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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 September 2022 (15.09.2022)

WIPO | PCT

(10) International Publication Number
WO 2022/192884 A1

(51) International Patent Classification:
H04N 19/13 (2014.01) *H04N 19/18* (2014.01)
H04N 19/154 (2014.01) *H04N 19/91* (2014.01)

(21) International Application Number:
PCT/US2022/071053

(22) International Filing Date:
09 March 2022 (09.03.2022)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
63/159,892 11 March 2021 (11.03.2021) US
17/653,962 08 March 2022 (08.03.2022) US

(71) Applicant: **QUALCOMM INCORPORATED** [US/US]; ATTN: International IP Administration, 5775 Morehouse Drive, San Diego, California 92121-1714 (US).

(72) Inventors: **RUSANOVSKYY, Dmytro**; 5775 Morehouse Drive, San Diego, California 92121-1714 (US). **KARCZEWICZ, Marta**; 5775 Morehouse Drive, San Diego, California 92121-1714 (US).

(74) Agent: **EVANS, Matthew J.**; Shumaker & Sieffert, P.A., 1625 Radio Drive, Suite 100, Woodbury, Minnesota 55125 (US).

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

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

(54) Title: RICE PARAMETER DERIVATION FOR HIGH BIT-DEPTH VIDEO CODING

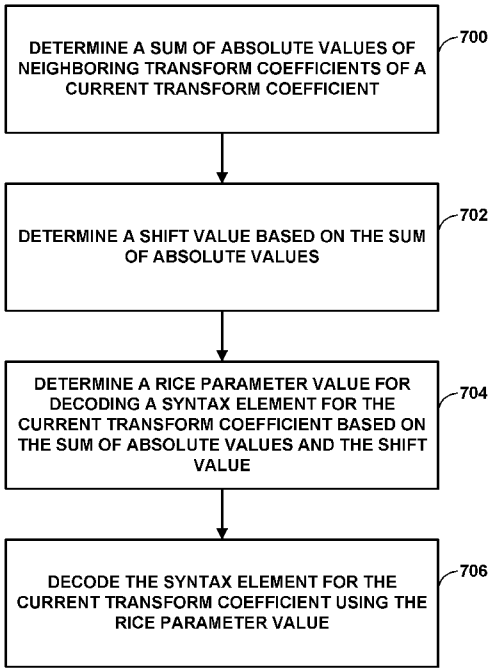


FIG. 11

(57) Abstract: A video coder determines of shift values based on the dynamic range of a sum of absolute values of neighboring transform coefficients. The shift value may then be used to extend possible values of a Rice parameter used to binarize a syntax element related to a currently coded transform coefficient.

WO 2022/192884 A1

[Continued on next page]

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))*

RICE PARAMETER DERIVATION FOR HIGH BIT-DEPTH VIDEO CODING

[0001] This application claims priority to U.S. Patent Application No. 17/653,962, filed March 08, 2022 and U.S. Provisional Application No. 63/159,892, filed March 11, 2021, the entire contents of each of which are incorporated by reference herein. U.S. Patent Application No. 17/653,962, filed March 08, 2022 claims the benefit of U.S. Provisional Application No. 63/159,892, filed March 11, 2021.

TECHNICAL FIELD

[0002] This disclosure relates to video encoding and video decoding.

BACKGROUND

[0003] Digital video capabilities can be incorporated into a wide range of devices, including digital televisions, digital direct broadcast systems, wireless broadcast systems, personal digital assistants (PDAs), laptop or desktop computers, tablet computers, e-book readers, digital cameras, digital recording devices, digital media players, video gaming devices, video game consoles, cellular or satellite radio telephones, so-called "smart phones," video teleconferencing devices, video streaming devices, and the like. Digital video devices implement video coding techniques, such as those described in the standards defined by MPEG-2, MPEG-4, ITU-T H.263, ITU-T H.264/MPEG-4, Part 10, Advanced Video Coding (AVC), ITU-T H.265/High Efficiency Video Coding (HEVC), and extensions of such standards. The video devices may transmit, receive, encode, decode, and/or store digital video information more efficiently by implementing such video coding techniques.

[0004] Video coding techniques include spatial (intra-picture) prediction and/or temporal (inter-picture) prediction to reduce or remove redundancy inherent in video sequences. For block-based video coding, a video slice (e.g., a video picture or a portion of a video picture) may be partitioned into video blocks, which may also be referred to as coding tree units (CTUs), coding units (CUs) and/or coding nodes. Video blocks in an intra-coded (I) slice of a picture are encoded using spatial prediction with respect to reference samples in neighboring blocks in the same picture. Video blocks in an inter-coded (P or B) slice of a picture may use spatial prediction with respect to reference samples in neighboring blocks in the same picture or temporal prediction with

respect to reference samples in other reference pictures. Pictures may be referred to as frames, and reference pictures may be referred to as reference frames.

SUMMARY

[0005] In general, this disclosure describes techniques for adaptively deriving a Rice parameter for regular residual coding (RRC) of a syntax element related to a transform coefficient in high bit-depth coding. In particular, this disclosure describes techniques for extending the range of possible Rice parameter values based on the dynamic range of a sum of absolute values of transform coefficients in a template around a currently coded transform coefficient.

[0006] In high-bit depth video coding, the values of some transform coefficients can become quite large, thus resulting in syntax elements with larger codewords. In general, large codewords reduce coding efficiency. The size of such codewords may be reduced by determining larger Rice parameter values in some circumstances. The techniques of this disclosure include the determination of shift values based on the dynamic range of a sum of absolute values of neighboring transform coefficients. The shift value may then be used to extend possible values of the Rice parameter. As such, smaller codeword values may be achieved for some syntax elements, thus improving coding efficiency. The techniques of this disclosure may be used with extensions of HEVC, VVC, or other video coding techniques.

[0007] In one example, this disclosure describes a method of decoding video data, the method comprising determining a sum of absolute values of neighboring transform coefficients of a current transform coefficient, determining a shift value based on the sum of absolute values, determining a Rice parameter value for decoding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value, and decoding the syntax element for the current transform coefficient using the Rice parameter value.

[0008] In another example, this disclosure describes an apparatus configured to decode video data, the apparatus comprising a memory configured to store video data, and one or more processors implemented in circuitry and in communication with the memory, the one or more processors configured to determine a sum of absolute values of neighboring transform coefficients of a current transform coefficient, determine a shift value based on the sum of absolute values, determine a Rice parameter value for

decoding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value, and decode the syntax element for the current transform coefficient using the Rice parameter value.

[0009] In another example, this disclosure describes an apparatus configured to decode video data, the apparatus comprising means for determining a sum of absolute values of neighboring transform coefficients of a current transform coefficient, means for determining a shift value based on the sum of absolute values, means for determining a Rice parameter value for decoding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value, and means for decoding the syntax element for the current transform coefficient using the Rice parameter value.

[0010] In another example, this disclosure describes a non-transitory computer-readable storage medium storing instructions that, when executed, cause one or more processors configured to decode video data to determine a sum of absolute values of neighboring transform coefficients of a current transform coefficient, determine a shift value based on the sum of absolute values, determine a Rice parameter value for decoding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value, and decode the syntax element for the current transform coefficient using the Rice parameter value.

[0011] In another example, this disclosure describes a method of encoding video data, the method comprising determining a sum of absolute values of neighboring transform coefficients of a current transform coefficient, determining a shift value based on the sum of absolute values, determining a Rice parameter value for decoding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value, and encoding the syntax element for the current transform coefficient using the Rice parameter value.

[0012] The details of one or more examples are set forth in the accompanying drawings and the description below. Other features, objects, and advantages will be apparent from the description, drawings, and claims.

BRIEF DESCRIPTION OF DRAWINGS

[0013] FIG. 1 is a block diagram illustrating an example video encoding and decoding system that may perform the techniques of this disclosure.

[0014] FIG. 2 is a conceptual diagram illustrating one example of neighboring coefficients that are used to derive a Rice parameter.

[0015] FIG. 3 illustrates one example of lookup table used to derive a Rice parameter.

[0016] FIG. 4 illustrates an example histogram of coded coefficients.

[0017] FIG. 5 illustrates example codeword lengths for different Rice parameters.

[0018] FIG. 6 is a block diagram illustrating an example video encoder that may perform the techniques of this disclosure.

[0019] FIG. 7 is a block diagram illustrating an example video decoder that may perform the techniques of this disclosure.

[0020] FIG. 8 is a flowchart illustrating an example method for encoding a current block in accordance with the techniques of this disclosure.

[0021] FIG. 9 is a flowchart illustrating an example method for decoding a current block in accordance with the techniques of this disclosure.

[0022] FIG. 10 is another flowchart illustrating an example method for encoding a current block in accordance with the techniques of this disclosure.

[0023] FIG. 11 is another flowchart illustrating an example method for decoding a current block in accordance with the techniques of this disclosure.

DETAILED DESCRIPTION

[0024] Regular residual coding (RRC) is an example technique for encoding and decoding a residual block. RRC is a term that may distinguish residual coding techniques that include transforms from other residual coding techniques, such as transform skip (TS) residual coding. From the context of a video decoder, RRC involves decoding a plurality of syntax elements that are used to determine the magnitudes and polarities (e.g., positive or negative) of transform coefficients in a block. Once the transform coefficient values are determined, the video decoder may apply one or more transforms to the transform coefficients to reconstruct the residual block. Example syntax elements that may define the position and values of transform coefficients may include one or more last significant coefficient position syntax elements, a coded block flag, a significant coefficient flag, one or more absolute level greater than X flags, a parity flag, a remainder syntax element, and an absolute value syntax element.

[0025] In some examples of RRC, or other residual coding techniques, certain syntax elements may be binarized according to a binarization process. For example, a remainder syntax element and/or an absolute value syntax element may be binarized according to a truncated Rice binarization process. The truncated Rice binarization process is performed based on a Rice parameter. A video decoder may be configured to determine a Rice parameter in order to perform an inverse binarization process to determine the values of such syntax elements. In some examples, a video decoder may be configured to derive a Rice parameter for RRC using a look-up table and the coefficient values of neighboring transform coefficients in a template.

[0026] For example, a video decoder may be configured to calculate a value of *locSumAbs*, which is the sum of absolute values of five available neighboring coefficients in a template relative to a current transform coefficient. The video decoder may normalize the value of *locSumAbs* (e.g., using subtraction and clipping). The video decoder may derive the Rice parameter using the normalized value of *locSumAbs* as an input to a look-up table.

[0027] In VVC, the Rice parameter has been tested for encoding and decoding video sources of 8-bit bit-depth or 10-bit bit-depth. In the VVC design, the Rice parameter value is dependent on the value of *locSumAbs* and is limited to be in the range from 0 to 3, inclusive, through the clipping process described above. When the bit-depth of the input video is increased, extended precision is enabled, and/or the quantization parameter (QP) is set very low, the range of values of transform coefficients, and therfore *locSumAbs* value, may significantly increase. In such cases, the range of allowed Rice parameter values in VVC may not be sufficient for optimal coding efficiency, since the restriction of Rice parameter values to be from 0 to 3 would require utilization of long codewords, e.g., codewords of length 32, for a coded coefficients having large values.

[0028] This disclosure describes techniques that address the limitations of Rice parameter derivation techniques in VVC, and other video coding processes, for various input bit-depths of video data. The techniques of this disclosure may improve compression efficiency of video coding designs, particular video coding deisgns using bit depths over 10 bits. With the proposed techniques of this disclsoure, the supported range of Rice paramters can be extended from 0 to 3 in VVC up to a larger number, e.g., 16, which would provide for a more efficient binarization process. The more efficient binarization process may result in a lower number of bits used for large values of

transform coefficients. In particular, the techniqeus of this disclsoure may result in smaller codeword lengths for remainder and/or absolute value syntax elements for transform coefficients having larger absolute values.

[0029] FIG. 1 is a block diagram illustrating an example video encoding and decoding system 100 that may perform the techniques of this disclosure. The techniques of this disclosure are generally directed to coding (encoding and/or decoding) video data. In general, video data includes any data for processing a video. Thus, video data may include raw, unencoded video, encoded video, decoded (e.g., reconstructed) video, and video metadata, such as signaling data.

[0030] As shown in FIG. 1, system 100 includes a source device 102 that provides encoded video data to be decoded and displayed by a destination device 116, in this example. In particular, source device 102 provides the video data to destination device 116 via a computer-readable medium 110. Source device 102 and destination device 116 may comprise any of a wide range of devices, including desktop computers, notebook (i.e., laptop) computers, mobile devices, tablet computers, set-top boxes, telephone handsets such as smartphones, televisions, cameras, display devices, digital media players, video gaming consoles, video streaming device, broadcast receiver devices, or the like. In some cases, source device 102 and destination device 116 may be equipped for wireless communication, and thus may be referred to as wireless communication devices.

[0031] In the example of FIG. 1, source device 102 includes video source 104, memory 106, video encoder 200, and output interface 108. Destination device 116 includes input interface 122, video decoder 300, memory 120, and display device 118. In accordance with this disclosure, video encoder 200 of source device 102 and video decoder 300 of destination device 116 may be configured to apply the techniques for Rice parameter derivation. Thus, source device 102 represents an example of a video encoding device, while destination device 116 represents an example of a video decoding device. In other examples, a source device and a destination device may include other components or arrangements. For example, source device 102 may receive video data from an external video source, such as an external camera. Likewise, destination device 116 may interface with an external display device, rather than include an integrated display device.

[0032] System 100 as shown in FIG. 1 is merely one example. In general, any digital video encoding and/or decoding device may perform techniques for Rice parameter

derivation. Source device 102 and destination device 116 are merely examples of such coding devices in which source device 102 generates coded video data for transmission to destination device 116. This disclosure refers to a "coding" device as a device that performs coding (encoding and/or decoding) of data. Thus, video encoder 200 and video decoder 300 represent examples of coding devices, in particular, a video encoder and a video decoder, respectively. In some examples, source device 102 and destination device 116 may operate in a substantially symmetrical manner such that each of source device 102 and destination device 116 includes video encoding and decoding components. Hence, system 100 may support one-way or two-way video transmission between source device 102 and destination device 116, e.g., for video streaming, video playback, video broadcasting, or video telephony.

[0033] In general, video source 104 represents a source of video data (i.e., raw, unencoded video data) and provides a sequential series of pictures (also referred to as "frames") of the video data to video encoder 200, which encodes data for the pictures. Video source 104 of source device 102 may include a video capture device, such as a video camera, a video archive containing previously captured raw video, and/or a video feed interface to receive video from a video content provider. As a further alternative, video source 104 may generate computer graphics-based data as the source video, or a combination of live video, archived video, and computer-generated video. In each case, video encoder 200 encodes the captured, pre-captured, or computer-generated video data. Video encoder 200 may rearrange the pictures from the received order (sometimes referred to as "display order") into a coding order for coding. Video encoder 200 may generate a bitstream including encoded video data. Source device 102 may then output the encoded video data via output interface 108 onto computer-readable medium 110 for reception and/or retrieval by, e.g., input interface 122 of destination device 116.

[0034] Memory 106 of source device 102 and memory 120 of destination device 116 represent general purpose memories. In some examples, memories 106, 120 may store raw video data, e.g., raw video from video source 104 and raw, decoded video data from video decoder 300. Additionally or alternatively, memories 106, 120 may store software instructions executable by, e.g., video encoder 200 and video decoder 300, respectively. Although memory 106 and memory 120 are shown separately from video encoder 200 and video decoder 300 in this example, it should be understood that video encoder 200 and video decoder 300 may also include internal memories for functionally similar or equivalent purposes. Furthermore, memories 106, 120 may store encoded video data,

e.g., output from video encoder 200 and input to video decoder 300. In some examples, portions of memories 106, 120 may be allocated as one or more video buffers, e.g., to store raw, decoded, and/or encoded video data.

[0035] Computer-readable medium 110 may represent any type of medium or device capable of transporting the encoded video data from source device 102 to destination device 116. In one example, computer-readable medium 110 represents a communication medium to enable source device 102 to transmit encoded video data directly to destination device 116 in real-time, e.g., via a radio frequency network or computer-based network. Output interface 108 may modulate a transmission signal including the encoded video data, and input interface 122 may demodulate the received transmission signal, according to a communication standard, such as a wireless communication protocol. The communication medium may comprise 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 source device 102 to destination device 116.

[0036] In some examples, source device 102 may output encoded data from output interface 108 to storage device 112. Similarly, destination device 116 may access encoded data from storage device 112 via input interface 122. Storage device 112 may include any of a variety of distributed or locally accessed data storage media such as a hard drive, Blu-ray discs, DVDs, CD-ROMs, flash memory, volatile or non-volatile memory, or any other suitable digital storage media for storing encoded video data.

[0037] In some examples, source device 102 may output encoded video data to file server 114 or another intermediate storage device that may store the encoded video data generated by source device 102. Destination device 116 may access stored video data from file server 114 via streaming or download.

[0038] File server 114 may be any type of server device capable of storing encoded video data and transmitting that encoded video data to the destination device 116. File server 114 may represent a web server (e.g., for a website), a server configured to provide a file transfer protocol service (such as File Transfer Protocol (FTP) or File Delivery over Unidirectional Transport (FLUTE) protocol), a content delivery network (CDN) device, a hypertext transfer protocol (HTTP) server, a Multimedia Broadcast

Multicast Service (MBMS) or Enhanced MBMS (eMBMS) server, and/or a network attached storage (NAS) device. File server 114 may, additionally or alternatively, implement one or more HTTP streaming protocols, such as Dynamic Adaptive Streaming over HTTP (DASH), HTTP Live Streaming (HLS), Real Time Streaming Protocol (RTSP), HTTP Dynamic Streaming, or the like.

[0039] Destination device 116 may access encoded video data from file server 114 through any standard data connection, including an Internet connection. This may include a wireless channel (e.g., a Wi-Fi connection), a wired connection (e.g., digital subscriber line (DSL), cable modem, etc.), or a combination of both that is suitable for accessing encoded video data stored on file server 114. Input interface 122 may be configured to operate according to any one or more of the various protocols discussed above for retrieving or receiving media data from file server 114, or other such protocols for retrieving media data.

[0040] Output interface 108 and input interface 122 may represent wireless transmitters/receivers, modems, wired networking components (e.g., Ethernet cards), wireless communication components that operate according to any of a variety of IEEE 802.11 standards, or other physical components. In examples where output interface 108 and input interface 122 comprise wireless components, output interface 108 and input interface 122 may be configured to transfer data, such as encoded video data, according to a cellular communication standard, such as 4G, 4G-LTE (Long-Term Evolution), LTE Advanced, 5G, or the like. In some examples where output interface 108 comprises a wireless transmitter, output interface 108 and input interface 122 may be configured to transfer data, such as encoded video data, according to other wireless standards, such as an IEEE 802.11 specification, an IEEE 802.15 specification (e.g., ZigBee™), a Bluetooth™ standard, or the like. In some examples, source device 102 and/or destination device 116 may include respective system-on-a-chip (SoC) devices. For example, source device 102 may include an SoC device to perform the functionality attributed to video encoder 200 and/or output interface 108, and destination device 116 may include an SoC device to perform the functionality attributed to video decoder 300 and/or input interface 122.

[0041] The techniques of this disclosure may be applied to video coding in support of any of a variety of multimedia applications, such as over-the-air television broadcasts, cable television transmissions, satellite television transmissions, Internet streaming video transmissions, such as dynamic adaptive streaming over HTTP (DASH), digital

video that is encoded onto a data storage medium, decoding of digital video stored on a data storage medium, or other applications.

[0042] Input interface 122 of destination device 116 receives an encoded video bitstream from computer-readable medium 110 (e.g., a communication medium, storage device 112, file server 114, or the like). The encoded video bitstream may include signaling information defined by video encoder 200, which is also used by video decoder 300, such as syntax elements having values that describe characteristics and/or processing of video blocks or other coded units (e.g., slices, pictures, groups of pictures, sequences, or the like). Display device 118 displays decoded pictures of the decoded video data to a user. Display device 118 may represent 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.

[0043] Although not shown in FIG. 1, in some examples, video encoder 200 and video decoder 300 may each be integrated with an audio encoder and/or audio decoder, and may include appropriate MUX-DEMUX units, or other hardware and/or software, to handle multiplexed streams including both audio and video in a common data stream. If applicable, MUX-DEMUX units may conform to the ITU H.223 multiplexer protocol, or other protocols such as the user datagram protocol (UDP).

[0044] Video encoder 200 and video decoder 300 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 the techniques are implemented partially in software, a 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 techniques of this disclosure. Each of video encoder 200 and video decoder 300 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. A device including video encoder 200 and/or video decoder 300 may comprise an integrated circuit, a microprocessor, and/or a wireless communication device, such as a cellular telephone.

[0045] Video encoder 200 and video decoder 300 may operate according to a video coding standard, such as ITU-T H.265, also referred to as High Efficiency Video Coding (HEVC) or extensions thereto, such as the multi-view and/or scalable video

coding extensions. Alternatively, video encoder 200 and video decoder 300 may operate according to other proprietary or industry standards, such as ITU-T H.266, also referred to as Versatile Video Coding (VVC). A draft of the VVC standard is described in Bross, et al. "Versatile Video Coding (Draft 10)," Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 18th Meeting: by teleconference, 22 June – 1 July 2020, JVET-S2001-vA (hereinafter "VVC Draft 10"). The techniques of this disclosure, however, are not limited to any particular coding standard.

[0046] In general, video encoder 200 and video decoder 300 may perform block-based coding of pictures. The term "block" generally refers to a structure including data to be processed (e.g., encoded, decoded, or otherwise used in the encoding and/or decoding process). For example, a block may include a two-dimensional matrix of samples of luminance and/or chrominance data. In general, video encoder 200 and video decoder 300 may code video data represented in a YUV (e.g., Y, Cb, Cr) format. That is, rather than coding red, green, and blue (RGB) data for samples of a picture, video encoder 200 and video decoder 300 may code luminance and chrominance components, where the chrominance components may include both red hue and blue hue chrominance components. In some examples, video encoder 200 converts received RGB formatted data to a YUV representation prior to encoding, and video decoder 300 converts the YUV representation to the RGB format. Alternatively, pre- and post-processing units (not shown) may perform these conversions.

[0047] This disclosure may generally refer to coding (e.g., encoding and decoding) of pictures to include the process of encoding or decoding data of the picture. Similarly, this disclosure may refer to coding of blocks of a picture to include the process of encoding or decoding data for the blocks, e.g., prediction and/or residual coding. An encoded video bitstream generally includes a series of values for syntax elements representative of coding decisions (e.g., coding modes) and partitioning of pictures into blocks. Thus, references to coding a picture or a block should generally be understood as coding values for syntax elements forming the picture or block.

[0048] HEVC defines various blocks, including coding units (CUs), prediction units (PUs), and transform units (TUs). According to HEVC, a video coder (such as video encoder 200) partitions a coding tree unit (CTU) into CUs according to a quadtree structure. That is, the video coder partitions CTUs and CUs into four equal, non-overlapping squares, and each node of the quadtree has either zero or four child nodes.

Nodes without child nodes may be referred to as "leaf nodes," and CUs of such leaf nodes may include one or more PUs and/or one or more TUs. The video coder may further partition PUs and TUs. For example, in HEVC, a residual quadtree (RQT) represents partitioning of TUs. In HEVC, PUs represent inter-prediction data, while TUs represent residual data. CUs that are intra-predicted include intra-prediction information, such as an intra-mode indication.

[0049] As another example, video encoder 200 and video decoder 300 may be configured to operate according to VVC. According to VVC, a video coder (such as video encoder 200) partitions a picture into a plurality of coding tree units (CTUs). Video encoder 200 may partition a CTU according to a tree structure, such as a quadtree-binary tree (QTBT) structure or Multi-Type Tree (MTT) structure. The QTBT structure removes the concepts of multiple partition types, such as the separation between CUs, PUs, and TUs of HEVC. A QTBT structure includes two levels: a first level partitioned according to quadtree partitioning, and a second level partitioned according to binary tree partitioning. A root node of the QTBT structure corresponds to a CTU. Leaf nodes of the binary trees correspond to coding units (CUs).

[0050] In an MTT partitioning structure, blocks may be partitioned using a quadtree (QT) partition, a binary tree (BT) partition, and one or more types of triple tree (TT) (also called ternary tree (TT)) partitions. A triple or ternary tree partition is a partition where a block is split into three sub-blocks. In some examples, a triple or ternary tree partition divides a block into three sub-blocks without dividing the original block through the center. The partitioning types in MTT (e.g., QT, BT, and TT), may be symmetrical or asymmetrical.

[0051] In some examples, video encoder 200 and video decoder 300 may use a single QTBT or MTT structure to represent each of the luminance and chrominance components, while in other examples, video encoder 200 and video decoder 300 may use two or more QTBT or MTT structures, such as one QTBT/MTT structure for the luminance component and another QTBT/MTT structure for both chrominance components (or two QTBT/MTT structures for respective chrominance components).

[0052] Video encoder 200 and video decoder 300 may be configured to use quadtree partitioning per HEVC, QTBT partitioning, MTT partitioning, or other partitioning structures. For purposes of explanation, the description of the techniques of this disclosure is presented with respect to QTBT partitioning. However, it should be

understood that the techniques of this disclosure may also be applied to video coders configured to use quadtree partitioning, or other types of partitioning as well.

[0053] In some examples, a CTU includes a coding tree block (CTB) of luma samples, two corresponding CTBs of chroma samples of a picture that has three sample arrays, or a CTB of samples of a monochrome picture or a picture that is coded using three separate color planes and syntax structures used to code the samples. A CTB may be an NxN block of samples for some value of N such that the division of a component into CTBs is a partitioning. A component is an array or single sample from one of the three arrays (luma and two chroma) that compose a picture in 4:2:0, 4:2:2, or 4:4:4 color format or the array or a single sample of the array that compose a picture in monochrome format. In some examples, a coding block is an MxN block of samples for some values of M and N such that a division of a CTB into coding blocks is a partitioning.

[0054] The blocks (e.g., CTUs or CUs) may be grouped in various ways in a picture. As one example, a brick may refer to a rectangular region of CTU rows within a particular tile in a picture. A tile may be a rectangular region of CTUs within a particular tile column and a particular tile row in a picture. A tile column refers to a rectangular region of CTUs having a height equal to the height of the picture and a width specified by syntax elements (e.g., such as in a picture parameter set). A tile row refers to a rectangular region of CTUs having a height specified by syntax elements (e.g., such as in a picture parameter set) and a width equal to the width of the picture.

[0055] In some examples, a tile may be partitioned into multiple bricks, each of which may include one or more CTU rows within the tile. A tile that is not partitioned into multiple bricks may also be referred to as a brick. However, a brick that is a true subset of a tile may not be referred to as a tile.

[0056] The bricks in a picture may also be arranged in a slice. A slice may be an integer number of bricks of a picture that may be exclusively contained in a single network abstraction layer (NAL) unit. In some examples, a slice includes either a number of complete tiles or only a consecutive sequence of complete bricks of one tile.

[0057] This disclosure may use "NxN" and "N by N" interchangeably to refer to the sample dimensions of a block (such as a CU or other video block) in terms of vertical and horizontal dimensions, e.g., 16x16 samples or 16 by 16 samples. In general, a 16x16 CU will have 16 samples in a vertical direction (y = 16) and 16 samples in a horizontal direction (x = 16). Likewise, an NxN CU generally has N samples in a

vertical direction and N samples in a horizontal direction, where N represents a nonnegative integer value. The samples in a CU may be arranged in rows and columns. Moreover, CUs need not necessarily have the same number of samples in the horizontal direction as in the vertical direction. For example, CUs may comprise NxM samples, where M is not necessarily equal to N.

[0058] Video encoder 200 encodes video data for CUs representing prediction and/or residual information, and other information. The prediction information indicates how the CU is to be predicted in order to form a prediction block for the CU. The residual information generally represents sample-by-sample differences between samples of the CU prior to encoding and the prediction block.

[0059] To predict a CU, video encoder 200 may generally form a prediction block for the CU through inter-prediction or intra-prediction. Inter-prediction generally refers to predicting the CU from data of a previously coded picture, whereas intra-prediction generally refers to predicting the CU from previously coded data of the same picture. To perform inter-prediction, video encoder 200 may generate the prediction block using one or more motion vectors. Video encoder 200 may generally perform a motion search to identify a reference block that closely matches the CU, e.g., in terms of differences between the CU and the reference block. Video encoder 200 may calculate a difference metric using a sum of absolute difference (SAD), sum of squared differences (SSD), mean absolute difference (MAD), mean squared differences (MSD), or other such difference calculations to determine whether a reference block closely matches the current CU. In some examples, video encoder 200 may predict the current CU using uni-directional prediction or bi-directional prediction.

[0060] Some examples of VVC also provide an affine motion compensation mode, which may be considered an inter-prediction mode. In affine motion compensation mode, video encoder 200 may determine two or more motion vectors that represent non-translational motion, such as zoom in or out, rotation, perspective motion, or other irregular motion types.

[0061] To perform intra-prediction, video encoder 200 may select an intra-prediction mode to generate the prediction block. Some examples of VVC provide sixty-seven intra-prediction modes, including various directional modes, as well as planar mode and DC mode. In general, video encoder 200 selects an intra-prediction mode that describes neighboring samples to a current block (e.g., a block of a CU) from which to predict samples of the current block. Such samples may generally be above, above and to the

left, or to the left of the current block in the same picture as the current block, assuming video encoder 200 codes CTUs and CUs in raster scan order (left to right, top to bottom).

[0062] Video encoder 200 encodes data representing the prediction mode for a current block. For example, for inter-prediction modes, video encoder 200 may encode data representing which of the various available inter-prediction modes is used, as well as motion information for the corresponding mode. For uni-directional or bi-directional inter-prediction, for example, video encoder 200 may encode motion vectors using advanced motion vector prediction (AMVP) or merge mode. Video encoder 200 may use similar modes to encode motion vectors for affine motion compensation mode.

[0063] Following prediction, such as intra-prediction or inter-prediction of a block, video encoder 200 may calculate residual data for the block. The residual data, such as a residual block, represents sample by sample differences between the block and a prediction block for the block, formed using the corresponding prediction mode. Video encoder 200 may apply one or more transforms to the residual block, to produce transformed data in a transform domain instead of the sample domain. For example, video encoder 200 may apply a discrete cosine transform (DCT), an integer transform, a wavelet transform, or a conceptually similar transform to residual video data. Additionally, video encoder 200 may apply a secondary transform following the first transform, such as a mode-dependent non-separable secondary transform (MDNSST), a signal dependent transform, a Karhunen-Loeve transform (KLT), or the like. Video encoder 200 produces transform coefficients following application of the one or more transforms.

[0064] As noted above, following any transforms to produce transform coefficients, video encoder 200 may perform quantization of the transform coefficients. 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. By performing the quantization process, video encoder 200 may reduce the bit depth associated with some or all of the transform coefficients. For example, video encoder 200 may round an n-bit value down to an m-bit value during quantization, where n is greater than m. In some examples, to perform quantization, video encoder 200 may perform a bitwise right-shift of the value to be quantized.

[0065] Following quantization, video encoder 200 may scan the transform coefficients, producing a one-dimensional vector from the two-dimensional matrix including the quantized transform coefficients. The scan may be designed to place higher energy (and therefore lower frequency) transform coefficients at the front of the vector and to place lower energy (and therefore higher frequency) transform coefficients at the back of the vector. In some examples, video encoder 200 may utilize a predefined scan order to scan the quantized transform coefficients to produce a serialized vector, and then entropy encode the quantized transform coefficients of the vector. In other examples, video encoder 200 may perform an adaptive scan. After scanning the quantized transform coefficients to form the one-dimensional vector, video encoder 200 may entropy encode the one-dimensional vector, e.g., according to context-adaptive binary arithmetic coding (CABAC). Video encoder 200 may also entropy encode values for syntax elements describing metadata associated with the encoded video data for use by video decoder 300 in decoding the video data.

[0066] To perform CABAC, video encoder 200 may assign a context within a context model to a symbol to be transmitted. The context may relate to, for example, whether neighboring values of the symbol are zero-valued or not. The probability determination may be based on a context assigned to the symbol.

[0067] Video encoder 200 may further generate syntax data, such as block-based syntax data, picture-based syntax data, and sequence-based syntax data, to video decoder 300, e.g., in a picture header, a block header, a slice header, or other syntax data, such as a sequence parameter set (SPS), picture parameter set (PPS), or video parameter set (VPS). Video decoder 300 may likewise decode such syntax data to determine how to decode corresponding video data.

[0068] In this manner, video encoder 200 may generate a bitstream including encoded video data, e.g., syntax elements describing partitioning of a picture into blocks (e.g., CUs) and prediction and/or residual information for the blocks. Ultimately, video decoder 300 may receive the bitstream and decode the encoded video data.

[0069] In general, video decoder 300 performs a reciprocal process to that performed by video encoder 200 to decode the encoded video data of the bitstream. For example, video decoder 300 may decode values for syntax elements of the bitstream using CABAC in a manner substantially similar to, albeit reciprocal to, the CABAC encoding process of video encoder 200. The syntax elements may define partitioning information for partitioning of a picture into CTUs, and partitioning of each CTU according to a

corresponding partition structure, such as a QTBT structure, to define CUs of the CTU. The syntax elements may further define prediction and residual information for blocks (e.g., CUs) of video data.

[0070] The residual information may be represented by, for example, quantized transform coefficients. Video decoder 300 may inverse quantize and inverse transform the quantized transform coefficients of a block to reproduce a residual block for the block. Video decoder 300 uses a signaled prediction mode (intra- or inter-prediction) and related prediction information (e.g., motion information for inter-prediction) to form a prediction block for the block. Video decoder 300 may then combine the prediction block and the residual block (on a sample-by-sample basis) to reproduce the original block. Video decoder 300 may perform additional processing, such as performing a deblocking process to reduce visual artifacts along boundaries of the block.

[0071] This disclosure may generally refer to "signaling" certain information, such as syntax elements. The term "signaling" may generally refer to the communication of values for syntax elements and/or other data used to decode encoded video data. That is, video encoder 200 may signal values for syntax elements in the bitstream. In general, signaling refers to generating a value in the bitstream. As noted above, source device 102 may transport the bitstream to destination device 116 substantially in real time, or not in real time, such as might occur when storing syntax elements to storage device 112 for later retrieval by destination device 116.

[0072] In accordance with the techniques of this disclosure, as will be explained in more detail below, video encoder 200 and video decoder 300 may be configured to determine a Rice parameter value for coding a current transform coefficient, wherein a possible range for Rice parameter values is from 0 to greater than 3, and code the current transform coefficient using the determined Rice parameter value. In a more specific example, video encoder 200 and video decoder 300 may be configured to determine a sum of absolute values of neighboring transform coefficients of a current transform coefficient, determine a shift value based on the sum of absolute values, determine a Rice parameter value for coding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value, and code the syntax element for the current transform coefficient using the Rice parameter value.

[0073] **Overview of Rice Parameter Derivation for Regular Residual Coding**

[0074] Regular residual coding (RRC) is an example technique for encoding and decoding a residual block. RRC is a term that may distinguish residual coding

techniques that include transforms from other residual coding techniques, such as transform skip (TS) residual coding. From the context of video decoder 300, RRC involves decoding a plurality of syntax elements that are used to determine the magnitudes and polarities (e.g., positive or negative) of transform coefficients in a block. Once the transform coefficient values are determined, video decoder 300 may apply one or more transforms to the transform coefficients to reconstruct the residual block. Example syntax elements that may define the position and values of transform coefficients may include one or more last significant coefficient position syntax elements, a coded block flag, a significant coefficient flag, one or more absolute level greater than X flags, a parity flag, a remainder syntax element, and an absolute value syntax element.

[0075] In general, the last significant coefficient position syntax elements may indicate the x and y position of a last significant (e.g., non-zero) coefficient in a transform block with respect to a forward scanning order. Video decoder 300 may determine the position of this last significant coefficient and then parse and decode the other syntax elements for other transform coefficients in a reverse scanning order starting for this last significant coefficient position.

[0076] The coded block flag indicates whether or not a block or subblock (e.g., a subblock of transform block) includes significant coefficients. For example, in VVC, an example coded block flag is sb_coded_flag[xS][yS], which specifies whether or not a subblock of a transform block includes non-zero transform coefficients.

[0077] The significant coefficient flag indicates whether or not the absolute value of a particular transform coefficient is non-zero. For example, in VVC, an example significant coefficient flag is sig_coeff_flag[xC][yC], which specifies for the transform coefficient location (xC, yC) within the current transform block whether the corresponding transform coefficient level at the location (xC, yC) is non-zero.

[0078] The one or more absolute level greater than X flags are syntax elements that indicate whether or not the absolute value of particular transform coefficient is greater than some value X (e.g., 1, 2, 3, etc.). In VVC, an example of an absolute value greater than X flag is abs_level_gtx_flag[n][j], which specifies whether the absolute value of the transform coefficient level (at scanning position n) is greater than (j << 1) + 1.

[0079] A parity flag is a flag that indicates whether the value of a transform coefficient is positive or negative. In VVC, an example parity flag is par_level_flag[n], which specifies the parity of the transform coefficient level at scanning position n.

[0080] A remainder syntax element is a syntax element whose value indicates the remaining absolute value of a transform coefficient above a certain other value (e.g., the largest value X coded for an absolute level greater than X flag). In VVC, an example of an remainder syntax element is abs_remainder[n], which indicates the remaining absolute value of a transform coefficient level that is coded with a Golomb-Rice code at the scanning position n.

[0081] An absolute value syntax element is another syntax element that may be coded to indicate the absolute value of a transform coefficient. In VVC, an example absolute value syntax element is dec_abs_level[n], which is an intermediate value that is coded with a Golomb-Rice code at the scanning position n.

[0082] In some examples of RRC, or other residual coding techniques, certain syntax elements may be binarized according to a binarization process. For example, abs_remainder[n] and dec_abs_level[n] are binarized according to a truncated Rice binarization process. The truncated Rice binarization process is performed based on a Rice parameter. Video decoder 300 may be configured to determine a Rice parameter in order to perform an inverse binarization process to determine the values of such syntax elements (e.g., abs_remainder[n] and dec_abs_level[n]). In some examples, such as VVC, video encoder 200 and video decoder 300 may be configured to derive a Rice parameter for RRC using a look-up table and the coefficient values of neighboring transform coefficients in a template. The template of the neighboring coefficients is shown in FIG. 2. FIG. 2 shows a transform block 390 with a current coefficient 392 and neighboring coefficients 394. The values of the neighboring coefficients 394 may be used to determine a Rice parameter for coding syntax elements related to current coefficient 392.

[0083] First, video encoder 200 and/or video decoder 300 may be configured to calculate a value of *locSumAbs*, which is the sum of absolute values of five available neighboring coefficients in a template relative to a current coefficient. In FIG. 2, the current coefficient 392 is shaded in black and the neighboring coefficients 394 in the template are shaded in gray. Video encoder 200 and video decoder 300 may normalize the value of *locSumAbs* (e.g., using subtraction and clipping) as follows: locSumAbs = Clip3(0, 31, *locSumAbs* - baseLevel * 5). Clip3 is a clipping function and limits the value of (locSumAbs - baseLevel * 5) to be between 0 and 31, inclusive. The variable baseLevel multiplied by 5 (with baseLevel not equal to zero) implements an offset to the estimate from the template locSumAbs value if a part of the transform coefficient

value is being coded with context-based arithmetic coding. Video encoder 200 and video decoder 300 may derive the Rice parameter using the normalized value of *locSumAbs* as an input to a look-up table, e.g., Table 1 shown in FIG. 3. As shown in FIG. 3, in some examples, the range of values for the Rice parameter is constrained from 0 to 3, inclusive. As further shown in FIG. 3, the range of values for the normalized *locSumAbs* ranges from 0 to 31, inclusive.

[0084] In VVC, the Rice parameter has been tested for encoding video sources of 8-bit bit-depth or 10-bit bit-depth. In the VVC design, the Rice parameter value is dependent on the value of *locSumAbs* and is limited to be in the range from 0 to 3, inclusive, through the clipping process described above. When the bit-depth of the input video is increased, extended precision is enabled, and/or the quantization parameter (QP) is set very low, the range of values of transform coefficients, and therfore *locSumAbs* value, may significantly increase. In such cases, the range of allowed Rice parameter values in VVC may not be sufficient for optimal coding efficiency, since the current design of VVC (e.g. ,VVC Draft 10) would require utilization of long codewords, e.g., codewords of length 32, for a coded coefficients having large values. Such a problem may be particularly noticeable for the abs_remainder[n] and dec_abs_level[n] syntax elements.

[0085] Visualizations of the problem are shown in FIG. 4 and FIG. 5. FIG. 4 illustrates an example histogram (Hist) of coded coefficients (in the form of a power of 2) for a 16 bit signal. FIG. 4 shows the total number of occurrences (Y axis) for particualr codeword lengths (X axis). Plot 400 is the histogram for all coefficients (coeffsHist), plot 402 is the histogram for DC coefficients (coeffsHistDC), and plot 404 is the histogram for the last significant coefficients (coeffsHistlast).

[0086] FIG. 5 illustrates example codeword lengths (e.g., codeword lengths for remainder and/or absolute value syntax elements) for different Rice parameters. As shown in FIG. 5, the normative range of Rice paramter values being limited to 3 (as defined in VVC), leads to a noticable increase in codeword length for coded values greater than 10 (e.g., see plot 502). In FIG. 5, plots 500, 502, 504, 506, and 508 show codeword length for Rice paramter values 0, 3, 8, 12, and 15, respectively. As shown in FIG. 5, Rice paramter values from 0 to 3 generally result in small codeword lengths for smaller coded values, but quickly produce large codeword lengths for larger coded values. This problem is mitigated by allowing a larger value for the Rice parameter for coefficients of larger magnitude.

[0087] This disclosure describes techniques that address the limitations of Rice parameter derivation techniques in VVC, and other video coding processes, for various input bit-depths of video data. The techniques of this disclosure may improve compression efficiency of video coding designs, particular video coding deisgns using bit depths over 10 bits. With the proposed techniques of this disclsoure, the supported range of Rice paramters can be extended from 0 to 3 in VVC up to a larger number, e.g., 16, which would provide for a more efficient binarization process. The more efficient binarization process may result in a lower number of bits used for large values of transform coefficients. In particular, the techniqeus of this disclsoure may result in smaller codeword lengths for remainder and/or absolute value syntax elements for transform coefficients having larger absolute values.

[0088] In one example of the disclosure, video encoder 200 and video decoder 300 may be configured to scale and/or nomralize the calcuated value of a template of neighboring coefficients (e.g., called *localSumbAbs)* to handle bitdepth increases and/or or a larger dynamical range of transform coefficients. Video encoder 200 and video decoder 300 may be configured scale and/or nomralize the value of *localSumbAbs* before being used to derive the Rice parameter. Video encoder 200 and video decoder 300 may be configured to determine the amount of scaling factor used based on the value of *localSumAbs* subtracted by the output of a template derivation computed to indicate local acitivity of transform coefficients. In another example, video decoder 300 may be configured to determine the scaling factor from a syntax element signalled in the bitstream or from set of tabulated values.

[0089] Video encoder 200 and video decoder 300 may compare the determined value of localSumAbs against a set of tabulated thresholds Tx = {Tid} (e.g., called g_riceT below) to identify a dynamical range id (rangeIdx) the value of localSumAbs falls within. In one example the tabluated thresholds g_riceT have the following eight values Tx[8] = { 32,128, 512, 2048, 8192, 32768, 131072, 524288 }. If the value of localSumAbs is less than 32, then the range id is 0. If the value of localSumAbs is less than 128, but greater than or equal to 32, then the range id is 1. If the value of localSumAbs is less than 512, but greater than or equal to 128, then the range id is 2, and so on.

[0090] Video encoder 200 and video decoder 300 may determine a scaler value (*normShift)* from a given set of tabulated scalers Rx = {Rid} (e.g., called g_riceShift below) determined by previously determined range id. The scalers in scaler array Rx

may be referred to as scale factors and/or shift values. Video encoder 200 and video decoder 300 may use the previously determined range id as an input to the set of tabulated scalers Rx (g_riceShift) to determine the *normShift* value. In one example, the tabluated thresholds g_riceT have the following nine values Rx[9] = { 0, 2, 4, 6, 8, 10, 12, 14, 16 }. If the value the range id is 0, then the *normShift* value is 0. If the value of range id is 1, then the *normShift* value 2. If the value the range id is 2, then the *normShift* value is 4, and so on.

[0091] Video encoder 200 and video decoder 300 may use the *normShift* value to normalize/scale the value of *localSumAbs* prior to determinig the Rice paramter. For example, video encoder 200 and video decoder 300 may right shift the value of *localSumAbs* by the value of *normShift* (e.g., the normalized *locSumAbs* = *locSumAbs* >> *normShift*). Video encoder 200 and video decoder 300 are configured to use the normalized *localSumAbs* to derive the Rice parameter using a predefined lookup table (e.g., Table 1 in FIG. 3 or another predefined table). More generally, video encoder 200 and video decoder 300 may use the normalized *localSumAbs* as an input to a function, lookup table, or process for determining a Rice paramter.

[0092] In one example, with the value of *localSumbAbs* being normalzied as described above, video encoder 200 and video decoder 300 may derive the Rice parameter from a predefined table. Vidoe encoder 200 and video decoder 300 may then modify the Rice paramter determied from the table by adding an offset equal to Rid (e.g., *normShift*) to extend the dynamical range of the Rice parameter. That is, the final Rice paramter is equal to the Rice paramter determined from the table plus the value of *normShift*.

[0093] In some examples, the set of thresholds Tx and scaler value Rx can be defined as follows:

Tx[8] = { 32,128, 512, 2048, 8192, 32768, 131072, 524288 };

Rx[9] = { 0, 2, 4, 6, 8, 10, 12, 14, 16 }.

[0094] In some examples, the values comprising the array Tx are restricted to be powers of two. In other examples, the values are not restricted to be powers of two (e.g., 2^x), but can be converted to such through a limited number of operations, such as offset, scaling or right/left shift.

[0095] The size of the arrays is not limited to 8 or 9 values, as above, but can be equal to N, where N is a positive integer larger than a lower boundary, e.g., greater than 1 or greater than 2, and less than an upper boundary, e.g., 9 or 5. In some examples, the size

of the Rx array is larger than the size of the Tx array by one entry. In some examples, the size of the Tx and Rx arrays can be restricted to 4.

[0096] The related Ricer parameter derivation part described in VVC Draft 10 may be modified as follows, with the size of the tabulated arrays (Rx and Tx) equal to 4. Added material relative to VVC Draft 10 is shown between the tags **<ADD>** and **</ADD>**.

[0097] 9.3.3.2 Rice parameter derivation process for abs_remainder[ ] and dec_abs_level[ ]

Inputs to this process are the base level baseLevel, the colour component index cIdx, the luma location (x0, y0) specifying the top-left sample of the current transform block relative to the top-left sample of the current picture, the current coefficient scan location (xC, yC), the binary logarithm of the transform block width log2TbWidth, and the binary logarithm of the transform block height log2TbHeight.

Output of this process is the Rice parameter cRiceParam.

Given the array AbsLevel[x][y] for the transform block with component index cIdx and the top-left luma location (x0, y0), the variable locSumAbs is derived as specified by the following pseudo-code process:

```
locSumAbs = 0
if( xC < ( 1 << log2TbWidth ) − 1 ) {
   locSumAbs += AbsLevel[ xC + 1 ][ yC ]
   if( xC < ( 1 << log2TbWidth ) − 2 )
      locSumAbs += AbsLevel[ xC + 2 ][ yC ]
   if( yC < ( 1 << log2TbHeight ) − 1 )
      locSumAbs += AbsLevel[ xC + 1 ][ yC + 1 ]                    (1517)
}
if( yC < ( 1 << log2TbHeight ) − 1 ) {
   locSumAbs += AbsLevel[ xC ][ yC + 1 ]
   if( yC < ( 1 << log2TbHeight ) − 2 )
      locSumAbs += AbsLevel[ xC ][ yC + 2 ]
}
<ADD>sum = localSumAbs
        if (sum < g_riceT[0])
          rangeIdx = 0
        else if (sum < g_riceT[1])
          rangeIdx = 1
```

```
            else if (sum < g_riceT[2])
              rangeIdx = 2
            else if (sum < g_riceT[3])
              rangeIdx = 3
            else
               rangeIdx = 4
      normShift = g_riceShift[rangeIdx]
      locSumAbs = locSumAbs >> normShift</ADD>
      locSumAbs = Clip3( 0, 31, locSumAbs − baseLevel * 5 )
```

<ADD>With g_riceT and g_riceShift being defined as following:

g_riceT[8] = { 32,128, 512, 2048};

g_riceShift[8] = { 0, 2, 4, 6, 8};</ADD>

Given the variable locSumAbs, the Rice parameter cRiceParam is derived as specified in Table 128.

<ADD> cRiceParam is then is refined as:

cRiceParam = cRiceParam + normShift </ADD>

[0098] In the above example, the threshold array Tx is called g_riceT and the scaler array Rx is called g_riceShift. The following pseudo code shows the comparison of the localSumAbs value to the various entries in the threshold array g_riceT:

```
      sum = localSumAbs
            if (sum < g_riceT[0])
              rangeIdx = 0
            else if (sum < g_riceT[1])
              rangeIdx = 1
            else if (sum < g_riceT[2])
              rangeIdx = 2
            else if (sum < g_riceT[3])
              rangeIdx = 3
            else
               rangeIdx = 4
```

[0099] As shown above, the localSumAbs value (sum) is sequentially compared with each of the tresholds (e.g., the sequential comparators) in array g_riceT to determine a range id (rangeIdx). Once the value of rangeIdx is determined, video encoder 200 and

video decoder 300 sets the shift value (normShift) according to following equation: normShift = g_riceShift[rangeIdx]. That is, rangeIdx is used as an input to the scale factor (or shift value) array g_riceShift.

[0100] In the example above, the g_riceT and g_riceShift arrays are defined as follows:

g_riceT[8] = { 32,128, 512, 2048};

g_riceShift[8] = { 0, 2, 4, 6, 8}

[0101] After the value of normShift is determined, video encoder 200 and video decoder 300 may normalize the value of locSumAbs using the following equation: locSumAbs = locSumAbs >> normShift, where >> is a right shift. Video encoder 200 and video decoder 300 may then clip the normalized locSumAbs value using the following equation, as described above: locSumAbs = Clip3(0, 31, locSumAbs – baseLevel * 5). This clipped locSumAbs value may then be used as an input to a lookup table to determine an initial Rice parameter value (e.g., the table in FIG. 3). Video encoder 200 and video decoder 300 may then add the normShift value to the initial Rice parameter value to get the final Rice parameter value (cRiceParam) as follows: cRiceParam = cRiceParam + normShift.

[0102] In some exampes, the g_riceT and/or g_riceShift arrays can be extended to 8 entries, with associated extensions to the sequential comparators (e.g., the if else comparisons) defined above.

[0103] In some examples, a parallel friendly impementation of the sequential compararisons of x vs {Tx} defined above can be utilized.

x = localSumAbs

y = 0

y |= (x >> 5)?1:0

y |= (x>>7)?2:0

y |= (x>>9)?4:0

y |= (x>>11)?8:0

normShift = Rx(y)

Rx= {0,2,4,4,6,6, 6,8,8,8,8,8,8, 8}

[0104] The implementation shown above is one possible technique for conducting normShift derivation (as Rx (rangeIdx)) from an x value derived as an output of template computation. To avoid sequential branching by comparison against the thresholds (e.g., as in the if/else statements described earlier), the value x can be downshifted in parallel with set of prespecified right shifts (>>). Depending on the

result of right shift, binary "or" operations aggregate the bits to produce the y variable. The variable y used to fetch the parameter normShift from an Rx table.

[0105] In another example, the sequential comparisons of x vs {Tx} defined above can be substitituted with approximations, e.g., with floor(log2(x)) operations, or through a search for most significant bit, leading zeros, and so on.

[0106] In some examples, separate Tx and Rx arrays can be defined for separate color components, color formats, or types of the syntax elements being decoded with the Rice parameter (e.g., a remainder or absolute decoded value, or another certain syntax element value). Such syntax element values can be determined using paramters such as the bit depth of the signal, internal bitdepth, or mode of the Rice derivation process.

[0107] In summary, in one example, video encoder 200 and video decoder 300 may be configured to determine a Rice parameter value for coding a current transform coefficient, wherein possible range for Rice parameter values is from 0 to greater than 3, and code the current transform coefficient using the determined Rice parameter value. In one example, the possible range for Rice parameter values is from 0 to 16.

[0108] In a more specific example, video encoder 200 and video decoder 300 may be configured to determine a sum of absolute values (e.g., locSumAbs) of neighboring transform coefficients of a current transform coefficient. Video encoder 200 and video decoder 300 may then determine a shift value (e.g., normShift) based on the sum of absolute values. Video encoder 200 and video decoder 300 may further determine a Rice parameter value (cRiceParam) for coding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value. As discussed above, possible syntax elements may include a remainder syntax elements and/or an absolute value syntax element. Video encoder 200 and video decoder 300 may then code (e.g., encode or decode, respectively) the syntax element for the current transform coefficient using the Rice parameter value.

[0109] In one example, to determine the shift value based on the sum of absolute values, video encoder 200 and video decoder 300 may determine a range id (e.g., rangeIdx) from the sum of absolute values, and determine the shift value from the range id. In order to determine the range id from the sum of absolute values, video encoder 200 and video decoder 300 may compare the sum of absolute values to an array of thresholds (e.g., Tx or g_riceT), determine the range id based on which two thresholds of the array of thresholds the sum of absolute values falls between. Comparing the sum of absolute values to the array of thresholds may include comparing the sum of absolute

values to all thresholds of the array of thresholds in parallel. In a further example, to determine the shift value from the range id, video encoder 200 and video decoder 300 may be configured to determine the shift value using the range id as an input to an array of scale factors (e.g., Rx or g_riceShift).

[0110] In another example, to determine the Rice parameter value for decoding the syntax element for the current transform coefficient based on the sum of absolute values and the shift value, video encoder 200 and video decoder 300 may be configured to normalize the sum of absolute values using the shift value to generate a normalized sum of absolute values. In one example, the normalization process is locSumAbs = locSumAbs >> normShift. Video encoder 200 and video decoder 300 may determine an initial Rice parameter value based on the normalized sum of absolute values, and add the shift value to the initial Rice parameter value to determine the Rice parameter value. For example, video encoder 200 and video decoder 300 may be configured to determine the Rice paramter value (cRiceParam) as cRiceParam = cRiceParam + normShift. To determine the initial Rice parameter based on the normalized sum of absolute values, video encoder 200 and video decoder 300 may be configured to determine the initial Rice parameter value using the normalized sum of absolute values as an input to a lookup table (e.g., the lookup table in FIG. 3). Before determining the intial Rice paramter value, video encoder 200 and video decoder 300 may be configured to clip the normalized sum of absolute values (e.g., locSumAbs =
Clip3(0, 31, *locSumAbs* - baseLevel * 5). Once the Rice parameter value has been determined video encoder 200 and video decoder 300 may be configured to binarize or inverse binarize, respectively, the syntax element based on the Rice parameter.

[0111] FIG. 6 is a block diagram illustrating an example video encoder 200 that may perform the techniques of this disclosure. FIG. 6 is provided for purposes of explanation and should not be considered limiting of the techniques as broadly exemplified and described in this disclosure. For purposes of explanation, this disclosure describes video encoder 200 according to the techniques of VVC (ITU-T H.266, under development), and HEVC (ITU-T H.265). However, the techniques of this disclosure may be performed by video encoding devices that are configured to other video coding standards.

[0112] In the example of FIG. 6, video encoder 200 includes video data memory 230, mode selection unit 202, residual generation unit 204, transform processing unit 206, quantization unit 208, inverse quantization unit 210, inverse transform processing unit

212, reconstruction unit 214, filter unit 216, decoded picture buffer (DPB) 218, and entropy encoding unit 220. Any or all of video data memory 230, mode selection unit 202, residual generation unit 204, transform processing unit 206, quantization unit 208, inverse quantization unit 210, inverse transform processing unit 212, reconstruction unit 214, filter unit 216, DPB 218, and entropy encoding unit 220 may be implemented in one or more processors or in processing circuitry. For instance, the units of video encoder 200 may be implemented as one or more circuits or logic elements as part of hardware circuitry, or as part of a processor, ASIC, or FPGA. Moreover, video encoder 200 may include additional or alternative processors or processing circuitry to perform these and other functions.

[0113] Video data memory 230 may store video data to be encoded by the components of video encoder 200. Video encoder 200 may receive the video data stored in video data memory 230 from, for example, video source 104 (FIG. 1). DPB 218 may act as a reference picture memory that stores reference video data for use in prediction of subsequent video data by video encoder 200. Video data memory 230 and DPB 218 may be formed by any of a variety of memory devices, such as dynamic random access memory (DRAM), including synchronous DRAM (SDRAM), magnetoresistive RAM (MRAM), resistive RAM (RRAM), or other types of memory devices. Video data memory 230 and DPB 218 may be provided by the same memory device or separate memory devices. In various examples, video data memory 230 may be on-chip with other components of video encoder 200, as illustrated, or off-chip relative to those components.

[0114] In this disclosure, reference to video data memory 230 should not be interpreted as being limited to memory internal to video encoder 200, unless specifically described as such, or memory external to video encoder 200, unless specifically described as such. Rather, reference to video data memory 230 should be understood as reference memory that stores video data that video encoder 200 receives for encoding (e.g., video data for a current block that is to be encoded). Memory 106 of FIG. 1 may also provide temporary storage of outputs from the various units of video encoder 200.

[0115] The various units of FIG. 6 are illustrated to assist with understanding the operations performed by video encoder 200. The units may be implemented as fixed-function circuits, programmable circuits, or a combination thereof. Fixed-function circuits refer to circuits that provide particular functionality, and are preset on the operations that can be performed. Programmable circuits refer to circuits that can be

programmed to perform various tasks, and provide flexible functionality in the operations that can be performed. For instance, programmable circuits may execute software or firmware that cause the programmable circuits to operate in the manner defined by instructions of the software or firmware. Fixed-function circuits may execute software instructions (e.g., to receive parameters or output parameters), but the types of operations that the fixed-function circuits perform are generally immutable. In some examples, one or more of the units may be distinct circuit blocks (fixed-function or programmable), and in some examples, one or more of the units may be integrated circuits.

[0116] Video encoder 200 may include arithmetic logic units (ALUs), elementary function units (EFUs), digital circuits, analog circuits, and/or programmable cores, formed from programmable circuits. In examples where the operations of video encoder 200 are performed using software executed by the programmable circuits, memory 106 (FIG. 1) may store the instructions (e.g., object code) of the software that video encoder 200 receives and executes, or another memory within video encoder 200 (not shown) may store such instructions.

[0117] Video data memory 230 is configured to store received video data. Video encoder 200 may retrieve a picture of the video data from video data memory 230 and provide the video data to residual generation unit 204 and mode selection unit 202. Video data in video data memory 230 may be raw video data that is to be encoded.

[0118] Mode selection unit 202 includes a motion estimation unit 222, a motion compensation unit 224, and an intra-prediction unit 226. Mode selection unit 202 may include additional functional units to perform video prediction in accordance with other prediction modes. As examples, mode selection unit 202 may include a palette unit, an intra-block copy unit (which may be part of motion estimation unit 222 and/or motion compensation unit 224), an affine unit, a linear model (LM) unit, or the like.

[0119] Mode selection unit 202 generally coordinates multiple encoding passes to test combinations of encoding parameters and resulting rate-distortion values for such combinations. The encoding parameters may include partitioning of CTUs into CUs, prediction modes for the CUs, transform types for residual data of the CUs, quantization parameters for residual data of the CUs, and so on. Mode selection unit 202 may ultimately select the combination of encoding parameters having rate-distortion values that are better than the other tested combinations.

[0120] Video encoder 200 may partition a picture retrieved from video data memory 230 into a series of CTUs, and encapsulate one or more CTUs within a slice. Mode selection unit 202 may partition a CTU of the picture in accordance with a tree structure, such as the QTBT structure or the quad-tree structure of HEVC described above. As described above, video encoder 200 may form one or more CUs from partitioning a CTU according to the tree structure. Such a CU may also be referred to generally as a "video block" or "block."

[0121] In general, mode selection unit 202 also controls the components thereof (e.g., motion estimation unit 222, motion compensation unit 224, and intra-prediction unit 226) to generate a prediction block for a current block (e.g., a current CU, or in HEVC, the overlapping portion of a PU and a TU). For inter-prediction of a current block, motion estimation unit 222 may perform a motion search to identify one or more closely matching reference blocks in one or more reference pictures (e.g., one or more previously coded pictures stored in DPB 218). In particular, motion estimation unit 222 may calculate a value representative of how similar a potential reference block is to the current block, e.g., according to sum of absolute difference (SAD), sum of squared differences (SSD), mean absolute difference (MAD), mean squared differences (MSD), or the like. Motion estimation unit 222 may generally perform these calculations using sample-by-sample differences between the current block and the reference block being considered. Motion estimation unit 222 may identify a reference block having a lowest value resulting from these calculations, indicating a reference block that most closely matches the current block.

[0122] Motion estimation unit 222 may form one or more motion vectors (MVs) that defines the positions of the reference blocks in the reference pictures relative to the position of the current block in a current picture. Motion estimation unit 222 may then provide the motion vectors to motion compensation unit 224. For example, for uni-directional inter-prediction, motion estimation unit 222 may provide a single motion vector, whereas for bi-directional inter-prediction, motion estimation unit 222 may provide two motion vectors. Motion compensation unit 224 may then generate a prediction block using the motion vectors. For example, motion compensation unit 224 may retrieve data of the reference block using the motion vector. As another example, if the motion vector has fractional sample precision, motion compensation unit 224 may interpolate values for the prediction block according to one or more interpolation filters. Moreover, for bi-directional inter-prediction, motion compensation unit 224 may

retrieve data for two reference blocks identified by respective motion vectors and combine the retrieved data, e.g., through sample-by-sample averaging or weighted averaging.

[0123] As another example, for intra-prediction, or intra-prediction coding, intra-prediction unit 226 may generate the prediction block from samples neighboring the current block. For example, for directional modes, intra-prediction unit 226 may generally mathematically combine values of neighboring samples and populate these calculated values in the defined direction across the current block to produce the prediction block. As another example, for DC mode, intra-prediction unit 226 may calculate an average of the neighboring samples to the current block and generate the prediction block to include this resulting average for each sample of the prediction block.

[0124] Mode selection unit 202 provides the prediction block to residual generation unit 204. Residual generation unit 204 receives a raw, unencoded version of the current block from video data memory 230 and the prediction block from mode selection unit 202. Residual generation unit 204 calculates sample-by-sample differences between the current block and the prediction block. The resulting sample-by-sample differences define a residual block for the current block. In some examples, residual generation unit 204 may also determine differences between sample values in the residual block to generate a residual block using residual differential pulse code modulation (RDPCM). In some examples, residual generation unit 204 may be formed using one or more subtractor circuits that perform binary subtraction.

[0125] In examples where mode selection unit 202 partitions CUs into PUs, each PU may be associated with a luma prediction unit and corresponding chroma prediction units. Video encoder 200 and video decoder 300 may support PUs having various sizes. As indicated above, the size of a CU may refer to the size of the luma coding block of the CU and the size of a PU may refer to the size of a luma prediction unit of the PU. Assuming that the size of a particular CU is 2Nx2N, video encoder 200 may support PU sizes of 2Nx2N or NxN for intra prediction, and symmetric PU sizes of 2Nx2N, 2NxN, Nx2N, NxN, or similar for inter prediction. Video encoder 200 and video decoder 300 may also support asymmetric partitioning for PU sizes of 2NxnU, 2NxnD, nLx2N, and nRx2N for inter prediction.

[0126] In examples where mode selection unit 202 does not further partition a CU into PUs, each CU may be associated with a luma coding block and corresponding chroma

coding blocks. As above, the size of a CU may refer to the size of the luma coding block of the CU. The video encoder 200 and video decoder 300 may support CU sizes of 2Nx2N, 2NxN, or Nx2N.

[0127] For other video coding techniques such as an intra-block copy mode coding, an affine-mode coding, and linear model (LM) mode coding, as some examples, mode selection unit 202, via respective units associated with the coding techniques, generates a prediction block for the current block being encoded. In some examples, such as palette mode coding, mode selection unit 202 may not generate a prediction block, and instead generate syntax elements that indicate the manner in which to reconstruct the block based on a selected palette. In such modes, mode selection unit 202 may provide these syntax elements to entropy encoding unit 220 to be encoded.

[0128] As described above, residual generation unit 204 receives the video data for the current block and the corresponding prediction block. Residual generation unit 204 then generates a residual block for the current block. To generate the residual block, residual generation unit 204 calculates sample-by-sample differences between the prediction block and the current block.

[0129] Transform processing unit 206 applies one or more transforms to the residual block to generate a block of transform coefficients (referred to herein as a "transform coefficient block"). Transform processing unit 206 may apply various transforms to a residual block to form the transform coefficient block. For example, transform processing unit 206 may apply a discrete cosine transform (DCT), a directional transform, a Karhunen-Loeve transform (KLT), or a conceptually similar transform to a residual block. In some examples, transform processing unit 206 may perform multiple transforms to a residual block, e.g., a primary transform and a secondary transform, such as a rotational transform. In some examples, transform processing unit 206 does not apply transforms to a residual block.

[0130] Quantization unit 208 may quantize the transform coefficients in a transform coefficient block, to produce a quantized transform coefficient block. Quantization unit 208 may quantize transform coefficients of a transform coefficient block according to a quantization parameter (QP) value associated with the current block. Video encoder 200 (e.g., via mode selection unit 202) may adjust the degree of quantization applied to the transform coefficient blocks associated with the current block by adjusting the QP value associated with the CU. Quantization may introduce loss of information, and

thus, quantized transform coefficients may have lower precision than the original transform coefficients produced by transform processing unit 206.

[0131] Inverse quantization unit 210 and inverse transform processing unit 212 may apply inverse quantization and inverse transforms to a quantized transform coefficient block, respectively, to reconstruct a residual block from the transform coefficient block. Reconstruction unit 214 may produce a reconstructed block corresponding to the current block (albeit potentially with some degree of distortion) based on the reconstructed residual block and a prediction block generated by mode selection unit 202. For example, reconstruction unit 214 may add samples of the reconstructed residual block to corresponding samples from the prediction block generated by mode selection unit 202 to produce the reconstructed block.

[0132] Filter unit 216 may perform one or more filter operations on reconstructed blocks. For example, filter unit 216 may perform deblocking operations to reduce blockiness artifacts along edges of CUs. Operations of filter unit 216 may be skipped, in some examples.

[0133] Video encoder 200 stores reconstructed blocks in DPB 218. For instance, in examples where operations of filter unit 216 are not performed, reconstruction unit 214 may store reconstructed blocks to DPB 218. In examples where operations of filter unit 216 are performed, filter unit 216 may store the filtered reconstructed blocks to DPB 218. Motion estimation unit 222 and motion compensation unit 224 may retrieve a reference picture from DPB 218, formed from the reconstructed (and potentially filtered) blocks, to inter-predict blocks of subsequently encoded pictures. In addition, intra-prediction unit 226 may use reconstructed blocks in DPB 218 of a current picture to intra-predict other blocks in the current picture.

[0134] In general, entropy encoding unit 220 may entropy encode syntax elements received from other functional components of video encoder 200. For example, entropy encoding unit 220 may entropy encode quantized transform coefficient blocks from quantization unit 208. As another example, entropy encoding unit 220 may entropy encode prediction syntax elements (e.g., motion information for inter-prediction or intra-mode information for intra-prediction) from mode selection unit 202. Entropy encoding unit 220 may perform one or more entropy encoding operations on the syntax elements, which are another example of video data, to generate entropy-encoded data. For example, entropy encoding unit 220 may perform a context-adaptive variable length coding (CAVLC) operation, a CABAC operation, a variable-to-variable (V2V) length

coding operation, a syntax-based context-adaptive binary arithmetic coding (SBAC) operation, a Probability Interval Partitioning Entropy (PIPE) coding operation, an Exponential-Golomb encoding operation, or another type of entropy encoding operation on the data. In some examples, entropy encoding unit 220 may operate in bypass mode where syntax elements are not entropy encoded.

[0135] In one example, entropy encoding unit 220 may be configured to perform one or more techniques of this disclosure, as described above. For example, entropy encoding unit 220 may be configured to determine a sum of absolute values of neighboring transform coefficients of a current transform coefficient, determine a shift value based on the sum of absolute values, determine a Rice parameter value for encoding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value, and encode the syntax element for the current transform coefficient using the Rice parameter value.

[0136] Video encoder 200 may output a bitstream that includes the entropy encoded syntax elements needed to reconstruct blocks of a slice or picture. In particular, entropy encoding unit 220 may output the bitstream.

[0137] The operations described above are described with respect to a block. Such description should be understood as being operations for a luma coding block and/or chroma coding blocks. As described above, in some examples, the luma coding block and chroma coding blocks are luma and chroma components of a CU. In some examples, the luma coding block and the chroma coding blocks are luma and chroma components of a PU.

[0138] In some examples, operations performed with respect to a luma coding block need not be repeated for the chroma coding blocks. As one example, operations to identify a motion vector (MV) and reference picture for a luma coding block need not be repeated for identifying a MV and reference picture for the chroma blocks. Rather, the MV for the luma coding block may be scaled to determine the MV for the chroma blocks, and the reference picture may be the same. As another example, the intra-prediction process may be the same for the luma coding block and the chroma coding blocks.

[0139] Video encoder 200 represents an example of a device configured to encode video data including a memory configured to store video data, and one or more processing units implemented in circuitry and configured to determine a Rice parameter value for coding a current transform coefficient, wherein a possible range for Rice

parameter values is from 0 to greater than 3, and code the current transform coefficient using the determined Rice parameter value.

[0140] FIG. 7 is a block diagram illustrating an example video decoder 300 that may perform the techniques of this disclosure. FIG. 7 is provided for purposes of explanation and is not limiting on the techniques as broadly exemplified and described in this disclosure. For purposes of explanation, this disclosure describes video decoder 300 according to the techniques of VVC (ITU-T H.266, under development), and HEVC (ITU-T H.265). However, the techniques of this disclosure may be performed by video coding devices that are configured to other video coding standards.

[0141] In the example of FIG. 7, video decoder 300 includes coded picture buffer (CPB) memory 320, entropy decoding unit 302, prediction processing unit 304, inverse quantization unit 306, inverse transform processing unit 308, reconstruction unit 310, filter unit 312, and decoded picture buffer (DPB) 314. Any or all of CPB memory 320, entropy decoding unit 302, prediction processing unit 304, inverse quantization unit 306, inverse transform processing unit 308, reconstruction unit 310, filter unit 312, and DPB 314 may be implemented in one or more processors or in processing circuitry. For instance, the units of video decoder 300 may be implemented as one or more circuits or logic elements as part of hardware circuitry, or as part of a processor, ASIC, or FPGA. Moreover, video decoder 300 may include additional or alternative processors or processing circuitry to perform these and other functions.

[0142] Prediction processing unit 304 includes motion compensation unit 316 and intra-prediction unit 318. Prediction processing unit 304 may include additional units to perform prediction in accordance with other prediction modes. As examples, prediction processing unit 304 may include a palette unit, an intra-block copy unit (which may form part of motion compensation unit 316), an affine unit, a linear model (LM) unit, or the like. In other examples, video decoder 300 may include more, fewer, or different functional components.

[0143] CPB memory 320 may store video data, such as an encoded video bitstream, to be decoded by the components of video decoder 300. The video data stored in CPB memory 320 may be obtained, for example, from computer-readable medium 110 (FIG. 1). CPB memory 320 may include a CPB that stores encoded video data (e.g., syntax elements) from an encoded video bitstream. Also, CPB memory 320 may store video data other than syntax elements of a coded picture, such as temporary data representing outputs from the various units of video decoder 300. DPB 314 generally stores decoded

pictures, which video decoder 300 may output and/or use as reference video data when decoding subsequent data or pictures of the encoded video bitstream. CPB memory 320 and DPB 314 may be formed by any of a variety of memory devices, such as DRAM, including SDRAM, MRAM, RRAM, or other types of memory devices. CPB memory 320 and DPB 314 may be provided by the same memory device or separate memory devices. In various examples, CPB memory 320 may be on-chip with other components of video decoder 300, or off-chip relative to those components.

[0144] Additionally or alternatively, in some examples, video decoder 300 may retrieve coded video data from memory 120 (FIG. 1). That is, memory 120 may store data as discussed above with CPB memory 320. Likewise, memory 120 may store instructions to be executed by video decoder 300, when some or all of the functionality of video decoder 300 is implemented in software to be executed by processing circuitry of video decoder 300.

[0145] The various units shown in FIG. 7 are illustrated to assist with understanding the operations performed by video decoder 300. The units may be implemented as fixed-function circuits, programmable circuits, or a combination thereof. Similar to FIG. 6, fixed-function circuits refer to circuits that provide particular functionality, and are preset on the operations that can be performed. Programmable circuits refer to circuits that can be programmed to perform various tasks, and provide flexible functionality in the operations that can be performed. For instance, programmable circuits may execute software or firmware that cause the programmable circuits to operate in the manner defined by instructions of the software or firmware. Fixed-function circuits may execute software instructions (e.g., to receive parameters or output parameters), but the types of operations that the fixed-function circuits perform are generally immutable. In some examples, one or more of the units may be distinct circuit blocks (fixed-function or programmable), and in some examples, one or more of the units may be integrated circuits.

[0146] Video decoder 300 may include ALUs, EFUs, digital circuits, analog circuits, and/or programmable cores formed from programmable circuits. In examples where the operations of video decoder 300 are performed by software executing on the programmable circuits, on-chip or off-chip memory may store instructions (e.g., object code) of the software that video decoder 300 receives and executes.

[0147] Entropy decoding unit 302 may receive encoded video data from the CPB and entropy decode the video data to reproduce syntax elements. Prediction processing unit

304, inverse quantization unit 306, inverse transform processing unit 308, reconstruction unit 310, and filter unit 312 may generate decoded video data based on the syntax elements extracted from the bitstream.

[0148] In general, video decoder 300 reconstructs a picture on a block-by-block basis. Video decoder 300 may perform a reconstruction operation on each block individually (where the block currently being reconstructed, i.e., decoded, may be referred to as a "current block").

[0149] Entropy decoding unit 302 may entropy decode syntax elements defining quantized transform coefficients of a quantized transform coefficient block, as well as transform information, such as a quantization parameter (QP) and/or transform mode indication(s). Inverse quantization unit 306 may use the QP associated with the quantized transform coefficient block to determine a degree of quantization and, likewise, a degree of inverse quantization for inverse quantization unit 306 to apply. Inverse quantization unit 306 may, for example, perform a bitwise left-shift operation to inverse quantize the quantized transform coefficients. Inverse quantization unit 306 may thereby form a transform coefficient block including transform coefficients.

[0150] In one example, entropy decoding unit 302 may be configured to perform one or more techniques of this disclosure, as described above. For example, entropy decoding unit 302 may be configured to determine a sum of absolute values of neighboring transform coefficients of a current transform coefficient, determine a shift value based on the sum of absolute values, determine a Rice parameter value for decoding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value, and decode the syntax element for the current transform coefficient using the Rice parameter value.

[0151] After inverse quantization unit 306 forms the transform coefficient block, inverse transform processing unit 308 may apply one or more inverse transforms to the transform coefficient block to generate a residual block associated with the current block. For example, inverse transform processing unit 308 may apply an inverse DCT, an inverse integer transform, an inverse Karhunen-Loeve transform (KLT), an inverse rotational transform, an inverse directional transform, or another inverse transform to the transform coefficient block.

[0152] Furthermore, prediction processing unit 304 generates a prediction block according to prediction information syntax elements that were entropy decoded by entropy decoding unit 302. For example, if the prediction information syntax elements

indicate that the current block is inter-predicted, motion compensation unit 316 may generate the prediction block. In this case, the prediction information syntax elements may indicate a reference picture in DPB 314 from which to retrieve a reference block, as well as a motion vector identifying a location of the reference block in the reference picture relative to the location of the current block in the current picture. Motion compensation unit 316 may generally perform the inter-prediction process in a manner that is substantially similar to that described with respect to motion compensation unit 224 (FIG. 6).

[0153] As another example, if the prediction information syntax elements indicate that the current block is intra-predicted, intra-prediction unit 318 may generate the prediction block according to an intra-prediction mode indicated by the prediction information syntax elements. Again, intra-prediction unit 318 may generally perform the intra-prediction process in a manner that is substantially similar to that described with respect to intra-prediction unit 226 (FIG. 6). Intra-prediction unit 318 may retrieve data of neighboring samples to the current block from DPB 314.

[0154] Reconstruction unit 310 may reconstruct the current block using the prediction block and the residual block. For example, reconstruction unit 310 may add samples of the residual block to corresponding samples of the prediction block to reconstruct the current block.

[0155] Filter unit 312 may perform one or more filter operations on reconstructed blocks. For example, filter unit 312 may perform deblocking operations to reduce blockiness artifacts along edges of the reconstructed blocks. Operations of filter unit 312 are not necessarily performed in all examples.

[0156] Video decoder 300 may store the reconstructed blocks in DPB 314. For instance, in examples where operations of filter unit 312 are not performed, reconstruction unit 310 may store reconstructed blocks to DPB 314. In examples where operations of filter unit 312 are performed, filter unit 312 may store the filtered reconstructed blocks to DPB 314. As discussed above, DPB 314 may provide reference information, such as samples of a current picture for intra-prediction and previously decoded pictures for subsequent motion compensation, to prediction processing unit 304. Moreover, video decoder 300 may output decoded pictures (e.g., decoded video) from DPB 314 for subsequent presentation on a display device, such as display device 118 of FIG. 1.

[0157] In this manner, video decoder 300 represents an example of a video decoding device including a memory configured to store video data, and one or more processing units implemented in circuitry and configured to determine a Rice parameter value for coding a current transform coefficient, wherein a possible range for Rice parameter values is from 0 to greater than 3, and code the current transform coefficient using the determined Rice parameter value.

[0158] FIG. 8 is a flowchart illustrating an example method for encoding a current block in accordance with the techniques of this disclosure. The current block may comprise a current CU. Although described with respect to video encoder 200 (FIGS. 1 and 6), it should be understood that other devices may be configured to perform a method similar to that of FIG. 8.

[0159] In this example, video encoder 200 initially predicts the current block (350). For example, video encoder 200 may form a prediction block for the current block. Video encoder 200 may then calculate a residual block for the current block (352). To calculate the residual block, video encoder 200 may calculate a difference between the original, unencoded block and the prediction block for the current block. Video encoder 200 may then transform the residual block and quantize transform coefficients of the residual block (354). Next, video encoder 200 may scan the quantized transform coefficients of the residual block (356). During the scan, or following the scan, video encoder 200 may entropy encode the transform coefficients (358). For example, video encoder 200 may encode the transform coefficients using CAVLC or CABAC. Video encoder 200 may then output the entropy encoded data of the block (360).

[0160] FIG. 9 is a flowchart illustrating an example method for decoding a current block of video data in accordance with the techniques of this disclosure. The current block may comprise a current CU. Although described with respect to video decoder 300 (FIGS. 1 and 7), it should be understood that other devices may be configured to perform a method similar to that of FIG. 9.

[0161] Video decoder 300 may receive entropy encoded data for the current block, such as entropy encoded prediction information and entropy encoded data for transform coefficients of a residual block corresponding to the current block (370). Video decoder 300 may entropy decode the entropy encoded data to determine prediction information for the current block and to reproduce transform coefficients of the residual block (372). Video decoder 300 may predict the current block (374), e.g., using an intra- or inter-prediction mode as indicated by the prediction information for the current block, to

calculate a prediction block for the current block. Video decoder 300 may then inverse scan the reproduced transform coefficients (376), to create a block of quantized transform coefficients. Video decoder 300 may then inverse quantize the transform coefficients and apply an inverse transform to the transform coefficients to produce a residual block (378). Video decoder 300 may ultimately decode the current block by combining the prediction block and the residual block (380).

[0162] FIG. 10 is another flowchart illustrating an example method for encoding a current block in accordance with the techniques of this disclosure. The techniques of FIG. 10 may be performed by one or more structural components of video encoder 200.

[0163] In one example of the disclosure, video encoder 200 may be configured to determine a sum of absolute values of neighboring transform coefficients of a current transform coefficient (600), and determine a shift value based on the sum of absolute values (602). Video encoder 200 may further determine a Rice parameter value for encoding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value (604), and encode the syntax element for the current transform coefficient using the Rice parameter value (606).

[0164] In one example, to determine the shift value based on the sum of absolute value, video encoder 200 is configured to determine a range id from the sum of absolute values, and determine the shift value from the range id. To determine the range id from the sum of absolute values, video encoder 200 may be configured to compare the sum of absolute values to an array of thresholds, and determine the range id based on which two thresholds of the array of thresholds the sum of absolute values falls between. Comparing the sum of absolute values to the array of thresholds may include comparing the sum of absolute values to all thresholds of the array of thresholds in parallel. In one example, to determine the shift value from the range id, video encoder 200 may be configured to determine the shift value using the range id as an input to an array of scale factors.

[0165] In another example, to determine the Rice parameter value for encoding the syntax element for the current transform coefficient based on the sum of absolute values and the shift value, video encoder 200 may be configured to normalize the sum of absolute values using the shift value to generate a normalized sum of absolute values, determine an initial Rice parameter value based on the normalized sum of absolute values, and the shift value to the initial Rice parameter value to determine the Rice parameter value. Video encoder 200 may be further configured to clip the normalized

sum of absolute values. In another example, to determine the initial Rice parameter based on the normalized sum of absolute values, video encoder 200 may be configured to determine the initial Rice parameter value using the normalized sum of absolute values as an input to a lookup table. Encoding the syntax element for the current transform coefficient using the Rice parameter value may include binarizing the syntax element based on the Rice parameter.

[0166] FIG. 11 is another flowchart illustrating an example method for decoding a current block in accordance with the techniques of this disclosure. The techniques of FIG. 11 may be performed by one or more structural components of video decoder 300.

[0167] In one example, video decoder 300 is configured to determine a sum of absolute values of neighboring transform coefficients of a current transform coefficient (700), and determine a shift value based on the sum of absolute values (702). Video decoder 300 may be further configured to determine a Rice parameter value for decoding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value (704), and decode the syntax element for the current transform coefficient using the Rice parameter value (706).

[0168] In one example, to determine the shift value based on the sum of absolute values, video decoder 300 is further configured to determine a range id from the sum of absolute values, and determine the shift value from the range id. To determine the range id from the sum of absolute values, video decoder 300 is further configured to compare the sum of absolute values to an array of thresholds, and determine the range id based on which two thresholds of the array of thresholds the sum of absolute values falls between. Video decoder 300 may configured to compare the sum of absolute values to all thresholds of the array of thresholds in parallel. Video decoder 300 may also determine the shift value using the range id as an input to an array of scale factors.

[0169] In another example, to determine the Rice parameter value for decoding the syntax element for the current transform coefficient based on the sum of absolute values and the shift value, video decoder 300 is further configured to normalize the sum of absolute values using the shift value to generate a normalized sum of absolute values, determine an initial Rice parameter value based on the normalized sum of absolute values, and add the shift value to the initial Rice parameter value to determine the Rice parameter value. Video decoder 300 may also clip the normalized sum of absolute values. In another example, to determine the initial Rice parameter based on the normalized sum of absolute values, video decoder 300 is further configured to

determine the initial Rice parameter value using the normalized sum of absolute values as an input to a lookup table. To decode the syntax element for the current transform coefficient using the Rice parameter value, video decoder 300 is further configured to inverse binarize the syntax element based on the Rice parameter.

[0170] Other illustrative aspects of the disclosure are described below.

[0171] Aspect 1A - A method of coding video data, the method comprising: determining a Rice parameter value for coding a current transform coefficient, wherein a possible range for Rice parameter values is from 0 to greater than 3; and coding the current transform coefficient using the determined Rice parameter value.

[0172] Aspect 2A - The method of Aspect 1A, wherein the possible range for Rice parameter values is from 0 to 16.

[0173] Aspect 3A - The method of Aspect 1A, wherein determining the Rice parameter value comprises: determining a value of localSumAbs, where localSumAbs is a sum of absolute values of neighboring transform coefficients to the current transform coefficient; and determining the Rice parameter value using the value of localSumAbs.

[0174] Aspect 4A - The method of Aspect 3A, further comprising: scaling the value of localSumAbs.

[0175] Aspect 5A - The method of Aspect 4A, wherein scaling the value of localSumAbs comprises: determining a dynamical range id for the value of localSumAbs, wherein determining the dynamical range id includes comparing the value of localSumAbs to a predetermined array of thresholds (Tx); determining a scale factor used to scale the value of localSumAbs using the dynamical range id and a predetermined array of scale factors (Rx); and scaling the value of localSumAbs using the scale factor.

[0176] Aspect 6A - The method of Aspect 5A, wherein determining the Rice parameter value comprises: determining the Rice parameter value using a lookup table and the scaled value of localSumAbs.

[0177] Aspect 7A - The method of any of Aspects 1A–6A, wherein coding comprises decoding.

[0178] Aspect 8A - The method of any of Aspects 1A–6A, wherein coding comprises encoding.

[0179] Aspect 9A - A device for coding video data, the device comprising one or more means for performing the method of any of Aspects 1A–8A.

[0180] Aspect 10A - The device of Aspect 9A, wherein the one or more means comprise one or more processors implemented in circuitry.

[0181] Aspect 11A - The device of any of Aspects 9A and 10A, further comprising a memory to store the video data.

[0182] Aspect 12A - The device of any of Aspects 9A-11A, further comprising a display configured to display decoded video data.

[0183] Aspect 13A - The device of any of Aspects 9A-12A, wherein the device comprises one or more of a camera, a computer, a mobile device, a broadcast receiver device, or a set-top box.

[0184] Aspect 14A - The device of any of Aspects 9A-13A, wherein the device comprises a video decoder.

[0185] Aspect 15A - The device of any of Aspects 9A-14A, wherein the device comprises a video encoder.

[0186] Aspect 16A - A computer-readable storage medium having stored thereon instructions that, when executed, cause one or more processors to perform the method of any of Aspects 1A–8A.

[0187] Aspect 1B - A method of decoding video data, the method comprising: determining a sum of absolute values of neighboring transform coefficients of a current transform coefficient; determining a shift value based on the sum of absolute values; determining a Rice parameter value for decoding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value; and decoding the syntax element for the current transform coefficient using the Rice parameter value.

[0188] Aspect 2B - The method of Aspect 1B, wherein determining the shift value based on the sum of absolute values comprises: determining a range id from the sum of absolute values; and determining the shift value from the range id.

[0189] Aspect 3B - The method of Aspect 2B, wherein determining the range id from the sum of absolute values comprises: comparing the sum of absolute values to an array of thresholds; and determining the range id based on which two thresholds of the array of thresholds the sum of absolute values falls between.

[0190] Aspect 4B - The method of Aspect 3B, wherein comparing the sum of absolute values to the array of thresholds comprises: comparing the sum of absolute values to all thresholds of the array of thresholds in parallel.

[0191] Aspect 5B - The method of Aspect 3B, wherein determining the shift value from the range id comprises: determining the shift value using the range id as an input to an array of scale factors.

[0192] Aspect 6B - The method of Aspect 1B, wherein determining the Rice parameter value for decoding the syntax element for the current transform coefficient based on the sum of absolute values and the shift value comprises: normalizing the sum of absolute values using the shift value to generate a normalized sum of absolute values; determining an initial Rice parameter value based on the normalized sum of absolute values; and adding the shift value to the initial Rice parameter value to determine the Rice parameter value.

[0193] Aspect 7B - The method of Aspect 6B, further comprising: clipping the normalized sum of absolute values.

[0194] Aspect 8B - The method of Aspect 6B, wherein determining the initial Rice parameter based on the normalized sum of absolute values comprises: determining the initial Rice parameter value using the normalized sum of absolute values as an input to a lookup table.

[0195] Aspect 9B - The method of Aspect 1B, wherein decoding the syntax element for the current transform coefficient using the Rice parameter value comprises: inverse binarizing the syntax element based on the Rice parameter.

[0196] Aspect 10B - The method of Aspect 1B, further comprising: reconstructing a residual block based at least in part on the syntax element; reconstructing a picture based at least in part on the residual block; and displaying the picture.

[0197] Aspect 11B - An apparatus configured to decode video data, the apparatus comprising: a memory configured to store video data; and one or more processors implemented in circuitry and in communication with the memory, the one or more processors configured to: determine a sum of absolute values of neighboring transform coefficients of a current transform coefficient; determine a shift value based on the sum of absolute values; determine a Rice parameter value for decoding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value; and decode the syntax element for the current transform coefficient using the Rice parameter value.

[0198] Aspect 12B - The apparatus of Aspect 11B, wherein to determine the shift value based on the sum of absolute values, the one or more processors are further configured

to: determine a range id from the sum of absolute values; and determine the shift value from the range id.

[0199] Aspect 13B - The apparatus of Aspect 12B, wherein to determine the range id from the sum of absolute values, the one or more processors are further configured to: compare the sum of absolute values to an array of thresholds; and determine the range id based on which two thresholds of the array of thresholds the sum of absolute values falls between.

[0200] Aspect 14B - The apparatus of Aspect 13B, wherein to compare the sum of absolute values to the array of thresholds, the one or more processors are further configured to: compare the sum of absolute values to all thresholds of the array of thresholds in parallel.

[0201] Aspect 15B - The apparatus of Aspect 13B, wherein to determine the shift value from the range id, the one or more processors are further configured to: determine the shift value using the range id as an input to an array of scale factors.

[0202] Aspect 16B - The apparatus of Aspect 11B, wherein to determine the Rice parameter value for decoding the syntax element for the current transform coefficient based on the sum of absolute values and the shift value, the one or more processors are further configured to: normalize the sum of absolute values using the shift value to generate a normalized sum of absolute values; determine an initial Rice parameter value based on the normalized sum of absolute values; and add the shift value to the initial Rice parameter value to determine the Rice parameter value.

[0203] Aspect 17B - The apparatus of Aspect 16B, wherein the one or more processors are further configured to: clip the normalized sum of absolute values.

[0204] Aspect 18B - The apparatus of Aspect 16B, wherein to determine the initial Rice parameter based on the normalized sum of absolute values, the one or more processors are further configured to: determine the initial Rice parameter value using the normalized sum of absolute values as an input to a lookup table.

[0205] Aspect 19B - The apparatus of Aspect 11B, wherein to decode the syntax element for the current transform coefficient using the Rice parameter value, the one or more processors are further configured to: inverse binarize the syntax element based on the Rice parameter.

[0206] Aspect 20B - The apparatus of Aspect 11B, wherein the one or more processors are further configured to: reconstruct a residual block based at least in part on the syntax

element; reconstruct a picture based at least in part on the residual block; and display the picture.

[0207] Aspect 21B - An apparatus configured to decode video data, the apparatus comprising: means for determining a sum of absolute values of neighboring transform coefficients of a current transform coefficient; means for determining a shift value based on the sum of absolute values; means for determining a Rice parameter value for decoding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value; and means for decoding the syntax element for the current transform coefficient using the Rice parameter value.

[0208] Aspect 22B - A non-transitory computer-readable storage medium storing instructions that, when executed, cause one or more processors configured to decode video data to: determine a sum of absolute values of neighboring transform coefficients of a current transform coefficient; determine a shift value based on the sum of absolute values; determine a Rice parameter value for decoding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value; and decode the syntax element for the current transform coefficient using the Rice parameter value.

[0209] Aspect 23B - A method of encoding video data, the method comprising: determining a sum of absolute values of neighboring transform coefficients of a current transform coefficient; determining a shift value based on the sum of absolute values; determining a Rice parameter value for encoding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value; and encoding the syntax element for the current transform coefficient using the Rice parameter value.

[0210] Aspect 24B - The method of Aspect 23B, wherein determining the shift value based on the sum of absolute values comprises: determining a range id from the sum of absolute values; and determining the shift value from the range id.

[0211] Aspect 25B - The method of Aspect 24B, wherein determining the range id from the sum of absolute values comprises: comparing the sum of absolute values to an array of thresholds; and determining the range id based on which two thresholds of the array of thresholds the sum of absolute values falls between.

[0212] Aspect 26B - The method of Aspect 25B, wherein comparing the sum of absolute values to the array of thresholds comprises: comparing the sum of absolute values to all thresholds of the array of thresholds in parallel.

[0213] Aspect 27B - The method of Aspect 25B, wherein determining the shift value from the range id comprises: determining the shift value using the range id as an input to an array of scale factors.

[0214] Aspect 28B - The method of Aspect 23B, wherein determining the Rice parameter value for encoding the syntax element for the current transform coefficient based on the sum of absolute values and the shift value comprises: normalizing the sum of absolute values using the shift value to generate a normalized sum of absolute values; determining an initial Rice parameter value based on the normalized sum of absolute values; and adding the shift value to the initial Rice parameter value to determine the Rice parameter value.

[0215] Aspect 29B - The method of Aspect 28B, further comprising: clipping the normalized sum of absolute values.

[0216] Aspect 30B - The method of Aspect 28B, wherein determining the initial Rice parameter based on the normalized sum of absolute values comprises: determining the initial Rice parameter value using the normalized sum of absolute values as an input to a lookup table.

[0217] Aspect 31B - The method of Aspect 23B, wherein encoding the syntax element for the current transform coefficient using the Rice parameter value comprises: binarizing the syntax element based on the Rice parameter.

[0218] Aspect 32B - The method of Aspect 23B, further comprising: capturing a picture of video data; generating a residual block from the picture of video data; and transforming the residual block to generate a transform block that includes the current transform coefficient.

[0219] Aspect 1C - A method of decoding video data, the method comprising: determining a sum of absolute values of neighboring transform coefficients of a current transform coefficient; determining a shift value based on the sum of absolute values; determining a Rice parameter value for decoding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value; and decoding the syntax element for the current transform coefficient using the Rice parameter value.

[0220] Aspect 2C - The method of Aspect 1C, wherein determining the shift value based on the sum of absolute values comprises: determining a range id from the sum of absolute values; and determining the shift value from the range id.

[0221] Aspect 3C - The method of Aspect 2C, wherein determining the range id from the sum of absolute values comprises: comparing the sum of absolute values to an array of thresholds; and determining the range id based on which two thresholds of the array of thresholds the sum of absolute values falls between.

[0222] Aspect 4C - The method of Aspect 3C, wherein comparing the sum of absolute values to the array of thresholds comprises: comparing the sum of absolute values to all thresholds of the array of thresholds in parallel.

[0223] Aspect 5C - The method of Aspect 3C, wherein determining the shift value from the range id comprises: determining the shift value using the range id as an input to an array of scale factors.

[0224] Aspect 6C - The method of any of Aspects 1C-5C, wherein determining the Rice parameter value for decoding the syntax element for the current transform coefficient based on the sum of absolute values and the shift value comprises: normalizing the sum of absolute values using the shift value to generate a normalized sum of absolute values; determining an initial Rice parameter value based on the normalized sum of absolute values; and adding the shift value to the initial Rice parameter value to determine the Rice parameter value.

[0225] Aspect 7C - The method of Aspect 6C, further comprising: clipping the normalized sum of absolute values.

[0226] Aspect 8C - The method of Aspect 6C, wherein determining the initial Rice parameter based on the normalized sum of absolute values comprises: determining the initial Rice parameter value using the normalized sum of absolute values as an input to a lookup table.

[0227] Aspect 9C - The method of any of Aspects 1C-8C, wherein decoding the syntax element for the current transform coefficient using the Rice parameter value comprises: inverse binarizing the syntax element based on the Rice parameter.

[0228] Aspect 10C - The method of any of Aspects 1C-9C, further comprising: reconstructing a residual block based at least in part on the syntax element; reconstructing a picture based at least in part on the residual block; and displaying the picture.

[0229] It is to be recognized that depending on the example, certain acts or events of any of the techniques described herein can be performed in a different sequence, may be added, merged, or left out altogether (e.g., not all described acts or events are necessary for the practice of the techniques). Moreover, in certain examples, acts or events may

be performed concurrently, e.g., through multi-threaded processing, interrupt processing, or multiple processors, rather than sequentially.

[0230] In one or more examples, the functions described may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored on or transmitted over as one or more instructions or code on a computer-readable medium and executed by a hardware-based processing unit. Computer-readable media may include computer-readable storage media, which corresponds to a tangible medium such as data storage media, or communication media including any medium that facilitates transfer of a computer program from one place to another, e.g., according to a communication protocol. In this manner, computer-readable media generally may correspond to (1) tangible computer-readable storage media which is non-transitory or (2) a communication medium such as a signal or carrier wave. Data storage media may be any available media that can be accessed by one or more computers or one or more processors to retrieve instructions, code and/or data structures for implementation of the techniques described in this disclosure. A computer program product may include a computer-readable medium.

[0231] By way of example, and not limitation, such computer-readable storage media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage, or other magnetic storage devices, flash memory, or any other medium that can be used to store desired program code in the form of instructions or data structures and that can be accessed by a computer. Also, any connection is properly termed a computer-readable medium. For example, if instructions are transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared, radio, and microwave, then the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio, and microwave are included in the definition of medium. It should be understood, however, that computer-readable storage media and data storage media do not include connections, carrier waves, signals, or other transitory media, but are instead directed to non-transitory, tangible storage media. Disk and disc, as used herein, includes compact disc (CD), laser disc, optical disc, digital versatile disc (DVD), floppy disk and Blu-ray disc, where disks usually reproduce data magnetically, while discs reproduce data optically with lasers. Combinations of the above should also be included within the scope of computer-readable media.

[0232] Instructions may be executed by one or more processors, such as one or more DSPs, general purpose microprocessors, ASICs, FPGAs, or other equivalent integrated or discrete logic circuitry. Accordingly, the terms "processor" and "processing circuitry," as used herein may refer to any of the foregoing structures or any other structure suitable for implementation of the techniques described herein. In addition, in some aspects, the functionality described herein may be provided within dedicated hardware and/or software modules configured for encoding and decoding, or incorporated in a combined codec. Also, the techniques could be fully implemented in one or more circuits or logic elements.

[0233] The techniques of this disclosure may be implemented in a wide variety of devices or apparatuses, including a wireless handset, an integrated circuit (IC) or a set of ICs (e.g., a chip set). Various components, modules, or units are described in this disclosure to emphasize functional aspects of devices configured to perform the disclosed techniques, but do not necessarily require realization by different hardware units. Rather, as described above, various units may be combined in a codec hardware unit or provided by a collection of interoperative hardware units, including one or more processors as described above, in conjunction with suitable software and/or firmware.

[0234] Various examples have been described. These and other examples are within the scope of the following claims.

WHAT IS CLAIMED IS:

1. A method of decoding video data, the method comprising:

determining a sum of absolute values of neighboring transform coefficients of a current transform coefficient;

determining a shift value based on the sum of absolute values;

determining a Rice parameter value for decoding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value; and

decoding the syntax element for the current transform coefficient using the Rice parameter value.

2. The method of claim 1, wherein determining the shift value based on the sum of absolute values comprises:

determining a range id from the sum of absolute values; and

determining the shift value from the range id.

3. The method of claim 2, wherein determining the range id from the sum of absolute values comprises:

comparing the sum of absolute values to an array of thresholds; and

determining the range id based on which two thresholds of the array of thresholds the sum of absolute values falls between.

4. The method of claim 3, wherein comparing the sum of absolute values to the array of thresholds comprises:

comparing the sum of absolute values to all thresholds of the array of thresholds in parallel.

5. The method of claim 3, wherein determining the shift value from the range id comprises:

determining the shift value using the range id as an input to an array of scale factors.

6. The method of claim 1, wherein determining the Rice parameter value for decoding the syntax element for the current transform coefficient based on the sum of absolute values and the shift value comprises:

normalizing the sum of absolute values using the shift value to generate a normalized sum of absolute values;

determining an initial Rice parameter value based on the normalized sum of absolute values; and

adding the shift value to the initial Rice parameter value to determine the Rice parameter value.

7. The method of claim 6, further comprising:

clipping the normalized sum of absolute values.

8. The method of claim 6, wherein determining the initial Rice parameter based on the normalized sum of absolute values comprises:

determining the initial Rice parameter value using the normalized sum of absolute values as an input to a lookup table.

9. The method of claim 1, wherein decoding the syntax element for the current transform coefficient using the Rice parameter value comprises:

inverse binarizing the syntax element based on the Rice parameter.

10. The method of claim 1, further comprising:

reconstructing a residual block based at least in part on the syntax element;

reconstructing a picture based at least in part on the residual block; and

displaying the picture.

11. An apparatus configured to decode video data, the apparatus comprising:

a memory configured to store video data; and

one or more processors implemented in circuitry and in communication with the memory, the one or more processors configured to:

determine a sum of absolute values of neighboring transform coefficients of a current transform coefficient;

determine a shift value based on the sum of absolute values;

determine a Rice parameter value for decoding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value; and

decode the syntax element for the current transform coefficient using the Rice parameter value.

12. The apparatus of claim 11, wherein to determine the shift value based on the sum of absolute values, the one or more processors are further configured to:

determine a range id from the sum of absolute values; and

determine the shift value from the range id.

13. The apparatus of claim 12, wherein to determine the range id from the sum of absolute values, the one or more processors are further configured to:

compare the sum of absolute values to an array of thresholds; and

determine the range id based on which two thresholds of the array of thresholds the sum of absolute values falls between.

14. The apparatus of claim 13, wherein to compare the sum of absolute values to the array of thresholds, the one or more processors are further configured to:

compare the sum of absolute values to all thresholds of the array of thresholds in parallel.

15. The apparatus of claim 13, wherein to determine the shift value from the range id, the one or more processors are further configured to:

determine the shift value using the range id as an input to an array of scale factors.

16. The apparatus of claim 11, wherein to determine the Rice parameter value for decoding the syntax element for the current transform coefficient based on the sum of absolute values and the shift value, the one or more processors are further configured to:

normalize the sum of absolute values using the shift value to generate a normalized sum of absolute values;

determine an initial Rice parameter value based on the normalized sum of absolute values; and

add the shift value to the initial Rice parameter value to determine the Rice parameter value.

17. The apparatus of claim 16, wherein the one or more processors are further configured to:

clip the normalized sum of absolute values.

18. The apparatus of claim 16, wherein to determine the initial Rice parameter based on the normalized sum of absolute values, the one or more processors are further configured to:

determine the initial Rice parameter value using the normalized sum of absolute values as an input to a lookup table.

19. The apparatus of claim 11, wherein to decode the syntax element for the current transform coefficient using the Rice parameter value, the one or more processors are further configured to:

inverse binarize the syntax element based on the Rice parameter.

20. The apparatus of claim 11, wherein the one or more processors are further configured to:

reconstruct a residual block based at least in part on the syntax element;

reconstruct a picture based at least in part on the residual block; and

display the picture.

21. An apparatus configured to decode video data, the apparatus comprising:

means for determining a sum of absolute values of neighboring transform coefficients of a current transform coefficient;

means for determining a shift value based on the sum of absolute values;

means for determining a Rice parameter value for decoding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value; and

means for decoding the syntax element for the current transform coefficient using the Rice parameter value.

22. A non-transitory computer-readable storage medium storing instructions that, when executed, cause one or more processors configured to decode video data to:

determine a sum of absolute values of neighboring transform coefficients of a current transform coefficient;

determine a shift value based on the sum of absolute values;

determine a Rice parameter value for decoding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value; and

decode the syntax element for the current transform coefficient using the Rice parameter value.

23. A method of encoding video data, the method comprising:

determining a sum of absolute values of neighboring transform coefficients of a current transform coefficient;

determining a shift value based on the sum of absolute values;

determining a Rice parameter value for encoding a syntax element for the current transform coefficient based on the sum of absolute values and the shift value; and

encoding the syntax element for the current transform coefficient using the Rice parameter value.

24. The method of claim 23, wherein determining the shift value based on the sum of absolute values comprises:

determining a range id from the sum of absolute values; and

determining the shift value from the range id.

25. The method of claim 24, wherein determining the range id from the sum of absolute values comprises:

comparing the sum of absolute values to an array of thresholds; and

determining the range id based on which two thresholds of the array of thresholds the sum of absolute values falls between.

26. The method of claim 25, wherein comparing the sum of absolute values to the array of thresholds comprises:

comparing the sum of absolute values to all thresholds of the array of thresholds in parallel.

27. The method of claim 25, wherein determining the shift value from the range id comprises:

determining the shift value using the range id as an input to an array of scale factors.

28. The method of claim 23, wherein determining the Rice parameter value for encoding the syntax element for the current transform coefficient based on the sum of absolute values and the shift value comprises:

normalizing the sum of absolute values using the shift value to generate a normalized sum of absolute values;

determining an initial Rice parameter value based on the normalized sum of absolute values; and

adding the shift value to the initial Rice parameter value to determine the Rice parameter value.

29. The method of claim 28, further comprising:

clipping the normalized sum of absolute values.

30. The method of claim 28, wherein determining the initial Rice parameter based on the normalized sum of absolute values comprises:

determining the initial Rice parameter value using the normalized sum of absolute values as an input to a lookup table.

31. The method of claim 23, wherein encoding the syntax element for the current transform coefficient using the Rice parameter value comprises:

binarizing the syntax element based on the Rice parameter.

32. The method of claim 23, further comprising:

capturing a picture of video data;

generating a residual block from the picture of video data; and

transforming the residual block to generate a transform block that includes the current transform coefficient.

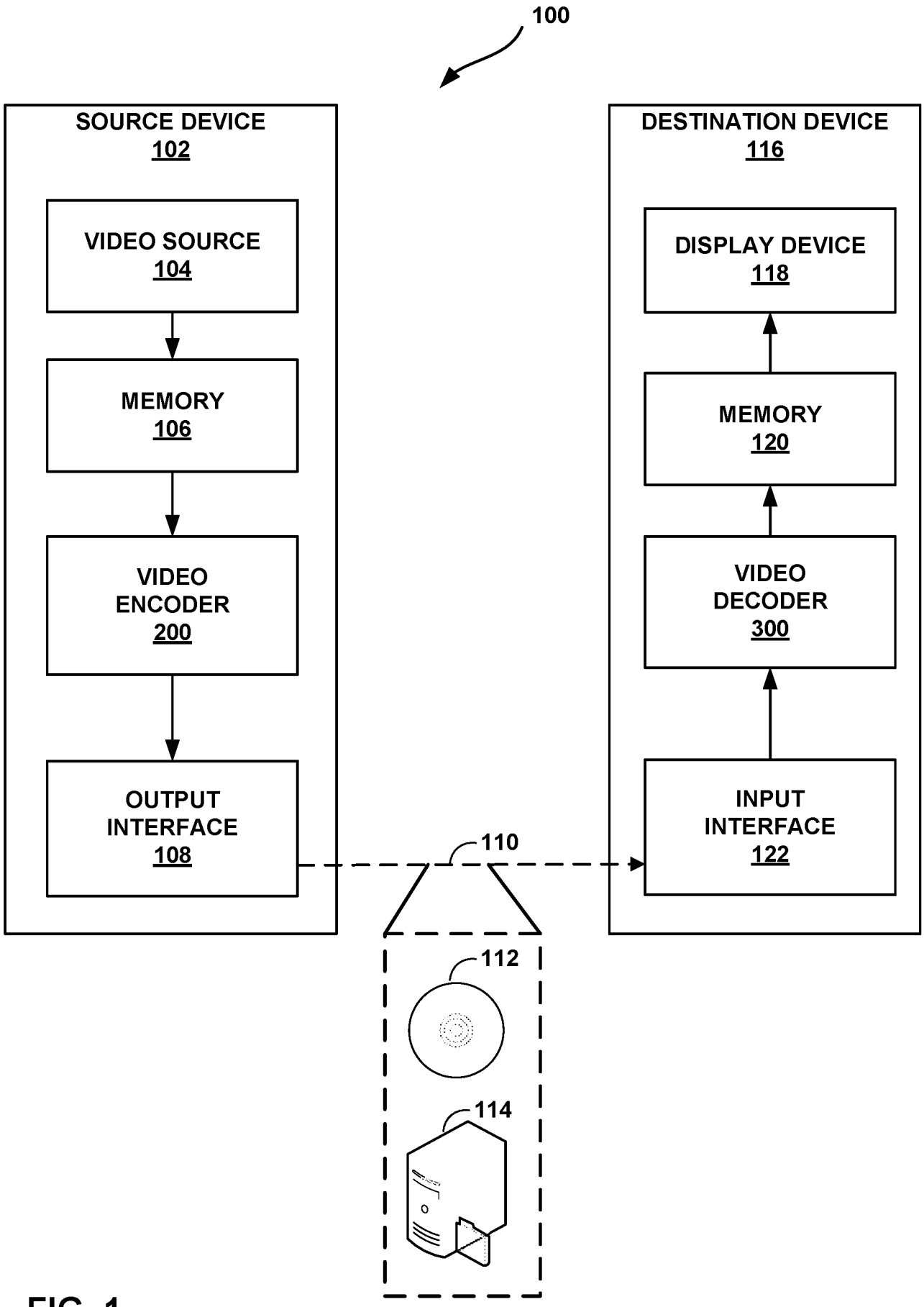

100
SOURCE DEVICE
102
VIDEO SOURCE
104
MEMORY
106
VIDEO
ENCODER
200
OUTPUT
INTERFACE
108
DESTINATION DEVICE
116
DISPLAY DEVICE
118
MEMORY
120
VIDEO
DECODER
300
INPUT
INTERFACE
122
110
112
114

FIG. 1

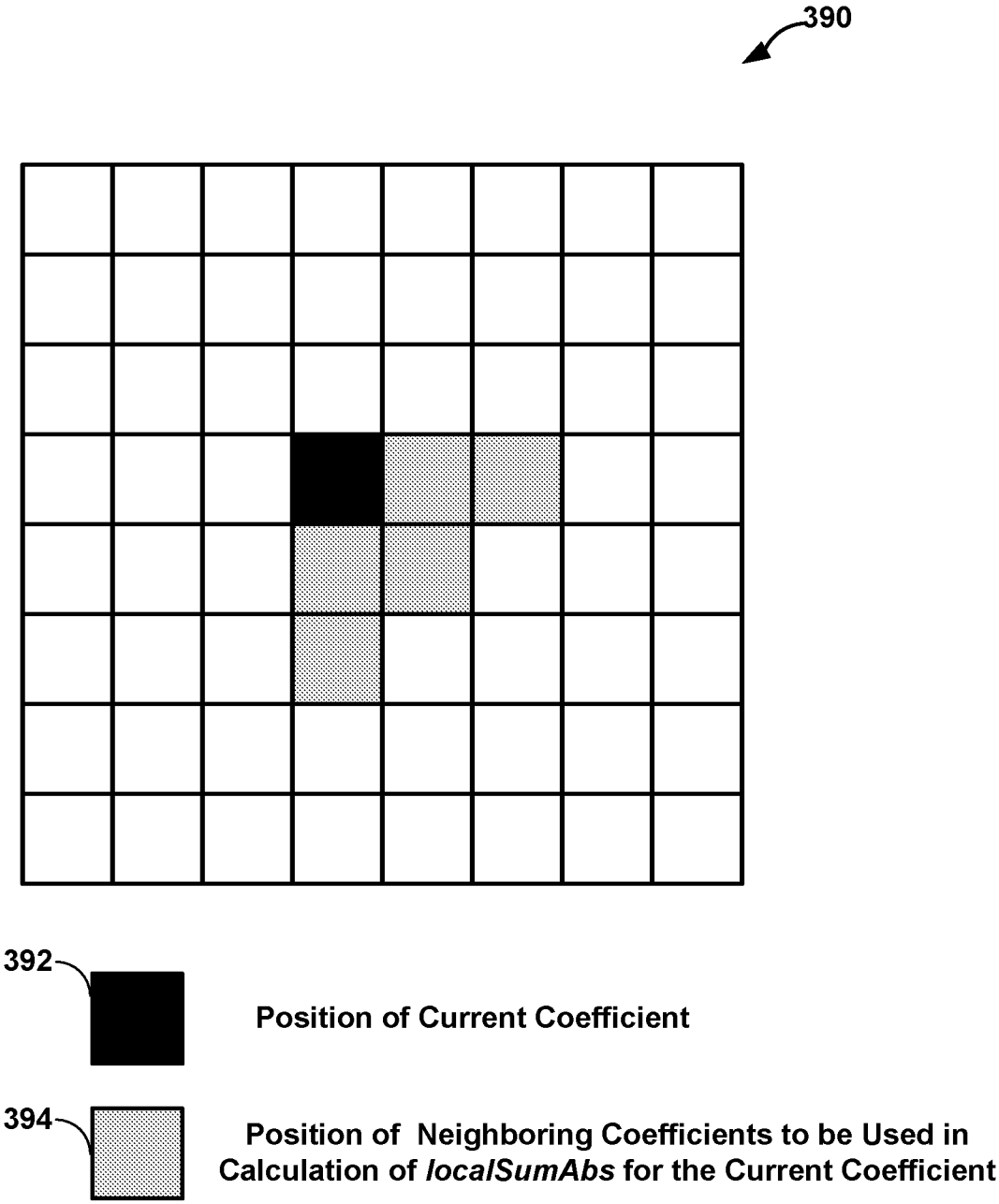

390
392
Position of Current Coefficient
394
Position of Neighboring Coefficients to be Used in Calculation of *localSumAbs* for the Current Coefficient

FIG. 2

Table 1. Look-up table for Rice Parameter based on locSumAbs in VVC

locSumAbs	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cRiceParam	0	0	0	0	0	0	0	1	1	1	1	1	1	1	2	2
locSumAbs	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
cRiceParam	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3

FIG. 3

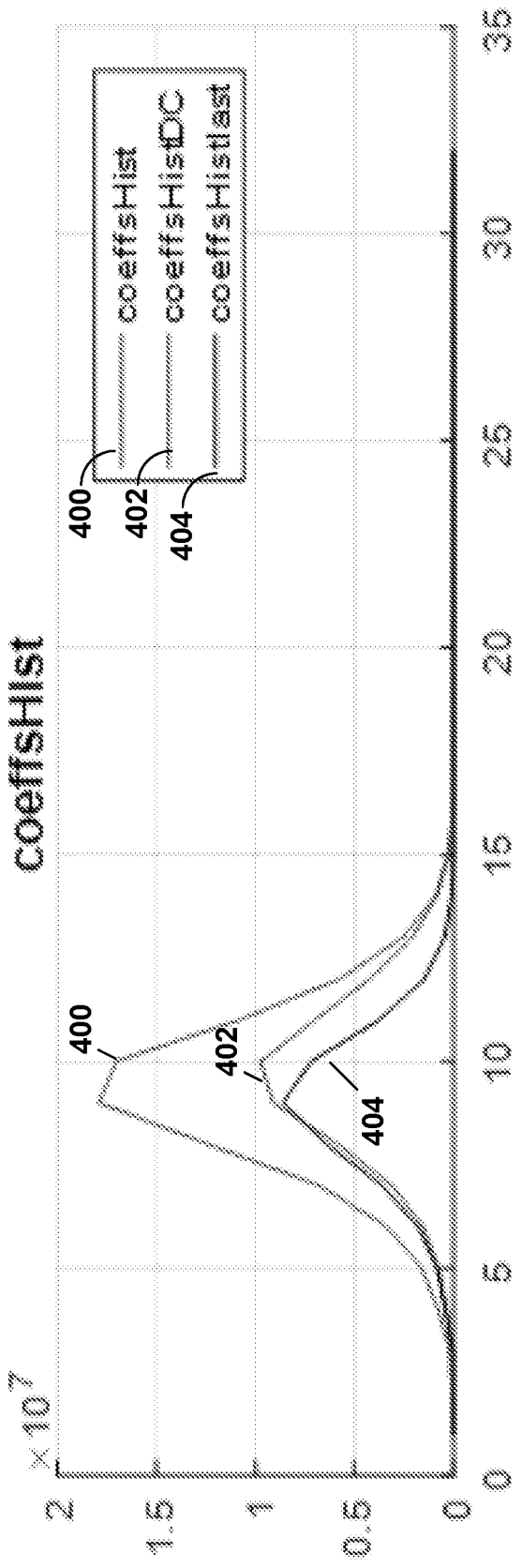

coeffsHist
coeffsHist
coeffsHistDC
coeffsHistlast
400
402
404
×10^7
2
1.5
1
0.5
0
0
5
10
15
20
25
30
35

FIG. 4

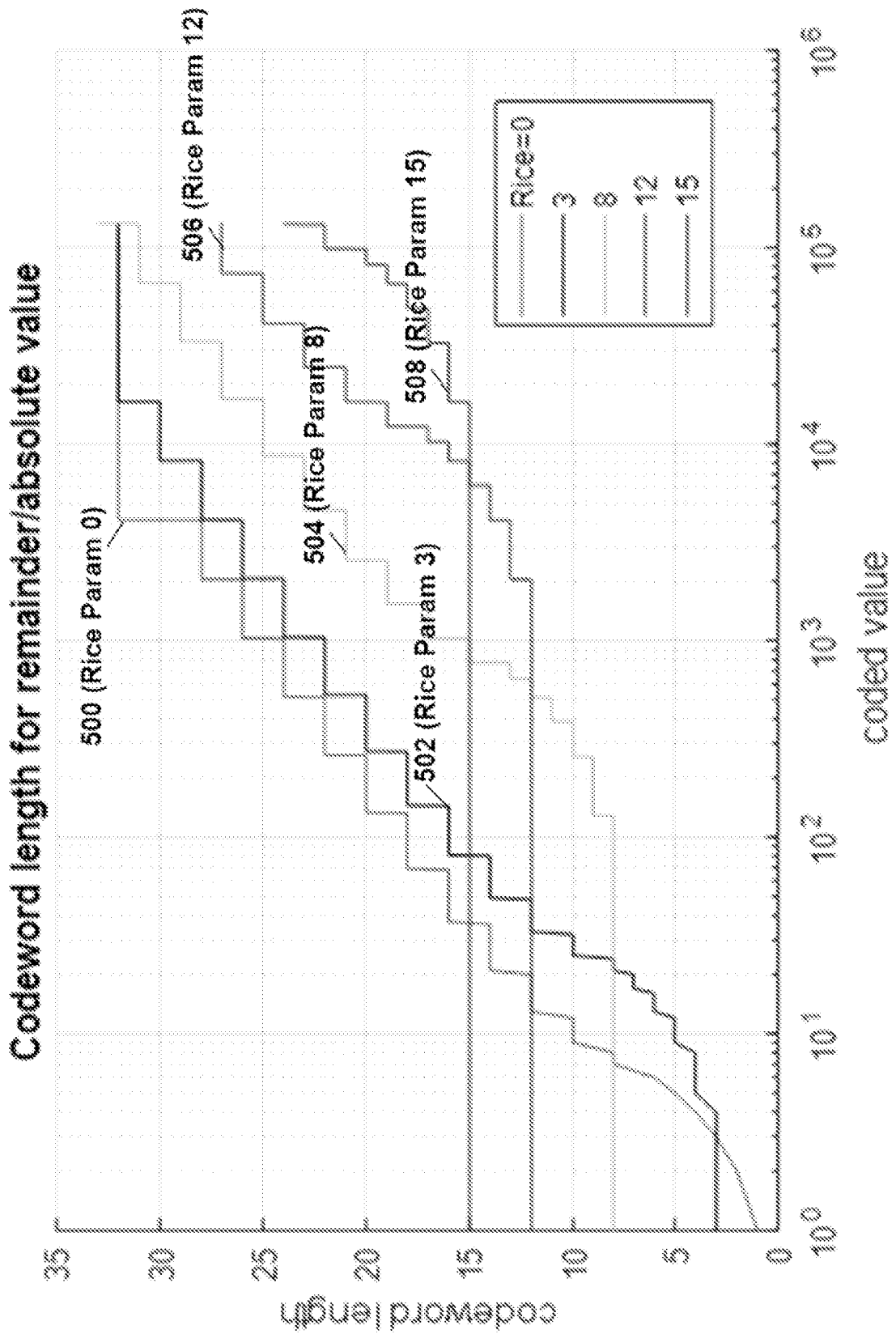

Codeword length for remainder/absolute value
500 (Rice Param 0)
502 (Rice Param 3)
504 (Rice Param 8)
506 (Rice Param 12)
508 (Rice Param 15)
Rice=0
3
8
12
15
codeword length
coded value
0
5
10
15
20
25
30
35
10^0
10^1
10^2
10^3
10^4
10^5
10^6

FIG. 5

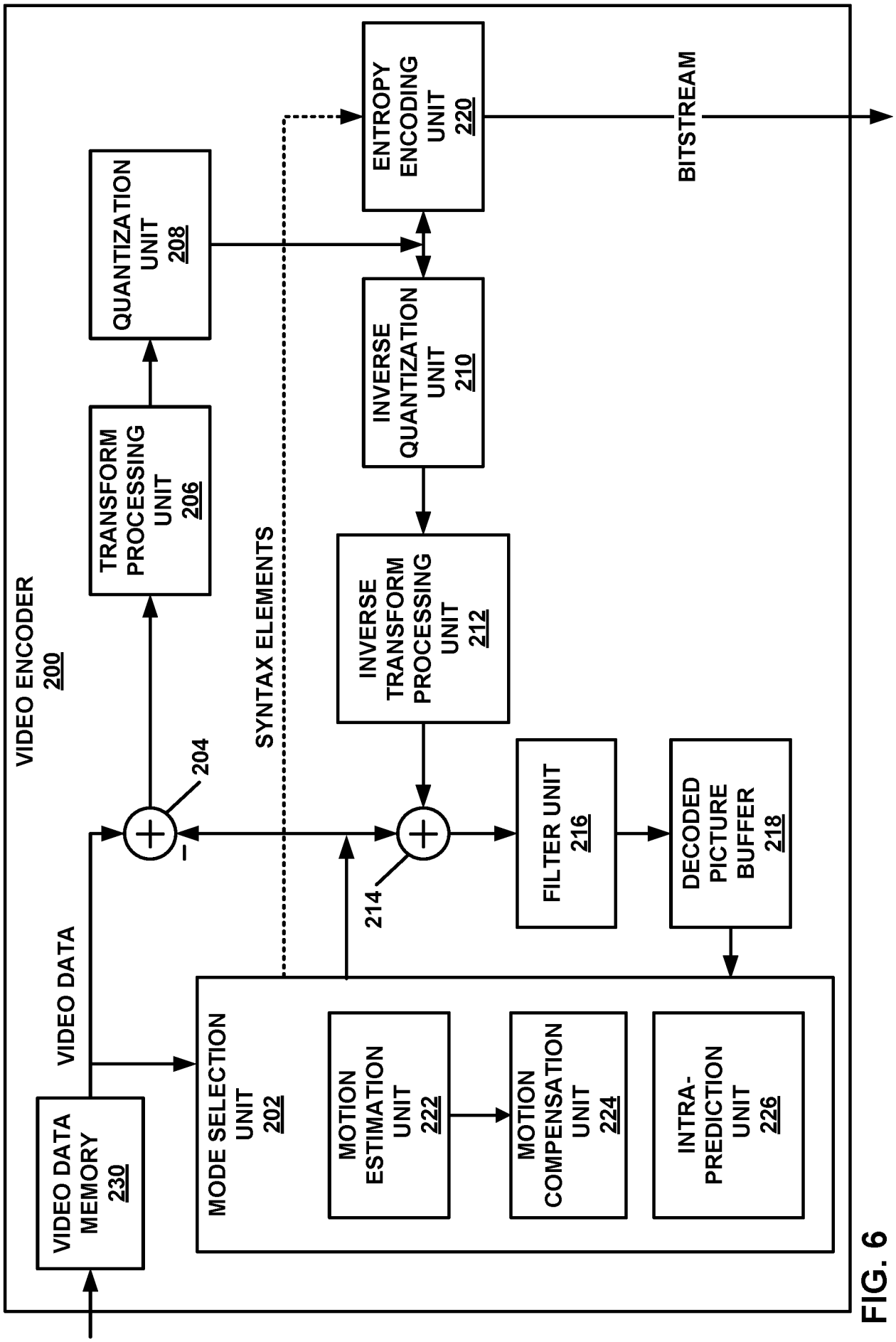

VIDEO ENCODER
200
VIDEO DATA MEMORY
230
VIDEO DATA
MODE SELECTION UNIT
202
MOTION ESTIMATION UNIT
222
MOTION COMPENSATION UNIT
224
INTRA-PREDICTION UNIT
226
204
214
TRANSFORM PROCESSING UNIT
206
QUANTIZATION UNIT
208
INVERSE QUANTIZATION UNIT
210
INVERSE TRANSFORM PROCESSING UNIT
212
FILTER UNIT
216
DECODED PICTURE BUFFER
218
SYNTAX ELEMENTS
ENTROPY ENCODING UNIT
220
BITSTREAM

FIG. 6

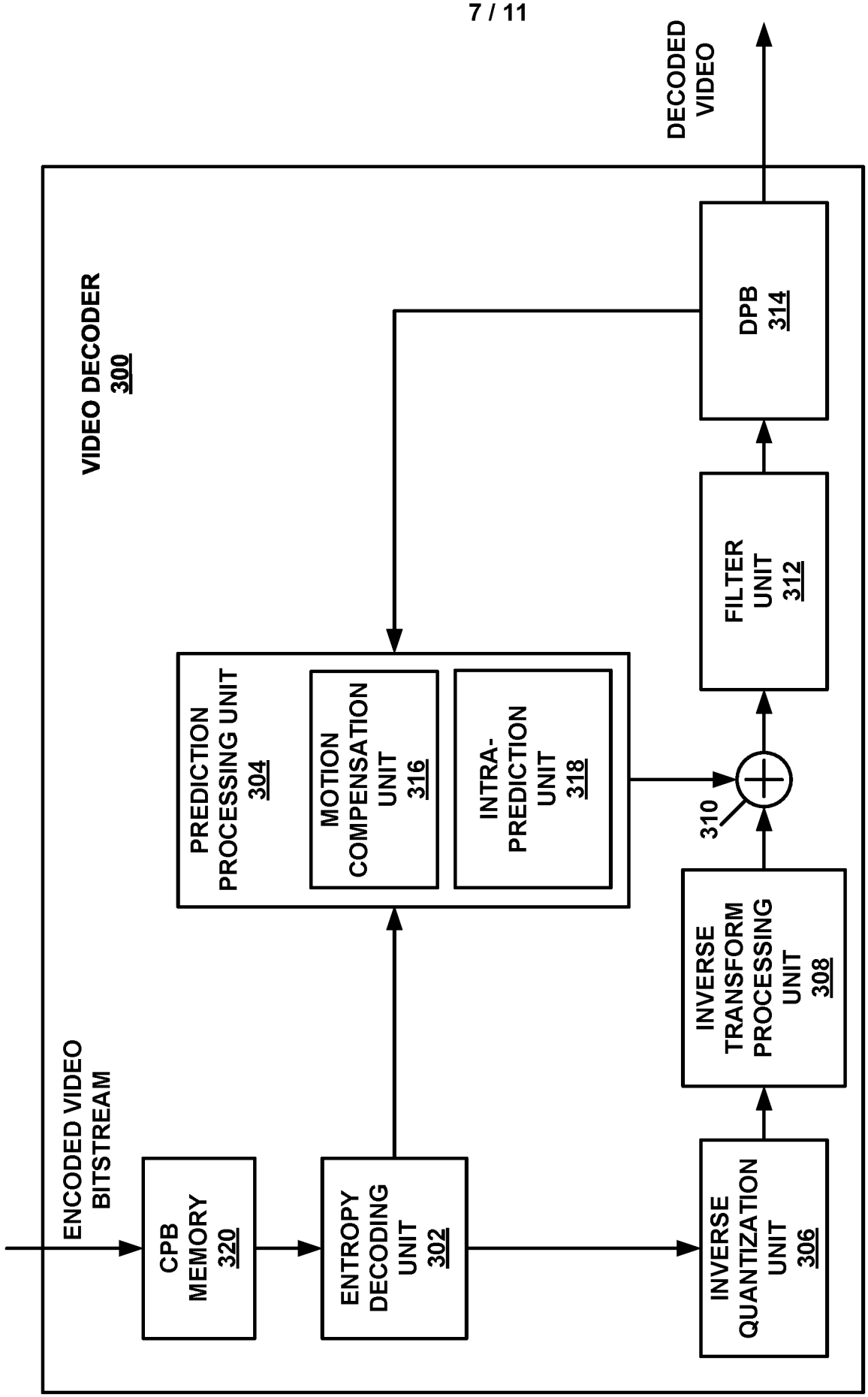

ENCODED VIDEO BITSTREAM
CPB MEMORY 320
ENTROPY DECODING UNIT 302
INVERSE QUANTIZATION UNIT 306
INVERSE TRANSFORM PROCESSING UNIT 308
310
PREDICTION PROCESSING UNIT 304
MOTION COMPENSATION UNIT 316
INTRA-PREDICTION UNIT 318
FILTER UNIT 312
DPB 314
VIDEO DECODER 300
DECODED VIDEO

FIG. 7

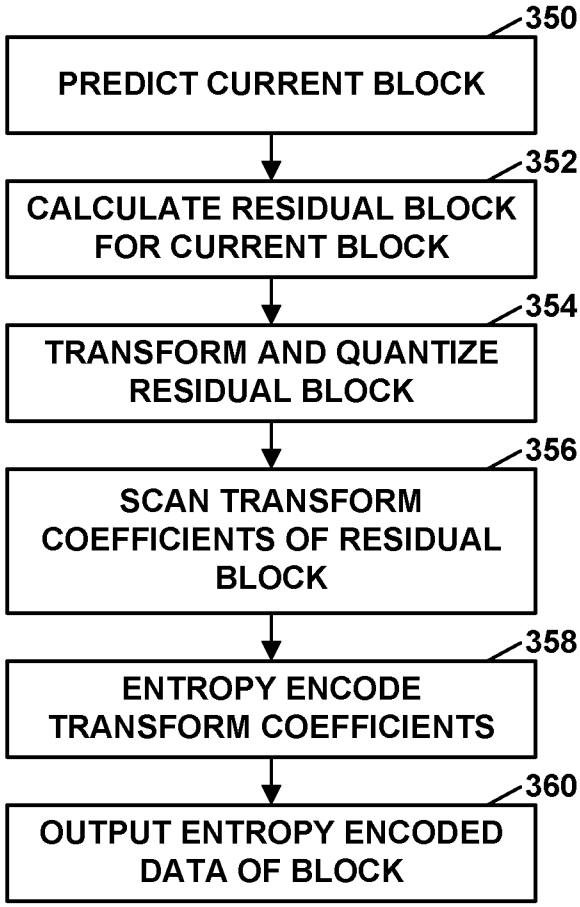

350
PREDICT CURRENT BLOCK
352
CALCULATE RESIDUAL BLOCK FOR CURRENT BLOCK
354
TRANSFORM AND QUANTIZE RESIDUAL BLOCK
356
SCAN TRANSFORM COEFFICIENTS OF RESIDUAL BLOCK
358
ENTROPY ENCODE TRANSFORM COEFFICIENTS
360
OUTPUT ENTROPY ENCODED DATA OF BLOCK

FIG. 8

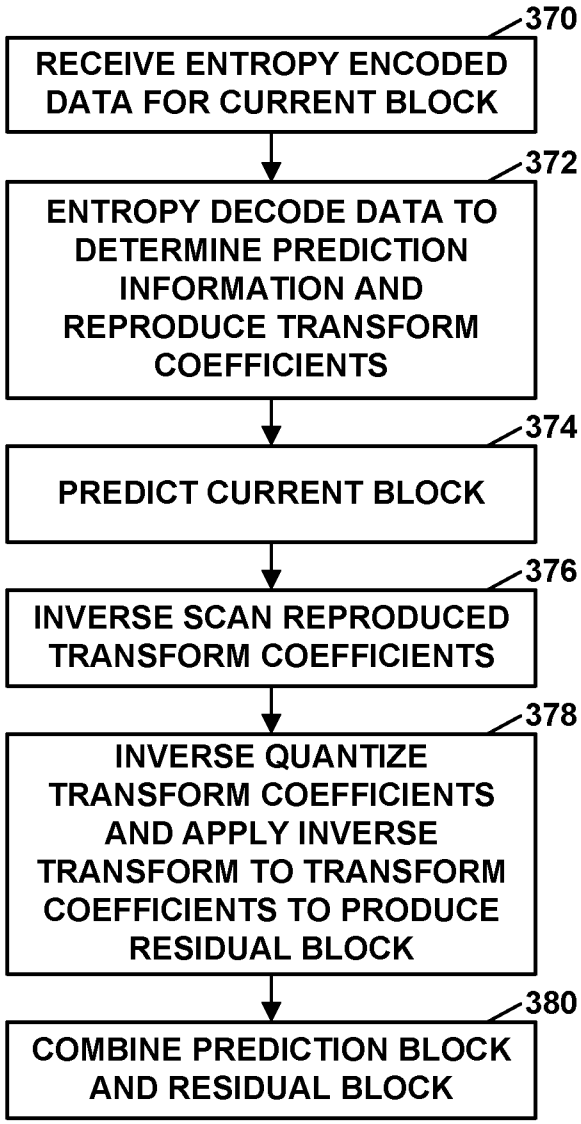

370
RECEIVE ENTROPY ENCODED DATA FOR CURRENT BLOCK
372
ENTROPY DECODE DATA TO DETERMINE PREDICTION INFORMATION AND REPRODUCE TRANSFORM COEFFICIENTS
374
PREDICT CURRENT BLOCK
376
INVERSE SCAN REPRODUCED TRANSFORM COEFFICIENTS
378
INVERSE QUANTIZE TRANSFORM COEFFICIENTS AND APPLY INVERSE TRANSFORM TO TRANSFORM COEFFICIENTS TO PRODUCE RESIDUAL BLOCK
380
COMBINE PREDICTION BLOCK AND RESIDUAL BLOCK

FIG. 9

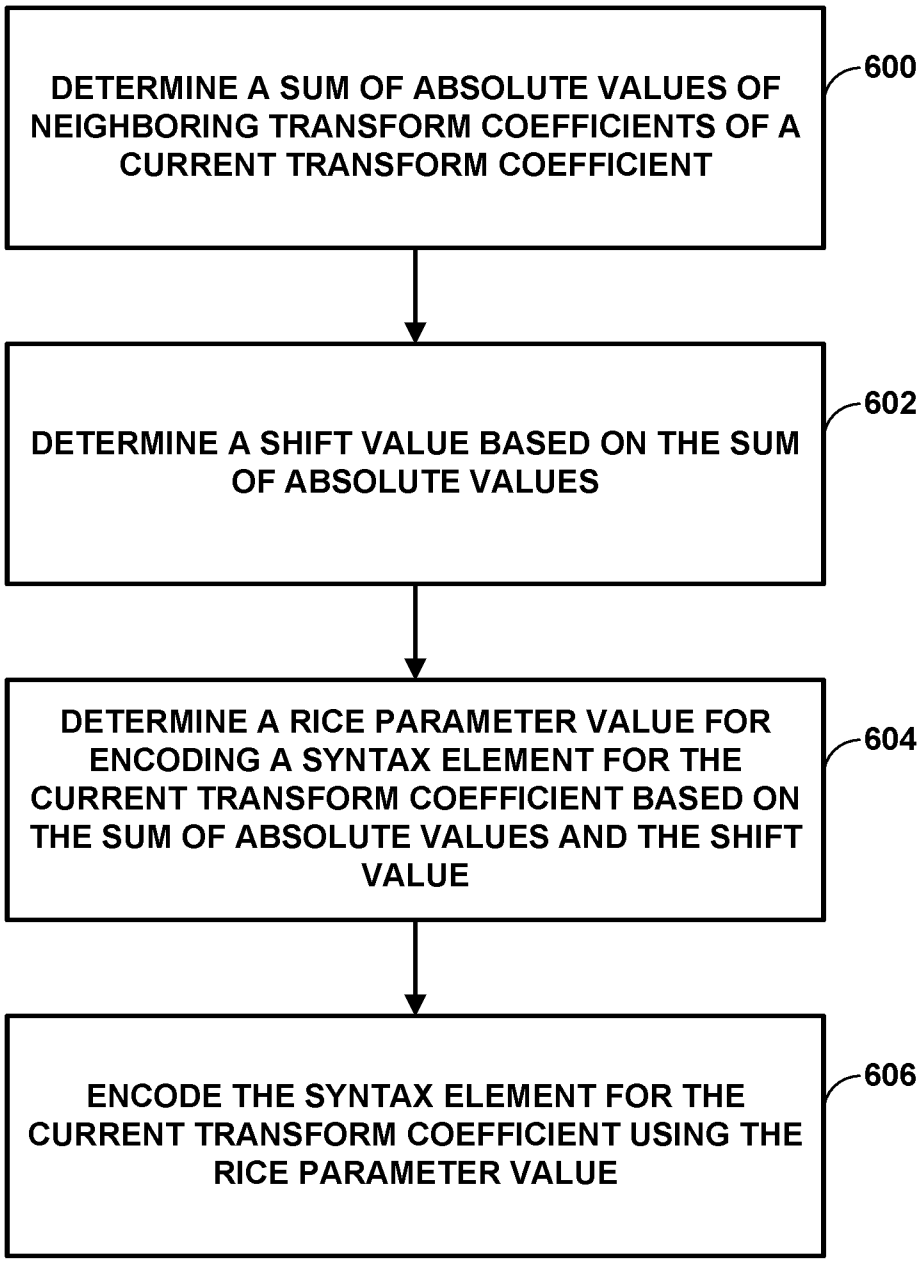

DETERMINE A SUM OF ABSOLUTE VALUES OF NEIGHBORING TRANSFORM COEFFICIENTS OF A CURRENT TRANSFORM COEFFICIENT
600
DETERMINE A SHIFT VALUE BASED ON THE SUM OF ABSOLUTE VALUES
602
DETERMINE A RICE PARAMETER VALUE FOR ENCODING A SYNTAX ELEMENT FOR THE CURRENT TRANSFORM COEFFICIENT BASED ON THE SUM OF ABSOLUTE VALUES AND THE SHIFT VALUE
604
ENCODE THE SYNTAX ELEMENT FOR THE CURRENT TRANSFORM COEFFICIENT USING THE RICE PARAMETER VALUE
606

FIG. 10

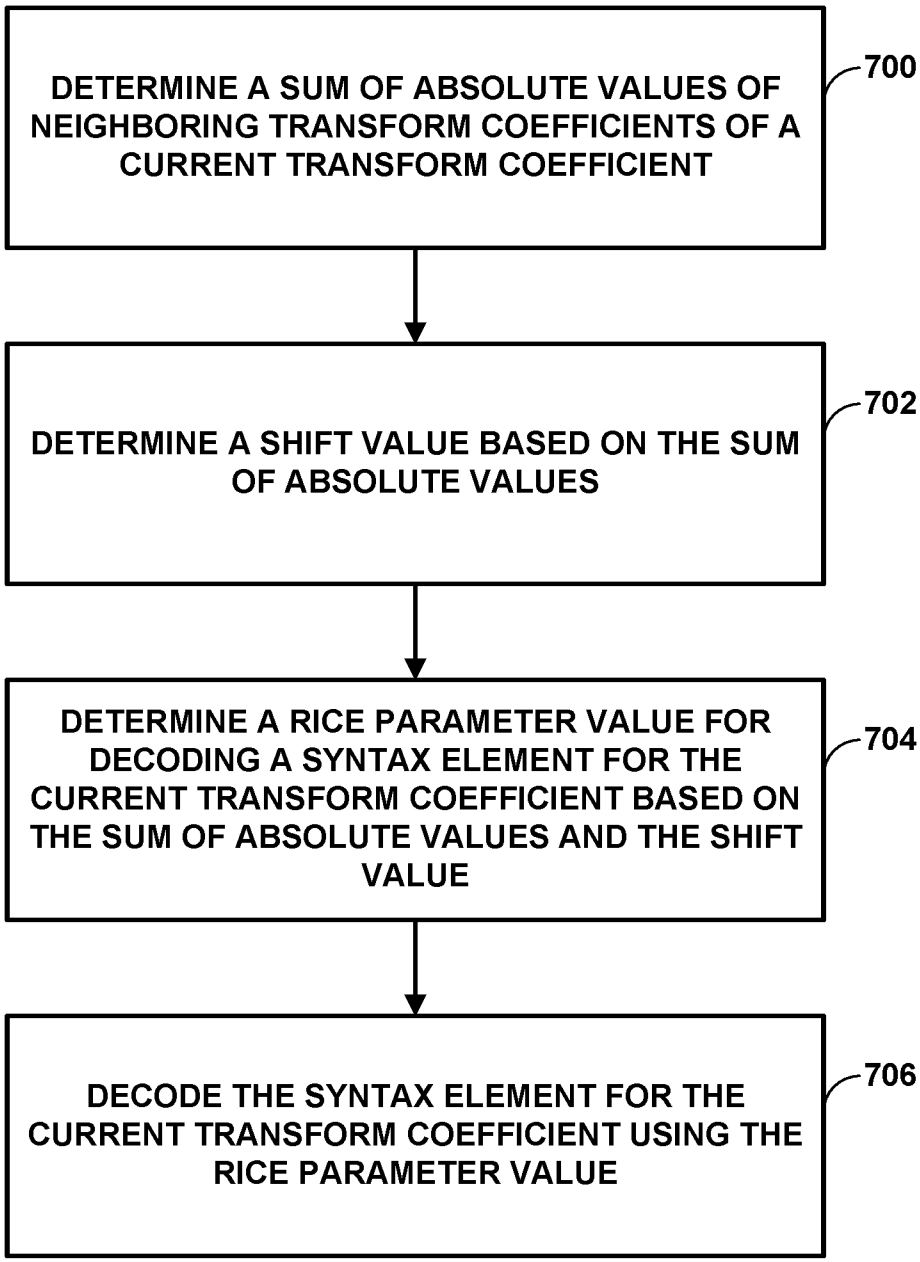

DETERMINE A SUM OF ABSOLUTE VALUES OF NEIGHBORING TRANSFORM COEFFICIENTS OF A CURRENT TRANSFORM COEFFICIENT
700
DETERMINE A SHIFT VALUE BASED ON THE SUM OF ABSOLUTE VALUES
702
DETERMINE A RICE PARAMETER VALUE FOR DECODING A SYNTAX ELEMENT FOR THE CURRENT TRANSFORM COEFFICIENT BASED ON THE SUM OF ABSOLUTE VALUES AND THE SHIFT VALUE
704
DECODE THE SYNTAX ELEMENT FOR THE CURRENT TRANSFORM COEFFICIENT USING THE RICE PARAMETER VALUE
706

FIG. 11

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2022/071053

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04N19/13 H04N19/154 H04N19/18 H04N19/91
ADD.

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04N

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	RUSANOVSKYY (QUALCOMM) D ET AL: "CE-1.1 and CE-1.2: On the Rice parameter derivation for high bit-depth coding", 21. JVET MEETING; 20210106 - 20210115; TELECONFERENCE; (THE JOINT VIDEO EXPLORATION TEAM OF ISO/IEC JTC1/SC29/WG11 AND ITU-T SG.16), , no. JVET-U0064 1 January 2021 (2021-01-01), XP030293128, Retrieved from the Internet: URL:https://jvet-experts.org/doc_end_user/documents/21_Teleconference/wg11/JVET-U0064-v2.zip JVET-U0064.docx [retrieved on 2021-01-01] section 2 ----- -/--	1-32

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
10 June 2022	22/06/2022

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Montoneri, Fabio

1

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2022/071053

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 2021/040492 A1 (LG ELECTRONICS INC [KR]) 4 March 2021 (2021-03-04) paragraphs [0253], [0256] - [0258], [0289] - [0293]	1-32
X	KAWAMURA (KDDI) K ET AL: "CE related: On Rice Parameter Derivation with Content Adaptation", 21. JVET MEETING; 20210106 - 20210115; TELECONFERENCE; (THE JOINT VIDEO EXPLORATION TEAM OF ISO/IEC JTC1/SC29/WG11 AND ITU-T SG.16), , no. JVET-U0062 29 December 2020 (2020-12-29), XP030293120, Retrieved from the Internet: URL:https://jvet-experts.org/doc_end_user/documents/21_Teleconference/wg11/JVET-U0062-v1.zip JVET-U0062.docx [retrieved on 2020-12-29] section 2	1-32
X	BROWNE (SONY) A ET AL: "CE-1.4, CE-1.5, CE-2.2 and CE-2.3: Rice parameter selection for high bit depths", 21. JVET MEETING; 20210106 - 20210115; TELECONFERENCE; (THE JOINT VIDEO EXPLORATION TEAM OF ISO/IEC JTC1/SC29/WG11 AND ITU-T SG.16), , no. JVET-U0057 5 January 2021 (2021-01-05), XP030293111, Retrieved from the Internet: URL:https://jvet-experts.org/doc_end_user/documents/21_Teleconference/wg11/JVET-U0057-v2.zip JVET-U0057/JVET-U0057_v2_clean.docx [retrieved on 2021-01-05] sections 2-2-2.4	1-32

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2022/071053

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2021040492 A1	04-03-2021	CN 114586356 A	03-06-2022
		EP 4024869 A1	06-07-2022
		KR 20220051341 A	26-04-2022
		WO 2021040492 A1	04-03-2021