

SCANS AND LAST COEFFICIENT POSITION CODING FOR ZERO-OUT TRANSFORMS

[0001] This application claims priority to U.S. Patent Application No. 16/655,950, filed October 17, 2019, which claims the benefit of U.S. Provisional Application No. 62/747,543, filed October 18, 2018, U.S. Provisional Application No. 62/786,285, filed December 28, 2018, U.S. Provisional Application No. 62/788,546, filed January 4, 2019, and U.S. Provisional Application No. 62/792,795, filed January 15, 2019, the entire content of each of which is incorporated by reference herein.

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 transform coefficient coding for video coding. In some examples, this disclosure describes a transform coefficient scan technique and a last non-zero coefficient position coding technique for large transforms that use zero-out techniques for high frequency coefficients. When using a zero-out technique, a video coder may be configured to set all transform coefficients in a certain region of a transform unit to have a value of zero. For example, for a 64x64 transform unit, the video coder may keep the values in a 32x32 region in the upper left corner of the transform unit (e.g., the lowest frequency components) with no change. However, the video coder will set all transform coefficients outside this region (i.e., the zero-out region) to have a value of zero.

[0006] In one example, this disclosure describes a scanning technique that includes only scanning transform coefficients in a transform unit that are outside the zero-out region (i.e., that are in the non zero-out region). In this way, scanning of transform coefficients known to have a value of zero may be avoided, thus increasing coder throughput.

[0007] In addition, this disclosure describes techniques of coding a position of a last non-zero transform coefficient in a transform unit that has a zero-out region. For example, this disclosure describes techniques where a video coder determines a context (e.g., a probability model) for coding a syntax element that indicates a position of a last non-zero transform coefficient. In one example, the context is determined based on the entire size of the transform unit, and not based on the size of non zero-out region of the transform unit. Even though the position of the last non-zero transform coefficient is guaranteed to be in the non zero-out region (e.g., the upper-left 32x32 region of a 64x64 transform unit), the context of coding the position of the last non-zero transform coefficient is more closely related to statistics of a 64x64 transform unit, not a 32x32 transform unit. Thus, coding efficiency of the position of the last non-zero transform coefficient may be increased.

[0008] In one example, a method includes determining a first region of a transform unit in which transform coefficients are subject to zero-out, determining a second region of

the transform unit in which transform coefficients are not subject to zero-out, and scanning only the second region of the transform unit.

[0009] In another example, a device includes a memory and one or more processors in communication with the memory, the one or more processors configured to determine a first region of a transform unit in which transform coefficients are subject to zero-out, determine a second region of the transform unit in which transform coefficients are not subject to zero-out, and scan only the second region of the transform unit.

[0010] In another example, a device includes means for determining a first region of a transform unit in which transform coefficients are subject to zero-out, means for determining a second region of the transform unit in which transform coefficients are not subject to zero-out, and means for scanning only the second region of the transform unit.

[0011] In another example, a computer-readable storage medium is encoded with instructions that, when executed, cause a programmable processor to determine a first region of a transform unit in which transform coefficients are subject to zero-out, determine a second region of the transform unit in which transform coefficients are not subject to zero-out, and scan only the second region of the transform unit.

[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] FIGS. 2A and 2B are conceptual diagrams illustrating an example quadtree binary tree (QTBT) structure, and a corresponding coding tree unit (CTU).

[0015] FIG. 3 is a block diagram illustrating an example video encoder that may perform the techniques of this disclosure.

[0016] FIG. 4 is a block diagram illustrating an example video decoder that may perform the techniques of this disclosure.

[0017] FIG. 5 is a conceptual diagram illustrating example transform unit scans including example scans according to the techniques of this disclosure.

[0018] FIG. 6 is a flowchart illustrating an example encoding method according to the techniques of this disclosure.

[0019] FIG. 7 is a flowchart illustrating an example decoding method according to the techniques of this disclosure.

DETAILED DESCRIPTION

[0020] FIG. 1 is a block diagram illustrating an example video encoding and decoding system 100 that may perform the techniques of this disclosure for scanning and coding zero-out transform units. 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, uncoded video, encoded video, decoded (e.g., reconstructed) video, and video metadata, such as signaling data.

[0021] 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 include any of a wide range of devices, including desktop computers, notebook (i.e., laptop) computers, tablet computers, set-top boxes, telephone handsets such as smart phones, televisions, cameras, display devices, digital media players, video gaming consoles, video streaming 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.

[0022] 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 transform coefficient coding. 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.

[0023] 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 transform coefficient coding. 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 include 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.

[0024] In general, video source 104 represents a source of video data (i.e., raw, uncoded 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.

[0025] 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 memories 106, 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.

[0026] 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 some examples, 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 include any wireless or wired communication medium, such as a radio frequency (RF) spectrum or one or more physical transmission lines. The communication medium may form part of a packet-based network, such as a local area network, a wide area network, or a global network such as the Internet. The communication medium may include routers, switches, base stations, or any other equipment that may be useful to facilitate communication from source device 102 to destination device 116.

[0027] In some examples, computer-readable medium 110 may include storage device 112. 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 hard drives, 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.

[0028] In some examples, computer-readable medium 110 may include file server 114 or another intermediate storage device that may store the encoded video data generated by source device 102. Source device 102 may output encoded video data to file server 114. Destination device 116 may access stored video data from file server 114 via streaming or download. 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 File Transfer Protocol (FTP) server, a content delivery network device, or a network attached storage (NAS) device. 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. File server 114 and input interface 122 may be configured to operate according to a streaming transmission protocol, a download transmission protocol, or a combination thereof.

[0029] 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 include 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 and input interface 122 include a wireless transmitter and/or wireless receiver, 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.

[0030] 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.

[0031] 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 from computer-readable medium 110 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 cathode ray tube (CRT), a liquid crystal display (LCD), a plasma display, an organic light emitting diode (OLED) display, or another type of display device.

[0032] 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).

[0033] 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 include an integrated circuit, a microprocessor, and/or a wireless communication device, such as a cellular telephone.

[0034] In general, this disclosure describes techniques for transform coefficient coding for video coding. In some examples, this disclosure describes a transform coefficient scan technique and last non-zero coefficient position coding for large transforms that use zero-out techniques for high frequency coefficients for complexity reduction. That

is, in some examples, a video coder (e.g., video encoder 200 and/or video decoder 300) may be configured to set all transform coefficients in a certain region of a transform unit to have a value of zero. For example, for a 64x64 transform unit, the video coder may keep the values in a 32x32 region in the upper left corner of the transform unit (e.g., the lowest frequency components) as is. However, the video coder will set all transform coefficients outside this region (i.e., the zero-out region) to have a value of zero.

[0035] In one example, this disclosure describes a scanning technique that includes only scanning transform coefficients in a transform unit that are outside the zero-out region. In this way, scanning of transform coefficients known to have a value of zero may be avoided, thus increasing coder throughput.

[0036] In addition, this disclosure describes techniques of coding a position of a last non-zero transform coefficient in a transform unit that has a zero-out region. For example, this disclosure describes techniques where a video coder determines a context (e.g., a probability model) for coding a syntax element that indicates a position of a last non-zero transform coefficient. In one example, the context is determined based on the entire size of the transform unit, and not based on the size of non zero-out region of the transform unit. Even though the position of the last non-zero transform coefficient is guaranteed to be in the non zero-out region (e.g., the upper-left 32x32 region of a 64x64 transform coefficient), the context of coding the position of the last non-zero transform coefficient is more closely related to statistics of a 64x64 transform unit, not a 32x32 transform unit. Thus, coding efficiency of the position of the last non-zero transform coefficient may be increased.

[0037] As will be explained in more detail below, video encoder 200 and video decoder 300 may be configured to determine a first region of a transform unit in which transform coefficients are subject to zero-out, determine a second region of the transform unit in which transform coefficients are not subject to zero-out, and scan only the second region of the transform unit.

[0038] 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 the Joint Exploration Test Model (JEM) and/or H.266/VVC (Versatile Video Coding). The techniques of this disclosure, however, are not limited to any particular coding standard.

The techniques of this disclosure may be applicable to encoding and decoding a variety of types of video data. In particular, video data in which blocks of data are predicted, a difference is calculated, and the difference or residual is transformed using one of a number of transforms to produce transform coefficients. The transform coefficients are then scanned. The video coding may involve such techniques according to a variety of different standards.

[0039] An early draft for new video coding standard, referred to as the H.266/Versatile Video Coding (VVC) standard, is available in the document JVET-J1001 “Versatile Video Coding (Draft 1)” by Benjamin Bross, and its algorithm description is available in the document JVET-J1002 “Algorithm description for Versatile Video Coding and Test Model 1 (VTM 1)” by Jianle Chen and Elena Alshina.

[0040] 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.

[0041] 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.

[0042] 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.

[0043] As another example, video encoder 200 and video decoder 300 may be configured to operate according to JEM or VVC. According to JEM or 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).

[0044] In an MTT partitioning structure, blocks may be partitioned using a quadtree (QT) partition, a binary tree (BT) partition, and/or 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.

[0045] 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).

[0046] 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, MTT partitioning, or other types of partitioning as well.

[0047] 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.

[0048] 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.

[0049] 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.

[0050] 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.

[0051] 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.

[0052] 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.

[0053] Some examples of JEM and 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.

[0054] To perform intra-prediction, video encoder 200 may select an intra-prediction mode to generate the prediction block. Some examples of JEM and 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).

[0055] 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 an

advanced motion vector prediction (AMVP) mode or a merge mode. Video encoder 200 may use similar modes to encode motion vectors for affine motion compensation mode.

[0056] 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.

[0057] 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.

[0058] 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.

[0059] 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.

[0060] 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.

[0061] 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.

[0062] 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 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.

[0063] 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.

[0064] This disclosure may generally refer to “signaling” certain information, such as syntax elements. The term “signaling” may generally refer to the communication of values of 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.

[0065] FIGS. 2A and 2B are conceptual diagrams illustrating an example quadtree binary tree (QTBT) structure 130, and a corresponding coding tree unit (CTU) 132. The solid lines represent quadtree splitting, and dotted lines indicate binary tree splitting. In each split (i.e., non-leaf) node of the binary tree, one flag is signaled to indicate which splitting type (i.e., horizontal or vertical) is used, where 0 indicates horizontal splitting and 1 indicates vertical splitting in this example. For the quadtree splitting, there is no need to indicate the splitting type, because quadtree nodes split a block horizontally and vertically into 4 sub-blocks with equal size. Accordingly, video encoder 200 may encode, and video decoder 300 may decode, syntax elements (such as splitting information) for a region tree level of QTBT structure 130 (i.e., the solid lines) and syntax elements (such as splitting information) for a prediction tree level of QTBT structure 130 (i.e., the dashed lines). Video encoder 200 may encode, and video decoder 300 may decode, video data, such as prediction and transform data, for CUs represented by terminal leaf nodes of QTBT structure 130.

[0066] In general, CTU 132 of FIG. 2B may be associated with parameters defining sizes of blocks corresponding to nodes of QTBT structure 130 at the first and second levels. As stated above, the first level may be partitioned according to quadtree partitioning, and the second level may be partitioned according to binary tree partitioning. These parameters may include a CTU size (representing a size of CTU 132 in samples), a minimum quadtree size (MinQTSIZE, representing a minimum allowed quadtree leaf node size), a maximum binary tree size (MaxBTSIZE, representing a maximum allowed binary tree root node size), a maximum binary tree depth (MaxBTDepth, representing a maximum allowed binary tree depth), and a minimum

binary tree size (MinBTSize, representing the minimum allowed binary tree leaf node size).

[0067] The root node of a QTBT structure corresponding to a CTU may have four child nodes at the first level of the QTBT structure, each of which may be partitioned according to quadtree partitioning. That is, nodes of the first level are either leaf nodes (having no child nodes) or have four child nodes. The example of QTBT structure 130 represents such nodes as including the parent node and child nodes having solid lines for branches. If nodes of the first level are not larger than the maximum allowed binary tree root node size (MaxBTSize), then the nodes can be further partitioned by respective binary trees. The binary tree splitting of one node can be iterated until the nodes resulting from the split reach the minimum allowed binary tree leaf node size (MinBTSize) or the maximum allowed binary tree depth (MaxBTDepth). The example of QTBT structure 130 represents such nodes as having dashed lines for branches. The binary tree leaf node is referred to as a coding unit (CU), which is used for prediction (e.g., intra-picture or inter-picture prediction) and transform, without any further partitioning. As discussed above, CUs may also be referred to as “video blocks” or “blocks.”

[0068] In one example of the QTBT partitioning structure, the CTU size is set as 128x128 (luma samples and two corresponding 64x64 chroma samples), the MinQTSIZE is set as 16x16, the MaxBTSize is set as 64x64, the MinBTSize (for both width and height) is set as 4, and the MaxBTDepth is set as 4. The quadtree partitioning is applied to the CTU first to generate quad-tree leaf nodes. The quadtree leaf nodes may have a size from 16x16 (i.e., the MinQTSIZE) to 128x128 (i.e., the CTU size). If the leaf quadtree node is 128x128, the leaf quadtree node will not be further split by the binary tree, because the size exceeds the MaxBTSize (i.e., 64x64, in this example). Otherwise, the leaf quadtree node will be further partitioned by the binary tree. Therefore, the quadtree leaf node is also the root node for the binary tree and has the binary tree depth as 0. When the binary tree depth reaches MaxBTDepth (4, in this example), no further splitting is permitted. When the binary tree node has a width equal to MinBTSize (4, in this example), it implies no further horizontal splitting is permitted. Similarly, a binary tree node having a height equal to MinBTSize implies no further vertical splitting is permitted for that binary tree node. As noted above, leaf nodes of the binary tree are referred to as CUs, and are further processed according to prediction and transform without further partitioning.

[0069] FIG. 3 is a block diagram illustrating an example video encoder 200 that may perform the techniques of this disclosure. FIG. 3 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 in the context of video coding standards such as the HEVC video coding standard and the H.266/VVC video coding standard in development. However, the techniques of this disclosure are not limited to these video coding standards, and are applicable generally to video encoding and decoding.

[0070] In the example of FIG. 3, 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.

[0071] 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.

[0072] In this disclosure, reference to video data memory 230 should not be interpreted as being limited to a memory internal to video encoder 200, unless specifically described as such, or a memory external to video encoder 200, unless specifically described as such. Rather, reference to video data memory 230 should be understood as a 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.

[0073] The various units of FIG. 3 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 causes 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, the one or more units may be integrated circuits.

[0074] 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.

[0075] 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.

[0076] Mode selection unit 202 includes a motion estimation unit 222, 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.

[0077] 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.

[0078] 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, MTT 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.”

[0079] 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.

[0080] Motion estimation unit 222 may form one or more motion vectors (MVs) that define 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.

[0081] 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.

[0082] Mode selection unit 202 provides the prediction block to residual generation unit 204. Residual generation unit 204 receives a raw, uncoded 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.

[0083] 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 $2N \times 2N$, video encoder 200 may support PU sizes of $2N \times 2N$ or $N \times N$ for intra prediction, and symmetric PU sizes of $2N \times 2N$, $2N \times N$, $N \times 2N$, $N \times N$, or similar for inter prediction. Video encoder 20 and video decoder 30 may also support asymmetric partitioning for PU sizes of $2N \times nU$, $2N \times nD$, $nL \times 2N$, and $nR \times 2N$ for inter prediction.

[0084] 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 $2N \times 2N$, $2N \times N$, or $N \times 2N$.

[0085] For other video coding techniques such as an intra-block copy mode coding, an affine-mode coding, and linear model (LM) mode coding, as a few 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.

[0086] 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.

[0087] 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.

[0088] 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.

[0089] 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.

[0090] 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.

[0091] Video encoder 200 stores reconstructed blocks in DPB 218. For instance, in examples where operations of filter unit 216 are not needed, reconstruction unit 214 may store reconstructed blocks to DPB 218. In examples where operations of filter unit 216 are needed, 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.

[0092] 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.

[0093] When entropy encoding syntax elements representing transform coefficients, entropy encoding unit 220 may be configured to scan through the transform coefficients in a predetermined fashion. In accordance with examples of this disclosure, as will be described in more detail below, entropy encoding unit 220 may be configured to perform a transform coefficient scan technique and last non-zero coefficient position coding technique for large transforms that use zero-out techniques for high frequency coefficients for complexity reduction. That is, in some examples, entropy encoding unit 220 and/or transform processing unit 206 may be configured to set all transform coefficients in a certain region of a transform unit to have a value of zero. For example, for a 64x64 transform unit, entropy encoding unit 220 and/or transform processing unit 206 may keep the values in a 32x32 region in the upper left corner of the transform unit (e.g., the lowest frequency components) as is. However, entropy encoding unit 220 and/or transform processing unit 206 will set all transform coefficients outside this region (i.e., the zero-out region) to have a value of zero.

[0094] In one example, this disclosure describes a scanning technique where entropy encoding unit 220 is configured to only scan transform coefficients in a transform unit that are outside the zero-out region. In this way, scanning of transform coefficients known to have a value of zero may be avoided, thus increasing coder throughput.

[0095] In addition, this disclosure describes techniques of encoding a position of a last non-zero transform coefficient in a transform unit that has a zero-out region. For example, this disclosure describes techniques where entropy encoding unit 220 determines a context (e.g., a probability model) for encoding a syntax element that indicates a position of a last non-zero transform coefficient. In one example, entropy encoding unit 220 determines the context based on the entire size of the transform unit, and not based on the size of non zero-out region of the transform unit. Even though the position of the last non-zero transform coefficient is guaranteed to be in the non zero-out region (e.g., the upper-left 32x32 region of a 64x64 transform coefficient), the context of encoding the position of the last non-zero transform coefficient is more

closely related to statistics of a 64x64 transform unit, not a 32x32 transform unit. Thus, encoding efficiency of the position of the last non-zero transform coefficient may be increased.

[0096] As will be explained in more detail below, in one example of the disclosure, entropy encoding unit 220 may be configured to determine a first region of a transform unit in which transform coefficients are subject to zero-out, determine a second region of the transform unit in which transform coefficients are not subject to zero-out, and scan only the second region of the transform unit.

[0097] 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.

[0098] 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.

[0099] 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 blocks and the chroma coding blocks.

[0100] FIG. 4 is a block diagram illustrating an example video decoder 300 that may perform the techniques of this disclosure. FIG. 4 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 JEM and HEVC. However, the techniques of this disclosure may be performed by video coding devices that are configured according to other video coding standards.

[0101] In the example of FIG. 4, 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. 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.

[0102] 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 dynamic random access memory (DRAM), including synchronous DRAM (SDRAM), magnetoresistive RAM (MRAM), resistive RAM (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.

[0103] 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.

[0104] The various units shown in FIG. 4 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. 3, 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 causes 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, the one or more units may be integrated circuits.

[0105] 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.

[0106] 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.

[0107] When entropy decoding encoded syntax elements representing transform coefficients of a transform block, entropy decoding unit 302 may be configured to scan through the transform block in a predetermined fashion. In accordance with examples of this disclosure, as will be described in more detail below, entropy decoding unit 302 may be configured to perform a transform coefficient scan technique and last non-zero coefficient position decoding technique for large transforms that use zero-out techniques for high frequency coefficients for complexity reduction. That is, in some examples, entropy decoding unit 302 and/or inverse transform processing unit 308 may be configured to set all transform coefficients in a certain region of a transform unit to have a value of zero. For example, for a 64x64 transform unit, entropy decoding unit 302 and/or inverse transform processing unit 308 may keep the values in a 32x32 region in the upper left corner of the transform unit (e.g., the lowest frequency components) as is. However, entropy decoding unit 302 and/or inverse transform processing unit 308 will set all transform coefficients outside this region (i.e., the zero-out region) to have a value of zero.

[0108] In one example, this disclosure describes a scanning technique where entropy decoding unit 302 is configured to only scan transform coefficients in a transform unit that are outside the zero-out region. In this way, scanning of transform coefficients known to have a value of zero may be avoided, thus increasing coder throughput.

[0109] In addition, this disclosure describes techniques of decoding one or more syntax elements indicating a position of a last non-zero transform coefficient in a transform unit that has a zero-out region. For example, this disclosure describes techniques where entropy decoding unit 302 determines a context (e.g., a probability model) for decoding one or more syntax elements that indicates a position of a last non-zero transform coefficient. In one example, entropy decoding unit 302 determines the context based on the entire size of the transform unit, and not based on the size of non zero-out region of the transform unit. Even though the position of the last non-zero transform coefficient is guaranteed to be in the non zero-out region (e.g., the upper-left 32x32 region of a 64x64 transform coefficient), the context of decoding one or more syntax elements indicating a position of the last non-zero transform coefficient is more closely related to statistics of a 64x64 transform unit, not a 32x32 transform unit. Thus, decoding efficiency of the position of the last non-zero transform coefficient may be increased.

[0110] As will be explained in more detail below, in one example of the disclosure, entropy decoding unit 302 may be configured to determine a first region of a transform unit in which transform coefficients are subject to zero-out, determine a second region of the transform unit in which transform coefficients are not subject to zero-out, and scan only the second region of the transform unit.

[0111] 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”).

[0112] 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.

[0113] 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.

[0114] 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. 3).

[0115] 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. 3). Intra-prediction unit 318 may retrieve data of neighboring samples to the current block from DPB 314.

[0116] 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.

[0117] 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.

[0118] Video decoder 300 may store the reconstructed blocks in DPB 314. In examples where operations of filter unit 312 are not needed, reconstruction unit 310 may store reconstructed blocks to DPB 314. In examples where operations of filter unit 312 are needed, 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.

[0119] In Bross, et al., Versatile Video Coding (Draft 2), Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29 WG 11, 11th Meeting: Ljubljana, SI, 10-18 July 2018, JVET-K1001, which is available at http://phenix.it-sudparis.eu/jvet/doc_end_user/documents/11_Ljubljana/wg11/JVET-K1001-v7.zip, the largest transform unit size is 64x64. In order to reduce the implementation complexity of transforms, VVC Draft 2 specifies that high frequency coefficients beyond 32 rows and columns (i.e., beyond the first 32 rows and columns of transform coefficients in a transform block starting from a DC coefficient) are set to zero (i.e., are zeroed out), keeping only the coefficients in the lower MxN region ($M \leq 32$, $N \leq 32$), where M is the width of the region in samples and N is the height of the region in samples.

[0120] In VVC Draft 2, the coefficient coding module of video encoder 200 and video decoder 300 (e.g., entropy encoding unit 220 and entropy decoding unit 302) ignores the fact that zero-out happens and treats the large transforms (where zero-out is used) as normal (i.e., non zero-out) transforms. Basically, video encoder 200 and video decoder 300 may unnecessarily handle coding of already known zero-out coefficients (i.e., coefficients whose value are set to zero due to the zero-out process described above). This disclosure addresses this issue by modifying the coefficient scan for zero-out transform coefficients and by modifying the last non-zero coefficient position coding method.

[0121] In VVC, coefficients are scanned from last to first, coefficient group (CG) by coefficient group. Each coefficient group may include a predetermined number of transform coefficients. Coefficients are also scanned within a coefficient group in a reverse diagonal scan (lower left to top right direction). In some examples, video

encoder 200 and video decoder 300 also scan coefficient groups in a reverse diagonal scan (e.g., lower left to top right direction) as shown for blocks 500 (64x64 transform unit) and 506 (32x64 transform unit) in FIG. 5. FIG. 5 is a conceptual diagram illustrating example coefficient group scans. The same scans may be used for transform units. Of course, other scan directions may be used.

[0122] As can be seen from blocks 500 and block 506, all coefficients are scanned, regardless of whether or not they have been subject to zero-out. For example, in block 500, video encoder 200 and video decoder 300 would scan both the non zero-out region 502 and the zero-out region 504. Likewise, for block 506, video encoder 200 and video decoder 300 would scan both non zero-out region 508 and zero-out region 510. Even if the scan starts from the position of the last non-zero transform coefficient, which would be in a non zero-out region, the scans shown for blocks 500 and 506 would still proceed into zero-out regions for many cases. In HEVC and VVC, for transform coefficient coding, the term “last” means the last non-zero coefficient present along the scan path going from DC coefficient (or lowest frequency coefficient) to the higher frequency coefficients. The term “first” means the DC or the lowest frequency coefficient within a coefficient group (scan within a CG) or transform unit (CG scan within a TU).

Coefficient groups may be of size 4x4 or 2x2.

[0123] According to the techniques of this disclosure, in a new coefficient group scan, the scan region is only kept in the non zero-out region, as shown in blocks 512, 518, 524, and 530. Block 512 is a 64x64 transform unit with a scan order that proceeds to the upper right. Block 512 includes a 32x32 non zero-out region 514. The remaining area of block 512 is the zero-out region 516. As described above, all transform coefficients in the zero-out regions are set to have a value of 0.

[0124] Block 524 is a 64x64 transform unit with a scan order that proceeds to the lower left. Block 524 includes a 32x32 non zero-out region 526. The remaining area of block 524 is the zero-out region 528.

[0125] Block 518 is a 32x64 transform unit with a scan order that proceeds to the upper right. Block 518 includes a 32x32 non zero-out region 520. The remaining area of block 518 is the zero-out region 522.

[0126] Block 530 is a 32x64 transform unit with a scan order that proceeds to the lower left. Block 530 includes a 32x32 non zero-out region 532. The remaining area of block 530 is the zero-out region 534.

[0127] As can be seen from blocks 512, 518, 524, and 530 of FIG. 3, regardless of the dimensions of the transform units and the scan direction, video encoder 200 and video decoder 300 are configured to only perform a scan within the non zero-out regions (e.g., regions 514, 520, 526, and 532) of each respective transform unit.

[0128] That is, video encoder 200 and video decoder 300 may be configured to determine which transform coefficients in a block/CU/TU have been subjected to the zero-out process. If a transform coefficient has been subject to the zero-out process, video encoder 200 and video decoder 300 will not scan the transform coefficient in the zero-out regions. If a transform coefficient has not been subject to the zero-out process (e.g., is in a non zero-out region), video encoder 200 and video decoder 300 do scan that transform coefficient. This scan method eliminates scanning of zero-out coefficients. In some examples, the scan within a CG is not altered.

[0129] Accordingly, in one example of the disclosure, video encoder 200 and video decoder 300 are configured to determine a first region of a transform unit (zero-out region) in which transform coefficients are subject to zero-out. Video encoder 200 and video decoder 300 may be further configured to determine a second region of the transform unit (non zero-out region) in which transform coefficients are not subject to zero-out. Video encoder 200 and video decoder 300 may then scan only the second region of the transform unit (non zero-out region).

[0130] For coding of a last non-zero coefficient position in the X and Y directions (e.g., an (X,Y) coordinate in the transform unit), VVC employs a similar scheme to HEVC, except that VVC extends the scheme to larger transform units. In HEVC, the largest transform unit is 32x32. The last non-zero coefficient position may be coded as a truncated unary coded prefix code (e.g., CABAC regular mode) followed by a fixed length coded suffix code (bypass coded).

[0131] For coding of the last non-zero coefficient position of zero-out transform units (e.g., larger transform units subject to zero-out), video encoder 200 and video decoder 300 may be configured to set the maximum transform unit dimension to a zero-out threshold (e.g., 32) instead of 64, thus shortening the prefix code length for last coefficient position to the range 24-31. Basically, the coding of the last non-zero coefficient position in transform units of dimension larger than 32 (e.g., one example zero-out threshold) would follow the rule for coding of the last coefficient position in transform units of dimension 32 (e.g., 32x32) instead of 64 (e.g., 64x64). Of course, the

threshold for apply zero-out to transform coefficients may be variable and may take on values other than 32.

[0132] As such, in accordance with the techniques of this disclosure, video encoder 200 and video decoder 300 may be configured to encode/decode a position of a last non-zero coefficient in the transform unit based on a threshold for performing a zero-out process. In one example, to encode/decode the position of the last non-zero coefficient in the transform unit based on the threshold for performing the zero-out process, video encoder 200 and video decoder 300 may be configured to encode/decode a prefix code that represents a first portion of the position of the last non-zero coefficient based on the threshold for performing the zero-out process, and encode/decode a suffix code that represents a second portion of the position of the last non-zero coefficient.

[0133] In another example for coding the last non-zero coefficient position of a zeroed out transform unit that is maxed out at a specified dimension (e.g., at 32 width or height), video encoder 200 and video decoder 300 may determine the entropy coding context used for coding the last non-zero coefficient position as the context for coding the last non-zero coefficient position for 64 dimension blocks (e.g., 64x64). That is, video encoder 200 and video decoder 300 may determine the context used for coding the position of the last non-zero coefficient based on the size of the transform unit, rather than the size of the non zero-out region. Even though the position of the last non-zero transform coefficient is guaranteed to be in the non zero-out region (e.g., the upper-left 32x32 region of a 64x64 transform unit), the context for coding the position of the last non-zero transform coefficient is more closely related to statistics of a 64x64 transform unit, not a 32x32 transform unit. Thus, coding efficiency of the position of the last non-zero transform coefficient may be increased. Accordingly, in one example of the disclosure, video encoder 200 and video decoder 300 may be configured to determine a context for entropy coding the position of a last non-zero coefficient based on an entire size of the transform unit.

[0134] The context for the last non-zero coefficient position for a regular (non zero-out) 32x32 block and the context for the last non-zero coefficient position of a 32x32 region of a zeroed out 64x64 block may still be separate context sets in that instance and similarly for other greater than 32 dimension transform blocks (e.g., transform blocks with a height or width of 32). In other examples, the context for coding of last non-zero coefficient position can be shared between the max dimension 32 blocks (e.g., transform blocks with a height or width at a maximum of 32) and max dimension 64 blocks (e.g.,

transform blocks with a height or width at a maximum of 64). For example, the context for the last non-zero coefficient position for a regular (non zero-out) 32x32 block and the context for the last non-zero coefficient position of a 32x32 region of a zeroed out 64x64 block may share the same context set.

[0135] In another example, as another alternative to scanning only non zero-out regions, the large scans can be kept as they are, but video encoder 200 and video decoder 300 may be configured to enforce that the encoded bitstream has only 0 coefficient levels in the zeroed out region of the large inverse transforms. In this case the last non-zero coefficient position would be enforced to be in the upper left non zero-out region and still be limited to max non-zero size of 32, as described above.

[0136] One or more techniques above can be extended to any of NxM, MxN, and NxN transform sizes, where N is the transform dimension where zero-out is applied from N/2 to N-1 for x dimension (i.e., horizontal dimension) or y dimension (i.e., vertical dimension), or both x and y dimensions, respectively. Only the coefficients in the non zero-out region would be scanned, and the last non-zero coefficient position coding would be for the size of the non zero-out region. However, as described above, the contexts used would be the contexts for the full size transform unit. For example, if zero-out is applied for 32x32 transforms, only the lower frequency 16x16 region (e.g., the upper left 16x16 region of the 32x32 transform unit) would be kept, and any coefficient outside of that region would be zeroed out. Video encoder 200 and video decoder 300 would only scan the 16x16 non zero-out region and the last non-zero coefficient position coding would use binarization for a 16x16 transform unit. However, video encoder 200 and video decoder 300 would determine CABAC contexts for the position of the last non-zero coefficient based on the 32x32 transform unit example.

[0137] Accordingly, in other examples of the disclosure, to determine the first region of the transform unit in which transform coefficients are subject to zero-out, video encoder 200 and video decoder 300 may be configured to determine the first region of the transform unit in which transform coefficients are subject to zero-out based on positions of the transform coefficients in the transform unit.

[0138] In another example of the disclosure, to determine the first region of the transform unit in which transform coefficients are subject to zero-out based on positions of the transform coefficients in the transform unit, video encoder 200 and video decoder 300 may be configured to determine that the first region includes transform coefficients

in the transform unit that are beyond the first 32 rows or columns of the transform unit are subject to zero-out.

[0139] In another example of the disclosure, to scan only the second region (non zero-out region) of the transform unit, video encoder 200 and video decoder 300 may be configured to scan only the second region of the transform unit starting at the position of the last non-zero coefficient.

[0140] In another example of the disclosure, to scan only the second region of the transform unit, video encoder 200 and video decoder 300 may be configured to scan only the second region of the transform unit coefficient group by coefficient group.

[0141] FIG. 6 is a flowchart illustrating an example method for encoding a current block. The current block may include a current CU. Although described with respect to video encoder 200 (FIGS. 1 and 3), it should be understood that other devices may be configured to perform a method similar to that of FIG. 6.

[0142] 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, uncoded block and the prediction block for the current block. Video encoder 200 may then transform and quantize coefficients of the residual block (354). Next, video encoder 200 may scan the quantized transform coefficients of the residual block (356).

[0143] In accordance with the techniques of this disclosure, to scan the transform coefficients, video encoder 200 may be configured to determine a first region of a transform unit in which transform coefficients are subject to zero-out (362), determine a second region of the transform unit in which transform coefficients are not subject to zero-out (364), and scan only the second region of the transform unit (366). During the scan, or following the scan, video encoder 200 may entropy encode the coefficients (358). Video encoder 200 may also entropy encode prediction information. In accordance with the techniques of this disclosure, video encoder 200 may be configured to determine a context for entropy encoding the position of a last non-zero coefficient based on an entire size of the transform unit. Video encoder 200 may encode the coefficients using CAVLC or CABAC. Video encoder 200 may then output the entropy coded data of the current block (360). For example, video encoder 200 may output

entropy coded data for transform coefficients of a residual block corresponding to the current block.

[0144] FIG. 7 is a flowchart illustrating an example method for decoding a current block of video data. The current block may include a current CU. Although described with respect to video decoder 300 (FIGS. 1 and 4), it should be understood that other devices may be configured to perform a method similar to that of FIG. 7.

[0145] Video decoder 300 may receive entropy coded data for the current block, such as entropy coded prediction information and entropy coded data for coefficients of a residual block corresponding to the current block (370). Video decoder 300 may entropy decode the entropy coded data to determine prediction information for the current block and to reproduce coefficients of the residual block (372). In accordance with the techniques of this disclosure, to reproduce the transform coefficients, video decoder 300 may be configured to determine a first region of a transform unit in which transform coefficients are subject to zero-out (382), determine a second region of the transform unit in which transform coefficients are not subject to zero-out (384), and scan only the second region of the transform unit (386). In accordance with the techniques of this disclosure, video decoder 300 may be further configured to determine a context for entropy decoding one or more syntax elements indicating a position of a last non-zero coefficient based on an entire size of the transform unit.

[0146] 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 coefficients (376), to create a block of quantized transform coefficients. Video decoder 300 may then inverse quantize and inverse transform the 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).

[0147] 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.

[0148] 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.

[0149] By way of example, and not limitation, such computer-readable storage media can include one or more of 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, include compact discs (CDs), laser discs, optical discs, digital versatile discs (DVDs), floppy disks and Blu-ray discs, 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.

[0150] Instructions may be executed by one or more processors, such as one or more digital signal processors (DSPs), general purpose microprocessors, application specific

integrated circuits (ASICs), field programmable gate arrays (FPGAs), or other equivalent integrated or discrete logic circuitry. Accordingly, the term “processor,” 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.

[0151] 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.

[0152] Various examples have been described. These and other examples are within the scope of the following claims.

CLAIMS

1. A method of decoding video data, the method comprising:
 - decoding a first prefix code that represents a first portion of a position of a last non-zero coefficient in a X direction based on a threshold for performing a zero-out process, wherein the threshold is 32, and wherein decoding the prefix code that represents the first portion of the position of the last non-zero coefficient based on the threshold for performing a zero-out process comprises decoding the prefix code based on a range of 24-31;
 - decoding a first suffix code that represents a second portion of the position of the last non-zero coefficient in the X direction;
 - decoding a second prefix code that represents a first portion of the position of the last non-zero coefficient in a Y direction based on the threshold for performing a zero-out process;
 - decoding a second suffix code that represents a second portion of the position of the last non-zero coefficient in the Y direction;
 - determining a first region of a transform unit in which transform coefficients are subject to zero-out based on the threshold;
 - determining a second region of the transform unit in which transform coefficients are not subject to zero-out based on the threshold; and
 - scanning only the second region of the transform unit starting at the position of the last non-zero coefficient.
2. The method of claim 1, further comprising:
 - determining a context for entropy decoding one or more syntax elements indicating a position of a last non-zero coefficient based on an entire size of the transform unit.
3. The method of claim 1, wherein determining the first region of the transform unit in which transform coefficients are subject to zero-out comprises:
 - determining the first region of the transform unit in which transform coefficients are subject to zero-out based on positions of the transform coefficients in the transform unit relative to the threshold.
4. The method of claim 3, wherein determining the first region of the transform unit in

which transform coefficients are subject to zero-out based on positions of the transform coefficients in the transform unit comprises:

determining that the first region includes transform coefficients in the transform unit that are beyond the first 32 rows or columns of the transform unit are subject to zero-out.

5. The method of claim 1, wherein scanning only the second region of the transform unit starting at the position of the last non-zero coefficient comprises:

scanning only the second region of the transform unit coefficient group by coefficient group.

6. The method of claim 1, wherein the transform unit has a width or a length of 64 transform coefficients, and wherein the second region is a 32x32 region of the transform unit.

7. The method of claim 1, further comprising:

inverse transforming the transform unit to create residual data;

determining a prediction block;

adding the residual data to the prediction block to decode a block of video data; and

displaying a picture that includes the decoded block of video data.

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

a memory configured to store a transform unit of video data; and

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

decode a first prefix code that represents a first portion of a position of a last non-zero coefficient in a X direction based on a threshold for performing a zero-out process, wherein the threshold is 32, and wherein to decode the prefix code that represents the first portion of the position of the last non-zero coefficient based on the threshold for performing a zero-out process, the one or more processors are further configured to decode the prefix code based on a range of 24-31;

decode a first suffix code that represents a second portion of the position of the last non-zero coefficient in the X direction;

decode a second prefix code that represents a first portion of the position of the

last non-zero coefficient in a Y direction based on the threshold for performing a zero-out process;

decode a second suffix code that represents a second portion of the position of the last non-zero coefficient in the Y direction;

determine a first region of a transform unit in which transform coefficients are subject to zero-out based on the threshold;

determine a second region of the transform unit in which transform coefficients are not subject to zero-out based on the threshold; and

scan only the second region of the transform unit starting at the position of the last non-zero coefficient.

9. The apparatus of claim 8, wherein the one or more processors are further configured to:
determine a context for entropy decoding one or more syntax elements indicating a position of a last non-zero coefficient based on an entire size of the transform unit.

10. The apparatus of claim 8, wherein to determine the first region of the transform unit in which transform coefficients are subject to zero-out, the one or more processors are further configured to:

determine the first region of the transform unit in which transform coefficients are subject to zero-out based on positions of the transform coefficients in the transform unit relative to the threshold.

11. The apparatus of claim 10, wherein to determine the first region of the transform unit in which transform coefficients are subject to zero-out based on positions of the transform coefficients in the transform unit, the one or more processors are further configured to:

determine that the first region includes transform coefficients in the transform unit that are beyond the first 32 rows or columns of the transform unit are subject to zero-out.

12. The apparatus of claim 8, wherein to scan only the second region of the transform unit starting at the position of the last non-zero coefficient, the one or more processors are further configured to:

scan only the second region of the transform unit coefficient group by coefficient group.

13. The apparatus of claim 8, wherein the transform unit has a width or a length of 64 transform coefficients, and wherein the second region is a 32x32 region of the transform unit.
14. The apparatus of claim 8, wherein the one or more processors are further configured to:
 - inverse transform the transform unit to create residual data;
 - determine a prediction block;
 - add the residual data to the prediction block to decode a block of video data; and
 - display a picture that includes the decoded block of video data.
15. An apparatus configured to decode video data, the apparatus comprising:
 - means for decoding a first prefix code that represents a first portion of a position of a last non-zero coefficient in a X direction based on a threshold for performing a zero-out process, wherein the threshold is 32, and wherein the means for decoding the prefix code that represents the first portion of the position of the last non-zero coefficient based on the threshold for performing a zero-out process comprises means for decoding the prefix code based on a range of 24-31;
 - means for decoding a first suffix code that represents a second portion of the position of the last non-zero coefficient in the X direction;
 - means for decoding a second prefix code that represents a first portion of the position of the last non-zero coefficient in a Y direction based on the threshold for performing a zero-out process;
 - means for decoding a second suffix code that represents a second portion of the position of the last non-zero coefficient in the Y direction;
 - means for determining a first region of a transform unit in which transform coefficients are subject to zero-out based on the threshold;
 - means for determining a second region of the transform unit in which transform coefficients are not subject to zero-out based on the threshold; and
 - means for scanning only the second region of the transform unit starting at the position of the last non-zero coefficient.

16. The apparatus of claim 15, further comprising:
 - means for determining a context for entropy decoding one or more syntax elements indicating a position of a last non-zero coefficient based on an entire size of the transform unit.
17. The apparatus of claim 15, wherein the means for determining the first region of the transform unit in which transform coefficients are subject to zero-out comprises:
 - means for determining the first region of the transform unit in which transform coefficients are subject to zero-out based on positions of the transform coefficients in the transform unit relative to the threshold.
18. The apparatus of claim 17, wherein the means for determining the first region of the transform unit in which transform coefficients are subject to zero-out based on positions of the transform coefficients in the transform unit comprises:
 - means for determining that the first region includes transform coefficients in the transform unit that are beyond the first 32 rows or columns of the transform unit are subject to zero-out.
19. The apparatus of claim 15, wherein the means for scanning only the second region of the transform unit starting at the position of the last non-zero coefficient comprises:
 - means for scanning only the second region of the transform unit coefficient group by coefficient group.
20. The apparatus of claim 15, wherein the transform unit has a width or a length of 64 transform coefficients, and wherein the second region is a 32x32 region of the transform unit.
21. The apparatus of claim 15, further comprising:
 - means for inverse transforming the transform unit to create residual data;
 - means for determining a prediction block;
 - means for adding the residual data to the prediction block to decode a block of video data; and
 - means for displaying a picture that includes the decoded block of video data.

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

decode a first prefix code that represents a first portion of a position of a last non-zero coefficient in a X direction based on a threshold for performing a zero-out process, wherein the threshold is 32, and wherein to decode the prefix code that represents the first portion of the position of the last non-zero coefficient based on the threshold for performing a zero-out process, the instructions further cause the one or more processors to decode the prefix code based on a range of 24-31;

decode a first suffix code that represents a second portion of the position of the last non-zero coefficient in the X direction;

decode a second prefix code that represents a first portion of the position of the last non-zero coefficient in a Y direction based on the threshold for performing a zero-out process;

decode a second suffix code that represents a second portion of the position of the last non-zero coefficient in the Y direction;

determine a first region of a transform unit in which transform coefficients are subject to zero-out based on the threshold;

determine a second region of the transform unit in which transform coefficients are not subject to zero-out based on the threshold; and

scan only the second region of the transform unit starting at the position of the last non-zero coefficient.

23. The non-transitory computer-readable storage medium of claim 22, wherein the instructions further cause the one or more processors to:

determine a context for entropy decoding one or more syntax elements indicating a position of a last non-zero coefficient based on an entire size of the transform unit.

24. The non-transitory computer-readable storage medium of claim 22, wherein to determine the first region of the transform unit in which transform coefficients are subject to zero-out, the instructions further cause the one or more processors to:

determine the first region of the transform unit in which transform coefficients are subject to zero-out based on positions of the transform coefficients in the transform unit relative to the threshold.

25. The non-transitory computer-readable storage medium of claim 24, wherein to determine the first region of the transform unit in which transform coefficients are subject to zero-out based on positions of the transform coefficients in the transform unit, the instructions further cause the one or more processors to:

determine that the first region includes transform coefficients in the transform unit that are beyond the first 32 rows or columns of the transform unit are subject to zero-out.

26. The non-transitory computer-readable storage medium of claim 22, wherein to scan only the second region of the transform unit starting at the position of the last non-zero coefficient, the instructions further cause the one or more processors to:

scan only the second region of the transform unit coefficient group by coefficient group.

27. The non-transitory computer-readable storage medium of claim 22, wherein the transform unit has a width or a length of 64 transform coefficients, and wherein the second region is a 32x32 region of the transform unit.

28. The non-transitory computer-readable storage medium of claim 22, wherein the instructions further cause the one or more processors to:

inverse transform the transform unit to create residual data;

determine a prediction block;

add the residual data to the prediction block to decode a block of video data; and

display a picture that includes the decoded block of video data.