

(51) International Patent Classification:
H04N 21/234 (2011.01)(21) International Application Number:
PCT/US2013/037058(22) International Filing Date:
18 April 2013 (18.04.2013)

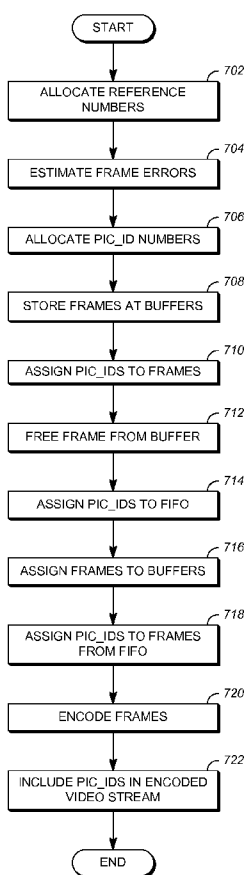
(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/636,944 23 April 2012 (23.04.2012) US(71) Applicant: **GOOGLE INC.** [US/US]; 1600 Amphitheatre Parkway, Mountain View, California 94043 (US).(72) Inventors: **GU, Qunshan**; c/o Google Inc., 1600 Amphitheatre Parkway, Mountain View, California 94043 (US).**PANICONI, Marco**; c/o Google Inc., 1600 Amphitheatre Parkway, Mountain View, California 94043 (US).(74) Agents: **BASILE, Andrew R., JR.** et al.; Young Basile Hanlon & Macfarlane, P.C., 3001 West Big Beaver Rd., Troy, Michigan 48084 (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, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, 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, 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.

[Continued on next page]

(54) Title: MANAGING MULTI-REFERENCE PICTURE BUFFERS FOR VIDEO DATA CODING



(57) **Abstract:** Encoding and decoding is accomplished herein using a multi-reference picture buffer. Reference frames are assigned to reference picture buffers when used for encoding and are identified with picture IDs unique relative to other picture IDs currently associated with the reference picture buffers. The maximum picture number used as the picture ID can be based on a number of reference picture buffers available to encode and decode the frames plus a value based on an expected error rate. The picture IDs can be assigned based on a least recently used policy. When a reference frame is no longer needed for encoding and decoding, a picture ID number associated with the reference frame can be released for re-assignment to a new reference frame.

FIG. 7



(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, 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, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

MANAGING MULTI-REFERENCE PICTURE BUFFERS FOR VIDEO DATA CODING

TECHNICAL FIELD

[0001] This disclosure relates to multi-reference picture buffers for encoding and decoding visual data, such as video stream data, for transmission or storage.

BACKGROUND

[0002] An increasing number of applications today make use of digital video for various purposes including, for example, remote business meetings via video conferencing, high definition video entertainment, video advertisements, and sharing of user-generated videos. As technology is evolving, users have higher expectations for video quality and expect high resolution video with smooth playback.

[0003] Digital video streams typically represent video using a sequence of frames. Each frame can include a number of blocks, which in turn may contain information describing the value of color, brightness or other attributes for pixels. The amount of data in a typical video stream is large, and transmission and storage of video can use significant computing or communications resources. Various approaches have been proposed to reduce the amount of data in video streams, including compression and other encoding techniques. Certain of these techniques use reference frames or portions of reference frames to generate differences between the reference and the frame being encoded. The differences can generally be encoded with fewer bits than the original frame.

SUMMARY

[0004] Disclosed herein are aspects of systems, methods and apparatuses for managing multi-reference picture buffers used to encode a video stream having a plurality of frames and/or decode an encoded video stream having a plurality of frames. One encoding method includes identifying an allocated number of reference frame buffers, calculating a maximum number of picture IDs as equal to or greater than the allocated number, assigning a picture ID to a respective reference frame assigned to a respective reference frame buffer of a plurality of reference frame buffers to no higher than the maximum number of picture IDs, each picture ID having a value not assigned to any other reference frames assigned to the plurality of reference frame buffers, freeing the picture ID when its associated reference

frame is no longer assigned to the reference frame buffer, periodically re-using a freed picture ID when assigning the picture IDs, encoding one or more frames of the plurality of frames using the reference frames assigned to the plurality of reference frame buffers, and including the picture ID numbers in the video stream.

[0005] One method for decoding an encoded video stream having a plurality of frames includes receiving an encoded video stream, decoding an expected error value representing possible frame errors, allocating reference picture buffers based on the expected error value, decoding a plurality of reference frames from the video stream, adding each of the plurality of reference frames to the reference picture buffers at least to a maximum value, determining an encoded video stream error value representing actual frame errors based on the decoded plurality of reference frames, and if the encoded video stream error value is greater than the decoded expected error value, signaling an error.

[0006] Variations in these and other aspects are described in additional detail below.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] The description herein makes reference to the accompanying drawings wherein like reference numerals refer to like parts throughout the several views, and wherein:

[0008] FIG. 1 is a schematic of a video encoding and decoding system;

[0009] FIG. 2 is a diagram of a typical video stream to be encoded and decoded;

[0010] FIG. 3 is a block diagram of a video encoder in accordance with an implementation;

[0011] FIG. 4 is a block diagram of a video decoder in accordance with another implementation;

[0012] FIG. 5 is a block diagram of a modified video encoder illustrating management of a multi-reference picture buffer;

[0013] FIG. 6 is a block diagram of a modified video decoder illustrating management of a multi-reference picture buffer;

[0014] FIG. 7 is a flowchart diagram illustrating a process of managing a multi-reference picture buffer and encoding reference pictures in an encoded video stream in accordance with an implementation; and

[0015] FIG. 8 is a flowchart diagram illustrating a process of decoding frames using a reference picture buffer in accordance with an implementation.

DETAILED DESCRIPTION

[0016] To permit transmission of digital video streams while limiting bandwidth consumption, video encoding and decoding schemes incorporate various compression formats. These digital video encoder/decoder formats, also called codecs, can be implemented using hardware and/or software programs executing from memory included in a computing device or devices.

[0017] Existing codecs can employ reference frames to permit video streams to maintain video quality while reducing bandwidth requirements. Reference frames can be used by an encoder to encode some frames of the video stream using a reduced number of bits to represent the residuals or differences between the blocks of the frame and blocks of the reference frame or frames. This is sometimes called inter-frame motion prediction. A number of reference frames can be temporarily stored or buffered in memory at the encoder or decoder during encoding or decoding. The encoder can assign numbers or symbols to the reference frame(s) to indicate which reference frame can be used to encode which frame to permit the decoder to correctly decode the frames of the video stream.

[0018] The teachings herein provide techniques for managing a multi-reference picture buffer including reference frames for encoding and decoding. The terms reference frame and reference picture are used interchangeably herein, and they may represent only a portion of a frame. Picture IDs are assigned to reference frames to permit correct encoding and decoding of frames of the video stream. A maximum number of picture IDs can be determined based on the maximum number of supported reference buffers and a margin for error resiliency. Once a reference picture is assigned a picture ID and inserted into a picture buffer, it can stay in the buffer for any period of time without conflicting with incoming pictures. The word-length requirement of the picture ID can be short depending on the number of reference pictures and the strength of packet-loss robustness. Implementations of the teachings herein can be relatively straightforward and can save overhead bits used to maintain encoder and decoder buffers in sync even under packet loss conditions. Details of certain aspects are described herein initially with reference to the environments in which they can be implemented.

[0019] FIG. 1 is a schematic of a video encoding and decoding system 10. An exemplary transmitting station 12 can be, for example, a computer having an internal configuration of hardware including a processor such as a central processing unit (CPU) 14 and a memory 16. CPU 14 is a controller for controlling the operations of transmitting station

12. CPU 14 is connected to memory 16 by, for example, a memory bus. Memory 16 can be read only memory (ROM), random access memory (RAM) or any other suitable memory device. Memory 16 can store data and program instructions that are used by CPU 14. Other suitable implementations of transmitting station 12 are possible. For example, the processing of transmitting station 12 can be distributed among multiple devices.

[0020] A network 28 connects transmitting station 12 and a receiving station 30 for encoding and decoding of the video stream. Specifically, the video stream can be encoded in transmitting station 12 and the encoded video stream can be decoded in receiving station 30. Network 28 can, for example, be the Internet. Network 28 can also be a local area network (LAN), wide area network (WAN), virtual private network (VPN), a cellular telephone network or any other means of transferring the video stream from transmitting station 12 to, in this example, receiving station 30.

[0021] Receiving station 30, in one example, can be a computer having an internal configuration of hardware including a processor such as a CPU 32 and a memory 34. CPU 32 is a controller for controlling the operations of receiving station 30. CPU 32 is connected to memory 34 by, for example, a memory bus. Memory 34 can be ROM, RAM or any other suitable memory device. Memory 34 can store data and program instructions that are used by CPU 32. Other suitable implementations of receiving station 30 are possible. For example, the processing of receiving station 30 can be distributed among multiple devices.

[0022] A display 36 configured to display a video stream can be connected to receiving station 30. Display 36 can be implemented in various ways, including by a liquid crystal display (LCD), a cathode-ray tube (CRT), or a light emitting diode display (LED), such as an OLED display. Display 36 is coupled to CPU 32 and can be configured to display a rendering 38 of the video stream decoded by a decoder in receiving station 30.

[0023] Other implementations of encoder and decoder system 10 are possible. In some of the implementations described, for example, an encoder is in transmitting station 12 and a decoder is in receiving station 30 as instructions in memory or a component separate from memory. However, an encoder or decoder can be coupled to a respective station 12, 30 rather than in it. Further, one implementation can omit network 28 and/or display 36. In another implementation, a video stream can be encoded and then stored for transmission at a later time to receiving station 30 or any other device having memory. In one implementation, a video stream is received by the receiving station 30 (e.g., via network 28, a computer bus, and/or some communication pathway) and stored for later decoding. In another

implementation, additional components can be added to encoder and decoder system 10. For example, a display or a video camera can be attached to transmitting station 12 to capture the video stream to be encoded. In an exemplary implementation, a real-time transport protocol (RTP) is used for transmission of the encoded video. In another implementation, a transport protocol other than RTP may be used, e.g., an HTTP-based video streaming protocol.

[0024] FIG. 2 is a diagram of a typical video stream 50 to be encoded and subsequently decoded. Video stream 50 includes a video sequence 52. At the next level, video sequence 52 includes a number of adjacent frames 54. While three frames are depicted as adjacent frames 54, video sequence 52 can include any number of adjacent frames. Adjacent frames 54 can then be further subdivided into individual frames, e.g., a single frame 56. At the next level, single frame 56 can be divided into a series of blocks 58, which can contain data corresponding to, for example, 16x16 pixels in frame 56. Each block 58 can contain luminance and chrominance data for the corresponding pixels. Blocks 58 can also be of any other suitable size such as 16x8 pixel groups or 8x16 pixel groups and can be further subdivided into smaller blocks depending on the application. Unless otherwise noted, the terms block and macroblock are used interchangeably herein.

[0025] FIG. 3 is a block diagram of a video encoder 70 in accordance with an implementation. Encoder 70 can be implemented, as described above, in transmitting station 12 such as by providing a computer software program stored in memory 16, for example. The computer software program can include machine instructions that, when executed by CPU 14, cause transmitting station 12 to encode video data in the manner described in FIG. 3. Encoder 70 can also be implemented as specialized hardware included, for example, in transmitting station 12. Encoder 70 has the following stages to perform the various functions in a forward path (shown by the solid connection lines) to produce an encoded or a compressed bitstream 88 using input video stream 50: an intra/inter prediction stage 72, a transform stage 74, a quantization stage 76, and an entropy encoding stage 78. Encoder 70 may also include a reconstruction path (shown by the dotted connection lines) to reconstruct a frame for prediction and encoding of future blocks. In FIG. 3, encoder 70 has the following stages to perform the various functions in the reconstruction path: a dequantization stage 80, an inverse transform stage 82, a reconstruction stage 84, and a loop filtering stage 86. Other structural variations of encoder 70 can be used to encode video stream 50.

[0026] When video stream 50 is presented for encoding, each frame 56 within video stream 50 is processed in units of blocks. At intra/inter prediction stage 72, each block can be

encoded using either intra-frame prediction (within a single frame) or inter-frame prediction (from frame to frame). In either case, a prediction block can be formed. In the case of intra-prediction, a prediction block can be formed from samples in the current frame that have been previously encoded and reconstructed. In the case of inter-prediction, a prediction block can be formed from samples in one or more previously constructed reference frames.

[0027] Next, still referring to FIG. 3, the prediction block can be subtracted from the current block at intra/inter prediction stage 72 to produce a residual block (residual). Transform stage 74 transforms the residual into transform coefficients in, for example, the frequency domain. Examples of block-based transforms include the Karhunen-Loève Transform (KLT), the Discrete Cosine Transform (DCT), the Singular Value Decomposition Transform (SVD) and the Asymmetric Discrete Sine Transform (ADST). In one example, the DCT transforms the block into the frequency domain. In the case of DCT, the transform coefficient values are based on spatial frequency, with the lowest frequency (DC) coefficient at the top-left of the matrix and the highest frequency coefficient at the bottom-right of the matrix.

[0028] Quantization stage 76 converts the transform coefficients into discrete quantum values, which are referred to as quantized transform coefficients. The quantized transform coefficients are then entropy encoded by entropy encoding stage 78. The entropy-encoded coefficients, together with other information used to decode the block, which may include for example the type of prediction used, motion vectors and quantizer value, are then output to compressed bitstream 88. Compressed bitstream 88 can be formatted using various techniques, such as variable length encoding (VLC) and arithmetic coding. Compressed bitstream 88 can also be referred to as an encoded video stream and the terms will be used interchangeably herein.

[0029] The reconstruction path in FIG. 3 (shown by the dotted connection lines) can be used to ensure that both encoder 70 and a decoder 100 (described below) use the same reference frames to decode compressed bitstream 88. The reconstruction path performs functions that are similar to functions that take place during the decoding process that are discussed in more detail below, including dequantizing the quantized transform coefficients at dequantization stage 80 and inverse transforming the dequantized transform coefficients at inverse transform stage 82 to produce a derivative residual block (derivative residual). At reconstruction stage 84, the prediction block that was predicted at intra/inter prediction stage

72 can be added to the derivative residual to create a reconstructed block. Loop filtering stage 86 can be applied to the reconstructed block to reduce distortion such as blocking artifacts.

[0030] Other variations of encoder 70 can be used to encode compressed bitstream 88. For example, a non-transform based encoder 70 can quantize the residual signal directly without transform stage 74. In another implementation, encoder 70 can have quantization stage 76 and dequantization stage 80 combined into a single stage.

[0031] FIG. 4 is a block diagram of a video decoder 100 in accordance with another implementation of this disclosure. Decoder 100 can be implemented in receiving station 30, for example, by providing a computer software program stored in memory 34. The computer software program can include machine instructions that, when executed by CPU 32, cause receiving station 30 to decode video data in the manner described in FIG. 4. Decoder 100 can also be implemented as specialized hardware included, for example, in transmitting station 12 or receiving station 30.

[0032] Decoder 100, similar to the reconstruction path of encoder 70 discussed above, includes in one example the following stages to perform various functions to produce an output video stream 116 from compressed bitstream 88: an entropy decoding stage 102, a dequantization stage 104, an inverse transform stage 106, an intra/inter prediction stage 108, a reconstruction stage 110, a loop filtering stage 112 and a deblocking filtering stage 114. Other structural variations of decoder 100 can be used to decode compressed bitstream 88.

[0033] When compressed bitstream 88 is presented for decoding, the data elements within compressed bitstream 88 can be decoded by entropy decoding stage 102 to produce a set of quantized transform coefficients. Dequantization stage 104 dequantizes the quantized transform coefficients, and inverse transform stage 106 inverse transforms the dequantized transform coefficients to produce a derivative residual that can be identical to that created by inverse transform stage 82 in encoder 70. Using header information decoded from compressed bitstream 88, decoder 100 can use intra/inter prediction stage 108 to create the same prediction block as was created in encoder 70, e.g., at intra/inter prediction stage 72. At reconstruction stage 110, the prediction block can be added to the derivative residual to create a reconstructed block. Loop filtering stage 112 can be applied to the reconstructed block to reduce blocking artifacts. Deblocking filtering stage 114 can be applied to the reconstructed block to reduce blocking distortion, and the result is output as output video stream

116. Output video stream 116 can also be referred to as a decoded video stream and the terms will be used interchangeably herein.

[0034] Other variations of decoder 100 can be used to decode compressed bitstream 88. For example, decoder 100 can produce output video stream 116 without deblocking filtering stage 114.

[0035] FIG. 5 is a block diagram of a modified video encoder 500 illustrating encoding of an input video stream using multiple reference pictures through a reference picture list in accordance with an implementation. Encoder 500 can be implemented as a software program using a computing device such as transmitting station 12, for example. The software program can be stored in memory 16 and can include instructions that, when executed by CPU 14, perform encoding steps described below. Encoder 500 can also be implemented as specialized hardware included, for example, in transmitting station 12. Encoder 500 has the following components to perform various functions to produce encoded or compressed bitstream 88 using input video stream 50: an encode picture list component 502, an encoder component 504 and reference picture buffers 506. Encoder component 504 performs processing as described for encoder 70 in FIG. 3 above, accepting as input a video stream 50 and outputting compressed bitstream 88. Encoder component 504 communicates with reference picture buffers 506 in cooperation with encode picture list component 502. Encoder component 504 can use the reference picture buffers 506 to temporarily store reference frames for use in inter-frame motion prediction using/at inter/intra prediction stage 72, for example.

[0036] Encode picture list component 502 manages reference picture buffers 506 and can keep track of which reference frames are currently being stored in which buffer in reference picture buffers 506, for example. Encode picture list component 502 assigns and encodes picture IDs and optionally encodes reference picture lists (e.g. lists of the reference picture IDs) to be included in encoded video stream 88. Details of assigning and encoding picture IDs are described below. Reference picture buffers 506 and encode picture list component 502 can also be incorporated into existing stages/components of encoder 70, such as in inter/intra prediction stage 72.

[0037] FIG. 6 is a block diagram of a modified video decoder 600 illustrating decoding of a compressed bitstream including picture IDs and in some instances a reference picture list in accordance with an implementation. Decoder 600 can be implemented as a software program using a computing device such as receiving station 30, for example. The

software program can be stored in memory 34 and can include instructions that, when executed by CPU 32, perform decoding steps described below. Decoder 600 can also be implemented as specialized hardware included, for example, in receiving station 30. Decoder 600 has the following components to perform various functions to produce decoded or output video stream 116 from compressed bitstream 88: a decode picture list component 602, a decoder component 604 and reference picture buffers 606. Decoder component 604 performs processing as described for decoder 100 in FIG. 4 above, accepting as input encoded video stream 88 and outputting an output video stream 116. Decoder component 604 communicates with reference picture buffers 606 in cooperation with decode picture list component 602. Decoder component 604 can use reference picture buffers 606 to temporarily store reference frames for use in inter-frame motion prediction using/at inter/intra prediction stage 108, for example.

[0038] Decode picture list component 602 can manage the reference picture buffers 606 and keep track of which reference frames are currently being stored in which buffer in reference picture buffers 606, for example. Decode picture list component 602 decodes picture IDs and optionally reference picture lists included in encoded video stream 88. Reference picture buffers 606 and decode picture list component 602 can also be incorporated into existing stages/components of decoder 100, such as in inter/intra prediction stage 108.

[0039] FIG. 7 is a flowchart diagram illustrating a process 700 of managing a multi-reference picture buffer and encoding reference pictures in an encoded video stream in accordance with an implementation. Process 700 can be implemented as a software program associated with encoder 500 using a computing device such as transmitting station 12, for example. The software program can be stored in memory 16 and can include instructions that, when executed by CPU 14, perform encoding steps described below. Process 700 can also be implemented as specialized hardware included, for example, in transmitting station 12. Process 700 can be implemented by the steps described below; however, the order in which the steps are performed may differ in different implementations and certain steps may occur concurrently. Further, steps may also be omitted, combined or added in certain implementations.

[0040] Process 700 employs a dedicated picture ID or number `pic_id` as a unique identification for a frame, sequence, slice or segment used as a reference in the reference picture buffers. The picture number `pic_id` may be assigned to a sequence header, picture

header, slice header, segment header or any synchronization point within the video stream. For ease of explanation, the video stream data used as a reference and assigned a `pic_id` will be referred to as a frame. However, any defined portion of a video stream can be used.

[0041] At step 702, a number `Nrb` of reference picture buffers are allocated in memory of a computing device used for the encoding processes, for example reference picture buffers 506. The number `Nrb` of reference picture buffers to allocate can be predetermined to be compatible with the encoder used, for example. For example, the number `Nrb` of reference picture buffers can be a maximum number of reference buffers supported by encoder 500. The term “allocate” as used herein means to assign, apportion, designate, provide or in any manner whatsoever allocate memory locations for the stated use.

[0042] At step 704, an expected frame error length `Mex` is estimated. `Mex` can be considered as a margin added for error resiliency. `Mex` can represent, for example, the expected number of consecutive frames that can be lost due to transmission or storage errors. Estimation can be performed based on previously gathered statistics, analysis of current network conditions, a predetermined value or any combination of these. As used herein, the term “estimate” can mean approximate, project, measure or in any manner whatsoever estimate a value or values. Alternatively, `Mex` can simply be assigned a positive number.

[0043] At step 706, the picture numbers `pic_id` are allocated. When the `pic_id` is used only for identifying pictures and thus reference buffers, a picture can be assigned a `pic_id` value as long as the value is uniquely identifiable from other picture IDs in the reference buffer. When the number of reference picture buffers is `Nrb`, the `pic_id` value can range from 0 to `PIDmax`, where `PIDmax` is a maximum number of picture numbers `pic_id`. The maximum number `PIDmax` of picture numbers `pic_id` is equal to or larger than the number of reference buffers `Nrb`. Accordingly, the maximum number `PIDmax` of picture numbers `pic_id` can be the sum of the number of reference picture buffers and the margin `Mex`, minus 1 to account for a 0 value. This can be calculated according to the formula:

$$PIDmax = Nrb + Mex - 1.$$

Accordingly, the maximum number `PIDmax` is fixed at a relatively small finite number.

[0044] The value range or the word-length of the picture number `pic_id` can be defined or signaled in the mentioned headers or through other means. The value of `pic_id` can be coded in the bitstream with a fixed word-length code or variable length code. For example, the `pic_id` value can also be represented by a variable length code such as an Exp-Golomb code. The value of `pic_id` can be signed or unsigned. In an exemplary implementation

described herein, the value of `pic_id` is unsigned (e.g. only represented by positive values). For example, with `Nrb = 6`, `Mex = 10` and `PIDmax = 15`, `pic_id` can assume values from 0 to 15 and can be represented by a 4-bit fixed-length code word. Note that the maximum value of `pic_id` can be, in certain implementations, larger than the number of reference buffers, e.g., for error robustness, so that a `pic_id` will not replicate within a period of certain pictures.

[0045] At step 708, reference pictures of the video stream can be stored at reference picture buffers, e.g., as needed. In this example, each reference picture is stored in a single buffer. Frames are stored temporarily in the reference picture buffer to permit encoding frames of the video stream using, for example, inter-frame motion prediction. Reference frames can be deleted from the reference picture buffer when they are no longer used to encode other frames in the video stream to make room for other reference frames.

[0046] At step 710, when a reference frame is encoded it will be assigned a picture number `pic_id` to be included in the picture header and possibly other headers such as a slice header, for example, for error resiliency. When this coded picture is to be stored in the reference picture buffer for future reference, the buffer at which the picture is stored will be marked with its picture number `pic_id` for identification purposes. As used herein the term “marked” can mean designated, signaled, indicated or otherwise marked in any manner whatsoever.

[0047] Before assigning a value of `pic_id` to reference frames and reference picture buffers, an array is created to represent the picture IDs of the number `Nrb` of allocated reference picture buffers. Where the array is referred to as `pic_buffer_id[Nrb]`, it can be initialized as follows:

`pic_buffer_id[i] = -1`, for `i = 0` to `Nrb-1`.

Initializing in this way indicates that the *i*th buffer is “unused for reference” and is free to store other pictures. As reference frames are added to the reference picture buffer, the value in the `pic_buffer_id` array is set to the value of the `pic_id` assigned to the frame, so `pic_buffer_id[i] = PID` means that the *i*th buffer is occupied by a frame with `pic_id = PID`, where `PID` is in the range from 0 to `PIDmax`.

[0048] At step 712, as a reference frame is no longer used for encoding frames, that reference frame may be removed from the reference picture buffer. The `pic_buffer_id[]` entry for that reference picture buffer is set to -1 to indicate that the reference picture buffer is free. To keep track of which `pic_id` values are no longer in use in the reference picture buffer, a first-in-first-out buffer (`PIDfifo[]`) can be constructed and initialized as follows:

$PIDfifo[i] = -1$, for $i = 0$ to $Mex - 1$.

Accordingly, the size of array $PIDfifo[]$ is Mex .

[0049] When the encoder starts, the first Nrb coded pic_ids can be assigned any value from 0 to $PIDmax$, as long as there is no replication. The first Nrb coded reference frames can be stored in the Nrb reference picture buffers. Each of these buffers will be assigned a pic_id corresponding to the pic_id of the reference frame stored at that buffer. For example, the first Nrb frames can be assigned pic_ids from 0 to $Nrb - 1$, respectively.

[0050] For follow-on coded reference pictures, as new reference pictures are used, pic_ids can be assigned to the reference pictures from the unused pic_ids as they are stored in the reference picture buffers. As reference pictures are no longer used for prediction, the buffers they were assigned to are freed and their pic_ids are entered into the $PIDfifo$, e.g., at step 714, to free up one or more reference picture buffers. For example, the pic_id value for a reference picture buffer that is no longer used for prediction can be entered into the array $PIDfifo[]$ until all of the unused pic_ids are assigned to the $PIDfifo$. At this point, all of the pic_ids have been used at least once. The array $PIDfifo[]$ can be used to keep track of pic_id reuse to insure that the least recently used pic_id is assigned to a reference picture buffer to prevent confusion at the decoder in case of missing or corrupt frames or pic_ids .

[0051] At step 716, one or more frames to be used as reference pictures are assigned to reference picture buffers and any buffers having frames no longer used for prediction are freed by assigning a value of -1 to the reference picture buffer and entering the pic_id of the no longer used frame into the array $PIDfifo[]$. At step 718, a pic_id can be assigned to the reference pictures in the reference buffers from the array $PIDfifo[]$ and the pic_id from the unused frame is entered into the array $PIDfifo[]$ as follows:

$LastPIDfifo = PIDfifo[Mex - 1]$;

$PIDfifo[i + 1] = PIDfifo[i]$, for $i = 0$ to $Mex - 1$; and

$PIDfifo[0] = PIDremoved$;

wherein $LastPIDfifo$ is the least recently used pic_id to be assigned to the reference frame and the reference picture buffer or array $pic\_buffer\_id[]$; and $PIDremoved$ is the pic_id of the frame that is no longer used for prediction and is most recently freed from the reference picture buffer.

[0052] As can be seen, the least recently used pic_id is shifted out of the $PIDfifo[]$ at location $Mex - 1$, each pic_id in the array $PIDfifo$ is moved up one index, and the most recently freed value for pic_id is pushed onto the $PIDfifo$ at index 0.

[0053] At step 718, a re-used pic_id can be selected for assignment to a reference frame and a reference picture buffer in which the reference frame is stored. If LastPIDfifo = -1, then the encoder can choose any PID value that is not the same as in the arrays of pic_buffer_id[] and PIDfifo[]. For example, the encoder may continue to increase the pic_id value by "1" (with a modulo operation with a wrap-around value at PIDmax) from the last picture ID. If LastPIDfifo does not equal -1, LastPIDfifo can be used as the value for pic_id for a reference frame and the associated reference buffer.

[0054] Following the above process, any picture may stay in the reference buffer perpetually without worrying about the possible conflict with the picture ID of the incoming pictures. Any picture ID that is removed from the reference buffer is not used within Mex number of coded pictures in this implementation.

[0055] At step 720, the frames are encoded by the encoder, and the pic_ids and optionally the pic_buffer_id[] array are encoded into the encoded video stream along with the frames for transmission or storage at step 722. The value of pic_id for each reference frame and the pic_buffer_id[] information are used to later decode the frames.

[0056] FIG. 8 is a flowchart diagram illustrating a process 800 of decoding frames using a reference picture buffer in accordance with an implementation. Process 800 can be implemented as a software program using a computing device such as receiving station 30, for example. The software program can be stored in memory 34 and can include instructions that, when executed by CPU 32, perform the decoding steps described below. Process 800 can also be implemented as specialized hardware included, for example, in receiving station 30. Process 800 can be implemented by the steps described below; however, the order in which the steps are performed may differ in different implementations and certain steps may occur concurrently. Further, steps may also be omitted, combined or added in certain implementations.

[0057] At step 802, an encoded video stream such as compressed bitstream 88 can be received at a computing device, for example receiving station 30. Herein, received means to read the data from a network that could be wired or wireless, from memory, or from a disk drive or removable media such as a CompactFlash (CF) card, Secure Digital (SD) card, or the like. Reference frames extracted from or generated by data from the video stream can be written to reference picture buffers 606 at decoder 600, for example, to be used to decode further frames.

[0058] At step 804, the decoder can decode Mex from the encoded video stream (e.g., compressed bitstream 88). The decoder can also decode in step 804 the picture numbers pic_ids and any reference picture lists included in the encoded video stream, e.g., in the picture header and/or other headers such as slice headers. The decoding process has been described generally above and can be the inverse of the encoding process at least in part.

[0059] At step 806, frames within the encoded video stream are decoded. This decoding includes decoding and/or generating one or more reference frames. At step 808, the reference frame is added to reference picture buffers of the decoder, such as buffers 606. In this description, each reference picture is assigned to its own buffer, so each buffer is assigned the pic_id of the frame stored therein. If a reference picture list is also included in the bitstream, the received reference picture list can be compared to a reference picture list generated at the decoder based on the received pic_ids to determine if a reference frame has been lost or corrupted, for example, by a transmission error or to determine if a reference frame buffer can be freed. Each decoder buffer may be marked “used for reference” or “unused for reference” based on the status of the reference picture within that buffer. This status can be received from the reference picture list referenced above or from other mechanisms that indicate the removal of a picture store in a reference buffer, e.g., Memory Management Control Operations (MMCOs). A decoded reference picture can be stored in any buffer that is marked as “unused for reference” or can be stored in other buffers through syntax elements.

[0060] At step 810, the error rate of the received video stream is determined. The error rate can be determined by checking for missing pic_ids as described above or from transport level information such as missing packets or Picture-Order-Count (POC) techniques, for example. At step 812, the determined error rate is compared to the received maximum expected error rate Mex. If the determined error rate is greater than the expected error rate, the number of missing frames may exceed the maximum number of missing frames that process 800 can handle without error. In this case, process 800 can take corrective action at step 814 by, for example, signaling the encoder that many frames have been lost in a row and the decoder buffers and encoder buffers have lost synchronization. The encoder may send an intra-frame to re-establish encoder-decoder buffer synchronization. The terms “signaling” or “sending” as used herein can mean indicating, communicating, signing or otherwise signaling in any manner whatsoever. Process 800 can repeat until all frames are decoded.

[0061] In brief, for example, an apparatus for decoding a video stream having a plurality of frames may include a memory and a processor configured to execute instructions stored in the memory to receive an encoded video stream, decode an expected error value representing possible frame errors, allocate reference picture buffers based on the expected error value, decode a plurality of reference frames from the video stream, add each of the plurality of reference frames to the reference picture buffers up to a maximum value, determine an encoded video stream error value representing actual frame errors based on the decoded plurality of reference frames, and if the encoded video stream error value is greater than the decoded expected error value, signaling an error. The processor can also be configured to decode one or more frames of the plurality of frames using the reference frames to perform intra-frame motion prediction and/or to decode picture IDs and decode a number of picture IDs. The picture IDs can be decoded from a sequence header, picture header, slice header or segment header, for example.

[0062] According to the teachings herein, a maximum picture ID number can be calculated using a number of buffers or storage locations available to temporarily store reference frames plus an extra value based on the possible number of frames that could be lost or corrupted due to unreliable network transmission. Calculating a maximum picture ID number and re-using picture ID numbers can limit the number of bits required to code the picture IDs in the bitstream for buffer management purposes. Further, it is possible to keep track of which picture ID numbers are in use to designate which frames are assigned to which buffers. As reference frames are no longer in use in a buffer, the picture ID number designating the reference frame can be recycled for use by a subsequent reference frame. The unused picture ID numbers can be kept in a first-in-first-out (FIFO) buffer in order to enforce a least-recently-used policy for re-using picture ID numbers.

[0063] As described, if picture loss is no more than Mex number of pictures in a row, the good reference pictures in the decoder buffers can be confidently used for prediction. A good reference picture is defined as a picture's worth of intact bitstream with good references. If the bitstream is corrupted or lost, or if reference pictures to be used for decoding the bistream do not exist or are corrupted, the decoded picture may be considered a bad reference picture. The number of picture losses in a row may be determined by other means, such as through picture numbers, transport level information, etc.

[0064] The aspects and implementations of encoding and decoding described above illustrate some exemplary encoding and decoding techniques. However, it is to be understood

that encoding and decoding, as those terms are used in the claims, could mean compression, decompression, transformation, or any other processing or change of data.

[0065] The words “example” or “exemplary” are used herein to mean serving as an example, instance, or illustration. Any aspect or design described herein as “example” or “exemplary” is not necessarily to be construed as preferred or advantageous over other aspects or designs. Rather, use of the words “example” or “exemplary” is intended to present concepts in a concrete fashion. As used in this application, the term “or” is intended to mean an inclusive “or” rather than an exclusive “or”. That is, unless specified otherwise, or clear from context, “X includes A or B” is intended to mean any of the natural inclusive permutations. That is, if X includes A; X includes B; or X includes both A and B, then “X includes A or B” is satisfied under any of the foregoing instances. In addition, the articles “a” and “an” as used in this application and the appended claims should generally be construed to mean “one or more” unless specified otherwise or clear from context to be directed to a singular form. Moreover, use of the term “an embodiment” or “one embodiment” or “an implementation” or “one implementation” throughout is not intended to mean the same embodiment or implementation unless described as such.

[0066] Implementations of transmitting station 12 and/or receiving station 30 (and the algorithms, methods, instructions, etc. stored thereon and/or executed thereby, including by encoder 70 and decoder 100) can be realized in hardware, software, or any combination thereof. The hardware can include, for example, computers, intellectual property (IP) cores, application-specific integrated circuits (ASICs), programmable logic arrays, optical processors, programmable logic controllers, microcode, microcontrollers, servers, microprocessors, digital signal processors or any other suitable circuit. In the claims, the term “processor” should be understood as encompassing any of the foregoing hardware, either singly or in combination. The terms “signal” and “data” are used interchangeably. Further, portions of transmitting station 12 and receiving station 30 do not necessarily have to be implemented in the same manner.

[0067] Further, in one embodiment, for example, transmitting station 12 or receiving station 30 can be implemented using a general purpose computer or general purpose processor with a computer program that, when executed, carries out any of the respective methods, algorithms and/or instructions described herein. In addition or alternatively, for example, a special purpose computer/processor can be utilized which can contain specialized hardware for carrying out any of the methods, algorithms, or instructions described herein.

[0068] Transmitting station 12 and receiving station 30 can, for example, be implemented on computers in a video conferencing system. Alternatively, transmitting station 12 can be implemented on a server and receiving station 30 can be implemented on a device separate from the server, such as a hand-held communications device. In this instance, transmitting station 12 can encode content using encoder 70 into an encoded video signal and transmit the encoded video signal to the communications device. In turn, the communications device can then decode the encoded video signal using decoder 100. Alternatively, the communications device can decode content stored locally on the communications device, for example, content that was not transmitted by transmitting station 12. Other suitable transmitting station 12 and receiving station 30 implementation schemes are available. For example, receiving station 30 can be a generally stationary personal computer rather than a portable communications device and/or a device including encoder 70 may also include decoder 100.

[0069] Further, all or a portion of embodiments of the present invention can take the form of a computer program product accessible from, for example, a tangible computer-usable or computer-readable medium. A computer-usable or computer-readable medium can be any device that can, for example, tangibly contain, store, communicate, or transport the program for use by or in connection with any processor. The medium can be, for example, an electronic, magnetic, optical, electromagnetic, or a semiconductor device. Other suitable mediums are also available.

[0070] The above-described embodiments, implementations and aspects have been described in order to allow easy understanding of the present invention and do not limit the present invention. On the contrary, the invention is intended to cover various modifications and equivalent arrangements included within the scope of the appended claims, which scope is to be accorded the broadest interpretation so as to encompass all such modifications and equivalent structure as is permitted under the law.

What is claimed is:

1. A method for encoding a video stream having a plurality of frames, comprising:
 - identifying an allocated number of reference frame buffers;
 - calculating a maximum number of picture IDs as equal to or greater than the allocated number;
 - assigning, using a processor, a picture ID to a respective reference frame assigned to a respective reference frame buffer of a plurality of reference frame buffers to no higher than the maximum number of picture IDs, each picture ID having a value not assigned to any other reference frames assigned to the plurality of reference frame buffers;
 - freeing the picture ID when its associated reference frame is no longer assigned to the reference frame buffer;
 - periodically re-using a freed picture ID when assigning the picture IDs;
 - encoding one or more frames of the plurality of frames using the reference frames assigned to the plurality of reference frame buffers; and
 - including the picture IDs in the video stream.
2. The method of claim 1 wherein periodically re-using the freed picture ID when assigning the picture IDs comprises assigning a least recently used picture ID for a new reference frame assigned to one of the plurality of reference frame buffers.
3. The method of claim 1 or claim 2 wherein assigning a least recently used picture ID is performed using a first-in-first-out array.
4. The method of claim 1 or claim 2, further comprising:
 - estimating an expected error value representing possible frame errors; wherein the maximum number of picture IDs is equal to the allocated number plus the expected error value.
5. The method of claim 1 or claim 2 wherein including the picture IDs in the video stream further comprises:

encoding the picture ID in a sequence header, picture header, slice header or segment header.

6. An apparatus for encoding a video stream having a plurality of frames, comprising:

a memory;

a processor configured to execute instructions stored in the memory to:

identify an allocated number of reference frame buffers;

calculate a maximum number of picture IDs as equal to or greater than the allocated number;

assign using a processor a picture ID to a respective reference frame assigned to a respective reference frame buffer of a plurality of reference frame buffers to no higher than the maximum number of picture IDs, each picture ID having a value not assigned to any other reference frames assigned to the plurality of reference frame buffers;

free the picture ID when its associated reference frame is no longer assigned to the reference frame buffer;

periodically re-use a freed picture ID when assigning the picture IDs;

encode one or more frames of the plurality of frames using the reference frames assigned to the plurality of reference frame buffers; and

include the picture IDs in the video stream.

7. The apparatus of claim 6 wherein the processor is configured to periodically re-use the freed picture ID when assigning the picture IDs by assigning a least recently used picture ID for a new reference frame assigned to one of the plurality of reference frame buffers.

8. The apparatus of claim 6 or claim 7 wherein the processor is configured to assign a least recently used picture ID using a first-in-first-out array.

9. The apparatus of claim 6 or claim 7 wherein the processor is configured to:

estimate an expected error value representing possible frame errors; and wherein the maximum number of picture IDs is equal to the allocated number plus the expected error value.

10. The apparatus of claim 6 or claim 7 wherein the processor is configured to:

encode one or more frames of the plurality of frames using the reference frames assigned to the plurality of reference frame buffers to perform inter-frame motion prediction.

11. A method for decoding an encoded video stream having a plurality of frames, comprising:

receiving an encoded video stream;

decoding an expected error value representing possible frame errors using a processor;

allocating reference picture buffers based on the expected error value;

decoding a plurality of reference frames from the video stream;

adding each of the plurality of reference frames to the reference picture buffers up to a maximum value;

determining an encoded video stream error value representing actual frame errors based on the decoded plurality of reference frames; and

if the encoded video stream error value is greater than the decoded expected error value, signaling an error.

12. The method of claim 11, further comprising:

decoding one or more frames of the plurality of frames using the reference frames to perform inter-frame motion prediction.

13. The method of claim 11 or claim 12, further comprising:

decoding picture IDs using the processor.

14. The method of claim 13, further comprising:

decoding a number of the picture IDs.

1/7

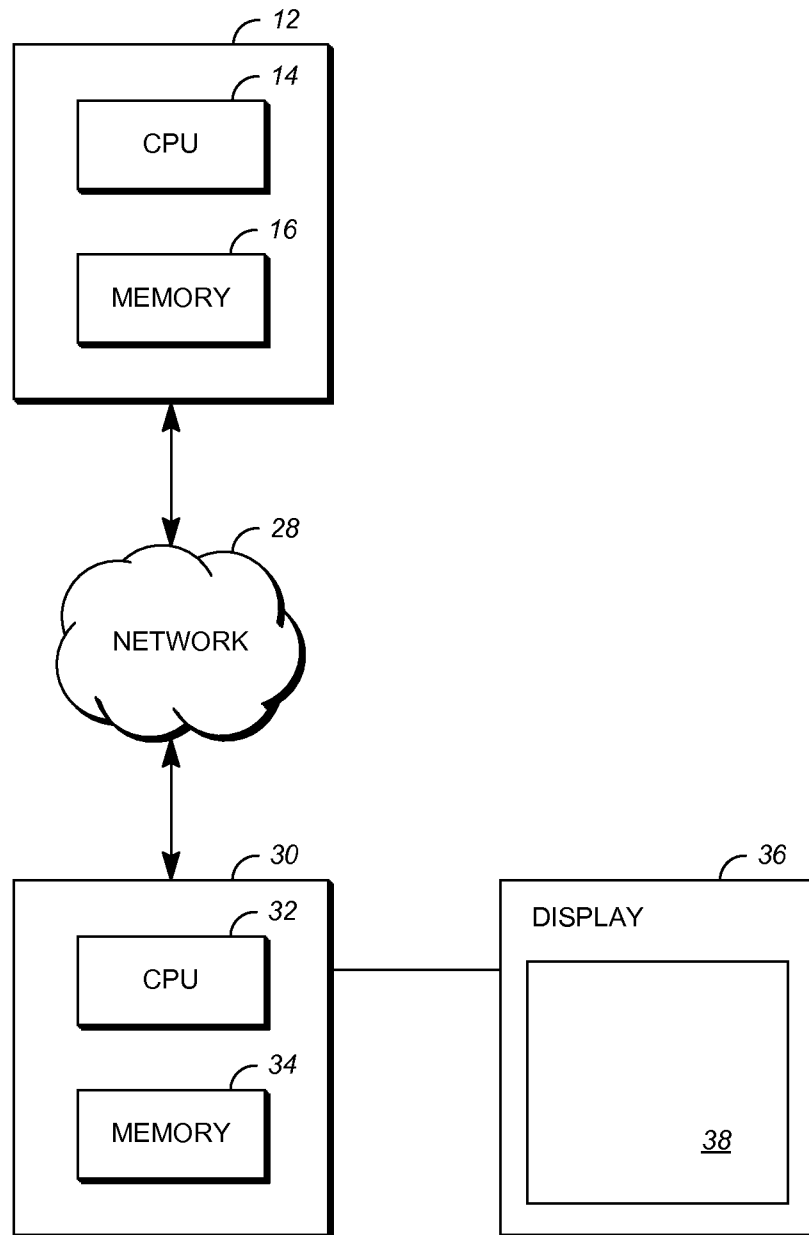
10

FIG. 1

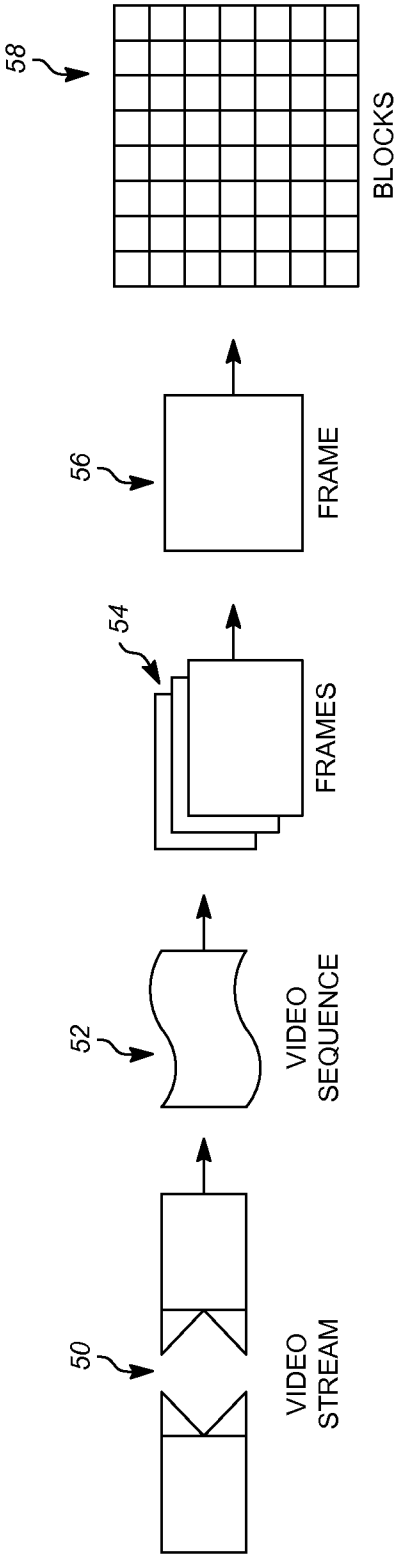


FIG. 2

3/7

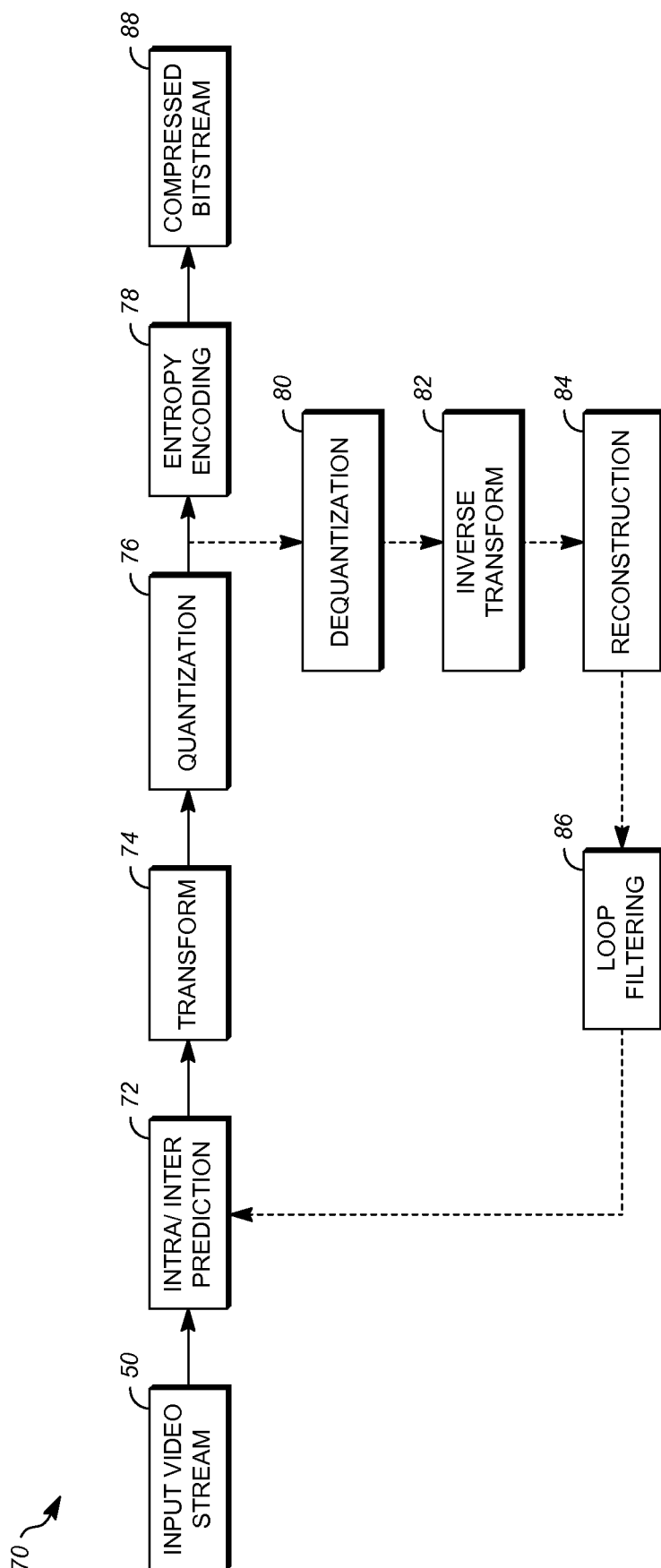


FIG. 3

4/7

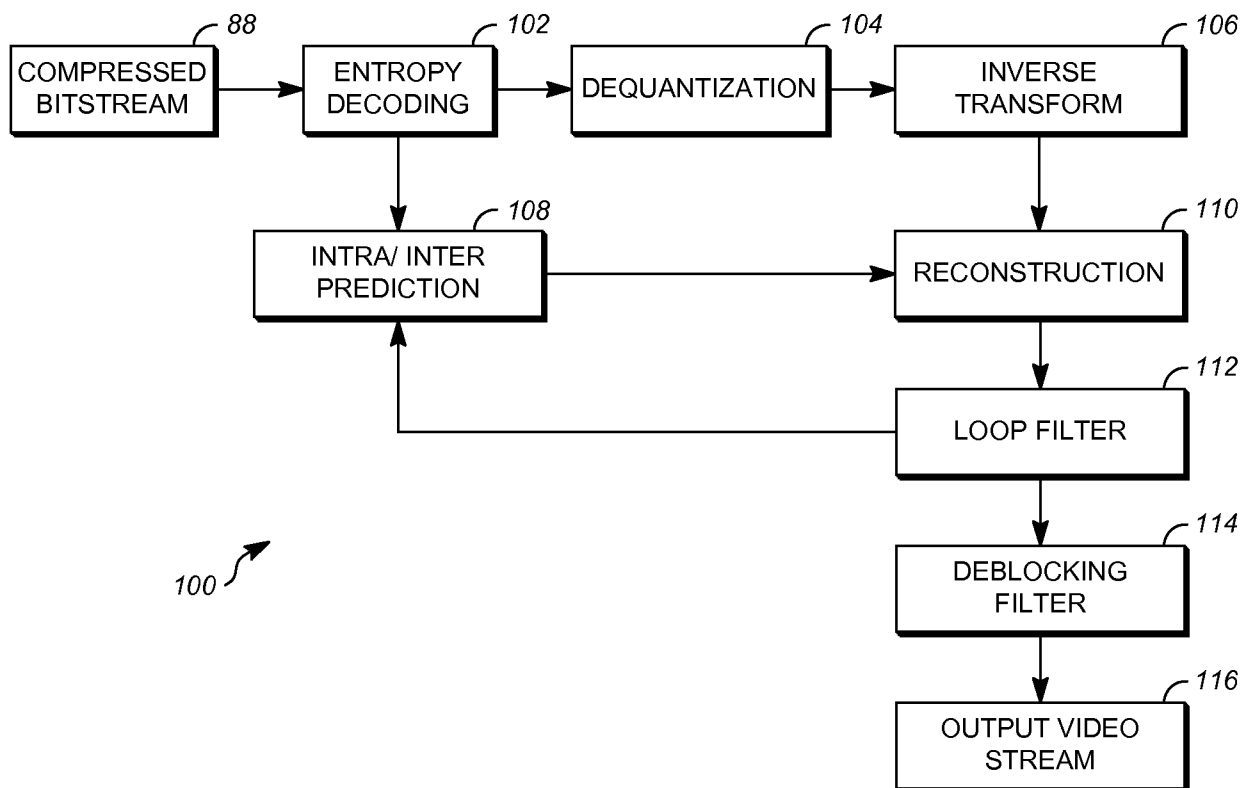


FIG. 4

5/7

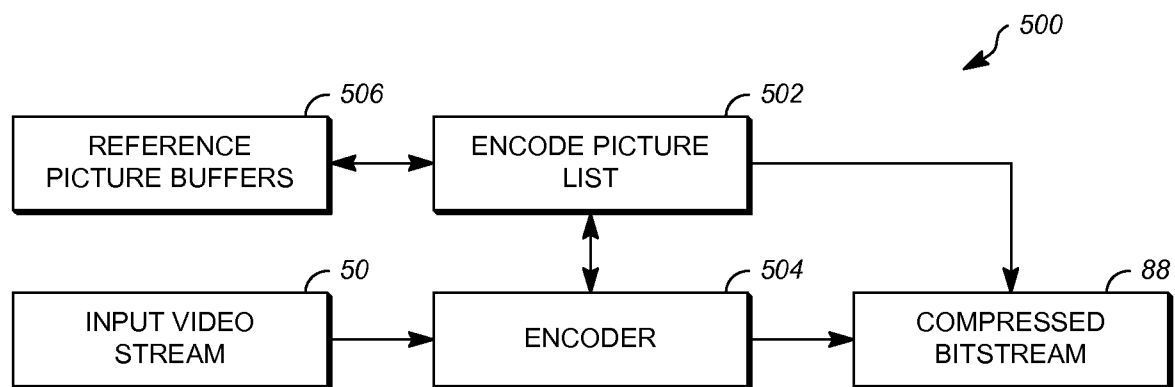


FIG. 5

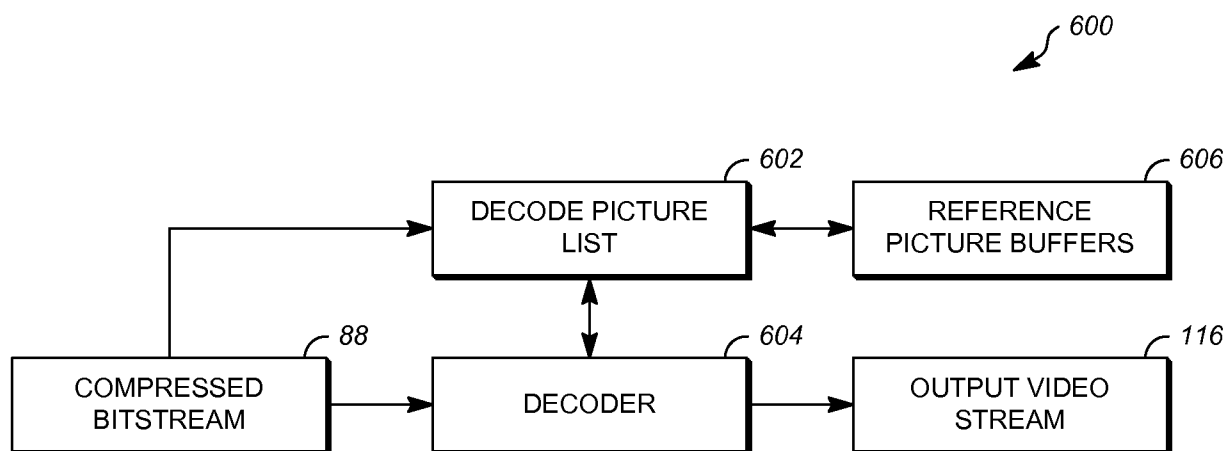


FIG. 6

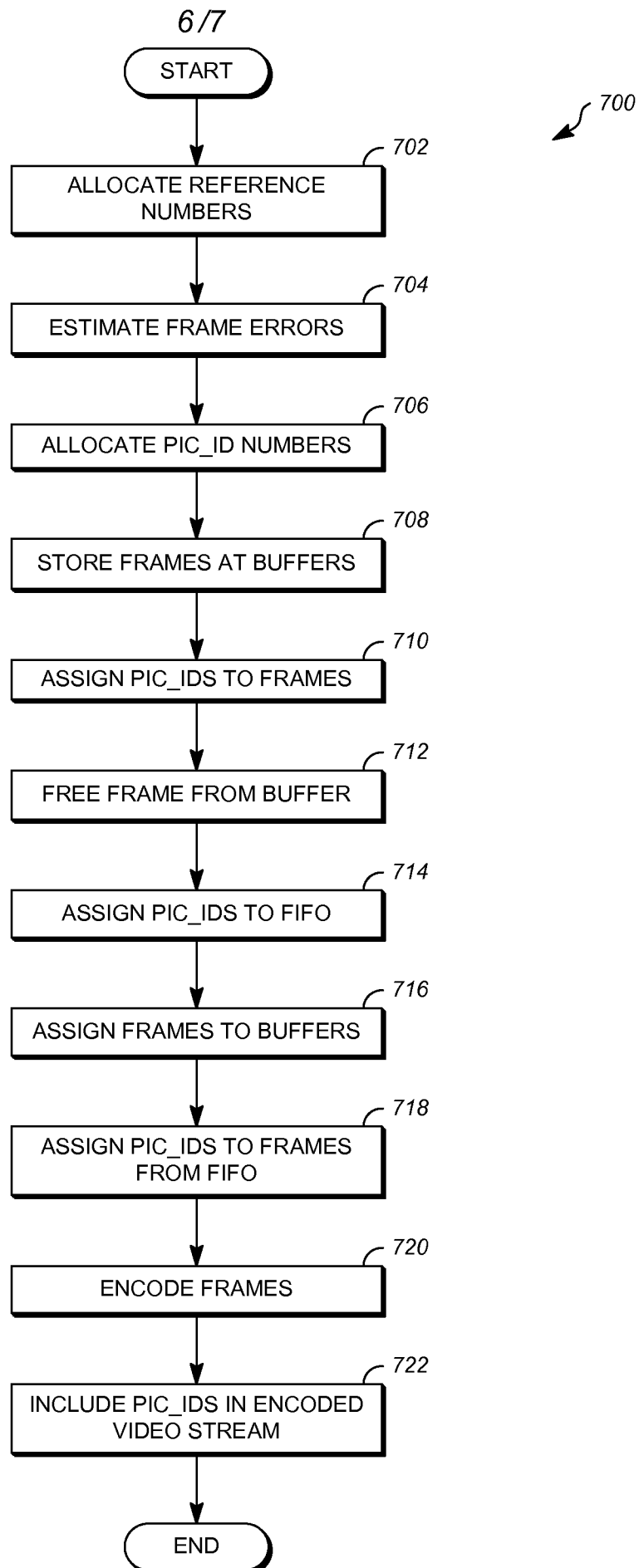


FIG. 7

7/7

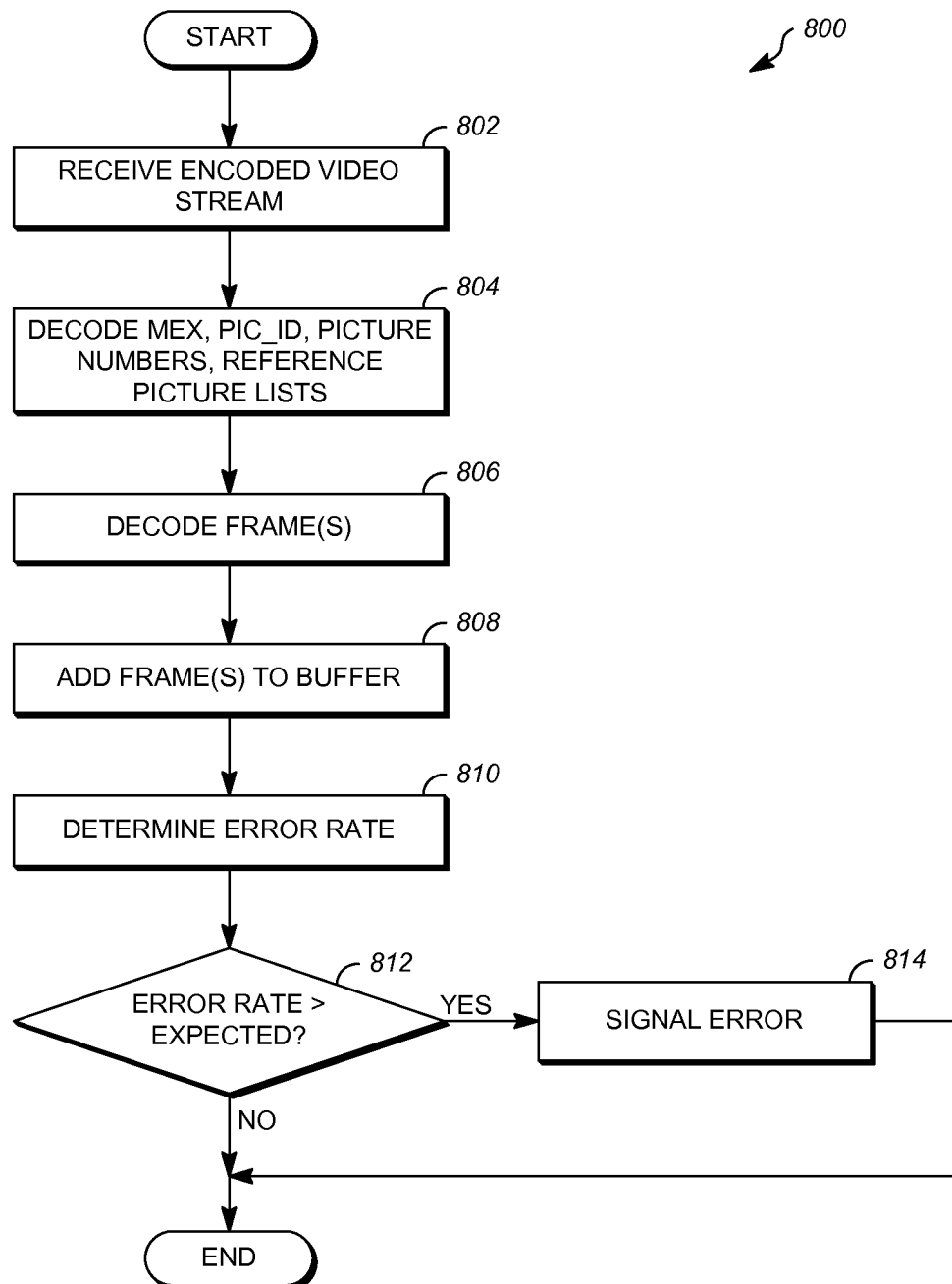


FIG. 8