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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
7 August 2014 (07.08.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/120575 A1

(51) International Patent Classification:
H04N 19/91 (2014.01) *H04N 19/13* (2014.01)

(21) International Application Number:
PCT/US2014/013027

(22) International Filing Date:
24 January 2014 (24.01.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/758,314 30 January 2013 (30.01.2013) US

(71) Applicant *(for all designated States except US)*: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, California 95054 (US).

(72) Inventors; and

(71) Applicants *(for US only)*: **PURI, Atul** [US/US]; 16080 NE 85th Street, Apt N204, Redmond, Washington 98052 (US). **GOKHALE, Neelesh N.** [US/US]; 1825 10th Ave W APT C, Seattle, Washington 98119 (US). **SOCEK, Daniel** [US/US]; 20125 NE 25th Avenue, Miami, Florida 33180 (US).

(74) Agent: **GREEN, Blayne D.**; Lynch Law Patent Group PC, C/O CPA Global, P.O. Box 52050, Minneapolis, Minnesota 55402 (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, 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, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM,

[Continued on next page]

(54) Title: CONTENT ADAPTIVE ENTROPY CODING OF PARTITIONS DATA FOR NEXT GENERATION VIDEO

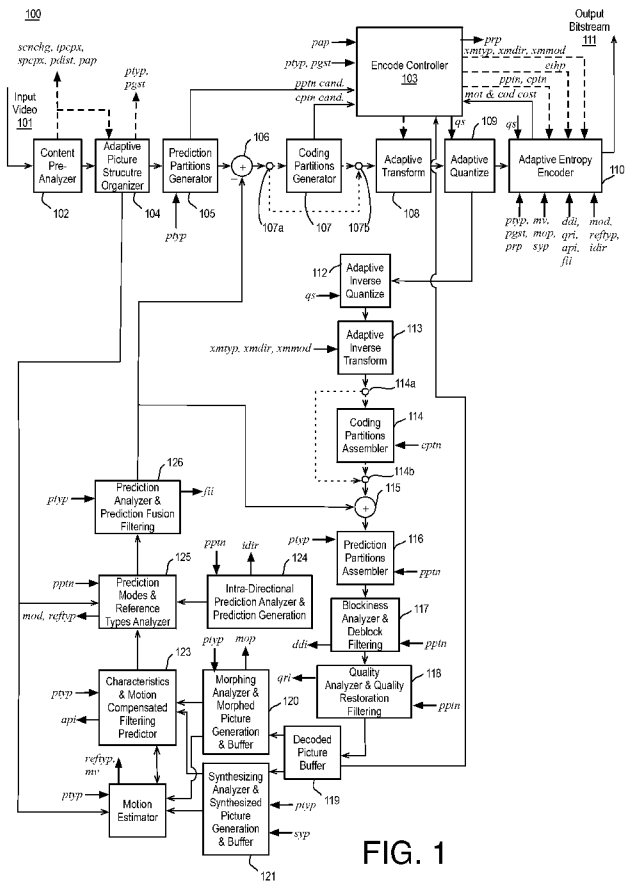


FIG. 1

(57) Abstract: Techniques related to content adaptive entropy coding of partitions data are described.

WO 2014/120575 A1

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, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

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

CONTENT ADAPTIVE ENTROPY CODING OF PARTITIONS DATA FOR NEXT GENERATION VIDEO

RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Application No. 61/758,314 filed 30 JAN 2013 and titled "NEXT GENERATION VIDEO CODING", the contents of which are hereby incorporated by reference in their entirety.

BACKGROUND

A video encoder compresses video information so that more information can be sent over a given bandwidth. The compressed signal may then be transmitted to a receiver having a decoder that decodes or decompresses the signal prior to display.

High Efficient Video Coding (HEVC) is the latest video compression standard, which is being developed by the Joint Collaborative Team on Video Coding (JCT-VC) formed by ISO/IEC Moving Picture Experts Group (MPEG) and ITU-T Video Coding Experts Group (VCEG). HEVC is being developed in response to the previous H.264/AVC (Advanced Video Coding) standard not providing enough compression for evolving higher resolution video applications. Similar to previous video coding standards, HEVC includes basic functional modules such as intra/inter prediction, transform, quantization, in-loop filtering, and entropy coding.

The ongoing HEVC standard may attempt to improve on limitations of the H.264/AVC standard such as limited choices for allowed prediction partitions and coding partitions, limited allowed multiple references and prediction generation, limited transform block sizes and actual transforms, limited mechanisms for reducing coding artifacts, and inefficient entropy encoding techniques. However, the ongoing HEVC standard may use iterative approaches to solving such problems.

For instance, with ever increasing resolution of video to be compressed and expectation of high video quality, the corresponding bitrate/bandwidth required for coding using existing video coding standards such as H.264 or even evolving standards such as H.265/HEVC, is relatively high. The aforementioned standards use expanded forms of traditional approaches to implicitly address the insufficient compression/quality problem, but often the results are limited.

The present description, developed within the context of a Next Generation Video (NGV) codec project, addresses the general problem of designing an advanced video codec that maximizes the achievable compression efficiency while remaining sufficiently

practical for implementation on devices. For instance, with ever increasing resolution of video and expectation of high video quality due to availability of good displays, the corresponding bitrate/bandwidth required using existing video coding standards such as earlier MPEG standards and even the more recent H.264/AVC standard, is relatively high. H.264/AVC was not perceived to be providing high enough compression for evolving higher resolution video applications.

BRIEF DESCRIPTION OF THE DRAWINGS

The material described herein is illustrated by way of example and not by way of limitation in the accompanying figures. For simplicity and clarity of illustration, elements illustrated in the figures are not necessarily drawn to scale. For example, the dimensions of some elements may be exaggerated relative to other elements for clarity. Further, where considered appropriate, reference labels have been repeated among the figures to indicate corresponding or analogous elements. In the figures:

FIG. 1 is an illustrative diagram of an example next generation video encoder;

FIG. 2 is an illustrative diagram of an example next generation video decoder;

FIG. 3(a) is an illustrative diagram of an example next generation video encoder and subsystems;

FIG. 3(b) is an illustrative diagram of an example next generation video decoder and subsystems;

FIG. 4 is an illustrative diagram of an example entropy encoder module;

FIG. 5 is an illustrative diagram of an example entropy decoder module;

FIG. 6 is an illustrative diagram of an example entropy encoder module;

FIG. 7 is an illustrative diagram of an example entropy decoder module;

FIG. 8 is a flow diagram illustrating an example process;

FIG 9 illustrates an example bitstream;

FIG. 10 is a flow diagram illustrating an example process;

FIG. 11 is an illustrative diagram of an example video coding system and video coding process in operation;

FIG. 11 is an illustrative diagram of an example video coding system;

FIG. 12 is a flow diagram illustrating an example encoding process;

FIG. 13 illustrates an example bitstream;

FIG. 14 is a flow diagram illustrating an example decoding process;

FIGS. 15(A), 15(B), and 15(C) provide an illustrative diagram of an example video coding system and video coding process in operation;

FIG. 16 is an illustrative diagram of an example video coding system;

FIG. 17 is an illustrative diagram of an example system;

FIG. 18 illustrates an example device;

FIG. is an illustrative diagram of an example multi-level pattern for example partitions of a picture portion;

FIG. 20 is a flow diagram illustrating an example process;

FIG. 21 is a flow diagram illustrating an example process;

FIGS. 22(A)–22(F) illustrate a flow diagram of an example process;

FIGS. 23(A)–23(C) illustrate a flow diagram of an example process;

FIG. 24 illustrates an example system for entropy encoding partitions data;

FIG. 25 illustrates an example current picture portion and example neighboring picture portions, all arranged in accordance with at least some implementations of the present disclosure.

DETAILED DESCRIPTION

One or more embodiments or implementations are now described with reference to the enclosed figures. While specific configurations and arrangements are discussed, it should be understood that this is done for illustrative purposes only. Persons skilled in the relevant art will recognize that other configurations and arrangements may be employed

without departing from the spirit and scope of the description. It will be apparent to those skilled in the relevant art that techniques and/or arrangements described herein may also be employed in a variety of other systems and applications other than what is described herein.

While the following description sets forth various implementations that may be manifested in architectures such as system-on-a-chip (SoC) architectures for example, implementation of the techniques and/or arrangements described herein are not restricted to particular architectures and/or computing systems and may be implemented by any architecture and/or computing system for similar purposes. For instance, various architectures employing, for example, multiple integrated circuit (IC) chips and/or packages, and/or various computing devices and/or consumer electronic (CE) devices such as set top boxes, smart phones, etc., may implement the techniques and/or arrangements described herein. Further, while the following description may set forth numerous specific details such as logic implementations, types and interrelationships of system components, logic partitioning/integration choices, etc., claimed subject matter may be practiced without such specific details. In other instances, some material such as, for example, control structures and full software instruction sequences, may not be shown in detail in order not to obscure the material disclosed herein.

The material disclosed herein may be implemented in hardware, firmware, software, or any combination thereof. The material disclosed herein may also be implemented as instructions stored on a machine-readable medium, which may be read and executed by one or more processors. A machine-readable medium may include any medium and/or mechanism for storing or transmitting information in a form readable by a machine (e.g., a computing device). For example, a machine-readable medium may include read only memory (ROM); random access memory (RAM); magnetic disk storage media; optical storage media; flash memory devices; electrical, optical, acoustical or other forms of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.), and others.

References in the specification to "one implementation", "an implementation", "an example implementation", etc., indicate that the implementation described may include a particular feature, structure, or characteristic, but every embodiment may not necessarily include the particular feature, structure, or characteristic. Moreover, such phrases are not necessarily referring to the same implementation. Further, when a particular feature, structure, or characteristic is described in connection with an embodiment, it is submitted

that it is within the knowledge of one skilled in the art to effect such feature, structure, or characteristic in connection with other implementations whether or not explicitly described herein.

Systems, apparatus, articles, and methods are described below related to content adaptive entropy coding for video systems.

Next generation video (NGV) systems, apparatus, articles, and methods are described below. NGV video coding may incorporate significant content based adaptivity in the video coding process to achieve higher compression. As discussed above, the H.264/AVC standard may have a variety of limitations and ongoing attempts to improve on the standard, such as, for example, the HEVC standard may use iterative approaches to address such limitations. Herein, an NGV system including an encoder and a decoder will be described.

For example, the H.264 standard may support two modes of entropy coding. In the first mode, Adaptive VLC (Variable Length Coding), transform coefficients may be coded using Context Adaptive VLC (CAVLC) and all other syntax elements (e.g., overhead data, modes, motion vectors, and so on) may be coded using Exponential Golomb coding. The CAVLC coded data and the Exponential Golomb coded data may be multiplexed to generate an encoded bitstream. In the second mode, all data may be encoded using Context Adaptive Binary Arithmetic Coding (CABAC). The corresponding decoder may also operate in two modes, disassembling and decoding the multiplexed bit stream in the first mode and decoding the CABAC encoded bitstream in the second mode. Such techniques may have limitations. For example, CABAC coding may be efficient but may be complex such that throughput in higher resolution contexts may be limited. Further, by grouping data types together for coding, efficiency and complexity of the entropy coding may not be optimized.

In some video codec implementations, entropy coding and decoding of various data such as overhead data, modes, motion vectors, and/or transform coefficients may be a significant contributor to the coding efficiency and complexity of the system. In some examples, the techniques discussed herein may balance coding efficiency and system complexity.

In some examples, first video data and second video data may be received for entropy encoding at an entropy encoder module. The first video data and the second video

data may be different data types (e.g., header data, morphing parameters, synthesizing parameters, or global maps data or motion vectors or intra-prediction partition data or so on, as is discussed further herein). A first entropy encoding technique may be determined for the first video data based on a parameter associated with the first video data such as, for example, a number of compressed bits of the first video data, a predetermined indicator or flag associated with the first video data, a predetermined threshold, or a heuristically determined threshold or the like. In some examples, the first entropy encoding technique may be chosen from one of an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique. The first video data may be entropy encoded using the first entropy encoding technique and the second video data may be entropy encoded using the first entropy encoding technique. In some examples, the second entropy encoding technique may be determined for the second video data based on a parameter as discussed with respect to the first video data. In some examples, the second entropy encoding technique may be determined from options including an adaptive symbol-run variable length coding technique, an adaptive proxy variable length coding technique, an adaptive vector variable length coding, an adaptive 1-dimensional variable length coding technique, an adaptive 2-dimensional variable length coding technique, or other techniques as discussed herein. In some examples, the second entropy encoding technique may be predetermined for the data type of the second video data. Entropy encoding the first video data and the second video data may generate first compressed video data and second compressed video data. The first and second compressed video data may be assembled to generate an output bitstream.

The output bitstream may be transmitted from the encoder to a decoder, which may disassemble the bitstream to determine the first and second compressed video data. The compressed video data may be entropy decoded to generate entropy decoded video data, which may be further decoded to generate a video frame. The video frame may be transmitted to a display device for presentment to a user.

In some examples, additional video data types may be received for entropy encoding at the entropy encoder module. For example, third, fourth, fifth, sixth, seventh, or more video data may be entropy encoded to generate associated compressed video data, which may be assembled in the output bitstream, transmitted, and subsequently entropy decoded via a decoder. The various data types and encoding / decoding technique options and implementations are discussed further herein.

As used herein, the term "coder" may refer to an encoder and/or a decoder. Similarly, as used herein, the term "coding" may refer to performing video encoding via an encoder and/or performing video decoding via a decoder. For example a video encoder and video decoder may both be examples of coders capable of coding video data. In addition, as used herein, the term "codec" may refer to any process, program or set of operations, such as, for example, any combination of software, firmware, and/or hardware, that may implement an encoder and/or a decoder. Further, as used herein, the phrase "video data" may refer to any type of data associated with video coding such as, for example, video frames, image data, encoded bitstream data, or the like.

FIG. 1 is an illustrative diagram of an example next generation video encoder 100, arranged in accordance with at least some implementations of the present disclosure. As shown, encoder 100 may receive input video 101. Input video 101 may include any suitable input video for encoding such as, for example, input frames of a video sequence. As shown, input video 101 may be received via a content pre-analyzer module 102. Content pre-analyzer module 102 may be configured to perform analysis of the content of video frames of input video 101 to determine various types of parameters for improving video coding efficiency and speed performance. For example, content pre-analyzer module 102 may determine horizontal and vertical gradient information (e.g., Rs, Cs), variance, spatial complexity per picture, temporal complexity per picture, scene change detection, motion range estimation, gain detection, prediction distance estimation, number of objects estimation, region boundary detection, spatial complexity map computation, focus estimation, film grain estimation, or the like. The parameters generated by content pre-analyzer module 102 may be used by encoder 100 (e.g., via encode controller 103) and/or quantized and communicated to a decoder. As shown, video frames and/or other data may be transmitted from content pre-analyzer module 102 to adaptive picture organizer module 104, which may determine the picture type (e.g., I-, P-, or F/B-picture) of each video frame and reorder the video frames as needed. In some examples, adaptive picture organizer module 104 may include a frame portion generator configured to generate frame portions. In some examples, content pre-analyzer module 102 and adaptive picture organizer module 104 may together be considered a pre-analyzer subsystem of encoder 100.

As shown, video frames and/or other data may be transmitted from adaptive picture organizer module 104 to prediction partitions generator module 105. In some examples, prediction partitions generator module 105 may divide a frame or picture into tiles or

super-fragments or the like. In some examples, an additional module (e.g., between modules 104 and 105) may be provided for dividing a frame or picture into tiles or super-fragments. Prediction partitions generator module 105 may divide each tile or super-fragment into potential prediction partitionings or partitions. In some examples, the potential prediction partitionings may be determined using a partitioning technique such as, for example, a k-d tree partitioning technique, a bi-tree partitioning technique, or the like, which may be determined based on the picture type (e.g., I-, P-, or F/B-picture) of individual video frames, a characteristic of the frame portion being partitioned, or the like. In some examples, the determined potential prediction partitionings may be partitions for prediction (e.g., inter- or intra-prediction) and may be described as prediction partitions or prediction blocks or the like.

In some examples, a selected prediction partitioning (e.g., prediction partitions) may be determined from the potential prediction partitionings. For example, the selected prediction partitioning may be based on determining, for each potential prediction partitioning, predictions using characteristics and motion based multi-reference predictions or intra-predictions, and determining prediction parameters. For each potential prediction partitioning, a potential prediction error may be determined by differencing original pixels with prediction pixels and the selected prediction partitioning may be the potential prediction partitioning with the minimum prediction error. In other examples, the selected prediction partitioning may be determined based on a rate distortion optimization including a weighted scoring based on number of bits for coding the partitioning and a prediction error associated with the prediction partitioning.

As shown, the original pixels of the selected prediction partitioning (e.g., prediction partitions of a current frame) may be differenced with predicted partitions (e.g., a prediction of the prediction partition of the current frame based on a reference frame or frames and other predictive data such as inter- or intra-prediction data) at differencer 106. The determination of the predicted partitions will be described further below and may include a decode loop as shown in Fig. 1. Any residuals or residual data (e.g., partition prediction error data) from the differencing may be transmitted to coding partitions generator module 107. In some examples, such as for intra-prediction of prediction partitions in any picture type (I-, F/B- or P-pictures), coding partitions generator module 107 may be bypassed via switches 107a and 107b. In such examples, only a single level of partitioning may be performed. Such partitioning may be described as prediction partitioning (as discussed) or coding partitioning or both. In various examples, such partitioning may be performed via prediction partitions generator module 105 (as

discussed) or, as is discussed further herein, such partitioning may be performed via a k-d tree intra-prediction/coding partitioner module or a bi-tree intra-prediction/coding partitioner module implemented via coding partitions generator module 107.

In some examples, the partition prediction error data, if any, may not be significant enough to warrant encoding. In other examples, where it may be desirable to encode the partition prediction error data and the partition prediction error data is associated with inter-prediction or the like, coding partitions generator module 107 may determine coding partitions of the prediction partitions. In some examples, coding partitions generator module 107 may not be needed as the partition may be encoded without coding partitioning (e.g., as shown via the bypass path available via switches 107a and 107b). With or without coding partitioning, the partition prediction error data (which may subsequently be described as coding partitions in either event) may be transmitted to adaptive transform module 108 in the event the residuals or residual data require encoding. In some examples, prediction partitions generator module 105 and coding partitions generator module 107 may together be considered a partitioner subsystem of encoder 100. In various examples, coding partitions generator module 107 may operate on partition prediction error data, original pixel data, residual data, or wavelet data.

Coding partitions generator module 107 may generate potential coding partitionings (e.g., coding partitions) of, for example, partition prediction error data using bi-tree and/or k-d tree partitioning techniques or the like. In some examples, the potential coding partitions may be transformed using adaptive or fixed transforms with various block sizes via adaptive transform module 108 and a selected coding partitioning and selected transforms (e.g., adaptive or fixed) may be determined based on a rate distortion optimization or other basis. In some examples, the selected coding partitioning and/or the selected transform(s) may be determined based on a predetermined selection method based on coding partitions size or the like.

For example, adaptive transform module 108 may include a first portion or component for performing a parametric transform to allow locally optimal transform coding of small to medium size blocks and a second portion or component for performing globally stable, low overhead transform coding using a fixed transform, such as a discrete cosine transform (DCT) or a picture based transform from a variety of transforms, including parametric transforms, or any other configuration as is discussed further herein. In some examples, for locally optimal transform coding a Parametric Haar Transform (PHT) may be performed, as is discussed further herein. In some examples, transforms may be performed on 2D blocks of rectangular sizes between about 4x4 pixels and 64x64

pixels, with actual sizes depending on a number of factors such as whether the transformed data is luma or chroma, or inter or intra, or if the determined transform used is PHT or DCT or the like.

As shown, the resultant transform coefficients may be transmitted to adaptive quantize module 109. Adaptive quantize module 109 may quantize the resultant transform coefficients. Further, any data associated with a parametric transform, as needed, may be transmitted to either adaptive quantize module 109 (if quantization is desired) or adaptive entropy encoder module 110. Also as shown in FIG. 1, the quantized coefficients may be scanned and transmitted to adaptive entropy encoder module 110. Adaptive entropy encoder module 110 may entropy encode the quantized coefficients and include them in output bitstream 111. In some examples, adaptive transform module 108 and adaptive quantize module 109 may together be considered a transform encoder subsystem of encoder 100.

As also shown in FIG. 1, encoder 100 includes a local decode loop. The local decode loop may begin at adaptive inverse quantize module 112. Adaptive inverse quantize module 112 may be configured to perform the opposite operation(s) of adaptive quantize module 109 such that an inverse scan may be performed and quantized coefficients may be de-scaled to determine transform coefficients. Such an adaptive quantize operation may be lossy, for example. As shown, the transform coefficients may be transmitted to an adaptive inverse transform module 113. Adaptive inverse transform module 113 may perform the inverse transform as that performed by adaptive transform module 108, for example, to generate residuals or residual values or partition prediction error data (or original data or wavelet data, as discussed) associated with coding partitions. In some examples, adaptive inverse quantize module 112 and adaptive inverse transform module 113 may together be considered a transform decoder subsystem of encoder 100.

As shown, the partition prediction error data (or the like) may be transmitted to optional coding partitions assembler 114. Coding partitions assembler 114 may assemble coding partitions into decoded prediction partitions as needed (as shown, in some examples, coding partitions assembler 114 may be skipped via switches 114a and 114b such that decoded prediction partitions may have been generated at adaptive inverse transform module 113) to generate prediction partitions of prediction error data or decoded residual prediction partitions or the like.

As shown, the decoded residual prediction partitions may be added to predicted partitions (e.g., prediction pixel data) at adder 115 to generate reconstructed prediction partitions. The reconstructed prediction partitions may be transmitted to prediction

partitions assembler 116. Prediction partitions assembler 116 may assemble the reconstructed prediction partitions to generate reconstructed tiles or super-fragments. In some examples, coding partitions assembler module 114 and prediction partitions assembler module 116 may together be considered an un-partitioner subsystem of encoder 100.

The reconstructed tiles or super-fragments may be transmitted to blockiness analyzer and deblock filtering module 117. Blockiness analyzer and deblock filtering module 117 may deblock and dither the reconstructed tiles or super-fragments (or prediction partitions of tiles or super-fragments). The generated deblock and dither filter parameters may be used for the current filter operation and/or coded in output bitstream 111 for use by a decoder, for example. The output of blockiness analyzer and deblock filtering module 117 may be transmitted to a quality analyzer and quality restoration filtering module 118. Quality analyzer and quality restoration filtering module 118 may determine QR filtering parameters (e.g., for a QR decomposition) and use the determined parameters for filtering. The QR filtering parameters may also be coded in output bitstream 111 for use by a decoder. As shown, the output of quality analyzer and quality restoration filtering module 118 may be transmitted to decoded picture buffer 119. In some examples, the output of quality analyzer and quality restoration filtering module 118 may be a final reconstructed frame that may be used for prediction for coding other frames (e.g., the final reconstructed frame may be a reference frame or the like). In some examples, blockiness analyzer and deblock filtering module 117 and quality analyzer and quality restoration filtering module 118 may together be considered a filtering subsystem of encoder 100.

In encoder 100, prediction operations may include inter- and/or intra-prediction. As shown in FIG. 1(a), inter-prediction may be performed by one or more modules including morphing analyzer and morphed picture generation module 120, synthesizing analyzer and generation module 121, and characteristics and motion filtering predictor module 123. Morphing analyzer and morphed picture generation module 120 may analyze a current picture to determine parameters for changes in gain, changes in dominant motion, changes in registration, and changes in blur with respect to a reference frame or frames with which itmay beto be coded. The determined morphing parameters may be quantized/de-quantized and used (e.g., by morphing analyzer and morphed picture generation module 120) to generate morphed reference frames that that may be used by motion estimator module 122 for computing motion vectors for efficient motion (and characteristics) compensated prediction of a current frame. Synthesizing analyzer and generation module 121 may generate super resolution (SR) pictures and projected interpolation (PI) pictures or the like

for motion for determining motion vectors for efficient motion compensated prediction in these frames.

Motion estimator module 122 may generate motion vector data based on morphed reference frame(s) and/or super resolution (SR) pictures and projected interpolation (PI) pictures along with the current frame. In some examples, motion estimator module 122 may be considered an inter-prediction module. For example, the motion vector data may be used for inter-prediction. If inter-prediction is applied, characteristics and motion compensated filtering predictor module 123 may apply motion compensation as part of the local decode loop as discussed.

Intra-prediction may be performed by intra-directional prediction analyzer and prediction generation module 124. Intra-directional prediction analyzer and prediction generation module 124 may be configured to perform spatial directional prediction and may use decoded neighboring partitions. In some examples, both the determination of direction and generation of prediction may be performed by intra-directional prediction analyzer and prediction generation module 124. In some examples, intra-directional prediction analyzer and prediction generation module 124 may be considered an intra-prediction module.

As shown in FIG. 1, prediction modes and reference types analyzer module 125 may allow for selection of prediction modes from among, “skip”, “auto”, “inter”, “split”, “multi”, and “intra”, for each prediction partition of a tile (or super-fragment), all of which may apply to P- and F/B-pictures. In addition to prediction modes, it also allows for selection of reference types that can be different depending on “inter” or “multi” mode, as well as for P- and F/B-pictures. The prediction signal at the output of prediction modes and reference types analyzer module 125 may be filtered by prediction analyzer and prediction fusion filtering module 126. Prediction analyzer and prediction fusion filtering module 126 may determine parameters (e.g., filtering coefficients, frequency, overhead) to use for filtering and may perform the filtering. In some examples, filtering the prediction signal may fuse different types of signals representing different modes (e.g., intra, inter, multi, split, skip, and auto). In some examples, intra-prediction signals may be different than all other types of inter-prediction signal(s) such that proper filtering may greatly enhance coding efficiency. In some examples, the filtering parameters may be encoded in output bitstream 111 for use by a decoder. The filtered prediction signal may provide the second input (e.g., prediction partition(s)) to differencer 106, as discussed above, that may determine the prediction difference signal (e.g., partition prediction error) for coding discussed earlier. Further, the same filtered prediction signal may provide the second input

to adder 115, also as discussed above. As discussed, output bitstream 111 may provide an efficiently encoded bitstream for use by a decoder for the presentment of video.

FIG. 1 illustrates example control signals associated with operation of video encoder 100, where the following abbreviations may represent the associated information:

scnchg	Scene change information
spcpx	Spatial complexity information
tpcpx	Temporal complexity information
pdist	Temporal prediction distance information
pap	Pre Analysis parameters (placeholder for all other pre analysis parameters except scnchg, spcpx, tpcpx, pdist)
ptyp	Picture types information
pgst	Picture group structure information
pptn cand.	Prediction partitioning candidates
cptn cand.	Coding Partitioning Candidates
prp	Preprocessing
xmtyp	Transform type information
xmdir	Transform direction information
xmmod	Transform mode
ethp	One eighth (1/8th) pel motion prediction
pptn	Prediction Partitioning
cptn	Coding Partitioning
mot&cod cost	Motion and Coding Cost
qs	quantizer information set (includes Quantizer parameter (Qp), Quantizer matrix (QM) choice)
mv	Motion vectors
mop	Morphing Paramters
syp	Synthesizing Parameters
ddi	Deblock and dither information
qri	Quality Restoration filtering index/information
api	Adaptive Precision filtering index/information
fii	Fusion Filtering index/information
mod	Mode information
reftyp	Reference type information
idir	Intra Prediction Direction

The various signals and data items that may need to be sent to the decoder, ie, *pgst*, *ptyp*, *prp*, *pptn*, *cptn*, *modes*, *reftype*, *ethp*, *xmtyp*, *xmdir*, *xmmod*, *idir*, *mv*, *qs*, *mop*, *syp*, *ddi*, *qri*, *api*, *fii*, quant coefficients and others may then be entropy encoded by adaptive entropy encoder 110 that may include different entropy coders collectively referred to as an entropy encoder subsystem. While these control signals are illustrated as being associated with specific example functional modules of encoder 100 in Fig. 1, other implementations may include a different distribution of control signals among the functional modules of encoder 300. The present disclosure is not limited in this regard and, in various examples, implementation of the control signals herein may include the undertaking of only a subset of the specific example control signals shown, additional control signals, and/or in a different arrangement than illustrated.

FIG. 2 is an illustrative diagram of an example next generation video decoder 200, arranged in accordance with at least some implementations of the present disclosure. As shown, decoder 200 may receive an input bitstream 201. In some examples, input bitstream 201 may be encoded via encoder 100 and/or via the encoding techniques discussed herein. As shown, input bitstream 201 may be received by an adaptive entropy decoder module 202. Adaptive entropy decoder module 202 may decode the various types of encoded data (e.g., overhead, motion vectors, transform coefficients, etc.). In some examples, adaptive entropy decoder 202 may use a variable length decoding technique. In some examples, adaptive entropy decoder 202 may perform the inverse operation(s) of adaptive entropy encoder module 110 discussed above.

The decoded data may be transmitted to adaptive inverse quantize module 203. Adaptive inverse quantize module 203 may be configured to inverse scan and de-scale quantized coefficients to determine transform coefficients. Such an adaptive quantize operation may be lossy, for example. In some examples, adaptive inverse quantize module 203 may be configured to perform the opposite operation of adaptive quantize module 109 (e.g., substantially the same operations as adaptive inverse quantize module 112). As shown, the transform coefficients (and, in some examples, transform data for use in a parametric transform) may be transmitted to an adaptive inverse transform module 204. Adaptive inverse transform module 204 may perform an inverse transform on the transform coefficients to generate residuals or residual values or partition prediction error data (or original data or wavelet data) associated with coding partitions. In some examples, adaptive inverse transform module 204 may be configured to perform the opposite

operation of adaptive transform module 108 (e.g., substantially the same operations as adaptive inverse transform module 113). In some examples, adaptive inverse transform module 204 may perform an inverse transform based on other previously decoded data, such as, for example, decoded neighboring partitions. In some examples, adaptive inverse quantize module 203 and adaptive inverse transform module 204 may together be considered a transform decoder subsystem of decoder 200.

As shown, the residuals or residual values or partition prediction error data may be transmitted to coding partitions assembler 205. Coding partitions assembler 205 may assemble coding partitions into decoded prediction partitions as needed (as shown, in some examples, coding partitions assembler 205 may be skipped via switches 205a and 205b such that decoded prediction partitions may have been generated at adaptive inverse transform module 204). The decoded prediction partitions of prediction error data (e.g., prediction partition residuals) may be added to predicted partitions (e.g., prediction pixel data) at adder 206 to generate reconstructed prediction partitions. The reconstructed prediction partitions may be transmitted to prediction partitions assembler 207. Prediction partitions assembler 207 may assemble the reconstructed prediction partitions to generate reconstructed tiles or super-fragments. In some examples, coding partitions assembler module 205 and prediction partitions assembler module 207 may together be considered an un-partitioner subsystem of decoder 200.

The reconstructed tiles or super-fragments may be transmitted to deblock filtering module 208. Deblock filtering module 208 may deblock and dither the reconstructed tiles or super-fragments (or prediction partitions of tiles or super-fragments). The generated deblock and dither filter parameters may be determined from input bitstream 201, for example. The output of deblock filtering module 208 may be transmitted to a quality restoration filtering module 209. Quality restoration filtering module 209 may apply quality filtering based on QR parameters, which may be determined from input bitstream 201, for example. As shown in FIG. 2, the output of quality restoration filtering module 209 may be transmitted to decoded picture buffer 210. In some examples, the output of quality restoration filtering module 209 may be a final reconstructed frame that may be used for prediction for coding other frames (e.g., the final reconstructed frame may be a reference frame or the like). In some examples, deblock filtering module 208 and quality restoration filtering module 209 may together be considered a filtering subsystem of decoder 200.

As discussed, compensation due to prediction operations may include inter- and/or intra-prediction compensation. As shown, inter-prediction compensation may be

performed by one or more modules including morphing generation module 211, synthesizing generation module 212, and characteristics and motion compensated filtering predictor module 213. Morphing generation module 211 may use de-quantized morphing parameters (e.g., determined from input bitstream 201) to generate morphed reference frames. Synthesizing generation module 212 may generate super resolution (SR) pictures and projected interpolation (PI) pictures or the like based on parameters determined from input bitstream 201. If inter-prediction is applied, characteristics and motion compensated filtering predictor module 213 may apply motion compensation based on the received frames and motion vector data or the like in input bitstream 201.

Intra-prediction compensation may be performed by intra-directional prediction generation module 214. Intra-directional prediction generation module 214 may be configured to perform spatial directional prediction and may use decoded neighboring partitions according to intra-prediction data in input bitstream 201.

As shown in FIG. 2, prediction modes selector module 215 may determine a prediction mode selection from among, "skip", "auto", "inter", "multi", and "intra", for each prediction partition of a tile, all of which may apply to P- and F/B-pictures, based on mode selection data in input bitstream 201. In addition to prediction modes, it also allows for selection of reference types that can be different depending on "inter" or "multi" mode, as well as for P- and F/B-pictures. The prediction signal at the output of prediction modes selector module 215 may be filtered by prediction fusion filtering module 216. Prediction fusion filtering module 216 may perform filtering based on parameters (e.g., filtering coefficients, frequency, overhead) determined via input bitstream 201. In some examples, filtering the prediction signal may fuse different types of signals representing different modes (e.g., intra, inter, multi, skip, and auto). In some examples, intra-prediction signals may be different than all other types of inter-prediction signal(s) such that proper filtering may greatly enhance coding efficiency. The filtered prediction signal may provide the second input (e.g., prediction partition(s)) to differencer 206, as discussed above.

As discussed, the output of quality restoration filtering module 209 may be a final reconstructed frame. Final reconstructed frames may be transmitted to an adaptive picture re-organizer 217, which may re-order or re-organize frames as needed based on ordering parameters in input bitstream 201. Re-ordered frames may be transmitted to content post-restorer module 218. Content post-restorer module 218 may be an optional module configured to perform further improvement of perceptual quality of the decoded video. The improvement processing may be performed in response to quality improvement parameters

in input bitstream 201 or it may be performed as standalone operation. In some examples, content post-restorer module 218 may apply parameters to improve quality such as, for example, an estimation of film grain noise or residual blockiness reduction (e.g., even after the deblocking operations discussed with respect to deblock filtering module 208). As shown, decoder 200 may provide display video 219, which may be configured for display via a display device (not shown).

FIG. 2 illustrates example control signals associated with operation of video decoder 200, where the indicated abbreviations may represent similar information as discussed with respect to FIG. 1 above. While these control signals are illustrated as being associated with specific example functional modules of decoder 200 in Fig. 4, other implementations may include a different distribution of control signals among the functional modules of encoder 100. The present disclosure is not limited in this regard and, in various examples, implementation of the control signals herein may include the undertaking of only a subset of the specific example control signals shown, additional control signals, and/or in a different arrangement than illustrated.

While FIGS. 1 through 2 illustrate particular encoding and decoding modules, various other coding modules or components not depicted may also be utilized in accordance with the present disclosure. Further, the present disclosure is not limited to the particular components illustrated in FIGS. 1 and 2 and/or to the manner in which the various components are arranged. Various components of the systems described herein may be implemented in software, firmware, and/or hardware and/or any combination thereof. For example, various components of encoder 100 and/or decoder 200 may be provided, at least in part, by hardware of a computing System-on-a-Chip (SoC) such as may be found in a computing system such as, for example, a mobile phone.

Further, it may be recognized that encoder 100 may be associated with and/or provided by a content provider system including, for example, a video content server system, and that output bitstream 111 may be transmitted or conveyed to decoders such as, for example, decoder 200 by various communications components and/or systems such as transceivers, antennae, network systems, and the like not depicted in FIGS. 1 and 2. It may also be recognized that decoder 200 may be associated with a client system such as a computing device (e.g., a desktop computer, laptop computer, tablet computer, convertible laptop, mobile phone, or the like) that is remote to encoder 100 and that receives input bitstream 201 via various communications components and/or systems such as

transceivers, antennae, network systems, and the like not depicted in FIGS. 1 and 2. Therefore, in various implementations, encoder 100 and decoder subsystem 200 may be implemented either together or independent of one another.

FIG. 3(a) is an illustrative diagram of an example next generation video encoder 300a, arranged in accordance with at least some implementations of the present disclosure. FIG. 3(a) presents a similar encoder to that shown in FIGS. 1(a) and 1(b), and similar elements will not be repeated for the sake of brevity. As shown in FIG. 3(a), encoder 300a may include preanalyzer subsystem 310a, partitioner subsystem 320a, prediction encoding subsystem 330a, transform encoder subsystem 340a, filtering encoding subsystem 350a, entropy encoder system 360a, transform decoder subsystem 370a, and/or unpartioner subsystem 380a. Preanalyzer subsystem 310a may include content pre-analyzer module 102 and/or adaptive picture organizer module 104. Partitioner subsystem 320a may include prediction partitions generator module 105, and/or coding partitions generator 107. Prediction encoding subsystem 330a may include motion estimator module 122, characteristics and motion compensated filtering predictor module 123, and/or intra-directional prediction analyzer and prediction generation module 124. Transform encoder subsystem 340a may include adaptive transform module 108 and/or adaptive quantize module 109. Filtering encoding subsystem 350a may include blockiness analyzer and deblock filtering module 117, quality analyzer and quality restoration filtering module 118, motion estimator module 122, characteristics and motion compensated filtering predictor module 123, and/or prediction analyzer and prediction fusion filtering module 126. Entropy coding subsystem 360a may include adaptive entropy encoder module 110. Transform decoder subsystem 370a may include adaptive inverse quantize module 112 and/or adaptive inverse transform module 113. Unpartioner subsystem 380a may include coding partitions assembler 114 and/or prediction partitions assembler 116.

Partitioner subsystem 320a of encoder 300a may include two partitioning subsystems: prediction partitions generator module 105 that may perform analysis and partitioning for prediction, and coding partitions generator module 107 that may perform analysis and partitioning for coding. Another partitioning method may include adaptive picture organizer 104 which may segment pictures into regions or slices may also be optionally considered as being part of this partitioner.

Prediction encoder subsystem 330a of encoder 300a may include motion estimator 122 and characteristics and motion compensated filtering predictor 123 that may perform

analysis and prediction of "inter" signal, and intra-directional prediction analyzer and prediction generation module 124 that may perform analysis and prediction of "intra" signal. Motion estimator 122 and characteristics and motion compensated filtering predictor 123 may allow for increasing predictability by first compensating for other sources of differences (such as gain, global motion, registration), followed by actual motion compensation. They may also allow for use of data modeling to create synthesized frames (super resolution, and projection) that may allow better predictions, followed by use of actual motion compensation in such frames.

Transform encoder subsystem 340a of encoder 300a may perform analysis to select the type and size of transform and may include two major types of components. The first type of component may allow for using parametric transform to allow locally optimal transform coding of small to medium size blocks; such coding however may require some overhead. The second type of component may allow globally stable, low overhead coding using a generic/fixed transform such as the DCT, or a picture based transform from a choice of small number of transforms including parametric transforms. For locally adaptive transform coding, PHT (Parametric Haar Transform) may be used. Transforms may be performed on 2D blocks of rectangular sizes between 4x4 and 64x64, with actual sizes that may depend on a number of factors such as if the transformed data is luma or chroma, inter or intra, and if the transform used is PHT or DCT. The resulting transform coefficients may be quantized, scanned and entropy coded.

Entropy encoder subsystem 360a of encoder 300a may include a number of efficient but low complexity components each with the goal of efficiently coding a specific type of data (various types of overhead, motion vectors, or transform coefficients). Components of this subsystem may belong to a generic class of low complexity variable length coding techniques, however, for efficient coding, each component may be custom optimized for highest efficiency. For instance, a custom solution may be designed for coding of "Coded/Not Coded" data, another for "Modes and Ref Types" data, yet another for "Motion Vector" data, and yet another one for "Prediction and Coding Partitions" data. Finally, because a very large portion of data to be entropy coded is "transform coefficient" data, multiple approaches for efficient handling of specific block sizes, as well as an algorithm that may adapt between multiple tables may be used.

Filtering encoder subsystem 350a of encoder 300a may perform analysis of parameters as well as multiple filtering of the reconstructed pictures based on these

parameters, and may include several subsystems. For example, a first subsystem, blockiness analyzer and deblock filtering module 117 may deblock and dither to reduce or mask any potential block coding artifacts. A second example subsystem, quality analyzer and quality restoration filtering module 118, may perform general quality restoration to reduce the artifacts due to quantization operation in any video coding. A third example subsystem, which may include motion estimator 122 and characteristics and motion compensated filtering predictor module 123, may improve results from motion compensation by using a filter that adapts to the motion characteristics (motion speed/degree of blurriness) of the content. A fourth example subsystem, prediction fusion analyzer and filter generation module 126, may allow adaptive filtering of the prediction signal (which may reduce spurious artifacts in prediction, often from intra prediction) thereby reducing the prediction error which needs to be coded.

Encode controller module 103 of encoder 300a may be responsible for overall video quality under the constraints of given resources and desired encoding speed. For instance, in full RDO (Rate Distortion Optimization) based coding without using any shortcuts, the encoding speed for software encoding may be simply a consequence of computing resources (speed of processor, number of processors, hyperthreading, DDR3 memory etc.) availability. In such case, encode controller module 103 may be input every single combination of prediction partitions and coding partitions and by actual encoding, and the bitrate may be calculated along with reconstructed error for each case and, based on lagrangian optimization equations, the best set of prediction and coding partitions may be sent for each tile of each frame being coded. The full RDO based mode may result in best compression efficiency and may also be the slowest encoding mode. By using content analysis parameters from content preanalyzer module 102 and using them to make RDO simplification (not test all possible cases) or only pass a certain percentage of the blocks through full RDO, quality versus speed tradeoffs may be made allowing speedier encoding. Up to now we have described a variable bitrate (VBR) based encoder operation. Encode controller module 103 may also include a rate controller that can be invoked in case of constant bitrate (CBR) controlled coding.

Lastly, preanalyzer subsystem 310a of encoder 300a may perform analysis of content to compute various types of parameters useful for improving video coding efficiency and speed performance. For instance, it may compute horizontal and vertical gradient information (Rs, Cs), variance, spatial complexity per picture, temporal complexity per picture, scene change detection, motion range estimation, gain detection, prediction

distance estimation, number of objects estimation, region boundary detection, spatial complexity map computation, focus estimation, film grain estimation etc. The parameters generated by preanalyzer subsystem 310a may either be consumed by the encoder or be quantized and communicated to decoder 200.

While subsystems 310a through 380a are illustrated as being associated with specific example functional modules of encoder 300a in Fig. 3(a), other implementations of encoder 300a herein may include a different distribution of the functional modules of encoder 300a among subsystems 310a through 380a. The present disclosure is not limited in this regard and, in various examples, implementation of the example subsystems 310a through 380a herein may include the undertaking of only a subset of the specific example functional modules of encoder 300a shown, additional functional modules, and/or in a different arrangement than illustrated.

FIG. 3(b) is an illustrative diagram of an example next generation video decoder 300b, arranged in accordance with at least some implementations of the present disclosure. FIG. 3(b) presents a similar decoder to that shown in FIG. 2, and similar elements will not be repeated for the sake of brevity. As shown in FIG. 3(b), decoder 300b may include prediction decoder subsystem 330b, filtering decoder subsystem 350b, entropy decoder subsystem 360b, transform decoder subsystem 370b, unpartitioner_2 subsystem 380b, unpartitioner_1 subsystem 351b, filtering decoder subsystem 350b, and/or postrestorer subsystem 390b. Prediction decoder subsystem 330b may include characteristics and motion compensated filtering predictor module 213 and/or intra-directional prediction generation module 214. Filtering decoder subsystem 350b may include deblock filtering module 208, quality restoration filtering module 209, characteristics and motion compensated filtering predictor module 213, and/or prediction fusion filtering module 216. Entropy decoder subsystem 360b may include adaptive entropy decoder module 202. Transform decoder subsystem 370b may include adaptive inverse quantize module 203 and/or adaptive inverse transform module 204. Unpartitioner_2 subsystem 380b may include coding partitions assembler 205. Unpartitioner_1 subsystem 351b may include prediction partitions assembler 207. Postrestorer subsystem 790 may include content post restorer module 218 and/or adaptive picture re-organizer 217.

Entropy decoding subsystem 360b of decoder 300b may perform the inverse operation of the entropy encoder subsystem 360a of encoder 300a, i.e., it may decode various data (types of overhead, motion vectors, transform coefficients) encoded by

entropy encoder subsystem 360a using a class of techniques loosely referred to as variable length decoding. Specifically, various types of data to be decoded may include "Coded/Not Coded" data, "Modes and Ref Types" data, "Motion Vector" data, "Prediction and Coding Partitions" data, and "Transform Coefficient" data.

Transform decoder subsystem 370b of decoder 300b may perform inverse operation to that of transform encoder subsystem 340a of encoder 300a. Transform decoder subsystem 370b may include two types of components. The first type of example component may support use of the parametric inverse PHT transform of small to medium block sizes, while the other type of example component may support inverse DCT transform for all block sizes. The PHT transform used for a block may depend on analysis of decoded data of the neighboring blocks. Output bitstream 111 and/or input bitstream 201may carry information about partition/block sizes for PHT transform as well as in which direction of the 2D block to be inverse transformed the PHT may be used (the other direction uses DCT). For blocks coded purely by DCT, the partition/ block sizes information may be also retrieved from output bitstream 111 and/or input bitstream 201and used to apply inverse DCT of appropriate size.

Unpartitioner subsystem 380b of decoder 300b may perform inverse operation to that of partitioner subsystem 320a of encoder 300a and may include two unpartitioning subsystems, coding partitions assembler module 205 that may perform unpartitioning of coded data and prediction partitions assembler module 207 that may perform unpartitioning for prediction. Further if optional adaptive picture organizer module 104 is used at encoder 300a for region segmentation or slices, adaptive picture re-organizer module 217 may be needed at the decoder.

Prediction decoder subsystem 330b of decoder 300b may include characteristics and motion compensated filtering predictor module 213 that may perform prediction of "inter" signal and intra-directional prediction generation module 214 that may perform prediction of "intra" signal. Characteristics and motion compensated filtering predictor module 213 may allow for increasing predictability by first compensating for other sources of differences (such as gain, global motion, registration) or creation of synthesized frames (super resolution, and projection), followed by actual motion compensation.

Filtering decoder subsystem 350b of decoder 300b may perform multiple filtering of the reconstructed pictures based on parameters sent by encoder 300a and may include several subsystems. The first example subsystem, deblock filtering modue 208, may

deblock and dither to reduce or mask any potential block coding artifacts. The second example subsystem, quality restoration filtering module 209, may perform general quality restoration to reduce the artifacts due to quantization operation in any video coding. The third example subsystem, characteristics and motion compensated filtering predictor module 213, may improve results from motion compensation by using a filter that may adapt to the motion characteristics (motion speed/degree of blurriness) of the content. The fourth example subsystem, prediction fusion filtering module 216, may allow adaptive filtering of the prediction signal (which may reduce spurious artifacts in prediction, often from intra prediction) thereby reducing the prediction error which may need to be coded.

Postrestorer subsystem 390b of decoder 300b is an optional block that may perform further improvement of perceptual quality of decoded video. This processing can be done either in response to quality improvement parameters sent by encoder 100, or it can be standalone decision made at the postrestorer subsystem 390b. In terms of specific parameters computed at encoder 100 that can be used to improve quality at postrestorer subsystem 390b may be estimation of film grain noise and residual blockiness at encoder 100 (even after deblocking). As regards the film grain noise, if parameters can be computed and sent via output bitstream 111 and/or input bitstream 201to decoder 200, then these parameters may be used to synthesize the film grain noise. Likewise, for any residual blocking artifacts at encoder 100, if they can be measured and parameters sent via output bitstream 111 and/or bitstream 201, postrestorer subsystem 390b may decode these parameters and may use them to optionally perform additional deblocking prior to display. In addition, encoder 100 also may have access to scene change, spatial complexity, temporal complexity, motion range, and prediction distance information that may help in quality restoration in postrestorer subsystem 390b.

While subsystems 330b through 390b are illustrated as being associated with specific example functional modules of decoder 300b in Fig. 3(b), other implementations of decoder 300b herein may include a different distribution of the functional modules of decoder 300b among subsystems 330b through 390b. The present disclosure is not limited in this regard and, in various examples, implementation of the example subsystems 330b through 390b herein may include the undertaking of only a subset of the specific example functional modules of decoder 300b shown, additional functional modules, and/or in a different arrangement than illustrated.

FIG. 4 is an illustrative diagram of an example entropy encoder module 110, arranged in accordance with at least some implementations of the present disclosure. As shown, entropy encoder module 110 may include bitstream headers, parameters and maps data encoder module 401, picture partitions, prediction partitions, and coding partitions encoder module 402, coding modes and reference types encoder module 403, coded/not-coded data encoder module 404, transform coefficients encoder module 405, motion vector prediction and differential motion vector encoder module 406, intra-prediction type and direction data encoder module 407, and/or bitstream assembler module 408. In the discussion herein, each of modules 401–407 may be shortened to encoder module 401, encoder module 404, or the like for the sake of brevity.

As shown, encoder modules 401–407 may receive video data 411–417, respectively, via adaptive entropy encoder 110. In some examples, received video data 411–417 may be received from various modules of encoder 100 as discussed herein. As shown, encoder modules 401–407 may compress the received video data 411–417 to generated compressed video data 421–427. Compressed video data 421–427 may be provided to bitstream assembler 408, which may assemble compressed video data 421–427 to generate output bitstream 111.

In some examples, encoder modules 401–407 may each include one or more specialized component(s) for efficiently encoding the type of data associated with received video data 411–417. In some examples, one or more of encoder modules 401–407 may preprocess the received video data 411–417 and/or select an entropy coding technique based on a parameter, parameters, or characteristics of the received video data 411–417 or other system information.

For example, encoder module 401 may receive overhead data 411, which may include bitstream header data (e.g., sequence and/or picture level bitstream headers), morphing parameters, synthesizing parameters, or global maps data (e.g., quantizer maps of pictures indicating quantizers to be used on a partition basis). As is discussed further below with respect to FIG. 6, in some examples, encoder module 401 may implement an adaptive symbol-run variable length coding technique, an adaptive proxy variable length coding technique, or a variable length coding table or tables compression of video data 411. In some examples, encoder module 411 may determine which technique provides the greatest compression efficiency (e.g., the fewest bits for compressed video data 421) such that the parameter(s) associated with video data 411 may be the number of bits needed for

each coding technique or the like. Encoder module 411 may entropy encode video data 411 to generate compressed video data 421 (e.g., compressed overhead data), which may be transmitted to bitstream assembler 408 as shown.

As discussed, in some examples, the parameter associated with the video data (e.g., any of video data 411–417) may be a fewest number of attainable bits, most efficient encoding technique, or the like. In other examples, the parameter associated with the video data may be a predefined or predetermine parameter such that encoding technique is predetermined. In some examples, the parameter associated with the video data may be based on a characteristic of the video data such that the determined encoding technique may be adaptive to the received video data as is discussed further herein.

As shown, in some examples, encoder module 402 may receive partition data 412, which may include picture slices or regions data, intra-prediction partition data, and/or inter-prediction partition and coding partition data. As is discussed further below with respect to FIG. 6, in some examples, encoder module 412 may implement an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique for the compression of the intra-prediction partition data and/or inter-prediction partition data portions of video data 412 based on a parameter associated with the intra-prediction partition data and/or inter-prediction partition data (e.g., a fewest number of attainable bits, most efficient encoding technique, a predetermined parameter, characteristics of video data 412, or the like), and encoder module 412 may implement adaptive codebook variable length coding for the slices or regions data portions of video data 412 to generate compressed video data 422 (e.g., compressed partition data), which may be transmitted to bitstream assembler 408 as shown. In some examples, the intra-prediction partition data and/or inter-prediction partition data may include data indicating the partitioning of tiles into partitions, partitions into sub-partitions, or the like. In some examples, the partitions and/or sub-partitions may include prediction partitions and/or sub-partitions. In some examples, partitions and/or sub-partitions may include coding partitions and/or sub-partitions.

Further as shown, in some examples, encoder module 403 may receive modes and reference types data 413, which may include modes (e.g., intra, split, skip, auto, inter, or multi) data and/or references data for each prediction partition. For example, the mode split information may indicate whether a partition is further divided or not. If a partition is further divided, the mode data may further include direction information indicating

whether the split is a horizontal split (e.g., hor) or a vertical split (e.g., vert). As is discussed further below with respect to FIG. 6, in some examples, encoder module 403 may implement an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique for separate coding of split/non-split partition information data, separate coding of split/non-split split information data, or prediction reference information data based on a parameter associated with the data (e.g., a fewest number of attainable bits, most efficient encoding technique, a predetermined parameter, characteristics of the pertinent portions of video data 413, or the like), and encoder module 403 may implement adaptive variable length coding for joint coding of modes and split information to generate compressed video data 423(e.g., compressed modes and reference types data), which may be transmitted to bitstream assembler 408 as shown.

Further, in some examples, encoder module 404 may receive coded / not-coded data 414, which may include coded/not-coded data as discussed herein. For example, a partition (or sub-partition) may be coded if it has any nonzero transform coefficients and a partition (or sub-partition) may be not-coded if it has all zero transform coefficients. In some examples, coded/not-coded data may not be needed for partitions having an intra or skip mode. In some examples, coded/not-coded data may be needed for partitions having an auto, inter, or multi mode. As is discussed further below with respect to FIG. 6, in some examples, encoder module 404 may implement an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique for coded/not-coded data based on a parameter associated with the coded/not-coded data (e.g., a fewest number of attainable bits, most efficient encoding technique, a predetermined parameter, characteristics of video data 414, or the like) to generate compressed video data 424 (e.g., compressed coded / not-coded data), which may be transmitted to bitstream assembler 408 as shown.

In some examples, encoder module 405 may receive transform data 415, which may include transform coefficient data. For example, for blocks or partitions or sub-partitions that are coded (e.g., have one or more nonzero transform coefficients), transform coefficient data may be received for entropy encoding. As is discussed further with respect to FIG. 6, encoder module 405 may implement adaptive vector variable length coding for blocks or partitions or sub-partitions having a block or partition or sub-partition size of 2 in one dimension (e.g., 2xK partitions or Kx2 sized partitions). Further, encoder module 405 may implement adaptive 1-dimensional variable length coding for blocks or partitions or sub-partitions of size 4x4 and adaptive 2-dimensional variable length coding for all other

block or partition or sub-partition sizes (e.g., 4x8, 8x4, 8x8, 16x16, 32x32, 64x64, and so on). The generated compressed video data 425 (e.g., compressed transform data) may be transmitted to bitstream assembler 408 as shown.

For example, the transform coefficient data may result from a forward transform of rectangular (or square or the like) partitions of pixel data or rectangular (or square or the like) of pixel difference values implemented via adaptive transform module 108 followed by a quantization of the resulting coefficients via adaptive quantize module 109. In some examples, the transform coefficient data may be scanned to convert it to a 1-dimensional frequency ordered partition via encoder module 405. Such conversion may be highly adaptive any partition size (e.g., 24 or more or 32 or more partition sizes), different data types (e.g., discrete cosine transform coefficients or hybrid parametric Haar transform coefficients or the like of either intra or inter partitions), and/or different quantizer set characteristics (e.g., various combinations of quantizer parameters and/or matrices). Further, a block or partition or sub-partition may belong to different picture types: I-picture (e.g., intra compensation only), P-picture (e.g., predictive) or F-picture (e.g., functional) and/or may represent different types of signal or data (e.g., luma or chroma or the like), which may be quantized with different quantizer setting.

Further, in some examples, encoder module 406 may receive motion data 416, which may include motion vector data. As is discussed further with respect to FIG. 6, motion vector prediction may be performed based on video data 416 to generate one or more predicted motion vectors. A predicted motion vector may be differenced with an original motion data of video data 416 to generate a difference motion vector. In some examples, encoder module 416 may implement an adaptive classified variable length coding for the difference motion vector(s) to generate compressed video data 426 (e.g., compressed motion data), which may be transmitted to bitstream assembler 408 as shown.

Further, in some examples, encoder module 407 may receive intra-prediction data 417, which may include intra-prediction type or intra-prediction direction data. For example, as discussed, intra coding may use prediction, which may use neighboring past decoded partition(s) within the same frame to generate spatial prediction. In such examples, there may be predictors for indicating a past decoded partition or partitions. For example, the predictors may include dc, slope, directional, BTPC, feature matching, or the like. Further, in some examples, the directional predictor may be adaptive for different partition sizes. For example, specifying a directional predictor may include providing an

technique for determining angular prediction pixel partition(s) for coding using causal neighboring decoded partitions and/or specifying a technique for entropy coding spatial prediction directions. In some examples, such techniques may be performed via encoder module 407. As is discussed further below with respect to FIG. 6, in some examples, encoder module 407 may implement an adaptive variable length coding technique or an arithmetic coding technique for intra-prediction type or intra-prediction direction data based on a parameter associated with the intra-prediction type or intra-prediction direction data (e.g., a fewest number of attainable bits, most efficient encoding technique, a predetermined parameter, characteristics of video data 417, or the like) to generate compressed video data 427 (e.g., compressed intra-prediction data), which may be transmitted to bitstream assembler 408 as shown.

As shown in FIG. 4, adaptive entropy encoder 110 may include bitstream assembler 408. In some examples, some or all of encoder modules 401–407 may provide entropy coded compressed video data 421–427 at different instances in time. Further, in some examples, one or some of compressed video data 421–427 may be picture based, region or slice based, tile based, prediction partition based, coding partition based, or any combination thereof. In some examples, bitstream assembler may multiplex (the potentially different) compressed video data 421–427 to create a valid bitstream such as, for example, output bitstream 111. For example, the valid bitstream may be a valid next generation video (NGV) coded bitstream, which may following a NGV bitstream syntax specification. In some examples, output bitstream 111 may be a video only bitstream. In some examples, output bitstream 111 may be multiplexed (e.g., Transport or a Media File Format) with uncoded or coded audio to create a multiplexed audio-visual stream. In any event, the bitstream may be used local decode, storage, or transmission to a decoder as discussed herein.

FIG. 5 is an illustrative diagram of an example entropy decoder module 202, arranged in accordance with at least some implementations of the present disclosure. As shown, entropy decoder module 202 may include bitstream headers, parameters and maps data decoder module 501, picture partitions, prediction partitions, and coding partitions decoder module 502, coding modes and reference types decoder module 503, coded/not-coded data decoder module 504, transform coefficients decoder module 505, motion vector and differential motion vector decoder module 506, intra-prediction and direction data decoder module 507, and/or bitstream disassembler module 508. In the discussion herein,

each of modules 501–507 may be shortened to decoder module 501, decoder module 505, or the like for the sake of brevity.

As shown, bitstream disassembler module 508 may receive input bitstream 201. In some examples, input bitstream 201 may be a valid bitstream such as, for example, a valid next generation video (NGV) coded bitstream, which may follow a NGV bitstream syntax specification. In some examples, input bitstream 201 may be a video only bitstream. In some examples, input bitstream 201 may be a multiplexed audio-visual stream as discussed herein. Bitstream disassembler module 508 may disassemble input bitstream 201 to determine compressed video data 511–517 as shown. For example, bitstream disassembler module 508 may use a predefined syntax or specification to divide input bitstream 201 into component compressed video data 511–517 by data type for decompression via decoder modules 501–507. In some examples, bitstream disassembler module 508 may perform an inverse operation with respect to bitstream assembler module 508.

As shown in FIG. 5, decoder modules 501–507 may receive compressed video data 511–517, respectively, and generate video data 521–527. Video data 521–527 may be transmitted to various components of decoder 200 for further decoding as discussed herein. Decoder 200 may thereby generate video frame(s) for presentment to a user via a display device (not shown). In some examples, decoder modules 501–507 may each perform an inverse operation with respect to encoder modules 401–407. In some examples, decoder modules 501–507 may each include one or more specialized component(s) for efficiently entropy decoding the type of data associated with compressed video data 511–517.

For example, decoder module 501 may receive compressed overhead data 511, which may include compressed bitstream header data (e.g., sequence and/or picture level bitstream headers), morphing parameters, synthesizing parameters, or global maps data (e.g., quantizer maps of pictures indicating quantizers to be used on a partition basis). In some examples, decoder module 511 may implement an adaptive symbol-run variable length coding technique, an adaptive proxy variable length coding technique, or a variable length coding table or tables for decompression of compressed overhead data 511 to generate overhead data 521. In some examples, decoder module 501 may determine which coding technique to implement based on a parameter or indicator provided via bitstream 201.

As shown, in some examples, decoder module 502 may receive compressed partition data 512, which may include compressed picture slices or regions data, intra-prediction

partition data, and/or inter-prediction partition data. In some examples, decoder module 512 may implement an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique for the decompression of the intra-prediction partition data and/or inter-prediction partition data portions of compressed partition data 512, and decoder module 512 may implement an adaptive codebook variable length coding for the decompression of the slices or regions data portions of compressed partition data 512 to generate partition data 522. In some examples, the intra-prediction partition data and/or inter-prediction partition data may include data indicating the partitioning of tiles into partitions, partitions into sub-partitions, or the like. In some examples, the partitions and/or sub-partitions may include prediction partitions and/or sub-partitions. In some examples, partitions and/or sub-partitions may include coding partitions and/or sub-partitions. In some examples, decoder module 502 may determine which coding technique to implement for the decompression of the intra-prediction partition data and/or inter-prediction partition data portions of compressed video data 512 based on a parameter or indicator provided via bitstream 201.

Further, in some examples, decoder module 503 may receive compressed modes and reference types data 513, which may include compressed modes (e.g., intra, split, skip, auto, inter, or multi) data and/or references data for each prediction partition. For example, the mode split information may indicate whether a partition is further divided or not. If a partition is further divided, the mode data may further include direction information indicating whether the split is a horizontal split (e.g., hor) or a vertical split (e.g., vert). In some examples, decoder module 503 may implement an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique for decompression of separate coding of split/non-split partition information data, separate coding of split/non-split split information data, or prediction reference information data, and decoder module 503 may implement an adaptive variable length coding for decompression of joint coding of modes and split information to generate modes and reference types data 523. In some examples, decoder module 503 may determine which coding technique to implement for decompression of separate coding of split/non-split partition information data, separate coding of split/non-split split information data, or prediction reference information data based on a parameter or indicator provided via bitstream 201.

Further, in some examples, decoder module 504 may receive compressed coded / not-coded data 514, which may include coded/not-coded data as discussed herein. For

example, a partition (or sub-partition) may be coded if it has any nonzero transform coefficients and a partition (or sub-partition) may be not-coded if it has all zero transform coefficients. In some examples, coded/not-coded data may not be needed for partitions having an intra or skip mode. In some examples, coded/not-coded data may be needed for partitions having an auto, inter, or multi mode. In some examples, decoder module 504 may implement an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique for decompression of coded/not-coded data to generate coded/not-coded data 524. In some examples, decoder module 504 may determine which coding technique to implement for decompression based on a parameter or indicator provided via bitstream 201.

As shown, in some examples, decoder module 505 may receive compressed transform data 515, which may include transform coefficient data. For example, for blocks or partitions or sub-partitions that are coded (e.g., have one or more nonzero transform coefficients) compressed transform data 515 may include transform coefficient data. In some examples, decoder module 505 may implement an adaptive vector variable length coding for decompression of blocks or partitions or sub-partitions having a block or partition or sub-partition size of 2 in one dimension (e.g., 2xK partitions or Kx2 sized partitions. Further, decoder module 505 may implement an adaptive 1-dimensional variable length coding for decompression of blocks or partitions or sub-partitions of size 4x4 and adaptive 2-dimensional variable length coding for decompression of all other block or partition or sub-partition sizes (e.g., 4x8, 8x4, 8x8, 16x16, 32x32, 64x64, and so on). The generated transform data 525 may be transmitted to other module(s) of decoder 200 as shown.

Further, in some examples, decoder module 506 may receive compressed motion data 516, which may include motion vector data. As is discussed further below with respect to FIG. 6, in some examples, compressed motion data 516 may be decompressed using an adaptive classified variable length coding technique to generate predicted difference motion vectors. The predicted difference motion vectors may be added to motion vector prediction to generate reconstructed motion vectors. The motion vector prediction may be generated based on previously decoded motion vectors of neighboring blocks or partitions using the inverse of the technique implemented via encoder module 506, for example, and/or motion vectors. The reconstructed motion vectors may be transmitted to other module(s) of decoder 200 via motion data 526 as shown.

Further, in some examples, decoder module 507 may receive compressed intra-prediction data 517, which may include intra-prediction type or intra-prediction direction data. For example, as discussed, intra coding may use prediction, which may use neighboring past decoded partition(s) within the same frame to generate spatial prediction. In such examples, there may be predictors for indicating a past decoded partition or partitions. For example, the predictors may include dc, slope, directional, BTPC, feature matching, or the like. Further, in some examples, the directional predictor may be adaptive for different partition sizes. For example, specifying a directional predictor may include providing an technique for determining angular prediction pixel partition(s) for coding using causal neighboring decoded partitions and/or specifying a technique for entropy coding spatial prediction directions. In some examples, decoder module 517 may implement an adaptive variable length coding technique or an arithmetic coding technique for decompression of intra-prediction type or intra-prediction direction data to generate intra-prediction data 527. In some examples, decoder module 507 may determine which coding technique to implement for decompression based on a parameter or indicator provided via bitstream 201.

As discussed above, a variety of entropy coding techniques may be implemented on various data types for lossless compression of video data to generate compressed video data at an entropy encoder and for decompression of the compressed video data to generate duplicate video data at an entropy decoder.

In some examples, an adaptive symbol-run variable length coding technique may be implemented. For example, encoder and decoder modules 401, 501, 402, 502, 403, 503, and/or 404, 504 may implement an adaptive symbol-run variable length coding technique on some or all of the video data or compressed video data received.

In some examples, an adaptive symbol-run variable length coding technique may include coding of relative difference in addresses between non-skipped blocks within a frame in video coding that allows one to determine the number of consecutive skipped blocks. For example, in the context of coded/not-coded data as encoded and decoded via encoder and decoder modules 504, 504, instead of sending one bit (e.g., bit-map) for each block to signal coded/not-coded (e.g., skipped) blocks, encoder module 504, for example, may encode a run of skipped blocks. In such implementations, the longer the run of skipped blocks, the more efficiently the data may be compressed.

Further, several types of adaptivity may be added to adaptive symbol-run variable length coding technique as described herein: adaptivity that may allow for use of multiple tables, adaptivity that may allow for use of either performing this type of coding on original bit map data, inverted bitmap, differential bitmap, or gradient predictive bitmap, or the like. For example, the adaptive symbol-run variable length coding technique may include converting the first video data from bit map data to at least one of an inverted bitmap, a differential bit map, or a gradient predictive bit map before applying adaptive symbol-run variable length coding. For example, adaptive symbol-run variable length coding technique may be used to entropy encode substantially any type of event (e.g., symbol/run combination). Further, symbol/run events may be used to code multi-level (e.g., 0, 1, 2, 3, etc.) or binary (e.g., 0,1) events. In examples where multi-level events are encoded, adaptive symbol-run variable length coding technique may be applied a number of consecutive times, breaking a multi-level map into a number of binary sub-maps, with each previous sub-map, excluded from next level's sub-map, or the like.

In some examples, an adaptive proxy variable length coding technique may be implemented. For example, encoder and decoder modules 401, 501, 402, 502, 403, 503, and/or 404, 504 may implement an adaptive proxy variable length coding technique on some or all of the video data or compressed video data received.

In some examples, an adaptive proxy variable length coding technique may include substitution of original fixed length 1D blocks (e.g., groups) of bits with variable length codes (e.g., patterns of sequence of bits) such that after the replacement the resulting bitstream may be smaller in size. In some examples, at the decoder, the process may be repeated (or inversed) resulting in original intended bitstream. In some examples, the original blocks of bits replaced may be of fixed small sizes (e.g., groups of 2 bits, groups of 3 bits, or groups of 4 bits, or the like). In some examples, the replacement codes may be of small size and variable length in nature. In some examples, the adaptive proxy variable length coding discussed herein may be characterized as Short VLCs (e.g., variable length codes). Further, the adaptive proxy variable length coding technique described herein may be adaptive to content by providing multiple replacement options. In some examples, 1-dimensional blocks/groups of 2 bits may be replaced with 1–3 bit long codes. In some examples, 1-dimensional blocks/groups (or collections of blocks/groups) of 3 bits with codes may be replaced with 1–5 bit long codes. In some examples, the adaptive proxy variable length coding technique may exploit statistical redundancy within a bitstream. In some examples, the adaptive proxy variable length coding technique may provide a

compression gain of about 1–1.3. In some examples, adaptive proxy variable length coding technique may offer the advantage of being amenable to application to short sequence of bits.

In some examples, an adaptive block-of-symbols variable length coding technique may be implemented. For example, encoder and decoder modules 405, 505 may implement an adaptive block-of-symbols variable length coding technique on some or all of the video data or compressed video data received.

In some examples, an adaptive block-of-symbols variable length coding technique may include two sub-coding techniques, as will be discussed further with respect to FIG. 7. For example, the adaptive block-of-symbols variable length coding technique may include an adaptive vector variable length coding technique and an adaptive 1D/2D (1-dimensional/2-dimensional) variable length coding technique. In some examples, the adaptive block-of-symbols variable length coding technique may be used to encode blocks of closely related symbols such as blocks of transform coefficients as discussed herein.

In some examples, the adaptive vector variable length coding technique of the adaptive block-of-symbols variable length coding technique may encode relatively small 2D blocks or partitions of symbols by use of a joint single codeword such that coding a block of symbols may result in a VLC (variable length coding) codebook. In some examples, the larger the size of the block or partition, the larger the size of the codebook. In some examples, the adaptive vector variable length coding technique may be applied to block or partition sizes having a size 2 in one dimension (e.g., 2xK or Kx2 blocks or partitions). By applying the adaptive vector variable length coding technique to blocks or partitions of such sizes, the size of the VLC codebook may be advantageously limited.

In some examples, the adaptive 1D variable length coding technique of the adaptive block-of-symbols variable length coding technique may be used for coding 4x4 transform coefficient block or partition sizes. is essentially same as the CAVLC coder. This coder is primarily used for coding 4x4. In some examples, the adaptive 1D variable length coding technique may be implemented via a content adaptive variable length coding technique with a number of different VLC Tables used based on the context of the coefficient(s) being coded. For example, based on the context of the coefficient(s) being coded encoder and/or decoder modules 505, 505 may switch VLC Tables.

In some examples, the adaptive 2D variable length coding technique of the adaptive block-of-symbols variable length coding technique may utilize two dimensional properties of a block of symbols to switch based on context between a number of different VCL Tables. In some examples, the adaptive 2D variable length coding technique may be characterized as a CA2DVLC (Content Adaptive 2D Variable Length) coder. In some examples, In some examples, the adaptive 2D variable length coding technique may be used to encode all remaining transform coefficient block or petition sizes besides 2xK, Kx2 blocks and 4x4 blocks (e.g., 4x8, 8x4, 8x8, 16x16, 32x32, 64x64, and so on).

FIG. 6 is an illustrative diagram of an example entropy encoder module 110, arranged in accordance with at least some implementations of the present disclosure. As shown and as discussed above with respect to FIG. 4, entropy encoder module 110 may include bitstream headers, parameters and maps data encoder module 401, picture partitions, prediction partitions, and coding partitions encoder module 402, coding modes and reference types encoder module 403, coded/not-coded data encoder module 404, transform coefficients encoder module 405, motion vector and differential motion vector encoder module 406, intra-prediction and direction data encoder module 407, and/or bitstream assembler module 408.

As shown, encoder module 401 may include adaptive VLC, symbol-run VLC, and/or proxy VLC encoder for bitstream headers, parameters, and maps data module 611 and may receive video data 411. Video data 411 may have a data type such that video data 411 may include bitstream header data (e.g., sequence and/or picture level bitstream headers), morphing parameters, synthesizing parameters, and/or global maps data (e.g., quantizer maps of pictures indicating quantizers to be used on a partition basis). In some examples, an entropy encoding technique may be determined for video data 411 based on a parameter, parameters or characteristics of video data 411 or other system parameters. In some examples, the entropy encoding technique for video data 411 may be one of an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique, as described above, or a variable length coding table or tables compression technique. The determined entropy encoding technique may be applied to video data 411 to generate compressed video data 421. A variable length coding table or tables compression technique may include a content adaptive variable length coding technique with one or more tables available for coding based on video data 411, for example. In some examples, encoder module 404 may determine which technique provides

the greatest compression efficiency (e.g., the fewest bits for compressed video data 421) such that the parameter(s) associated with video data 411 may be the number of bits needed for each coding technique or the like. In some examples, the parameter associated with video data 411 may be based on a characteristic of the video data such that the determined encoding technique may be adaptive to the received video data.

As shown in FIG. 6, encoder module 402 may include adaptive codebook VLC encoder for picture partitions module 621, symbol-run VLC and/or proxy VLC encoder for intra-prediction partitions module 622, and/or symbol-run VLC and/or proxy VLC encoder for inter-prediction partitions and coding partitions module 623. Also as shown, encoder module 402 may receive video data 412. In some examples, video data 412 may include picture slices or regions data 624, intra-prediction partition data 625, and/or inter-prediction and coding partition data 626.

As shown, picture slices or regions data 624 may be received via adaptive codebook VLC encoder for picture partitions module 621, which may apply adaptive codebook variable length coding to picture slices or regions data 624 to generate compressed picture slices or regions data 627. In some examples, picture slices or regions data 624 may include region boundaries for pictures, slices, regions, or the like. In some examples, adaptive codebook variable length coding may include content adaptive variable length coding using an codebook adaptive to the content of picture slices or regions data 624 or other system parameters or the like.

As shown, intra-prediction partition data 625 may be received via symbol-run VLC and/or proxy VLC encoder for intra-prediction partitions module 622. In some examples, an entropy encoding technique may be determined for intra-prediction partition data 625 based on a parameter, parameters or characteristics of intra -prediction partition data 625 or other system parameters (e.g., compression efficiency, a characteristic of the data, and so on), as discussed herein. The determined entropy encoding technique may be applied to intra-prediction partition data 625 to generate compressed intra-prediction partition data 628. As shown, in some examples, the entropy encoding technique for intra-prediction partition data 625 may be one of an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique, as described above. In some examples, intra-prediction partition data 625 may include partitions based on bi-tree partitioning or k-d tree partitioning, or the like.

As shown, inter-prediction and coding partition data 626 may be received via symbol-run VLC and/or proxy VLC encoder for inter-prediction partitions and coding partitions module 623. In some examples, an entropy encoding technique may be determined for or inter-prediction and coding partition data 626 based on a parameter, parameters or characteristics of or inter-prediction and coding partition data 626 or other system parameters (e.g., compression efficiency, a characteristic of the data, and so on), as discussed herein. The determined entropy encoding technique may be applied to or inter-prediction and coding partition data 626 to generate compressed or inter-prediction and coding partition data 629. As shown, in some examples, the entropy encoding technique for or inter-prediction and coding partition data 626 may be one of an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique, as described above. In some examples, or inter-prediction and coding partition data 426 may include inter-partitions and coding tree partitions, or the like.

As shown in FIG. 6, encoder module 403 may include adaptive VLC encoder for joint modes and splits module 631, symbol-run VLC and/or proxy VLC encoder for modes module 632, symbol-run VLC and/or proxy VLC encoder for splits module 633, and/or symbol-run VLC and/or proxy VLC encoder for reference types module 634. Also as shown, encoder module 403 may receive video data 412. In some examples, video data 412 may joint coding of modes and splits data 635, modes information data 636, split/not-split information data 637, and/or prediction reference information data 638.

As shown, joint coding of modes and splits data 635 may be received via adaptive VLC encoder for joint modes and splits module 631, which may apply adaptive variable length coding to joint coding of modes and splits data 635 to generate compressed joint coding of modes and splits data 639. In some examples, adaptive variable length coding may include content adaptive variable length coding adaptive to the content of joint coding of modes and splits data 635 or other system parameters or the like.

As discussed, in some examples, modes and splits data may be coded jointly via adaptive VLC encoder for joint modes and splits module 631. In some examples, modes and splits data may be coded separately via symbol-run VLC and/or proxy VLC encoder for modes module 632 and symbol-run VLC and/or proxy VLC encoder for splits module 633, as is discussed below. In some examples, encoder 100 (via, e.g., adaptive entropy encoder 110 and/or encode controller 103) to code jointly or separately based on results of

a comparison of the coding techniques to determine which technique compresses the data most efficiently.

As shown, modes information data 636 may be received via symbol-run VLC and/or proxy VLC encoder for modes module 632. In some examples, an entropy encoding technique may be determined for modes information data 636 based on a parameter, parameters or characteristics of modes information data 636 or other system parameters (e.g., compression efficiency, a characteristic of the data, and so on), as discussed herein. The determined entropy encoding technique may be applied to modes information data 636 to generate compressed modes information data 642. As shown, in some examples, the entropy encoding technique for modes information data 636 may be one of an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique, as described above.

As shown, split/not-split information data 637 may be received via symbol-run VLC and/or proxy VLC encoder for splits module 633. In some examples, an entropy encoding technique may be determined for split/not-split information data 637 based on a parameter, parameters or characteristics of split/not-split information data 637 or other system parameters (e.g., compression efficiency, a characteristic of the data, and so on), as discussed herein. The determined entropy encoding technique may be applied to split/not-split information data 637 to generate compressed split/not-split information data 643. As shown, in some examples, the entropy encoding technique for split/not-split information data 637 may be one of an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique, as described above.

As shown, prediction reference information data 638 may be received via symbol-run VLC and/or proxy VLC encoder for reference types module 634. In some examples, an entropy encoding technique may be determined for prediction reference information data 638 based on a parameter, parameters or characteristics of prediction reference information data 638 or other system parameters (e.g., compression efficiency, a characteristic of the data, and so on), as discussed herein. The determined entropy encoding technique may be applied to prediction reference information data 638 to generate compressed prediction reference information data 644. As shown, in some examples, the entropy encoding technique for prediction reference information data 638 may be one of an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique, as described above.

As shown, encoder module 404 may include symbol-run VLC and/or proxy VLC encoder for coded/not-coded data 641 and may receive video data 414. Video data 414 may have a data type such that video data 411 may coded/not-coded data. For example, a partition (or sub-partition) may be coded if it has any nonzero transform coefficients and a partition (or sub-partition) may be not-coded if it has all zero transform coefficients. In some examples, coded/not-coded data may not be needed for partitions having an intra or skip mode. In some examples, coded/not-coded data may be needed for partitions having an auto, inter, or multi mode. In some examples, an entropy encoding technique may be determined for video data 414 based on a parameter, parameters or characteristics of video data 414 or other system parameters. In some examples, the entropy encoding technique for video data 414 may be one of an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique, as described above. The determined entropy encoding technique may be applied to video data 414 to generate compressed video data 424. In some examples, encoder module 404 may determine which technique provides the greatest compression efficiency as discussed such that the parameter(s) associated with video data 411 may be the number of bits needed for each coding technique or the like. In some examples, the parameter associated with video data 411 may be based on a characteristic of the video data such that the determined encoding technique may be adaptive to the received video data.

As shown, in some examples, encoder module 405 may include an adaptive vector VLC encoder for transform coefficients module 651 and/or an adaptive 1D and 2D VLC encoder for transform coefficients module 652.

As shown, adaptive vector VLC encoder for transform coefficients module 651 may receive transform coefficients data 653, which may include transform coefficient data for blocks or partitions or sub-partitions having a block or partition or sub-partition size of 2 in one dimension (e.g., 2xK partitions or Kx2 sized partitions). An adaptive vector variable length coding technique may be applied to transform coefficients data 653 to generate compressed transform coefficients data 655. In some examples, the adaptive vector variable length coding technique may include a quad-tree division of the block or partition or the, representing each quadrant generated via the quad-tree division with a single vector codeword that represents all coefficients with a single index value, and entropy coding the codeword using a variable length coding technique or the like.

Also as shown, adaptive 1D and 2D VLC encoder for transform coefficients module 652 may receive transform coefficient data 654 and may implement an adaptive 1-dimensional variable length coding for blocks or partitions or sub-partitions of size 4x4 and an adaptive 2-dimensional variable length coding for all other block or partition or sub-partition sizes (e.g., 4x8, 8x4, 8x8, 16x16, 32x32, 64x64, and so on) to generate compressed transform coefficient data 656. As discussed, transform coefficient data 653, 654 may result from a forward transform of rectangular (or square or the like) partitions of pixel data or rectangular (or square or the like) of pixel difference values implemented via adaptive transform module 108 followed by a quantization of the resulting coefficients via adaptive quantize module 109. In some examples, the transform coefficient data may be scanned to convert it to a 1-dimensional frequency ordered partition via encoder module 405. Such conversion may be highly adaptive any partition size (e.g., 24 or more or 32 or more partition sizes), different data types (e.g., discrete cosine transform coefficients or hybrid parametric Haar transform coefficients or the like of either intra or inter partitions), and/or different quantizer set characteristics (e.g., various combinations of quantizer parameters and/or matrices). Further, a block or partition or sub-partition may belong to different picture types: I-picture (e.g., intra compensation only), P-picture (e.g., predictive) or F-picture (e.g., functional) and/or may represent different types of signal or data (e.g., luma or chroma or the like), which may be quantized with different quantizer setting.

As shown, encoder module 406 may include a motion vector predictor module 661, an adaptive VLC encoder for motion vector differences module 662, and/or a differencer 663. As shown, encoder module 406 may receive video data 416, which may include motion vector data, via motion vector predictor module 661. Motion vector predictor module 661 may perform motion vector prediction based on video data 416 (e.g., the motion vector data of video data 516) using original motion vector(s) to generate associated predicted motion vector(s). In some examples, the motion vector prediction may be based on immediate neighbors to left, right, above, or below the motion vector being predicted. In some examples, other spatial neighbors that may share the same or similar characteristics may be used. For example, a number of different types of prediction may be adaptively selected and the selection information may provided to decoder 200 via bitstream 111. Differencer 663 may difference the predicted motion vector(s) and the original motion vector(s) to generate difference motion vector(s) for entropy coding. As shown adaptive VLC encoder for motion vector differences module 662 may apply an adaptive variable length coding technique to the difference motion vector(s) to generate

compressed video data 526. In some examples, differential (e.g., difference) motion vectors may have twice the range of original motion vectors. Further 1/8th pel precision motion compensation may expand the range of the difference motion vector by a factor of 8. In some examples, to address such expansion, classification of large space(s) into smaller subintervals and indexing of vectors inside the subinterval may be used.

As shown, encoder module 407 may include adaptive VLC and/or arithmetic encoder for intra-prediction and directions data module 671 and may receive video data 417. Video data 517 may have a data type such that video data 417 may include intra-prediction type or intra-prediction direction data. In some examples, an entropy encoding technique may be determined for video data 417 based on a parameter, parameters or characteristics of video data 417 or other system parameters (e.g., compression efficiency or the like) as discussed herein. In some examples, the entropy encoding technique for video data 417 may be one of an adaptive variable length coding technique or an arithmetic coding technique, as described above. The determined entropy encoding technique may be applied to video data 417 to generate compressed video data 427. In some examples, the adaptive variable length coding technique may include a content adaptive variable length coding based on the content of video data 417. In some examples, the arithmetic coding technique may include a content adaptive binary arithmetic coding based on the content of video data 417. In some examples, video data 417 may support 9 or more prediction directions and a variety of prediction types including planar, Binary Tree Predictive Coding (BTPC), or the like.

As shown in FIG. 6 and discussed above with respect to FIG. 4, the output encoder modules 401–407 (via the associated sub-modules) may be input to bitstream assembler 408, which may output a multiplexed bitstream formatted per the bitstream syntax, as discussed above.

FIG. 7 is an illustrative diagram of an example entropy decoder module 202, arranged in accordance with at least some implementations of the present disclosure. As shown and as discussed above with respect to FIG. 5, entropy decoder module 202 may include headers, parameters and maps data decoder module 501, picture partitions, prediction partitions, and coding partitions decoder module 502, coding modes and reference types decoder module 503, coded/not-coded data decoder module 504, transform coefficients decoder module 505, motion vector and differential motion vector decoder module 506, intra-prediction and direction data decoder module 507, and/or bitstream

disassembler module 508. In some examples, entropy decoder module 202 (and the pertinent sub-modules) may perform an inverse technique with respect to entropy encoder module 110 (and the pertinent sub-modules) such that there may be a one-to-one correspondence between encoder modules (and sub-modules) and decoder modules (and sub-modules)

As shown, bitstream disassembler module 508 may receive input bitstream 201. In some examples, input bitstream 201 may be a valid bitstream such as, for example, a valid next generation video (NGV) coded bitstream, which may follow a NGV bitstream syntax specification or any valid bitstream as discussed herein. As discussed with respect to FIG. 5, bitstream disassembler module 508 may disassemble input bitstream 201 to determine compressed video data 511–517, which may each have one or more component parts as discussed further below. For example, bitstream disassembler module 508 may use a predefined syntax or specification to divide input bitstream 201 into component compressed video data 511–517 by data type for decompression via decoder modules 501–507. In some examples, bitstream disassembler module 508 may perform an inverse operation with respect to bitstream assembler module 308. In some examples, the disassembling of input bitstream 201 may be characterized as a de-multiplexing.

As shown, decoder module 501 may include adaptive VLC, symbol-run VLC, and/or proxy VLC decoder for headers, parameters, and maps data module 711 and may receive compressed video data 511. In some examples, compressed video data 511 may include header data (e.g., sequence and/or picture level bitstream headers), morphing parameters, synthesizing parameters, and/or global maps data entropy encoded using one of an adaptive symbol-run variable length coding technique, an adaptive proxy variable length coding technique, or a variable length coding table or tables compression technique. In some examples, adaptive VLC, symbol-run VLC, and/or proxy VLC decoder for headers, parameters, and maps data module 711 may determine an entropy decoding technique applicable to compressed video data 511 and decode compressed video data 511 using the applicable technique to generate video data 521. In some examples, the applicable technique may be determined based on an indicator, parameter, header data, or the like conveyed via input bitstream 201.

As shown, decoder module 502 may include adaptive codebook VLC decoder for picture partitions module 721, symbol-run VLC and/or proxy VLC decoder for intra-prediction partitions data module 722, and/or symbol-run VLC and/or proxy VLC decoder

for inter-prediction partitions and coding partitions data module 723 and may receive compressed video data 512.

As shown, compressed picture slices or regions data 724 may be received via adaptive codebook VLC decoder for picture partitions module 721. In some examples, adaptive codebook VLC decoder for picture partitions module 721 may apply adaptive codebook variable length coding to compressed picture slices or regions data 724 to generate picture slices or regions data 727. As discussed, adaptive codebook variable length coding may include content adaptive variable length coding using an codebook adaptive to the content of compressed picture slices or regions data 724 or other system parameters or the like. In some examples, the codebook may be implemented via adaptive codebook VLC decoder for picture partitions module 721.

As shown, symbol-run VLC and/or proxy VLC decoder for intra-prediction partitions data 722 may receive compressed intra-prediction partition data 725. In some examples, compressed intra-prediction partition data 725 may include compressed intra-prediction partition data entropy encoded using one of an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique, as described herein. In some examples, symbol-run VLC and/or proxy VLC decoder for intra-prediction partitions data 722 may determine an entropy decoding technique applicable to compressed intra-prediction partition data 725 and entropy decode compressed intra-prediction partition data 725 using the applicable technique to generate intra-prediction partition data 728. In some examples, the applicable coding technique may be determined based on an indicator, parameter, header data, or the like conveyed via input bitstream 201.

As shown, symbol-run VLC and/or proxy VLC decoder for inter-prediction partitions and coding partitions data 723 may receive compressed inter-prediction and coding partition data 726. In some examples, compressed inter-prediction and coding partition data 726 may include compressed inter-prediction and coding partition data entropy encoded using one of an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique, as described herein. In some examples, symbol-run VLC and/or proxy VLC decoder for inter-prediction partitions and coding partitions data 723 may determine an entropy decoding technique applicable to compressed inter-prediction and coding partition data 726 and entropy decode compressed inter-prediction and coding partition data 726 using the applicable technique to generate inter-prediction and coding partition data 729. In some examples, the applicable coding

technique may be determined based on an indicator, parameter, header data, or the like conveyed via input bitstream 201.

As shown, decoder module 503 may include adaptive VLC decoder for joint modes and splits module 731, symbol-run VLC and/or proxy VLC decoder for modes module 732, symbol-run VLC and/or proxy VLC decoder for splits module 733, and/or symbol-run VLC and/or proxy VLC decoder for reference types module 734 and may receive compressed video data 513.

As discussed above with respect to encoder module 403, in some examples, modes and splits data may be coded jointly and, in some examples, modes and splits data may be coded separately. In some examples, adaptive VLC decoder for joint modes and splits module 731 may decode jointly coded data, and symbol-run VLC and/or proxy VLC decoder for modes module 732 and symbol-run VLC and/or proxy VLC decoder for splits module 733 may decode separately coded data. In some examples, whether data is jointly or separately coded may be indicated via input bitstream 201.

As shown, compressed joint coding of modes and splits data 735 may be received via adaptive VLC decoder for joint modes and splits module 731. In some examples, adaptive VLC decoder for joint modes and splits module 731 may apply adaptive variable length coding to compressed joint coding of modes and splits data 735 to generate joint coding of modes and splits data 739. As discussed, adaptive variable length coding may be content adaptive variable length coding adaptive to the content of compressed joint coding of modes and splits data 735 or other system parameters or the like.

As shown, symbol-run VLC and/or proxy VLC decoder for modes module 732 may receive compressed modes information data 736. In some examples, compressed modes information data 736 may include compressed modes information data entropy encoded using one of an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique, as described herein. In some examples, symbol-run VLC and/or proxy VLC decoder for modes module 732 may determine an entropy decoding technique applicable to compressed modes information data 736 and entropy decode compressed modes information data 736 using the applicable technique to generate modes information data 742. In some examples, the applicable coding technique may be determined based on an indicator, parameter, header data, or the like conveyed via input bitstream 201.

As shown, symbol-run VLC and/or proxy VLC decoder for splits module 733 may receive compressed splits information data 737. In some examples, compressed splits information data 737 may include compressed splits information data entropy encoded using one of an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique, as described herein. In some examples, symbol-run VLC and/or proxy VLC decoder for splits module 733 may determine an entropy decoding technique applicable to compressed splits information data 737 and entropy decode compressed splits information data 737 using the applicable technique to generate splits information data 743. In some examples, the applicable coding technique may be determined based on an indicator, parameter, header data, or the like conveyed via input bitstream 201.

As shown, symbol-run VLC and/or proxy VLC decoder for reference types module 734 may receive compressed reference types information data 738. In some examples, compressed reference types information data 738 may include compressed reference types information data entropy encoded using one of an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique, as described herein. In some examples, symbol-run VLC and/or proxy VLC decoder for reference types module 734 may determine an entropy decoding technique applicable to compressed reference types information data 738 and entropy decode compressed reference types information data 738 using the applicable technique to generate reference types information data 744. In some examples, the applicable coding technique may be determined based on an indicator, parameter, header data, or the like conveyed via input bitstream 201.

As shown, decoder module 504 may include symbol-run VLC and/or proxy VLC decoder for coded/not-coded data module 741 and may receive compressed video data 514. In some examples, compressed video data 514 may include coded/not-coded, as discussed herein, entropy encoded using one of an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique. In some examples, symbol-run VLC and/or proxy VLC decoder for coded/not-coded data module 741 may determine an entropy decoding technique applicable to compressed video data 514 and decode compressed video data 514 using the applicable technique to generate video data 524. In some examples, the applicable technique may be determined based on an indicator, parameter, header data, or the like conveyed via input bitstream 201. In some examples,

the applicable technique may be one of an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique.

As shown, decoder module 505 may include adaptive vector VLC decoder for transform coefficients module 751 and/or adaptive 1D and 2D VLC decoder for transform coefficients module 752 and may receive compressed video data 515.

As shown, adaptive vector VLC decoder for transform coefficients module 751 may receive compressed transform coefficients data 753, which may include compressed transform coefficients data for blocks or partitions or sub-partitions having a size of 2 in one dimension (e.g., 2xK partitions or Kx2 sized partitions), as discussed herein. In some examples, adaptive vector VLC decoder for transform coefficients module 751 may apply an adaptive vector variable length coding technique to entropy decode compressed transform coefficients data 753 to generate transform coefficients data 755. As discussed, In some examples, the adaptive vector variable length coding technique may include using a variable length codeword to generate all coefficients of a quad-tree division of the block, which may be generated via merging the quad-tree division.

As shown, adaptive 1D and 2D VLC decoder for transform coefficients module 752 may receive compressed transform coefficients data 754, which may include compressed transform coefficients data for blocks or partitions or sub-partitions of size 4x4 and all other block or partition or sub-partition sizes (e.g., 4x8, 8x4, 8x8, 16x16, 32x32, 64x64, and so on). In some examples, adaptive 1D and 2D VLC decoder for transform coefficients module 752 may apply an adaptive 1-dimensional variable length coding for blocks or partitions or sub-partitions of size 4x4 and an adaptive 2-dimensional variable length coding for all other block or partition or sub-partition sizes (e.g., 4x8, 8x4, 8x8, 16x16, 32x32, 64x64, and so on) to entropy decode compressed transform coefficients data 754 to generate transform coefficients data 756.

As shown, decoder module 506 may include an adaptive VLC decoder for motion vector differences module 762, a motion vector predictor 761 and an adder 763, and may receive compressed video data 516. In some examples, adaptive VLC decoder for motion vector differences module 762 may decode compressed video data 516 to generate motion vector differences. Furthermore, motion vector predictor 761 may generate prediction motion vectors using previously decoded neighboring motion vectors in analogy to the techniques discussed with respect to motion vector predictor module 661. As shown, decoded difference motion vector(s) may be added via adder 763 to prediction motion

vector(s) to generate reconstructed motion vector(s), which may be output as a part of video data 526 and further used to perform motion vector prediction for other motion vectors via motion vector predictor module 761.

As shown, decoder module 507 may include adaptive VLC and/or arithmetic decoder for intra-prediction type and direction module 771 and may receive compressed video data 517. In some examples, compressed video data 517 may include intra-prediction type and direction data entropy encoded using one of an adaptive VLC technique or an arithmetic coding technique. In some examples, adaptive VLC and/or arithmetic decoder for intra-prediction type and direction module 771 may determine an entropy decoding technique applicable to compressed video data 517 and decode compressed video data 517 using the applicable technique to generate video data 527. In some examples, the applicable technique may be determined based on an indicator, parameter, header data, or the like conveyed via input bitstream 201.

As discussed, video data 521–527 (including various sub-components as discussed) may be transmitted to various components of decoder 200. Further, as discussed, decoder 200 may use entropy decoded video data 521–527 to generate video frames, which may be output, via display video 219, for presentment or display via a display device for a user.

FIG. 8 is a flow diagram illustrating an example process 800, arranged in accordance with at least some implementations of the present disclosure. Process 800 may include one or more operations, functions or actions as illustrated by one or more of operations 802, 804, 806, and/or 808. Process 800 may form at least part of a next generation video coding process. By way of non-limiting example, process 800 may form at least part of a next generation video encoding process as undertaken by encoder system 100 of FIG. 1 and/or entropy encoder module 110 of FIGS. 5 or 7.

Process 800 may begin at operation 802, "Obtain First and Second Video Data of Different Types for Entropy Encoding", where first and second (or additional) video data of different types may be obtained or received for entropy coding. For example, two or more of video data 411–417 (and/or any sub-components of video data 511–517) may be received. In some examples, video data 411–417 may be received via adaptive entropy encoder 110. As discussed, the first and second (or additional) video data may be of different types such as any of the types or sub-component types as discussed with respect to video data 411–417 or elsewhere herein.

Process 800 may continue at operation 804, "Determine an Entropy Encoding Technique for the First Video Data", where a first entropy encoding technique may be determined for the first video data. As discussed, in some examples, one or more of encoder modules 401–407 may determine a coding technique for video data 411–417 from various coding technique options. For example, encoder module 401 may determine a coding technique for video data 411 from one of an adaptive symbol-run variable length coding technique, an adaptive proxy variable length coding technique, or a variable length coding table or tables compression technique. Further, encoder module 402 may determine a coding technique for intra-prediction partition data 625 of video data 402 from one of an adaptive symbol-run variable length coding technique and an adaptive proxy variable length coding technique, and so on. A wide range of examples have been provided herein with respect to FIGS. 4 and 6 and will not be repeated here for the sake of brevity. As discussed, in some examples, two ore more types of video data may be received. In some examples, an entropy encoding technique may be determined for two, three, or more types of video data as described herein.

As discussed, an entropy encoding technique may be determined for video data. In some examples, a selected coding table associated with the first encoding technique may be determined from two or more available tables. For example, the table selection may be made based on a bit count comparison, a coding efficiency, or the like between the available tables. For example, any of the described entropy encoding techniques discussed herein may have multiple available tables. In some examples, a table indicator may be generated and provided in bitstream 900 indicating the selected table for the video data. For example, bitstream 900 may include indicators indicating the selected encoding technique and the selected coding table. Further an indicator or indicators associated with the length of the video data may be included into bitstream 900. For example, the length or number of tuples in the video data (e.g. the length of the string of video) may be provided and any remainder portion (e.g., if the length is not evenly divided into tuples) may be coded using a bit map technique and provided into the output bitstream.

Process 800 may continue at operation 806, "Entropy Encode the First Video Data using the First Entropy Encoding Technique and Entropy Encode the Second Video Data to Generate First and Second Compressed Video Data", where the first video data may be entropy encoded using the first entropy encoding technique to generate first compressed video data and the second video data may be compressed using a second entropy encoding technique to generate first and second compressed video data.

Process 800 may continue at operation 808, "Assemble the First and Second Compressed Video Data to Generate an Output Bitstream", where the first compressed video data and the second compressed video data may be assembled to generate an output bitstream. For example, bitstream assembler 508 may assemble or multiplex the first compressed video data and the second compressed video data to generate output bitstream 111 as discussed herein.

As discussed, in some examples, output bitstream 111 may be multiplexed with an audio stream (coded or uncoded) to generate a multiplexed audio-visual stream. Further, as discussed, in some examples, one or more of the video data may be preprocessed or otherwise manipulated prior to entropy coding. For example, for motion vector data, motion vector prediction may be performed to generate predicted motion vector(s), the predicted motion vector(s) may be differenced with the original motion vector(s) to generate difference motion vector(s), and the difference motion vector(s) may be entropy coded as discussed herein. Further, in some examples, each of the seven encoder modules 401–407 may be implemented simultaneously to operate on seven types of video data 411–417. Process 800 may be implemented via adaptive entropy encoder module 110 as discussed herein. Further, process 800 may be repeated either in serial or in parallel on any number of instantiations of video data.

As discussed, video data of different types may be entropy coded using a variety of determined (or predetermined) adaptive entropy coding techniques to generate compressed video data. The compressed video data may be assembled to generate an output bitstream.

FIG. 9 illustrates an example bitstream 900, arranged in accordance with at least some implementations of the present disclosure. In some examples, bitstream 900 may correspond to output bitstream 111 as shown in FIGS. 1, 3a, 4, and 6 and/or input bitstream 201 as shown in FIGS. 2, 3b, 5, and 7. Although not shown in FIG. 9 for the sake of clarity of presentation, in some examples bitstream 900 may include a header portion and a data portion. In various examples, bitstream 900 may include data, indicators, index values, mode selection data, or the like associated with encoding compressed video as discussed herein. As shown, in some examples, bitstream 900 may include indicator data 901, compressed video data 421, compressed video data 422, compressed video data 423, compressed video data 424, compressed video data 425, compressed video data 426, and/or compressed video data 427. The illustrated data may be in any order in bitstream 900 and may be adjacent or separated by any other of a variety of additional data for

coding video. As discussed, bitstream 900 may also include indicators indicating the selected encoding technique and the selected coding table (e.g., in indicator data 901). Further an indicator or indicators associated with the length of the video data may be included into bitstream 900. For example, the length or number of tuples in the video data (e.g. the length of the string of video) may be provided and any remainder portion (e.g., if the length is not evenly divided into tuples) may be coded using a bit map technique and provided into the output bitstream.

In some examples, compressed video data 421–427 may include any compressed video data encoded via any technique as discussed herein. In some examples, indicator data 901 may include header data, mode indicator data, and/or data indicating entropy encoding techniques associated with compressed video data 421–427. For example, indicator data 901 may indicate an entropy coding technique used for compressing video data 421, an entropy coding technique used for compressing portions of video data 422, an entropy coding technique used for compressing portions of video data 423, an entropy coding technique used for compressing video data 424, and/or an entropy coding technique used for compressing video data 427, as discussed herein with respect to FIGS. 4 and 6.

As discussed, bitstream 900 may be generated by an encoder such as, for example, encoder 100 and/or received by a decoder 200 for decoding such that video frames may be presented via a display device.

FIG. 10 is a flow diagram illustrating an example process 1000, arranged in accordance with at least some implementations of the present disclosure. Process 1000 may include one or more operations, functions or actions as illustrated by one or more of operations 1002, 1004, 1006, and/or 1008. Process 1000 may form at least part of a next generation video coding process. By way of non-limiting example, process 1000 may form at least part of a next generation video decoding process as undertaken by decoder system 200 of FIG. 2.

Process 1000 may begin at operation 1002, "Receive Entropy Encoded Bitstream", where an entropy encoded bitstream may be received. For example, a bitstream encoded as discussed herein may be received at a video decoder. In some examples, bitstream 100 or 900 or the like may be received via decoder 200.

Process 1000 may continue at operation 1004, "Disassemble the Entropy Encoded Bitstream to First Compressed Video Data and Second Compressed Video Data", where

the received bitstream may be disassembled to determine different types of compressed video data. For example, bitstream 201 may be disassembled via bitstream disassembler 508 to generate compressed video data 511–517 (and any sub-component video data) as illustrated in FIG. 7. For example, the disassembled compressed video data may include first compressed video and second compressed video data.

Process 1000 may continue at operation 1006, "Entropy Decode the First and Second Compressed Video Data to Generate First Video Data and Second Video Data", where first and second compressed video data may be entropy decoded to generate (decompressed) first and second video data. As discussed, in some examples, one or more of decoder modules 501–507 may determine a coding technique for compressed video data 511–517 from various coding technique options. For example, decoder module 501 may determine a coding technique for compressed video data 511 from one of an adaptive symbol-run variable length coding technique, an adaptive proxy variable length coding technique, or a variable length coding table or tables compression technique based on an indicator or indicators provided via the input bitstream. Further, decoder module 502 may determine a coding technique for compressed intra-prediction partition data 725 of video data 512 from one of an adaptive symbol-run variable length coding technique and an adaptive proxy variable length coding technique, and so on. A wide range of examples have been provided herein with respect to FIGS. 5 and 7 and will not be repeated here for the sake of brevity. As discussed, in some examples, two ore more types of video data may be received. In some examples, an entropy decoding technique may be determined for two, three, or more types of video data as described herein.

Process 1000 may continue at operation 1008, "Decode the First and Second Video Data to Generate a Video Frame", where the first and second video data (and any other entropy decoded video data) may be decoded to generate video frame(s). The video frames may be suitable for presentment to a user via a display device, for example.

As discussed, in some examples, one or more of the entropy decoded video data may be post-processed or otherwise manipulated prior to further decoding. For example, entropy decoded difference motion vector(s) may be added to prediction motion vector(s) to generate reconstructed motion vector(s), which may be output for use in motion vector prediction (e.g., inter-prediction) via decoder 200.

Various components of the systems described herein may be implemented in software, firmware, and/or hardware and/or any combination thereof. For example,

various components of encoder 100 or encoder 200 may be provided, at least in part, by hardware of a computing System-on-a-Chip (SoC) such as may be found in a computing system such as, for example, a smart phone. Those skilled in the art may recognize that systems described herein may include additional components that have not been depicted in the corresponding figures. For example, the systems discussed herein may include additional components such as bit stream multiplexer or de-multiplexer modules and the like that have not been depicted in the interest of clarity.

Some additional and/or alternative details related to process 800, 1000 and other processes discussed herein may be illustrated in one or more examples of implementations discussed herein and, in particular, with respect to FIG. 11 below.

FIG. 11 is an illustrative diagram of an example video coding system 1610 and video coding process 1100 in operation, arranged in accordance with at least some implementations of the present disclosure. In the illustrated implementation, process 1100 may include one or more operations, functions or actions as illustrated by one or more of actions 1100–1112. By way of non-limiting example, process 1100 will be described herein with reference to example video coding system 1610 including encoder 100 of FIG. 1 and decoder 200 of FIG. 2, as is discussed further herein below with respect to FIG. 16. In various examples, process 1100 may be undertaken by a system including both an encoder and decoder or by separate systems with one system employing an encoder (and optionally a decoder) and another system employing a decoder (and optionally an encoder). It is also noted, as discussed above, that an encoder may include a local decode loop employing a local decoder as a part of the encoder system.

In the illustrated implementation, video coding system 1610 may include logic circuitry 1150, the like, and/or combinations thereof. For example, logic circuitry 1150, may include encoder 100 and may include any modules as discussed with respect to FIG. 1 and/or FIGS. 3 and 5 and decoder 200 and may include any modules as discussed with respect to FIG. 2 and/or FIGS. 4 and 6. Although video coding system 1610, as shown in FIG. 16, may include one particular set of blocks or actions associated with particular modules, these blocks or actions may be associated with different modules than the particular module illustrated here. Although process 1100, as illustrated, is directed to encoding and decoding, the concepts and/or operations described may be applied to encoding and/or decoding separately, and, more generally, to video coding.

Process 1100 may begin at operation 1101, "Obtain Video Data of Different Types", where video data of different types may be received for entropy encoding. For example two or more types of video data may be received for entropy encoding. For example, two or more of video data 411–417 (and/or any sub-components of video data 411–417) may be received via adaptive entropy encoder 110. As discussed, the first and second (or additional) video data may be of different types such as any of the types or sub-component types as discussed with respect to video data 411–417 or elsewhere herein.

Process 1100 may continue from operation 1101 to operation 1102, "Determine Entropy Encoding Technique(s) for one or more Types of Video Data", where entropy encoding technique(s) may be determined for one or more of the video data types. As discussed, in some examples, one or more of encoder modules 401–407 may determine a coding technique for video data 411–417 from various coding technique options. For example, encoder module 401 may determine a coding technique for video data 411 from one of an adaptive symbol-run variable length coding technique, an adaptive proxy variable length coding technique, or a variable length coding table or tables compression technique. Further, encoder module 402 may determine a coding technique for intra-prediction partition data 625 of video data 402 from one of an adaptive symbol-run variable length coding technique and an adaptive proxy variable length coding technique, and so on as discussed herein.

Process 1100 may continue at operation 1103, "Entropy Encode the Video Data using the Determined Technique(s)", the video data may be entropy encoded using the determined technique(s). For example, first video data may be entropy encoded using a first entropy encoding technique to generate first compressed video data. In some examples, second, third, or more additional video data may be entropy encoded using second, third or more additional entropy encoding techniques as discussed herein to generate first, second, third, and so on respective compressed video data.

Process 1100 may continue from operation 1103 to operation 1104, "Assemble the Compressed Video Data", where the compressed video data of different types may be assembled to generate an output bitstream. For example, bitstream assembler 408 may assemble or multiplex the compressed video data to generate output bitstream 111 as discussed herein.

Process 1100 may continue from operation 1104 to operation 1105, "Optionally Multiplex Video Bitstream with Audio Stream", where the video bitstream may be

optionally multiplexed with a coded or uncoded audio stream to generate an audio-visual bitstream.

Process 1100 may continue from operation 1105 or operation 1104 to operation 1106, "Transmit Bitstream", where the bitstream may be transmitted. For example, video coding system 1610 may transmit output bitstream 111 or bitstream 800 or the like via an antenna 1102 (please refer to FIG. 16).

Operations 1101–1106 may provide for video encoding and bitstream transmission techniques, which may be employed by an encoder system as discussed herein. The following operations, operations 1107–1112 may provide for video decoding and video display techniques, which may be employed by a decoder system as discussed herein.

Process 1100 may continue at operation 1107, "Receive Bitstream", where the encoded bitstream may be received. For example, input bitstream 100, 201, or bitstream 800 or the like may be received via decoder 200. In some examples, the bitstream may include different types of entropy encoded data as discussed herein.

Process 1100 may continue from operation 1107 to operation 1108, "Disassamble Bitstream to Generate Compressed Video Data of Different Types", where the received bitstream may be disassembled to determine different types of compressed video data. For example, bitstream 201 may be disassembled via bitstream disassembler 508 to generate compressed video data 511–517 (and any sub-component video data) as illustrated in FIG. 7. For example, the disassembled compressed video data may include first, second, third, or more compressed video data.

Process 1100 may continue from operation 1108 to operation 1109, "Determine Entropy Decoding Technique(s) for one or more Types of Compressed Video Data", entropy decoding technique(s) may be determined for one or more of the compressed video data types. In some examples, the entropy decoding technique(s) may be determined based on a flag or indicator or the like conveyed via the received bitstream. As discussed, in some examples, one or more of decoder modules 501–507 may determine a coding technique for compressed video data 511–517 from various coding technique options. For example, decoder module 401 may determine a coding technique for compressed video data 511 from one of an adaptive symbol-run variable length coding technique, an adaptive proxy variable length coding technique, or a variable length coding table or tables compression technique based on an indicator or indicators provided via the input bitstream.

Further, decoder module 502 may determine a coding technique for compressed intra-prediction partition data 725 of video data 512 from one of an adaptive symbol-run variable length coding technique and an adaptive proxy variable length coding technique, and so on. A wide range of examples have been provided herein and will not be repeated here for the sake of brevity. As discussed, in some examples, two ore more types of video data may be received. In some examples, an entropy decoding technique may be determined for two, three, or more types of video data as described herein.

Process 1100 may continue from operation 1109 to operation 1610, "Entropy Decode the Compressed Video Data", where the compressed video data may be entropy decoded based on the determined entropy decoding techniques. For example, compressed video data 511–517 may be entropy decoded via decode modules 501–507.

Process 1100 may continue from operation 1610 to operation 1111, "Decode the Entropy Decoded Video Data to Generate Video Frame(s)", where the first and second video data (and any other entropy decoded video data) may be decoded to generate video frame(s). The video frames may be suitable for presentment to a user via a display device, for example. For example, the video frame may be determined based on the implementation of decode techniques discussed with respect to decoder 200.

Process 1100 may continue from operation 1111 to operation 1112, "Transmit Video Frames for Presentment via a Display Device", where generated video frame(s) may be transmitted for presentment via a display device. For example, the video frame(s) may be transmitted to a display device 1105 (as shown in FIG. 16) for presentment. In some examples, display device 1105 may display the video frames to a user, for example.

While implementation of the example processes herein may include the undertaking of all operations shown in the order illustrated, the present disclosure is not limited in this regard and, in various examples, implementation of the example processes herein may include the undertaking of only a subset of the operations shown and/or in a different order than illustrated.

In addition, any one or more of the operations discussed herein may be undertaken in response to instructions provided by one or more computer program products. Such program products may include signal bearing media providing instructions that, when executed by, for example, a processor, may provide the functionality described herein. The computer program products may be provided in any form of one or more machine-readable

media. Thus, for example, a processor including one or more processor core(s) may undertake one or more of the operations of the example processes herein in response to program code and/or instructions or instruction sets conveyed to the processor by one or more machine-readable media. In general, a machine-readable medium may convey software in the form of program code and/or instructions or instruction sets that may cause any of the devices and/or systems described herein to implement at least portions of the video systems as discussed herein.

As used in any implementation described herein, the term "module" refers to any combination of software logic, firmware logic and/or hardware logic configured to provide the functionality described herein. The software may be embodied as a software package, code and/or instruction set or instructions, and "hardware", as used in any implementation described herein, may include, for example, singly or in any combination, hardwired circuitry, programmable circuitry, state machine circuitry, and/or firmware that stores instructions executed by programmable circuitry. The modules may, collectively or individually, be embodied as circuitry that forms part of a larger system, for example, an integrated circuit (IC), system on-chip (SoC), and so forth. For example, a module may be embodied in logic circuitry for the implementation via software, firmware, or hardware of the coding systems discussed herein.

FIG. 12 is a flow diagram illustrating an example process 1200, arranged in accordance with at least some implementations of the present disclosure. Process 1200 may include one or more operations, functions or actions as illustrated by one or more operations. Process 1200 may form at least part of a next generation video coding process. By way of non-limiting example, process 1200 may form at least part of a next generation video encoding process as undertaken by encoder system 100 of FIG. 1 and/or any other encoder system or subsystems described herein.

Process 1200 may begin at operation 1202, "Receive Input Video Frames of a Video Sequence", where input video frames of a video sequence may be received via encoder 100 for example.

Process 1200 may continue at operation 1204, "Associate a Picture Type with each Video Frame", where a picture type may be associated with each video frame in a group of pictures via content pre-analyzer module 102 for example. For example, the picture type may be F/B-picture , P-picture, or I-picture, or the like. In some examples, a video sequence may include groups of pictures and the processing described herein (e.g.,

operations 1203 through 1211) may be performed on a frame or picture of a group of pictures and the processing may be repeated for all frames or pictures of a group and then repeated for all groups of pictures in a video sequence.

Process 1200 may continue at operation 1206, "Divide a Picture into Tiles and/or Super-fragments and Potential Prediction Partitionings", where a picture may be divided into tiles or super-fragments and potential prediction partitions via prediction partitions generator 105 for example.

Process 1200 may continue at operation 1210, "For Potential Prediction Partitioning, Determine Potential Prediction Error", where, for each potential prediction partitioning, a potential prediction error may be determined. For example, for each prediction partitioning (and associated prediction partitions, prediction(s), and prediction parameters), a prediction error may be determined. For example, determining the potential prediction error may include differencing original pixels (e.g., original pixel data of a prediction partition) with prediction pixels. In some examples, the associated prediction parameters may be stored. As discussed, in some examples, the prediction error data partition may include prediction error data generated based at least in part on a previously decoded frame generated using at least one of a morphing technique or a synthesizing technique.

Process 1200 may continue at operation 1212, "Select Prediction Partitioning and Prediction Type and Save Parameters", where a prediction partitioning and prediction type may be selected and the associated parameters may be saved. In some examples, the potential prediction partitioning with a minimum prediction error may be selected. In some examples, the potential prediction partitioning may be selected based on a rate distortion optimization (RDO).

Process 1200 may continue at operation 1214, "Perform Transforms on Potential Coding Partitionings", where fixed or content adaptive transforms with various block sizes may be performed on various potential coding partitionings of partition prediction error data. For example, partition prediction error data may be partitioned to generate a plurality of coding partitions. For example, the partition prediction error data may be partitioned by a bi-tree coding partitioner module or a k-d tree coding partitioner module of coding partitions generator module 107 as discussed herein. In some examples, partition prediction error data associated with an F/B- or P-picture may be partitioned by a bi-tree coding partitioner module. In some examples, video data associated with an I-picture (e.g., tiles or super-fragments in some examples) may be partitioned by a k-d tree coding

partitioner module. In some examples, a coding partitioner module may be chosen or selected via a switch or switches. For example, the partitions may be generated by coding partitions generator module 107.

Process 1200 may continue at operation 1216, "Determine the Best Coding Partitioning, Transform Block Sizes, and Actual Transform", where the best coding partitioning, transform block sizes, and actual transforms may be determined. For example, various coding partitionings (e.g., having various coding partitions) may be evaluated based on RDO or another basis to determine a selected coding partitioning (which may also include further division of coding partitions into transform blocks when coding partitions to not match a transform block size as discussed). For example, the actual transform (or selected transform) may include any content adaptive transform or fixed transform performed on coding partition or block sizes as described herein.

Process 1200 may continue at operation 1218, "Quantize and Scan Transform Coefficients", where transform coefficients associated with coding partitions (and/or transform blocks) may be quantized and scanned in preparation for entropy coding.

Process 1200 may continue at operation 1222, "Entropy Encode Data associated with Each Tile or Super-fragment Decode, Such As Coding Partition Indicator(s), Block Size Data, Transform Type Data, Quantizer (Qp), and Quantized Transform Coefficients, Motion Vectors and Reference Type Data, Characteristic Parameters (e.g., mop, syp)", where data may be entropy encoded. For example, the entropy encoded data may include the coding partition indicators, block size data, transform type data, quantizer (Qp), quantized transform coefficients, motion vectors and reference type data, characteristic parameters (e.g., mop, syp), the like, and/or combinations thereof. Additionally or alternatively, the entropy encoded data may include prediction partitioning, prediction parameters, the selected coding partitioning, the selected characteristics data, motion vector data, quantized transform coefficients, filter parameters, selection data (such as mode selection data), and indictors.

Process 1200 may continue at operation 1223 "Apply DD/DB Filter, Reconstruct Pixel Data, Assemble into a Picture", where deblock filtering (e.g., DD or DB filters) may be applied, pixel data may be reconstructed, and assembled into a picture. For example, after a local decode loop (e.g., including inverse scan, inverse transform, and assembling coding partitions), prediction error data partitions may be generated. The prediction error data partitions may be added with a prediction partition to generate reconstructed

prediction partitions, which may be assembled into tiles or super-fragments. The assembled tiles or super-fragments may be optionally processed via deblock filtering and/or quality restoration filtering and assembled to generate a picture.

Process 1200 may continue at operation 1224 "Apply QR/LF Filter Save in Reference Picture Buffers", where quality restoration filtering (e.g., QR or LF filtering) may be applied, and the assembled picture may be saved in reference picture buffers. For example, in addition to or in the alternative to the DD/DB filtering, the assembled tiles or super-fragments may be optionally processed via quality restoration filtering and assembled to generate a picture. The picture may be saved in decoded picture buffer 119 as a reference picture for prediction of other (e.g., following) pictures.

Process 1200 may continue at operation 1225, "Apply AP/AM Filter, Determine Modifying (e.g., Morphing or Synthesizing) Characteristic Parameters for Generating Morphed or Synthesized Prediction Reference(s) and Perform Prediction(s)", where, modifying (e.g., morphing or synthesizing) characteristic parameters and prediction(s) may be performed and adaptive motion filtering or adaptive precision filtering (e.g., AP/AM Filter) may be applied. For example, modifying (e.g., morphing or synthesizing) characteristic parameters for generating morphed or synthesized prediction reference(s) may be generated and prediction(s) may be performed. Additionally, adaptive motion filtering or adaptive precision filtering may be applied at this point in the process.

As discussed, in some examples, inter-prediction may be performed. In some examples, up to 4 decoded past and/or future pictures and several morphing/synthesis predictions may be used to generate a large number of reference types (e.g., reference pictures). For instance in 'inter' mode, up to nine reference types may be supported in P-pictures, and up to ten reference types may be supported for F/B-pictures. Further, 'multi' mode may provide a type of inter prediction mode in which instead of 1 reference picture, 2 reference pictures may be used and P- and F/B-pictures respectively may allow 3, and up to 8 reference types. For example, prediction may be based on a previously decoded frame generated using at least one of a morphing technique or a synthesizing technique. In such examples, and the bitstream (discussed below with respect to operation 1212) may include a frame reference, morphing parameters, or synthesizing parameters associated with the prediction partition.

Process 1200 may continue at operation 1229 "Optionally Apply EP Filter and/or Optionally apply FI/FP Filter", where enhanced predicted partition (e.g., EP Filtering) or

FI/FP Filtering (e.g., fusion filtering or fusion improvement filtering) may be optionally applied. In some examples, a decision may be made regarding whether to utilize some form or FI/FP Filter (fusion improvement filtering/fusion filtering) or not to use FI/FP Filtering. When some form or FI/FP Filter (e.g., fusion filtering or fusion improvement filtering) is to be applied to the selected predicted partition the selected predicted partition and a second selected predicted partition may be assembled to generate at least a portion of an assembled picture. FI/FP Filtering may be applied to filter the portion of the assembled picture. FI/FP Filtering parameters (e.g., filtering parameters or fusion improvement filtering parameters) associated with the FI/FP Filtering may be generated and sent to the entropy coder subsystem.

In implementations where both EP Filtering or FI/FP Filtering are available, an indicator may be generated that indicates to the decoder system whether to use the enhanced predicted partition (e.g., EP Filtering) or the predicted partition data as the selected predicted partition for the prediction partition.

Operations 1202 through 1229 may provide for video encoding and bitstream transmission techniques, which may be employed by an encoder system as discussed herein.

FIG. 13 illustrates an example bitstream 1300, arranged in accordance with at least some implementations of the present disclosure. In some examples, bitstream 1300 may correspond to output bitstream 111 as shown in FIG. 1 and/or input bitstream 201 as shown in FIG. 2. Although not shown in FIG. 29 for the sake of clarity of presentation, in some examples bitstream 1300 may include a header portion and a data portion. In various examples, bitstream 1300 may include data, indicators, index values, mode selection data, or the like associated with encoding a video frame as discussed herein.

As discussed, bitstream 1300 may be generated by an encoder such as, for example, encoder 100 and/or received by a decoder 200 for decoding such that decoded video frames may be presented via a display device.

FIG. 14 is a flow diagram illustrating an example process 1400, arranged in accordance with at least some implementations of the present disclosure. Process 1400 may include one or more operations, functions or actions as illustrated by one or more operations. Process 1400 may form at least part of a next generation video coding process. By way of non-limiting example, process 1400 may form at least part of a next generation

video decoding process as undertaken by decoder system 200 and/or any other decoder system or subsystems described herein.

Process 1400 may begin at operation 1402, "Receive Encoded Bitstream", where a bitstream may be received. For example, a bitstream encoded as discussed herein may be received at a video decoder. In some examples, bitstream 900 or 1300 may be received via decoder 200.

Process 1400 may continue at operation 1404, "Decode the Entropy Encoded Bitstream to Determine Coding Partition Indicator(s), Block Size Data, Transform Type Data, Quantizer (Qp), Quantized Transform Coefficients, Motion Vectors and Reference Type Data, Characteristic Parameters (e.g., mop, syp)", where the bitstream may be decoded to determine coding partition indicators, block size data, transform type data, quantizer (Qp), quantized transform coefficients, motion vectors and reference type data, characteristic parameters (e.g., mop, syp), the like, and/or combinations thereof. Additionally or alternatively, the entropy encoded data may include prediction partitioning, prediction parameters, the selected coding partitioning, the selected characteristics data, motion vector data, quantized transform coefficients, filter parameters, selection data (such as mode selection data), and indictors.

Process 1400 may continue at operation 1406, "Apply Quantizer (Qp) on Quantized Coefficients to Generate Inverse Quantized Transform Coefficients", where quantizer (Qp) may be applied to quantized transform coefficients to generate inverse quantized transform coefficients. For example, operation 1406 may be applied via adaptive inverse quantize module 203.

Process 1400 may continue at operation 1408, "On each Decoded Block of Coefficients in a Coding (or Intra Predicted) Partition Perform Inverse Transform based on Transform Type and Block Size Data to Generate Decoded Prediction Error Partitions", where, on each decode block of transform coefficients in a coding (or intra predicted) partition, an inverse transform based on the transform type and block size data may be performed to generate decoded prediction error partitions. In some examples, the inverse transform may include an inverse fixed transform. In some examples, the inverse transform may include an inverse content adaptive transform. In such examples, performing the inverse content adaptive transform may include determining basis functions associated with the inverse content adaptive transform based on a neighboring block of decoded video data, as discussed herein. Any forward transform used for encoding as discussed herein

may be used for decoding using an associated inverse transform. In some examples, the inverse transform may be performed by adaptive inverse transform module 204. In some examples, generating the decoded prediction error partitions may also include assembling coding partitions via coding partitions assembler 205.

Process 1400 may continue at operation 1423 "Apply DD/DB Filter, Reconstruct Pixel Data, Assemble into a Picture", where deblock filtering (e.g., DD or DB filters) may be applied, pixel data may be reconstructed, and assembled into a picture. For example, after inverse scan, inverse transform, and assembling coding partitions, the prediction error data partitions may be added with a prediction partition to generate reconstructed prediction partitions, which may be assembled into tiles or super-fragments. The assembled tiles or super-fragments may be optionally processed via deblock filtering.

Process 1400 may continue at operation 1424 "Apply QR/LF Filter Save in Reference Picture Buffers", where quality restoration filtering (e.g., QR or LF filtering) may be applied, and the assembled picture may be saved in reference picture buffers. For example, in addition to or in the alternative to the DD/DB filtering, the assembled tiles or super-fragments may be optionally processed via quality restoration filtering and assembled to generate a picture. The picture may be saved in decoded picture buffer 119 as a reference picture for prediction of other (e.g., following) pictures.

Process 1400 may continue at operation 1425, "Apply AP/AM Filter, Use Decoded Modifying Characteristics (e.g., mop, syp) to Generate Modified References for Prediction and Use Motion Vectors and Reference Info, Predicted Partition Info, and Modified References to Generate Predicted Partition", where modified references for prediction may be generated and predicted partitions may be generated as well, and where adaptive motion filtering or adaptive precision filtering (e.g., AP/AM Filter) may be applied. For example, where modified references for prediction may be generated based at least in part on decoded modifying characteristics (e.g., mop, syp) and predicted partitions may be generated based at least in part on motion vectors and reference information, predicted partition information, and modified references. Additionally, adaptive motion filtering or adaptive precision filtering may be applied at this point in the process.

Process 1400 may continue at operation 1429 "Optionally Apply EP Filter and/or Optionally apply FI/FP Filter", where enhanced predicted partition (e.g., EP Filtering) or FI/FP Filtering (e.g., fusion filtering or fusion improvement filtering) may be optionally applied. In some examples, a decision may be made regarding whether to utilize some

form or FI/FP Filter (fusion improvement filtering/fusion filtering) or not to use FI/FP Filtering. When some form or FI/FP Filter (e.g., fusion filtering or fusion improvement filtering) is to be applied to the selected predicted partition the selected predicted partition and a second selected predicted partition may be assembled to generate at least a portion of an assembled picture. FI/FP Filtering may be applied to filter the portion of the assembled picture. FI/FP Filtering parameters (e.g., filtering parameters or fusion improvement filtering parameters) associated with the FI/FP Filtering may be generated and sent to the entropy coder subsystem.

In implementations where both EP Filtering or FI/FP Filtering are available, an indicator may be received from the encoder system that indicates to the decoder system whether to use the enhanced predicted partition (e.g., EP Filtering) or the predicted partition data as the selected predicted partition for the prediction partition.

Process 1400 may continue at operation 1430, "Add Prediction Partition to the Decoded Prediction Error Data Partition to Generate a Reconstructed Partition", where a prediction partition my be added to the decoded prediction error data partition to generate a reconstructed prediction partition. For example, the decoded prediction error data partition may be added to the associated prediction partition via adder 206.

Process 1400 may continue at operation 1432, "Assemble Reconstructed Partitions to Generate a Tile or Super-Fragment", where the reconstructed prediction partitions may be assembled to generate tiles or super-fragments. For example, the reconstructed prediction partitions may be assembled to generate tiles or super-fragments via prediction partitions assembler module 207.

Process 1400 may continue at operation 1434, "Assemble Tiles or Super-Fragments of a Picture to Generate a Full Decoded Picture", where the tiles or super-fragments of a picture may be assembled to generate a full decoded picture. For example, after optional deblock filtering and/or quality restoration filtering, tiles or super-fragments may be assembled to generate a full decoded picture, which may be stored via decoded picture buffer 210 and/or transmitted for presentment via a display device after processing via adaptive picture re-organizer module 217 and content post-restorer module 218.

FIGS. 15(A), 15(B), and 15(C) provide an illustrative diagram of an example video coding system 1600 and video coding process 1500 in operation, arranged in accordance with at least some implementations of the present disclosure. In the illustrated

implementation, process 1500 may include one or more operations, functions or actions as illustrated by one or more of actions 1501 through 1580. By way of non-limiting example, process 1500 will be described herein with reference to example video coding system 1600 including encoder 100 of FIG. 1 and decoder 200 of FIG. 2, as is discussed further herein below with respect to FIG. 16. In various examples, process 1500 may be undertaken by a system including both an encoder and decoder or by separate systems with one system employing an encoder (and optionally a decoder) and another system employing a decoder (and optionally an encoder). It is also noted, as discussed above, that an encoder may include a local decode loop employing a local decoder as a part of the encoder system.

In the illustrated implementation, video coding system 1600 may include logic circuitry 1650, the like, and/or combinations thereof. For example, logic circuitry 1650 may include encoder system 100 of FIG. 1 and/or decoder system 200 of FIG. 2 and may include any modules as discussed with respect to any of the encoder systems or subsystems described herein and/or decoder systems or subsystems described herein. Although video coding system 1600, as shown in FIGS. 15(A)-(C) may include one particular set of blocks or actions associated with particular modules, these blocks or actions may be associated with different modules than the particular modules illustrated here. Although process 1500, as illustrated, is directed to encoding and decoding, the concepts and/or operations described may be applied to encoding and/or decoding separately, and, more generally, to video coding.

Process 1500 may begin at operation 1501, "Receive Input Video Frames of a Video Sequence", where input video frames of a video sequence may be received via encoder 100 for example.

Process 1500 may continue at operation 1502, "Associate a Picture Type with each Video Frame in a Group of Pictures", where a picture type may be associated with each video frame in a group of pictures via content pre-analyzer module 102 for example. For example, the picture type may be F/B-picture , P-picture, or I-picture, or the like. In some examples, a video sequence may include groups of pictures and the processing described herein (e.g., operations 1503 through 1511) may be performed on a frame or picture of a group of pictures and the processing may be repeated for all frames or pictures of a group and then repeated for all groups of pictures in a video sequence.

Process 1500 may continue at operation 1503, "Divide a Picture into Tiles and/or Super-fragments and Potential Prediction Partitionings", where a picture may be divided

into tiles or super-fragments and potential prediction partitions via prediction partitions generator 105 for example.

Process 1500 may continue at operation 1504, "For Each Potential Prediction Partitioning, Perform Prediction(s) and Determine Prediction Parameters", where, for each potential prediction partitionings, prediction(s) may be performed and prediction parameters may be determined. For example, a range of potential prediction partitionings (each having various prediction partitions) may be generated and the associated prediction(s) and prediction parameters may be determined. For example, the prediction(s) may include prediction(s) using characteristics and motion based multi-reference predictions or intra-predictions.

As discussed, in some examples, inter-prediction may be performed. In some examples, up to 4 decoded past and/or future pictures and several morphing/synthesis predictions may be used to generate a large number of reference types (e.g., reference pictures). For instance in 'inter' mode, up to 9 reference types may be supported in P-pictures, and up to 10 reference types may be supported for F/B-pictures. Further, 'multi' mode may provide a type of inter prediction mode in which instead of 1 reference picture, 2 reference pictures may be used and P- and F/B-pictures respectively may allow 3, and up to 8 reference types. For example, prediction may be based on a previously decoded frame generated using at least one of a morphing technique or a synthesizing technique. In such examples, and the bitstream (discussed below with respect to operation 1512) may include a frame reference, morphing parameters, or synthesizing parameters associated with the prediction partition.

Process 1500 may continue at operation 1505, "For Each Potential Prediction Partitioning, Determine Potential Prediction Error", where, for each potential prediction partitioning, a potential prediction error may be determined. For example, for each prediction partitioning (and associated prediction partitions, prediction(s), and prediction parameters), a prediction error may be determined. For example, determining the potential prediction error may include differencing original pixels (e.g., original pixel data of a prediction partition) with prediction pixels. In some examples, the associated prediction parameters may be stored. As discussed, in some examples, the prediction error data partition may include prediction error data generated based at least in part on a previously decoded frame generated using at least one of a morphing technique or a synthesizing technique.

Process 1500 may continue at operation 1506, "Select Prediction Partitioning and Prediction Type and Save Parameters", where a prediction partitioning and prediction type may be selected and the associated parameters may be saved. In some examples, the potential prediction partitioning with a minimum prediction error may be selected. In some examples, the potential prediction partitioning may be selected based on a rate distortion optimization (RDO).

Process 1500 may continue at operation 1507, "Perform Fixed or Content Adaptive Transforms with Various Block Sizes on Various Potential Coding Partitionings of Partition Prediction Error Data", where fixed or content adaptive transforms with various block sizes may be performed on various potential coding partitionings of partition prediction error data. For example, partition prediction error data may be partitioned to generate a plurality of coding partitions. For example, the partition prediction error data may be partitioned by a bi-tree coding partitioner module or a k-d tree coding partitioner module of coding partitions generator module 107 as discussed herein. In some examples, partition prediction error data associated with an F/B- or P-picture may be partitioned by a bi-tree coding partitioner module. In some examples, video data associated with an I-picture (e.g., tiles or super-fragments in some examples) may be partitioned by a k-d tree coding partitioner module. In some examples, a coding partitioner module may be chosen or selected via a switch or switches. For example, the partitions may be generated by coding partitions generator module 107.

Process 1500 may continue at operation 1508, "Determine the Best Coding Partitioning, Transform Block Sizes, and Actual Transform", where the best coding partitioning, transform block sizes, and actual transforms may be determined. For example, various coding partitionings (e.g., having various coding partitions) may be evaluated based on RDO or another basis to determine a selected coding partitioning (which may also include further division of coding partitions into transform blocks when coding partitions to not match a transform block size as discussed). For example, the actual transform (or selected transform) may include any content adaptive transform or fixed transform performed on coding partition or block sizes as described herein.

Process 1500 may continue at operation 1509, "Quantize and Scan Transform Coefficients", where transform coefficients associated with coding partitions (and/or transform blocks) may be quantized and scanned in preparation for entropy coding.

Process 1500 may continue at operation 1511, "Entropy Encode Data associated with Each Tile or Super-fragment", where data associated with each tile or super-fragment may be entropy encoded. For example, data associated with each tile or super-fragment of each picture of each group of pictures of each video sequence may be entropy encoded. The entropy encoded data may include the prediction partitioning, prediction parameters, the selected coding partitioning, the selected characteristics data, motion vector data, quantized transform coefficients, filter parameters, selection data (such as mode selection data), and indictors.

Process 1500 may continue at operation 1512, "Generate Bitstream" where a bitstream may be generated based on the entropy encoded data. As discussed, in some examples, the bitstream may include a frame or picture reference, morphing parameters, or synthesizing parameters associated with a prediction partition.

Process 1500 may continue at operation 1513, "Transmit Bitstream", where the bitstream may be transmitted. For example, video coding system 2400 may transmit output bitstream 111, bitstream 2100, or the like via an antenna 2402 (please refer to FIG. 34).

Process 1500 may continue at operation 1520, "Reconstruct Pixel Data, Assemble into a Picture, and Save in Reference Picture Buffers", where pixel data may be reconstructed, assembled into a picture, and saved in reference picture buffers. For example, after a local decode loop (e.g., including inverse scan, inverse transform, and assembling coding partitions), prediction error data partitions may be generated. The prediction error data partitions may be added with a prediction partition to generate reconstructed prediction partitions, which may be assembled into tiles or super-fragments. The assembled tiles or super-fragments may be optionally processed via deblock filtering and/or quality restoration filtering and assembled to generate a picture. The picture may be saved in decoded picture buffer 119 as a reference picture for prediction of other (e.g., following) pictures.

Process 1500 may continue at operation 1523 "Apply DD/DB Filter, Reconstruct Pixel Data, Assemble into a Picture", where deblock filtering (e.g., DD or DB filters) may be applied, pixel data may be reconstructed, and assembled into a picture. For example, after a local decode loop (e.g., including inverse scan, inverse transform, and assembling coding partitions), prediction error data partitions may be generated. The prediction error data partitions may be added with a prediction partition to generate reconstructed prediction partitions, which may be assembled into tiles or super-fragments. The

assembled tiles or super-fragments may be optionally processed via deblock filtering and/or quality restoration filtering and assembled to generate a picture.

Process 1500 may continue at operation 1524 "Apply QR/LF Filter Save in Reference Picture Buffers", where quality restoration filtering (e.g., QR or LF filtering) may be applied, and the assembled picture may be saved in reference picture buffers. For example, in addition to or in the alternative to the DD/DB filtering, the assembled tiles or super-fragments may be optionally processed via quality restoration filtering and assembled to generate a picture. The picture may be saved in decoded picture buffer 119 as a reference picture for prediction of other (e.g., following) pictures.

Process 1500 may continue at operation 1525, "Generate Modifying Characteristic Parameters", where, modified characteristic parameters may be generated. For example, a second modified prediction reference picture and second modifying characteristic parameters associated with the second modified prediction reference picture may be generated based at least in part on the second decoded prediction reference picture, where the second modified reference picture may be of a different type than the first modified reference picture.

Process 1500 may continue at operation 1526, "Generate Modified Prediction Reference Pictures", where modified prediction reference pictures may be generated, for example, a first modified prediction reference picture and first modifying characteristic parameters associated with the first modified prediction reference picture may be generated based at least in part on the first decoded prediction reference picture.

Process 1500 may continue at operation 1527, "Generate Motion Data", where, motion estimation data may be generated. For example, motion data associated with a prediction partition of a current picture may be generated based at least in part on one of the first modified prediction reference picture or the second modified prediction reference picture.

Process 1500 may continue at operation 1528, "Apply AP/AM Filter Perform Motion Compensation", where, motion compensation may be performed. For example, motion compensation may be performed based at least in part on the motion data and at least one of the first modified prediction reference picture or the second modified prediction reference picture to generate prediction partition data for the prediction partition and adaptive motion filtering or adaptive precision filtering (e.g., AP/AM Filter) may be

applied. Process 1500 may feed this information back to operation 1504 where each decoded prediction error partition (e.g., including zero prediction error partitions) may be added to the corresponding prediction partition to generate a reconstructed prediction partition. Additionally, adaptive motion filtering or adaptive precision filtering may be applied at this point in the process.

Process 1500 may continue at operation 1529 "Optionally Apply EP", where enhanced predicted partition (e.g., EP Filtering) may be optionally applied. In some examples, where both EP Filtering or FI/FP Filtering are available, an indicator may be generated that indicates to the decoder system whether to use the enhanced predicted partition (e.g., EP Filtering) or the predicted partition data as the selected predicted partition for the prediction partition.

Process 1500 may continue at operation 1530 "Optionally apply FI/FP Filter", where FI/FP Filtering (e.g., fusion filtering or fusion improvement filtering) may be optionally applied. In some examples, a decision may be made regarding whether to utilize some form or FI/FP Filter (fusion improvement filtering/fusion filtering) or not to use FI/FP Filtering. When some form or FI/FP Filter (e.g., fusion filtering or fusion improvement filtering) is to be applied to the selected predicted partition the selected predicted partition and a second selected predicted partition may be assembled to generate at least a portion of an assembled picture. FI/FP Filtering may be applied to filter the portion of the assembled picture. FI/FP Filtering parameters (e.g., filtering parameters or fusion improvement filtering parameters) associated with the FI/FP Filtering may be generated and sent to the entropy coder subsystem.

Operations 1501 through 1530 may provide for video encoding and bitstream transmission techniques, which may be employed by an encoder system as discussed herein. The following operations, operations 1554 through 1568 may provide for video decoding and video display techniques, which may be employed by a decoder system as discussed herein.

Process 1500 may continue at operation 1554, "Receive Bitstream", where the bitstream may be received. For example, input bitstream 201, bitstream 2100, or the like may be received via decoder 200. In some examples, the bitstream may include data associated with a coding partition, one or more indicators, and/or data defining coding partition(s) as discussed above. In some examples, the bitstream may include the prediction partitioning, prediction parameters, the selected coding partitioning, the selected

characteristics data, motion vector data, quantized transform coefficients, filter parameters, selection data (such as mode selection data), and indictors.

Process 1500 may continue at operation 1555, "Decode Bitstream", where the received bitstream may be decoded via adaptive entropy decoder module 202 for example. For example, received bitstream may be entropy decoded to determine the prediction partitioning, prediction parameters, the selected coding partitioning, the selected characteristics data, motion vector data, quantized transform coefficients, filter parameters, selection data (such as mode selection data), and indictors.

Process 1500 may continue at operation 1556, "Perform Inverse Scan and Inverse Quantization on Each Block of Each Coding Partition", where an inverse scan and inverse quantization may be performed on each block of each coding partition for the prediction partition being processed. For example, the inverse scan and inverse quantization may be performed via adaptive inverse quantize module 203.

Process 1500 may continue at operation 1557, "Perform Fixed or Content Adaptive Inverse Transform to Decode Transform Coefficients to Determine Decoded Prediction Error Data Partitions", where a fixed or content adaptive inverse transform may be performed to decode transform coefficients to determine decoded prediction error data partitions. For example, the inverse transform may include an inverse content adaptive transform such as a hybrid parametric Haar inverse transform such that the hybrid parametric Haar inverse transform may include a parametric Haar inverse transform in a direction of the parametric transform direction and a discrete cosine inverse transform in a direction orthogonal to the parametric transform direction. In some examples, the fixed inverse transform may include a discrete cosine inverse transform or a discrete cosine inverse transform approximator. For example, the fixed or content adaptive transform may be performed via adaptive inverse transform module 204. As discussed, the content adaptive inverse transform may be based on other previously decoded data, such as, for example, decoded neighboring partitions or blocks. In some examples, generating the decoded prediction error data partitions may include assembling decoded coding partitions via coding partitions assembler module 205.

Process 1500 may continue at operation 1558, "Generate Prediction Pixel Data for Each Prediction Partition", where prediction pixel data may be generated for each prediction partition. For example, prediction pixel data may be generated using the

selected prediction type (e.g., based on characteristics and motion, or intra-, or other types) and associated prediction parameters.

Process 1500 may continue at operation 1559, "Add to Each Decoded Prediction Error Partition the Corresponding Prediction Partition to Generate Reconstructed Prediction Partition", where each decoded prediction error partition (e.g., including zero prediction error partitions) may be added to the corresponding prediction partition to generated a reconstructed prediction partition. For example, prediction partitions may be generated via the decode loop illustrated in FIG. 2 and added via adder 206 to decoded prediction error partitions.

Process 1500 may continue at operation 1560, "Assemble Reconstructed Prediction Partitions to Generate Decoded Tiles or Super-fragments", where reconstructed prediction partitions may be assembled to generate decoded tiles or super-fragments. For example, prediction partitions may be assembled to generate decoded tiles or super-fragments via prediction partitions assembler module 207.

Process 1500 may continue at operation 1561, "Apply Deblock Filtering and/or QR Filtering to Generate Final Decoded Tiles or Super-fragments", where optional deblock filtering and/or quality restoration filtering may be applied to the decoded tiles or super-fragments to generate final decoded tiles or super-fragments. For example, optional deblock filtering may be applied via deblock filtering module 208 and/or optional quality restoration filtering may be applied via quality restoration filtering module 209.

Process 1500 may continue at operation 1562, "Assemble Decoded Tiles or Super-fragments to Generate a Decoded Video Picture, and Save in Reference Picture Buffers", where decoded (or final decoded) tiles or super-fragments may be assembled to generate a decoded video picture, and the decoded video picture may be saved in reference picture buffers (e.g., decoded picture buffer 210) for use in future prediction.

Process 1500 may continue at operation 1563, "Transmit Decoded Video Frames for Presentment via a Display Device", where decoded video frames may be transmitted for presentment via a display device. For example, decoded video pictures may be further processed via adaptive picture re-organizer 217 and content post restorer module 218 and transmitted to a display device as video frames of display video 219 for presentment to a user. For example, the video frame(s) may be transmitted to a display device 2405 (as shown in FIG. 34) for presentment.

Process 1500 may continue at operation 1573 "Apply DD/DB Filter, Reconstruct Pixel Data, Assemble into a Picture", where deblock filtering (e.g., DD or DB filters) may be applied, pixel data may be reconstructed, and assembled into a picture. For example, after inverse scan, inverse transform, and assembling coding partitions, the prediction error data partitions may be added with a prediction partition to generate reconstructed prediction partitions, which may be assembled into tiles or super-fragments. The assembled tiles or super-fragments may be optionally processed via deblock filtering.

Process 1500 may continue at operation 1574 "Apply QR/LF Filter Save in Reference Picture Buffers", where quality restoration filtering (e.g., QR or LF filtering) may be applied, and the assembled picture may be saved in reference picture buffers. For example, in addition to or in the alternative to the DD/DB filtering, the assembled tiles or super-fragments may be optionally processed via quality restoration filtering and assembled to generate a picture. The picture may be saved in a picture buffer as a reference picture for prediction of other (e.g., following) pictures.

Process 1500 may continue at operation 1576, "Generate Modified Prediction Reference Pictures", where modified prediction reference pictures may be generated, for example, at least a portion of a third modified prediction reference picture may be generated based at least in part on the third modifying characteristic parameters. Similarly, at least a portion a fourth modified prediction reference picture may be generated based at least in part on the second modifying characteristic parameters associated.

Process 1500 may continue at operation 1577, "Generate Motion Data", where, motion estimation data may be generated. For example, motion data associated with a prediction partition of a current picture may be generated based at least in part on one of the third modified prediction reference picture or the third modified prediction reference picture.

Process 1500 may continue at operation 1578, "Apply AP/AM Filter and Perform Motion Compensation", where, motion compensation may be performed and where adaptive motion filtering or adaptive precision filtering (e.g., AP/AM Filter) may be applied. For example, motion compensation may be performed based at least in part on the motion data and at least one of the third modified prediction reference picture or the fourth modified prediction reference picture to generate prediction partition data for the prediction partition. Process 1300 may feed this information back to operation 1559 where each decoded prediction error partition (e.g., including zero prediction error partitions)

may be added to the corresponding prediction partition to generate a reconstructed prediction partition. Additionally, adaptive motion filtering or adaptive precision filtering may be applied at this point in the process.

Process 1500 may continue at operation 1579 "Optionally Apply EP Filter", where enhanced predicted partition (e.g., EP Filtering) may be optionally applied. In some examples, where both EP Filtering or FI/FP Filtering are available, an indicator may be received from the encoder system that indicates to the decoder system whether to use the enhanced predicted partition (e.g., EP Filtering) or the predicted partition data as the selected predicted partition for the prediction partition.

Process 1500 may continue at operation 1580 "Optionally apply FI/FP Filter", where FI/FP Filtering (e.g., fusion filtering or fusion improvement filtering) may be optionally applied. In some examples, a decision may be made regarding whether to utilize some form or FI/FP Filter (fusion improvement filtering/fusion filtering) or not to use FI/FP Filtering. When some form or FI/FP Filter (e.g., fusion filtering or fusion improvement filtering) is to be applied to the selected predicted partition the selected predicted partition and a second selected predicted partition may be assembled to generate at least a portion of an assembled picture. FI/FP Filtering may be applied to filter the portion of the assembled picture. FI/FP Filtering parameters (e.g., filtering parameters or fusion improvement filtering parameters) associated with the FI/FP Filtering may be generated and sent to the entropy coder subsystem.

Process 1500 may be implemented via any of the coder systems as discussed herein. Further, process 1500 may be repeated either in serial or in parallel on any number of instantiations of video data such as prediction error data partitions, original data partitions, or wavelet data or the like.

While implementation of the example processes herein may include the undertaking of all operations shown in the order illustrated, the present disclosure is not limited in this regard and, in various examples, implementation of the example processes herein may include the undertaking of only a subset of the operations shown and/or in a different order than illustrated.

FIG. 16 is an illustrative diagram of example video coding system 1600, arranged in accordance with at least some implementations of the present disclosure. In the illustrated implementation, video coding system 1600 may include imaging device(s) 1601, video

encoder 100 and/or a video encoder implemented via logic circuitry 1650 of processing unit(s) 1620, video decoder 200 and/or a video decoder implemented via logic circuitry 1650 of processing unit(s) 1620, an antenna 1602, one or more processor(s) 1603, one or more memory store(s) 2004, and/or a display device 1605.

As illustrated, imaging device(s) 1601, antenna 1602, processing unit(s) 1620, logic circuitry 1650, video encoder 100, video decoder 200, processor(s) 1603, memory store(s) 1604, and/or display device 1605 may be capable of communication with one another. As discussed, although illustrated with both video encoder 100 and video decoder 200, video coding system 1600 may include only video encoder 100 or only video decoder 200 in various examples.

As shown, in some examples, video coding system 1600 may include antenna 1602. Antenna 1602 may be configured to transmit or receive an encoded bitstream of video data, for example. Further, in some examples, video coding system 1600 may include display device 1605. Display device 1605 may be configured to present video data. As shown, in some example, logic circuitry 1650 may be implemented via processing unit(s) 1620. Processing unit(s) 1620 may include application-specific integrated circuit (ASIC) logic, graphics processor(s), general purpose processor(s), or the like. Video coding system 1600 also may include optional processor(s) 1603, which may similarly include application-specific integrated circuit (ASIC) logic, graphics processor(s), general purpose processor(s), or the like. In some examples, logic circuitry 1650 may be implemented via hardware or video coding dedicated hardware or the like, and processor(s) 1603 may implemented general purpose software or operating systems or the like. In addition, memory stores 1604 may be any type of memory such as volatile memory (e.g., Static Random Access Memory (SRAM), Dynamic Random Access Memory (DRAM), etc.) or non-volatile memory (e.g., flash memory, etc.), and so forth. In a non-limiting example, memory stores 1604 may be implemented by cache memory. In some examples, logic circuitry 1650 may access memory stores 1604 (for implementation of an image buffer for example). In other examples, logic circuitry 1650 and/or processing unit(s) 1620 may include memory stores (e.g., cache or the like) for the implementation of an image buffer or the like.

In some examples, video encoder 100 implemented via logic circuitry may include an image buffer (e.g., via either processing unit(s) 1620 or memory store(s) 1604)) and a graphics processing unit (e.g., via processing unit(s) 1620). The graphics processing unit

may be communicatively coupled to the image buffer. The graphics processing unit may include video encoder 100 as implemented via logic circuitry 1650 to embody the various modules as discussed with respect to FIG. 1 and FIGS. 3 and 5. For example, the graphics processing unit may include entropy encoder logic circuitry, and so on. The logic circuitry may be configured to perform the various operations as discussed herein. For example, the entropy encoder logic circuitry may be configured to receive first video data and second video data of different types for entropy encoding, determine a first entropy encoding technique for the first video data based at least in part on a parameter associated with the first video data such that the first entropy encoding technique comprises at least one of an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique, entropy encode the first video data using the first encoding technique to generate first compressed video data and the second video data using a second encoding technique to generate second compressed video data, and assemble the first compressed video data and the second compressed video data to generate an output bitstream. Video decoder 200 may be implemented in a similar manner.

In some examples, antenna 1602 of video coding system 1600 may be configured to receive an entropy encoded bitstream of video data. As discussed, the bitstream may include two or more compressed video data types. Video coding system 1600 may also include video decoder 200 coupled to antenna 1602 and configured to decode the encoded bitstream. For example, video decoder 200 may be configured to disassemble the entropy encoded bitstream to determine first compressed video data and second compressed video data, determine a first entropy decoding technique for the first compressed video data, such that the first entropy decoding technique comprises at least one of an adaptive symbol-run variable length coding technique or an adaptive proxy variable length coding technique, entropy decode the first compressed video data based on the first entropy decoding technique to generate first video data and the second compressed video data based on a second entropy decoding technique to generate second video data, and decode the first video data and the second video data to generate a video frame.

In embodiments, features described herein may be undertaken in response to instructions provided by one or more computer program products. Such program products may include signal bearing media providing instructions that, when executed by, for example, a processor, may provide the functionality described herein. The computer program products may be provided in any form of one or more machine-readable media. Thus, for example, a processor including one or more processor core(s) may undertake one

or more features described herein in response to program code and/or instructions or instruction sets conveyed to the processor by one or more machine-readable media. In general, a machine-readable medium may convey software in the form of program code and/or instructions or instruction sets that may cause any of the devices and/or systems described herein to implement at least portions of the features described herein.

FIG. 17 is an illustrative diagram of an example system 1700, arranged in accordance with at least some implementations of the present disclosure. In various implementations, system 1700 may be a media system although system 1700 is not limited to this context. For example, system 1700 may be incorporated into a personal computer (PC), laptop computer, ultra-laptop computer, tablet, touch pad, portable computer, handheld computer, palmtop computer, personal digital assistant (PDA), cellular telephone, combination cellular telephone/PDA, television, smart device (e.g., smart phone, smart tablet or smart television), mobile internet device (MID), messaging device, data communication device, cameras (e.g. point-and-shoot cameras, super-zoom cameras, digital single-lens reflex (DSLR) cameras), and so forth.

In various implementations, system 1700 includes a platform 1702 coupled to a display 1720. Platform 1702 may receive content from a content device such as content services device(s) 1730 or content delivery device(s) 1740 or other similar content sources. A navigation controller 1750 including one or more navigation features may be used to interact with, for example, platform 1702 and/or display 1720. Each of these components is described in greater detail below.

In various implementations, platform 1702 may include any combination of a chipset 1705, processor 1710, memory 1712, antenna 1713, storage 1714, graphics subsystem 1715, applications 1716 and/or radio 1718. Chipset 1705 may provide intercommunication among processor 1710, memory 1712, storage 1714, graphics subsystem 1715, applications 1716 and/or radio 1718. For example, chipset 1705 may include a storage adapter (not depicted) capable of providing intercommunication with storage 1714.

Processor 1710 may be implemented as a Complex Instruction Set Computer (CISC) or Reduced Instruction Set Computer (RISC) processors, x86 instruction set compatible processors, multi-core, or any other microprocessor or central processing unit (CPU). In various implementations, processor 1710 may be dual-core processor(s), dual-core mobile processor(s), and so forth.

Memory 1712 may be implemented as a volatile memory device such as, but not limited to, a Random Access Memory (RAM), Dynamic Random Access Memory (DRAM), or Static RAM (SRAM).

Storage 1714 may be implemented as a non-volatile storage device such as, but not limited to, a magnetic disk drive, optical disk drive, tape drive, an internal storage device, an attached storage device, flash memory, battery backed-up SDRAM (synchronous DRAM), and/or a network accessible storage device. In various implementations, storage 1714 may include technology to increase the storage performance enhanced protection for valuable digital media when multiple hard drives are included, for example.

Graphics subsystem 1715 may perform processing of images such as still or video for display. Graphics subsystem 1715 may be a graphics processing unit (GPU) or a visual processing unit (VPU), for example. An analog or digital interface may be used to communicatively couple graphics subsystem 1715 and display 1720. For example, the interface may be any of a High-Definition Multimedia Interface, DisplayPort, wireless HDMI, and/or wireless HD compliant techniques. Graphics subsystem 1715 may be integrated into processor 1710 or chipset 1705. In some implementations, graphics subsystem 1715 may be a stand-alone device communicatively coupled to chipset 1705.

The graphics and/or video processing techniques described herein may be implemented in various hardware architectures. For example, graphics and/or video functionality may be integrated within a chipset. Alternatively, a discrete graphics and/or video processor may be used. As still another implementation, the graphics and/or video functions may be provided by a general purpose processor, including a multi-core processor. In further embodiments, the functions may be implemented in a consumer electronics device.

Radio 1718 may include one or more radios capable of transmitting and receiving signals using various suitable wireless communications techniques. Such techniques may involve communications across one or more wireless networks. Example wireless networks include (but are not limited to) wireless local area networks (WLANs), wireless personal area networks (WPANs), wireless metropolitan area network (WMANs), cellular networks, and satellite networks. In communicating across such networks, radio 1718 may operate in accordance with one or more applicable standards in any version.

In various implementations, display 1720 may include any television type monitor or display. Display 1720 may include, for example, a computer display screen, touch screen display, video monitor, television-like device, and/or a television. Display 1720 may be digital and/or analog. In various implementations, display 1720 may be a holographic display. Also, display 1720 may be a transparent surface that may receive a visual projection. Such projections may convey various forms of information, images, and/or objects. For example, such projections may be a visual overlay for a mobile augmented reality (MAR) application. Under the control of one or more software applications 1716, platform 1702 may display user interface 1722 on display 1720.

In various implementations, content services device(s) 1730 may be hosted by any national, international and/or independent service and thus accessible to platform 1702 via the Internet, for example. Content services device(s) 1730 may be coupled to platform 1702 and/or to display 1720. Platform 1702 and/or content services device(s) 1730 may be coupled to a network 1760 to communicate (e.g., send and/or receive) media information to and from network 1760. Content delivery device(s) 1740 also may be coupled to platform 1702 and/or to display 1720.

In various implementations, content services device(s) 1730 may include a cable television box, personal computer, network, telephone, Internet enabled devices or appliance capable of delivering digital information and/or content, and any other similar device capable of unidirectionally or bidirectionally communicating content between content providers and platform 1702 and/display 1720, via network 1760 or directly. It will be appreciated that the content may be communicated unidirectionally and/or bidirectionally to and from any one of the components in system 1700 and a content provider via network 1760. Examples of content may include any media information including, for example, video, music, medical and gaming information, and so forth.

Content services device(s) 1730 may receive content such as cable television programming including media information, digital information, and/or other content. Examples of content providers may include any cable or satellite television or radio or Internet content providers. The provided examples are not meant to limit implementations in accordance with the present disclosure in any way.

In various implementations, platform 1702 may receive control signals from navigation controller 1750 having one or more navigation features. The navigation features of controller 1750 may be used to interact with user interface 1722, for example. In various

embodiments, navigation controller 1750 may be a pointing device that may be a computer hardware component (specifically, a human interface device) that allows a user to input spatial (e.g., continuous and multi-dimensional) data into a computer. Many systems such as graphical user interfaces (GUI), and televisions and monitors allow the user to control and provide data to the computer or television using physical gestures.

Movements of the navigation features of controller 1750 may be replicated on a display (e.g., display 1720) by movements of a pointer, cursor, focus ring, or other visual indicators displayed on the display. For example, under the control of software applications 1716, the navigation features located on navigation controller 1750 may be mapped to virtual navigation features displayed on user interface 1722, for example. In various embodiments, controller 1750 may not be a separate component but may be integrated into platform 1702 and/or display 1720. The present disclosure, however, is not limited to the elements or in the context shown or described herein.

In various implementations, drivers (not shown) may include technology to enable users to instantly turn on and off platform 1702 like a television with the touch of a button after initial boot-up, when enabled, for example. Program logic may allow platform 1702 to stream content to media adaptors or other content services device(s) 1730 or content delivery device(s) 1740 even when the platform is turned "off." In addition, chipset 1705 may include hardware and/or software support for 5.1 surround sound audio and/or high definition 7.1 surround sound audio, for example. Drivers may include a graphics driver for integrated graphics platforms. In various embodiments, the graphics driver may comprise a peripheral component interconnect (PCI) Express graphics card.

In various implementations, any one or more of the components shown in system 1700 may be integrated. For example, platform 1702 and content services device(s) 1730 may be integrated, or platform 1702 and content delivery device(s) 1740 may be integrated, or platform 1702, content services device(s) 1730, and content delivery device(s) 1740 may be integrated, for example. In various embodiments, platform 1702 and display 1720 may be an integrated unit. Display 1720 and content service device(s) 1730 may be integrated, or display 1720 and content delivery device(s) 1740 may be integrated, for example. These examples are not meant to limit the present disclosure.

In various embodiments, system 1700 may be implemented as a wireless system, a wired system, or a combination of both. When implemented as a wireless system, system 1700 may include components and interfaces suitable for communicating over a wireless

shared media, such as one or more antennas, transmitters, receivers, transceivers, amplifiers, filters, control logic, and so forth. An example of wireless shared media may include portions of a wireless spectrum, such as the RF spectrum and so forth. When implemented as a wired system, system 1700 may include components and interfaces suitable for communicating over wired communications media, such as input/output (I/O) adapters, physical connectors to connect the I/O adapter with a corresponding wired communications medium, a network interface card (NIC), disc controller, video controller, audio controller, and the like. Examples of wired communications media may include a wire, cable, metal leads, printed circuit board (PCB), backplane, switch fabric, semiconductor material, twisted-pair wire, co-axial cable, fiber optics, and so forth.

Platform 1702 may establish one or more logical or physical channels to communicate information. The information may include media information and control information. Media information may refer to any data representing content meant for a user. Examples of content may include, for example, data from a voice conversation, videoconference, streaming video, electronic mail ("email") message, voice mail message, alphanumeric symbols, graphics, image, video, text and so forth. Data from a voice conversation may be, for example, speech information, silence periods, background noise, comfort noise, tones and so forth. Control information may refer to any data representing commands, instructions or control words meant for an automated system. For example, control information may be used to route media information through a system, or instruct a node to process the media information in a predetermined manner. The embodiments, however, are not limited to the elements or in the context shown or described in FIG. 17.

As described above, system 1700 may be embodied in varying physical styles or form factors. FIG. 18 illustrates implementations of a small form factor device 1800 in which system 1800 may be embodied. In various embodiments, for example, device 1800 may be implemented as a mobile computing device a having wireless capabilities. A mobile computing device may refer to any device having a processing system and a mobile power source or supply, such as one or more batteries, for example.

As described above, examples of a mobile computing device may include a personal computer (PC), laptop computer, ultra-laptop computer, tablet, touch pad, portable computer, handheld computer, palmtop computer, personal digital assistant (PDA), cellular telephone, combination cellular telephone/PDA, television, smart device (e.g., smart phone, smart tablet or smart television), mobile internet device (MID), messaging device,

data communication device, cameras (e.g. point-and-shoot cameras, super-zoom cameras, digital single-lens reflex (DSLR) cameras), and so forth.

Examples of a mobile computing device also may include computers that are arranged to be worn by a person, such as a wrist computer, finger computer, ring computer, eyeglass computer, belt-clip computer, arm-band computer, shoe computers, clothing computers, and other wearable computers. In various embodiments, for example, a mobile computing device may be implemented as a smart phone capable of executing computer applications, as well as voice communications and/or data communications. Although some embodiments may be described with a mobile computing device implemented as a smart phone by way of example, it may be appreciated that other embodiments may be implemented using other wireless mobile computing devices as well. The embodiments are not limited in this context.

As shown in FIG. 18, device 1800 may include a housing 1802, a display 1804, an input/output (I/O) device 1806, and an antenna 1808. Device 1800 also may include navigation features 1812. Display 1804 may include any suitable display unit for displaying information appropriate for a mobile computing device. I/O device 1806 may include any suitable I/O device for entering information into a mobile computing device. Examples for I/O device 1806 may include an alphanumeric keyboard, a numeric keypad, a touch pad, input keys, buttons, switches, rocker switches, microphones, speakers, voice recognition device and software, and so forth. Information also may be entered into device 1800 by way of microphone (not shown). Such information may be digitized by a voice recognition device (not shown). The embodiments are not limited in this context.

Various embodiments may be implemented using hardware elements, software elements, or a combination of both. Examples of hardware elements may include processors, microprocessors, circuits, circuit elements (e.g., transistors, resistors, capacitors, inductors, and so forth), integrated circuits, application specific integrated circuits (ASIC), programmable logic devices (PLD), digital signal processors (DSP), field programmable gate array (FPGA), logic gates, registers, semiconductor device, chips, microchips, chip sets, and so forth. Examples of software may include software components, programs, applications, computer programs, application programs, system programs, machine programs, operating system software, middleware, firmware, software modules, routines, subroutines, functions, methods, procedures, software interfaces, application program interfaces (API), instruction sets, computing code, computer code,

code segments, computer code segments, words, values, symbols, or any combination thereof. Determining whether an embodiment is implemented using hardware elements and/or software elements may vary in accordance with any number of factors, such as desired computational rate, power levels, heat tolerances, processing cycle budget, input data rates, output data rates, memory resources, data bus speeds and other design or performance constraints.

One or more aspects of at least one embodiment may be implemented by representative instructions stored on a machine-readable medium which represents various logic within the processor, which when read by a machine causes the machine to fabricate logic to perform the techniques described herein. Such representations, known as "IP cores" may be stored on a tangible, machine readable medium and supplied to various customers or manufacturing facilities to load into the fabrication machines that actually make the logic or processor.

As discussed, systems, apparatus, articles, and methods are described herein related to content adaptive entropy coding for video systems. In some examples, systems, apparatus, articles, and methods are described herein related to content adaptive entropy coding for data defining partitions for picture portions of video frames for video systems. For example, the partitions may be k-d tree partitions, bi-tree partitions, quad-tree partitions, structured codebook partitions, or the like of a tile, a largest coding unit, a super block, or the like of a video frame. The partitions may be prediction partitions or coding partitions for intra- or inter- coded picture partitions.

In some examples, multi-level patterns (e.g., patterns having 1 or more patterns) may be determined for picture portions. The multi-level patterns may have a only base pattern if the base pattern has all terminating portions (e.g., the portions of the base pattern match the input partitions) or the multi-level patterns may have a base pattern, a level one pattern or patterns for non-terminating portions of the base pattern, a level two pattern or patterns for non-terminating portions of the level one pattern, and so on until each pattern has all terminating portions at each level. The number of levels needed (base only, level one, level two, etc.) may vary by picture portion based on complexity, number of partitions, or the like. The multi-level pattern may include patterns (e.g., base, level one, and so on) determined from a number of available patterns. In some examples, the available patterns may be from a codebook of available patterns or the like. Further, termination bits (e.g., bits indicating whether or not a portion of a pattern terminates) may be determined for each

level of the multi-level pattern. The termination bits may be determined and gathered for a video frame, a portion (e.g., a slice) of a video frame, or for sequences of video frames. Entropy coded codewords may be determined for each pattern of each multi-level pattern and the termination bits may be entropy coded. The entropy coded codewords and termination bits may be written to a bitstream and transmitted to a receiving device such as a video decoder.

In some examples, the video decoder may receive and decode the bitstream. The video decoder may determine the base pattern and optional higher level patterns based on the entropy coded codewords. The video decoder may also determine the termination bits based on entropy decoding the entropy encoded termination bits (optionally based on an indicated variable length coding table or the like). The video decoder may reconstruct the partitions of the video portions based on the multi-level patterns and termination bits. Further, as discussed herein, the video decoder may decode the bitstream to determine transform coefficients for error data associated with the partitions, predict the partitions using techniques discussed herein, add the error data and the predicted partitions to generate final predicted partitions, and so on to generate decoded video frames for presentment via a display device.

As discussed herein, encoder module 402 (please refer to FIG. 6) may receive video data 402, which may include input data defining partitions of picture portions of a video frame or frames. Encoder module 402

As discussed, next generation video coding may use 3 picture types, I-, P-, and B/F-pictures. For example, I- and P-pictures, while they may have differences in details of prediction and coding, may be functionally similar. For prediction and coding, pictures in next generation video coding may be divided into picture partitions such as tiles. For common intermediate format (CIF) or lower resolutions, P-pictures may use 32x32 pixel tiles and I- and B/F-pictures may use 64x64 pixel tiles. For resolutions higher than CIF, all picture types may use 64x64 pixel tiles.

As discussed herein, each picture portion may be further divided into partitions for prediction (e.g., prediction partitions) and coding (e.g., coding partitions). The partitioning process may be adaptive to content (e.g., flat, detailed or mixed pictures) and/or coding bitrate (e.g., low, medium, high). The partitioning may also be based on Rate Distortion Optimization (RDO , which may incorporate content and bitrate adaptive properties. In various examples, the partitioning may be rectangular or arbitrary (e.g., having more

detailed shapes). For example, rectangular partitioning of a picture portion may be based bi-tree partitioning, quad-tree partitioning, k-d tree partitioning, pattern codebook partitioning, or the like. The techniques discussed herein with respect to coding partitions may be performed based on any of the described partitioning techniques discussed herein. Also as discussed, partitioning may be performed for prediction (e.g., prediction partitions) or coding (e.g., coding partitions). The discussed techniques may performed with respect to prediction or coding partitioning.

FIG. is an illustrative diagram of an example multi-level pattern for example partitions of a picture portion, arranged in accordance with at least some implementations of the present disclosure. As shown in FIG. , a picture portion 01 may include partitions 02 (which are not each labeled for the sake of clarity). As discussed, picture portion 01 may be a picture portion of a video frame. For example, picture portion 01 may be a tile, a largest coding unit, or a super block of a video frame. In some examples, picture portion 01 may be a partition of a picture portion of a video frame. In some examples, the video frame may be an I-picture picture portion 01 may be a tile of the video frame. In some I-picture examples, partitions 02 may be k-d tree partitions. In some examples, the video frame may be a P-picture or a B/F-picture and picture portion 01 may be a partition of a tile of the video frame. For example, the partition of the tile may be a prediction partition. In some P or B/F-picture examples, partitions 02 may be bi-tree partitions.

As discussed, picture portion 01 may be any size. In some examples, picture portion 01 may be 32x32 pixels, 64x64 pixels, or the like. Further, as discussed, partitions 02 may be any type of partitions. In some examples, partitions 02 may be k-d tree partitions, bi-tree partitions, quad-tree partitions, structured codebook partitions, or the like. Picture portion 01 and partitions 02 may be for either luma or chroma data. Further, picture portion 01 and partitions 02 as well as other picture portions and their respective partitions may be defined via input data such as, for example, partition data 412 or the like.

As shown in FIG. , a multi-level pattern 10 may be determined for picture portion 01. As discussed, multi-level patterns may include any number of levels (e.g., only a base level, a base level and one or more level one pattern(s), a base level, one or more level one pattern(s), and one or more level two pattern(s), and so on). In the example of FIG. , multi-level pattern 10 includes a base level pattern 11, four level one patterns 12, 13, 14, and 15, and seven level two patterns 21–27.

For example, base pattern (e.g., level 0 pattern) 11, level one patterns 12–15, and/or level two patterns 21–27 may be determined for picture portion 1901 and partitions 02 using a pattern based approach such as a hierarchical pattern representation. For example, base pattern 11 may be determined from a plurality of available patterns as the most suitable pattern (of the available patterns) that covers or represents as many partitions of partitions 02 as it can. In some examples, base pattern 11 may be the pattern of the available pattern that covers or represents a maximum number of partitions 02. As shown, one or more portions of base pattern 11 may be non-terminating such that the portion of base pattern 11 does not match (e.g., fully represent) partitions 02 of the corresponding portion of picture portion 01. For example, a representative portion 30 of picture portion 01 has a corresponding portion of picture portion 01 with seven partitions such that representative portion 30 of base pattern 11 does not fully represent the corresponding portion of picture portion 01 and a further level or levels of patterns are needed to fully represent the corresponding portion of picture portion 01.

In some examples, a base pattern may be fully terminating such that the base pattern fully represents the corresponding picture portion. In such examples, no further pattern levels may be needed and the base pattern may be entropy encoded using a entropy coded codeword that indicates the base pattern is fully terminating, as is discussed further herein.

As shown in FIG. , each non-terminating portion of base pattern 11 may be further represented by level one patterns 12–15. As discussed with respect to base pattern 11, each level one pattern 12–15 may be determined from a plurality of available patterns as the most suitable pattern (of the available patterns) that covers or represents as many partitions of partitions 02 as it can. This process may be repeated for any number of additional levels (e.g., level two patterns in the illustrated example) and going to smaller and smaller levels until all partitions terminate.

For example, level one pattern 12 may be determined for portion 30 of base pattern 11. As shown, level one pattern may have three terminating portions and one non-terminating portion. Level two pattern 21 may be determined for the non-terminating portion of level one pattern 12. As shown, level two pattern 21 may be fully terminating. To determine whether a pattern is terminating, as discussed, the pattern may be compared to the associated portion of picture portion 01. If a partition in the pattern fully matches the associated partition in picture portion 01, the partition in the pattern is terminating. If it does not fully match, the partition in the pattern is non-terminating.

Similarly, level one pattern 13 may be determined for the associated portion of base pattern 1911. Level one pattern 13 may have four non-terminating portions (and no terminating portions). Level two patterns 22–25 may be determined for the non-terminating portions. As shown, level two patterns in the illustrated example may all be terminating (and no further level pattern may be needed). Further, level one pattern 14 may be determined the associated portion of base pattern 1911 and may be fully terminating. Further, level one pattern 14 may be determined the associated portion of base pattern 1911 and may have two terminating portions and two non-terminating portions. Level two patterns 26 and 27 may be determined for the non-terminating portions of level one pattern 15, as shown.

Using the described techniques, multi-level pattern 10 may be determined for picture portion 01. Multi-level pattern 10 may be complete when each level is terminating (e.g., all leaves of the pattern tree are terminating). Using similar techniques, multi-level patterns may be determined for picture portions of a video frame, picture portions of a portion or slice of a video frame, picture portions of video frames of a sequence of video frames, or the like.

In this process, at each level, for each non-terminating pattern, termination bits (e.g., one bit per each partition of a non-fully terminating pattern) may be determined. For example, for base pattern 11, the termination bits may include four bits each indicating a non terminating portion (or partition) of base pattern 11. Similarly, level one pattern 12 may have associated with it four termination bits with one indicating a non-terminating portion (or partition) and three indicating terminating portions (or partitions). For example, if 1 indicates non-termination and 0 indicates termination, the bits for level one pattern 12 may be 0001. In some examples, the portions of the pattern may be ordered from a top-left to a bottom-right of the pattern, with a scan running left to right and down the pattern such that the termination bits are ordered in such a manner. Such termination bits may thereby be associated with the patterns of multi-level pattern 10.

For each pattern of multi-level pattern 10 (e.g., base pattern 11, level one patterns 12–15, and level two patterns 21–27 in the illustrated example), an entropy coded codeword may be determined based on a table such as a proxy VLC table. As will be described further herein, the applicable (or selected) proxy VLC table for entropy coding may be selected in a variety of ways such as predetermined, automatic, or adaptive. Further, the method for selecting the proxy VLC table may depend on a variable length

code tables selection mode, which may further be selected on a frame, portion of a frame, or sequence of frames basis.

In any event, entropy coded codewords for the patterns may be determined and written to a bitstream. Further, the termination bits may be entropy coded and written to the bitstream. In some examples, the termination bits may be grouped by pattern levels and/or pattern characteristics (e.g., the number of portions or partitions in a pattern), and each group may coded with a combination of proxy VLC and/or symbol run coding. Such groupings may increase the coding efficiency of the techniques discussed herein.

FIG. 20 is a flow diagram illustrating an example process 2000, arranged in accordance with at least some implementations of the present disclosure. Process 2000 may include one or more operations, functions or actions as illustrated by one or more of operations 2002, 2004, 2006, 2008, 2010, 2012, 2014, and/or 2016. Process 2000 may form at least part of a next generation video coding process. By way of non-limiting example, process 2000 may form at least part of a next generation video encoding process as undertaken by encoder system 100 of FIG. 1.

Process 2000 may begin at operation 2002, "Load Input Data Defining Partitions", where input data defining partitions may be loaded. For example, input data defining partitions 02 or the like of picture portions 01 or the like of video frame(s) may be loaded by encoder module 402 or the like. For example, input data defining partitions may be loaded via a video encoder such as video encoder 100. As discussed herein, in various examples, the input data may include luma data defining luma partitions and/or chroma data defining chroma partitions (optionally in U and V planes, for example).

Process 2000 may continue at operation 2004, "Set Variable Length Code Tables Selection Mode and/or Switching Mode", where a variable length code tables selection mode and/or a switching mode may be set. For example, as discussed, entropy coded codewords representing patterns of multi-level patterns may be determined based on a variable length coding (VLC) table. In some examples, a variable length code tables selection mode may determine whether an entropy coded codeword is determined based on a single VLC table (e.g., first mode or mode 0) or an entropy coded codeword is determined based on selecting a VLC table from multiple available multiple VLC tables (e.g., a second mode or mode 1). In various examples, the variable length code tables selection mode may be determined or set on a video frame basis, a portion or slice of a video frame basis, or on a sequence of video frames basis.

For example, a variable length code tables selection mode may be determined for a video frame, a slice of a video frame, or a sequence of video frames from a first variable length code tables selection mode where a single variable length coding table is used for determining entropy coded codewords for every picture portion of the video frame and a second variable length code tables selection mode where a picture portion-based variable length coding table is selected for each picture portion of the video frame from two or more available variable length coding tables and the selected picture portion-based variable length coding table is used for determining entropy coded codewords for the associated picture portion. The variable length code tables selection mode may be indicated via a variable length code tables selection mode indicator (e.g., a bit or bits), which may be encoded to the bitstream.

In examples where the variable length code tables selection mode is set such that a single VLC table is used for all codewords, no switching mode may be needed or set. In such examples, the single VLC table may be an average VLC table or the like.

In examples where the variable length code tables selection mode is set such that VLC table is selected from available VLC tables, then a switching mode may be set. In such examples, the encoder and decoder may need to determine which of the available VLC tables should be implemented on a picture portion basis or the like. In such examples, the switching mode may be an automatic switching mode or an adaptive switching mode or the like.

In the automatic switching mode, a selected VLC table may be determined based on a technique that may be performed at both the encoder and decoder (e.g., such that no indicator needs to be sent in the bitstream). For example, in the automatic switching mode, the VLC table may be determined for a picture portion automatically based on a prediction of complexity for the first portion. For example, the prediction of complexity may include determining a number of splits for picture portions neighboring the first picture portion and using a selected VLC table based on the number of splits. In some examples, a high complexity VLC coding table may be selected when the number of splits is above a threshold and the low complexity variable length coding table may be selected when the number of splits is at or below the threshold.

In the adaptive switching mode, a selected VLC table may be determined at the encoder and transmitted via an adaptive switching mask. For example, the adaptive switching mask may generated by including indications (e.g., bits) indicating which of a

first VLC table and a second VLC table is selected for each picture portion. The VLC table may be determined based on a prediction of complexity as discussed, a rate distortion optimization, or the like. As discussed herein, the adaptive switching mask may be entropy encoded and written to the bitstream.

As discussed, when the variable length code tables selection mode is set such that VLC table is selected from available VLC tables, then a switching mode may be set from one of an automatic switching mode or an adaptive switching mode. In some examples, the switching mode may be determined based on a quantizer value currently being implemented by the video system. For example, the adaptive switching mode may be the selected switching mode if the quantizer value is less than a predetermined threshold. For example, the predetermined may be 32 or the like.

Process 2000 may continue at operation 2006, "Optionally Generate Binary Mask of Picture Portions with Terminating 32x32 Partitions", where a binary mask of picture portions with terminating 32x32 partitions may optionally be generated. As discussed, multi-level patterns may be determined for picture portions. In some examples, picture portions may be 64x64 pixels. Such picture portions may include 32x32 terminating portions such that identifying and providing a binary mask of picture portions with terminating 32x32 partitions may be advantageous to coding efficiency. For example, a binary mask of picture portions may be generated that includes a plurality of binary indicators (e.g., bits) each indicating whether an associated picture portion comprises four 32x32 non-terminating partitions or four 32x32 partitions with at least one terminating 32x32 partition. The binary mask of picture portions may be entropy coded and written to the bitstream.

Process 2000 may continue at operation 2008, "Optionally Encode the Binary Mask of Picture Partitions with Terminating 32x32 Partitions", where the binary mask of picture portions may optionally be entropy coded. The binary mask may be encoded using bitmap coding or symbol run coding, for example, and the selected coding technique may be transmitted via the bitstream as a binary mask of picture portions coding technique indicator.

Process 2000 may continue at operation 2010, "Optionally Generate and Encode an Adaptive Switching Mask", where an adaptive switching mask may be optionally generated and encoded. As discussed above, an adaptive switching mask may be implemented when the variable length code tables selection mode is set such that VLC

table is selected from available VLC tables and the switching mode is set to adaptive switching mode. In such examples, the described adaptive switching mask may be generated and entropy encoded. For example, seven coding techniques may be evaluated for a smallest bit cost and the adaptive switching mask may be coded using the selected technique. Further, the selected technique may be provided via the bitstream as an adaptive switching mask coding technique indicator.

Process 2000 may continue at operation 2012, "Determine Optimal Multi-Level Patterns for Picture Portions and Encode Into Buffer", where optimal multi-level patterns for picture portions may be determined and encoded into a buffer. For example, multi-level patterns may be determined as discussed with respect to FIG. or elsewhere herein. The multi-level patterns may be used to determine, for each pattern of a multi-level pattern, a entropy coded codeword. As discussed, depending on the variable length code tables selection mode and switching mode, the table used to determine entropy coded codewords may vary (as is discussed herein below via example tables). Also as discussed, in some examples, the selected codeword for a pattern may indicate the pattern is terminating such that additional termination bits may not be required for terminating patterns (and coding efficiency may be enhanced).

Process 2000 may continue at operation 2014, "Generate and Determine Optimal Method for Coding Termination Bits and Encode Into Buffers", where termination bits may be generated, an optimal method for coding the termination bits may be determined, and the termination bits may be encoded into a buffer. For example, the termination bits may be generated as discussed with respect to FIG. and elsewhere herein. The termination bits may be grouped as is discussed further below, and each group of termination bits may be entropy encoded. The entropy encoded bits may be stored in a buffer.

Process 2000 may continue at operation 2016, "Write Termination Bits Buffer to Bitstream", where the entropy encoded termination bits may be written from the buffer to the bitstream.

Process 2000 may continue at operation 2018, "Write Patterns Buffer to Bitstream", where the entropy coded codewords (for patterns of the multi-level patterns) may be written from the buffer to the bitstream.

As discussed, a variety of modes and options may be available in the techniques discussed herein. Table A below provides example descriptions for some of the terms used herein.

Term	Example Description
Mode=0	VLC Tables Selection Mode 0: One (e.g., average) VLC pattern table is used
Mode=1	VLC Tables Selection Mode 1: Two (high and low complexity) or more VLC pattern tables are used
asw_mode=0	(Automatic) Switching Mode 0 for VLC Tables Selection Mode 1: Switch between two VLC tables at picture portion level by doing a prediction of complexity (e.g., based on number of splits in the neighboring tiles). If the predicted complexity bit is 0 select low-complexity pattern VLC table, if it is 1 then select the high complexity one.
asw_mode=1	(Adaptive) Switching Mode 1 for VLC Tables Selection Mode 1: Switch between two or VLC tables at picture portion level based on asw_mask. If asw_mask bit for the tile is 0 select low-complexity pattern VLC table, if it is 1 then select the high complexity one.
Mode=1&Qp<32	An example criterion for setting the asw_mode to 1 (else it is set to 0)
Mode=1&asw_mode=1	(Adaptive) Switching Mode 1 for VLC Tables Selection Mode 1 case: Two VLC tables are used for patterns and switching per picture portion level is done as per asw_mask.
Mode=1&asw_mode=0	(Automatic) Switching Mode 0 for VLC Tables Selection Mode 1: Two VLC tables are used for patterns and switching per tile level is done using complexity prediction based on neighboring tiles.
mask[]	Binary Mask of Picture Portions with Terminating 32x32 Partitions (value is '0' when all 4, 32x32's are nonterminating, else value is '1', e.g., at least one

	32x32 is terminating)
asw_mask[]	Binary Mask of VLC table switching per picture portion (0=select low-complexity pattern VLC table, 1=select the high complexity one.)

Table A: Example Terms and Descriptions

FIG. 21 is a flow diagram illustrating an example process 2100, arranged in accordance with at least some implementations of the present disclosure. Process 2100 may include one or more operations, functions or actions as illustrated by one or more of operations 2102, 2104, 2106, 2108, 2110, and/or 2112. Process 2100 may form at least part of a next generation video coding process. By way of non-limiting example, process 2100 may form at least part of a next generation video decoding process as undertaken by decoder system 200 of FIG. 2.

Process 2100 may begin at operation 2102, "Receive Encoded Bitstream and Input Data for Processing", where an encoded bitstream may be received and/or data may be input for processing. For example, a bitstream encoded as discussed herein may be received at a video decoder or the like. The bitstream may include any encoded data as discussed herein such as entropy coded codewords representing patterns of multi-level patterns, entropy coded termination bits, header data, indicators, or the like. The bitstream may be considered data or input data, which may be input for processing. For example, the bitstream may be input to adaptive entropy decoder 202 or the like for processing.

Process 2100 may continue at operation 2104, "Decode and Set Variable Length Code Tables Selection Mode and/or Switching Mode", where the bitstream may be decoded to determine a variable length code tables selection mode and/or a switching mode. For example, the bitstream may include a variable length code tables selection mode indicator for a video frame, a portion or slice of the video frame, or a sequence of video frames. The variable length code tables selection mode indicator may indicate (e.g., for the video frame, the portion or slice of the video frame, or the sequence of video frames), a variable length code tables selection mode from a first variable length code tables selection mode where a single variable length coding table is used for determining entropy coded codewords for every picture portion of the video frame and a second variable length code tables selection mode where a picture portion-based variable length coding table is selected for each picture portion of the video frame from two or more available variable length

coding tables and the selected picture portion-based variable length coding table is used for determining entropy coded codewords for the associated picture portion or any variable length code tables selection mode as discussed herein.

As discussed above, in examples where the variable length code tables selection mode is set such that a single VLC table is used for all codewords, no switching mode may be needed or set. In such examples, the single VLC table may be an average VLC table or the like.

In examples, where the variable length code tables selection mode is set such that VLC table is selected from available VLC tables, then a switching mode may be determined. For example, the bitstream may include a a selected switching mode indicator that indicates whether a switching mode between available variable length coding tables. For example, the switching mode may be an automatic switching mode or an adaptive switching mode.

As discussed herein, in the automatic switching mode, a selected VLC table may be determined based on a technique that may be performed at both the encoder and decoder (e.g., such that no indicator needs to be sent in the bitstream). For example, in the automatic switching mode, the VLC table may be determined for a picture portion automatically based on a prediction of complexity for the first portion. For example, the prediction of complexity may include determining a number of splits for picture portions neighboring the first picture portion and using a selected VLC table based on the number of splits. For example, a high complexity VLC coding table may be selected when the number of splits is above a threshold and the low complexity variable length coding table may be selected when the number of splits is at or below the threshold.

In the adaptive switching mode, a selected VLC table may be determined at the encoder and transmitted via an adaptive switching mask, as is discussed further herein and, in particular, with respect to operation 2108 below.

Process 2100 may continue at operation 2106, "Decode Binary Mask of Picture Portions with Terminating 32x32 Partitions, if Applicable", where a binary mask of picture portions with terminating 32x32 partitions may be decoded, if applicable. As discussed, in some examples, a binary mask of picture portions may be included in the bitstream that includes a plurality of binary indicators (e.g., bits) each indicating whether associated picture portions have four 32x32 non-terminating partitions or four 32x32 partitions with

at least one terminating 32x32 partition. Such a mask may be decoded at operation 2106 and implemented in the decoding to skip such terminating 32x32 partitions.

Process 2100 may continue at operation 2108, "Decode Adaptive Switching Mask, if Applicable", where an adaptive switching mask may be decoded, if applicable. For example, an optional adaptive switching mask may be received as discussed above. The adaptive switching mask may include indications (e.g., bits) indicating which of a first VLC table and a second VLC table is selected for each picture portion. The adaptive switching mask may indicate for entropy coded codewords of a picture portion which VLC table of available VLC tables is used for entropy decoding the codewords.

Process 2100 may continue at operation 2110, "Decode Termination Bits", where the termination bits may be decoded. As discussed the termination bits may have been grouped by pattern levels and/or pattern characteristics (e.g., the number of portions or partitions in a pattern), and each group may coded with a combination of proxy VLC and/or symbol run coding. In such examples, the received entropy coded termination bits may be decoded using proxy VLC and/or symbol run coding and ungrouped. The details of example groupings are discussed further below.

Process 2100 may continue at operation 2112, "Decode Patterns from Entropy Coded Codewords and Reconstruct Partitions", where patterns may be decoded from the entropy coded codewords and partitions may be reconstructed. For example, the entropy coded codewords may indicate patterns from available patterns. For example, the patterns may be contained in a codebook or the like. The decoded patterns and termination bits may be used to reconstruct partitions of picture portions such as partitions 02 of picture portion 01 or the like.

As discussed, a multi-level pattern may be determined for partitions of a picture portion (e.g., a tile or the like). The multi-level pattern may include a base pattern and/or one or more higher level patterns. The patterns may be coded via entropy coded codewords with the codewords determined from a single table or a selected table from a variety of available tables, depending on a variable length code tables selection mode. When the table is selectable, the method for selecting the VLC table may be based on a switching mode (e.g., an automatic switching mode or an adaptive switching mode or the like). The variable length code tables selection mode and the switching mode (if needed) may be entropy coded and written to a bitstream via indicators. If the switching mode is the adaptive switching mode, an adaptive switching mask may also be entropy coded and

written to the bitstream. Further, a binary mask of picture portions indicating picture portions with non-terminating sub-portions (e.g., 32x32 partitions) may be optionally encoded and written to the bitstream as discussed. The entropy coded codewords may be written to the bitstream. Termination bits may be gathered, optionally grouped (as detailed further herein), entropy encoded, and written to the bitstream.

Also as discussed, the encoded bitstream may be received and the discussed processes and techniques may be substantially reversed to reconstruct the partitions of the picture portion. FIGS. 22 and 23 illustrate an example encoding process 2200 and an example decoding process 2300, respectively, for implementing certain aspects of the described techniques.

FIGS. 22(A)–22(F) illustrate a flow diagram of an example process 2200, arranged in accordance with at least some implementations of the present disclosure. Process 2200 may include operations or operations groups 2002, 2004, 2006, 2008, 2010, 2012, 2014, 2016, and/or 2018, which may provide the same or similar techniques as those operations discussed with respect to FIG. 20 herein. In some examples, the process illustrated by FIGS. 22(A)–22(F) may provide processing for encoding partitions of 64x64 picture portions of luma data.

Process 2200 may begin with operation or operation group 2002, which may include the operation “Input Data: Intra partitions (Luma 64x64 tiles), Mode, Qp”, where input data may be loaded. For example, the input data may include intra partitions for luma with a picture portion (e.g., tile) size of 64x64 pixels. In some examples, as shown, the input data may also include a variable length code tables selection mode and a quantizer value, Qp.

Process 2200 may continue at operation or operation group 2004, which may include decision operation “Mode=1 & Qp<32?”, operation “asw_mode = 0”, and operation “asw_mode = 1”, where the variable length code tables selection mode may be determined. If the variable length code tables selection mode is 1 (e.g., such that a VLC table is selected from two or more available tables), then a switching mode may be determined. As discussed, such a mode may be based on the quantizer value. In the illustrated example, if the quantizer value is above a threshold of 32, the switching mode is set to adaptive an if it is at or below a threshold of 32, the switching mode is set to automatic. For example, the operations may set adaptive switching mode—if VLC table selection mode 0 is used and Qp < 32, enable adaptive switching mode, else disable it (e.g., use automatic switching).

Process 2200 may continue at operation or operation group 2006, which may include operation “i = 0”, operation “Set t to the first tile in the picture”, decision operation “t has terminating 32x32 partitions?”, operation “mask[i]=0”, operation “mask[i]=1”, operation “i = i + 1”, and decision operation “All tiles done?”. Operations 2006 may generate a binary mask of picture portions with terminating 32x32 partitions. For example, the operations may get the mask of picture portions (e.g., tiles) with terminating 32x32 partitions—for all tiles, collect a mask bit; mask bit is set to 1 if there is a terminating 32x32 partition in the picture portion; otherwise it is set to 0. The length of the mask is the number of picture portions in the video frame (e.g., picture).

Process 2200 may continue at operation or operation group 2008, which may include decision operation “mask = allo 0s?”, operation “Encode "1"”, operation “Encode "0"”, operation “cost[0] = cost of coding with Bitmap, cost[1] = cost of coding with Symbol Run (Tbl0)”, operation “cost[j] = min(cost[0],cost[1])”, operation “Encode header for method j (1 bit)”, and operation “Encode mask with method j”. Operations 2008 may optionally encode a binary mask of picture portions with terminating 32x32 partitions. For example, the operations may encode the mask of picture portions (e.g., tiles) with terminating 32x32 partitions—if all bits are 0 use 1 bit (0) to indicate it; otherwise, spend 1 bit header (1) and code the mask as follows: compute cost of coding the mask with symbol run method and set cost of bitmap coding to the mask length, select the minimum of the 2 costs as the coding method, encode the 1 bit header indicating the selected method, and encode the mask with the selected method.

Process 2200 may continue at operation or operation group 2010, which may include decision operation “asw_mode=1?”, operation “i = 0”, operation “Set t to the first tile in the picture”, decision operation “Num. part. in t<1/16th of the num. of 4x4 blks in t?”, operation “asw_mask[i] = 0”, operation “asw_mask[i] = 1”, operation “i = i +1”, decision operation “All tiles done?”, decision operation “asw_mask=all 0s?”, operation “Encode “1””, operation “Encode “0””, operation “cost[0] = cost of coding asw_mask with Bitmap, cost[1] = cost of coding asw_mask with Symbol Run (Tbl0), cost[2] = cost of coding inv.asw_mask with Symbol Run (Tbl0), cost[3] = cost of coding diff.asw_mask with Symbol Run (Tbl0), cost[4] = cost of coding asw_mask with Symbol Run (Tbl5), cost[5] = cost of coding inv.asw_mask with Symbol Run (Tbl0), cost[6] = cost of coding diff.asw_mask with Symbol Run (Tbl0)”, operation “cost[j] = min(cost[k]), k=0,1,...,6”, operation “Encode header for method j (2-3 bits)”, and operation “Encode asw_mask with method j”. Operations 2010 may optionally generate and encode an adaptive switching

mask. For example, the operations may compute and encode the adaptive switching mask—if adaptive switching mode is enabled, then derive and encode the adaptive switching mask as follows: compute the adaptive switching mask bit for each tile: if the number of partitions in a tile is small then set the bit to 0; otherwise set it to 1, encode the adaptive switching mask—if all bits are 0 use 1 bit (0) to indicate it, otherwise, spend 1 bit header (1) and code the mask as follows: compute cost of coding the mask with 2 symbol run methods, applied on the mask bits, inverted mask bits and differenced mask bits, and set cost of bitmap coding to the mask length (total of 7 costs computed), select the minimum of the 7 costs as the coding method, encode the 2-3 bit header indicating the selected method, and encode the mask with the selected method.

Process 2200 may continue at operation or operation group 2012, which may include operation "i = 0", operation "Set t to the first tile in the picture", decision operation "Mode=1 & asw_mode=1?", decision operation "asw_mask[i]=0?", decision operation "Mode=1 & asw_mode=0?", operation "N = predicted number of partitions in t from neighbors of t (top,left,topleft)", decision operation "N < T_low?", operation "Set pattern VLC table to low-complexity table", operation "Set pattern VLC table to high-complexity table", operation "Set pattern VLC table to average table", operation "P = pattern corresponding to tile t partitions", operation "Patt_ind = search pattern codebook (P's dimensions) to find exact match", decision operation "Patt_ind > -1?", operation "Add pattern codeword (corresponding to Patt_ind) to picture_buffer", operation "ind = 0", operation "minCT = ∞", operation "Set T to ind pattern in codebook (of t dimension)", decision operation "T is valid containter of P?", operation "Approximate the coding cost CT of T and its sub-patterns (Level 1,2,...). In the process accumulate: tmp_pattern_bits_buffer and tmp_termiantion_bits_buffers", decision operation "minCT>CT?", operation "minCT=CT, tile_patt_bits_buff = temp_pattern_bits_buffer, tile_term_bits_buff = tmp_termination_bits_buffers", operation "ind = ind +1", decision operation "All T patterns processed?", operation "Add tile_patt_bits_buff to pic_buff, Add tile_term_bits_buff to pic_tb_buffs", operation "i = i +1", and decision operation "All tiles done?". Operations 2012 may determine optimal multi-level patterns for picture portions and encode them into a buffer. For example, the operations may select pattern VLC table: if VLC tables selection mode 1 is used and adaptive switching mode is enabled, then if the adaptive switching mask is 0 select low-complexity pattern VLC table, if it is 1 then select the high complexity one; if VLC tables selection mode 1 is used and adaptive switching mode is disabled (e.g., set to automatic switching), then do a prediction of complexity

based on number of splits in the neighboring tiles—if the predicted complexity bit is 0 select low-complexity pattern VLC table, if it is 1 then select the high complexity one; and if VLC tables selection mode 0 is used, set the pattern VLC table to the average one. Further the operations may get the pattern corresponding to the tile dimensions and partitions. For example, the operations may search the pattern codebook (e.g., Tables 38(a-e)) for an exact match; if the match exists, set the matching index to the matching pattern index in the codebook table, otherwise set it to -1; if the index is greater than -1, write the pattern codeword bits to a buffer and move to the next picture portion (e.g., tile), else continue to go through the tile-dimension pattern codebook (Table 38(a)) and find the valid Level 0 patterns (e.g., ones without invalid cuts); for all such patterns do the following: set *i* to 0, accumulate termination bits for the current non-terminating level *i* pattern, for all non-terminating partitions (e.g., at level *i*+1) of the Level *i* pattern, search for either the exact match or the best valid match in the corresponding pattern codebook; if exact match is found, write the pattern's codeword bits to a temporary buffer and move to next non-terminating partition of the current pattern at level *i*; if no exact match is found, write bits of the best valid container partitions (non-terminating), increase level *i* by 1 and repeat until all partitions are terminating. The operations may evaluate the cost of bits accumulated in the temporary buffer, and if it is smallest so far, copy accumulated termination bits and pattern bits to tile buffers. The operations may also, at the end of processing a non-terminating tile, copy tile pattern buffer into the picture buffer, and copy tile termination bits buffers into picture termination bits buffers.

Process 2200 may continue at operation or operation group 2014, which may include operation “Invert (bitwise) pic_tb_buffs”, decision operation “pic_tb_buff[0] = all 0s?”, operation “Encode "1"”, operation “Encode "0"”, operation “Encode pic_tb_buff[0] using Proxy VLC coding (Table 2)”, operation “cost[0] = cost of coding pic_tb_buff[1] with Proxy VLC (Table 4), cost[1] = cost of coding pic_tb_buff[1] with Proxy VLC (Table 5), cost[2] = cost of coding pic_tb_buff[1] with Proxy VLC (Table 6)”, operation “cost[j] = min(cost[0], cost[1], cost[2])”, operation “Encode header for method j (1-2 bits)”, operation “Encode pic_tb_buff[1] with method j”, operation “Encode pic_tb_buff[2] using Proxy VLC coding (Table 7)”, operation “cost[0] = cost of coding pic_tb_buff[3] with Proxy VLC (Table 9), cost[1] = cost of coding pic_tb_buff[3] with Proxy VLC (Table 10), cost[2] = cost of coding pic_tb_buff[3] with Proxy VLC (Table 11), cost[3] = cost of coding pic_tb_buff[3] with Proxy VLC (Table 12), cost[4] = cost of coding pic_tb_buff[3] with Proxy VLC (Table 13), cost[5] = cost of coding pic_tb_buff[3] with Proxy VLC

(Table 14), cost[6] = cost of coding pic_tb_buff[3] with Proxy VLC (Table 15)", operation "cost[j] = min(cost[k]), k=0,1,...6", operation "Encode header for method j (2-3 bits)", operation "Encode pic_tb_buff[3] with method j", decision operation "pic_tb_buff[4] = all 0s?", operation "Encode "1"", operation "Encode "0"", operation "cost[0] = cost of coding pic_tb_buff[4] with Symbol Run (Table 17), cost[1] = cost of coding pic_tb_buff[4] with Symbol Run (Table 18), cost[2] = cost of coding pic_tb_buff[4] with Symbol Run (Table 19), cost[3] = cost of coding pic_tb_buff[4] with Proxy VLC (Table 20)", operation "cost[j] = min(cost[k]), k=0,1,2,3", operation "Encode header for method j (2 bits)", operation "Encode pic_tb_buff[4] with method j", operation "cost[0] = cost of coding pic_tb_buff[5] withProxy VLC (Table 22), cost[1] = cost of coding pic_tb_buff[5] withProxy VLC (Table 23), cost[2] = cost of coding pic_tb_buff[5] withProxy VLC (Table 24), cost[3] = cost of coding pic_tb_buff[5] withProxy VLC (Table 25)", operation "cost[j] = min(cost[k]), k=0,1,2,3", operation "Encode header for method j (2 bits)", and operation "Encode pic_tb_buff[5] with method j". Operations 2014 may generate and determine an optimal method for coding termination bits and encode them into a buffer. For example, the operations may encode termination bits by: termination bits of non-terminating patterns at level 0 with 2 partitions are inverted and coded as follows: if all bits are 0 use 1 bit (0) to indicate it. Otherwise, spend 1 bit header (1) and code the mask with proxy VLC coding using the event map shown in Table 2, termination bits of non-terminating patterns at level 1 (or higher) with 2 partitions are inverted and coded as follows: evaluate cost of coding the bits with Proxy VLC coding using the event map shown in Table 4, Table 5 and Table 6 (3 costs), find the minimal cost and select the corresponding method, encode 1-2 bit header to indicate the selected method into the buffer, encode the termination bits with the selected method into the buffer; termination bits of non-terminating patterns at level 0 with 3 partitions are inverted and coded with the Proxy VLC coding using the event map shown in Table 7; termination bits of non-terminating patterns at level 1 (or higher) with 3 partitions are inverted and coded as follows: evaluate cost of coding the bits with Proxy VLC coding using the event map shown in Table 9, Table 10, Table 11, Table 12, Table 13, Table 14, and Table 15 (7 costs), find the minimal cost and select the corresponding method, encode 2-3 bit header to indicate the selected method, and encode the termination bits with the selected method into the buffer; termination bits of non-terminating patterns at level 0 with 4 partitions are inverted and if all bits are 0 use 1 bit (0) to indicate it, otherwise, spend 1 bit header (1) and code the bits as follows: evaluate cost of coding the bits with Symbol Run with VLC shown in Table 17, 18 and 19, and Proxy VLC coding using the event map shown in Table 20 (4 costs), find the minimal cost and select the

corresponding method, encode 2 bit header to indicate the selected method, encode the termination bits with the selected method into the buffer; termination bits of non-terminating patterns at level 1 (or higher) with 4 partitions are inverted and coded as follows: Evaluate cost of coding the bits with Proxy VLC coding using the event map shown in Table 22, Table 23, Table 24, and Table 25 (4 costs), find the minimal cost and select the corresponding method, encode 2 bit header to indicate the selected method, and encode the termination bits with the selected method into the buffer.

Process 2200 may continue at operation or operation groups 2016 and 2018, which may include operation "write pic_buff to bitstream" and operation "Finish processing current picture". Operations 2016 and 2018 may write the termination bits buffer to the bitstream, write the patterns buffer to the bitstream, and finish processing of a current video frame (e.g., picture). For example, the operations may write termination bits buffers to bitstream and write pattern bits buffer to bitstream.

FIGS. 23(A)–23(C) illustrate a flow diagram of an example process 2300, arranged in accordance with at least some implementations of the present disclosure. Process 2300 may include operations or operations groups 2102, 2104, 2106, 2108, 2110, and/or 2112, which may provide the same or similar techniques as those operations discussed with respect to FIG. 21 herein. In some examples, the process illustrated by FIGS. 23(A)–23(C may provide processing for decoding partitions of 64x64 picture portions of luma data.

Process 2300 may begin with operation or operation group 2102, which may include the operation "Input Data: Bitstream, Mode, Qp", where a bitstream may be received and/or input data may be loaded. For example, the bitstream may include entropy encoded data as discussed herein. In some examples, as shown, the bitstream and/or input data may include a variable length code tables selection mode and a quantizer value, Qp.

Process 2300 may continue at operation or operation group 2104, which may include decision operation "Mode=1 & Qp<32?', operation "asw_mode = 0", and operation "asw_mode = 1". Operations 2104 may decode and set a VLC code tables selection mode and/or switching mode as discussed herein. For example, the options and modes discussed with respect to FIGS. 22(A)–22(F) and elsewhere herein may be available via the decode of the received bitstream.

Process 2300 may continue at operation or operation group 2106, which may include operation "Decode header for mask of tiles with non-terminating 32x32 partitions" and

operation “Decode mask of tiles with non-terminating 32x32 partitions”. Operations 2106 may decode a binary mask of picture portions (e.g., tiles) with terminating 32x32 partitions, if applicable. For example, the operations may provide for decoding a header indicating a coding technique (e.g. bitmap or symbol run) and for decoding the mask of tiles with non-terminating 32x32 partitions, if applicable. Such techniques may provide for decoding operations for the encoding operations as discussed with respect to FIGS. 22(A)–22(F) and elsewhere herein.

Process 2300 may continue at operation or operation group 2108, which may include decision operation “asw_mode=1?”, operation “Decode header for asw_mask”, and operation “Decode asw_mask”. Operations 2108 may decode an adaptive switching mask, if applicable. For example, the operations may for decoding a header indicating a coding technique (e.g. bitmap, symbol run, symbol run on inverted bits, symbol run on differential bits, or the like) and for adaptive switching mask, if applicable. Such techniques may provide for decoding operations for the encoding operations as discussed with respect to FIGS. 22(A)–22(F) and elsewhere herein.

Process 2300 may continue at operation or operation group 2110, which may include operation “i = 0”, operation “Decode header for pic_tb_buff[i]”, operation “Decode pic_tb_buff[i], the termination bits group i”, operation “i = i + 1”, and decision operation “i > 6?”. Operations 2110 may decode termination bits. For example, the operations may provide for decoding and ungrouping termination bits. Such techniques may provide for decoding operations for the encoding operations as discussed with respect to FIGS. 22(A)–22(F) and elsewhere herein. For example, a more detailed discussion of grouping termination bits is provided below.

Process 2300 may continue at operation or operation group 2112, which may include operation “i = 0”, operation “Set t to the first tile in the picture”, decision operation “Mode=1 & asw_mode=1?”, decision operation “asw_mask[i]0?”, decision operation “Mode=1 & asw_mode=0?”, operation “N = predicted number of partitions in t from neighbors of t (top,left,topleft)”, decision operation “N < T_low?”, operation “Set pattern VLC table to low-complexity table”, operation “Set pattern VLC table to high-complexity table”, operation “Set pattern VLC table to average table”, operation “Decode pattern P at Level 0 (Tables 38a-e)”, decision operation “P is terminating?”, operation “Decode (move pointer) terminating bits of P (from tb group depending on the number of part. in P)”, operation “N = numb. of non-term. partitions in P”, operation “n = 0”, operation “Decode

sub-pattern of P corresponding to part. n", operation "n = n+1", decision operation "n > N?", operation "i = I+1", decision operation "All tiles decoded?", and operation "Finish decoding intra partitions for current picture". Operations 2112 may decode patterns from entropy coded codewords and reconstruct partitions of picture portions. Operations 2112 may also finish decoding partitions (e.g., intra partitions) for a current video frame (e.g., picture).

FIG. 24 illustrates an example system 2400 for entropy encoding partitions data, arranged in accordance with at least some implementations of the present disclosure. As shown, system 2400 may include a base pattern matcher module 2401, a base patterns codebook module 2402, a partition termination analyzer module 2403, a level '0' termination indicator stream collector module 2404, a partition pattern matcher module 2405, a partition pattern codebook module 2406, a sub-partition termination analyzer module 2407, a level 'n' termination indictor stream collector module 2408, a VLC pattern index encoder module 2409, a recursion level analyzer module 2410, and/or a VLC (symbol-run/proxy) termination stream encoder module 2411.

In some examples, picture portions that have been partitioned (e.g., a partitioned picture portion) may be input to base pattern matcher module 2401. For example, the partitions may be k-d tree partition or Bi-tree partitions of size 64x64 or 32x32 or the like for prediction or coding. Base pattern matcher module 2401 may search base patterns codebook module 2402, which may include candidate patterns from which the closest possible base (e.g., "coarse") match to the partitioned input picture portion may be selected. For example, the selected pattern may be referred to as a base pattern as discussed herein. As shown, the base pattern may be input to partition termination analyzer module 2403, which may output partitioning details corresponding to next non-terminating partition (e.g., level 0 non-termination) and may insert a 1-bit indicator (e.g., with value of '0' or '1') in a bitstream segment level 0 termination indicator stream. The bitstream segment level 0 termination indicator stream may be collected, as shown, via level '0' termination indicator stream collector module 2404. As shown, the details of nonterminating partition (e.g., level 0 non-termination) may be available as a first input to a switch 2421. As shown, a second input to switch 2421 may be provided as either current level or previous level non-terminating sub-partitions.

For a first non-terminating partition of the base pattern is being processed, level 0 non-termination (e.g., a next non-terminating partition) may be input to partition pattern

matcher module 2405. Partition pattern matcher module 2405 may search partition pattern codebook module 2406, which may include patterns from which a closest next level (e..g, "coarse") match to the partition (e.g., the next non-terminating partition) may be selected. The pattern may be referred to as partition pattern and may be input to sub-partition termination analyzer module 2407. Sub-partition termination analyzer module 2407 may output a detailed non-terminating partition of a current level (e.g., curr lvl non-term) at one input to a switch 2422. A second input to switch 2422 may be a next detailed non-terminating partition of a previous level. The original picture portion partitioning (e.g., the partitioned picture portion) as well as the "current level non-termination sub-partition (e.g., curr levl non-term) may be input to recursion level analyzer module 2410. Recursion level analyzer module 2410 may output the next detailed prev lvl nonterminating partition (if needed) or nothing, which may signal the completion of processing of a current picture partition. As shown, the output of switch 2422, which selects between "curr lvl non term" partition and "any prev lvl non term" partition may be fed back to switch 2422 at the input of partition pattern matcher module 2405 completing the loop. The indices and/or codewords of various pattern/sub-patterns selected to represent each non-terminating partitions may be encoded by VLC pattern index encoder module 2411. Further, the termination indicator stream for level 0 (e.g., lvl 0 term ind stream), and the various other termination indicator streams for higher levels (e.g., lvl n term ind stream) may be encoded by VLC (symbol-run/proxy) termination stream encoder module 2411. The outputs of VLC pattern index encoder module 2411 and/or VLC (symbol-run/proxy) termination stream encoder module 2411 may be transmitted to bitstream assembler 408 for assembly into a bitstream and output as discussed herein.

In various examples, system 2400 may be implemented via encoder 100, such as via adaptive entropy encoder 110 of encoder 100 (please refer to FIG. 1). In some examples, system 2400 may be implemented via picture partitions, prediction partitions, and coding partitions encoder module 402 (please refer to FIG. 4). For example, system 2400 may be implemented via one or more of adaptive codebook VLC encoder for picture partitions module 621, symbol-run VLC and/or proxy VLC encoder for intra-prediction partitions module 622, and/or symbol-run VLC and/or proxy VLC encoder for inter-prediction partitions and coding partitions module 623. For example, one or more of system 2400 may be implemented to encode picture partitions, intra-prediction partitions, inter-prediction partitions, and/or coding partitions as discussed herein.

The discussion now turns to various examples of coding. For example, luma partitions coding may include two parts: (1) coding of patterns (e.g., patterns of multi-level patterns) and (2) coding of termination bits. In some examples, chroma partitions coding may include three parts: (1) coding of patterns, (2) coding of follow bits, and (3) coding of termination bits.

As discussed, partitions in each picture portion (e.g., tile) may be represented by a set of hierarchically ordered patterns (e.g., multi-level patterns). At the 0 (base) level, a best suited starting pattern may be selected as discussed. The pattern itself may have two modes of representation. For example, the pattern may either it terminates (e.g., it faithfully or fully represents) the given partitions of the picture portion, or it may not terminate (e.g., one, some, or all of its portions or partitions need to be further coded and represented with smaller patterns—at higher levels).

Referring again to FIG. , the illustrated example shows that input picture portion partitions may be represented in a hierarchical manner. In the illustrated example, there are 12 patterns used to specify the picture portion (1 at base level, 4 at level one, and 7 at level two).

As discussed, in such examples, the terminating patterns do not need to carry any additional bits other than their actual VLC codes (e.g., no termination bits are required). However, information (e.g., termination bits) about which partitions of a non-terminating patterns may be further coded, which information needs to be carried to the decoder so the decoder may reconstruct the partitions of a picture portion. For example, a non-terminating pattern must have at least 1 partition being non-terminating so that the termination bits of a pattern have one bitwise-event less. In other words, if denote 0 represents termination and 1 represents non-termination, then in non-terminating patterns with 3 partitions event 000 will never occur (otherwise the pattern would be terminating). In some examples, this fact may be used to derive a small VLC-based coding of termination bits. In addition, in some examples, termination bits may be coded with a symbol run method.

Also as discussed, there may be two modes of operation in coding of patterns: (1) variable length code tables selection mode 0 (e.g., a single Pattern VLC Table), and (2) variable length code tables selection mode 1 (e.g., switching between VLC tables such as a complex pattern VLC Table and simple pattern VLC table).

As discussed, in luma examples, variable length code tables selection mode 0 may be based on using a single VLC table for patterns. In chroma examples, variable length code tables selection mode 0 may use a frame-based switching of coding methods. Such switching may be based on quantization parameter (Qp). The switching modes may include method A where level 0 patterns (e.g., solids) may be disabled from the VLC table and coded with a Symbol Run coder and method B where a symbol run in the V plane may be performed on a differential of the U plane and V plane solid/non-solid masks. In chroma examples, variable length code tables selection mode 1 may use the same methods as mode 0.

Returning to luma examples, as discussed, switching may be based on a prediction on a picture portion basis that uses neighboring picture portions. For example, predetermined statistics for the outcome for a current picture portion may be provided for the table switching bits for the immediate neighboring picture portions (e.g., previously decoded picture portions). For example, statistics may be computed using 16 sequences having 3 frames each using quantizers of 4, 8, 12, 16, 24, 32, 48, and 64. Further, each picture portion may be doubly coded and the best of the 2 may chosen per picture portion level. Using the predetermined statistics data, tables of probabilities may be constructed where tables were separated according to spatial complexity (SC) on a frame level and quantizer (Qp). The decision of what table to switch at a given picture portion may be determined using the corresponding probability from a table. For example, a lookup may be performed based on Qp, SC, and switch bits of neighboring picture portions. A switching mask may be generated or approximated using a counting of pattern bits for each picture portion using bits tables. If the probability of a particular table is greater than about 65%, the resultant switching mask may be used. If the probability of a particular table is less than about 65%, the switching mask may be based on the following techniques using a number of partitions in neighboring picture portions, which is illustrated with respect to FIG. 25.

FIG. 25 illustrates an example current picture portion 2501 and example neighboring picture portions 2511, 2512, and 2513, arranged in accordance with at least some implementations of the present disclosure. As shown in FIG. 25, current picture portion 2501 (at position X,Y) may have previously decoded neighboring picture portion 2511 (at position X-1, Y-1; top-left with respect to current picture portion 2501), previously decoded neighboring picture portion 2512 (at position X, Y-1; top with respect to current picture portion 2501), and previously decoded neighboring picture portion 2513 (at

position X-1, Y; left with respect to current picture portion 2501). For example, a technique using a number of partitions in neighboring picture portions may determine a number of partitions to predict the switch bit of a current picture portion. For example, a number of partitions for current picture portion 2501 may be estimated using known number of partitions of the immediate neighboring picture portions 2511, 2512, and 2513. For inner picture portions, such as current picture portion 2501, the estimated number of partitions may be 60% of the number of partitions of picture portion 2513, 30% of the number of partitions of picture portion 2512, and 10% of the number of partitions of picture portion 2511. The estimated number of picture portion partitions may be compared to a predetermined threshold such that if the estimated number of picture portion partitions is greater than the predetermined threshold, the VLC table is set to the high partition or high complexity VLC table (corresponding to the switch bit 1) and if the estimated number of picture portion partitions is less than the predetermined threshold, VLC table is set to the low partition or low complexity VLC table (corresponding to the switch bit 0). In some examples, the predetermined threshold may be 16.

As discussed, for all non-terminating patterns, termination bits may be specified such that the decoder may determine which partitions in a non-terminating pattern do not terminate and are further coded. Also as discussed, the termination bits may be grouped, entropy coded as grouped, and written to a bitstream. For example, in the luma plane for 64x64 pixel picture portions, the termination bits may be accumulated and grouped into 7 different groups as follows:

Group	Description
1st	Termination bits of non-terminating patterns at level 0 with 2 partitions
2nd	Termination bits of non-terminating patterns at level 1 (or higher) with 2 partitions
3rd	Termination bits of non-terminating patterns at level 0 with 3 partitions
4th	Termination bits of non-terminating patterns at level 1 (or higher) with 3 partitions
5th	Termination bits of non-terminating patterns at level 0 with 4 partitions

6th	Termination bits of non-terminating patterns at level 1 (or higher) with 4 partitions
7th	Termination bits of patterns with more than 4 partitions at any level

Table B: Example Luma Plane Termination Bit Groupings

For example, the luma picture portions may be 64x64 pixels. In such examples (e.g., luma for picture portions of 64x64 size) the following methods may be used to code the termination bits. For example, the 1st group of termination bits (termination bits of non-terminating patterns at level 0 with 2 partitions) may be inverted and coded with the one of 2 methods from Table 1 which produces the smallest bit cost.

Method No.	**VLC Header**	**Description**
0	0	All bits are all 0
1	1	Proxy VLC coding using the event map shown in Table 2

Table 1: Methods and their header bits used in coding of termination bits of non-terminating patterns with 2 partitions at level 0 (in luma plane where 64x64 tile size is used)

Event	**VLC**
00	0
01	10
10	11
11	-

Table 2: Proxy VLC table used in Method 1 of coding the inverted termination bits of non-terminating patterns with 2 partitions at level 0 (in luma plane where 64x64 tile size is used)

For example, the 2nd group of termination bits (termination bits of non-terminating patterns at level 1 or higher with 2 partitions) may be inverted and coded with the one of 3 methods from Table 3 which produces the smallest bit cost.

Method No.	VLC Header	Description
0	0	Proxy VLC coding using the event map shown in Table 4
1	10	Proxy VLC coding using the event map shown in Table 5
2	11	Proxy VLC coding using the event map shown in Table 6

Table 3: Methods and their header bits used in coding of termination bits of non-terminating patterns with 2 partitions at level 1 or higher (in luma plane where 64x64 tile size is used)

Event	VLC
00	10
01	0
10	11
11	-

Table 4: Proxy VLC table used in Method 0 of coding the inverted termination bits of non-terminating patterns with 2 partitions at level 1 or higher (in luma plane where 64x64 tile size is used)

Event	VLC
00	0
01	10
10	11
11	-

Table 5: Proxy VLC table used in Method 1 of coding the inverted termination bits of non-terminating patterns with 2 partitions at level 1 or higher (in luma plane where 64x64 tile size is used)

Event	VLC
00	11
01	10
10	0
11	-

Table 6: Proxy VLC table used in Method 2 of coding the inverted termination bits of non-terminating patterns with 2 partitions at level 1 or higher (in luma plane where 64x64 tile size is used)

For example, the 3rd group of termination bits (termination bits of non-terminating patterns at level 0 with 3 partitions) may be inverted and coded with the Proxy VLC coding using the event map shown in Table 7.

Event	VLC
000	00
001	010
100	011

010	100
011	101
110	110
101	111
111	-

Table 7: Proxy VLC table used for coding the inverted termination bits of non-terminating patterns with 3 partitions at level 0 (in luma plane where 64x64 tile size is used)

For example, the 4th group of termination bits (termination bits of non-terminating patterns at level 1 (or higher) with 3 partitions) may be inverted and coded with the one of 7 methods from Table 8 which produces the smallest bit cost.

Method No.	**VLC Header**	**Description**
0	00	Proxy VLC coding using the event map shown in Table 9
1	010	Proxy VLC coding using the event map shown in Table 10
2	011	Proxy VLC coding using the event map shown in Table 11
3	100	Proxy VLC coding using the event map shown in Table 12
4	101	Proxy VLC coding using the event map shown in Table 13
5	110	Proxy VLC coding using the event map shown in Table 14
6	111	Proxy VLC coding using the event map shown in Table 15

Table 8: Methods and their header bits used in coding of termination bits of non-terminating patterns with 3 partitions at level 1 or higher (in luma plane where 64x64 tile size is used)

Event	VLC
000	00
001	010
100	011
010	100
011	101
110	110
101	111
111	-

Table 9: Proxy VLC table used in Method 0 of coding the inverted termination bits of non-terminating patterns with 3 partitions at level 1 or higher (in luma plane where 64x64 tile size is used)

Event	VLC
000	010
001	00
100	011
010	100
011	101
110	110
101	111
111	-

Table 10: Proxy VLC table used in Method 1 of coding the inverted termination bits of non-terminating patterns with 3 partitions at level 1 or higher (in luma plane where 64x64 tile size is used)

Event	VLC
000	011
001	010
100	00
010	100
011	101
110	110
101	111
111	-

Table 11: Proxy VLC table used in Method 2 of coding the inverted termination bits of non-terminating patterns with 3 partitions at level 1 or higher (in luma plane where 64x64 tile size is used)

Event	VLC
000	100
001	010
100	011
010	00
011	101
110	110
101	111
111	-

Table 12: Proxy VLC table used in Method 3 of coding the inverted termination bits of non-terminating patterns with 3 partitions at level 1 or higher (in luma plane where 64x64 tile size is used)

Event	VLC

000	101
001	010
100	011
010	100
011	00
110	110
101	111
111	-

Table 13: Proxy VLC table used in Method 4 of coding the inverted termination bits of non-terminating patterns with 3 partitions at level 1 or higher (in luma plane where 64x64 tile size is used)

5

Event	**VLC**
000	110
001	010
100	011
010	100
011	101
110	00
101	111
111	-

Table 14: Proxy VLC table used in Method 5 of coding the inverted termination bits of non-terminating patterns with 3 partitions at level 1 or higher (in luma plane where 64x64 tile size is used)

10

Event	**VLC**
000	111
001	010
100	011

010	100
011	101
110	110
101	00
111	-

Table 15: Proxy VLC table used in Method 6 of coding the inverted termination bits of non-terminating patterns with 3 partitions at level 1 or higher (in luma plane where 64x64 tile size is used)

5

For example, the 5th group of termination bits (termination bits of non-terminating patterns at level 0 with 4 partitions) may be inverted and coded with the one of 5 methods from Table 16 which produces the smallest bit cost.

Method No.	**VLC Header**	**Description**
0	0	All bits are all 0
1	100	Symbol Run with VLC shown in Table 17
2	101	Symbol Run with VLC shown in Table 18
3	110	Symbol Run with VLC shown in Table 19
4	111	Proxy VLC coding using the event map shown in Table 20

10

Table 16: Methods and their header bits used in coding of termination bits of non-terminating patterns with 4 partitions at level 0 (in luma plane where 64x64 tile size is used)

Run	**Len.**	**Code**	**Run**	**Len.**	**Code**
0	2	11	17	6	101101
1	4	0001	18	6	100001

2	4	0101	19	7	0000001
3	4	0100	20	7	0000000
4	5	10101	21	7	1000101
5	5	10100	22	7	1000100
6	5	01101	23	7	1011000
7	5	10111	24	7	1011001
8	6	000011	25	8	10000010
9	6	011001	26	8	10001101
10	6	100110	27	8	10000011
11	6	100100	28	8	10001111
12	6	000010	29	8	10000000
13	6	100101	30	8	10000001
14	6	011000	31	8	10001110
15	6	100111	32	8	10001100
16	6	000001	ESC	4	0111
-	-	-	EOB	3	001

Table 17: Symbol Run VLC table used in Method 1 of coding the inverted termination bits of non-terminating patterns with 4 partitions at level 0 (in luma plane where 64x64 tile size is used)

5

Run	**Len.**	**Code**	**Run**	**Len.**	**Code**
0	2	01	17	7	1011001
1	3	001	18	8	00000001
2	3	111	19	8	10100001
3	4	0001	20	8	10100010
4	4	1001	21	8	00000101
5	4	1100	22	8	10100011
6	4	1101	23	8	10110001
7	5	10101	24	9	101000000
8	5	10111	25	9	000000000
9	6	100000	26	9	000001001

10	6	100011	27	9	101100000
11	6	101101	28	9	000000001
12	6	101001	29	9	101000001
13	7	0000001	30	9	101100001
14	7	1000100	31	10	0000010000
15	7	0000011	32	10	0000010001
16	7	1000101	ESC	6	100001
-	-	-	EOB	5	00001

Table 18: Symbol Run VLC table used in Method 2 of coding the inverted termination bits of non-terminating patterns with 4 partitions at level 0 (in luma plane where 64x64 tile size is used)

5

Run	**Len.**	**Code**	**Run**	**Len.**	**Code**
0	1	1	17	11	00000000001
1	2	01	18	11	00000001101
2	4	0001	19	12	000000000000
3	4	0011	20	12	000000000001
4	5	00001	21	12	000000011001
5	5	00101	22	14	00000001100011
6	6	000001	23	14	00000001100000
7	7	0000001	24	15	000000011000101
8	7	0010010	25	16	0000000110001001
9	7	0010011	26	17	00000001100010001
10	8	00100000	27	18	000000011000100001
11	8	00100001	28	19	0000000110001000001
12	9	000000001	29	20	00000001100010000001
13	10	0000000001	30	21	000000011000100000001
14	10	0000000111	31	22	0000000110001000000000
15	10	0000000100	32	22	0000000110001000000001
16	10	0000000101	ESC	14	00000001100001
-	-	-	EOB	7	0010001

Table 19: Symbol Run VLC table used in Method 3 of coding the inverted termination bits of non-terminating patterns with 4 partitions at level 0 (in luma plane where 64x64 tile size is used)

Event	VLC
0000	0
0001	1000
0100	1001
0010	1010
0011	1011
0110	11000
0101	11001
0111	11010
1000	11011
1001	11100
1100	11101
1010	111100
1011	111101
1110	111110
1101	111111
1111	-

Table 20: Proxy VLC table used in Method 4 of coding the inverted termination bits of non-terminating patterns with 4 partitions at level 0 (in luma plane where 64x64 tile size is used)

For example, the 6th group of termination bits (termination bits of non-terminating patterns at level 1 or higher with 4 partitions) may be inverted and coded with the one of 4 methods from Table 21 which produces the smallest bit cost.

Method No.	VLC Header	Description
0	00	Proxy VLC coding using the event map shown in Table 22
1	01	Proxy VLC coding using the event map shown in Table 23
2	10	Proxy VLC coding using the event map shown in Table 24
3	11	Proxy VLC coding using the event map shown in Table 25

Table 21: Methods and their header bits used in coding of termination bits of non-terminating patterns with 4 partitions at level 1 or higher (in luma plane where 64x64 tile size is used)

5

Event	VLC
0000	0
0001	1000
0100	1001
0010	1010
0011	1011
0110	11000
0101	11001
0111	11010
1000	11011
1001	11100
1100	11101
1010	111100
1011	111101
1110	111110
1101	111111
1111	-

Table 22: Proxy VLC table used in Method 0 of coding the inverted termination bits of non-terminating patterns with 4 partitions at level 1 or higher (in luma plane where 64x64 tile size is used)

Event	VLC
0000	00
0001	010
0100	011
0010	100
0011	101
0110	11000
0101	11001
0111	11010
1000	11011
1001	11100
1100	11101
1010	111100
1011	111101
1110	111110
1101	111111
1111	-

Table 23: Proxy VLC table used in Method 1 of coding the inverted termination bits of non-terminating patterns with 4 partitions at level 1 or higher (in luma plane where 64x64 tile size is used)

Event	VLC
0000	00
0001	0100
0100	0101
0010	0110

0011	0111
0110	1000
0101	1001
0111	1010
1000	1011
1001	1100
1100	1101
1010	11100
1011	11101
1110	11110
1101	11111
1111	-

Table 24: Proxy VLC table used in Method 2 of coding the inverted termination bits of non-terminating patterns with 4 partitions at level 1 or higher (in luma plane where 64x64 tile size is used)

5

Event	**VLC**
0000	000
0001	0010
0100	0011
0010	0100
0011	0101
0110	0110
0101	0111
0111	1000
1000	1001
1001	1010
1100	1011
1010	1100
1011	1101
1110	1110

1101	1111
1111	-

Table 25: Proxy VLC table used in Method 3 of coding the inverted termination bits of non-terminating patterns with 4 partitions at level 1 or higher (in luma plane where 64x64 tile size is used)

For example, the 7th group of termination bits (termination bits of non-terminating patterns at any level with more than 4 partitions) may be coded uncompressed (e.g., "as is" with bitmap coding. For example, termination bits of non-terminating patterns at any level with more than 4 partitions may be a substantially rare event.

For example, in the luma plane for 32x32 pixel picture portions, the 2nd, 3rd, 4th, 6th, and 7th group of termination bits may coded using the same techniques as those for luma plane with 64x64 pixel picture portions. However, 1st and 5th group of termination bits be coded using different techniques. For example, in the luma plane for 32x32 pixel picture portions, the 1st group of termination bits (termination bits of non-terminating patterns at level 0 with 2 partitions) may be inverted and coded with the one of 3 methods from Table 26 which produces the smallest bit cost.

Method No.	**VLC Header**	**Description**
0	10	All bits are all 0
1	0	Proxy VLC coding using the event map shown in Table 27
2	11	Symbol Run with VLC shown in Table 28

Table 26: Methods and their header bits used in coding of termination bits of non-terminating patterns with 2 partitions at level 0 (in luma plane where 32x32 tile size is used)

Event	VLC
00	0
01	10
10	11
11	-

Table 27: Proxy VLC table used in Method 1 of coding the inverted termination bits of non-terminating patterns with 2 partitions at level 0 (in luma plane where 32x32 tile size is used)

5

Run	Len.	Code	Run	Len.	Code
0	2	11	17	6	101101
1	4	0001	18	6	100001
2	4	0101	19	7	0000001
3	4	0100	20	7	0000000
4	5	10101	21	7	1000101
5	5	10100	22	7	1000100
6	5	01101	23	7	1011000
7	5	10111	24	7	1011001
8	6	000011	25	8	10000010
9	6	011001	26	8	10001101
10	6	100110	27	8	10000011
11	6	100100	28	8	10001111
12	6	000010	29	8	10000000
13	6	100101	30	8	10000001
14	6	011000	31	8	10001110
15	6	100111	32	8	10001100
16	6	000001	ESC	4	0111
-	-	-	EOB	3	001

Table 28: Symbol Run VLC table used in Method 2 of coding the inverted termination bits of non-terminating patterns with 2 partitions at level 1 or higher (in luma plane where 32x32 tile size is used)

For example, in the luma plane for 32x32 pixel picture portions, the 5th group of termination bits (termination bits of non-terminating patterns at level 0 with 4 partitions) may be inverted and coded with the one of 3 methods from Table 29 which produces the smallest bit cost.

Method No.	**VLC Header**	**Description**
0	10	All bits are all 0
1	0	Proxy VLC coding using the event map shown in Table 30
2	11	Symbol Run with VLC shown in Table 31

Table 29: Methods and their header bits used in coding of termination bits of non-terminating patterns with 4 partitions at level 0 (in luma plane where 32x32 tile size is used)

Event	**VLC**
0000	0
0001	1000
0010	1001
0011	1010
0100	1011
0101	11000
0110	11001
0111	11010
1000	11011

1001	11100
1010	11101
1011	111100
1100	111101
1101	111110
1110	111111
1111	-

Table 30: Proxy VLC table used in Method 1 of coding the inverted termination bits of non-terminating patterns with 4 partitions at level 0 (in luma plane where 32x32 tile size is used)

5

Run	Len.	Code	Run	Len.	Code
0	2	11	17	6	101101
1	4	0001	18	6	100001
2	4	0101	19	7	0000001
3	4	0100	20	7	0000000
4	5	10101	21	7	1000101
5	5	10100	22	7	1000100
6	5	01101	23	7	1011000
7	5	10111	24	7	1011001
8	6	000011	25	8	10000010
9	6	011001	26	8	10001101
10	6	100110	27	8	10000011
11	6	100100	28	8	10001111
12	6	000010	29	8	10000000
13	6	100101	30	8	10000001
14	6	011000	31	8	10001110
15	6	100111	32	8	10001100
16	6	000001	ESC	4	0111
-	-	-	EOB	3	001

Table 31: Symbol Run VLC table used in Method 2 of coding the inverted termination bits of non-terminating patterns with 4 partitions at level 1 or higher (in luma plane where 32x32 tile size is used)

For example, in a chroma plane, the termination bits may be accumulated and grouped into 5 different groups as follows:

Group	Description
1st	Termination bits of non-terminating patterns at level 0 with 2 partitions
2nd	Termination bits of non-terminating patterns at level 1 (or higher) with 2 partitions
3rd	Termination bits of non-terminating patterns with 3 partitions (all levels)
4th	Termination bits of non-terminating patterns with 4 partitions (all levels)
5th	Termination bits of patterns with more than 4 partitions at any level

Table C: Example Chroma Plane Termination Bit Groupings

For example, in a chroma plane, the 1st group of termination bits (termination bits of non-terminating patterns at level 0 with 2 partitions) may be coded with the Proxy VLC coding using the event map shown in Table 32.

Event	VLC
00	-
01	10
10	11
11	0

Table 32: Proxy VLC table used in coding the termination bits of non-terminating patterns with 2 partitions at level 0 (in Chroma plane where 32x32 tile size is used (Luma 64x64))

For example, in a chroma plane, the 2nd group of termination bits (termination bits of non-terminating patterns at level 1 or higher with 2 partitions) may be coded with the one of 2 methods from Table 33 which produces the smallest bit cost.

Method No.	VLC Header	Description
0	0	Proxy VLC coding using the event map shown in Table 34
1	1	Proxy VLC coding using the event map shown in Table 35

Table 33: Methods and their header bits used in coding of termination bits of non-terminating patterns with 2 partitions at level 1 or higher (in Chroma plane where 32x32 tile size is used (Luma 64x64))

Event	VLC
00	-
01	0
10	10
11	11

Table 34: Proxy VLC table used in coding the termination bits of non-terminating patterns with 2 partitions at level 1 or higher (in Chroma plane)

Event	VLC

00	-
01	10
10	0
11	11

Table 35: Proxy VLC table used in coding the termination bits of non-terminating patterns with 2 partitions at level 1 or higher (in Chroma plane where 32x32 tile size is used (Luma 64x64))

For example, in a chroma plane, the 3rd group of termination bits (termination bits of non-terminating patterns with 3 partitions) is coded with the Proxy VLC coding using the event map shown in Table 36.

Event	**VLC**
000	-
001	010
010	011
011	100
100	00
101	101
110	110
111	111

Table 36: Proxy VLC table used in coding the termination bits of non-terminating patterns with 3 partitions (in Chroma plane where 32x32 tile size is used (Luma 64x64))

For example, in a chroma plane, the 4th group of termination bits (termination bits of non-terminating patterns with 4 partitions) may be coded with the Proxy VLC coding using the event map shown in Table 37.

Event	VLC
0000	-
0001	11111100
0010	11111101
0011	1111000
0100	11111110
0101	1111001
0110	1111011
0111	101
1000	11111111
1001	1111010
1010	1111100
1011	1110
1100	1111101
1101	110
1110	100
1111	0

Table 37: Proxy VLC table used in coding the termination bits of non-terminating patterns with 4 partitions (in Chroma plane where 32x32 tile size is used (Luma 64x64))

For example, in a chroma plane, the 5th group of termination bits (termination bits of non-terminating patterns at any level with more than 4 partitions) may be coded coded uncompressed (i.e., "as is" bitmap). For example, termination bits of non-terminating patterns at any level with more than 4 partitions may be a rare event.

As discussed, termination bits may be grouped the termination bits into a first group comprising termination bits for base level patterns having two partitions, a second group comprising termination bits for level one or higher patterns having two partitions, a third group comprising termination bits for base level patterns having three partitions, a fourth group comprising termination bits for level one or higher patterns having three partitions, a fifth group comprising termination bits for base level patterns having four partitions, a

sixth group comprising termination bits for level one or higher patterns having four partitions, and a seventh group comprising termination bits for base level or higher patterns having more than four partitions. The termination bits may be entropy coded by inverting at least one of the first through seventh groups of termination bits and entropy encoding the inverted group of termination bits based on a selected variable length coding table from a plurality of variable length coding tables giving the smallest bit cost for the inverted group of termination bits, generating a variable length coding table header associated with the selected variable length coding table, and writing the variable length coding table header and the entropy encoded inverted group of termination bits to the bitstream.

In some examples, chroma picture portion partitions may be coded based on luma picture portion partitions. For example, follow bits may describe whether chroma picture portion partition(s) may be copied from corresponding luma picture portion partitions or chroma picture portion partitions may explicitly specified (e.g., the chroma picture portion partitions are different and may not be copied). For example, a follow bit of '1' may indicate if a particular picture portion partitioning from luma can be copied for use by chroma and a follow bit with value of '0' may indicate that the partitioning information for the chroma picture portion partitioning is sent explicitly via the bitstream.

In some examples, U chroma plane follow bits may be directly coded with bitmap or symbol run VLC and V chroma plane follow bits may be predicted from the U plane with prediction differences (e.g., bits correcting the prediction of V chroma plane follow bits) may be coded with being coded with bitmap or symbol run VLC.

For example, input data for coding may include luma data and chroma data defining luma partitions and chroma partitions for a video frame. As discussed, follow bits may be generated for the video frame. The follow bits may indicate, for picture portions of the video frame whether the chroma partitions match the luma partitions (e.g., a bit of '1') or the chroma partitions do not match the luma partitions (e.g., a bit of '0'). A selected coding technique for the follow bits may be determined from a bitmap coding or a symbol run variable length coding and a follow bits selected coding technique header indicating coding technique may be generated. The follow bits may be entropy encoded based on the selected coding technique and written to a bitstream.

In some examples, input data may include luma data defining luma partitions and chroma data defining U plane chroma partitions and V plane chroma partitions for a video frame. U plane follow bits may be generated for the video frame such that the U plane

follow bits indicate, for picture portions of the video frame, whether the U plane chroma partitions match the luma partitions (e.g., a bit of '1') or the U plane chroma partitions do not match the luma partitions (e.g., a bit of '0'). Further, U plane follow bits may be generated for the video frame such that the V plane follow bits indicate, for picture portions of the video frame, whether the V plane chroma partitions match the luma partitions (e.g., a bit of '1') or the V plane chroma partitions do not match the luma partitions (e.g., a bit of '0'). A selected coding technique for the U plane follow bits may be determined from a bitmap coding or a symbol run variable length coding and a U plane follow bits selected coding technique header indicating coding technique may be generated. Further, V plane predicted follow bits may be determined based on the U plane follow bits. For example, the V plane predicted follow bits may match the U plane follow bits. The V plane follow bits and the V plane predicted follow bits may be differenced to generate V plane difference follow bits. A V plane difference follow bits selected coding technique for the V plane difference follow bits may be determined from a bitmap coding or a symbol run variable length coding and a V plane difference follow bits selected coding technique header indicating the selected coding technique may be generated. The U plane follow bits and V plane difference follow bits may be entropy encoded based on the selected coding techniques and the U plane follow bits selected coding technique header, the V plane difference follow bits selected coding technique header, the entropy encoded U plane follow bits, and the entropy encoded V plane difference follow bits may be written to a bitstream.

As discussed herein, patterns of multi-level patterns (e.g., base level patterns and higher level patterns) may be entropy coded based on a single (average) VLC table or selectively based on two (e.g., a low-complexity and high complexity) VLC tables (or more). When two or more tables are available, they may be selected automatically or adaptively. Based on the applicable VLC table or tables, an entropy coded codeword may be determined for each pattern (of the multi-level patterns) and the entropy coded codewords may be written to a bitstream. Table D provides a key for an example listing of VLC types. In Table D, column "Type of VLC" provides example VLC types, modes, and so on as discussed herein and column "T" provides a key for Tables 38(a)–(e) such that in Tables 38(a)–(e), the listing under "T" represents the description provided by the "Type of VLC" in Table D. The codebooks described herein may be implemented together or they may be implemented separately. Further, the implementation of a multi-level pattern as described may limit the number of entries within the codebooks as compared to a single

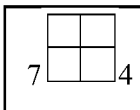
codebook. The provided exemplary codebook tables are presented for luma. In some examples, chroma tables having the same or similar numbers of entries may also be implemented.

T	Type of VLC
a	Average VLC Table for Terminating Patterns
b	Average VLC Table for Non-Terminating Patterns
c	Low-complexity VLC Table for Terminating Patterns (asw_mode off)
d	Low-complexity VLC Table for Non-Terminating Patterns (asw_mode off)
e	High-complexity VLC Table for Terminating Patterns (asw_mode off)
f	High-complexity VLC Table for Non-Terminating Patterns (asw_mode off)
g	Low-complexity VLC Table for Terminating Patterns (asw_mode on)
h	Low-complexity VLC Table for Non-Terminating Patterns (asw_mode on)
i	High-complexity VLC Table for Terminating Patterns (asw_mode on)
j	High-complexity VLC Table for Non-Terminating Patterns (asw_mode on)

Table D: Key for the following Tables 38(a)–(e) and example descriptions of Types of VLCs

In the following Tables 38(a)–(e), example codebooks or codebook portions are provided. In the examples, in the first column, visual patterns (e.g., of partitions) of the codebook are provided. To the bottom left of the visual pattern, an index value of the

visual pattern is provided and to the bottom right, a number of partitions is provided. In the other columns, example, entropy coded codewords are provided corresponding to the "Type of VLC" provided in Table D (e.g., "a" corresponds to Type of VLC "Average VLC Table for Terminating Patterns", "b" corresponds to Type of VLC "Average VLC Table for Non-Terminating Patterns", and so on). In the tables, the header "T" represents "Type of VLC". Further, a value under a header of "L" is given, which is the length of the codeword. The codebooks provided are exemplary and, in some cases partial for the sake of brevity.



a	1	0		b	1	1
c	1	0		d	1	1
e	1	0		f	1	1
g	1	0		h	1	1
i	1	0		j	1	1

Table 38(a): Luma 64x64 Picture Portions Codebook for 64x64 Patterns

h/v pattern	h/v pattern code					
	T	L	Code	T	L	Code
0 1	a	18	001000000000000110	b	19	0010000000000000100
	c	15	111010010001010	d	16	1110100100000101
	e	15	010001100000010	f	15	010001100011111
	g	17	01100000100010010	h	16	0110000010001111
	i	17	00011100000001010	j	17	00011100000010110
 1 2	a	5	11010	b	5	01111
	c	5	01111	d	5	00011
	e	7	0100000	f	6	110000
	g	5	01000	h	5	00000
	i	17	00011100000001011	j	9	000111001
 2 2	a	7	0010001	b	6	011101
	c	6	100000	d	6	011101
	e	9	010001101	f	7	0100110
	g	6	110000	h	5	10011
	i	17	00011100000001000	j	17	00011100000010111
 3 3	a	7	0111001	b	5	01001
	c	6	001000	d	4	1111
	e	8	01000111	f	5	11001
	g	9	001010000	h	5	00011
	i	17	00011100000001001	j	6	000011
 4 3	a	7	1001010	b	5	00101
	c	6	010100	d	4	1101

	e	15	010001100000011	f	5	01101
	g	9	001000010	h	5	00001
	i	17	00011100000001110	j	5	10000
 5 3	a	11	00100000001	b	7	0000000
	c	9	111010011	d	7	1110001
	e	15	010001100001010	f	6	000010
	g	10	0110000011	h	6	100100
	i	17	00011100000001111	j	6	110101
 6 3	a	14	00100000000001	b	6	111011
	c	15	111010010001011	d	6	111001
	e	15	010001100001011	f	6	000011
	g	12	011000001001	h	6	101001
	i	17	00011100000001100	j	7	1100000
 7 4	a	10	0010000001	b	4	0011
	c	8	10011000	d	4	0011
	e	15	010001100001000	f	3	100
	g	17	01100000100010011	h	4	0111
	i	17	00011100000001101	j	3	111
 8 2	a	7	0110011	b	8	01100001
	c	6	100111	d	9	100100001
	e	15	010001100001001	f	6	110001
	g	7	1000000	h	7	1110001
	i	17	00011100000101010	j	17	00011100000010100
 9 2	a	7	0100000	b	8	10010001
	c	6	100101	d	9	101100000
	e	15	010001100001110	f	15	010001100011100
	g	7	0001000	h	7	1000001
	i	17	00011100000101011	j	11	00000000010
 10 2	a	18	001000000000000111	b	8	11110001
	c	16	1110100100001010	d	8	01100000
	e	15	010001100001111	f	15	010001100011101
	g	17	01100000100010000	h	9	001000000
	i	17	00011100000101000	j	17	00011100000010101
 11 2	a	18	001000000000001010	b	9	100100000
	c	16	1110100100001011	d	8	01100001
	e	15	010001100001100	f	15	010001100010010
	g	17	01100000100010001	h	10	1010001001
	i	17	00011100000101001	j	18	000111000000000010
 12 2	a	18	001000000000001011	b	19	0010000000000000101
	c	16	1110100100001000	d	15	111010010001110
	e	15	010001100001101	f	15	010001100010011
	g	17	01100000100001010	h	16	0110000010001100
	i	17	00011100000101110	j	18	000111000000000011
 13 2	a	18	001000000000001000	b	19	0010000000000001000
	c	16	1110100100001001	d	15	111010010001111
	e	15	010001100101010	f	15	010001100010000
	g	17	01100000100001011	h	16	0110000010001101
	i	17	00011100000101111	j	18	000111000000000000
 14 2	a	8	10010010	b	8	10010011
	c	8	00010100	d	9	000100011
	e	15	010001100101011	f	15	010001100010001

	g	7	0110100	h	7	0101001
	i	17	00011100000101100	j	11	00000000011
 15 2	a	7	1110101	b	9	111010001
	c	7	0100000	d	9	000100000
	e	7	0100001	f	15	010001100010110
	g	7	0010010	h	9	010100000
	i	17	00011100000101101	j	18	000111000000000001
 16 2	a	18	001000000000001001	b	11	00100000101
	c	16	1110100100001110	d	9	111010101
	e	15	010001100101000	f	15	010001100010111
	g	17	01100000100001000	h	9	111110101
	i	17	00011100000100010	j	18	000111000000000110
 17 2	a	18	001000000000001110	b	10	0110000000
	c	16	1110100100001111	d	10	1110111000
	e	15	010001100101001	f	15	010001100010100
	g	17	01100000100001001	h	9	010100001
	i	17	00011100000100011	j	18	000111000000000111
 18 2	a	18	001000000000001111	b	19	0010000000000001001
	c	16	1110100100001100	d	15	111010010001100
	e	15	010001100101110	f	15	010001100010101
	g	17	01100000100001110	h	17	01100000100010100
	i	17	00011100000100000	j	18	000111000000000100
 19 2	a	18	001000000000001100	b	18	001000000000000101
	c	16	1110100100001101	d	15	111010010001101
	e	15	010001100101111	f	15	010001100000110
	g	17	01100000100001111	h	17	01100000100010101
	i	17	00011100000100001	j	18	000111000000000101
 20 3	a	8	01100100	b	5	00001
	c	7	0111000	d	5	00101
	e	15	010001100101100	f	5	00111
	g	8	11111001	h	4	1101
	i	17	00011100000100110	j	5	10001
 21 3	a	7	0110001	b	5	11000
	c	6	110000	d	5	10001
	e	15	010001100101101	f	6	011100
	g	7	0001001	h	5	01011
	i	17	00011100000100111	j	6	110010
 22 3	a	7	1110000	b	5	11111
	c	8	00010101	d	5	10101
	e	15	010001100100010	f	9	001100001
	g	6	111111	h	5	11001
	i	17	00011100000100100	j	7	1100001
 23 3	a	8	00000101	b	5	10000
	c	7	1011011	d	5	01001
	e	15	010001100100011	f	6	011101
	g	7	0110101	h	5	00110
	i	17	00011100000100101	j	6	000001
 24 3	a	8	01100101	b	5	10101
	c	7	1110000	d	5	10111
	e	15	010001100100000	f	5	11100
	g	8	00101001	h	6	001011

	i	11	00011100001	j	5	11011
 25 3	a	7	1111001	b	5	10011
	c	7	1001101	d	5	01011
	e	15	010001100100001	f	6	000000
	g	7	0110001	h	5	00111
	i	17	00011100000111010	j	6	000100

Table 38(b): Luma 64x64 Picture Portions Codebook for 64x64 Patterns

h/v pattern	h/v pattern code					
	T	L	Code	T	L	Code
 0 2	a	6	010110	b	5	11001
	c	6	010010	d	5	11100
	e	11	11010010010	f	12	110100011000
	g	6	000011	h	5	01001
	i	14	00000100001010	j	14	00000100011001
 1 2	a	9	001000011	b	6	110000
	c	6	110110	d	6	101000
	e	11	11010010011	f	12	110100011001
	g	6	111100	h	7	0100001
	i	14	00000100001011	j	7	0000110
 2 2	a	4	1011	b	15	001000010000101
	c	4	0110	d	14	11000100000110
	e	12	110100100010	f	12	110100011110
	g	4	0101	h	14	11111010000110
	i	7	0000011	j	14	00000100011110
 3 2	a	4	1000	b	16	0010000100000000
	c	4	1000	d	14	11000100000111
	e	12	110100100011	f	12	110100011111
	g	4	1000	h	14	11111010000111
	i	9	000001001	j	14	00000100011111
 4 2	a	4	1001	b	4	0111
	c	4	0111	d	4	0101
	e	12	110100100000	f	3	111
	g	4	0011	h	4	1101
	i	7	0000000	j	5	01100
 5 2	a	4	1101	b	4	0011
	c	4	1001	d	4	0001
	e	6	110101	f	3	010
	g	4	1001	h	4	0111
	i	7	0000001	j	4	0001
 6 2	a	15	001000010000010	b	5	00101
	c	14	11000100001010	d	5	00001
	e	12	110100100001	f	5	00000
	g	13	1111101000110	h	5	00101
	i	14	00000100001000	j	5	00100

	a	15	001000010000011	b	5	00001
	c	14	11000100001011	d	4	1011
7 2	e	11	11010011010	f	4	0011
	g	13	1111101000111	h	5	11100
	i	14	00000100001001	j	4	0101

Table 38(c): Luma 64x64 Tiles Codebook for 32x16 Patterns (partial)

h/v pattern	h/v pattern code					
	T	L	Code	T	L	Code
	a	4	0011	b	16	0101000000000101
	c	3	101	d	15	011100000000100
	e	5	01010	f	14	00001100000101
0 2	g	3	011	h	14	10000100000100
	i	6	000001	j	16	0101000000000101
	a	5	00011	b	4	1110
	c	4	0001	d	4	0011
	e	5	10011	f	4	1100
1 2	g	4	1001	h	4	1101
	i	7	0100001	j	5	01001
	a	6	001000	b	6	000100
	c	4	1101	d	5	01100
	e	9	000011001	f	7	0000101
2 2	g	5	00110	h	5	10001
	i	10	0101000001	j	6	101001
	a	5	10111	b	5	10100
	c	5	00100	d	5	00101
	e	5	10000	f	5	01011
3 2	g	4	1100	h	4	1110
	i	7	0000001	j	6	010001
	a	17	01010000000000010	b	7	1011001
	c	14	01110000000110	d	15	011100000000101
	e	13	0000110001010	f	7	0010000
4 2	g	14	10000100001010	h	7	1000001
	i	15	010100000000110	j	7	0101001
	a	17	01010000000000011	b	9	010100001
	c	14	01110000000111	d	15	011100000001000
	e	13	0000110001011	f	13	0000110000110
5 2	g	14	10000100001011	h	14	10000100000101
	i	15	010100000000111	j	8	10100001
	a	17	01010000000000000	b	17	01010000000000100
	c	15	011100000000010	d	15	011100000001001
	e	13	0000110001000	f	13	0000110000111
6 2	g	14	10000100001000	h	14	10000100001100
	i	16	0101000000000010	j	16	0101000000001000

	a	17	01010000000000001	b	17	01010000000000101
	c	15	011100000000011	d	14	01110000000101
	e	13	0000110001001	f	13	0000110000100
7 2	g	14	10000100001001	h	14	10000100001101
	i	16	0101000000000011	j	16	0101000000001001

Table 38(d): Luma 64x64 Tiles Codebook for 32x8 Patterns (partial)

h/v pattern	h/v pattern code					
	T	L	Code	T	L	Code
	a	5	11000	b	14	01000000100010
	c	5	00010	d	13	1000000000000
	e	6	000000	f	12	000011001100
	g	5	00001	h	13	1110011000000
0 2	i	7	0001000	j	14	01001100000011
	a	5	01001	b	14	01000000100011
	c	5	00001	d	13	1000000000001
	e	6	000001	f	12	000011001101
	g	5	10000	h	13	1110011000001
1 2	i	5	00011	j	14	01001100000000
	a	8	00000000	b	6	010001
	c	7	1000001	d	5	11000
	e	8	00001101	f	5	00101
	g	9	111001101	h	5	10001
2 3	i	8	00000000	j	7	1000001
	a	7	0000100	b	6	010110
	c	6	000001	d	5	11001
	e	6	000110	f	6	000101
	g	8	11100111	h	6	110000
3 3	i	7	1000000	j	8	01001101
	a	8	01000001	b	7	0000001
	c	7	0100010	d	13	1000000000110
	e	7	0000111	f	12	000011000010
	g	8	11100100	h	6	110001
4 3	i	13	0100110000010	j	6	000101
	a	7	0101000	b	7	0101001
	c	9	100000001	d	8	10000100
	e	7	0000100	f	7	0011000
	g	7	0000001	h	5	11101
5 3	i	6	100001	j	7	0000001
	a	10	0100000011	b	6	010111
	c	13	1000000000010	d	8	10000101
	e	12	000011001010	f	12	000011000011
	g	8	11100101	h	6	000001
6 4	i	10	0100110001	j	7	0001001

	a	4	0001	b	14	01000000100000
	c	3	111	d	13	1000000000111
	e	5	00111	f	12	000011000000
	g	3	101	h	13	1110011000110
7 2	i	4	0110	j	14	01001100000001

Table 38(e): Luma 64x64 Tiles Codebook for 32x8 Patterns (partial)

As discussed with respect to FIG. 16, example video coding system 1600 may include imaging device(s) 1601, video encoder 100 and/or a video encoder implemented via logic circuitry 1650 of processing unit(s) 1620, video decoder 200 and/or a video decoder implemented via logic circuitry 1650 of processing unit(s) 1620, an antenna 1602, one or more processor(s) 1603, one or more memory store(s) 2004, and/or a display device 2005.

In some examples, video encoder 100 implemented via logic circuitry 1650 may include an image buffer (e.g., via either processing unit(s) 1620 or memory store(s) 1604)) and a graphics processing unit (e.g., via processing unit(s) 1620). The graphics processing unit may be communicatively coupled to the image buffer. The graphics processing unit may include video encoder 100 as implemented via logic circuitry 1650 to embody the various modules as discussed with respect to FIG. 1 and FIGS. 4, 6 and others herein. For example, the graphics processing unit may include entropy encoder logic circuitry, and so on. The logic circuitry may be configured to perform the various operations as discussed herein. For example, the entropy encoder logic circuitry may be configured to load input data defining partitions of picture portions of a video frame, determine a multi-level pattern for a first picture portion such that the multi-level pattern may include a base pattern selected from a plurality of available patterns and at least one level one pattern selected from the plurality of available patterns for at least one non-terminating portion of the base pattern, determine termination bits indicating terminations of pattern levels of the picture portions of the video frame such that the termination bits may include first termination bits associated with the base pattern of the multi-level pattern and the first picture portion, determine a first entropy coded codeword associated with the base pattern and a second entropy coded codeword associated with the level one pattern, entropy encode the termination bits, and write the first entropy coded codeword, the second entropy coded codeword, and the entropy coded termination bits to a bitstream.

In some examples, antenna 1602 of video coding system 1600 may be configured to receive an entropy encoded bitstream of video data. As discussed, the bitstream may include compressed video data of various types. Video coding system 1600 may also include video decoder 200 coupled to antenna 1602 and configured to decode the encoded bitstream. For example, video decoder 200 may be configured to decode the encoded bitstream to determine a base pattern of a multi-level pattern for a first picture portion of a video frame based on a first entropy coded codeword in the encoded bitstream and a level one pattern of the multi-level pattern based on a second entropy coded codeword in the encoded bitstream, and termination bits indicating terminations of pattern levels of picture portions of the video frame, and reconstruct first partitions for the first picture portion based on the decoded base pattern, level one pattern, and termination bits.

While certain features set forth herein have been described with reference to various implementations, this description is not intended to be construed in a limiting sense. Hence, various modifications of the implementations described herein, as well as other implementations, which are apparent to persons skilled in the art to which the present disclosure pertains are deemed to lie within the spirit and scope of the present disclosure.

The following examples pertain to further embodiments.

In one example, a computer-implemented method for video coding may include loading input data defining partitions of picture portions of a video frame, determining a multi-level pattern for a first picture portion such that the multi-level pattern may include a base pattern selected from a plurality of available patterns and at least one level one pattern selected from the plurality of available patterns for at least one non-terminating portion of the base pattern, determining termination bits indicating terminations of pattern levels of the picture portions of the video frame such that the termination bits may include first termination bits associated with the base pattern of the multi-level pattern and the first picture portion, determining a first entropy coded codeword associated with the base pattern and a second entropy coded codeword associated with the level one pattern, entropy encoding the termination bits, and writing the first entropy coded codeword, the second entropy coded codeword, and the entropy coded termination bits to a bitstream.

In another example, a computer-implemented method for video coding may further include determining a variable length code tables selection mode for at least one of the video frame, a slice of the video frame, or a sequence of video frames from at least one of a first variable length code tables selection mode where a single variable length coding

table is used for determining entropy coded codewords for every picture portion of the video frame and a second variable length code tables selection mode where a picture portion-based variable length coding table is selected for each picture portion of the video frame from two or more available variable length coding tables and the selected picture portion-based variable length coding table is used for determining entropy coded codewords for the associated picture portion, and grouping the termination bits into a first group comprising termination bits for base level patterns having two partitions, a second group comprising termination bits for level one or higher patterns having two partitions, a third group comprising termination bits for base level patterns having three partitions, a fourth group comprising termination bits for level one or higher patterns having three partitions, a fifth group comprising termination bits for base level patterns having four partitions, a sixth group comprising termination bits for level one or higher patterns having four partitions, and a seventh group comprising termination bits for base level or higher patterns having more than four partitions. Entropy encoding the termination bits may include inverting at least one of the first through seventh groups of termination bits and entropy encoding the inverted group of termination bits based on a selected variable length coding table from a plurality of variable length coding tables giving the smallest bit cost for the inverted group of termination bits, generating a variable length coding table header associated with the selected variable length coding table, and writing the variable length coding table header and the entropy encoded inverted group of termination bits to the bitstream. A first variable length coding table and a second variable length coding table may be selectively available for determining entropy coded codewords for picture portions of the video frame, and the method may further include determining a selected switching mode for switching between the first and second variable length coding tables for the picture portions of the video frame from at least one of an automatic switching mode or an adaptive switching mode, and such that the adaptive switching mode is the selected switching mode if a quantizer value is less than a predetermined threshold. A first variable length coding table and a second variable length coding table may be selectively available for determining entropy coded codewords for picture portions of the video frame such that a first selected variable length coding table from at least one of the first variable length coding table or the second variable length coding table may be determined for the first picture portion automatically based on a prediction of complexity for the first picture portion. A high complexity variable length coding table and a low complexity variable length coding table may be selectively available for determining entropy coded codewords for picture portions of the video frame such that a first selected variable length coding table

from the high complexity variable length coding table and the low complexity variable length coding table may be determined for the first picture portion automatically based on a prediction of complexity for the first picture portion. The prediction of complexity for the first picture portion may include determining a number of splits for picture portions neighboring the first picture portion and using the high complexity variable length coding table when the number of splits is above a threshold and the low complexity variable length coding table when the number of splits is at or below the threshold. A first variable length coding table and a second variable length coding table may be selectively available for determining entropy coded codewords for picture portions of the video frame such that a first selected variable length coding table from the first variable length coding table and the second variable length coding table may be determined for the first picture portion, and the method may further include determining a selected variable length coding table from the first variable length coding table and the second variable length coding table for picture portions of the video frame, generating an adaptive switching mask for the video frame such that the adaptive switching mask may include indicators indicating the selected variable length coding tables for the picture portions, and entropy encoding and writing the adaptive switching mask to the bitstream. The second entropy coded codeword may indicate the level one pattern is a terminating pattern. The picture portions may be 64x64 pixels, and the method may further include generating a binary mask of picture portions comprising a plurality of binary indicators each indicating whether the associated picture portion comprises four 32x32 non-terminating partitions or four 32x32 partitions with at least one terminating 32x32 partition, determining whether to encode the binary mask of picture portions using a symbol run coding method or a bitmap coding method, generating a binary mask of picture portions coding method indicator, entropy encoding the binary mask of picture portions based on the selected coding method, and writing the binary mask of picture portions coding method indicator and the entropy encoded binary mask of picture portions to the bitstream. The input data may include luma data and chroma data defining luma partitions and chroma partitions, and the method may further include generating follow bits for the video frame such that the follow bits indicate, for picture portions of the video frame, at least one of the chroma partitions match the luma partitions or the chroma partitions do not match the luma partitions, determining a follow bits selected coding technique for the follow bits from at least one of a bitmap coding or a symbol run variable length coding, generating a follow bits selected coding technique header indicating at least one of the bitmap coding or the symbol run variable length coding, entropy encoding the follow bits based on the selected coding technique, and

writing the follow bits selected coding technique and the entropy encoded follow bits to the bitstream. The input data may include luma data defining luma partitions and chroma data defining U plane chroma partitions and V plane chroma partitions, and the method may further include generating U plane follow bits for the video frame such that the U plane follow bits indicate, for picture portions of the video frame, at least one of the U plane chroma partitions match the luma partitions or the U plane chroma partitions do not match the luma partitions, generating V plane follow bits for the video frame such that the V plane follow bits indicate, for picture portions of the video frame, at least one of the V plane chroma partitions match the luma partitions or the V plane chroma partitions do not match the luma partitions, determining a U plane follow bits selected coding technique for the U plane follow bits from at least one of a bitmap coding or a symbol run variable length coding and a U plane follow bits selected coding technique header indicating at least one of the bitmap coding or the symbol run variable length coding, generating V plane predicted follow bits based at least in part on the U plane follow bits, differencing the V plane follow bits and the V plane predicted follow bits to generate V plane difference follow bits, determining a V plane difference follow bits selected coding technique for the V plane difference follow bits from at least one of a bitmap coding or a symbol run variable length coding and a V plane difference follow bits selected coding technique header indicating at least one of the bitmap coding or the symbol run variable length coding, entropy encoding the U plane follow bits and V plane difference follow bits, based on the selected coding techniques, and writing the U plane follow bits selected coding technique header, the V plane difference follow bits selected coding technique header, the entropy encoded U plane follow bits, and the entropy encoded V plane difference follow bits to the bitstream. The input data may include at least one of luma data or chroma data. The video frame may include an I-picture, and wherein the first picture portion comprises a tile of the video frame. The video frame may include an I-picture and the first picture portion may include a tile of the video frame, and wherein the partitions comprise k-d tree partitions. The video frame may include at least one of a P-picture or a B/F-picture, and the picture portion may include a partition of a tile of the video frame. The partitions may include at least one of k-d tree partitions, bi-tree partitions, quad-tree partitions, or structured codebook partitions. The picture portion may include at least one of a tile, a largest coding unit, or a super block. The termination bits may be associated with at least one of the video frame, a slice of the video frame, or a sequence of video frames. The plurality of available patterns may include a codebook of available patterns.

In other examples, a video encoder may include an image buffer and a graphics processing unit having entropy encoder logic circuitry. The graphics processing unit may be communicatively coupled to the image buffer and the entropy encoder logic circuitry may be configured to load input data defining partitions of picture portions of a video frame, determine a multi-level pattern for a first picture portion such that the multi-level pattern may include a base pattern selected from a plurality of available patterns and at least one level one pattern selected from the plurality of available patterns for at least one non-terminating portion of the base pattern, determine termination bits indicating terminations of pattern levels of the picture portions of the video frame such that the termination bits comprise first termination bits associated with the base pattern of the multi-level pattern and the first picture portion, determine a first entropy coded codeword associated with the base pattern and a second entropy coded codeword associated with the level one pattern, entropy encode the termination bits, and write the first entropy coded codeword, the second entropy coded codeword, and the entropy coded termination bits to a bitstream.

In a further example video encoder, the entropy encoder logic circuitry may be further configured to determine a variable length code tables selection mode for at least one of the video frame, a slice of the video frame, or a sequence of video frames from at least one of a first variable length code tables selection mode where a single variable length coding table is used for determining entropy coded codewords for every picture portion of the video frame and a second variable length code tables selection mode where a picture portion-based variable length coding table is selected for each picture portion of the video frame from two or more available variable length coding tables and the selected picture portion-based variable length coding table is used for determining entropy coded codewords for the associated picture portion, determine a selected switching mode for switching between the first and second variable length coding tables for the picture portions of the video frame from at least one of an automatic switching mode or an adaptive switching mode such that the adaptive switching mode is the selected switching mode if a quantizer value is less than a predetermined threshold, and group the termination bits into a first group comprising termination bits for base level patterns having two partitions, a second group comprising termination bits for level one or higher patterns having two partitions, a third group comprising termination bits for base level patterns having three partitions, a fourth group comprising termination bits for level one or higher patterns having three partitions, a fifth group comprising termination bits for base level

patterns having four partitions, a sixth group comprising termination bits for level one or higher patterns having four partitions, and a seventh group comprising termination bits for base level or higher patterns having more than four partitions. To entropy encode the termination bits may include the entropy encoder logic being configured to invert at least one of the first through seventh groups of termination bits and entropy encoding the inverted group of termination bits based on a selected variable length coding table from a plurality of variable length coding tables giving the smallest bit cost for the inverted group of termination bits, generate a variable length coding table header associated with the selected variable length coding table, and write the variable length coding table header and the entropy encoded inverted group of termination bits to the bitstream. A high complexity variable length coding table and a low complexity variable length coding table may be selectively available for determining entropy coded codewords for picture portions of the video frame, and the entropy encoder logic circuitry may be configured to determine a first selected variable length coding table from the high complexity variable length coding table and the low complexity variable length coding table for the first picture portion automatically based on a prediction of complexity for the first picture portion, and such that to determine the prediction of complexity for the first picture portion comprises the entropy encoder logic being configured to determine a number of splits for picture portions neighboring the first picture portion and using the high complexity variable length coding table when the number of splits is above a threshold and the low complexity variable length coding table when the number of splits is at or below the threshold. A first variable length coding table and a second variable length coding table may be selectively available for determining entropy coded codewords for picture portions of the video frame, and the entropy encoder logic circuitry may be further configured to determine a selected variable length coding table from the first variable length coding table and the second variable length coding table for picture portions of the video frame, generate an adaptive switching mask for the video frame such that the adaptive switching mask comprises indicators indicating the selected variable length coding tables for the picture portions, and entropy encode and write the adaptive switching mask to the bitstream. The second entropy coded codeword may indicate the level one pattern is a terminating pattern. The picture portions may include 64x64 pixels, and the entropy encoder logic circuitry may be further configured to generate a binary mask of picture portions comprising a plurality of binary indicators each indicating whether the associated picture portion comprises four 32x32 non-terminating partitions or four 32x32 partitions with at least one terminating 32x32 partition, determine whether to encode the binary mask of picture portions using a symbol

run coding method or a bitmap coding method, generate a binary mask of picture portions coding method indicator, entropy encode the binary mask of picture portions based on the selected coding method, and write the binary mask of picture portions coding method indicator and the entropy encoded binary mask of picture portions to the bitstream. The input data may include luma data and chroma data defining luma partitions and chroma partitions, and the entropy encoder logic circuitry may be further configured to generate follow bits for the video frame such that the follow bits indicate, for picture portions of the video frame, at least one of the chroma partitions match the luma partitions or the chroma partitions do not match the luma partitions, determine a follow bits selected coding technique for the follow bits from at least one of a bitmap coding or a symbol run variable length coding, generate a follow bits selected coding technique header indicating at least one of the bitmap coding or the symbol run variable length coding, entropy encode the follow bits based on the selected coding technique, and write the follow bits selected coding technique and the entropy encoded follow bits to the bitstream. The input data may include luma data defining luma partitions and chroma data defining U plane chroma partitions and V plane chroma partitions, and the entropy encoder logic circuitry is further configured to generate U plane follow bits for the video frame such that the U plane follow bits indicate, for picture portions of the video frame, at least one of the U plane chroma partitions match the luma partitions or the U plane chroma partitions do not match the luma partitions, generate V plane follow bits for the video frame such that the V plane follow bits indicate, for picture portions of the video frame, at least one of the V plane chroma partitions match the luma partitions or the V plane chroma partitions do not match the luma partitions, determine a U plane follow bits selected coding technique for the U plane follow bits from at least one of a bitmap coding or a symbol run variable length coding and a U plane follow bits selected coding technique header indicating at least one of the bitmap coding or the symbol run variable length coding, generate V plane predicted follow bits based at least in part on the U plane follow bits, difference the V plane follow bits and the V plane predicted follow bits to generate V plane difference follow bits, determine a V plane difference follow bits selected coding technique for the V plane difference follow bits from at least one of a bitmap coding or a symbol run variable length coding and a V plane difference follow bits selected coding technique header indicating at least one of the bitmap coding or the symbol run variable length coding, entropy encode the U plane follow bits and V plane difference follow bits, based on the selected coding techniques, and write the U plane follow bits selected coding technique header, the V plane difference follow bits selected coding technique header, the entropy encoded U plane

follow bits, and the entropy encoded V plane difference follow bits to the bitstream. The video frame may include an I-picture, and wherein the first picture portion comprises a tile of the video frame. The video frame may include an I-picture and the first picture portion may include a tile of the video frame, and wherein the partitions comprise k-d tree partitions. The video frame may include at least one of a P-picture or a B/F-picture, and the picture portion may include a partition of a tile of the video frame. The partitions may include at least one of k-d tree partitions, bi-tree partitions, quad-tree partitions, or structured codebook partitions. The picture portion may include at least one of a tile, a largest coding unit, or a super block. The termination bits may be associated with at least one of the video frame, a slice of the video frame, or a sequence of video frames. The plurality of available patterns may include a codebook of available patterns.

In yet another example, a system may include a video decoder configured to decode an encoded bitstream. The video decoder is configured to decode the encoded bitstream to determine a base pattern of a multi-level pattern for a first picture portion of a video frame based on a first entropy coded codeword in the encoded bitstream and a level one pattern of the multi-level pattern based on a second entropy coded codeword in the encoded bitstream, and termination bits indicating terminations of pattern levels of picture portions of the video frame, and reconstruct first partitions for the first picture portion based on the decoded base pattern, level one pattern, and termination bits.

In a further example system, the system may also include an antenna configured to receive the entropy encoded bitstream and a display device configured to present video frames. To decode the encoded bitstream may further include the video decoder being configured to determine variable length code tables selection mode indicator for at least one of the video frame, a slice of the video frame, or a sequence of video frames indicating a variable length code tables selection mode from at least one of a first variable length code tables selection mode where a single variable length coding table is used to entropy decode entropy coded codewords for every picture portion of the video frame and a second variable length code tables selection mode where a picture portion-based variable length coding table is selected for each picture portion of the video frame from two or more variable length coding tables and the selected picture portion-based variable length coding table is used to entropy decode entropy coded codewords for the associated picture portion. A first variable length coding table and a second variable length coding table may be selectively available to entropy decode entropy coded codewords for picture portions of the video frame, and to decode the encoded bitstream may further include the video

decoder being configured to determine a selected switching mode for switching between the first and second variable length coding tables for the picture portions of the video frame from at least one of an automatic switching mode or an adaptive switching mode. A first variable length coding table and a second variable length coding table may be selectively available to entropy decode entropy coded codewords for picture portions of the video frame, and the video decoder may be further configured to determine a first selected variable length coding table from at least one of the high complexity variable length coding table or the low complexity variable length coding table automatically based on a prediction of complexity for the first picture portion sch that the prediction of complexity for the first picture portion may include the video decoder being configured to determine a number of splits for picture portions neighboring the first picture portion and use the high complexity variable length coding table when the number of splits is above a threshold and the low complexity variable length coding table when the number of splits is at or below the threshold. A first variable length coding table and a second variable length coding table may be selectively available to entropy decode entropy coded codewords for picture portions of the video frame, and to decode the encoded bitstream may further include the video decoder being configured to determine an adaptive switching mask for the video frame such that the adaptive switching mask comprises an indication of the first selected variable length coding table for the first picture portion. The second entropy coded codeword may indicate the selected level one pattern is a terminating pattern. The first picture portion may include 64x64 pixels, and to decode the encoded bitstream may further include the video decoder being configured to determine a binary mask of picture portions comprising a plurality of binary indicators each indicating whether the associated picture portions comprises four 32x32 non-terminating partitions or four 32x32 partitions with at least one terminating 32x32 partition. The video decoder may be further configured to entropy decode the termination bits based on a grouping of the termination bits into a first group comprising termination bits for base level patterns having two partitions, a second group comprising termination bits for level one or higher patterns having two partitions, a third group comprising termination bits for base level patterns having three partitions, a fourth group comprising termination bits for level one or higher patterns having three partitions, a fifth group comprising termination bits for base level patterns having four partitions, a sixth group comprising termination bits for level one or higher patterns having four partitions, and a seventh group comprising termination bits for base level or higher patterns having more than four partitions and entropy decode the termination bits based on a grouping of the termination bits, wherein to entropy decode the termination bits

comprises the video decoder being configured to determine a variable length coding table header associated with a first group of termination bits, and decode the first group of termination bits based on a selected variable length coding table based on the variable length coding table header. The multi-level pattern may include a luma pattern, and to decode the encoded bitstream may further include the video decoder being configured to determine follow bits for the video frame, and the video decoder may be further configured to determine a chroma pattern for the first picture portion matches the luma pattern based at least in part on the decoded follow bits.

In a further example, at least one machine readable medium may include a plurality of instructions that in response to being executed on a computing device, causes the computing device to perform the method according to any one of the above examples.

In a still further example, an apparatus may include means for performing the methods according to any one of the above examples.

The above examples may include specific combination of features. However, such the above examples are not limited in this regard and, in various implementations, the above examples may include the undertaking only a subset of such features, undertaking a different order of such features, undertaking a different combination of such features, and/or undertaking additional features than those features explicitly listed. For example, all features described with respect to the example methods may be implemented with respect to the example apparatus, the example systems, and/or the example articles, and *vice versa*.

WHAT IS CLAIMED:

1. A computer-implemented method for video coding, comprising:

loading input data defining partitions of picture portions of a video frame;

determining a multi-level pattern for a first picture portion, wherein the multi-level pattern comprises a base pattern selected from a plurality of available patterns and at least one level one pattern selected from the plurality of available patterns for at least one non-terminating portion of the base pattern;

determining termination bits indicating terminations of pattern levels of the picture portions of the video frame, wherein the termination bits comprise first termination bits associated with the base pattern of the multi-level pattern and the first picture portion;

determining a first entropy coded codeword associated with the base pattern and a second entropy coded codeword associated with the level one pattern;

entropy encoding the termination bits; and

writing the first entropy coded codeword, the second entropy coded codeword, and the entropy coded termination bits to a bitstream.

2. The method of claim 1, further comprising:

determining a variable length code tables selection mode for at least one of the video frame, a slice of the video frame, or a sequence of video frames from at least one of a first variable length code tables selection mode where a single variable length coding table is used for determining entropy coded codewords for every picture portion of the video frame and a second variable length code tables selection mode where a picture portion-based variable length coding table is selected for each picture portion of the video frame from two or more available variable length coding tables and the selected picture portion-based variable length coding table is used for determining entropy coded codewords for the associated picture portion.

3. The method of claim 1, wherein a first variable length coding table and a second variable length coding table are selectively available for determining entropy coded codewords for picture portions of the video frame, and wherein the method further comprises:

determining a selected switching mode for switching between the first and second variable length coding tables for the picture portions of the video frame from at least one of an automatic switching mode or an adaptive switching mode.

4. The method of claim 1, wherein a first variable length coding table and a second variable length coding table are selectively available for determining entropy coded codewords for picture portions of the video frame, and wherein the method further comprises:

determining a selected switching mode for switching between the first and second variable length coding tables for the picture portions of the video frame from at least one of an automatic switching mode or an adaptive switching mode, and wherein the adaptive switching mode is the selected switching mode if a quantizer value is less than a predetermined threshold.

5. The method of claim 1, wherein a first variable length coding table and a second variable length coding table are selectively available for determining entropy coded codewords for picture portions of the video frame, and wherein a first selected variable length coding table from at least one of the first variable length coding table or the second variable length coding table is determined for the first picture portion automatically based on a prediction of complexity for the first picture portion.

6. The method of claim 1, wherein a high complexity variable length coding table and a low complexity variable length coding table are selectively available for determining entropy coded codewords for picture portions of the video frame, and wherein a first selected variable length coding table from the high complexity variable length coding table and the low complexity variable length coding table is determined for the first picture portion automatically based on a prediction of complexity for the first picture portion, and wherein the prediction of complexity for the first picture portion comprises determining a number of splits for picture portions

neighboring the first picture portion and using the high complexity variable length coding table when the number of splits is above a threshold and the low complexity variable length coding table when the number of splits is at or below the threshold.

7. The method of claim 1, wherein a first variable length coding table and a second variable length coding table are selectively available for determining entropy coded codewords for picture portions of the video frame, wherein a first selected variable length coding table from the first variable length coding table and the second variable length coding table is determined for the first picture portion, and wherein the method further comprises:

generating an adaptive switching mask for the video frame, wherein the adaptive switching mask comprises an indication of the first selected variable length coding table for the first picture portion.

8. The method of claim 1, wherein a first variable length coding table and a second variable length coding table are selectively available for determining entropy coded codewords for picture portions of the video frame, and wherein the method further comprises:

determining a selected variable length coding table from the first variable length coding table and the second variable length coding table for picture portions of the video frame;

generating an adaptive switching mask for the video frame, wherein the adaptive switching mask comprises indicators indicating the selected variable length coding tables for the picture portions; and

entropy encoding and writing the adaptive switching mask to the bitstream.

9. The method of claim 1, wherein the second entropy coded codeword indicates the level one pattern is a terminating pattern.

10. The method of claim 1, wherein the picture portions comprise 64x64 pixels, the method further comprising:

generating a binary mask of picture portions comprising a plurality of binary indicators each indicating whether the associated picture portion comprises four 32x32 non-terminating partitions or four 32x32 partitions with at least one terminating 32x32 partition; and

entropy encoding and writing the binary mask of picture portions to the bitstream.

11. The method of claim 1, wherein the picture portions comprise 64x64 pixels, the method further comprising:

generating a binary mask of picture portions comprising a plurality of binary indicators each indicating whether the associated picture portion comprises four 32x32 non-terminating partitions or four 32x32 partitions with at least one terminating 32x32 partition;

determining whether to encode the binary mask of picture portions using a symbol run coding method or a bitmap coding method;

generating a binary mask of picture portions coding method indicator;

entropy encoding the binary mask of picture portions based on the selected coding method; and

writing the binary mask of picture portions coding method indicator and the entropy encoded binary mask of picture portions to the bitstream.

12. The method of claim 1, further comprising:

grouping the termination bits into a first group comprising termination bits for base level patterns having two partitions, a second group comprising termination bits for level one or higher patterns having two partitions, a third group comprising termination bits for base level patterns having three partitions, a fourth group comprising termination bits for level one or higher patterns having three partitions, a fifth group comprising termination bits for base level patterns having four partitions, a sixth group comprising termination bits for level one or higher patterns having four partitions, and a seventh group comprising termination bits for base level or higher patterns having more than four partitions.

13. The method of claim 1, further comprising:

grouping the termination bits into a first group comprising termination bits for base level patterns having two partitions, a second group comprising termination bits for level one or higher patterns having two partitions, a third group comprising termination bits for base level patterns having three partitions, a fourth group comprising termination bits for level one or higher patterns having three partitions, a fifth group comprising termination bits for base level patterns having four partitions, a sixth group comprising termination bits for level one or higher patterns having four partitions, and a seventh group comprising termination bits for base level or higher patterns having more than four partitions,

wherein entropy encoding the termination bits comprises:

inverting at least one of the first through seventh groups of termination bits and entropy encoding the inverted group of termination bits based on a selected variable length coding table from a plurality of variable length coding tables giving the smallest bit cost for the inverted group of termination bits;

generating a variable length coding table header associated with the selected variable length coding table; and

writing the variable length coding table header and the entropy encoded inverted group of termination bits to the bitstream.

14. The method of claim 1, wherein the input data comprises luma data and chroma data defining luma partitions and chroma partitions, the method further comprising:

generating follow bits for the video frame, wherein the follow bits indicate, for picture portions of the video frame, at least one of the chroma partitions match the luma partitions or the chroma partitions do not match the luma partitions.

15. The method of claim 1, wherein the input data comprises luma data and chroma data defining luma partitions and chroma partitions, the method further comprising:

generating follow bits for the video frame, wherein the follow bits indicate, for picture portions of the video frame, at least one of the chroma partitions match the luma partitions or the chroma partitions do not match the luma partitions;

determining a follow bits selected coding technique for the follow bits from at least one of a bitmap coding or a symbol run variable length coding;

generating a follow bits selected coding technique header indicating at least one of the bitmap coding or the symbol run variable length coding;

entropy encoding the follow bits based on the selected coding technique; and

writing the follow bits selected coding technique and the entropy encoded follow bits to the bitstream.

16. The method of claim 1, wherein the input data comprises luma data defining luma partitions and chroma data defining U plane chroma partitions and V plane chroma partitions, the method further comprising:

generating U plane follow bits for the video frame, wherein the U plane follow bits indicate, for picture portions of the video frame, at least one of the U plane chroma partitions match the luma partitions or the U plane chroma partitions do not match the luma partitions;

generating V plane follow bits for the video frame, wherein the V plane follow bits indicate, for picture portions of the video frame, at least one of the V plane chroma partitions match the luma partitions or the V plane chroma partitions do not match the luma partitions;

determining a U plane follow bits selected coding technique for the U plane follow bits from at least one of a bitmap coding or a symbol run variable length coding and a U plane follow bits selected coding technique header indicating at least one of the bitmap coding or the symbol run variable length coding;

generating V plane predicted follow bits based at least in part on the U plane follow bits;

differencing the V plane follow bits and the V plane predicted follow bits to generate V plane difference follow bits;

determining a V plane difference follow bits selected coding technique for the V plane difference follow bits from at least one of a bitmap coding or a symbol run variable length coding and a V plane difference follow bits selected coding technique header indicating at least one of the bitmap coding or the symbol run variable length coding;

entropy encoding the U plane follow bits and V plane difference follow bits, based on the selected coding techniques; and

writing the U plane follow bits selected coding technique header, the V plane difference follow bits selected coding technique header, the entropy encoded U plane follow bits, and the entropy encoded V plane difference follow bits to the bitstream.

17. The method of claim 1, wherein the input data comprises at least one of luma data or chroma data.

18. The method of claim 1, wherein the video frame comprises an I-picture, and wherein the first picture portion comprises a tile of the video frame.

19. The method of claim 1, wherein the video frame comprises an I-picture, wherein the first picture portion comprises a tile of the video frame, and wherein the partitions comprise k-d tree partitions.

20. The method of claim 1, wherein the video frame comprises at least one of a P-picture or a B/F-picture, and wherein the picture portion comprises a partition of a tile of the video frame.

21. The method of claim 1, wherein the partitions comprise at least one of k-d tree partitions, bi-tree partitions, quad-tree partitions, or structured codebook partitions.

22. The method of claim 1, wherein the picture portion comprises at least one of a tile, a largest coding unit, or a super block.

23. The method of claim 1, wherein the termination bits are associated with at least one of the video frame, a slice of the video frame, or a sequence of video frames.

24. The method of claim 1, wherein the plurality of available patterns comprises a codebook of available patterns.

25. The method of claim 1, further comprising:

determining a variable length code tables selection mode for at least one of the video frame, a slice of the video frame, or a sequence of video frames from at least one of a first variable length code tables selection mode where a single variable length coding table is used for determining entropy coded codewords for every picture portion of the video frame and a second variable length code tables selection mode where a picture portion-based variable length coding table is selected for each picture portion of the video frame from two or more available variable length coding tables and the selected picture portion-based variable length coding table is used for determining entropy coded codewords for the associated picture portion; and

grouping the termination bits into a first group comprising termination bits for base level patterns having two partitions, a second group comprising termination bits for level one or higher patterns having two partitions, a third group comprising termination bits for base level patterns having three partitions, a fourth group comprising termination bits for level one or higher patterns having three partitions, a fifth group comprising termination bits for base level patterns having four partitions, a sixth group comprising termination bits for level one or higher patterns having four partitions, and a seventh group comprising termination bits for base level or higher patterns having more than four partitions,

wherein entropy encoding the termination bits comprises:

inverting at least one of the first through seventh groups of termination bits and entropy encoding the inverted group of termination bits based on a selected variable length coding table from a plurality of variable length coding tables giving the smallest bit cost for the inverted group of termination bits;

generating a variable length coding table header associated with the selected variable length coding table; and

writing the variable length coding table header and the entropy encoded inverted group of termination bits to the bitstream,

wherein a first variable length coding table and a second variable length coding table are selectively available for determining entropy coded codewords for picture portions of the video frame, and wherein the method further comprises:

determining a selected switching mode for switching between the first and second variable length coding tables for the picture portions of the video frame from at least one of an automatic switching mode or an adaptive switching mode, and wherein the adaptive switching mode is the selected switching mode if a quantizer value is less than a predetermined threshold,

wherein a first variable length coding table and a second variable length coding table are selectively available for determining entropy coded codewords for picture portions of the video frame, and wherein a first selected variable length coding table from at least one of the first variable length coding table or the second variable length coding table is determined for the first picture portion automatically based on a prediction of complexity for the first picture portion,

wherein a high complexity variable length coding table and a low complexity variable length coding table are selectively available for determining entropy coded codewords for picture portions of the video frame, and wherein a first selected variable length coding table from the high complexity variable length coding table and the low complexity variable length coding table is determined for the first picture portion automatically based on a prediction of complexity for the first picture portion, and wherein the prediction of complexity for the first picture portion comprises determining a number of splits for picture portions neighboring the first picture portion and using the high complexity variable length coding table when the number of splits is above a threshold and the low complexity variable length coding table when the number of splits is at or below the threshold,

wherein a first variable length coding table and a second variable length coding table are selectively available for determining entropy coded codewords for picture portions of the video frame, wherein a first selected variable length coding table from the first variable length coding

table and the second variable length coding table is determined for the first picture portion, and wherein the method further comprises:

determining a selected variable length coding table from the first variable length coding table and the second variable length coding table for picture portions of the video frame;

generating an adaptive switching mask for the video frame, wherein the adaptive switching mask comprises indicators indicating the selected variable length coding tables for the picture portions; and

entropy encoding and writing the adaptive switching mask to the bitstream,

wherein the second entropy coded codeword indicates the level one pattern is a terminating pattern,

wherein the picture portions comprise 64x64 pixels, the method further comprising:

generating a binary mask of picture portions comprising a plurality of binary indicators each indicating whether the associated picture portion comprises four 32x32 non-terminating partitions or four 32x32 partitions with at least one terminating 32x32 partition;

determining whether to encode the binary mask of picture portions using a symbol run coding method or a bitmap coding method;

generating a binary mask of picture portions coding method indicator;

entropy encoding the binary mask of picture portions based on the selected coding method; and

writing the binary mask of picture portions coding method indicator and the entropy encoded binary mask of picture portions to the bitstream,

wherein the input data comprises luma data and chroma data defining luma partitions and chroma partitions, the method further comprising:

generating follow bits for the video frame, wherein the follow bits indicate, for picture portions of the video frame, at least one of the chroma partitions match the luma partitions or the chroma partitions do not match the luma partitions;

determining a follow bits selected coding technique for the follow bits from at least one of a bitmap coding or a symbol run variable length coding;

generating a follow bits selected coding technique header indicating at least one of the bitmap coding or the symbol run variable length coding;

entropy encoding the follow bits based on the selected coding technique; and

writing the follow bits selected coding technique and the entropy encoded follow bits to the bitstream,

wherein the input data comprises luma data defining luma partitions and chroma data defining U plane chroma partitions and V plane chroma partitions, the method further comprising:

generating U plane follow bits for the video frame, wherein the U plane follow bits indicate, for picture portions of the video frame, at least one of the U plane chroma partitions match the luma partitions or the U plane chroma partitions do not match the luma partitions;

generating V plane follow bits for the video frame, wherein the V plane follow bits indicate, for picture portions of the video frame, at least one of the V plane chroma partitions match the luma partitions or the V plane chroma partitions do not match the luma partitions;

determining a U plane follow bits selected coding technique for the U plane follow bits from at least one of a bitmap coding or a symbol run variable length coding and a U plane follow bits selected coding technique header indicating at least one of the bitmap coding or the symbol run variable length coding;

generating V plane predicted follow bits based at least in part on the U plane follow bits;

differencing the V plane follow bits and the V plane predicted follow bits to generate V plane difference follow bits;

determining a V plane difference follow bits selected coding technique for the V plane difference follow bits from at least one of a bitmap coding or a symbol run variable length coding and a V plane difference follow bits selected coding technique header indicating at least one of the bitmap coding or the symbol run variable length coding;

entropy encoding the U plane follow bits and V plane difference follow bits, based on the selected coding techniques; and

writing the U plane follow bits selected coding technique header, the V plane difference follow bits selected coding technique header, the entropy encoded U plane follow bits, and the entropy encoded V plane difference follow bits to the bitstream,

wherein the input data comprises at least one of luma data or chroma data,

wherein the video frame comprises an I-picture, and wherein the first picture portion comprises a tile of the video frame,

wherein the video frame comprises an I-picture, wherein the first picture portion comprises a tile of the video frame, and wherein the partitions comprise k-d tree partitions,

wherein the video frame comprises at least one of a P-picture or a B/F-picture, and wherein the picture portion comprises a partition of a tile of the video frame,

wherein the partitions comprise at least one of k-d tree partitions, bi-tree partitions, quad-tree partitions, or structured codebook partitions,

wherein the picture portion comprises at least one of a tile, a largest coding unit, or a super block,

wherein the termination bits are associated with at least one of the video frame, a slice of the video frame, or a sequence of video frames, and

wherein the plurality of available patterns comprises a codebook of available patterns.

26. A video encoder comprising:

an image buffer;

a graphics processing unit comprising entropy encoder logic circuitry, wherein the graphics processing unit is communicatively coupled to the image buffer and wherein the entropy encoder logic circuitry is configured to:

load input data defining partitions of picture portions of a video frame;

determine a multi-level pattern for a first picture portion, wherein the multi-level pattern comprises a base pattern selected from a plurality of available patterns and at least one level one pattern selected from the plurality of available patterns for at least one non-terminating portion of the base pattern;

determine termination bits indicating terminations of pattern levels of the picture portions of the video frame, wherein the termination bits comprise first termination bits associated with the base pattern of the multi-level pattern and the first picture portion;

determine a first entropy coded codeword associated with the base pattern and a second entropy coded codeword associated with the level one pattern;

entropy encode the termination bits; and

write the first entropy coded codeword, the second entropy coded codeword, and the entropy coded termination bits to a bitstream.

27. The video encoder of claim 26, wherein the entropy encoder logic circuitry is further configured to:

determine a variable length code tables selection mode for at least one of the video frame, a slice of the video frame, or a sequence of video frames from at least one of a first variable length code tables selection mode where a single variable length coding table is used for determining entropy coded codewords for every picture portion of the video frame and a second variable length code tables selection mode where a picture portion-based variable length coding

table is selected for each picture portion of the video frame from two or more available variable length coding tables and the selected picture portion-based variable length coding table is used for determining entropy coded codewords for the associated picture portion.

28. The video encoder of claim 26, wherein the entropy encoder logic circuitry is further configured to:

determine a selected switching mode for switching between the first and second variable length coding tables for the picture portions of the video frame from at least one of an automatic switching mode or an adaptive switching mode, and wherein the adaptive switching mode is the selected switching mode if a quantizer value is less than a predetermined threshold.

29. The video encoder of claim 26, wherein a high complexity variable length coding table and a low complexity variable length coding table are selectively available for determining entropy coded codewords for picture portions of the video frame, and wherein the entropy encoder logic circuitry is configured to determine a first selected variable length coding table from the high complexity variable length coding table and the low complexity variable length coding table for the first picture portion automatically based on a prediction of complexity for the first picture portion, and wherein to determine the prediction of complexity for the first picture portion comprises the entropy encoder logic being configured to determine a number of splits for picture portions neighboring the first picture portion and using the high complexity variable length coding table when the number of splits is above a threshold and the low complexity variable length coding table when the number of splits is at or below the threshold.

30. The video encoder of claim 26, wherein a first variable length coding table and a second variable length coding table are selectively available for determining entropy coded codewords for picture portions of the video frame, and wherein the entropy encoder logic circuitry is further configured to:

determine a selected variable length coding table from the first variable length coding table and the second variable length coding table for picture portions of the video frame;

generate an adaptive switching mask for the video frame, wherein the adaptive switching mask comprises indicators indicating the selected variable length coding tables for the picture portions; and

entropy encode and write the adaptive switching mask to the bitstream.

31. The video encoder of claim 26, wherein the second entropy coded codeword indicates the level one pattern is a terminating pattern.

32. The video encoder of claim 26, wherein the picture portions comprise 64x64 pixels, and wherein the entropy encoder logic circuitry is further configured to:

generate a binary mask of picture portions comprising a plurality of binary indicators each indicating whether the associated picture portion comprises four 32x32 non-terminating partitions or four 32x32 partitions with at least one terminating 32x32 partition;

determine whether to encode the binary mask of picture portions using a symbol run coding method or a bitmap coding method;

generate a binary mask of picture portions coding method indicator;

entropy encode the binary mask of picture portions based on the selected coding method; and

write the binary mask of picture portions coding method indicator and the entropy encoded binary mask of picture portions to the bitstream.

33. The video encoder of claim 26, wherein the entropy encoder logic circuitry is further configured to:

group the termination bits into a first group comprising termination bits for base level patterns having two partitions, a second group comprising termination bits for level one or higher patterns having two partitions, a third group comprising termination bits for base level patterns having three partitions, a fourth group comprising termination bits for level one or higher patterns having three partitions, a fifth group comprising termination bits for base level patterns having four partitions, a sixth group comprising termination bits for level one or higher

patterns having four partitions, and a seventh group comprising termination bits for base level or higher patterns having more than four partitions,

wherein to entropy encode the termination bits comprises the entropy encoder logic being configured to:

invert at least one of the first through seventh groups of termination bits and entropy encoding the inverted group of termination bits based on a selected variable length coding table from a plurality of variable length coding tables giving the smallest bit cost for the inverted group of termination bits;

generate a variable length coding table header associated with the selected variable length coding table; and

write the variable length coding table header and the entropy encoded inverted group of termination bits to the bitstream.

34. The video encoder of claim 26, wherein the input data comprises luma data and chroma data defining luma partitions and chroma partitions, and wherein the entropy encoder logic circuitry is further configured to:

generate follow bits for the video frame, wherein the follow bits indicate, for picture portions of the video frame, at least one of the chroma partitions match the luma partitions or the chroma partitions do not match the luma partitions;

determine a follow bits selected coding technique for the follow bits from at least one of a bitmap coding or a symbol run variable length coding;

generate a follow bits selected coding technique header indicating at least one of the bitmap coding or the symbol run variable length coding;

entropy encode the follow bits based on the selected coding technique; and

write the follow bits selected coding technique and the entropy encoded follow bits to the bitstream.

35. The video encoder of claim 26, wherein the input data comprises luma data defining luma partitions and chroma data defining U plane chroma partitions and V plane chroma partitions, and wherein the entropy encoder logic circuitry is further configured to:

generate U plane follow bits for the video frame, wherein the U plane follow bits indicate, for picture portions of the video frame, at least one of the U plane chroma partitions match the luma partitions or the U plane chroma partitions do not match the luma partitions;

generate V plane follow bits for the video frame, wherein the V plane follow bits indicate, for picture portions of the video frame, at least one of the V plane chroma partitions match the luma partitions or the V plane chroma partitions do not match the luma partitions;

determine a U plane follow bits selected coding technique for the U plane follow bits from at least one of a bitmap coding or a symbol run variable length coding and a U plane follow bits selected coding technique header indicating at least one of the bitmap coding or the symbol run variable length coding;

generate V plane predicted follow bits based at least in part on the U plane follow bits;

difference the V plane follow bits and the V plane predicted follow bits to generate V plane difference follow bits;

determine a V plane difference follow bits selected coding technique for the V plane difference follow bits from at least one of a bitmap coding or a symbol run variable length coding and a V plane difference follow bits selected coding technique header indicating at least one of the bitmap coding or the symbol run variable length coding;

entropy encode the U plane follow bits and V plane difference follow bits, based on the selected coding techniques; and

write the U plane follow bits selected coding technique header, the V plane difference follow bits selected coding technique header, the entropy encoded U plane follow bits, and the entropy encoded V plane difference follow bits to the bitstream.

36. The video encoder of claim 26, wherein the video frame comprises an I-picture, wherein the first picture portion comprises a tile of the video frame, and wherein the partitions comprise k-d tree partitions.

37. The video encoder of claim 26, wherein the video frame comprises at least one of a P-picture or a B/F-picture, and wherein the picture portion comprises a partition of a tile of the video frame.

38. The video encoder of claim 26, wherein the partitions comprise at least one of k-d tree partitions, bi-tree partitions, quad-tree partitions, or structured codebook partitions.

39. The video encoder of claim 26, wherein the picture portion comprises at least one of a tile, a largest coding unit, or a super block.

40. The video encoder of claim 26, wherein the termination bits are associated with at least one of the video frame, a slice of the video frame, or a sequence of video frames.

41. The video encoder of claim 26, wherein the plurality of available patterns comprises a codebook of available patterns.

42. The video encoder of claim 26, wherein the entropy encoder logic circuitry is further configured to:

determine a variable length code tables selection mode for at least one of the video frame, a slice of the video frame, or a sequence of video frames from at least one of a first variable length code tables selection mode where a single variable length coding table is used for determining entropy coded codewords for every picture portion of the video frame and a second variable length code tables selection mode where a picture portion-based variable length coding table is selected for each picture portion of the video frame from two or more available variable length coding tables and the selected picture portion-based variable length coding table is used for determining entropy coded codewords for the associated picture portion;

determine a selected switching mode for switching between the first and second variable length coding tables for the picture portions of the video frame from at least one of an automatic

switching mode or an adaptive switching mode, and wherein the adaptive switching mode is the selected switching mode if a quantizer value is less than a predetermined threshold; and

group the termination bits into a first group comprising termination bits for base level patterns having two partitions, a second group comprising termination bits for level one or higher patterns having two partitions, a third group comprising termination bits for base level patterns having three partitions, a fourth group comprising termination bits for level one or higher patterns having three partitions, a fifth group comprising termination bits for base level patterns having four partitions, a sixth group comprising termination bits for level one or higher patterns having four partitions, and a seventh group comprising termination bits for base level or higher patterns having more than four partitions,

wherein to entropy encode the termination bits comprises the entropy encoder logic being configured to:

invert at least one of the first through seventh groups of termination bits and entropy encoding the inverted group of termination bits based on a selected variable length coding table from a plurality of variable length coding tables giving the smallest bit cost for the inverted group of termination bits;

generate a variable length coding table header associated with the selected variable length coding table; and

write the variable length coding table header and the entropy encoded inverted group of termination bits to the bitstream,

wherein a high complexity variable length coding table and a low complexity variable length coding table are selectively available for determining entropy coded codewords for picture portions of the video frame, and wherein the entropy encoder logic circuitry is configured to determine a first selected variable length coding table from the high complexity variable length coding table and the low complexity variable length coding table for the first picture portion automatically based on a prediction of complexity for the first picture portion, and

wherein to determine the prediction of complexity for the first picture portion comprises the entropy encoder logic being configured to determine a number of splits for picture portions neighboring the first picture portion and using the high complexity variable length coding table when the number of splits is above a threshold and the low complexity variable length coding table when the number of splits is at or below the threshold,

wherein a first variable length coding table and a second variable length coding table are selectively available for determining entropy coded codewords for picture portions of the video frame, and wherein the entropy encoder logic circuitry is further configured to:

determine a selected variable length coding table from the first variable length coding table and the second variable length coding table for picture portions of the video frame;

generate an adaptive switching mask for the video frame, wherein the adaptive switching mask comprises indicators indicating the selected variable length coding tables for the picture portions; and

entropy encode and write the adaptive switching mask to the bitstream,

wherein the second entropy coded codeword indicates the level one pattern is a terminating pattern,

wherein the picture portions comprise 64x64 pixels, and wherein the entropy encoder logic circuitry is further configured to:

generate a binary mask of picture portions comprising a plurality of binary indicators each indicating whether the associated picture portion comprises four 32x32 non-terminating partitions or four 32x32 partitions with at least one terminating 32x32 partition;

determine whether to encode the binary mask of picture portions using a symbol run coding method or a bitmap coding method;

generate a binary mask of picture portions coding method indicator;

entropy encode the binary mask of picture portions based on the selected coding method; and

write the binary mask of picture portions coding method indicator and the entropy encoded binary mask of picture portions to the bitstream,

wherein the input data comprises luma data and chroma data defining luma partitions and chroma partitions, and wherein the entropy encoder logic circuitry is further configured to:

generate follow bits for the video frame, wherein the follow bits indicate, for picture portions of the video frame, at least one of the chroma partitions match the luma partitions or the chroma partitions do not match the luma partitions;

determine a follow bits selected coding technique for the follow bits from at least one of a bitmap coding or a symbol run variable length coding;

generate a follow bits selected coding technique header indicating at least one of the bitmap coding or the symbol run variable length coding;

entropy encode the follow bits based on the selected coding technique; and

write the follow bits selected coding technique and the entropy encoded follow bits to the bitstream,

wherein the input data comprises luma data defining luma partitions and chroma data defining U plane chroma partitions and V plane chroma partitions, and wherein the entropy encoder logic circuitry is further configured to:

generate U plane follow bits for the video frame, wherein the U plane follow bits indicate, for picture portions of the video frame, at least one of the U plane chroma partitions match the luma partitions or the U plane chroma partitions do not match the luma partitions;

generate V plane follow bits for the video frame, wherein the V plane follow bits indicate, for picture portions of the video frame, at least one of the V plane chroma

partitions match the luma partitions or the V plane chroma partitions do not match the luma partitions;

determine a U plane follow bits selected coding technique for the U plane follow bits from at least one of a bitmap coding or a symbol run variable length coding and a U plane follow bits selected coding technique header indicating at least one of the bitmap coding or the symbol run variable length coding;

generate V plane predicted follow bits based at least in part on the U plane follow bits;

difference the V plane follow bits and the V plane predicted follow bits to generate V plane difference follow bits;

determine a V plane difference follow bits selected coding technique for the V plane difference follow bits from at least one of a bitmap coding or a symbol run variable length coding and a V plane difference follow bits selected coding technique header indicating at least one of the bitmap coding or the symbol run variable length coding;

entropy encode the U plane follow bits and V plane difference follow bits, based on the selected coding techniques; and

write the U plane follow bits selected coding technique header, the V plane difference follow bits selected coding technique header, the entropy encoded U plane follow bits, and the entropy encoded V plane difference follow bits to the bitstream,

wherein the video frame comprises an I-picture, wherein the first picture portion comprises a tile of the video frame, and wherein the partitions comprise k-d tree partitions,

wherein the video frame comprises at least one of a P-picture or a B/F-picture, and wherein the picture portion comprises a partition of a tile of the video frame,

wherein the partitions comprise at least one of k-d tree partitions, bi-tree partitions, quad-tree partitions, or structured codebook partitions,

wherein the picture portion comprises at least one of a tile, a largest coding unit, or a super block,

wherein the termination bits are associated with at least one of the video frame, a slice of the video frame, or a sequence of video frames, and

wherein the plurality of available patterns comprises a codebook of available patterns.

43. A system comprising:

a video decoder configured to decode an encoded bitstream, wherein the video decoder is configured to:

decode the encoded bitstream to determine a base pattern of a multi-level pattern for a first picture portion of a video frame based on a first entropy coded codeword in the encoded bitstream and a level one pattern of the multi-level pattern based on a second entropy coded codeword in the encoded bitstream, and termination bits indicating terminations of pattern levels of picture portions of the video frame; and

reconstruct first partitions for the first picture portion based on the decoded base pattern, level one pattern, and termination bits.

44. The system of claim 43, further comprising:

an antenna configured to receive the encoded bitstream, wherein the antenna is communicatively coupled to the video decoder; and

a display device configured to present video frames.

45. The system of claim 43, wherein to decode the encoded bitstream further comprises the video decoder being configured to determine variable length code tables selection mode indicator for at least one of the video frame, a slice of the video frame, or a sequence of video frames indicating a variable length code tables selection mode from at least one of a first variable length code tables selection mode where a single variable length coding table is used to entropy decode entropy coded codewords for every picture portion of the video frame and a second variable length code tables selection mode where a picture portion-based variable length coding

table is selected for each picture portion of the video frame from two or more variable length coding tables and the selected picture portion-based variable length coding table is used to entropy decode entropy coded codewords for the associated picture portion.

46. The system of claim 43, wherein a first variable length coding table and a second variable length coding table are selectively available to entropy decode entropy coded codewords for picture portions of the video frame, and wherein to decode the encoded bitstream further comprises the video decoder being configured to determine a selected switching mode indicator for indicating a switching mode between the first and second variable length coding tables for the picture portions of the video frame from at least one of an automatic switching mode or an adaptive switching mode.

47. The system of claim 43, wherein a first variable length coding table and a second variable length coding table are selectively available to entropy decode entropy coded codewords for picture portions of the video frame, and wherein the video decoder is further configured to:

determine a first selected variable length coding table from at least one of the first variable length coding table or the second variable length coding table automatically based on a prediction of complexity for the first picture portion.

48. The system of claim 43, wherein a high complexity variable length coding table and a low complexity variable length coding table are selectively available to entropy decode entropy coded codewords for picture portions of the video frame, and wherein the video decoder is further configured to:

determine a first selected variable length coding table from at least one of the high complexity variable length coding table or the low complexity variable length coding table automatically based on a prediction of complexity for the first picture portion, wherein the prediction of complexity for the first picture portion comprises the video decoder being configured to determine a number of splits for picture portions neighboring the first picture portion and use the high complexity variable length coding table when the number of splits is

above a threshold and the low complexity variable length coding table when the number of splits is at or below the threshold.

49. The system of claim 43, wherein a first variable length coding table and a second variable length coding table are selectively available to entropy decode entropy coded codewords for picture portions of the video frame, wherein to decode the encoded bitstream further comprises the video decoder being configured to determine an adaptive switching mask for the video frame, wherein the adaptive switching mask comprises an indication of the first selected variable length coding table for the first picture portion.

50. The system of claim 43, wherein the second entropy coded codeword indicates the selected level one pattern is a terminating pattern.

51. The system of claim 43, wherein the first picture portion comprises 64x64 pixels, wherein to decode the encoded bitstream further comprises the video decoder being configured to determine a binary mask of picture portions comprising a plurality of binary indicators each indicating whether the associated picture portions comprises four 32x32 non-terminating partitions or four 32x32 partitions with at least one terminating 32x32 partition.

52. The system of claim 43, wherein the video decoder is further configured to:

entropy decode the termination bits based on a grouping of the termination bits into a first group comprising termination bits for base level patterns having two partitions, a second group comprising termination bits for level one or higher patterns having two partitions, a third group comprising termination bits for base level patterns having three partitions, a fourth group comprising termination bits for level one or higher patterns having three partitions, a fifth group comprising termination bits for base level patterns having four partitions, a sixth group comprising termination bits for level one or higher patterns having four partitions, and a seventh group comprising termination bits for base level or higher patterns having more than four partitions.

53. The system of claim 43, wherein the video decoder is further configured to:

entropy decode the termination bits based on a grouping of the termination bits, wherein to entropy decode the termination bits comprises the video decoder being configured to determine a variable length coding table header associated with a first group of termination bits, and decode the first group of termination bits based on a selected variable length coding table based on the variable length coding table header.

54. The system of claim 43, wherein the multi-level pattern comprises a luma pattern, wherein to decode the encoded bitstream further comprises the video decoder being configured to determine follow bits for the video frame, and wherein the video decoder is further configured to:

determine a chroma pattern for the first picture portion matches the luma pattern based at least in part on the decoded follow bits.

55. The system of claim 43, further comprising:

an antenna configured to receive the encoded bitstream, wherein the antenna is communicatively coupled to the video decoder; and

a display device configured to present video frames,

wherein to decode the encoded bitstream further comprises the video decoder being configured to determine variable length code tables selection mode indicator for at least one of the video frame, a slice of the video frame, or a sequence of video frames from at least one of a first variable length code tables selection mode where a single variable length coding table is used to entropy decode entropy coded codewords for every picture portion of the video frame and a second variable length code tables selection mode where a picture portion-based variable length coding table is selected for each picture portion of the video frame from two or more variable length coding tables and the selected picture portion-based variable length coding table is used to entropy decode entropy coded codewords for the associated picture portion,

wherein a first variable length coding table and a second variable length coding table are selectively available to entropy decode entropy coded codewords for picture portions of the

video frame, and wherein to decode the encoded bitstream further comprises the video decoder being configured to determine a selected switching mode for switching between the first and second variable length coding tables for the picture portions of the video frame from at least one of an automatic switching mode or an adaptive switching mode,

wherein a first variable length coding table and a second variable length coding table are selectively available to entropy decode entropy coded codewords for picture portions of the video frame, and wherein the video decoder is further configured to:

determine a first selected variable length coding table from at least one of the first variable length coding table or the second variable length coding table automatically based on a prediction of complexity for the first picture portion,

wherein a high complexity variable length coding table and a low complexity variable length coding table are selectively available to entropy decode entropy coded codewords for picture portions of the video frame, and wherein the video decoder is further configured to:

determine a first selected variable length coding table from at least one of the high complexity variable length coding table or the low complexity variable length coding table automatically based on a prediction of complexity for the first picture portion, wherein the prediction of complexity for the first picture portion comprises the video decoder being configured to determine a number of splits for picture portions neighboring the first picture portion and use the high complexity variable length coding table when the number of splits is above a threshold and the low complexity variable length coding table when the number of splits is at or below the threshold,

wherein a first variable length coding table and a second variable length coding table are selectively available to entropy decode entropy coded codewords for picture portions of the video frame, wherein to decode the encoded bitstream further comprises the video decoder being configured to determine an adaptive switching mask for the video frame, wherein the adaptive

switching mask comprises an indication of the first selected variable length coding table for the first picture portion,

wherein the second entropy coded codeword indicates the selected level one pattern is a terminating pattern,

wherein the first picture portion comprises 64x64 pixels, wherein to decode the encoded bitstream further comprises the video decoder being configured to determine a binary mask of picture portions comprising a plurality of binary indicators each indicating whether the associated picture portions comprises four 32x32 non-terminating partitions or four 32x32 partitions with at least one terminating 32x32 partition,

wherein the video decoder is further configured to:

entropy decode the termination bits based on a grouping of the termination bits into a first group comprising termination bits for base level patterns having two partitions, a second group comprising termination bits for level one or higher patterns having two partitions, a third group comprising termination bits for base level patterns having three partitions, a fourth group comprising termination bits for level one or higher patterns having three partitions, a fifth group comprising termination bits for base level patterns having four partitions, a sixth group comprising termination bits for level one or higher patterns having four partitions, and a seventh group comprising termination bits for base level or higher patterns having more than four partitions; and

entropy decode the termination bits based on a grouping of the termination bits, wherein to entropy decode the termination bits comprises the video decoder being configured to determine a variable length coding table header associated with a first group of termination bits, and decode the first group of termination bits based on a selected variable length coding table based on the variable length coding table header, and

wherein the multi-level pattern comprises a luma pattern, wherein to decode the encoded bitstream further comprises the video decoder being configured to determine follow bits for the video frame, and wherein the video decoder is further configured to:

determine a chroma pattern for the first picture portion matches the luma pattern based at least in part on the decoded follow bits.

56. At least one machine readable medium comprising:

a plurality of instructions that in response to being executed on a computing device, causes the computing device to perform the method according to any one of claims 1–25.

57. An apparatus, comprising:

means for performing the methods according to any one of claims 1–25.

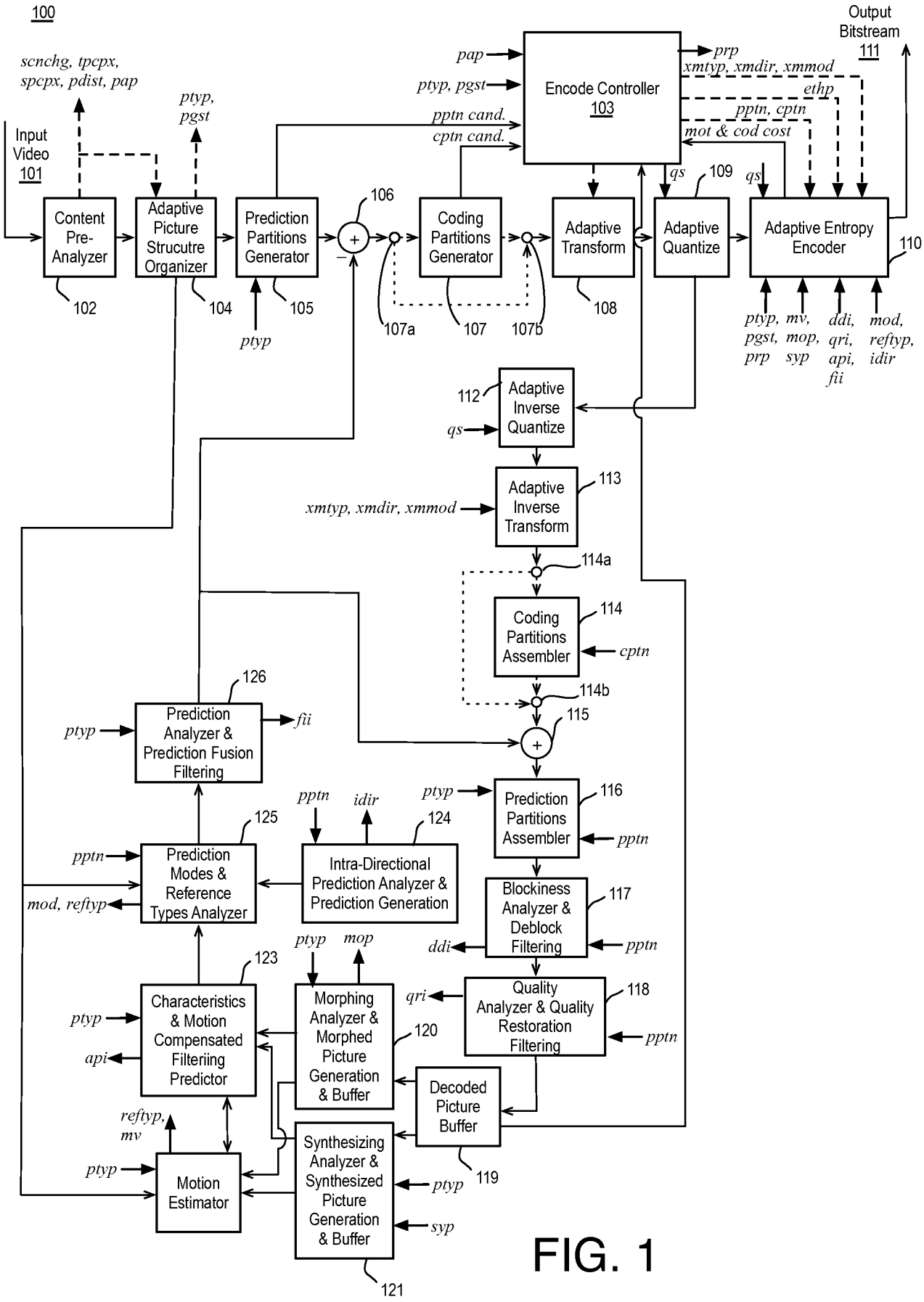

100
Output Bitstream 111
scnchg, tpcpx, spcpx, pdist, pap
ptyp, pgst
Input Video 101
pap
ptyp, pgst
pptn cand.
cptn cand.
Encode Controller 103
prp
xmtyp, xmdir, xmmod
ethp
pptn, cptn
mot & cod cost
qs
109
Content Pre-Analyzer
Adaptive Picture Strucutre Organizer
Prediction Partitions Generator
106
Coding Partitions Generator
Adaptive Transform
Adaptive Quantize
Adaptive Entropy Encoder
110
102
104
105
107a
107
107b
108
ptyp
ptyp, pgst, prp
mv, mop, syp
ddi, qri, api, fii
mod, reftyp, idir
112
Adaptive Inverse Quantize
113
Adaptive Inverse Transform
xmtyp, xmdir, xmmod
114a
Coding Partitions Assembler
114
cptn
114b
115
126
Prediction Analyzer & Prediction Fusion Filtering
fii
pptn
idir
124
125
Prediction Modes & Reference Types Analyzer
Intra-Directional Prediction Analyzer & Prediction Generation
mod, reftyp
Prediction Partitions Assembler
116
Blockiness Analyzer & Deblock Filtering
117
ddi
Quality Analyzer & Quality Restoration Filtering
118
qri
mop
123
Characteristics & Motion Compensated Filteriing Predictor
Morphing Analyzer & Morphed Picture Generation & Buffer
120
api
Decoded Picture Buffer
reftyp, mv
Synthesizing Analyzer & Synthesized Picture Generation & Buffer
119
Motion Estimator
syp
121

FIG. 1

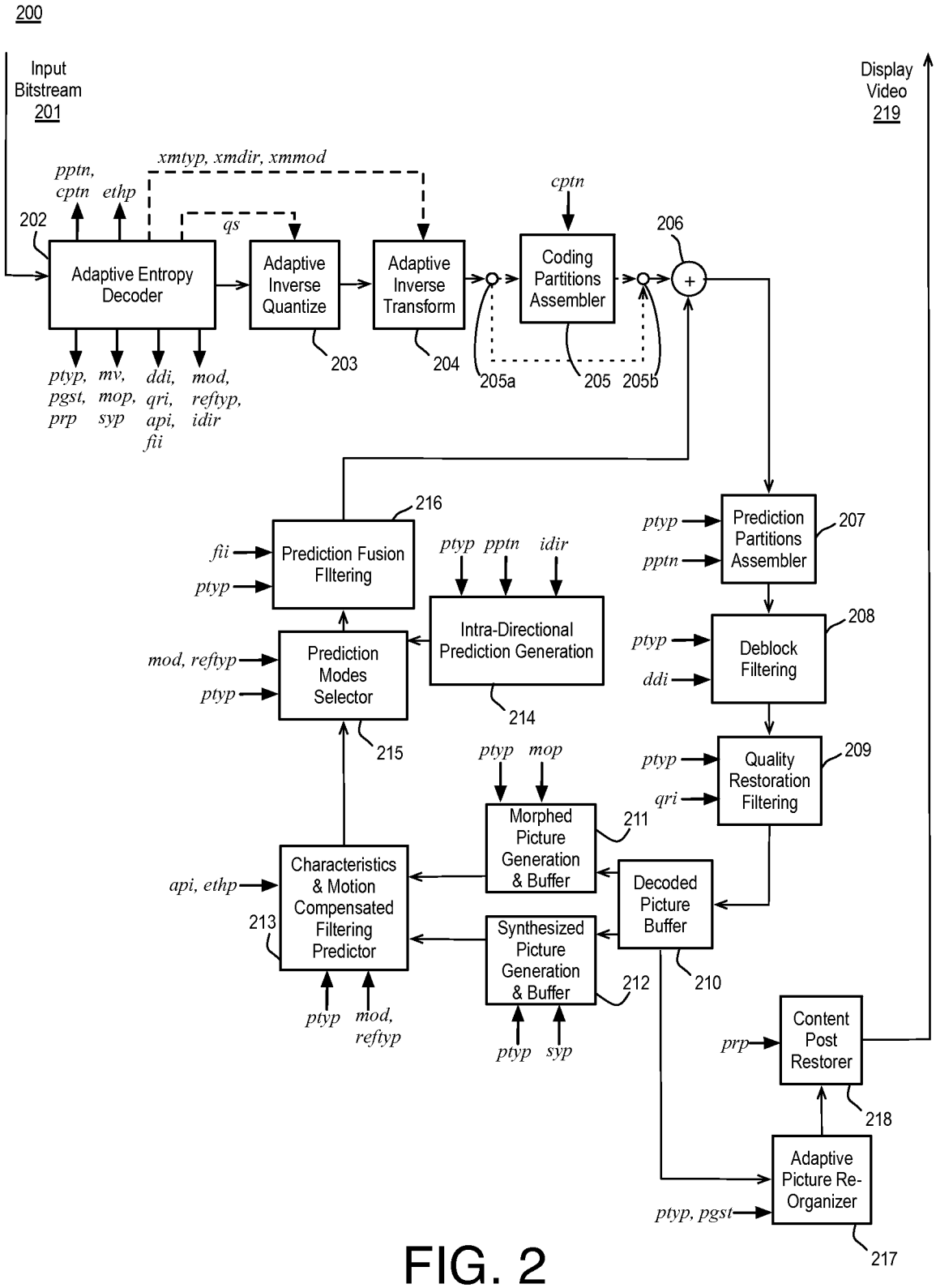

200
Input Bitstream 201
Display Video 219
pptn, cptn ethp
xmtyp, xmdir, xmmod
qs
cptn
202
206
Adaptive Entropy Decoder
Adaptive Inverse Quantize
Adaptive Inverse Transform
Coding Partitions Assembler
203
204
205a
205
205b
ptyp, pgst, prp
mv, mop, syp
ddi, qri, api, fii
mod, reftyp, idir
216
fii
ptyp
Prediction Fusion FIltering
ptyp pptn idir
Intra-Directional Prediction Generation
214
mod, reftyp
ptyp
Prediction Modes Selector
215
ptyp
pptn
Prediction Partitions Assembler
207
ptyp
ddi
Deblock Filtering
208
ptyp
qri
Quality Restoration Filtering
209
ptyp mop
Morphed Picture Generation & Buffer
211
api, ethp
213
Characteristics & Motion Compensated Filtering Predictor
Synthesized Picture Generation & Buffer
212
Decoded Picture Buffer
210
ptyp mod, reftyp
ptyp syp
prp
Content Post Restorer
218
Adaptive Picture Re-Organizer
ptyp, pgst
217

FIG. 2

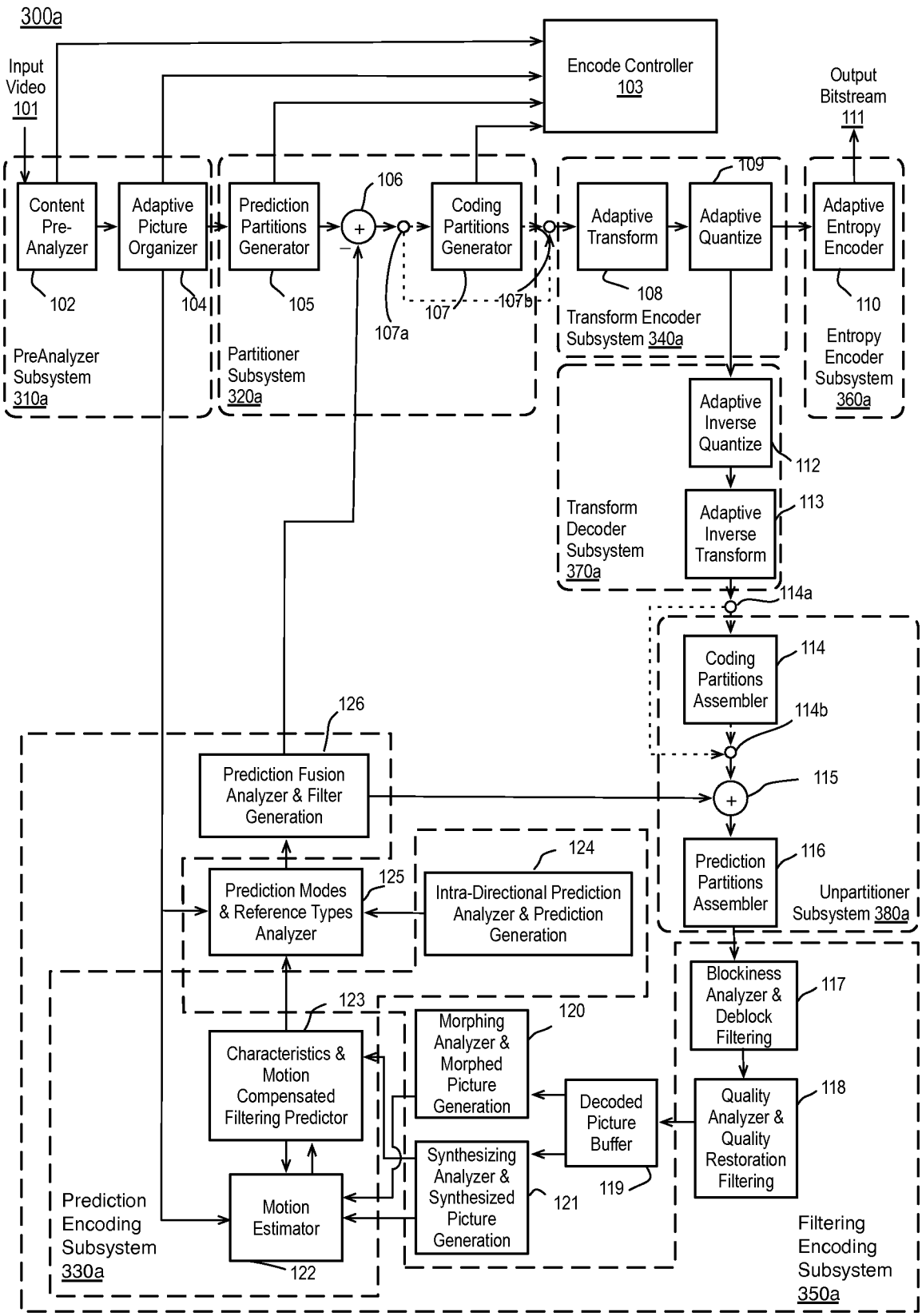

300a
Input Video 101
Encode Controller 103
Output Bitstream 111
Content Pre-Analyzer
Adaptive Picture Organizer
Prediction Partitions Generator
Coding Partitions Generator
Adaptive Transform
Adaptive Quantize
Adaptive Entropy Encoder
102
104
105
106
107
107a
107b
108
109
110
PreAnalyzer Subsystem 310a
Partitioner Subsystem 320a
Transform Encoder Subsystem 340a
Entropy Encoder Subsystem 360a
Adaptive Inverse Quantize
Adaptive Inverse Transform
112
113
Transform Decoder Subsystem 370a
114a
Coding Partitions Assembler
114
114b
115
Prediction Partitions Assembler
116
Unpartitioner Subsystem 380a
126
Prediction Fusion Analyzer & Filter Generation
124
Intra-Directional Prediction Analyzer & Prediction Generation
125
Prediction Modes & Reference Types Analyzer
Blockiness Analyzer & Deblock Filtering
117
123
Characteristics & Motion Compensated Filtering Predictor
120
Morphing Analyzer & Morphed Picture Generation
Decoded Picture Buffer
Quality Analyzer & Quality Restoration Filtering
118
119
Synthesizing Analyzer & Synthesized Picture Generation
121
Motion Estimator
122
Prediction Encoding Subsystem 330a
Filtering Encoding Subsystem 350a

FIG. 3(a)

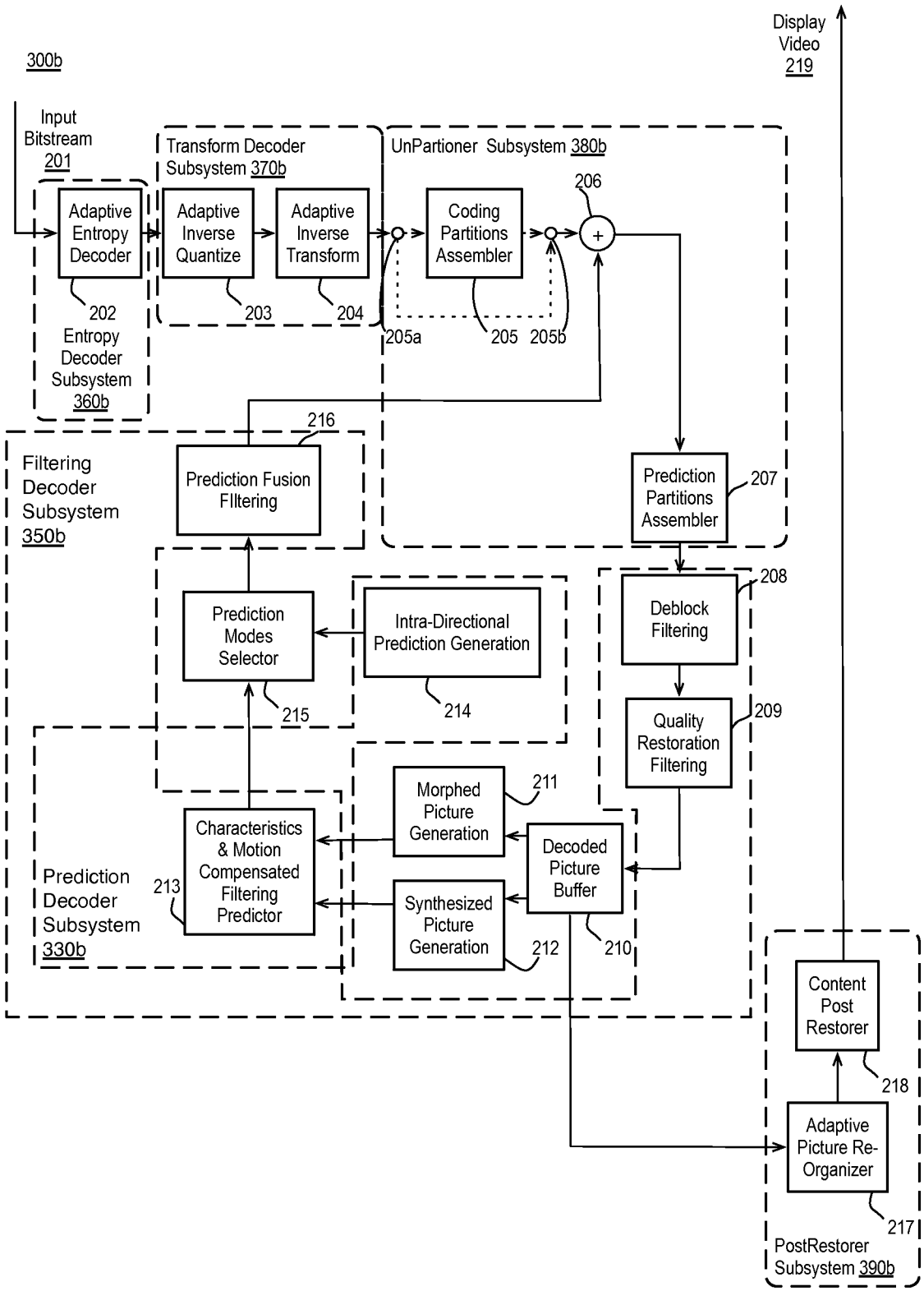

300b
Input Bitstream 201
Transform Decoder Subsystem 370b
UnPartioner Subsystem 380b
Display Video 219
Adaptive Entropy Decoder
Adaptive Inverse Quantize
Adaptive Inverse Transform
Coding Partitions Assembler
206
202
203
204
205a
205
205b
Entropy Decoder Subsystem 360b
216
Filtering Decoder Subsystem 350b
Prediction Fusion FIltering
Prediction Partitions Assembler
207
208
Deblock Filtering
Prediction Modes Selector
Intra-Directional Prediction Generation
215
214
209
Quality Restoration Filtering
211
Morphed Picture Generation
Characteristics & Motion Compensated Filtering Predictor
213
Prediction Decoder Subsystem 330b
Decoded Picture Buffer
Synthesized Picture Generation
212
210
Content Post Restorer
218
Adaptive Picture Re-Organizer
217
PostRestorer Subsystem 390b

FIG. 3(b)

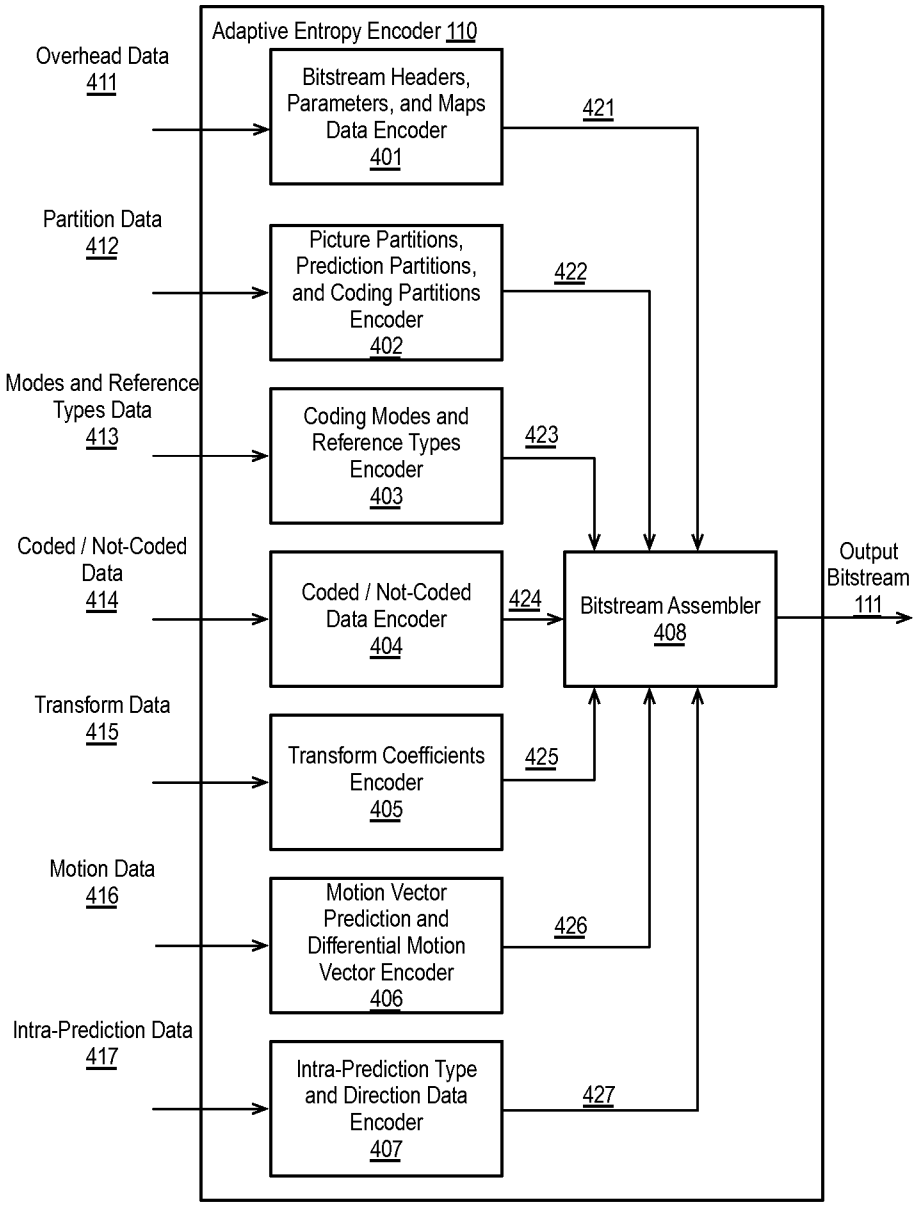

Adaptive Entropy Encoder 110
Overhead Data 411
Bitstream Headers, Parameters, and Maps Data Encoder 401
421
Partition Data 412
Picture Partitions, Prediction Partitions, and Coding Partitions Encoder 402
422
Modes and Reference Types Data 413
Coding Modes and Reference Types Encoder 403
423
Coded / Not-Coded Data 414
Coded / Not-Coded Data Encoder 404
424
Bitstream Assembler 408
Output Bitstream 111
Transform Data 415
Transform Coefficients Encoder 405
425
Motion Data 416
Motion Vector Prediction and Differential Motion Vector Encoder 406
426
Intra-Prediction Data 417
Intra-Prediction Type and Direction Data Encoder 407
427

FIG. 4

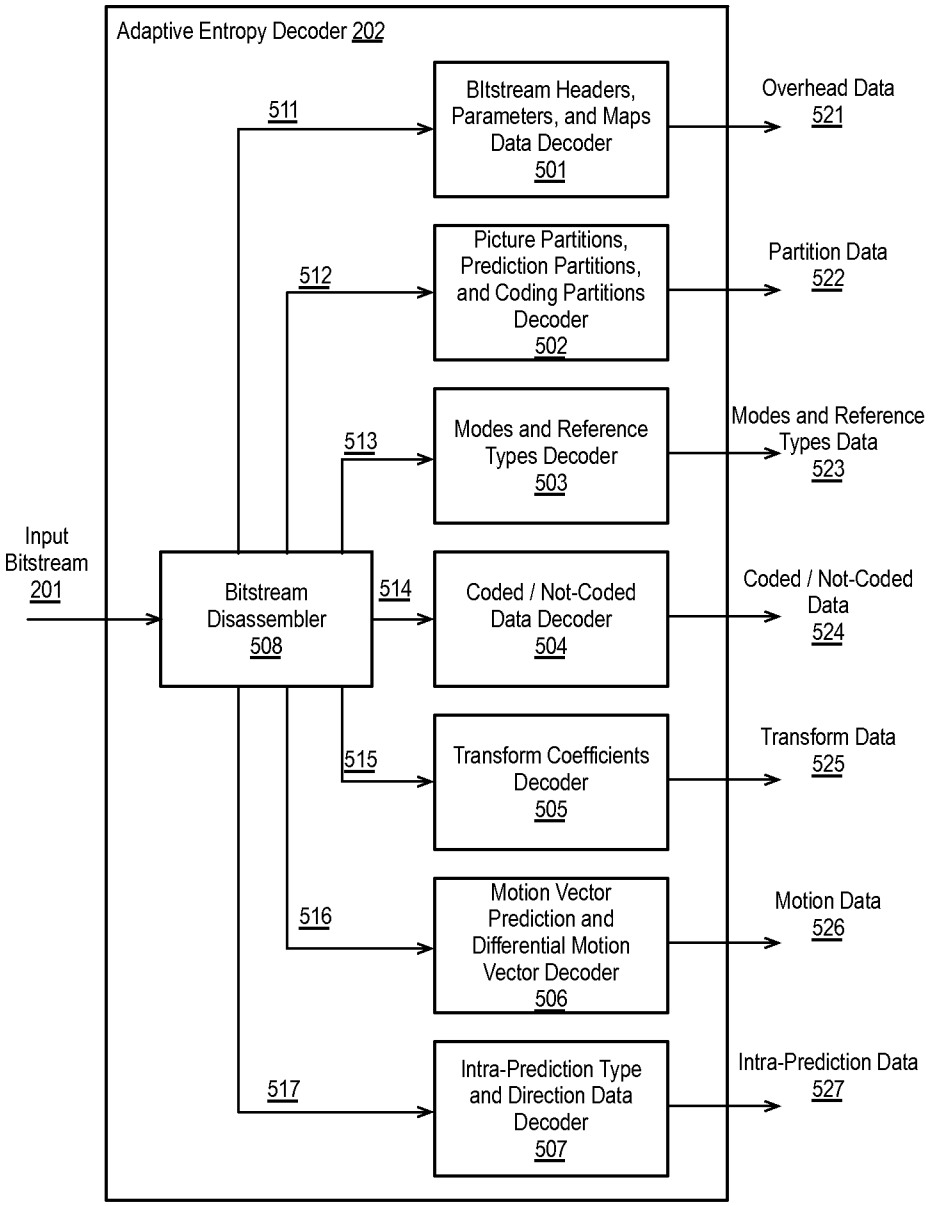

Adaptive Entropy Decoder 202
Input Bitstream 201
Bitstream Disassembler 508
511
512
513
514
515
516
517
BItstream Headers, Parameters, and Maps Data Decoder 501
Picture Partitions, Prediction Partitions, and Coding Partitions Decoder 502
Modes and Reference Types Decoder 503
Coded / Not-Coded Data Decoder 504
Transform Coefficients Decoder 505
Motion Vector Prediction and Differential Motion Vector Decoder 506
Intra-Prediction Type and Direction Data Decoder 507
Overhead Data 521
Partition Data 522
Modes and Reference Types Data 523
Coded / Not-Coded Data 524
Transform Data 525
Motion Data 526
Intra-Prediction Data 527

FIG. 5

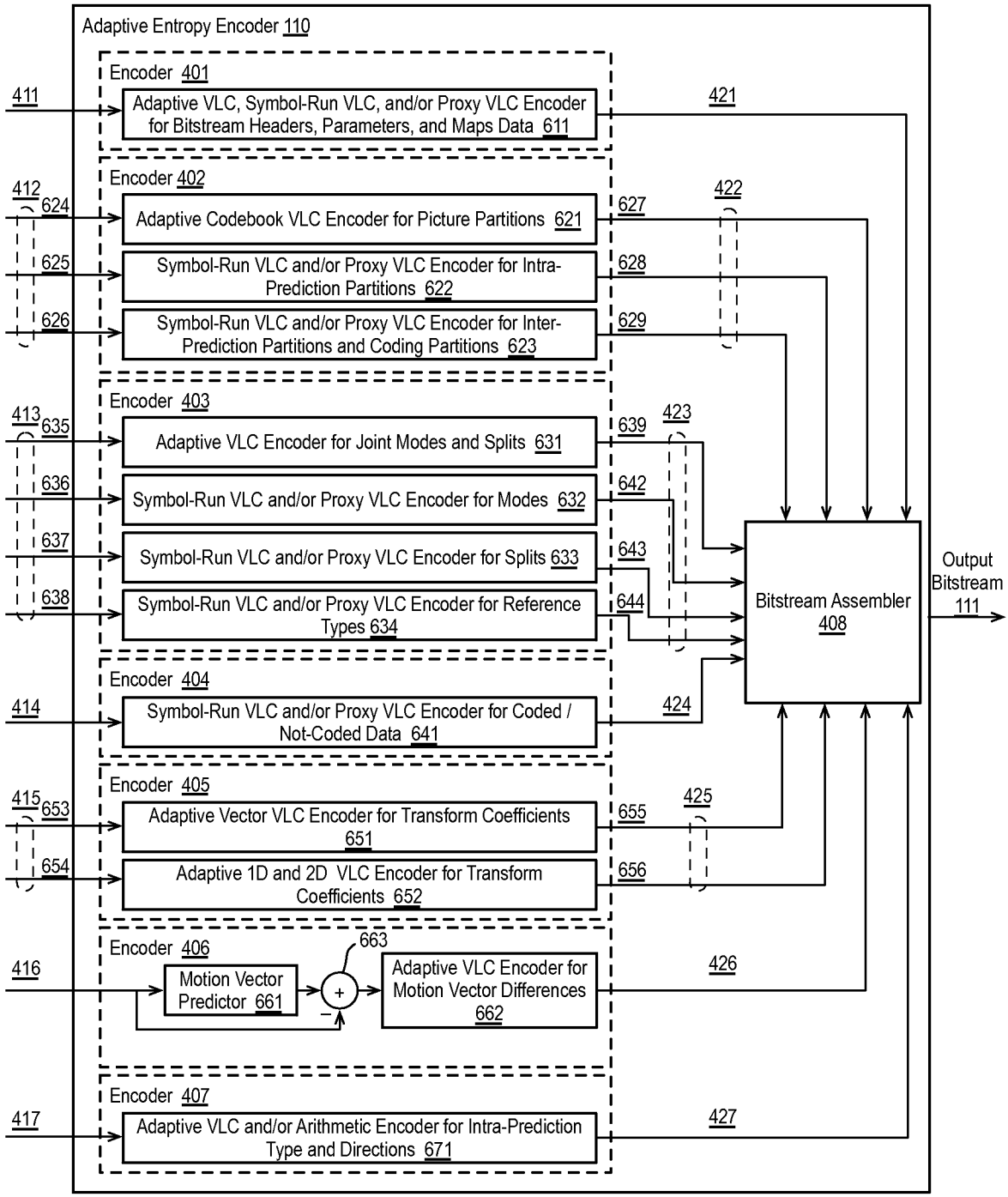

Adaptive Entropy Encoder 110
Encoder 401
411
Adaptive VLC, Symbol-Run VLC, and/or Proxy VLC Encoder for Bitstream Headers, Parameters, and Maps Data 611
421
Encoder 402
412
624
Adaptive Codebook VLC Encoder for Picture Partitions 621
627
422
625
Symbol-Run VLC and/or Proxy VLC Encoder for Intra-Prediction Partitions 622
628
626
Symbol-Run VLC and/or Proxy VLC Encoder for Inter-Prediction Partitions and Coding Partitions 623
629
Encoder 403
413
635
Adaptive VLC Encoder for Joint Modes and Splits 631
639
423
636
Symbol-Run VLC and/or Proxy VLC Encoder for Modes 632
642
637
Symbol-Run VLC and/or Proxy VLC Encoder for Splits 633
643
638
Symbol-Run VLC and/or Proxy VLC Encoder for Reference Types 634
644
Bitstream Assembler 408
Output Bitstream 111
Encoder 404
414
Symbol-Run VLC and/or Proxy VLC Encoder for Coded / Not-Coded Data 641
424
Encoder 405
415
653
Adaptive Vector VLC Encoder for Transform Coefficients 651
655
425
654
Adaptive 1D and 2D VLC Encoder for Transform Coefficients 652
656
Encoder 406
663
416
Motion Vector Predictor 661
+
−
Adaptive VLC Encoder for Motion Vector Differences 662
426
Encoder 407
417
Adaptive VLC and/or Arithmetic Encoder for Intra-Prediction Type and Directions 671
427

FIG. 6

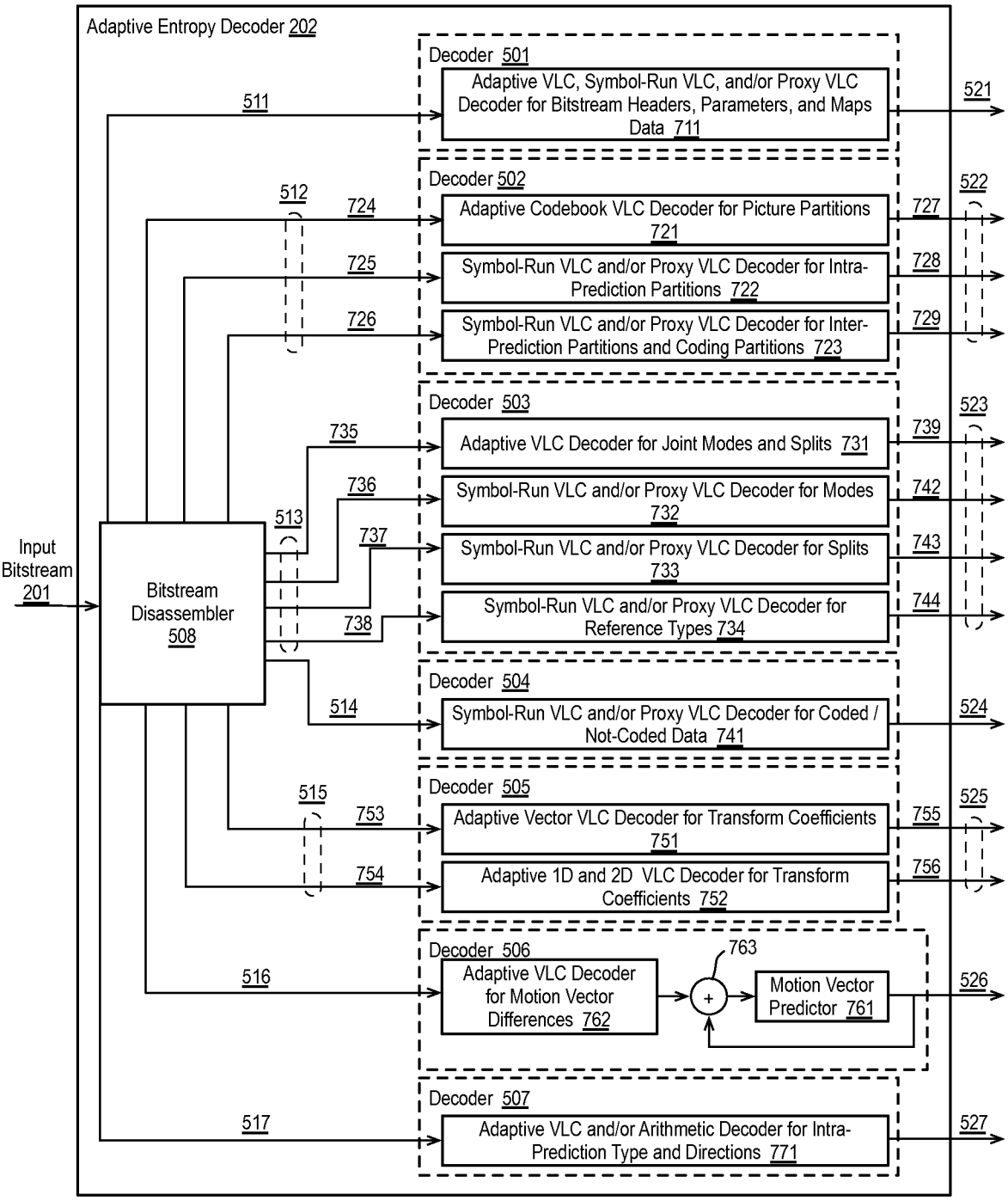

Adaptive Entropy Decoder 202
Input Bitstream 201
Bitstream Disassembler 508
Decoder 501
Adaptive VLC, Symbol-Run VLC, and/or Proxy VLC Decoder for Bitstream Headers, Parameters, and Maps Data 711
Decoder 502
Adaptive Codebook VLC Decoder for Picture Partitions 721
Symbol-Run VLC and/or Proxy VLC Decoder for Intra-Prediction Partitions 722
Symbol-Run VLC and/or Proxy VLC Decoder for Inter-Prediction Partitions and Coding Partitions 723
Decoder 503
Adaptive VLC Decoder for Joint Modes and Splits 731
Symbol-Run VLC and/or Proxy VLC Decoder for Modes 732
Symbol-Run VLC and/or Proxy VLC Decoder for Splits 733
Symbol-Run VLC and/or Proxy VLC Decoder for Reference Types 734
Decoder 504
Symbol-Run VLC and/or Proxy VLC Decoder for Coded / Not-Coded Data 741
Decoder 505
Adaptive Vector VLC Decoder for Transform Coefficients 751
Adaptive 1D and 2D VLC Decoder for Transform Coefficients 752
Decoder 506
Adaptive VLC Decoder for Motion Vector Differences 762
Motion Vector Predictor 761
Decoder 507
Adaptive VLC and/or Arithmetic Decoder for Intra-Prediction Type and Directions 771
511
512
513
514
515
516
517
521
522
523
524
525
526
527
724
725
726
727
728
729
735
736
737
738
739
742
743
744
753
754
755
756
763

FIG. 7

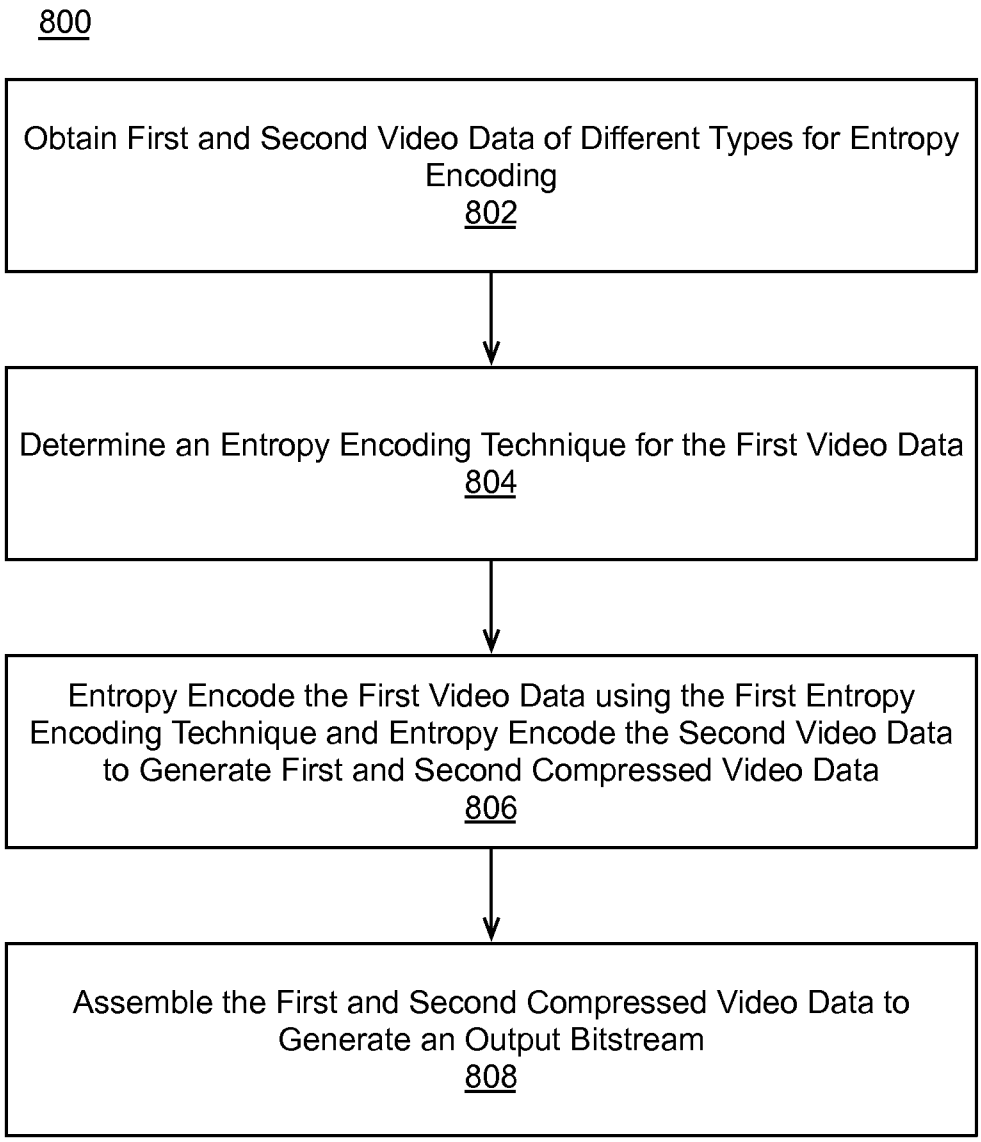

800
Obtain First and Second Video Data of Different Types for Entropy Encoding
802
Determine an Entropy Encoding Technique for the First Video Data
804
Entropy Encode the First Video Data using the First Entropy Encoding Technique and Entropy Encode the Second Video Data to Generate First and Second Compressed Video Data
806
Assemble the First and Second Compressed Video Data to Generate an Output Bitstream
808

FIG. 8

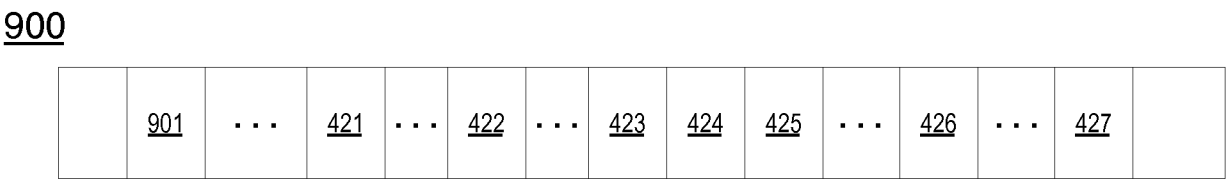

900
901
421
422
423
424
425
426
427

FIG. 9

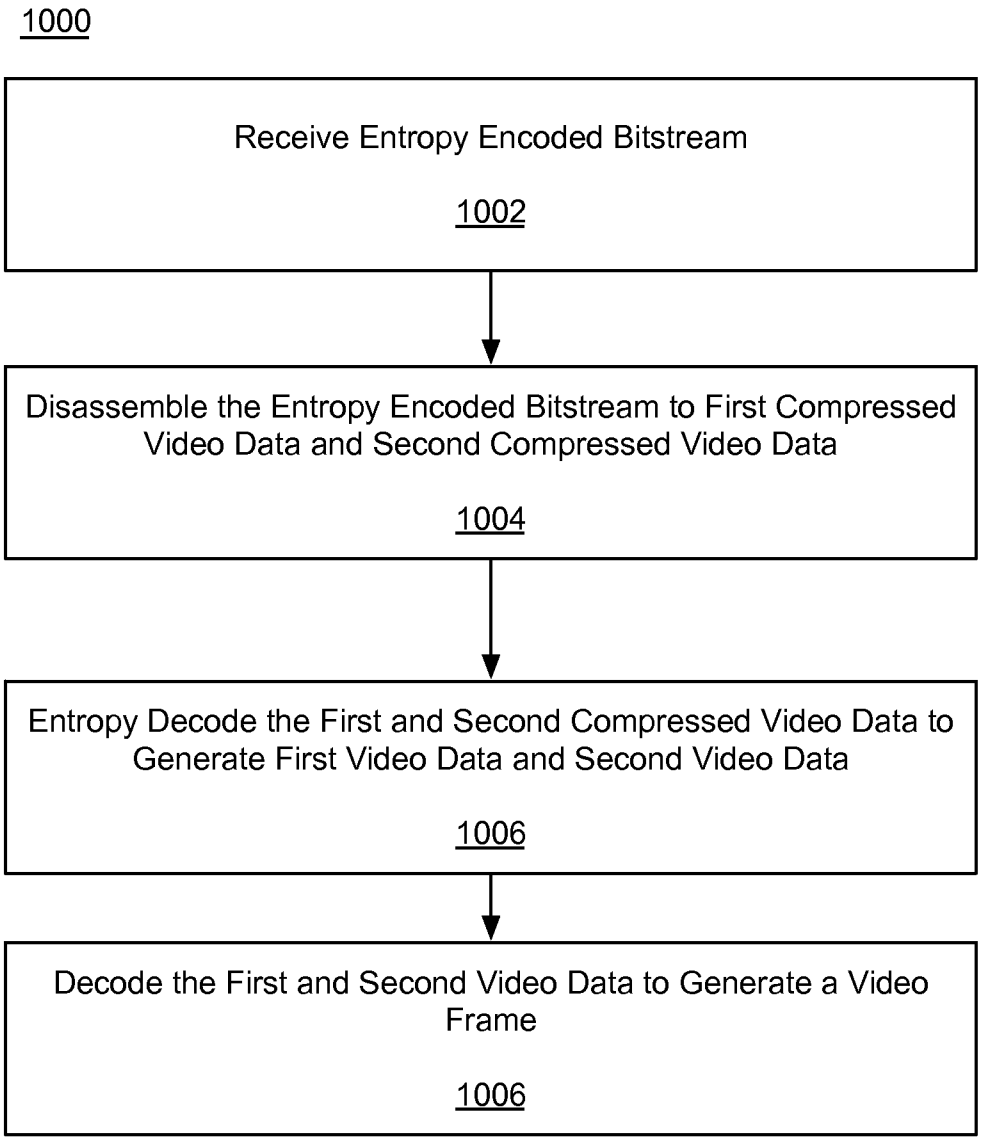

1000
Receive Entropy Encoded Bitstream
1002
Disassemble the Entropy Encoded Bitstream to First Compressed Video Data and Second Compressed Video Data
1004
Entropy Decode the First and Second Compressed Video Data to Generate First Video Data and Second Video Data
1006
Decode the First and Second Video Data to Generate a Video Frame
1006

FIG. 10

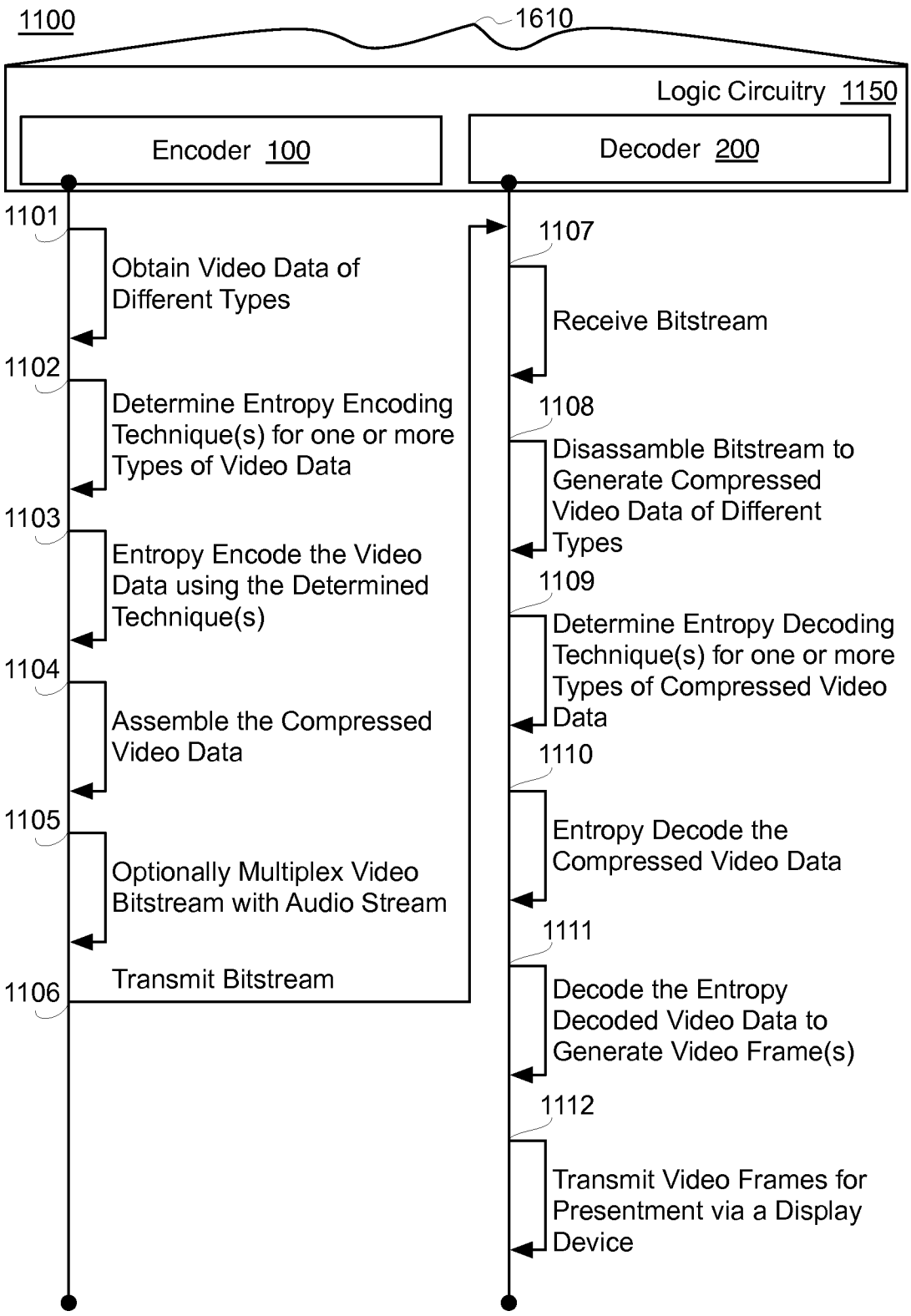

1100
1610
Logic Circuitry 1150
Encoder 100
Decoder 200
1101
Obtain Video Data of Different Types
1102
Determine Entropy Encoding Technique(s) for one or more Types of Video Data
1103
Entropy Encode the Video Data using the Determined Technique(s)
1104
Assemble the Compressed Video Data
1105
Optionally Multiplex Video Bitstream with Audio Stream
1106
Transmit Bitstream
1107
Receive Bitstream
1108
Disassamble Bitstream to Generate Compressed Video Data of Different Types
1109
Determine Entropy Decoding Technique(s) for one or more Types of Compressed Video Data
1110
Entropy Decode the Compressed Video Data
1111
Decode the Entropy Decoded Video Data to Generate Video Frame(s)
1112
Transmit Video Frames for Presentment via a Display Device

FIG. 11

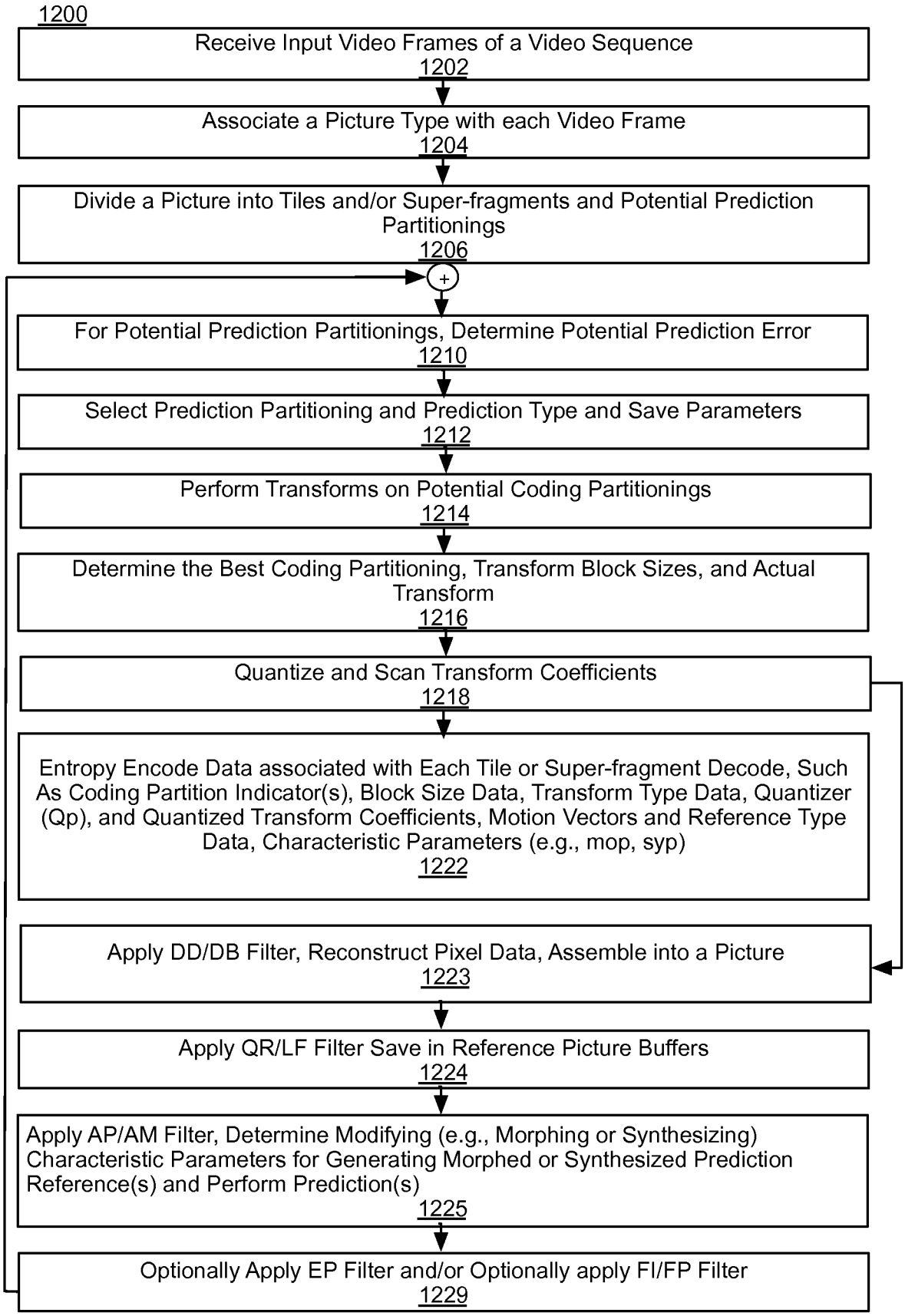

1200
Receive Input Video Frames of a Video Sequence
1202
Associate a Picture Type with each Video Frame
1204
Divide a Picture into Tiles and/or Super-fragments and Potential Prediction Partitionings
1206
+
For Potential Prediction Partitionings, Determine Potential Prediction Error
1210
Select Prediction Partitioning and Prediction Type and Save Parameters
1212
Perform Transforms on Potential Coding Partitionings
1214
Determine the Best Coding Partitioning, Transform Block Sizes, and Actual Transform
1216
Quantize and Scan Transform Coefficients
1218
Entropy Encode Data associated with Each Tile or Super-fragment Decode, Such As Coding Partition Indicator(s), Block Size Data, Transform Type Data, Quantizer (Qp), and Quantized Transform Coefficients, Motion Vectors and Reference Type Data, Characteristic Parameters (e.g., mop, syp)
1222
Apply DD/DB Filter, Reconstruct Pixel Data, Assemble into a Picture
1223
Apply QR/LF Filter Save in Reference Picture Buffers
1224
Apply AP/AM Filter, Determine Modifying (e.g., Morphing or Synthesizing) Characteristic Parameters for Generating Morphed or Synthesized Prediction Reference(s) and Perform Prediction(s)
1225
Optionally Apply EP Filter and/or Optionally apply FI/FP Filter
1229

FIG. 12

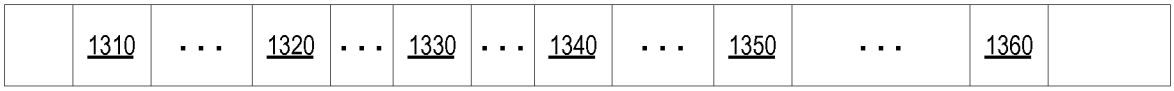

1310
1320
1330
1340
1350
1360
1300

FIG. 13

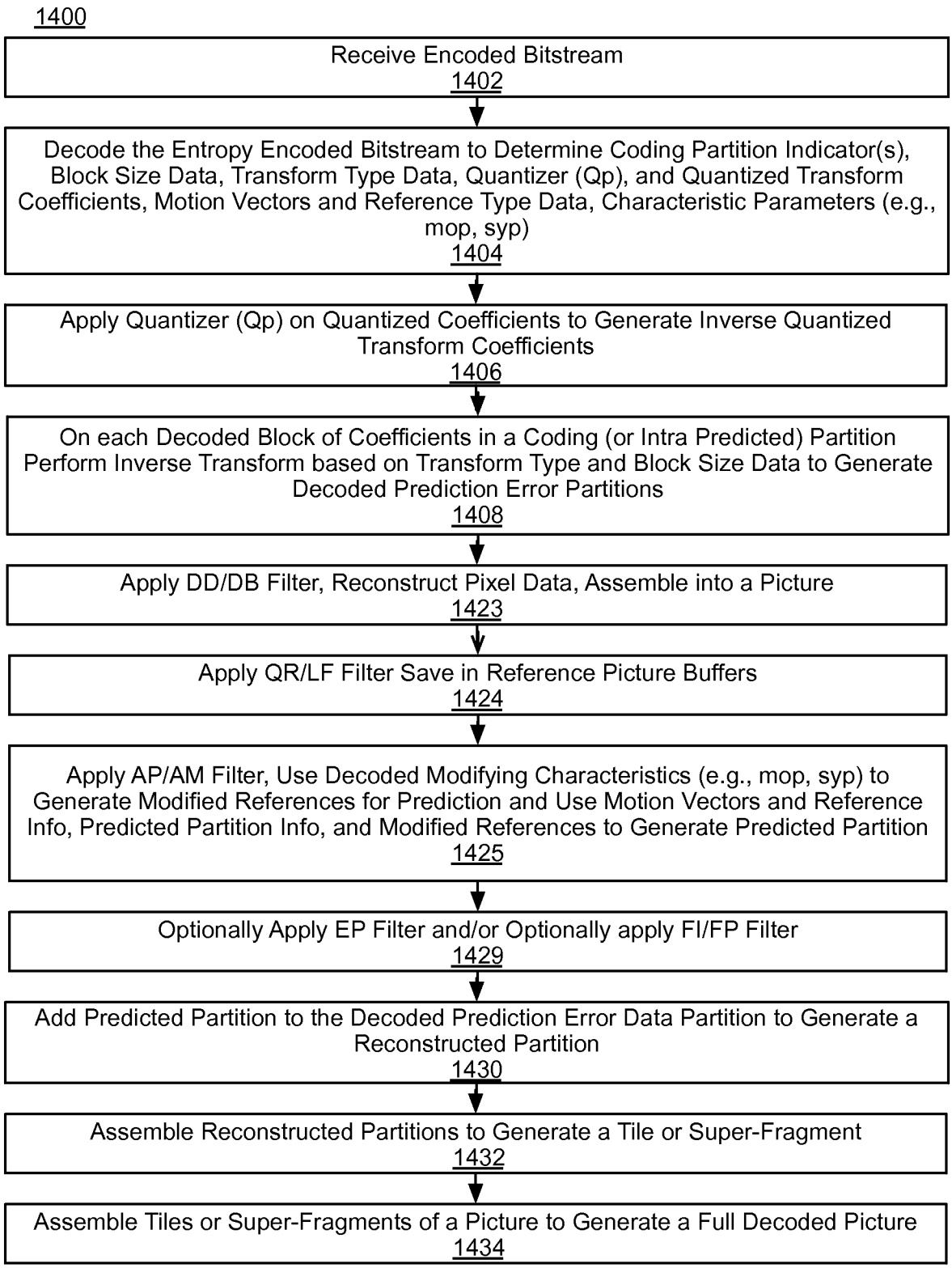

1400
Receive Encoded Bitstream
1402
Decode the Entropy Encoded Bitstream to Determine Coding Partition Indicator(s), Block Size Data, Transform Type Data, Quantizer (Qp), and Quantized Transform Coefficients, Motion Vectors and Reference Type Data, Characteristic Parameters (e.g., mop, syp)
1404
Apply Quantizer (Qp) on Quantized Coefficients to Generate Inverse Quantized Transform Coefficients
1406
On each Decoded Block of Coefficients in a Coding (or Intra Predicted) Partition Perform Inverse Transform based on Transform Type and Block Size Data to Generate Decoded Prediction Error Partitions
1408
Apply DD/DB Filter, Reconstruct Pixel Data, Assemble into a Picture
1423
Apply QR/LF Filter Save in Reference Picture Buffers
1424
Apply AP/AM Filter, Use Decoded Modifying Characteristics (e.g., mop, syp) to Generate Modified References for Prediction and Use Motion Vectors and Reference Info, Predicted Partition Info, and Modified References to Generate Predicted Partition
1425
Optionally Apply EP Filter and/or Optionally apply FI/FP Filter
1429
Add Predicted Partition to the Decoded Prediction Error Data Partition to Generate a Reconstructed Partition
1430
Assemble Reconstructed Partitions to Generate a Tile or Super-Fragment
1432
Assemble Tiles or Super-Fragments of a Picture to Generate a Full Decoded Picture
1434

FIG. 14

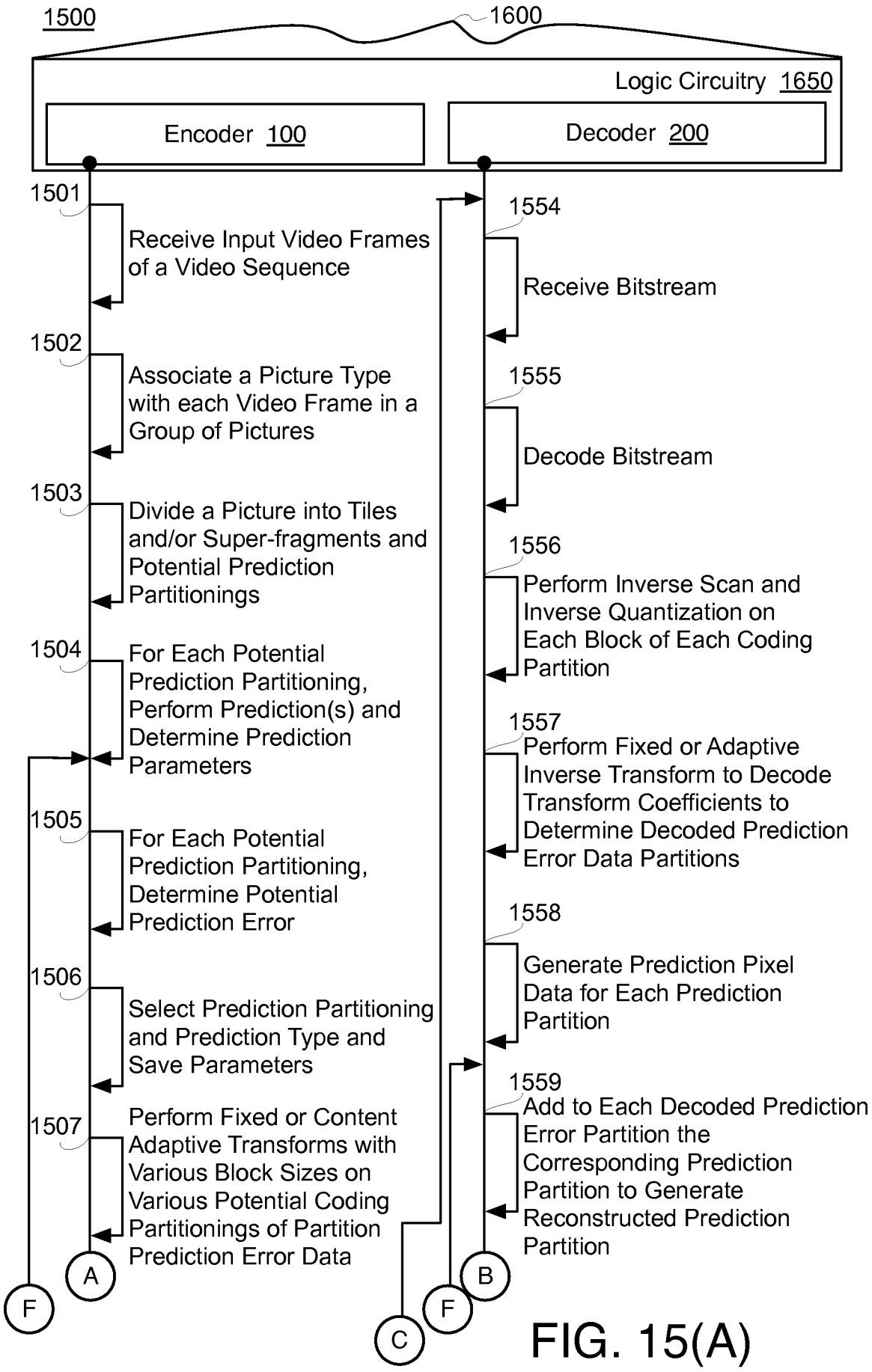

1500
1600
Logic Circuitry 1650
Encoder 100
Decoder 200
1501
Receive Input Video Frames of a Video Sequence
1502
Associate a Picture Type with each Video Frame in a Group of Pictures
1503
Divide a Picture into Tiles and/or Super-fragments and Potential Prediction Partitionings
1504
For Each Potential Prediction Partitioning, Perform Prediction(s) and Determine Prediction Parameters
1505
For Each Potential Prediction Partitioning, Determine Potential Prediction Error
1506
Select Prediction Partitioning and Prediction Type and Save Parameters
1507
Perform Fixed or Content Adaptive Transforms with Various Block Sizes on Various Potential Coding Partitionings of Partition Prediction Error Data
A
F
1554
Receive Bitstream
1555
Decode Bitstream
1556
Perform Inverse Scan and Inverse Quantization on Each Block of Each Coding Partition
1557
Perform Fixed or Adaptive Inverse Transform to Decode Transform Coefficients to Determine Decoded Prediction Error Data Partitions
1558
Generate Prediction Pixel Data for Each Prediction Partition
1559
Add to Each Decoded Prediction Error Partition the Corresponding Prediction Partition to Generate Reconstructed Prediction Partition
B
F
C

FIG. 15(A)

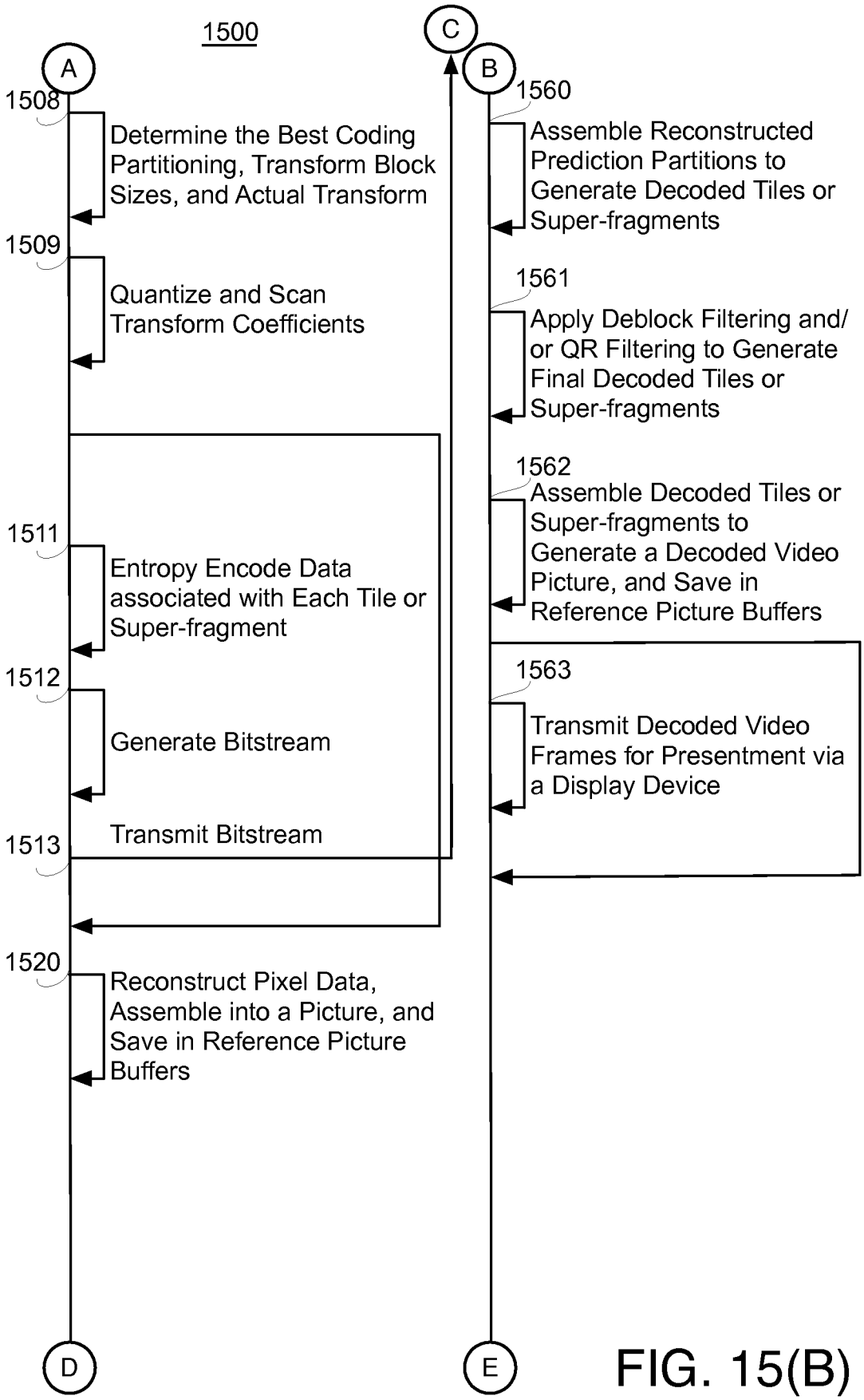

1500
A
1508
Determine the Best Coding Partitioning, Transform Block Sizes, and Actual Transform
1509
Quantize and Scan Transform Coefficients
1511
Entropy Encode Data associated with Each Tile or Super-fragment
1512
Generate Bitstream
1513
Transmit Bitstream
1520
Reconstruct Pixel Data, Assemble into a Picture, and Save in Reference Picture Buffers
D
C
B
1560
Assemble Reconstructed Prediction Partitions to Generate Decoded Tiles or Super-fragments
1561
Apply Deblock Filtering and/ or QR Filtering to Generate Final Decoded Tiles or Super-fragments
1562
Assemble Decoded Tiles or Super-fragments to Generate a Decoded Video Picture, and Save in Reference Picture Buffers
1563
Transmit Decoded Video Frames for Presentment via a Display Device
E

FIG. 15(B)

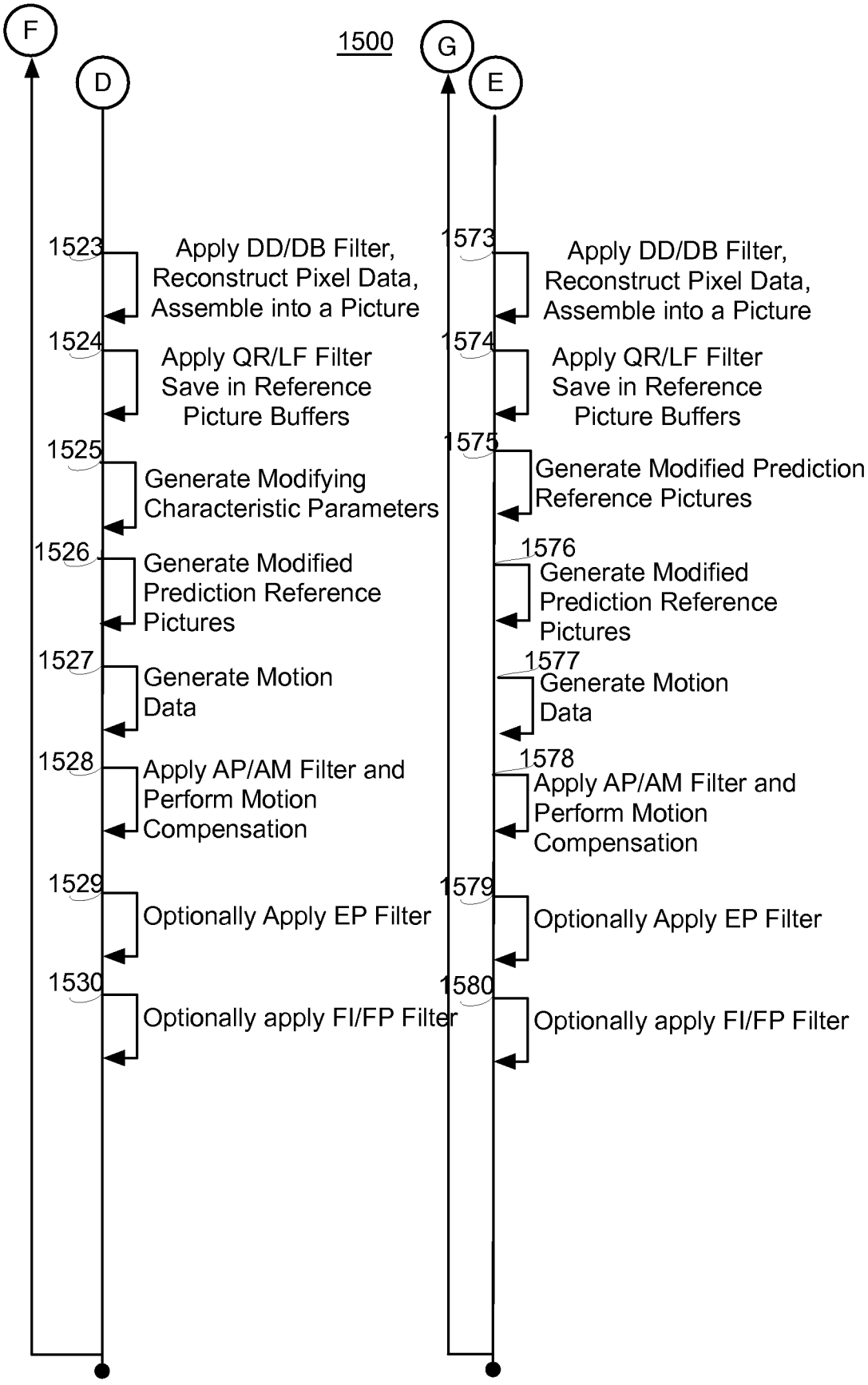

F
D
1500
G
E
1523
Apply DD/DB Filter, Reconstruct Pixel Data, Assemble into a Picture
1524
Apply QR/LF Filter Save in Reference Picture Buffers
1525
Generate Modifying Characteristic Parameters
1526
Generate Modified Prediction Reference Pictures
1527
Generate Motion Data
1528
Apply AP/AM Filter and Perform Motion Compensation
1529
Optionally Apply EP Filter
1530
Optionally apply FI/FP Filter
1573
Apply DD/DB Filter, Reconstruct Pixel Data, Assemble into a Picture
1574
Apply QR/LF Filter Save in Reference Picture Buffers
1575
Generate Modified Prediction Reference Pictures
1576
Generate Modified Prediction Reference Pictures
1577
Generate Motion Data
1578
Apply AP/AM Filter and Perform Motion Compensation
1579
Optionally Apply EP Filter
1580
Optionally apply FI/FP Filter

FIG. 15(C)

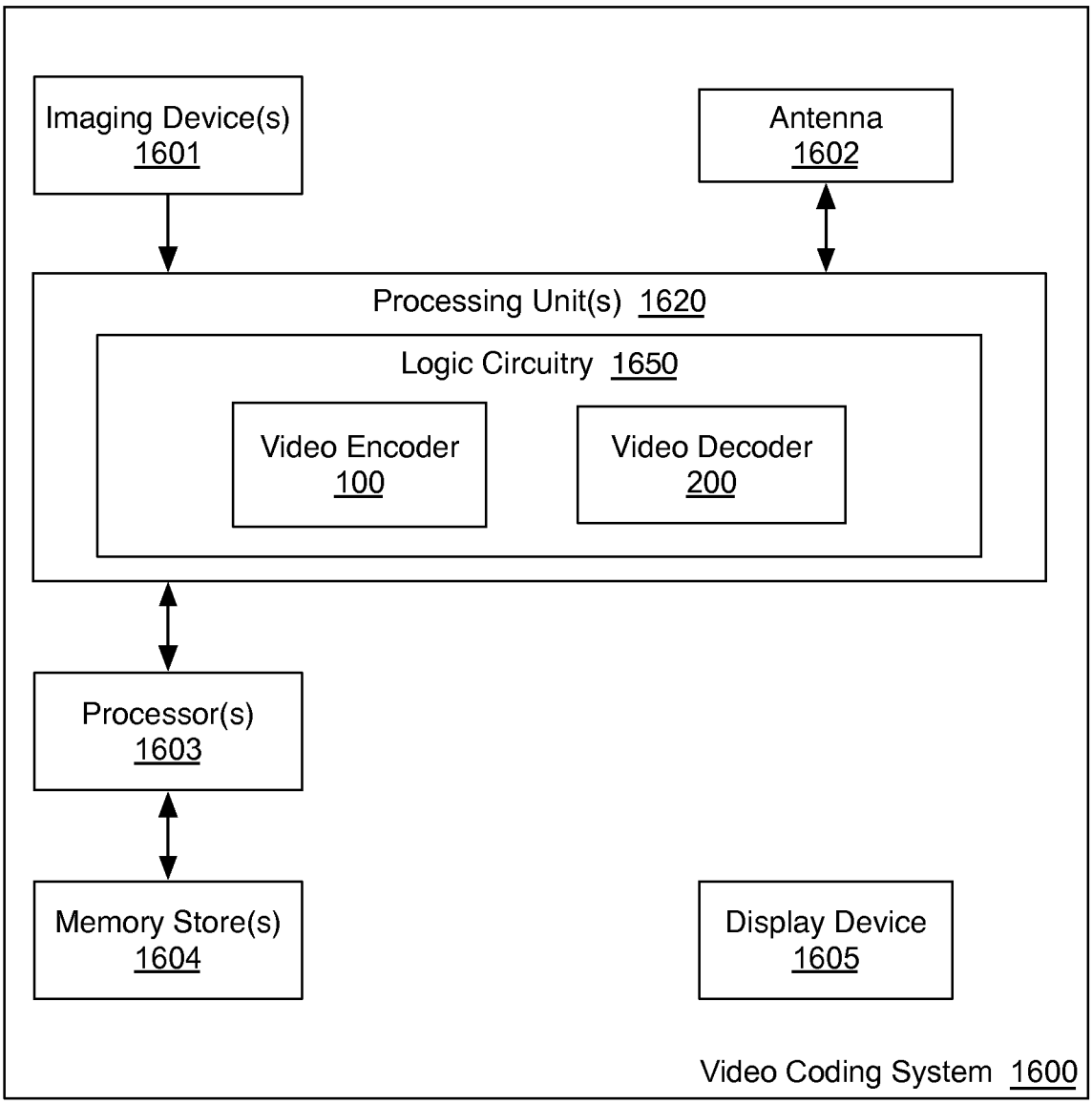

Imaging Device(s)
1601
Antenna
1602
Processing Unit(s) 1620
Logic Circuitry 1650
Video Encoder
100
Video Decoder
200
Processor(s)
1603
Memory Store(s)
1604
Display Device
1605
Video Coding System 1600

FIG. 16

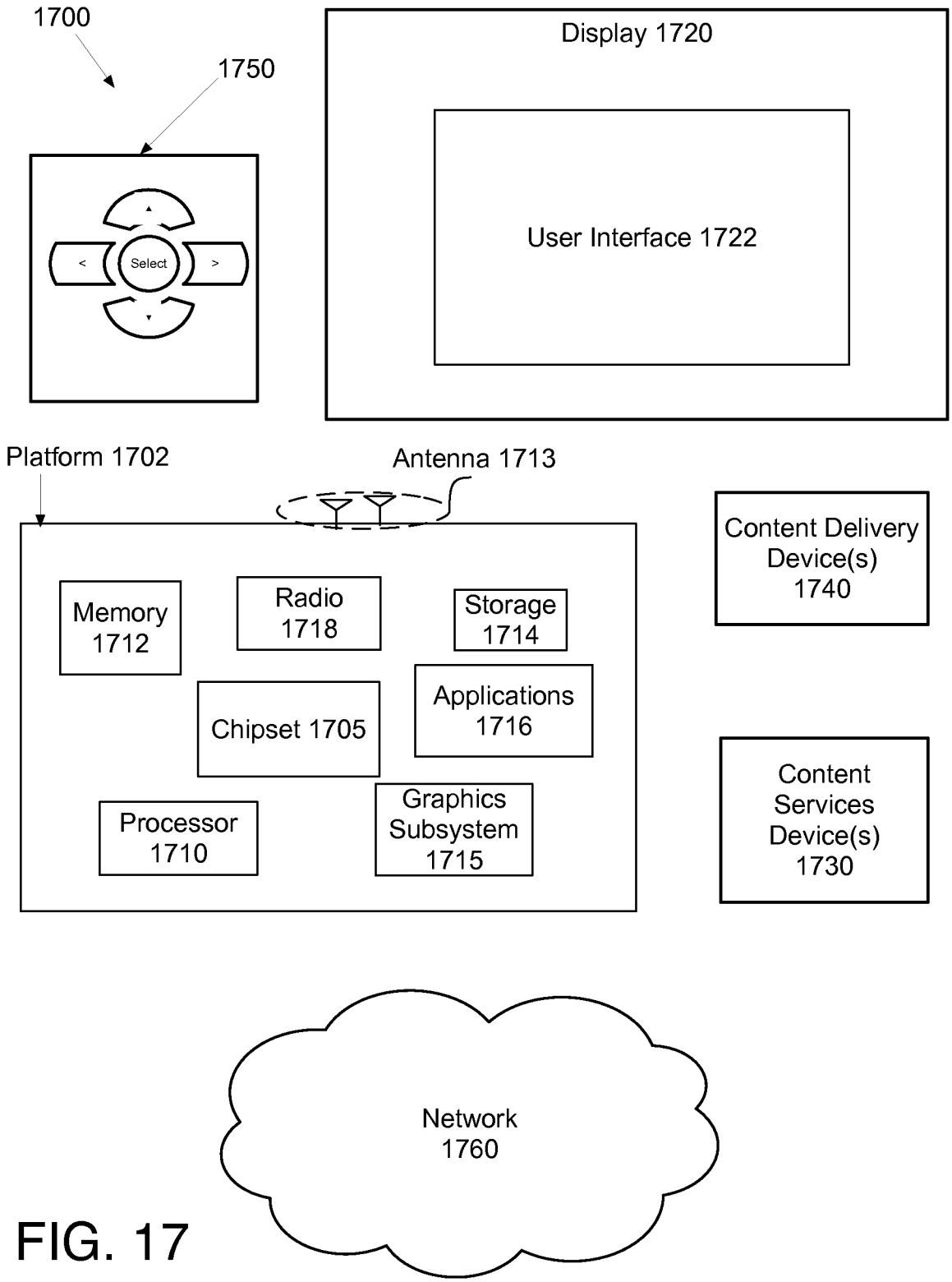

1700
1750
Select
Display 1720
User Interface 1722
Platform 1702
Antenna 1713
Memory 1712
Radio 1718
Storage 1714
Chipset 1705
Applications 1716
Processor 1710
Graphics Subsystem 1715
Content Delivery Device(s) 1740
Content Services Device(s) 1730
Network 1760

FIG. 17

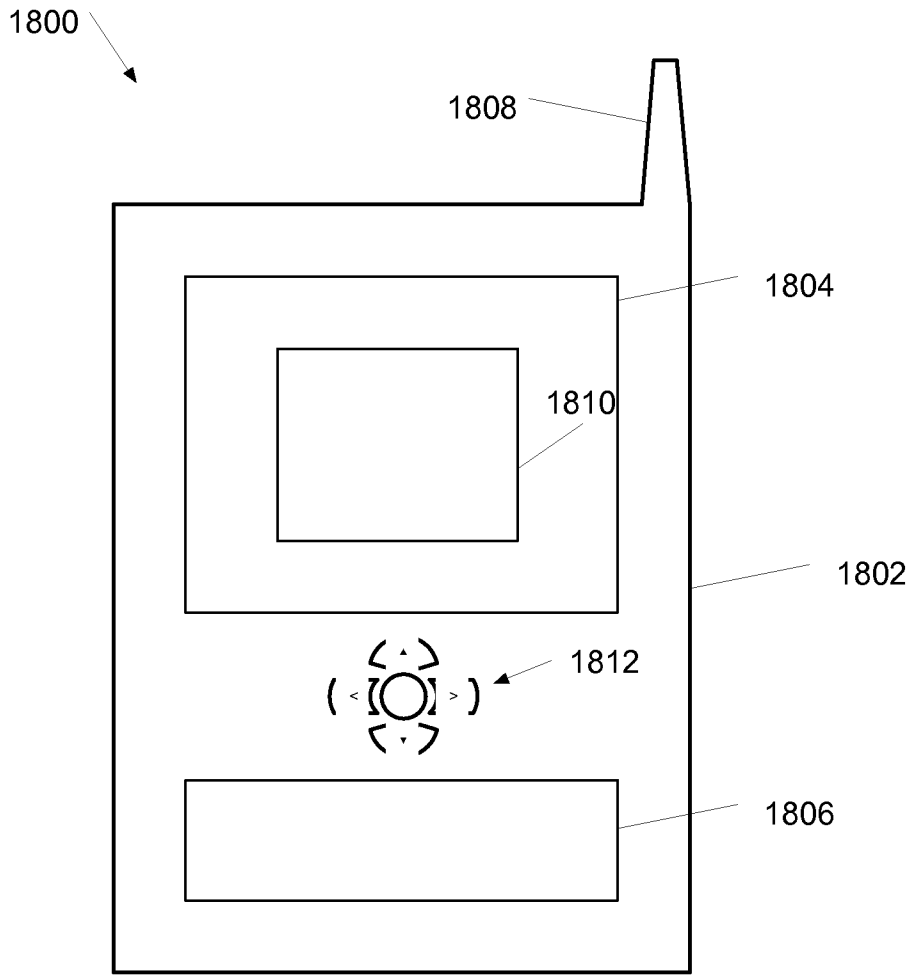


FIG. 18

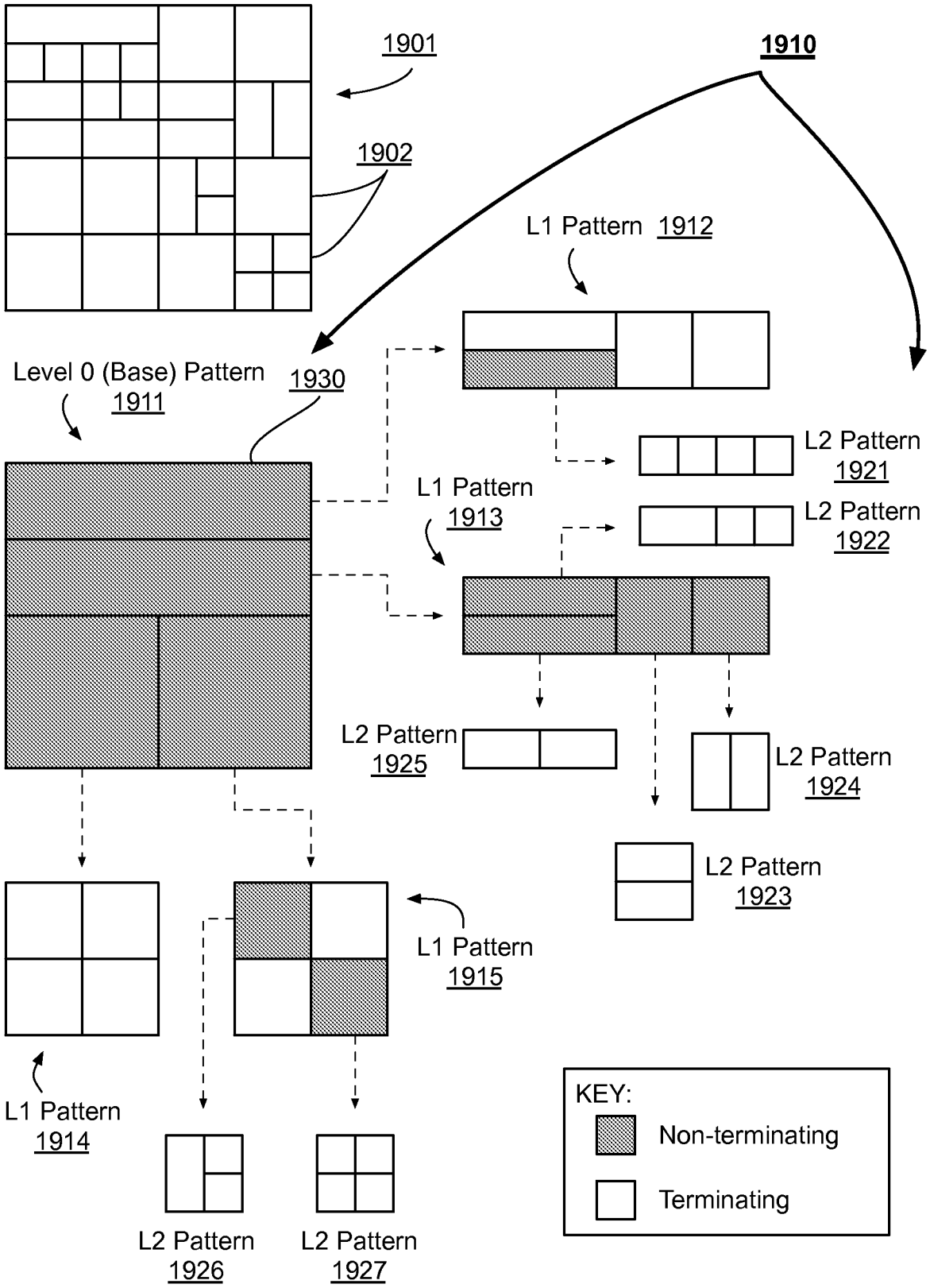

1901
1902
1910
L1 Pattern 1912
Level 0 (Base) Pattern 1911
1930
L2 Pattern 1921
L1 Pattern 1913
L2 Pattern 1922
L2 Pattern 1925
L2 Pattern 1924
L2 Pattern 1923
L1 Pattern 1915
L1 Pattern 1914
L2 Pattern 1926
L2 Pattern 1927
KEY:
Non-terminating
Terminating

FIG. 19

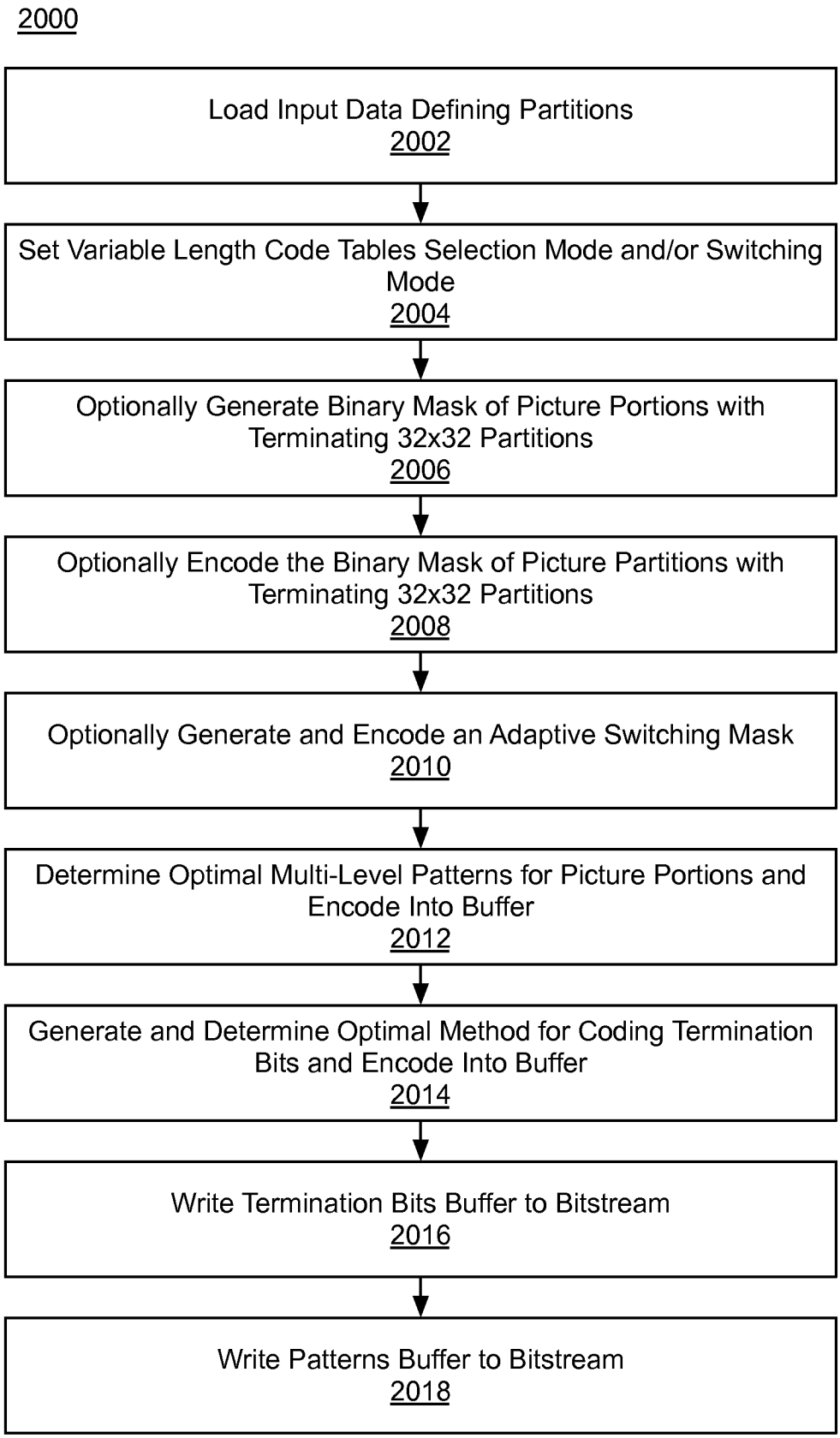

2000
Load Input Data Defining Partitions
2002
Set Variable Length Code Tables Selection Mode and/or Switching Mode
2004
Optionally Generate Binary Mask of Picture Portions with Terminating 32x32 Partitions
2006
Optionally Encode the Binary Mask of Picture Partitions with Terminating 32x32 Partitions
2008
Optionally Generate and Encode an Adaptive Switching Mask
2010
Determine Optimal Multi-Level Patterns for Picture Portions and Encode Into Buffer
2012
Generate and Determine Optimal Method for Coding Termination Bits and Encode Into Buffer
2014
Write Termination Bits Buffer to Bitstream
2016
Write Patterns Buffer to Bitstream
2018

FIG. 20

2100

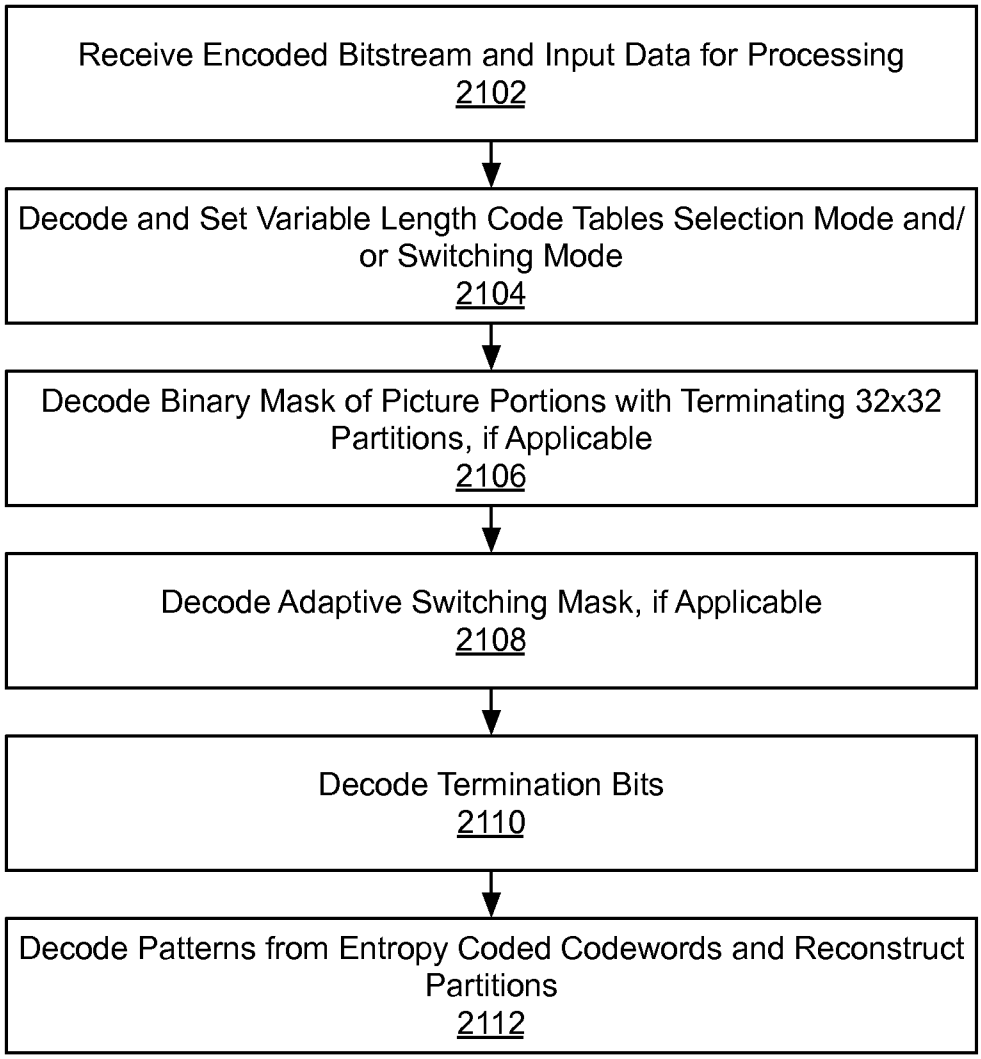

Receive Encoded Bitstream and Input Data for Processing
2102
Decode and Set Variable Length Code Tables Selection Mode and/or Switching Mode
2104
Decode Binary Mask of Picture Portions with Terminating 32x32 Partitions, if Applicable
2106
Decode Adaptive Switching Mask, if Applicable
2108
Decode Termination Bits
2110
Decode Patterns from Entropy Coded Codewords and Reconstruct Partitions
2112

FIG. 21

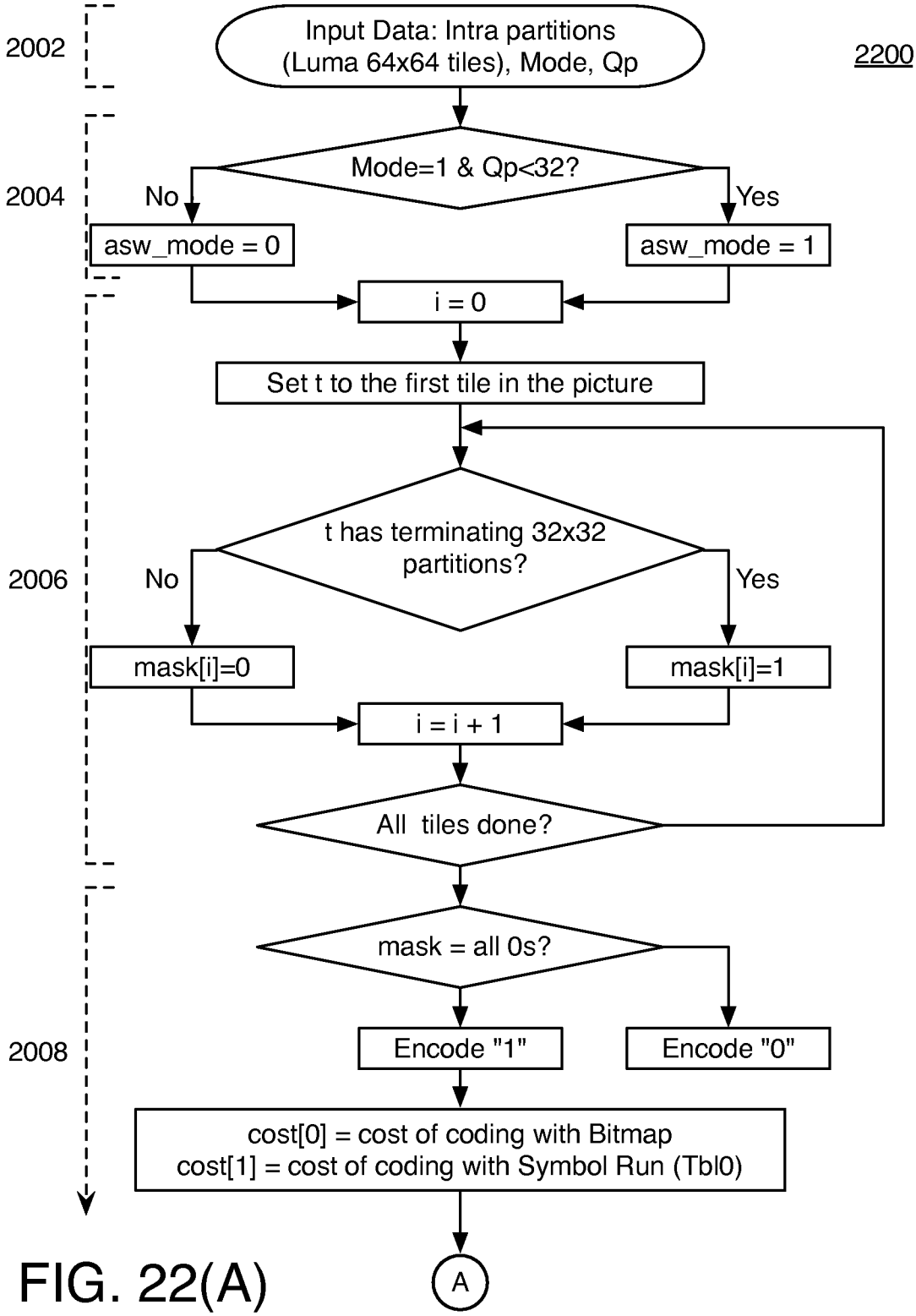

2002
Input Data: Intra partitions
(Luma 64x64 tiles), Mode, Qp
2200
2004
Mode=1 & Qp<32?
No
Yes
asw_mode = 0
asw_mode = 1
i = 0
Set t to the first tile in the picture
2006
t has terminating 32x32
partitions?
No
Yes
mask[i]=0
mask[i]=1
i = i + 1
All tiles done?
mask = all 0s?
2008
Encode "1"
Encode "0"
cost[0] = cost of coding with Bitmap
cost[1] = cost of coding with Symbol Run (Tbl0)
A

FIG. 22(A)

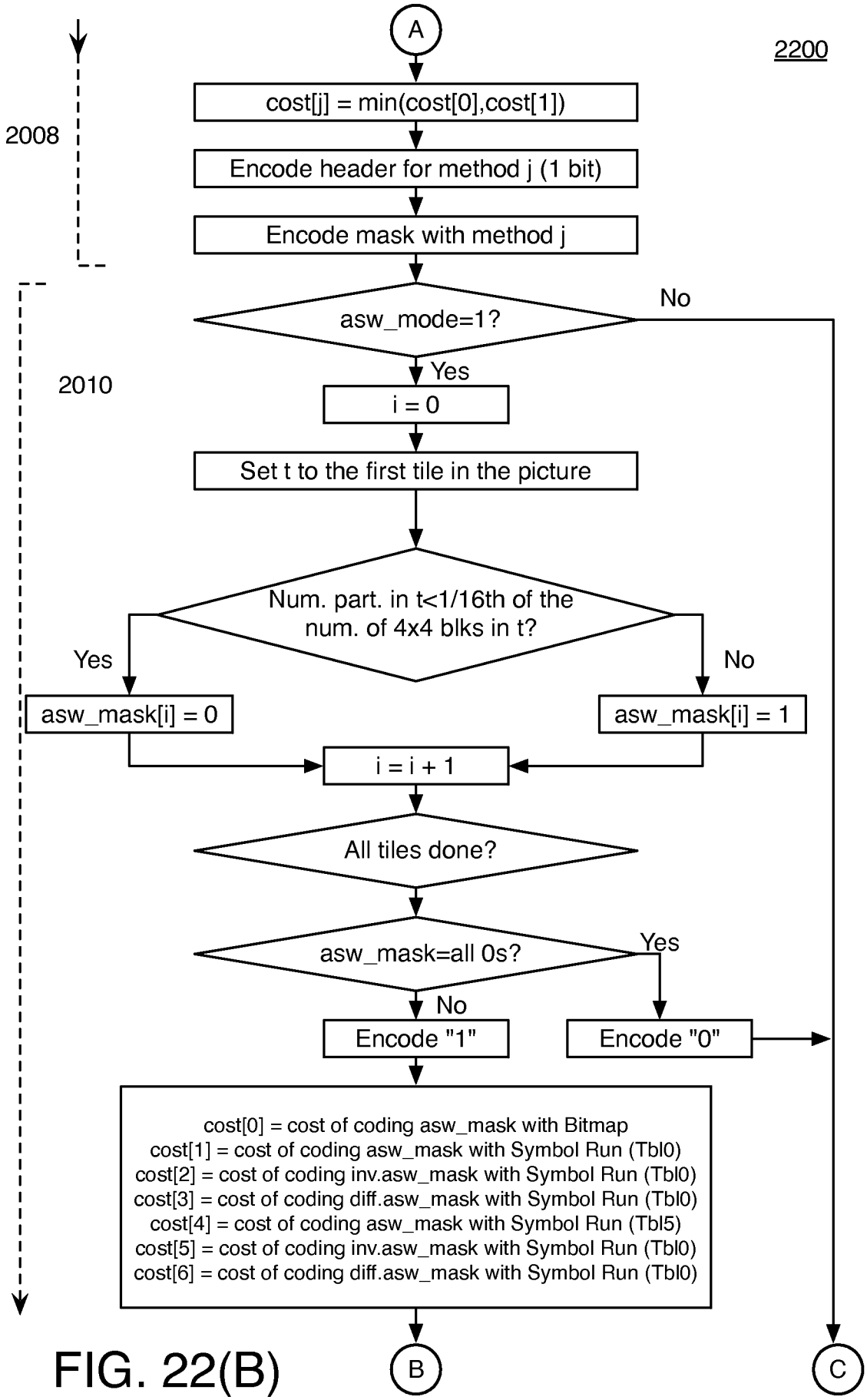

A
2200
2008
cost[j] = min(cost[0],cost[1])
Encode header for method j (1 bit)
Encode mask with method j
asw_mode=1?
No
Yes
2010
i = 0
Set t to the first tile in the picture
Num. part. in t<1/16th of the num. of 4x4 blks in t?
Yes
No
asw_mask[i] = 0
asw_mask[i] = 1
i = i + 1
All tiles done?
asw_mask=all 0s?
Yes
No
Encode "1"
Encode "0"
cost[0] = cost of coding asw_mask with Bitmap
cost[1] = cost of coding asw_mask with Symbol Run (Tbl0)
cost[2] = cost of coding inv.asw_mask with Symbol Run (Tbl0)
cost[3] = cost of coding diff.asw_mask with Symbol Run (Tbl0)
cost[4] = cost of coding asw_mask with Symbol Run (Tbl5)
cost[5] = cost of coding inv.asw_mask with Symbol Run (Tbl0)
cost[6] = cost of coding diff.asw_mask with Symbol Run (Tbl0)
B
C

FIG. 22(B)

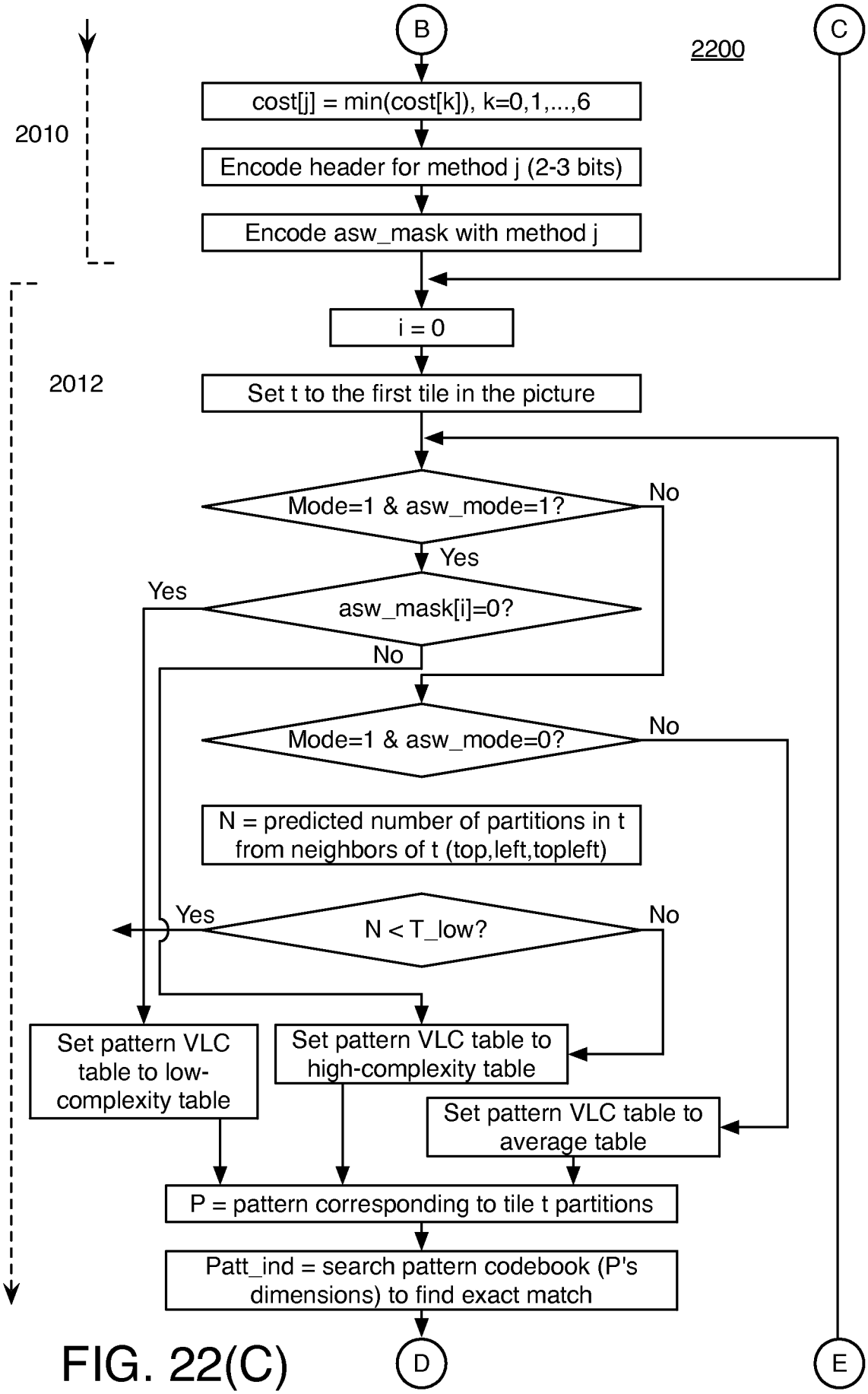

B
C
2200
2010
cost[j] = min(cost[k]), k=0,1,...,6
Encode header for method j (2-3 bits)
Encode asw_mask with method j
i = 0
2012
Set t to the first tile in the picture
Mode=1 & asw_mode=1?
No
Yes
Yes
asw_mask[i]=0?
No
Mode=1 & asw_mode=0?
No
N = predicted number of partitions in t from neighbors of t (top,left,topleft)
Yes
N < T_low?
No
Set pattern VLC table to low-complexity table
Set pattern VLC table to high-complexity table
Set pattern VLC table to average table
P = pattern corresponding to tile t partitions
Patt_ind = search pattern codebook (P's dimensions) to find exact match
D
E

FIG. 22(C)

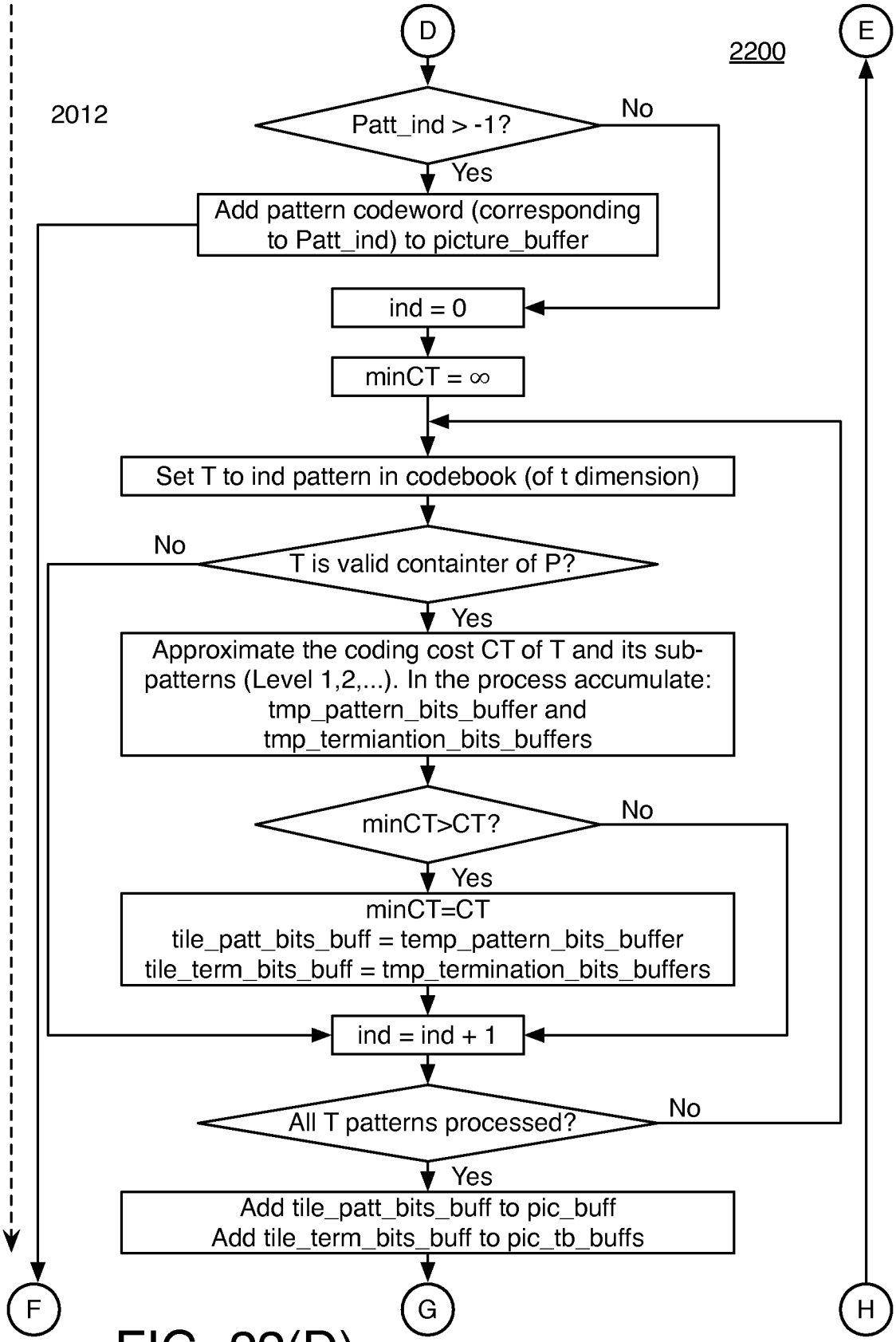

D
E
2200
2012
Patt_ind > -1?
No
Yes
Add pattern codeword (corresponding to Patt_ind) to picture_buffer
ind = 0
minCT = ∞
Set T to ind pattern in codebook (of t dimension)
No
T is valid containter of P?
Yes
Approximate the coding cost CT of T and its sub-patterns (Level 1,2,...). In the process accumulate: tmp_pattern_bits_buffer and tmp_termiantion_bits_buffers
minCT>CT?
No
Yes
minCT=CT
tile_patt_bits_buff = temp_pattern_bits_buffer
tile_term_bits_buff = tmp_termination_bits_buffers
ind = ind + 1
All T patterns processed?
No
Yes
Add tile_patt_bits_buff to pic_buff
Add tile_term_bits_buff to pic_tb_buffs
F
G
H

FIG. 22(D)

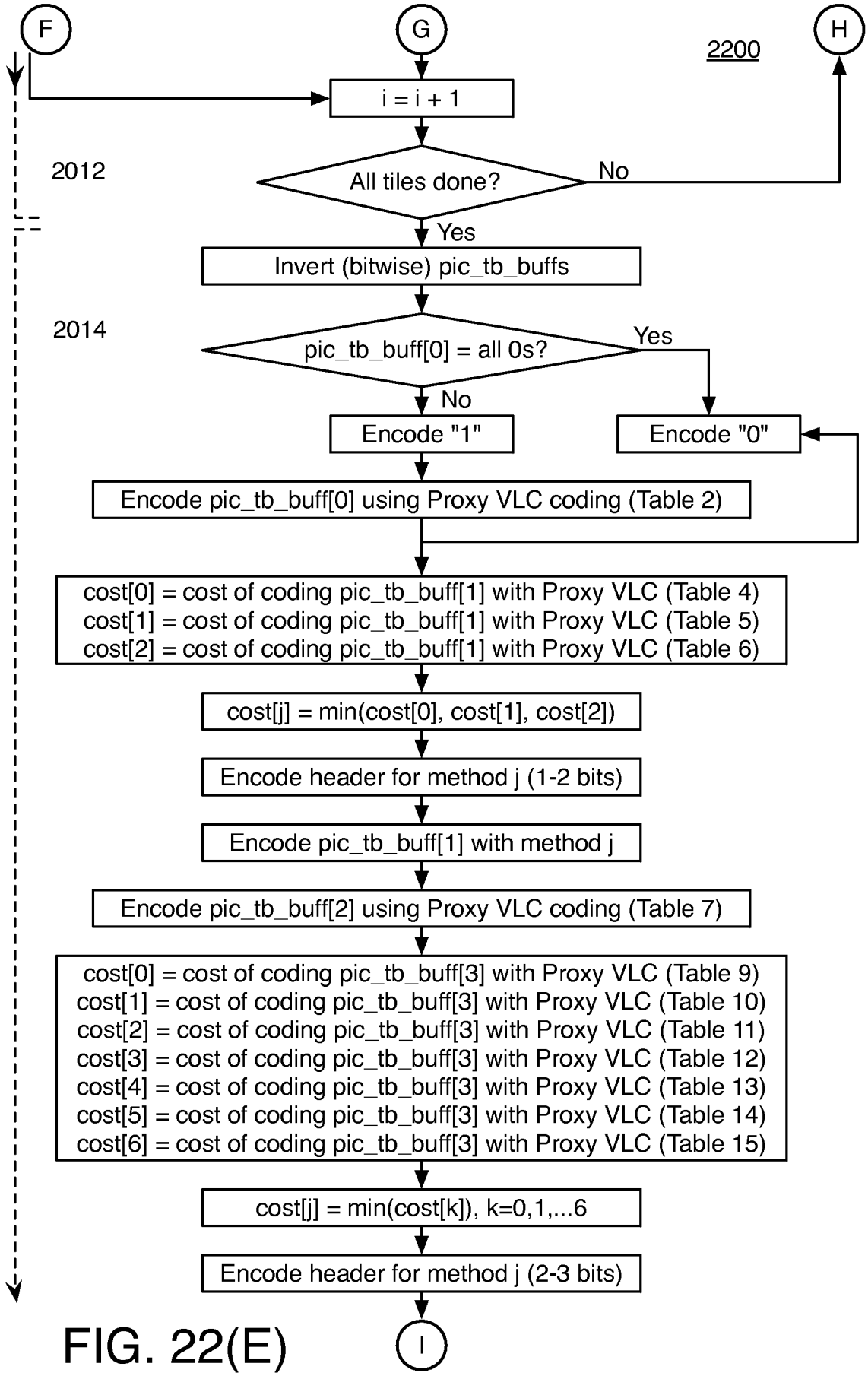

F
G
H
2200
i = i + 1
2012
All tiles done?
No
Yes
Invert (bitwise) pic_tb_buffs
2014
pic_tb_buff[0] = all 0s?
Yes
No
Encode "1"
Encode "0"
Encode pic_tb_buff[0] using Proxy VLC coding (Table 2)
cost[0] = cost of coding pic_tb_buff[1] with Proxy VLC (Table 4)
cost[1] = cost of coding pic_tb_buff[1] with Proxy VLC (Table 5)
cost[2] = cost of coding pic_tb_buff[1] with Proxy VLC (Table 6)
cost[j] = min(cost[0], cost[1], cost[2])
Encode header for method j (1-2 bits)
Encode pic_tb_buff[1] with method j
Encode pic_tb_buff[2] using Proxy VLC coding (Table 7)
cost[0] = cost of coding pic_tb_buff[3] with Proxy VLC (Table 9)
cost[1] = cost of coding pic_tb_buff[3] with Proxy VLC (Table 10)
cost[2] = cost of coding pic_tb_buff[3] with Proxy VLC (Table 11)
cost[3] = cost of coding pic_tb_buff[3] with Proxy VLC (Table 12)
cost[4] = cost of coding pic_tb_buff[3] with Proxy VLC (Table 13)
cost[5] = cost of coding pic_tb_buff[3] with Proxy VLC (Table 14)
cost[6] = cost of coding pic_tb_buff[3] with Proxy VLC (Table 15)
cost[j] = min(cost[k]), k=0,1,...6
Encode header for method j (2-3 bits)
I

FIG. 22(E)

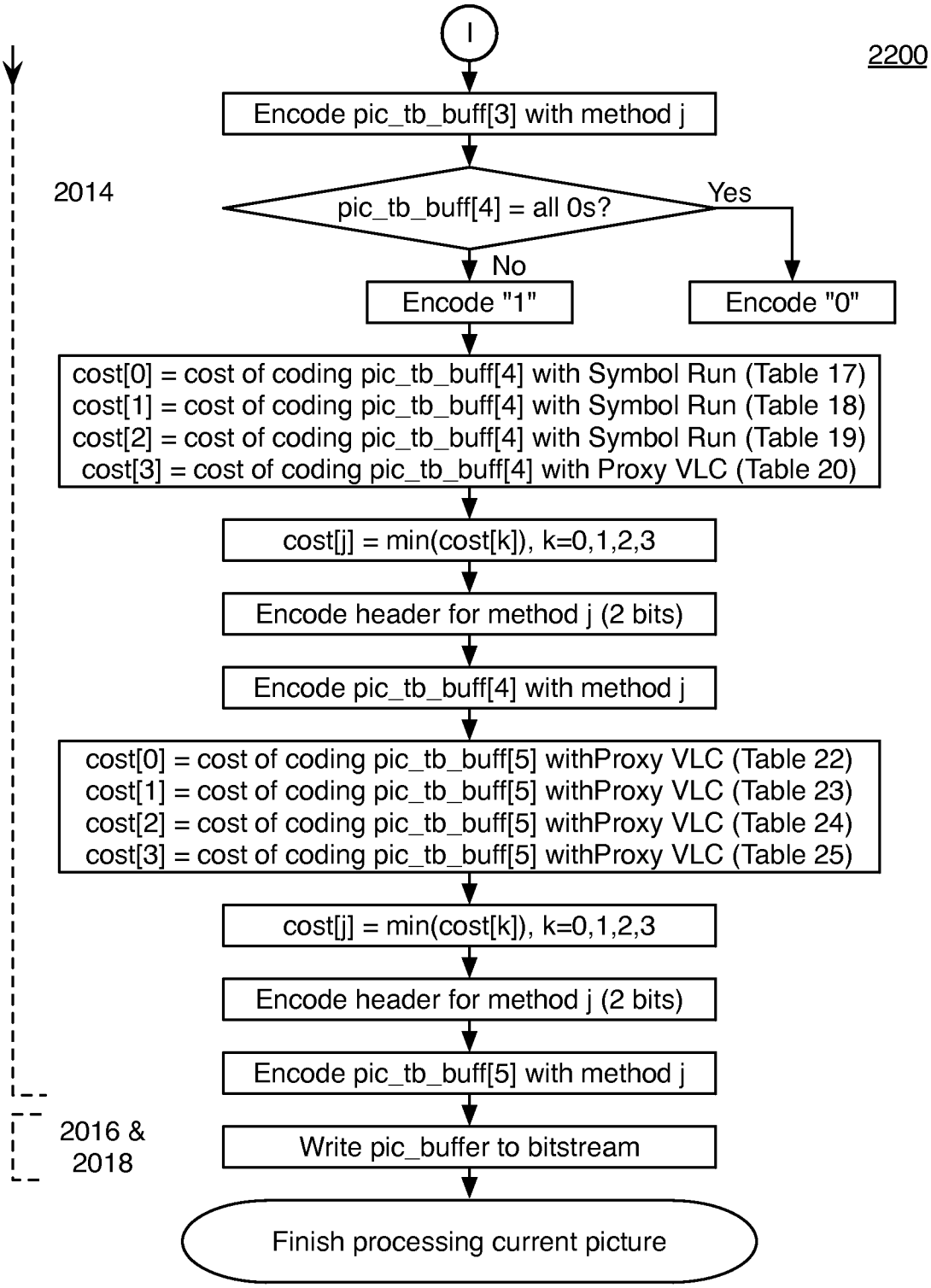

I
2200
Encode pic_tb_buff[3] with method j
2014
pic_tb_buff[4] = all 0s?
Yes
No
Encode "1"
Encode "0"
cost[0] = cost of coding pic_tb_buff[4] with Symbol Run (Table 17)
cost[1] = cost of coding pic_tb_buff[4] with Symbol Run (Table 18)
cost[2] = cost of coding pic_tb_buff[4] with Symbol Run (Table 19)
cost[3] = cost of coding pic_tb_buff[4] with Proxy VLC (Table 20)
cost[j] = min(cost[k]), k=0,1,2,3
Encode header for method j (2 bits)
Encode pic_tb_buff[4] with method j
cost[0] = cost of coding pic_tb_buff[5] withProxy VLC (Table 22)
cost[1] = cost of coding pic_tb_buff[5] withProxy VLC (Table 23)
cost[2] = cost of coding pic_tb_buff[5] withProxy VLC (Table 24)
cost[3] = cost of coding pic_tb_buff[5] withProxy VLC (Table 25)
cost[j] = min(cost[k]), k=0,1,2,3
Encode header for method j (2 bits)
Encode pic_tb_buff[5] with method j
2016 &
2018
Write pic_buffer to bitstream
Finish processing current picture

FIG. 22(F)

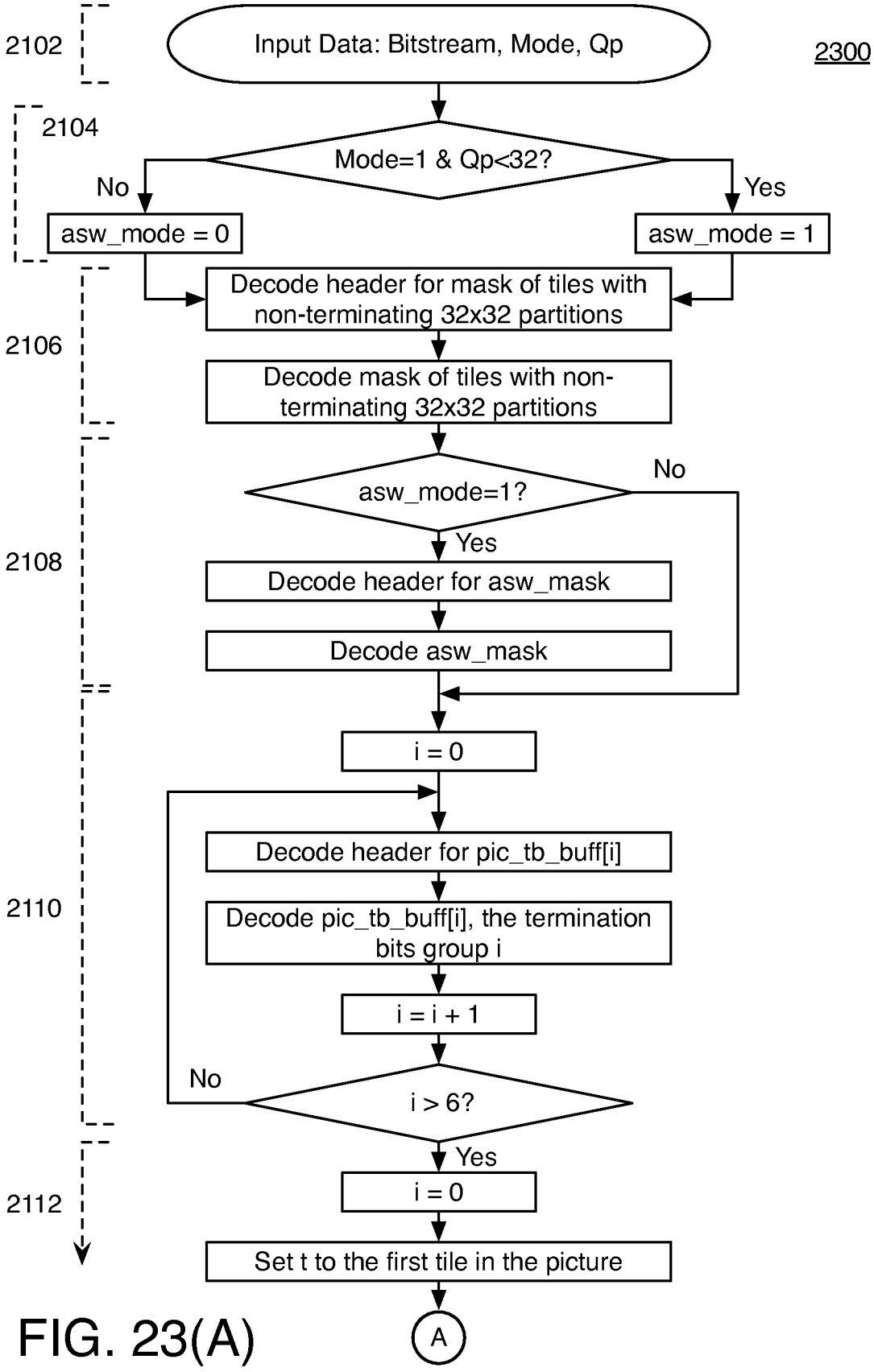

2102
Input Data: Bitstream, Mode, Qp
2300
2104
Mode=1 & Qp<32?
No
Yes
asw_mode = 0
asw_mode = 1
Decode header for mask of tiles with non-terminating 32x32 partitions
2106
Decode mask of tiles with non-terminating 32x32 partitions
asw_mode=1?
No
Yes
2108
Decode header for asw_mask
Decode asw_mask
i = 0
Decode header for pic_tb_buff[i]
2110
Decode pic_tb_buff[i], the termination bits group i
i = i + 1
No
i > 6?
Yes
i = 0
2112
Set t to the first tile in the picture
A

FIG. 23(A)

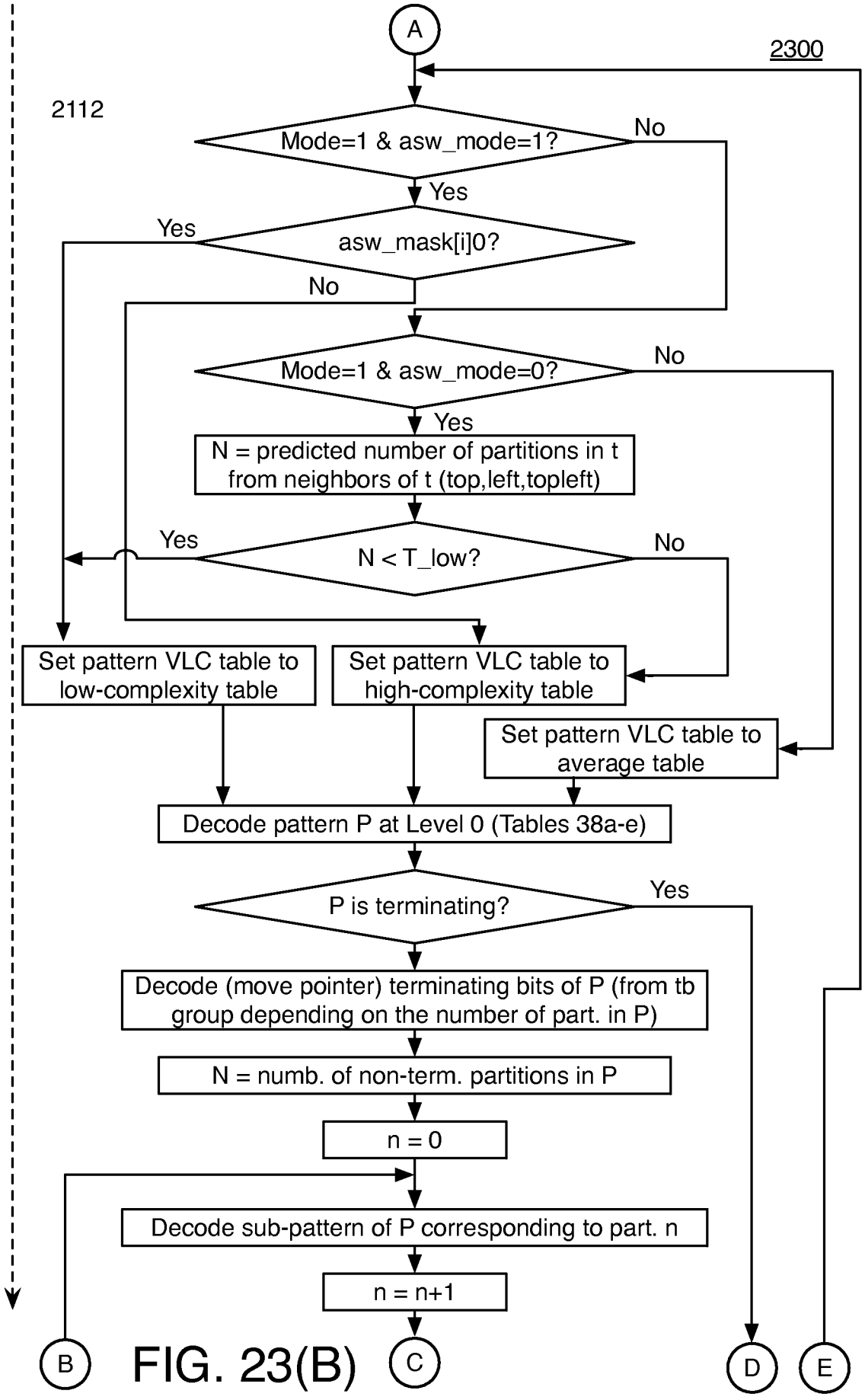

A
2300
2112
Mode=1 & asw_mode=1?
No
Yes
asw_mask[i]0?
Yes
No
Mode=1 & asw_mode=0?
No
Yes
N = predicted number of partitions in t from neighbors of t (top,left,topleft)
N < T_low?
Yes
No
Set pattern VLC table to low-complexity table
Set pattern VLC table to high-complexity table
Set pattern VLC table to average table
Decode pattern P at Level 0 (Tables 38a-e)
P is terminating?
Yes
Decode (move pointer) terminating bits of P (from tb group depending on the number of part. in P)
N = numb. of non-term. partitions in P
n = 0
Decode sub-pattern of P corresponding to part. n
n = n+1
B
C
D
E

FIG. 23(B)

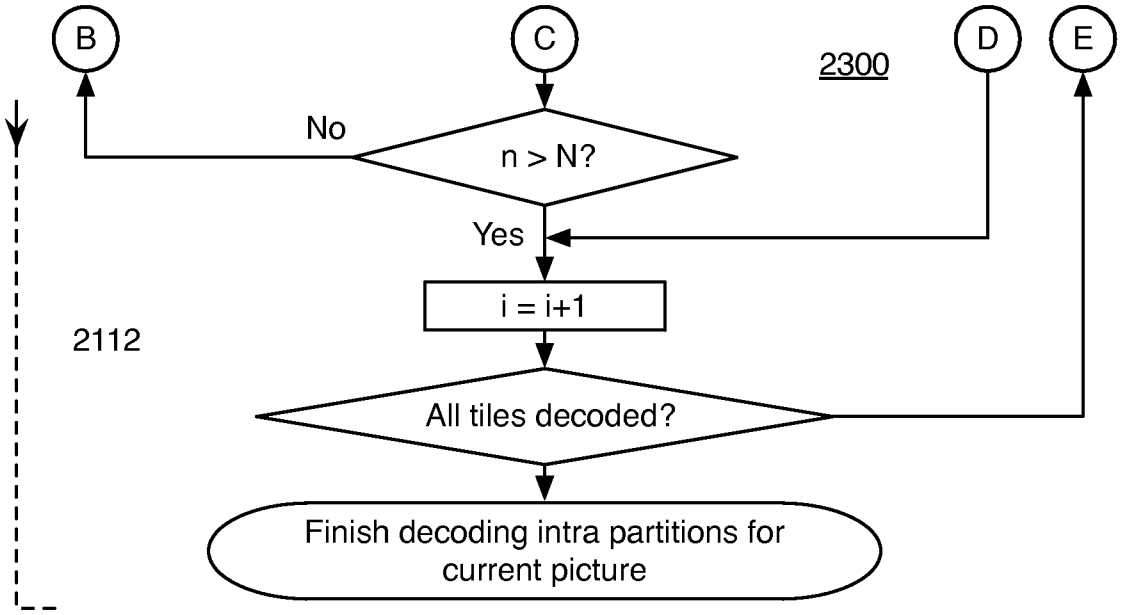

B
C
D
E
2300
No
n > N?
Yes
i = i+1
2112
All tiles decoded?
Finish decoding intra partitions for current picture

FIG. 23(C)

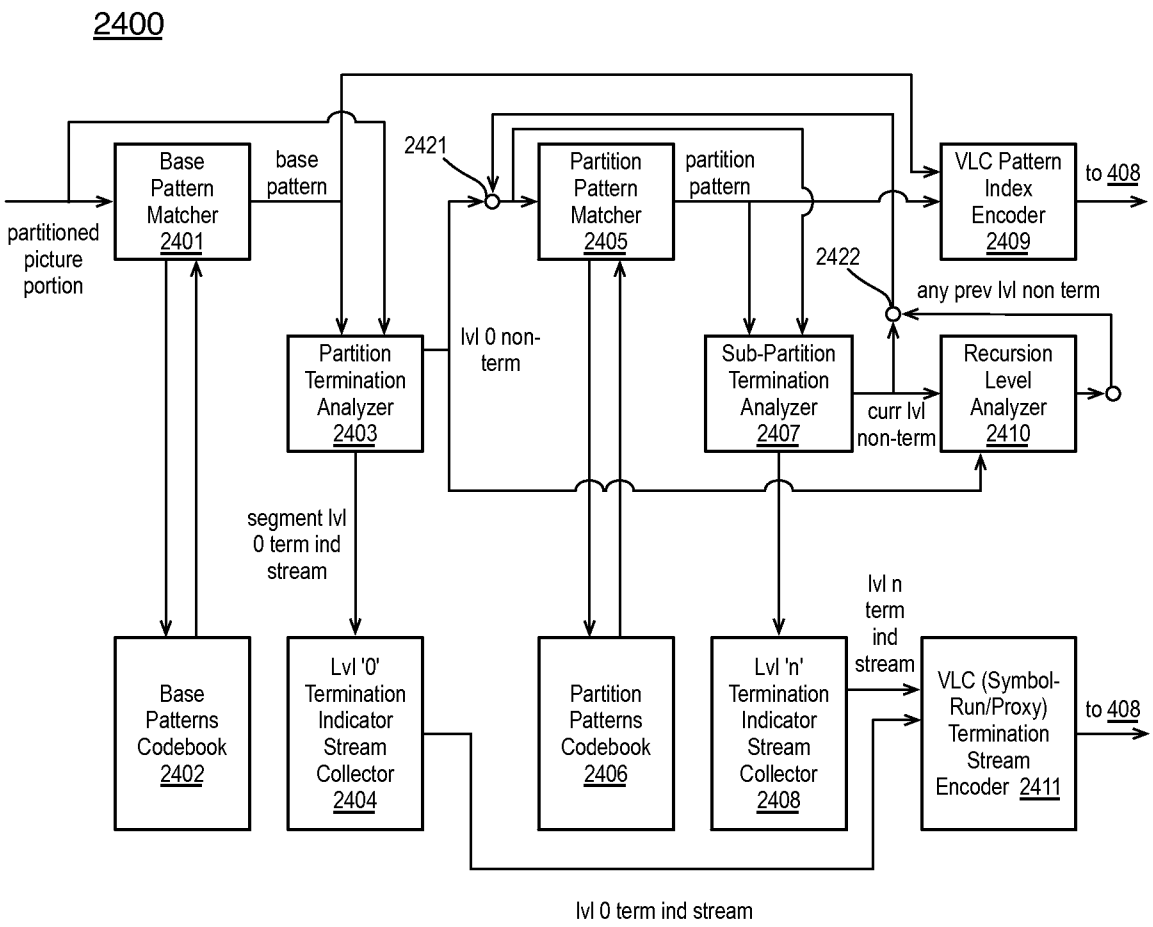

2400
partitioned picture portion
Base Pattern Matcher 2401
base pattern
2421
Partition Pattern Matcher 2405
partition pattern
VLC Pattern Index Encoder 2409
to 408
2422
any prev lvl non term
Partition Termination Analyzer 2403
lvl 0 non-term
Sub-Partition Termination Analyzer 2407
curr lvl non-term
Recursion Level Analyzer 2410
segment lvl 0 term ind stream
lvl n term ind stream
Base Patterns Codebook 2402
Lvl '0' Termination Indicator Stream Collector 2404
Partition Patterns Codebook 2406
Lvl 'n' Termination Indicator Stream Collector 2408
VLC (Symbol-Run/Proxy) Termination Stream Encoder 2411
to 408
lvl 0 term ind stream

FIG. 24

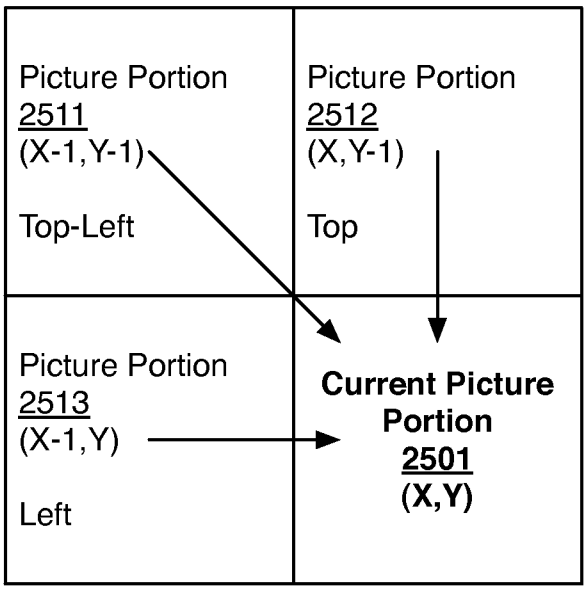

Picture Portion
2511
(X-1,Y-1)
Top-Left
Picture Portion
2512
(X,Y-1)
Top
Picture Portion
2513
(X-1,Y)
Left
Current Picture Portion
2501
(X,Y)

FIG. 25

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2014/013027

A. CLASSIFICATION OF SUBJECT MATTER

H04N 19/91(2014.01)i, H04N 19/13(2014.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04N 19/91; H04N 7/30; H03M 13/05; H04N 7/32; G06F 11/10; H04N 7/26; H04N 7/50; H04N 19/13

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & keywords: encode, pattern, terminate, bit

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	JP 2012-080213 A (MITSUBISHI ELECTRIC CORP.) 19 April 2012 See paragraphs [0002], [0017]-[0025], [0043]-[0047], [0081]-[0087]; and figures 1-2, 7-8.	1-9,17-23,26-31,36-40,43-50,56-57
A		10-16,24-25,32-35,41-42,51-55
Y	US 2010-0269009 A1 (SHUTAI OKAMURA et al.) 21 October 2010 See paragraphs [0123]-[0128]; and figure 8.	1-9,17-23,26-31,36-40,43-50,56-57
A	KR 10-2013-0010124 A (QUALCOMM INCORPORATED) 25 January 2013 See abstract; paragraphs [0117]-[0118]; claim 1; and figure 6.	1-57
A	US 2013-0003837 A1 (YUE YU et al.) 03 January 2013 See abstract; paragraphs [0055]-[0087]; and figures 1-4.	1-57
A	US 2012-0207222 A1 (JIAN LOU et al.) 16 August 2012 See abstract; paragraphs [0029]-[0036]; and figures 2B-3B.	1-57

☐ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 May 2014 (19.05.2014)	**20 May 2014 (20.05.2014)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea
Facsimile No. +82-42-472-7140

Authorized officer
KIM, Seong Woo
Telephone No. +82-42-481-3348

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2014/013027

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
JP 2012-080213 A	19/04/2012	None	
US 2010-0269009 A1	21/10/2010	CN 101904101 A	01/12/2010
		EP 2234275 A1	29/09/2010
		EP 2234275 A4	07/12/2011
		JP 2009-170952 A	30/07/2009
		TW 200943738 A	16/10/2009
		US 8458577 B2	04/06/2013
KR 10-2013-0010124 A	25/01/2013	CN 103098463 A	08/05/2013
		EP 2572509 A2	27/03/2013
		JP 2013-524734 A	17/06/2013
		US 2011-0249721 A1	13/10/2011
		US 2011-0249754 A1	13/10/2011
		WO 2011-130333 A1	20/10/2011
		WO 2011-130334 A2	20/10/2011
		WO 2011-130334 A3	06/12/2012
US 2013-0003837 A1	03/01/2013	KR 10-2014-0016994 A	10/02/2014
		US 2013-0003857 A1	03/01/2013
		WO 2013-003777 A1	03/01/2013
		WO 2013-003791 A1	03/01/2013
US 2012-0207222 A1	16/08/2012	EP 2676443 A1	25/12/2013
		KR 10-2013-0119475 A	31/10/2013
		WO 2012-112384 A1	23/08/2012