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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 July 2023 (20.07.2023)

WIPO | PCT

(10) International Publication Number
WO 2023/137217 A1

(51) International Patent Classification:
H04N 19/60 (2014.01) *H04N 19/423* (2014.01)
H04N 19/18 (2014.01) *H04N 19/176* (2014.01)
H04N 19/159 (2014.01)

(21) International Application Number:
PCT/US2023/010901

(22) International Filing Date:
17 January 2023 (17.01.2023)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
63/300,187 17 January 2022 (17.01.2022) US
PCT/US2022/040442
16 August 2022 (16.08.2022) US
PCT/US2022/043607
15 September 2022 (15.09.2022) US
PCT/US2022/049264
08 November 2022 (08.11.2022) US
PCT/US2022/053150
16 December 2022 (16.12.2022) US

(71) Applicant: **BEIJING DAJIA INTERNET INFORMATION TECHNOLOGY CO., LTD.** [CN/CN]; Room 101D1-7, 1st Floor, Building 1, No. 6 Shangdi West Road, Haidian District, Beijing, 100085 (CN).

(72) Inventor; and

(71) Applicant *(for MN only)*: **WANG, Xianglin** [US/US]; 8910 University Center Lane, #400, San Diego, CA 92122 (US).

(72) Inventors: **XIU, Xiaoyu**; 8910 University Center Lane, #400, San Diego, CA 92122 (US). **YAN, Ning**; Room 702, Delixi, Building, No. 28, Xueyuen Road, Xihu District, Hangzhou, Zhejiang Province (CN). **CHEN, Yi-wen**; 8910 University Center Lane, #400, San Diego, CA 92122 (US). **KUO, Che-wei**; Room 101D1-7, 1st Floor, Building 1, No. 6, Shangdi West Road, Haidian District, Beijing (CN). **CHEN, Wei**; 8910 University Center Lane, #400, San Diego, CA 92122 (US). **JHU, Hong-jheng**; Room 101D1-7, 1st Floor, Building 1, No. 6, Shangdi West Road, Haidian District, Beijing (CN). **GAO, Han**; 8910 University Center Lane, #400, San Diego, CA 92122 (US). **YU, Bing**; Room 101D1-7, 1st Floor, Building 1, No. 6, Shangdi West Road, Haidian District, Beijing (CN).

(74) Agent: **SONG, Jiayu**; Bayes PLLC, 8260 Greensboro Drive, Suite 625, McLean, VA 22102 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM,

(54) Title: SIGN PREDICTION FOR BLOCK-BASED VIDEO CODING

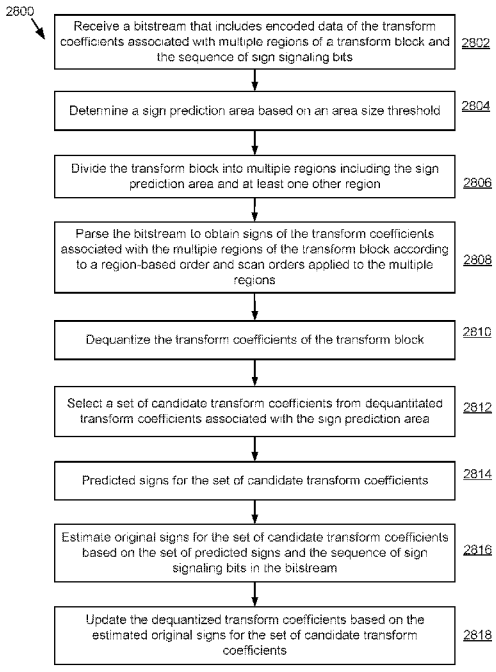


FIG. 28

(57) Abstract: Implementations of the disclosure provide a video decoding apparatus and method for transform coefficient sign prediction on a video decoder side. The method may include receiving a bitstream comprising encoded data of transform coefficients of a transform block of a video frame from a video. The method may further include determining, by a decoder, multiple regions in the transform block, where the multiple regions include a sign prediction area for performing the sign prediction and at least one other region. The method may further include parsing the bitstream, by the decoder, to obtain signs of transform coefficients associated with the multiple regions of the transform block according to a region-based order used to signal the signs of the transform coefficients. The method may also include estimating, by the decoder, original signs for a set of candidate transform coefficients associated with the sign prediction area of the transform block, and updating the transform coefficients associated with the sign prediction area based on the estimated original signs.

WO 2023/137217 A1

[Continued on next page]

AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, 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, CV, 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, ME, 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))*

SIGN PREDICTION FOR BLOCK-BASED VIDEO CODING

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application is based upon and claims priority to U.S. Provisional Application No. 63/300,187, filed January 17, 2022. This application is further based upon and claims priority to PCT Application No. PCT/US22/53150, filed December 16, 2022, which claims priority to U.S. Provisional Application No. 63/290,307, filed December 16, 2021. This application is further based upon and claims priority to PCT Application No. PCT/US22/49264, filed November 8, 2022, which claims priority to U.S. Provisional Application No. 63/277,705, filed November 10, 2021. This application is further based upon and claims priority to PCT Application No. PCT/US22/43607, filed September 15, 2022, which further claims priority to U.S. Provisional Application No. 63/244,317, filed September 15, 2021, and U.S. Provisional Application No. 63/250,797, filed September 30, 2021. This application is also based upon and claims priority to PCT Application No. PCT/US22/40442, filed August 16, 2022, which further claims priority to U.S. Provisional Application No. 63/233,940, filed August 17, 2021. The contents of all above applications are incorporated herein by reference in their entireties.

TECHNICAL FIELD

[0002] This application is related to video coding and compression. More specifically, this application relates to video processing systems and methods for sign prediction in block-based video coding.

BACKGROUND

[0003] Digital video is supported by a variety of electronic devices, such as digital televisions, laptop or desktop computers, tablet computers, digital cameras, digital recording devices, digital media players, video gaming consoles, smart phones, video teleconferencing devices, video streaming devices, etc. The electronic devices transmit and receive or otherwise communicate digital video data across a communication network, and/or store the digital video data on a storage device. Due to a limited bandwidth capacity of the communication network and limited memory resources of the storage device, video coding may be used to compress the video data according to one or more video coding standards before it is communicated or stored. For example, video coding standards include Versatile Video Coding (VVC), Joint

Exploration test Model (JEM), High-Efficiency Video Coding (HEVC/H.265), Advanced Video Coding (AVC/H.264), Moving Picture Expert Group (MPEG) coding, or the like. Video coding generally utilizes prediction methods (e.g., inter-prediction, intra-prediction, or the like) that take advantage of redundancy inherent in the video data. Video coding aims to compress video data into a form that uses a lower bit rate, while avoiding or minimizing degradations to video quality.

SUMMARY

[0004] Implementations of the present disclosure provide a video decoding method for sign prediction of transform coefficients. The method may include receiving a bitstream comprising encoded data of transform coefficients of a transform block of a video frame from a video. The method may further include determining, by a decoder, multiple regions in the transform block, where the multiple regions include a sign prediction area for performing the sign prediction and at least one other region. The method may further include parsing the bitstream, by the decoder, to obtain encoded data of transform coefficients associated with the multiple regions of the transform block according to a region-based order used to signal the encoded data of the transform coefficients. The method may also include estimating, by the decoder, original signs for a set of candidate transform coefficients associated with the sign prediction area of the transform block, and updating the transform coefficients associated with the sign prediction area based on the estimated original signs.

[0005] Implementations of the present disclosure also provide a video decoding apparatus. The video decoding apparatus may include a memory configured to perform the video decoding method for sign prediction of transform coefficients, as summarized above.

[0006] Implementations of the present disclosure also provide a non-transitory computer-readable storage medium having stored therein instructions which, when executed by a processor, cause the processor to perform a video decoding method for sign prediction of transform coefficients as summarized above. Implementations of the present disclosure further provide a non-transitory computer-readable storage medium having stored therein a bitstream including encoded data of transform coefficients of a transform block. The bitstream is decodable by the video decoding method for sign prediction of transform coefficients, as summarized above.

[0007] Implementations of the present disclosure also provide a video encoding method for sign prediction of transform coefficients. The video encoding method may include

generating, by an encoder, transform coefficients of a transform block of a video frame from a video. The method may further include determining, by the encoder, multiple regions in the transform block, wherein the multiple regions include a sign prediction area for performing the sign prediction and at least one other region. The method may also include predicting, by the encoder, signs for a set of candidate transform coefficients associated with the sign prediction area of the transform block; and generating, by the encoder, a bitstream to signal signs of the transform coefficients associated with the multiple regions of the transform block according to a region-based order.

[0008] Implementations of the present disclosure also provide a video encoding apparatus. The video decoding apparatus may include a memory configured to perform the video encoding method for sign prediction of transform coefficients, as summarized above.

[0009] Implementations of the present disclosure also provide a non-transitory computer-readable storage medium having stored therein instructions which, when executed by a processor, cause the processor to perform a video encoding method for sign prediction of transform coefficients as summarized above. Implementations of the present disclosure further provide a non-transitory computer-readable storage medium having stored therein a bitstream generated by the video encoding method for sign prediction of transform coefficients, as summarized above.

[0010] It is to be understood that both the foregoing general description and the following detailed description are examples only and are not restrictive of the present disclosure.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate examples consistent with the present disclosure and, together with the description, serve to explain the principles of the disclosure.

[0012] FIG. 1 is a block diagram illustrating an exemplary system for encoding and decoding video blocks in accordance with some implementations of the present disclosure.

[0013] FIG. 2 is a block diagram illustrating an exemplary video encoder in accordance with some implementations of the present disclosure.

[0014] FIG. 3 is a block diagram illustrating an exemplary video decoder in accordance with some implementations of the present disclosure.

[0015] FIGS. 4A through 4E are graphical representations illustrating how a frame is recursively partitioned into multiple video blocks of different sizes and shapes in accordance

with some implementations of the present disclosure.

[0016] FIG. 5 is a graphical representation illustrating a top-left scan order of transform coefficients in a coefficient group in accordance with some examples.

[0017] FIG. 6 is a graphical representation illustrating a low-frequency non-separable transform (LFNST) process in accordance with some examples.

[0018] FIG. 7 is a graphical representation illustrating a top-left region of primary transform coefficients input to a forward LFNST in accordance with some examples.

[0019] FIG. 8 is a graphical representation illustrating a search area of intra template matching in accordance with some examples.

[0020] FIG. 9 is a graphical representation illustrating an exemplary process of sign prediction in accordance with some examples.

[0021] FIG. 10 is a graphical representation illustrating calculation of a cost function for sign prediction in accordance with some examples.

[0022] FIG. 11 is a graphical representation illustrating two exemplary scalar quantizers used in a dependent scalar quantization in accordance with some examples.

[0023] FIG. 12A is a graphical representation illustrating state transition using a state machine with four states used in a dependent scalar quantization in accordance with some examples.

[0024] FIG. 12B is a table showing an exemplary quantizer selection according to the state transition of FIG. 12A in accordance with some examples.

[0025] FIG. 13 is a block diagram illustrating an exemplary sign prediction process in block-based video coding in accordance with some implementations of the present disclosure.

[0026] FIG. 14 is a graphical representation illustrating exemplary hypothesis generation based on a linear combination of templates in accordance with some implementations of the present disclosure.

[0027] FIG. 15A is a graphical representation illustrating an exemplary implementation of an existing sign prediction scheme in accordance with some examples.

[0028] FIG. 15B is a graphical representation illustrating an exemplary implementation of a vector-based sign prediction scheme in accordance with some implementations of the present disclosure.

[0029] FIG. 16A is a graphical representation illustrating exemplary calculation of a left-diagonal cost function along a left diagonal direction in accordance with some implementations of the present disclosure.

[0030] FIG. 16B is a graphical representation illustrating exemplary calculation of a right-

diagonal cost function along a right diagonal direction in accordance with some implementations of the present disclosure.

[0031] FIG. 17 is a flow chart of a method for capturing a dominate gradient direction within the neighboring reconstructed samples of a current block in accordance with some implementations of the present disclosure.

[0032] FIG. 18A is a graphical representation illustrating exemplary template samples and gradient filter windows in a gradient-based selection of a sample extrapolation direction for a cost function in accordance with some implementations of the present disclosure.

[0033] FIG. 18B a graphical representation illustrating an exemplary histogram of gradient (HoG) in a gradient-based selection of a sample extrapolation direction for a cost function in accordance with some implementations of the present disclosure.

[0034] FIG. 19 is a graphical representation illustrating a sign prediction area for predicting signs of transform coefficients in accordance with some examples.

[0035] FIG. 20 is a graphical representation illustrating a division of a transform block for implementing an exemplary order for signaling the signs of transform coefficients from an encoder to a decoder in accordance with some examples.

[0036] FIG. 21 is a graphical representation illustrating various scan orders for scanning the signs of transform coefficients in accordance with some examples.

[0037] FIG. 22 is a graphical representation illustrating implementation of different scan orders for the transform block divided according to FIG. 20, in accordance with some examples.

[0038] FIG. 23 is a graphical representation illustrating a division of a transform block for implementing another exemplary order for signaling the signs of transform coefficients from an encoder to a decoder in accordance with some examples.

[0039] FIG. 24 is a graphical representation illustrating implementation of different scan orders for the transform block divided according to FIG. 20, in accordance with some examples.

[0040] FIG. 25A is a graphical representation illustrating exemplary context modeling for transform coefficient sign prediction in accordance with some implementations of the present disclosure.

[0041] FIG. 25B is a graphical representation illustrating another exemplary context modeling for transform coefficient sign prediction in accordance with some implementations of the present disclosure.

[0042] FIG. 26 is a flow chart of an exemplary method for sign prediction in block-based video coding in accordance with some implementations of the present disclosure.

[0043] FIG. 27 is a flow chart of an exemplary video encoding method for transform

coefficient sign prediction performed by a video encoder in accordance with some implementations of the present disclosure.

[0044] FIG. 28 is a flow chart of an exemplary video decoding method for transform coefficient sign prediction performed by a video decoder in accordance with some implementations of the present disclosure.

[0045] FIG. 29 is a block diagram illustrating a computing environment coupled with a user interface in accordance with some implementations of the present disclosure.

DETAILED DESCRIPTION

[0046] Reference will now be made in detail to specific implementations, examples of which are illustrated in the accompanying drawings. In the following detailed description, numerous non-limiting specific details are set forth in order to assist in understanding the subject matter presented herein. But it will be apparent to one of ordinary skill in the art that various alternatives may be used without departing from the scope of claims and the subject matter may be practiced without these specific details. For example, it will be apparent to one of ordinary skill in the art that the subject matter presented herein can be implemented on many types of electronic devices with digital video capabilities.

[0047] It should be illustrated that the terms "first," "second," and the like used in the description, claims of the present disclosure, and the accompanying drawings are used to distinguish objects, and not used to describe any specific order or sequence. It should be understood that the data used in this way may be interchanged under an appropriate condition, such that the embodiments of the present disclosure described herein may be implemented in orders besides those shown in the accompanying drawings or described in the present disclosure.

[0048] FIG. 1 is a block diagram illustrating an exemplary system 10 for encoding and decoding video blocks in parallel in accordance with some implementations of the present disclosure. As shown in FIG. 1, the system 10 includes a source device 12 that generates and encodes video data to be decoded at a later time by a destination device 14. The source device 12 and the destination device 14 may include any of a wide variety of electronic devices, including desktop or laptop computers, tablet computers, smart phones, set-top boxes, digital televisions, cameras, display devices, digital media players, video gaming consoles, video streaming device, or the like. In some implementations, the source device 12 and the destination device 14 are equipped with wireless communication capabilities.

[0049] In some implementations, the destination device 14 may receive the encoded video data to be decoded via a link 16. The link 16 may include any type of communication medium or device capable of forwarding the encoded video data from the source device 12 to the destination device 14. In one example, the link 16 may include a communication medium to enable the source device 12 to transmit the encoded video data directly to the destination device 14 in real time. The encoded video data may be modulated according to a communication standard, such as a wireless communication protocol, and transmitted to the destination device 14. The communication medium may include any wireless or wired communication medium, such as a Radio Frequency (RF) spectrum or one or more physical transmission lines. The communication medium may form part of a packet-based network, such as a local area network, a wide-area network, or a global network such as the Internet. The communication medium may include routers, switches, base stations, or any other equipment that may be useful to facilitate communication from the source device 12 to the destination device 14.

[0050] In some other implementations, the encoded video data may be transmitted from an output interface 22 to a storage device 32. Subsequently, the encoded video data in the storage device 32 may be accessed by the destination device 14 via an input interface 28. The storage device 32 may include any of a variety of distributed or locally accessed data storage media such as a hard drive, Blu-ray discs, Digital Versatile Disks (DVDs), Compact Disc Read-Only Memories (CD-ROMs), flash memory, volatile or non-volatile memory, or any other suitable digital storage media for storing the encoded video data. In a further example, the storage device 32 may correspond to a file server or another intermediate storage device that may store the encoded video data generated by the source device 12. The destination device 14 may access the stored video data from the storage device 32 via streaming or downloading. The file server may be any type of computer capable of storing the encoded video data and transmitting the encoded video data to the destination device 14. Exemplary file servers include a web server (e.g., for a website), a File Transfer Protocol (FTP) server, Network Attached Storage (NAS) devices, or a local disk drive. The destination device 14 may access the encoded video data through any standard data connection, including a wireless channel (e.g., a Wireless Fidelity (Wi-Fi) connection), a wired connection (e.g., Digital Subscriber Line (DSL), cable modem, etc.), or any combination thereof that is suitable for accessing encoded video data stored on a file server. The transmission of the encoded video data from the storage device 32 may be a streaming transmission, a download transmission, or a combination of both.

[0051] As shown in FIG. 1, the source device 12 includes a video source 18, a video encoder 20, and the output interface 22. The video source 18 may include a source such as a

video capturing device, e.g., a video camera, a video archive containing previously captured video, a video feeding interface to receive video data from a video content provider, and/or a computer graphics system for generating computer graphics data as the source video, or a combination of such sources. As one example, if the video source 18 is a video camera of a security surveillance system, the source device 12 and the destination device 14 may include camera phones or video phones. However, the implementations described in the present disclosure may be applicable to video coding in general, and may be applied to wireless and/or wired applications.

[0052] The captured, pre-captured, or computer-generated video may be encoded by the video encoder 20. The encoded video data may be transmitted directly to the destination device 14 via the output interface 22 of the source device 12. The encoded video data may also (or alternatively) be stored onto the storage device 32 for later access by the destination device 14 or other devices, for decoding and/or playback. The output interface 22 may further include a modem and/or a transmitter.

[0053] The destination device 14 includes the input interface 28, a video decoder 30, and a display device 34. The input interface 28 may include a receiver and/or a modem and receive the encoded video data over the link 16. The encoded video data communicated over the link 16, or provided on the storage device 32, may include a variety of syntax elements generated by the video encoder 20 for use by the video decoder 30 in decoding the video data. Such syntax elements may be included within the encoded video data transmitted on a communication medium, stored on a storage medium, or stored on a file server.

[0054] In some implementations, the destination device 14 may include the display device 34, which can be an integrated display device and an external display device that is configured to communicate with the destination device 14. The display device 34 displays the decoded video data for a user, and may include any of a variety of display devices such as a Liquid Crystal Display (LCD), a plasma display, an Organic Light Emitting Diode (OLED) display, or another type of display device.

[0055] The video encoder 20 and the video decoder 30 may operate according to proprietary or industry standards, such as VVC, HEVC, MPEG-4, Part 10, AVC, or extensions of such standards. It should be understood that the present disclosure is not limited to a specific video encoding/decoding standard and may be applicable to other video encoding/decoding standards. It is generally contemplated that the video encoder 20 of the source device 12 may be configured to encode video data according to any of these current or future standards. Similarly, it is also generally contemplated that the video decoder 30 of the destination device

14 may be configured to decode video data according to any of these current or future standards.

[0056] The video encoder 20 and the video decoder 30 each may be implemented as any of a variety of suitable encoder and/or decoder circuitry, such as one or more microprocessors, Digital Signal Processors (DSPs), Application Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs), discrete logic, software, hardware, firmware or any combinations thereof. When implemented partially in software, an electronic device may store instructions for the software in a suitable, non-transitory computer-readable medium and execute the instructions in hardware using one or more processors to perform the video encoding/decoding operations disclosed in the present disclosure. Each of the video encoder 20 and the video decoder 30 may be included in one or more encoders or decoders, either of which may be integrated as part of a combined encoder/decoder (CODEC) in a respective device.

[0057] FIG. 2 is a block diagram illustrating an exemplary video encoder 20 in accordance with some implementations described in the present application. The video encoder 20 may perform intra and inter predictive coding of video blocks within video frames. Intra predictive coding relies on spatial prediction to reduce or remove spatial redundancy in video data within a given video frame or picture. Inter predictive coding relies on temporal prediction to reduce or remove temporal redundancy in video data within adjacent video frames or pictures of a video sequence. It should be noted that the term "frame" may be used as synonyms for the term "image" or "picture" in the field of video coding.

[0058] As shown in FIG. 2, the video encoder 20 includes a video data memory 40, a prediction processing unit 41, a Decoded Picture Buffer (DPB) 64, a summer 50, a transform processing unit 52, a quantization unit 54, and an entropy encoding unit 56. The prediction processing unit 41 further includes a motion estimation unit 42, a motion compensation unit 44, a partition unit 45, an intra prediction processing unit 46, and an intra Block Copy (BC) unit 48. In some implementations, the video encoder 20 also includes an inverse quantization unit 58, an inverse transform processing unit 60, and a summer 62 for video block reconstruction. An in-loop filter 63, such as a deblocking filter, may be positioned between the summer 62 and the DPB 64 to filter block boundaries to remove block artifacts from reconstructed video data. Another in-loop filter, such as an SAO filter and/or Adaptive in-Loop Filter (ALF), may also be used in addition to the deblocking filter to filter an output of the summer 62. In some examples, the in-loop filters may be omitted, and the decoded video block may be directly provided by the summer 62 to the DPB 64. The video encoder 20 may take the form of a fixed or programmable hardware unit or may be divided among one or more of

the illustrated fixed or programmable hardware units.

[0059] The video data memory 40 may store video data to be encoded by the components of the video encoder 20. The video data in the video data memory 40 may be obtained, for example, from the video source 18 as shown in FIG. 1. The DPB 64 is a buffer that stores reference video data (for example, reference frames or pictures) for use in encoding video data by the video encoder 20 (e.g., in intra or inter predictive coding modes). The video data memory 40 and the DPB 64 may be formed by any of a variety of memory devices. In various examples, the video data memory 40 may be on-chip with other components of the video encoder 20, or off-chip relative to those components.

[0060] As shown in FIG. 2, after receiving the video data, the partition unit 45 within the prediction processing unit 41 partitions the video data into video blocks. This partitioning may also include partitioning a video frame into slices, tiles (for example, sets of video blocks), or other larger Coding Units (CUs) according to predefined splitting structures such as a Quad-Tree (QT) structure associated with the video data. The video frame is or may be regarded as a two-dimensional array or matrix of samples with sample values. A sample in the array may also be referred to as a pixel or a pel. A number of samples in horizontal and vertical directions (or axes) of the array or picture define a size and/or a resolution of the video frame. The video frame may be divided into multiple video blocks by, for example, using QT partitioning. The video block again is or may be regarded as a two-dimensional array or matrix of samples with sample values, although of smaller dimension than the video frame. A number of samples in horizontal and vertical directions (or axes) of the video block define a size of the video block. The video block may further be partitioned into one or more block partitions or sub-blocks (which may form again blocks) by, for example, iteratively using QT partitioning, Binary-Tree (BT) partitioning, Triple-Tree (TT) partitioning or any combination thereof. It should be noted that the term "block" or "video block" as used herein may be a portion, in particular a rectangular (square or non- square) portion, of a frame or a picture. With reference to, for example, HEVC and VVC, the block or video block may be or correspond to a Coding Tree Unit (CTU), a CU, a Prediction Unit (PU), or a Transform Unit (TU), and/or may be or correspond to a corresponding block, e.g., a Coding Tree Block (CTB), a Coding Block (CB), a Prediction Block (PB), or a Transform Block (TB). Alternatively or additionally, the block or video block may be or correspond to a sub-block of a CTB, a CB, a PB, a TB, etc.

[0061] The prediction processing unit 41 may select one of a plurality of possible predictive coding modes, such as one of a plurality of intra predictive coding modes or one of a plurality of inter predictive coding modes, for the current video block based on error results (e.g., coding

rate and the level of distortion). The prediction processing unit 41 may provide the resulting intra or inter prediction coded block (e.g., a predictive block) to the summer 50 to generate a residual block and to the summer 62 to reconstruct the encoded block for use as part of a reference frame subsequently. The prediction processing unit 41 also provides syntax elements, such as motion vectors, intra-mode indicators, partition information, and other such syntax information to the entropy encoding unit 56.

[0062] In order to select an appropriate intra predictive coding mode for the current video block, the intra prediction processing unit 46 within the prediction processing unit 41 may perform intra predictive coding of the current video block relative to one or more neighbor blocks in the same frame as the current block to be coded to provide spatial prediction. The motion estimation unit 42 and the motion compensation unit 44 within the prediction processing unit 41 perform inter predictive coding of the current video block relative to one or more predictive blocks in one or more reference frames to provide temporal prediction. The video encoder 20 may perform multiple coding passes, e.g., to select an appropriate coding mode for each block of video data.

[0063] In some implementations, the motion estimation unit 42 determines the inter prediction mode for a current video frame by generating a motion vector, which indicates the displacement of a video block within the current video frame relative to a predictive block within a reference frame, according to a predetermined pattern within a sequence of video frames. Motion estimation, performed by the motion estimation unit 42, may be a process of generating motion vectors, which may estimate motion for video blocks. A motion vector, for example, may indicate the displacement of a video block within a current video frame or picture relative to a predictive block within a reference frame. The predetermined pattern may designate video frames in the sequence as P frames or B frames. The intra BC unit 48 may determine vectors, e.g., block vectors, for intra BC coding in a manner similar to the determination of motion vectors by the motion estimation unit 42 for inter prediction, or may utilize the motion estimation unit 42 to determine the block vectors.

[0064] A predictive block for the video block may be or may correspond to a block or a reference block of a reference frame that is deemed as closely matching the video block to be coded in terms of pixel difference, which may be determined by Sum of Absolute Difference (SAD), Sum of Square Difference (SSD), or other difference metrics. In some implementations, the video encoder 20 may calculate values for sub-integer pixel positions of reference frames stored in the DPB 64. For example, the video encoder 20 may interpolate values of one-quarter pixel positions, one-eighth pixel positions, or other fractional pixel positions of the reference

frame. Therefore, the motion estimation unit 42 may perform a motion search relative to the full pixel positions and fractional pixel positions and output a motion vector with fractional pixel precision.

[0065] The motion estimation unit 42 calculates a motion vector for a video block in an inter prediction coded frame by comparing the position of the video block to the position of a predictive block of a reference frame selected from a first reference frame list (List 0) or a second reference frame list (List 1), each of which identifies one or more reference frames stored in the DPB 64. The motion estimation unit 42 sends the calculated motion vector to the motion compensation unit 44 and then to the entropy encoding unit 56.

[0066] Motion compensation, performed by the motion compensation unit 44, may involve fetching or generating the predictive block based on the motion vector determined by the motion estimation unit 42. Upon receiving the motion vector for the current video block, the motion compensation unit 44 may locate a predictive block to which the motion vector points in one of the reference frame lists, retrieve the predictive block from the DPB 64, and forward the predictive block to the summer 50. The summer 50 then forms a residual block of pixel difference values by subtracting pixel values of the predictive block provided by the motion compensation unit 44 from the pixel values of the current video block being coded. The pixel difference values forming the residual block may include luma or chroma difference components or both. The motion compensation unit 44 may also generate syntax elements associated with the video blocks of a video frame for use by the video decoder 30 in decoding the video blocks of the video frame. The syntax elements may include, for example, syntax elements defining the motion vector used to identify the predictive block, any flags indicating the prediction mode, or any other syntax information described herein. It is noted that the motion estimation unit 42 and the motion compensation unit 44 may be integrated together, which are illustrated separately for conceptual purposes in FIG. 2.

[0067] In some implementations, the intra BC unit 48 may generate vectors and fetch predictive blocks in a manner similar to that described above in connection with the motion estimation unit 42 and the motion compensation unit 44, but with the predictive blocks being in the same frame as the current block being coded and with the vectors being referred to as block vectors as opposed to motion vectors. In particular, the intra BC unit 48 may determine an intra-prediction mode to use to encode a current block. In some examples, the intra BC unit 48 may encode a current block using various intra-prediction modes, e.g., during separate encoding passes, and test their performance through rate-distortion analysis. Next, the intra BC unit 48 may select, among the various tested intra-prediction modes, an appropriate intra-

prediction mode to use and generate an intra-mode indicator accordingly. For example, the intra BC unit 48 may calculate rate-distortion values using a rate-distortion analysis for the various tested intra-prediction modes, and select the intra-prediction mode having the best rate-distortion characteristics among the tested modes as the appropriate intra-prediction mode to use. Rate-distortion analysis generally determines an amount of distortion (or error) between an encoded block and an original, unencoded block that was encoded to produce the encoded block, as well as a bitrate (i.e., a number of bits) used to produce the encoded block. Intra BC unit 48 may calculate ratios from the distortions and rates for the various encoded blocks to determine which intra-prediction mode exhibits the best rate-distortion value for the block.

[0068] In other examples, the intra BC unit 48 may use the motion estimation unit 42 and the motion compensation unit 44, in whole or in part, to perform such functions for Intra BC prediction according to the implementations described herein. In either case, for intra block copy, a predictive block may be a block that is deemed as closely matching the block to be coded, in terms of pixel difference, which may be determined by SAD, SSD, or other difference metrics, and identification of the predictive block may include calculation of values for sub-integer pixel positions.

[0069] Whether the predictive block is from the same frame according to intra prediction, or from a different frame according to inter prediction, the video encoder 20 may form a residual block by subtracting pixel values of the predictive block from the pixel values of the current video block being coded, forming pixel difference values. The pixel difference values forming the residual block may include both luma and chroma component differences.

[0070] The intra prediction processing unit 46 may intra-predict a current video block, as an alternative to the inter-prediction performed by the motion estimation unit 42 and the motion compensation unit 44, or the intra block copy prediction performed by the intra BC unit 48, as described above. In particular, the intra prediction processing unit 46 may determine an intra prediction mode to use to encode a current block. For example, the intra prediction processing unit 46 may encode a current block using various intra prediction modes, e.g., during separate encoding passes, and the intra prediction processing unit 46 (or a mode selection unit, in some examples) may select an appropriate intra prediction mode to use from the tested intra prediction modes. The intra prediction processing unit 46 may provide information indicative of the selected intra-prediction mode for the block to the entropy encoding unit 56. The entropy encoding unit 56 may encode the information indicating the selected intra-prediction mode in a bitstream.

[0071] After the prediction processing unit 41 determines the predictive block for the

current video block via either inter prediction or intra prediction, the summer 50 forms a residual block by subtracting the predictive block from the current video block. The residual video data in the residual block may be included in one or more TUs and is provided to the transform processing unit 52. The transform processing unit 52 transforms the residual video data into transform coefficients using a transform, such as a Discrete Cosine Transform (DCT) or a conceptually similar transform.

[0072] The transform processing unit 52 may send the resulting transform coefficients to the quantization unit 54. The quantization unit 54 quantizes the transform coefficients to further reduce the bit rate. The quantization process may also reduce the bit depth associated with some or all of the coefficients. The degree of quantization may be modified by adjusting a quantization parameter. In some examples, the quantization unit 54 may then perform a scan of a matrix including the quantized transform coefficients. Alternatively, the entropy encoding unit 56 may perform the scan.

[0073] Following quantization, the entropy encoding unit 56 may use an entropy encoding technique to encode the quantized transform coefficients into a video bitstream, e.g., using Context Adaptive Variable Length Coding (CAVLC), Context Adaptive Binary Arithmetic Coding (CABAC), Syntax-based context-adaptive Binary Arithmetic Coding (SBAC), Probability Interval Partitioning Entropy (PIPE) coding, or another entropy encoding methodology or technique. The encoded bitstream may then be transmitted to the video decoder 30 as shown in FIG. 1 or archived in the storage device 32 as shown in FIG. 1 for later transmission to or retrieval by the video decoder 30. The entropy encoding unit 56 may also use an entropy encoding technique to encode the motion vectors and the other syntax elements for the current video frame being coded.

[0074] The inverse quantization unit 58 and the inverse transform processing unit 60 apply inverse quantization and inverse transformation, respectively, to reconstruct the residual block in the pixel domain for generating a reference block for prediction of other video blocks. A reconstructed residual block may be generated thereof. As noted above, the motion compensation unit 44 may generate a motion compensated predictive block from one or more reference blocks of the frames stored in the DPB 64. The motion compensation unit 44 may also apply one or more interpolation filters to the predictive block to calculate sub-integer pixel values for use in motion estimation.

[0075] The summer 62 adds the reconstructed residual block to the motion compensated predictive block produced by the motion compensation unit 44 to produce a reference block for storage in the DPB 64. The reference block may then be used by the intra BC unit 48, the

motion estimation unit 42, and the motion compensation unit 44 as a predictive block to inter predict another video block in a subsequent video frame.

[0076] FIG. 3 is a block diagram illustrating an exemplary video decoder 30 in accordance with some implementations of the present application. The video decoder 30 includes a video data memory 79, an entropy decoding unit 80, a prediction processing unit 81, an inverse quantization unit 86, an inverse transform processing unit 88, a summer 90, and a DPB 92. The prediction processing unit 81 further includes a motion compensation unit 82, an intra prediction unit 84, and an intra BC unit 85. The video decoder 30 may perform a decoding process generally reciprocal to the encoding process described above with respect to the video encoder 20 in connection with FIG. 2. For example, the motion compensation unit 82 may generate prediction data based on motion vectors received from the entropy decoding unit 80, while the intra prediction unit 84 may generate prediction data based on intra-prediction mode indicators received from the entropy decoding unit 80.

[0077] In some examples, a unit of the video decoder 30 may be tasked to perform the implementations of the present application. Also, in some examples, the implementations of the present disclosure may be divided among one or more of the units of the video decoder 30. For example, the intra BC unit 85 may perform the implementations of the present application, alone, or in combination with other units of the video decoder 30, such as the motion compensation unit 82, the intra prediction unit 84, and the entropy decoding unit 80. In some examples, the video decoder 30 may not include the intra BC unit 85 and the functionality of intra BC unit 85 may be performed by other components of the prediction processing unit 81, such as the motion compensation unit 82.

[0078] The video data memory 79 may store video data, such as an encoded video bitstream, to be decoded by the other components of the video decoder 30. The video data stored in the video data memory 79 may be obtained, for example, from the storage device 32, from a local video source, such as a camera, via wired or wireless network communication of video data, or by accessing physical data storage media (e.g., a flash drive or hard disk). The video data memory 79 may include a Coded Picture Buffer (CPB) that stores encoded video data from an encoded video bitstream. The DPB 92 of the video decoder 30 stores reference video data for use in decoding video data by the video decoder 30 (e.g., in intra or inter predictive coding modes). The video data memory 79 and the DPB 92 may be formed by any of a variety of memory devices, such as dynamic random access memory (DRAM), including Synchronous DRAM (SDRAM), Magneto-resistive RAM (MRAM), Resistive RAM (RRAM), or other types of memory devices. For illustrative purpose, the video data memory 79 and the DPB 92

are depicted as two distinct components of the video decoder 30 in FIG. 3. But it will be apparent to one skilled in the art that the video data memory 79 and the DPB 92 may be provided by the same memory device or separate memory devices. In some examples, the video data memory 79 may be on-chip with other components of the video decoder 30, or off-chip relative to those components.

[0079] During the decoding process, the video decoder 30 receives an encoded video bitstream that represents video blocks of an encoded video frame and associated syntax elements. The video decoder 30 may receive the syntax elements at the video frame level and/or the video block level. The entropy decoding unit 80 of the video decoder 30 may use an entropy decoding technique to decode the bitstream to obtain quantized coefficients, motion vectors or intra-prediction mode indicators, and other syntax elements. The entropy decoding unit 80 then forwards the motion vectors or intra-prediction mode indicators and other syntax elements to the prediction processing unit 81.

[0080] When the video frame is coded as an intra predictive coded (e.g., I) frame or for intra coded predictive blocks in other types of frames, the intra prediction unit 84 of the prediction processing unit 81 may generate prediction data for a video block of the current video frame based on a signaled intra prediction mode and reference data from previously decoded blocks of the current frame.

[0081] When the video frame is coded as an inter-predictive coded (i.e., B or P) frame, the motion compensation unit 82 of the prediction processing unit 81 produces one or more predictive blocks for a video block of the current video frame based on the motion vectors and other syntax elements received from the entropy decoding unit 80. Each of the predictive blocks may be produced from a reference frame within one of the reference frame lists. The video decoder 30 may construct the reference frame lists, e.g., List 0 and List 1, using default construction techniques based on reference frames stored in the DPB 92.

[0082] In some examples, when the video block is coded according to the intra BC mode described herein, the intra BC unit 85 of the prediction processing unit 81 produces predictive blocks for the current video block based on block vectors and other syntax elements received from the entropy decoding unit 80. The predictive blocks may be within a reconstructed region of the same picture as the current video block processed by the video encoder 20.

[0083] The motion compensation unit 82 and/or the intra BC unit 85 determines prediction information for a video block of the current video frame by parsing the motion vectors and other syntax elements, and then uses the prediction information to produce the predictive blocks for the current video block being decoded. For example, the motion compensation unit

82 uses some of the received syntax elements to determine a prediction mode (e.g., intra or inter prediction) used to code video blocks of the video frame, an inter prediction frame type (e.g., B or P), construction information for one or more of the reference frame lists for the frame, motion vectors for each inter predictive encoded video block of the frame, inter prediction status for each inter predictive coded video block of the frame, and other information to decode the video blocks in the current video frame.

[0084] Similarly, the intra BC unit 85 may use some of the received syntax elements, e.g., a flag, to determine that the current video block was predicted using the intra BC mode, construction information of which video blocks of the frame are within the reconstructed region and should be stored in the DPB 92, block vectors for each intra BC predicted video block of the frame, intra BC prediction status for each intra BC predicted video block of the frame, and other information to decode the video blocks in the current video frame.

[0085] The motion compensation unit 82 may also perform interpolation using the interpolation filters as used by the video encoder 20 during encoding of the video blocks to calculate interpolated values for sub-integer pixels of reference blocks. In this case, the motion compensation unit 82 may determine the interpolation filters used by the video encoder 20 from the received syntax elements and use the interpolation filters to produce predictive blocks.

[0086] The inverse quantization unit 86 inversely quantizes the quantized transform coefficients provided in the bitstream and decoded by the entropy decoding unit 80 using the same quantization parameter calculated by the video encoder 20 for each video block in the video frame to determine a degree of quantization. The inverse transform processing unit 88 applies an inverse transform, e.g., an inverse DCT, an inverse integer transform, or a conceptually similar inverse transform process, to the transform coefficients in order to reconstruct the residual blocks in the pixel domain.

[0087] After the motion compensation unit 82 or the intra BC unit 85 generates the predictive block for the current video block based on the vectors and other syntax elements, the summer 90 reconstructs a decoded video block for the current video block by summing the residual block from the inverse transform processing unit 88 and a corresponding predictive block generated by the motion compensation unit 82 and the intra BC unit 85. The decoded video block may also be referred to as a reconstructed block for the current video block. An in-loop filter 91 such as a deblocking filter, SAO filter, and/or ALF may be positioned between the summer 90 and the DPB 92 to further process the decoded video block. In some examples, the in-loop filter 91 may be omitted, and the decoded video block may be directly provided by the summer 90 to the DPB 92. The decoded video blocks in a given frame are then stored in

the DPB 92, which stores reference frames used for subsequent motion compensation of next video blocks. The DPB 92, or a memory device separate from the DPB 92, may also store decoded video for later presentation on a display device, such as the display device 34 of FIG. 1.

[0088] In a typical video coding process (e.g., including a video encoding process and a video decoding process), a video sequence typically includes an ordered set of frames or pictures. Each frame may include three sample arrays, denoted SL, SCb, and SCr. SL is a two-dimensional array of luma samples. SCb is a two-dimensional array of Cb chroma samples. SCr is a two-dimensional array of Cr chroma samples. In other instances, a frame may be monochrome and therefore includes only one two-dimensional array of luma samples.

[0089] As shown in FIG. 4A, the video encoder 20 (or more specifically the partition unit 45) generates an encoded representation of a frame by first partitioning the frame into a set of CTUs. A video frame may include an integer number of CTUs arranged consecutively in a raster scan order from left to right and from top to bottom. Each CTU is a largest logical coding unit and the width and height of the CTU are signaled by the video encoder 20 in a sequence parameter set, such that all the CTUs in a video sequence have the same size being one of 128×128, 64×64, 32×32, and 16×16. But it should be noted that a CTU in the present disclosure is not necessarily limited to a particular size. As shown in FIG. 4B, each CTU may include one CTB of luma samples, two corresponding coding tree blocks of chroma samples, and syntax elements used to code the samples of the coding tree blocks. The syntax elements describe properties of different types of units of a coded block of pixels and how the video sequence can be reconstructed at the video decoder 30, including inter or intra prediction, intra prediction mode, motion vectors, and other parameters. In monochrome pictures or pictures having three separate color planes, a CTU may include a single coding tree block and syntax elements used to code the samples of the coding tree block. A coding tree block may be an N×N block of samples.

[0090] To achieve a better performance, the video encoder 20 may recursively perform tree partitioning such as binary-tree partitioning, ternary-tree partitioning, quad-tree partitioning or a combination thereof on the coding tree blocks of the CTU and divide the CTU into smaller CUs. As depicted in FIG. 4C, the 64×64 CTU 400 is first divided into four smaller CUs, each having a block size of 32×32. Among the four smaller CUs, CU 410 and CU 420 are each divided into four CUs of 16×16 by block size. The two 16×16 CUs 430 and 440 are each further divided into four CUs of 8×8 by block size. FIG. 4D depicts a quad-tree data structure illustrating the end result of the partition process of the CTU 400 as depicted in FIG. 4C, each

leaf node of the quad-tree corresponding to one CU of a respective size ranging from 32×32 to 8×8. Like the CTU depicted in FIG. 4B, each CU may include a CB of luma samples and two corresponding coding blocks of chroma samples of a frame of the same size, and syntax elements used to code the samples of the coding blocks. In monochrome pictures or pictures having three separate colour planes, a CU may include a single coding block and syntax structures used to code the samples of the coding block. It should be noted that the quad-tree partitioning depicted in FIGS. 4C and 4D is only for illustrative purposes and one CTU can be split into CUs to adapt to varying local characteristics based on quad/ternary/binary-tree partitions. In the multi-type tree structure, one CTU is partitioned by a quad-tree structure and each quad-tree leaf CU can be further partitioned by a binary and ternary tree structure. As shown in FIG. 4E, there are multiple possible partitioning types of a coding block having a width W and a height H, i.e., quaternary partitioning, vertical binary partitioning, horizontal binary partitioning, vertical ternary partitioning, vertical extended ternary partitioning, horizontal ternary partitioning, and horizontal extended ternary partitioning.

[0091] In some implementations, the video encoder 20 may further partition a coding block of a CU into one or more M×N PBs. A PB may include a rectangular (square or non-square) block of samples on which the same prediction, inter or intra, is applied. A PU of a CU may include a PB of luma samples, two corresponding PBs of chroma samples, and syntax elements used to predict the PBs. In monochrome pictures or pictures having three separate color planes, a PU may include a single PB and syntax structures used to predict the PB. The video encoder 20 may generate predictive luma, Cb, and Cr blocks for luma, Cb, and Cr PBs of each PU of the CU.

[0092] The video encoder 20 may use intra prediction or inter prediction to generate the predictive blocks for a PU. If the video encoder 20 uses intra prediction to generate the predictive blocks of a PU, the video encoder 20 may generate the predictive blocks of the PU based on decoded samples of the frame associated with the PU. If the video encoder 20 uses inter prediction to generate the predictive blocks of a PU, the video encoder 20 may generate the predictive blocks of the PU based on decoded samples of one or more frames other than the frame associated with the PU.

[0093] After the video encoder 20 generates predictive luma, Cb, and Cr blocks for one or more PUs of a CU, the video encoder 20 may generate a luma residual block for the CU by subtracting the CU's predictive luma blocks from its original luma coding block such that each sample in the CU's luma residual block indicates a difference between a luma sample in one of the CU's predictive luma blocks and a corresponding sample in the CU's original luma coding

block. Similarly, the video encoder 20 may generate a Cb residual block and a Cr residual block for the CU, respectively, such that each sample in the CU's Cb residual block indicates a difference between a Cb sample in one of the CU's predictive Cb blocks and a corresponding sample in the CU's original Cb coding block, and each sample in the CU's Cr residual block may indicate a difference between a Cr sample in one of the CU's predictive Cr blocks and a corresponding sample in the CU's original Cr coding block.

[0094] Furthermore, as illustrated in FIG. 4C, the video encoder 20 may use quad-tree partitioning to decompose the luma, Cb, and Cr residual blocks of a CU into one or more luma, Cb, and Cr transform blocks respectively. A transform block may include a rectangular (square or non-square) block of samples on which the same transform is applied. A TU of a CU may include a transform block of luma samples, two corresponding transform blocks of chroma samples, and syntax elements used to transform the transform block samples. Thus, each TU of a CU may be associated with a luma transform block, a Cb transform block, and a Cr transform block. In some examples, the luma transform block associated with the TU may be a sub-block of the CU's luma residual block. The Cb transform block may be a sub-block of the CU's Cb residual block. The Cr transform block may be a sub-block of the CU's Cr residual block. In monochrome pictures or pictures having three separate color planes, a TU may include a single transform block and syntax structures used to transform the samples of the transform block.

[0095] The video encoder 20 may apply one or more transforms to a luma transform block of a TU to generate a luma coefficient block for the TU. A coefficient block may be a two-dimensional array of transform coefficients. A transform coefficient may be a scalar quantity. The video encoder 20 may apply one or more transforms to a Cb transform block of a TU to generate a Cb coefficient block for the TU. The video encoder 20 may apply one or more transforms to a Cr transform block of a TU to generate a Cr coefficient block for the TU.

[0096] After generating a coefficient block (e.g., a luma coefficient block, a Cb coefficient block, or a Cr coefficient block), the video encoder 20 may quantize the coefficient block. Quantization generally refers to a process in which transform coefficients are quantized to possibly reduce the amount of data used to represent the transform coefficients, providing further compression. After the video encoder 20 quantizes a coefficient block, the video encoder 20 may apply an entropy encoding technique to encode syntax elements indicating the quantized transform coefficients. For example, the video encoder 20 may perform CABAC on the syntax elements indicating the quantized transform coefficients. Finally, the video encoder 20 may output a bitstream that includes a sequence of bits that form a representation of coded

frames and associated data, which is either saved in the storage device 32 or transmitted to the destination device 14.

[0097] After receiving a bitstream generated by the video encoder 20, the video decoder 30 may parse the bitstream to obtain syntax elements from the bitstream. The video decoder 30 may reconstruct the frames of the video data based at least in part on the syntax elements obtained from the bitstream. The process of reconstructing the video data is generally reciprocal to the encoding process performed by the video encoder 20. For example, the video decoder 30 may perform inverse transforms on the coefficient blocks associated with TUs of a current CU to reconstruct residual blocks associated with the TUs of the current CU. The video decoder 30 also reconstructs the coding blocks of the current CU by adding the samples of the predictive blocks for PUs of the current CU to corresponding samples of the transform blocks of the TUs of the current CU. After reconstructing the coding blocks for each CU of a frame, video decoder 30 may reconstruct the frame.

[0098] As noted above, video coding achieves video compression using primarily two modes, i.e., intra-frame prediction (or intra-prediction) and inter-frame prediction (or inter-prediction). It is noted that intra block copy (IBC) could be regarded as either intra-frame prediction or a third mode. Between the two modes, inter-frame prediction contributes more to the coding efficiency than intra-frame prediction because of the use of motion vectors for predicting a current video block from a reference video block.

[0099] But with the ever-improving video data capturing technology and more refined video block size for preserving details in the video data, the amount of data required for representing motion vectors for a current frame also increases substantially. One way of overcoming this challenge is to benefit from the fact that not only a group of neighboring CUs in both the spatial and temporal domains have similar video data for predicting purpose but the motion vectors between these neighboring CUs are also similar. Therefore, it is possible to use the motion information of spatially neighboring CUs and/or temporally co-located CUs as an approximation of the motion information (e.g., motion vector) of a current CU by exploring their spatial and temporal correlation, which is also referred to as "Motion Vector Predictor (MVP)" of the current CU.

[00100] Instead of encoding an actual motion vector of the current CU into the video bitstream (e.g., the actual motion vector being determined by the motion estimation unit 42 as described above in connection with FIG. 2), the motion vector predictor of the current CU is subtracted from the actual motion vector of the current CU to produce a Motion Vector Difference (MVD) for the current CU. By doing so, there is no need to encode the motion

vector determined by the motion estimation unit 42 for each CU of a frame into the video bitstream, and the amount of data used for representing motion information in the video bitstream can be significantly decreased.

[00101] Like the process of choosing a predictive block in a reference frame during inter-frame prediction of a code block, a set of rules can be adopted by both the video encoder 20 and the video decoder 30 for constructing a motion vector candidate list (also known as a "merge list") for a current CU using those potential candidate motion vectors associated with spatially neighboring CUs and/or temporally co-located CUs of the current CU and then selecting one member from the motion vector candidate list as a motion vector predictor for the current CU. By doing so, there is no need to transmit the motion vector candidate list itself from the video encoder 20 to the video decoder 30, and an index of the selected motion vector predictor within the motion vector candidate list is sufficient for the video encoder 20 and the video decoder 30 to use the same motion vector predictor within the motion vector candidate list for encoding and decoding the current CU. Thus, only the index of the selected motion vector predictor needs to be sent from the video encoder 20 to the video decoder 30.

[00102] A brief discussion with respect to transform coefficient coding in a block-based video coding process (e.g., in an Enhanced Compression Model (ECM)) is provided herein. Specifically, each transform block is firstly divided into multiple coefficient groups (CGs), with each coefficient group includes transform coefficients of a 4×4 subblock for the luma component and a 2×2 subblock for the chroma components. The coding of the transform coefficients in the transform block is performed in the units of coefficient groups. For example, the coefficient groups inside the transform block are scanned and coded based on a first predetermined scan order. When coding each coefficient group, the transform coefficients of the coefficient group are scanned based on a second predetermined scan order within each subblock. In the ECM, the same top-left scan order is applied to scan the coefficient groups inside the transform block and different transform coefficients in each coefficient group (e.g., both the first and second predetermined scan orders are the top-left scan order). FIG. 5 is a graphical representation illustrating the top-left scan order of transform coefficients in a coefficient group in accordance with some examples. The numbers 0-15 in FIG. 5 indicates the corresponding scan order of each transform coefficient in the coefficient group.

[00103] According to a transform coefficient coding scheme in the ECM, a flag is firstly signaled for each transform block to indicate whether the transform block includes any non-zero transform coefficients. In a case where there is at least a non-zero transform coefficient in the transform block, a position of the last non-zero transform coefficient scanned according

to the top-left scan order is explicitly signaled from video encoder 20 to video decoder 30. With the position of the last non-zero transform coefficient being signaled, flags are further signaled for all the coefficient groups that are coded before the last coefficient group (i.e., the coefficient group including the last non-zero coefficient). Correspondingly, the flags indicate whether the respective coefficient groups include non-zero transform coefficients or not. If a flag of a coefficient group is equal to zero (indicating all the transform coefficients in the coefficient group are zeros), no further information needs to be sent for the coefficient group. Otherwise (e.g., the flag of the coefficient group is equal to one), an absolute value and a sign of each transform coefficient in the coefficient group are signaled in a bitstream according to the scan order. However, in existing designs, signs of the transform coefficients are bypass coded (e.g., no context model is applied), leading to inefficient transform coding in the current design. Consistent with the present disclosure, an improved LFNST process with sign prediction of the transform coefficients is described below in more detail, so that the transform coding efficiency can be improved.

[00104] FIG. 6 is a graphical representation illustrating an LFNST process in accordance with some examples. In the VVC, a secondary transform tool (e.g., an LFNST) is applied to compact the energy of transform coefficients of intra-coded blocks after a primary transform. As shown in FIG. 6, a forward LFNST 604 is applied between a forward primary transform 603 and a quantization 605 at video encoder 20, and an inverse LFNST 608 is applied between a de-quantization 607 and an inverse primary transform 609 at video decoder 30. For example, the LFNST process may include both forward LFNST 604 and inverse LFNST 608. As some examples, for a 4×4 forward LFNST 604, there may be 16 input coefficients; for an 8×8 forward LFNST 604, there may be 64 input coefficients; for a 4×4 inverse LFNST 608, there may be 8 input coefficients; and for an 8×8 inverse LFNST 608, there may be 16 input coefficients.

[00105] In forward LFNST 604, non-separable transforms with a varying transform size are applied based on a size of a coding block, which can be depicted using a matrix multiplication process. For example, assume that forward LFNST 604 is applied to a 4×4 block. Samples in the 4×4 block can be expressed using a matrix X as shown in the following expression (1):

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

[00106] The matrix X can be serialized as a vector $\vec{X}$ as shown in the following expression

(2):

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

[00107] In the above expression (1) or (2), X denotes a coefficient matrix obtained through forward primary transform 603, and X_{ij} denotes a primary transform coefficient in the matrix X. Then, forward LFNST 604 is applied as follows according to expression (3):

$$\vec{F} = T \cdot \vec{X} \qquad (3).$$

[00108] In the above expression (3), $\vec{F}$ denotes transform coefficients after forward LFNST 604, and T denotes a transform kernel (e.g., a transform matrix). In the above example, because X is a 4×4 matrix, T is a 16×16 matrix. The 16×1 vector $\vec{F}$ is subsequently re-organized as a 4×4 block (e.g., a coefficient matrix with a size of 4×4) according to a predetermined scan order. A first transform coefficient located before a second transform coefficient in the vector $\vec{F}$ may be associated with a smaller scanning index than the second transform coefficient in the 4×4 block.

[00109] In some implementations, a reduced non-separable transform kernel can be applied in the LFNST process. For example, based on the above expression (3), forward LFNST 604 is based on a direct matrix multiplication which is expensive in terms of computational operations and memory source to store transform coefficients. Therefore, a reduced non-separable transform kernel can be used in the LFNST design to reduce the implementation cost of the LFNST process, by mapping an N-dimension vector to a R-dimension vector in a different space, where $R < N$. For example, instead of using an $N \times N$ matrix for the transform kernel, a $R \times N$ matrix is used as the transform kernel in forward LFNST 604 as shown in expression (4):

$$T_{R\times N} = \begin{bmatrix} t_{0,0} & t_{0,1} & \cdots & t_{0,N-1} \\ t_{1,0} & t_{1,1} & \cdots & t_{1,N-1} \\ \vdots & \vdots & \ddots & t_{2,N-1} \\ t_{R-1,0} & t_{R-1,1} & \cdots & t_{R-1,N-1} \end{bmatrix} \qquad (4).$$

[00110] In the above expression (4), the R base vectors in $T_{R\times N}$ are generated by selecting the first R bases of the original N-dimension transform kernel (i.e., $N \times N$). Additionally, given that $T_{R\times N}$ is orthogonal, an inverse transform matrix for inverse LFNST 608 is the transpose of the forward transform matrix $T_{R\times N}$.

[00111] For 8×8 LFNST, if a factor $N/R = 4$ is applied, a 64×64 transform matrix is reduced to a 16×48 transform matrix for forward LFNST 604, and a 64×64 inverse transform matrix is reduced to a 48×16 inverse transform matrix for inverse LFNST 608. This is achieved by applying the LFNST process to an 8×8 subblock in the top-left region of primary transform

coefficients. Specifically, when the 16×48 forward LFNST is applied, it takes 48 transform coefficients from three 4×4 subblocks in the top-left 8×8 subblock (excluding its right-bottom 4×4 subblock) as input. In some examples, the LFNST process is restricted to be applicable only if all transform coefficients outside the top-left 4×4 subblock are zeros, which indicates that when LFNST is applied, all primary-only transform coefficients have to be zeros. Furthermore, to control complexity in a worst case (in terms of multiplications per pixel), the LFNST matrix for 4×4 and 8×8 coding blocks are forced to 8×16 and 8×48 transforms, respectively. For 4×M and M×4 coding blocks (M > 4), the non-separable transform matrix of the LFNST is 16×16.

[00112] In LFNST transform signaling, there are four transform sets in total, and two non-separable transform kernels per transform set are enabled in the LFNST design. A transform set is selected from the four transform sets according to an intra prediction mode of an intra block. The mapping from the intra prediction mode to the transform set is predetermined as shown in the following Table 1. If one of three Cross-Component Linear Model (CCLM) modes (e.g., INTRA_LT_CCLM, INTRA_T_CCLM, or INTRA_L_CCLM) is used for a current block (81 <= predModeIntra <= 83), a transform set "0" is selected for the current chroma block. For each transform set, the selected non-separable secondary transform candidate is indicated by signaling an LFNST index in a bitstream.

Table 1

IntraPredMode	**set**
IntraPredMode < 0	1
0 <= IntraPredMode <= 1	0
2 <= IntraPredMode <= 12	1
13 <= IntraPredMode <= 23	2
24 <= IntraPredMode <= 44	3
45 <= IntraPredMode <= 55	2
56 <= IntraPredMode<= 80	1
81 <= IntraPredMode<= 83	0

[00113] In some examples, because the LFNST is restricted to be applied to an intra block if all transform coefficients outside the first 16×16 subblock are zeros, the LFNST index signaling is dependent on the position of the last significant (i.e., non-zero) transform coefficient. For example, for 4×4 and 8×8 coding blocks, the LFNST index is only signaled when the position of the last significant transform coefficient is less than 8. For other coding block sizes, the LFNST index is only signaled when the position of the last significant transform coefficient is less than 16. Otherwise (i.e., the LFNST index is not signaled), the LFNST index is inferred to be zero, i.e., the LFNST is disabled.

[00114] Furthermore, to reduce a size of a buffer for caching transform coefficients, the LFNST is disallowed when either the width or the height of the current coding block is greater than the maximum transform size (i.e., 64) as signaled in a sequence parameter set (SPS). Meanwhile, the LFNST is only applied when the primary transform is DCT2. Besides, the LFNST is applied to intra coding blocks in both intra and inter slices and for both luma and chroma components. If a dual-tree or local-tree is enabled (where the partitions of luma and chroma components are misaligned), the LFNST indices are signaled separately for the luma and chroma components (i.e., luma and chroma components can apply different LFNST transforms). Otherwise, when a single-tree is applied (where the partitions of luma and chroma components are aligned), the LFSNT is only applied to the luma component with a single LFNST index being signaled.

[00115] The LFNST design in the ECM is like that in the VVC, except that additional LFNST kernels are introduced to provide better energy compaction of the residual samples of large block sizes. Specifically, when the width or height of the transform block is no smaller than 16, a new LFNST transform is introduced to the top-left region of the low-frequency transform coefficients generated from the primary transform. In the current ECM, as shown in FIG. 7, the low-frequency region includes six 4×4 subblocks in the top-left corner of the primary transform coefficients (e.g., six 4×4 subblocks shown with gray color in FIG. 7A). In this case, the number of coefficient inputs to forward LFNST 604 is 96. Additionally, to control the worst-case computational complexity, the number of coefficient outputs of forward LFNST 604 is set to 32.

[00116] Specifically, for W×H transform blocks where W >= 16 and H >= 16, a 32×96 forward LFNST is applied, which takes 96 transform coefficients from six 4×4 subblocks in the top-left region as input and outputs 32 transform coefficients. On the other side, the 8×8 LFNST in the ECM utilizes the transform coefficients in all four 4×4 subblocks as input and outputs 32 transform coefficients (i.e., a 32×64 matrix for forward LFNST 604 and a 64×32

matrix for inverse LFNST 608). This is different from the VVC where the 8×8 LFNST is only applied to the three 4x4 subblocks in the top left region and generates only 16 transform coefficients (i.e., a 16×48 matrix for forward LFNST 604 and a 48×16 matrix for inverse LFNST 608). Additionally, the total number of LFNST sets is increased from 4 in the VVC to 35 in ECM. Similar to the VVC, the selection of the LFNST set is dependent on the intra prediction mode of the current coding unit, and each LFNST set includes three different transform kernels.

[00117] In some examples, a multiple transform selection (MTS) scheme is applied for transforming residuals of both inter and intra coded blocks in addition to DCT2 transform used in HEVC. The MTS scheme uses multiple selected transforms from the DCT8 and DST7 transforms.

[00118] For example, two control flags are specified at a sequence level to enable the MTS scheme for intra and inter modes, separately. When the MTS scheme is enabled at the sequence level, another CU level flag is further signaled to indicate whether the MTS scheme is applied or not. In some implementations, the MTS scheme is only applied to the luma component. Furthermore, the MTS scheme is signaled only when the following conditions are satisfied: (a) both the width and height are smaller than or equal to 32; are (b) the Coded Block Flag (CBF) is equal to one. If the MTS CU flag is equal to zero, then DCT2 is applied in both horizontal and vertical directions. If the MTS CU flag is equal to one, then two other flags are additionally signaled to indicate the transform types for the horizontal and vertical directions, respectively. The mapping between the MTS horizontal and vertical control flags and the applied transforms is illustrated in the following Table 2.

Table 2

MTS_CU_flag	MTS_Hor_flag	MTS_Ver_flag	Intra/inter	
			Horizontal	Vertical
0			DCT2	
1	0	0	DST7	DST7
1	0	1	DCT8	DST7
1	1	0	DST7	DCT8
1	1	1	DCT8	DCT8

[00119] With respect to precision of a transform matrix, all the MTS transform coefficients are in 6-bit precision, which is the same as the DCT2 core transforms. Given that the VVC supports all the transform sizes used in the HEVC, all the transform cores used in the HEVC are kept to be the same as the VVC, including 4-point, 8-point, 16-point, and 32-point DCT-2 transforms and 4-point DST-7 transform. Meanwhile, other transform cores including 64-point DCT-2, 4-point DCT-8, 8-point, 16-point, 32-point DST-7 and DCT-8 are additionally supported in the VVC transform design. Furthermore, to reduce the complexity of the large size DST-7 and DCT-8, high frequency transform coefficients that are located outside the 16×16 low-frequency region are set to zeros (also known as zero-out) for DST-7 and DCT-8 transform blocks when either the width or height is equal to 32.

[00120] In the VVC, in addition to DCT2, only DST7 and DCT8 transform kernels are utilized for intra and inter coding. For intra coding, usually the statistical characteristics of the residual signal depend on the intra prediction mode. Additional primary transforms may be beneficial to handle the diversity of the residual characteristics.

[00121] Additional primary transforms including DCT5, DST4, DST1, and identity transform (IDT) are employed in the ECM. Also, the MTS set is made dependent on the TU size and intra mode information. 16 different TU sizes can be considered, and 5 different classes can be considered for each TU size depending on the intra-mode information. For each class, 4 different transform pairs are considered (the same as that of VVC). Although a total of 80 different classes may be considered, some of those different classes often share the same transform set. Thus, there are 58 (less than 80) unique entries in a resultant Lookup Table (LUT).

[00122] For angular modes, a joint symmetry over the TU shape and intra prediction is considered. Thus, a mode i (i>34) with a TU shape A×B may be mapped to the same class corresponding to the mode j=(68 – i) with a TU shape B×A. However, for each transform pair, the orders of the horizontal and vertical transform kernels are swapped. For example, a 16×4 block with mode 18 (horizontal prediction) and a 4×16 block with mode 50 (vertical prediction) are mapped to the same class, with the vertical and horizontal transform kernels being swapped. For the wide-angle modes, the nearest conventional angular mode is used for the transform set determination. For example, mode 2 is used for all the modes between -2 and -14. Similarly, mode 66 is used for mode 67 to mode 80.

[00123] Intra template matching prediction is an example of the intra prediction mode that copies a prediction block from a reconstructed part of a current frame, whose L-shaped template matches a current template. For a predetermined search range, video encoder 20

searches for the most similar template to the current template (e.g., based on an SAD cost) in a reconstructed part of the current frame, and uses the corresponding block as a prediction block. Video encoder 20 then signals the usage of this mode, and the same prediction operation is performed at a decoder side. A prediction signal is generated by matching an L-shaped causal neighbor of a current block with another block in a predetermined search area as shown in FIG. 8, which includes: (a) a current CTU (R1); (b) a top-left CTU (R2); (c) an above CTU (R3); and (d) a left CTU (R4). The intra template matching is enabled for CUs with a size less than or equal to 64 in width and height. Meanwhile, the intra template matching prediction mode is indicated by signaling a flag at a CU level. When the intra template matching is applied to a coding block with the width or height between 4 and 16 (inclusive), the primary transform that is applied to the corresponding dimension is set to DST-VII. Otherwise (i.e., the width or height is smaller than 4 or larger than 16), DCT-II is applied in the dimension.

[00124] FIG. 9 is a graphical representation illustrating an exemplary process of sign prediction in accordance with some examples. In some implementations, sign prediction can be performed to estimate signs of transform coefficients in a transform block from samples of its neighboring block. The correctness of the estimated signs can be coded according to a context model to indicate whether the sign prediction is correct. For example, in one context model, the difference between each estimated sign can be coded and a corresponding true sign with "0" (or "1") to indicate that the estimated sign is the same (or not the same) as the true sign. In a case when the signs can be accurately estimated in a high percentage (e.g., 90% or 95% of the signs are estimated correctly), the difference between the estimated signs and the true signs tends to be 0, which can be efficiently entropy-coded by CABAC when compared to the bypass-coded signs for the transform coefficients in the VVC. Consistent with the disclosure, other context models can be used to entropy-code the correctness of the sign prediction, to further improve the coding efficiency. These other exemplary context models will be described in more detail below.

[00125] In general, there is a high correlation among samples at the boundaries between a current block and its neighboring blocks, which can be utilized by a sign prediction scheme to predict the signs of the transform coefficients of the current block. As shown in FIG. 9, assume that there are M non-zero transform coefficients in the current block (with M signs each being either + or -). Then, a total number of possible combinations of the signs is 2^M. The sign prediction scheme generates a corresponding hypothesis (e.g., reconstructed samples at the top and left boundaries of the current block) using each combination of the signs, and compares the reconstructed samples in the corresponding hypothesis with extrapolated samples from

neighboring blocks to obtain a sample difference (e.g., SSD or SAD) between the reconstructed samples and the extrapolated samples. A combination of the signs (among the 2^M possible combinations of the signs) which minimizes the sample difference is selected as predicted signs in the current block.

[00126] In some implementations as shown in FIG. 9, in order to generate a corresponding hypothesis for each combination of the M signs, M corresponding transform coefficients can be processed by a de-quantization operation and an inverse transform to obtain residual samples. The residual samples can be added with the prediction samples to obtain reconstructed samples, which include the reconstructed samples at the top and left boundaries of the current block (shown in an L-shaped gray area 902).

[00127] In some implementations, a cost function which measures spatial discontinuity between the samples at the boundaries of the current block and its neighboring block is used for the selection of the combination of the signs. Instead of using the L2 norm (SSD), the cost function can be based on the L1 norm (SAD) as shown below in expression (5):

$$\text{costHV} = \sum_{n=0}^{N-1} \left| 2B_{-1,n} - B_{-2,n} - P_{0,n} \right| + \sum_{m=0}^{M-1} \left| 2C_{m,-1} - C_{m,-2} - P_{m,0} \right| \quad (5).$$

[00128] In the above expression (5), $B_{i,n}$ (with $i = -2, -1$) represents neighboring samples of the current block from its top neighboring block. $C_{m,j}$ (with $j = -2, -1$) represents neighboring samples of the current block from its left neighboring block. $P_{0,n}$ and $P_{m,0}$ represent corresponding reconstructed samples on the top and left boundaries of the current block, respectively. N and M represent a width and a height of the current block, respectively. FIG. 10 shows the corresponding samples $P_{0,n}$ and $P_{m,0}$ of the current block and the corresponding samples $B_{i,n}$ and $C_{m,j}$ of the neighboring blocks that are used to calculate the cost function for the sign prediction.

[00129] In some implementations, a template-based hypothesis reconstruction method can be applied in the sign prediction scheme to avoid the complexity of conducting multiple inverse transforms. Each template can be a set of reconstructed samples on the top and left boundaries of the current block, and can be acquired by applying the inverse transform to a coefficient matrix where a specific coefficient is set to one while all the other coefficients are equal to zeros. Given that the inverse transforms (e.g., DCT, DST) are linear, the corresponding hypothesis can be generated by a linear combination of a set of pre-calculated templates.

[00130] In some implementations, the predicted signs are classified into two sets and each

set is coded by a single CABAC context. For example, a first set includes the predicted signs of the transform coefficients at the top-left corner of a transform block, and a second set includes the predicted signs of the transform coefficients at all the other locations of the transform block.

[00131] Like HEVC, scalar quantization is used in the VVC. In some implementations, the scalar quantization in the VVC may be is implemented as a dependent scalar quantization. Dependent scalar quantization refers to an approach in which the set of admissible reconstruction values for a transform coefficient depends on the values of the transform coefficient levels that precede the current transform coefficient level in reconstruction order. The main effect of this approach is that, in comparison to conventional independent scalar quantization used in the HEVC, the admissible reconstruction vectors are packed more densely in the N-dimensional vector space (N represents the number of transform coefficients in a transform block). That means, for a given average number of admissible reconstruction vectors per N-dimensional unit volume, the average distortion between an input vector and its closest reconstruction vector is reduced.

[00132] The dependent scalar quantization can be implemented by: (a) defining two scalar quantizers with different reconstruction levels and (b) defining a process for switching between the two scalar quantizers. FIG. 11 illustrates two exemplary scalar quantizers used in a dependent scalar quantization of the VVC in accordance with some implementations of the present disclosure. As shown in FIG. 11, the two scalar quantizers, denoted by Q0 and Q1, are applied in the quantization design of the VVC. The location of the available reconstruction levels is uniquely specified by a quantization step size Δ. In this implementation, the selection between the two scalar quantizers Q0 and Q1 is not explicitly signalled in bitstream. Instead, the quantizer used for a current transform coefficient is determined by the parities of the transform coefficient levels that precede the current transform coefficient in coding order by video encoder 20 or in reconstruction order by video decoder 30.

[00133] In some implementations, the switching between the two scalar quantizers is done via a state machine. For example, FIG. 12A is a graphical representation illustrating state transition using a state machine with four states used in a dependent scalar quantization in accordance with some examples. As illustrated in FIG. 12, the states can take four different values of 0, 1, 2, 3. It is uniquely determined by the parities of the transform coefficient levels preceding the current transform coefficient in coding/reconstruction order.

[00134] In some implementations, at the start of the inverse quantization for a transform block, the state is set equal to 0. The transform coefficients are reconstructed in a scanning

order (i.e., in the same order they are entropy decoded).

[00135] After a current transform coefficient is reconstructed, the state is updated according to the state machine. For example, in FIG. 12A, k denotes the value of the transform coefficient level. At each state, the next state is determined based on the parity of the transform coefficient level k, i.e., (k & 1). The next state when (k & 1)==1 is different from the next state when (k & 1)==0. As shown in FIG. 12A, the state machine includes two arrows pointing to two different states from each of the four states. FIG. 12B is a table showing an exemplary quantizer selection according to the state transition of FIG. 12A in accordance with some examples. For example, according to FIG. 12A and FIG. 12B, at state 1, the next state will be 2 if (k & 1)==0 and 0 if (k & 1)==1.

[00136] Correspondingly, at the decoder, a reconstructed quantization index of one transform coefficient can calculated according to expression (6):

$$quantIdx = (abs(k) \ll 1) - (state \,\&\, 1) \qquad (6).$$

where abs() is the function of calculating the absolute value of the input and *state* is the current state of the state transition machine when parsing the level of the current transform coefficient. Further, on the decoder side, the reconstructed transform coefficient after de-quantization can be obtained according to expression (7):

$$transCoeff = quantIdx \cdot \Delta \qquad (7).$$

[00137] Several exemplary deficiencies that exist in a current design of the sign prediction scheme are identified herein. In a first example, the sign prediction in the current ECM is only applicable to the prediction of signs for transform coefficients in transform blocks where only primary transforms (e.g., DCT and DST transforms) are applied. As mentioned above, the LFNST can be applied to transform coefficients from a primary transform to provide better energy compaction of residual samples of intra-coded blocks. However, the sign prediction is bypassed for transform blocks to which the LFNST is applied in the current ECM design.

[00138] In a second example, a predetermined maximum number of predicted signs (denoted as "L_{max}") is determined for a transform block to control the complexity of the sign prediction. In the current ECM, a video encoder determines the value of the maximum number (e.g., L_{max} =8) based on tradeoff between the complexity and the coding efficiency, and sends the value to a video decoder. Moreover, for each transform block, the video encoder or decoder may scan all the transform coefficients in a raster-scan order and the first L_{max} non-zero

transform coefficients are selected as candidate transform coefficients for the sign prediction. Such equal treatment of the different transform coefficients in the transform block may not be optimal in terms of the accuracy of the sign prediction. For example, for transform coefficients with relatively large magnitudes, the prediction of their signs may have a better chance to achieve a correct prediction. This is because using incorrect signs for those transform coefficients tends to produce greater impacts on reconstructed samples on the block boundaries than that produced by using transform coefficients with relatively small magnitudes.

[00139] In a third example, instead of directly coding the explicit sign values, the video encoder or decoder can code the correctness of the predicted signs. For instance, for a transform coefficient which has a positive sign, if its predicted sign is also positive, only a bin "0" needs to be indicated in a bitstream from the video encoder to the video decoder. In this case, the predicted sign is the same as the true sign (or the original sign) of the transform coefficient, indicating that the sign prediction for this transform coefficient is correct. Otherwise (e.g., if the predicted sign is negative whereas the true sign is positive), a bin "1" may be included in the bitstream from the video encoder to the video decoder. If all the signs are predicted correctly, the corresponding bins indicated in the bitstream are zeros, which can be efficiently entropy-coded by CABAC. If some of the signs are predicted incorrectly, the corresponding bins indicated in the bitstream are ones. Although an arithmetic coding plus appropriate context model can be efficient to code the bins according to their corresponding probabilities, there are still non-negligible bits generated in the bitstream for the indication of the sign values.

[00140] In a fourth example, spatial discontinuity between samples at the boundaries between a current block and its neighboring block is used for the selection of the best sign prediction combination in the current design of sign prediction in ECM. An L1 norm of a gradient difference along a vertical direction and a horizontal direction are utilized to capture the spatial discontinuity. However, since the distribution of image signals is usually non-uniform, using only the vertical and horizontal directions may fail to capture the spatial discontinuity accurately.

[00141] Consistent with the present disclosure, a video processing method and system for sign prediction in block-based video coding are provided herein to address one or more of the above exemplary deficiencies. The method and system disclosed herein can improve the coding efficiency of the sign prediction while taking friendliness of hardware codec implementations into considerations. The method and system disclosed herein can improve the coding efficiency of transform blocks which apply the sign prediction technology to

transform coefficients of the blocks.

[00142] For example, as mentioned above, the sign prediction may predict signs of transform coefficients in a transform block based on a correlation among boundary samples (also referred to as border samples) located at the boundaries or close to the boundaries of the transform block and its spatial neighboring blocks. Given that an existence of the correlation is independent on which specific transform is applied, two coding tools (i.e., the LFNST and the sign prediction) do not interfere with each other and can be applied jointly. Additionally, because the LFNST further compacts the energy of transform coefficients of a primary transform, the sign prediction for transform coefficients of the LFNST can be more accurate than that for the primary transform. This is because an incorrect sign prediction of a transform coefficient from the LFNST can yield more discrepancy on the smoothness of the boundary samples. Thus, consistent with the present disclosure, a harmonization scheme is disclosed herein to enable a combination of the LFNST and the sign prediction to enhance the coding efficiency of the transform coefficient coding. Furthermore, a template-based hypothesis generation scheme is also disclosed herein to reconstruct border samples for different combinations of predicted signs in order to reduce the number of inverse transforms.

[00143] In another example, instead of giving equal treatment to different transform coefficients in a transform block for the selection of candidate transform coefficients for the sign prediction as described above, higher weights may be given to transform coefficients which can lead to discrepancy between the boundary samples of neighboring blocks, given that the signs of those transform coefficients are more easily predicted. Consistent with the present disclosure, the method and system disclosed herein can select candidate transform coefficients for the sign prediction (e.g., transform coefficients whose signs are to be predicted for a transform block) based on one or more selection criteria to improve the accuracy of the sign prediction. For example, transform coefficients which have more impacts on the reconstructed border samples (rather than transform coefficients which have less impacts on the reconstructed border samples) are selected as candidate transform coefficients for the sign prediction, so that the accuracy of the sign prediction can be improved.

[00144] In yet another example, when signs of transform coefficients in a transform block are predicted with a high accuracy rate (e.g., correctness of the predicted signs is higher than a threshold such as 80% or 90%), a strong correlation exists between the boundary samples of the transform block and its neighboring blocks. In this case, it commonly occurs that there may be consecutive transform coefficients (e.g., especially for a few of non-zero transform

coefficients at the beginning of the transform block) which can be correctly predicted for most scenarios. In such scenarios, a single bin (instead of multiple bins) can be used to indicate whether the signs of all the consecutive transform coefficients are correctly predicted or not to save a signaling overhead of the sign prediction. Consistent with the present disclosure, a vector-based sign prediction scheme is disclosed herein to reduce the signaling overhead of the sign prediction. Different from an existing sign prediction which predicts a sign of each non-zero transform coefficient separately, the disclosed vector-based sign prediction scheme groups a set of consecutive non-zero candidate transform coefficients and predict their corresponding signs together such that an average number of bins (or bits) used to indicate the correctness of the predicted signs can be efficiently reduced.

[00145] In still yet another example, using only the vertical and horizontal directions may fail to capture the spatial discontinuity between samples at the boundaries between the current block and its neighboring block accurately. Thus, more directions may be introduced to capture the spatial discontinuity more accurately. Consistent with the present disclosure, an improved cost function is disclosed herein which considers both the gradients of the vertical and horizontal directions and the gradients of diagonal directions to capture the spatial discontinuity more accurately.

[00146] FIG. 13 is a block diagram illustrating an exemplary sign prediction process 1100 in block-based video coding in accordance with some implementations of the present disclosure. In some implementations, sign prediction process 1300 may be performed by transform processing unit 52. In some implementations, sign prediction process 1300 may be performed by one or more processors (e.g., one or more video processors) of video encoder 20 or decoder 30. Throughout the present disclosure, the LFNST is used as an example of the secondary transform without loss of generality. It is contemplated that other examples of the secondary transform can also be applied herein.

[00147] In an existing design of ECM, sign prediction is disabled for a transform block to which the LFNST is applied. However, the principle of the sign prediction is to predict signs of transform coefficients based on the correlation between border samples of the transform block and its spatial neighboring blocks, which does not depend on specific transform types (e.g., whether it is a primary transform or secondary transform) or transform cores (e.g., whether it is DCT or DST) applied to the transform block. Therefore, the sign prediction and the LFNST can be applied together to further improve the efficiency of the transform coding herein. Consistent with the present disclosure, sign prediction process 1300 can be applied to predict signs of transform coefficients in the transform block where a primary transform and a

secondary transform are applied jointly.

[00148] An exemplary overview of sign prediction process 1300 is provided herein. Initially, sign prediction process 1300 may perform a coefficient generation operation 1302 by applying a primary transform and a secondary transform to a transform block of a video frame from a video to generate transform coefficients for the transform block. Next, sign prediction process 1300 may perform a coefficient selection operation 1304 by selecting a set of candidate transform coefficients from the transform coefficients for the sign prediction. Subsequently, sign prediction process 1300 may perform a hypothesis generation operation 1306 by applying a template-based hypothesis generation scheme to select a hypothesis from a plurality of candidate hypotheses for the set of candidate transform coefficients. Additionally, sign prediction process 1300 may perform a sign generation operation 1108 by determining a combination of sign candidates associated with the selected hypothesis to be a set of predicted signs for the set of candidate transform coefficients. Operations 1302, 1304, 1306, and 1308 are respectively described below in more detail.

[00149] For example, transform processing unit 52 of video encoder 20 may transform residual video data into transform coefficients of a transform block by applying the primary transform and the secondary transform jointly (e.g., as shown in FIG. 6 where forward primary transform 603 and forward LFNST 604 are applied together). A predetermined number (e.g., L) of non-zero transform coefficients from the transform coefficients of the transform block can be selected as candidate transform coefficients based on one or more selection criteria as described below, with $1 \leq L \leq$ a maximum number of signs that can be predicted. Next, by applying a template-based hypothesis generation scheme, a plurality of candidate hypotheses can be generated using different combinations of sign candidates for the L candidate transform coefficients, respectively, which may result in a total of 2^L candidate hypotheses. Each candidate hypothesis may include reconstructed samples on the top and left boundaries of the transform block. Then, a cost can be calculated for each candidate hypothesis reconstruction using a cost function that incorporates a combined gradient along horizontal, vertical, and diagonal directions. A candidate hypothesis associated with a minimal cost from the plurality of candidate hypotheses can be determined as a hypothesis for predicting the signs of the L candidate transform coefficients. For example, a combination of sign candidates used to generate the candidate hypothesis associated with the minimal cost is used as the predicted signs for the L candidate transform coefficients.

[00150] To begin with, sign prediction process 1300 may perform coefficient generation operation 1302, in which the primary transform (e.g., DCT, DST, etc.) and the secondary

transform (e.g., LFNST) may be applied to a transform block jointly to generate transform coefficients for the transform block. For example, the primary transform may be applied to the transform block to generate primary transform coefficients for the transform block. Then, the LFNST may be applied to the transform block to generate LFNST transform coefficients based on the primary transform coefficients.

[00151] Sign prediction process 1300 may continue to perform coefficient selection operation 1304, in which a set of candidate transform coefficients for the sign prediction can be selected from the transform coefficients of the transform block based on one or more selection criteria. Through the selection of the candidate transform coefficients, the number of the candidate transform coefficients that can be correctly predicted can be maximized, and the accuracy of the sign prediction can be improved.

[00152] In some implementations, the set of candidate transform coefficients can be selected from the transform coefficients of the transform block based on magnitudes of the transform coefficients. For example, the set of candidate transform coefficients can include one or more transform coefficients that have larger magnitudes than remaining transform coefficients in the transform block.

[00153] In general, for transform coefficients with larger magnitudes, predicted signs of these transform coefficients are more likely to be correct. This is because these transform coefficients with larger magnitudes tend to have more impacts on the quality of the reconstructed samples, and using incorrect signs for these transform coefficients may be more likely to generate discontinuities between the boundary samples of the transform block and its spatial neighboring blocks. Based on this rationale, the set of candidate transform coefficients for the sign prediction can be selected from the transform coefficients of the transform block based on magnitudes of the non-zero transform coefficients in the transform block.

[00154] There could be multiple ways to implement the transform coefficient magnitude based reordering scheme for sign prediction. In a first implementation, the scheme can directly use the transform coefficients that are reconstructed after the de-quantization (i.e., dequantized transform coefficients) to the sorting such that the dequantized transform coefficients with larger magnitudes will be placed ahead the ones with smaller magnitudes for the sign prediction. For example, all the non-zero transform coefficients in the transform block can be scanned and sorted to form a coefficient list according to the descending order of their magnitudes. A transform coefficient with the largest magnitude can be selected from the coefficient list and placed as a first candidate transform coefficient in the set of candidate transform coefficients, a transform coefficient with the second largest magnitude can be selected from the coefficient

list and placed as a second candidate transform coefficient in the set of candidate transform coefficients, so on and so forth until the number of the selected candidate transform coefficients reaches the predetermined number L.

[00155] In a second implementation, instead of directly using the dequantized transform coefficients, the quantization index (e.g., *quantIdx* as obtained according to expression (6)) of each transform coefficient can be used to represent the magnitude of the transform coefficient and used for such sorting. As shown in expression (7), because the value of one dequantized transform coefficient is equal to the production of its quantization index *quantIdx* and the corresponding step size Δ and the step size is the same for the dequantization of all the transform coefficients in one transform block, the two implementations are actually mathematically identical. However, given that the quantization index *quantIdx* can be obtained at the parsing stage (which is earlier than the acquisition of the dequantized transform coefficients), the second implementation may be provide certain benefits when implemented by some specific hardware.

[00156] In some implementations, the set of candidate transform coefficients can be selected from the transform coefficients of the transform block based on a coefficient scan order of entropy coding applied in the video coding. Since natural video content may have abundant low-frequency information, magnitudes of non-zero transform coefficients obtained from the processing of the video content tend to be larger at low frequency positions and smaller towards high frequency positions. Thus, a coefficient scan order (such as a zig-zag scan, a top-left scan, a horizontal scan, or a vertical scan, etc.) can be used in modern video codecs to scan the transform coefficients in the transform block for entropy coding. By using this coefficient scan order, transform coefficients with larger magnitudes (usually corresponding to lower frequencies) are scanned before transform coefficients with smaller magnitudes (usually corresponding to higher frequencies). Based on this rationale, the set of candidate transform coefficients for the sign prediction disclosed herein can be selected from the transform coefficients of the transform block based on the coefficient scan order for entropy coding. For example, a coefficient list can be obtained by scanning all the transform coefficients in the transform block using the coefficient scan order. Then, the first L non-zero transform coefficients in the coefficient list can be automatically selected as the set of candidate transform coefficients for the sign prediction.

[00157] In some implementations, for an intra coded block, the set of candidate transform coefficients for the sign prediction can be selected from the transform coefficients of the block

based on an intra prediction direction of the block. For example, multiple scan orders coherent with the intra prediction directions (e.g., the 67 intra prediction directions in the VVC and ECM) can be determined and stored as look-up-tables at both video encoder 20 and video decoder 30. When coding transform coefficients of an intra block, video encoder 20 or video decoder 30 may identify a scan order from the scan orders that is closest to an intra prediction of the intra block. Video encoder 20 or video decoder 30 may use the identified scan order to scan all the non-zero transform coefficients of the intra block to obtain a coefficient list, and select the first L non-zero transform coefficients from the coefficient list as the set of candidates transform coefficients.

[00158] In some implementations, video encoder 20 may determine a scan order for the transform coefficients of the transform block and signal the determined scan order to video decoder 30. One or more new syntax elements which indicate the determined scan order can be signaled through the bitstream. For example, multiple fixed scan orders (e.g., for different transform block sizes and coding modes) can be predetermined by video encoder 20 and pre-shared with video decoder 30. Then, after selecting a scan order from the fixed scan orders, video encoder 20 only needs to signal a single index to indicate the selected scan order to video decoder 30. In another example, the one or more new syntax elements may be used to enable the signaling of any selected scan order of the transform coefficients. In some implementations, the one or more syntax elements can be signaled at various coding levels, e.g., a sequence parameter set (SPS), a picture parameter set (PPS), a picture (or slice) level, a CTU (or CU) level, etc.

[00159] In some implementations, the set of candidate transform coefficients can be selected from the transform coefficients of the transform block based on influence scores of the transform coefficients on reconstructed border samples of the transform block. Specifically, as shown in the above expression (5), the selection of the combination of the signs (i.e., predicted signs, or sign predictors) are based on a cost function to minimize the discontinuity of the gradients of the samples between the current transform block and its spatial neighboring blocks. Therefore, signs of transform coefficients with relatively large influence on the reconstructed samples on the top and left borders of the current transform block tend to be more likely to be accurately predicted, because the flipping of these signs may result in large variations on the smoothness between the boundary samples as calculated in (5). To maximize a percentage of accurate sign predictions, the signs of these transform coefficients (i.e., the ones with more impacts on reconstructed border samples) may be predicted before the other transform coefficients (i.e.,

the ones with less impacts on the reconstructed border samples). Based on this rationale, the set of candidate transform coefficients for the sign prediction disclosed herein can be selected based on their influence scores on the reconstructed samples on the top and left borders of the current transform block.

[00160] For example, video encoder 20 or decoder 30 may sort all the transform coefficients based on the measurement of their corresponding influence scores to the reconstructed border samples of the transform block. If a transform coefficient has a larger influence score to the reconstructed border samples, it may be assigned with a smaller index in a sign prediction candidate list because it is easier to be correctly predicted. The set of candidate transform coefficients disclosed herein can be L transform coefficients with the L smallest indices in the sign prediction candidate list.

[00161] In some implementations, different criteria may be applied to quantify an influence score of a transform coefficient on the reconstructed border samples. For example, a value which measures the energy of variations on the reconstructed border samples caused by the transform coefficient can be used as the influence score, which can be obtained as follows (in the L1 norm):

$$\mathrm{V} = \left|C_{i,j}\right| \cdot \left(\sum_{n=0}^{N-1} \left|T_{i,j}(0,n)\right| + \sum_{m=0}^{M-1} \left|T_{i,j}(m,0)\right| \right) \tag{8}$$

[00162] In the above expression (6), $C_{i,j}$ represents a transform coefficient at a position *(i, j)* in the transform block. $T_{i,j}(l,k)$ represents a corresponding border sample at a position *(l,k)* of a template that is associated with the transform coefficient $C_{i,j}$. N and M represent a width and a height of the transform block, respectively. V represents an influence score of the transform coefficient at the position *(i, j)*.

[00163] In another example, the L1 norm in the above expression (8) can be replaced by the L2 norm, so that the influence score (e.g., the measurement on the energy of the variations on the reconstructed border samples caused by the transform coefficient) can be calculated using the L2 norm as follows:

$$\mathrm{V} = \left|C_{i,j}\right| \cdot \left(\sum_{n=0}^{N-1} \left|T_{i,j}(0,n)\right|^2 + \sum_{m=0}^{M-1} \left|T_{i,j}(m,0)\right|^2 \right) \tag{9}$$

[00164] Consistent with the present disclosure, although only top and left boundary samples (e.g., as indicated by $T_{i,j}(0,n)$ and $T_{i,j}(m,0)$) are used for the calculation in the above expressions (8) and (9), the transform coefficient selection scheme disclosed herein can also be

applied to any sign prediction scheme by changing the reconstructed samples of the current transform block that are used in a corresponding cost function.

[00165] Similar to the transform coefficient magnitude base scheme, there could be different ways to implement the influence scores based scheme. In a first implementation, the schemes can directly use the magnitudes of the dequantized transform coefficients $C_{i,j}$, as shown in expressions (8) and (9). In a second implementation, the quantization index values *quantIdx* can be applied instead to replace the dequantized transform coefficients $C_{i,j}$ to calculate the corresponding cost measurement. Specifically, when quantization index values *quantIdx* are applied, expressions (8) and (9) become expressions (10) and (11) below:

$$\mathrm{M} = \left|quantIdx_{i,j}\right| \cdot \left(\sum_{n=0}^{N-1}\left|T_{i,j}(0,n)\right| + \sum_{m=0}^{M-1}\left|T_{i,j}(m,0)\right|\right) \tag{10}$$

$$\mathrm{M} = \left|quantIdx_{i,j}\right| \cdot \left(\sum_{n=0}^{N-1}\left|T_{i,j}(0,n)\right|^2 + \sum_{m=0}^{M-1}\left|T_{i,j}(m,0)\right|^2\right) \tag{11}$$

where $quantIdx_{i,j}$ represents the quantization index of the transform coefficient at position *(i, j)* in the transform block.

[00166] Because the value of one dequantized transform coefficient is equal to the production of its quantization index *quantIdx* and the corresponding step size Δ and the step size is the same for the dequantization of all the transform coefficients in one transform block, the two methods are actually mathematically identical.

[00167] Sign prediction process 1300 may continue to perform hypothesis generation operation 1106, in which a template-based hypothesis generation scheme can be applied to select a hypothesis for the set of candidate transform coefficients from a plurality of candidate hypotheses. Initially, a plurality of combinations of sign candidates can be determined for the set of candidate transform coefficients based on a total number of coefficients included in the set of candidate transform coefficients. For example, if there are L candidate transform coefficients in total, then the plurality of combinations of sign candidates for the set of candidate transform coefficients can be 2^L combinations of sign candidates. Each sign candidate can be either a negative sign (-) or a positive sign (+). Each combination of sign candidates may include a total of L negative or positive signs. For example, if L=2, then the plurality of combinations of sign candidates may include 2^2=4 combinations of sign candidates, which are (+, +), (+, -), (-, -), and (-, -), respectively.

[00168] Next, a template-based hypothesis generation scheme can be applied to generate the plurality of candidate hypotheses for the plurality of combinations of sign candidates,

respectively. To reduce the complexity of inverse primary and secondary transforms that need to be performed, the template-based hypothesis generation scheme disclosed herein can be used to optimize the generation of reconstructed border samples of the transform block. Two exemplary approaches for implementing the template-based hypothesis generation scheme are disclosed herein. It is contemplated that other exemplary approaches for implementing the template-based hypothesis generation scheme are possible, which is not limited herein.

[00169] In a first exemplary approach, a corresponding candidate hypothesis for each combination of sign candidates can be generated based on a linear combination of templates, so that a plurality of candidate hypotheses can be generated for the plurality of combinations of sign candidates, respectively. Each template may correspond to a candidate transform coefficient from the set of candidate transform coefficients. Each template may represent a group of reconstructed samples on the top and left boundaries of the transform block. Each template can be generated by applying an inverse secondary transform and an inverse primary transform to the transform block, where each of the set of candidate transform coefficients is set to zero except the candidate transform coefficient corresponding to the template which is set to one (e.g., the candidate transform coefficient corresponding to the template is set to be 1, whereas each of the remaining candidate transform coefficients is set to be 0).

[00170] For example, the corresponding candidate hypothesis for each combination of sign candidates can be set to be a linear combination of templates. For a template that corresponds to a respective candidate transform coefficient, a respective weight for the template can be set to be a magnitude of a dequantized transform coefficient corresponding to the respective candidate transform coefficient. An example of the hypothesis generation based on a linear combination of templates is illustrated in FIG. 14, which is described below in more detail.

[00171] To predict the signs of the candidate transform coefficients, video encoder 20 or decoder 30 may go through all the candidate hypotheses before identifying a hypothesis associated with a combination of sign candidates which can minimize a cost value calculated from a cost function. In the first exemplary approach described above, each candidate hypothesis can be generated based on a combination of multiple templates, which is relatively complicate when considering the per-sample calculations (e.g., additions, multiplications, and shifts) involved in such combination. In order to reduce the computation complexity associated with the identification of the hypothesis that minimizes the cost value calculated from the cost function, a second exemplary approach is introduced herein.

[00172] In the second exemplary approach, the plurality of combinations of sign candidates associated with the plurality of candidate hypotheses can be treated as a plurality of hypothesis

indices for the plurality of candidate hypotheses, respectively. For example, digitals 0 and 1 can be configured to represent the positive sign (+) and the negative sign (-), respectively. A combination of sign candidates corresponding to a candidate hypothesis can be used as a unique representation (i.e., a hypothesis index) for the candidate hypothesis. For instance, assume that there are three signs to be predicted (e.g., L=3). A hypothesis index 000 may represent a candidate hypothesis which is generated by setting all three sign candidates to be positive (e.g., the three sign candidates being (+, +, +)). Similarly, a hypothesis index 010 may represent a candidate hypothesis which is generated by setting the first sign candidate and the third sign candidate to be positive while setting the second sign candidate to be negative (e.g., the three sign candidates being (+, -, +)).

[00173] Then, the plurality of candidate hypotheses can be generated based on a Gary code order of the plurality of hypothesis indices, such that reconstructed samples of a previous candidate hypothesis with a previous hypothesis index can be used to generate a current candidate hypothesis with a current hypothesis index. The current hypothesis index of the current candidate hypothesis can be immediately after the previous hypothesis index of the previous candidate hypothesis in the Gary code order of the plurality of hypothesis indices. The current hypothesis index can be generated by changing a sign candidate associated with the previous hypothesis index from positive (or negative) to negative (or positive). For example, the current hypothesis index can be obtained by changing a single "0" (or "1") in the previous hypothesis index to "1" (or "0").

[00174] For example, the plurality of hypothesis indices can be reordered based on the Gary code order of the plurality of hypothesis indices to generate a reordered sequence of hypothesis indices. For a first hypothesis index in the reordered sequence of hypothesis indices, a first candidate hypothesis corresponding to the first hypothesis index can be generated by applying an inverse secondary transform and an inverse primary transform to the transform block where each of the set of candidate transform coefficients is set to one. For a second hypothesis index in the reordered sequence of hypothesis indices that is immediately after the first hypothesis index, a second candidate hypothesis corresponding to the second hypothesis index can be generated based on (a) the first candidate hypothesis corresponding to the first hypothesis index and (b) an adjusting term for the second candidate hypothesis. The following Table 3 illustrates an exemplary process of generating all the candidate hypotheses for LFNST when the number of the candidate transform coefficients is 3 (e.g., L=3).

Table 3

Sign Candidates	Hypothesis Index	Candidate Hypothesis	Hypothesis generation
(+, +, +)	000	H000	Applying inverse transform to the coefficient matrix where all the sign candidates are set to be positive.
(+, +, -)	001	H001	H000 – C_2 * T001
(+, -, -)	011	H011	H001 – C_1 * T010
(+, -, +)	010	H010	H011 + C_2 * T001
(-, -, +)	110	H110	H010 – C_0 * T100
(-, -, -)	111	H111	H110 – C_2 * T001
(-, +, -)	101	H101	H111 + C_1 * T010
(-, +, +)	100	H100	H101 – C_2 * T001

[00175] In the above Table 3, a first column illustrates 2^3=8 combinations of sign candidates, respectively. A second column illustrates hypothesis indices corresponding to the combinations of sign candidates, respectively, by using digitals 0 and 1 to represent the positive sign (+) and the negative sign (-), respectively. The hypothesis indices in the second column are ordered according to the order of the Gary codes (e.g., in the order of 000, 001, 011, 010, 110, 111, 101, 100). A third column illustrates candidate hypotheses corresponding to the combinations of sign candidates and the hypothesis indices, respectively. A fourth column illustrates calculations of the candidate hypotheses, respectively.

[00176] In Table 3, TXYZ in the fourth column represents a corresponding template (i.e., reconstructed samples on the top and left boundaries of the transform block), which can be generated by applying an inverse transform to a coefficient matrix of the transform block where a specific transform coefficient is set to one while all the other transform coefficients are equal to zeros. For example, T100 represents a corresponding template which is generated by applying an inverse transform to the coefficient matrix where only the transform coefficient corresponding to the first sign candidate is set to one while all the transform coefficients in the coefficient matrix are set to zeros. C_0, C_1, and C_2 represent absolutes of dequantized transform coefficients that are associated with the first, second, and third sign candidates, respectively.

[00177] Referring to Table 3, for a first hypothesis index 000, a first candidate hypothesis H000 be generated by applying an inverse secondary transform and an inverse primary transform to a coefficient matrix associated with the transform block where each of the candidate transform coefficients is set to one. For a second hypothesis index 001 that is immediately after the first hypothesis index 000, a second candidate hypothesis H001 can be generated based on (a) the first candidate hypothesis H000 and (b) an adjusting term for the second candidate hypothesis (e.g., $-C_2$*T001). Similarly, for a third hypothesis index 011 that is immediately after the second hypothesis index 001, a third candidate hypothesis H011 can be generated based on (a) the second candidate hypothesis H001 and (b) an adjusting term for the third candidate hypothesis (e.g., $-C_1$*T010). For a fourth hypothesis index 010 that is immediately after the third hypothesis index 011, a fourth candidate hypothesis H010 can be generated based on (a) the third candidate hypothesis H011 and (b) an adjusting term for the fourth candidate hypothesis (e.g., C_2*T001).

[00178] Subsequently, a hypothesis associated with a minimal cost can be determined from the plurality of candidate hypotheses based on a cost function that incorporates a combined gradient along horizontal, vertical, and diagonal directions. As described above, if the cost function only utilizes a gradient of the horizontal and vertical directions (e.g., as shown above in the expression (5)), it may not work well for highly non-uniform image signals. Consistent with the present disclosure, a gradient along one or more diagonal directions is also utilized to improve the accuracy of the cost function. For example, two diagonal directions including a left diagonal direction and a right diagonal direction can also be incorporated into the cost function. For example, cost functions for the two diagonal directions can be described according to the following expressions (12) and (13):

$$\text{costD1} = \sum_{n=0}^{N-1} \left|2B_{-1,n-1} - B_{-2,n-2} - P_{0,n}\right| + \sum_{m=0}^{M-1} \left|2C_{m-1,-1} - C_{m-2,-2} - P_{m,0}\right| \quad (12),$$

$$\text{costD2} = \sum_{n=0}^{N-1} \left|2B_{-1,n+1} - B_{-2,n+2} - P_{0,n}\right| + \sum_{m=0}^{M-1} \left|2C_{m+1,-1} - C_{m+2,-2} - P_{m,0}\right| \quad (13).$$

[00179] In the above expression (8) or (9), $B_{-1,n-1}$, $B_{-2,n-2}$, $B_{-1,n+1}$, and $B_{-2,n+2}$ represent neighboring samples of the transform block from its top neighboring block. $C_{m-1,-1}$, $C_{m-2,-2}$, $C_{m+1,-1}$, and $C_{m+2,-2}$ represent neighboring samples of the transform block from its left neighboring block. $P_{0,n}$ and $P_{m,0}$ represent reconstructed samples on the top and left boundaries of the transform block, respectively. N and M represent a width and a height of the

transform block, respectively. costD1 and costD1 represent a left-diagonal cost function and a right-diagonal cost function for the left diagonal direction and the right diagonal direction, respectively.

[00180] The two cost functions of the diagonal directions (e.g., costD1 and costD2) can be used jointly with that of a horizontal-and-vertical cost function (e.g., costHV shown in the above expression (5)). Then, the cost function for the sign prediction can be determined based on the horizontal-and-vertical cost function that incorporates a gradient along the horizontal and vertical directions, the left-diagonal cost function that incorporates a gradient along a left diagonal direction, and the right-diagonal cost function that incorporates a gradient along a right diagonal direction. For example, the cost function may be a weighted sum of the horizontal-and-vertical cost function, the left-diagonal cost function, and the right-diagonal cost function as described in expression (14):

$$\text{cost} = \text{costHV} + \omega(\text{costD1} + \text{costD2}) \tag{14}$$

[00181] In the above expression (14), ω denotes a weight for the left-diagonal cost function and the right-diagonal cost function.

[00182] In another example, the cost function may be a minimum of the horizontal-and-vertical cost function, the left-diagonal cost function, and the right-diagonal cost function as described in expression (15):

$$\text{cost} = \min\{\text{costHV}, \text{costD1}, \text{costD2}\} \tag{15}$$

[00183] Compared with the expression (5) shown above, the cost function in the expression (14) or (15) disclosed herein may need more neighboring pixels to support the cost functions costD1, costD2 along the diagonal directions, which is described below in more detail with reference to FIGs. 16A-16B.

[00184] In some implementations, a corresponding cost for each candidate hypothesis may be determined using the above expression (14) or (15). Then, a plurality of costs can be calculated for the plurality of candidate hypotheses, respectively. A minimal cost among the plurality of costs can be determined. A candidate hypothesis associated with the minimal cost can be determined from the plurality of candidate hypotheses and selected to be a hypothesis for the sign prediction.

[00185] Sign prediction process 1300 may continue to perform sign generation operation 1108, in which a combination of sign candidates associated with the selected hypothesis is determined to be a set of predicted signs for the set of candidate transform coefficients. For

example, a combination of sign candidates (e.g., L sign candidates) used to generate the selected hypothesis can be used as the predicted signs for the L candidate transform coefficients.

[00186] In some implementations, sign generation operation 1308 may also include applying a vector-based sign prediction scheme to the set of predicted signs to generate a sequence of sign signaling bits for the set of candidate transform coefficients. A bitstream including the sequence of sign signaling bits may be generated by video encoder 20 and stored in storage device 32 of FIG. 1. Alternatively or additionally, the bitstream may be transmitted to video decoder 30 through link 16 of FIG. 1.

[00187] As described above, when signs of transform coefficients in a transform block are well predicted, it is very likely that signs of multiple consecutive transform coefficients can be correctly predicted. In this case, since a signaling scheme in existing sign prediction designs needs to signal multiple bins of "0" to separately indicate that a corresponding sign of each transform coefficient can be correctly predicted, it is obviously inefficient in terms of an overhead for signaling sign values of the transform block. An exemplary implementation of an existing sign prediction scheme is described below in more detail with reference to FIG. 15A.

[00188] Consistent with the present disclosure, the efficiency of sign signaling can be improved by applying the vector-based sign prediction scheme disclosed herein. Specifically, the candidate transform coefficients for the transform block can be divided into multiple groups and signs of candidate transform coefficients in each group can be predicted together. In this case, when original signs (or true signs) of candidate transform coefficients in a group are the same as their predicted signs respectively, only a bin with a value of "0" needs to be sent in a bitstream to indicate that all the signs in the group are correctly predicted. Otherwise (i.e., there is at least a candidate transform coefficient in the group whose original sign is different from the predicted sign), a bin with a value of "1" may be firstly signaled in the bitstream to indicate that not all the signs for the candidate transform coefficients in the group are correctly predicted. Then, additional bins may also be signaled in the bitstream from video encoder 20 to video decoder 30 to inform the corresponding correctness of each predicted sign in the group separately. An exemplary implementation of the vector-based sign prediction scheme disclosed herein is described below in more detail with reference to FIG. 15B.

[00189] In some implementations, the set of candidate transform coefficients can be divided into a plurality of groups of candidate transform coefficients. For each group of candidate transform coefficients, one or more sign signaling bits may be generated for the group of candidate transform coefficients to indicate the correctness of the predicted signs.

[00190] In one example, the sign signaling bits may be generated based on whether original signs of the group of candidate transform coefficients are identical to predicted signs of the group of candidate transform coefficients. Responsive to the original signs of the group of candidate transform coefficients being identical to the predicted signs of the group of candidate transform coefficients, a bin with a value of zero ("0") may be generated and added as a sign signaling bit to the bitstream. For example, the bitstream may include a "0" to indicate that the predicted signs of the group of candidate transform coefficients are correctly predicted.

[00191] On the other hand, responsive to the original signs of the group of candidate transform coefficients being not identical to the predicted signs of the group of candidate transform coefficients, a bin with a value of one ("1") may be generated. A set of additional bins for informing corresponding correctness of the predicted signs of the group of candidate transform coefficients may also be generated. Then, the bin with the value of one and the set of additional bins can be added as sign signaling bits to the bitstream. For example, the set of additional bins can be an XOR result of the original signs and the predicted signs of the group of candidate transform coefficients. An additional bin with a value of "0" may indicate that a predicted sign of a candidate transform coefficient corresponding to the additional bin is correctly predicted, whereas an additional bin with a value of "1" may indicate that a predicted sign of a candidate transform coefficient corresponding to the additional bin is incorrectly predicted. The bitstream may include: (a) a "1" to indicate that the predicted signs of the group of candidate transform coefficients are not correctly predicted; and (b) the set of additional bins to indicate which predicted signs are correctly predicted and which predicted signs are incorrectly predicted.

[00192] The sign signaling bits can be similarly generated using other context models. For example, the sign signaling bits may include a "0" to indicate that the predicted signs of the group of candidate transform coefficients are correctly predicted, and a bin with a value of one ("1") to indicate the predicted signs are not correct. A set of additional bins for informing corresponding correctness of the predicted signs of the group of candidate transform coefficients with incorrect sign prediction may also be generated.

[00193] In some implementations, a size of each group of candidate transform coefficients can be adaptively changed based on one or more predetermined criteria. The one or more predetermined criteria may include a width or height of the transform block, a coding mode of the transform block (e.g., intra or inter coding), and the number of non-zero transform coefficients within the transform block, etc. In some implementations, the size of each group of candidate transform coefficients can be signaled in the bitstream at various coding levels

such as SPS, PPS, a slice or picture level, a CTU or CU level, or a transform block level.

[00194] In some implementations, one or more constraints may be applied to limit the application scenarios of the vector-based sign prediction scheme disclosed herein. For example, the vector-based sign prediction scheme disclosed herein can be applied to process signs for a first part of the transform coefficients in the transform block, whereas signs of a second part of the transform coefficients in the transform block can be processed using an existing sign prediction scheme. In a further example, the vector-based sign prediction scheme disclosed herein can be applicable to the first N (e.g., $N = 2, 3, 4, 5, 6, 7,$ or 8, etc.) non-zero candidate transform coefficients from the transform block, whereas signs of other candidate transform coefficients form the transform block can be processed using an existing sign prediction scheme shown in FIG. 15A, which will be described later in this disclosure.

[00195] Consistent with the present disclosure, sign prediction process 1100 disclosed herein may be disabled under some scenarios. For example, when the LFNST is applied to coding blocks that are coded by intra template matching mode, the primary transform can be DST-VII. Given that LFNST core transforms in ECM are mainly trained when the primary transform is DCT-II, the corresponding LFNST transform coefficients of intra template matching blocks may show different characteristics when compared to that of other LFNST blocks. Based on this rationale, sign prediction process 1300 may be disabled when a current coding block is an intra template matching block and is coded using the LFNST.

[00196] Consistent with the present disclosure, a maximum number of the predicted signs for LFNST blocks and a maximum number of the predicted signs for non-LFNST blocks may be different to control the computation complexity of the sign prediction. For example, the maximum number of the predicted signs for LFNST blocks may be set to be 6 (or 4), whereas the maximum number of the predicted signs for non-LFNST blocks may have a value different from 6 (or 4). Additionally, different values of the maximum number of predicted signs may be applied for video blocks that apply the LFNST and not apply the LFNST. In some implementations, video encoder 20 may determine the maximum number of the predicted signs for the LFNST blocks based on corresponding complexity or performance preference of the encoder, and may signal the maximum number to video decoder 30. When the maximum number of the predicted signs for the LFNST blocks is signaled to video decoder 30, it may be signaled at various coding levels, e.g., a sequence parameter set (SPS), a picture parameter set (PPS), a picture or slice level, or a CTU or CU level. In some implementations, video encoder 20 may determine different values of the maximum number of predicted signs for video blocks that apply the LFNST and not apply the LFNST and signal the values of the maximum number

from video encoder 20 to video decoder 30.

[00197] Consistent with the present disclosure, given that the transform coefficients of both primary and secondary transforms are fixed, video encoder 20 or video decoder 30 may pre-calculate templates (e.g., samples of the templates) for different transform block sizes and different combinations of primary and secondary transform combinations. Video encoder 20 or video decoder 30 may store the templates (e.g., the samples of the templates) in an internal or external memory to avoid the complexity of producing the samples of the templates on-the-fly for optimized implementations. The samples of the templates may be stored with different fractional precisions to achieve different tradeoffs between a storage size and a sample precision. For example, video encoder 20 or video decoder 30 may scale floating samples of the templates with a fixed factor (e.g., 64, 128, or 256) and round the scaled samples to their nearest integers. The rounded samples may be stored in the memory. Then, when a template is used to reconstruct a candidate hypothesis, its corresponding samples may be firstly de-scaled to the original precision to ensure that generated samples in the candidate hypothesis are in a correct dynamic range.

[00198] FIG. 14 is a graphical representation illustrating exemplary hypothesis generation based on a linear combination of templates in accordance with some implementations of the present disclosure. In FIG 14, four patterned blocks 0-3 represent candidate transform coefficients whose signs are to be predicted. Factors C_0, C_1, C_2, and C_3 represent corresponding values of dequantized transform coefficients of the four candidate transform coefficients. Templates 0-3 may correspond to the four candidate transform coefficients 0-3, respectively. For example, template 0 which corresponds to the candidate transform coefficient 0 can be generated by applying an inverse secondary transform and an inverse primary transform to the transform block, where the candidate transform coefficient 0 is set to be one, and the remaining candidate transform coefficients in the transform block are set to be zero. Similarly, templates 1-3 can be generated respectively. A candidate hypothesis can be generated by adding templates 0-1 together with weights C_0-C_3, respectively.

[00199] FIG. 15A is a graphical representation illustrating an exemplary implementation of an existing sign prediction scheme in accordance with some examples. FIG. 15B is a graphical representation illustrating an exemplary implementation of a vector-based sign prediction scheme in accordance with some implementations of the present disclosure. An exemplary comparison between the existing sign prediction scheme and the vector-based sign prediction scheme disclosed herein is illustrated herein with reference to FIGs. 15A-15B.

[00200] In FIGs. 15A-15B, there are six non-zero transform coefficients in a transform block

which are selected as candidate transform coefficients for the sign prediction. The candidate transform coefficients are scanned from a coefficient matrix of the transform block using a raster scan order. Original signs and predicted signs of the candidate transform coefficients are also shown in FIGs. 15A-155B. For example, both an original sign and a predicted sign of a first candidate transform coefficient with a value of "-2" are "-" (represented as "1" in FIGs. 15A-155B). Both an original sign and a predicted sign of a second candidate transform coefficient with a value of "3" are "+" (represented as "0" in FIGs. 13A-13B). An original sign and a predicted sign of a third candidate transform coefficient with a value of "1" are "+" and "-," respectively (represented as "0" and "1" in FIGs. 13A-13B, respectively). The original sign of the third candidate transform coefficient is incorrectly predicted. As shown in FIGs. 15A-15B, except the third transform coefficient, original signs of all the other candidate transform coefficients are the same as their corresponding predicted signs (i.e., being correctly predicted).

[00201] Referring to FIG. 15A, 6 bins (i.e., 0, 0, 1, 0, 0 and 0) are generated in total with each bin corresponding to a candidate transform coefficient. The 6 bines can be generated by performing an XOR operation between the original signs and the predicted signs of the 6 candidate transform coefficients. The 6 bins can be used to indicate the corresponding correctness of the 6 predicted signs. For example, a first bin and a second bin each having a value of "0" indicate that the predicted signs for the first and second candidate transform coefficients are correct. A third bin having a value of "1" indicates that the predicted sign for the third transform coefficient is not correct. The 6 bins can be sent to CABAC for entropy coding.

[00202] Referring to FIG. 15B, the vector-based sign prediction scheme disclosed herein divides the 6 candidate transform coefficients into 3 groups, and each group includes 2 consecutive candidate transform coefficients. Because the signs of the candidate transform coefficients in Groups #0 and #2 can be correctly predicted, only two bins each having a value of "0" are generated for the two groups. For Group #1, since it includes the third candidate transform coefficient whose sign cannot be correctly predicted, a bin with a value of "1" (which is underlined in FIG. 15B) is generated and signaled in the bitstream to indicate that the group includes at least a candidate transform coefficient whose original sign is different from its predicted sign. Subsequently, two additional bins with values of "1" and "0" are generated for the third and fourth coefficients in Group #1 to indicate whether their signs can be correctly predicted. Correspondingly, there are 5 bins in total generated for CABAC when the vector-based sign prediction scheme disclosed herein is applied, which has less bits than the bins

generated by the existing sign prediction scheme shown in FIG. 15A. Thus, by applying the vector-based sign prediction scheme disclosed herein, the signaling overhead can be reduced, and the coding efficiency of the transform block can be improved.

[00203] Consistent with the present disclosure, although the raster scan order is used to obtain the candidate transform coefficients from the coefficient matrix of the transform block as shown in FIG. 15B, any other scan orders can also be used for selecting the candidate transform coefficients for sign prediction. For example, the candidate transform coefficients can be selected based on one or more selection criteria described above. The similar description is not repeated herein.

[00204] FIG. 16A is a graphical representation illustrating exemplary calculation of a left-diagonal cost function along a left diagonal direction in accordance with some implementations of the present disclosure. FIG. 16B is a graphical representation illustrating exemplary calculation of a right-diagonal cost function along a right diagonal direction in accordance with some implementations of the present disclosure. Compared with the above expression (8) for the calculation of costHV, the left-diagonal cost function costD1 or the right-diagonal cost function costD2 shown in the above expression (14) or (15) may need more neighboring pixels to support the calculation of the cost functions costD1, costD2 along the diagonal directions (illustrated as pixels marked within areas 1602, 1604, and 16606 in FIGs. 16A-16B). When these pixels in areas 1602, 1604, and 1606 are not available, then a nearest padding method can be adopted to fill these unavailable positions. For example, if $B_{-1,4}$ in area 1606 is unavailable, $B_{-1,3}$ which is an available pixel closest to $B_{-1,4}$ is used to fill the position of $B_{-1,4}$ (e.g., $B_{-1,4} = B_{-1,3}$). If $B_{-1,-1}$ (also denoted as $C_{-1,-1}$), $B_{-1,-2}$ ($C_{-1,-1}$), $B_{-2,-1}$ ($C_{-1,-1}$), and $B_{-2,-2}$ ($C_{-1,-1}$) in area 1602 are unavailable, several exemplary methods are disclosed herein to fill the unavailable positions.

[00205] In a first exemplary method, each unavailable position can be filled by weighting its nearest available pixels as shown below in expressions (16)-(19):

$$B_{-2,-2} = (B_{-2,0} + C_{0,-2}) * 0.5 \quad (16)$$

$$B_{-2,-1} = (2 * B_{-2,0} + C_{0,-1})/3 \quad (17)$$

$$B_{-1,-2} = (B_{-1,0} + 2 * C_{0,-2})/3 \quad (18)$$

$$B_{-1,-1} = (B_{-1,0} + C_{0,-1}) * 0.5 \quad (19).$$

[00206] In a second exemplary method, some of the unavailable positions can be filled with their nearest available pixels, respectively. For example, if $B_{-1,-2}$ in area 1602 is unavailable, it is filled with $C_{0,-2}$. If $B_{-2,-1}$ is unavailable, it is filled with $B_{-2,0}$. However, if $B_{-2,-2}$ and

$B_{-1,-1}$ are unavailable, they can be filled with an average of their two nearest neighboring pixels calculated according to the above expressions (16) and (17).

[00207] In a third exemplary method, only available neighboring reconstructed samples are used to calculate the cost function. In case when the cost calculation of one boundary sample along the top/left boundary of the current block involves any unavailable reconstructed samples, they will not be used in calculating the cost in the corresponding direction. For example, in Figure 16B, only the boundary samples $P_{0,0}$, $P_{0,1}$ and $P_{1,0}$ are used for calculating the value of the cost function in the top-right direction while $P_{0,2}$, $P_{0,3}$, $P_{2,0}$ and $P_{3,0}$ will not be used given that their cost computations refer to at least one reference samples which are not available.

[00208] Consistent with the present disclosure, although the left diagonal and the right diagonal (i.e., 135° and 45° shown in FIGs. 16A-16B) are used for illustration purpose in the calculation of the cost function shown in the above expression (14) or (15), it is contemplated any other measurement components (e.g., a continuity measurement along one or more arbitrary directions) can be incorporated into the calculation of the cost function for the sign prediction.

[00209] In a fourth implementation, a sample extrapolation method, which is based on the gradient analysis of the texture information among the neighboring samples of one current block, can be implemented to improve the precision of the cost function for the sign prediction. Instead of always using a fixed extrapolation direction (e.g., vertical extrapolation for the above neighboring samples and horizontal extrapolation for the horizontal neighboring samples), a texture analysis of the top and left neighboring reconstructed samples of the current block can be performed at both encoder and decoder, and the most dominate direction of the neighboring samples' gradients can be selected to extrapolate the boundary samples along the top and left boundaries of the current block.

[00210] For example, FIG. 17 is a flow chart of a method 1700 for capturing a dominate gradient direction within the neighboring reconstructed samples of a current block in accordance with some implementations of the present disclosure. Method 1700 may be implemented by a video processor associated with video encoder 20 or video decoder 30, and may include steps 1702-1712 as described below. Some of the steps may be optional to perform the disclosure provided herein. Further, some of the steps may be performed simultaneously, or in a different order than shown in FIG. 17.

[00211] In step 1702, reference samples are selected for a current block for the gradient derivation. In some implementations, reference samples form a template. For example, FIG.

18A is a graphical representation illustrating exemplary template samples and gradient filter windows in a gradient-based selection of a sample extrapolation direction for a cost function in accordance with some implementations of the present disclosure. As shown in FIG. 18A, a template of *N* rows and columns of the neighboring reconstructed samples above and left to the current block are used as the reference samples for the gradient derivation. In the example of FIG. 18A, the template size is equal to 3.

[00212] In step 1704, a histogram of gradient (HoG) is initialized. For example, the HoG may be generated with a number of entries, where each entry of the histogram of gradient corresponds to an accumulated magnitude of the gradients at a predefined angular direction. Each entry may be initialized as zero. For example, FIG. 18B a graphical representation illustrating an exemplary histogram of gradient (HoG) in a gradient-based selection of a sample extrapolation direction for a cost function in accordance with some implementations of the present disclosure. In practice, different predefined directions may be used in the disclosed gradient analysis scheme. In one implementation, the same directions as defined for 65 angular directions of the regular intra prediction in the VVC/ECM is used.

[00213] In step 1706, a gradient filter window is applied to the reference samples to calculate the respective gradients. As shown in FIG. 18A, one N×N gradient filter window is applied to each template sample located at the middle row/column of the template (i.e., the filter window is centered at the sample position) to calculate its corresponding horizontal gradient G_h and vertical gradient G_v, respectively.

[00214] In step 1708, the angle (Angle) and magnitude (Amp) of the gradient of each reference sample is calculated. For example, the gradient of a sample can be calculated according to expressions (20) and (21):

$$\text{Angle} = \tan^{-1}\left(\frac{G_h}{G_v}\right) \tag{3}$$

$$\text{Amp} = |G_h| + |G_v| \tag{21}.$$

[00215] In step 1710, the angle of the gradient can be converted into one of the predefined directions and the corresponding entry in the HoG is updated. For example, as shown in FIG. 18, the Amplitude in the HoG for each Angle is updated by adding the magnitude (Amp) of the sample gradient of that Angle. As a result, the Amplitude is an accumulated magnitude.

[00216] In step 1712, the entry of the HoG with the largest accumulated magnitude will be selected as the direction used to extrapolate the neighboring samples for the cost function of the current block. For example, as shown in FIG. 18, the largest entry is circled.

[00217] In the above method, the direction with the largest magnitude is selected as the direction for sample extrapolation. Such method may not necessarily be reliable when there is some noise (e.g., quantization errors and/or coding noise caused by other coding modules). To solve such problem, certain conditions may be applied before the dominant gradient direction is adopted as the extrapolation direction.

[00218] For example, in one implementation, the selected dominant gradient direction may be enabled for sample extrapolation of the cost function calculation only when there are enough template samples that belong to the selected gradient direction (e.g., the percentage of the samples belonging to the selected direction is large enough, such as exceeding a predetermined threshold). Otherwise (e.g., the number of template sample belonging to the selected direction is not large enough), the default extrapolation (e.g., vertical extrapolation for the above neighbor and horizontal extrapolation for the left neighbor) is still applied.

[00219] In another implementation, the selected dominant gradient direction may be enabled for sample extrapolation of the cost function calculation only when the gradient magnitude associated with the dominant gradient direction is large enough (e.g., the ratio of the magnitude of the selected gradient direction to the sum of the magnitudes of all the gradient directions is larger than another predetermined threshold). Otherwise (e.g., the gradient magnitude of the selected direction is not significant), the default extrapolation is still applied.

[00220] In yet another implementation, the above constrains are applied together. That is, the selected direction is only enabled for the sample extrapolation of the current block only when the number of template samples associated with selected direction are large enough and the gradient magnitude is significant enough. Otherwise, the default extrapolation is still applied.

[00221] In yet another implementation, when the selected dominant gradient direction is enabled, the direction that is orthogonal to the selected direction is used for the sample extrapolation in the calculation of the cost function when predicting the signs within one block. For instance, when the selected direction is 45-degree, then 135-degree will be used as the direction to extrapolate the samples along the top and left boundaries of the current block. Similarly, when the selected direction is 135-degree, then 45-degree will be used as the direction to extrapolate the samples along the top and left boundaries of the current block.

[00222] In the above disclosed implementations, the sign prediction is only applicable to predict the signs of the coefficients that are located in the top-left 4×4 sub-blocks of one transform block (e.g., as shown in FIG. 7). In general, such method is reasonable because the majority of the energy of common transform blocks is usually concentrated among a few of

transform coefficients at low frequencies. Correspondingly, the signs of the transform coefficients at the top-left corners are statistically easier than the transform coefficients at other places. However, this assumption may not be always true. For instance, for inter blocks with complicated motion field (e.g., subblock inter mode where one inter block is partitioned into multiple subblocks and each subblock can have its own MV), there could be a lot of edges generated along the boundary between different motions in their prediction signals. In such case, after applying the DCT/DST transform, there will be non-negligible high-frequency transform coefficients being generated and those coefficients could stand out of the top-left 4×4 corners. The signs of those high-frequency coefficients cannot be predicted according to the above disclosed sign prediction designs.

[00223] In some alternative implementations, the area of the transform coefficients that are inputted to a forward LFNST and therefore later used for sign prediction can be extended from the few top-left 4×4 sub-blocks (e.g., as shown in FIG. 7) to further improve the coding performance. For example, FIG. 19 is a graphical representation illustrating a sign prediction area for predicting signs of transform coefficients in accordance with some examples.

[00224] Specifically, the signs of the top-left A × B area of the current transform block can be selected for prediction, where A and B are calculated according to expression (22):

$$\begin{aligned} A &= \min(width, TH) \\ B &= \min(height, TH) \end{aligned} \tag{22}$$

where *width* and *height* are the width and height of the transform block and *TH* is the maximum area size for sign prediction (also referred to as the area size threshold). Manipulating the value of *TH* could provide different performance and complexity trade-off for the sign prediction technique. Enlarging the area of the sign prediction can increase the number of the transform coefficients whose signs are going to be predicted, although it may come at the expense of increased computational complexity due to the increasing number of sign combinations that need to be tested at both encoder and decoder.

[00225] In some implementations, different methods can be used for the determination of the value of *TH*. In one implementation, a fixed value (e.g., 8, 16, 32 and so forth) may be used for *TH* in all sequences and coding scenarios. Given the area size threshold *TH* is fixed, encoder and decoder will use the same value (i.e., without being signaled) to determine the corresponding area when locating the transform coefficients for the sign prediction. In another implementation, the encoder may be given the flexibility to decide the optimal area size threshold according to the specific characteristics of the video sequence as well as its preferred

performance/complexity tradeoff and signal the corresponding value from encoder to decoder.

[00226] In one exemplary method, the area size threshold may be selected from a set of fixed number such that only a fixed number of bits need to be signaled in the bitstream to the decoder to indicate which size is selected. For example, assuming there are four allowed area size threshold values {4, 8, 16, 32}, only two bits need to be signaled to indicate the particular value selected from the group. In another exemplary method, the area size threshold may be adaptively determined for the transform block on the encoder side. Therefore, the value of the area size threshold may be an arbitrary number. In such case, due to its unknown values, fixed-length codeword cannot be used; instead, some variable-length codeword (e.g., exponential Golomb, unary code and so forth) may be applied for indicating the value of the determined area size threshold in bitstream.

[00227] Additionally, when the value of *TH* is sent in bitstream, it can be signaled in different levels, e.g., in video parameter set (VPS), sequence parameter set (SPS), picture parameter set (PPS), slice header and coding block level. When it is signaled at sequence level (e.g., in the VPS or SPS), it means one same area size threshold will be shared for all the transform blocks in the video sequence. When it is signaled at picture/slice level (i.e., in the PPS or slice header), it allows the area size threshold to vary from picture to picture or slice to slice, respectively. Similarly, when the value is signaled at coding blocks, it provides the maximum adaption capability of the sign prediction regions which however consume more coding bits given the value has to be signaled for each individual coding block separately.

[00228] On the decoder side, the decoder may determine the sign prediction area based on the size of the transform block and the area size threshold *TH*. When *TH* is a fixed value, it may not be signaled in the bitstream, but rather preprogramed in the decoder. The decoder may use this preprogramed fixed value for all sequences and scenarios. Otherwise, when *TH* is not a fixed value, but a value determined by the encoder, it will be signaled to the decoder in the bitstream. Accordingly, the decoder determines the sign prediction area based on the size of the transform block and the area size threshold *TH* signaled.

[00229] In some implementations, the decoder may first determine in which level the area size threshold is signaled. As described above, *TH* can be signaled in different levels, such as VPS, SPS, PPS, slice header and coding block level, etc. Based on the level it is signaled, decoder can determine the extent that the signaled *TH* should be applied. For example, when *TH* is signaled in a VPS or SPS level, the decoder will apply it to all the transform blocks in the video sequence. When *TH* is signaled in a PPS or slice header level, the decoder will apply it to all the transform blocks in the current picture/slice, and will read a different *TH* for each

different picture/slice. When *TH* is signaled at coding block level, the decoder will apply a different *TH* read from the bitstream to each different coding block.

[00230] When the area size threshold is a value adaptively determined for the transform block, the decoder may determine the value of *TH* based on the codeword signaled in the bitstream. As described above, *TH* can be signaled using a fixed-length codeword when it is selected from a group of predetermined values, or signaled using a variable-length codeword when it is an arbitrary number adaptively determined for the transform block.

[00231] In some implementations, after determining what value of *TH* to be applied to the current transform block, the decoder may determine the sign prediction area based on the size of the transform block and the area size threshold *TH*, e.g., according to expression (22). For example, as shown in FIG. 19, a width of the sign prediction area A is determined to be the smaller of a *width* of the transform block and the area size threshold *TH*, and a height of the sign prediction area B is determined to be the smaller of a *height* of the transform block and the area size threshold *TH*.

[00232] As described above, to improve the efficiency of the sign prediction, the sign prediction area where the sign prediction of the transform coefficients is applied could be signaled from encoder to decoder. To implement this extended sign prediction area on the decoder side, the decoder has to be capable of handling all the possible extended region size that may be selected by the encoder, this could possibly introduce ambiguity when the decoder parses the signs for the blocks in various sizes. To facilitate the implementation of extended sign prediction area, region-based methods may be used to signal the signs of one transform block in a certain order.

[00233] Assume that there are *N* possible area size threshold values, TH_0, TH_1, ..., TH_{N-1}, that may be signaled from encoder to decoder and be applied by the decoder to locate the transform coefficients for the sign prediction. Consistent with descriptions above, these area size threshold values can be predetermined values or values adaptively determined by the encoder. TH_{max} is the maximum area size threshold among these possible threshold values, i.e., $TH_{max} = max\{TH_0, TH_1, ..., TH_{N-1}\}$. TH_{cur} is the selected threshold value used for determining the sign prediction area and therefore is the value signaled to the decoder, where $TH_{cur} \in \{TH_0, TH_1, ..., TH_{N-1}\}$. For simplification, the descriptions below assume that both the width and height of the current transform block are larger than TH_{max}. That is, applying Equation (22), $A=B=TH_{max}$.

[00234] In some implementations, to signal the signs of one transform block in a certain

order, the transform block is first divided into multiple regions, and then the signs of the transform coefficients associated with the multiple regions are signaled one region after another.

[00235] For example, FIG. 20 is a graphical representation illustrating a division of a transform block for implementing an exemplary order for signaling the signs of transform coefficients from an encoder to a decoder in accordance with some examples. As shown in FIG. 20, the whole region of the transform block can be divided into three different regions:

(1) the first region (region #1) contains the transform coefficients that are within the top-left region enclosed by the signed sign prediction region size TH_{cur}, i.e., $0 <= x < TH_{cur}$ *and* $0 <= y < TH_{cur}$;

(2) the second region (region #2) contains the transform coefficients that are in-between the regions enclosed by the signed sign prediction region and the maximum sign prediction region size, i.e., $\{TH_{cur} <= x < TH_{max}$ *and* $0 <= y < TH_{cur}\}$, $\{0 <= x < TH_{cur}$ *and* $TH_{cur} <= y < TH_{max}\}$, $\{TH_{cur} <= x < TH_{max}$ *and* $TH_{cur} <= y < TH_{max}\}$; and

(3) the third region (region #3) contains all the other transform coefficients that are in the transform block, i.e., $x > TH_{max}$ or $y > TH_{max}$,

where x and y are the horizontal and vertical coordinates of transform coefficients in the transform block.

[00236] FIG. 20 illustrates the resulting regions after the above partition/division scheme is applied to one transform block. Region #1 contains the transform coefficients whose signs can be either predicted or not predicted while the signs of the transform coefficients in region #2 and region #3 are purely not predictable, i.e., they are directly signaled in bitstream. In some implementations, under the partition structure of FIG. 20, the signs of the transform coefficients in region #3 may be signaled first, followed by the signs of the transform coefficients in region #1, and finally, the signs of the transform coefficients in region #2.

[00237] In some implementations, to code the signs of the transform coefficients from different regions, different scanning orders may be applied by the encoder. For example, FIG. 21 is a graphical representation illustrating various scan orders for scanning the signs of transform coefficients in accordance with some examples. Part (a) of FIG. 21 shows a diagonal scan, part (b) shows a horizontal scan and Part (c) shows a vertical scan.

[00238] In some implementations, different scan orders may be applied to the divided regions of the transform block. For example, FIG. 22 is a graphical representation illustrating implementation of different scan orders for the transform block divided according to FIG. 20, in accordance with some examples. The signs of the transform coefficients in region #1 and #2 may be scanned by horizontal scan and the signs of the transform coefficients in region #3

may be by diagonal scan, as illustrated in FIG. 22.

[00239] As another example, FIG. 23 is a graphical representation illustrating a division of a transform block for implementing another exemplary order for signaling the signs of transform coefficients from an encoder to a decoder in accordance with some examples. In this example, the transform block is only partitioned into two regions:

(1) the first region (region #1) contains the transform coefficients that are within the top-left region enclosed by the signed sign prediction region size TH_{cur}, i.e., $0 <= x < TH_{cur}$ and $0 <= y < TH_{cur}$; and

(2) the second region (region #2) contains all the other transform coefficients that are in the transform block, i.e., $x > TH_{cur}$ or $y > TH_{cur}$,

where x and y are the horizontal and vertical coordinates of transform coefficients in the transform block.

[00240] FIG. 22 illustrates the resulting regions after the above partition scheme is applied to one transform block. Region #1 contains the transform coefficients whose signs can be either predicted or not predicted while the signs of the transform coefficients in region #2 are purely not predictable, i.e., they are directly signaled in bitstream. Under the partition structure of FIG. 22, in some implementations, the signs of the transform coefficients in region #2 are signaled first, followed by the signs of the transform coefficients in region #1.

[00241] In some embodiments, different scan orders may be applied by the encoder to scan the signs of transform coefficients belonging to the two regions. For example, FIG. 24 is a graphical representation illustrating implementation of different scan orders for the transform block divided according to FIG. 22, in accordance with some examples. In one example, the encoder may scan the signs of the transform coefficients in region #1 by horizontal scan and scan the signs of the transform coefficients in region #2 by diagonal scan, as illustrated in FIG. 24.

[00242] Although in the examples of FIG. 22 and FIG. 24, only horizontal and diagonal scans are applied to scan the signs of transform coefficients, persons skilled in modern video coding technologies will readily recognize that other combinations of scan orders may be applicable to the disclosed region-based signaling methods to signal the signs of one transform block.

[00243] Because on the encoder side, the signals are encoded in a bitstream in a region-based order (e.g., region #3, region #1, and then region #2, for the partition shown in FIG. 20), on the decoder side, the encoded data are parsed according to the same order. The decoder may first determine the regions using the signaled TH_{cur} (and optionally also TH_{max}) and read the

data for those respective regions, before conducting sign prediction and other video decoding steps. Using the partition of FIG. 20 as an example, the decoder may parse the bitstream to obtain signs and encoded transform coefficients for region #3 first, followed by region #1, and then region #2. The decoder may then perform sign prediction for the transform coefficients in region #1 (the sign prediction area for the transform block).

[00244] It is contemplated that the extended sign prediction area scheme does not interfere with any of the sign prediction techniques disclosed above, which can be easily adapted and combined with the extended sign prediction area to offer an improved coding performance. As just one specific example, the extended sign prediction area can be combined with the sign prediction reordering and the LFNST sign prediction.

[00245] In the forgoing descriptions, one context model is described as an example to code the correctness of the predicted signs in the sign signaling bits. Specifically, the difference between each predicted sign can be coded and a corresponding true sign with "0" (or "1") to indicate that the estimated sign is the same (or not the same) as the true sign. Consistent with some implementations of the disclosure, other context models can be used to entropy-code the correctness of the sign prediction, to further improve the coding efficiency.

[00246] In a first exemplary method, the context of the level value (i.e., magnitude) of the transform coefficients can be used for entropy-coding the correctness of the sign prediction. This is because one larger magnitude tends to have a higher probability of making a correct prediction. For instance, when such method is applied, the range of the possible magnitude values may be divided into several segments and different contexts may be assigned to the sign prediction entropy-coding of the coefficients in different segments.

[00247] In some implementations, the range of the magnitudes of the transform coefficients into a plurality of magnitude segments, and a context may be assigned to each magnitude segment for entropy-encoding the sign signaling bits for the transform coefficients belonging to the magnitude segment. Every two consecutive magnitude segments, among the plurality of magnitude segments, may be separated by a threshold, so that the magnitude segments do not overlap.

[00248] In one specific example, a threshold *thres* may be used to divide the range of the transform coefficient magnitude into two segments, i.e., (1, thres) and [thres, $+\infty$]. With such division, if one transform coefficient belongs to the first segment, one first context is assigned to code the sign of the transform coefficient. Otherwise, if the transform coefficient falls into the second segment, then one second context is assigned to code the sign of the transform

coefficient.

[00249] In various implementations, the value of threshold (or values of the thresholds) can be determined and specified in different ways. In one exemplary method, the value may be fixed (e.g., 2, 3, …, and so forth) and preprogramed in both encoder and decoder. Accordingly, the fixed value is used at both the encoder and decoder sides without the need to signal it. In another exemplary method, the encoder may be given the freedom of selecting the optimal threshold and signal the corresponding value to decoder.

[00250] As described above, the magnitude of each transform coefficient is used for determining the context for coding the sign value of the transform coefficient on the encoder side. Accordingly, on the decoder side, the magnitude of each transform coefficient is accordingly used for determining the context in order to decode the sign value of the transform coefficient. Given that the transform coefficients can only be obtained after dequantization is done on the decoder side, such scheme may significantly increase the complexity of parsing bitstreams due to the fact that dequantization needs to be conducted at the parsing stage. To solve such complexity issue, in one implementation, the decoder may directly use the level $\underline{k}$ of the transform coefficient (i.e., the quantized value of the transform coefficient directly parsed from the bitstream without dequantization being applied) for determining the context of coding the sign of one transform coefficient.

[00251] In another implementation, instead of directly using the level k that is parsed from the bitstream, the decoder may use the quantization index (i.e., *quantIdx*) to determine the context of encoding the sign of one transform coefficient. As described, in the VVC/ECM, two scalar quantizers may be applied for the quantization/dequantization. The actual quantizer used for one transform coefficient is determined by the parities of the transform coefficients that precede the current transform coefficient. Therefore, the quantization indices for the transform coefficients may be used to determine the contexts for encoding the signs. As described earlier, the quantization index can be calculated from the value of the level k based on the transition state (i.e., *state*) of the current transform coefficient, e.g., according to Expression (6).

[00252] In various implementations, all the above magnitude/level-based context modeling methods can be combined with the disclosed transform coefficient selection schemes as described in connection with FIG. 13. For example, such combination could be implemented in two different ways. In one method, the transform coefficients may be reordered and the magnitudes/levels of the reordered transform coefficients (i.e., the ones after the transform coefficients are reordered) to determine the corresponding contexts that are applied to code the signs of transform coefficients in one transform block. In another method, the transform

coefficients are not reordered and the magnitudes/levels of the transform coefficients are used without reordering (i.e., the ones before the transform coefficient are reordered) to determine the contexts applied to coding the signs of transform coefficients in the transform block.

[00253] FIGs. 25A and 25B show two examples of context modeling for transform coefficient sign prediction, implementing the two schemes described above, respectively. As shown in FIG. 25A, when the first scheme is applied, the transform coefficients are reordered according to their magnitudes/levels, e.g., in the order of 6, 3, 2, 2, 1, 1, 1, and 1. The magnitudes/levels of the reordered transform coefficients are used for selecting the contexts of the sign signaling bits that are included in the bitstream. The benefit of this first scheme is that the corresponding magnitudes/levels used for context modeling are matched with the transform coefficients whose signs are being coded. When this first scheme is applied, the encoder/decoder needs to reorder the transform coefficients before the signs are being coded/decoded, which may increase the complexity of bitstream parsing and therefore reducing the parsing throughput. FIG. 25B shows the implementation of a second scheme that may avoid this parsing problem. In this second scheme, the second method the magnitudes/levels of the transform coefficients without being reordered are used for the context determination for encoding the signs. Therefore, the reordering does not need to be applied at the parsing stage. That said, the magnitudes/levels used for context modeling may be misaligned with the signs.

[00254] In a second method, the context of the scan positions of the transform coefficients may be used for entropy-coding the sign prediction. For instance, one single context may be applied to the coding of the transform coefficients located at the first L positions of one transform block while another context may be applied to coding the signs of the coefficients at other positions. In one specific example, the value of L may be set to be equal to 1. Accordingly, one context is assigned to code the sign of the first transform coefficient (i.e., the DC coefficient) of one transform coefficient and another context is assigned to code the signs of the all the other transform coefficients (i.e., the AC coefficients). Alternatively, all the sign prediction positions in one transform block can be classified into different groups according to their respective importance (or probability of correct predictions) and different contexts may be assigned to each group, separately.

[00255] In a third method, coding related information such as the context of the coding mode, block size and component channel information may be used for the sign prediction coding. For instance, different contexts can be used for inter and intra modes. In another example, different context models may be used for different blocks. In yet another example, different contexts may be used for luma and chroma components.

[00256] In a fourth method, the transform types for determining the transform coefficients may be used as the context for the sign prediction. As described above, there are multiple transform cores that are applied to compact the energy of coefficients in one transform block, such as MTS and LFNST. Given that the transform coefficients resulting from different transform cores may present different statistic characteristics, different contexts may be used for entropy encoding the signs of the transform coefficients of different transforms.

[00257] Although the above context modeling schemes are described separately, it is contemplated that any of them may be used together. In practice, each method can be freely combined with others to achieve different context design.

[00258] FIG. 26 is a flow chart of an exemplary method 2600 for sign prediction in block-based video coding in accordance with some implementations of the present disclosure. Method 2600 may be implemented by a video processor associated with video encoder 20 or video decoder 30, and may include steps 2602-2608 as described below. Some of the steps may be optional to perform the disclosure provided herein. Further, some of the steps may be performed simultaneously, or in a different order than shown in FIG. 26.

[00259] In step 2602, the video processor may apply a primary transform and a secondary transform to a transform block of a video frame from a video to generate transform coefficients for the transform block.

[00260] In step 2604, the video processor may select a set of candidate transform coefficients from the transform coefficients for the sign prediction.

[00261] In step 2606, the video processor may apply a template-based hypothesis generation scheme to select a hypothesis from a plurality of candidate hypotheses for the set of candidate transform coefficients.

[00262] In step 2608, the video processor may determine a combination of sign candidates associated with the selected hypothesis to be a set of predicted signs for the set of candidate transform coefficients.

[00263] Consistent with the present disclosure, methods 2600 and FIG. 26 may be performed on a video encoder side or a video decoder side. When method 2600 is performed on the video encoder side, it may be considered as an encoding method for transform coefficient sign prediction on the video encoder side. Sign signaling bits, indicating the correctness of the sign prediction, are then generated and encoded to be included in a bitstream sent to the decoder. When method 2600 is performed on the video decoder side, it may be considered as a decoding method for transform coefficient sign prediction on the video decoder side. The sign signaling bits are signaled to the decoder in the bitstream that also contains the encoded data of the

quantized levels of the transform coefficients. The decoder may use the sign signaling bits to correct its sign prediction in order to estimate the original signs of the received transform coefficients. An exemplary encoding method and an exemplary decoding method for transform coefficient sign prediction on the video decoder side are respectively provided below with reference to FIG. 27 and FIG. 28.

[00264] FIG. 27 is a flow chart of an exemplary video encoding method 2700 for transform coefficient sign prediction performed by a video encoder, in accordance with some implementations of the present disclosure. Method 2700 may be implemented by a video processor associated with video encoder 20, and may include steps 2702-2716 as described below. Some of the steps may be optional to perform the disclosure provided herein. Further, some of the steps may be performed simultaneously, or in a different order than shown in FIG. 27.

[00265] In step 2702, the video processor may apply a primary transform and a secondary transform to a transform block of a video frame from a video to generate transform coefficients for the transform block. In some implementations, the secondary transform may be a forward LFNST.

[00266] In step 2704, the video processor may determine a sign prediction area for performing transform coefficient sign prediction based on an area size threshold. In some implementations, the video processor may select the area size threshold and signal the area size threshold. In some implementations, the video processor may determine the sign prediction area in a top-left corner of the transform block according to Equation (22).

[00267] In step 2706, the video processor may divide/partition the transform block into multiple regions including the sign prediction area and at least one other region. In some implementations, the transform block may be divided into three regions, including a first region of the sign prediction area, a second region encompassing an area of a rectangle with a width and a height equal to a maximum area size threshold and outside the sign prediction area, and a third region covering the remaining area of the transform block outside the first and second regions. In some implementations, the transform block may be divided into two regions, including a first region of the sign prediction area, and a second region covering the remaining area of the transform block outside the first region.

[00268] In step 2708, the video processor may select a set of candidate transform coefficients from the transform coefficients in the sign prediction area for the sign prediction.

[00269] In step 2710, the video processor may predict signs for the set of candidate transform coefficients. In some implementations, the video processor may determine a

plurality of combinations of sign candidates for the set of candidate transform coefficients based on a total number of candidate transform coefficients in the set of candidate transform coefficients. The video processor may then apply a template-based hypothesis generation scheme to generate a plurality of candidate hypotheses for the plurality of combinations of sign candidates, respectively. The video processor may select a hypothesis associated with a minimal cost from the plurality of candidate hypotheses based on a cost function. The video processor may determine a combination of sign candidates associated with the selected hypothesis to be a set of predicted signs for the set of candidate transform coefficients.

[00270] In step 2712, the video processor may generate a sequence of sign signaling bits for the set of candidate transform coefficients according to one or more contexts. Various context models and their combinations may be used to code the correctness of the predicted signs. In some implementations, the sequence of sign signaling bits may be generated by applying a vector-based sign prediction scheme.

[00271] In step 2714, the video processor may code signs of the transform coefficients of the multiple regions using respective scan orders. In some implementations, different scan orders are applied to the multiple regions. For example, when the transform block is divided into three regions in step 2706, the first region and the second region may be scanned in a horizontal scan order and the third region may be scanned in a diagonal scan order. As another example, when the transform block is divided into two regions in step 2706, the first region may be scanned in a horizontal scan order and the second region may be scanned in a diagonal scan order.

[00272] In step 2716, the video processor may generate a bitstream to signal the signs of the transform coefficients associated with the multiple regions according to a region-based order and the sequence of sign signaling bits. In some implementations, when the transform block is divided into three regions in step 2706, the signs of transform coefficients of the third region are signaled in the bitstream first, followed by those of the first region, and finally those of the second region. In some implementations, when the transform block is divided into two regions in step 2706, the signs of transform coefficients of the second region are signaled in the bitstream first, followed by those of the first region.

[00273] The bitstream generated in step 2716 may be sent to a decoder for reconstructing the transform block of the video frame. If the area size threshold is selected by the encoder (in step 2704), it may be included in the bitstream as well to be signaled to the decoder.

[00274] FIG. 28 is a flow chart of an exemplary video decoding method 2800 for transform coefficient sign prediction performed by a video decoder, in accordance with some

implementations of the present disclosure. Method 2800 may be implemented by a video processor associated with video decoder 30, and may include steps 2802-2818 as described below. Some of the steps may be optional to perform the disclosure provided herein. Further, some of the steps may be performed simultaneously, or in a different order than shown in FIG. 28.

[00275] In step 2802, the video processor may receive a bitstream that includes the coded data of the transform coefficients associated with multiple regions of a transform block and the sequence of sign signaling bits. In some implementations, the bitstream may be generated using a video encoding method such as method 2700 of FIG. 27.

[00276] In step 2804, the video processor may determine a sign prediction area for performing transform coefficient sign prediction based on an area size threshold. In some implementations, the area size threshold may be selected by an encoder and signaled to the decoder in the bitstream. In some implementations, the video processor may determine the sign prediction area in a top-left corner of the transform block according to Equation (22).

[00277] In step 2806, the video processor may divide/partition the transform block into multiple regions including the sign prediction area and at least one other region. In some implementations, the transform block may be divided into three regions, including a first region of the sign prediction area, a second region encompassing an area of a rectangle with a width and a height equal to a maximum area size threshold and outside the sign prediction area, and a third region covering the remaining area of the transform block outside the first and second regions. In some implementations, the transform block may be divided into two regions, including a first region of the sign prediction area, and a second region covering the remaining area of the transform block outside the first region.

[00278] In step 2808, the video processor may parse the bitstream to obtain coded data of the transform coefficients associated with the multiple regions of the transform block according to a region-based order and scan orders applied to the multiple regions. In some implementations, when the transform block is divided into three regions in step 2806, the signs of transform coefficients of the third region are read from the bitstream first, followed by those of the first region, and finally those of the second region. In some implementations, when the transform block is divided into two regions in step 2806, the signs of transform coefficients of the second region are read from the bitstream first, followed by those of the first region.

[00279] Within each region, the data are parsed according to the respective scan orders applied by the encoder to code the signs of the transform coefficients. For example, when the transform block is divided into three regions in step 2806, the first region and the second region

may be scanned in a horizontal scan order and the third region may be scanned in a diagonal scan order. As another example, when the transform block is divided into two regions in step 2806, the first region may be scanned in a horizontal scan order and the second region may be scanned in a diagonal scan order.

[00280] In step 2810, the video processor may dequantize the transform coefficients of the transform block. The video processor may generate the dequantized transform coefficients from the quantized transform coefficients through inverse quantization unit 86 of FIG. 3. The dequantized transform coefficients of the transform block in video decoder 30 may be equivalent to transform coefficients of the transform block in the video encoder 20.

[00281] In step 2812, the video processor may select a set of candidate transform coefficients from the dequantized transform coefficients associated with the sign prediction area for the sign prediction.

[00282] In step 2814, the video processor may predict signs for the set of candidate transform coefficients. In some implementations, the video processor may determine a plurality of combinations of sign candidates for the set of candidate transform coefficients based on a total number of candidate transform coefficients in the set of candidate transform coefficients. The video processor may then apply a template-based hypothesis generation scheme to generate a plurality of candidate hypotheses for the plurality of combinations of sign candidates, respectively. The video processor may select a hypothesis associated with a minimal cost from the plurality of candidate hypotheses based on a cost function. The video processor may determine a combination of sign candidates associated with the selected hypothesis to be a set of predicted signs for the set of candidate transform coefficients.

[00283] In step 2816, the video processor may estimate original signs for the set of candidate transform coefficients based on the set of predicted signs and the sequence of decoded sign signaling bits in the bitstream received in step 2802.

[00284] For example, with reference to FIG. 15B, the set of predicted signs can include Group #0 with values (1, 0), Group #2 with values (1, 0), and Group #3 with values (1, 0), where 1 indicates a negative sign and 0 indicates a positive sign. The sequence of sign signaling bits can include a bit "0" for Group #0, bits "1, 1, 0" for Group #2, and a bit "0" for Group #3. Since the bit for Group #0 has a value of "0," indicating the predicted signs for this group with values of (1, 0) are correct (e.g., the predicted signs are the same as the original signs), the estimated original signs for Group #0 are determined to be (1, 0). Since a first bit in the bits for Group #1 has a value of "1," indicating the predicted signs for this group (1, 0) are incorrect (e.g., the predicted signs are not the same as the original signs), then the estimated original

signs for Group #1 (1, 0) are determined to be an XOR result of the predicted signs for this group (1, 0) and the second and third bits "1, 0" for Group #1 (e.g., the estimated original signs = XOR ((1, 0), (1, 0)) = (0, 0)). Since the bit for Group #2 has a value of "0," indicating the predicted signs for this group with values of (1, 0) are correct (e.g., the predicted signs are the same as the original signs), the estimated original signs for Group #2 are determined to be (1, 0). Then, the estimated original signs for the set of candidate transform coefficients are formed by concatenating the estimated original signs of Groups #0, #1, and #2, respectively, which includes (1, 0, 0, 0, 1, 0). As described above, the video encoder may use different context models to code the correctness of the predicted signs, and correction of predicted signs are adapted according to those context models.

[00285] In step 2818, the video processor may update the dequantized transform coefficients based on the estimated original signs for the set of candidate transform coefficients. For example, the video processor may use the estimated original signs as true signs for dequantized transform coefficients in the transform block that correspond to the set of candidate transform coefficients.

[00286] In some implementations, after the dequantized transform coefficients are updated, the video processor may further apply an inverse primary transform and an inverse secondary transform to the dequantized transform coefficients to generate residual samples in a residual block corresponding to the transform block. The inverse secondary transform corresponds to a secondary transform that includes an LFNST. The inverse primary transform corresponds to a primary transform that includes DCT-II, DCT-V, DCT-VIII, DST-I, DST-IV, DST-VII, or and identity transforms.

[00287] In some implementations, the sequence of sign signaling bits for the set of candidate transform coefficients are generated by the video encoder by: applying, by the video encoder, a vector-based sign prediction scheme to another set of predicted signs for another set of candidate transform coefficients selected on a video encoder side; and generating a bitstream including the sequence of sign signaling bits. The other set of candidate transform coefficients are transform coefficients on the video encoder side that correspond to the set of candidate transform coefficients on the video decoder side.

[00288] In some implementations, applying the vector-based sign prediction scheme to the other set of predicted signs for the other set of candidate transform coefficients further includes: dividing the other set of candidate transform coefficients into a plurality of groups of candidate transform coefficients; and for each group of candidate transform coefficients, generating one or more sign signaling bits for the group of candidate transform coefficients based on whether

original signs of the group of candidate transform coefficients are identical to predicted signs of the group of candidate transform coefficients.

[00289] In some implementations, generating the one or more sign signaling bits for the group of candidate transform coefficients includes: responsive to the original signs of the group of candidate transform coefficients being identical to the predicted signs of the group of candidate transform coefficients, generating a bin with a value of zero; and adding the bin as a sign signaling bit to a bitstream. In some implementations, generating the one or more sign signaling bits for the group of candidate transform coefficients includes: responsive to the original signs of the group of candidate transform coefficients being not identical to the predicted signs of the group of candidate transform coefficients, generating a bin with a value of one; generating a set of additional bins for informing corresponding correctness of the predicted signs of the group of candidate transform coefficients; and adding the bin and the set of additional bins as sign signaling bits to the bitstream.

[00290] FIG. 29 shows a computing environment 2910 coupled with a user interface 2950, according to some implementations of the present disclosure. The computing environment 2910 can be part of a data processing server. For example, the video processor in video encoder 20 or video decoder 30, disclosed above, can be implemented using computing environment 2910. The computing environment 2910 includes a processor 2920, a memory 2930, and an Input/Output (I/O) interface 2940.

[00291] The processor 2920 typically controls overall operations of the computing environment 2910, such as the operations associated with display, data acquisition, data communications, and image processing. The processor 2920 may include one or more processors to execute instructions to perform all or some of the steps in the above-described methods. Moreover, the processor 2920 may include one or more modules that facilitate the interaction between the processor 2920 and other components. The processor 2920 may be a Central Processing Unit (CPU), a microprocessor, a single chip machine, a Graphical Processing Unit (GPU), or the like.

[00292] The memory 2930 is configured to store various types of data to support the operation of the computing environment 2910. The memory 2930 may include predetermined software 2932. Examples of such data includes instructions for any applications or methods operated on the computing environment 2910, video datasets, image data, etc. The memory 2930 may be implemented by using any type of volatile or non-volatile memory devices, or a combination thereof, such as a Static Random Access Memory (SRAM), an Electrically Erasable Programmable Read-Only Memory (EEPROM), an Erasable Programmable Read-

Only Memory (EPROM), a Programmable Read-Only Memory (PROM), a Read-Only Memory (ROM), a magnetic memory, a flash memory, a magnetic or optical disk.

[00293] The I/O interface 2940 provides an interface between the processor 2920 and peripheral interface modules, such as a keyboard, a click wheel, buttons, or the like. The buttons may include but are not limited to, a home button, a start scan button, and a stop scan button. The I/O interface 2940 can be coupled with an encoder and decoder.

[00294] In some implementations, there is also provided a non-transitory computer-readable storage medium comprising a plurality of programs, for example, in the memory 2930, executable by the processor 2920 in the computing environment 2910, for performing the above-described methods. In one example, the plurality of programs may be executed by the processor 2920 in the computing environment 2910 to receive (for example, from the video encoder 20 in FIG. 2) a bitstream or data stream including encoded video information (for example, video blocks representing encoded video frames, and/or associated one or more syntax elements, etc.), and may also be executed by the processor 2920 in the computing environment 2910 to perform the decoding method described above according to the received bitstream or data stream. In another example, the plurality of programs may be executed by the processor 2920 in the computing environment 2910 to perform the encoding method described above to encode video information (for example, video blocks representing video frames, and/or associated one or more syntax elements, etc.) into a bitstream or data stream, and may also be executed by the processor 2920 in the computing environment 2910 to transmit the bitstream or data stream (for example, to the video decoder 30 in FIG. 3). Alternatively, the non-transitory computer-readable storage medium may have stored therein a bitstream or a data stream comprising encoded video information (for example, video blocks representing encoded video frames, and/or associated one or more syntax elements, etc.) generated by an encoder (for example, video encoder 20 in FIG. 2) using, for example, the encoding method described above for use by a decoder (for example, video decoder 30 in FIG. 3) in decoding video data. The non-transitory computer-readable storage medium may be, for example, a ROM, a Random Access Memory (RAM), a CD-ROM, a magnetic tape, a floppy disc, an optical data storage device or the like.

[00295] In some implementations, there is also provided a computing device comprising one or more processors (for example, the processor 2920); and the non-transitory computer-readable storage medium or the memory 2930 having stored therein a plurality of programs executable by the one or more processors, wherein the one or more processors, upon execution of the plurality of programs, are configured to perform the above-described methods.

[00296] In some implementations, there is also provided a computer program product comprising a plurality of programs, for example, in the memory 2930, executable by the processor 2920 in the computing environment 2910, for performing the above-described methods. For example, the computer program product may include the non-transitory computer-readable storage medium.

[00297] In some implementations, the computing environment 2910 may be implemented with one or more ASICs, DSPs, Digital Signal Processing Devices (DSPDs), Programmable Logic Devices (PLDs), FPGAs, GPUs, controllers, micro-controllers, microprocessors, or other electronic components, for performing the above methods.

[00298] The description of the present disclosure has been presented for purposes of illustration and is not intended to be exhaustive or limited to the present disclosure. Many modifications, variations, and alternative implementations will be apparent to those of ordinary skill in the art having the benefit of the teachings presented in the foregoing descriptions and the associated drawings.

[00299] Unless specifically stated otherwise, an order of steps of the method according to the present disclosure is only intended to be illustrative, and the steps of the method according to the present disclosure are not limited to the order specifically described above, but may be changed according to practical conditions. In addition, at least one of the steps of the method according to the present disclosure may be adjusted, combined or deleted according to practical requirements.

[00300] The examples were chosen and described in order to explain the principles of the disclosure and to enable others skilled in the art to understand the disclosure for various implementations and to best utilize the underlying principles and various implementations with various modifications as are suited to the particular use contemplated. Therefore, it is to be understood that the scope of the disclosure is not to be limited to the specific examples of the implementations disclosed and that modifications and other implementations are intended to be included within the scope of the present disclosure.

CLAIMS

What is claimed is:

1. A video decoding method for sign prediction of transform coefficients, comprising:

receiving a bitstream comprising encoded data of transform coefficients of a transform block of a video frame from a video;

determining, by a decoder, multiple regions in the transform block, wherein the multiple regions include a sign prediction area for performing the sign prediction and at least one other region;

parsing the bitstream, by the decoder, to obtain signs of transform coefficients associated with the multiple regions of the transform block according to a region-based order used to signal the signs of the transform coefficients;

estimating, by the decoder, original signs for a set of candidate transform coefficients associated with the sign prediction area of the transform block; and

updating the transform coefficients associated with the sign prediction area based on the estimated original signs.

2. The video decoding method of claim 1, wherein the multiple regions in the transform block are determined based on an area size threshold.

3. The video decoding method of claim 2, wherein the sign prediction area is determined to be a first region in a top-left corner of the transform block with a width and a height equal to the area size threshold.

4. The video decoding method of claim 3, wherein the at least one other region comprises a second region encompassing the remaining area of the transform block outside the first region.

5. The video decoding method of claim 4, wherein the signs of the transform coefficients in the first region are coded according to a first scan order and the signs of the transform coefficients in the second region are coded according to a second scan order different from the first scan order.

6. The video decoding method of claim 5, wherein the first scan order is a horizontal scan order and the second scan order is a diagonal scan order.

7. The video decoding method of claim 4, wherein parsing the bitstream to obtain signs of transform coefficients associated with the multiple regions of the transform block according to a region-based order used to signal the signs of the transform coefficients further comprises:

reading the signs of the transform coefficients from the bitstream according to the region-based order of the second region and then the first region.

8. The video decoding method of claim 3, wherein the at least one other region comprises a second region encompassing an area in the top-left corner of the transform block with a width and a height equal to a maximum area size threshold and outside the first region, wherein the at least one other region further comprises a third region compassing the remaining area of the transform block outside the first region and the second region.

9. The video decoding method of claim 8, wherein the signs of the transform coefficients in the first region, the second region, and the third region are coded according to at least two different scan orders.

10. The video decoding method of claim 9, wherein the signs of the transform coefficients in the first region and the second region are coded according to a horizontal scan order and the transform coefficients in the third region are coded according to a diagonal scan order.

11. The video decoding method of claim 5, wherein parsing the bitstream to obtain signs of transform coefficients associated with the multiple regions of the transform block according to a region-based order used to signal the signs of the transform coefficients further comprises:

reading the signs of the transform coefficients from the bitstream according to the region-based order of the third region, then the first region, and then the second region.

12. The video decoding method of claim 2, wherein the area size threshold is a fixed

value for all transform blocks of the video, wherein the area size threshold is not signaled in the bitstream.

13. The video decoding method of claim 2, wherein the area size threshold is adaptively determined for the transform block, wherein the area size threshold is signaled in the bitstream.

14. The video decoding method of claim 13, wherein the area size threshold is signaled in a video parameter set, a sequence parameter set, a picture parameter set, or a slice header and coding block level.

15. The video decoding method of claim 1, wherein the bitstream further comprises a sequence of sign signaling bits, wherein estimating the original signs for the set of candidate transform coefficients associated with the sign prediction area of the transform block further comprises:

predicting signs for a set of candidate transform coefficients associated with the sign prediction area of the transform block; and

estimating original signs for the set of candidate transform coefficients based on the set of predicted signs and the sequence of sign signaling bits.

16. The video decoding method of claim 15, wherein predicting signs for a set of candidate transform coefficients associated with the sign prediction area of the transform block further comprises:

determining a plurality of combinations of sign candidates for the set of candidate transform coefficients based on a total number of candidate transform coefficients in the set of candidate transform coefficients;

applying a template-based hypothesis generation scheme to generate a plurality of candidate hypotheses for the plurality of combinations of sign candidates, respectively; and

selecting a hypothesis from the plurality of candidate hypotheses as a set of predicted signs for the set of candidate transform coefficients.

17. The video decoding method of claim 1, further comprising:

applying an inverse primary transform and an inverse low-frequency non-separable transform (LFNST) to the transform coefficients of the transform block to generate residual

samples in a residual block corresponding to the transform block.

18. A video decoding apparatus, comprising:

a memory configured to store a bitstream ; and

a processor coupled to the memory and configured to perform the video decoding method for sign prediction of transform coefficients according to any of claims 1-17.

19. A non-transitory computer-readable storage medium having stored therein instructions which, when executed by a processor, cause the processor to perform the video decoding method for sign prediction of transform coefficients according to any of claims 1-17,

wherein the bitstream is stored in the computer-readable storage medium.

20. A non-transitory computer-readable storage medium having stored therein a bitstream comprising encoded data of transform coefficients of a transform block, wherein the bitstream is decodable by the video decoding method for sign prediction of transform coefficients according to any of claims 1-17.

21. A video encoding method for sign prediction of transform coefficients, comprising:

generating, by an encoder, transform coefficients of a transform block of a video frame from a video;

determining, by the encoder, multiple regions in the transform block, wherein the multiple regions include a sign prediction area for performing the sign prediction and at least one other region;

predicting, by the encoder, signs for a set of candidate transform coefficients associated with the sign prediction area of the transform block; and

generating, by the encoder, a bitstream to signal signs of the transform coefficients associated with the multiple regions of the transform block according to a region-based order.

22. The video encoding method of claim 21, wherein the multiple regions in the transform block are determined based on an area size threshold.

23. The video encoding method of claim 22, wherein the sign prediction area is determined to be a first region in a top-left corner of the transform block with a width and a

height equal to the area size threshold.

24. The video encoding method of claim 23, wherein the at least one other region comprises a second region encompassing the remaining area of the transform block outside the first region.

25. The video encoding method of claim 24, further comprising:

encoding the transform coefficients in the first region according to a first scan order; and

encoding the transform coefficients in the second region according to a second scan order different from the first scan order.

26. The video encoding method of claim 25, wherein the first scan order is a horizontal scan order and the second scan order is a diagonal scan order.

27. The video encoding method of claim 24, wherein the signs of the transform coefficients associated with the multiple regions of the transform block are signaled according to the region-based order of the second region and then the first region.

28. The video encoding method of claim 23, wherein the at least one other region comprises a second region encompassing an area in the top-left corner of the transform block with a width and a height equal to a maximum area size threshold and outside the first region, wherein the at least one other region further comprises a third region compassing the remaining area of the transform block outside the first region and the second region.

29. The video encoding method of claim 28, further comprising:

coding the signs of the transform coefficients in the first region, the second region, and the third region according to at least two different scan orders.

30. The video encoding method of claim 29, wherein the transform coefficients in the first region and the second region are coded according to a horizontal scan order and the transform coefficients in the third region are coded according to a diagonal scan order.

31. The video encoding method of claim 25, wherein the signs of transform

coefficients associated with the multiple regions of the transform block are signaled according to the region-based order of the third region, then the first region, and then the second region.

32. The video encoding method of claim 22, wherein the area size threshold is a fixed value for all transform blocks of the video, wherein the area size threshold is not signaled in the bitstream.

33. The video encoding method of claim 22, further comprising:

adaptively determining the area size threshold for the transform block; and

signaling the area size threshold in the bitstream.

34. The video encoding method of claim 33, wherein the area size threshold is signaled in a video parameter set, a sequence parameter set, a picture parameter set, or a slice header and coding block level.

35. The video encoding method of claim 21, further comprising:

generating a sequence of sign signaling bits based on the predicted signs for the set of candidate transform coefficients; and

signaling the sequence of sign signaling bits in the bitstream.

36. The video encoding method of claim 21, wherein predicting signs for a set of candidate transform coefficients associated with the sign prediction area of the transform block further comprises:

determining a plurality of combinations of sign candidates for the set of candidate transform coefficients based on a total number of candidate transform coefficients in the set of candidate transform coefficients;

applying a template-based hypothesis generation scheme to generate a plurality of candidate hypotheses for the plurality of combinations of sign candidates, respectively; and

selecting a hypothesis from the plurality of candidate hypotheses as a set of predicted signs for the set of candidate transform coefficients.

37. The video encoding method of claim 1, further comprising:

applying a primary transform and a low-frequency non-separable transform (LFNST) to generate the transform coefficients of the transform block.

38. A video encoding apparatus, comprising:

a processor configured to perform the video encoding method for sign prediction of transform coefficients according to any of claims 21-37; and

a memory coupled to the processor and configured to store the bitstream.

39. A non-transitory computer-readable storage medium having stored therein and instructions which, when executed by a processor, cause the processor to perform the video encoding method for sign prediction of transform coefficients according to any of claims 21-37,

wherein the bitstream is stored in the computer-readable storage medium.

40. A non-transitory computer-readable storage medium having stored therein a bitstream generated by the video encoding method for sign prediction of transform coefficients according to any of claims 21-37.

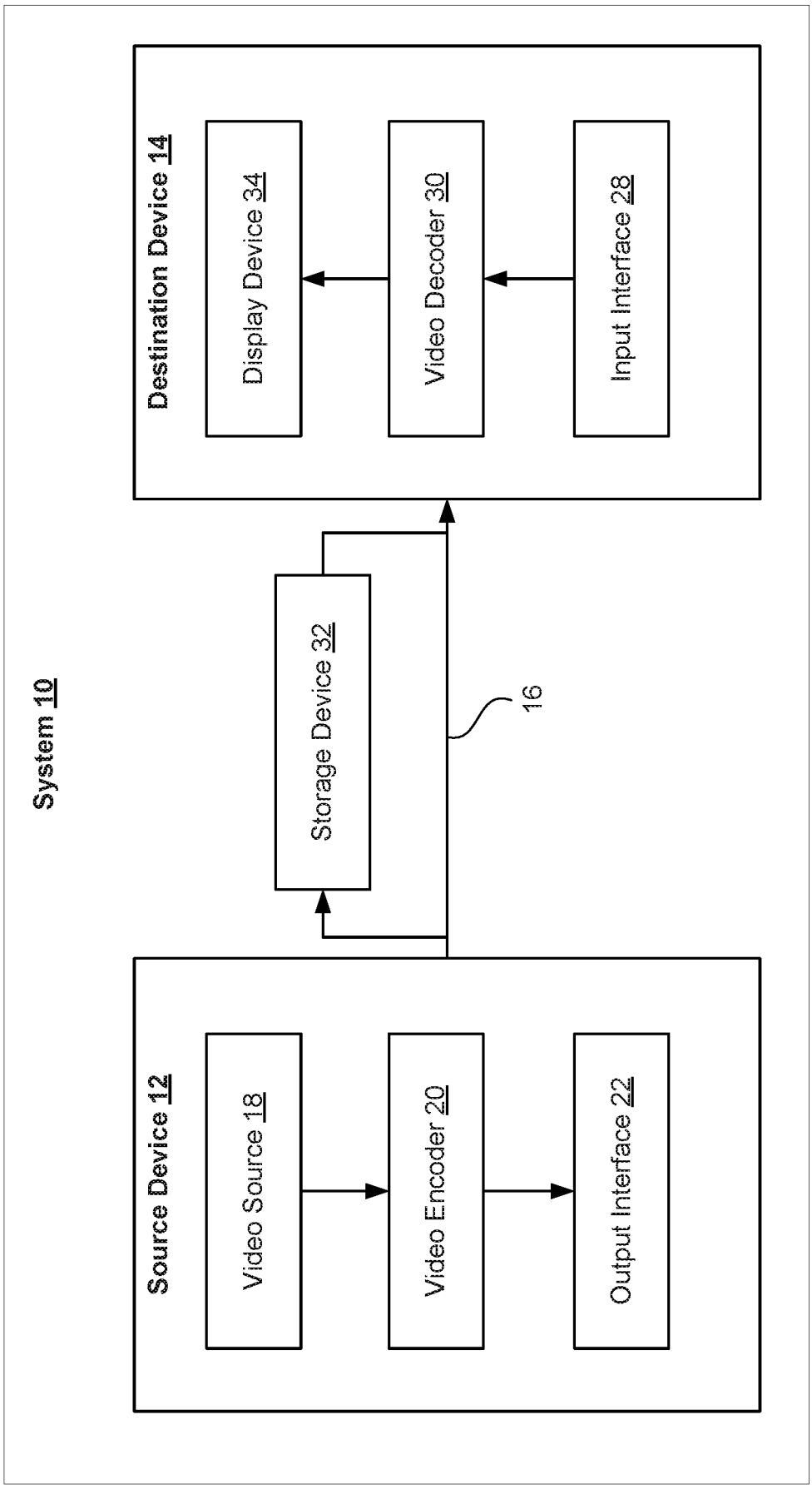

System 10
Source Device 12
Video Source 18
Video Encoder 20
Output Interface 22
16
Storage Device 32
Destination Device 14
Display Device 34
Video Decoder 30
Input Interface 28

FIG. 1

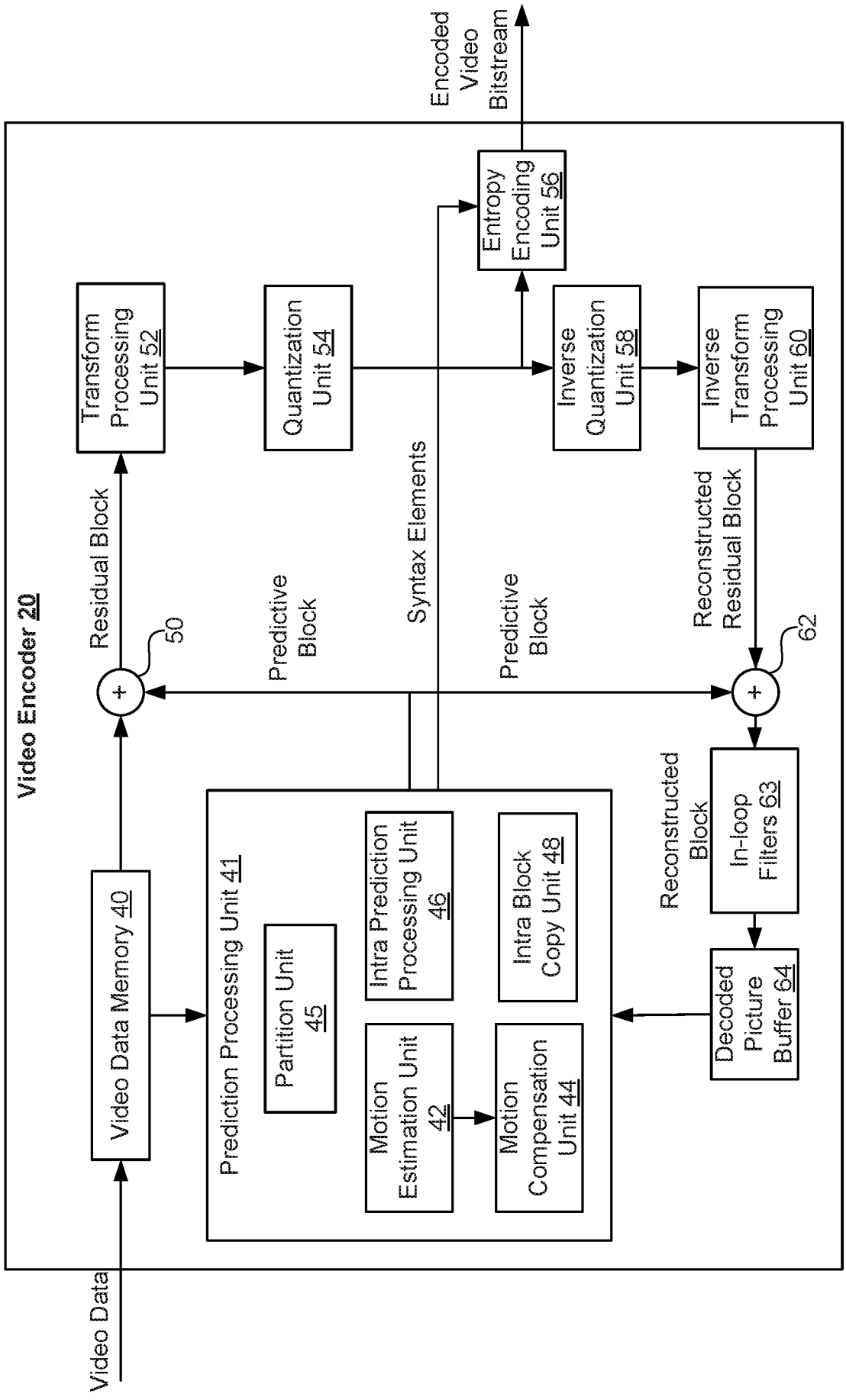

Video Data
Video Encoder 20
Video Data Memory 40
Prediction Processing Unit 41
Partition Unit 45
Motion Estimation Unit 42
Motion Compensation Unit 44
Intra Prediction Processing Unit 46
Intra Block Copy Unit 48
Residual Block
50
Transform Processing Unit 52
Quantization Unit 54
Entropy Encoding Unit 56
Encoded Video Bitstream
Syntax Elements
Predictive Block
Predictive Block
Inverse Quantization Unit 58
Inverse Transform Processing Unit 60
Reconstructed Residual Block
62
Reconstructed Block
In-loop Filters 63
Decoded Picture Buffer 64

FIG. 2

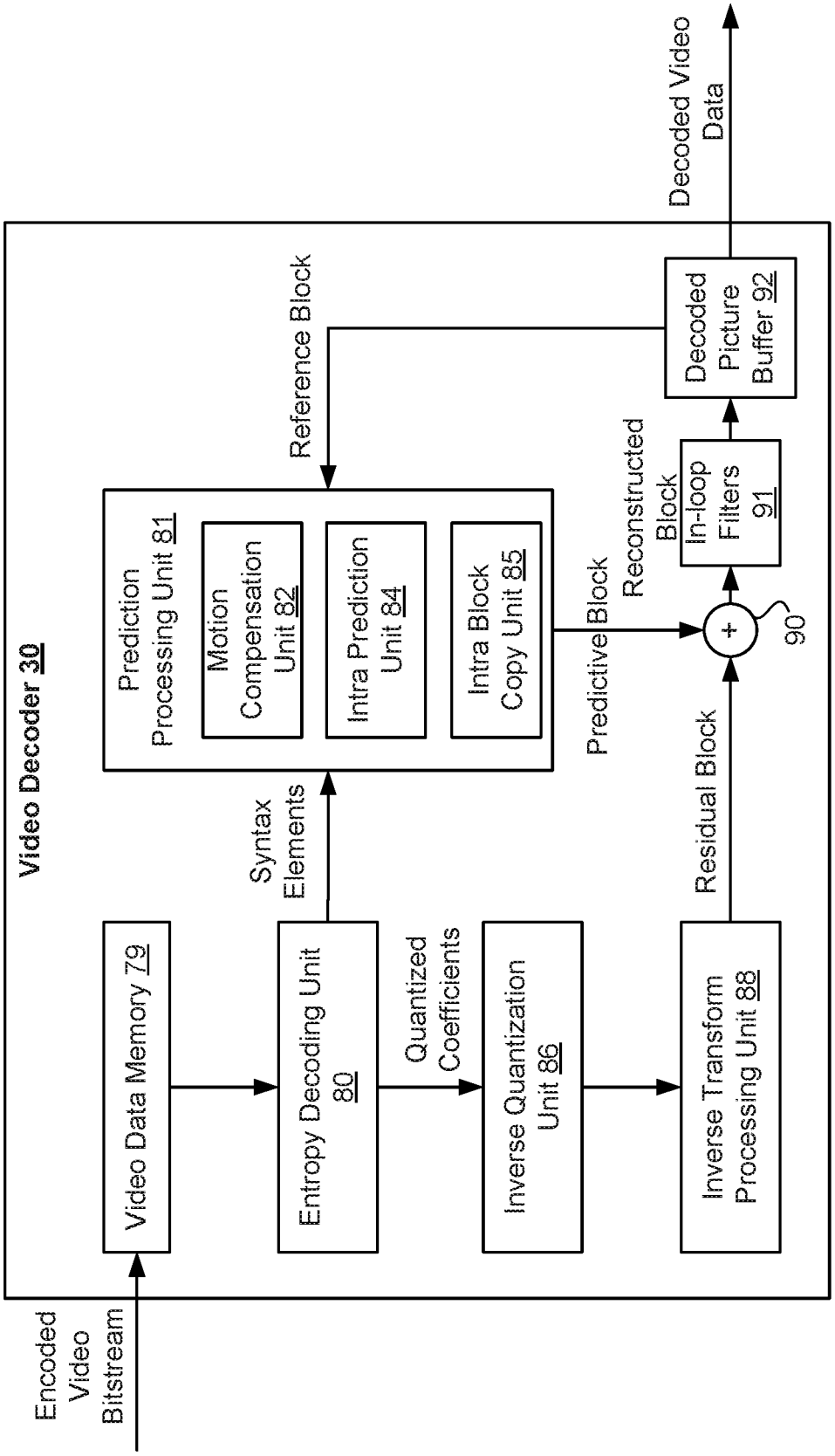

Video Decoder 30
Video Data Memory 79
Entropy Decoding Unit 80
Inverse Quantization Unit 86
Inverse Transform Processing Unit 88
Prediction Processing Unit 81
Motion Compensation Unit 82
Intra Prediction Unit 84
Intra Block Copy Unit 85
In-loop Filters 91
Decoded Picture Buffer 92
Encoded Video Bitstream
Syntax Elements
Quantized Coefficients
Residual Block
Predictive Block
Reconstructed Block
Reference Block
Decoded Video Data
90

FIG. 3

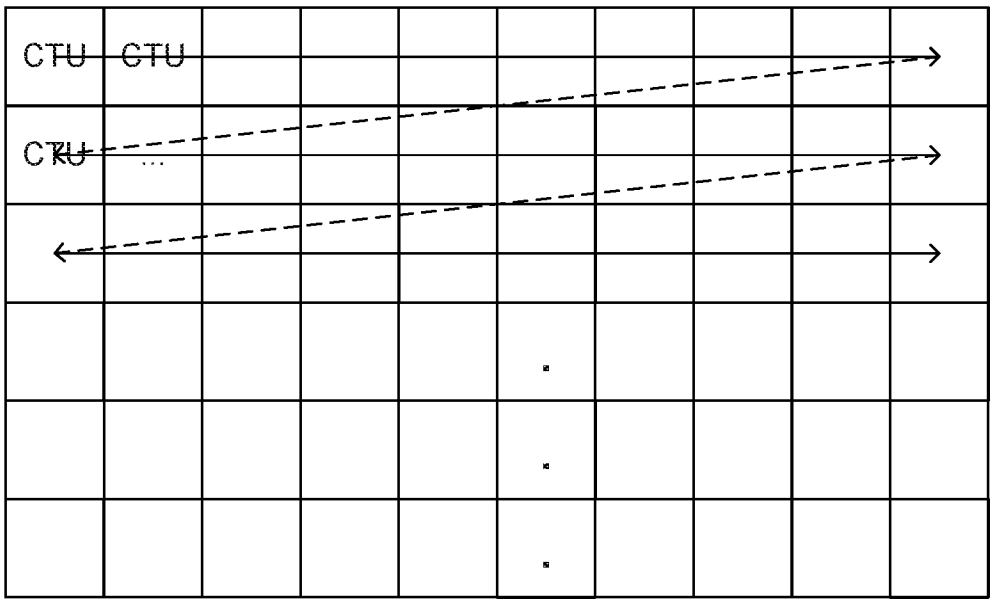

CTU
CTU
CTU
...

FIG. 4A

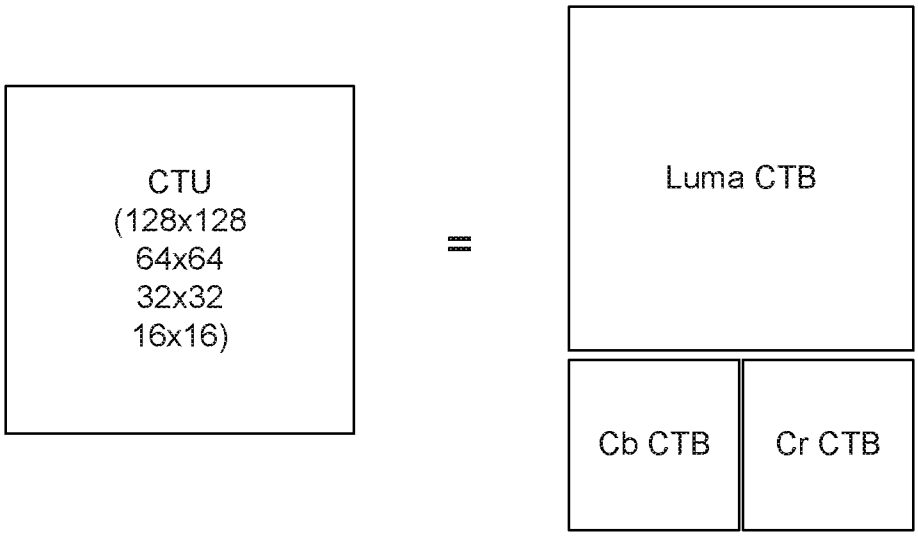

CTU
(128x128
64x64
32x32
16x16)
=
Luma CTB
Cb CTB
Cr CTB

FIG. 4B

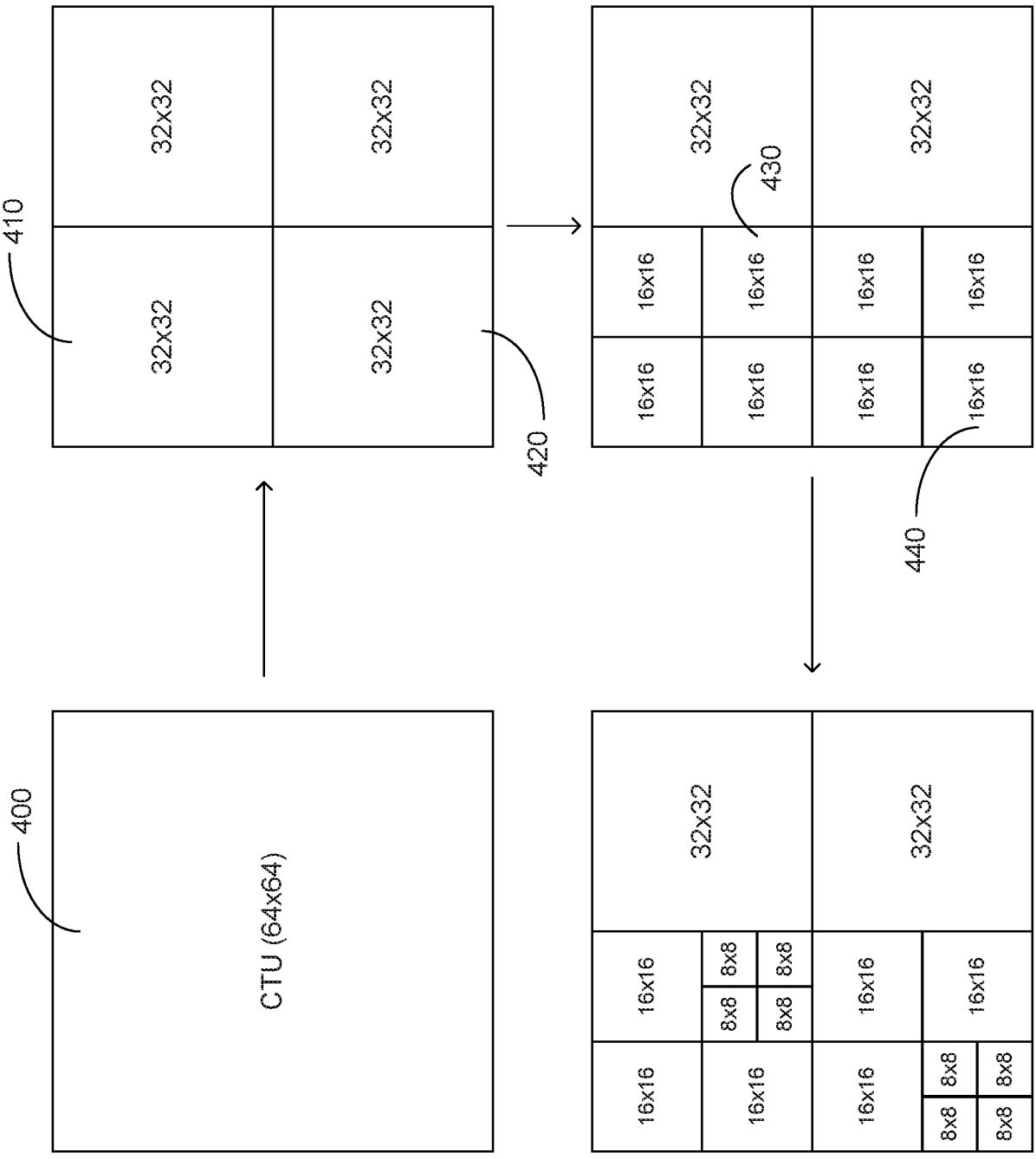

400
CTU (64x64)
410
32x32
32x32
32x32
32x32
420
430
440
16x16
8x8

FIG. 4C

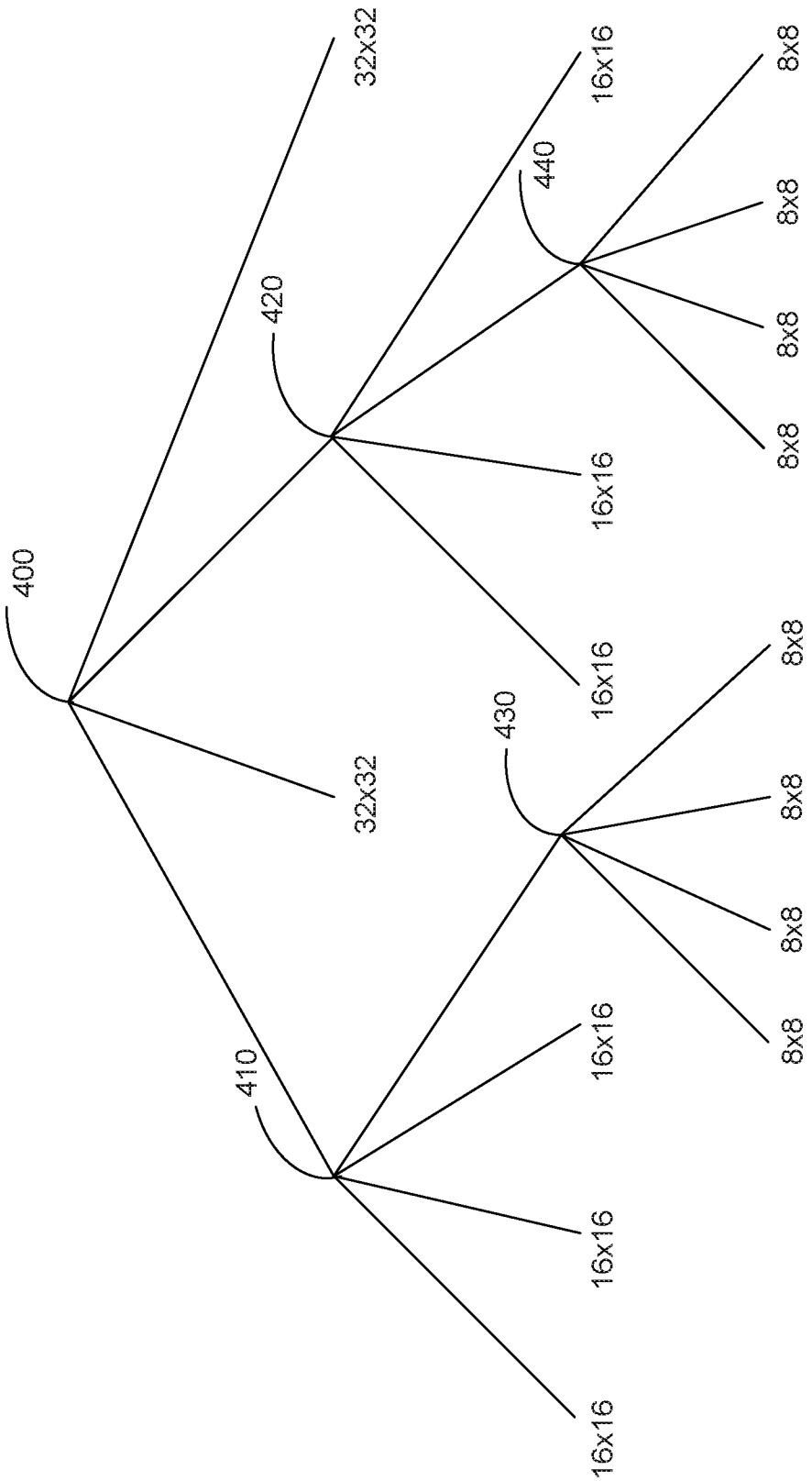

400
32x32
420
16x16
440
8x8
8x8
8x8
16x16
16x16
32x32
430
8x8
8x8
8x8
410
16x16
16x16
16x16

FIG. 4D

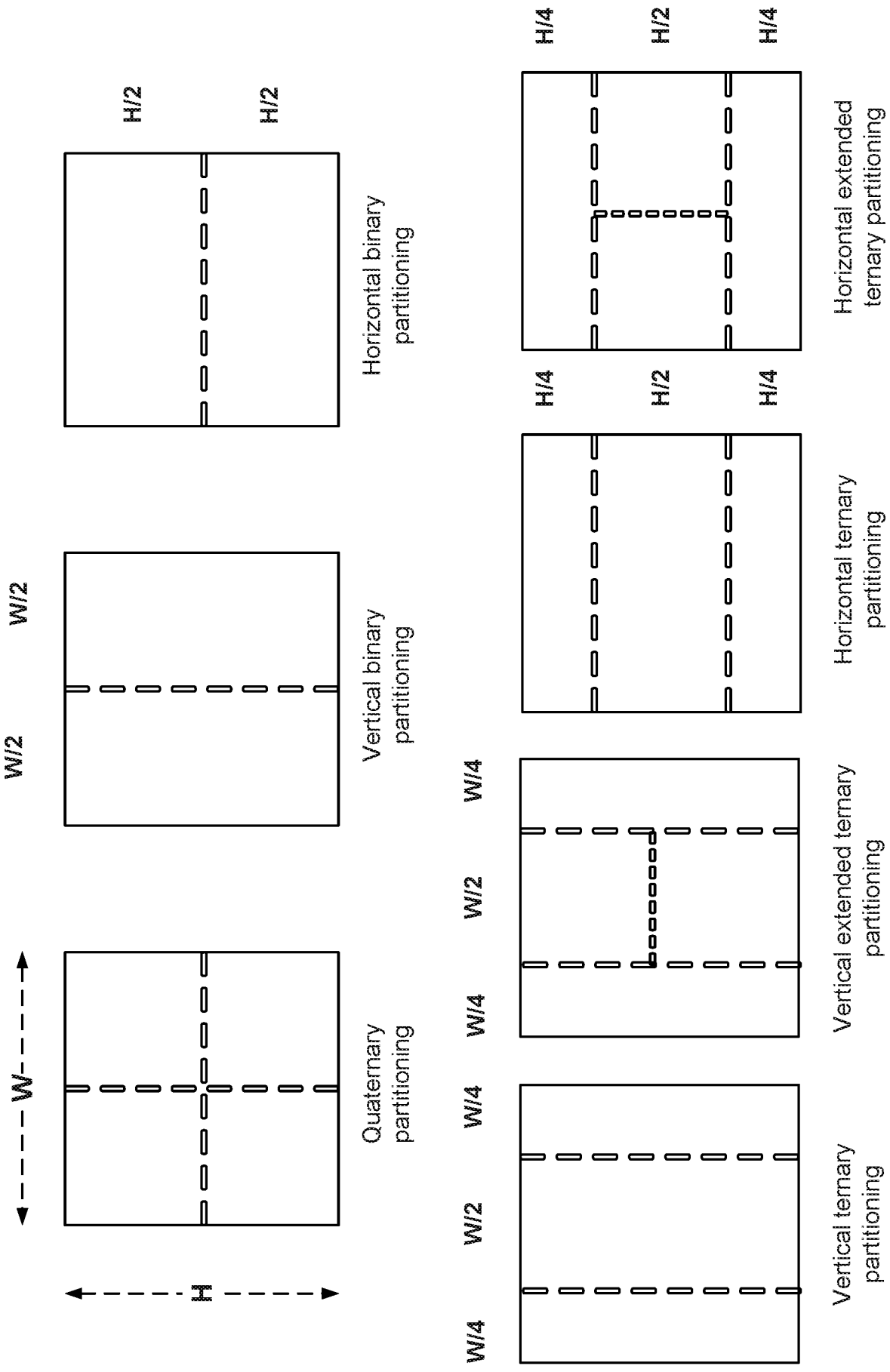

W
H
W/2
W/2
H/2
H/2
Quaternary partitioning
Vertical binary partitioning
Horizontal binary partitioning
W/4
W/2
W/4
W/4
W/2
W/4
H/4
H/2
H/4
H/4
H/2
H/4
Vertical ternary partitioning
Vertical extended ternary partitioning
Horizontal ternary partitioning
Horizontal extended ternary partitioning

FIG. 4E

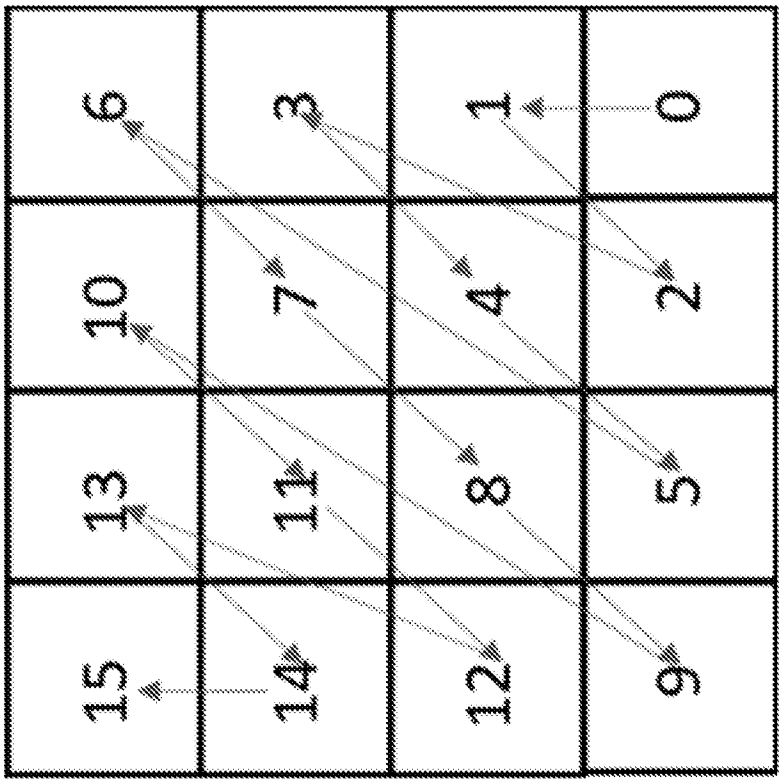


FIG. 5

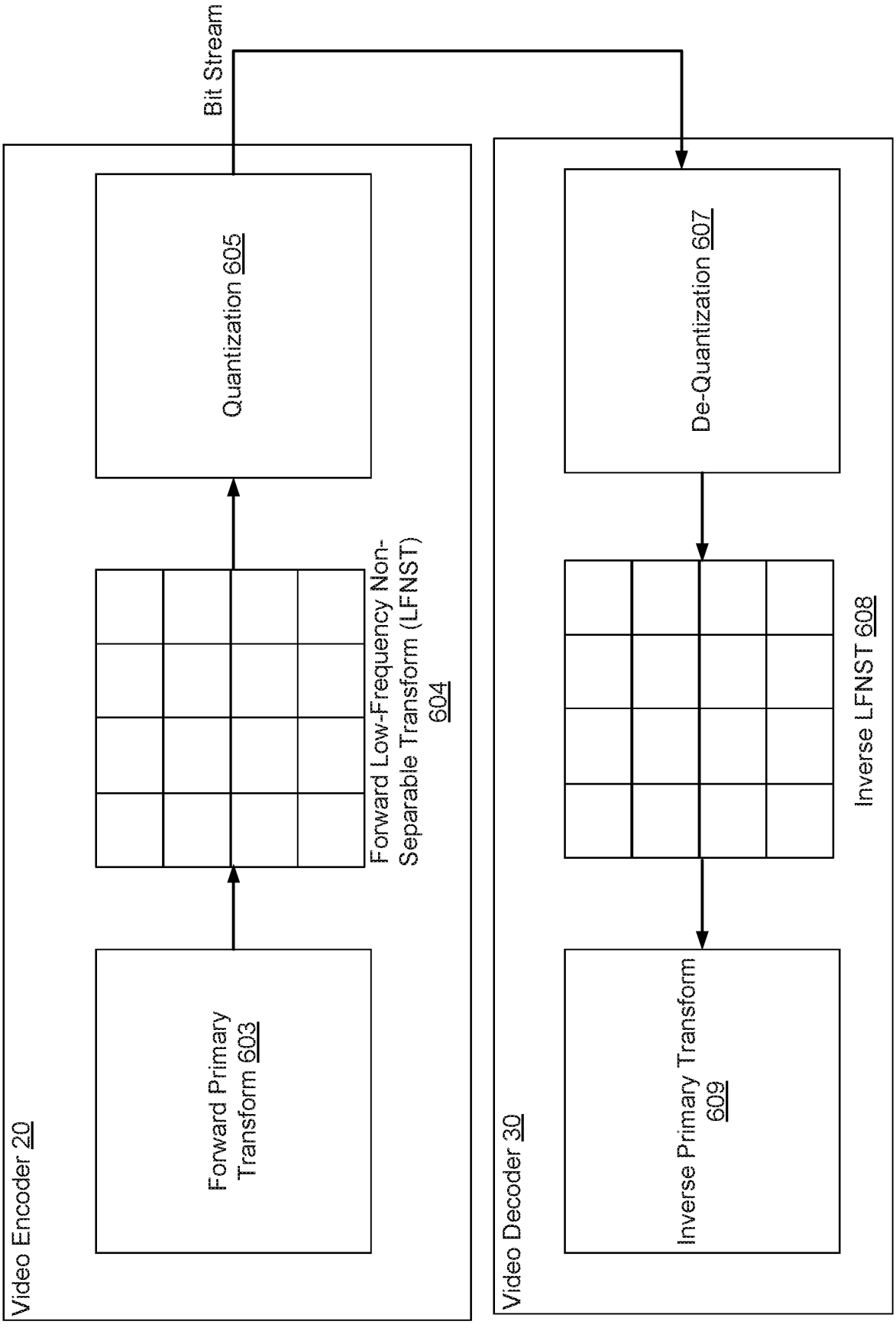

Bit Stream
Video Encoder 20
Forward Primary Transform 603
Forward Low-Frequency Non-Separable Transform (LFNST) 604
Quantization 605
Video Decoder 30
Inverse Primary Transform 609
Inverse LFNST 608
De-Quantization 607

FIG. 6

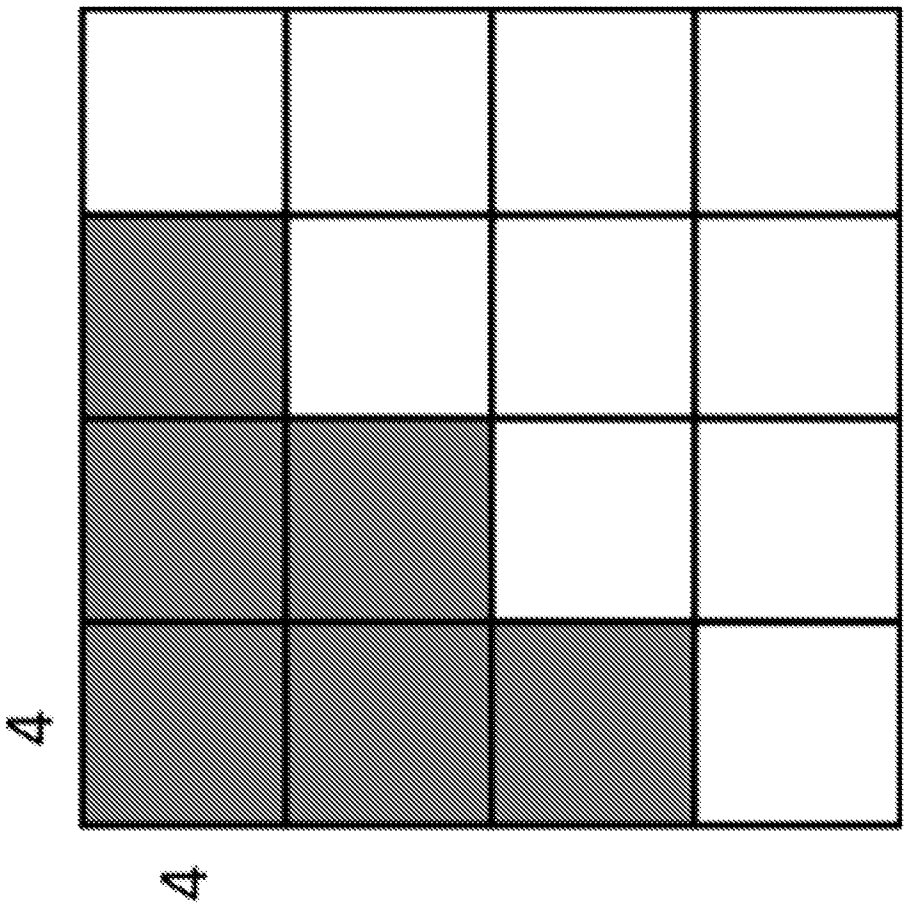


FIG. 7

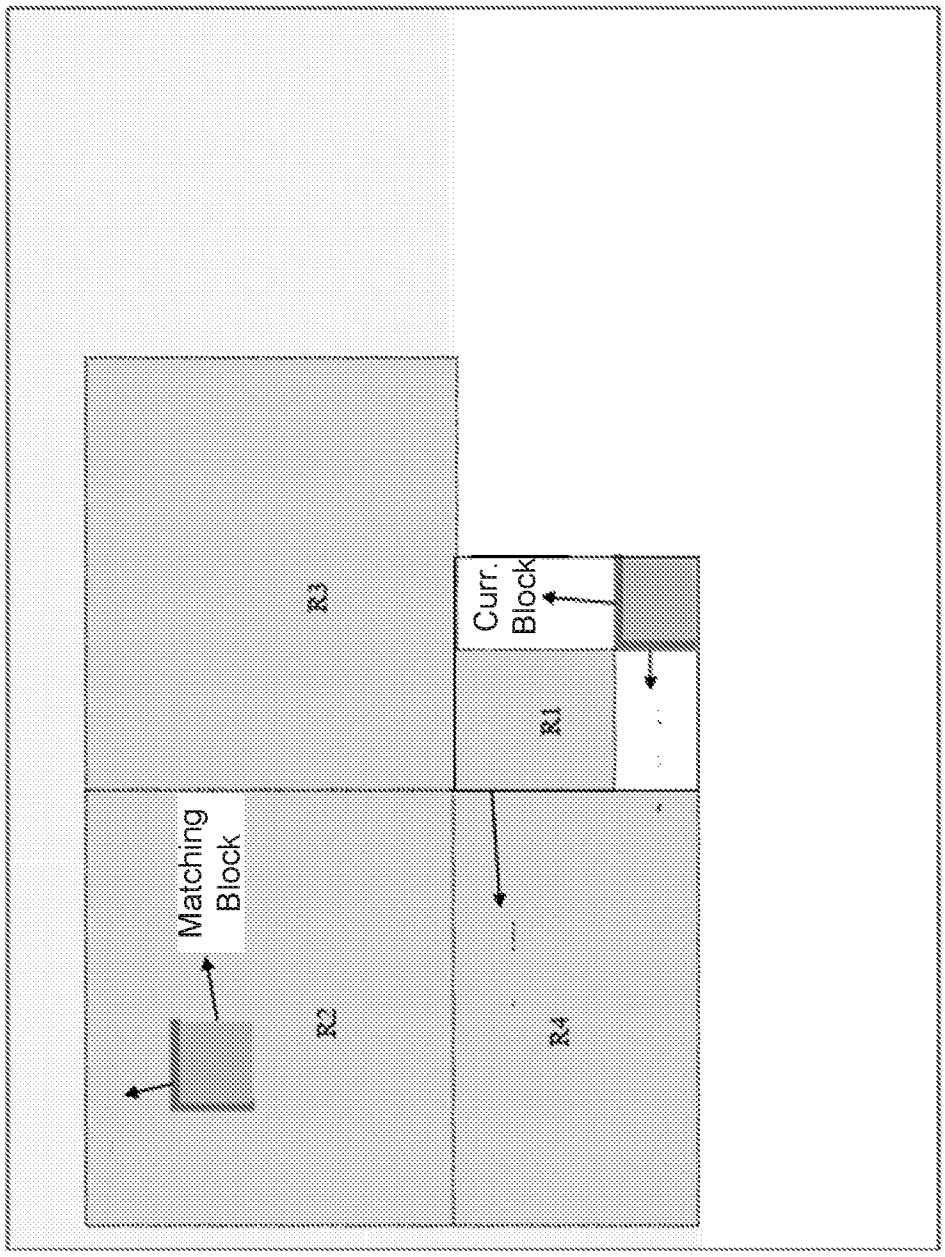


FIG. 8

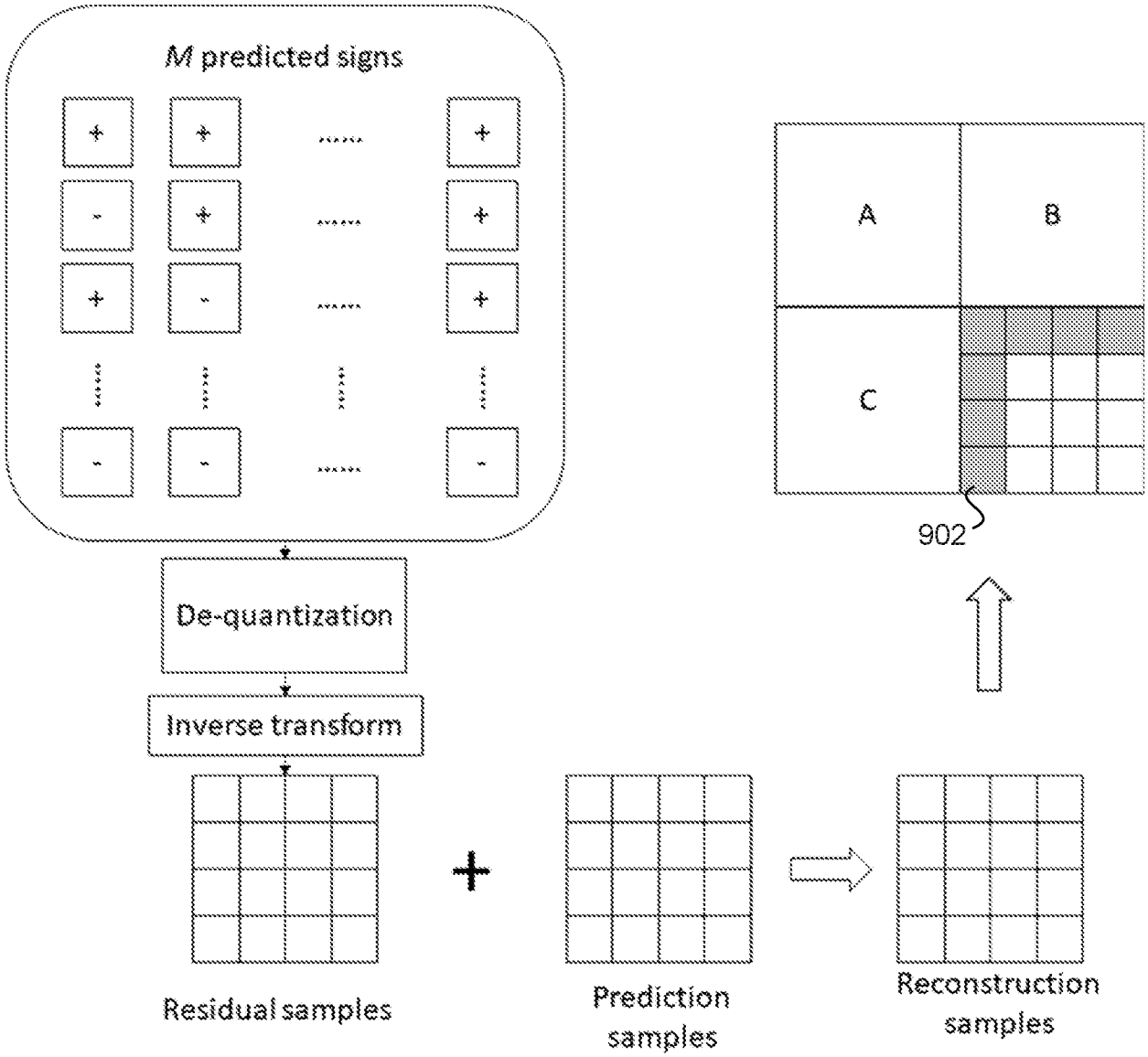

M predicted signs
De-quantization
Inverse transform
Residual samples
Prediction samples
Reconstruction samples
A
B
C
902

FIG. 9

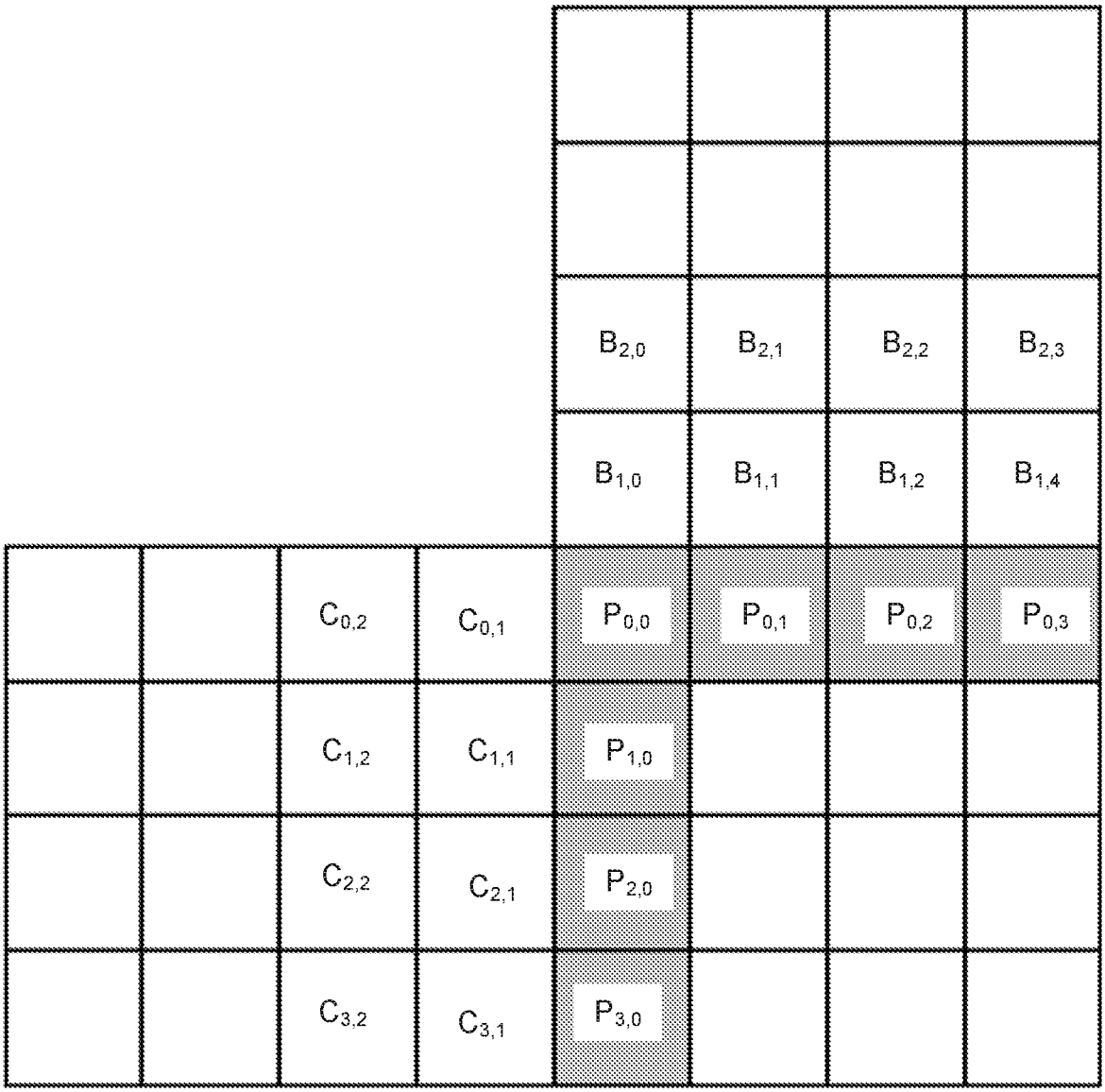

B2,0
B2,1
B2,2
B2,3
B1,0
B1,1
B1,2
B1,4
C0,2
C0,1
P0,0
P0,1
P0,2
P0,3
C1,2
C1,1
P1,0
C2,2
C2,1
P2,0
C3,2
C3,1
P3,0

FIG. 10

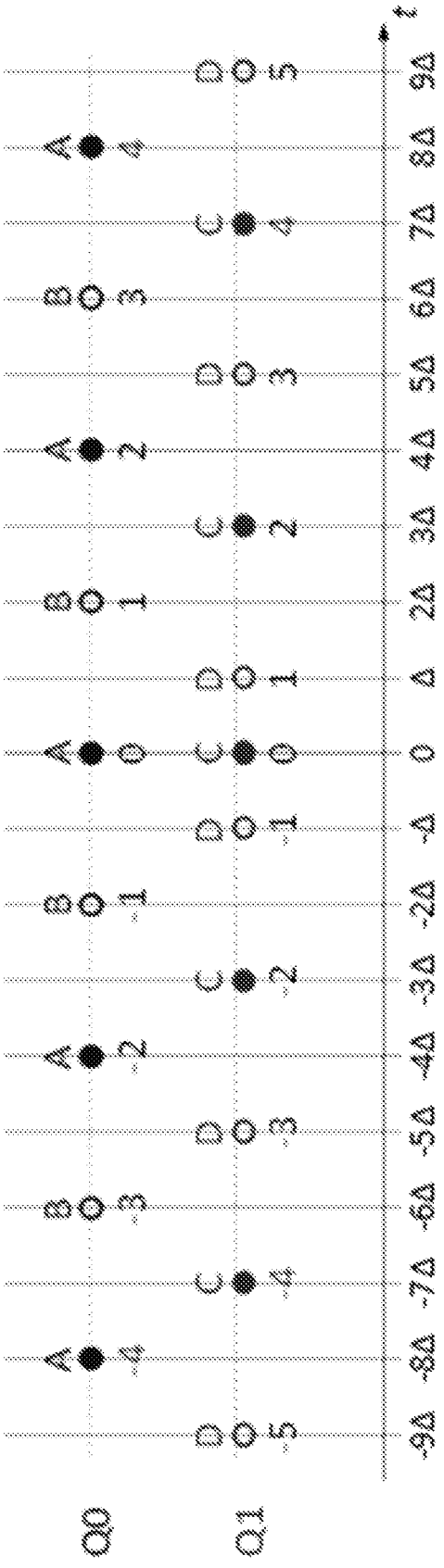


FIG. 11

current state	next state for ...	
	(k & 1) == 0	(k & 1) == 1
0	0	2
1	2	0
2	1	3
3	3	1

FIG. 12B

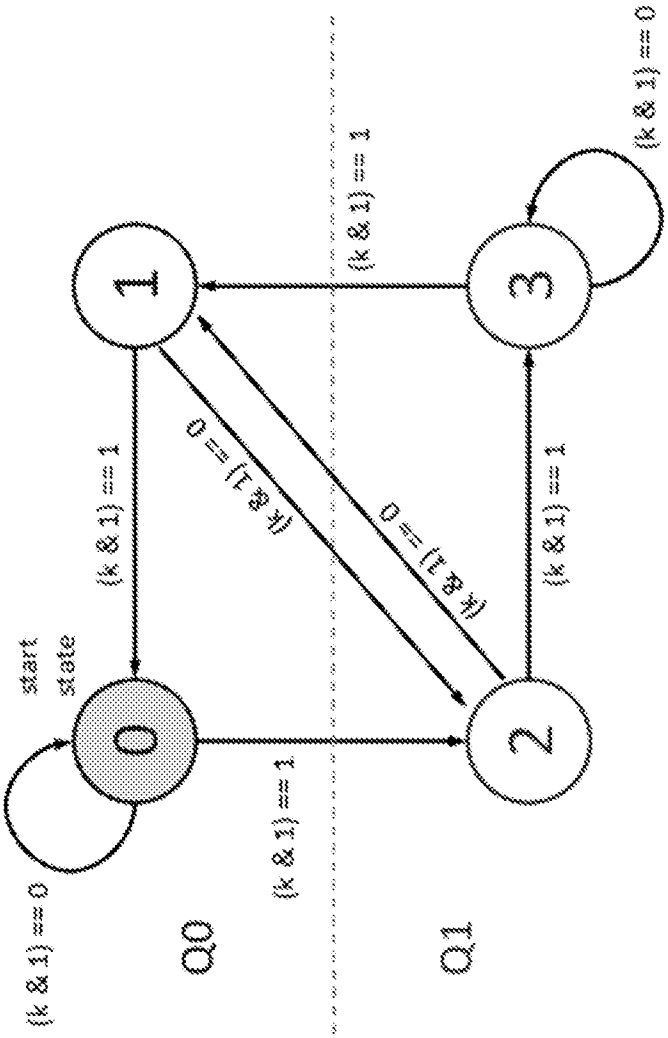


FIG. 12A

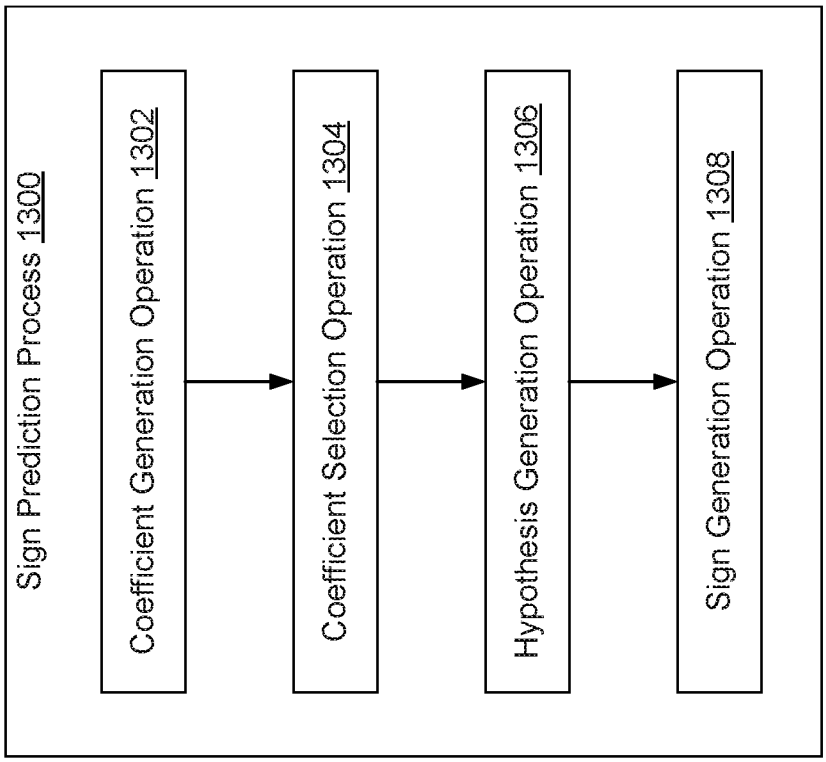

Sign Prediction Process 1300
Coefficient Generation Operation 1302
Coefficient Selection Operation 1304
Hypothesis Generation Operation 1306
Sign Generation Operation 1308

FIG. 13

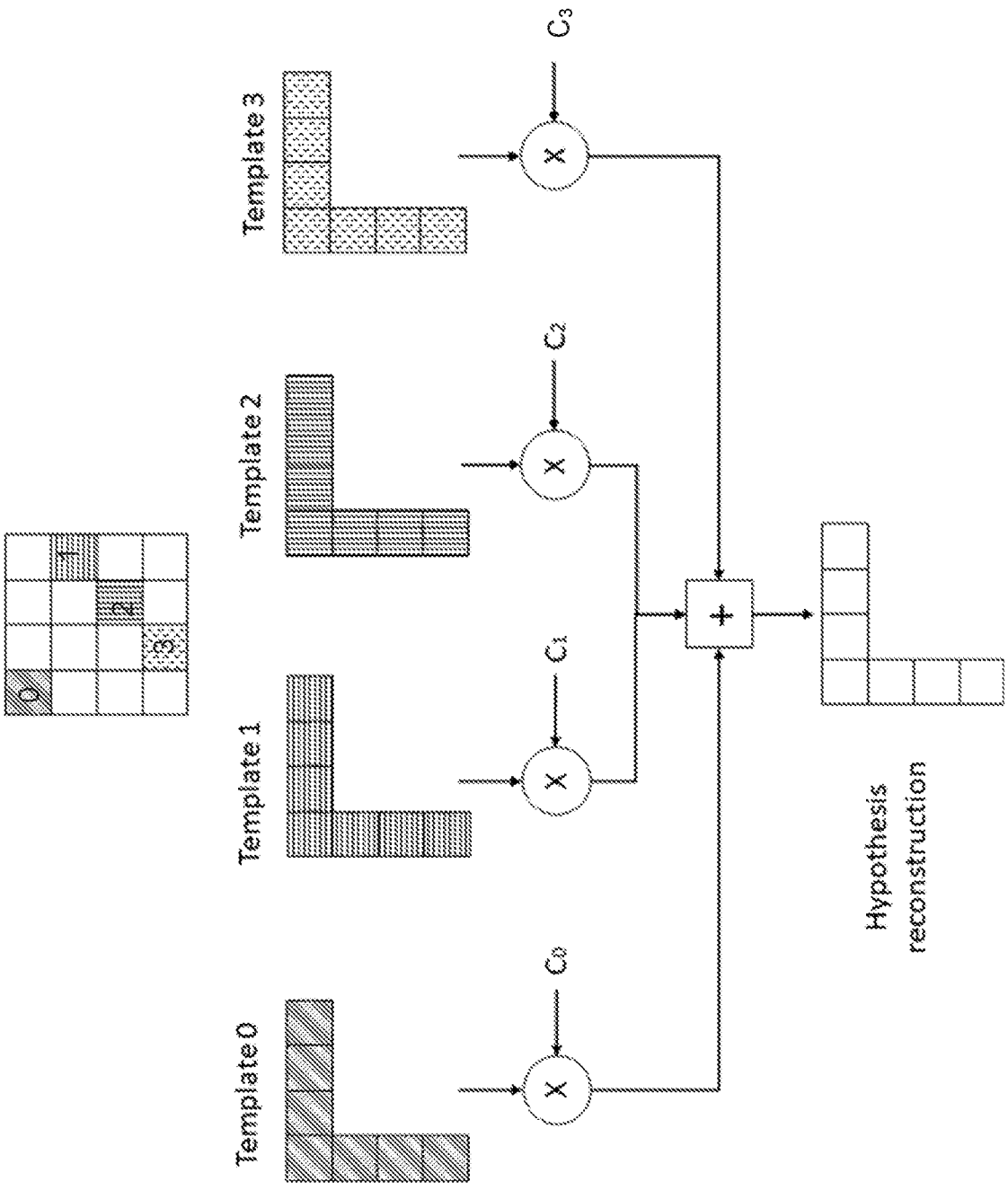

0
1
2
3
Template 0
Template 1
Template 2
Template 3
C0
C1
C2
C3
X
+
Hypothesis reconstruction

FIG. 14

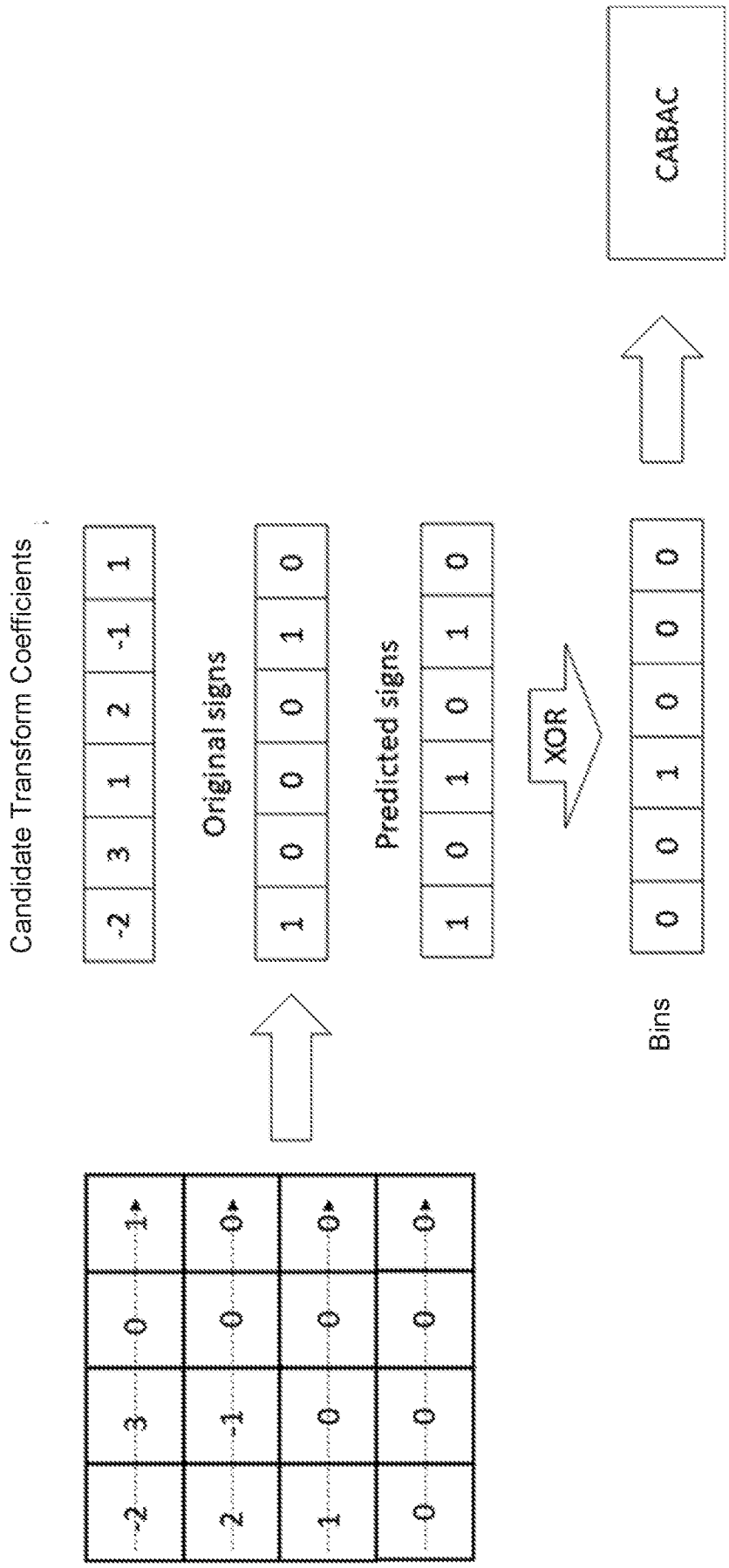

Candidate Transform Coefficients
-2 3 1 2 -1 1
Original signs
1 0 0 0 1 0
Predicted signs
1 0 1 0 1 0
XOR
0 0 1 0 0 0
Bins
CABAC
-2 3 1 2 -1 1
2 -1 0 0 0 0
1 0 0 0 0 0
0 0 0 0 0 0

FIG. 15A

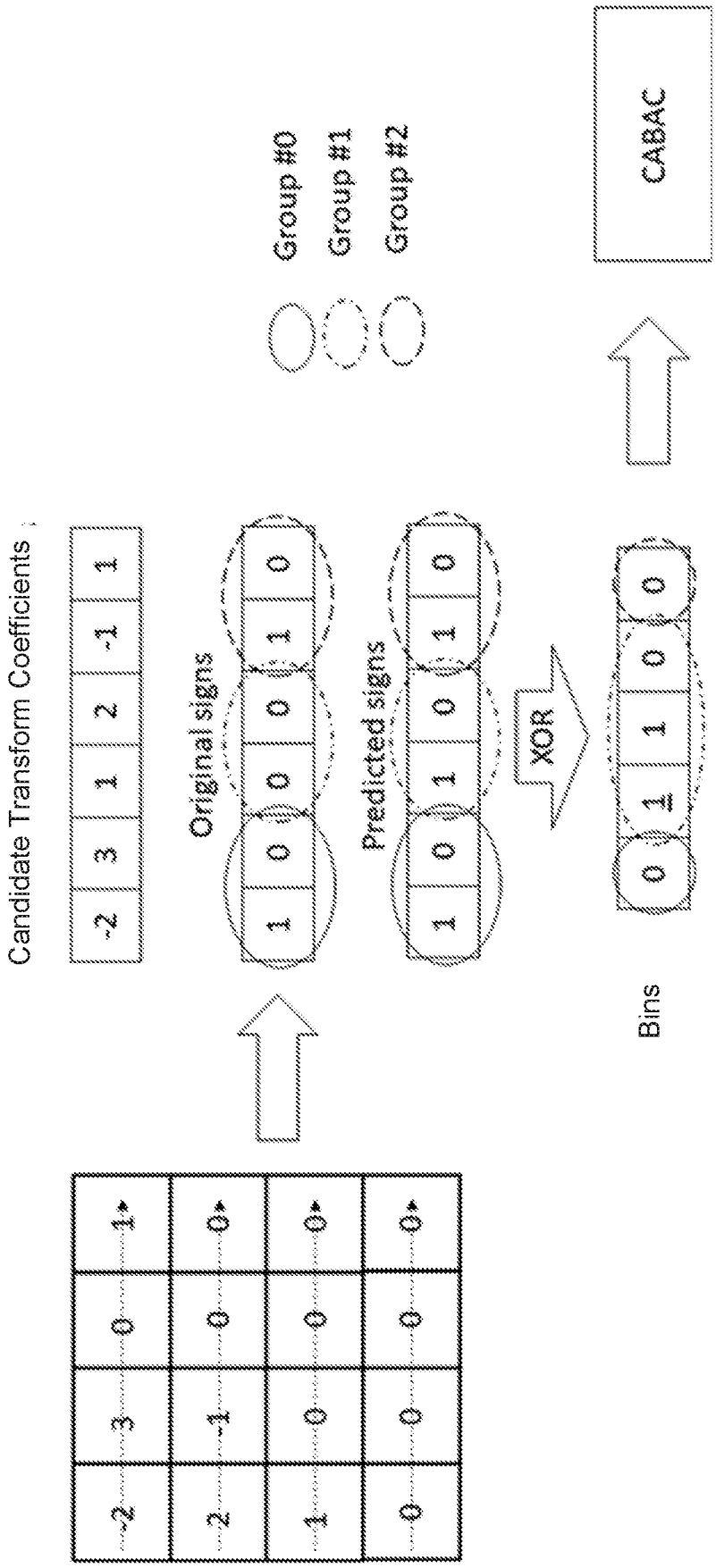

Candidate Transform Coefficients
-2 3 1 2 -1 1
Original signs
1 0 0 0 1 0
Predicted signs
1 0 1 0 1 0
XOR
0 1 1 0 0
Bins
Group #0
Group #1
Group #2
CABAC

FIG. 15B

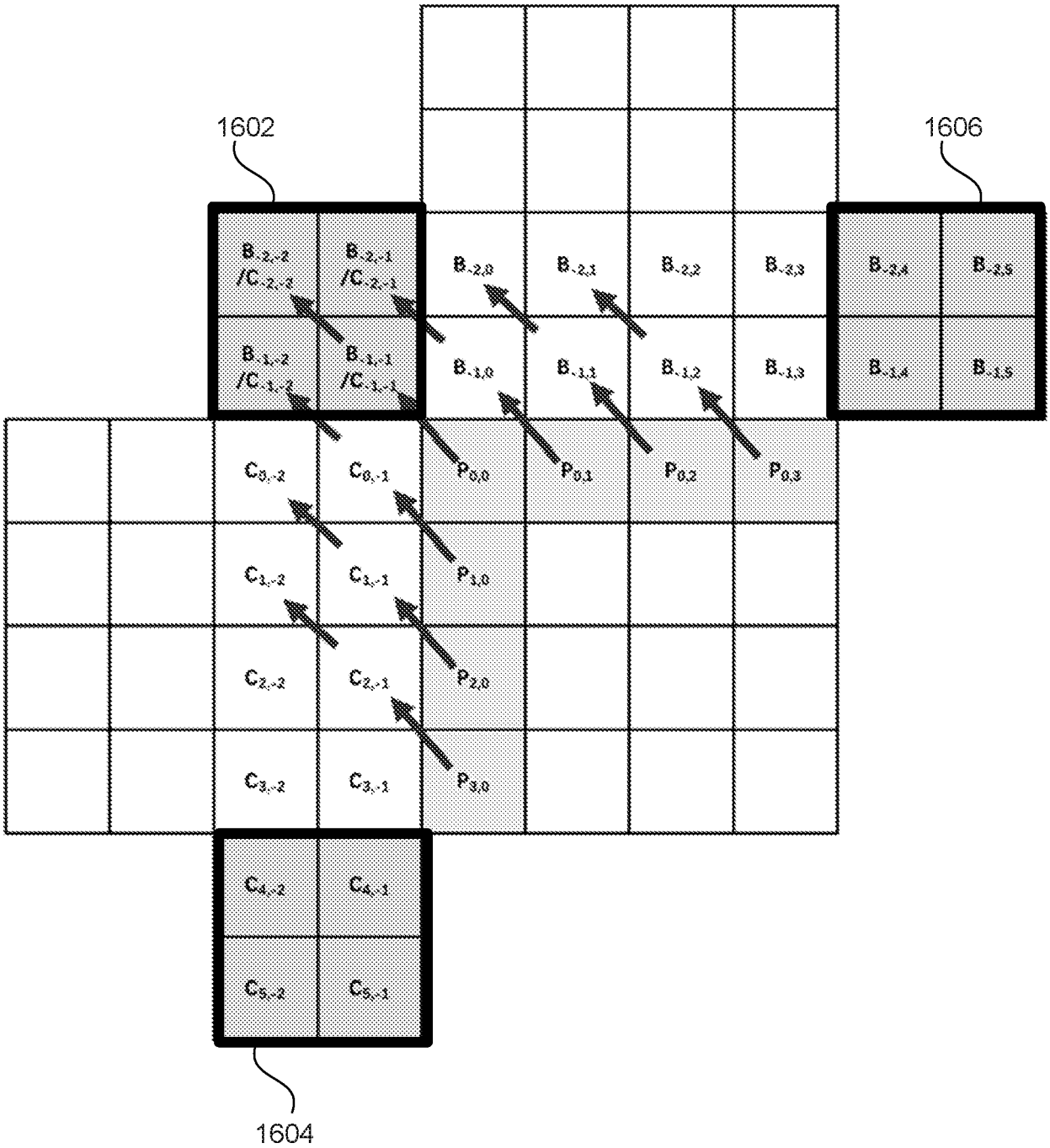

1602
1606
1604
B-2,-2 /C-2,-2
B-2,-1 /C-2,-1
B-2,0
B-2,1
B-2,2
B-2,3
B-2,4
B-2,5
B-1,-2 /C-1,-2
B-1,-1 /C-1,-1
B-1,0
B-1,1
B-1,2
B-1,3
B-1,4
B-1,5
C0,-2
C0,-1
P0,0
P0,1
P0,2
P0,3
C1,-2
C1,-1
P1,0
C2,-2
C2,-1
P2,0
C3,-2
C3,-1
P3,0
C4,-2
C4,-1
C5,-2
C5,-1

FIG. 16A

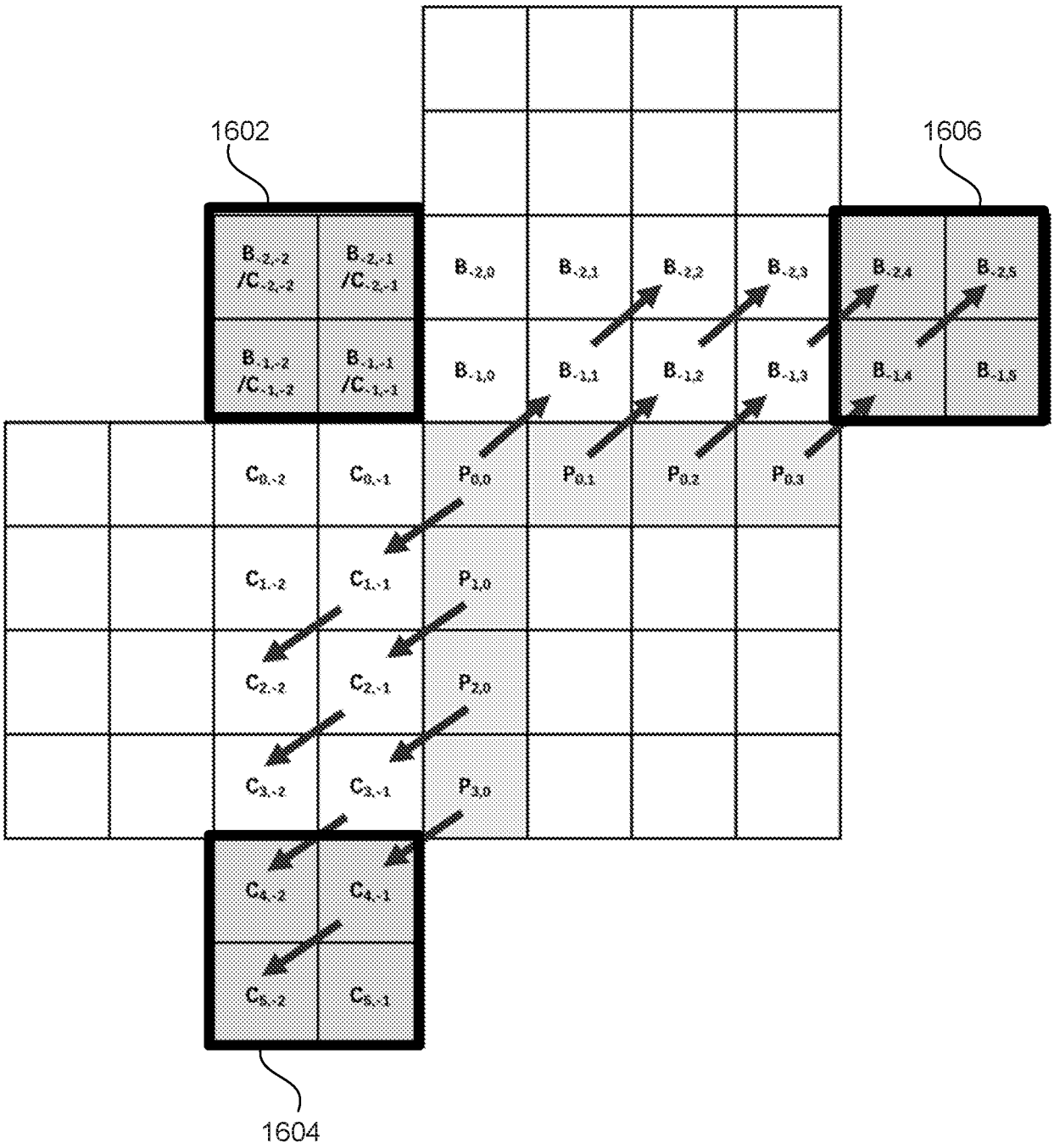

1602
1606
1604
B-2,-2 /C-2,-2
B-2,-1 /C-2,-1
B-2,0
B-2,1
B-2,2
B-2,3
B-2,4
B-2,5
B-1,-2 /C-1,-2
B-1,-1 /C-1,-1
B-1,0
B-1,1
B-1,2
B-1,3
B-1,4
B-1,5
C0,-2
C0,-1
P0,0
P0,1
P0,2
P0,3
C1,-2
C1,-1
P1,0
C2,-2
C2,-1
P2,0
C3,-2
C3,-1
P3,0
C4,-2
C4,-1
C5,-2
C5,-1

FIG. 16B

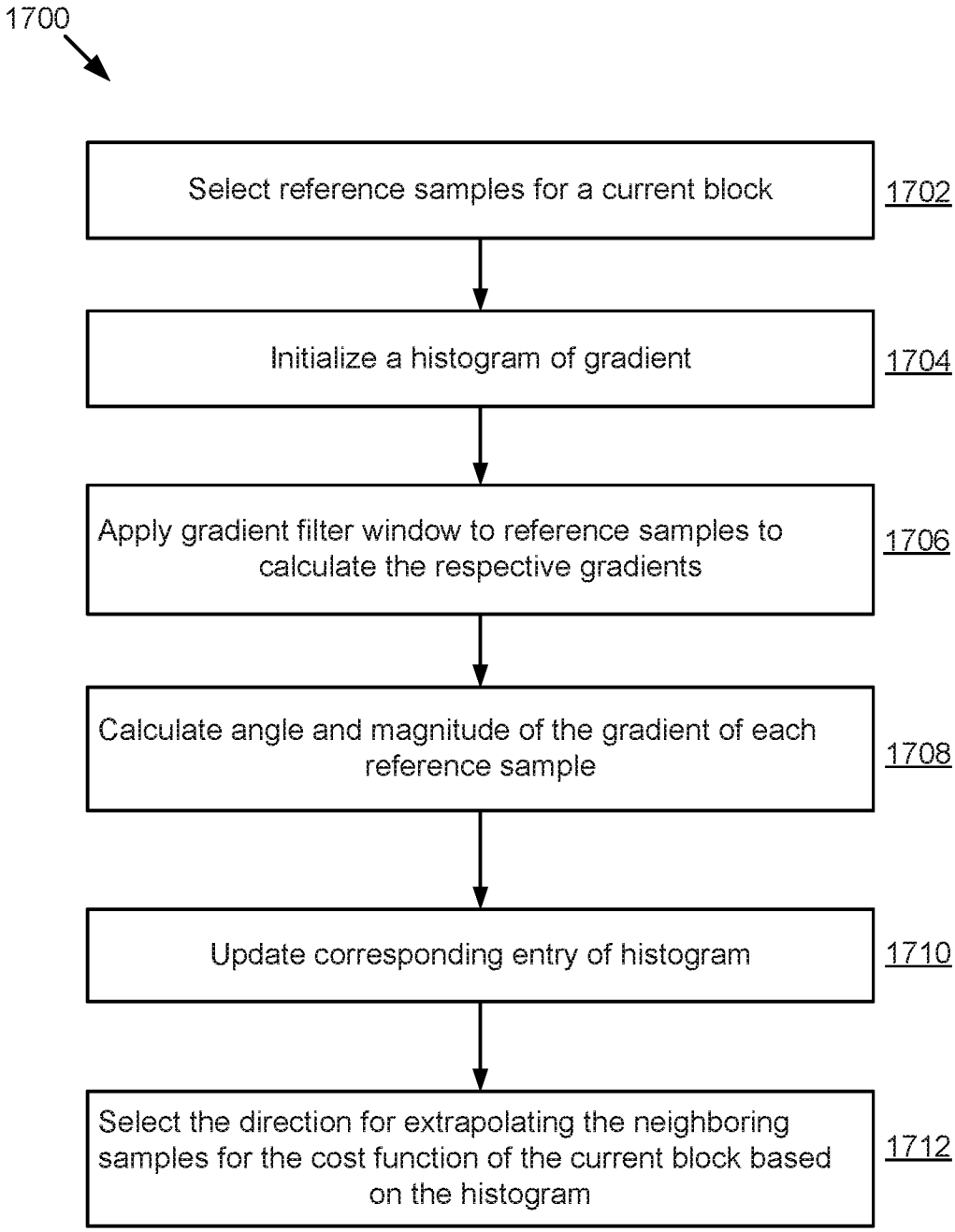

1700
Select reference samples for a current block
1702
Initialize a histogram of gradient
1704
Apply gradient filter window to reference samples to calculate the respective gradients
1706
Calculate angle and magnitude of the gradient of each reference sample
1708
Update corresponding entry of histogram
1710
Select the direction for extrapolating the neighboring samples for the cost function of the current block based on the histogram
1712

FIG. 17

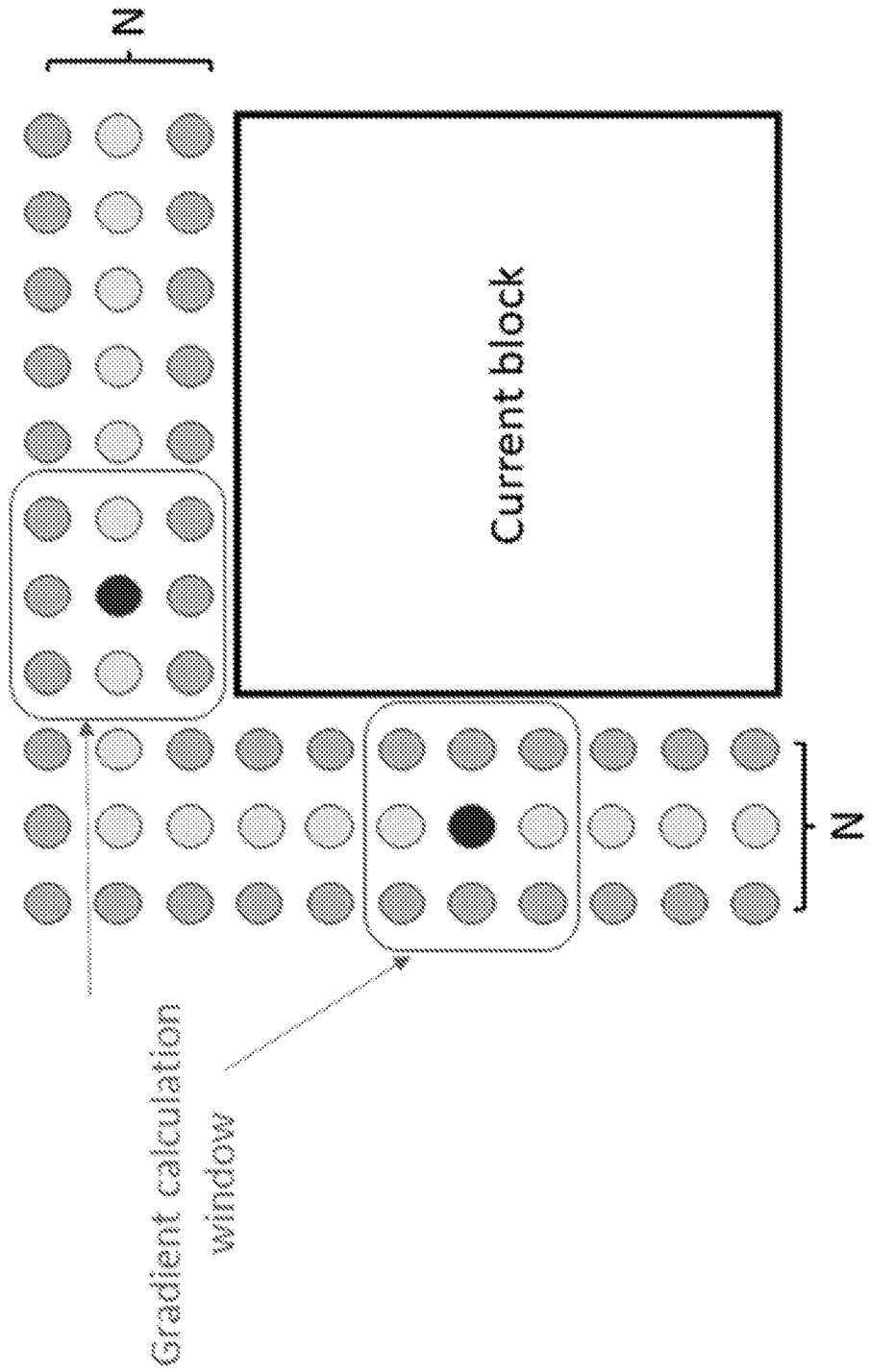


FIG. 18A

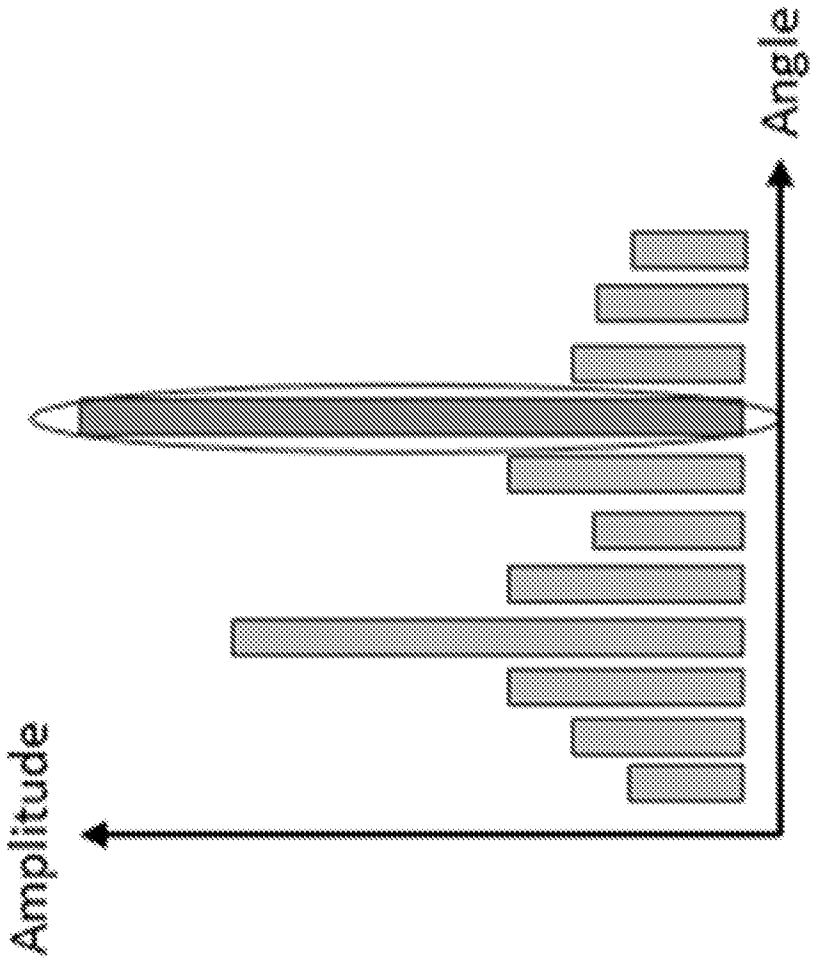


FIG. 18B

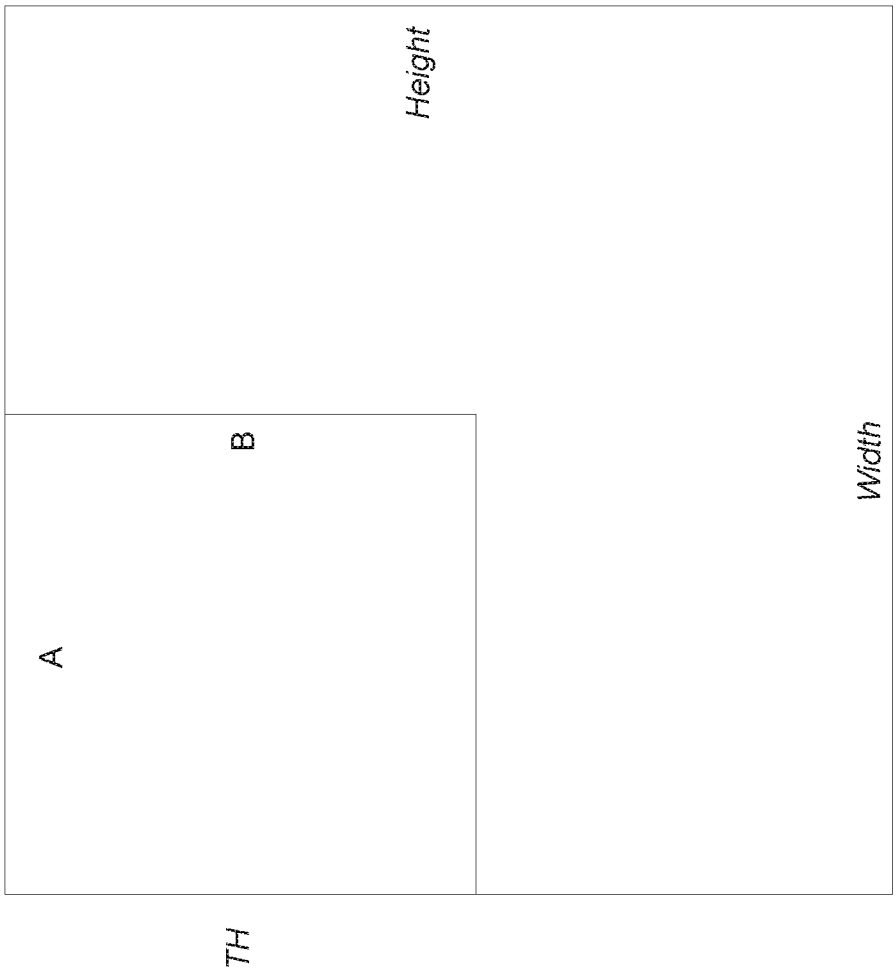


FIG. 19

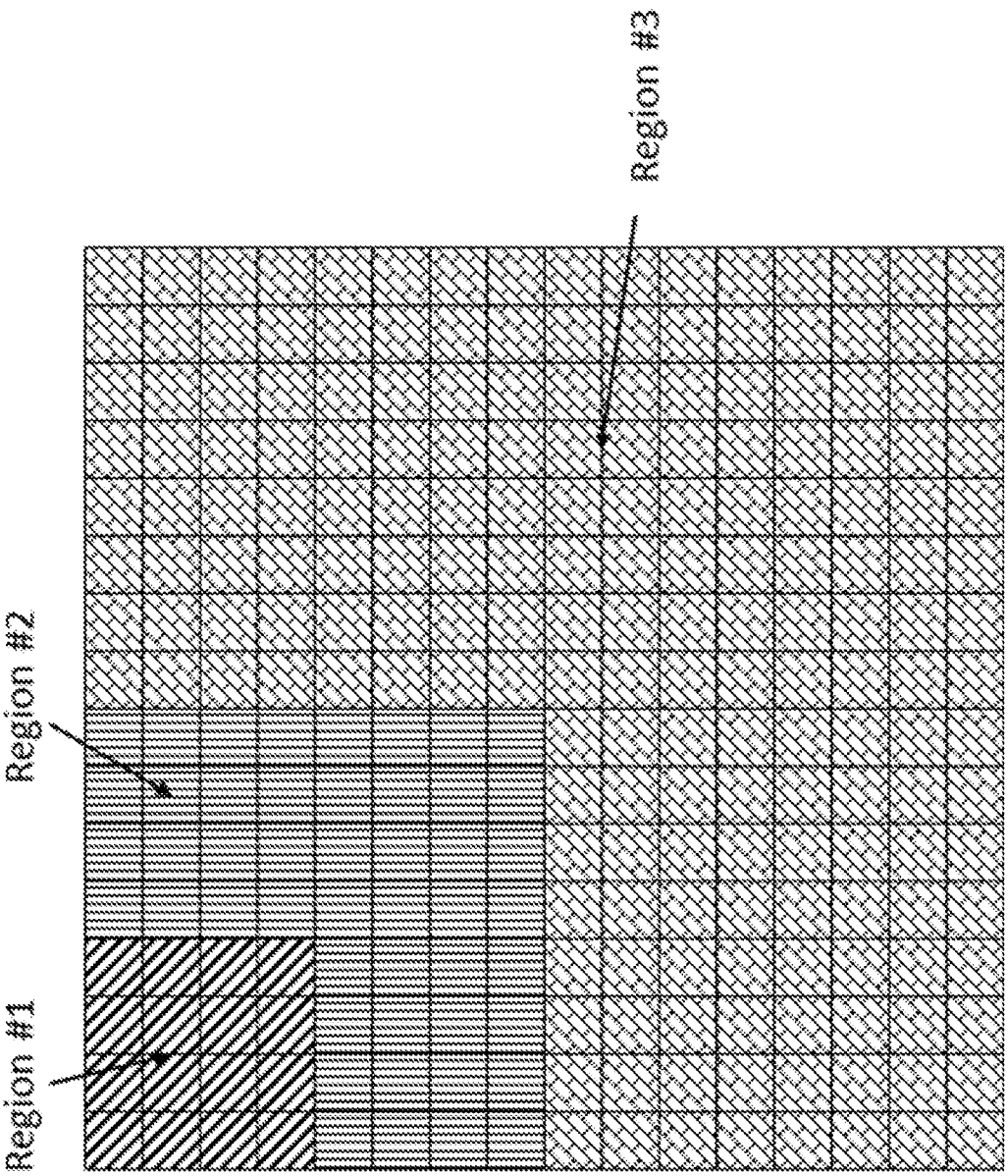


FIG. 20

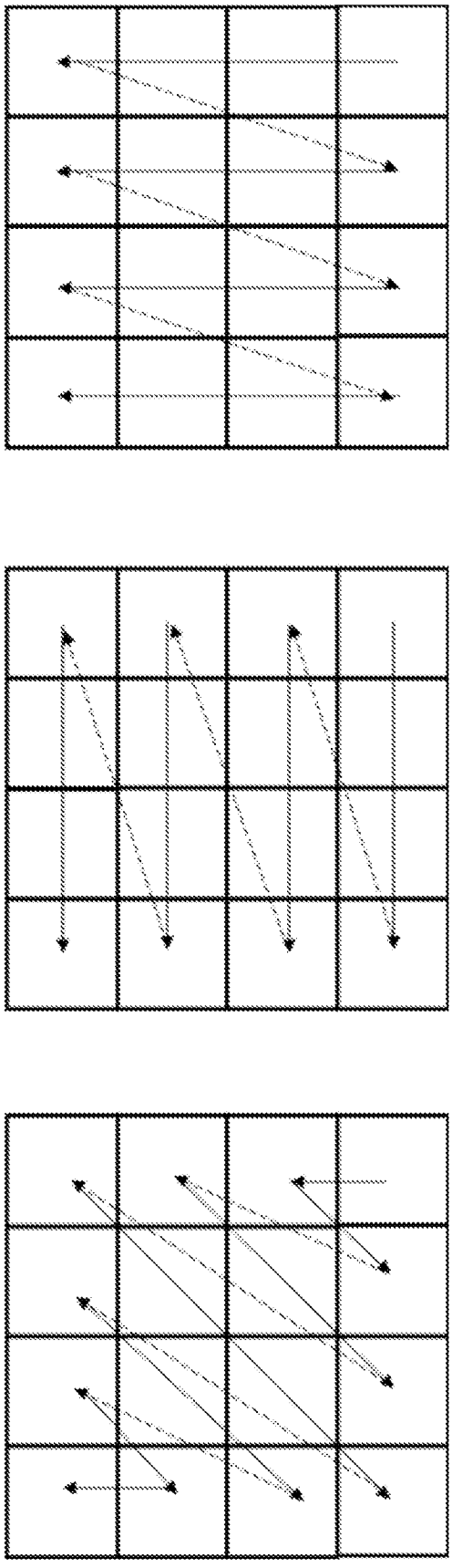


FIG. 21

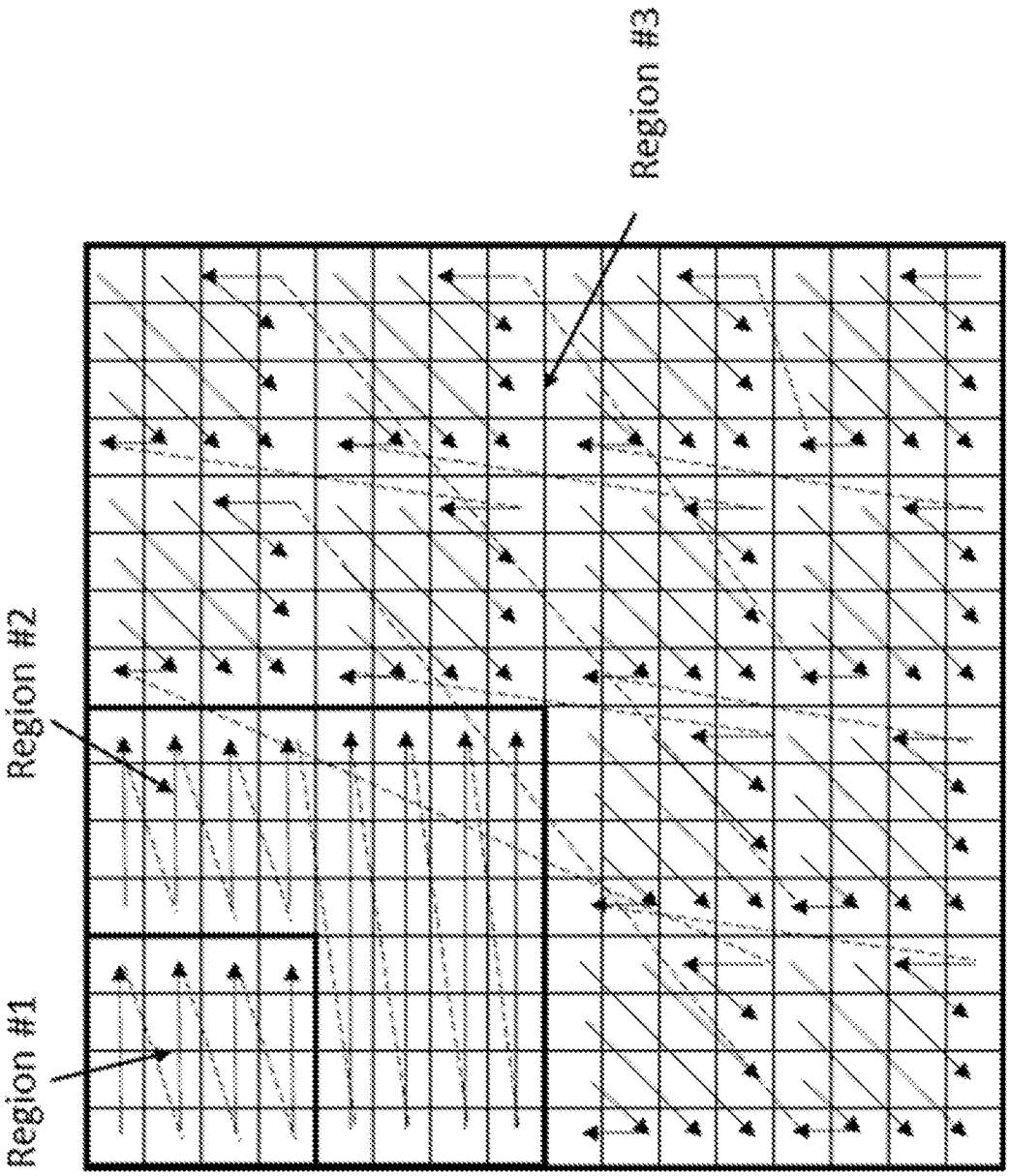


FIG. 22

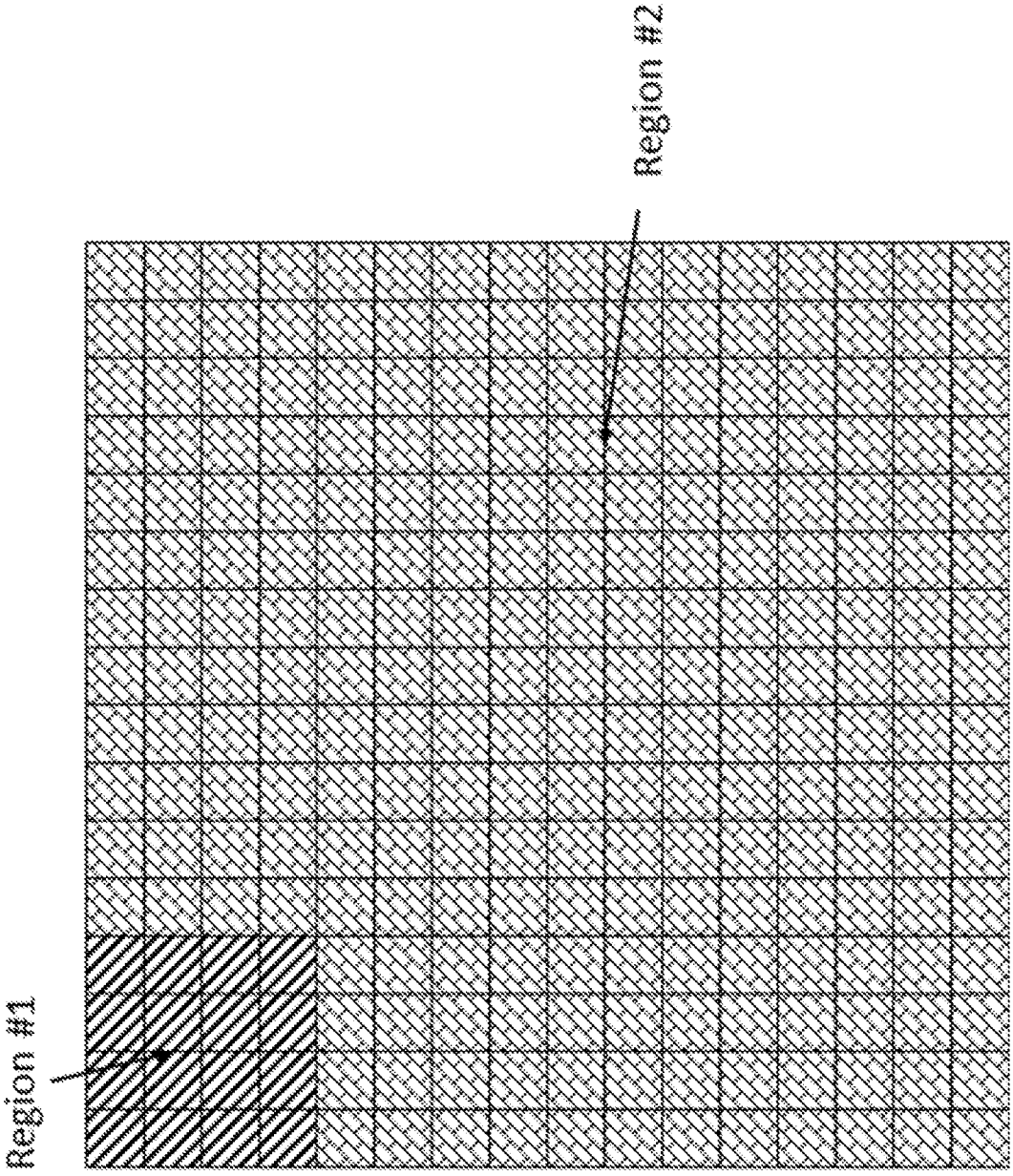


FIG. 23

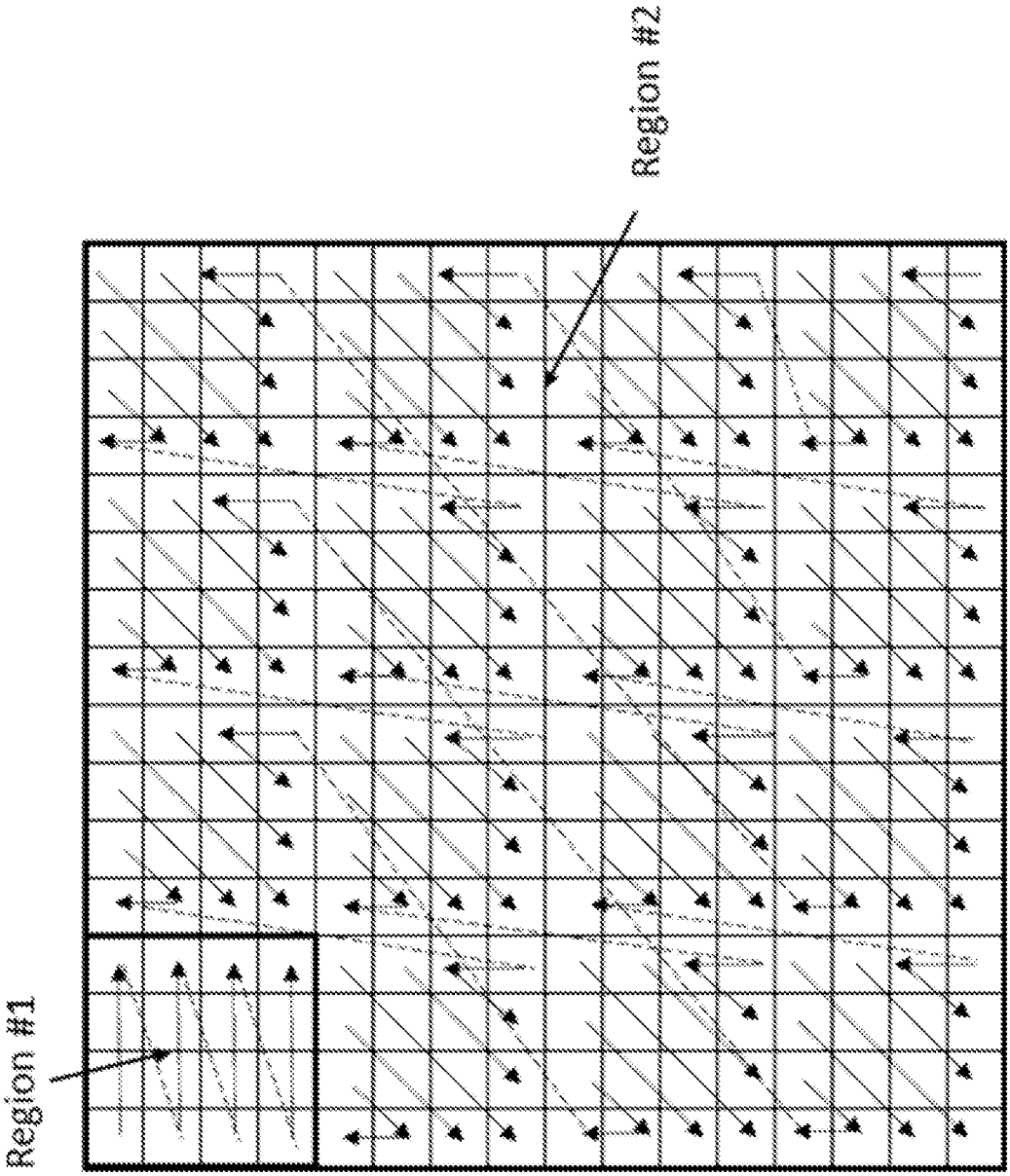


FIG. 24

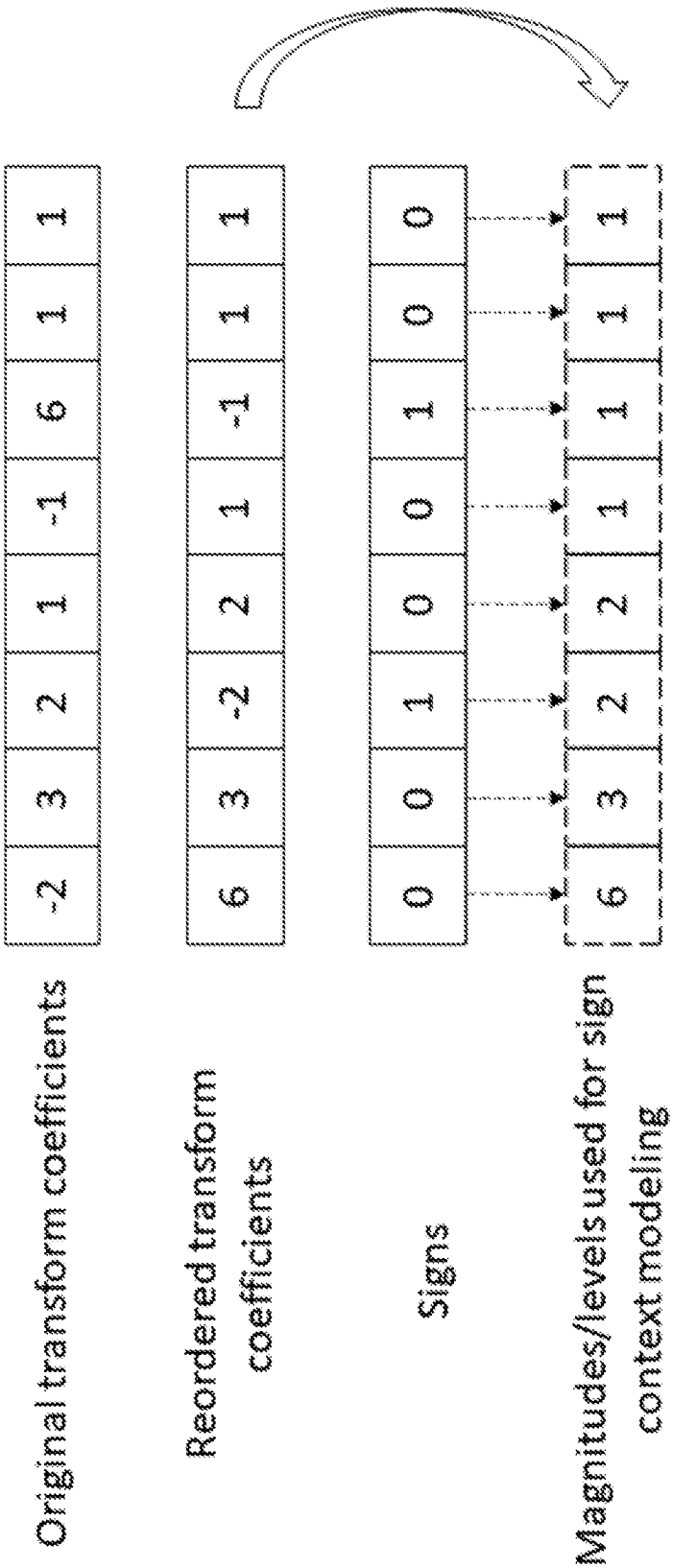

Original transform coefficients
-2 3 2 1 -1 6 1 1
Reordered transform coefficients
6 3 -2 2 1 -1 1 1
Signs
0 0 1 0 0 1 0 0
Magnitudes/levels used for sign context modeling
6 3 2 2 1 1 1 1

FIG. 25A

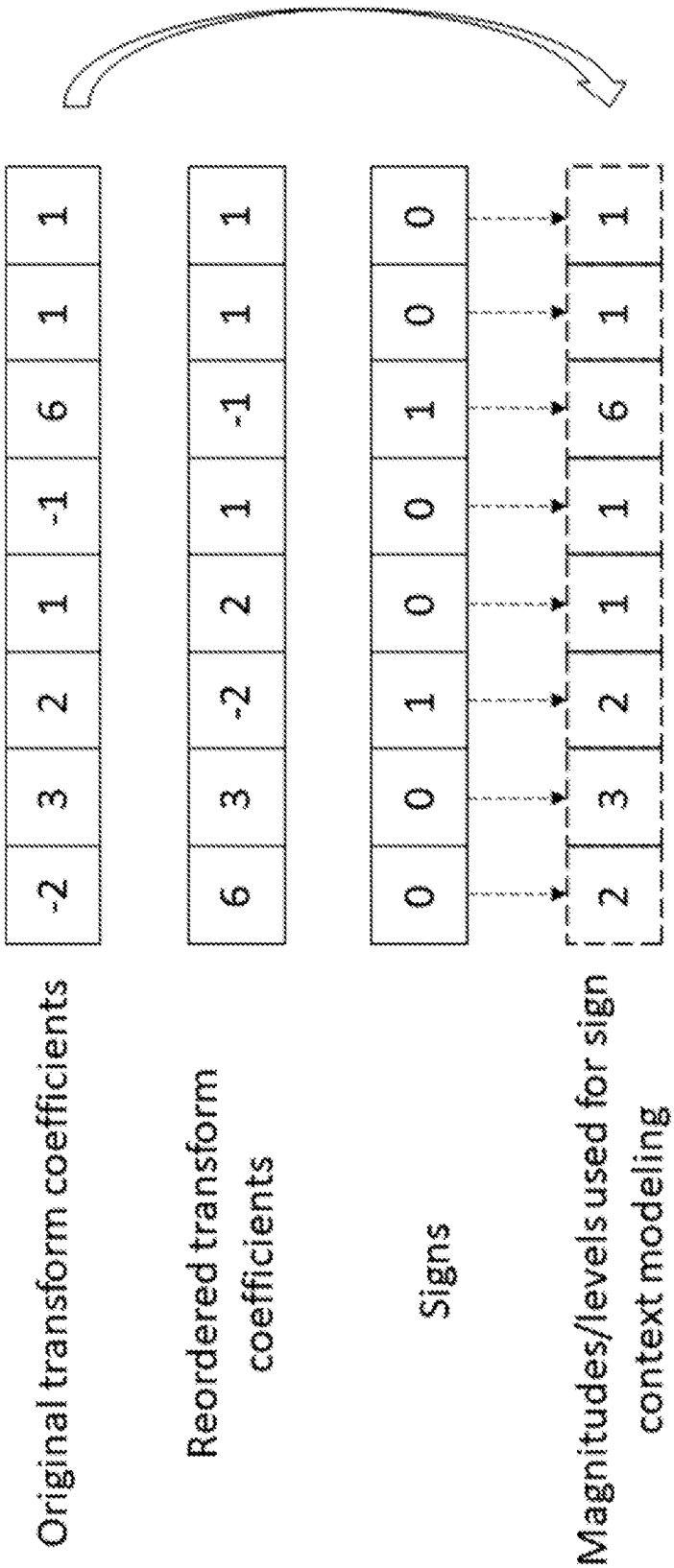

Original transform coefficients
-2 3 2 1 -1 6 1 1
Reordered transform coefficients
6 3 -2 2 1 -1 1 1
Signs
0 0 1 0 0 1 0 0
Magnitudes/levels used for sign context modeling
2 3 2 1 1 6 1 1

FIG. 25B

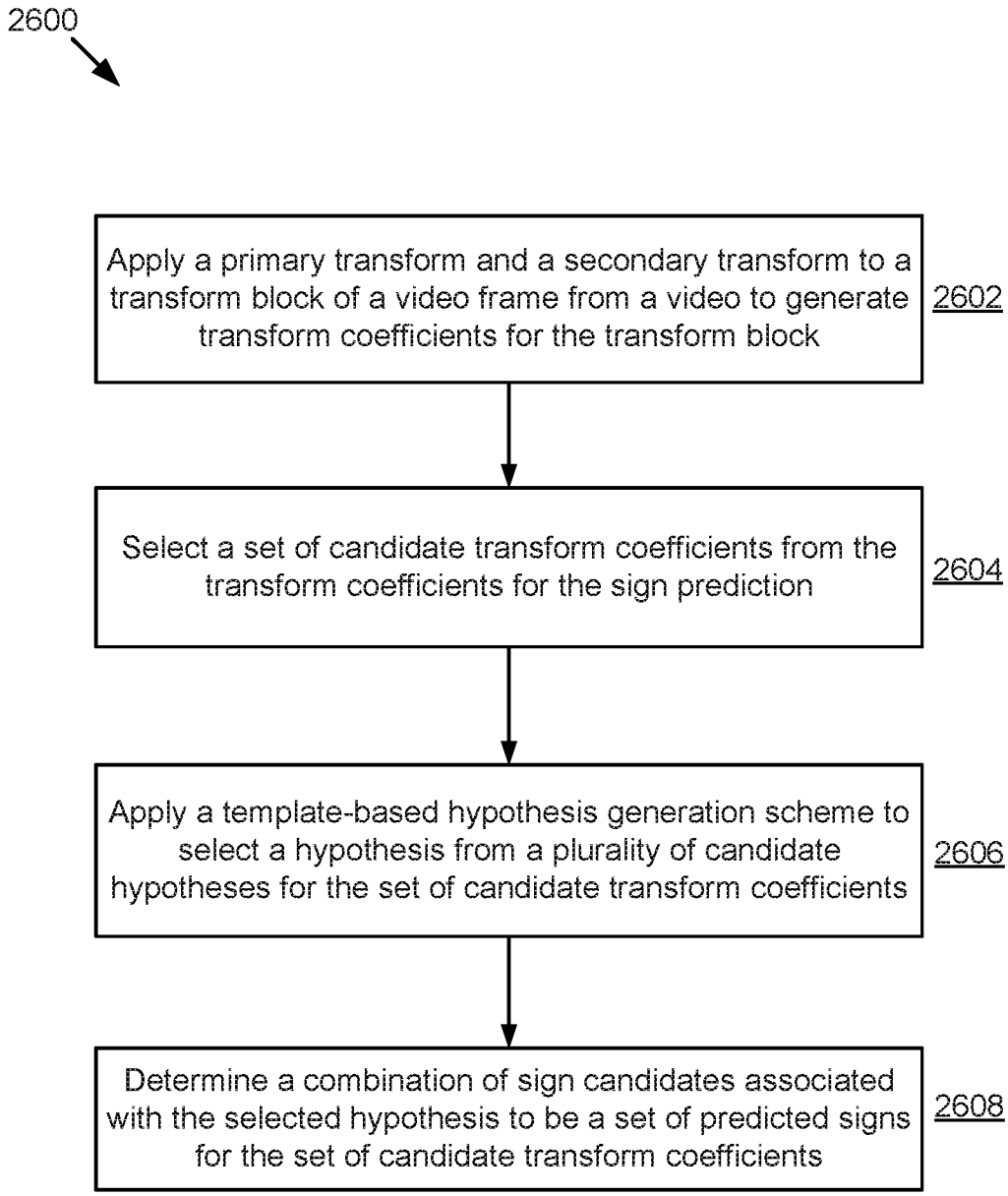

2600
Apply a primary transform and a secondary transform to a transform block of a video frame from a video to generate transform coefficients for the transform block
2602
Select a set of candidate transform coefficients from the transform coefficients for the sign prediction
2604
Apply a template-based hypothesis generation scheme to select a hypothesis from a plurality of candidate hypotheses for the set of candidate transform coefficients
2606
Determine a combination of sign candidates associated with the selected hypothesis to be a set of predicted signs for the set of candidate transform coefficients
2608

FIG. 26

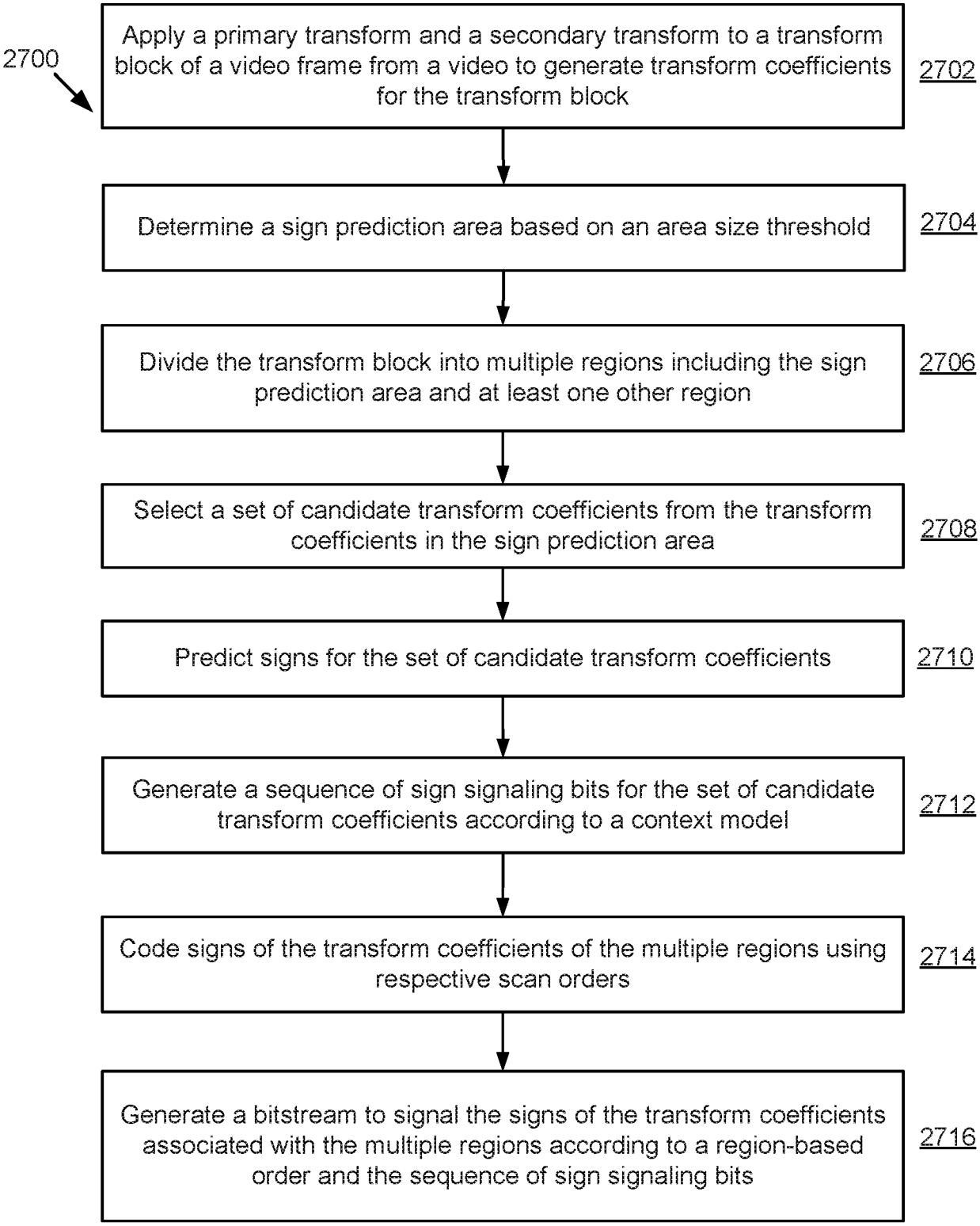

2700
Apply a primary transform and a secondary transform to a transform block of a video frame from a video to generate transform coefficients for the transform block
2702
Determine a sign prediction area based on an area size threshold
2704
Divide the transform block into multiple regions including the sign prediction area and at least one other region
2706
Select a set of candidate transform coefficients from the transform coefficients in the sign prediction area
2708
Predict signs for the set of candidate transform coefficients
2710
Generate a sequence of sign signaling bits for the set of candidate transform coefficients according to a context model
2712
Code signs of the transform coefficients of the multiple regions using respective scan orders
2714
Generate a bitstream to signal the signs of the transform coefficients associated with the multiple regions according to a region-based order and the sequence of sign signaling bits
2716

FIG. 27

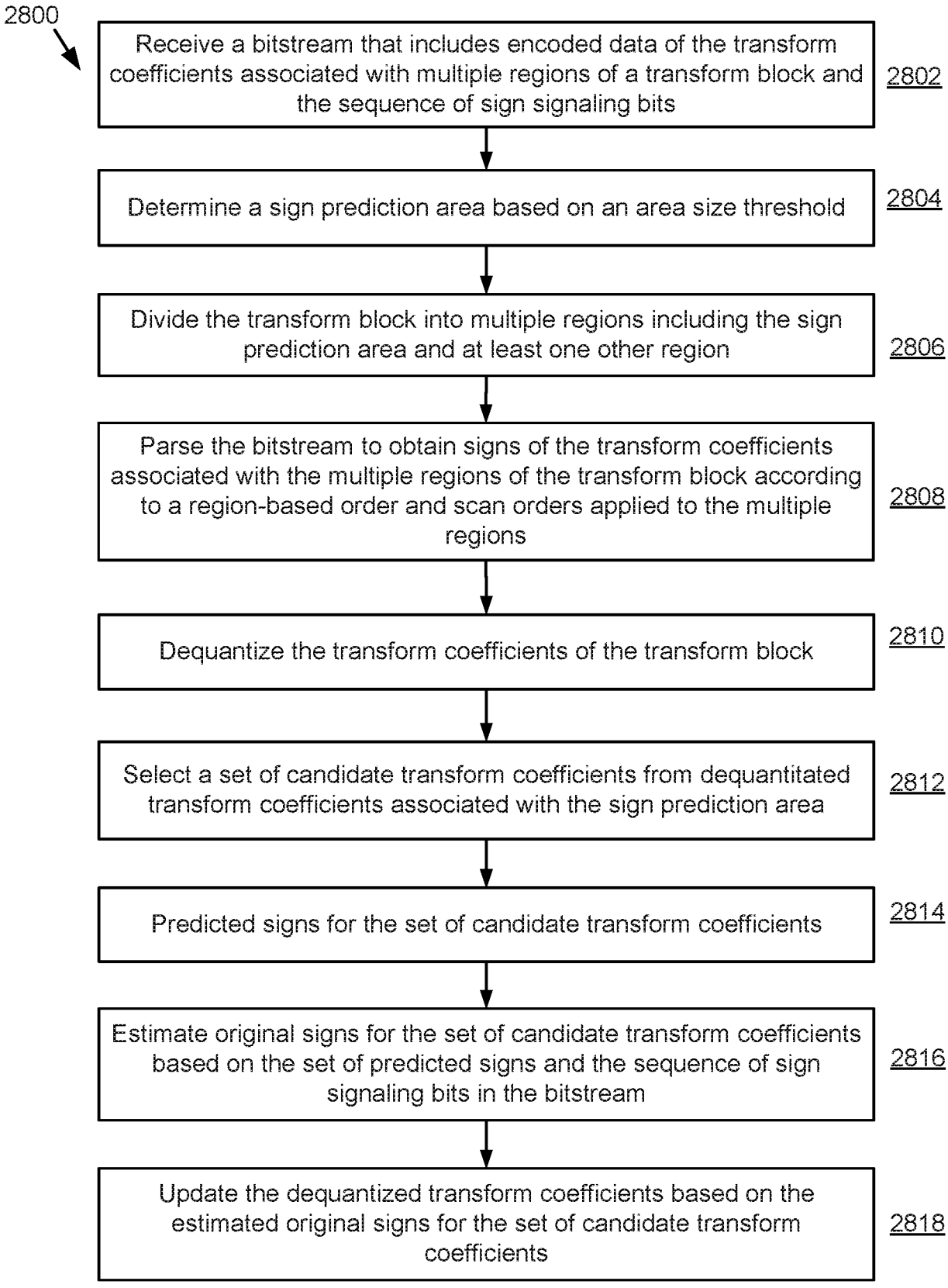

2800
Receive a bitstream that includes encoded data of the transform coefficients associated with multiple regions of a transform block and the sequence of sign signaling bits
2802
Determine a sign prediction area based on an area size threshold
2804
Divide the transform block into multiple regions including the sign prediction area and at least one other region
2806
Parse the bitstream to obtain signs of the transform coefficients associated with the multiple regions of the transform block according to a region-based order and scan orders applied to the multiple regions
2808
Dequantize the transform coefficients of the transform block
2810
Select a set of candidate transform coefficients from dequantitated transform coefficients associated with the sign prediction area
2812
Predicted signs for the set of candidate transform coefficients
2814
Estimate original signs for the set of candidate transform coefficients based on the set of predicted signs and the sequence of sign signaling bits in the bitstream
2816
Update the dequantized transform coefficients based on the estimated original signs for the set of candidate transform coefficients
2818

FIG. 28

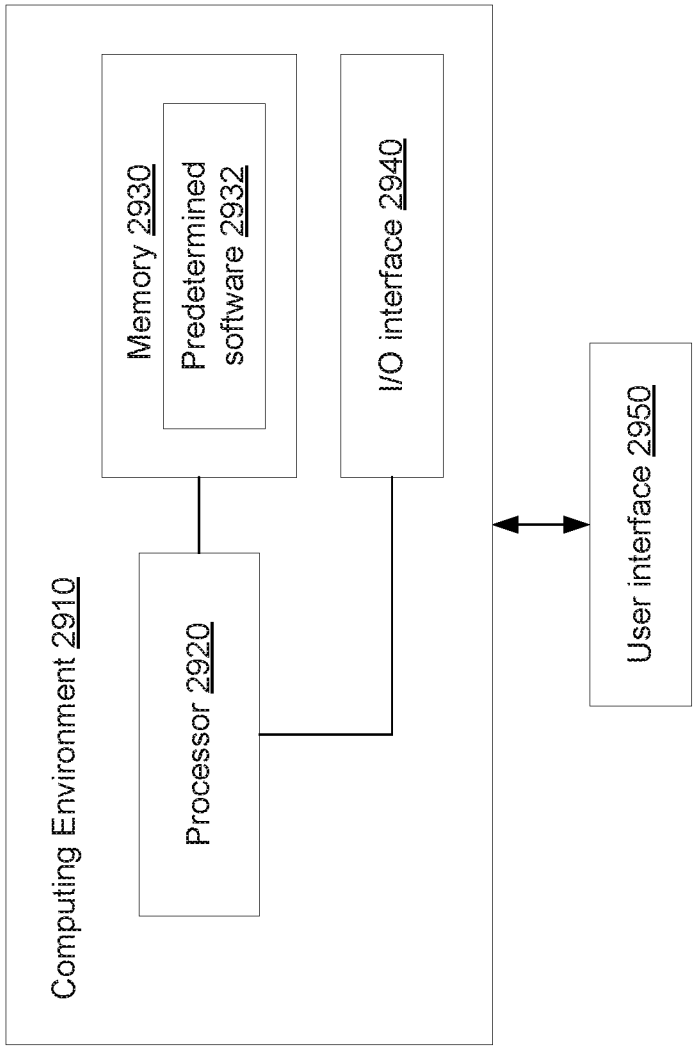

Computing Environment 2910
Processor 2920
Memory 2930
Predetermined software 2932
I/O interface 2940
User interface 2950

FIG. 29

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2023/010901

A. CLASSIFICATION OF SUBJECT MATTER

H04N 19/60(2014.01)i; **H04N 19/18**(2014.01)i; **H04N 19/159**(2014.01)i; **H04N 19/423**(2014.01)i; **H04N 19/176**(2014.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04N 19/60(2014.01); H04N 19/115(2014.01); H04N 19/13(2014.01); H04N 19/176(2014.01); H04N 19/48(2014.01); H04N 19/61(2014.01)

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

Korean utility models and applications for utility models
Japanese utility models and applications for utility models

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

eKOMPASS(KIPO internal) & Keywords: decoding, transform, coefficient, sign, predict, area, size, reconstruct

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2020-0404311 A1 (HUAWEI TECHNOLOGIES CO., LTD.) 24 December 2020 (2020-12-24) paragraphs [0039], [0103]-[0104], [0122], [0146], [0162], [0166], [0168], [0171], [0180], [0190]; claim 16; and figure 2	1-40
Y	JIE CHEN et al., `EE2-4.3 related: More combined test results for sign prediction`, JVET-Y0141-v2, Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29, 25th Meeting, by teleconference, pp. 1-5, 12 January 2022 page 2	1-40
A	US 2021-0067807 A1 (NOKIA TECHNOLOGIES OY) 04 March 2021 (2021-03-04) claims 17-23	1-40
A	US 2019-0208225 A1 (QUALCOMM INCORPORATED) 04 July 2019 (2019-07-04) claims 1-10	1-40
A	US 2017-0142444 A1 (B<>COM et al.) 18 May 2017 (2017-05-18) claims 1-7	1-40

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

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

Date of the actual completion of the international search	Date of mailing of the international search report
10 May 2023	**10 May 2023**

Name and mailing address of the ISA/KR	Authorized officer
Korean Intellectual Property Office **189 Cheongsa-ro, Seo-gu, Daejeon** **35208, Republic of Korea**	**BYUN, SUNG CHEAL**
Facsimile No. **+82-42-481-8578**	Telephone No. **+82-42-481-8262**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2023/010901

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
US 2020-0404311 A1	24 December 2020	CN 111819853 A	23 October 2020
		EP 3741123 A1	25 November 2020
		JP 2021-516016 A	24 June 2021
		JP 7047119 B2	04 April 2022
		KR 10-2020-0112964 A	05 October 2020
		KR 10-2419112 B1	07 July 2022
		US 11438618 B2	06 September 2022
		US 2021-0014509 A1	14 January 2021
		WO 2019-172798 A1	12 September 2019
		WO 2019-172802 A1	12 September 2019
US 2021-0067807 A1	04 March 2021	US 11223849 B2	11 January 2022
		WO 2019-185980 A1	03 October 2019
US 2019-0208225 A1	04 July 2019	WO 2019-135930 A1	11 July 2019
US 2017-0142444 A1	18 May 2017	CN 106797472 A	31 May 2017
		EP 3183879 A1	28 June 2017
		JP 2017-521980 A	03 August 2017
		KR 10-2017-0026536 A	08 March 2017
		WO 2015-197945 A1	30 December 2015