(12) STANDARD PATENT (11) Application No. **AU 2014374141 B2**
(19) AUSTRALIAN PATENT OFFICE

(54) Title
Innovations in block vector prediction and estimation of reconstructed sample values within an overlap area

(51) International Patent Classification(s)
H04N 19/105 (2014.01) ***H04N 19/52*** (2014.01)
H04N 19/167 (2014.01) ***H04N 19/593*** (2014.01)
H04N 19/176 (2014.01)

(21) Application No: **2014374141** (22) Date of Filing: **2014.12.22**

(87) WIPO No: **WO15/102975**

(30) Priority Data

(31) Number	(32) Date	(33) Country
14/222,580	**2014.03.21**	**US**
61/923,628	**2014.01.03**	**US**

(43) Publication Date: **2015.07.09**
(44) Accepted Journal Date: **2018.11.08**

(71) Applicant(s)
Microsoft Technology Licensing, LLC

(72) Inventor(s)
Zhu, Lihua;Sullivan, Gary J.;Wu, Yongjun

(74) Agent / Attorney
Davies Collison Cave Pty Ltd, Level 15 1 Nicholson Street, MELBOURNE, VIC, 3000, AU

(56) Related Art
Flynn D et al., "BoG report on Range Extensions topics", 15. JCT-VC MEETING; 23-10-2013 - 1-11-2013; GENEVA; (JOINT COLLABORATIVE TEAM ON VIDEO CODING OF ISO/IEC JTC1/SC29/WG11 AND ITU-T SG.16)
Sole et al., "HEVC Range Extensions Core Experiment 3 (RCE3): Intra block copy refinement," JCTVC-01123, 6 pp. ,23 October 2013- 1 November 2013
Li B et al., "On Intra BC mode", 15. JCT-VC MEETING; 23-10-2013 - 1-11-2013; GENEVA; (JOINT COLLABORATIVE TEAM ON VIDEO CODING OF ISO/IEC JTC1/SC29/WG11 AND ITU-T SG.16)

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
9 July 2015 (09.07.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/102975 A3

(51) International Patent Classification:
H04N 19/105 (2014.01) *H04N 19/593* (2014.01)
H04N 19/52 (2014.01) *H04N 19/167* (2014.01)
H04N 19/176 (2014.01)

(21) International Application Number:
PCT/US2014/071780

(22) International Filing Date:
22 December 2014 (22.12.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/923,628 3 January 2014 (03.01.2014) US
14/222,580 21 March 2014 (21.03.2014) US

(71) Applicant: **MICROSOFT TECHNOLOGY LICENSING, LLC** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(72) Inventors: **ZHU, Lihua**; c/o Microsoft Corporation, LCA - International Patents (8/1172), One Microsoft Way, Redmond, Washington 98052-6399 (US). **SULLIVAN, Gary J.**; c/o Microsoft Corporation, LCA - International Patents (8/1172), One Microsoft Way, Redmond, Washington 98052-6399 (US). **WU, Yongjun**; c/o Microsoft Corporation, LCA - International Patents (8/1172), One Microsoft Way, Redmond, Washington 98052-6399 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, 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).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

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

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

[Continued on next page]

(54) Title: INNOVATIONS IN BLOCK VECTOR PREDICTION AND ESTIMATION OF RECONSTRUCTED SAMPLE VALUES WITHIN AN OVERLAP AREA

Figure 22a

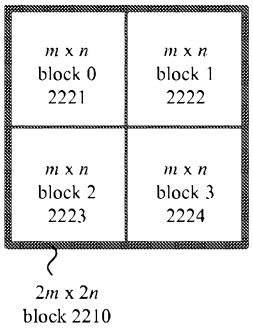
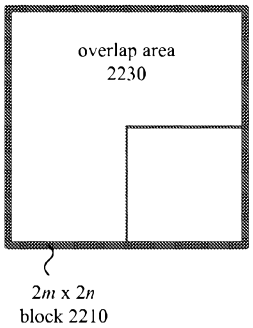


Figure 22b



(57) Abstract: Innovations in encoder-side options for intra block copy ("BC") prediction mode facilitate intra BC prediction that is more effective in terms of rate-distortion performance and/or computational efficiency of encoding. For example, some of the innovations relate to estimation of sample values within an overlap area of a current block during block vector estimation. Other innovations relate to prediction of block vector ("BV") values during encoding or decoding using "ping-pong" approaches.

WO 2015/102975 A3

(88) Date of publication of the international search report:
27 August 2015

INNOVATIONS IN BLOCK VECTOR PREDICTION AND ESTIMATION OF RECONSTRUCTED SAMPLE VALUES WITHIN AN OVERLAP AREA

BACKGROUND

[001] Engineers use compression (also called source coding or source encoding) to reduce the bit rate of digital video. Compression decreases the cost of storing and transmitting video information by converting the information into a lower bit rate form. Decompression (also called decoding) reconstructs a version of the original information from the compressed form. A "codec" is an encoder/decoder system.

[002] Over the last two decades, various video codec standards have been adopted, including the ITU-T H.261, H.262 (MPEG-2 or ISO/IEC 13818-2), H.263 and H.264 (MPEG-4 AVC or ISO/IEC 14496-10) standards, the MPEG-1 (ISO/IEC 11172-2) and MPEG-4 Visual (ISO/IEC 14496-2) standards, and the SMPTE 421M (VC-1) standard. More recently, the HEVC standard (ITU-T H.265 or ISO/IEC 23008-2) has been approved. Extensions to the HEVC standard (*e.g.*, for scalable video coding/decoding, for coding/decoding of video with higher fidelity in terms of sample bit depth or chroma sampling rate, or for multi-view coding/decoding) are currently under development. A video codec standard typically defines options for the syntax of an encoded video bitstream, detailing parameters in the bitstream when particular features are used in encoding and decoding. In many cases, a video codec standard also provides details about the decoding operations a decoder should perform to achieve conforming results in decoding. Aside from codec standards, various proprietary codec formats define other options for the syntax of an encoded video bitstream and corresponding decoding operations.

[003] Intra block copy ("BC") is a prediction mode under development for HEVC extensions. For intra BC prediction mode, the sample values of a current block of a picture are predicted using previously reconstructed sample values in the same picture. A block vector ("BV") indicates a displacement from the current block to a region of the picture that includes the previously reconstructed sample values used for prediction. The BV is signaled in the bitstream. Intra BC prediction is a form of intra-picture prediction – intra BC prediction for a block of a picture does not use any sample values other than sample values in the same picture.

[004] As currently specified in draft extensions to the HEVC standard and implemented in some reference software for the draft extensions to the HEVC standard,

2014374141 19 Sep 2018

intra BC prediction mode has several problems. For example, encoder-side decisions about how to use intra BC prediction are not made effectively. As another example, BV values are not signaled efficiently in many scenarios.

[004A] It is desired to overcome or alleviate one or more difficulties of the prior art, or to at least provide a useful alternative.

SUMMARY

[005] In summary, the detailed description presents innovations in encoder-side operations for intra block copy ("BC") prediction mode. For example, some of the innovations relate to ways of estimating reconstructed sample values within an overlap area of a current block during block vector ("BV") estimation. Other innovations relate to prediction of BV values during encoding or decoding using a "ping-pong" approach, in which an encoder or decoder selects between a pair of candidate BV values when predicting the BV value for a block.

[005A] In accordance with some embodiments of the present invention, there is provided a computing device comprising:

an encoder configured to perform operations to produce encoded data for a picture, wherein the encoder is a video encoder or image encoder, and wherein the operations include:

estimating reconstructed sample values within an overlap area of a current block of the picture, wherein the current block has multiple smaller blocks, the overlap area covering parts of the current block that are in potential intra-prediction regions for at least one of the multiple smaller blocks, wherein the estimating the reconstructed sample values within the overlap area uses corresponding original sample values from the overlap area to estimate the reconstructed sample values, respectively, within the overlap area without using reconstructed sample values from outside the overlap area in place of the reconstructed sample values, respectively, within the overlap area; and

performing block vector ("BV") estimation to determine a BV value for the current block, the BV value indicating a displacement from the current block to a region of the picture that includes sample values used for prediction, and wherein the BV estimation uses at least some of the estimated reconstructed sample values within the overlap area of the current block; and

a buffer configured to store the encoded data for output as part of a bitstream.

2014374141 19 Sep 2018

[005B] In accordance with some embodiments of the present invention, there is provided in a computing device that implements a video encoder or image encoder, a method comprising:

encoding a picture to produce encoded data, wherein the encoding includes:

estimating reconstructed sample values within an overlap area of a current block of the picture, wherein the current block has multiple smaller blocks, the overlap area covering parts of the current block that are in potential intra-prediction regions for at least one of the multiple smaller blocks, wherein the estimating the reconstructed sample values within the overlap area uses corresponding original sample values from the overlap area to estimate the reconstructed sample values, respectively, within the overlap area without using reconstructed sample values from outside the overlap area in place of the reconstructed sample values, respectively, within the overlap area; and

performing block vector ("BV") estimation to determine a BV value for the current block, the BV value indicating a displacement from the current block to a region of the picture that includes sample values used for prediction, and wherein the BV estimation uses at least some of the estimated reconstructed sample values within the overlap area of the current block; and

outputting the encoded data as part of a bitstream.

[005C] In accordance with some embodiments of the present invention, there is provided one or more computer-readable media storing computer-executable instructions for causing a computing device, when programmed thereby, to perform operations, wherein the one or more computer-readable media are selected from the group consisting of volatile memory, non-volatile memory, magnetic disk, CD-ROM, and DVD, the operations comprising:

encoding a picture to produce encoded data, wherein the encoding includes:

estimating reconstructed sample values within an overlap area of a current block of the picture, wherein the current block has multiple smaller blocks, the overlap area covering parts of the current block that are in potential intra-prediction regions for at least one of the multiple smaller blocks, wherein the estimating the reconstructed sample values within the overlap area uses corresponding original sample values from the overlap area to estimate the reconstructed sample values, respectively, within the overlap area without using reconstructed sample values from outside the overlap area in place of the reconstructed sample values, respectively, within the overlap area; and

2014374141 19 Sep 2018

performing block vector ("BV") estimation to determine a BV value for the current block, the BV value indicating a displacement from the current block to a region of the picture that includes sample values used for prediction, and wherein the BV estimation uses at least some of the estimated reconstructed sample values within the overlap area of the current block; and

outputting the encoded data as part of a bitstream.

[006] According to a one aspect of the innovations described herein, an encoder or decoder processes a coding unit ("CU") of a current picture, where the CU has four prediction units ("PUs"). The CU is a set of one or more blocks for purposes of encoding and decoding, generally, and each of the PUs is a set of one or more blocks, within the CU, for purposes of signaling of prediction information and/or prediction processing. For each of the four PUs, the encoder or decoder (1) predicts a BV value using one of a pair of candidate BV values, where a flag value indicates the selection between the pair of candidate BV values, and (2) processes the BV value for the PU using the predicted BV value. The pair of candidate BV values can be different for different PUs of the CU. For example, for the first PU, the pair of candidate BV values includes a first initial candidate BV value (labeled BV_{init_0}) and a second initial candidate BV value (labeled BV_{init_1}). For the second PU, the pair of candidate BV values includes the BV value for the first PU and BV_{init_0}. For the third PU, the pair of candidate BV values includes the BV values for the first and second PUs of the current CU. Finally, for the fourth PU, the pair of candidate BV values includes the BV values for the second and third PUs of the current CU. For a subsequent CU of the current picture, however, the pair of candidate BV values includes the BV values for the first and second PUs of the current CU (even though the BV values for the first and second PUs of the current CU were processed before the BV values for the third and fourth PUs of the current CU).

[007] According to another aspect of the innovations described herein, an encoder or decoder processes multiple CUs of a current picture. At least one of the CUs has a single PU, and at least one of the CUs has multiple PUs. For a given CU of the multiple CUs, if the given CU has a single PU, the encoder or decoder (1) predicts a BV value using one of

a pair of candidate BV values (including a first initial candidate BV value stored in a first BV buffer and a second initial candidate BV value stored in a second BV buffer), where a flag value indicates the selection between the pair of candidate BV values, and (2) processes the BV value for the single PU using the predicted BV value. In doing so, the encoder or decoder selectively updates the first BV buffer and the second BV buffer depending on whether the BV value for the PU equals the BV value stored in the first BV buffer.

[008] Otherwise (the given CU has multiple PUs), for each of the PUs, the encoder or decoder (1) predicts a BV value using one of the pair of candidate BV values, where a flag value indicates the selection between the pair of candidate BV values, and (2) processes the BV value for the PU using the predicted BV value. The pair of candidate BV values can be different for different PUs. If the PU is a first or second PU of the given CU, the encoder or decoder selectively updates the first BV buffer and the second BV buffer depending on whether the BV value for the PU equals the BV value stored in the first BV buffer. For a third PU or fourth PU of the given CU, the encoder skips updating the BV buffers.

[009] According to another aspect of the innovations described herein, an encoder estimates reconstructed sample values within an overlap area of a current block of a picture. The current block has multiple smaller blocks. The overlap area covers parts of the current block that are in potential intra-prediction regions for at least one of the multiple smaller blocks. The encoder then performs BV estimation to determine a BV value for the current block. The BV estimation uses at least some of the estimated reconstructed sample values within the overlap area of the current block.

[010] The innovations for BV prediction and encoder-side options for intra BC prediction mode can be implemented as part of a method, as part of a computing device adapted to perform the method or as part of a tangible computer-readable media storing computer-executable instructions for causing a computing device to perform the method. The various innovations can be used in combination or separately.

[011] The foregoing and other objects, features, and advantages of the invention will become more apparent from the following detailed description, which proceeds with reference to the accompanying figures.

2014374141 19 Sep 2018

BRIEF DESCRIPTION OF THE DRAWINGS

[011A] Some embodiments of the present invention are hereinafter described, by way of example only, with reference to the accompanying drawings, wherein:

[012] Figure 1 is a diagram of an example computing system in which some described embodiments can be implemented.

[013] Figures 2a and 2b are diagrams of example network environments in which some described embodiments can be implemented.

[014] Figure 3 is a diagram of an example encoder system in conjunction with which some described embodiments can be implemented.

[015] Figure 4 is a diagram of an example decoder system in conjunction with which some described embodiments can be implemented.

[016] Figures 5a and 5b are diagrams illustrating an example video encoder in conjunction with which some described embodiments can be implemented.

[017] Figure 6 is a diagram illustrating an example video decoder in conjunction with which some described embodiments can be implemented.

[018] Figure 7 is diagram illustrating intra BC prediction for a block of a picture.

[019] Figure 8 is a diagram illustrating example constraints on search range for BV values.

[020] Figure 9 is a flowchart illustrating a generalized technique for encoding with an intra BC prediction mode, subject to one or more constraints on selection of BV values.

[021] Figure 10 is a diagram illustrating example z-scan order for blocks of a picture.

[022] Figure 11 is a diagram illustrating candidate BV values in BV buffers, and Figure 12 is a diagram illustrating candidate BV values in another data structure.

[023] Figure 13 is a diagram illustrating an example of BV prediction for a 2*N*x2*N* PU of a 2*N*x2*N* CU according to example ping-pong approaches.

[024] Figure 14 is a diagram illustrating examples of BV prediction for *N*x*N* PUs of a 2*N*x2*N* CU according to example ping-pong approaches.

[025] Figure 15 is a diagram illustrating examples of BV prediction for *N*x2*N* PUs or 2*N*x*N* PUs of a 2*N*x2*N* CU according to example ping-pong approaches.

[026] Figure 16 is a flowchart illustrating an example technique for predicting BV values for a CU with multiple PUs according to example ping-pong approaches.

[027] Figure 17 is a flowchart illustrating an example technique for predicting BV values for CUs according to example ping-pong approaches.

[028] Figure 18 is a flowchart illustrating a generalized technique for selectively merging blocks into a larger block during BV estimation.

[029] Figure 19 is a diagram illustrating an advantage of selectively merging blocks into a larger block during BV estimation.

[030] Figure 20 is a flowchart illustrating a generalized technique for concurrently performing BV estimation and making block splitting decisions for a block.

[031] Figure 21 is a flowchart illustrating an example technique for concurrently evaluating a candidate BV value and block splitting decisions for a block.

[032] Figures 22a and 22b are diagrams illustrating an overlap area of a current block during BV estimation.

[033] Figure 23 is a flowchart illustrating an example technique for BV estimation in which an encoder estimates reconstructed sample values within an overlap area.

[034] Figure 24 is a diagram illustrating an overlap area of blocks with minimum transform size for video in YUV 4:2:2 format during BV estimation.

DETAILED DESCRIPTION

[035] The detailed description presents innovations in block vector ("BV") prediction during encoding and decoding, and in encoder-side decisions for intra block copy ("BC") prediction mode during encoding. In particular, the detailed description presents innovations for estimating sample values within an overlap area of a current block during BV estimation. The detailed description also presents innovations for BV prediction during encoding or decoding using "ping-pong" approaches, according to which an encoder or decoder selects between a pair of candidate BV values when predicting the BV value for a block.

[036] Although operations described herein are in places described as being performed by a video encoder or video decoder, in many cases the operations can be performed by another type of media processing tool (*e.g.*, image encoder or image decoder).

[037] Some of the innovations described herein are illustrated with reference to syntax elements and operations specific to the HEVC standard. For example, reference is made to the draft version JCTVC-O1005 of the HEVC standard – "High Efficiency Video Coding (HEVC) Range Extensions Text Specification: Draft 5," JCTVC-O1005-v3, November 2013. The innovations described herein can also be implemented for other standards or formats.

[038] More generally, various alternatives to the examples described herein are possible. For example, some of the methods described herein can be altered by changing the ordering of the method acts described, by splitting, repeating, or omitting certain

method acts, etc. The various aspects of the disclosed technology can be used in combination or separately. Different embodiments use one or more of the described innovations. Some of the innovations described herein address one or more of the problems noted in the background. Typically, a given technique/tool does not solve all such problems.

I. Example Computing Systems.

[039] Figure 1 illustrates a generalized example of a suitable computing system (100) in which several of the described innovations may be implemented. The computing system (100) is not intended to suggest any limitation as to scope of use or functionality, as the innovations may be implemented in diverse general-purpose or special-purpose computing systems.

[040] With reference to Figure 1, the computing system (100) includes one or more processing units (110, 115) and memory (120, 125). The processing units (110, 115) execute computer-executable instructions. A processing unit can be a general-purpose central processing unit ("CPU"), processor in an application-specific integrated circuit ("ASIC") or any other type of processor. In a multi-processing system, multiple processing units execute computer-executable instructions to increase processing power. For example, Figure 1 shows a central processing unit (110) as well as a graphics processing unit or co-processing unit (115). The tangible memory (120, 125) may be volatile memory (*e.g.*, registers, cache, RAM), non-volatile memory (*e.g.*, ROM, EEPROM, flash memory, etc.), or some combination of the two, accessible by the processing unit(s). The memory (120, 125) stores software (180) implementing one or more innovations for BV prediction and/or estimating sample values within an overlap area of a current block during BV estimation, or other encoder-side options for intra BC prediction mode, in the form of computer-executable instructions suitable for execution by the processing unit(s).

[041] A computing system may have additional features. For example, the computing system (100) includes storage (140), one or more input devices (150), one or more output devices (160), and one or more communication connections (170). An interconnection mechanism (not shown) such as a bus, controller, or network interconnects the components of the computing system (100). Typically, operating system software (not shown) provides an operating environment for other software executing in the computing system (100), and coordinates activities of the components of the computing system (100).

[042] The tangible storage (140) may be removable or non-removable, and includes magnetic disks, magnetic tapes or cassettes, CD-ROMs, DVDs, or any other medium which can be used to store information and which can be accessed within the computing system (100). The storage (140) stores instructions for the software (180) implementing one or more innovations for BV prediction and/or estimating sample values within an overlap area of a current block during BV estimation, or other encoder-side options for intra BC prediction mode.

[043] The input device(s) (150) may be a touch input device such as a keyboard, mouse, pen, or trackball, a voice input device, a scanning device, or another device that provides input to the computing system (100). For video, the input device(s) (150) may be a camera, video card, TV tuner card, or similar device that accepts video input in analog or digital form, or a CD-ROM or CD-RW that reads video input into the computing system (100). The output device(s) (160) may be a display, printer, speaker, CD-writer, or another device that provides output from the computing system (100).

[044] The communication connection(s) (170) enable communication over a communication medium to another computing entity. The communication medium conveys information such as computer-executable instructions, audio or video input or output, or other data in a modulated data signal. A modulated data signal is a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media can use an electrical, optical, RF, or other carrier.

[045] The innovations can be described in the general context of computer-readable media. Computer-readable media are any available tangible media that can be accessed within a computing environment. By way of example, and not limitation, with the computing system (100), computer-readable media include memory (120, 125), storage (140), and combinations of any of the above.

[046] The innovations can be described in the general context of computer-executable instructions, such as those included in program modules, being executed in a computing system on a target real or virtual processor. Generally, program modules include routines, programs, libraries, objects, classes, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The functionality of the program modules may be combined or split between program modules as desired in various embodiments. Computer-executable instructions for program modules may be executed within a local or distributed computing system.

[047] The terms "system" and "device" are used interchangeably herein. Unless the context clearly indicates otherwise, neither term implies any limitation on a type of computing system or computing device. In general, a computing system or computing device can be local or distributed, and can include any combination of special-purpose hardware and/or general-purpose hardware with software implementing the functionality described herein.

[048] The disclosed methods can also be implemented using specialized computing hardware configured to perform any of the disclosed methods. For example, the disclosed methods can be implemented by an integrated circuit (*e.g.*, an ASIC (such as an ASIC digital signal processor ("DSP"), a graphics processing unit ("GPU"), or a programmable logic device ("PLD"), such as a field programmable gate array ("FPGA")) specially designed or configured to implement any of the disclosed methods.

[049] For the sake of presentation, the detailed description uses terms like "determine" and "use" to describe computer operations in a computing system. These terms are high-level abstractions for operations performed by a computer, and should not be confused with acts performed by a human being. The actual computer operations corresponding to these terms vary depending on implementation. As used herein, the term "optimiz*" (including variations such as optimization and optimizing) refers to a choice among options under a given scope of decision, and does not imply that an optimized choice is the "best" or "optimum" choice for an expanded scope of decisions.

II. Example Network Environments.

[050] Figures 2a and 2b show example network environments (201, 202) that include video encoders (220) and video decoders (270). The encoders (220) and decoders (270) are connected over a network (250) using an appropriate communication protocol. The network (250) can include the Internet or another computer network.

[051] In the network environment (201) shown in Figure 2a, each real-time communication ("RTC") tool (210) includes both an encoder (220) and a decoder (270) for bidirectional communication. A given encoder (220) can produce output compliant with a variation or extension of the HEVC standard (also known as H.265), SMPTE 421M standard, ISO-IEC 14496-10 standard (also known as H.264 or AVC), another standard, or a proprietary format, with a corresponding decoder (270) accepting encoded data from the encoder (220). The bidirectional communication can be part of a video conference, video telephone call, or other two-party or multi-party communication scenario. Although the network environment (201) in Figure 2a includes two real-time communication tools

(210), the network environment (201) can instead include three or more real-time communication tools (210) that participate in multi-party communication.

[052] A real-time communication tool (210) manages encoding by an encoder (220). Figure 3 shows an example encoder system (300) that can be included in the real-time communication tool (210). Alternatively, the real-time communication tool (210) uses another encoder system. A real-time communication tool (210) also manages decoding by a decoder (270). Figure 4 shows an example decoder system (400), which can be included in the real-time communication tool (210). Alternatively, the real-time communication tool (210) uses another decoder system.

[053] In the network environment (202) shown in Figure 2b, an encoding tool (212) includes an encoder (220) that encodes video for delivery to multiple playback tools (214), which include decoders (270). The unidirectional communication can be provided for a video surveillance system, web camera monitoring system, remote desktop conferencing presentation or other scenario in which video is encoded and sent from one location to one or more other locations. Although the network environment (202) in Figure 2b includes two playback tools (214), the network environment (202) can include more or fewer playback tools (214). In general, a playback tool (214) communicates with the encoding tool (212) to determine a stream of video for the playback tool (214) to receive. The playback tool (214) receives the stream, buffers the received encoded data for an appropriate period, and begins decoding and playback.

[054] Figure 3 shows an example encoder system (300) that can be included in the encoding tool (212). Alternatively, the encoding tool (212) uses another encoder system. The encoding tool (212) can also include server-side controller logic for managing connections with one or more playback tools (214). Figure 4 shows an example decoder system (400), which can be included in the playback tool (214). Alternatively, the playback tool (214) uses another decoder system. A playback tool (214) can also include client-side controller logic for managing connections with the encoding tool (212).

III. Example Encoder Systems.

[055] Figure 3 is a block diagram of an example encoder system (300) in conjunction with which some described embodiments may be implemented. The encoder system (300) can be a general-purpose encoding tool capable of operating in any of multiple encoding modes such as a low-latency encoding mode for real-time communication, a transcoding mode, and a higher-latency encoding mode for producing media for playback from a file or stream, or it can be a special-purpose encoding tool

adapted for one such encoding mode. The encoder system (300) can be implemented as an operating system module, as part of an application library or as a standalone application. Overall, the encoder system (300) receives a sequence of source video frames (311) from a video source (310) and produces encoded data as output to a channel (390). The encoded data output to the channel can include content encoded using intra BC prediction mode.

[056] The video source (310) can be a camera, tuner card, storage media, or other digital video source. The video source (310) produces a sequence of video frames at a frame rate of, for example, 30 frames per second. As used herein, the term "frame" generally refers to source, coded or reconstructed image data. For progressive-scan video, a frame is a progressive-scan video frame. For interlaced video, in example embodiments, an interlaced video frame is de-interlaced prior to encoding. Alternatively, two complementary interlaced video fields are encoded together as a single video frame or encoded as two separately-encoded fields. Aside from indicating a progressive-scan video frame or interlaced-scan video frame, the term "frame" or "picture" can indicate a single non-paired video field, a complementary pair of video fields, a video object plane that represents a video object at a given time, or a region of interest in a larger image. The video object plane or region can be part of a larger image that includes multiple objects or regions of a scene.

[057] An arriving source frame (311) is stored in a source frame temporary memory storage area (320) that includes multiple frame buffer storage areas (321, 322, … , 32*n*). A frame buffer (321, 322, etc.) holds one source frame in the source frame storage area (320). After one or more of the source frames (311) have been stored in frame buffers (321, 322, etc.), a frame selector (330) selects an individual source frame from the source frame storage area (320). The order in which frames are selected by the frame selector (330) for input to the encoder (340) may differ from the order in which the frames are produced by the video source (310), *e.g.*, the encoding of some frames may be delayed in order, so as to allow some later frames to be encoded first and to thus facilitate temporally backward prediction. Before the encoder (340), the encoder system (300) can include a pre-processor (not shown) that performs pre-processing (*e.g.*, filtering) of the selected frame (331) before encoding. The pre-processing can include color space conversion into primary (*e.g.*, luma) and secondary (*e.g.*, chroma differences toward red and toward blue) components and resampling processing (*e.g.*, to reduce the spatial resolution of chroma components) for encoding. Typically, before encoding, video has been converted to a

color space such as YUV, in which sample values of a luma (Y) component represent brightness or intensity values, and sample values of chroma (U, V) components represent color-difference values. The chroma sample values may be sub-sampled to a lower chroma sampling rate (*e.g.*, for YUV 4:2:0 format or YUV 4:2:2), or the chroma sample values may have the same resolution as the luma sample values (*e.g.*, for YUV 4:4:4 format). In YUV 4:2:0 format, chroma components are downsampled by a factor of two horizontally and by a factor of two vertically. In YUV 4:2:2 format, chroma components are downsampled by a factor of two horizontally. Or, the video can be encoded in another format (*e.g.*, RGB 4:4:4 format).

[058] The encoder (340) encodes the selected frame (331) to produce a coded frame (341) and also produces memory management control operation ("MMCO") signals (342) or reference picture set ("RPS") information. If the current frame is not the first frame that has been encoded, when performing its encoding process, the encoder (340) may use one or more previously encoded/decoded frames (369) that have been stored in a decoded frame temporary memory storage area (360). Such stored decoded frames (369) are used as reference frames for inter-frame prediction of the content of the current source frame (331). The MMCO/RPS information (342) indicates to a decoder which reconstructed frames may be used as reference frames, and hence should be stored in a frame storage area.

[059] Generally, the encoder (340) includes multiple encoding modules that perform encoding tasks such as partitioning into tiles, intra prediction estimation and prediction, motion estimation and compensation, frequency transforms, quantization and entropy coding. The exact operations performed by the encoder (340) can vary depending on compression format. The format of the output encoded data can be a variation or extension of HEVC format (H.265), Windows Media Video format, VC-1 format, MPEG-x format (*e.g.*, MPEG-1, MPEG-2, or MPEG-4), H.26x format (*e.g.*, H.261, H.262, H.263, H.264), or another format.

[060] The encoder (340) can partition a frame into multiple tiles of the same size or different sizes. For example, the encoder (340) splits the frame along tile rows and tile columns that, with frame boundaries, define horizontal and vertical boundaries of tiles within the frame, where each tile is a rectangular region. Tiles are often used to provide options for parallel processing. A frame can also be organized as one or more slices, where a slice can be an entire frame or region of the frame. A slice can be decoded independently of other slices in a frame, which improves error resilience. The content of a

slice or tile is further partitioned into blocks or other sets of samples for purposes of encoding and decoding.

[061] For syntax according to the HEVC standard, the encoder splits the content of a frame (or slice or tile) into coding tree units. A coding tree unit ("CTU") includes luma sample values organized as a luma coding tree block ("CTB") and corresponding chroma sample values organized as two chroma CTBs. The size of a CTU (and its CTBs) is selected by the encoder, and can be, for example, 64x64, 32x32 or 16x16 sample values. A CTU includes one or more coding units. A coding unit ("CU") has a luma coding block ("CB") and two corresponding chroma CBs. For example, a CTU with a 64x64 luma CTB and two 64x64 chroma CTBs (YUV 4:4:4 format) can be split into four CUs, with each CU including a 32x32 luma CB and two 32x32 chroma CBs, and with each CU possibly being split further into smaller CUs. Or, as another example, a CTU with a 64x64 luma CTB and two 32x32 chroma CTBs (YUV 4:2:0 format) can be split into four CUs, with each CU including a 32x32 luma CB and two 16x16 chroma CBs, and with each CU possibly being split further into smaller CUs. The smallest allowable size of CU (*e.g.*, 8x8, 16x16) can be signaled in the bitstream.

[062] Generally, a CU has a prediction mode such as inter or intra. A CU includes one or more prediction units for purposes of signaling of prediction information (such as prediction mode details, displacement values, etc.) and/or prediction processing. A prediction unit ("PU") has a luma prediction block ("PB") and two chroma PBs. For an intra-predicted CU, the PU has the same size as the CU, unless the CU has the smallest size (*e.g.*, 8x8). In that case, the CU can be split into four smaller PUs (*e.g.*, each 4x4 if the smallest CU size is 8x8) or the PU can have the smallest CU size, as indicated by a syntax element for the CU. A CU also has one or more transform units for purposes of residual coding/decoding, where a transform unit ("TU") has a transform block ("TB") and two chroma TBs. A PU in an intra-predicted CU may contain a single TU (equal in size to the PU) or multiple TUs. As used herein, the term "block" can indicate a CB, PB, TB or other set of sample values, depending on context. The encoder decides how to partition video into CTUs, CUs, PUs, TUs, etc.

[063] Returning to Figure 3, the encoder represents an intra-coded block of a source frame (331) in terms of prediction from other, previously reconstructed sample values in the frame (331). For intra BC prediction, an intra-picture estimator estimates displacement of a block with respect to the other, previously reconstructed sample values. An intra-frame prediction reference region (or intra-prediction region, for short) is a region

of samples in the frame that are used to generate BC-prediction values for the block. The intra-frame prediction region can be indicated with a block vector ("BV") value (determined in BV estimation). For intra spatial prediction for a block, the intra-picture estimator estimates extrapolation of the neighboring reconstructed sample values into the block. The intra-picture estimator can output prediction information (such as BV values for intra BC prediction or prediction mode (direction) for intra spatial prediction), which is entropy coded. An intra-frame prediction predictor applies the prediction information to determine intra prediction values.

[064] The encoder (340) represents an inter-frame coded, predicted block of a source frame (331) in terms of prediction from reference frames. A motion estimator estimates the motion of the block with respect to one or more reference frames (369). When multiple reference frames are used, the multiple reference frames can be from different temporal directions or the same temporal direction. A motion-compensated prediction reference region is a region of samples in the reference frame(s) that are used to generate motion-compensated prediction values for a block of samples of a current frame. The motion estimator outputs motion information such as motion vector ("MV") information, which is entropy coded. A motion compensator applies MVs to reference frames (369) to determine motion-compensated prediction values for inter-frame prediction.

[065] The encoder can determine the differences (if any) between a block's prediction values (intra or inter) and corresponding original values. These prediction residual values are further encoded using a frequency transform, quantization and entropy encoding. For example, the encoder (340) sets values for quantization parameter ("QP") for a picture, tile, slice and/or other portion of video, and quantizes transform coefficients accordingly. The entropy coder of the encoder (340) compresses quantized transform coefficient values as well as certain side information (*e.g.*, MV information, BV values, QP values, mode decisions, parameter choices). Typical entropy coding techniques include Exponential-Golomb coding, Golomb-Rice coding, arithmetic coding, differential coding, Huffman coding, run length coding, variable-length-to-variable-length ("V2V") coding, variable-length-to-fixed-length ("V2F") coding, Lempel-Ziv ("LZ") coding, dictionary coding, probability interval partitioning entropy coding ("PIPE"), and combinations of the above. The entropy coder can use different coding techniques for different kinds of information, can apply multiple techniques in combination (*e.g.*, by applying Golomb-Rice coding followed by arithmetic coding), and can choose from among multiple code tables within a particular coding technique.

[066] An adaptive deblocking filter is included within the motion compensation loop in the encoder (340) to smooth discontinuities across block boundary rows and/or columns in a decoded frame. Other filtering (such as de-ringing filtering, adaptive loop filtering ("ALF"), or sample-adaptive offset ("SAO") filtering; not shown) can alternatively or additionally be applied as in-loop filtering operations.

[067] The coded frames (341) and MMCO/RPS information (342) (or information equivalent to the MMCO/RPS information (342), since the dependencies and ordering structures for frames are already known at the encoder (340)) are processed by a decoding process emulator (350). The decoding process emulator (350) implements some of the functionality of a decoder, for example, decoding tasks to reconstruct reference frames. In a manner consistent with the MMCO/RPS information (342), the decoding process emulator (350) determines whether a given coded frame (341) needs to be reconstructed and stored for use as a reference frame in inter-frame prediction of subsequent frames to be encoded. If a coded frame (341) needs to be stored, the decoding process emulator (350) models the decoding process that would be conducted by a decoder that receives the coded frame (341) and produces a corresponding decoded frame (351). In doing so, when the encoder (340) has used decoded frame(s) (369) that have been stored in the decoded frame storage area (360), the decoding process emulator (350) also uses the decoded frame(s) (369) from the storage area (360) as part of the decoding process.

[068] The decoded frame temporary memory storage area (360) includes multiple frame buffer storage areas (361, 362, …, 36*n*). In a manner consistent with the MMCO/RPS information (342), the decoding process emulator (350) manages the contents of the storage area (360) in order to identify any frame buffers (361, 362, etc.) with frames that are no longer needed by the encoder (340) for use as reference frames. After modeling the decoding process, the decoding process emulator (350) stores a newly decoded frame (351) in a frame buffer (361, 362, etc.) that has been identified in this manner.

[069] The coded frames (341) and MMCO/RPS information (342) are buffered in a temporary coded data area (370). The coded data that is aggregated in the coded data area (370) contains, as part of the syntax of an elementary coded video bitstream, encoded data for one or more pictures. The coded data that is aggregated in the coded data area (370) can also include media metadata relating to the coded video data (*e.g.*, as one or more parameters in one or more supplemental enhancement information ("SEI") messages or video usability information ("VUI") messages).

[070] The aggregated data (371) from the temporary coded data area (370) are processed by a channel encoder (380). The channel encoder (380) can packetize and/or multiplex the aggregated data for transmission or storage as a media stream (*e.g.*, according to a media program stream or transport stream format such as ITU-T H.222.0 | ISO/IEC 13818-1 or an Internet real-time transport protocol format such as IETF RFC 3550), in which case the channel encoder (380) can add syntax elements as part of the syntax of the media transmission stream. Or, the channel encoder (380) can organize the aggregated data for storage as a file (*e.g.*, according to a media container format such as ISO/IEC 14496-12), in which case the channel encoder (380) can add syntax elements as part of the syntax of the media storage file. Or, more generally, the channel encoder (380) can implement one or more media system multiplexing protocols or transport protocols, in which case the channel encoder (380) can add syntax elements as part of the syntax of the protocol(s). The channel encoder (380) provides output to a channel (390), which represents storage, a communications connection, or another channel for the output. The channel encoder (380) or channel (390) may also include other elements (not shown), *e.g.*, for forward-error correction ("FEC") encoding and analog signal modulation.

IV. Example Decoder Systems.

[071] Figure 4 is a block diagram of an example decoder system (400) in conjunction with which some described embodiments can be implemented. The decoder system (400) can be a general-purpose decoding tool capable of operating in any of multiple decoding modes such as a low-latency decoding mode for real-time communication and a higher-latency decoding mode for media playback from a file or stream, or it can be a special-purpose decoding tool adapted for one such decoding mode. The decoder system (400) can be implemented as an operating system module, as part of an application library or as a standalone application. Overall, the decoder system (400) receives coded data from a channel (410) and produces reconstructed frames as output for an output destination (490). The coded data can include content encoded using intra BC prediction mode.

[072] The decoder system (400) includes a channel (410), which can represent storage, a communications connection, or another channel for coded data as input. The channel (410) produces coded data that has been channel coded. A channel decoder (420) can process the coded data. For example, the channel decoder (420) de-packetizes and/or demultiplexes data that has been aggregated for transmission or storage as a media stream (*e.g.*, according to a media program stream or transport stream format such as ITU-T

H.222.0 | ISO/IEC 13818-1 or an internet real-time transport protocol format such as IETF RFC 3550), in which case the channel decoder (420) can parse syntax elements added as part of the syntax of the media transmission stream. Or, the channel decoder (420) separates coded video data that has been aggregated for storage as a file (*e.g.*, according to a media container format such as ISO/IEC 14496-12), in which case the channel decoder (420) can parse syntax elements added as part of the syntax of the media storage file. Or, more generally, the channel decoder (420) can implement one or more media system demultiplexing protocols or transport protocols, in which case the channel decoder (420) can parse syntax elements added as part of the syntax of the protocol(s). The channel (410) or channel decoder (420) may also include other elements (not shown), *e.g.*, for FEC decoding and analog signal demodulation.

[073] The coded data (421) that is output from the channel decoder (420) is stored in a temporary coded data area (430) until a sufficient quantity of such data has been received. The coded data (421) includes coded frames (431) and MMCO/RPS information (432). The coded data (421) in the coded data area (430) contain, as part of the syntax of an elementary coded video bitstream, coded data for one or more pictures. The coded data (421) in the coded data area (430) can also include media metadata relating to the encoded video data (*e.g.*, as one or more parameters in one or more SEI messages or VUI messages).

[074] In general, the coded data area (430) temporarily stores coded data (421) until such coded data (421) is used by the decoder (450). At that point, coded data for a coded frame (431) and MMCO/RPS information (432) are transferred from the coded data area (430) to the decoder (450). As decoding continues, new coded data is added to the coded data area (430) and the oldest coded data remaining in the coded data area (430) is transferred to the decoder (450).

[075] The decoder (450) decodes a coded frame (431) to produce a corresponding decoded frame (451). As appropriate, when performing its decoding process, the decoder (450) may use one or more previously decoded frames (469) as reference frames for inter-frame prediction. The decoder (450) reads such previously decoded frames (469) from a decoded frame temporary memory storage area (460). Generally, the decoder (450) includes multiple decoding modules that perform decoding tasks such as entropy decoding, intra-frame prediction, motion-compensated inter-frame prediction, inverse quantization, inverse frequency transforms, and merging of tiles. The exact operations performed by the decoder (450) can vary depending on compression format.

[076] For example, the decoder (450) receives encoded data for a compressed frame or sequence of frames and produces output including decoded frame (451). In the decoder (450), a buffer receives encoded data for a compressed frame and, at an appropriate time, makes the received encoded data available to an entropy decoder. The entropy decoder entropy decodes entropy-coded quantized data as well as entropy-coded side information, typically applying the inverse of entropy encoding performed in the encoder. A motion compensator applies motion information to one or more reference frames to form motion-compensated prediction values for any inter-coded blocks of the frame being reconstructed. An intra-frame prediction module can spatially predict sample values of a current block from neighboring, previously reconstructed sample values or, for intra BC prediction, predict sample values of a current block using previously reconstructed sample values of an intra-frame prediction region in the frame. The intra-frame prediction region can be indicated with a BV value. The decoder (450) also reconstructs prediction residual values. An inverse quantizer inverse quantizes entropy-decoded data. For example, the decoder (450) sets values for QP for a picture, tile, slice and/or other portion of video based on syntax elements in the bitstream, and inverse quantizes transform coefficients accordingly. An inverse frequency transformer converts the quantized, frequency-domain data into spatial-domain data. For an inter-frame predicted block, the decoder (450) combines reconstructed prediction residual values with motion-compensated prediction values. The decoder (450) can similarly combine prediction residual values with prediction values from intra-frame prediction. An adaptive deblocking filter is included within the motion compensation loop in the video decoder (450) to smooth discontinuities across block boundary rows and/or columns in the decoded frame (451). Other filtering (such as de-ringing filtering, ALF, or SAO filtering; not shown) can alternatively or additionally be applied as in-loop filtering operations.

[077] The decoded frame temporary memory storage area (460) includes multiple frame buffer storage areas (461, 462, ..., 46*n*). The decoded frame storage area (460) is an example of a decoded picture buffer. The decoder (450) uses the MMCO/RPS information (432) to identify a frame buffer (461, 462, etc.) in which it can store a decoded frame (451). The decoder (450) stores the decoded frame (451) in that frame buffer.

[078] An output sequencer (480) identifies when the next frame to be produced in output order is available in the decoded frame storage area (460). When the next frame (481) to be produced in output order is available in the decoded frame storage area (460),

it is read by the output sequencer (480) and output to the output destination (490) (*e.g.*, display). In general, the order in which frames are output from the decoded frame storage area (460) by the output sequencer (480) may differ from the order in which the frames are decoded by the decoder (450).

V. Example Video Encoders.

[079] Figures 5a and 5b are a block diagram of a generalized video encoder (500) in conjunction with which some described embodiments may be implemented. The encoder (500) receives a sequence of video pictures including a current picture as an input video signal (505) and produces encoded data in a coded video bitstream (595) as output.

[080] The encoder (500) is block-based and uses a block format that depends on implementation. Blocks may be further sub-divided at different stages, *e.g.*, at the prediction, frequency transform and/or entropy encoding stages. For example, a picture can be divided into 64x64 blocks, 32x32 blocks or 16x16 blocks, which can in turn be divided into smaller blocks of sample values for coding and decoding. In implementations of encoding for the HEVC standard, the encoder partitions a picture into CTUs (CTBs), CUs (CBs), PUs (PBs) and TU (TBs).

[081] The encoder (500) compresses pictures using intra-picture coding and/or inter-picture coding. Many of the components of the encoder (500) are used for both intra-picture coding and inter-picture coding. The exact operations performed by those components can vary depending on the type of information being compressed.

[082] A tiling module (510) optionally partitions a picture into multiple tiles of the same size or different sizes. For example, the tiling module (510) splits the picture along tile rows and tile columns that, with picture boundaries, define horizontal and vertical boundaries of tiles within the picture, where each tile is a rectangular region.

[083] The general encoding control (520) receives pictures for the input video signal (505) as well as feedback (not shown) from various modules of the encoder (500). Overall, the general encoding control (520) provides control signals (not shown) to other modules (such as the tiling module (510), transformer/scaler/quantizer (530), scaler/inverse transformer (535), intra-picture estimator (540), motion estimator (550) and intra/inter switch) to set and change coding parameters during encoding. In particular, the general encoding control (520) can decide whether and how to use intra BC prediction during encoding. The general encoding control (520) can also evaluate intermediate results during encoding, for example, performing rate-distortion analysis. The general encoding control (520) produces general control data (522) that indicates decisions made

during encoding, so that a corresponding decoder can make consistent decisions. The general control data (522) is provided to the header formatter/entropy coder (590).

[084] If the current picture is predicted using inter-picture prediction, a motion estimator (550) estimates the motion of blocks of sample values of the current picture of the input video signal (505) with respect to one or more reference pictures. The decoded picture buffer (570) buffers one or more reconstructed previously coded pictures for use as reference pictures. When multiple reference pictures are used, the multiple reference pictures can be from different temporal directions or the same temporal direction. The motion estimator (550) produces as side information motion data (552) such as MV data, merge mode index values and reference picture selection data. The motion data (552) is provided to the header formatter/entropy coder (590) as well as the motion compensator (555).

[085] The motion compensator (555) applies MVs to the reconstructed reference picture(s) from the decoded picture buffer (570). The motion compensator (555) produces motion-compensated predictions for the current picture.

[086] In a separate path within the encoder (500), an intra-picture estimator (540) determines how to perform intra-picture prediction for blocks of sample values of a current picture of the input video signal (505). The current picture can be entirely or partially coded using intra-picture coding. Using values of a reconstruction (538) of the current picture, for intra spatial prediction, the intra-picture estimator (540) determines how to spatially predict sample values of a current block of the current picture from neighboring, previously reconstructed sample values of the current picture. Or, for intra BC prediction using BV values, the intra-picture estimator (540) estimates displacement of the sample values of the current block to different candidate regions within the current picture. BV values can be predicted using a ping-pong approach to BV prediction, as described below. For intra BC prediction, the intra-prediction estimator (540) can constrain the BV selection process using one or more constraints described below, and it can estimate reconstructed sample values in an overlap area of a current block for purposes of BV estimation.

[087] The intra-picture estimator (540) produces as side information intra prediction data (542), such as information indicating whether intra prediction uses spatial prediction or intra BC prediction (*e.g.*, a flag value per intra block), prediction mode direction (for intra spatial prediction), and BV values (for intra BC prediction). The intra prediction data

(542) is provided to the header formatter/entropy coder (590) as well as the intra-picture predictor (545).

[088] According to the intra prediction data (542), the intra-picture predictor (545) spatially predicts sample values of a current block of the current picture from neighboring, previously reconstructed sample values of the current picture. Or, for intra BC prediction, the intra-picture predictor (545) predicts the sample values of the current block using previously reconstructed sample values of an intra-prediction region, which is indicated by a BV value for the current block. When the chroma data for a picture has the same resolution as the luma data (*e.g.* when the format is YUV 4:4:4 format or RGB 4:4:4 format), the BV value that is applied for the chroma block may be the same as the BV value applied for the luma block. On the other hand, when the chroma data for a picture has reduced resolution relative to the luma data (*e.g.* when the format is YUV 4:2:0 format or YUV 4:2:2 format), the BV value that is applied for the chroma block may be scaled down and possibly rounded to adjust for the difference in chroma resolution (*e.g.*, for YUV 4:2:0 format, by dividing the vertical and horizontal components of the BV value by two and truncating or rounding them to integer values; for YUV 4:2:2 format, by dividing the horizontal component of the BV value by two and truncating or rounding it to an integer value).

[089] The intra/inter switch selects values of a motion-compensated prediction or intra-picture prediction for use as the prediction (558) for a given block. The difference (if any) between a block of the prediction (558) and a corresponding part of the original current picture of the input video signal (505) provides values of the residual (518). During reconstruction of the current picture, reconstructed residual values are combined with the prediction (558) to produce a reconstruction (538) of the original content from the video signal (505). In lossy compression, however, some information is still lost from the video signal (505).

[090] In the transformer/scaler/quantizer (530), a frequency transformer converts spatial-domain video data into frequency-domain (*i.e.*, spectral, transform) data. For block-based video coding, the frequency transformer applies a discrete cosine transform ("DCT"), an integer approximation thereof, or another type of forward block transform (*e.g.*, a discrete sine transform or an integer approximation thereof) to blocks of prediction residual data (or sample value data if the prediction (558) is null), producing blocks of frequency transform coefficients. The encoder (500) may also be able to indicate that such transform step is skipped. The scaler/quantizer scales and quantizes the transform

coefficients. For example, the quantizer applies dead-zone scalar quantization to the frequency-domain data with a quantization step size that varies on a frame-by-frame basis, tile-by-tile basis, slice-by-slice basis, block-by-block basis, frequency-specific basis or other basis. The quantized transform coefficient data (532) is provided to the header formatter/entropy coder (590).

[091] In the scaler/inverse transformer (535), a scaler/inverse quantizer performs inverse scaling and inverse quantization on the quantized transform coefficients. An inverse frequency transformer performs an inverse frequency transform, producing blocks of reconstructed prediction residual values or sample values. The encoder (500) combines reconstructed residual values with values of the prediction (558) (*e.g.*, motion-compensated prediction values, intra-picture prediction values) to form the reconstruction (538).

[092] For intra-picture prediction, the values of the reconstruction (538) can be fed back to the intra-picture estimator (540) and intra-picture predictor (545). Also, the values of the reconstruction (538) can be used for motion-compensated prediction of subsequent pictures. The values of the reconstruction (538) can be further filtered. A filtering control (560) determines how to perform deblock filtering and SAO filtering on values of the reconstruction (538), for a given picture of the video signal (505). The filtering control (560) produces filter control data (562), which is provided to the header formatter/entropy coder (590) and merger/filter(s) (565).

[093] In the merger/filter(s) (565), the encoder (500) merges content from different tiles into a reconstructed version of the picture. The encoder (500) selectively performs deblock filtering and SAO filtering according to the filter control data (562), so as to adaptively smooth discontinuities across boundaries in the frames. Other filtering (such as de-ringing filtering or ALF; not shown) can alternatively or additionally be applied. Tile boundaries can be selectively filtered or not filtered at all, depending on settings of the encoder (500), and the encoder (500) may provide syntax within the coded bitstream to indicate whether or not such filtering was applied. The decoded picture buffer (570) buffers the reconstructed current picture for use in subsequent motion-compensated prediction.

[094] The header formatter/entropy coder (590) formats and/or entropy codes the general control data (522), quantized transform coefficient data (532), intra prediction data (542), motion data (552) and filter control data (562). For example, the header formatter/entropy coder (590) uses context-adaptive binary arithmetic coding for entropy

coding of various syntax elements such as syntax elements for BV values. The header formatter/entropy coder (590) can use a ping-pong approach to BV prediction, as described below, when encoding BV values.

[095] The header formatter/entropy coder (590) provides the encoded data in the coded video bitstream (595). The format of the coded video bitstream (595) can be a variation or extension of HEVC format, Windows Media Video format, VC-1 format, MPEG-x format (*e.g.*, MPEG-1, MPEG-2, or MPEG-4), H.26x format (*e.g.*, H.261, H.262, H.263, H.264), or another format.

[096] Depending on implementation and the type of compression desired, modules of the encoder can be added, omitted, split into multiple modules, combined with other modules, and/or replaced with like modules. In alternative embodiments, encoders with different modules and/or other configurations of modules perform one or more of the described techniques. Specific embodiments of encoders typically use a variation or supplemented version of the encoder (500). The relationships shown between modules within the encoder (500) indicate general flows of information in the encoder; other relationships are not shown for the sake of simplicity.

VI. Example Video Decoders.

[097] Figure 6 is a block diagram of a generalized decoder (600) in conjunction with which some described embodiments can be implemented. The decoder (600) receives encoded data in a coded video bitstream (605) and produces output including pictures for reconstructed video (695). The format of the coded video bitstream (605) can be a variation or extension of HEVC format, Windows Media Video format, VC-1 format, MPEG-x format (*e.g.*, MPEG-1, MPEG-2, or MPEG-4), H.26x format (*e.g.*, H.261, H.262, H.263, H.264), or another format.

[098] The decoder (600) is block-based and uses a block format that depends on implementation. Blocks may be further sub-divided at different stages. For example, a picture can be divided into 64x64 blocks, 32x32 blocks or 16x16 blocks, which can in turn be divided into smaller blocks of sample values. In implementations of decoding for the HEVC standard, a picture is partitioned into CTUs (CTBs), CUs (CBs), PUs (PBs) and TU (TBs).

[099] The decoder (600) decompresses pictures using intra-picture decoding and/or inter-picture decoding. Many of the components of the decoder (600) are used for both intra-picture decoding and inter-picture decoding. The exact operations performed by those components can vary depending on the type of information being decompressed.

[0100] A buffer receives encoded data in the coded video bitstream (605) and makes the received encoded data available to the parser/entropy decoder (610). The parser/entropy decoder (610) entropy decodes entropy-coded data, typically applying the inverse of entropy coding performed in the encoder (500) (*e.g.*, context-adaptive binary arithmetic decoding). As a result of parsing and entropy decoding, the parser/entropy decoder (610) produces general control data (622), quantized transform coefficient data (632), intra prediction data (642), motion data (652) and filter control data (662). In particular, for the intra prediction data (642), the parser/entropy decoder (610) entropy decodes syntax elements for BV values, *e.g.*, using context-adaptive binary arithmetic decoding. The parser/entropy decoder (610) can use a ping-pong approach to BV prediction, as described below, when decoding BV values.

[0101] The general decoding control (620) receives the general control data (622) and provides control signals (not shown) to other modules (such as the scaler/inverse transformer (635), intra-picture predictor (645), motion compensator (655) and intra/inter switch) to set and change decoding parameters during decoding.

[0102] If the current picture is predicted using inter-picture prediction, a motion compensator (655) receives the motion data (652), such as MV data, reference picture selection data and merge mode index values. The motion compensator (655) applies MVs to the reconstructed reference picture(s) from the decoded picture buffer (670). The motion compensator (655) produces motion-compensated predictions for inter-coded blocks of the current picture. The decoded picture buffer (670) stores one or more previously reconstructed pictures for use as reference pictures.

[0103] In a separate path within the decoder (600), the intra-frame prediction predictor (645) receives the intra prediction data (642), such as information indicating whether intra prediction uses spatial prediction or intra BC prediction (*e.g.*, a flag value per intra block), prediction mode direction (for intra spatial prediction) and BV values (for intra BC prediction). For intra spatial prediction, using values of a reconstruction (638) of the current picture, according to prediction mode data, the intra-picture predictor (645) spatially predicts sample values of a current block of the current picture from neighboring, previously reconstructed sample values of the current picture. Or, for intra BC prediction using BV values, the intra-picture predictor (645) predicts the sample values of the current block using previously reconstructed sample values of an intra-prediction region, which is indicated by a BV value for the current block.

[0104] The intra/inter switch selects values of a motion-compensated prediction or intra-picture prediction for use as the prediction (658) for a given block. For example, when HEVC syntax is followed, the intra/inter switch can be controlled based on a syntax element encoded for a CU of a picture that can contain intra-predicted CUs and inter-predicted CUs. The decoder (600) combines the prediction (658) with reconstructed residual values to produce the reconstruction (638) of the content from the video signal.

[0105] To reconstruct the residual, the scaler/inverse transformer (635) receives and processes the quantized transform coefficient data (632). In the scaler/inverse transformer (635), a scaler/inverse quantizer performs inverse scaling and inverse quantization on the quantized transform coefficients. An inverse frequency transformer performs an inverse frequency transform, producing blocks of reconstructed prediction residual values or sample values. For example, the inverse frequency transformer applies an inverse block transform to frequency transform coefficients, producing sample value data or prediction residual data. The inverse frequency transform can be an inverse DCT, an integer approximation thereof, or another type of inverse frequency transform (*e.g.*, an inverse discrete sine transform or an integer approximation thereof).

[0106] For intra-picture prediction, the values of the reconstruction (638) can be fed back to the intra-picture predictor (645). For inter-picture prediction, the values of the reconstruction (638) can be further filtered. In the merger/filter(s) (665), the decoder (600) merges content from different tiles into a reconstructed version of the picture. The decoder (600) selectively performs deblock filtering and SAO filtering according to the filter control data (662) and rules for filter adaptation, so as to adaptively smooth discontinuities across boundaries in the frames. Other filtering (such as de-ringing filtering or ALF; not shown) can alternatively or additionally be applied. Tile boundaries can be selectively filtered or not filtered at all, depending on settings of the decoder (600) or a syntax indication within the encoded bitstream data. The decoded picture buffer (670) buffers the reconstructed current picture for use in subsequent motion-compensated prediction.

[0107] The decoder (600) can also include a post-processing filter. The post-processing filter can include de-ringing filtering, adaptive Wiener filtering, film-grain reproduction filtering, SAO filtering or another kind of filtering.

[0108] Depending on implementation and the type of decompression desired, modules of the decoder can be added, omitted, split into multiple modules, combined with other modules, and/or replaced with like modules. In alternative embodiments, decoders

with different modules and/or other configurations of modules perform one or more of the described techniques. Specific embodiments of decoders typically use a variation or supplemented version of the decoder (600). The relationships shown between modules within the decoder (600) indicate general flows of information in the decoder; other relationships are not shown for the sake of simplicity.

VII. Intra Block Copy Prediction Mode.

[0109] This section presents various features of intra block copy ("BC") prediction mode. Some of the features relate to selection of block vector ("BV") values, while others relate to encoding/decoding of BV values. These features can facilitate intra BC prediction that is more effective in terms of rate-distortion performance and/or computational efficiency of encoding and decoding. In particular, intra BC prediction can improve rate-distortion performance when encoding certain "artificially" created video content such as screen-capture content. Screen-capture content typically includes repeated structures (*e.g.*, graphics, text characters), which provide opportunities for intra BC prediction to improve performance. Screen capture content is usually encoded in a format (*e.g.*, YUV 4:4:4 or RGB 4:4:4) with high chroma sampling resolution, although it may also be encoded in a format with lower chroma sampling resolution (*e.g.,* YUV 4:2:0, YUV 4:2:2).

A. Intra BC Prediction Mode – Introduction.

[0110] For intra BC prediction mode, the sample values of a current block of a picture are predicted using sample values in the same picture. A BV indicates a displacement from the current block to a region of the picture that includes the sample values used for prediction. Typically, the sample values used for prediction are previously reconstructed sample values. The BV is signaled in the bitstream. Intra BC prediction is a form of intra-picture prediction – intra BC prediction for a block of a picture does not use any sample values other than sample values in the same picture.

[0111] Figure 7 illustrates intra BC prediction for a current block (730) of a current frame (710). The current block can be a coding block ("CB") of a coding unit ("CU"), prediction block ("PB") of a prediction unit ("PU"), transform block ("TB") of a transform unit ("TU") or other block. The size of the current block can be 64x64, 32x32, 16x16, 8x8 or some other size. More generally, the size of the current block is *m* x *n*, where each of *m* and *n* is a whole number, and where *m* and *n* can be equal to each other or can have different values. Alternatively, the current block can have some other shape (*e.g.*, an area of a coded video object with a non-rectangular shape).

[0112] The BV (740) indicates a displacement (or offset) from the current block (730) to a region (750) of the picture that includes the sample values used for prediction. Suppose the top left position of a current block is at position (x_0, y_0) in the current frame, and suppose the top left position of the intra-prediction region is at position (x_1, y_1) in the current frame. The BV indicates the displacement $(x_1 - x_0, y_1 - y_0)$. For example, if the top left position of the current block is at position (320, 256), and the top left position of the intra-prediction region is at position (295, 270), the BV value is (-25, 14). In this example, a negative horizontal displacement indicates a position to the left of the current block, and a negative vertical displacement indicates a position above the current block.

[0113] In some example implementations, the intra-predicted region (750) is constrained to be within the same slice and tile as the current block (730). Such intra BC prediction does not use sample values in other slices or tiles. The location of the intra-predicted region (750) may be subject to one or more other constraints (*e.g.*, for search range, regarding use of reconstructed sample values of inter-coded blocks).

[0114] A block with prediction mode of intra BC prediction can be a CB, PB or other block. When the block is a CB, the BV for the block can be signaled at CU level (and other CBs in the CU use the same BV or a scaled version thereof). Or, when the block is a PB, the BV for the block can be signaled at PU level (and other PBs in the PU use the same BV or a scaled version thereof). More generally, the BV for an intra-BC prediction block is signaled at an appropriate syntax level for the block.

[0115] The block copying operations of prediction according to the intra BC prediction mode can be performed at the level of CB (when a BV is signaled per CB) or PB (when a BV is signaled per PB). For example, suppose a 16x16 CB has a single 16x16 PB. The BV (for the PB) is applied to block copy a 16x16 region. When the intra-prediction region is constrained to not overlap the 16x16 block being predicted, the BV has a magnitude (absolute value) of at least 16 horizontally or vertically.

[0116] Alternatively, the block copying operations can be performed at the level of TBs within a PB or CB, even when the BV is signaled for the PB or CB. In this way, a BV, as applied for a TB, can reference positions of other TBs in the same PB or CB. For example, suppose a 16x16 CB has a single 16x16 PB but is split into sixteen 4x4 TBs for purposes of residual coding/decoding. The BV (for the PB) is applied to block copy a 4x4 region for the first TB in raster scan order, then the same BV is applied to block copy a 4x4 region for the second TB in raster scan order, and so on. The 4x4 region used in the BC operations for a TB can include positions in previously reconstructed TBs in the same

CB, after combining residual values with predicted values for those previously reconstructed TBs. (A BV still does not reference positions in the same TB that is being predicted). Applying BC operations at the TB level facilitates use of BVs with relatively small magnitudes.

[0117] Intra BC prediction operations for chroma blocks of a CU generally correspond to intra BC prediction operations for the luma block of the CU. Normally, the segmentation of chroma PBs and chroma TBs corresponds directly to the segmentation of the luma PBs and luma TBs in the CU. When the format of video is YUV 4:4:4, the sizes of chroma PBs and TBs match the sizes of corresponding luma PBs and TBs. When the format of video is YUV 4:2:0, chroma PBs and TBs are half the width and half the height of corresponding luma PBs and TBs. If a luma TB has minimum transform size, however, a single chroma TB having that minimum transform size is used. When the format of video is YUV 4:2:2, chroma PBs and TBs are half the width of corresponding luma PBs and TBs.

[0118] In some implementations, for an intra BC predicted CU, intra BC prediction for a chroma block in a PU uses the same BV value as intra BC prediction for the luma block in the PU, possibly after scaling and rounding when the chroma data has reduced resolution relative to the luma data (*e.g.* when the format is YUV 4:2:0 format or YUV 4:2:2 format). Alternatively, different BV values can be signaled for the luma block and chroma blocks of a PU.

[0119] In some implementations, if the prediction mode of the luma block of a PU is intra BC prediction, the prediction mode for the chroma blocks of the PU is also intra BC predicted. For example, the prediction mode is signaled for the PU. Alternatively, the prediction mode can be intra BC prediction for the luma block or chroma blocks of the PU, but not both.

B. Constraining BV Search for Intra BC Prediction Mode.

[0120] In some example implementations, an encoder limits BV range according to one or more constraints. By limiting BV range, the area of reconstructed sample values that is referenced by fast memory access for intra BC prediction during encoding and decoding can be reduced, which tends to lower implementation cost.

[0121] Figure 8 illustrates example constraints on search range for BV values. In addition to a current block (830) of a current frame (810), Figure 8 shows a search range defined by two CTBs (820, 822). The current CTB (820) is part of the current CTU and includes the current block (830). With the CTB (822) to its left, the current CTB (820)

defines a search range within which allowable BVs can be found for the current block (830). BVs (842, 844) reference regions that are outside the search range, so those BV values (842, 844) are not allowed.

[0122] In some example implementations, the search range for BV values for a current block is the current CTB and the CTB to its left. For example, a CTB can have size of 64x64, 32x32 or 16x16 sample values, which yields a search range of 128x64, 64x32 or 32x16 sample values. Only sample value in the current CTB and left CTB are used for intra BC prediction for the current block. This simplifies encoder implementation by constraining the search process. It also simplifies decoder implementation by limiting the number of sample values that the decoder buffers in fast memory for intra prediction. Another constraint is that intra prediction cannot reference sample values from another slice or tile. For a current m x n block with a top left position at (x_0, y_0) and CTB(s) each having dimensions CTB_{sizeY} x CTB_{sizeY}, an encoder can check these constraints for a two-dimensional BV having a horizontal component $BV[0]$ and vertical component $BV[1]$ as follows.

- $BV[0] \geq -((x_0 \,\%\, CTB_{sizeY}) + CTB_{sizeY})$
- $BV[1] \geq -(y_0 \,\%\, CTB_{sizeY})$
- The sample values at positions (x_0, y_0), $(x_0+BV[0], y_0+BV[1])$ and $(x_0+BV[0]+m-1, y_0+BV[1]+n-1)$ shall be in the same slice.
- The sample values at positions (x_0, y_0), $(x_0+BV[0], y_0+BV[1])$ and $(x_0+BV[0]+m-1, y_0+BV[1]+n-1)$ shall be in the same tile.

[0123] Figure 9 shows a technique (900) for encoding with an intra BC prediction mode, subject to one or more constraints on selection of BV values. An encoder such as one described with reference to Figure 3 or Figures 5a-5b can perform the technique (900).

[0124] To start, the encoder determines (910) a BV for a current block of a picture. The current block can be a CB, PB or other block. The BV indicates a displacement to a region within the picture. In determining the BV, the encoder checks one or more constraints.

[0125] According to one possible constraint, the encoder checks range of sample values used for intra BC prediction. The encoder can check that a candidate intra-prediction region is within a range defined by a current CTB and one or more other CTBs (*e.g.*, CTB to the left of the current CTB). For example, when the BV has a first component $BV[0]$ and a second component $BV[1]$, the current block has a top left position

at position (x_0, y_0), and each of the CTB(s) has width CTB_{width} and height CTB_{height}, the constraint is satisfied if $BV[0] >= -((x_0 \% CTB_{width}) + CTB_{width})$ and $BV[1] >= -(y_0 \% CTB_{height})$. The encoder can similarly check upper limits on values of $BV[0]$ and $BV[1]$ within the search range: $BV[0] < (CTB_{width} - m - (x0 \% CTB_{width}))$ and $BV[1] < (CTB_{height} - n - (y0 \% CTB_{height}))$. Alternatively, the search range includes more or fewer CTBs, or the search range is defined in some other way.

[0126] According to another possible constraint, the encoder limits searching to the current slice and tile (*i.e.*, the current block and region are part of no more than one slice of the picture and no more than one tile of the picture). The encoder can check that a top left position of the current block, a top left position of a candidate intra-prediction region and a bottom right position of the candidate intra-prediction region are part of a single slice and single tile. For example, the constraint is satisfied if (x_0, y_0), ($x_0 + BV[0]$, $y_0 + BV[1]$) and ($x_0 + BV[0] + m - 1$, $y_0 + BV[1] + n - 1$) are part of a single slice and single tile.

[0127] Alternatively, the encoder checks other and/or additional constraints.

[0128] The encoder performs (920) intra BC prediction for the current block using the BV. For example, the encoder performs intra BC prediction for the entire current block. Or, the encoder performs intra BC prediction for multiple blocks associated with the current block (*e.g.*, for multiple TBs on a TB-by-TB basis, where the TBs are associated with a current PB that has the BV).

[0129] The encoder encodes (930) the BV. For example, the encoder encodes (930) the BV as described below. The encoder can repeat the technique (900) for another intra BC prediction mode block.

[0130] For intra BC prediction, the encoder and decoder use reconstructed sample values. Unreconstructed sample values might be present as parts of a picture that have not been encoded and reconstructed yet. To avoid using unreconstructed sample values for intra BC prediction, the encoder can set constraints on allowable values of BV such that only actual, previously reconstructed sample values are used for intra BC prediction according to a BV.

[0131] In some example implementations, the encoder checks a BV value by considering the *z*-scan orders of the current block and the block that contains the bottom right position of the candidate intra-prediction region. More specifically, the encoder checks that the *z*-scan order of the block containing the position ($x_0+BV[0]+m-1$, $y_0+BV[1]+n-1$) is smaller than *z*-scan order of the block containing (x_0, y_0). If so, the block that contains the bottom right position of the intra-prediction region has been previously

reconstructed (and hence so has the rest of the intra-prediction region). The BV also satisfies at least one of the conditions $BV[0]+m \leq 0$ and $BV[1]+n \leq 0$, ensuring that the intra-prediction region does not overlap the current block.

[0132] The *z*-scan order follows a sequentially specified ordering of blocks that partition a picture. Figure 10 shows example *z*-scan order (1000) for a current block (1030) and blocks that might include the bottom right position of an intra-prediction region for a candidate BV. The current block (1030) can be a CB, PB or other block. The *z*-scan orders are generally assigned to blocks sequentially from left-to-right in a row, repeating in successive rows from top-to-bottom. When a block is split, *z*-scan orders are assigned within the split block, recursively. For implementations of encoding/decoding for the HEVC standard, the *z*-scan order proceeds CTB-to-CTB by a CTB raster scan pattern (left-to-right in a CTB row, repeating in successive CTB rows from top-to-bottom). If a CTB is split, the *z*-scan order follows a raster scan pattern for CBs of a quadtree within the split CTB. And, if a CB is split (*e.g.*, into multiple CBs, or into multiple PBs), the z-scan order follows a raster scan pattern for blocks within the split CB.

[0133] Alternatively, when intra BC prediction can be performed on a TB-by-TB basis, the encoder and decoder can check for possible overlap between an intra-prediction region and a current block (TB), then use the results of the check to decide whether the current TB should be split into smaller TBs for application of intra BC prediction operations. Suppose a current TB has a size of m x n, where m and n can be equal to each other or can have different values. If $BV[0] > -m$ and $BV[1] > -n$, the intra-prediction region overlaps the current m x n TB, which is problematic unless the current m x n TB is split into smaller TBs for application of intra BC prediction operations. Thus, if $BV[0] > -m$ and $BV[1] > -n$, the encoder and decoder split the current TB into smaller TBs. The same condition is checked (*e.g.*, checked recursively) for the smaller TBs, which may be further split if $BV[0] > -m$ and $BV[1] > -n$ even for the smaller values of m and n after splitting.

[0134] For example, suppose the BV for a PB is (-9, -5), and the current TB is a 32x32 block. The encoder and decoder determine that -9 > -32 and -5 > -32, indicating that the intra-prediction region (whose top left corner is displaced -9, -5) would overlap the current 32x32 TB. The encoder and decoder split the 32x32 TB into four 16x16 TBs. For each of the 16x16 TBs, the encoder and decoder determine that -9 > -16 and -5 > -16, indicating that the intra-prediction region (whose top left corner is displaced -9, -5) would overlap the current 16x16 TB. The encoder and decoder split each 16x16 TB, in

succession, into four 8x8 TBs. For an 8x8 TB, the BV of (-9, -5) is not problematic, so the 8x8 TB is not forced to be further split.

[0135] In this scenario, when a TB is split due to a BV value and size of the TB, the encoder can skip signaling of the flag value that would otherwise signal whether to split the current TB into smaller TBs. The bitstream of encoded data lacks the flag value directing the decoder to split the current TB into smaller TBs. Instead, the decoder can infer that a TB should be split due to a BV value and the size of the TB. This can save bits that would otherwise be spent signaling information about splitting TBs.

C. Encoding and Decoding of BV Values.

[0136] Collectively, BV values for blocks encoded using intra BC prediction can consume a significant number of bits. The BV values can be entropy encoded to reduce bit rate. To further reduce bit rate for BV values, an encoder can use prediction of the BV values. BV values often exhibit redundancy – the BV value for a given block is often similar to, or even the same as, the BV values of previous blocks in the picture. For BV prediction, the BV value for the given block is predicted using a BV predictor. The difference (or BV differential) between the BV value for the given block and the BV predictor is then entropy coded. Typically, the BV differential is computed for horizontal and vertical components of the BV value and BV predictor. When BV prediction works well, BV differentials have a probability distribution that supports efficient entropy coding. An encoder and decoder can use basic BV prediction and/or merge mode/BV competition when encoding/decoding BV values.

[0137] For basic BV prediction during encoding and decoding, the BV value for a current block can be predicted based on the BV values of one or more previous blocks. For example, the BV value of a neighboring block (*e.g.*, block left of the current block) can be used to determine a BV predictor for the BV value of the current block. If the BV value of the current block is (-80, -24) and the BV predictor is (-80, -32), the BV differential of (0, 8) is entropy encoded. Or, the BV predictor for the BV value of the current block can be the component-wise median or average of the BV values of multiple neighboring blocks (*e.g.*, blocks to the left, above and above-left of the current block).

[0138] During decoding, a decoder receives and entropy decodes the entropy coded BV differential, if any, for a BV value. The decoder also determines a BV predictor for the BV value. (The BV predictor determined by the decoder is the same as the BV predictor determined by the encoder.) The reconstructed BV difference, if any, is then combined with the BV predictor.

[0139] The encoder and decoder can use a default BV predictor when an actual BV value from a previous block is not available (*e.g.*, when determining a BV value for the first intra BC-predicted block in a given CTU). For example, the default BV predictor can be (0, 0). Or, the default BV predictor can be (-*W*, 0) or (-2**W*, 0), where *W* is the width of the current block. Or, the default BV predictor can be (0, -*H*) or (0, -2**H*), where *H* is the height of the current block. Compared to a zero-value default BV predictor, a default BV predictor with a non-zero component tends to be closer to the BV value of the current block, which results more efficient entropy coding of the BV differential. Or, a component of the default BV predictor can have a fixed non-zero value (*e.g.*, 8 or 16).

[0140] In any case, when the BV value that is possible for the current block is constrained to fall within a specific search range (*e.g.*, the current CTB and CTB to the left of the current CTB, as described in the previous section), and BV values cannot result in overlap between a current block and its intra-prediction region, in some cases the BV value for the current block can only have one possible value. For example, if (1) the current CTB has a single CB as the current block, (2) the search range for BV values is the current CTB and CTB to its left, and (3) overlapping between the current block and its intra-prediction region is not allowed, then the BV value for the single CB must have the BV value of (-*W*, 0). Other BV values either reference locations outside the search range or reference locations within the current block. In this situation, the encoder can skip signaling of the BV value (or BV differential) for the current block. The decoder can check conditions to identify the situation and infer the BV value of (-*W*, 0) without parsing and decoding any BV value (or BV differential) for the current block.

[0141] Or, an encoder and decoder determine one or more candidate BV values for a current block among the BV values used for reconstructed blocks that spatially neighbor the current block (*e.g.*, block to the left of the current block, block above the current block, and so on). The candidate BV value(s) can also include one or more BV values used for reconstructed blocks that temporally neighbor the current block, where a temporally neighboring block is at a corresponding position as the current block in another picture (*e.g.*, same position or overlapping position). The list of candidate BV value(s) can be determined by rules during encoding and decoding to eliminate redundant BV values. During encoding, the encoder can signal one or more syntax elements indicating which of the candidate BV value(s) to use as the BV predictor for the current block. In some modes, that BV predictor can be used as the BV value for the current block, which effectively "merges" the BV value of the current block with the BV value the neighbor

providing the candidate BV value. Or, the encoder can determine and encode a BV difference based on the BV value and BV predictor. During decoding, the decoder can receive one or more syntax elements indicating which of the candidate BV value(s) to use as the BV predictor for the current block. In some modes, that BV predictor can be used as the BV value for the current block, which effectively "merges" the BV value of the current block with the BV value the neighbor providing the candidate BV value. Or, the decoder can receive and decode a BV difference, which the decoder combines with the BV predictor to reconstruct the BV value. A BV "skip" or BV "direct" mode can be provided in which the BV predictor (selected by rule) is used as the BV value of the current block, with no residual values signaled for the current block. Examples of BV prediction using a ping-pong approach, in which an encoder or decoder selects between a pair of candidate BV values when predicting the BV value for a block, are described below.

D. Example Implementation Combining Features of Intra BC Prediction.

[0142] As noted, the preceding features of intra BC prediction can be used separately and individually. Or, the preceding features of intra BC prediction can be used can be used in combination.

[0143] For example, in one combined implementation that generally follows HEVC syntax, a BV value is signaled for a PU (which can be a CU, or part of a CU). The PU can include one or more TUs. Intra BC prediction processes operate at the level of TBs, on a TB-by-TB basis, using the BV value signaled for the PU. (All TBs use the same BV value, and intra BC prediction for a current TB can use reconstructed sample values of other, earlier TBs in the same CU). The BV value can be predicted using the BV values of one or more neighboring PUs (*e.g.*, using a ping-pong approach to BV prediction). The selection of BV values is constrained: (a) such that the encoder is prohibited from selecting BV values that would cause any sample values to be referenced that lie within areas that have not yet been encoded/reconstructed (*i.e.*, the sample values of an intra-prediction region for a current TB must be in areas covered by *other* TBs that precede the current TB in decoding/bitstream order; that is, for a given TB, the BV value is constrained to reference a region that is outside of the TB); (b) to reduce the necessary memory capacity in the decoder (*e.g.*, by constraining references according to BV values to be within the current CTB and one or two CTBs to the left of the current CTB); and (c) to prohibit references according to BV values from being outside the current slice, outside the current tile or outside the picture.

VIII. Innovations in Block Vector Prediction.

[0144] According to one aspect of the innovations described herein, an encoder and decoder use block vector ("BV") prediction according to a "ping-pong" approach. In particular, example ping-pong approaches to BV prediction improve coding efficiency when encoding screen content video in some scenarios.

[0145] In the example ping-pong approaches, the encoder and decoder maintain a pair of candidate BV values as state information. Figure 11 shows a pair of candidate BV values in BV buffers (1100) that can be used in the example ping-pong approaches. In the BV buffers (1100), BV buffer 0 stores a first initial candidate BV value, which is labeled BV_{init_0}. BV buffer 1 stores a second initial candidate BV value, which is labeled BV_{init_1}. In general, the BV value variable for BV buffer 0 can be labeled PBV0, and the BV value variable for BV buffer 1 can be labeled PBV1.

[0146] Figure 12 shows another data structure (1200) that can store a pair of candidate BV values according to the example ping-pong approaches. The data structure (1200) stores a first initial candidate BV value, which is labeled BV_{init_0} and is associated with the index value idx0. A second initial candidate BV value (BV_{init_1}) is associated with the index value idx1 in the data structure (1200). In general, the BV value variable associated with idx0 can be labeled PBV0, and the BV value variable associated with idx1 can be labeled PBV1.

[0147] Alternatively, the encoder and decoder use another data structure to track a pair of candidate BV values for BV prediction according to a ping-pong approach.

[0148] According to the example ping-pong approaches, from a pair of candidate BV values, an encoder selects the BV predictor to use for a BV value of a PU. For example, when the PU will be encoded using intra BC prediction with the selected candidate BV value, the encoder selects the candidate BV value for which the referenced intra-prediction region most closely matches the PU by some metric (*e.g.*, sum of absolute difference, mean squared error). Or, when the PU has a BV value (identified through BV estimation) that will be used for intra BC prediction, the encoder selects the candidate BV value that most closely matches the BV value for the PU. This selection results in the smallest BV differentials, which tends to improve the efficiency of entropy coding.

[0149] For a BV value, the encoder signals a flag value that indicates the selected one of the pair of candidate BV values, so as to indicate to a decoder which of the pair of candidate BV values should be used as the BV predictor. The flag value can be entropy coded or signaled as a fixed-length value. The flag value can be (conditionally) signaled

in the bitstream as a separate syntax element, or the flag value can be signaled in the bitstream jointly with another syntax element.

[0150] When each candidate BV value in the pair of candidate BV values is equally likely to be the BV predictor, the flag value can efficiently be signaled as a fixed-length value (*e.g.*, 1 bit), or using binary arithmetic coding in bypass mode (assuming the two possible values are equally probable). A first value (*e.g.*, 0) indicates selection of the first candidate BV value, and a second value (*e.g.*, 1) indicates selection of the second candidate BV value. The flag value can also be signaled as a fixed-length value when one of the pair of candidate BV values is more likely than the other, but coding efficiency may be reduced.

[0151] When one of the candidate BV values in the pair is more likely to be selected as BV predictor, it may be more efficient to use context-based arithmetic coding to signal flag values. For example, the probability content for the flag values can be initialized to equal probability for the two possible flag values when coding or decoding begins for a slice, then updated during coding / decoding to account for occurrences of the different flag values. Having a separate context for BV predictor flag values may improve coding efficiency, but adds to the complexity of encoding and decoding due to context modeling, etc. Typically, the candidate BV value that is expected to be selected more frequently is stored in a first BV buffer (*e.g.*, BV buffer 0) and indicated with a first flag value, and the candidate BV value that is expected to be selected less frequently is stored in the other BV buffer (*e.g.*, BV buffer 1) and indicated with the other flag value. Thus, flag values indicating the first BV buffer can, on average, be encoded using a smaller number of fractional bits in arithmetic coding. For some content, however, the more common candidate BV values may be stored in the other BV buffer (*e.g.*, BV buffer 1), with the context modeling accounting for the change in BV predictor distribution.

[0152] A decoder parses/decodes the flag value for a BV value and uses the flag value to determine which of the pair of candidate BV values is the BV predictor for the BV value. The decoder uses the selected candidate BV value as the BV predictor.

[0153] The encoder and decoder track and update which BV values are in the pair of candidate BV values. In previous ping-pong approaches, when a new BV value is produced (or reconstructed), the older candidate BV value (PBV1) in the pair is replaced by the newer BV value (PBV0) in the pair, and the newer BV value (PBV0) in the pair is replaced by the just produced / reconstructed BV value (BV_{new}). (That is, PBV1 = PBV0,

and PBV0 = BV_{new}.) The pair of candidate BV values is updated as each BV value is produced or reconstructed.

[0154] In example ping-pong approaches described herein, an encoder and decoder can automatically update the pair of candidate BV values to include a BV value after the BV value is produced or reconstructed. This can result in the pair of candidate BV values having two identical candidate BV values, however. In this situation, the encoder (recognizing that both candidate BV values are identical) can skip signaling of a flag value that indicates the selection between the pair of candidate BV values, and a decoder (recognizing that both candidate BV values are identical) can skip parsing of the flag value, instead simply using either of the identical candidate BV values as the BV predictor.

[0155] Or, in example ping-pong approaches described herein, the encoder and decoder can selectively update the pair of candidate BV values (and BV buffers) to avoid having identical candidate BV values. The encoder or decoder updates the pair of candidate BV values only when the new BV value does not equal the candidate BV value that will remain in the pair. If the new BV value equals the candidate BV value that will remain in the pair, the encoder or decoder skips the update operation. For example, the encoder or decoder selectively updates the pair of candidate BV values (PBV0 and PBV1) to include the new BV value BV_{new} as follows.

```
if (BVnew != PBV0) {
        PBV1 = PBV0;
        PBV0 = BVnew;
} else {
        do nothing;
}
```

By testing the condition BV_{new} != PBV0, the encoder and decoder avoid the situation where PBV0 equals PBV1, in which case the flag value indicating the BV predictor is wasted. On the other hand, checking the condition adds complexity to BV prediction, which might not be acceptable given resource requirements or justified by gain in coding efficiency.

[0156] For example, suppose PBV0 is (-12, 0) and PBV1 is (-16, 0). If BV_{new} is (-17, 0), (-12, 0) replaces (-16, 0) for PBV1, and (-17, 0) replaces (-12, 0) for PBV0. The previous BV value (-16, 0) from PBV1 is discarded. On the other hand, if BV_{new} is (-12, 0), PBV0 and PBV1 are unchanged, since BV_{new} matches PBV0. If BV_{new} matches PBV1

(both (-16, 0), in this example), the order of PBV0 and PBV1 is effectively switched – (-12, 0) replaces (-16, 0) for PBV1, and (-16, 0) (for BV_{new}) replaces (-12, 0) for PBV0.

[0157] In example ping-pong approaches described herein, for CUs that have four PUs, an encoder and decoder use a different pattern when updating the pair of candidate BV values used for BV prediction. Conceptually, the different pattern uses CU-level updating of the pair of candidate BV values, as opposed to PU-level updating. After BV prediction is finished for a CU, the pair of candidate BV values includes the BV values of the first two PUs of the CU – the BV value for PU1 in BV buffer 0, and the BV value for PU0 in BV buffer 1. In contrast, according to previous ping-pong approaches, after BV prediction is finished for CU, the pair of candidate BV values includes the BV values of the last two PUs of the CU, which could be PU3 and PU2.

[0158] Alternatively, the encoder and decoder can use a different pattern when updating the pair of candidate BV values used for BV prediction. If flag values are coded using fixed-length values, for example, the encoder and decoder can "toggle" (alternate) between updating two BV buffers with the BV value that was just produced or reconstructed. In this way, the encoder and decoder can use a single operation to replace the older candidate BV value (instead of shifting the newer candidate BV value to the spot of the older candidate BV value, then adding the new BV value to the previous spot of the newer candidate BV value), but the encoder and decoder need to track which BV buffer should be updated next. Also, toggling may hurt coding efficiency for flag values by interfering with context-adaptive binary arithmetic coding.

[0159] When no actual BV values are available for use as candidate BV values (*e.g.*, before coding or decoding the first CU of a CTU), the pair of candidate BV values can be initialized to default values. For example, both of the candidate BV values are initialized to $(-2W, 0)$, where W is the width of the CU. Or, one of the candidate BV values can be initialized to $(-2W, 0)$, and the other candidate BV value can be initialized to $(0, -2H)$, where H is the height of the CU (and may be the same as W). Or, a candidate BV value can be initialized to another value such as $(-W, 0)$ or $(0, -H)$. Or, for the sake of simplicity, instead of using a default value that depends on actual CU size, a candidate BV value can be initialized to a fixed value such as (-8, 0), (-16, 0), (0, -8) or (0, -16), or to a value signaled in the bitstream. This may be appropriate, for example, when CUs of size 8 or 16 are most common for intra BC prediction. In any case, after the start of BV prediction, actual BV values replace the default BV values in the pair of candidate BV values.

[0160] In all of the following examples, unless otherwise indicated, BV buffer 0 stores a "favored" candidate BV value, which is more likely to be selected than the other candidate BV value stored in BV buffer 1. Context-adaptive binary arithmetic coding can exploit the higher probability of the more common BV predictor flag value to improve coding efficiency for the flag values.

A. Example BV Prediction for 2*N*x2*N* PU of a 2*N*x2*N* CU.

[0161] Figure 13 shows an example of BV prediction for a 2*N*x2*N* PU of a 2*N*x2*N* CU according to example ping-pong approaches. The 2*N*x2*N* CU includes a single 2*N*x2*N* PU (PU0) whose BV is predicted using a pair of candidate BV values.

[0162] In Figure 13, BV buffer 0 initially stores a first initial candidate BV value BV_{init_0} (PBV0), and BV buffer 1 initially stores a second initial candidate BV value BV_{init_1} (PBV1). The BV value for the single 2*N*x2*N* PU (PU0) is predicted using either PBV0 or PBV1, as indicated by a flag value for the 2*N*x2*N* PU. The BV value for PU0 is processed using the selected BV predictor (*e.g.*, during encoding or decoding).

[0163] The pair of candidate BV values is then updated as follows. PBV1 is replaced with PBV0, and PBV0 is replaced with the BV value for PU0. That is, PBV1 = PBV0, and PBV0 = BV PU0. In Figure 13, BV buffer 1 is updated to store BV_{init_0}, and BV buffer 0 is updated to store the BV value for PU0 (BV_{PU0}). Alternatively, as explained above, the encoder or decoder selectively updates the pair of candidate BV values (and BV buffers) depending on whether the new BV value BV_{new} (here, the BV_{PU0}) equals PBV0.

B. Example BV Prediction for *N*x*N* PUs of a 2*N*x2*N* CU.

[0164] Figure 14 shows an example of BV prediction for four *N*x*N* PUs of a 2*N*x2*N* CU according to example ping-pong approaches. The 2*N*x2*N* CU includes four *N*x*N* PUs (PU0, PU1, PU2 and PU3) whose BV values are predicted using a pair of candidate BV values, which is updated during the BV prediction process for the CU.

[0165] In Figure 14, BV buffer 0 initially stores a first initial candidate BV value BV_{init_0} (PBV0), and BV buffer 1 initially stores a second initial candidate BV value BV_{init_1} (PBV1). The BV value for the first *N*x*N* PU (PU0) is predicted using either PBV0 or PBV1, as indicated by a flag value for PU0. The BV value for PU0 (BV_{PU0}) is processed using the selected BV predictor (*e.g.*, during encoding or decoding).

[0166] The pair of candidate BV values is then updated by replacing PBV1 with PBV0, and by replacing PBV0 with the BV value for PU0. That is, PBV1 = PBV0, and PBV0 = BV_{PU0}. In Figure 14, BV buffer 1 is updated to store BV_{init_0}, and BV buffer 0 is updated to store BV_{PU0}. Alternatively, as explained above, the encoder or decoder

selectively updates the pair of candidate BV values (and BV buffers) depending on whether the new BV value BV_{new} (here, the BV_{PU0}) equals PBV0.

[0167] Next, the BV value for the second *N*x*N* PU (PU1) is predicted using either PBV0 (in Figure 14, BV_{PU0} of its left neighbor) or PBV1 (in Figure 14, BV_{init_0}), as indicated by a flag value for PU1. The BV value for PU1 (BV_{PU1}) is processed using the selected BV predictor (*e.g.*, during encoding or decoding).

[0168] The pair of candidate BV values is then updated by replacing PBV1 with PBV0, and by replacing PBV0 with the BV value for PU1. That is, PBV1 = PBV0, and PBV0 = BV_{PU1}. In Figure 14, BV buffer 1 is updated to store BV_{PU0}, and BV buffer 0 is updated to store BV_{PU1}. Alternatively, as explained above, the encoder or decoder selectively updates the pair of candidate BV values (and BV buffers) depending on whether the new BV value BV_{new} (here, the BV_{PU1}) equals PBV0.

[0169] Next, the BV value for the third *N*x*N* PU (PU2) is predicted using either PBV0 (in Figure 14, BV_{PU1} of its top-right neighbor) or PBV1 (in Figure 14, BV_{PU0} of its top neighbor), as indicated by a flag value for PU2. For this flag value, BV buffer 1 stores the favored candidate BV value, not BV buffer 0. This adjustment accounts for BV_{PU0} (top neighbor) being a more likely candidate than BV_{PU1} (top-right neighbor). The BV value for PU2 (BV_{PU2}) is processed using the selected BV predictor (*e.g.*, during encoding or decoding). Afterwards, the candidate BV values in BV buffer 0 (in Figure 14, BV_{PU1}) and BV buffer 1 (in Figure 14, BV_{PU0}) are unchanged.

[0170] Finally, the BV value for the fourth *N*x*N* PU (PU3) is predicted using either the BV value for PU2 (BV_{PU2}), which is buffered in another location, or PBV1 (in Figure 14, BV_{PU1} of its top neighbor), as indicated by a flag value for PU3. For this flag value, BV buffer 0 again stores the favored candidate BV value, although the other candidate BV value is not stored in BV buffer 1. The BV value for PU3 (BV_{PU3}) is processed using the selected BV predictor (*e.g.*, during encoding or decoding). Afterwards, the candidate BV values in BV buffer 0 and BV buffer 1 are unchanged. In Figure 14, at the end of BV prediction for the four *N*x*N* PUs of the CU, the pair of candidate BV values is BV_{PU1} (in BV buffer 0) and BV_{PU0} (in BV buffer 1).

[0171] In contrast, according to previous ping-pong approaches, for BV prediction for *N*x*N* PUs of a 2*N*x2*N* CU, when a BV value is produced or reconstructed, the candidate BV value in BV buffer 0 is moved to BV buffer 1, and the just produced / reconstructed BV value is stored in BV buffer 0. At the end of BV prediction for four *N*x*N* PUs of a 2*N*x2*N* CU, the pair of candidate BV values would be BV_{PU3} (in BV buffer 0) and BV_{PU2}

(in BV buffer 1). There are several disadvantages to the previous ping-pong approaches, in practice. If the next CU has a single 2*N*x2*N* PU, BV_{PU0} of the current CU usually would be the best BV predictor, but BV_{PU0} of the current CU is not retained as one of the candidate BV values. Or, if the next CU has two *N*x2*N* PUs, BV_{PU0} and BV_{PU1} of the current CU usually would be the best BV predictors for PU0 and PU1 of the next CU, respectively, but BV_{PU0} and BV_{PU1} of the current CU are not retained as the candidate BV values.

[0172] According to the example ping-pong approaches described herein, however, after BV prediction for *N*x*N* PUs of a 2*N*x2*N* CU (as shown in Figure 14), the pair of candidate BV values is BV_{PU1} (in BV buffer 0) and BV_{PU0} (in BV buffer 1). This retains the candidate BV values that are likely to be most useful for BV prediction for the next CU.

C. Example BV Prediction for *N*x2*N* PUs or 2*N*x*N* PUs of a 2*N*x2*N* CU.

[0173] Figure 15 shows an example of BV prediction for two *N*x2*N* PUs or two 2*N*x*N* PUs of a 2*N*x2*N* CU according to example ping-pong approaches. The 2*N*x2*N* CU includes two *N*x2*N* PUs (left PU0 and right PU1) or two 2*N*x*N* PUs (top PU0 and bottom PU1), whose BV values are predicted using a pair of candidate BV values, which is updated during the BV prediction process for the CU.

[0174] In Figure 15, BV buffer 0 initially stores a first initial candidate BV value BV_{init_0} (PBV0), and BV buffer 1 initially stores a second initial candidate BV value BV_{init_1} (PBV1). The BV value for first PU (PU0) is predicted using either PBV0 or PBV1, as indicated by a flag value for PU0. The BV value for PU0 (BV_{PU0}) is processed using the selected BV predictor (*e.g.*, during encoding or decoding).

[0175] The pair of candidate BV values is then updated by replacing PBV1 with PBV0, and by replacing PBV0 with the BV value for PU0. That is, PBV1 = PBV0, and PBV0 = BV_{PU0}. In Figure 15, BV buffer 1 is updated to store BV_{init_0}, and BV buffer 0 is updated to store BV_{PU0}. Alternatively, as explained above, the encoder or decoder selectively updates the pair of candidate BV values (and BV buffers) depending on whether the new BV value BV_{new} (here, the BV_{PU0}) equals PBV0.

[0176] Next, the BV value for the second PU (PU1) is predicted using either PBV0 (in Figure 15, BV_{PU0} of its neighbor) or PBV1 (in Figure 15, BV_{init_0}), as indicated by a flag value for PU1. The BV value for PU1 (BV_{PU1}) is processed using the selected BV predictor (*e.g.*, during encoding or decoding).

[0177] The pair of candidate BV values is then updated by replacing PBV1 with PBV0, and by replacing PBV0 with the BV value for PU1. That is, PBV1 = PBV0, and PBV0 = BV_{PU1}. In Figure 15, BV buffer 1 is updated to store BV_{PU0}, and BV buffer 0 is updated to store BV_{PU1}. Alternatively, as explained above, the encoder or decoder selectively updates the pair of candidate BV values (and BV buffers) depending on whether the new BV value BV_{new} (here, the BV_{PU1}) equals PBV0.

D. Example BV Prediction Techniques According to Ping-Pong Approaches.

[0178] Figure 16 shows an example technique for predicting BV values for a CU with multiple PUs according to example ping-pong approaches. The CU is a set of one or more blocks for purposes of encoding and decoding, generally, and each of the PUs is a set of one or more blocks, within the CU, for purposes of signaling of prediction information and/or prediction processing. An image encoder or video encoder such as one described with reference to Figure 3 or Figures 5a-5b, or other encoder, can perform the technique (1600). Or, an image decoder or video decoder such as one described with reference to Figure 4 or Figure 6, or other decoder, can perform the technique (1600).

[0179] To start, the pair of candidate BV values is a first initial candidate BV value BV_{init_0} (PBV0) and a second initial candidate BV value BV_{init_1} (PBV1), which are stored in BV buffers 0 and 1, respectively.

[0180] For each of multiple PUs of a current CU of a current picture, the encoder or decoder predicts (1610) a BV value for the PU using one of a pair of candidate BV values. A flag value indicates the selection between the pair of candidate BV values. The pair of candidate BV values can be different for different PUs of the CU. For example, when the current CU includes four PUs, for the first PU, the pair of candidate BV values PBV0 and PBV1 can be BV_{init_0} and BV_{init_1}, respectively. For the second PU, the pair of candidate BV values PBV0 and PBV1 can be the BV for the first PU (BV_{PU0}) and BV_{init_0}, respectively. For the third PU, the pair of candidate BV values PBV0 and PBV1 can be the BV value for the second PU (BV_{PU1}) and BV_{PU0}, respectively. Finally, for the fourth PU, the pair of candidate BV values can be BV_{PU1} and the BV value for the third PU (BV_{PU2}).

[0181] The encoder or decoder then processes (1620) the BV value for the PU using the predicted BV value for the PU. For example, during encoding, the encoder determines a BV differential value using the BV value for the PU and the predicted BV value. Or,

during decoding, the decoder reconstructs the BV value for the PU based at least in part on the predicted BV value.

[0182] The BV buffers can be updated automatically or selectively (depending on whether a new BV value BV_{new} equals PBV0). For example, (as shown in Figure 14), when the current CU includes four PUs, before the BV prediction for the first PU (PU0), BV buffer 0 stores BV_{init_0}, and BV buffer 1 stores BV_{init_1}. Before the BV prediction for the second PU (PU1), BV buffer 0 stores BV_{PU0}, and BV buffer 1 stores BV_{init_1}. Before the BV prediction for the third PU (PU2), BV buffer 0 stores BV_{PU1}, and BV buffer 1 stores BV_{PU0}. After the processing for PU2, BV buffer 0 retains BV_{PU1}, and BV buffer 1 retains BV_{PU0}.

[0183] After BV prediction for the current CU, when updates to the BV buffers have occurred, the pair of candidate BV values is BV_{PU1} and BV_{PU0} of the current CU. These will be the initial candidate BV values for BV prediction for a subsequent CU of the current picture. Thus, when the current CU includes four PUs, the final pair of candidate BV values can include the BV values for the first and second PUs of the current CU, even though the BV values for the first and second PUs of the current CU were processed before the BV values for the third and fourth PUs of the current CU.

[0184] Figure 17 shows another example technique for predicting BV values for CUs according to example ping-pong approaches, with selective updating of BV buffers. An image encoder or video encoder such as one described with reference to Figure 3 or Figures 5a-5b, or other encoder, can perform the technique (1700). Or, an image decoder or video decoder such as one described with reference to Figure 4 or Figure 6, or other decoder, can perform the technique (1700).

[0185] The encoder or decoder processes multiple CUs of a current picture. For example, the encoder processes the multiple CUs as part of encoding and outputs encoded data as part of a bitstream. Or, the decoder processes the multiple CUs as part of decoding and outputs reconstructed sample values. Any given CU of the multiple CUs can include a single PU or multiple PUs.

[0186] To start, the pair of candidate BV values is a first initial candidate BV value BV_{init_0} (PBV0) and a second initial candidate BV value BV_{init_1} (PBV1), which are stored in BV buffers 0 and 1, respectively.

[0187] For a given CU of the multiple CUs, the encoder or decoder checks (1710) whether the given CU has a single PU or multiple PUs. If the given CU has a single PU, the encoder or decoder predicts (1720) a BV value for the single PU using one of a pair of

candidate BV values PBV0 and PBV1 (in Figure 17, BV_{init_0} and BV_{init_1}), where a flag value indicates the selection between the pair of candidate BV values. The encoder or decoder processes (1730) the BV value for the single PU using the predicted BV value. The encoder or decoder selectively updates BV buffers 0 and 1 depending on whether the BV value for the PU (BV_{PU0}) equals the BV value stored in BV buffer 0 (BV_{init_0}). For example, the encoder or decoder compares BV_{PU0} and the BV value stored in BV buffer 0 (BV_{init_0}). If BV_{PU0} is different than the BV value stored in BV buffer 0 (BV_{init_0}), the encoder or decoder stores the BV value from BV buffer 0 (BV_{init_0}) in BV buffer 1, and stores BV_{PU0} in BV buffer 0.

[0188] After the processing (1730), when updates to the BV buffers have occurred, BV buffer 0 stores the BV value for the single PU (BV_{PU0}), and BV buffer 1 stores BV_{init_0}. The encoder or decoder checks (1740) whether to continue with the next CU of the picture. If so, the pair of buffered BV values is treated as the initial candidate BV values BV_{init_0} and BV_{init_1} for the next CU.

[0189] Otherwise, when the given CU has multiple PUs, the encoder or decoder performs BV prediction and processing for the PUs of the given CU. For each of the multiple PUs of the given CU, the encoder or decoder predicts (1750) a BV value for the PU using one of a pair of candidate BV values PBV0 and PBV1 (initially BV_{init_0} and BV_{init_1}, but later updated). A flag value for the PU indicates the selection between the pair of candidate BV values. The encoder or decoder processes (1760) the BV value for the PU using the predicted BV value. If the PU is the first or second PU of the given CU, the encoder or decoder selectively updates BV buffers 0 and 1 depending on whether the BV value for the PU (BV_{PUx}) equals the BV value stored in BV buffer 0 (PBV0). For example, the encoder or decoder compares BV_{PUx} and PBV0. If BV_{PUx} is different than PBV0, the encoder or decoder stores PBV0 in BV buffer 1, and stores BV_{PUx} in BV buffer 0.

[0190] The encoder or decoder checks (1770) whether the PU is the last PU in the given CU. If not, the encoder or decoder continues with BV prediction (1750) for the next PU of the given CU. For example, the PUs are four *N*x*N* blocks, two *N*x2*N* blocks or two 2*N*x*N* blocks of a 2*N*x2*N* CU.

[0191] After the processing (1760) for the last PU of the given CU, when both updates to the BV buffers have occurred, BV buffer 0 stores the BV value for the second PU (BV_{PU1}) of the given CU, and BV buffer 1 stores the BV value for the first PU (BV_{PU0}) of the given CU. The encoder or decoder checks (1740) whether to continue with the next

CU of the picture. If so, the pair of buffered BV values BV_{PU1} and BV_{PU0} is treated as the initial candidate BV values BV_{init_0} and BV_{init_1} for the next CU.

IX. Encoder-side Options for Intra Block Copy Prediction Mode.

[0192] This section presents various innovations for encoder-side options for intra block copy ("BC") prediction. Some of the innovations relate to concurrently performing block vector ("BV") estimation and making block splitting decisions for a block. Other innovations relate to selectively merging blocks into a larger block during BV estimation. Still other innovations relate to estimation of sample values within an overlap area of a current block during BV estimation. In general, these innovations improve coding efficiency of intra BC prediction.

[0193] As part of BV estimation, the encoder can use any of several approaches. The encoder can use a full search, evaluating every candidate BV value allowed in a search range. Or, the encoder can use a partial search, evaluating only some of the candidate BV values allowed in a search range. For example, the encoder can start a partial search at the predicted BV value for a current block (*e.g.*, predicted based on BV values of one or more neighboring blocks). After evaluating the candidate BV value at the starting position for the partial search, the encoder can evaluate one or more other candidate BV values at increasing distances from the starting position (*e.g.*, according to a spiral search pattern or some other pattern). When evaluating a given candidate BV value, the encoder can compare all sample values in the intra-prediction region and current block. Or, the encoder can evaluate a subset of the sample values (that is, sub-sample which values are evaluated). When comparing sample values between the intra-prediction region and current block to determine a distortion cost, the encoder can compute mean square error, sum of squared differences, sum of absolute differences, or some other measure of distortion. The encoder can also determine a rate cost associated with encoding of the candidate BV value.

[0194] During encoding, at various stages, an encoder can use rate-distortion optimization ("RDO") in which the rate cost and distortion cost of various options are evaluated. In general, rate-distortion cost for an option is given by $D+\lambda R$ (or $R+\lambda D$), where R represents the rate cost in terms of bits of encoded data, D represents the distortion cost using a metric such as mean squared error or a perceptual distortion measure, and λ is a Lagrangian multiplier (example of weighting parameter) that weights the rate cost R compared to the distortion cost D. The encoder typically selects the option that gives the lowest rate-distortion cost.

A. Selectively Merging Blocks into Larger Block During BV Estimation.

[0195] During BV estimation, an encoder can merge blocks into a larger block. In this way, the encoder can reduce the number of bits used to signal BV information. In particular, when intra BC prediction is performed on a TB-by-TB basis, the encoder can perform BV estimation on a CB-by-CB basis then merge small CBs into a larger CB, relying on splitting of the CB into smaller TBs during intra BC prediction to avoid overlap between a TB and its intra-prediction region.

[0196] Figure 18 shows a technique (1800) for selectively merging blocks into a larger block during BV estimation. The technique (1800) can be performed by an encoder such as one described with reference to Figure 3 or Figures 5a and 5b, or by another encoder.

[0197] According to the technique, when the encoder encodes data for a picture using intra BC prediction, for each of multiple blocks with a first size, the encoder identifies (1810) a BV value using BV estimation. The first size can be 8x8, 16x16, 32x32 or some other size. In general, for each of the blocks with the first size, the BV estimation includes determining a cost for a candidate BV value. The cost can include a rate cost and/or a distortion cost. The identified BV value references an intra-prediction region that does not overlap the block with the first size.

[0198] The encoder selectively merges (1820) two or more of the blocks into a block with a second size larger than the first size. The second size can be 16x16, 32x32, 64x64 or some other size. For example, the encoder compares the BV values of two or more adjacent blocks among the multiple blocks with the first size and, if the compared BV values are identical, merges the two or more adjacent blocks into the block with the second size. The block with the second size is assigned the identical BV value. Unlike the blocks with the first size, the BV value can reference an intra-prediction region that overlaps the block with the second size.

[0199] The encoder can repeat the technique (1800) for another block. Eventually, the encoder outputs the encoded data for the picture.

[0200] In some example implementations, each of the multiple blocks with the first size is a CB, and the block with the second size is also a CB. Intra BC prediction, however, is performed on a TB-by-TB basis. The BV estimation can use a full search of a search range for each of the multiple blocks with the first size. Or, the BV estimation can use a partial search of a search range for each of the multiple blocks with the first size, for example, starting at a predicted BV value for the block, and potentially terminating the

partial search based at least in part on the cost for a candidate BV value that has been evaluated.

[0201] Figure 19 illustrates an advantage of selective merging (1900) of blocks into a larger block during BV estimation. In Figure 19, four blocks numbered 0, 1, 2 and 3 (indicated with light solid lines) have BV values that reference corresponding intra-prediction regions 0', 1', 2' and 3', respectively. That is, the BV value references intra-prediction region 0' for block 0, references intra-prediction region 1' for block 1, and so on. The intra-prediction regions are indicated with dotted lines. None of the blocks 0, 1, 2 or 3 overlaps its corresponding intra-prediction region. The larger block that includes blocks 0, 1, 2 and 3 (indicated with a heavy solid line) does overlap its corresponding intra-prediction region, however. If the encoder were to perform BV estimation for the larger block, when overlap between intra-prediction region and block is prohibited, the encoder would not be able to identify the BV value shown in Figure 19 as an allowable BV value. In contrast, when the encoder performs BV estimation for the smaller blocks 0, 1, 2 and 3, the encoder identifies the same BV value for all four blocks, and the encoder can merge the four blocks into the larger block with the same BV value. During encoding or decoding, the larger block will be split into smaller blocks for the actual intra BC prediction operations. For example, a CB having a given size is split into smaller TBs for performance of the intra BC prediction operations on a TB-by-TB basis, with no overlap between a TB and its intra-prediction region.

B. Concurrent Block Vector Estimation and Block Splitting Decisions.

[0202] The encoder can perform BV estimation and block splitting decisions concurrently for a current block. In particular, when intra BC prediction operations are performed on a TB-by-TB basis, the encoder can concurrently evaluate a candidate BV value and possible block splitting decisions for a current block.

[0203] Figure 20 is a flowchart illustrating a generalized technique for concurrently performing BV estimation and making block splitting decisions for a block. The technique (2000) can be performed by an encoder such as one described with reference to Figure 3 or Figures 5a and 5b, or by another encoder.

[0204] When the encoder encodes data for a picture using intra BC prediction, the encoder encodes (2010) data for a current block. As part of the encoding (2010), the encoder performs BV estimation operations and block splitting decision operations concurrently. In some example implementations, the current block is a CB corresponding in size to a TB, and intra BC prediction is performed on a TB-by-TB basis. For example,

as part of the encoding (2010), the encoder performs the technique (2100) shown in Figure 21 to concurrently evaluate a candidate BV value and possible block splitting decisions for the current block.

[0205] Figure 21 is flowchart illustrating an example technique for concurrently evaluating a candidate BV value and block splitting decisions for a block. The technique (2100) can be performed by an encoder such as one described with reference to Figure 3 or Figures 5a and 5b, or by another encoder.

[0206] The encoder identifies (2110) a candidate BV value for the current block using BV estimation operations. The current block has a first size. For example, the first size is 8x8, 16x16, 32x32, 64x64 or some other size.

[0207] The encoder checks (2120) whether to split the current block. For example, the encoder evaluates whether intra BC prediction with the candidate BV value for the current block results in overlap between the current block and an intra-prediction region referenced by the candidate BV value.

[0208] Depending on the results of the check (2120), the encoder selectively splits (2130) the current block into multiple blocks each having a second size smaller than the first size. For example, the second size is 4x4, 8x8, 16x16, 32x32, or some other size. Then (the current block having been split), the encoder repeats (2140) the technique (2100) for each of the smaller blocks. That is, for each of the blocks having the second size, the encoder repeats the identifying, the evaluating, the selectively splitting, etc., with the block having the second size being treated as the current block.

[0209] On the other hand, if the current block is not split, the encoder encodes (2150) the current block and measures (2160) the cost of encoding the current block. For example, when it encodes the current block, the encoder (a) predicts sample values of the current block using intra BC prediction, (b) determines residual values for the current block using the predicted sample values and the sample values of the current block, (c) optionally applies a frequency transform to the residual values to produce transform coefficients and quantizes the transform coefficients, (d) optionally inverse quantizes the transform coefficients and applies an inverse frequency transform to reconstruct the residual values, and (e) combines the residual values and the predicted sample values. The measured cost can be a rate-distortion cost, rate cost or distortion cost. The rate cost can account for both cost of signaling the BV value and cost of signaling block splitting decision information.

[0210] Alternatively, the encoder evaluates a candidate BV value and possible block splitting decisions for the current block in some other way. For example, in addition to considering overlap between an intra-prediction region and block, the encoder also considers whether the block already has the smallest allowable size. If so, the block is not split, but the candidate BV value is disallowed.

[0211] Returning to Figure 20, as part of the encoding (2010), the encoder can record the cost for the candidate BV value. The encoder can then evaluate one or more other candidate BV values. For example, the encoder repeats the technique (2100) for another candidate BV value. Eventually, the encoder selects one or more BVs and makes block splitting decisions for the current block (and blocks within the current block) that result in lowest cost (*e.g.*, rate-distortion cost, rate cost or distortion cost).

[0212] The encoder outputs (2020) the encoded data for the block. The encoder can repeat the technique (2000) for another block.

C. Estimating Reconstructed Sample Values Within an Overlap Area.

[0213] In some examples described herein, BVs are signaled at the level of CUs and applied to blocks at the CB level. Overlap between a CB and corresponding intra-prediction region is not allowed. In other examples described herein, an encoder applies a BV value to smaller blocks on a TB-by-TB basis, even though the BV value is signaled at a higher syntax level (*e.g.*, for a CU). For example, a BV value is signaled for a CU with a 32x32 CB, but the BV value is applied to blocks at the level of 4x4 TBs or 8x8 TBs. This permits the BV value to have a smaller magnitude, while still not having overlap between the block at TB level and intra-prediction region that is copied.

[0214] When a given block is split into smaller blocks for purposes of intra BC prediction operations, the given block includes an overlap area in which sample values of the given block may be part of intra-prediction regions for one or more of the smaller blocks. This section describes approaches to estimating reconstructed sample values within the overlap area. In particular, this section describes approaches to estimating reconstructed sample values within an overlap area of a block when a BV value for the block is applied to smaller blocks on a TB-by-TB basis.

[0215] Figure 22a illustrates a current block (2210) having size $2m$ x $2n$. The current block (2210) includes four m x n blocks (2221, 2222, 2223, 2224) for purposes of intra BC operations. For example, the current block (2210) is an 8x8 block of a CU, and the four smaller blocks (2221, 2222, 2223, 2224) are 4x4 blocks with 4x4 TBs. Or, the current block (2210) is a 16x16 block of a CU, and the four smaller blocks (2221, 2222, 2223,

2224) are 8x8 blocks with 8x8 TBs. When intra BC prediction operations are applied to the *m* x *n* blocks in zigzag scan order (in Figure 22a, block 0, block 1, block 2, then block 3), the intra-prediction region for the first *m* x *n* block (2221) is outside the current 2*m* x 2*n* block (2210). For the second *m* x *n* block (2222), however, the intra-prediction region may overlap the first *m* x *n* block (2221) in part or completely. For the third m x n block (2223), the intra-prediction region may overlap the first *m* x *n* block (2221) or the second *m* x *n* block (2222) in part or completely. And for the fourth *m* x *n* block (2224), the intra-prediction region may overlap any of the first three *m* x *n* block (2221, 2222, 2223) of the current block (2210) in part or completely.

[0216] Figure 22b shows the overlap area (2230) of the current block (2210). The overlap area (2230) covers the first three *m* x *n* blocks (2221, 2222, 2223) in zigzag scan order, since sample values in these three *m* x *n* blocks might be part of an intra-prediction region for at least one *m* x *n* block of the current block (2210). For another size of smaller block (*e.g.*, 4x4 blocks of a 16x16 block or 32x32 block), the overlap area could cover even more of the current block.

[0217] During BV estimation, when an encoder evaluates a candidate BV value for a current block that is split into smaller blocks for purposes of intra BC prediction operations, the encoder can calculate actual reconstructed sample values within the overlap area as the encoder applies the candidate BV value across the respective smaller blocks in zigzag order. This can be computationally intensive, since it involves encoding operations and reconstruction operations applied per candidate BV value per smaller block. Instead, the encoder can estimate reconstructed sample values within the overlap area. In particular, when intra BC prediction operations are performed for smaller blocks on a TB-by-TB basis, the encoder can estimate reconstructed sample values within the overlap area of a current block that includes the multiple smaller blocks.

[0218] Figure 23 is flowchart illustrating an example technique for BV estimation in which an encoder estimates reconstructed sample values within the overlap area. The technique (2300) can be performed by an encoder such as one described with reference to Figure 3 or Figures 5a and 5b, or by another encoder.

[0219] For a current block of a picture, the encoder estimates (2310) reconstructed sample values within an overlap area of the current block. The current block has multiple smaller blocks. For example, the current block has a first size (*e.g.*, 32x32, 16x16 or 8x8), and each of the multiple blocks has a second size smaller than the first size (*e.g.*, 16x16, 8x8 or 4x4). The current block can be a block of a CU, in which case each of the multiple

smaller blocks can be at the level of a TB of a TU. The overlap area covers parts of the current block that are in potential intra-prediction regions for at least one of the multiple smaller blocks.

[0220] According to a first approach to estimating reconstructed sample values within the overlap area, the encoder uses corresponding original sample values as the reconstructed sample values, respectively, within the overlap area. Outside the current block, the BV estimation can use original sample values within the picture. Alternatively, outside the current block, the BV estimation can use reconstructed sample values within the picture (that is, reconstructed sample values outside the current block; original sample values within the overlap area of the current block). In this case, original sample values may be used in the intra-prediction region when no actual reconstructed sample values are available during BV estimation, but otherwise actual reconstructed sample values are used.

[0221] According to a second approach to estimating reconstructed sample values within the overlap area, the encoder processes the original sample values within the overlap area. In particular, the encoder applies a frequency transform, quantization, inverse quantization and an inverse frequency transform to the original sample values within the overlap area, yielding an approximation of the reconstructed sample values within the overlap area. The quantization and inverse quantization can use a quantization parameter that has been applied (or is expected to be applied) in a neighborhood around the current block.

[0222] According to a third approach to estimating reconstructed sample values within the overlap area, the encoder uses intra BC prediction operations when estimating the reconstructed sample values within the overlap area. The encoder predicts sample values within the overlap area using at least one BV predictor (such as a BV value for a previous block in the picture, *e.g.*, a neighbor block, or a default BV predictor). The encoder determines residual values based on differences between the predicted sample values and corresponding original sample values within the overlap area. To the residual values, the encoder applies a frequency transform, quantization, inverse quantization and an inverse frequency transform. The quantization and inverse quantization can use a quantization parameter that has been applied (or is expected to be applied) in a neighborhood around the current block. The encoder combines the results of the processing (that is, the reconstructed residual values) with the predicted sample values, which yields an approximation of the reconstructed sample values within the overlap area.

[0223] In one variation of the third approach, the encoder predicts sample values within the overlap area using at least one BV predictor (such as a BV value for a previous block in the picture, *e.g.*, a neighbor block, or a default BV predictor), but skips the processing of the residual values. The estimated reconstructed sample values in the overlap area are the predicted sample values.

[0224] In another variation of the third approach, the encoder can estimate whether the residual values for the overlap area will be significant (*e.g.*, have non-zero transform coefficient values after transform, quantization, etc.). For example, the encoder can evaluate whether the residual values have small magnitude, individually or collectively, or check some other condition for the residual values. If the residual values are not significant, the encoder can skip the residual processing and simply use the predicted sample values as the estimated reconstructed sample values in the overlap area. Otherwise (residual values are significant), the encoder can encode then reconstruct the residual values, then combine them with the predicted sample values to produce the estimated reconstructed sample values in the overlap area, as described above.

[0225] The encoder performs (2320) BV estimation to determine a BV value for the current block. The BV estimation uses at least some of the estimated reconstructed sample values within the overlap area of the current block. The BV estimation also uses reconstructed sample values (or original sample values, in one approach) within the picture outside the current block.

[0226] The BV estimation can include, for each of one or more candidate BV values for the current block, performing block matching operations between the smaller blocks, respectively, and candidate intra-prediction regions according to the candidate BV value. For example, for a given candidate BV value for a current block with four smaller blocks, the encoder performs block matching operations between the four smaller blocks and their corresponding candidate intra-prediction regions according to the given candidate BV value.

[0227] Alternatively, the BV estimation can include (1) for each of the multiple smaller blocks, determining a BV value for the smaller block, and then (2) synthesizing the BV values for the multiple smaller blocks into the BV value for the current block. For example, for a current block with four smaller blocks, the encoder determines, in succession, a BV value for the first smaller block, a BV value for the second smaller block, a BV value for the third smaller block and a BV value for the fourth smaller block. After the first smaller block, the BV estimation may include evaluation of intra-prediction

regions in the overlap area of the current block. The encoder then synthesizes the BV values for the four smaller blocks into the BV value for the current block. For example, the encoder can (a) average the BV values for the multiple smaller blocks, (b) calculate a median of the BV values for the multiple smaller blocks, (c) identify the BV value for the current block in a neighborhood around the BV values for the multiple smaller blocks, (d) identify the BV value for the current block using RDO for candidate BV values in a neighborhood around the BV values for the multiple smaller blocks, or (e) synthesize the BV values for the multiple smaller blocks into the BV value for the current block in some other way.

[0228] After the BV value for the current block has been identified by BV estimation, the encoder can perform intra BC prediction for the current block using the BV value for the current block. In doing so, the encoder performs BC operations on a block-by-block basis for the multiple smaller blocks. Thus, the encoder can perform BV estimation at the level of the current block but perform BV compensation at the level of the smaller blocks. For example, the encoder can perform BV estimation for an 8x8 block of an 8x8 CU, but perform intra BC operations (and reconstruction operations) on a TB-by-TB basis for four 4x4 blocks in zigzag scan order. The 4x4 blocks share the same BV value (of the 8x8 CU), and the encoder performs 4x4 transforms, quantization, inverse quantization, 4x4 inverse transforms, etc. to successive 4x4 blocks of residual values. The first 4x4 block (for TB 0) is reconstructed, then may be used in an intra-prediction region for the second 4x4 block (for TB 1). The second 4x4 block is reconstructed, then may be used in an intra-prediction region for the third 4x4 block (for TB 2). For the fourth 4x4 block (for TB 3), any of the first three 4x4 blocks may be used in an intra-prediction region.

[0229] At the decoder side, as described above, the decoder can check whether a BV value results in overlap between an intra-prediction region and current block. If so, the block (at TB level) is split into four smaller blocks (at the level of a smaller TB), then the four smaller blocks are successively decoded (using intra BC prediction and reconstruction operations) in zigzag order. Thus, the reconstructed sample values for one of the smaller blocks are available for use in the intra-prediction region for the next smaller block in zigzag order. If the transform block flag of a TU is zero, intra BC prediction is applied only at the CU level, and chroma de-correlation is disabled.

D. Alternatives and Variations.

[0230] Many of the preceding examples address BV values and intra BC prediction operations for luma blocks. For video in YUV 4:4:4 format, chroma components have the

same resolution as a luma component. The BV value used for a current luma block can be used for corresponding chroma blocks. When the current luma block is split into smaller luma blocks for intra BC prediction operations, the corresponding chroma blocks are also split into smaller chroma blocks in the same manner for intra BC prediction operations.

[0231] For video in YUV 4:2:0 format, chroma components are downsampled by a factor of two horizontally and by a factor of two vertically. For example, if the current luma block has size of 16x16, the corresponding chroma blocks have size of 8x8. The BV value used for a current luma block can be used for corresponding chroma blocks, after appropriate scaling and rounding (and/or truncation) operations. In general, when the current luma block is split into smaller luma blocks for intra BC prediction operations, the corresponding chroma blocks are also split into smaller chroma blocks in the same manner for intra BC prediction operations.

[0232] In some implementations, however, this is not always the case. In some implementations, if a luma TB of video in YUV 4:2:0 format has the minimum transform size, chroma TBs having that minimum transform size are used for the chroma counterparts for that luma TB *and* its three neighboring luma TBs for the same CB. For example, if the current luma TB has minimum size of 4x4 (and is associated with an 8x8 luma CB), corresponding chroma CBs have the size of 4x4. Each 4x4 chroma CB has a single 4x4 chroma TB. Chroma TBs with size 2x2, as might be expected with downsampling for the YUV 4:2:0 format, are not supported. In such implementations, splitting of the current luma block into smaller luma blocks for intra BC prediction operations is not supported if the smaller luma blocks would have the minimum transform size because, when applied to the corresponding chroma blocks, the splitting would result in chroma blocks smaller than the minimum transform size. In such implementations, splitting of the current luma block (and corresponding chroma blocks) into smaller blocks for intra BC prediction operations is still permitted for larger sizes.

[0233] For video in YUV 4:2:2 format, chroma components are downsampled by a factor of two horizontally. For example, if the current luma block has size of 16x16, the corresponding chroma blocks have size of 8x16. The BV value used for a current luma block can be used for corresponding chroma blocks, after appropriate scaling and rounding (and/or truncation) operations. In general, when the current luma block is split into smaller luma blocks for intra BC prediction operations, the corresponding chroma blocks can also be split into smaller chroma blocks in the same manner for intra BC prediction operations.

[0234] In some implementations, however, this is not always the case. In some implementations, if a luma TB of video in YUV 4:2:2 format has minimum transform size, chroma TBs having that minimum transform size are used for the chroma counterparts for that luma TB *and* a neighboring luma TB in the same CB. For example, if the current luma TB has minimum size of 4x4 (and is associated with an 8x8 CB), corresponding 4x8 chroma CBs each have two 4x4 chroma TBs. Chroma TBs with size 2x4, as might be expected with downsampling for the YUV 4:2:2 format, are not supported. Instead, each of the 4x4 chroma TBs of a 4x8 chroma CB includes residual values for two 2x4 chroma areas. In such implementations, splitting of the current luma block into smaller luma blocks for intra BC prediction operations as described above (with corresponding splitting of chroma blocks into smaller chroma blocks for intra BC prediction operations, as described above) is permitted for block sizes larger than the minimum transform size. If the smaller blocks have the minimum transform size, splitting of the current luma block into smaller luma blocks for intra BC prediction operations is supported if chroma blocks of residual values are rearranged as shown in Figure 24.

[0235] Figure 24 shows blocks with minimum transform size of 4x4 for video in YUV 4:2:2 format. An 8x8 luma block (2410) includes four 4x4 luma blocks (2421, 2422, 2423, 2424), which are numbered 0…3. A corresponding 4x8 chroma block (2430) includes corresponding chroma components, which are downsampled by a factor of two horizontally. The transform size of 2x4 is not supported.

[0236] According to the current draft of the HEVC standard, for a 4x8 chroma block, 2x4 blocks 0 and 1 are processed together as a 4x4 chroma TB with 4x4 transform size for residual coding and decoding, and 2x4 blocks 2 and 3 are processed together as a 4x4 chroma TB with 4x4 transform size for residual coding and decoding. As a result, since 2x4 block 0 will not be reconstructed until 2x4 block 1 is reconstructed, intra BC prediction operations for 2x4 block 1 cannot reference reconstructed sample values in 2x4 block 0. Similarly, intra BC prediction operations for 2x4 block 3 cannot reference reconstructed sample values in 2x4 block 2.

[0237] As shown in Figure 24, the 2x4 blocks of the 4x8 chroma block (2440) can be rearranged for purposes of residual coding and decoding. In this case, 2x4 blocks 0 and 2 are processed together as a 4x4 chroma TB with 4x4 transform size for residual coding and decoding, and 2x4 blocks 1 and 3 are processed together as a 4x4 chroma TB with 4x4 transform size for residual coding and decoding. Intra BC prediction operations still happen in the "un-rearranged" 2x4 blocks of the 4x8 chroma block (2430). When the 4x8

chroma block is decoded, intra BC prediction for 2x4 blocks 0 and 2 can happen first, followed by residual decoding for 2x4 blocks 0 and 2 and reconstruction of sample values for 2x4 blocks 0 and 2. Then, 2x4 blocks 1 and 3 are reconstructed. Thus, 2x4 block 0 can be reconstructed before 2x4 block 1 is reconstructed, so intra BC prediction operations for 2x4 block 1 can reference reconstructed sample values in 2x4 block 0. Similarly, intra BC prediction operations for 2x4 block 3 can reference reconstructed sample values in 2x4 block 2. This rearrangement of 2x4 chroma blocks of residual values happens at the encoder and at the decoder.

[0238] With the 4x8 chroma block (2440) having rearranged 2x4 blocks for residual coding and decoding, an overlap area is supported for the left half of the luma block and corresponding chroma blocks, as shown in gray. During BV estimation, an encoder can estimate reconstructed sample values in the overlap area (as described above) and evaluate candidate BV values that reference intra-prediction regions that partially or completely cover the overlap area.

[0239] In this approach, BV values with non-zero horizontal BV components are allowed, but the BV values have zero-value vertical BV components. This is appropriate for screen capture content, for example, which typically exhibits horizontal continuity in sample values.

[0240] In example implementations, intra BC prediction uses BV values with integer-sample precision for luma blocks and for chroma blocks. Fractional displacements (and fractional interpolation between reconstructed sample values) are not used for BV values. When chroma data for a picture has reduced resolution relative to the luma data for the picture (*e.g.*, when the format is YUV 4:2:0 format or YUV 4:2:2 format), the BV value from the luma block, before it is applied to a corresponding chroma block, may be scaled down to adjust for the difference in chroma resolution, but is rounded and/or truncated to a value with integer-sample precision.

[0241] Alternatively, BV values have integer-sample precision for luma blocks but may have fractional-sample precision for corresponding chroma blocks. In particular, when chroma data for a picture has reduced resolution relative to the luma data for the picture, the BV value from the luma block, before it is applied to a corresponding chroma block, may be scaled down to adjust for the difference in chroma resolution. For YUV 4:2:0 format, the vertical and horizontal components of a BV value can be divided by two, and truncated or rounded, yielding a value with either integer-sample precision or a fractional-sample precision such as half-sample precision. For example, a BVy

component value of 3 for a luma block is scaled to a BVy component value of 1.5 for a corresponding chroma block. For YUV 4:2:2 format, the horizontal component of a BV value can be divided by two, and truncated or rounded, yielding a value with either integer-precision or a fractional-sample precision such as half-sample position precision. When a BV value has fractional-sample precision, interpolation between reconstructed sample values can use bilinear interpolation, bicubic interpolation or another form of interpolation.

[0242] Alternatively, BV values can have fractional-sample prediction for luma blocks and corresponding chroma blocks. Again, when a BV value has fractional-sample precision, interpolation between reconstructed sample values can use bilinear interpolation, bicubic interpolation or another form of interpolation.

[0243] In view of the many possible embodiments to which the principles of the disclosed invention may be applied, it should be recognized that the illustrated embodiments are only preferred examples of the invention and should not be taken as limiting the scope of the invention. Rather, the scope of the invention is defined by the following claims. We therefore claim as our invention all that comes within the scope of these claims.

2014374141 19 Sep 2018

[0244] Throughout this specification and claims which follow, unless the context requires otherwise, the word "comprise", and variations such as "comprises" and "comprising", will be understood to imply the inclusion of a stated integer or step or group of integers or steps but not the exclusion of any other integer or step or group of integers or steps.

[0245] The reference in this specification to any prior publication (or information derived from it), or to any matter which is known, is not, and should not be taken as an acknowledgment or admission or any form of suggestion that that prior publication (or information derived from it) or known matter forms part of the common general knowledge in the field of endeavour to which this specification relates.

2014374141 19 Sep 2018

CLAIMS

1. A computing device comprising:

an encoder configured to perform operations to produce encoded data for a picture, wherein the encoder is a video encoder or image encoder, and wherein the operations include:

estimating reconstructed sample values within an overlap area of a current block of the picture, wherein the current block has multiple smaller blocks, the overlap area covering parts of the current block that are in potential intra-prediction regions for at least one of the multiple smaller blocks, wherein the estimating the reconstructed sample values within the overlap area uses corresponding original sample values from the overlap area to estimate the reconstructed sample values, respectively, within the overlap area without using reconstructed sample values from outside the overlap area in place of the reconstructed sample values, respectively, within the overlap area; and

performing block vector ("BV") estimation to determine a BV value for the current block, the BV value indicating a displacement from the current block to a region of the picture that includes sample values used for prediction, and wherein the BV estimation uses at least some of the estimated reconstructed sample values within the overlap area of the current block; and

a buffer configured to store the encoded data for output as part of a bitstream.

2. The computing device of claim 1 wherein the estimating the reconstructed sample values within the overlap area includes using the corresponding original sample values from the overlap area as the reconstructed sample values, respectively, within the overlap area.

3. The computing device of claim 1 wherein the estimating the reconstructed sample values within the overlap area includes:

processing the corresponding original sample values from the overlap area using a frequency transform, quantization, inverse quantization and an inverse frequency transform; and

using the processed sample values as the reconstructed sample values, respectively, within the overlap area.

2014374141 19 Sep 2018

4. The computing device of claim 1 wherein the estimating the reconstructed sample values within the overlap area includes:

predicting sample values within the overlap area using at least one BV predictor;

determining residual values based on differences between the predicted sample values and the corresponding original sample values from the overlap area;

processing the residual values using a frequency transform, quantization, inverse quantization and an inverse frequency transform;

combining results of the processing with the predicted sample values; and

using results of the combining as the reconstructed sample values, respectively, within the overlap area.

5. The computing device of claim 1 wherein the BV estimation includes, for each of one or more candidate BV values for the current block:

performing block matching operations between the multiple smaller blocks, respectively, and candidate intra-prediction regions according to the candidate BV value.

6. The computing device of claim 1 wherein the BV estimation includes:

for each of the multiple smaller blocks, determining a BV value for the smaller block; and

synthesizing the BV values for the multiple smaller blocks into the BV value for the current block.

7. The computing device of claim 1 wherein the operations further comprise:

performing intra block copy ("BC") prediction for the current block using the BV value for the current block, including performing BC operations on a block-by-block basis for the multiple smaller blocks.

8. The computing device of claim 1 wherein the current block is a coding block of a coding unit, and wherein each of the multiple blocks is a transform block of a transform unit.

9. In a computing device that implements a video encoder or image encoder, a method comprising:

encoding a picture to produce encoded data, wherein the encoding includes:

2014374141 19 Sep 2018

estimating reconstructed sample values within an overlap area of a current block of the picture, wherein the current block has multiple smaller blocks, the overlap area covering parts of the current block that are in potential intra-prediction regions for at least one of the multiple smaller blocks, wherein the estimating the reconstructed sample values within the overlap area uses corresponding original sample values from the overlap area to estimate the reconstructed sample values, respectively, within the overlap area without using reconstructed sample values from outside the overlap area in place of the reconstructed sample values, respectively, within the overlap area; and

performing block vector ("BV") estimation to determine a BV value for the current block, the BV value indicating a displacement from the current block to a region of the picture that includes sample values used for prediction, and wherein the BV estimation uses at least some of the estimated reconstructed sample values within the overlap area of the current block; and

outputting the encoded data as part of a bitstream.

10. The method of claim 9 wherein the estimating the reconstructed sample values within the overlap area includes using the corresponding original sample values from the overlap area as the reconstructed sample values, respectively, within the overlap area.

11. The method of claim 9 wherein the estimating the reconstructed sample values within the overlap area includes:

processing the corresponding original sample values from the overlap area using a frequency transform, quantization, inverse quantization and an inverse frequency transform; and

using the processed sample values as the reconstructed sample values, respectively, within the overlap area.

12. The method of claim 9 wherein the estimating the reconstructed sample values within the overlap area includes:

predicting sample values within the overlap area using at least one BV predictor;

determining residual values based on differences between the predicted sample values and the corresponding original sample values from the overlap area;

processing the residual values using a frequency transform, quantization, inverse quantization and an inverse frequency transform;

2014374141 19 Sep 2018

combining results of the processing with the predicted sample values; and

using results of the combining as the reconstructed sample values, respectively, within the overlap area.

13. The method of claim 9 wherein the BV estimation includes, for each of one or more candidate BV values for the current block:

performing block matching operations between the multiple smaller blocks, respectively, and candidate intra-prediction regions according to the candidate BV value.

14. The method of claim 9 wherein the BV estimation includes:

for each of the multiple smaller blocks, determining a BV value for the smaller block; and

synthesizing the BV values for the multiple smaller blocks into the BV value for the current block.

15. The method of claim 9 wherein the encoding further comprises:

performing intra block copy ("BC") prediction for the current block using the BV value for the current block, including performing BC operations on a block-by-block basis for the multiple smaller blocks.

16. The method of claim 9 wherein the current block is a coding block of a coding unit, and wherein each of the multiple blocks is a transform block of a transform unit.

17. One or more computer-readable media storing computer-executable instructions for causing a computing device, when programmed thereby, to perform operations, wherein the one or more computer-readable media are selected from the group consisting of volatile memory, non-volatile memory, magnetic disk, CD-ROM, and DVD, the operations comprising:

encoding a picture to produce encoded data, wherein the encoding includes:

estimating reconstructed sample values within an overlap area of a current block of the picture, wherein the current block has multiple smaller blocks, the overlap area covering parts of the current block that are in potential intra-prediction regions for at least one of the multiple smaller blocks, wherein the estimating the reconstructed sample values within the overlap area uses corresponding original sample values from the overlap

2014374141 19 Sep 2018

area to estimate the reconstructed sample values, respectively, within the overlap area without using reconstructed sample values from outside the overlap area in place of the reconstructed sample values, respectively, within the overlap area; and

performing block vector ("BV") estimation to determine a BV value for the current block, the BV value indicating a displacement from the current block to a region of the picture that includes sample values used for prediction, and wherein the BV estimation uses at least some of the estimated reconstructed sample values within the overlap area of the current block; and

outputting the encoded data as part of a bitstream.

18. The one or more computer-readable media of claim 17 wherein the estimating the reconstructed sample values within the overlap area includes using the corresponding original sample values from the overlap area as the reconstructed sample values, respectively, within the overlap area.

19. The one or more computer-readable media of claim 17 wherein the estimating the reconstructed sample values within the overlap area includes:

processing the corresponding original sample values from the overlap area using a frequency transform, quantization, inverse quantization and an inverse frequency transform; and

using the processed sample values as the reconstructed sample values, respectively, within the overlap area.

20. The one or more computer-readable media of claim 17 wherein the estimating the reconstructed sample values within the overlap area includes:

predicting sample values within the overlap area using at least one BV predictor;

determining residual values based on differences between the predicted sample values and the corresponding original sample values from the overlap area;

processing the residual values using a frequency transform, quantization, inverse quantization and an inverse frequency transform;

combining results of the processing with the predicted sample values; and

using results of the combining as the reconstructed sample values, respectively, within the overlap area.

Figure 1

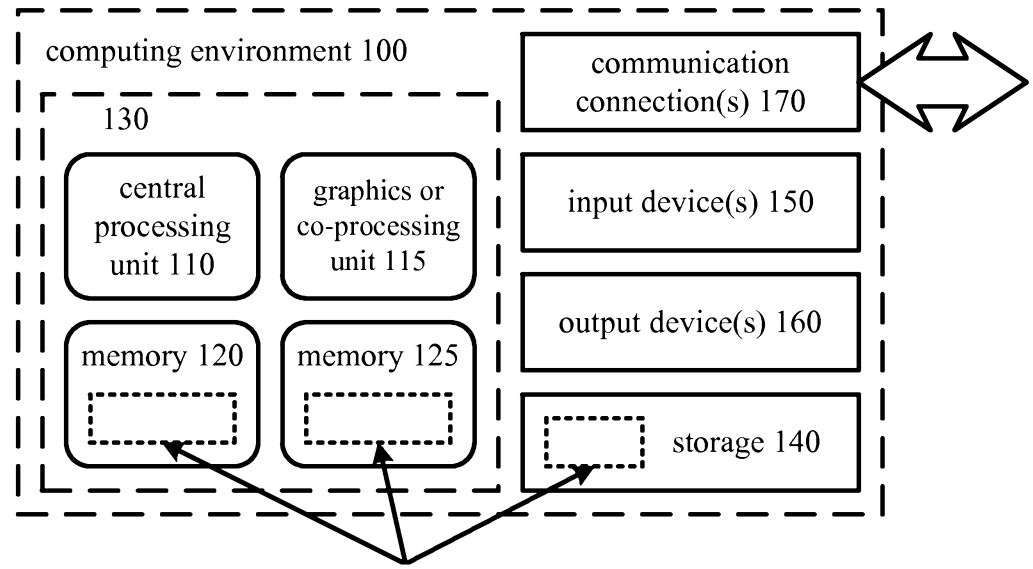


Figure 2a

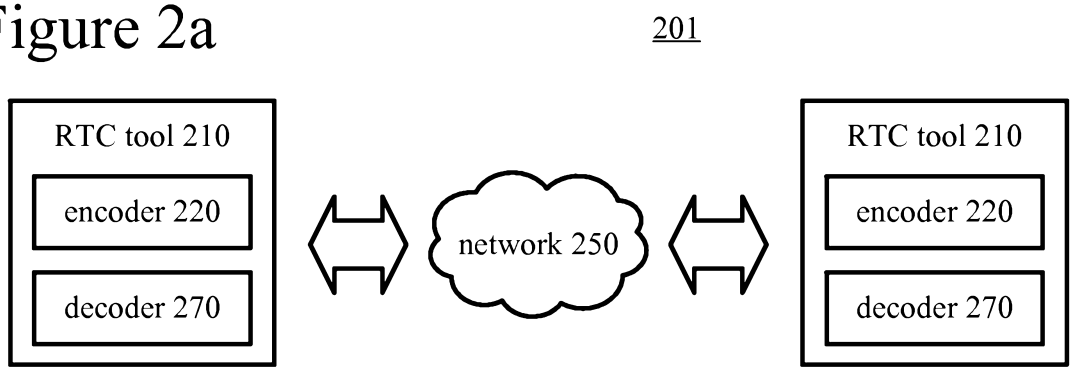


Figure 2b

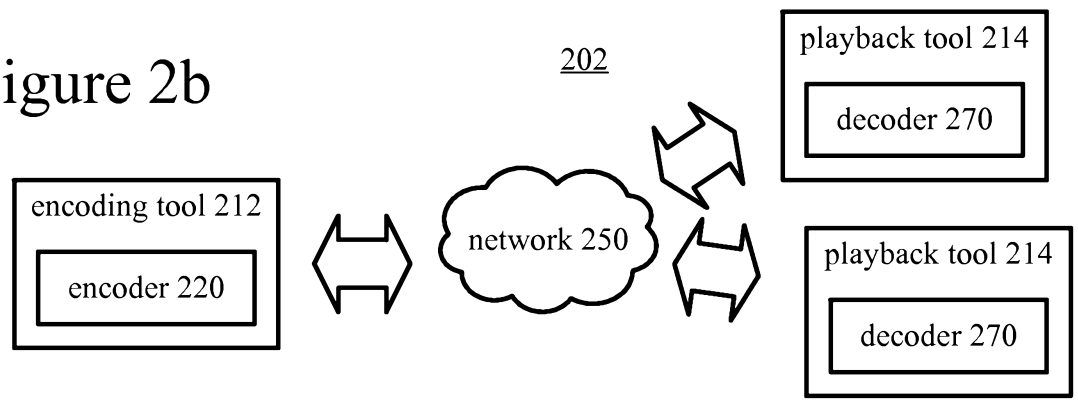


Figure 3

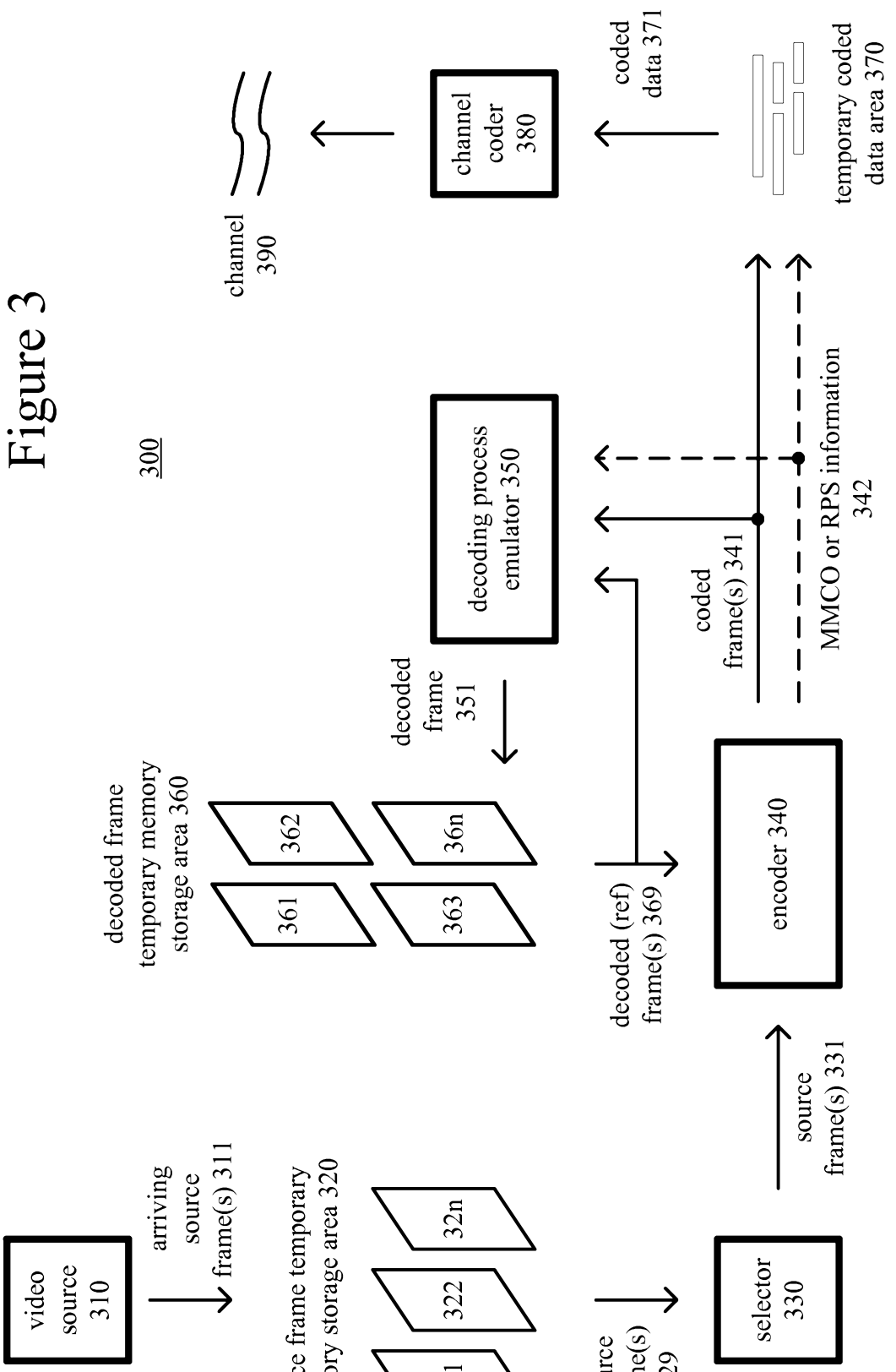

300
video source 310
arriving source frame(s) 311
source frame temporary memory storage area 320
321
322
32n
source frame(s) 329
selector 330
source frame(s) 331
encoder 340
decoded (ref) frame(s) 369
coded frame(s) 341
MMCO or RPS information 342
decoded frame temporary memory storage area 360
361
362
363
36n
decoded frame 351
decoding process emulator 350
channel coder 380
coded data 371
temporary coded data area 370
channel 390

Figure 4

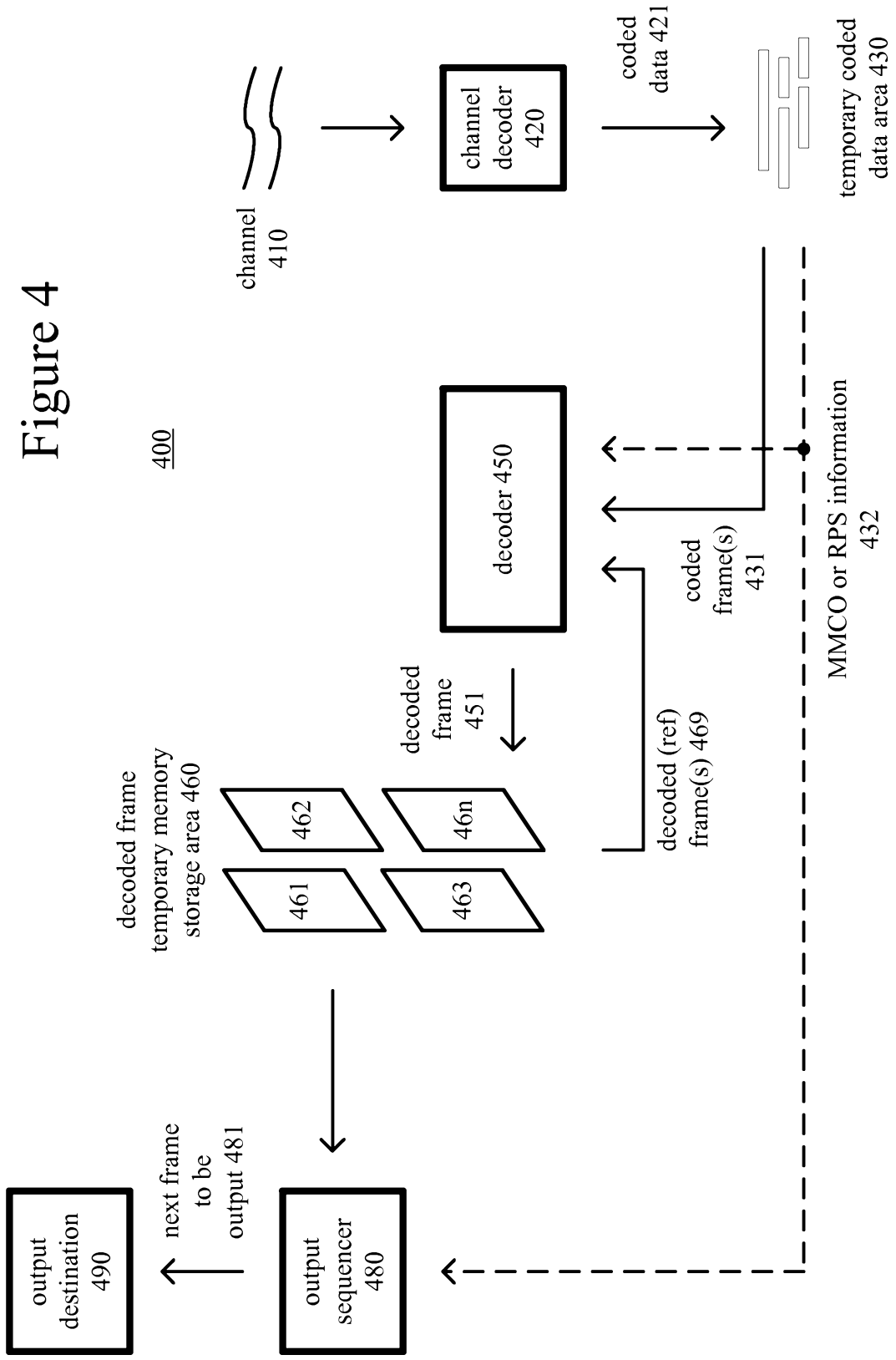
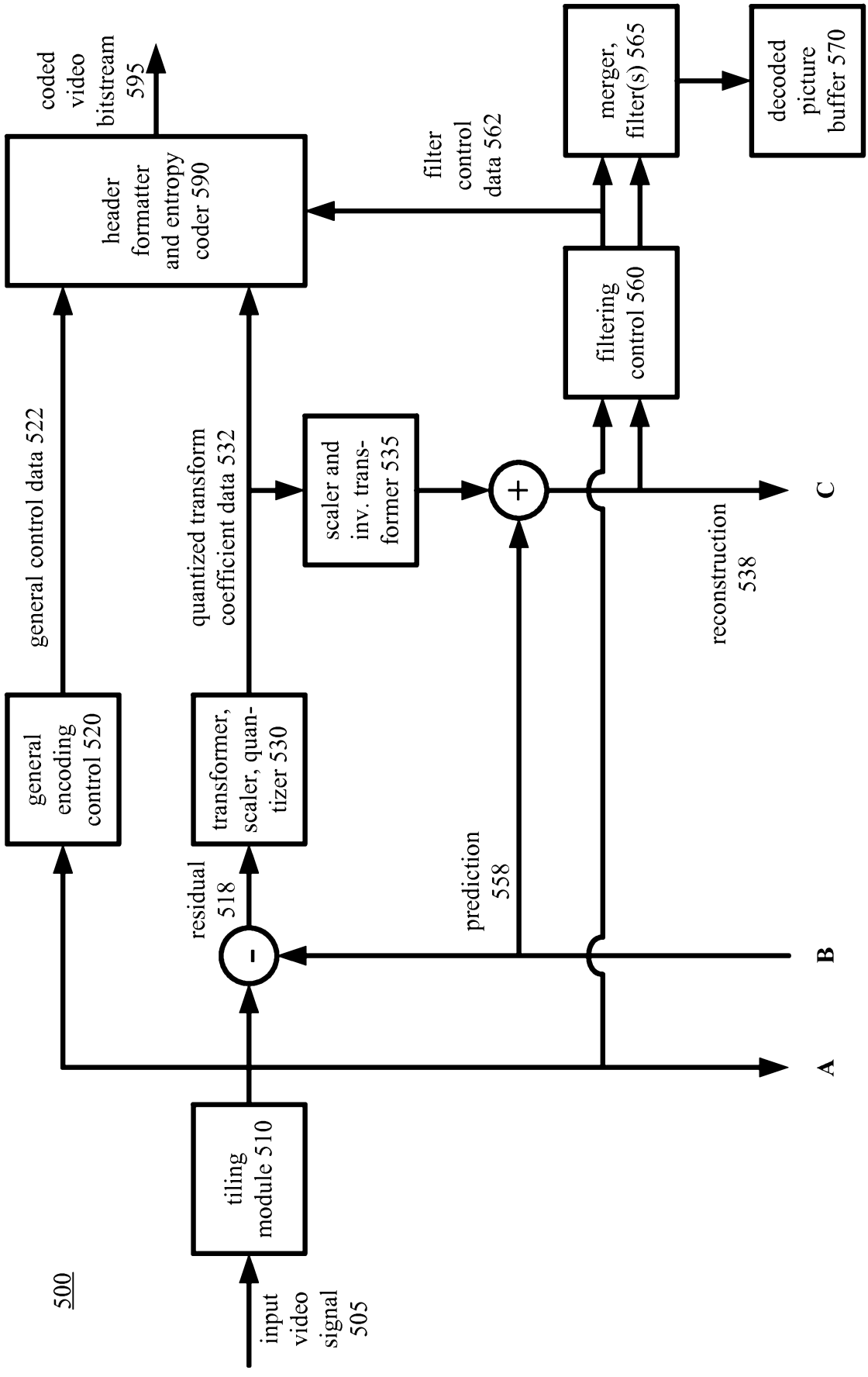

400
channel 410
channel decoder 420
coded data 421
temporary coded data area 430
decoder 450
decoded frame temporary memory storage area 460
461
462
463
46n
decoded frame 451
decoded (ref) frame(s) 469
coded frame(s) 431
MMCO or RPS information 432
output sequencer 480
next frame to be output 481
output destination 490

Figure 5a


500
input video signal 505
tiling module 510
general encoding control 520
transformer, scaler, quantizer 530
scaler and inv. transformer 535
filtering control 560
merger, filter(s) 565
decoded picture buffer 570
header formatter and entropy coder 590
coded video bitstream 595
general control data 522
quantized transform coefficient data 532
residual 518
prediction 558
reconstruction 538
filter control data 562
A
B
C

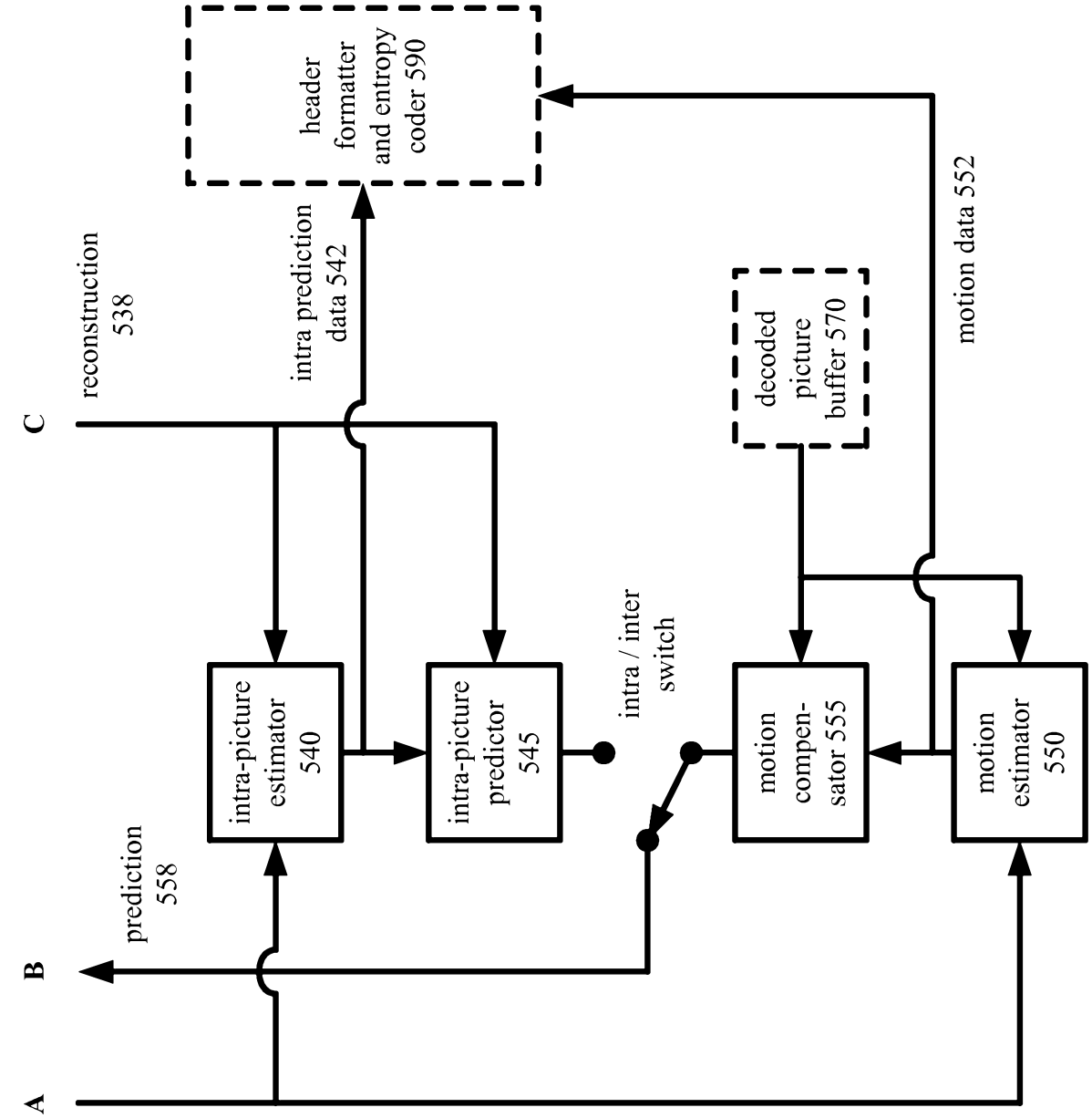

500
A
B
C
prediction 558
reconstruction 538
intra-picture estimator 540
intra-picture predictor 545
intra / inter switch
motion compen-sator 555
motion estimator 550
decoded picture buffer 570
intra prediction data 542
header formatter and entropy coder 590
motion data 552

Figure 5b

Figure 6

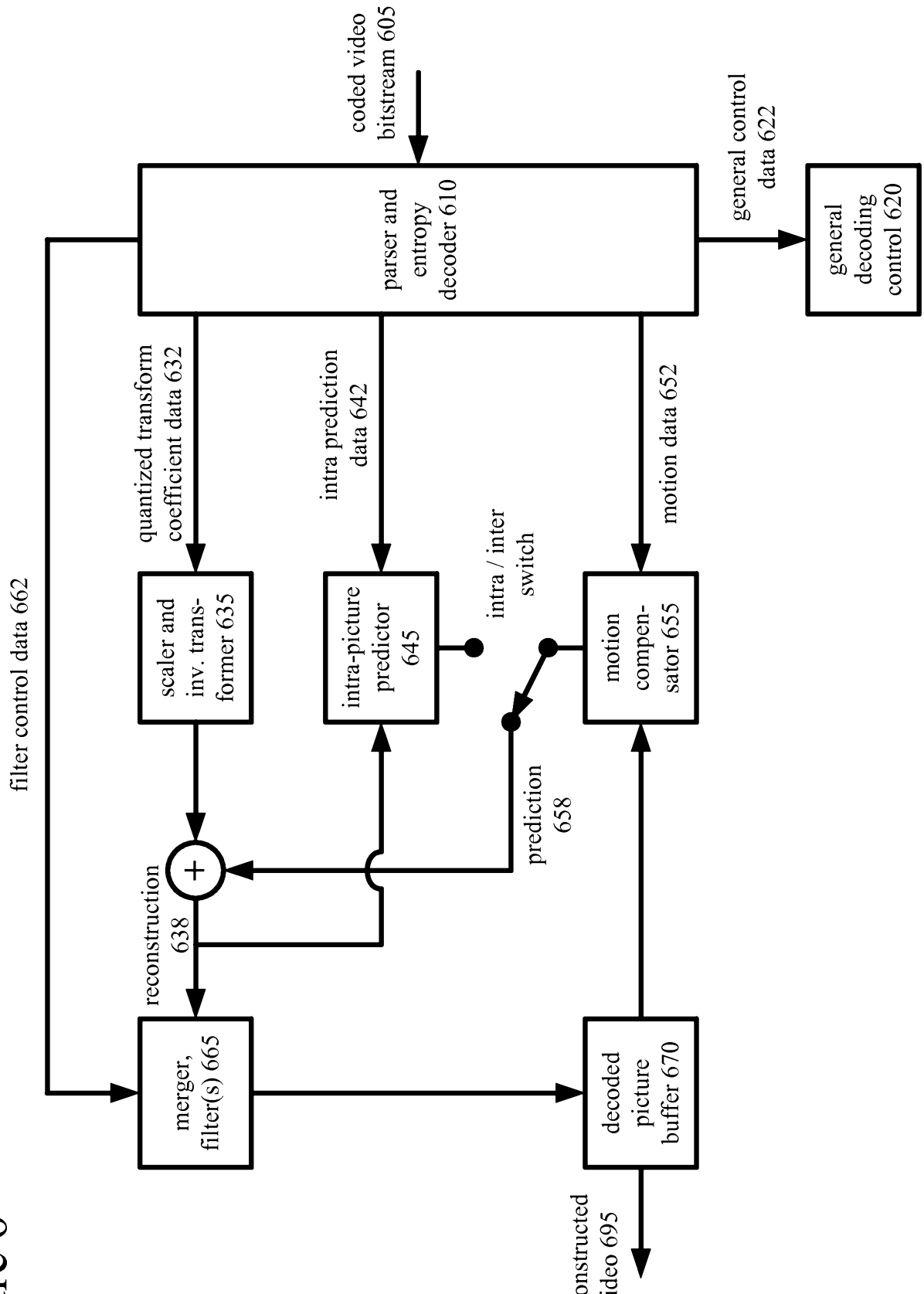
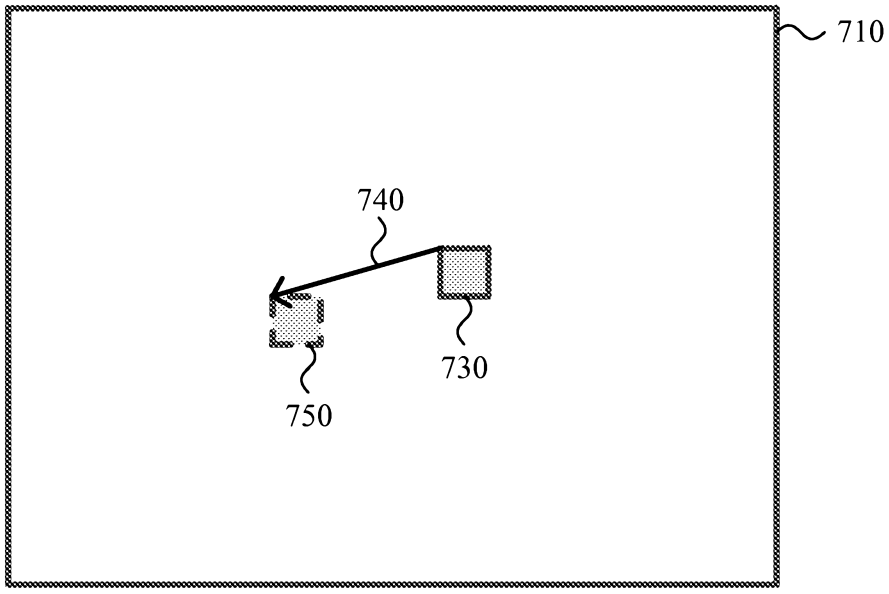
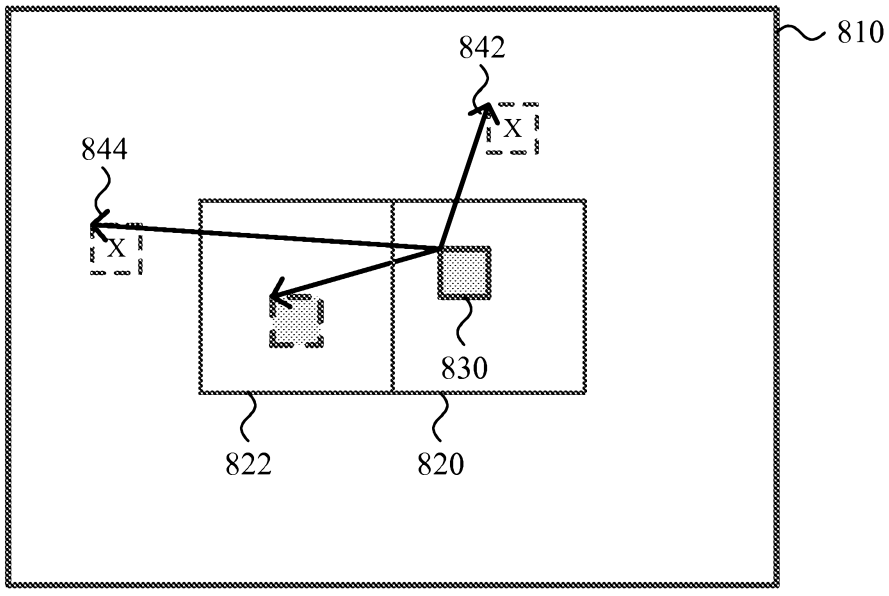

600
coded video bitstream 605
parser and entropy decoder 610
general control data 622
general decoding control 620
quantized transform coefficient data 632
intra prediction data 642
motion data 652
filter control data 662
scaler and inv. trans-former 635
intra-picture predictor 645
intra / inter switch
motion compen-sator 655
prediction 658
reconstruction 638
merger, filter(s) 665
decoded picture buffer 670
reconstructed video 695

Figure 7



block vector (740) for current block (730) of current frame (710), indicating a displacement to a region (750) in the current frame (710)

Figure 8



candidate block vectors (842, 844) indicating displacements to regions that are outside of search range for current block (830) of current frame (810)

Figure 9

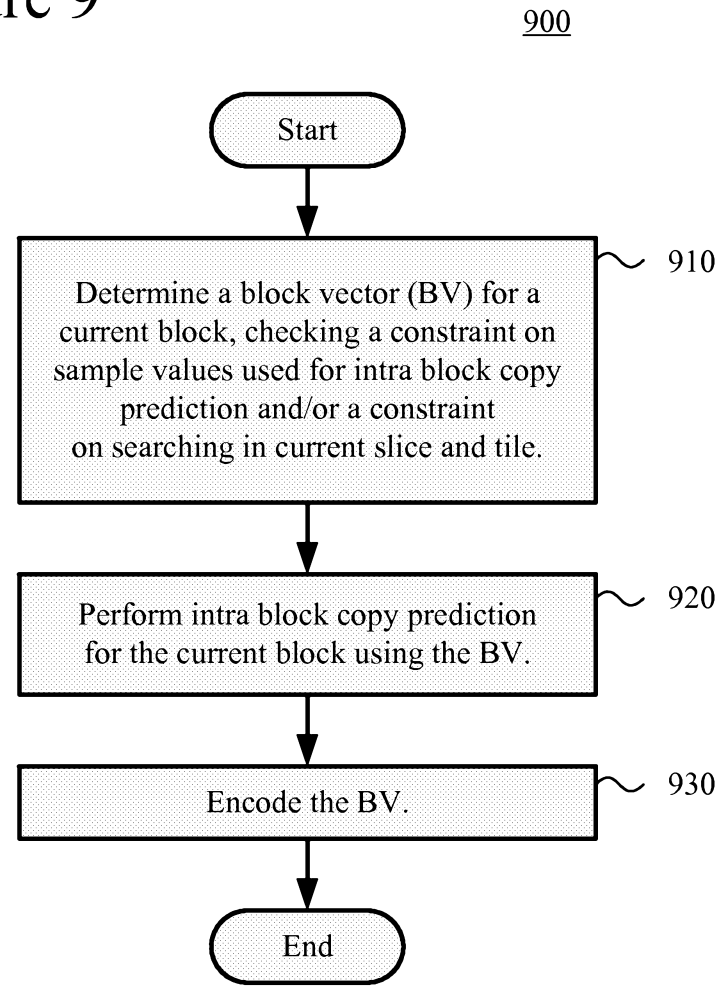


Figure 10

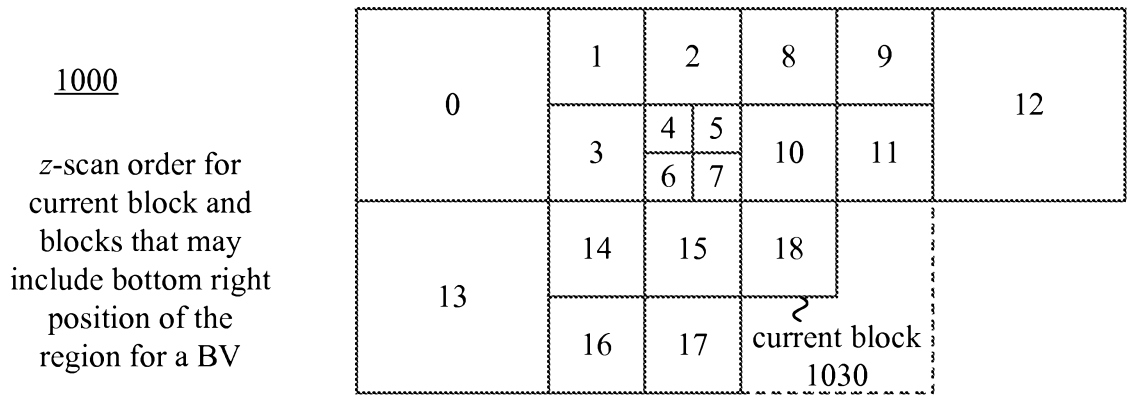


Figure 11

1100

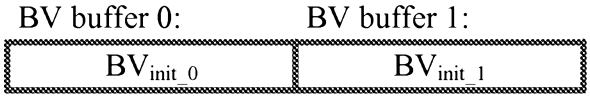


Figure 12

1200

index value:	BV predictor:
idx 0 :	first candidate BV value (BV_{init_0})
idx 1 :	second candidate BV value (BV_{init_1})

Figure 13

1300

2*N* x 2*N* CU
with single PU

operation	pair of candidate BV values
predict BV for PU0	1st initial candidate BV value (PBV0), 2nd initial candidate BV value (PBV1)
process BV for PU0	

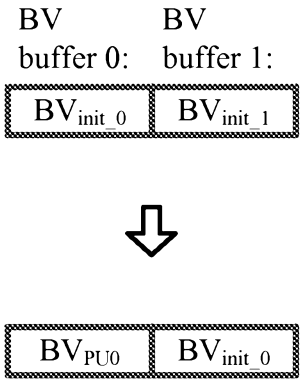
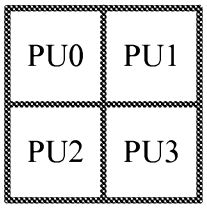


Figure 14

1400



2*N* x 2*N* CU with four *N* x *N* PUs

operation	pair of candidate BV values
predict BV for PU0	1st initial candidate BV value (PBV0), 2nd initial candidate BV value (PBV1)
process BV for PU0	
predict BV for PU1	BV for PU0, 1st initial candidate BV value (PBV1)
process BV for PU1	
predict BV for PU2	BV for PU1, BV for PU0
process BV for PU2	
predict BV for PU3	BV for PU1, BV for PU2
process BV for PU3	

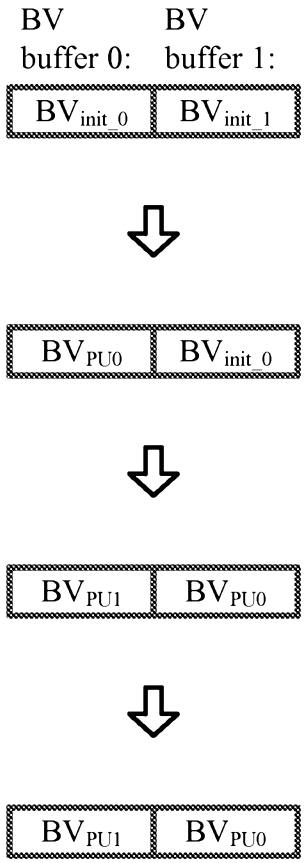
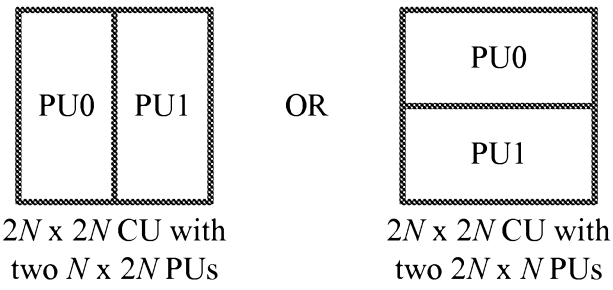


Figure 15

1500



2*N* x 2*N* CU with two *N* x 2*N* PUs

2*N* x 2*N* CU with two 2*N* x *N* PUs

operation	pair of candidate BV values
predict BV for PU0	1st initial candidate BV value (PBV0), 2nd initial candidate BV value (PBV1)
process BV for PU0	
predict BV for PU1	BV for PU0, 1st initial candidate BV value (PBV0)
process BV for PU1	

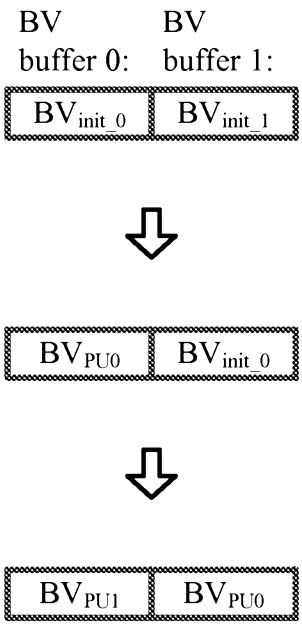


Figure 16

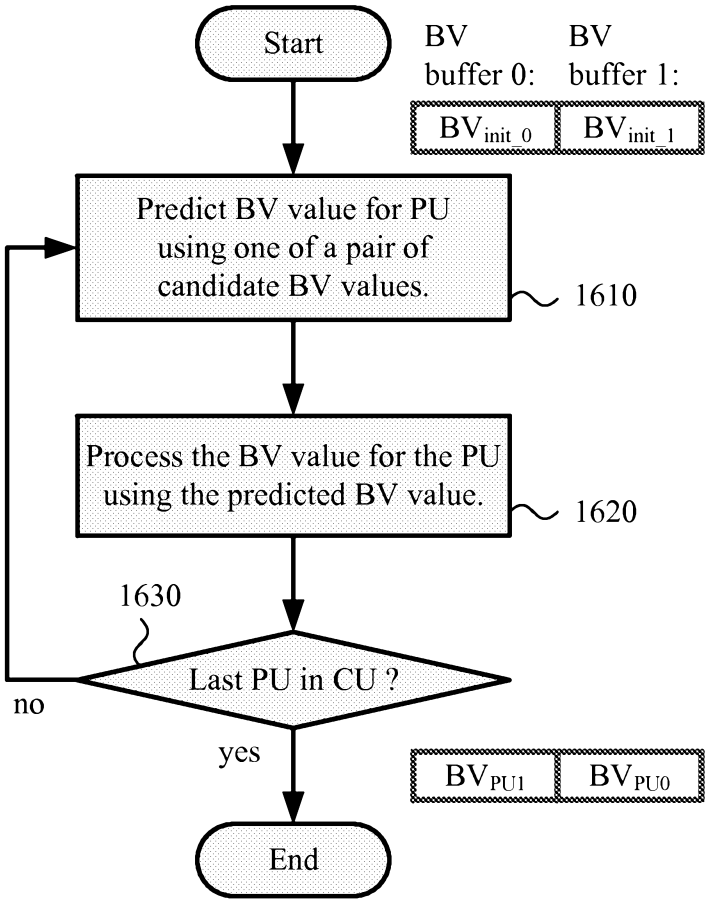

1600
Start
BV buffer 0:
BV buffer 1:
BVinit_0
BVinit_1
Predict BV value for PU using one of a pair of candidate BV values.
1610
Process the BV value for the PU using the predicted BV value.
1620
1630
Last PU in CU ?
no
yes
BVPU1
BVPU0
End

Figure 17

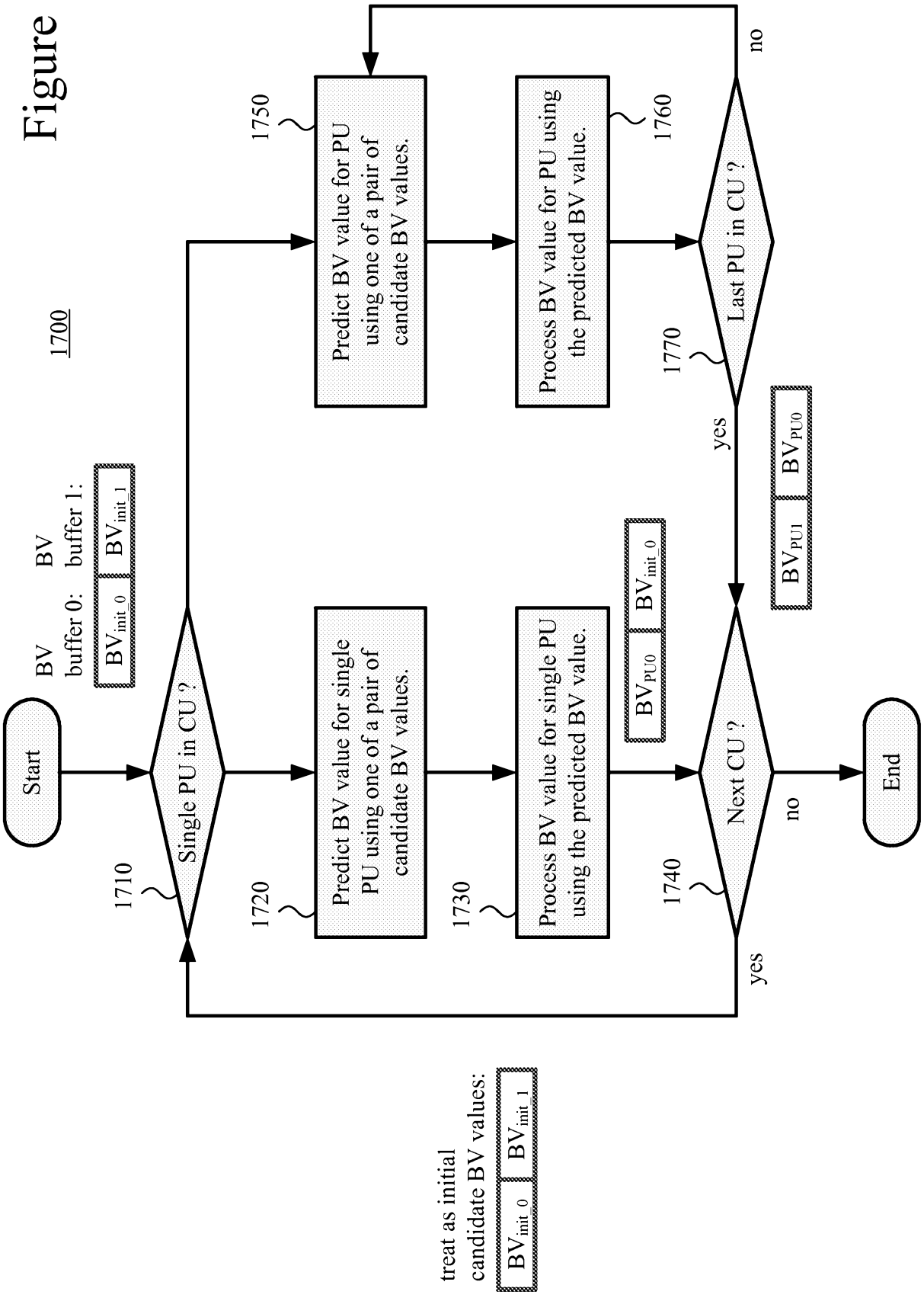

1700
Start
BV buffer 0: BV buffer 1:
BV_init_0 BV_init_1
Single PU in CU ?
1710
Predict BV value for single PU using one of a pair of candidate BV values.
1720
Process BV value for single PU using the predicted BV value.
1730
BV_PU0 BV_init_0
Next CU ?
1740
yes
no
Predict BV value for PU using one of a pair of candidate BV values.
1750
Process BV value for PU using the predicted BV value.
1760
Last PU in CU ?
1770
yes
no
BV_PU1 BV_PU0
End
treat as initial candidate BV values:
BV_init_0 BV_init_1

Figure 18

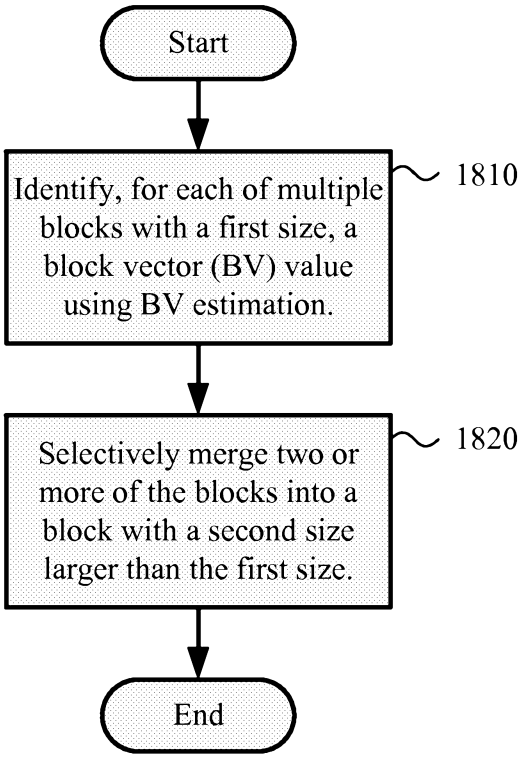


Figure 20

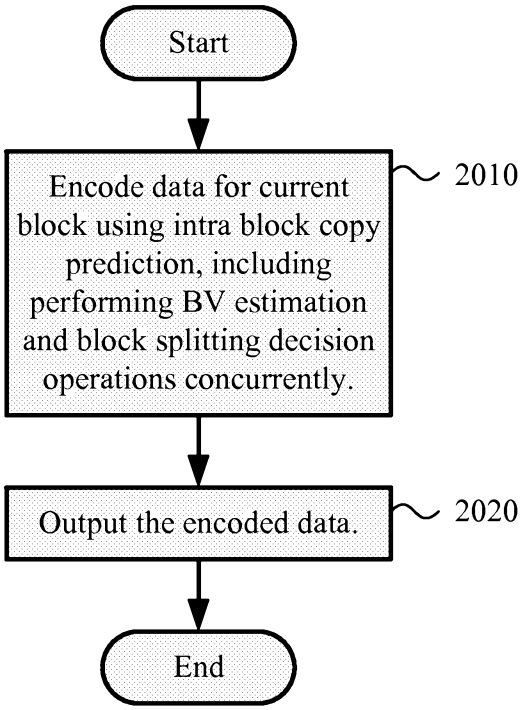


Figure 19

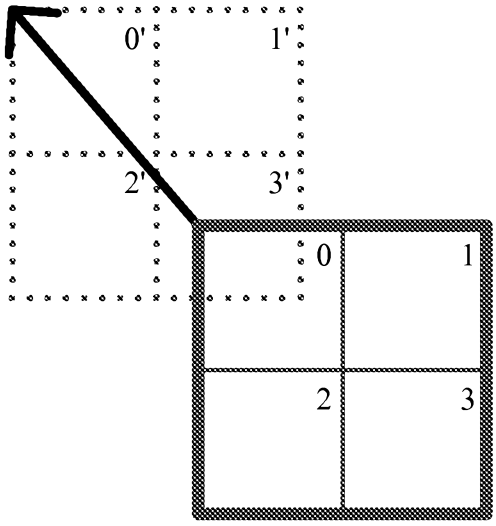


Figure 21

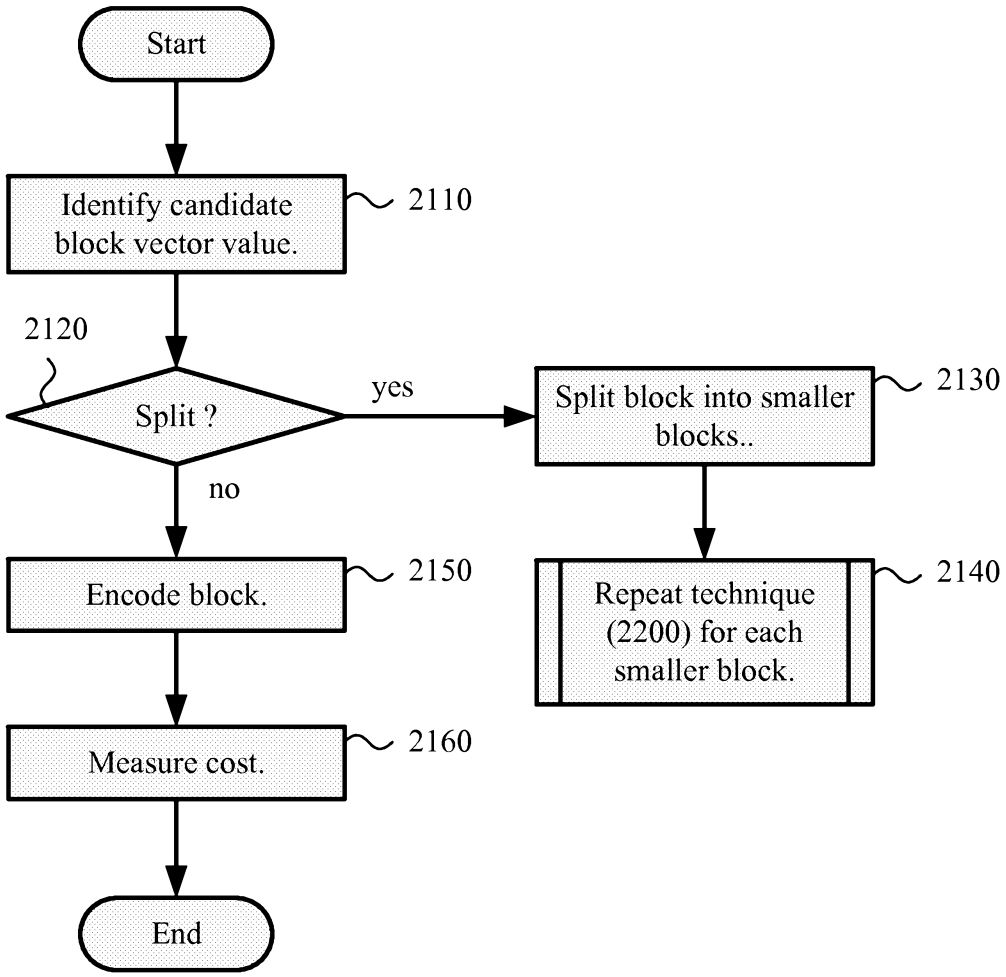

2100
Start
Identify candidate block vector value.
2110
2120
Split ?
yes
Split block into smaller blocks..
2130
no
Encode block.
2150
Repeat technique (2200) for each smaller block.
2140
Measure cost.
2160
End

Figure 22a

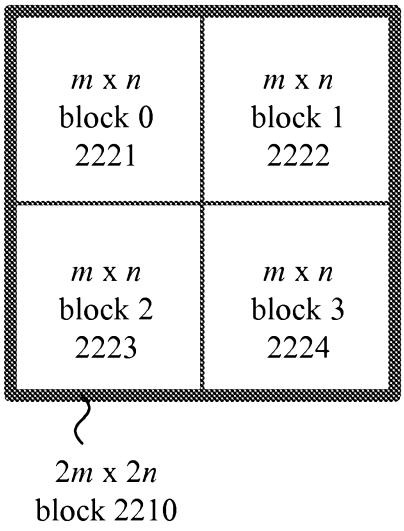


Figure 22b

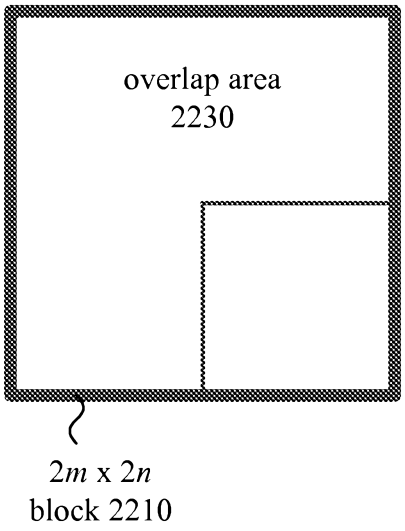


Figure 23

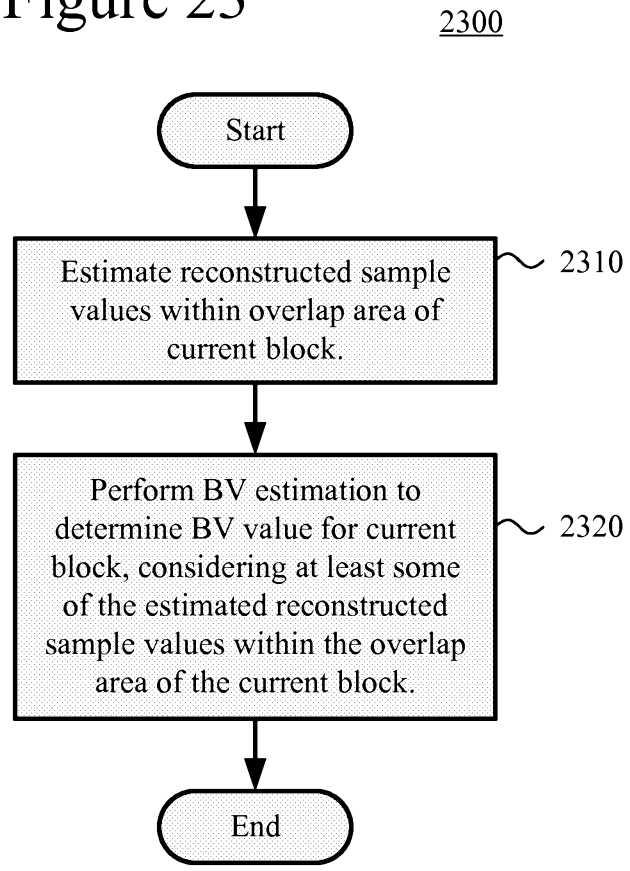
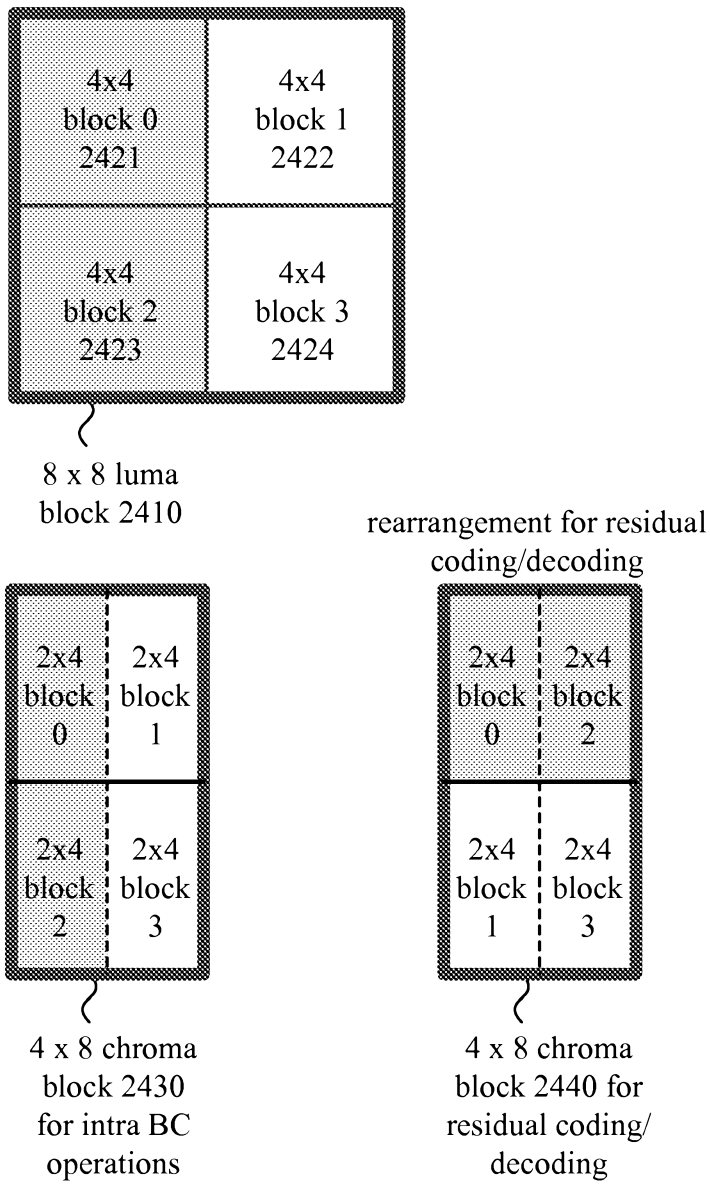


Figure 24


4x4 block 0 2421
4x4 block 1 2422
4x4 block 2 2423
4x4 block 3 2424
8 x 8 luma block 2410
rearrangement for residual coding/decoding
2x4 block 0
2x4 block 1
2x4 block 2
2x4 block 3
4 x 8 chroma block 2430 for intra BC operations
2x4 block 0
2x4 block 2
2x4 block 1
2x4 block 3
4 x 8 chroma block 2440 for residual coding/ decoding