

CHROMA DELTA QUANTIZATION PARAMETER IN VIDEO CODING

[0001] This Application claims priority to U.S. Patent Application No. 16/912,232, filed 25 June 2020, which claims the benefit of U.S. Provisional Application No. 62/868,771, filed 28 June 2019, the entire contents of each of which are incorporated by reference.

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] This disclosure describes techniques related to the quantization process and, more specifically, to determining and signaling values for determining a chroma quantization parameter (QP), using a chroma delta QP. A video decoder can be configured to determine a predicted QP value for a block based on QP values used for previously decoded blocks and without receiving any explicit syntax elements in the bitstream. The video decoder can then receive a delta QP value representing a difference between the predicted QP value and an actual QP value for the block. As the delta QP value is typically small compared to the actual QP value, the delta QP value can be signaled with relatively fewer bits compared to the actual QP value. A quantization parameter, or QP, is a variable used by the decoding process for scaling transform coefficient levels. The QP effectively represents the amount of quantization applied to the coefficient levels.

[0006] According to one example, a method of decoding a bitstream of encoded video data includes determining a predicted luma quantization parameter (QP) for a luma component of a coding unit; receiving, in the bitstream of encoded video data, first syntax indicating a luma delta QP value for the luma component; determining a QP value for the luma component based on the predicted luma QP and the luma delta QP value; determining a predicted chroma QP for a chroma component of the coding unit; receiving, in the bitstream of encoded video data, second syntax indicating a chroma delta QP value for the chroma component of the coding unit; determining a QP value for the chroma component of the coding unit based on the predicted chroma QP and the chroma delta QP value; dequantizing a block of luma transform coefficients for the coding unit based on the QP value for the luma component; dequantizing a block of chroma transform coefficients for the coding unit based on the QP value of the chroma component; and decoding the coding unit based on the dequantized block of luma transform coefficients and the dequantized block of chroma transform coefficients.

[0007] According to another example, a method of encoding video data includes determining a quantization parameter (QP) value for a luma component of a coding unit of the video data; determining a predicted luma QP for the luma component of the coding unit, based on the QP value for the luma component and the predicted luma QP,

determining a delta QP value for the luma component of the coding unit; generating, for inclusion in a bitstream of encoded video data, first syntax indicating the delta QP value for the luma component of the coding unit; determining a QP value for a chroma component of the coding unit of the video data; determining a predicted chroma QP for the chroma component of the coding unit; based on the QP value for the chroma component and the predicted chroma QP, determining a delta QP value for the chroma component of the coding unit; generating, for inclusion in the bitstream of the encoded video data, second syntax indicating the delta QP value for the chroma component of the coding unit; and outputting, in the bitstream of encoded video data, the first syntax and the second syntax.

[0008] According to another example, a device for decoding video data includes a memory configured to store video data and one or more processors implemented in circuitry and configured to determine a predicted luma quantization parameter (QP) for a luma component of a coding unit; receive, in the bitstream of encoded video data, first syntax indicating a luma delta QP value for the luma component; determine a QP value for the luma component based on the predicted luma QP and the luma delta QP value; determine a predicted chroma QP for a chroma component of the coding unit; receive, in the bitstream of encoded video data, second syntax indicating a chroma delta QP value for the chroma component of the coding unit; determine a QP value for the chroma component of the coding unit based on the predicted chroma QP and the chroma delta QP value; dequantize a block of luma transform coefficients for the coding unit based on the QP value for the luma component; dequantize a block of chroma transform coefficients for the coding unit based on the QP value of the chroma component; and decode the coding unit based on the dequantized block of luma transform coefficients and the dequantized block of chroma transform coefficients.

[0009] According to another example, a device for encoding video data includes a memory configured to store video data and one or more processors implemented in circuitry and configured to determine a quantization parameter (QP) value for a luma component of a coding unit of the video data; determine a predicted luma QP for the luma component of the coding unit; based on the QP value for the luma component and the predicted luma QP, determine a delta QP value for the luma component of the coding unit; generate, for inclusion in a bitstream of encoded video data, first syntax indicating the delta QP value for the luma component of the coding unit; determine a QP value for a chroma component of the coding unit of the video data; determine a

predicted chroma QP for the chroma component of the coding unit; based on the QP value for the chroma component and the predicted chroma QP, determine a delta QP value for the chroma component of the coding unit; generate, for inclusion in the bitstream of the encoded video data, second syntax indicating the delta QP value for the chroma component of the coding unit; and output, in the bitstream of encoded video data, the first syntax and the second syntax.

[0010] According to another example, an apparatus for decoding a bitstream of encoded video data includes means for determining a predicted luma quantization parameter (QP) for a luma component of a coding unit; means for receiving, in the bitstream of encoded video data, first syntax indicating a luma delta QP value for the luma component; means for determining a QP value for the luma component based on the predicted luma QP and the luma delta QP value; means for determining a predicted chroma QP for a chroma component of the coding unit; means for receiving, in the bitstream of encoded video data, second syntax indicating a chroma delta QP value for the chroma component of the coding unit; means for determining a QP value for the chroma component of the coding unit based on the predicted chroma QP and the chroma delta QP value; means for dequantizing a block of luma transform coefficients for the coding unit based on the QP value for the luma component; means for dequantizing a block of chroma transform coefficients for the coding unit based on the QP value of the chroma component; and means for decoding the coding unit based on the dequantized block of luma transform coefficients and the dequantized block of chroma transform coefficients.

[0011] According to another example, a computer-readable storage medium stores instructions that when executed by one or more processors cause the one or more processor to determine a predicted luma quantization parameter (QP) for a luma component of a coding unit; receive, in the bitstream of encoded video data, first syntax indicating a luma delta QP value for the luma component; determine a QP value for the luma component based on the predicted luma QP and the luma delta QP value; determine a predicted chroma QP for a chroma component of the coding unit; receive, in the bitstream of encoded video data, second syntax indicating a chroma delta QP value for the chroma component of the coding unit; determine a QP value for the chroma component of the coding unit based on the predicted chroma QP and the chroma delta QP value; dequantizing a block of luma transform coefficients for the coding unit based on the QP value for the luma component; dequantize a block of chroma transform

coefficients for the coding unit based on the QP value of the chroma component; and decode the coding unit based on the dequantized block of luma transform coefficients and the dequantized block of chroma transform coefficients.

[0012] According to another example, an apparatus for encoding video data includes means for determining a quantization parameter (QP) value for a luma component of a coding unit of the video data; means for determining a predicted luma QP for the luma component of the coding unit; means for determining a delta QP value for the luma component of the coding unit based on the QP value for the luma component and the predicted luma QP; means for generating, for inclusion in a bitstream of encoded video data, first syntax indicating the delta QP value for the luma component of the coding unit; means for determining a QP value for a chroma component of the coding unit of the video data; means for determining a predicted chroma QP for the chroma component of the coding unit; means for determining a delta QP value for the chroma component of the coding unit based on the QP value for the chroma component and the predicted chroma QP; means for generating, for inclusion in the bitstream of the encoded video data, second syntax indicating the delta QP value for the chroma component of the coding unit; and means for outputting, in the bitstream of encoded video data, the first syntax and the second syntax.

[0013] A computer-readable storage medium storing instructions that when executed by one or more processors cause the one or more processor to determine a quantization parameter (QP) value for a luma component of a coding unit of the video data; determine a predicted luma QP for the luma component of the coding unit; based on the QP value for the luma component and the predicted luma QP, determine a delta QP value for the luma component of the coding unit; generate, for inclusion in a bitstream of encoded video data, first syntax indicating the delta QP value for the luma component of the coding unit; determine a QP value for a chroma component of the coding unit of the video data; determine a predicted chroma QP for the chroma component of the coding unit; based on the QP value for the chroma component and the predicted chroma QP, determine a delta QP value for the chroma component of the coding unit; generate, for inclusion in the bitstream of the encoded video data, second syntax indicating the delta QP value for the chroma component of the coding unit; and output, in the bitstream of encoded video data, the first syntax and the second syntax.

[0014] 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

[0015] FIG. 1 is a block diagram illustrating an example video encoding and decoding system that may perform the techniques of this disclosure.

[0016] FIGS. 2A and 2B are conceptual diagrams illustrating an example quadtree binary tree (QTBT) structure, and a corresponding coding tree unit (CTU).

[0017] FIG. 3 shows an example process for determining a delta QP value for video data that supports dual tree partitioning.

[0018] FIG. 4 shows an example process for decoding a chroma QP offset value for video data that supports dual tree partitioning.

[0019] FIG. 5 shows an example video coding process.

[0020] FIG. 6 shows an example of a chroma CU with corresponding luma CUs in a separate tree.

[0021] FIG. 7 is a block diagram illustrating an example video encoder that may perform the techniques of this disclosure.

[0022] FIG. 8 is a block diagram illustrating an example video decoder that may perform the techniques of this disclosure.

[0023] FIG. 9 is a flowchart illustrating an example video encoding process.

[0024] FIG. 10 is a flowchart illustrating an example video decoding process.

[0025] FIG. 11 is a flowchart illustrating an example video encoding process.

[0026] FIG. 12 is a flowchart illustrating an example video decoding process.

DETAILED DESCRIPTION

[0027] Video coding (e.g., video encoding and/or video decoding) typically involves predicting a block of video data from either an already coded block of video data in the same picture (e.g., intra prediction) or an already coded block of video data in a different picture (e.g., inter prediction). In some instances, the video encoder also calculates residual data by comparing the prediction block to the original block. Thus, the residual data represents a difference between the prediction block and the original block. To reduce the number of bits needed to signal the residual data, the video

encoder transforms and quantizes the residual data and signals the transformed and quantized residual data in the encoded bitstream. The compression achieved by the transform and quantization processes may be lossy, meaning that transform and quantization processes may introduce distortion into the decoded video data.

[0028] A video decoder decodes and adds the residual data to the prediction block to produce a reconstructed video block that matches the original video block more closely than the prediction block alone. Due to the loss introduced by the transforming and quantizing of the residual data, the first reconstructed block may have distortion or artifacts. One common type of artifact or distortion is referred to as blockiness, where the boundaries of the blocks used to code the video data are visible.

[0029] To further improve the quality of decoded video, a video decoder can perform one or more filtering operations on the reconstructed video blocks. Examples of these filtering operations include deblocking filtering, sample adaptive offset (SAO) filtering, and adaptive loop filtering (ALF). Parameters for these filtering operations may either be determined by a video encoder and explicitly signaled in the encoded video bitstream or may be implicitly determined by a video decoder without needing the parameters to be explicitly signaled in the encoded video bitstream.

[0030] This disclosure describes techniques related to the quantization process and, more specifically, to determining and signaling values for determining a chroma quantization parameter (QP), using a chroma delta QP. A video decoder can be configured to determine a predicted QP value for a block based on QP values used for previously decoded blocks and without receiving any explicit syntax elements in the bitstream. The video decoder can then receive a delta QP value representing a difference between the predicted QP value and an actual QP value for the block. As the delta QP value is typically small compared to the actual QP value, the delta QP value can be signaled with relatively fewer bits compared to the actual QP value.

[0031] A quantization parameter, or QP, is a variable used by the decoding process for scaling transform coefficient levels. The QP effectively represents the amount of quantization applied to the coefficient levels. By configuring a video decoder to receive, in a bitstream of encoded video data, syntax indicating a chroma delta QP value for a chroma component and to determine a QP value for the chroma component based on the predicted chroma QP and the chroma delta QP value, the techniques of this disclosure may achieve the advantage of an improved rate-distortion tradeoff achieved in some coding scenarios by allowing for chroma QP values to be adjusted independent

of luma QP values.

[0032] The described techniques may be used in conjunction with any of the existing video codecs, such as High Efficiency Video Coding (HEVC), Versatile Video Coding (VVC), or be an efficient coding tool in any future video coding standards. The techniques of this disclosure will be described with respect HEVC, JEM, and VVC, but the techniques described herein are not limited to any particular standard.

[0033] FIG. 1 is a block diagram illustrating an example video encoding and decoding system 100 that may perform the techniques of this disclosure. The techniques of this disclosure are generally directed to coding (encoding and/or decoding) video data. In general, video data includes any data for processing a video. Thus, video data may include raw, unencoded video, encoded video, decoded (e.g., reconstructed) video, and video metadata, such as signaling data.

[0034] As shown in FIG. 1, system 100 includes a source device 102 that provides encoded video data to be decoded and displayed by a destination device 116, in this example. In particular, source device 102 provides the video data to destination device 116 via a computer-readable medium 110. Source device 102 and destination device 116 may comprise any of a wide range of devices, including desktop computers, notebook (i.e., laptop) computers, mobile devices, tablet computers, set-top boxes, telephone handsets such as smartphones, televisions, cameras, display devices, digital media players, video gaming consoles, video streaming device, broadcast receiver devices, or the like. In some cases, source device 102 and destination device 116 may be equipped for wireless communication, and thus may be referred to as wireless communication devices.

[0035] 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 chroma delta QP coding described herein. 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.

[0036] 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 chroma delta QP coding described herein. Source device 102 and destination device 116 are merely examples of such coding devices in which source device 102 generates coded video data for transmission to destination device 116. This disclosure refers to a “coding” device as a device that performs coding (encoding and/or decoding) of data. Thus, video encoder 200 and video decoder 300 represent examples of coding devices, in particular, a video encoder and a video decoder, respectively. In some examples, source device 102 and destination device 116 may operate in a substantially symmetrical manner such that each of source device 102 and destination device 116 includes video encoding and decoding components. Hence, system 100 may support one-way or two-way video transmission between source device 102 and destination device 116, e.g., for video streaming, video playback, video broadcasting, or video telephony.

[0037] In general, video source 104 represents a source of video data (i.e., raw, unencoded video data) and provides a sequential series of pictures (also referred to as “frames”) of the video data to video encoder 200, which encodes data for the pictures. Video source 104 of source device 102 may include a video capture device, such as a video camera, a video archive containing previously captured raw video, and/or a video feed interface to receive video from a video content provider. As a further alternative, video source 104 may generate computer graphics-based data as the source video, or a combination of live video, archived video, and computer-generated video. In each case, video encoder 200 encodes the captured, pre-captured, or computer-generated video data. Video encoder 200 may rearrange the pictures from the received order (sometimes referred to as “display order”) into a coding order for coding. Video encoder 200 may generate a bitstream including encoded video data. Source device 102 may then output the encoded video data via output interface 108 onto computer-readable medium 110 for reception and/or retrieval by, e.g., input interface 122 of destination device 116.

[0038] Memory 106 of source device 102 and memory 120 of destination device 116 represent general purpose memories. In some examples, memories 106, 120 may store raw video data, e.g., raw video from video source 104 and raw, decoded video data from video decoder 300. Additionally or alternatively, memories 106, 120 may store software instructions executable by, e.g., video encoder 200 and video decoder 300, respectively.

Although memory 106 and memory 120 are shown separately from video encoder 200 and video decoder 300 in this example, it should be understood that video encoder 200 and video decoder 300 may also include internal memories for functionally similar or equivalent purposes. Furthermore, memories 106, 120 may store encoded video data, e.g., output from video encoder 200 and input to video decoder 300. In some examples, portions of memories 106, 120 may be allocated as one or more video buffers, e.g., to store raw, decoded, and/or encoded video data.

[0039] Computer-readable medium 110 may represent any type of medium or device capable of transporting the encoded video data from source device 102 to destination device 116. In one example, computer-readable medium 110 represents a communication medium to enable source device 102 to transmit encoded video data directly to destination device 116 in real-time, e.g., via a radio frequency network or computer-based network. Output interface 108 may modulate a transmission signal including the encoded video data, and input interface 122 may demodulate the received transmission signal, according to a communication standard, such as a wireless communication protocol. The communication medium may comprise any wireless or wired communication medium, such as a radio frequency (RF) spectrum or one or more physical transmission lines. The communication medium may form part of a packet-based network, such as a local area network, a wide-area network, or a global network such as the Internet. The communication medium may include routers, switches, base stations, or any other equipment that may be useful to facilitate communication from source device 102 to destination device 116.

[0040] In some examples, source device 102 may output encoded data from output interface 108 to storage device 112. Similarly, destination device 116 may access encoded data from storage device 112 via input interface 122. Storage device 112 may include any of a variety of distributed or locally accessed data storage media such as a hard drive, Blu-ray discs, DVDs, CD-ROMs, flash memory, volatile or non-volatile memory, or any other suitable digital storage media for storing encoded video data.

[0041] In some examples, source device 102 may output encoded video data to file server 114 or another intermediate storage device that may store the encoded video data generated by source device 102. Destination device 116 may access stored video data from file server 114 via streaming or download.

[0042] File server 114 may be any type of server device capable of storing encoded video data and transmitting that encoded video data to the destination device 116. File

server 114 may represent a web server (e.g., for a website), a server configured to provide a file transfer protocol service (such as File Transfer Protocol (FTP) or File Delivery over Unidirectional Transport (FLUTE) protocol), a content delivery network (CDN) device, a hypertext transfer protocol (HTTP) server, a Multimedia Broadcast Multicast Service (MBMS) or Enhanced MBMS (eMBMS) server, and/or a network attached storage (NAS) device. File server 114 may, additionally or alternatively, implement one or more HTTP streaming protocols, such as Dynamic Adaptive Streaming over HTTP (DASH), HTTP Live Streaming (HLS), Real Time Streaming Protocol (RTSP), HTTP Dynamic Streaming, or the like.

[0043] Destination device 116 may access encoded video data from file server 114 through any standard data connection, including an Internet connection. This may include a wireless channel (e.g., a Wi-Fi connection), a wired connection (e.g., digital subscriber line (DSL), cable modem, etc.), or a combination of both that is suitable for accessing encoded video data stored on file server 114. Input interface 122 may be configured to operate according to any one or more of the various protocols discussed above for retrieving or receiving media data from file server 114, or other such protocols for retrieving media data.

[0044] Output interface 108 and input interface 122 may represent wireless transmitters/receivers, modems, wired networking components (e.g., Ethernet cards), wireless communication components that operate according to any of a variety of IEEE 802.11 standards, or other physical components. In examples where output interface 108 and input interface 122 comprise wireless components, output interface 108 and input interface 122 may be configured to transfer data, such as encoded video data, according to a cellular communication standard, such as 4G, 4G-LTE (Long-Term Evolution), LTE Advanced, 5G, or the like. In some examples where output interface 108 comprises a wireless transmitter, output interface 108 and input interface 122 may be configured to transfer data, such as encoded video data, according to other wireless standards, such as an IEEE 802.11 specification, an IEEE 802.15 specification (e.g., ZigBee™), a Bluetooth™ standard, or the like. In some examples, source device 102 and/or destination device 116 may include respective system-on-a-chip (SoC) devices. For example, source device 102 may include an SoC device to perform the functionality attributed to video encoder 200 and/or output interface 108, and destination device 116 may include an SoC device to perform the functionality attributed to video decoder 300 and/or input interface 122.

[0045] 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.

[0046] Input interface 122 of destination device 116 receives an encoded video bitstream from computer-readable medium 110 (e.g., a communication medium, storage device 112, file server 114, or the like). The encoded video bitstream may include signaling information defined by video encoder 200, which is also used by video decoder 300, such as syntax elements having values that describe characteristics and/or processing of video blocks or other coded units (e.g., slices, pictures, groups of pictures, sequences, or the like). Display device 118 displays decoded pictures of the decoded video data to a user. Display device 118 may represent any of a variety of display devices such as a liquid crystal display (LCD), a plasma display, an organic light emitting diode (OLED) display, or another type of display device.

[0047] 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).

[0048] Video encoder 200 and video decoder 300 each may be implemented as any of a variety of suitable encoder and/or decoder circuitry, such as one or more microprocessors, digital signal processors (DSPs), application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), discrete logic, software, hardware, firmware or any combinations thereof. When the techniques are implemented partially in software, a device may store instructions for the software in a suitable, non-transitory computer-readable medium and execute the instructions in hardware using one or more processors to perform the techniques of this disclosure. Each of video encoder 200 and video decoder 300 may be included in one or more encoders or decoders, either of which may be integrated as part of a combined encoder/decoder (CODEC) in a respective device. A device including video encoder 200 and/or video

decoder 300 may comprise an integrated circuit, a microprocessor, and/or a wireless communication device, such as a cellular telephone.

[0049] Video encoder 200 and video decoder 300 may operate according to a video coding standard, such as ITU-T H.265, also referred to as High Efficiency Video Coding (HEVC) or extensions thereto, such as the multi-view and/or scalable video coding extensions. Alternatively, video encoder 200 and video decoder 300 may operate according to other proprietary or industry standards, such as ITU-T H.266, also referred to as Versatile Video Coding (VVC). A draft of the VVC standard is described in Bross, et al. “Versatile Video Coding (Draft 5),” Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 14th Meeting: Geneva, CH, 19-27 March 2019, JVET-N1001-v9 (hereinafter “VVC Draft 5”). Another draft of the VVC standard is described in Bross, et al. “Versatile Video Coding (Draft 9),” Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 18th Meeting: by teleconference, 15–24 Apr. 2020, JVET-R2001-v11 (hereinafter “VVC Draft 9”). The techniques of this disclosure, however, are not limited to any particular coding standard.

[0050] 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.

[0051] 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.

[0052] 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.

[0053] As another example, video encoder 200 and video decoder 300 may be configured to operate according to VVC. According to VVC, a video coder (such as video encoder 200) partitions a picture into a plurality of coding tree units (CTUs). Video encoder 200 may partition a CTU according to a tree structure, such as a quadtree-binary tree (QTBT) structure or Multi-Type Tree (MTT) structure. The QTBT structure removes the concepts of multiple partition types, such as the separation between CUs, PUs, and TUs of HEVC. A QTBT structure includes two levels: a first level partitioned according to quadtree partitioning, and a second level partitioned according to binary tree partitioning. A root node of the QTBT structure corresponds to a CTU. Leaf nodes of the binary trees correspond to coding units (CUs).

[0054] In an MTT partitioning structure, blocks may be partitioned using a quadtree (QT) partition, a binary tree (BT) partition, and one or more types of triple tree (TT) (also called ternary tree (TT)) partitions. A triple or ternary tree partition is a partition where a block is split into three sub-blocks. In some examples, a triple or ternary tree partition divides a block into three sub-blocks without dividing the original block through the center. The partitioning types in MTT (e.g., QT, BT, and TT), may be symmetrical or asymmetrical.

[0055] 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).

[0056] Video encoder 200 and video decoder 300 may be configured to use quadtree partitioning per HEVC, QTBT partitioning, MTT partitioning, or other partitioning structures. For purposes of explanation, the description of the techniques of this disclosure is presented with respect to QTBT partitioning. However, it should be understood that the techniques of this disclosure may also be applied to video coders configured to use quadtree partitioning, or other types of partitioning as well.

[0057] In some examples, a CTU includes a coding tree block (CTB) of luma samples, two corresponding CTBs of chroma samples of a picture that has three sample arrays, or a CTB of samples of a monochrome picture or a picture that is coded using three separate color planes and syntax structures used to code the samples. A CTB may be an NxN block of samples for some value of N such that the division of a component into CTBs is a partitioning. A component is an array or single sample from one of the three arrays (luma and two chroma) that compose a picture in 4:2:0, 4:2:2, or 4:4:4 color format or the array or a single sample of the array that compose a picture in monochrome format. In some examples, a coding block is an MxN block of samples for some values of M and N such that a division of a CTB into coding blocks is a partitioning.

[0058] 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.

[0059] 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.

[0060] 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.

[0061] 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.

[0062] 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.

[0063] 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.

[0064] Some examples of VVC also provide an affine motion compensation mode, which may be considered an inter-prediction mode. In affine motion compensation mode, video encoder 200 may determine two or more motion vectors that represent non-

translational motion, such as zoom in or out, rotation, perspective motion, or other irregular motion types.

[0065] To perform intra-prediction, video encoder 200 may select an intra-prediction mode to generate the prediction block. Some examples of VVC provide sixty-seven intra-prediction modes, including various directional modes, as well as planar mode and DC mode. In general, video encoder 200 selects an intra-prediction mode that describes neighboring samples to a current block (e.g., a block of a CU) from which to predict samples of the current block. Such samples may generally be above, above and to the left, or to the left of the current block in the same picture as the current block, assuming video encoder 200 codes CTUs and CUs in raster scan order (left to right, top to bottom).

[0066] Video encoder 200 encodes data representing the prediction mode for a current block. For example, for inter-prediction modes, video encoder 200 may encode data representing which of the various available inter-prediction modes is used, as well as motion information for the corresponding mode. For uni-directional or bi-directional inter-prediction, for example, video encoder 200 may encode motion vectors using advanced motion vector prediction (AMVP) or merge mode. Video encoder 200 may use similar modes to encode motion vectors for affine motion compensation mode.

[0067] 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.

[0068] 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.

[0069] 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.

[0070] 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.

[0071] 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.

[0072] 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.

[0073] In general, video decoder 300 performs a reciprocal process to that performed by video encoder 200 to decode the encoded video data of the bitstream. For example, video decoder 300 may decode values for syntax elements of the bitstream using CABAC in a manner substantially similar to, albeit reciprocal to, the CABAC encoding process of video encoder 200. The syntax elements may define partitioning information for partitioning of a picture into CTUs, and partitioning of each CTU according to a corresponding partition structure, such as a QTBT structure, to define CUs of the CTU. The syntax elements may further define prediction and residual information for blocks (e.g., CUs) of video data.

[0074] 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.

[0075] By determining a predicted chroma QP for a chroma component of the coding unit, signalling syntax indicating a chroma delta QP value for the chroma component of the coding unit, and determining a QP value for the chroma component of the coding unit based on the predicted chroma QP and the chroma delta QP value, video encoder 200 and video decoder 300 may enable for chroma QP values to be flexibly adjusted independent of luma QP values, in a manner that does not create an undesirable amount of additional signalling overhead.

[0076] This disclosure may generally refer to “signaling” certain information, such as syntax elements. The term “signaling” may generally refer to the communication of values for syntax elements and/or other data used to decode encoded video data. That is, video encoder 200 may signal values for syntax elements in the bitstream. In general, signaling refers to generating a value in the bitstream. As noted above, source device 102 may transport the bitstream to destination device 116 substantially in real

time, or not in real time, such as might occur when storing syntax elements to storage device 112 for later retrieval by destination device 116.

[0077] 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. A CTU may be partitioned with either single tree partitioning or dual tree partitioning. With single tree partitioning, the chroma component of the CTU and the luma component of the CTU have the same partitioning structure. With dual tree partitioning, the chroma component of the CTU and the luma component of the CTU potentially have different partitioning structure.

[0078] 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. 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).

[0079] 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.”

[0080] 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 quadtree leaf 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 quadtree leaf 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. A binary tree node having a width equal to MinBTSIZE (4, in this example) it implies that no further vertical splitting (that is, dividing of the width) is permitted for that binary tree node. Similarly, a binary tree node having a height equal to MinBTSIZE implies no further horizontal splitting (that is, dividing of the height) 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.

[0081] Video encoder 200 and video decoder 300 may be configured to process luma delta QP values. That is, video decoder 300 can be configured to determine a predicted luma QP value for a block based on luma QP values used for previously decoded blocks and without receiving any explicit syntax elements in the bitstream. Video decoder 300

can then receive a luma delta QP value representing a difference between the predicted luma QP value and an actual luma QP value for the block.

[0082] In VVC Draft 5, values for delta QP are singled in a similar way to HEVC. The syntax included in VVC Draft 5 is as follows:

transform_unit(x0, y0, tbWidth, tbHeight, treeType, subTuIndex) {	Descriptor
...	
if((tu_cbf_luma[x0][y0] tu_cbf_cb[x0][y0] tu_cbf_cr[x0][y0]) && treeType != DUAL_TREE_CHROMA) {	
if(cu_qp_delta_enabled_flag && !IsCuQpDeltaCoded) {	
cu_qp_delta_abs	ae(v)
if(cu_qp_delta_abs)	
cu_qp_delta_sign_flag	ae(v)
}	
}	
...	
}	

[0083] The syntax element `cu_qp_delta_abs` specifies the absolute value of the difference between the quantization parameter of the current coding unit and a predicted QP value for the coding unit. As one example, video encoder 200 and video decoder 300 may, for example, determine the predicted QP value for a current CU to be equal to the QP value used for a previously coded CU. If one or both of an above neighboring CU or a left neighboring CU are available, then video encoder 200 and video decoder 300 may use one of the QP for the above neighbor or the QP for the left neighbor as the predicted QP value. If a previously coded CU does not have a QP value, for instance if the previously coded CU was coded in a skip mode, then video encoder 200 and video decoder 300 may be configured to set the predicted QP value equal to a QP value for a last CU that utilized quantization or to a default value. In the description below, `CuQpDeltaVal` represents a delta QP value.

[0084] The syntax element `cu_qp_delta_sign_flag` specifies the sign of `CuQpDeltaVal` as follows:

- If `cu_qp_delta_sign_flag` is equal to 0, the corresponding `CuQpDeltaVal` has a positive value.

- Otherwise (cu_qp_delta_sign_flag is equal to 1), the corresponding CuQpDeltaVal has a negative value.

When cu_qp_delta_sign_flag is not present, a value for cu_qp_delta_sign_flag is inferred to be equal to 0.

When cu_qp_delta_abs is present, the variables IsCuQpDeltaCoded and CuQpDeltaVal are derived as follows:

$$\text{IsCuQpDeltaCoded} = 1 \quad (7-130)$$

$$\text{CuQpDeltaVal} = \text{cu_qp_delta_abs} * (1 - 2 * \text{cu_qp_delta_sign_flag}) \quad (7-131)$$

The value of CuQpDeltaVal shall be in the range of $-(32 + \text{QpBdOffsetY} / 2)$ to $+(31 + \text{QpBdOffsetY} / 2)$, inclusive.

[0085] The syntax element cu_qp_delta_enabled_flag equal to 1 specifies that the cu_qp_delta_subdiv syntax element is present in the PPS and that cu_qp_delta_abs may be present in the transform unit syntax. The syntax element cu_qp_delta_enabled_flag equal to 0 specifies that the cu_qp_delta_subdiv syntax element is not present in the PPS and that cu_qp_delta_abs is not present in the transform unit syntax.

[0086] The syntax element cu_qp_delta_subdiv specifies the maximum cbSubdiv value of coding units that convey cu_qp_delta_abs and cu_qp_delta_sign_flag. The value range of cu_qp_delta_subdiv is specified as follows:

- If slice_type is equal to I, the value of cu_qp_delta_subdiv shall be in the range of 0 to

$$2 * (\log_2 \text{ctu_size_minus2} - \log_2 \text{min_qt_size_intra_slice_minus2} + \text{MaxMttDepthY}), \text{ inclusive.}$$

- Otherwise (slice_type is not equal to I), the value of cu_qp_delta_subdiv shall be in the range of 0 to

$$2 * (\log_2 \text{ctu_size_minus2} - \log_2 \text{min_qt_size_inter_slice_minus2} + \text{MaxMttDepthY}), \text{ inclusive.}$$

When not present, the value of cu_qp_delta_subdiv is inferred to be equal to 0.

[0087] CBF flags of tu_cbf_luma[x0][y0], tu_cbf_cb[x0][y0], tu_cbf_cr[x0][y0] being equal to 1 means there is a coding bin in the coding block. Otherwise, a CBF equal to 0 mean there is no coding bin in the coding block.

[0088] FIG. 3 shows an example process for determining a delta QP value for video data that supports dual tree partitioning. Video decoder 300 determines if a CU is partitioned in a dual tree structure (140). If the CU is not partitioned in a dual tree structure (140, NO), then video decoder 300 determines a single partitioning for both

the luma and chroma components of the CU (142) and decodes a delta QP value (142). If the CU is partitioned in a dual tree structure (140, YES), then video decoder 300 determines a partitioning for a luma component of the CU (146) and separately determines a partitioning of a chroma component of the CU (148). Video decoder 300 also decodes a delta QP value (150).

[0089] Video encoder 200 and video decoder 300 may be configured to perform luma QP prediction. Video encoder 200 and video decoder 300 derive the predicted luma quantization parameter, qP_{Y_PRED} , as follows:

- If all the following conditions are true, then qP_{Y_PRED} is set equal to the luma quantization parameter Qp_Y of the coding unit containing the luma coding block covering ($xQg, yQg - 1$):
 - availableB is equal to TRUE.
 - the current quantization group is the first quantization group in a CTB row within a brick
- Otherwise, qP_{Y_PRED} is derived as follows:

$$qP_{Y_PRED} = (qP_{Y_A} + qP_{Y_B} + 1) \ggg 1$$

[0090] Video encoder 200 and video decoder 300 may be configured to process a previously luma QP. Video encoder 200 and video decoder 300 can derive the previous luma quantization parameter, qP_{Y_PREV} , as follows:

- If one or more of the following conditions are true, qP_{Y_PREV} is set equal to $SliceQp_Y$:
 - The current quantization group is the first quantization group in a slice.
 - The current quantization group is the first quantization group in a brick.
- Otherwise, qP_{Y_PREV} is set equal to the luma quantization parameter Qp_Y of the last luma coding unit in the previous quantization group in decoding order.

[0091] Video encoder 200 and video decoder 300 may be configured to determine a chroma QP. In VVC Draft 5, a value for chroma QP is calculated based on a value for luma QP.

[0092] When `treeType` is equal to `DUAL_TREE_CHROMA`, the variable Qp_Y is set equal to the luma quantization parameter Qp_Y of the luma coding unit that covers the luma location ($xCb + cbWidth / 2, yCb + cbHeight / 2$).

- The variables $qPCb$, $qPCr$ and $qPCbCr$ are derived as follows:

$$qPiCb = Clip3(-QpBdOffsetC, 69, Qp_Y + pps_cb_qp_offset + slice_cb_qp_offset)$$

$$qPiCr = Clip3(-QpBdOffsetC, 69, Qp_Y + pps_cr_qp_offset + slice_cr_qp_offset)q$$

$PiCbCr = Clip3(-QpBdOffsetC, 69, QpY + pps_joint_cbcr_qp_offset + slice_joint_cbcr_qp_offset)$

- If ChromaArrayType is equal to 1, the variables qPCb, qPCr and qPCbCr are set equal to the value of QpC as specified in Table below, based on the index qPi equal to qPiCb, qPiCr and qPiCbCr, respectively.
- Otherwise, the variables qPCb, qPCr and qPCbCr are set equal to $\text{Min}(qPi, 63)$, based on the index qPi equal to qPiCb, qPiCr and qPiCbCr, respectively.
- The chroma quantization parameters for the Cb and Cr components, Qp'Cb and Qp'Cr, and joint Cb-Cr coding Qp'CbCr are derived as follows:

$$Qp'Cb = qPCb + QpBdOffsetC$$

$$Qp'Cr = qPCr + QpBdOffsetC$$

$$Qp'CbCr = qPCbCr + QpBdOffsetC$$

[0093] The table below specifies Qpc as a function of qPi for ChromaArrayType equal to 1

Pi	30	0	1	2	3	4	5	6	7	8	9	0	1	2	3	43
pc	qPi	9	0	1	2	3	3	4	4	5	5	6	6	7	7	qPi – 6

[0094] Video encoder 200 and video decoder 300 may be configured to determine a chroma QP offset. The chroma QP offset may, for example, be an index value. Video encoder 200 and video decoder 300 can maintain tables that map these index values to delta QP values.

[0095] FIG. 4 shows an example process for decoding a chroma QP offset value for video data that supports dual tree partitioning. Video decoder 300 determines if a CU is partitioned in a dual tree structure (152). If the CU is not partitioned in a dual tree structure (152, NO), then video decoder 300 determines a single partitioning for both the luma and chroma components of the CU (154) and decodes a chroma QP offset value (156). If the CU is partitioned in a dual tree structure (152, YES), then video decoder 300 determines a partitioning for a luma component of the CU (158) and separately determines a partitioning of a chroma component of the CU (160). Video decoder 300 also decodes a chroma QP offset value (162).

[0096] Video encoder 200 and video decoder 300 may be configured to utilize quantization groups. All CUs in a quantization group typically use the same QP predictor but not necessarily the same QP. The current quantization group is a rectangular region inside a coding tree block that shares the same qP_{Y_PRED} . The quantization group width and height are equal to the width and height of the coding tree node, of which the top-left luma sample position is assigned to the variables $CuQgTopLeftX$ and $CuQgTopLeftY$.

[0097] FIG. 5 shows an example video coding process for determining a QP predictor, QP_pred . The techniques of FIG. 5 will be described with respect to video encoder 200 but may also be performed by video decoder 300. Video encoder 200 determines if a CU is the first CU in a CTU line and if the above CU is available (170). If the CU is the first CU in the CTU line and the above CU is available (170, Yes), then video encoder 200 sets the QP predictor equal to the QP of the above CU (172). If the CU is not the first CU in the CTU line or the above CU is not available (170, No), the video encoder 200 determines if a QP for an above CU is available (174) and if a CU for a left CU is available (176). If a QP for an above CU is available (174, Yes), then video encoder 200 sets the value for a variable (QP_above) to the value of the QP for the above CU. If a QP for an above CU is not available (174, No), then video encoder 200 sets the value for QP_above to a previously used QP value (178). If a QP for a left CU is available (176, Yes), then video encoder 200 sets the value for a variable (QP_left) to the value of the QP for the left CU. If a QP for a left CU is not available (176, No), then video encoder 200 sets the value for QP_left to a previously used QP value (180). Video encoder 200 sets a value for the QP predictor to an average of the values for QP_left and QP_above , which may be expressed mathematically as $QP_pred = (QP_above + QP_left + 1) >> 1$ (182).

[0098] In the example above, the previously used QP value, QP_prev , may be any previously used QP value, and need not necessarily be a QP value for a left neighboring CU or an above neighboring CU. If, for example, a left neighboring CU was coded in a skip mode without transformation or quantization, then the previously used QP value may belong to the most recently decoded CU that had a QP value even if that CU is not adjacent to the current CU.

[0099] Current techniques for determining chroma QP values include some potential problems. For example, in VVC Draft 5, chroma QP values are derived as follows:

$$qPic_b = \text{Clip3}(-QpBdOffset_c, 69, Qp_Y + pps_cb_qp_offset + slice_cb_qp_offset)$$

$$qPic_{Cr} = \text{Clip3}(-QpBdOffset_c, 69, Qp_Y + pps_cr_qp_offset + slice_cr_qp_offset)$$

As can be seen above, the signaling for chroma QP values is controlled at the PPS and slice level.

[0100] When a tree type for a coding unit is set to dual trees, the variable Qp_Y is set equal to the luma QP of the luma CB that covers the luma position $(xCb + cbWidth / 2, yCb + cbHeight / 2)$. FIG. 6 shows an example of a chroma CU with corresponding luma CUs in a separate tree. As shown in Fig. 6, one chroma CB 190 may cover more than one luma CB (192A-192D), and these luma CBs 192A-192D may be from different QGs with different QPs. The predicted QP derived from the center position may be not a accurate prediction for the chroma CB. And pps/slice level QP offset adjustment is limited. VVC draft 5 provides no scheme for flexibly adjusting chroma QP values.

[0101] This disclosure describes techniques that may address some of these problems. Unless stated to the contrary, the various techniques of this disclosure may be performed either separately or in combination.

[0102] Video encoder 200 and video decoder 300 may be configured to signal chroma delta QP values for chroma QGs. In one example, video encoder 200 and video decoder 300 may be configured to signal chroma delta QP values using the same scheme used for luma delta QP signaling in VVC Draft 5.

[0103] Video encoder 200 and video decoder 300 may be configured to define the chroma QG based on a split depth. In one example, the chroma QG may be defined in the same way as a luma QG but based on the depth of the chroma split. For example, $chroma_cu_qp_delta_subdiv$ can represent the maximum chroma delta QP signaling depth for chroma. For a leaf node with a depth smaller than or equal to the maximum chroma delta QP signalling depth, video encoder 200 and video decoder 300 can signal one chroma delta QP value for the CU of this leaf node, if the CU has at least one non-zero coefficient. For a CU with a depth greater than the maximum chroma delta QP signalling depth, video encoder 200 and video decoder 300 can signal one chroma delta QP value for all CUs of the split node at the maximum chroma delta QP signalling depth, if any of CUs of the child nodes of the split node have at least one non-zero coefficient.

[0104] Video encoder 200 and video decoder 300 may be configured to define the chroma QG by an area. In one example, the area may be a rectangular region specified by width and height. In one example, the area may be specified by the number of

pixels in the QG. In this example, the width/height of the region, or the specified number of pixels may be predefined in both encoder side and decoder side, or set as a value signaled from the encoder to the decoder at sequence level, picture level, slice level. For example, this value may be signaled in an SPS, PPS, or slice header.

[0105] Video encoder 200 and video decoder 300 may be configured to signal the value of chroma delta QP by signaling an absolute value and sign flag in the same way as luma delta QP. That is, video encoder 200 and video decoder 300 may signal the value of chroma delta QP using two different syntax elements for an absolute value of the chroma delta QP value and a sign flag indicating the sign of the chroma delta QP value.

[0106] Video encoder 200 and video decoder 300 may be configured to signal the value of chroma delta QP by signaling an absolute value only without needing to signal a sign flag. That is, the values for chroma delta QP may be restricted to only values that are equal to or greater than zero, meaning the values for chroma delta QP are restricted to not include negative value.

[0107] Video encoder 200 and video decoder 300 may be configured to use the same chroma delta QP for different chroma color components can use. For example, Cb and Cr components use the same chroma delta QP.

[0108] Video encoder 200 and video decoder 300 may be configured to signal chroma delta QP for different color components separately. For example, video encoder 200 and video decoder 300 may signal a chroma delta QP for a Cb chroma component, delta QP_{cb}, and separately signal a chroma delta QP for a Cr component, chroma delta QP_{cr}.

[0109] Video encoder 200 and video decoder 300 may be configured to use signal chroma QP values using delta QP values, PPS chroma QP offsets, slice chroma QP offsets, and cu level chroma QP offsets together for chroma QP generation. For example, video decoder 300 may be configured to derive chroma QP values as follows:

$$qP_{cb} = \text{Clip3}(\text{minnum}, \text{maximum}, \text{predQP} + \text{pps_cb_qp_offset} + \text{slice_cb_qp_offset} + \text{cu_chroma_qp_offset} + \text{ChromaCuQpDeltaVal})$$

$$qP_{cr} = \text{Clip3}(\text{minnum}, \text{maximum}, \text{predQP} + \text{pps_cr_qp_offset} + \text{slice_cr_qp_offset} + \text{cu_chroma_qp_offset} + \text{ChromaCuQpDeltaVal})$$

[0110] Video encoder 200 and video decoder 300 may be configured to use chroma delta QP values, PPS chroma QP offsets, and slice chroma QP offsets to generate chroma QP. For example, video decoder 300 may be configured to derive chroma QP values as follows:

$$qPic_b = \text{Clip3}(\text{minnum}, \text{maximum}, \text{predQP} + \text{pps_cb_qp_offset} + \text{slice_cb_qp_offset} + \text{ChromaCuQpDeltaVal})$$

$$qPic_r = \text{Clip3}(\text{minnum}, \text{maximum}, \text{predQP} + \text{pps_cr_qp_offset} + \text{slice_cr_qp_offset} + \text{ChromaCuQpDeltaVal})$$

[0111] In some examples, video encoder 200 and video decoder 300 may be configured to use different combinations of chroma delta QP values, PPS chroma QP offsets, slice chroma QP offsets, and cu level chroma QP offsets to generate chroma QP.

[0112] If signaling chroma delta QP is enabled, video encoder 200 and video decoder 300 may be configured to use chroma delta QP values to determine chroma delta QP values. Otherwise, if signaling chroma delta QP is not enabled, then video encoder 200 and video decoder 300 may be configured to use SPS level chroma QP offsets, PPS level chroma QP offsets, slice level chroma QP offsets, and/or cu level chroma QP offset.

[0113] Video encoder 200 and video decoder 300 may be configured to use the QPs of neighboring chroma blocks to predict the QP of current chroma block. In one example, video encoder 200 and video decoder 300 may use the same prediction techniques used for luma QP prediction described above with respect to FIG. 5. For example, if the chroma CU is the first CU in the CTU line, then video encoder 200 and video decoder 300 may use the chroma QP of an above CU as a QP predictor. If the above CU is outside the current CTU, or otherwise not available, then video encoder 200 and video decoder 300 may use a previous chroma QP that may or may not correspond to an adjacent CU, as discussed above with respect to FIG. 5. Otherwise, video encoder 200 and video decoder 300 may use the average chroma QPs of the above neighbor chroma CU and left neighbor chroma CU, if the above and left neighbor CUs are inside the current CTU. If any of the neighboring CUs are unavailable, then video encoder 200 and video decoder 300 may use the previous chroma QP instead of the QP of neighbor CU.

[0114] An example technique for signaling chroma delta QP will now be described. Based on JVET-N1001 version 8, the newly added parts are shown between <new text> and </end new text> and the text to be removed is marked with <remove text> and </end remove text>.

transform_unit(x0, y0, tbWidth, tbHeight, treeType, subTuIndex) {	Descriptor
...	

<new text> If(treeType != DUAL_TREE_CHROMA) { </end new text>	
if((tu_cbf_luma[x0][y0] tu_cbf_cb[x0][y0] tu_cbf_cr[x0][y0]) && <remove text>treeType != DUAL_TREE_CHROMA</end remove text>) {	
if(cu_qp_delta_enabled_flag && !IsCuQpDeltaCoded) {	
cu_qp_delta_abs	ae(v)
if(cu_qp_delta_abs)	
cu_qp_delta_sign_flag	ae(v)
}	
}	
<new text> </end new text>	
else if(tu_cbf_cb[x0][y0] tu_cbf_cr[x0][y0]) {	
if(chroma_cu_qp_delta_enabled_flag && !IsChromaCuQpDeltaCoded) {	
chroma_cu_qp_delta_abs	ae(v)
if(chroma_cu_qp_delta_abs)	
chroma_cu_qp_delta_sign_flag	ae(v)
}	
}	
}	
</end new text>	
...	
}	

[0115] In this process, video encoder 200 and video decoder 300 may signal a chroma delta QP using a chroma delta QP absolute value and a chroma delta QP sign value for both Cb and Cr components. The QPs of Cb and Cr are defined as qP_{Cb} , and qP_{Cr} which can be derived as follows:

$$\text{ChromaCuQpDeltaVal} = \text{chroma_cu_qp_delta_abs} * (1 - 2 * \text{chroma_cu_qp_delta_sign_flag})$$

$$qPiCb = \text{Clip3}(-QpBdOffsetC, 69, QpY + pps_cb_qp_offset + slice_cb_qp_offset + cu_chroma_qp_offset + ChromaCuQpDeltaVal)$$

$$qPiCr = \text{Clip3}(-QpBdOffsetC, 69, QpY + pps_cr_qp_offset + slice_cr_qp_offset + cu_chroma_qp_offset + ChromaCuQpDeltaVal)$$

[0116] In this example, video encoder 200 and video decoder 300 may signal the syntax elements `chroma_cu_qp_delta_enable_flag` and `chroma_cu_qp_delta_subdiv` in a PPS for chroma. In another example, video encoder 200 and video decoder 300 can use the same flags for chroma and luma, such as, for example, `cu_qp_delta_enable_flag` and `cu_qp_delta_subdiv` for luma in VVC Draft 5.

[0117] The syntax element `chroma_cu_qp_delta_enabled_flag` equal to 1 specifies that the `chroma_cu_qp_delta_subdiv` syntax element is present in the PPS and that `chroma_cu_qp_delta_abs` may be present in the transform unit syntax. The syntax element `chroma_cu_qp_delta_enabled_flag` equal to 0 specifies that the `chroma_cu_qp_delta_subdiv` syntax element is not present in the PPS and that the syntax element `chroma_cu_qp_delta_abs` is not present in the transform unit syntax.

[0118] The syntax element `chroma_cu_qp_delta_subdiv` specifies the maximum `cbSubdiv` value of coding units that convey `chroma_cu_qp_delta_abs` and `chroma_cu_qp_delta_sign_flag`. The value range of `chroma_cu_qp_delta_subdiv` is specified as follows:

- If `slice_type` is equal to I, the value of `chroma_cu_qp_delta_subdiv` shall be in the range of 0 to $2 * (\log_2_ctu_size_minus2 - \log_2_min_qt_size_intra_slice_minus2 + \text{MaxMttDepthY})$, inclusive.

When not present, the value of `chroma_cu_qp_delta_subdiv` is inferred to be equal to 0.

[0119] In this example, for dual tree chroma, if the current chroma CB depth (`subdiv`) is equal to or smaller than a value for `chroma_cu_qp_delta_subdiv`, then video encoder 200 and video decoder 300 may not signal a chroma delta QP.

<pre><new text> if((treeType == DUAL_TREE_CHROMA && chroma_cu_qp_delta_enabled_flag && chromaCbSubdiv <= chroma_cu_qp_delta_subdiv)) {</pre>	
<pre> IsCuQpDeltaCoded = 0</pre>	

CuQpDeltaVal = 0	
CuQgTopLeftX = x0	
CuQgTopLeftY = y0	
}	
</end new text>	

[0120] FIG. 7 is a block diagram illustrating an example video encoder 200 that may perform the techniques of this disclosure. FIG. 7 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 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.

[0121] In the example of FIG. 7, video encoder 200 includes video data memory 230, mode selection unit 202, residual generation unit 204, transform processing unit 206, quantization unit 208, inverse quantization unit 210, inverse transform processing unit 212, reconstruction unit 214, filter unit 216, decoded picture buffer (DPB) 218, and entropy encoding unit 220. Any or all of video data memory 230, mode selection unit 202, residual generation unit 204, transform processing unit 206, quantization unit 208, inverse quantization unit 210, inverse transform processing unit 212, reconstruction unit 214, filter unit 216, DPB 218, and entropy encoding unit 220 may be implemented in one or more processors or in processing circuitry. Moreover, video encoder 200 may include additional or alternative processors or processing circuitry to perform these and other functions.

[0122] 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.

[0123] In this disclosure, reference to video data memory 230 should not be interpreted as being limited to memory internal to video encoder 200, unless specifically described as such, or memory external to video encoder 200, unless specifically described as such. Rather, reference to video data memory 230 should be understood as reference memory that stores video data that video encoder 200 receives for encoding (e.g., video data for a current block that is to be encoded). Memory 106 of FIG. 1 may also provide temporary storage of outputs from the various units of video encoder 200.

[0124] The various units of FIG. 7 are illustrated to assist with understanding the operations performed by video encoder 200. The units may be implemented as fixed-function circuits, programmable circuits, or a combination thereof. Fixed-function circuits refer to circuits that provide particular functionality, and are preset on the operations that can be performed. Programmable circuits refer to circuits that can be programmed to perform various tasks, and provide flexible functionality in the operations that can be performed. For instance, programmable circuits may execute software or firmware that cause the programmable circuits to operate in the manner defined by instructions of the software or firmware. Fixed-function circuits may execute software instructions (e.g., to receive parameters or output parameters), but the types of operations that the fixed-function circuits perform are generally immutable. In some examples, the 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.

[0125] 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 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.

[0126] 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.

[0127] 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.

[0128] 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.

[0129] Video encoder 200 may partition a picture retrieved from video data memory 230 into a series of CTUs, and encapsulate one or more CTUs within a slice. Mode selection unit 202 may partition a CTU of the picture in accordance with a tree structure, such as the QTBT structure or the quad-tree structure of HEVC described above. As described above, video encoder 200 may form one or more CUs from partitioning a CTU according to the tree structure. Such a CU may also be referred to generally as a “video block” or “block.”

[0130] 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.

[0131] Motion estimation unit 222 may form one or more motion vectors (MVs) that defines the positions of the reference blocks in the reference pictures relative to the position of the current block in a current picture. Motion estimation unit 222 may then provide the motion vectors to motion compensation unit 224. For example, for uni-directional inter-prediction, motion estimation unit 222 may provide a single motion vector, whereas for bi-directional inter-prediction, motion estimation unit 222 may provide two motion vectors. Motion compensation unit 224 may then generate a prediction block using the motion vectors. For example, motion compensation unit 224 may retrieve data of the reference block using the motion vector. As another example, if the motion vector has fractional sample precision, motion compensation unit 224 may interpolate values for the prediction block according to one or more interpolation filters. Moreover, for bi-directional inter-prediction, motion compensation unit 224 may retrieve data for two reference blocks identified by respective motion vectors and combine the retrieved data, e.g., through sample-by-sample averaging or weighted averaging.

[0132] 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.

[0133] Mode selection unit 202 provides the prediction block to residual generation unit 204. Residual generation unit 204 receives a raw, unencoded version of the current block from video data memory 230 and the prediction block from mode selection unit 202. Residual generation unit 204 calculates sample-by-sample differences between the current block and the prediction block. The resulting sample-by-sample differences define a residual block for the current block. In some examples, residual generation unit 204 may also determine differences between sample values in the residual block to

generate a residual block using residual differential pulse code modulation (RDPCM). In some examples, residual generation unit 204 may be formed using one or more subtractor circuits that perform binary subtraction.

[0134] 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 200 and video decoder 300 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.

[0135] In examples where mode selection unit 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 $N \times 2N$.

[0136] For other video coding techniques such as an intra-block copy mode coding, an affine-mode coding, and linear model (LM) mode coding, as 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.

[0137] 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.

[0138] 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.

[0139] 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.

[0140] 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.

[0141] 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.

[0142] Video encoder 200 stores reconstructed blocks in DPB 218. For instance, in examples where operations of filter unit 216 are not performed, reconstruction unit 214 may store reconstructed blocks to DPB 218. In examples where operations of filter unit 216 are performed, filter unit 216 may store the filtered reconstructed blocks to DPB 218. Motion estimation unit 222 and motion compensation unit 224 may retrieve a

reference picture from DPB 218, formed from the reconstructed (and potentially filtered) blocks, to inter-predict blocks of subsequently encoded pictures. In addition, intra-prediction unit 226 may use reconstructed blocks in DPB 218 of a current picture to intra-predict other blocks in the current picture.

[0143] 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.

[0144] 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.

[0145] 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.

[0146] In some examples, operations performed with respect to a luma coding block need not be repeated for the chroma coding blocks. As one example, operations to identify a motion vector (MV) and reference picture for a luma coding block need not be repeated for identifying a MV and reference picture for the chroma blocks. Rather, the MV for the luma coding block may be scaled to determine the MV for the chroma blocks, and the reference picture may be the same. As another example, the intra-

prediction process may be the same for the luma coding block and the chroma coding blocks.

[0147] Video encoder 200 represents an example of a device configured to encode video data including a memory configured to store video data, and one or more processing units implemented in circuitry and configured to perform the chroma delta QP signaling techniques described herein. Video encoder 200 may, for example, generate for inclusion in a bitstream of encode video data the syntax elements `chroma_cu_qp_delta_abs` and `chroma_cu_qp_delta_sign_flag`, as described above.

[0148] FIG. 8 is a block diagram illustrating an example video decoder 300 that may perform the techniques of this disclosure. FIG. 8 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, VVC, and HEVC. However, the techniques of this disclosure may be performed by video coding devices that are configured to other video coding standards.

[0149] In the example of FIG. 8, video decoder 300 includes coded picture buffer (CPB) memory 320, entropy decoding unit 302, prediction processing unit 304, inverse quantization unit 306, inverse transform processing unit 308, reconstruction unit 310, filter unit 312, and decoded picture buffer (DPB) 314. Any or all of CPB memory 320, entropy decoding unit 302, prediction processing unit 304, inverse quantization unit 306, inverse transform processing unit 308, reconstruction unit 310, filter unit 312, and DPB 314 may be implemented in one or more processors or in processing circuitry. Moreover, video decoder 300 may include additional or alternative processors or processing circuitry to perform these and other functions.

[0150] Prediction processing unit 304 includes motion compensation unit 316 and intra-prediction unit 318. Prediction processing unit 304 may include addition 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.

[0151] CPB memory 320 may store video data, such as an encoded video bitstream, to be decoded by the components of video decoder 300. The video data stored in CPB memory 320 may be obtained, for example, from computer-readable medium 110 (FIG.

1). CPB memory 320 may include a CPB that stores encoded video data (e.g., syntax elements) from an encoded video bitstream. Also, CPB memory 320 may store video data other than syntax elements of a coded picture, such as temporary data representing outputs from the various units of video decoder 300. DPB 314 generally stores decoded pictures, which video decoder 300 may output and/or use as reference video data when decoding subsequent data or pictures of the encoded video bitstream. CPB memory 320 and DPB 314 may be formed by any of a variety of memory devices, such as DRAM, including SDRAM, MRAM, RRAM, or other types of memory devices. CPB memory 320 and DPB 314 may be provided by the same memory device or separate memory devices. In various examples, CPB memory 320 may be on-chip with other components of video decoder 300, or off-chip relative to those components.

[0152] 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.

[0153] The various units shown in FIG. 8 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. 7, fixed-function circuits refer to circuits that provide particular functionality, and are preset on the operations that can be performed. Programmable circuits refer to circuits that can be programmed to perform various tasks, and provide flexible functionality in the operations that can be performed. For instance, programmable circuits may execute software or firmware that cause the programmable circuits to operate in the manner defined by instructions of the software or firmware. Fixed-function circuits may execute software instructions (e.g., to receive parameters or output parameters), but the types of operations that the fixed-function circuits perform are generally immutable. In some examples, the 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.

[0154] 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.

[0155] 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.

[0156] 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”).

[0157] 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.

[0158] 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 coefficient block.

[0159] 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. 7).

[0160] 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. 7). Intra-prediction unit 318 may retrieve data of neighboring samples to the current block from DPB 314.

[0161] 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.

[0162] 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.

[0163] Video decoder 300 may store the reconstructed blocks in 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 from DPB 314 for subsequent presentation on a display device, such as display device 118 of FIG. 1.

[0164] In this manner, video decoder 300 represents an example of a video decoding device including a memory configured to store video data, and one or more processing units implemented in circuitry and configured to perform the chroma delta QP signaling techniques described herein. Video decoder 300 may, for example, decode and parse the syntax elements `chroma_cu_qp_delta_abs` and `chroma_cu_qp_delta_sign_flag`, as described above.

[0165] Entropy decoding unit 302 may, for example, receive, in a bitstream of encoded video data, first syntax indicating a luma delta QP value for the luma component and second syntax indicating a chroma delta QP value for the chroma component. Inverse

quantization unit 306 may, for example, determine a predicted luma quantization parameter (QP) for a luma component of a coding unit; determine a QP value for the luma component based on the predicted luma QP and the luma delta QP value; determine a predicted chroma QP for a chroma component of the coding unit; determine a QP value for the chroma component based on the predicted chroma QP and the chroma delta QP value; dequantize a block of luma transform coefficients based on the QP value for the luma component; and dequantize a block of chroma transform coefficients based on the QP value of the chroma component. Video decoder 300 can decode the coding unit based on the dequantized block of luma transform coefficients and the dequantized block of chroma transform coefficients.

[0166] FIG. 9 is a flowchart illustrating an example method for encoding a current block. The current block may comprise a current CU. Although described with respect to video encoder 200 (FIGS. 1 and 7), it should be understood that other devices may be configured to perform a method similar to that of FIG. 9.

[0167] In this example, video encoder 200 initially predicts the current block (350). For example, video encoder 200 may form a prediction block for the current block. Video encoder 200 may then calculate a residual block for the current block (352). To calculate the residual block, video encoder 200 may calculate a difference between the original, unencoded block and the prediction block for the current block. Video encoder 200 may then transform and quantize coefficients of the residual block (354). Video encoder 200 may, for example, signal QP values for luma components and chroma components of the video data using techniques described above. Next, video encoder 200 may scan the quantized transform coefficients of the residual block (356). During the scan, or following the scan, video encoder 200 may entropy encode the coefficients (358). For example, video encoder 200 may encode the coefficients using CAVLC or CABAC. Video encoder 200 may then output the entropy coded data of the block (360).

[0168] FIG. 10 is a flowchart illustrating an example method for decoding a current block of video data. The current block may comprise a current CU. Although described with respect to video decoder 300 (FIGS. 1 and 8), it should be understood that other devices may be configured to perform a method similar to that of FIG. 10.

[0169] 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). 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, for example, receive syntax elements indicating QP values for luma components and chroma components of the video data using techniques described above. Video decoder 300 may ultimately decode the current block by combining the prediction block and the residual block (380).

[0170] FIG. 11 is a flowchart illustrating an example method for encoding a current block. The current block may comprise a current CU. Although described with respect to video encoder 200 (FIGS. 1 and 7), it should be understood that other devices may be configured to perform a method similar to that of FIG. 11.

[0171] In this example, video encoder 200 determines a QP value for a luma component of a coding unit of the video data (400). Video encoder 200 may quantize a block of luma transform coefficients based on the QP value for the luma component of the coding unit. To signal the QP value for the luma component of the coding unit, video encoder 200 determines a predicted luma QP for the luma component of the coding unit (402). Video encoder 200 determines a delta QP value for the luma component of the coding unit based on the QP value for the luma component and the predicted luma QP (404). Video encoder 200 generates, for inclusion in a bitstream of encoded video data, first syntax indicating the delta QP value for the luma component of the coding unit (406).

[0172] Video encoder 200 determines a QP value for a chroma component of the coding unit of the video data (408). Video encoder 200 may quantize a block of chroma transform coefficients based on the QP value of the chroma component of the coding unit. To signal the QP value for the chroma component of the coding unit, video encoder 200 determines a predicted chroma QP for the chroma component of the coding unit (410). Video encoder 200 determines a delta QP value for the chroma component of the coding unit based on the QP value for the chroma component and the predicted chroma QP (412).

[0173] In some examples, a CTU that includes the coding unit may be partitioned using a single tree structure, such that a luma component of the CTU and a chroma component of the CTU have a same partitioning. In other examples, a CTU that includes the coding unit may be partitioned using a dual tree structure, such that a luma component of the CTU and a chroma component of the CTU have a different partitioning. If the CTU that includes the coding unit is partitioned using a dual tree structure, video encoder 200 may be configured to determine a maximum chroma delta QP signaling depth for the chroma component of the CTU; determine that the coding unit and at least one other coding unit belong to a split node of the chroma component of the CTU; in response to the maximum chroma delta QP signaling depth corresponding to the split node, determine a QP value for a chroma component of the at least one other coding unit to be equal to the QP value for the chroma component of the coding unit; and quantize a block of chroma transform coefficients for the at least one other CU based on the QP value for the chroma component of the at least one other coding unit.

[0174] Video encoder 200 generates, for inclusion in the bitstream of the encoded video data, second syntax indicating the delta QP value for the chroma component of the coding unit (414). In some examples, video encoder 200 may generate, for inclusion in the bitstream of encoded video data, the second syntax indicating the chroma delta QP value for the chroma component in response to determining that signaling of chroma delta QP values is enabled for the coding unit. The second syntax may, for example, include a syntax element indicating an absolute value of the chroma delta QP value and a syntax element indicating a sign for the chroma delta QP value.

[0175] Video encoder 200 outputs, in the bitstream of encoded video data, the first syntax and the second syntax (416). Video encoder 200 may also output, in the bitstream of encoded video data, syntax indicating values for the quantized block of luma transform coefficients and the quantized block of chroma transform coefficients.

[0176] Video encoder 200 may also determine a second QP value for a second chroma component of the coding unit; determine a second predicted chroma QP for the second chroma component of the coding unit; based on the second QP value for the second chroma component and the second predicted chroma QP, determine a second delta QP value for the second chroma component; generating, for inclusion in the bitstream of the encoded video data, third syntax indicating the second delta QP value for the second chroma component; and output, in the bitstream of encoded video data, the third syntax.

[0177] FIG. 12 is a flowchart illustrating an example method for decoding a current block of video data. The current block may comprise a current CU. Although described with respect to video decoder 300 (FIGS. 1 and 8), it should be understood that other devices may be configured to perform a method similar to that of FIG. 10.

[0178] Video decoder 300 determines a predicted luma QP for a luma component of a coding unit (420). Video decoder 300 may, for example, determine the predicted luma QP based on the QPs of neighboring CUs using any of the techniques described above.

[0179] Video decoder 300 receives, in the bitstream of encoded video data, first syntax indicating a luma delta QP value for the luma component (422). The first syntax may, for example, include a syntax element that indicates an absolute value of the luma delta QP value and a syntax element that indicates a sign for the luma delta QP value. Video decoder 300 determines a QP value for the luma component based on the predicted luma QP and the luma delta QP value (424). Video decoder 300 may, for example, add the luma delta QP value to a value of the predicted luma QP to determine the QP value for the luma component of the coding unit.

[0180] Video decoder 300 determines a predicted chroma QP for a chroma component of the coding unit (426). Video decoder 300 may, for example, determine the predicted chroma QP based on the QPs of neighboring CUs using any of the techniques described above. Video decoder 300 receives, in the bitstream of encoded video data, second syntax indicating a chroma delta QP value for the chroma component of the coding unit (428). Video decoder 300 may, for example, receive, in the bitstream of encoded video data, the second syntax indicating the chroma delta QP value for the chroma component in response to determining that signaling of chroma delta QP values is enabled for the coding unit. The second syntax may, for example, include a syntax element that indicates an absolute value of the chroma delta QP value and a syntax element that indicates a sign for the chroma delta QP value.

[0181] In some examples, the CTU that include the coding unit may be partitioned using a single tree structure, such that a luma component of the CTU and a chroma component of the CTU have a same partitioning. In other examples, a CTU that includes the coding unit may be partitioned using a dual tree structure, such that a luma component of the CTU and a chroma component of the CTU have a different partitioning. In examples, where the luma component of the CTU and the chroma component of the CTU have a different partitioning, then to receive the second syntax indicating the chroma delta QP value for the chroma component, video decoder 300

may, for example, determine a maximum chroma delta QP signaling depth for the chroma component of the CTU; determine that the coding unit and at least one other coding unit belong to a split node of the chroma component of the CTU; in response to the maximum chroma delta QP signaling depth corresponding to the split node, determine a QP value for a chroma component of the at least one other coding unit based on the chroma delta QP value; and dequantize a block of chroma transform coefficients for the at least one other CU based on the QP value for the chroma component of the at least one other coding unit.

[0182] Video decoder 300 determines a QP value for the chroma component of the coding unit based on the predicted chroma QP and the chroma delta QP value (430). Video decoder 300 may, for example, add the chroma delta QP value to a value of the predicted chroma QP to determine the QP value for the chroma component of the coding unit.

[0183] Video decoder 300 dequantizes a block of luma transform coefficients for the coding unit based on the QP value for the luma component (432) and dequantizes a block of chroma transform coefficients for the coding unit based on the QP value of the chroma component (434). Video decoder 300 decodes the coding unit based on the dequantized block of luma transform coefficients and the dequantized block of chroma transform coefficients (436).

[0184] To decode the coding unit based on the dequantized block of luma transform coefficients and the dequantized block of chroma transform coefficients, video decoder 300 may inverse transform the dequantized block of luma transform coefficients to determine a luma residual block; inverse transform the dequantized block of chroma transform coefficients to determine a chroma residual block; determine a luma prediction block; determine a chroma prediction block; adding the luma residual block to the luma prediction block to determine a reconstructed luma block of the coding unit; and add the chroma residual block to the chroma prediction block to determine a reconstructed chroma block of the coding unit.

[0185] Video decoder 300 may also determine a second predicted chroma QP for a second chroma component of the coding unit; receive, in the bitstream of encoded video data, third syntax indicating a second chroma delta QP value for the second chroma component of the coding unit; determine a second QP value for the second chroma component of the coding unit based on the second predicted chroma QP and the second chroma delta QP value; dequantize a second block of chroma transform coefficients for

the coding unit based on the second QP value of the second chroma component; and decode the coding unit based on the dequantized second block of chroma transform coefficients.

[0186] 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.

[0187] 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.

[0188] By way of example, and not limitation, such computer-readable storage media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage, or other magnetic storage devices, flash memory, or any other medium that can be used to store desired program code in the form of instructions or data structures and that can be accessed by a computer. Also, any connection is properly termed a computer-readable medium. For example, if instructions are transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared, radio, and microwave, then the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio, and microwave are included in the definition of medium. It should be understood, however, that computer-readable storage media and

data storage media do not include connections, carrier waves, signals, or other transitory media, but are instead directed to non-transitory, tangible storage media. Disk and disc, as used herein, includes compact disc (CD), laser disc, optical disc, digital versatile disc (DVD), floppy disk and Blu-ray disc, where disks usually reproduce data magnetically, while discs reproduce data optically with lasers. Combinations of the above should also be included within the scope of computer-readable media.

[0189] 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 terms “processor” and “processing circuitry,” as used herein may refer to any of the foregoing structures or any other structure suitable for implementation of the techniques described herein. In addition, in some aspects, the functionality described herein may be provided within dedicated hardware and/or software modules configured for encoding and decoding, or incorporated in a combined codec. Also, the techniques could be fully implemented in one or more circuits or logic elements.

[0190] 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.

[0191] Various examples have been described. These and other examples are within the scope of the following claims.