

BLOCK LEVEL SIGNALLING OF CHROMA QUANTIZATION PARAMETER OFFSETS IN VIDEO CODING

[0001] This application claims priority to U.S. Application No. **16/918,741**, filed July 1, 2020 which claims the benefit of U.S. Provisional Application No. **62/871,028**, filed July 5, 2019, the entire contents of each of which are hereby 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] In general, this disclosure describes techniques for determining chroma quantization parameter (QP). The techniques may be applied to any of the existing video codecs, such as HEVC (High Efficiency Video Coding), VVC (Versatile Video Coding), or be an efficient coding tool in any future video coding standards.

[0006] In video coding, a chroma QP indicates an amount of quantization that a video encoder applies to coefficient values of a residual block and an amount of inverse-quantization that a video decoder applies to quantized coefficient values to generate the residual block. In some examples, rather than a video encoder signaling the chroma QP, a video decoder may utilize a QP predictor, which may be based on a luma QP of a corresponding luma block to the chroma block, and one or more quantization parameter offsets to determine the chroma QP.

[0007] This disclosure describes example techniques of block-level signaling of quantization parameter offsets. Such block-level signaling of quantization parameter offsets provides block level flexibility to determine a more precise chroma QP for a chroma block. For instance, some techniques relied on quantization parameter offsets that are signaled at a picture level or slice level, so that the quantization parameter offsets were the same for each chroma block in the picture or slice. Such high level signaling (e.g., at picture and/or slice level) limits granularity in defining the chroma QP for a chroma block. With the block-level quantization parameter offset signaling described in this disclosure, there is more flexibility in defining the chroma QP, resulting in more accurate determination of chroma QP on a chroma block-by-chroma block basis.

[0008] In this way, the example techniques provide technical solutions to a technical problem by providing signaling that increases flexibility to more accurately define the chroma QP. The example techniques described in more detail provide a practical application to video coding, such as a way in which to determine the chroma QP that is more accurate, resulting in better video coding, as compared to other techniques that rely on high level signaling such as picture level or slice level signaling to define the offsets for determining the chroma QP.

[0009] In one example, the disclosure describes a method of decoding video data, the method comprising determining a quantization parameter predictor for a chroma block of the video data based on a quantization parameter for a corresponding luma block,

determining a block level quantization parameter offset for the chroma block, determining a quantization parameter for the chroma block based on the block level quantization parameter offset and the quantization parameter predictor, determining a residual block based on the quantization parameter, and reconstructing the chroma block based on the residual block.

[0010] In one example, the disclosure describes a device for decoding video data, the device comprising memory configured to store a quantization parameter for a corresponding luma block of a chroma block of the video data and processing circuitry coupled to the memory and configured to determine a quantization parameter predictor for the chroma block of the video data based on the quantization parameter for the corresponding luma block, determine a block level quantization parameter offset for the chroma block, determine a quantization parameter for the chroma block based on the block level quantization parameter offset and the quantization parameter predictor, determine a residual block based on the quantization parameter, and reconstruct the chroma block based on the residual block.

[0011] In one example, the disclosure describes a computer-readable storage medium storing instructions thereon that when executed cause one or more processors of a device for decoding video data to determine a quantization parameter predictor for a chroma block of the video data based on a quantization parameter for a corresponding luma block, determine a block level quantization parameter offset for the chroma block, determine a quantization parameter for the chroma block based on the block level quantization parameter offset and the quantization parameter predictor, determine a residual block based on the quantization parameter, and reconstruct the chroma block based on the residual block.

[0012] In one example, the disclosure describes a device for decoding video data, the device comprising means for determining a quantization parameter predictor for a chroma block of the video data based on a quantization parameter for a corresponding luma block, means for determining a block level quantization parameter offset for the chroma block, means for determining a quantization parameter for the chroma block based on the block level quantization parameter offset and the quantization parameter predictor, means for determining a residual block based on the quantization parameter, and means for reconstructing the chroma block based on the residual block.

[0013] In one example, the disclosure describes a method of encoding video data, the method comprising determining a quantization parameter predictor for a chroma block

of the video data based on a quantization parameter for a corresponding luma block, determining a block level quantization parameter offset for the chroma block, determining a quantization parameter for the chroma block based on the block level quantization parameter offset and the quantization parameter predictor, quantizing coefficient values for a residual block based on the determined quantization parameter for the chroma block, and signaling information indicative of the quantized coefficient values.

[0014] In one example, the disclosure describes a device for encoding video data, the device comprising memory configured to store a quantization parameter for a corresponding luma block of a chroma block of the video data and processing circuitry coupled to the memory and configured to determine a quantization parameter predictor for a chroma block of the video data based on a quantization parameter for a corresponding luma block, determine a block level quantization parameter offset for the chroma block, determine a quantization parameter for the chroma block based on the block level quantization parameter offset and the quantization parameter predictor, quantize coefficient values for a residual block based on the determined quantization parameter for the chroma block, and signal information indicative of the quantized coefficient values.

[0015] 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

[0016] FIG. 1 is a block diagram illustrating an example video encoding and decoding system that may perform the techniques of this disclosure.

[0017] FIGS. 2A and 2B are conceptual diagrams illustrating an example quadtree binary tree (QTBT) structure, and a corresponding coding tree unit (CTU).

[0018] FIG. 3 is a block diagram illustrating an example video encoder that may perform the techniques of this disclosure.

[0019] FIG. 4 is a block diagram illustrating an example video decoder that may perform the techniques of this disclosure.

[0020] FIG. 5 is a conceptual diagram illustrating an example of a chroma coding unit (CU) and its corresponding luma CUs in separate tree.

[0021] FIG. 6 is a flowchart illustrating an example method for encoding a current block of video data.

[0022] FIG. 7 is a flowchart illustrating an example method for decoding a current block of video data.

DETAILED DESCRIPTION

[0023] In video coding, a video encoder determines a prediction block for a current block being encoded and determines a residual block indicative of difference between the prediction block and the current block. The video encoder may perform a transform that transforms the residual values to transform coefficients. The transform may be skipped in some cases. After transform (or where transform is skipped), the video encoder performs a quantization of the coefficients. The video encoder signals the quantized coefficients to the video decoder which performs the reciprocal operations as the video encoder to generate the residual block. Video decoder utilizes the same techniques as the video encoder to determine the prediction block and adds the residual block to the prediction block to reconstruct the current block.

[0024] The video coding process is performed on luma and chroma components. In other words, the current block includes a luma block and one or more chroma blocks. The chroma block(s) may be subsampled relative to the luma block based on the particular color format.

[0025] A quantization parameter determines how much quantization the video encoder applies and amount of inverse quantization the video decoder applies. In some examples, rather than signaling the quantization parameter for a chroma block, the quantization parameter for a luma block corresponding to the chroma block may be used to determine a quantization parameter predictor for the quantization parameter for the chroma block. For example, the quantization parameter for the luma block is the quantization parameter predictor, but other ways to determine the quantization parameter predictor based on the quantization parameter for the luma block are possible (e.g., such as average of quantization parameters of neighboring luma blocks). In such examples, for a chroma block, the video encoder may signal offsets that the video decoder adds to the quantization parameter predictor.

[0026] In some techniques, the offsets that the video encoder signals are offsets that are constant for the entire picture or entire slice of the current block. For example, the

offset may be pps_cb_qp_offset for the Cb chroma block or pps_cr_qp_offset for the Cr chroma block. These example offsets are at a picture level (e.g., applicable to blocks in the picture). Another example is the $\text{slice_cb_qp_offset}$ for the Cb chroma block or $\text{slice_cr_qp_offset}$ for the Cr chroma block. These example offsets are at a slice level (e.g., applicable to blocks in the slice). Accordingly, if the Q_{pY} (e.g., QP for luma) is the quantization parameter predictor, then in some techniques, the Q_{pCR} (quantization parameter for Cr chroma block) is $Q_{pY} + \text{pps_cb_qp_offset} + \text{slice_cb_qp_offset}$, clipped to within a specified range. The Q_{pCB} (quantization parameter for Cb chroma block) is $Q_{pY} + \text{pps_cr_qp_offset} + \text{slice_cr_qp_offset}$, clipped to within a specified range. It should be understood that Q_{pY} being the quantization parameter predictor is merely one example and should not be considered limiting. In some examples, the quantization parameter predictor may be determined from Q_{pY} , and may not explicitly be Q_{pY} .

[0027] There may be problems with such techniques. The picture level and slice level offsets (e.g., pps_cb or cr_qp_offset and slice_cb or cr_qp_offset , respectively) may not provide sufficient flexibility to allow the video decoder to accurately determine the quantization parameter for a chroma block. For example, if the actual chroma block quantization parameter for the Cb chroma block is different than $Q_{pY} + \text{pps_cb_qp_offset} + \text{slice_cb_qp_offset}$, there may not be a way for the video encoder to further refine the quantization parameter for the Cb chroma block because the quantization parameter for the Cb chroma block may be limited to three factors: Q_{pY} , pps_cb_qp_offset , and $\text{slice_cb_qp_offset}$.

[0028] This disclosure describes block level signaling of offset that the video encoder or video decoder can add to a quantization parameter predictor to determine a quantization parameter for a chroma block. As one example, the video encoder and the video decoder each construct a list of quantization parameter offsets. The video encoder signals an index into the list of quantization parameter offsets, and the video decoder determines the block level quantization parameter offset based on the index into the list of quantization parameter offsets and adds the offset to the quantization parameter predictor.

[0029] For example, assume that CuQpOffsetCb is the block level quantization parameter offset that the video decoder determines based on an index into the list of quantization parameter offsets for the Cb chroma block, and CuQpOffsetCr is the block level quantization parameter offset that the video decoder determines based on the index

into the list of quantization parameter offsets for the Cr chroma block. For such examples, the quantization parameter for the Cb chroma block may be $Q_{pY} + pps_cb_qp_offset + slice_cb_qp_offset + CuQpOffsetCb$, and the quantization parameter for the Cr chroma block may be $Q_{pY} + pps_cr_qp_offset + slice_cr_qp_offset + CuQpOffsetCr$. The same may apply for joint Cb Cr blocks.

[0030] In this way, $CuQpOffsetCb$ and $CuQpOffsetCr$ provide block level flexibility for determining the quantization parameter for Cb and Cr chroma blocks. For instance, without $CuQpOffsetCb$ and $CuQpOffsetCr$, the quantization parameter for Cb and Cr chroma blocks would be limited by the quantization parameter predictor and quantization parameter offsets signaled at picture level or slice level, and there would not be a way in which to define the quantization parameter based on block level quantization parameter offsets. With the example techniques described in this disclosure, the video encoder can signal information to define block level quantization parameter offsets, rather than being limited to picture level or slice level signaling of quantization parameter offsets.

[0031] For example, the block level quantization parameter offset for two chroma blocks (e.g., two Cb chroma blocks or two Cr chroma blocks) in the same picture or slice may be different, and the block level quantization parameter offsets may be determined on a chroma block-by-chroma block basis. The $pps_cb_qp_offset$ and $pps_cr_qp_offset$ are picture level offset, so that two Cb chroma blocks or two Cr chroma blocks in the same picture have the same $pps_cb_qp_offset$ or the same $pps_cr_qp_offset$, respectively. The $slice_cb_qp_offset$ and $slice_cr_qp_offset$ are slice level offsets, so that two Cb chroma blocks or two Cr chroma blocks in the same slice have the same $slice_cb_qp_offset$ or $slice_cr_qp_offset$, respectively.

[0032] 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.

[0033] 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, 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.

[0034] 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 signaling and parsing chroma delta quantization parameter (QP). 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 including an integrated display device.

[0035] 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 signaling and parsing chroma delta QP. 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, devices 102, 116 may operate in a substantially symmetrical manner such that each of devices 102, 116 include video encoding and decoding components. Hence, system 100 may support one-way or two-way video transmission between video devices 102, 116, e.g., for video streaming, video playback, video broadcasting, or video telephony.

[0036] 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.

[0037] 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 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.

[0038] 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.

[0039] 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.

[0040] 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.

[0041] 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.

[0042] 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.

[0043] 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.

[0044] 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.

[0045] Input interface 122 of destination device 116 receives an encoded video bitstream from computer-readable medium 110 (e.g., 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 cathode ray tube (CRT), a liquid crystal display (LCD), a plasma display, an organic light emitting diode (OLED) display, or another type of display device.

[0046] 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).

[0047] 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.

[0048] 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. Description of HEVC is available at G. J. Sullivan; J.-R. Ohm; W.-J. Han; T. Wiegand (December 2012). "Overview of the High Efficiency Video Coding (HEVC) Standard" (PDF). IEEE Transactions on Circuits and Systems for Video Technology (IEEE) 22 (12). Retrieved 2012-09-14.

[0049] 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"). A more recent 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-v8 (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) partitions. A triple tree partition is a partition where a block is split into three sub-blocks. In some examples, a triple 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] 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.

[0058] 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.

[0059] 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.

[0060] 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.

[0061] 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.

[0062] 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.

[0063] VVC may 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.

[0064] To perform intra-prediction, video encoder 200 may select an intra-prediction mode to generate the prediction block. VVC may 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).

[0065] 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.

[0066] 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.

[0067] 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 coefficients, providing further compression. By performing the quantization process, video encoder 200 may reduce the bit depth associated with some or all of the 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.

[0068] This disclosure describes example ways in which video encoder 200 may determine the quantization parameter (QP) for a chroma block for quantizing. The QP may be a value that defines an amount of quantization that is to be applied the transform coefficients. In some examples, video encoder 200 may determine the actual QP for a chroma block (e.g., Cb or Cr chroma block) based on rate-distortion measurements and signal information indicative of the actual QP.

[0069] However, to reduce the number of bits that need to be signaled (e.g., increase bandwidth), video encoder 200 may utilize a QP predictor for the chroma block. In some examples, the QP predictor may be the QP for the luma block corresponding to the chroma block or based on QPs of a plurality of luma blocks. For instance, as described above, a coding unit may include a luminance component and/or a chrominance component. A luma block includes the samples of the luminance component, and a chroma block includes the samples of the chrominance component. Accordingly, for a coding unit there may be one luma block and two chroma blocks (e.g., Cb chroma block and Cr chroma block). A luma block and the two chroma blocks that together form the coding unit may be referred to as corresponding to one another because of the location of the chroma block and the location of the luma block overlap.

[0070] Based on the color format, the size of the chroma blocks may be different than the size of the luma blocks (e.g., the chroma blocks are downsampled relative to the luma blocks). For instance, in 4:2:0 color format, the luma block is $N \times N$ and in size the chroma block is $N/2 \times N/2$ in size. In this example, one chroma block of size $N \times N$ would correspond with four $N \times N$ sized luma blocks.

[0071] In some examples, the QP for the luma block (e.g., luma QP) may be a QP predictor for the QPs for both of the corresponding chroma blocks. In some examples, a plurality of luma blocks may correspond to one chroma block, and some average of the luma QPs of the plurality of luma blocks may be a QP predictor for the QPs for both of the corresponding chroma blocks. In any event, video encoder 200 may determine a quantization parameter predictor for a chroma block of video data based on a quantization parameter for a corresponding luma block.

[0072] Video encoder 200 may also determine an offset value, such that when the offset value is added to the QP predictor, the result is the quantization parameter for the chroma block. In accordance with one or more examples described in this disclosure, video encoder 200 may determine block level quantization parameter offset for a chroma block that when added to the QP predictor (possibly with some additional offsets), the result is the quantization parameter for a chroma block.

[0073] Block level quantization parameter offset may refer to quantization parameter offset that can be different on block-by-block basis. For instance, in some techniques, video encoder 200 may be limited to signaling high-level offsets, such as picture level offsets or slice level offsets, that are the same for all chroma blocks in a picture or slice. In some of those techniques, there may be no way in which to define, at a block level, what the offset should be. That is, there may not be block level granularity to define the offset for a chroma block. Rather, the offset would be the same for all blocks in the picture or slice.

[0074] In some of those techniques, where the offset is the same for all blocks in the picture or slice, utilizing the same offset to determine the quantization parameters for each of the chroma blocks may result in some of the chroma blocks having suboptimal quantization parameters. For instance, if the slice level or picture level offset is equal to A and B, respectively, Y1 is the quantization parameter predictor for a first chroma block in the slice, and Y2 is the quantization parameter predictor for a second chroma block in the slice, then the quantization parameter for the first chroma block would be $Y1 + A + B$ and the quantization parameter for the second chroma block would be $Y2 + A + B$. In some of those techniques, there may not be a way in which to define a block level quantization parameter offset for the first chroma block that is different than the block level quantization parameter offset for the second chroma block. Rather, the quantization parameter offset is limited to A and B for both the first and the second chroma blocks.

[0075] In accordance with techniques described in this disclosure, video encoder 200 may determine a block level quantization parameter offset for a chroma block, and determine a quantization parameter for the chroma block based on the block level quantization parameter offset and the quantization parameter predictor. For instance, there may be N number of possible block level quantization parameter offsets to select from, and video encoder 200 may select a block level quantization parameter offset that when added to the quantization parameter predictor (possibly with other offsets as well) results in the best quantization parameter for the chroma block. Video encoder 200 may then quantize coefficient values for a residual block based on the determined quantization parameter for the chroma block, and signal information indicative of the quantized coefficient values.

[0076] As described above, there may be two chroma blocks (e.g., Cb and Cr chroma blocks). In one or more examples, video encoder 200 may perform similar operations for both Cb chroma block and Cr chroma block. In some examples, there may be a joint chroma block that is a combination of the Cb and Cr chroma blocks (e.g., JointCbCr chroma block), and the example techniques described in this disclosure are applicable to the JointCbCr chroma block as well. Accordingly, in the above examples that refer to a chroma block, the chroma block may be a Cb chroma block, a Cr chroma block, or a JointCbCr chroma block.

[0077] 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) 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.

[0078] 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.

[0079] 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.

[0080] 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.

[0081] In general, video decoder 300 performs a reciprocal process to that performed by video encoder 200 to decode the encoded video data of the bitstream. For example, video decoder 300 may decode values for syntax elements of the bitstream using CABAC in a manner substantially similar to, albeit reciprocal to, the CABAC encoding process of video encoder 200. The syntax elements may define partitioning information of a picture into CTUs, and partitioning of each CTU according to a corresponding partition structure, such as a QTBT structure, to define CUs of the CTU. The syntax elements may further define prediction and residual information for blocks (e.g., CUs) of video data.

[0082] 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. For example, video decoder 300 may be configured to determine a quantization parameter for a chroma block using the example techniques described in this disclosure.

[0083] As described above, video encoder 200 may be configured to determine a quantization parameter predictor for a chroma block. Video decoder 300 may perform similar operations to determine a quantization parameter predictor for a chroma block of video data based on a quantization parameter for a corresponding luma block. The quantization parameter predictor may be the quantization parameter of the corresponding luma block or may be some average of the quantization parameters of a

plurality of luma blocks that may correspond to the chroma block. In general, the quantization parameter predictor may be determined based on the quantization parameter of the corresponding luma block, and need not necessarily be the quantization parameter of the corresponding luma block.

[0084] Video decoder 300 may also determine a block level quantization parameter offset for the chroma block. For instance, video encoder 200 may signal information to video decoder 300 that video decoder 300 uses to determine the block level quantization parameter offset. As one example, as described above, there may be N number of possible block level quantization parameter offsets for video encoder 200 to select from, and video encoder 200 may signal information indicating which one of the N number of possible block level quantization parameter offsets video decoder 300 is to select.

[0085] To determine the quantization parameter for the chroma block, video decoder 300 may determine a quantization parameter for the chroma block based on the block level quantization parameter offset and the quantization parameter predictor. For instance, video decoder 300 may add the block level quantization parameter offset to the quantization parameter predictor (and possibly add some additional other offsets) to determine the quantization parameter for the chroma block.

[0086] Video decoder 300 may determine a residual block based on the quantization parameter. For instance, video decoder 300 may perform the inverse quantization based on the quantization parameter to determine the residual block.

[0087] Video decoder 300 may use 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. For example, video decoder 300 may reconstruct the chroma block based on the residual block. Video decoder 300 may perform additional processing, such as performing a deblocking process to reduce visual artifacts along boundaries of the block.

[0088] 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.

[0089] As described above, in one or more examples, video encoder 200 and video decoder 300 may be configured to determine quantization parameters for chroma blocks using block level quantization parameter offsets. For instance, video encoder 200 and video decoder 300 may utilize similar techniques to determine a quantization parameter predictor for a chroma block of the video data based on a quantization parameter for a corresponding luma block. The quantization parameter predictor may be the quantization parameter of the corresponding luma block or may be an average of quantization parameters of a plurality of corresponding luma blocks, or some other technique that utilizes the quantization parameter of the corresponding luma block may be used to determine the quantization parameter predictor.

[0090] Video encoder 200 and video decoder 300 may determine a block level quantization parameter offset for the chroma block. The block level quantization parameter offset for the chroma block may mean that the block level quantization parameter offset can be different for different chroma blocks in same slice or picture. For instance, a slice may include two or more chroma blocks. In one or more examples, a first block level quantization parameter offset for a first chroma block in the slice and a second block level quantization parameter offset for a second chroma block in the same slice may be different.

[0091] In one or more examples, video encoder 200 and video decoder 300 may determine a quantization parameter for the chroma block based on the block level quantization parameter offset and the quantization parameter predictor. In addition to the block level quantization parameter offset, there may be a first quantization parameter offset in a picture parameter set and a second quantization parameter offset in a slice parameter set (e.g., slice header). For instance, video encoder 200 may signal and video decoder 300 may receive the first quantization parameter offset from a picture parameter set (PPS), and video encoder 200 may signal and video decoder 300 may receive the second quantization parameter from a slice parameter set (e.g., information signaled at the slice level). Video encoder 200 and video decoder 300 may add the first quantization parameter offset, the second quantization parameter offset, the block level quantization parameter offset, and the quantization parameter predictor to determine the quantization parameter for the chroma block.

[0092] Based on the quantization parameter, video encoder 200 and video decoder 300 may perform quantizing or inverse-quantizing. For example, video encoder 200 may quantize coefficient values for a residual block based on the determined quantization parameter for the chroma block. Video decoder 300 may inverse-quantize received values to generate the residual block based on the determined quantization parameter for the chroma block. Video encoder 200 may signal information indicative of the quantized coefficient values. Video decoder 300 may add the residual block to a prediction block to reconstruct the chroma block.

[0093] As described, video encoder 200 and video decoder 300 may be configured to determine the block level quantization parameter offset for the chroma block. In some examples, there may be N block level quantization parameter offsets from which to select the block level quantization parameter offset for the chroma block. For instance, video encoder 200 may construct a list of quantization parameter offsets, which include the N block level quantization parameter offsets from which to select the block level quantization parameter offset for a chroma block. Video encoder 200 may signal information indicating the values in the list of quantization parameter offsets to video decoder 300, and video decoder 300 may construct the list of quantization parameter offsets based on the signaled information.

[0094] However, in some examples, the list of quantization parameter offsets may be predefined and pre-stored. In some examples, video encoder 200 and video decoder 300 may each follow similar operations to construct the list of quantization parameter offsets using implicit techniques that do not require video encoder 200 to signal to video decoder 300 the information for constructing the list of quantization parameter offsets. There may be various ways in which to construct the list of quantization parameter offsets (e.g., signaling, pre-stored, or implicit), and the example techniques are not limited to any particular way in which video encoder 200 and video decoder 300 construct the list of quantization parameter offsets.

[0095] In one or more examples, video encoder 200 may determine which quantization parameter offset to select from the list of quantization parameter offsets and signal an index into the list of quantization parameter offsets. Video decoder 300 may receive an index into the list of quantization parameter offsets and determine the block level quantization parameter offset based on the index into the list of quantization parameter offsets.

[0096] Video encoder 200 and video decoder 300 may perform similar operations to determine block level quantization parameter offsets for each of the chroma blocks for the different chroma components. For instance, video encoder 200 and video decoder 300 may construct a first list of quantization parameter offsets for the Cb chroma block and a second list of quantization parameter offsets for the Cr chroma block.

[0097] For instance, there may be a luma block and two chroma blocks (e.g., one chroma block for each chroma component). There may be a Cb chroma block (e.g., a first chroma block of a first chroma component) and a Cr chroma block (e.g., a second chroma block of a second chroma component). In this example, the Cb chroma block and the Cr chroma block may share the same quantization parameter predictor (e.g., because the Cb chroma block and Cr chroma block correspond to the same luma block or plurality of luma blocks). However, the block level quantization parameter offset for the Cb chroma block and the Cr chroma block may be different. It is possible for the quantization parameter predictor for the Cb chroma block and the Cr chroma block to be different.

[0098] Although the block level quantization parameter offset for the Cb chroma block and Cr chroma block may be different, video encoder 200 and video decoder 300 may utilize the same index into respective lists of quantization parameter offsets. For example, video encoder 200 may signal and video decoder 300 may receive an index into the first list of quantization parameter offsets for the first chroma block of the first color component (e.g., Cb chroma block) to determine the quantization parameter for the first chroma block of the first color component. Video encoder 200 and video decoder 300 may utilize the same index into the second list of quantization parameter offsets for the second chroma block of the second color component (e.g., Cr chroma block) to determine the quantization parameter for the second chroma block of the second color component. Accordingly, there may be two lists of quantization parameter offsets, but only one index may be needed.

[0099] FIGS. 2A and 2B are conceptual diagram illustrating an example quadtree binary tree (QTBT) structure 130, and a corresponding coding tree unit (CTU) 132. The solid lines represent quadtree splitting, and the 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, since 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 dotted 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.

[0100] 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).

[0101] 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 dotted 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.”

[0102] In one example of the QTBT partitioning structure, the CTU size is set as 128x128 (luma samples and two corresponding 64x64 chroma samples), the MinQTSIZE is set as 16x16, the MaxBTSIZE is set as 64x64, the MinBTSIZE (for both

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

[0103] The following describes some video coding standards. Video coding standards include ITU-T H.261, ISO/IEC MPEG-1 Visual, ITU-T H.262 or ISO/IEC MPEG-2 Visual, ITU-T H.263, ISO/IEC MPEG-4 Visual and ITU-T H.264 (also known as ISO/IEC MPEG-4 AVC), including its Scalable Video Coding (SVC) and Multi-view Video Coding (MVC) extensions.

[0104] In addition, High Efficiency Video Coding (HEVC) or ITU-T H.265, including its range extension, multiview extension (MV-HEVC) and scalable extension (SHVC), has been developed by the Joint Collaboration Team on Video Coding (JCT-VC) as well as Joint Collaboration Team on 3D Video Coding Extension Development (JCT-3V) of ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Motion Picture Experts Group (MPEG). The latest HEVC draft specification, and referred to as HEVC WD hereinafter, is available from http://phenix.int-evry.fr/jct/doc_end_user/documents/14_Vienna/wg11/JCTVC-N1003-v1.zip.

[0105] There may be certain issues in current techniques for determining chroma quantization parameter (QP). As described above, the QP indicates the amount by which to quantize or inverse-quantize coefficients in a block.

[0106] In VVC Draft 5, a chroma QP is derived as below:

$$\begin{aligned} qP_{Cb} &= \text{Clip3}(-QpBdOffsetc, 69, Qp_Y + pps_cb_qp_offset + slice_cb_qp_offset) \\ qP_{Cr} &= \text{Clip3}(-QpBdOffsetc, 69, Qp_Y + pps_cr_qp_offset + slice_cr_qp_offset) \end{aligned}$$

[0107] In the above equation, qP_{Cb} is the quantization parameter for the Cb chroma block, and qP_{Cr} is the quantization parameter for the Cr chroma block. Q_{PY} is a quantization parameter predictor, and in some examples is the quantization parameter for a corresponding luma block to the Cb chroma block and the Cr chroma block. Q_{PY} being the quantization parameter predictor is merely one example. The quantization parameter predictor may be based on Q_{PY} without necessarily being equal to Q_{PY} (e.g., the quantization parameter predictor may be determined as a function of Q_{PY}).

[0108] The syntax element $pps_cb_qp_offset$ is an offset value signaled in the picture parameter set for the Cb chroma block, and the syntax element $pps_cr_qp_offset$ is an offset value signaled in the picture parameter set for the Cr chroma block. The syntax element $slice_cb_qp_offset$ is an offset value signaled in the slice parameter set (e.g., such as slice header) for the Cb chroma block, and the syntax element $slice_cr_qp_offset$ is an offset value signaled in the slice parameter set for the Cr chroma block. The value of $Q_{pBdOffsetC}$ may be a preselected or determined value.

[0109] Accordingly, in VVC Draft 5, the value of qP_{Cb} is equal to $Q_{PY} + pps_cb_qp_offset + slice_cb_qp_offset$, but clipped to within $-Q_{pBdOffsetC}$ and 69 if $Q_{PY} + pps_cb_qp_offset + slice_cb_qp_offset$ is outside that range. Similarly, in VVC Draft 5, the value of qP_{Cr} is equal to $Q_{PY} + pps_cr_qp_offset + slice_cr_qp_offset$, but clipped to within $-Q_{pBdOffsetC}$ and 69 if $Q_{PY} + pps_cr_qp_offset + slice_cr_qp_offset$ is outside that range.

[0110] As described above and also described in more detail below, $pps_cb_qp_offset$ and $slice_cb_qp_offset$ apply to all Cb chroma blocks in a picture or slice. Therefore, there may not be block level quantization parameter offsets that can be used to determine the qP_{Cb} value for a Cb chroma block. Similarly, $pps_cr_qp_offset$ and $slice_cr_qp_offset$ apply to all Cr chroma blocks in a picture or slice. Therefore, there may not be block level quantization parameter offsets that can be used to determine the qP_{Cr} value for a Cr chroma block. This disclosure describes example techniques for utilizing block level quantization parameter offset for determining the quantization parameter for the Cb chroma block (e.g., qP_{Cb}) and the quantization parameter for the Cr chroma block (e.g., qP_{Cr}).

[0111] For instance, when the tree type is a separate tree type (e.g., chroma and luma blocks are partitioned independently), the variable Q_{PY} is set equal to the luma QP of the luma CB (coding block) that covers the luma position $(xCb + cbWidth / 2, yCb + cbHeight / 2)$. As shown in FIG. 5, one chroma coding

block (CB) 402 may cover more than one luma CBs. For example, chroma CB 402 covers luma CBs 400A–400D. These luma CBs 400A–400D can be from different quantization groups (QGs) with different QPs. The predicted QP derived from the center position may be not an accurate prediction for the chroma CB, and picture parameter set (PPS)/slice level QP offset adjustment is limited (e.g., pps_cr_qp_offset and slice_cr_qp_offset and pps_cb_qp_offset and slice_cb_qp_offset may be limited). There may be no scheme of flexible adjusting chroma QP in the current VVC.

[0112] One example way in which to address some deficiencies with flexible adjustment of chroma QP is to fix the luma delta QP signaling in VVC. For instance, in VVC Draft 5, the condition of signaling luma delta QP may not be correct. Partial syntax is as below.

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)
}	
}	

[0113] According to the syntax, when treeType equals to DUAL_TREE_LUMA, and tu_cbf_luma equals to 0, and tu_cbf_cb or tu_cbf_cr equals to 1, luma delta QP still can be signaled. Signaling of luma delta QP in this case may be erroneous (e.g., unnecessary or cause video decoder 300 to perform in an unexpected manner). When the tree type is dual tree luma, the luma delta QP signaling may not (e.g., should not) depend on the CBF flag (coded bit flag indicating whether any bit in the block is significant) of chroma (tu_cbf_cb, tu_cbf_cr). For example, CBF flags of tu_cbf_luma[x0][y0], tu_cbf_cb[x0][y0], tu_cbf_cr[x0][y0] equals to 1 means there is coding bin in the coding block. Otherwise, CBF equals to 0 means there is no coding bin in the coding block. In one example, the disclosure describes modifications to the condition of luma delta QP signaling. When the tree type is dual tree luma, the luma delta QP signaling may

depend on luma CBF flag, and may not (e.g., does not) depend on chroma CBF flag. The partial syntax is as below: where new syntax is shown with text between <ADD> and </ADD> and removed syntax is shown with text between <DELETE> and </DELETE>

<ADD> If(treeType != DUAL_TREE_CHROMA) { </ADD>	
if(((tu_cbf_luma[x0][y0] tu_cbf_cb[x0][y0] tu_cbf_cr[x0][y0]) <DELETE> && treeType != DUAL_TREE_CHROMA </DELETE> <ADD> && treeType != DUAL_TREE_LUMA) (tu_cbf_luma[x0][y0] && treeType == DUAL_TREE_LUMA </ADD>)) {	
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)
}	
}	
<ADD> } </ADD>	

[0114] In one example, video encoder 200 and video decoder 300 may be configured to apply the CU level chroma QP offset method of HEVC range extension directly to VVC (e.g., extended CU chroma QP offset from Cb, Cr components to joint_cbr (joint coding of chrominance residuals, JCCR)).

[0115] The partial syntax is as below. Underling shows addition.

<ADD> If((tu_cbf_cb[x0][y0] tu_cbf_cr[x0][y0]) && cu_chroma_qp_offset_enabled_flag && !IsCuChromaQpOffsetCoded) {	
cu_chroma_qp_offset_flag	ae(v)
if(cu_chroma_qp_offset_flag && chroma_qp_offset_list_len_minus1 > 0)	
cu_chroma_qp_offset_idx	ae(v)
} </ADD>	

[0116] In the above syntax elements, the syntax elements tu_cbf_cb and tu_cbf_cr indicate whether there are any residual values for a Cb chroma block or the Cr chroma block. If there are no residual values, then no quantization or inverse quantization is needed. However, if there are residual values for the Cb chroma block or the Cr chroma

block, then video encoder 200 may determine quantization parameters for quantizing and video decoder 300 may determine quantization parameters for inverse quantizing.

[0117] As described above and in more detail below, this disclosure describes examples of block level quantization parameter offsets for determining the quantization parameter for a chroma block. Because the example quantization parameter offsets are at the block level, the block level quantization parameter offsets may be considered as being at the CU level.

[0118] In the above syntax elements, `cu_chroma_qp_offset_enabled_flag` indicates whether block level quantization parameter offsets are enabled or not (e.g., for blocks in a slice or a picture). The `cu_chroma_qp_offset_flag` indicates whether block level quantization parameter offsets are enabled or not for a particular chroma block. For example, if `cu_chroma_qp_offset_enabled_flag` is true, it means that it is possible for the quantization parameter of a chroma block to be determined based on a block level quantization parameter offset. However, it does not necessarily mean that the quantization parameter of a particular chroma block is to be determined based on a block level quantization parameter offset. The `cu_chroma_qp_offset_flag` may indicate whether the quantization parameter of a particular chroma block is to be determined based on a block level quantization parameter offset.

[0119] If the quantization parameter of a particular chroma block is to be determined based on a block level quantization parameter offset (e.g., `cu_chroma_qp_offset_flag` is true), then video encoder 200 may signal and video decoder 300 may receive the `cu_chroma_qp_offset_idx`, which is an index into a list of quantization parameter offsets, referred to as `cb_qp_offset_list` for the Cb chroma block and `cr_qp_offset_list` for the Cr chroma block. The `cu_chroma_qp_offset_idx` may identify a block level quantization parameter offset, and video encoder 200 and video decoder 300 may utilize the block level quantization parameter offset to determine the quantization parameter for the chroma block. In one or more examples, the same `cu_chroma_qp_offset_idx` may be used to identify the block level quantization parameter offset for the Cb chroma block from `cb_qp_offset_list` and to identify the block level quantization parameter offset for the Cr chroma block from `cr_qp_offset_list`.

[0120] In other words, a chroma QP offset index (e.g., `cu_chroma_qp_offset_idx`) is signaled at CU level (e.g., block level). Video decoder 300 may use this index to determine the block level quantization parameter offset. The Cb block level quantization parameter offset for a Cb chroma block may be referred to as

CuQpOffsetCb. The Cr block level quantization parameter offset for a Cr chroma block may be referred to as CuQpOffsetCr. The JointCbCr block level quantization parameter offset for a JointCbCr chroma block may be referred to as CuQpOffsetJointCbCr.

CuQpOffsetCb, CuQpOffsetCr, and CuQpOffsetJointCbCr are identified from the list of quantization parameter offsets, which may be a lookup table.

[0121] In some examples, even if block level quantization parameter offset is to be used for a particular block, video encoder 200 may not signal and video decoder 300 may not receive an index into the list of quantization parameter offsets. For example, if the size of list of quantization parameter offsets is only one, then video encoder 200 and video decoder 300 may implicitly determine that the index into the list of quantization parameter offsets is the first entry in the list of quantization parameter offsets without needing to signal any index.

[0122] In one example, the chroma QP offset lookup table (e.g., the list of quantization parameter offsets) can be predefined in both video encoder 200 and video decoder 300 or signaled from video encoder 200 to video decoder 300 at sequence level, picture level, or slice level. For example, this value can be signaled in a Sequence Parameter Set (SPS), a Picture Parameter Set (PPS), and/or a Slice header (SH). That is, the values that make the list of quantization parameter offset (e.g., cb_qp_offset_list, cr_qp_offset_list, or joint_cbc_r_qp_offset_list) may be pre-stored in video encoder 200 and video decoder 300 or video encoder 200 may signal and video decoder 300 may receive the values in an SPS, PPS, and/or SH.

[0123] The following describes example ways in which the block level quantization parameter offset is utilized to determine the quantization parameter for a chroma block. The block level quantization parameter offset may be CuQpOffsetCb, CuQpOffsetCr, and CuQpOffsetJointCbCr that are identified from respective lists of quantization parameter offsets (e.g., cb_qp_offset_list, cr_qp_offset_list, or joint_cbc_r_qp_offset_list) based on a signaled index (e.g., cu_chroma_qp_offset_idx).

[0124] In one example, the variables qP_{Cb} , qP_{Cr} and qP_{CbCr} can be derived as follows:

$$\begin{aligned}
 qP_{Cb} &= \text{Clip3}(-QpBdOffset_c, 69, Qp_Y + pps_cb_qp_offset + slice_cb_qp_offset + \dots \\
 &\quad + CuQpOffsetCb) \\
 qP_{Cr} &= \text{Clip3}(-QpBdOffset_c, 69, Qp_Y + pps_cr_qp_offset + slice_cr_qp_offset + \dots \\
 &\quad + CuQpOffsetCr)
 \end{aligned}$$

$$qP_{iCbCr} = \text{Clip3}(-QpBdOffset_c, 69, Qp_Y + pps_joint_cbcr_qp_offset + slice_joint_c \\ bcr_qp_offset + \dots \\ + CuQpOffsetJointCbCr)$$

[0125] In the above, $CuQpOffsetCb = cb_qp_offset_list[cu_chroma_qp_offset_idx]$ is the CU level QP offset for Cb and $CuQpOffsetCr = cr_qp_offset_list[cu_chroma_qp_offset_idx]$ is the CU level QP offset for Cr, and $CuQpOffsetJointCbCr = joint_cbcr_qp_offset_list[cu_chroma_qp_offset_idx]$ is the CU level QP offset for joint CbCr. For example, to determine $CuQpOffsetCb$ (e.g., block level quantization parameter offset for Cb chroma block), video encoder 200 and video decoder 300 may construct a list of quantization parameter offsets (e.g., $cb_qp_offset_list$). Video encoder 200 may signal an index (e.g., $cu_chroma_qp_offset_idx$) into the list of quantization parameter offsets. Video decoder 300 may determine that the value stored in $cb_qp_offset_list[cu_chroma_qp_offset_idx]$ is equal to $CuQpOffsetCb$. To determine $CuQpOffsetCr$ (e.g., block level quantization parameter offset for Cr chroma block), video encoder 200 and video decoder 300 may construct a list of quantization parameter offsets (e.g., $cr_qp_offset_list$). Video encoder 200 may signal an index (e.g., $cu_chroma_qp_offset_idx$) into the list of quantization parameter offsets. Video decoder 300 may determine that the value stored in $cr_qp_offset_list[cu_chroma_qp_offset_idx]$ is equal to $CuQpOffsetCr$. In this example, the same index (e.g., $cu_chroma_qp_offset_idx$) is used to identify the block level quantization parameter offset for the Cb chroma block and the Cr chroma block. To determine $CuQpOffsetJointCbCr$ (e.g., block level quantization parameter offset for JointCbCr chroma block), video encoder 200 and video decoder 300 may construct a list of quantization parameter offsets (e.g., $jointcbcr_qp_offset_list$). Video encoder 200 may signal an index (e.g., $cu_chroma_qp_offset_idx$) into the list of quantization parameter offsets. Video decoder 300 may determine that the value stored in $jointcbcr_qp_offset_list[cu_chroma_qp_offset_idx]$ is equal to $CuQpOffsetJointCbCr$.

[0126] As described above, the equations to determine the quantization parameter for the Cb chroma block, Cr chroma block, and JointCbCr chroma block may be as follows:

$$qP_{iCb} = \text{Clip3}(-QpBdOffset_c, 69, Qp_Y + pps_cb_qp_offset + slice_cb_qp_offset + \dots \\ + CuQpOffsetCb)$$

$$qP_{iCr} = \text{Clip3}(-QpBdOffset_c, 69, Qp_Y + pps_cr_qp_offset + slice_cr_qp_offset + \dots \\ + CuQpOffsetCr)$$

$$qP_{CbCr} = \text{Clip3}(-QpBdOffset_c, 69, Qp_Y + pps_joint_cbr_qp_offset + slice_joint_cbr_qp_offset + \dots + CuQpOffsetJointCbCr)$$

[0127] Accordingly, in accordance with one or more examples described in this disclosure, video decoder 300 may be configured to determine a quantization parameter predictor for a chroma block of video data based on a quantization parameter for a corresponding luma block. As one example, the quantization parameter predictor is Qp_Y , and may be the quantization parameter for a corresponding luma block or average of plurality of corresponding luma blocks. In some examples, the quantization parameter predictor may be determined based on Qp_Y without necessarily being equal to Qp_Y . For example, the Qp_Y may form as a value into a look up table from which the quantization parameter predictor is determined. The quantization parameter predictor may be different for the different chroma components or may be the same for the different chroma components.

[0128] Video decoder 300 may determine a block level quantization parameter offset for the chroma block. Examples of the block level quantization parameter offset include $CuQpOffsetCb$ for the Cb chroma block, $CuQpOffsetCr$ for the Cr chroma block, and $CuQpOffsetJointCbCr$ for the JointCbCr chroma block.

[0129] In one or more examples, the block level quantization parameter offset is different for at least one other chroma block in same slice or picture as the chroma block and is determined on a chroma block-by-chroma block basis. For instance, a first chroma block in a slice or picture may have a different block level quantization parameter offset as a second chroma block in the same slice or picture, and the block level quantization parameter offset for the first block and the second block may be determined on a chroma block-by-chroma block basis.

[0130] As one example, the first chroma block may be of a first chroma component (e.g., Cb chroma block or Cr chroma block) and the second chroma block may be of a second chroma component (e.g., the other of Cb chroma block or Cr chroma block). In this example, video decoder 300 may determine a first block level quantization parameter offset for the first chroma block of the first chroma component and determine a second block level quantization parameter offset for the second chroma block of the second chroma component. The first block level quantization parameter offset and the second block level quantization parameter offset may be different.

[0131] In the above example, the first chroma block and the second chroma block are of the same CU. However, in some examples, the first chroma block and the second

chroma block may be of different CUs, and the block level quantization parameter offset for the first chroma block and the second chroma block of different CUs may be different.

[0132] There may be various ways in which video decoder 300 may determine the block level quantization parameter offset. For example, video decoder 300 may construct a list of quantization parameter offsets (e.g., `cb_qp_offset_list`, `cr_qp_offset_list`, and/or `joint_cbr_qp_offset_list`), where the values for the list of quantization parameter offsets may be pre-stored or signaled. Video decoder 300 may receive an index into the list of quantization parameter offsets (e.g., `cu_chroma_qp_offset_idx`). Video decoder 300 may determine the block level quantization parameter offset based on the index into the list of quantization parameter offsets. For example, video decoder 300 may determine that `cb_qp_offset_list[cu_chroma_qp_offset_idx]` is the CU level QP offset for Cb and `CuQpOffsetCr = cr_qp_offset_list[cu_chroma_qp_offset_idx]` is the CU level QP offset for Cr, and `CuQpOffsetJointCbCr = joint_cbr_qp_offset_list[cu_chroma_qp_offset_idx]` is the CU level QP offset for JointCbCr.

[0133] In one or more examples, where a first chroma block and a second chroma block are of the same CU, the index into respective lists of quantization parameter offsets may be the same. For example, `cu_chroma_qp_offset_idx` is the index for both `cb_qp_offset_list` and `cr_qp_offset_list`. That is, video decoder 300 may be configured to determine a first block level quantization parameter offset from an index into a first list of quantization parameter offsets for the first chroma component (e.g., determine `CuQpOffsetCb` based on `cu_chroma_qp_offset_idx` into `cb_qp_offset_list`), and determine the second block level quantization parameter offset from the same index into a second list of quantization parameter offsets for the second chroma component (e.g., determine `CuQpOffsetCr` based on `cu_chroma_qp_offset_idx` into `cr_qp_offset_list`).

[0134] Video decoder 300 may determine a quantization parameter for the chroma block based on the block level quantization parameter offset and the quantization parameter predictor. For example, video decoder 300 may receive at least one of a first quantization parameter offset for chroma component in a picture parameter set (e.g., `pps_cb_qp_offset` for Cb chroma block or `pps_cr_qp_offset` for Cr chroma block) or a second quantization parameter offset for chroma component in a slice parameter set

(e.g., slice_cb_qp_offset for Cb chroma block or slice_cr_qp_offset for Cr chroma block signaled in slice header).

[0135] Video decoder 300 may determine the quantization parameter for the chroma block by adding the block level quantization parameter offset and one or both of the first quantization parameter offset and the second quantization parameter offset. For example, as described above:

$$qP_{Cb} = \text{Clip3}(-QpBdOffset_c, 69, Qp_Y + pps_cb_qp_offset + slice_cb_qp_offset + \dots + CuQpOffsetCb)$$

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

$$qP_{CbCr} = \text{Clip3}(-QpBdOffset_c, 69, Qp_Y + pps_joint_cbcr_qp_offset + slice_joint_cbcr_qp_offset + \dots + CuQpOffsetJointCbCr).$$

[0136] Video decoder 300 may determine a residual block based on the quantization parameter. For example, video decoder 300 may inverse-quantize a plurality of coefficient values based on the quantization parameter to generate inverse-quantized coefficient values and inverse-transform the inverse-quantized coefficient values to generate the residual block. In some examples, such as transform skip mode, inverse-transform may be skipped.

[0137] Video decoder 300 may reconstruct the chroma block based on the residual block. For example, video decoder 300 may determine a prediction block for the chroma block and add the residual block to the prediction block for reconstructing the chroma block.

[0138] In the above example, video decoder 300 is described as determining the block level quantization parameter offset for the chroma block. However, in some cases, it may be possible that determining block level quantization parameter offset for a particular chroma block is disabled. Accordingly, in one or more examples, video decoder 300 may receive a syntax element (e.g., cu_chroma_qp_offset_flag) indicating that the block level quantization parameter offset is to be used for determining the block level quantization parameter offset for the chroma block. In such examples, video decoder 300 may determine the block level quantization parameter offset based on the syntax element indicating that the block level quantization parameter offset is to be used (e.g., based on cu_chroma_qp_offset_flag being true).

[0139] In accordance with one or more examples described in this disclosure, video encoder 200 may be configured to determine a quantization parameter predictor for a chroma block of video data based on a quantization parameter for a corresponding luma block. As one example, the quantization parameter predictor is Q_{PY} , and may be the quantization parameter for a corresponding luma block or average of plurality of corresponding luma blocks. In some examples, the quantization parameter predictor may be determined based on Q_{PY} without necessarily being equal to Q_{PY} . For example, the Q_{PY} may form as a value into a look up table from which the quantization parameter predictor is determined. The quantization parameter predictor may be different for the different chroma components or may be the same for the different chroma components.

[0140] Video encoder 200 may determine a block level quantization parameter offset for the chroma block. Examples of the block level quantization parameter offset include $CuQpOffsetCb$ for the Cb chroma block, $CuQpOffsetCr$ for the Cr chroma block, and $CuQpOffsetJointCbCr$ for the JointCbCr chroma block.

[0141] Video encoder 200 may construct a list of quantization parameter offsets (e.g., $cb_qp_offset_list$, $cr_qp_offset_list$, and/or $joint_cbcr_qp_offset_list$), where the values for the list of quantization parameter offsets may be pre-stored or signaled. Video encoder 200 may signal an index into the list of quantization parameter offsets (e.g., $cu_chroma_qp_offset_idx$).

[0142] In one or more examples, where a first chroma block and a second chroma block are of the same CU, the index into respective lists of quantization parameter offsets may be the same. For example, $cu_chroma_qp_offset_idx$ is the index for both $cb_qp_offset_list$ and $cr_qp_offset_list$.

[0143] Video encoder 200 may determine a quantization parameter for the chroma block based on the block level quantization parameter offset and the quantization parameter predictor. For example, video encoder 200 may determine at least one of a first quantization parameter offset for chroma component in a picture parameter set (e.g., $pps_cb_qp_offset$ for Cb chroma block or $pps_cr_qp_offset$ for Cr chroma block) or a second quantization parameter offset for chroma component in a slice parameter set (e.g., $slice_cb_qp_offset$ for Cb chroma block or $slice_cr_qp_offset$ for Cr chroma block signaled in slice header).

[0144] Video encoder 200 may determine the quantization parameter for the chroma block by adding the block level quantization parameter offset and one or both of the first

quantization parameter offset and the second quantization parameter offset. For example, as described above:

$$qP_{iCb} = \text{Clip3}(-QpBdOffset_c, 69, Qp_Y + pps_cb_qp_offset + slice_cb_qp_offset + \dots + CuQpOffsetCb)$$

$$qP_{iCr} = \text{Clip3}(-QpBdOffset_c, 69, Qp_Y + pps_cr_qp_offset + slice_cr_qp_offset + \dots + CuQpOffsetCr)$$

$$qP_{iCbCr} = \text{Clip3}(-QpBdOffset_c, 69, Qp_Y + pps_joint_cbcr_qp_offset + slice_joint_cbcr_qp_offset + \dots + CuQpOffsetJointCbCr).$$

[0145] Video encoder 200 may quantize coefficient values for a residual block based on the determined quantization parameter for the chroma block. For example, video encoder 200 may generate the coefficient values for the residual block using a transform (although transform skip is possible) and quantize coefficient values based on the quantization parameter to generate quantized coefficient values. Video encoder 200 may signal information indicative of the quantized coefficient values.

[0146] In the above example, video encoder 200 is described as determining the block level quantization parameter offset for the chroma block. However, in some cases, it may be possible that determining block level quantization parameter offset for a particular chroma block is disabled. Accordingly, in one or more examples, video encoder 200 may signal a syntax element (e.g., `cu_chroma_qp_offset_flag`) indicating that the block level quantization parameter offset is to be used for determining the block level quantization parameter offset for the chroma block.

[0147] In some examples, video encoder 200 may signal and video decoder 300 may receive CU level chroma delta QP for chroma components. As one example, video encoder 200 may signal CU level chroma delta QP for chroma components when dual tree is enabled. In this example, a chroma delta QP value may be signaled using both absolute value for chroma delta QP and sign values for chroma delta QP for both Cb and Cr components. The variables qP_{iCb} , qP_{iCr} and qP_{iCbCr} can be derived as follows:

$$qP_{iCb} = \text{Clip3}(-QpBdOffset_c, 69, Qp_Y + pps_cb_qp_offset + slice_cb_qp_offset + \dots + ChromaCuQpDeltaVal + CuQpOffsetCb)$$

$$qP_{iCr} = \text{Clip3}(-QpBdOffset_c, 69, Qp_Y + pps_cr_qp_offset + slice_cr_qp_offset + \dots + ChromaCuQpDeltaVal + CuQpOffsetCr)$$

$$qP_{iCbCr} = \text{Clip3}(-QpBdOffset_c, 69, Qp_Y + pps_joint_cbr_qp_offset + slice_joint_cbr_qp_offset + \dots ChromaCuQpDeltaVal + CuQpOffsetJointCbCr)$$

[0148] In the above, $ChromaCuQpDeltaVal = chroma_cu_qp_delta_abs * (1 - 2 * chroma_cu_qp_delta_sign_flag)$. $CuQpOffsetCb = cb_qp_offset_list[cu_chroma_qp_offset_idx]$ is the CU level QP offset for Cb and $CuQpOffsetCr = cr_qp_offset_list[cu_chroma_qp_offset_idx]$ is the CU level QP offset for Cr, and $CuQpOffsetJointCbCr = joint_cbr_qp_offset_list[cu_chroma_qp_offset_idx]$ is the CU level QP offset for joint CbCr. In this example, video encoder 200 and video decoder 300 may use both PPS, slice and CU level QP offsets for chroma when dual tree is enabled.

[0149] Partial syntax is as below. <ADD>...</ADD> shows addition and <DELETE>...</DELETE> shows deletion.

transform_unit(x0, y0, tbWidth, tbHeight, treeType, subTuIndex) {	Descriptor
...	
<ADD> If(treeType != DUAL_TREE_CHROMA) {</ADD>	
if(((tu_cbf_luma[x0][y0] tu_cbf_cb[x0][y0] tu_cbf_cr[x0][y0]) <DELETE>&& treeType != DUAL_TREE_CHROMA </DELETE> && <ADD> treeType != DUAL_TREE_LUMA) (tu_cbf_luma[x0][y0] && treeType == DUAL_TREE_LUMA</ADD>)) {	
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)
}	
}	
<ADD>}	
If((tu_cbf_cb[x0][y0] tu_cbf_cr[x0][y0]) && cu_chroma_qp_offset_enabled_flag && !IsCuChromaQpOffsetCoded) {	
cu_chroma_qp_offset_flag	ae(v)
if(cu_chroma_qp_offset_flag && chroma_qp_offset_list_len_minus1 > 0)	
cu_chroma_qp_offset_idx	ae(v)
if(treeType == DUAL_TREE_CHROMA && cu_chroma_qp_offset_flag) {	
chroma_cu_qp_delta_abs	ae(v)
if(chroma_cu_qp_delta_abs)	
chroma_cu_qp_delta_sign_flag	ae(v)
}	
} </ADD>	
...	
}	

[0150] In some examples, video encoder 200 may signal and video decoder 300 may receive chroma delta QPs for chroma components Cb and Cr and joint_CbCr separately. In one example, in order to provide more flexible, in the example, the delta QPs for chroma components Cb and Cr and joint_CbCr are signaled separately. In the example,

a chroma delta QP value is signaled using both absolute value for chroma delta QP and sign values for chroma delta QP for both Cb and Cr components. Variables qPi_{Cb} and qPi_{Cr} and qPi_{CbCr} can be derived as follows:

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

...

$$\text{ChromaCuQpDeltaVal})$$

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

...

$$\text{ChromaCuQpDeltaVal})$$

$$qPi_{CbCr} = \text{Clip3}(-QpBdOffset_c, 69, Qp_Y + pps_joint_cbcr_qp_offset + slice_joint_cbcr_qp_offset + \dots \text{ChromaCuQpDeltaVal})$$

[0151] Partial syntax is as below. <ADD>...</ADD> shows addition and <DELETE>...</DELETE> shows deletion.

transform_unit(x0, y0, tbWidth, tbHeight, treeType, subTuIndex) {	Descriptor
...	
<ADD> If(treeType != DUAL_TREE_CHROMA) { </ADD>	
if((tu_cbf_luma[x0][y0] tu_cbf_cb[x0][y0] tu_cbf_cr[x0][y0]) <DELETE>&& treeType != DUAL_TREE_CHROMA </DELETE> && <ADD> treeType != DUAL_TREE_LUMA) (tu_cbf_luma[x0][y0] && treeType == DUAL_TREE_LUMA </ADD>)) {	
if(cu_qp_delta_enabled_flag && !IsCuQpDeltaCoded) {	
cu_qp_delta_abs	<u>ac(v)</u>
if(cu_qp_delta_abs)	
cu_qp_delta_sign_flag	<u>ac(v)</u>
}	
}	
}	
<ADD>...	
if(tu_cbf_cr[x0][y0]) {	
if(tu_cbf_cb[x0][y0])	
tu_joint_cbr_residual[x0][y0]	<u>ac(v)</u>
...	
If (treeType != DUAL_TREE_LUMA) {	
if(chroma_cu_qp_delta_enabled_flag && !IsChromaCuQpDeltaCoded)	
{	
If (tu_joint_cbr_residual[x0][y0]) {	
cbr_cu_qp_delta_abs	<u>ac(v)</u>
if(cbr_cu_qp_delta_abs)	
cbr_cu_qp_delta_sign_flag	<u>ac(v)</u>
} else {	
if(tu_cbf_cb[x0][y0]) {	
cb_cu_qp_delta_abs	<u>ac(v)</u>
if(cb_cu_qp_delta_abs)	
cb_cu_qp_delta_sign_flag	<u>ac(v)</u>
}	
if(tu_cbf_cr[x0][y0]) {	
cr_cu_qp_delta_abs	<u>ac(v)</u>

if(cr_cu_qp_delta_abs)	
cr_cu_qp_delta_sign_flag	<u>ac(v)</u>
}	
}	
}	
}	
} </ADD>	
...	
}	

[0152] In one example, the condition of signaling the cu level chroma delta QP and cu level chroma QP offset depends on the CBF flag of the components. For example, video encoder 200 may only signal delta QP and/or QP offset when the CBF of corresponding component is true (e.g. equal to 1).

[0153] In one example, if the size of chroma coding block equals to or is bigger than a predefined size (for example, virtual pipeline data unit (VPDU) size, or 32x32), video encoder 200 may signal the chroma delta QP and/or chroma QP offset at the first coding block, no matter the CBF of the first CB is 0 or 1. The specified size can be predefined in both encoder side and decoder side, or set as a value signaled from video encoder 200 to video decoder 300 at sequence level, picture level, slice level. For example, this value can be signaled in Sequence Parameter Set (SPS), Picture Parameter Set (PPS), Slice header (SH). In one example, when dual tree is enabled, video encoder 200 may signal the cu level chroma delta QP and/or cu level chroma QP offset no matter the cbf of chroma component is 0 or 1.

[0154] FIG. 3 is a block diagram illustrating an example video encoder 200 that may perform the techniques of this disclosure. FIG. 3 is provided for purposes of explanation and should not be considered limiting of the techniques as broadly exemplified and described in this disclosure. For purposes of explanation, this disclosure describes video encoder 200 in the context of video coding standards such as the HEVC video coding standard and the H.266 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.

[0155] In the example of FIG. 3, video encoder 200 includes video data memory 230, mode selection unit 202, residual generation unit 204, transform processing unit 206,

quantization unit 208, inverse quantization unit 210, inverse transform processing unit 212, reconstruction unit 214, filter unit 216, decoded picture buffer (DPB) 218, and entropy encoding unit 220. 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. In other examples, video encoder 200 may include more, fewer, or different units.

[0156] 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.

[0157] 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.

[0158] The various units of FIG. 3 are illustrated to assist with understanding the operations performed by video encoder 200. The units may be implemented as fixed-function circuits, programmable circuits, or a combination thereof. Fixed-function circuits refer to circuits that provide particular functionality, and are preset on the operations that can be performed. Programmable circuits refer to circuits that can be

programmed to perform various tasks, and provide flexible functionality in the operations that can be performed. For instance, programmable circuits may execute software or firmware that 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.

[0159] 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.

[0160] 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.

[0161] 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.

[0162] 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.

[0163] 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.”

[0164] 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.

[0165] 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.

[0166] 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.

[0167] 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.

[0168] 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.

[0169] 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 $N \times 2N$.

[0170] 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.

[0171] 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.

[0172] 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.

[0173] 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.

[0174] In accordance with one or more examples described in this disclosure, mode selection unit 202 may be configured to determine a quantization parameter predictor for a chroma block of the video data based on a quantization parameter for a corresponding luma block. For example, mode selection unit 202 may have determined a quantization parameter for the corresponding luma block or plurality of corresponding luma blocks, and the quantization parameter for the corresponding luma block or some average of the quantization parameters for the plurality of corresponding luma blocks may form the quantization parameter predictor. The quantization parameter predictor may be referred to as Q_{py} . However, in some examples, the quantization parameter predictor may be determined from the quantization parameter for the corresponding luma block (e.g., the quantization parameter predictor is determined from Q_{py}).

[0175] Mode selection unit 202 may determine a block level quantization parameter offset for the chroma block. The chroma block may be Cb chroma block, Cr chroma block, or a JointCbCr chroma block. Examples of the block level quantization parameter offsets include $CuOffsetCb$ for the Cb chroma block, $CuOffsetCr$ for the Cr chroma block, and $CuQpOffsetJointCbCr$ for the JointCbCr chroma block.

[0176] In some examples, mode selection unit 202 may first determine if block level quantization parameter offset for the particular chroma block should be enabled. If mode selection unit 202 determines that block level quantization parameter offset is enabled for the chroma block, mode selection unit 202 may determine that the $cu_chroma_qp_offset_flag$ is true, and cause entropy encoding unit 220 to signal information indicating that $cu_chroma_qp_offset_flag$ is true.

[0177] Mode selection unit 202 may construct a list of quantization parameter offsets. Mode selection unit 202 may determine an index into the list of quantization parameter offsets for the determined block level quantization parameter offset. Entropy encoding unit 220 may signal information indicative of the index. Example of the index includes $cu_chroma_qp_off_idx$.

[0178] Mode selection unit 202 may determine a quantization parameter for the chroma block based on the block level quantization parameter offset and the quantization parameter predictor. As one example, mode selection unit 202 may determine a first quantization parameter offset (e.g., $pps_cb_qp_offset$, $pps_cr_qp_offset$, or $pps_joint_cbcr_qp_offset$) in a picture parameter set and a second quantization

parameter offset (e.g., slice_cb_qp_offset, slice_cr_qp_offset, or slice_joint_cbr_qp_offset). Mode selection unit 202 may add the first quantization parameter, second quantization parameter, and the block level quantization parameter offset, and clip the result to within a certain range to determine the quantization parameter for the chroma block.

[0179] Quantization unit 208 may quantize coefficient values for a residual block based on the determined quantization parameter for the chroma block. Entropy encoding unit 220 may signal information indicative of the quantized coefficient values.

[0180] In the above examples, mode selection unit 202, quantization unit 208, and entropy encoding unit 220 are described as performing the example techniques. However, any one or combination of components of video encoder 200 may be configured to perform the example techniques.

[0181] 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.

[0182] 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.

[0183] Video encoder 200 stores reconstructed blocks in DPB 218. For instance, in examples where operations of filter unit 216 are not needed, reconstruction unit 214 may store reconstructed blocks to DPB 218. In examples where operations of filter unit 216 are needed, filter unit 216 may store the filtered reconstructed blocks to DPB 218. Motion estimation unit 222 and motion compensation unit 224 may retrieve a reference picture from DPB 218, formed from the reconstructed (and potentially filtered) blocks, to inter-predict blocks of subsequently encoded pictures. In addition, intra-prediction unit 226 may use reconstructed blocks in DPB 218 of a current picture to intra-predict other blocks in the current picture.

[0184] 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.

[0185] 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.

[0186] 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.

[0187] 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.

[0188] FIG. 4 is a block diagram illustrating an example video decoder 300 that may perform the techniques of this disclosure. FIG. 4 is provided for purposes of

explanation and is not limiting on the techniques as broadly exemplified and described in this disclosure. For purposes of explanation, this disclosure describes video decoder 300 according to the techniques of VVC and HEVC. However, the techniques of this disclosure may be performed by video coding devices that are configured to other video coding standards.

[0189] In the example of FIG. 4, video decoder 300 includes coded picture buffer (CPB) memory 320, entropy decoding unit 302, prediction processing unit 304, inverse quantization unit 306, inverse transform processing unit 308, reconstruction unit 310, filter unit 312, and decoded picture buffer (DPB) 314. 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. In other examples, video decoder 300 may include more, fewer, or different units.

[0190] 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.

[0191] 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.

[0192] 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.

[0193] The various units shown in FIG. 4 are illustrated to assist with understanding the operations performed by video decoder 300. The units may be implemented as fixed-function circuits, programmable circuits, or a combination thereof. Similar to FIG. 3, fixed-function circuits refer to circuits that provide particular functionality, and are preset on the operations that can be performed. Programmable circuits refer to circuits that can be programmed to perform various tasks, and provide flexible functionality in the operations that can be performed. For instance, programmable circuits may execute software or firmware that 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.

[0194] 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.

[0195] 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.

[0196] 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”).

[0197] 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.

[0198] 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.

[0199] Furthermore, prediction processing unit 304 generates a prediction block according to prediction information syntax elements that were entropy decoded by entropy decoding unit 302. For example, if the prediction information syntax elements indicate that the current block is inter-predicted, motion compensation unit 316 may generate the prediction block. In this case, the prediction information syntax elements may indicate a reference picture in DPB 314 from which to retrieve a reference block, as well as a motion vector identifying a location of the reference block in the reference picture relative to the location of the current block in the current picture. Motion compensation unit 316 may generally perform the inter-prediction process in a manner that is substantially similar to that described with respect to motion compensation unit 224 (FIG. 3).

[0200] As another example, if the prediction information syntax elements indicate that the current block is intra-predicted, intra-prediction unit 318 may generate the prediction block according to an intra-prediction mode indicated by the prediction information syntax elements. Again, intra-prediction unit 318 may generally perform the intra-prediction process in a manner that is substantially similar to that described

with respect to intra-prediction unit 226 (FIG. 3). Intra-prediction unit 318 may retrieve data of neighboring samples to the current block from DPB 314.

[0201] 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.

[0202] 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.

[0203] 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.

[0204] In one or more examples, prediction processing unit 304 and inverse quantization unit 306, along with reconstruction unit 310 may be configured to perform one or more example techniques described in this disclosure. For example, prediction processing unit 304 may determine a quantization parameter predictor for a chroma block of the video data based on a quantization parameter for a corresponding luma block. For example, prediction processing unit 304 may have determined a quantization parameter for the corresponding luma block or plurality of corresponding luma blocks, and determined the quantization parameter predictor for the chroma block based on the quantization parameter for the corresponding luma block or plurality of corresponding luma blocks. One example of the quantization parameter predictor is referred to as Qp_Y . However, in some examples, the quantization parameter predictor may be determined from the quantization parameter for the corresponding luma block (e.g., the quantization parameter predictor is determined from Qp_Y).

[0205] Prediction processing unit 304 may determine whether block level quantization parameter offset is enabled for the chroma block. For example, prediction processing unit 304 may receive a syntax element (e.g., `cu_chroma_qp_offset_flag`) indicating that the block level quantization parameter offset is to be used for determining the block level quantization parameter offset for the chroma block. Prediction processing unit 304

may determine a block level quantization parameter offset for the chroma block. For example, prediction processing unit 304 may determine the block level quantization parameter offset based on the syntax element indicating that the block level quantization parameter offset is to be used.

[0206] As one example, prediction processing unit 304 may construct a list of quantization parameter offsets (e.g., `cb_qp_offset_list`, `cr_qp_offset_list`, and/or `joint_cbc_r_qp_offset_list`) and receive an index (e.g., `cu_chroma_qp_offset_idx`) into the list of quantization parameter offsets. Prediction processing unit 304 may determine the block level quantization parameter offset based on the index into the list of quantization parameter offsets. Examples of the block level quantization parameter offset include `CuQpOffsetCb`, `CuQpOffsetCr`, and `CuQpOffsetJointCbCr`.

[0207] The chroma block may be a first chroma block of a first chroma component (e.g., Cb chroma block) and the list of quantization parameter offsets may be a first list of quantization parameter offsets (e.g., `cb_qp_offset_list`). Prediction processing unit 304 may also construct a list of quantization parameter offsets (e.g., `cr_qp_offset_list`) for a second chroma block of a second chroma component (e.g., Cr chroma block). In one or more examples, prediction processing unit 304 may utilize the same index (`cu_chroma_qp_offset_idx`) to determine the block level quantization parameter offset for the first chroma block and to determine the block level quantization parameter offset for the second chroma block (e.g., `cu_chroma_qp_offset_idx` is an index in `cb_qp_offset_list` and `cr_qp_offset_list`).

[0208] In one or more examples, the block level quantization parameter offset may be different for different chroma blocks in the same picture or slice. For example, the block level quantization parameter offset may be determined on a chroma block-by-chroma block basis. For instance, prediction processing unit 304 may determine the block level quantization parameter for a first chroma block in a slice or picture, and determine the block level quantization parameter for a second chroma block in the same slice or picture, where the two block level quantization parameters may be different.

[0209] Prediction processing unit 304 may determine a quantization parameter for the chroma block based on the block level quantization parameter offset and the quantization parameter predictor. For example, prediction processing unit 304 may receive at least one of a first quantization parameter offset for chroma component (e.g., `pps_cb_qp_offset`, `pps_cr_qp_offset`, or `pps_joint_cbc_r_qp_offset`) in a picture parameter set or a second quantization parameter offset for chroma component (e.g.,

slice_cb_qp_offset, slice_cr_qp_offset, or slice_joint_cbr_qp_offset) in a slice parameter set (e.g., slice header). Prediction processing unit 304 may determine the quantization parameter for the chroma block based on the block level quantization parameter offset, at least one of the first quantization parameter offset or the second quantization parameter offset, and the quantization parameter predictor.

[0210] For example, prediction processing unit 304 may perform the following operations to determine the block level quantization parameter offset, examples of which include qP_{Cb} , qP_{Cr} , and qP_{CbCr} .

$$qP_{Cb} = \text{Clip3}(-QpBdOffset_C, 69, Qp_Y + pps_cb_qp_offset + slice_cb_qp_offset + \dots + CuQpOffsetCb)$$

$$qP_{Cr} = \text{Clip3}(-QpBdOffset_C, 69, Qp_Y + pps_cr_qp_offset + slice_cr_qp_offset + \dots + CuQpOffsetCr)$$

$$qP_{CbCr} = \text{Clip3}(-QpBdOffset_C, 69, Qp_Y + pps_joint_cbr_qp_offset + slice_joint_cbr_qp_offset + \dots + CuQpOffsetJointCbCr).$$

[0211] Inverse quantization unit 306 and inverse transform processing unit 308 may determine a residual block based on the quantization parameter. For example, inverse quantization unit 306 may inverse-quantize (e.g., performing inverse-quantizing) a plurality of coefficient values based on the quantization parameter to generate inverse-quantized coefficient values. Inverse transform processing unit 308 may inverse-transform (e.g., performing inverse-transforming) the inverse-quantized coefficient values to generate the residual block.

[0212] Reconstruction unit 310 may reconstruct the chroma block based on the residual block. For example, prediction processing unit 304 may determine a prediction block for the chroma block. Reconstruction unit 310 may add the residual block to the prediction block to reconstruct the chroma block.

[0213] FIG. 6 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 3), it should be understood that other devices may be configured to perform a method similar to that of FIG. 6. The example techniques are described with respect to processing circuitry. Examples of the processing circuitry include the components of video encoder 200.

[0214] In one or more examples, memory (e.g., memory 106, video data memory 230, DPB 218, or some other memory) may be configured to store a quantization parameter for a corresponding luma block of a chroma block of the video data. For instance, the

processing circuitry of video encoder 200 may have already determined the quantization parameter for the corresponding luma block and stored it in memory. The memory may be coupled to the processing circuitry of video encoder 200.

[0215] The processing circuitry of video encoder 200 may be configured to determine a quantization parameter predictor for a chroma block of the video data based on a quantization parameter for a corresponding luma block (500). For example, the processing circuitry may utilize the quantization parameter for the corresponding luma block as an input into a process to determine the quantization parameter predictor. In some examples, the quantization parameter predictor may be the quantization parameter of the corresponding luma block.

[0216] The processing circuitry of video encoder 200 may determine a block level quantization parameter offset for the chroma block (502). In some examples, the processing circuitry may construct a list of quantization parameter offsets and determine an index into the list of quantization parameter offsets for the determined block level quantization parameter offset. The processing circuitry may signal information indicative of the index.

[0217] The processing circuitry of video encoder 200 may determine a quantization parameter for the chroma block based on the block level quantization parameter offset and the quantization parameter predictor (504). In some examples, the processing circuitry may add the quantization parameter predictor, the block level quantization parameter offset, a first quantization parameter offset signaled in the PPS, and a second quantization parameter offset signaled in the slice parameter set (e.g., slice header) to determine the quantization parameter for the chroma block.

[0218] The processing circuitry may quantize coefficient values for a residual block based on the determined quantization parameter for the chroma block (506). The processing circuitry may then signal information indicative of the quantized coefficient values (508).

[0219] FIG. 7 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 4), it should be understood that other devices may be configured to perform a method similar to that of FIG. 7. The example techniques are described with respect to processing circuitry. Examples of the processing circuitry include the components of video decoder 300.

[0220] In one or more examples, memory (e.g., memory 120, CPB memory 320, DPB 314, or some other memory) may be configured to store a quantization parameter for a corresponding luma block of a chroma block of the video data. For instance, the processing circuitry of video decoder 300 may have already determined the quantization parameter for the corresponding luma block and stored it in memory. The memory may be coupled to the processing circuitry of video decoder 300.

[0221] The processing circuitry of video decoder 300 may be configured to determine a quantization parameter predictor for a chroma block of the video data based on a quantization parameter for a corresponding luma block (600). For example, the processing circuitry may utilize the quantization parameter for the corresponding luma block as an input into a process to determine the quantization parameter predictor. In some examples, the quantization parameter predictor may be the quantization parameter of the corresponding luma block.

[0222] The processing circuitry of video decoder 300 may determine a block level quantization parameter offset for the chroma block (602). In some examples, the processing circuitry may construct a list of quantization parameter offsets. The processing circuitry may receive an index into the list of quantization parameter offsets, and determine the block level quantization parameter offset based on the received index.

[0223] The processing circuitry of video decoder 300 may determine a quantization parameter for the chroma block based on the block level quantization parameter offset and the quantization parameter predictor (604). In some examples, the processing circuitry may add the quantization parameter predictor, the block level quantization parameter offset, a first quantization parameter offset received in the PPS, and a second quantization parameter offset received in the slice parameter set (e.g., slice header) to determine the quantization parameter for the chroma block.

[0224] The processing circuitry may determine a residual block based on the quantization parameter (606). For example, the processing circuitry may inverse-quantize a plurality of coefficient values based on the quantization parameter to generate inverse-quantized coefficient values, and inverse-transform the inverse-quantized coefficient values to generate the residual block. In some examples, inverse-transform may be skipped, such as for transform skip mode.

[0225] The processing circuitry may reconstruct the chroma block based on the residual block (608). For example, the processing circuitry may determine a prediction block for

the chroma block and add the residual block to the prediction block to reconstruct the chroma block.

[0226] 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.

[0227] 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.

[0228] 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.

[0229] 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.

[0230] 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.

[0231] Various examples have been described. These and other examples are within the scope of the following claims.