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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
07 May 2020 (07.05.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/089824 A1

(51) International Patent Classification:
H04N 19/117 (2014.01) *H04N 19/82* (2014.01)
H04N 19/176 (2014.01)

(21) International Application Number:
PCT/IB2019/059342

(22) International Filing Date:
31 October 2019 (31.10.2019)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PCT/CN2018/112945
31 October 2018 (31.10.2018) CN

(71) Applicants: **BEIJING BYTEDANCE NETWORK TECHNOLOGY CO., LTD.** [CN/CN]; Room B-0035, 2/F, No.3 Building, No.30, Shixing Road, Shijingshan District, Beijing 100041 (CN). **BYTEDANCE INC.** [US/US]; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US).

(72) Inventors: **LIU, Hongbin**; Jinritoutiao Post Office, China Satellite Communications Tower, No.63, Zhichun Road, Haidian District, Beijing 100080 (CN). **ZHANG, Li**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US). **ZHANG, Kai**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US). **WANG, Yue**; Jinritoutiao Post Office, China Satellite Communications Tower, No.63, Zhichun Road, Haidian District, Beijing 100080 (CN).

(74) Agent: **LIU, SHEN & ASSOCIATES**; 10th Floor, Building 1, 10 Caihefang Road, Haidian District, Beijing 100080 (CN).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO,

(54) Title: DEBLOCKING FILTERING UNDER DEPENDENT QUANTIZATION

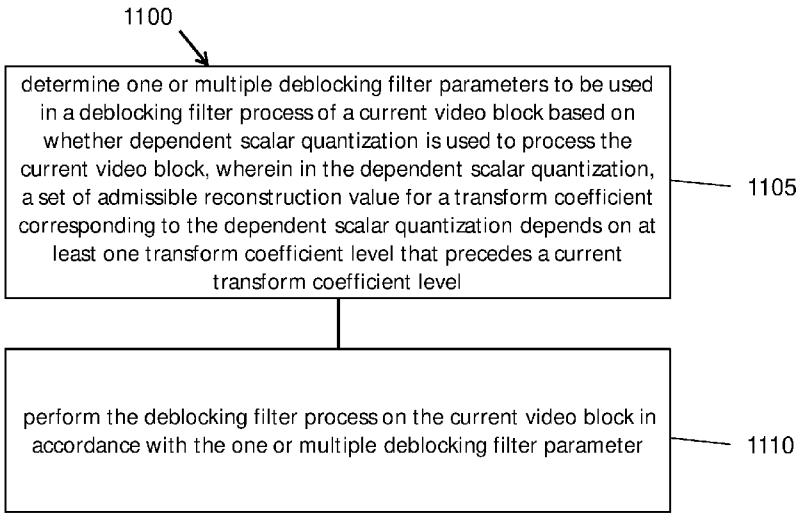

FIG. 11

(57) Abstract: Video processing method, apparatus and computer program product are disclosed. The video processing method comprises: determining one or multiple deblocking filter parameters to be used in a deblocking filter process of a current video block based on whether dependent scalar quantization is used to process the current video block, wherein a set of admissible reconstruction value for a transform coefficient corresponding to the dependent scalar quantization depends on at least one transform coefficient level that precedes a current transform coefficient level; and performing the deblocking filter process on the current video block in accordance with the one or multiple deblocking filter parameter.

WO 2020/089824 A1

[Continued on next page]

DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

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

Published:

— *with international search report (Art. 21(3))*

— *in black and white; the international application as filed contained color or greyscale and is available for download from PATENTSCOPE*

DEBLOCKING FILTERING UNDER DEPENDENT QUANTIZATION

CROSS-REFERENCE TO RELATED APPLICATION

[001] Under the applicable patent law and/or rules pursuant to the Paris Convention, this application is made to timely claim the priority to and benefits of International Patent Application No. PCT/CN2018/112945, filed on October 31, 2018. The entire disclosure of International Patent Application No. PCT/CN2018/112945 is incorporated by reference as part of the disclosure of this application.

TECHNICAL FIELD

[002] This document is related to video and image coding technologies.

BACKGROUND

[003] Digital video accounts for the largest bandwidth use on the internet and other digital communication networks. As the number of connected user devices capable of receiving and displaying video increases, it is expected that the bandwidth demand for digital video usage will continue to grow.

SUMMARY

[004] The disclosed techniques may be used by video or image decoder or encoder embodiments for in which quantization parameters are under the coding tool of dependent quantization.

[005] In one example aspect, a method of processing video is disclosed. The method includes performing a determination, by a processor, that dependent scalar quantization is used to process a first video block; determining, by the processor, a first quantization parameter (QP) to be used for a deblocking filter for the first video block based on the determination that dependent scalar quantization being used to process the first video block; and performing further processing of the first video block using the deblocking filter in accordance with the first QP.

[006] In another example aspect, a video processing method is disclosed. The method comprises determining one or multiple deblocking filter parameters to be used in a deblocking filter process of a current video block based on whether dependent scalar quantization is used to

process the current video block, wherein a set of admissible reconstruction value for a transform coefficient corresponding to the dependent scalar quantization depends on at least one transform coefficient level that precedes a current transform coefficient level; and performing the deblocking filter process on the current video block in accordance with the one or multiple deblocking filter parameter.

[007] In another example aspect, a video processing method is disclosed. The method comprises determining whether to apply deblocking filter process based on whether dependent scalar quantization is used to process a current video block, wherein a set of admissible reconstruction value for a transform coefficient corresponding to the dependent scalar quantization depends on at least one transform coefficient level that precedes a current transform coefficient level; and performing the deblocking filter process on the current video block base on the determination of applying the deblocking filter process.

[008] In another example aspect, a video processing method is disclosed. The method comprises determining a quantization parameter to be used in a dependent scalar quantization of a current video block in case that the dependent scalar quantization is enabled for the current video block, wherein a set of admissible reconstruction value for a transform coefficient corresponding to the dependent scalar quantization depends on at least one transform coefficient level that precedes a current transform coefficient level; and performing the dependent scalar quantization on the current video block in accordance with the determined quantization parameter, wherein the determined quantization parameter is also applied to a video process different from the dependent scalar quantization using a quantization parameter as an input parameter of the current video block.

[009] In another example aspect, a video processing method is disclosed. The method comprises determining a quantization parameter to be used in a dependent scalar dequantization of a current video block in case that the dependent scalar dequantization is enabled for the current video block, wherein a set of admissible reconstruction value for a transform coefficient corresponding to the dependent scalar dequantization depends on at least one transform coefficient level that precedes a current transform coefficient level; performing the dependent scalar dequantization on the current video block in accordance with the determined quantization parameter, wherein the determined quantization parameter is also applied to a video process different from the

dependent scalar dequantization using a quantization parameter as an input parameter of the current video block.

[0010] In another example aspect, the above-described method may be implemented by a video decoder apparatus that comprises a processor.

[0011] In another example aspect, the above-described method may be implemented by a video encoder apparatus that comprises a processor.

[0012] In another example aspect, the above-described method may be implemented by an apparatus in a video system that comprises a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to implement the above-described method.

[0013] In yet another example aspect, these methods may be embodied in the form of processor-executable instructions and stored on a computer-readable program medium.

[0014] These, and other, aspects are further described in the present document.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] FIG. 1 shows an example of two scalar quantizers used in dependent quantization.

[0016] FIG. 2 shows an example of state transition and quantizer selection for dependent quantization.

[0017] FIG. 3 shows an example of an overall processing flow of a deblocking filter process.

[0018] FIG. 4 shows an example of a flow diagram for Bs calculation.

[0019] FIG. 5 shows examples of referred information for Bs calculation at a coding tree unit (CTU) boundary.

[0020] FIG. 6 shows an example of a pixels involved in a filter on/off decision and a strong/weak filter selection.

[0021] FIG. 7 shows an example of a deblocking behavior in a 4:2:2 chroma format.

[0022] FIG. 8 is a block diagram of an example of a video processing apparatus.

[0023] FIG. 9 shows a block diagram of an example implementation of a video encoder.

[0024] FIG. 10 is a flowchart for an example of a video processing method.

[0025] FIG. 11 is a flowchart for an example of a video processing method.

[0026] FIG. 12 is a flowchart for an example of a video processing method.

[0027] FIG. 13 is a flowchart for an example of a video processing method.

[0028] FIG. 14 is a flowchart for an example of a video processing method.

DETAILED DESCRIPTION

[0029] The present document provides various techniques that can be used by a decoder of image or video bitstreams to improve the quality of decompressed or decoded digital video or images. For brevity, the term "video" is used herein to include both a sequence of pictures (traditionally called video) and individual images. Furthermore, a video encoder may also implement these techniques during the process of encoding in order to reconstruct decoded frames used for further encoding.

[0030] Section headings are used in the present document for ease of understanding and do not limit the embodiments and techniques to the corresponding sections. As such, embodiments from one section can be combined with embodiments from other sections.

[0031] 1. Summary

[0032] This patent document is related to video coding technologies. Specifically, it is related to the usage of quantization parameters when dependent quantization is utilized. It may be applied to the existing video coding standard like HEVC, or the standard (Versatile Video Coding) to be finalized. It may be also applicable to future video coding standards or video codec.

[0033] 2. Background

[0034] Video coding standards have evolved primarily through the development of the well-known ITU-T and ISO/IEC standards. The ITU-T produced H.261 and H.263, ISO/IEC produced MPEG-1 and MPEG-4 Visual, and the two organizations jointly produced the H.262/MPEG-2 Video and H.264/MPEG-4 Advanced Video Coding (AVC) and H.265/HEVC standards. Since H.262, the video coding standards are based on the hybrid video coding structure wherein temporal prediction plus transform coding are utilized. To explore the future video coding technologies beyond HEVC, Joint Video Exploration Team (JVET) was founded by VCEG and MPEG jointly in 2015. Since then, many new methods have been adopted by JVET and put into the reference software named Joint Exploration Model (JEM). In April 2018, the Joint Video Expert Team (JVET) between VCEG (Q6/16) and ISO/IEC JTC1 SC29/WG11 (MPEG) was created to work on the VVC standard targeting at 50% bitrate reduction compared to HEVC.

[0035] FIG. 9 is a block diagram of an example implementation of a video encoder. FIG. 9 shows that the encoder implementation has a feedback path built in in which the video encoder also performs video decoding functionality (reconstructing compressed representation of video data for use in encoding of next video data).

[0036] 2.1 Dependent Scalar Quantization

[0037] Dependent scalar quantization is proposed, and it refers to an approach in which the set of admissible reconstruction values for a transform coefficient depends on the values of the transform coefficient levels that precede the current transform coefficient level in reconstruction order. The main effect of this approach is that, in comparison to conventional independent scalar quantization (as used in HEVC and VTM-1), the admissible reconstruction vectors (given by all reconstructed transform coefficients of a transform block) are packed denser in the N-dimensional vector space (N represents the number of transform coefficients in a transform block). That means, for a given average number of admissible reconstruction vectors per N-dimensional unit volume, the average distance (or MSE distortion) between an input vector and the closest reconstruction vector is reduced (for typical distributions of input vectors). Eventually, this effect can result in an improved rate-distortion efficiency.

[0038] The approach of dependent scalar quantization is realized by: (a) defining two scalar quantizers with different reconstruction levels and (b) defining a process for switching between the two scalar quantizers.

[0039] Figure 1 is an illustration of the two scalar quantizers used in the proposed approach of dependent quantization.

[0040] The two scalar quantizers used, denoted by Q0 and Q1, are illustrated in Figure 1. The location of the available reconstruction levels is uniquely specified by a quantization step size Δ. If we neglect the fact that the actual reconstruction of transform coefficients uses integer arithmetic, the two scalar quantizers Q0 and Q1 are characterized as follows:

[0041] Q0: The reconstruction levels of the first quantizer Q0 are given by the even integer multiples of the quantization step size Δ. When this quantizer is used, a reconstructed transform coefficient t' is calculated according to

[0042] $t' = 2 \cdot k \cdot \Delta$,

[0043] where k denotes the associated transform coefficient level (transmitted quantization index).

[0044] **Q1**: The reconstruction levels of the second quantizer Q1 are given by the odd integer multiples of the quantization step size Δ and, in addition, the reconstruction level equal to zero. The mapping of transform coefficient levels k to reconstructed transform coefficients t' is specified by

[0045] $t' = (2 \cdot k - sgn(k)) \cdot \Delta$,

[0046] where sgn(·) denotes the signum function

[0047] sgn(x) = (k == 0 ? 0 : (k < 0 ? –1 : 1)).

[0048] The scalar quantizer used (Q0 or Q1) is not explicitly signalled in the bitstream. Instead, the quantizer used for a current transform coefficient is determined by the parities of the transform coefficient levels that precede the current transform coefficient in coding/reconstruction order.

[0049] Figure 2 is an example of a state transition and quantizer selection for the proposed dependent quantization.

[0050] As illustrated in Figure 2, the switching between the two scalar quantizers (Q0 and Q1) is realized via a state machine with four states. The state can take four different values: 0, 1, 2, 3. It is uniquely determined by the parities of the transform coefficient levels preceding the current transform coefficient in coding/reconstruction order. At the start of the inverse quantization for a transform block, the state is set equal to 0. The transform coefficients are reconstructed in scanning order (i.e., in the same order they are entropy decoded). After a current transform coefficient is reconstructed, the state is updated as shown in Figure 2, where k denotes the value of the transform coefficient level. Note that the next state only depends on the current state and the parity (k & 1) of the current transform coefficient level k. With k representing the value of the current transform coefficient level, the state update can be written as

[0051] state = stateTransTable[state][k & 1],

[0052] where stateTransTable represents the table shown in Figure 2 and the operator & specifies the bit-wise “and” operator in two’s-complement arithmetic. Alternatively, the state transition can also be specified without a table look-up:

[0053] state = (32040 >> ((state << 2) + ((k & 1) << 1))) & 3

[0054] At this, the 16-bit value 32040 specifies the state transition table.

[0055] The state uniquely specifies the scalar quantizer used. If the state for a current transform coefficient is equal to 0 or 1, the scalar quantizer Q0 is used. Otherwise (the state is equal to 2 or 3), the scalar quantizer Q1 is used.

[0056] The detailed scaling process is described as follows.

[0057] 7.3.4.9 Residual coding syntax

residual_coding(x0, y0, log2TbWidth, log2TbHeight, cIdx) {	**Descriptor**
if(transform_skip_enabled_flag && (cIdx != 0 || cu_mts_flag[x0][y0] == 0) && (log2TbWidth <= 2) && (log2TbHeight <= 2))	
transform_skip_flag[x0][y0][cIdx]	ae(v)
last_sig_coeff_x_prefix	ae(v)
last_sig_coeff_y_prefix	ae(v)
if(last_sig_coeff_x_prefix > 3)	
last_sig_coeff_x_suffix	ae(v)
if(last_sig_coeff_y_prefix > 3)	
last_sig_coeff_y_suffix	ae(v)
log2SbSize = (Min(log2TbWidth, log2TbHeight) < 2 ? 1 : 2)	
numSbCoeff = 1 << (log2SbSize << 1)	
lastScanPos = numSbCoeff	
lastSubBlock = (1 << (log2TbWidth + log2TbHeight − 2 * log2SbSize)) − 1	
do {	
if(lastScanPos == 0) {	
lastScanPos = numSbCoeff	
lastSubBlock− −	
}	
lastScanPos− −	
xS = DiagScanOrder[log2TbWidth − log2SbSize][log2TbHeight − log2SbSize] [lastSubBlock][0]	
yS = DiagScanOrder[log2TbWidth − log2SbSize][log2TbHeight − log2SbSize] [lastSubBlock][1]	
xC = (xS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][lastScanPos][0]	
yC = (yS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][lastScanPos][1]	
} while((xC != LastSignificantCoeffX) || (yC != LastSignificantCoeffY))	
numSigCoeff = 0	
QState = 0	
for(i = lastSubBlock; i >= 0; i− −) {	
startQStateSb = QState	
xS = DiagScanOrder[log2TbWidth − log2SbSize][log2TbHeight − log2SbSize] [lastSubBlock][0]	
yS = DiagScanOrder[log2TbWidth − log2SbSize][log2TbHeight − log2SbSize] [lastSubBlock][1]	
inferSbDcSigCoeffFlag = 0	
if((i < lastSubBlock) && (i > 0)) {	
coded_sub_block_flag[xS][yS]	ae(v)
inferSbDcSigCoeffFlag = 1	

}	
firstSigScanPosSb = numSbCoeff	
lastSigScanPosSb = −1	
for(n = (i == lastSubBlock) ? lastScanPos − 1 : numSbCoeff − 1; n >= 0; n− −) {	
xC = (xS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][n][0]	
yC = (yS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][n][1]	
if(coded_sub_block_flag[xS][yS] && (n > 0 || !inferSbDcSigCoeffFlag)) {	
sig_coeff_flag[xC][yC]	ae(v)
if(sig_coeff_flag[xC][yC])	
inferSbDcSigCoeffFlag = 0	
}	
if(sig_coeff_flag[xC][yC]) {	
numSigCoeff++	
par_level_flag[n]	ae(v)
rem_abs_gt1_flag[n]	ae(v)
if(lastSigScanPosSb == −1)	
lastSigScanPosSb = n	
firstSigScanPosSb = n	
}	
AbsLevelPass1[xC][yC] = sig_coeff_flag[xC][yC] + par_level_flag[n] + 2 * rem_abs_gt1_flag[n]	
if(dep_quant_enabled_flag)	
QState = QStateTransTable[QState][par_level_flag[n]]	
}	
for(n = numSbCoeff − 1; n >= 0; n− −) {	
if(rem_abs_gt1_flag[n])	
rem_abs_gt2_flag[n]	ae(v)
}	
for(n = numSbCoeff − 1; n >= 0; n− −) {	
xC = (xS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][n][0]	
yC = (yS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][n][1]	
if(rem_abs_gt2_flag[n])	
abs_remainder[n]	
AbsLevel[xC][yC] = AbsLevelPass1[xC][yC] + 2 * (rem_abs_gt2_flag[n] + abs_remainder[n])	
}	
if(dep_quant_enabled_flag || !sign_data_hiding_enabled_flag)	
signHidden = 0	
else	
signHidden = (lastSigScanPosSb − firstSigScanPosSb > 3 ? 1 : 0)	
for(n = numSbCoeff − 1; n >= 0; n− −) {	
xC = (xS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][n][0]	
yC = (yS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][n][1]	

if(sig_coeff_flag[xC][yC] && (!signHidden || (n != firstSigScanPosSb)))	
coeff_sign_flag[n]	ae(v)
}	
if(dep_quant_enabled_flag) {	
QState = startQStateSb	
for(n = numSbCoeff − 1; n >= 0; n− −) {	
xC = (xS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][n][0]	
yC = (yS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][n][1]	
if(sig_coeff_flag[xC][yC])	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = (2 * AbsLevel[xC][yC] − (QState > 1 ? 1 : 0)) * (1 − 2 * coeff_sign_flag[n])	
QState = QStateTransTable[QState][par_level_flag[n]]	
} else {	
sumAbsLevel = 0	
for(n = numSbCoeff − 1; n >= 0; n− −) {	
xC = (xS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][n][0]	
yC = (yS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][n][1]	
if(sig_coeff_flag[xC][yC]) {	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = AbsLevel[xC][yC] * (1 − 2 * coeff_sign_flag[n])	
if(signHidden) {	
sumAbsLevel += AbsLevel[xC][yC]	
if((n == firstSigScanPosSb) && (sumAbsLevel % 2) == 1))	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = −TransCoeffLevel[x0][y0][cIdx][xC][yC]	
}	
}	
}	
}	
}	
if(cu_mts_flag[x0][y0] && (cIdx == 0) && ((CuPredMode[x0][y0] == MODE_INTRA && numSigCoeff > 2) || (CuPredMode[x0][y0] == MODE_INTER))) {	
mts_idx[x0][y0]	ae(v)
}	

[0058] 8.4.3 Scaling process for transform coefficients

[0059] Inputs to this process are:

[0060] a luma location (xTbY, yTbY) specifying the top-left sample of the current luma transform block relative to the top left luma sample of the current picture,

[0061] a variable nTbW specifying the transform block width,

[0062] a variable nTbH specifying the transform block height,

[0063] a variable cIdx specifying the colour component of the current block,

[0064] a variable bitDepth specifying the bit depth of the current colour component.

[0065] Output of this process is the (nTbW)x(nTbH) array d of scaled transform coefficients with elements d[x][y].

[0066] The quantization parameter qP is derived as follows:

[0067] If cIdx is equal to 0, the following applies:

[0068] qP = Qp′Y (8-383)

[0069] Otherwise, if cIdx is equal to 1, the following applies:

[0070] qP = Qp′Cb (8-384)

[0071] Otherwise (cIdx is equal to 2), the following applies:

[0072] qP = Qp′Cr (8-385)

[0073] The variables bdShift, rectNorm and bdOffset are derived as follows:

[0074] bdShift = bitDepth + (((Log2(nTbW) + Log2(nTbH)) & 1) * 8 +
(Log2(nTbW) + Log2(nTbH)) / 2) − 5 + dep_quant_enabled_flag (8-386)

[0075] rectNorm = ((Log2(nTbW) + Log2(nTbH)) & 1) == 1 ? 181 : 1 (8-387)

[0076] bdOffset = (1 << bdShift) >> 1 (8-388)

[0077] The list levelScale[ ] is specified as levelScale[k] = { 40, 45, 51, 57, 64, 72 } with k = 0..5.

[0078] For the derivation of the scaled transform coefficients d[x][y] with x = 0..nTbW − 1, y = 0..nTbH − 1, the following applies:

[0079] The intermediate scaling factor m[x][y] is set equal to 16.

[0080] The scaling factor ls[x][y] is derived as follows:

[0081] – If dep_quant_enabled_flag is equal to 1, the following applies:

[0082] ls[x][y] = (m[x][y] * levelScale[(qP + 1) % 6]) << ((qP + 1) / 6) (8-389)

[0083] – Otherwise (dep_quant_enabled_flag is equal to 0), the following applies:

[0084] ls[x][y] = (m[x][y] * levelScale[qP % 6]) << (qP / 6)(8-390)

[0085] The value dnc[x][y] is derived as follows:

[0086] dnc[x][y] = (8-391)

[0087] (TransCoeffLevel[xTbY][yTbY][cIdx][x][y] * ls[x][y] * rectNorm +bdOffset) >> bdShift

[0088] The scaled transform coefficient d[x][y] is derived as follows:

[0089] d[x][y] = Clip3(CoeffMin, CoeffMax, dnc[x][y])(8-392)

[0090] Suppose quantization parameter (QP) QPC is the used QP of the current CU, in dependent quantization, it actually uses QPC + 1 for quantization according to equation 8-389. However, if dependent quantization is not used, QPC is used for quantization.

[0091] 2.2 Deblocking filter

[0092] A deblocking filter process is performed for each CU in the same order as the decoding process. First vertical edges are filtered (horizontal filtering) then horizontal edges are filtered (vertical filtering). Filtering is applied to 8×8 block boundaries which are determined to be filtered, both for luma and chroma components. 4×4 block boundaries are not processed in order to reduce the complexity.

[0093] Figure 3 illustrates the overall flow of deblocking filter processes. A boundary can have three filtering status values: no filtering, weak filtering and strong filtering. Each filtering decision is based on boundary strength, Bs, and threshold values, β and t_C.

[0094] Figure 3 is an example of an overall processing flow of a deblocking filter process.

[0095] 2.2.1 Boundary decision

[0096] Two kinds of boundaries are involved in the deblocking filter process: TU boundaries and PU boundaries. CU boundaries are also considered, since CU boundaries are necessarily also TU and PU boundaries. When PU shape is 2NxN (N > 4) and RQT depth is equal to 1, TU boundaries at 8x8 block grid and PU boundaries between each PU inside the CU are also involved in the filtering.

[0097] 2.2.2. Boundary strength calculation

[0098] The boundary strength (Bs) reflects how strong a filtering process may be needed for the boundary. A value of 2 for Bs indicates strong filtering, 1 indicates weak filtering and 0 indicates no deblocking filtering.,

[0099] Let P and Q be defined as blocks which are involved in the filtering, where P represents the block located to the left (vertical edge case) or above (horizontal edge case) the boundary and Q represents the block located to the right (vertical edge case) or above (horizontal edge case) the

boundary. Figure 4 illustrates how the Bs value is calculated based on the intra coding mode, the existence of non-zero transform coefficients, reference picture, number of motion vectors and motion vector difference.

[00100] At the CTU boundary, information on every second block (on a 4×4 grid) to the left or above is re-used as depicted in Figure 5 in order to reduce line buffer memory requirement. Figure 5 is an example of referred information for Bs calculation at a CTU boundary.

[00101] Threshold variables

[00102] Threshold values β' and t_C' are involved in the filter on/off decision, strong and weak filter selection and weak filtering process. These are derived from the value of the luma quantization parameter Q as shown in Table 2-1. The derivation process of Q is described in 2.2.3.1

Q	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
β'	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	7	8
t_C'	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Q	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
β'	9	10	11	12	13	14	15	16	17	18	20	22	24	26	28	30	32	34	36
t_C'	1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4
Q	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53			
β'	38	40	42	44	46	48	50	52	54	56	58	60	62	64	-	-			
t_C'	5	5	6	6	7	8	9	10	11	13	14	16	18	20	22	24			

[00103] The variable β is derived from β' as follows:

[00104] $\beta = \beta' * (1 << (BitDepth_Y - 8))$

[00105] The variable t_C is derived from t_C' as follows:

[00106] $t_C = t_C' * (1 << (BitDepth_Y - 8))$

[00107] How to dervie t_C' and β' are described as follow.

[00108] 2.2.3.1 tc′ and β′

[00109] The decoding process of HEVC design for t_C' and β' is described in sub-clause 8.7.2.5.3.

[00110] 8.7.2.5.3 Decision process for luma block edges

[00111] Inputs to this process are:

[00112] – a luma picture sample array recPictureL,

[00113] – a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

[00114] – a luma location (xBl, yBl) specifying the top-left sample of the current luma block relative to the top-left sample of the current luma coding block,

[00115] – a variable edgeType specifying whether a vertical (EDGE_VER) or a horizontal (EDGE_HOR) edge is filtered,

[00116] – a variable bS specifying the boundary filtering strength.

[00117] Outputs of this process are:

[00118] – the variables dE, dEp, and dEq containing decisions,

[00119] – the variables β and tc.

[00120] If edgeType is equal to EDGE_VER, the sample values pi,k and qi,k with i = 0..3 and k = 0 and 3 are derived as follows:

[00121] qi,k = recPictureL[xCb + xBl + i][yCb + yBl + k] (8-284)

[00122] pi,k = recPictureL[xCb + xBl − i − 1][yCb + yBl + k] (8-285)

[00123] Otherwise (edgeType is equal to EDGE_HOR), the sample values pi,k and qi,k with i = 0..3 and k = 0 and 3 are derived as follows:

[00124] qi,k = recPictureL[xCb + xBl + k][yCb + yBl + i] (8-286)

[00125] pi,k = recPictureL[xCb + xBl + k][yCb + yBl − i − 1] (8-287)

[00126] The variables QpQ and QpP are set equal to the QpY values of the coding units which include the coding blocks containing the sample q0,0 and p0,0, respectively.

[00127] A variable qPL is derived as follows:

[00128] qPL = ((QpQ + QpP + 1) >> 1) (8-288)

[00129] The value of the variable β′ is determined as specified in Table 8 11 based on the luma quantization parameter Q derived as follows:

[00130] Q = Clip3(0, 51, qPL + (slice_beta_offset_div2 << 1)) (8-289)

[00131] where slice_beta_offset_div2 is the value of the syntax element slice_beta_offset_div2 for the slice that contains sample q0,0.

[00132] The variable β is derived as follows:

[00133] β = β′ * (1 << (BitDepthY − 8)) (8-290)

[00134] The value of the variable tc′ is determined as specified in Table 8 11 based on the luma quantization parameter Q derived as follows:

[00135] Q = Clip3(0, 53, qPL + 2 * (bS − 1) + (slice_tc_offset_div2 << 1)) (8-291)

[00136] where slice_tc_offset_div2 is the value of the syntax element slice_tc_offset_div2 for the slice that contains sample q0,0.

[00137] The variable tc is derived as follows:

[00138] tc = tc′ * (1 << (BitDepthY − 8)) (8-292)

[00139] Depending on the value of edgeType, the following applies:

[00140] − If edgeType is equal to EDGE_VER, the following ordered steps apply:

[00141] The variables dpq0, dpq3, dp, dq, and d are derived as follows:

[00142] dp0 = Abs(p2,0 − 2 * p1,0 + p0,0) (8-293)

[00143] dp3 = Abs(p2,3 − 2 * p1,3 + p0,3) (8-294)

[00144] dq0 = Abs(q2,0 − 2 * q1,0 + q0,0) (8-295)

[00145] dq3 = Abs(q2,3 − 2 * q1,3 + q0,3) (8-296)

[00146] dpq0 = dp0 + dq0 (8-297)

[00147] dpq3 = dp3 + dq3 (8-298)

[00148] dp = dp0 + dp3 (8-299)

[00149] dq = dq0 + dq3 (8-300)

[00150] d = dpq0 + dpq3 (8-301)

[00151] The variables dE, dEp, and dEq are set equal to 0.

[00152] When d is less than β, the following ordered steps apply:

[00153] The variable dpq is set equal to 2 * dpq0.

[00154] For the sample location (xCb + xBl, yCb + yBl), the decision process for a luma sample as specified in subclause 8.7.2.5.6 is invoked with sample values pi,0, qi,0 with i = 0..3, the variables dpq, β, and tc as inputs, and the output is assigned to the decision dSam0.

[00155] The variable dpq is set equal to 2 * dpq3.

[00156] For the sample location (xCb + xBl, yCb + yBl + 3), the decision process for a luma sample as specified in subclause 8.7.2.5.6 is invoked with sample values pi,3, qi,3 with i = 0..3, the variables dpq, β, and tc as inputs, and the output is assigned to the decision dSam3.

[00157] The variable dE is set equal to 1.

[00158] When dSam0 is equal to 1 and dSam3 is equal to 1, the variable dE is set equal to 2.

[00159] When dp is less than (β + (β >> 1)) >> 3, the variable dEp is set equal to 1.

[00160] When dq is less than (β + (β >> 1)) >> 3, the variable dEq is set equal to 1.

[00161] – Otherwise (edgeType is equal to EDGE_HOR), the following ordered steps apply:

[00162] The variables dpq0, dpq3, dp, dq, and d are derived as follows:

[00163] dp0 = Abs(p2,0 − 2 * p1,0 + p0,0) (8-302)

[00164] dp3 = Abs(p2,3 − 2 * p1,3 + p0,3) (8-303)

[00165] dq0 = Abs(q2,0 − 2 * q1,0 + q0,0) (8-304)

[00166] dq3 = Abs(q2,3 − 2 * q1,3 + q0,3) (8-305)

[00167] dpq0 = dp0 + dq0 (8-306)

[00168] dpq3 = dp3 + dq3 (8-307)

[00169] dp = dp0 + dp3 (8-308)

[00170] dq = dq0 + dq3 (8-309)

[00171] d = dpq0 + dpq3 (8-310)

[00172] The variables dE, dEp, and dEq are set equal to 0.

[00173] When d is less than β, the following ordered steps apply:

[00174] The variable dpq is set equal to 2 * dpq0.

[00175] For the sample location (xCb + xBl, yCb + yBl), the decision process for a luma sample as specified in subclause 8.7.2.5.6 is invoked with sample values p0,0, p3,0, q0,0, and q3,0, the variables dpq, β, and tc as inputs, and the output is assigned to the decision dSam0.

[00176] The variable dpq is set equal to 2 * dpq3.

[00177] For the sample location (xCb + xBl + 3, yCb + yBl), the decision process for a luma sample as specified in subclause 8.7.2.5.6 is invoked with sample values p0,3, p3,3, q0,3, and q3,3, the variables dpq, β, and tc as inputs, and the output is assigned to the decision dSam3.

[00178] The variable dE is set equal to 1.

[00179] When dSam0 is equal to 1 and dSam3 is equal to 1, the variable dE is set equal to 2.

[00180] When dp is less than (β + (β >> 1)) >> 3, the variable dEp is set equal to 1.

[00181] When dq is less than (β + (β >> 1)) >> 3, the variable dEq is set equal to 1.

[00182] Table 8-11 below shows derivation of threshold variables β′ and tc′ from input Q.

Q	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
β′	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	7	8
t_C'	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Q	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
β′	9	10	11	12	13	14	15	16	17	18	20	22	24	26	28	30	32	34	36
t_C'	1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4
Q	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53			
β′	38	40	42	44	46	48	50	52	54	56	58	60	62	64	-	-			
t_C'	5	5	6	6	7	8	9	10	11	13	14	16	18	20	22	24			

[00183] 2.2.4 Filter on/off decision for 4 lines

[00184] The filter on/off decision is made using 4 lines grouped as a unit, to reduce computational complexity. Figure 6 illustrates the pixels involving in the decision. The 6 pixels in the two red boxes in the first 4 lines are used to determine whether the filter is on or off for those 4 lines. The 6 pixels in the two red boxes in the second group of 4 lines are used to determine whether the filter is on or off for the second group of 4 lines.

[00185] Figure 6 shows an example of pixels involved in an on/off decisions and a strong/weak filter selection.

[00186] The following variables are defined:

[00187] $dp0 = | p_{2,0} - 2*p_{1,0} + p_{0,0} |$

[00188] $dp3 = | p_{2,3} - 2*p_{1,3} + p_{0,3} |$

[00189] $dq0 = | q_{2,0} - 2*q_{1,0} + q_{0,0} |$

[00190] $dq3 = | q_{2,3} - 2*q_{1,3} + q_{0,3} |$

[00191] If $dp0+dq0+dp3+dq3 < \beta$, filtering for the first four lines is turned on and the strong/weak filter selection process is applied. If this condition is not met, no filtering is done for the first 4 lines.

[00192] Additionally, if the condition is met, the variables dE, dEp1 and dEp2 are set as follows:

[00193] dE is set equal to 1

[00194] If $dp0 + dp3 < (\beta + (\beta >> 1)) >> 3$, the variable dEp1 is set equal to 1

[00195] If $dq0 + dq3 < (\beta + (\beta >> 1)) >> 3$, the variable dEq1 is set equal to 1

[00196] A filter on/off decision is made in a similar manner as described above for the second group of 4 lines.

[00197] 2.2.5. Strong/weak filter selection for 4 lines

[00198] If filtering is turned on, a decision is made between strong and weak filtering. The pixels involved are the same as those used for the filter on/off decision. If the following two sets of conditions are met, a strong filter is used for filtering of the first 4 lines. Otherwise, a weak filter is used.

[00199] 1) $2*(dp0+dq0) < (\beta >> 2)$, $| p3_0 - p0_0 | + | q0_0 - q3_0 | < (\beta >> 3)$ and $| p0_0 - q0_0 | < (5* t_C + 1) >> 1$

[00200] 2) $2*(dp3+dq3) < (\beta >> 2)$, $| p3_3 - p0_3 | + | q0_3 - q3_3 | < (\beta >> 3)$ and $| p0_3 - q0_3 | < (5* t_C + 1) >> 1$

[00201] The decision on whether to select strong or weak filtering for the second group of 4 lines is made in a similar manner.

[00202] 2.2.6 Strong filtering

[00203] For strong filtering, the filtered pixel values are obtained by the following equations. Note that three pixels are modified using four pixels as an input for each P and Q block, respectively.

[00204] $p_0' = (p_2 + 2*p_1 + 2*p_0 + 2*q_0 + q_1 + 4) >> 3$

[00205] $q_0' = (p_1 + 2*p_0 + 2*q_0 + 2*q_1 + q_2 + 4) >> 3$

[00206] $p_1' = (p_2 + p_1 + p_0 + q_0 + 2) >> 2$

[00207] $q_1' = (p_0 + q_0 + q_1 + q_2 + 2) >> 2$

[00208] $p_2' = (2*p_3 + 3*p_2 + p_1 + p_0 + q_0 + 4) >> 3$

[00209] $q_2' = (p_0 + q_0 + q_1 + 3*q_2 + 2*q_3 + 4) >> 3$

[00210] 2.2.7 Weak filtering

[00211] Δ is defined as follows.

[00212] $\Delta = (9 * (q_0 - p_0) - 3 * (q_1 - p_1) + 8) >> 4$

[00213] When $abs(\Delta)$ is less than $t_C * 10$,

[00214] $\Delta = Clip3(- t_C , t_C , \Delta)$

[00215] $p_0' = Clip1_Y(p_0 + \Delta)$

[00216] $q_0' = Clip1_Y(q_0 - \Delta)$

[00217] If dEp1 is equal to 1,

[00218] $\Delta p = Clip3(-(t_C >> 1), t_C >> 1, (((p_2 + p_0 + 1) >> 1) - p_1 + \Delta) >>1)$

[00219] $p_1' = Clip1_Y(p_1 + \Delta p)$

[00220] If dEq1 is equal to 1,

[00221] $\Delta q = Clip3(-(t_C >> 1), t_C >> 1, (((q_2 + q_0 + 1) >> 1) - q_1 - \Delta) >>1)$

[00222] $q_1' = Clip1_Y(q_1 + \Delta q)$

[00223] Note that a maximum of two pixels are modified using three pixels as an input for each P and Q block, respectively.

[00224] 2.2.8 Chroma Filtering

[00225] The boundary strength Bs for chroma filtering is inherited from luma. If Bs > 1, chroma filtering is performed. No filter selection process is performed for chroma, since only one filter can be applied. The filtered sample values p_0' and q_0' are derived as follows.

[00226] $\Delta = Clip3(-t_C, t_C, ((((q_0 - p_0) << 2) + p_1 - q_1 + 4) >> 3))$

[00227] $p_0' = Clip1_C(p_0 + \Delta)$

[00228] $q_0' = Clip1_C(q_0 - \Delta)$

[00229] When the 4:2:2 chroma format is in use, each chroma block has a rectangular shape and is coded using up to two square transforms. This process introduces additional boundaries between the transform blocks in chroma. These boundaries are not deblocked (thick dashed lines running horizontally through the center in Figure 7).

[00230] Figure 7 is an example of a deblocking behavior in a 4:2:2 chroma format.

[00231] 2.3 Extension of quantization parameter value range

[00232] QP range is extended from [0, 51] to [0, 63], and the derivation of tC′ and β′ are as follows. The table size of β and tc are increased from 52 and 54 to 64 and 66 respectively.

[00233] 8.7.2.5.3 Decision process for luma block edges

[00234] The variable qPL is derived as follows:

[00235] qPL = ((QpQ + QpP + 1) >> 1) (2-38)

[00236] The value of the variable β′ is determined as specified in Table 2-3 based on the luma quantization parameter Q derived as follows:

[00237] Q = Clip3(0, 63~~-51~~, qPL + (slice_beta_offset_div2 << 1)) (2-39)

[00238] where slice_beta_offset_div2 is the value of the syntax element slice_beta_offset_div2 for the slice that contains sample q0,0.

[00239] The variable β is derived as follows:

[00240] $\beta = \beta' * (1 << (BitDepthY - 8))$ (2-40)

[00241] The value of the variable tc′ is determined as specified in Table 2 3 based on the luma quantization parameter Q derived as follows:

[00242] Q = Clip3(0, 65 ~~53~~, qPL + 2 * (bS − 1) + (slice_tc_offset_div2 << 1)) (2-41)

[00243] where slice_tc_offset_div2 is the value of the syntax element slice_tc_offset_div2 for the slice that contains sample q0,0.

[00244] The variable tc is derived as follows:

[00245] $tc = tc' * (1 << (BitDepthY - 8))$

[00246] Table 2-3 below is for a derivation of threshold variables β' and tc' from input Q.

Q	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
β'	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	7	8
t_C'	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Q	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
β'	9	10	11	12	13	14	15	16	17	18	20	22	24	26	28	30	32	34	36
t_C'	1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4
Q	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
β'	38	40	42	44	46	48	50	52	54	56	58	60	62	64	66	68	70	72	74
t_C'	5	5	6	6	7	8	9	10	11	13	14	16	18	20	22	24	26	28	30
Q	57	58	59	60	61	62	63	64	65										
β'	76	78	80	82	84	86	88	-	-										
t_C'	32	34	36	38	40	42	44	46	48										

[00247] 2.4 Initialization of context variables

[00248] In context-based adaptive binary arithmetic coding (CABAC), initial state of context variables depends on QP of the slice. The initialization process is described as follows.

[00249] 9.3.2.2 Initialization process for context variables

[00250] Outputs of this process are the initialized CABAC context variables indexed by ctxTable and ctxIdx.

[00251] Table 9-5 to Table 9-31 contain the values of the 8 bit variable initValue used in the initialization of context variables that are assigned to all syntax elements in subclauses 7.3.8.1 through 7.3.8.11, except end_of_slice_segment_flag, end_of_sub_stream_one_bit, and pcm_flag.

[00252] For each context variable, the two variables pStateIdx and valMps are initialized.

[00253] NOTE 1 – The variable pStateIdx corresponds to a probability state index and the variable valMps corresponds to the value of the most probable symbol as further described in subclause 9.3.4.3.

[00254] From the 8 bit table entry initValue, the two 4 bit variables slopeIdx and offsetIdx are derived as follows:

[00255] slopeIdx = initValue >> 4

offsetIdx = initValue & 15 (9-4)

[00256] The variables m and n, used in the initialization of context variables, are derived from slopeIdx and offsetIdx as follows:

[00257] m = slopeIdx * 5 − 45

n = (offsetIdx << 3) − 16 (9-5)

[00258] The two values assigned to pStateIdx and valMps for the initialization are derived from $SliceQp_Y$, which is derived in Equation 7-40. Given the variables m and n, the initialization is specified as follows:

[00259] preCtxState = Clip3(1, 126, ((m * Clip3(0, 51, $SliceQp_Y$)) >> 4) + n)

valMps = (preCtxState <= 63) ? 0 : 1

pStateIdx = valMps ? (preCtxState − 64) : (63 − preCtxState) (9-6)

[00260] In Table 9-4, the ctxIdx for which initialization is needed for each of the three initialization types, specified by the variable initType, are listed. Also listed is the table number that includes the values of initValue needed for the initialization. For P and B slice types, the derivation of initType depends on the value of the cabac_init_flag syntax element. The variable initType is derived as follows:

[00261] if(slice_type == I)

initType = 0

else if(slice_type == P)

initType = cabac_init_flag ? 2 : 1 (9-7)

else

initType = cabac_init_flag ? 1 : 2

[00262] 3.Examples of Problems Solved by Embodiments

[00263] In dependent quantization, $QP_C + 1$ is used for quantization. However, in deblocking filter process, QP_C is used, which is inconsistent.

[00264] In addition, if $QP_C + 1$ is used in the deblocking filter process, how to handle the mapping table between Q and t_C/β is unknown since QP_C may be set to the maximum value, i.e., 63.

[00265] 4. Examples of embodiments

[00266] To address the problem, several methods can be applied to the deblocking filter process which relies on the quantization parameters of blocks to be filtered. It may also be applicable to

other kinds of procedures, like bilateral filter, which depend on the quantization parameter associated with one block.

[00267] The detailed listing of techniques below should be considered as examples to explain general concepts. These inventions should not be interpreted in a narrow way. Furthermore, techniques can be combined in any manner. Denote the allowed minimum and maximum QP are QPmin and QPmax respectively. Denote signaled quantization parameter of the current CU is QP_C and the quantization/dequantization process relies on $QP_C + N$ to derive the quantization step (e.g., N = 1 in current dependent quantization design). Denote Tc'[n] and β'[n] as the n-th entry of Tc' and β' table.

1. It is proposed that whether to and how to apply deblocking filter may depend on whether dependent scalar quantization is used or not.
    a. For example, QP used in deblocking filter depends on whether dep_quant_enabled_flag is equal to 0 or 1.
2. It is proposed that one same QP is used in dependent quantization, deblocking filter or/and any other process using QP as an input parameter.
    a. In one example, QP_C instead of $QP_C + N$ is used in dependent quantization. N is an integer such as 1, 3, 6, 7 or -1, -3, -6, -7.
    b. In one example, $QP_C + N$ is used in deblocking filter or/and any other process using QP as an input parameter. N is an integer such as 1, 3, 6, 7 or -1, -3, -6, -7.
    c. $QP_C + N$ is clipped to a valid range before being used.
3. It is proposed that the allowed QP range is set to $[QP_{min} - N, QP_{max} - N]$ instead of $[QP_{min}, QP_{max}]$ for dependent quantization when $QP_C + N$ is used in it.
    a. Alternatively, the allowed QP range is set to $[Max(QP_{min} - N, QP_{min}), Min(QP_{max} - N, QP_{max})]$.
4. It is proposed that, as compared with no dependent quantization is used, a weaker/stronger deblocking filter is used when dependent quantization is used.
    a. In one example, the encoder selects a weaker/stronger deblocking filter when dependent quantization is enabled and signal it to the decoder.
    b. In one example, smaller/larger threshold values Tc and β are used implictly at both encoder and decoder, when dependent quantization is enabled.

5. When $QP_C + N$ is used in deblocking filter process, more entries may be required in Tc' and β' table (e.g., Table 2-3) for $QP_{max} + N$.
    a. Alternatively, the same table may be utilized, however, whenever $QP_C + N$ is firstly clipped to be within the same range $[QP_{min}, QP_{max}]$.
    b. In one example, Tc' table is extended as: tc'[66] = 50 and tc'[67] = 52.
    c. In one example, Tc' table is extended as: tc'[66] = 49 and tc'[67] = 50.
    d. In one example, Tc' table is extended as: tc'[66] = 49 and tc'[67] = 51.
    e. In one example, Tc' table is extended as: tc'[66] = 48 and tc'[67] = 50.
    f. In one example, Tc' table is extended as: tc'[66] = 50 and tc'[67] = 51.
    g. In one example, β' table is extended as: β'[64] = 90 and β'[65] = 92.
    h. In one example, β' table is extended as: β'[64] = 89 and β'[65] = 90.
    i. In one example, β' table is extended as: β'[64] = 89 and β'[65] = 91.
    j. In one example, β' table is extended as: β'[64] = 88 and β'[65] = 90.
    k. In one example, β' table is extended as: β'[64] = 90 and β'[65] = 91.
6. The initialization of CABAC context depends on the $QP_C + N$ instead of QP_C when dependent quantization is enabled.
7. The high-level signaled quantization parameter may be assigned with different semantics based on whether dependent quantization is used or not.
    a. In one example, for the qp indicated in the picture parameter set/picture header (i.e., **init_qp_minus26** in HEVC), it may have different semantics.
        i. When dependent quantization is OFF, **init_qp_minus26** plus 26 specifies the initial value of $SliceQp_Y$ for each slice referring to the PPS or initial value of all tiles' quantization parameters referring to the PPS/picture header.
        ii. When dependent quantization is ON, **init_qp_minus26** plus 27 specifies the initial value of $SliceQp_Y$ for each slice referring to the PPS or initial value of all tiles' quantization parameters referring to the PPS/picture header.
    b. In one example, for the delta qp indicated in slice header/tile header/tile groups header (i.e., **slice_qp_delta** in HEVC), it may have different semantics.

i. When dependent quantization is OFF, **slice_qp_delta** specifies the initial value of Qp_Y to be used for the coding blocks in the slice/tile/tile groups until modified by the value of CuQpDeltaVal in the coding unit layer. The initial value of the Qp_Y quantization parameter for the slice/tile/tile groups, $SliceQp_Y$, is derived as follows:

$SliceQp_Y = 26 + init\_qp\_minus26 + slice\_qp\_delta$

ii. When dependent quantization is ON, **slice_qp_delta** specifies the initial value of Qp_Y to be used for the coding blocks in the slice/tile/tile groups until modified by the value of CuQpDeltaVal in the coding unit layer. The initial value of the Qp_Y quantization parameter for the slice/tile/tile groups, $SliceQp_Y$, is derived as follows:

$SliceQp_Y = 26 + init\_qp\_minus26 + slice\_qp\_delta + 1$

8. Whether to enable or disable the proposed method may be signaled in SPS/PPS/VPS/sequence header/picture header/slice header/tile group header/group of CTUs, etc. al.

[00268] 5. Example of another Embodiment

[00269] In one embodiment, $QP_C + 1$ is used in deblocking filter. The newly added parts are highlighted.

[00270] 8.7.2.5.3 Decision process for luma block edges

[00271] Inputs to this process are:

[00272] – a luma picture sample array $recPicture_L$,

[00273] – a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

[00274] – a luma location (xBl, yBl) specifying the top-left sample of the current luma block relative to the top-left sample of the current luma coding block,

[00275] – a variable edgeType specifying whether a vertical (EDGE_VER) or a horizontal (EDGE_HOR) edge is filtered,

[00276] – a variable bS specifying the boundary filtering strength.

[00277] Outputs of this process are:

[00278] – the variables dE, dEp, and dEq containing decisions,

[00279] – the variables β and t_C.

[00280] If edgeType is equal to EDGE_VER, the sample values $p_{i,k}$ and $q_{i,k}$ with i = 0..3 and k = 0 and 3 are derived as follows:

[00281] $q_{i,k} = recPicture_L[xCb + xBl + i][yCb + yBl + k]$ (8-284)

[00282] $p_{i,k} = recPicture_L[xCb + xBl - i - 1][yCb + yBl + k]$ (8-285)

[00283] Otherwise (edgeType is equal to EDGE_HOR), the sample values $p_{i,k}$ and $q_{i,k}$ with i = 0..3 and k = 0 and 3 are derived as follows:

[00284] $q_{i,k} = recPicture_L[xCb + xBl + k][yCb + yBl + i]$ (8-286)

[00285] $p_{i,k} = recPicture_L[xCb + xBl + k][yCb + yBl - i - 1]$ (8-287)

[00286] The variables Qp_Q and Qp_P are set equal to the Qp_Y values of the coding units which include the coding blocks containing the sample $q_{0,0}$ and $p_{0,0}$, respectively.

[00287] If dep_quant_enabled_flag of the coding unit which includes the coding block containing the sample $q_{0,0}$ is equal to 1, Qp_Q is set equal to $Qp_Q + 1$. If dep_quant_enabled_flag of the coding unit which includes the coding block containing the sample $p_{0,0}$ is equal to 1, Qp_P is set equal to $Qp_P + 1$.

[00288] A variable qP_L is derived as follows:

[00289] $qP_L = ((Qp_Q + Qp_P + 1) >> 1)$ (8-288)

[00290] The value of the variable β' is determined as specified in Table 8-11 based on the luma quantization parameter Q derived as follows:

[00291] $Q = Clip3(0, 51, qP_L + (slice\_beta\_offset\_div2 << 1))$ (8-289)

[00292] where slice_beta_offset_div2 is the value of the syntax element slice_beta_offset_div2 for the slice that contains sample $q_{0,0}$.

[00293] The variable β is derived as follows:

[00294] $\beta = \beta' * (1 << (BitDepth_Y - 8))$ (8-290)

[00295] The value of the variable t_C' is determined as specified in Table 8-11 based on the luma quantization parameter Q derived as follows:

[00296] $Q = Clip3(0, 53, qP_L + 2 * (bS - 1) + (slice\_tc\_offset\_div2 << 1))$ (8-291)

[00297] where slice_tc_offset_div2 is the value of the syntax element slice_tc_offset_div2 for the slice that contains sample $q_{0,0}$.

[00298] The variable t_C is derived as follows:

[00299] $t_C = t_C' * (1 << (BitDepth_Y - 8))$ (8-292)

[00300] Depending on the value of edgeType, the following applies:

[00301] – If edgeType is equal to EDGE_VER, the following ordered steps apply:

[00302] The variables dpq0, dpq3, dp, dq, and d are derived as follows:

[00303] $dp0 = Abs(p_{2,0} - 2 * p_{1,0} + p_{0,0})$ (8-293)

[00304] $dp3 = Abs(p_{2,3} - 2 * p_{1,3} + p_{0,3})$ (8-294)

[00305] $dq0 = Abs(q_{2,0} - 2 * q_{1,0} + q_{0,0})$ (8-295)

[00306] $dq3 = Abs(q_{2,3} - 2 * q_{1,3} + q_{0,3})$ (8-296)

[00307] $dpq0 = dp0 + dq0$ (8-297)

[00308] $dpq3 = dp3 + dq3$ (8-298)

[00309] $dp = dp0 + dp3$ (8-299)

[00310] $dq = dq0 + dq3$ (8-300)

[00311] $d = dpq0 + dpq3$ (8-301)

[00312] The variables dE, dEp, and dEq are set equal to 0.

[00313] When d is less than β, the following ordered steps apply:

[00314] The variable dpq is set equal to 2 * dpq0.

[00315] For the sample location (xCb + xBl, yCb + yBl), the decision process for a luma sample as specified in subclause 8.7.2.5.6 is invoked with sample values $p_{i,0}$, $q_{i,0}$ with i = 0..3, the variables dpq, β, and t_C as inputs, and the output is assigned to the decision dSam0.

[00316] The variable dpq is set equal to 2 * dpq3.

[00317] For the sample location (xCb + xBl, yCb + yBl + 3), the decision process for a luma sample as specified in subclause 8.7.2.5.6 is invoked with sample values $p_{i,3}$, $q_{i,3}$ with i = 0..3, the variables dpq, β, and t_C as inputs, and the output is assigned to the decision dSam3.

[00318] The variable dE is set equal to 1.

[00319] When dSam0 is equal to 1 and dSam3 is equal to 1, the variable dE is set equal to 2.

[00320] When dp is less than $(\beta + (\beta >> 1)) >> 3$, the variable dEp is set equal to 1.

[00321] When dq is less than $(\beta + (\beta >> 1)) >> 3$, the variable dEq is set equal to 1.

[00322] – Otherwise (edgeType is equal to EDGE_HOR), the following ordered steps apply:

[00323] The variables dpq0, dpq3, dp, dq, and d are derived as follows:

[00324] $dp0 = Abs(p_{2,0} - 2 * p_{1,0} + p_{0,0})$ (8-302)

[00325] $dp3 = Abs(p_{2,3} - 2 * p_{1,3} + p_{0,3})$ (8-303)

[00326] $dq0 = Abs(q_{2,0} - 2 * q_{1,0} + q_{0,0})$ (8-304)

[00327] $dq3 = Abs(q_{2,3} - 2 * q_{1,3} + q_{0,3})$ (8-305)

[00328] dpq0 = dp0 + dq0 (8-306)

[00329] dpq3 = dp3 + dq3 (8-307)

[00330] dp = dp0 + dp3 (8-308)

[00331] dq = dq0 + dq3 (8-309)

[00332] d = dpq0 + dpq3 (8-310)

[00333] The variables dE, dEp, and dEq are set equal to 0.

[00334] When d is less than β, the following ordered steps apply:

[00335] The variable dpq is set equal to 2 * dpq0.

[00336] For the sample location (xCb + xBl, yCb + yBl), the decision process for a luma sample as specified in subclause 8.7.2.5.6 is invoked with sample values $p_{0,0}$, $p_{3,0}$, $q_{0,0}$, and $q_{3,0}$, the variables dpq, β, and t_C as inputs, and the output is assigned to the decision dSam0.

[00337] The variable dpq is set equal to 2 * dpq3.

[00338] For the sample location (xCb + xBl + 3, yCb + yBl), the decision process for a luma sample as specified in subclause 8.7.2.5.6 is invoked with sample values $p_{0,3}$, $p_{3,3}$, $q_{0,3}$, and $q_{3,3}$, the variables dpq, β, and t_C as inputs, and the output is assigned to the decision dSam3.

[00339] The variable dE is set equal to 1.

[00340] When dSam0 is equal to 1 and dSam3 is equal to 1, the variable dE is set equal to 2.

[00341] When dp is less than (β + (β >> 1)) >> 3, the variable dEp is set equal to 1.

[00342] When dq is less than (β + (β >> 1)) >> 3, the variable dEq is set equal to 1.

[00343] Table 8-11 below is for the derivation of threshold variables β′ and tc′ from input Q

Q	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
β′	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	7	8
t_C′	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Q	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
β′	9	10	11	12	13	14	15	16	17	18	20	22	24	26	28	30	32	34	36
t_C′	1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4
Q	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53			
β′	38	40	42	44	46	48	50	52	54	56	58	60	62	64	-	-			
t_C′	5	5	6	6	7	8	9	10	11	13	14	16	18	20	22	24			

[00344] FIG. 8 is a block diagram of a video processing apparatus 800. The apparatus 800 may be used to implement one or more of the methods described herein. The apparatus 800 may be

embodied in a smartphone, tablet, computer, Internet of Things (IoT) receiver, and so on. The apparatus 800 may include one or more processors 802, one or more memories 804 and video processing hardware 806. The processor(s) 802 may be configured to implement one or more methods described in the present document. The memory (memories) 804 may be used for storing data and code used for implementing the methods and techniques described herein. The video processing hardware 806 may be used to implement, in hardware circuitry, some techniques described in the present document.

[00345] FIG. 10 is a flowchart for a method 1000 of processing a video. The method 1000 includes performing (1005) a determination that dependent scalar quantization is used to process a first video block; determining (1010) a first quantization parameter (QP) to be used for a deblocking filter for the first video block based on the determination that dependent scalar quantization being used to process the first video block; and performing (1015) further processing of the first video block using the deblocking filter in accordance with the first QP.

[00346] With reference to method 1000, some examples of determining a candidate for encoding and their use are described in Section 4 of the present document. For example, as described in Section 4, a quantization parameter for deblocking filter can be determined depending on the usage of dependent scalar quantization.

[00347] With reference to method 1000, a video block may be encoded in the video bitstream in which bit efficiency may be achieved by using a bitstream generation rule related to motion information prediction.

[00348] The method can include wherein the determination that dependent scalar quantization is used is based on a value of a flag signal.

[00349] The method can include wherein the first QP used for the deblocking filter is used for dependent scalar quantization and other processing techniques of the first video block.

[00350] The method can include wherein the first QP is QPc.

[00351] The method can include wherein the first QP is QPc + N, wherein N is an integer.

[00352] The method can include wherein QPc + N is modified from a prior value to fit within a threshold value range.

[00353] The method can include wherein the threshold value range is [Max(QPmin – N, QPmin), Min(QPmax – N, QPmax)].

[00354] The method can include determining, by the processor, that dependent scalar quantization is not used to process a second video block; and performing further processing of the second video block using another deblocking filter, wherein the deblocking filter used for the first video block is stronger or weaker than the another deblocking filter used to process the second video block based on dependent scalar quantization being used for the first video block.

[00355] The method can include wherein the deblocking filter is selected by an encoder, the method further comprises: signaling to a decoder that dependent scalar quantization is enabled for the first video block.

[00356] The method can include wherein smaller or larger threshold values Tc and β are used by the encoder and the decoder based on the use of dependent scalar quantization.

[00357] The method can include wherein the first QP is QPc + N, and additional entries are used for Tc' and β' tables for QPmax + N.

[00358] The method can include wherein the first QP is QPc + N and Tc' and β' table is same as for QPmax + N based on QPc + N being clipped to be within a threshold value range.

[00359] The method can include wherein the Tc' table is extended as: tc'[66] = 50 and tc'[67] = 52.

[00360] The method can include wherein the Tc' and β' table is extended as: tc'[66] = 49 and tc'[67] = 50.

[00361] The method can include wherein the Tc' table is extended as: tc'[66] = 49 and tc'[67] = 51.

[00362] The method can include wherein the Tc' table is extended as: tc'[66] = 48 and tc'[67] = 50.

[00363] The method can include wherein the Tc' table is extended as: tc'[66] = 50 and tc'[67] = 51.

[00364] The method can include wherein the β' table is extended as: β'[64] = 90 and β'[65] = 92.

[00365] The method can include wherein the β' table is extended as: β'[64] = 89 and β'[65] = 90.

[00366] The method can include wherein the β' table is extended as: β'[64] = 89 and β'[65] = 91.

[00367] The method can include wherein the β' table is extended as: β'[64] = 88 and β'[65] = 90.

[00368] The method can include wherein the β' table is extended as: β'[64] = 90 and β'[65] = 91.

[00369] The method can include wherein initialization of context-based adaptive binary arithmetic coding (CABAC) is based on the first QP being QPc + N and dependent scalar quantization being used to process the first video block.

[00370] The method can include wherein the determination that dependent scalar quantization is used to process the first video block is signaled with semantics based on the use of the dependent scalar quantization.

[00371] The method can include wherein the first QP is indicated in a picture parameter set or a picture header.

[00372] The method can include wherein the picture parameter set or the picture header indicate init_qp_minus26 plus 26 that specifies an initial value of a SliceQPy for a slice referring to a PPS or an initial value of quantization parameters of tiles referred to in the PPS or the picture header based on dependent scalar quantization being off.

[00373] The method can include wherein the picture parameter set or the picture header indicate init_qp_minus26 plus 27 that specifies an initial value of a SliceQPy for a slice referring to a PPS or an initial value of quantization parameters of tiles referred to in the PPS or the picture header based on dependent scalar quantization being used.

[00374] The method can include wherein the first QP is indicated in a slice header, a tile header, or a tile groups header.

[00375] The method can include wherein the slice header, the tile header, or the tile groups header indicate slice_qp_delta specifies an initial value of a QpY for coding blocks in slice, tile. Or tile groups until modified by a value of CuQpDeltaVal in a coding unit layer, wherein an initial value of QpY is SliceQpY, wherein SliceQpY = 26 + init_qp_minus26 + slice_qp_delta based on dependent scalar quantization being off.

[00376] The method can include wherein the slice header, the tile header, or the tile groups header indicate slice_qp_delta specifies an initial value of a QpY for coding blocks in slice, tile. Or tile

groups until modified by a value of CuQpDeltaVal in a coding unit layer, wherein an initial value of QpY is SliceQpY, wherein SliceQpY = 26 + init_qp_minus26 + slice_qp_delta based on dependent scalar quantization being used.

[00377] The method can include wherein the method is applied based on being signaled in a SPS, a PPS, a VPS, a sequence header, a picture header, a slice header, a tile group header, or a group of coding tree units (CTUs).

[00378] FIG. 11 is a flowchart for a video processing method 1100 of processing a video. The method 1100 includes determining (1105) one or multiple deblocking filter parameters to be used in a deblocking filter process of a current video block based on whether dependent scalar quantization is used to process the current video block, wherein a set of admissible reconstruction value for a transform coefficient corresponding to the dependent scalar quantization depends on at least one transform coefficient level that precedes a current transform coefficient level; and performing (1110) the deblocking filter process on the current video block in accordance with the one or multiple deblocking filter parameter.

[00379] The method can include that wherein the determining one or multiple deblocking filter parameters to be used in a deblocking filter process of a current video block further comprises: determining the one or multiple deblocking filter parameters corresponding to a weaker deblocking filter in case that the dependent scalar quantization is used for the current video block; or determining the one or multiple deblocking filter parameters corresponding to a stronger deblocking filter in case that the dependent scalar quantization is used for the current video block.

[00380] The method can include that wherein the stronger deblocking filter modifies more pixels, and the weaker deblocking filter modifies less pixels.

[00381] The method can include that wherein the determining one or multiple deblocking filter parameters to be used in a deblocking filter process of a current video block further comprises: selecting smaller threshold values Tc and β in case that the dependent scalar quantization is used for the current video block; or selecting larger threshold values Tc and β in case that the dependent scalar quantization is used for the current video block.

[00382] The method can include that wherein the determining one or multiple deblocking filter parameters to be used in a deblocking filter process of a current video block further comprises: determining a quantization parameter included in the one or multiple deblocking filter

parameters based on whether dependent scalar quantization is used to process the current video block.

[00383] The method can include wherein the quantization parameter used for the deblocking filter process is set equal to QPc + N in case that the dependent scalar quantization is used for the current video block, wherein QPc is a signaled quantization parameter of the current video block, QPc + N is a quantization parameter used for the dependent scalar quantization, N is an integer and N>=1.

[00384] The method can include wherein at least one additional entry is set in a mapping table in the case that the dependent scalar quantization is used for the current video block, wherein the mapping table indicates mapping relationships between quantization parameters and threshold values β', or indicates mapping relationships between quantization parameters and threshold values Tc'.

[00385] The method can include wherein the mapping table is extended according to any of the following options: tc'[66] = 50 and tc'[67] = 52, tc'[66] = 49 and tc'[67] = 50, tc'[66] = 49 and tc'[67] = 51, tc'[66] = 48 and tc'[67] = 50, tc'[66] = 50 and tc'[67] = 51.

[00386] The method can include wherein the mapping table is extended according to any of the following options: β'[64] = 90 and β'[65] = 92, β'[64] = 89 and β'[65] = 90, β'[64] = 89 and β'[65] = 91, β'[64] = 88 and β'[65] = 90, β'[64] = 90 and β'[65] = 91.

[00387] The method can include wherein the QPc + N is clipped to be within a range [QPmin, QPmax] in case that it greater than QPmax or less than QPmin, wherein QPmin and QPmax are respectively the minimum and the maximum allowable quantization parameter.

[00388] FIG. 13 is a flowchart for a video processing method 1300. The method 1300 includes determining (1305) whether to apply deblocking filter process based on whether dependent scalar quantization is used to process a current video block, wherein a set of admissible reconstruction value for a transform coefficient corresponding to the dependent scalar quantization depends on at least one transform coefficient level that precedes a current transform coefficient level; and performing (1310) the deblocking filter process on the current video block base on the determination of applying the deblocking filter process.

[00389] FIG. 12 is a flowchart for a video processing method 1200 of processing a video. The method 1200 includes determining (1205) a quantization parameter to be used in a dependent scalar quantization of a current video block in case that the dependent scalar

quantization is enabled for the current video block, wherein a set of admissible reconstruction value for a transform coefficient corresponding to the dependent scalar quantization depends on at least one transform coefficient level that precedes a current transform coefficient level; and performing (1210) the dependent scalar quantization on the current video block in accordance with the determined quantization parameter, wherein the determined quantization parameter is also applied to a video process different from the dependent scalar quantization using a quantization parameter as an input parameter of the current video block.

[00390] The method can include wherein the video processing different from the dependent scalar quantization includes a deblocking filtering process.

[00391] The method can include wherein the determined quantization parameter is QPc in case that the dependent scalar quantization is enabled for the current video block, wherein QPc is a signaled quantization parameter of the current video block.

[00392] The method can include wherein the determined quantization parameter is QPc+N in case that the dependent scalar quantization is enabled for the current video block, wherein QPc is a signaled quantization parameter of the current video block, and N is an integer.

[00393] The method can further include clipping QPc + N to a threshold value range before using it.

[00394] The method can include wherein the threshold value range is[QPmin, QPmax], wherein QPmin and QPmax are respectively allowed minimum and maximum quantization parameters.

[00395] The method can include wherein an allowed QP_C range for the dependent scalar quantization is [QPmin – N, QPmax – N] in case that the dependent scalar quantization is enabled for the current video block, wherein QPmin and QPmax are the allowed minimum value and maximum value of QPc in case that the dependent scalar quantization is not enabled for the current video block , respectively.

[00396] The method can include wherein an allowed QP_C range for the dependent scalar quantization is [Max(QPmin – N, QPmin), Min(QPmax – N, QPmax)] in case that the dependent scalar quantization is enabled for the current video block, wherein QPmin and QPmax are the allowed minimum value and maximum value of QPc in case that the dependent scalar quantization is not enabled for the current video block, respectively.

[00397] The method can include wherein initialization of context-based adaptive binary arithmetic coding (CABAC) is based on the QPc + N in case that the dependent scalar quantization is enabled.

[00398] The method can include wherein high-level quantization parameters are assigned with different semantics based on whether the dependent scalar quantization is enabled.

[00399] The method can include wherein the quantization parameter is signaled in a picture parameter set or a picture header by means of a first parameter.

[00400] The method can include wherein the first parameter is init_qp_minus26, and init_qp_minus26 plus 26 specifies an initial quantization parameter value SliceQPy for a slice referring to the picture parameter set or an initial value of quantization parameters of tiles referring to the picture parameter set or the picture header, in case that the dependent scalar quantization is disabled.

[00401] The method can include wherein the first parameter is init_qp_minus26, and init_qp_minus26 plus 27 specifies an initial quantization parameter value SliceQPy for a slice referring to the picture parameter set or an initial value of quantization parameters of tiles referring to the picture parameter set or the picture header, in case that the dependent scalar quantization is enabled.

[00402] The method can include wherein the quantization parameter is signaled in a slice header, a tile header, or a tile groups header by means of a second parameter.

[00403] The method can include wherein the second parameter is slice_qp_delta, and slice_qp_delta is used to derive an initial quantization parameter value QpY for coding blocks in a slice, tile or tile groups until modified by a value of CuQpDeltaVal in a coding unit layer, wherein an initial value of QpY is set equal to SliceQpY, and SliceQpY = 26 + init_qp_minus26 + slice_qp_delta, in case that the dependent scalar quantization is disabled.

[00404] The method can include wherein the second parameter is slice_qp_delta, and slice_qp_delta is used to derive an initial quantization parameter value QpY for coding blocks in a slice, tile or tile groups until modified by a value of CuQpDeltaVal in a coding unit layer, wherein an initial value of QpY is set equal to SliceQpY, and SliceQpY = 26 + init_qp_minus26 + slice_qp_delta + 1, in case that the dependent scalar quantization is enabled.

[00405] The method can be applied in case of being signaled in a SPS, a PPS, a VPS, a sequence header, a picture header, a slice header, a tile group header, or a group of coding tree units (CTUs).

[00406] FIG. 14 is a flowchart for a video processing method 1400 of processing a video. The method 1400 includes determining (1405) a quantization parameter to be used in a dependent scalar dequantization of a current video block in case that the dependent scalar dequantization is enabled for the current video block, wherein a set of admissible reconstruction value for a transform coefficient corresponding to the dependent scalar dequantization depends on at least one transform coefficient level that precedes a current transform coefficient level; performing (1410) the dependent scalar dequantization on the current video block in accordance with the determined quantization parameter, wherein the determined quantization parameter is also applied to a video process different from the dependent scalar dequantization using a quantization parameter as an input parameter of the current video block.

[00407] It will be appreciated that the disclosed techniques may be embodied in video encoders or decoders to improve compression efficiency when the coding units being compressed have shaped that are significantly different than the traditional square shaped blocks or rectangular blocks that are half-square shaped. For example, new coding tools that use long or tall coding units such as 4x32 or 32x4 sized units may benefit from the disclosed techniques.

[00408] It will be appreciated that the disclosed techniques may be embodied in a video system comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to implement the above disclosed method.

[00409] The disclosed and other solutions, examples, embodiments, modules and the functional operations described in this document can be implemented in digital electronic circuitry, or in computer software, firmware, or hardware, including the structures disclosed in this document and their structural equivalents, or in combinations of one or more of them. The disclosed and other embodiments can be implemented as one or more computer program products, i.e., one or more modules of computer program instructions encoded on a computer readable medium for execution by, or to control the operation of, data processing apparatus. The computer readable medium can be a machine-readable storage device, a machine-readable storage substrate, a memory device, a composition of matter effecting a machine-readable

propagated signal, or a combination of one or more them. The term "data processing apparatus" encompasses all apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can include, in addition to hardware, code that creates an execution environment for the computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them. A propagated signal is an artificially generated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal, that is generated to encode information for transmission to suitable receiver apparatus.

[00410] A computer program (also known as a program, software, software application, script, or code) can be written in any form of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program does not necessarily correspond to a file in a file system. A program can be stored in a portion of a file that holds other programs or data (e.g., one or more scripts stored in a markup language document), in a single file dedicated to the program in question, or in multiple coordinated files (e.g., files that store one or more modules, sub programs, or portions of code). A computer program can be deployed to be executed on one computer or on multiple computers that are located at one site or distributed across multiple sites and interconnected by a communication network.

[00411] The processes and logic flows described in this document can be performed by one or more programmable processors executing one or more computer programs to perform functions by operating on input data and generating output. The processes and logic flows can also be performed by, and apparatus can also be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit).

[00412] Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read only memory or a random-access memory or both. The essential elements of a computer are a processor for performing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to

receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices. Computer readable media suitable for storing computer program instructions and data include all forms of non-volatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto optical disks; and CD ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

[00413] While this patent document contains many specifics, these should not be construed as limitations on the scope of any subject matter or of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular techniques. Certain features that are described in this patent document in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

[00414] Similarly, while operations are depicted in the drawings in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. Moreover, the separation of various system components in the embodiments described in this patent document should not be understood as requiring such separation in all embodiments.

[00415] Only a few implementations and examples are described and other implementations, enhancements and variations can be made based on what is described and illustrated in this patent document.

CLAIMS

1. A video processing method, comprising:

determining one or multiple deblocking filter parameters to be used in a deblocking filter process of a current video block based on whether dependent scalar quantization is used to process the current video block, wherein a set of admissible reconstruction value for a transform coefficient corresponding to the dependent scalar quantization depends on at least one transform coefficient level that precedes a current transform coefficient level; and

performing the deblocking filter process on the current video block in accordance with the one or multiple deblocking filter parameter.

2. The video processing method of claim 1, wherein the determining one or multiple deblocking filter parameters to be used in a deblocking filter process of a current video block further comprises:

determining the one or multiple deblocking filter parameters corresponding to a weaker deblocking filter in case that the dependent scalar quantization is used for the current video block; or

determining the one or multiple deblocking filter parameters corresponding to a stronger deblocking filter in case that the dependent scalar quantization is used for the current video block.

3. The video processing method of claim 2, wherein the stronger deblocking filter modifies more pixels, and the weaker deblocking filter modifies less pixels.

4. The video processing method of claim 1, wherein the determining one or multiple deblocking filter parameters to be used in a deblocking filter process of a current video block further comprises:

selecting smaller threshold values Tc and β in case that the dependent scalar quantization is used for the current video block; or

selecting larger threshold values Tc and β in case that the dependent scalar quantization is used for the current video block.

5. The video processing method of claim 1, wherein the determining one or multiple deblocking filter parameters to be used in a deblocking filter process of a current video block further comprises:

determining a quantization parameter included in the one or multiple deblocking filter parameters based on whether dependent scalar quantization is used to process the current video block.

6. The video processing method of claim 5, wherein the quantization parameter used for the deblocking filter process is set equal to QPc + N in case that the dependent scalar quantization is used for the current video block, wherein QPc is a signaled quantization parameter of the current video block, QPc + N is a quantization parameter used for the dependent scalar quantization, N is an integer and N>=1.

7. The video processing method of claim 1 or 6, wherein at least one additional entry is set in a mapping table in the case that the dependent scalar quantization is used for the current video block, wherein the mapping table indicates mapping relationships between quantization parameters and threshold values β', or indicates mapping relationships between quantization parameters and threshold values Tc'.

8. The video processing method of claim 7, wherein the mapping table is extended according to any of the following options: tc'[66] = 50 and tc'[67] = 52, tc'[66] = 49 and tc'[67] = 50, tc'[66] = 49 and tc'[67] = 51, tc'[66] = 48 and tc'[67] = 50, tc'[66] = 50 and tc'[67] = 51.

9. The video processing method of claim 7, wherein the mapping table is extended according to any of the following options: β'[64] = 90 and β'[65] = 92, β'[64] = 89 and β'[65] = 90, β'[64] = 89 and β'[65] = 91, β'[64] = 88 and β'[65] = 90, β'[64] = 90 and β'[65] = 91.

10. The video processing method of claim 6, wherein the QPc + N is clipped to be within a range [QPmin, QPmax] in case that it greater than QPmax or less than QPmin, wherein QPmin and QPmax are respectively the minimum and the maximum allowable quantization parameter.

11. A video processing method, comprising:

determining whether to apply deblocking filter process based on whether dependent scalar quantization is used to process a current video block, wherein a set of admissible reconstruction value for a transform coefficient corresponding to the dependent scalar quantization depends on at least one transform coefficient level that precedes a current transform coefficient level; and

performing the deblocking filter process on the current video block base on the determination of applying the deblocking filter process.

12. An apparatus in a video system comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to implement the method in any one of claims 1 to 11.

13. A computer program product stored on a non-transitory computer readable media, the computer program product including program code for carrying out the method in any one of claims 1 to 11.

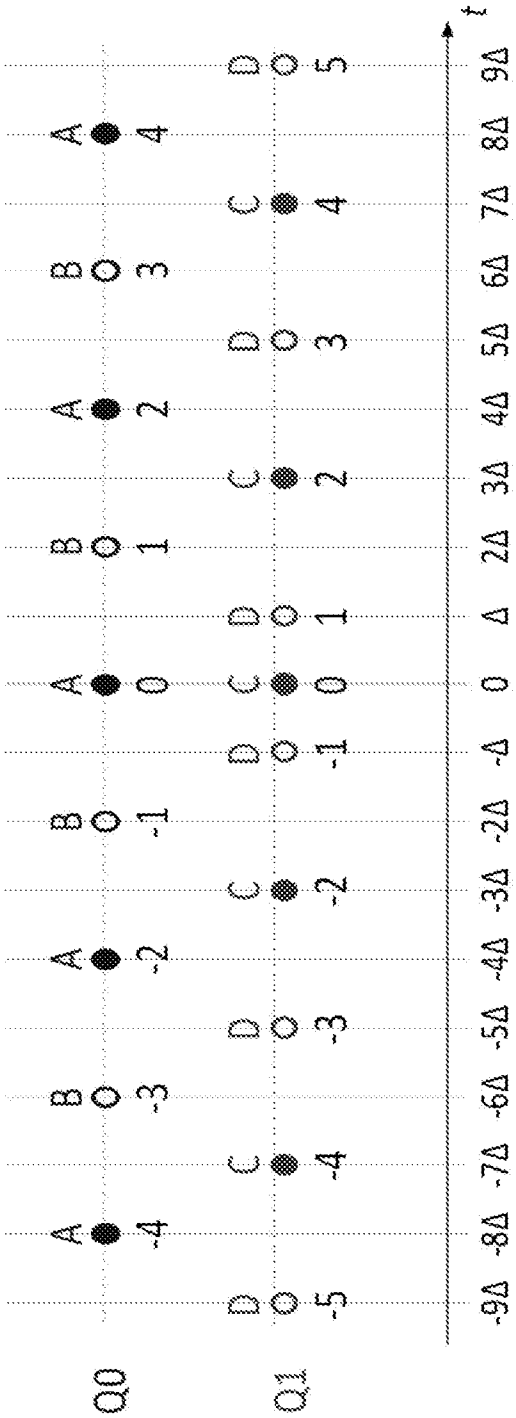
Q0
Q1
t
A
B
C
D
-9Δ -8Δ -7Δ -6Δ -5Δ -4Δ -3Δ -2Δ -Δ 0 Δ 2Δ 3Δ 4Δ 5Δ 6Δ 7Δ 8Δ 9Δ
-5 -4 -3 -2 -1 0 1 2 3 4 5

FIG. 1

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

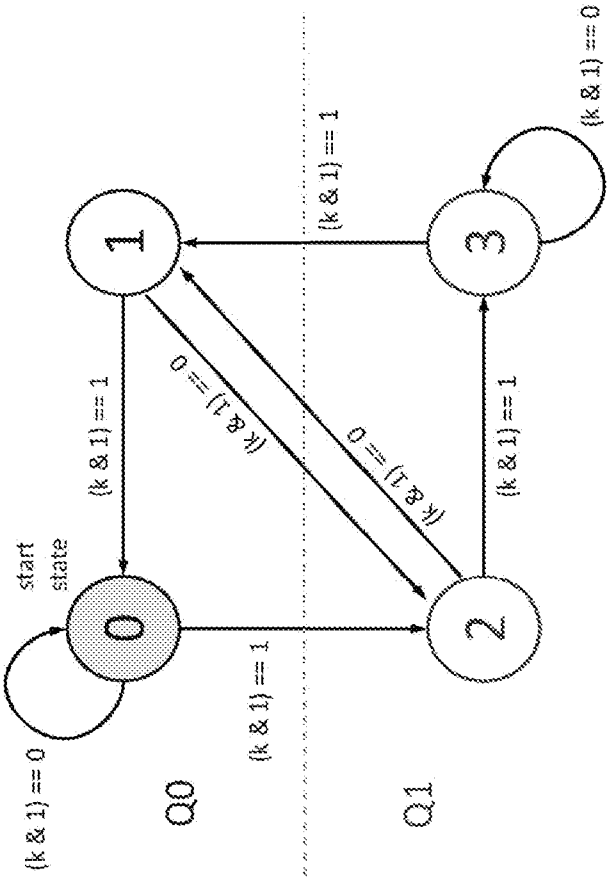

FIG. 2

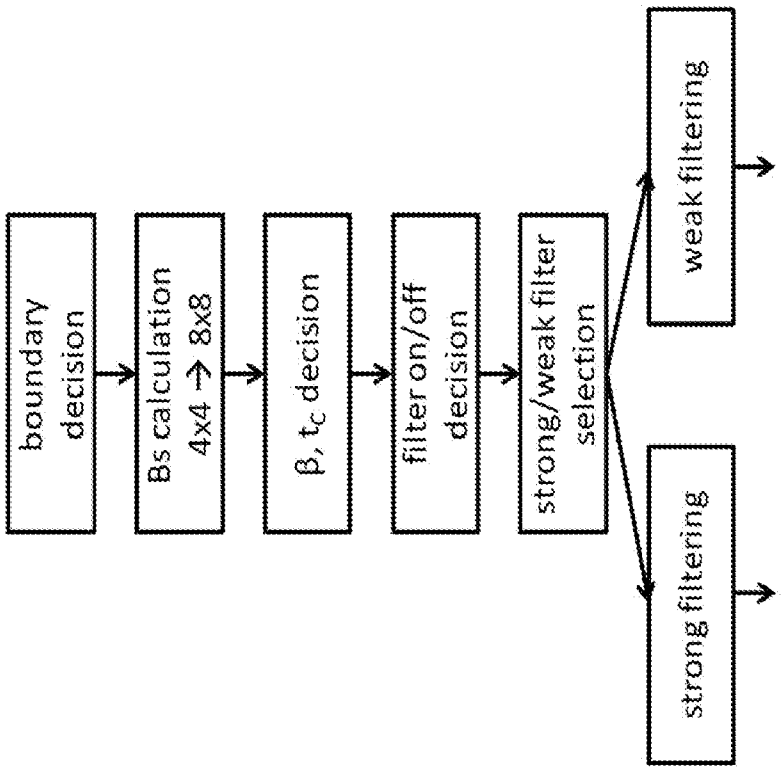
boundary decision
Bs calculation 4x4 → 8x8
β, t_C decision
filter on/off decision
strong/weak filter selection
weak filtering
strong filtering

FIG. 3

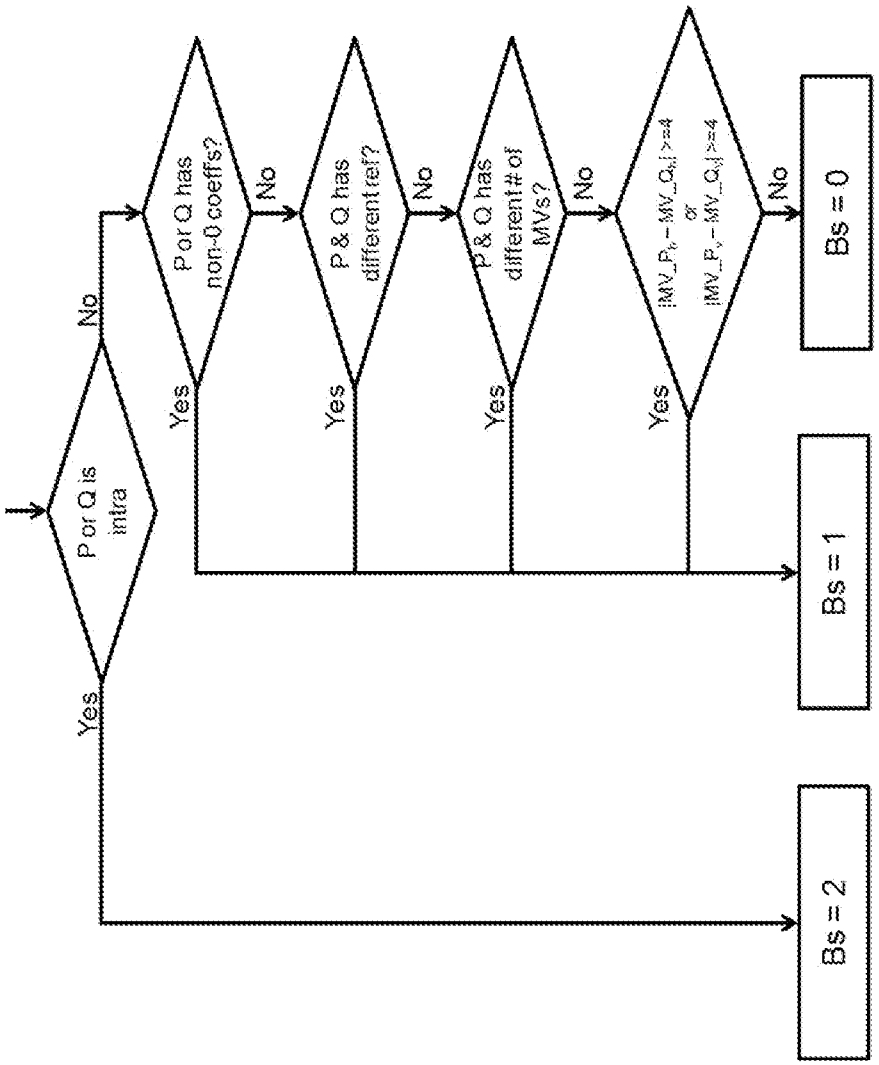
P or Q is intra
No
Yes
P or Q has non-0 coeffs?
No
Yes
P & Q has different ref?
No
Yes
P & Q has different # of MVs?
No
Yes
|MV_P_h − MV_Q_h| >=4 or |MV_P_v − MV_Q_v| >=4
No
Yes
Bs = 0
Bs = 1
Bs = 2

FIG. 4

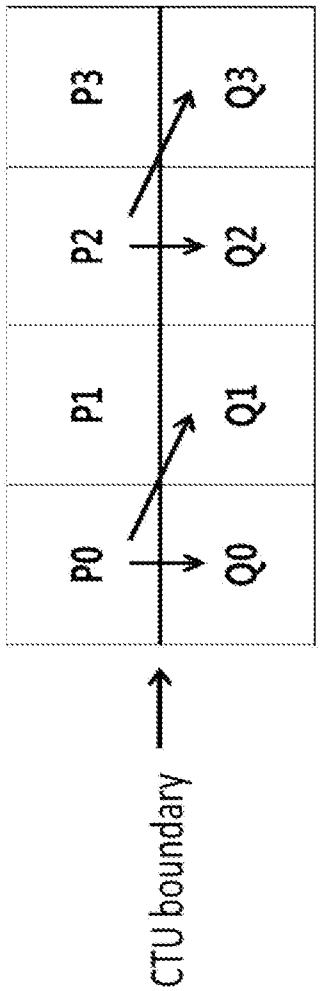

FIG. 5

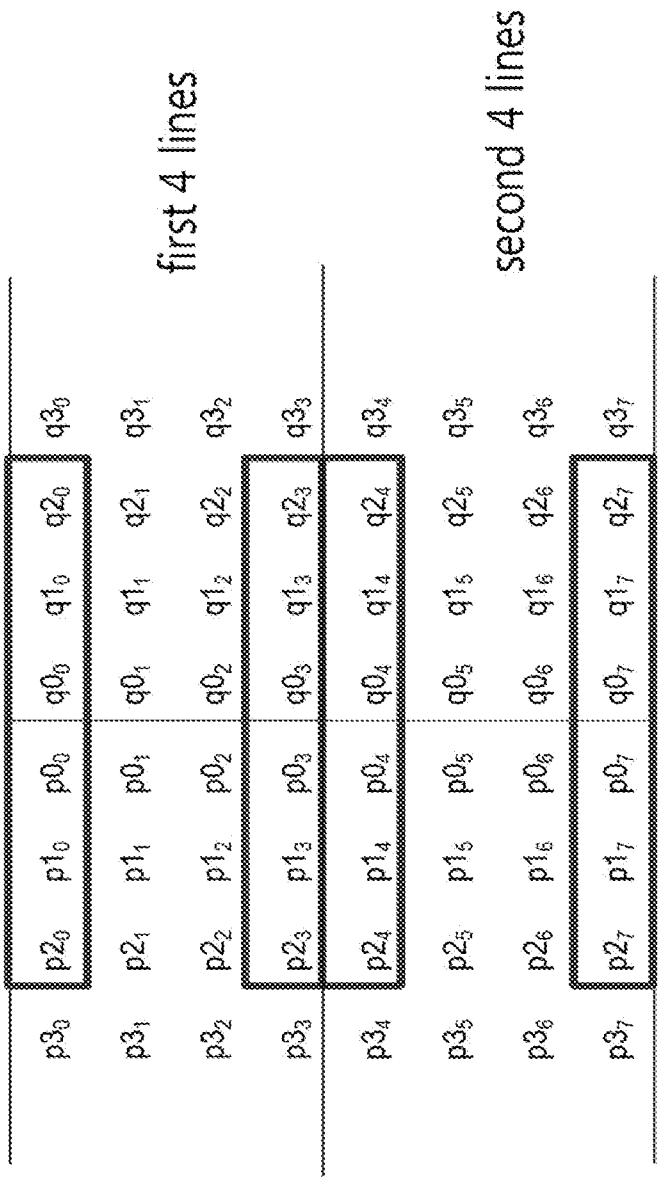
first 4 lines
second 4 lines
p3$_0$ p2$_0$ p1$_0$ p0$_0$ q0$_0$ q1$_0$ q2$_0$ q3$_0$
p3$_1$ p2$_1$ p1$_1$ p0$_1$ q0$_1$ q1$_1$ q2$_1$ q3$_1$
p3$_2$ p2$_2$ p1$_2$ p0$_2$ q0$_2$ q1$_2$ q2$_2$ q3$_2$
p3$_3$ p2$_3$ p1$_3$ p0$_3$ q0$_3$ q1$_3$ q2$_3$ q3$_3$
p3$_4$ p2$_4$ p1$_4$ p0$_4$ q0$_4$ q1$_4$ q2$_4$ q3$_4$
p3$_5$ p2$_5$ p1$_5$ p0$_5$ q0$_5$ q1$_5$ q2$_5$ q3$_5$
p3$_6$ p2$_6$ p1$_6$ p0$_6$ q0$_6$ q1$_6$ q2$_6$ q3$_6$
p3$_7$ p2$_7$ p1$_7$ p0$_7$ q0$_7$ q1$_7$ q2$_7$ q3$_7$

FIG. 6

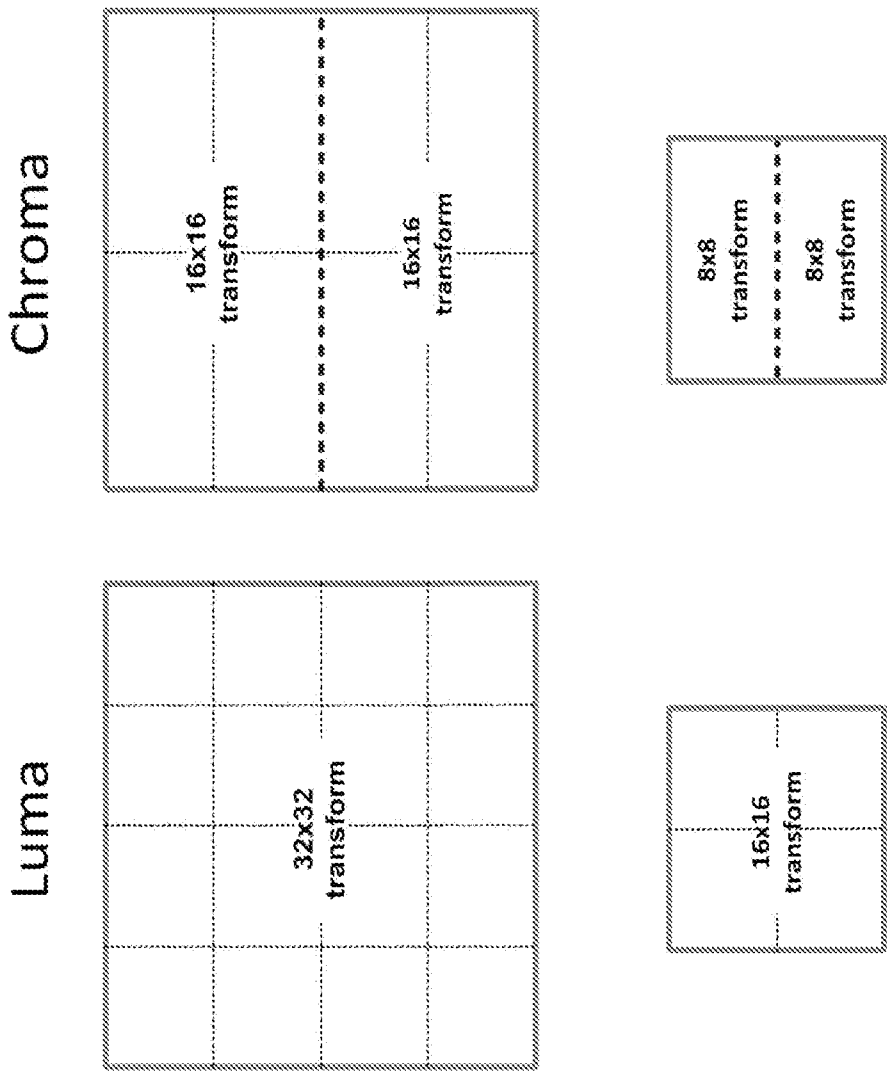
Luma
32x32
transform
16x16
transform
Chroma
16x16
transform
16x16
transform
8x8
transform
8x8
transform

FIG. 7

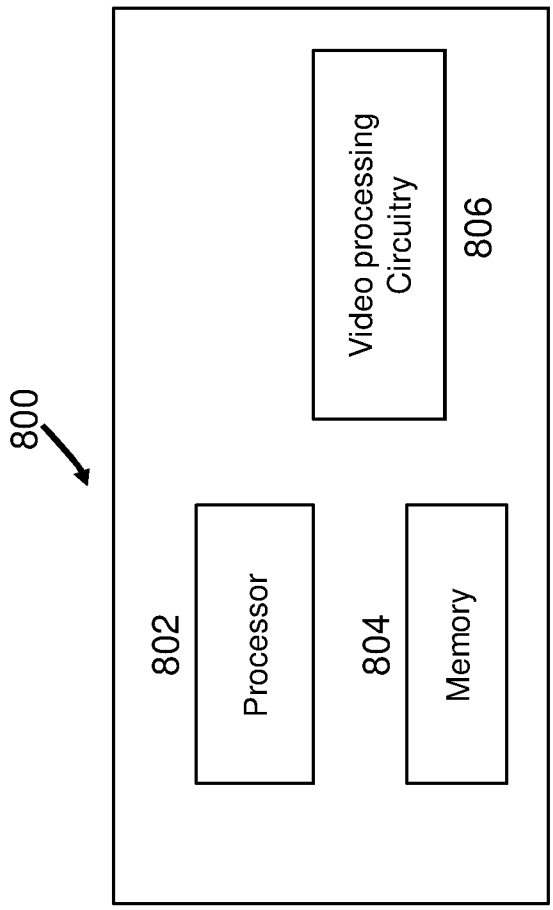
800
802
Processor
804
Memory
806
Video processing Circuitry

FIG. 8

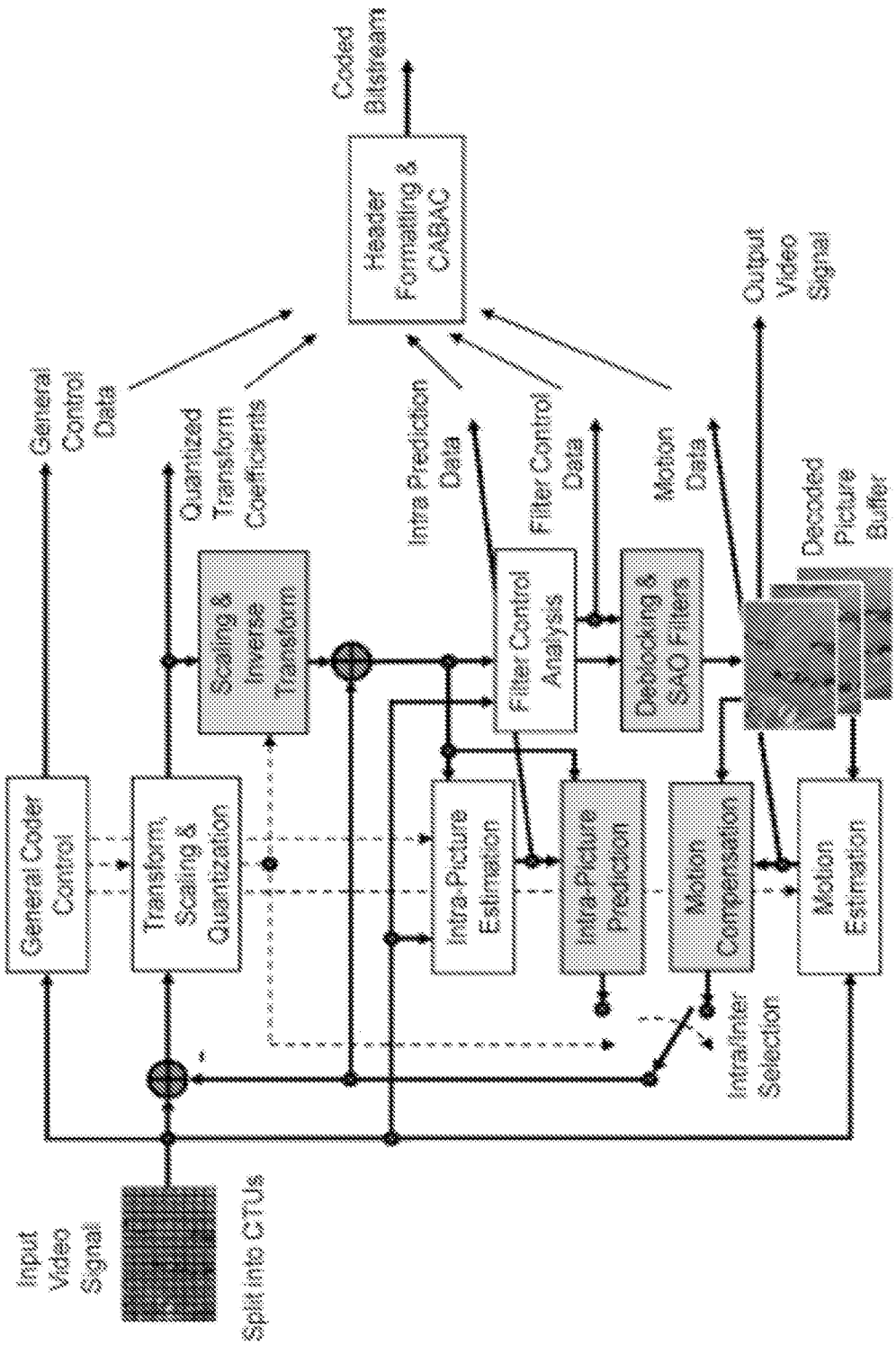
Input Video Signal
Split into CTUs
General Coder Control
Transform, Scaling & Quantization
Scaling & Inverse Transform
Intra-Picture Estimation
Intra-Picture Prediction
Motion Compensation
Motion Estimation
Filter Control Analysis
Deblocking & SAO Filters
Decoded Picture Buffer
Header Formatting & CABAC
Coded Bitstream
Output Video Signal
General Control Data
Quantized Transform Coefficients
Intra Prediction Data
Filter Control Data
Motion Data
Intra/Inter Selection

FIG. 9

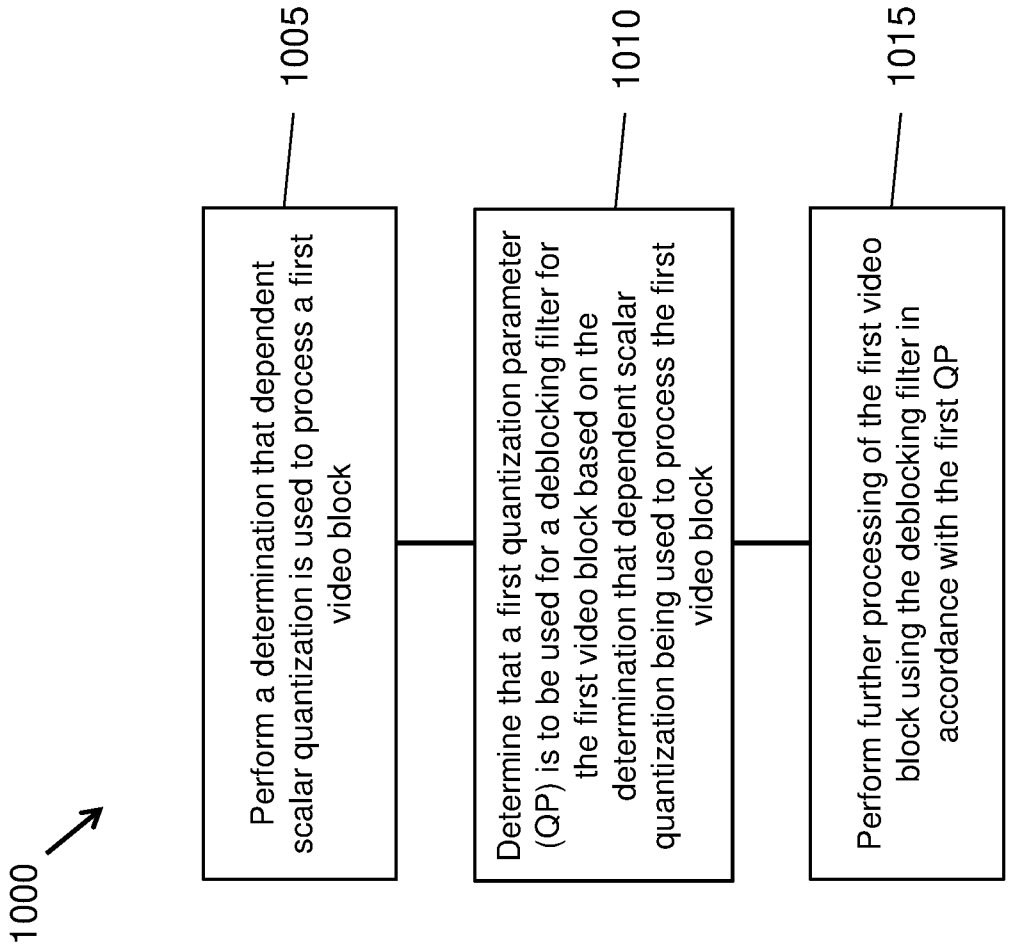
1000
1005
Perform a determination that dependent scalar quantization is used to process a first video block
1010
Determine that a first quantization parameter (QP) is to be used for a deblocking filter for the first video block based on the determination that dependent scalar quantization being used to process the first video block
1015
Perform further processing of the first video block using the deblocking filter in accordance with the first QP

FIG. 10

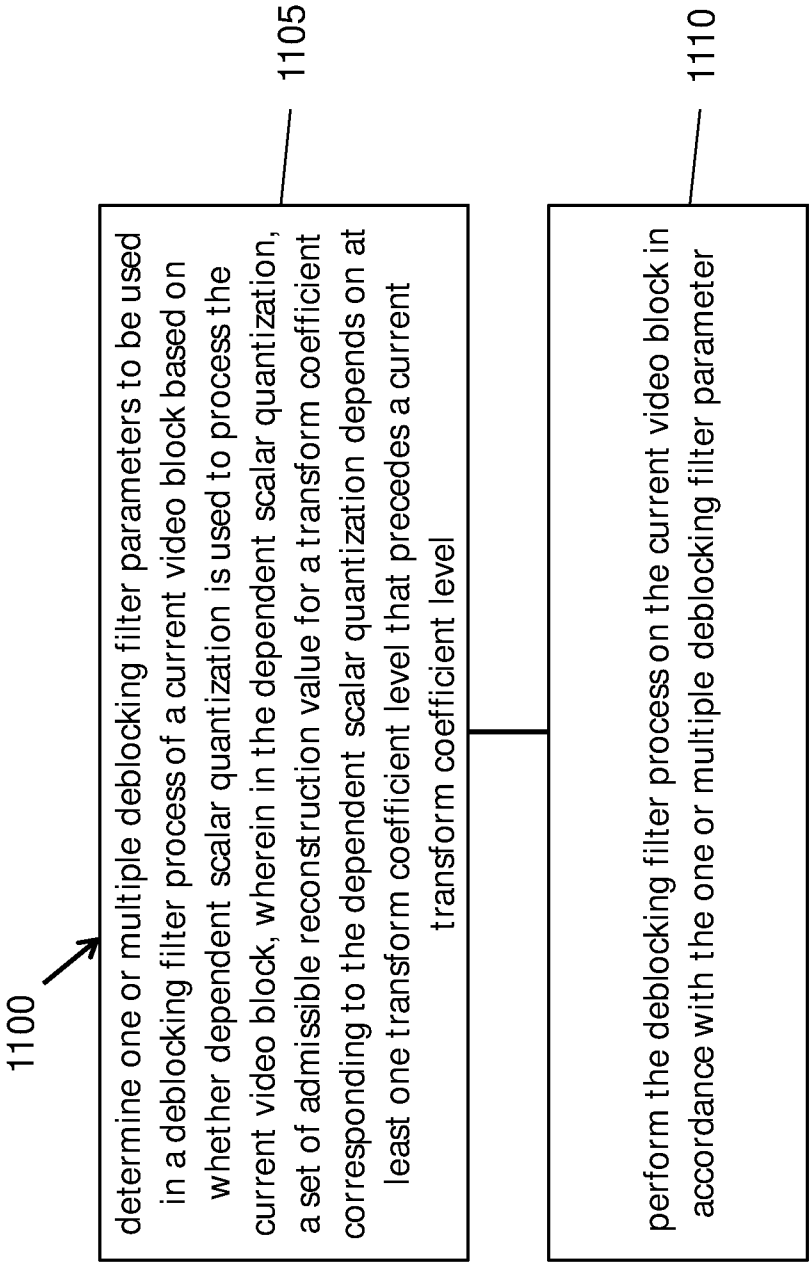
1100
determine one or multiple deblocking filter parameters to be used in a deblocking filter process of a current video block based on whether dependent scalar quantization is used to process the current video block, wherein in the dependent scalar quantization, a set of admissible reconstruction value for a transform coefficient corresponding to the dependent scalar quantization depends on at least one transform coefficient level that precedes a current transform coefficient level
1105
perform the deblocking filter process on the current video block in accordance with the one or multiple deblocking filter parameter
1110

FIG. 11

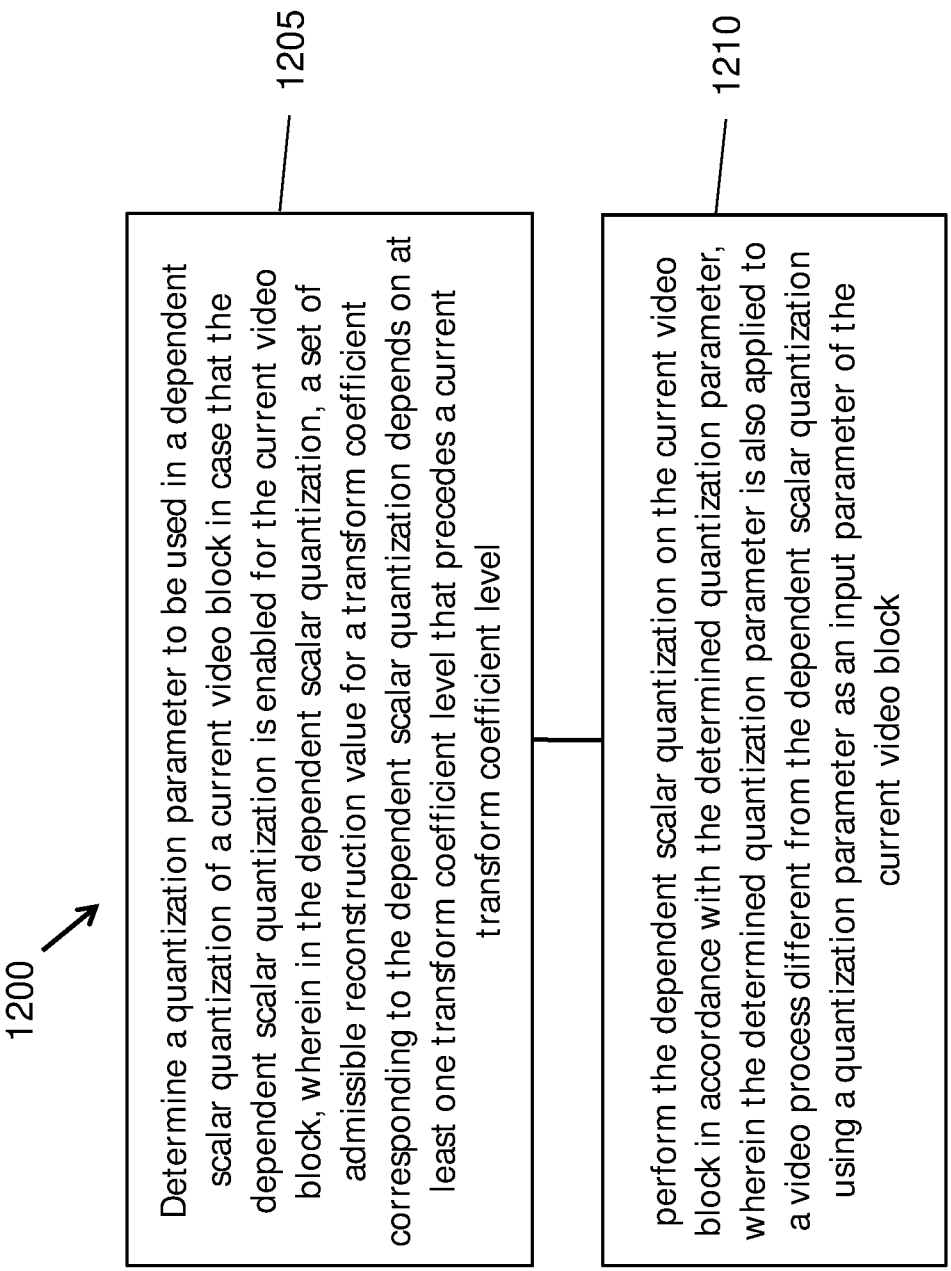
1200
1205
Determine a quantization parameter to be used in a dependent scalar quantization of a current video block in case that the dependent scalar quantization is enabled for the current video block, wherein in the dependent scalar quantization, a set of admissible reconstruction value for a transform coefficient corresponding to the dependent scalar quantization depends on at least one transform coefficient level that precedes a current transform coefficient level
1210
perform the dependent scalar quantization on the current video block in accordance with the determined quantization parameter, wherein the determined quantization parameter is also applied to a video process different from the dependent scalar quantization using a quantization parameter as an input parameter of the current video block

FIG. 12

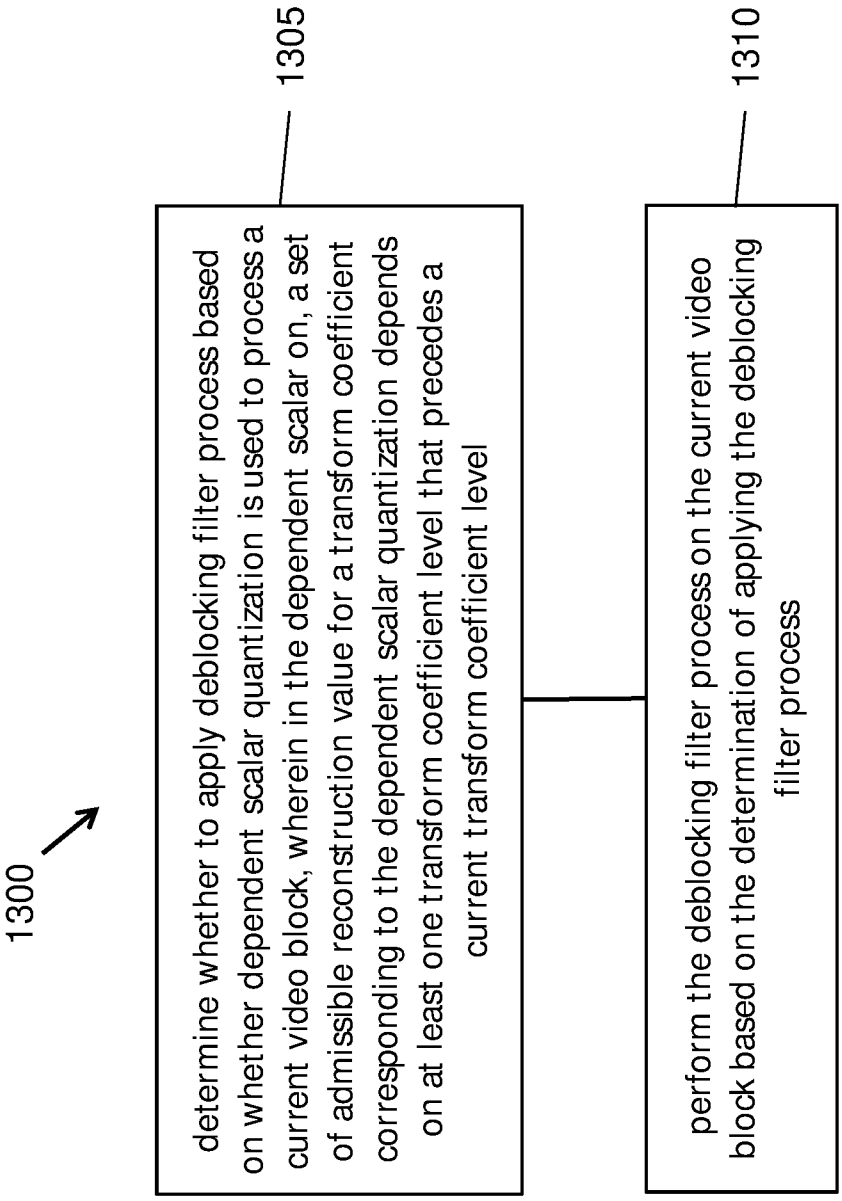
1300
1305
determine whether to apply deblocking filter process based on whether dependent scalar quantization is used to process a current video block, wherein in the dependent scalar on, a set of admissible reconstruction value for a transform coefficient corresponding to the dependent scalar quantization depends on at least one transform coefficient level that precedes a current transform coefficient level
1310
perform the deblocking filter process on the current video block based on the determination of applying the deblocking filter process

FIG. 13

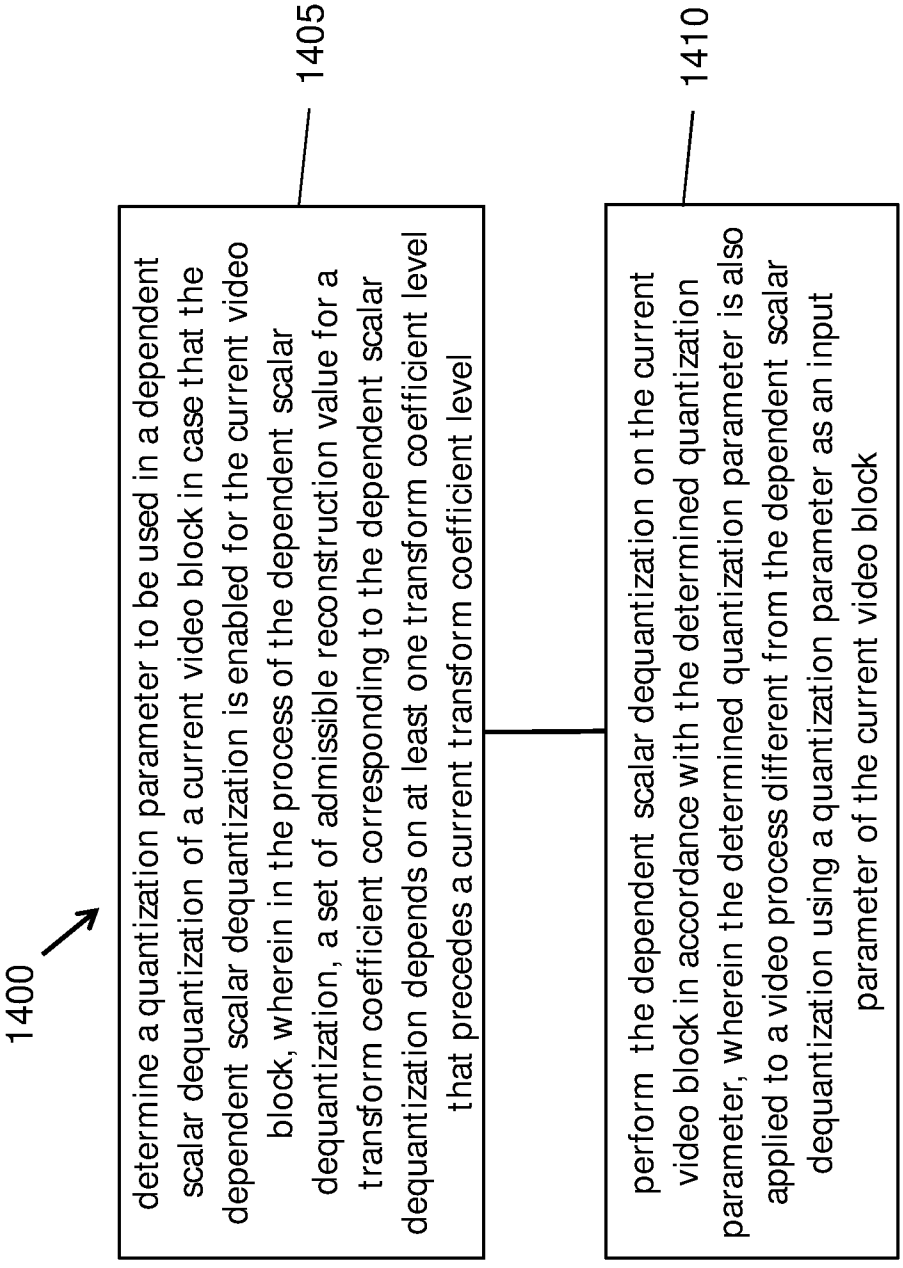
1400
determine a quantization parameter to be used in a dependent scalar dequantization of a current video block in case that the dependent scalar dequantization is enabled for the current video block, wherein in the process of the dependent scalar dequantization, a set of admissible reconstruction value for a transform coefficient corresponding to the dependent scalar dequantization depends on at least one transform coefficient level that precedes a current transform coefficient level
1405
perform the dependent scalar dequantization on the current video block in accordance with the determined quantization parameter, wherein the determined quantization parameter is also applied to a video process different from the dependent scalar dequantization using a quantization parameter as an input parameter of the current video block
1410

FIG. 14

INTERNATIONAL SEARCH REPORT

International application No
PCT/IB2019/059342

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04N19/117 H04N19/176 H04N19/82
ADD.

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

B. FIELDS SEARCHED

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

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

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	SCHWARZ (FRAUNHOFER) H ET AL: "CE7: Transform coefficient coding and dependent quantization (Tests 7.1.2, 7.2.1)", 11. JVET MEETING; 20180711 - 20180718; LJUBLJANA; (THE JOINT VIDEO EXPLORATION TEAM OF ISO/IEC JTC1/SC29/WG11 AND ITU-T SG.16), , no. JVET-K0071 11 July 2018 (2018-07-11), XP030199394, Retrieved from the Internet: URL:http://phenix.int-evry.fr/jvet/doc_end_user/documents/11_Ljubljana/wg11/JVET-K0071-v2.zip JVET-K0071.doc [retrieved on 2018-07-11] sections 2, 3; figure 2 ----- -/--	1-13

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents :

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
7 January 2020	23/01/2020

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

INTERNATIONAL SEARCH REPORT

International application No
PCT/IB2019/059342

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	VAN DER AUWERA G ET AL: "Varying QP Deblocking", 98. MPEG MEETING; 28-11-2011 - 2-12-2011; GENEVA; (MOTION PICTURE EXPERT GROUP OR ISO/IEC JTC1/SC29/WG11),, no. m21946, 18 November 2011 (2011-11-18), XP030050509, sections 1, 2 -----	1-13