US 20250267270A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2025/0267270 A1

Jumakulyyev et al. (43) Pub. Date: Aug. 21, 2025

(54) SEPARATE PROBABILITY INITIALIZATION FOR CONTEXT ADAPTIVE BINARY ARITHMETIC CODING FOR VIDEO CODING

(71) Applicant: **QUALCOMM Incorporated**, San Diego, CA (US)

(72) Inventors: **Ikram Jumakulyyev**, Gröbenzell (DE); **Muhammed Zeyd Coban**, Carlsbad, CA (US); **Vadim Seregin**, San Diego, CA (US); **Marta Karczewicz**, San Diego, CA (US)

(21) Appl. No.: **19/018,825**

(22) Filed: **Jan. 13, 2025**

Related U.S. Application Data

(60) Provisional application No. 63/556,274, filed on Feb. 21, 2024.

Publication Classification

(51) **Int. Cl.**

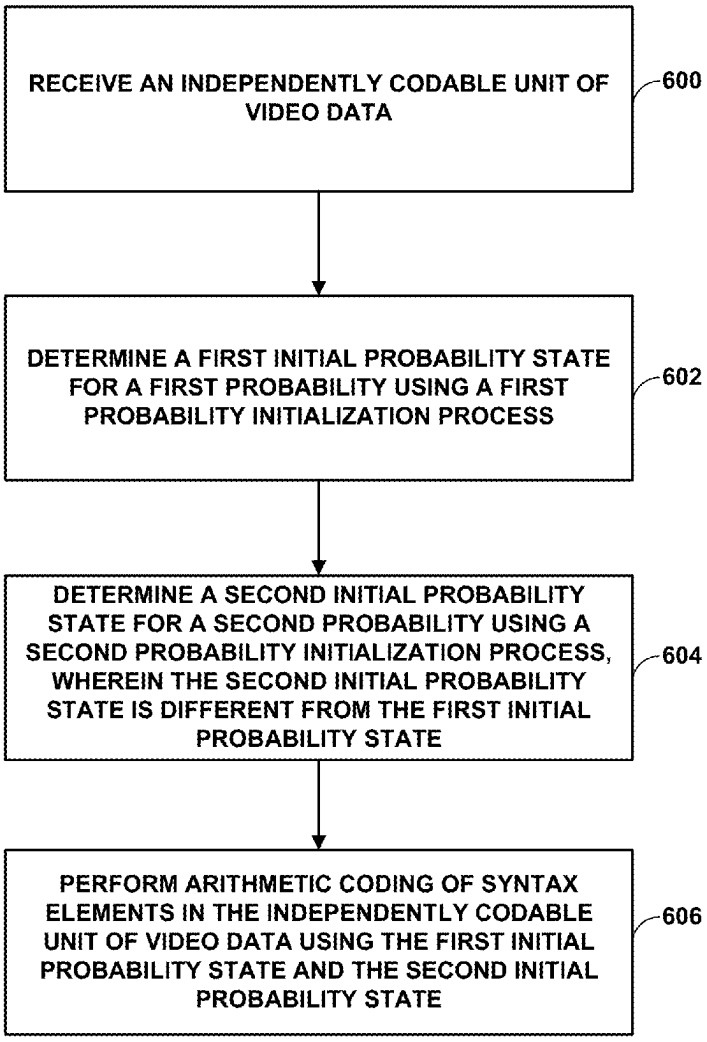
H04N 19/13	(2014.01)
H04N 19/174	(2014.01)
H04N 19/196	(2014.01)
H04N 19/70	(2014.01)

(52) **U.S. Cl.**

CPC ***H04N 19/13*** (2014.11); ***H04N 19/174*** (2014.11); ***H04N 19/196*** (2014.11); ***H04N 19/70*** (2014.11)

(57) **ABSTRACT**

A method of coding video data includes receiving an independently codable unit of video data, determining a first initial probability state for a first probability using a first probability initialization process, determining a second initial probability state for a second probability using a second probability initialization process, wherein the second initial probability state is different from the first initial probability state, and performing arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state.



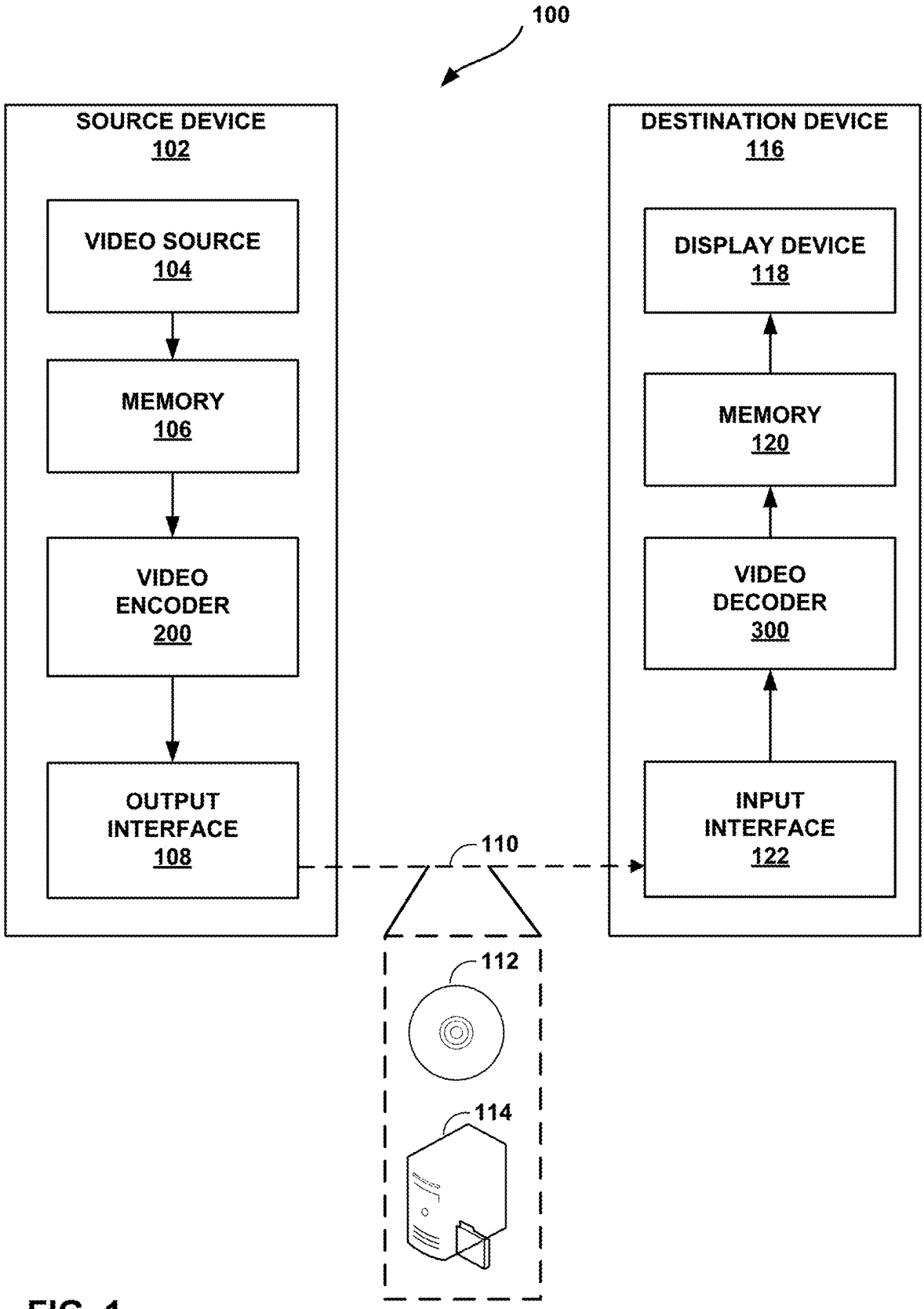

100
SOURCE DEVICE
102
VIDEO SOURCE
104
MEMORY
106
VIDEO
ENCODER
200
OUTPUT
INTERFACE
108
110
112
114
DESTINATION DEVICE
116
DISPLAY DEVICE
118
MEMORY
120
VIDEO
DECODER
300
INPUT
INTERFACE
122

FIG. 1

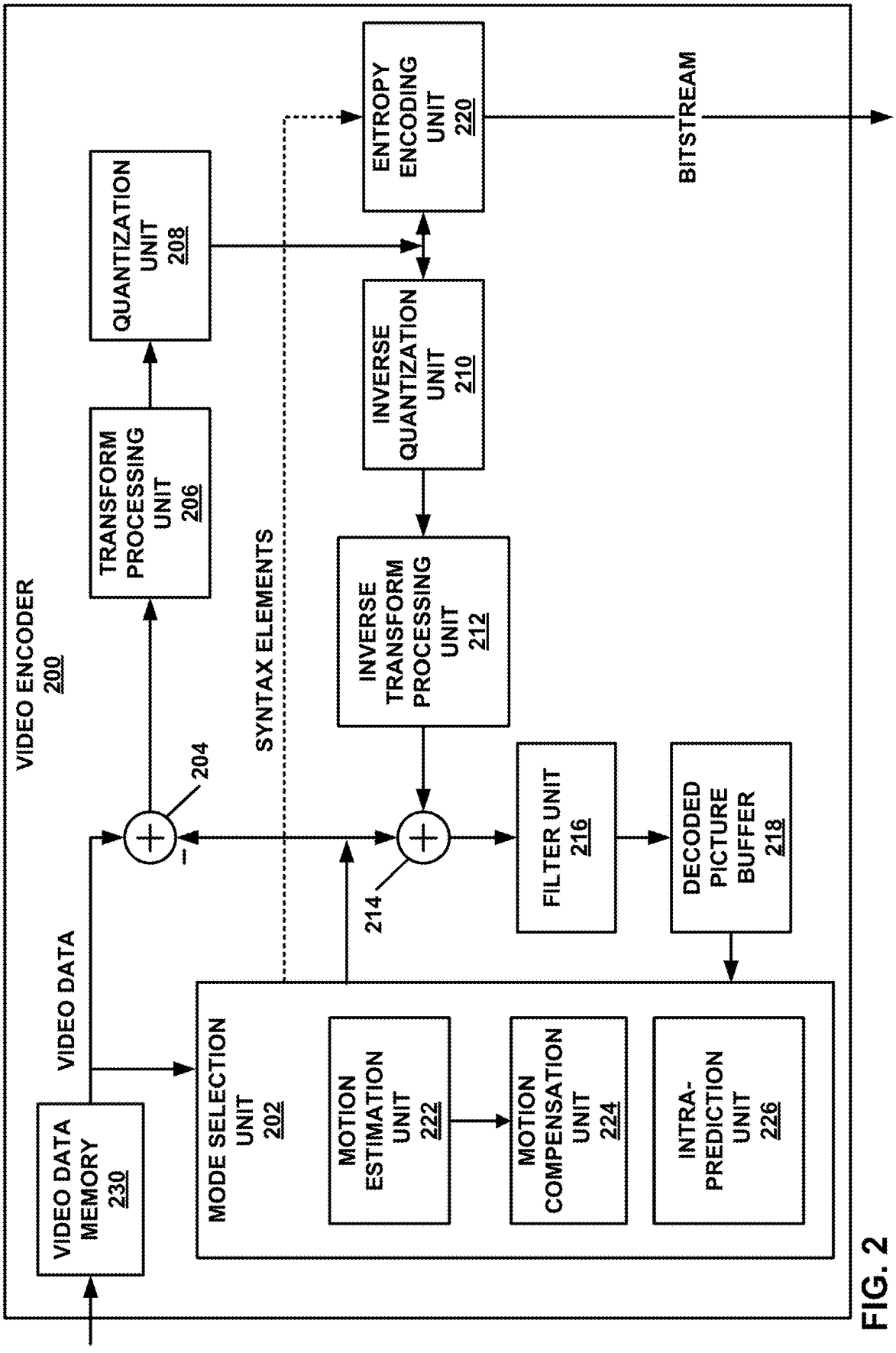

VIDEO ENCODER
200
VIDEO DATA MEMORY
230
VIDEO DATA
MODE SELECTION UNIT
202
MOTION ESTIMATION UNIT
222
MOTION COMPENSATION UNIT
224
INTRA-PREDICTION UNIT
226
204
214
TRANSFORM PROCESSING UNIT
206
QUANTIZATION UNIT
208
INVERSE QUANTIZATION UNIT
210
INVERSE TRANSFORM PROCESSING UNIT
212
FILTER UNIT
216
DECODED PICTURE BUFFER
218
SYNTAX ELEMENTS
ENTROPY ENCODING UNIT
220
BITSTREAM

FIG. 2

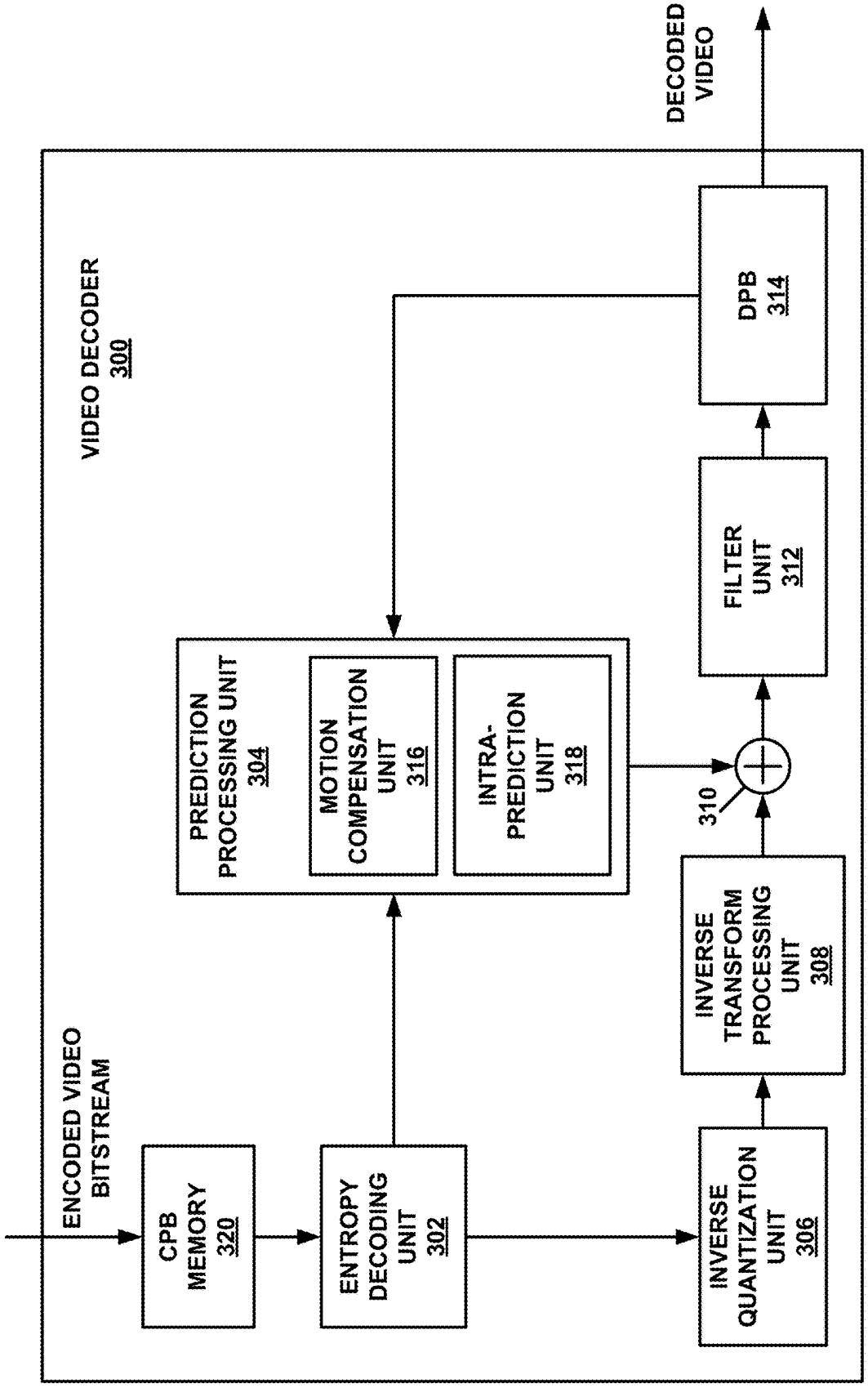

ENCODED VIDEO BITSTREAM
CPB MEMORY 320
ENTROPY DECODING UNIT 302
INVERSE QUANTIZATION UNIT 306
INVERSE TRANSFORM PROCESSING UNIT 308
310
PREDICTION PROCESSING UNIT 304
MOTION COMPENSATION UNIT 316
INTRA-PREDICTION UNIT 318
FILTER UNIT 312
DPB 314
VIDEO DECODER 300
DECODED VIDEO

FIG. 3

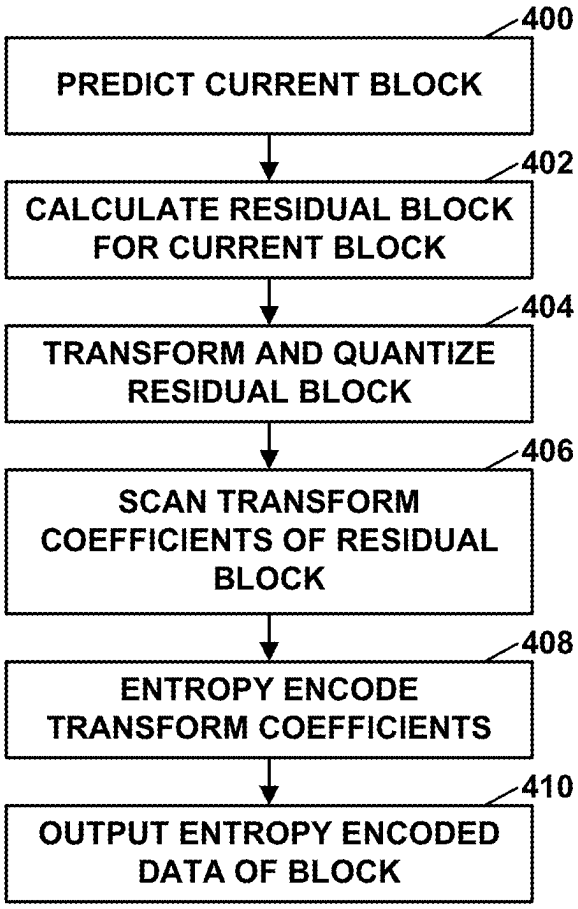

400
PREDICT CURRENT BLOCK
402
CALCULATE RESIDUAL BLOCK FOR CURRENT BLOCK
404
TRANSFORM AND QUANTIZE RESIDUAL BLOCK
406
SCAN TRANSFORM COEFFICIENTS OF RESIDUAL BLOCK
408
ENTROPY ENCODE TRANSFORM COEFFICIENTS
410
OUTPUT ENTROPY ENCODED DATA OF BLOCK

FIG. 4

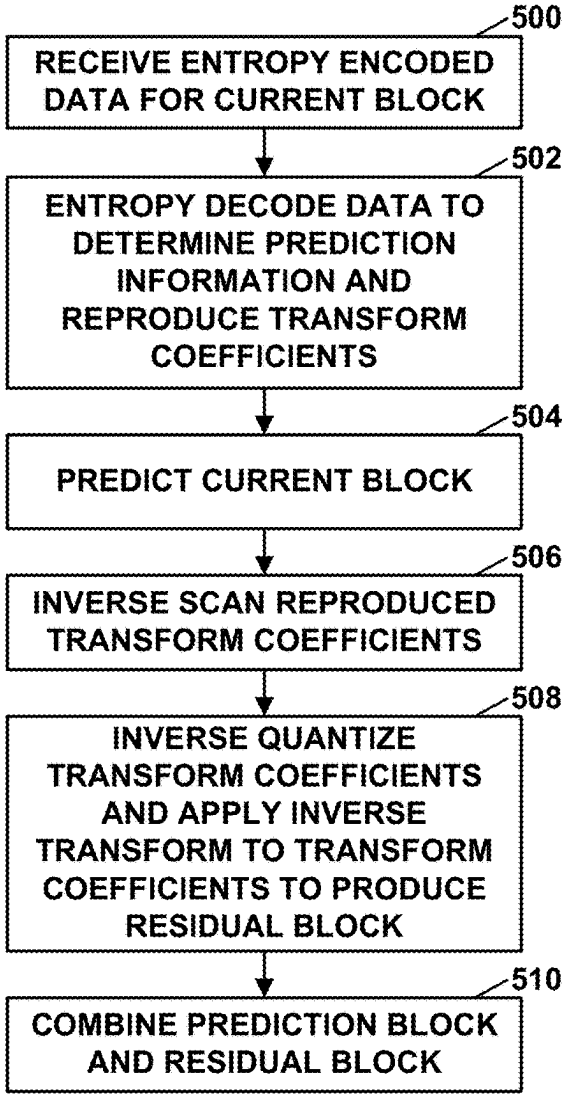

500
RECEIVE ENTROPY ENCODED DATA FOR CURRENT BLOCK
502
ENTROPY DECODE DATA TO DETERMINE PREDICTION INFORMATION AND REPRODUCE TRANSFORM COEFFICIENTS
504
PREDICT CURRENT BLOCK
506
INVERSE SCAN REPRODUCED TRANSFORM COEFFICIENTS
508
INVERSE QUANTIZE TRANSFORM COEFFICIENTS AND APPLY INVERSE TRANSFORM TO TRANSFORM COEFFICIENTS TO PRODUCE RESIDUAL BLOCK
510
COMBINE PREDICTION BLOCK AND RESIDUAL BLOCK

FIG. 5

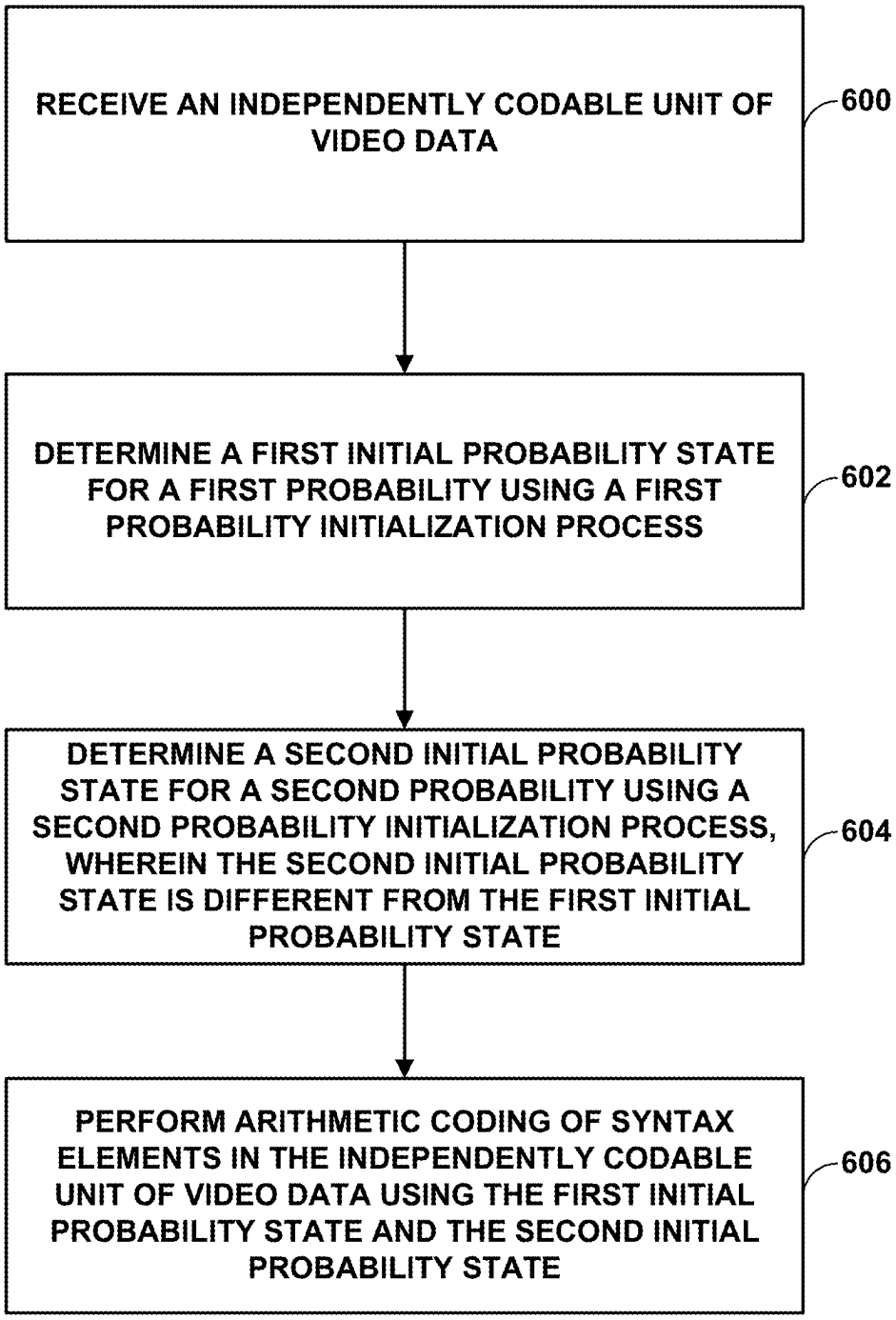

RECEIVE AN INDEPENDENTLY CODABLE UNIT OF VIDEO DATA
600
DETERMINE A FIRST INITIAL PROBABILITY STATE FOR A FIRST PROBABILITY USING A FIRST PROBABILITY INITIALIZATION PROCESS
602
DETERMINE A SECOND INITIAL PROBABILITY STATE FOR A SECOND PROBABILITY USING A SECOND PROBABILITY INITIALIZATION PROCESS, WHEREIN THE SECOND INITIAL PROBABILITY STATE IS DIFFERENT FROM THE FIRST INITIAL PROBABILITY STATE
604
PERFORM ARITHMETIC CODING OF SYNTAX ELEMENTS IN THE INDEPENDENTLY CODABLE UNIT OF VIDEO DATA USING THE FIRST INITIAL PROBABILITY STATE AND THE SECOND INITIAL PROBABILITY STATE
606

FIG. 6

SEPARATE PROBABILITY INITIALIZATION FOR CONTEXT ADAPTIVE BINARY ARITHMETIC CODING FOR VIDEO CODING

[0001] This application claims the benefit of U.S. Provisional Patent Application No. 63/556,274, filed Feb. 21, 2024, the entire content of which is incorporated by reference herein.

TECHNICAL FIELD

[0002] This disclosure relates to video encoding and video decoding.

BACKGROUND

[0003] Digital video capabilities can be incorporated into a wide range of devices, including digital televisions, digital direct broadcast systems, wireless broadcast systems, personal digital assistants (PDAs), laptop or desktop computers, tablet computers, e-book readers, digital cameras, digital recording devices, digital media players, video gaming devices, video game consoles, cellular or satellite radio telephones, so-called "smart phones," video teleconferencing devices, video streaming devices, and the like. Digital video devices implement video coding techniques, such as those described in the standards defined by MPEG-2, MPEG-4, ITU-T H.263, ITU-T H.264/MPEG-4, Part 10, Advanced Video Coding (AVC), ITU-T H.265/High Efficiency Video Coding (HEVC), ITU-T H.266/Versatile Video Coding (VVC), and extensions of such standards, as well as proprietary video codecs/formats such as AOMedia Video 1 (AV1) that was developed by the Alliance for Open Media. The video devices may transmit, receive, encode, decode, and/or store digital video information more efficiently by implementing such video coding techniques.

[0004] Video coding techniques include spatial (intra-picture) prediction and/or temporal (inter-picture) prediction to reduce or remove redundancy inherent in video sequences. For block-based video coding, a video slice (e.g., a video picture or a portion of a video picture) may be partitioned into video blocks, which may also be referred to as coding tree units (CTUs), coding units (CUs) and/or coding nodes. Video blocks in an intra-coded (I) slice of a picture are encoded using spatial prediction with respect to reference samples in neighboring blocks in the same picture. Video blocks in an inter-coded (P or B) slice of a picture may use spatial prediction with respect to reference samples in neighboring blocks in the same picture or temporal prediction with respect to reference samples in other reference pictures. Pictures may be referred to as frames, and reference pictures may be referred to as reference frames.

SUMMARY

[0005] In general, this disclosure describes techniques for a context initialization process that sets the separate initial probability states of each context model used in a binary arithmetic coder. The context initialization process is invoked at the beginning of entropy encoding or decoding an independently decodable unit, such as a slice.

[0006] The techniques of this disclosure are suitable for use with binary arithmetic coders where the probability state represents the real probability in the linear domain, such as the Context Adaptive Binary Arithmetic Coder (CABAC) adopted in Versatile Video Coding (VVC) or other video coding standards and video coding codecs.

[0007] In one example, a method includes receiving an independently codable unit of video data; determining a first initial probability state for a first probability using a first probability initialization process; determining a second initial probability state for a second probability using a second probability initialization process, wherein the second initial probability state is different from the first initial probability state; and performing arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state.

[0008] In another example, a device includes a memory and one or more processors implemented in circuitry and in communication with the memory, the one or more processors configured to receive an independently codable unit of video data; determine a first initial probability state for a first probability using a first probability initialization process; determine a second initial probability state for a second probability using a second probability initialization process, wherein the second initial probability state is different from the first initial probability state; and perform arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state.

[0009] In another example, a device includes means for receiving an independently codable unit of video data; means for determining a first initial probability state for a first probability using a first probability initialization process; means for determining a second initial probability state for a second probability using a second probability initialization process, wherein the second initial probability state is different from the first initial probability state; and means for performing arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state.

[0010] In another example, a computer-readable storage medium is encoded with instructions that, when executed, cause a programmable processor to receive an independently codable unit of video data; determine a first initial probability state for a first probability using a first probability initialization process; determine a second initial probability state for a second probability using a second probability initialization process, wherein the second initial probability state is different from the first initial probability state; and perform arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state.

[0011] The details of one or more examples are set forth in the accompanying drawings and the description below. Other features, objects, and advantages will be apparent from the description, drawings, and claims.

BRIEF DESCRIPTION OF DRAWINGS

[0012] FIG. **1** is a block diagram illustrating an example video encoding and decoding system that may perform the techniques of this disclosure.

[0013] FIG. **2** is a block diagram illustrating an example video encoder that may perform the techniques of this disclosure.

[0014] FIG. **3** is a block diagram illustrating an example video decoder that may perform the techniques of this disclosure.

[0015] FIG. **4** is a flowchart illustrating an example method for encoding a current block in accordance with the techniques of this disclosure.

[0016] FIG. **5** is a flowchart illustrating an example method for decoding a current block in accordance with the techniques of this disclosure.

[0017] FIG. **6** is a flowchart illustrating an example method for coding video data in accordance with the techniques of this disclosure.

DETAILED DESCRIPTION

[0018] Some example video codecs use the same initialization states for two different probability states (e.g., probability states associated with long and short windows) when performing arithmetic coding of syntax elements. Using the same initialization state for two probability states may not lead to optimal compression of syntax elements for all video sequences.

[0019] In general, this disclosure describes techniques for a context initialization process that sets the separate initial probability states of each context model used in a binary arithmetic coder. A video coder may invoke the context initialization process at the beginning of entropy encoding or entropy decoding an independently decodable unit of video data, such as a slice of video data. The techniques of this disclosure are suitable for use with binary arithmetic coders where the probability state represents the real probability in the linear domain, such as the CABAC adopted in Versatile Video Coding (VVC) or other video coding standards and video coding codecs.

[0020] FIG. **1** is a block diagram illustrating an example video encoding and decoding system **100** that may perform the techniques of this disclosure. The techniques of this disclosure are generally directed to coding (encoding and/or decoding) video data. In general, video data includes any data for processing a video. Thus, video data may include raw, unencoded video, encoded video, decoded (e.g., reconstructed) video, and video metadata, such as signaling data.

[0021] As shown in FIG. **1**, system **100** includes a source device **102** that provides encoded video data to be decoded and displayed by a destination device **116**, in this example. In particular, source device **102** provides the video data to destination device **116** via a computer-readable medium **110**. Source device **102** and destination device **116** may be or include any of a wide range of devices, such as desktop computers, notebook (i.e., laptop) computers, mobile devices, tablet computers, set-top boxes, telephone handsets such as smartphones, televisions, cameras, display devices, digital media players, video gaming consoles, video streaming device, broadcast receiver devices, or the like. In some cases, source device **102** and destination device **116** may be equipped for wireless communication, and thus may be referred to as wireless communication devices.

[0022] In the example of FIG. **1**, source device **102** includes video source **104**, memory **106**, video encoder **200**, and output interface **108**. Destination device **116** includes input interface **122**, video decoder **300**, memory **120**, and display device **118**. In accordance with this disclosure, video encoder **200** of source device **102** and video decoder **300** of destination device **116** may be configured to apply the techniques for separate probability initialization for arithmetic coding. Thus, source device **102** represents an example of a video encoding device, while destination device **116** represents an example of a video decoding device. In other examples, a source device and a destination device may include other components or arrangements. For example, source device **102** may receive video data from an external video source, such as an external camera. Likewise, destination device **116** may interface with an external display device, rather than include an integrated display device.

[0023] System **100** as shown in FIG. **1** is merely one example. In general, any digital video encoding and/or decoding device may perform techniques for separate probability initialization for arithmetic coding. Source device **102** and destination device **116** are merely examples of such coding devices in which source device **102** generates coded video data for transmission to destination device **116**. This disclosure refers to a "coding" device as a device that performs coding (encoding and/or decoding) of data. Thus, video encoder **200** and video decoder **300** represent examples of coding devices, in particular, a video encoder and a video decoder, respectively. In some examples, source device **102** and destination device **116** may operate in a substantially symmetrical manner such that each of source device **102** and destination device **116** includes video encoding and decoding components. Hence, system **100** may support one-way or two-way video transmission between source device **102** and destination device **116**, e.g., for video streaming, video playback, video broadcasting, or video telephony.

[0024] In general, video source **104** represents a source of video data (i.e., raw, unencoded video data) and provides a sequential series of pictures (also referred to as "frames") of the video data to video encoder **200**, which encodes data for the pictures. Video source **104** of source device **102** may include a video capture device, such as a video camera, a video archive containing previously captured raw video, and/or a video feed interface to receive video from a video content provider. As a further alternative, video source **104** may generate computer graphics-based data as the source video, or a combination of live video, archived video, and computer-generated video. In each case, video encoder **200** encodes the captured, pre-captured, or computer-generated video data. Video encoder **200** may rearrange the pictures from the received order (sometimes referred to as "display order") into a coding order for coding. Video encoder **200** may generate a bitstream including encoded video data. Source device **102** may then output the encoded video data via output interface **108** onto computer-readable medium **110** for reception and/or retrieval by, e.g., input interface **122** of destination device **116**.

[0025] Memory **106** of source device **102** and memory **120** of destination device **116** represent general purpose memories. In some examples, memories **106**, **120** may store raw video data, e.g., raw video from video source **104** and raw, decoded video data from video decoder **300**. Additionally or alternatively, memories **106**, **120** may store software instructions executable by, e.g., video encoder **200** and video decoder **300**, respectively. Although memory **106** and memory **120** are shown separately from video encoder **200** and video decoder **300** in this example, it should be understood that video encoder **200** and video decoder **300** may also include internal memories for functionally similar or equivalent purposes. Furthermore, memories **106**, **120** may store encoded video data, e.g., output from video encoder

200 and input to video decoder 300. In some examples, portions of memories 106, 120 may be allocated as one or more video buffers, e.g., to store raw, decoded, and/or encoded video data.

[0026] Computer-readable medium 110 may represent any type of medium or device capable of transporting the encoded video data from source device 102 to destination device 116. In one example, computer-readable medium 110 represents a communication medium to enable source device 102 to transmit encoded video data directly to destination device 116 in real-time, e.g., via a radio frequency network or computer-based network. Output interface 108 may modulate a transmission signal including the encoded video data, and input interface 122 may demodulate the received transmission signal, according to a communication standard, such as a wireless communication protocol. The communication medium may include any wireless or wired communication medium, such as a radio frequency (RF) spectrum or one or more physical transmission lines. The communication medium may form part of a packet-based network, such as a local area network, a wide-area network, or a global network such as the Internet. The communication medium may include routers, switches, base stations, or any other equipment that may be useful to facilitate communication from source device 102 to destination device 116.

[0027] In some examples, source device 102 may output encoded data from output interface 108 to storage device 112. Similarly, destination device 116 may access encoded data from storage device 112 via input interface 122. Storage device 112 may include any of a variety of distributed or locally accessed data storage media such as a hard drive, Blu-ray discs, DVDs, CD-ROMs, flash memory, volatile or non-volatile memory, or any other suitable digital storage media for storing encoded video data.

[0028] In some examples, source device 102 may output encoded video data to file server 114 or another intermediate storage device that may store the encoded video data generated by source device 102. Destination device 116 may access stored video data from file server 114 via streaming or download.

[0029] File server 114 may be any type of server device capable of storing encoded video data and transmitting that encoded video data to the destination device 116. File server 114 may represent a web server (e.g., for a website), a server configured to provide a file transfer protocol service (such as File Transfer Protocol (FTP) or File Delivery over Unidirectional Transport (FLUTE) protocol), a content delivery network (CDN) device, a hypertext transfer protocol (HTTP) server, a Multimedia Broadcast Multicast Service (MBMS) or Enhanced MBMS (cMBMS) server, and/or a network attached storage (NAS) device. File server 114 may, additionally or alternatively, implement one or more HTTP streaming protocols, such as Dynamic Adaptive Streaming over HTTP (DASH), HTTP Live Streaming (HLS), Real Time Streaming Protocol (RTSP), HTTP Dynamic Streaming, or the like.

[0030] Destination device 116 may access encoded video data from file server 114 through any standard data connection, including an Internet connection. This may include a wireless channel (e.g., a Wi-Fi connection), a wired connection (e.g., digital subscriber line (DSL), cable modem, etc.), or a combination of both that is suitable for accessing encoded video data stored on file server 114. Input interface 122 may be configured to operate according to any one or more of the various protocols discussed above for retrieving or receiving media data from file server 114, or other such protocols for retrieving media data.

[0031] Output interface 108 and input interface 122 may represent wireless transmitters/receivers, modems, wired networking components (e.g., Ethernet cards), wireless communication components that operate according to any of a variety of IEEE 802.11 standards, or other physical components. In examples where output interface 108 and input interface 122 include wireless components, output interface 108 and input interface 122 may be configured to transfer data, such as encoded video data, according to a cellular communication standard, such as 4G, 4G-LTE (Long-Term Evolution), LTE Advanced, 5G, or the like. In some examples where output interface 108 includes a wireless transmitter, output interface 108 and input interface 122 may be configured to transfer data, such as encoded video data, according to other wireless standards, such as an IEEE 802.11 specification, an IEEE 802.15 specification (e.g., ZigBee™), a Bluetooth™ standard, or the like. In some examples, source device 102 and/or destination device 116 may include respective system-on-a-chip (SoC) devices. For example, source device 102 may include an SoC device to perform the functionality attributed to video encoder 200 and/or output interface 108, and destination device 116 may include an SoC device to perform the functionality attributed to video decoder 300 and/or input interface 122.

[0032] The techniques of this disclosure may be applied to video coding in support of any of a variety of multimedia applications, such as over-the-air television broadcasts, cable television transmissions, satellite television transmissions, Internet streaming video transmissions, such as dynamic adaptive streaming over HTTP (DASH), digital video that is encoded onto a data storage medium, decoding of digital video stored on a data storage medium, or other applications.

[0033] Input interface 122 of destination device 116 receives an encoded video bitstream from computer-readable medium 110 (e.g., a communication medium, storage device 112, file server 114, or the like). The encoded video bitstream may include signaling information defined by video encoder 200, which is also used by video decoder 300, such as syntax elements having values that describe characteristics and/or processing of video blocks or other coded units (e.g., slices, pictures, groups of pictures, sequences, or the like). Display device 118 displays decoded pictures of the decoded video data to a user. Display device 118 may represent any of a variety of display devices such as a liquid crystal display (LCD), a plasma display, an organic light emitting diode (OLED) display, or another type of display device.

[0034] Although not shown in FIG. 1, in some examples, video encoder 200 and video decoder 300 may each be integrated with an audio encoder and/or audio decoder (e.g., audio codec), and may include appropriate MUX-DEMUX units, or other hardware and/or software, to handle multiplexed streams including both audio and video in a common data stream. Example audio codecs may include AAC, AC-3, AC-4, ALAC, ALS, AMBE, AMR, AMR-WB (G.722.2), AMR-WB+, aptx (various versions), ATRAC, BroadVoice (BV16, BV32), CELT, Enhanced AC-3 (E-AC-3), EVS, FLAC, G.711, G.722, G.722.1, G.722.2 (AMR-WB). G.723.1, G.726, G.728, G.729, G.729.1, GSM-FR, HE-AAC, iLBC, iSAC, LA Lyra, Monkey's Audio, MP1,

MP2 (MPEG-1, 2 Audio Layer II), MP3, Musepack, Nellymoser Asao, OptimFROG, Opus, Sac, Satin, SBC, SILK, Siren 7, Speex, SVOPC, True Audio (TTA), TwinVQ, USAC, Vorbis (Ogg), WavPack, and Windows Media Aud.

[0035] Video encoder **200** and video decoder **300** each may be implemented as any of a variety of suitable encoder and/or decoder circuitry that includes a processing system, such as one or more microprocessors, digital signal processors (DSPs), application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), discrete logic, software, hardware, firmware or any combinations thereof. When the techniques are implemented partially in software, a device may store instructions for the software in a suitable, non-transitory computer-readable medium and execute the instructions in hardware using one or more processors to perform the techniques of this disclosure. Each of video encoder **200** and video decoder **300** may be included in one or more encoders or decoders, either of which may be integrated as part of a combined encoder/decoder (CODEC) in a respective device. A device including video encoder **200** and/or video decoder **300** may implement video encoder **200** and/or video decoder **300** in processing circuitry such as an integrated circuit and/or a microprocessor. Such a device may be a wireless communication device, such as a cellular telephone, or any other type of device described herein.

[0036] Video encoder **200** and video decoder **300** may operate according to a video coding standard, such as ITU-T H.265, also referred to as High Efficiency Video Coding (HEVC) or extensions thereto, such as the multi-view and/or scalable video coding extensions. Alternatively, video encoder **200** and video decoder **300** may operate according to other proprietary or industry standards, such as ITU-T H.266, also referred to as Versatile Video Coding (VVC). In other examples, video encoder **200** and video decoder **300** may operate according to a proprietary video codec/format, such as AOMedia Video 1 (AV1), extensions of AV1, and/or successor versions of AV1 (e.g., AV2). In other examples, video encoder **200** and video decoder **300** may operate according to other proprietary formats or industry standards. The techniques of this disclosure, however, are not limited to any particular coding standard or format. In general, video encoder **200** and video decoder **300** may be configured to perform the techniques of this disclosure in conjunction with any video coding techniques that use probability initialization for arithmetic coding.

[0037] In general, video encoder **200** and video decoder **300** may perform block-based coding of pictures. The term "block" generally refers to a structure including data to be processed (e.g., encoded, decoded, or otherwise used in the encoding and/or decoding process). For example, a block may include a two-dimensional matrix of samples of luminance and/or chrominance data. In general, video encoder **200** and video decoder **300** may code video data represented in a YUV (e.g., Y, Cb, Cr) format. That is, rather than coding red, green, and blue (RGB) data for samples of a picture, video encoder **200** and video decoder **300** may code luminance and chrominance components, where the chrominance components may include both red hue and blue hue chrominance components. In some examples, video encoder **200** converts received RGB formatted data to a YUV representation prior to encoding, and video decoder **300** converts the YUV representation to the RGB format. Alternatively, pre- and post-processing units (not shown) may perform these conversions.

[0038] This disclosure may generally refer to coding (e.g., encoding and decoding) of pictures to include the process of encoding or decoding data of the picture. Similarly, this disclosure may refer to coding of blocks of a picture to include the process of encoding or decoding data for the blocks, e.g., prediction and/or residual coding. An encoded video bitstream generally includes a series of values for syntax elements representative of coding decisions (e.g., coding modes) and partitioning of pictures into blocks. Thus, references to coding a picture or a block should generally be understood as coding values for syntax elements forming the picture or block.

[0039] HEVC defines various blocks, including coding units (CUs), prediction units (PUs), and transform units (TUs). According to HEVC, a video coder (such as video encoder **200**) partitions a coding tree unit (CTU) into CUs according to a quadtree structure. That is, the video coder partitions CTUs and CUs into four equal, non-overlapping squares, and each node of the quadtree has either zero or four child nodes. Nodes without child nodes may be referred to as "leaf nodes," and CUs of such leaf nodes may include one or more PUs and/or one or more TUs. The video coder may further partition PUs and TUs. For example, in HEVC, a residual quadtree (RQT) represents partitioning of TUs. In HEVC, PUs represent inter-prediction data, while TUs represent residual data. CUs that are intra-predicted include intra-prediction information, such as an intra-mode indication.

[0040] As another example, video encoder **200** and video decoder **300** may be configured to operate according to VVC. According to VVC, a video coder (such as video encoder **200**) partitions a picture into a plurality of CTUs. Video encoder **200** may partition a CTU according to a tree structure, such as a quadtree-binary tree (QTBT) structure or Multi-Type Tree (MTT) structure. The QTBT structure removes the concepts of multiple partition types, such as the separation between CUs, PUs, and TUs of HEVC. A QTBT structure includes two levels: a first level partitioned according to quadtree partitioning, and a second level partitioned according to binary tree partitioning. A root node of the QTBT structure corresponds to a CTU. Leaf nodes of the binary trees correspond to CUs.

[0041] In an MTT partitioning structure, blocks may be partitioned using a quadtree (QT) partition, a binary tree (BT) partition, and one or more types of triple tree (TT) (also called ternary trec (TT)) partitions. A triple or ternary tree partition is a partition where a block is split into three sub-blocks. In some examples, a triple or ternary tree partition divides a block into three sub-blocks without dividing the original block through the center. The partitioning types in MTT (e.g., QT, BT, and TT), may be symmetrical or asymmetrical.

[0042] When operating according to the AV1 codec, video encoder **200** and video decoder **300** may be configured to code video data in blocks. In AV1, the largest coding block that can be processed is called a superblock. In AV1, a superblock can be either 128×128 luma samples or 64×64 luma samples. However, in successor video coding formats (e.g., AV2), a superblock may be defined by different (e.g., larger) luma sample sizes. In some examples, a superblock is the top level of a block quadtree. Video encoder **200** may further partition a superblock into smaller coding blocks. Video encoder **200** may partition a superblock and other coding blocks into smaller blocks using square or non-

square partitioning. Non-square blocks may include N/2×N, N×N/2, N/4×N, and N×N/4 blocks. Video encoder **200** and video decoder **300** may perform separate prediction and transform processes on each of the coding blocks.

[0043] AV1 also defines a tile of video data. A tile is a rectangular array of superblocks that may be coded independently of other tiles. That is, video encoder **200** and video decoder **300** may encode and decode, respectively, coding blocks within a tile without using video data from other tiles. However, video encoder **200** and video decoder **300** may perform filtering across tile boundaries. Tiles may be uniform or non-uniform in size. Tile-based coding may enable parallel processing and/or multi-threading for encoder and decoder implementations.

[0044] In some examples, video encoder **200** and video decoder **300** may use a single QTBT or MTT structure to represent each of the luminance and chrominance components, while in other examples, video encoder **200** and video decoder **300** may use two or more QTBT or MTT structures, such as one QTBT/MTT structure for the luminance component and another QTBT/MTT structure for both chrominance components (or two QTBT/MTT structures for respective chrominance components).

[0045] Video encoder **200** and video decoder **300** may be configured to use quadtree partitioning, QTBT partitioning, MTT partitioning, superblock partitioning, or other partitioning structures.

[0046] In some examples, a CTU includes a coding tree block (CTB) of luma samples, two corresponding CTBs of chroma samples of a picture that has three sample arrays, or a CTB of samples of a monochrome picture or a picture that is coded using three separate color planes and syntax structures used to code the samples. A CTB may be an N×N block of samples for some value of N such that the division of a component into CTBs is a partitioning. A component is an array or single sample from one of the three arrays (luma and two chroma) that compose a picture in 4:2:0, 4:2:2, or 4:4:4 color format or the array or a single sample of the array that compose a picture in monochrome format. In some examples, a coding block is an M×N block of samples for some values of M and N such that a division of a CTB into coding blocks is a partitioning.

[0047] The blocks (e.g., CTUs or CUs) may be grouped in various ways in a picture. As one example, a brick may refer to a rectangular region of CTU rows within a particular tile in a picture. A tile may be a rectangular region of CTUs within a particular tile column and a particular tile row in a picture. A tile column refers to a rectangular region of CTUS having a height equal to the height of the picture and a width specified by syntax elements (e.g., such as in a picture parameter set). A tile row refers to a rectangular region of CTUs having a height specified by syntax elements (e.g., such as in a picture parameter set) and a width equal to the width of the picture.

[0048] In some examples, a tile may be partitioned into multiple bricks, each of which may include one or more CTU rows within the tile. A tile that is not partitioned into multiple bricks may also be referred to as a brick. However, a brick that is a true subset of a tile may not be referred to as a tile. The bricks in a picture may also be arranged in a slice. A slice may be an integer number of bricks of a picture that may be exclusively contained in a single network abstraction layer (NAL) unit. In some examples, a slice includes either a number of complete tiles or only a consecutive sequence of complete bricks of one tile.

[0049] This disclosure may use "N×N" and "N by N" interchangeably to refer to the sample dimensions of a block (such as a CU or other video block) in terms of vertical and horizontal dimensions, e.g., 16×16 samples or 16 by 16 samples. In general, a 16×16 CU will have 16 samples in a vertical direction (y=16) and 16 samples in a horizontal direction (x=16). Likewise, an N×N CU generally has N samples in a vertical direction and N samples in a horizontal direction, where N represents a nonnegative integer value. The samples in a CU may be arranged in rows and columns. Moreover, CUs need not necessarily have the same number of samples in the horizontal direction as in the vertical direction. For example, CUs may include N×M samples, where M is not necessarily equal to N.

[0050] Video encoder **200** encodes video data for CUs representing prediction and/or residual information, and other information. The prediction information indicates how the CU is to be predicted in order to form a prediction block for the CU. The residual information generally represents sample-by-sample differences between samples of the CU prior to encoding and the prediction block.

[0051] To predict a CU, video encoder **200** may generally form a prediction block for the CU through inter-prediction or intra-prediction. Inter-prediction generally refers to predicting the CU from data of a previously coded picture, whereas intra-prediction generally refers to predicting the CU from previously coded data of the same picture. To perform inter-prediction, video encoder **200** may generate the prediction block using one or more motion vectors. Video encoder **200** may generally perform a motion search to identify a reference block that closely matches the CU, e.g., in terms of differences between the CU and the reference block. Video encoder **200** may calculate a difference metric using a sum of absolute difference (SAD), sum of squared differences (SSD), mean absolute difference (MAD), mean squared differences (MSD), or other such difference calculations to determine whether a reference block closely matches the current CU. In some examples, video encoder **200** may predict the current CU using unidirectional prediction or bi-directional prediction.

[0052] Some examples of VVC also provide an affine motion compensation mode, which may be considered an inter-prediction mode. In affine motion compensation mode, video encoder **200** may determine two or more motion vectors that represent non-translational motion, such as zoom in or out, rotation, perspective motion, or other irregular motion types.

[0053] To perform intra-prediction, video encoder **200** may select an intra-prediction mode to generate the prediction block. Some examples of VVC provide sixty-seven intra-prediction modes, including various directional modes, as well as planar mode and DC mode. In general, video encoder **200** selects an intra-prediction mode that describes neighboring samples to a current block (e.g., a block of a CU) from which to predict samples of the current block. Such samples may generally be above, above and to the left, or to the left of the current block in the same picture as the current block, assuming video encoder **200** codes CTUs and CUs in raster scan order (left to right, top to bottom).

[0054] Video encoder **200** encodes data representing the prediction mode for a current block. For example, for inter-prediction modes, video encoder **200** may encode data

representing which of the various available inter-prediction modes is used, as well as motion information for the corresponding mode. For uni-directional or bi-directional inter-prediction, for example, video encoder **200** may encode motion vectors using advanced motion vector prediction (AMVP) or merge mode. Video encoder **200** may use similar modes to encode motion vectors for affine motion compensation mode.

[0055] AV1 includes two general techniques for encoding and decoding a coding block of video data. The two general techniques are intra prediction (e.g., intra frame prediction or spatial prediction) and inter prediction (e.g., inter frame prediction or temporal prediction). In the context of AV1, when predicting blocks of a current frame of video data using an intra prediction mode, video encoder **200** and video decoder **300** do not use video data from other frames of video data. For most intra prediction modes, video encoder **200** encodes blocks of a current frame based on the difference between sample values in the current block and predicted values generated from reference samples in the same frame. Video encoder **200** determines predicted values generated from the reference samples based on the intra prediction mode.

[0056] Following prediction, such as intra-prediction or inter-prediction of a block, video encoder **200** may calculate residual data for the block. The residual data, such as a residual block, represents sample by sample differences between the block and a prediction block for the block, formed using the corresponding prediction mode. Video encoder **200** may apply one or more transforms to the residual block, to produce transformed data in a transform domain instead of the sample domain. For example, video encoder **200** may apply a discrete cosine transform (DCT), an integer transform, a wavelet transform, or a conceptually similar transform to residual video data. Additionally, video encoder **200** may apply a secondary transform following the first transform, such as a mode-dependent non-separable secondary transform (MDNSST), a signal dependent transform, a Karhunen-Loeve transform (KLT), or the like. Video encoder **200** produces transform coefficients following application of the one or more transforms.

[0057] As noted above, following any transforms to produce transform coefficients, video encoder **200** may perform quantization of the transform coefficients. Quantization generally refers to a process in which transform coefficients are quantized to possibly reduce the amount of data used to represent the transform coefficients, providing further compression. By performing the quantization process, video encoder **200** may reduce the bit depth associated with some or all of the transform coefficients. For example, video encoder **200** may round an n-bit value down to an m-bit value during quantization, where n is greater than m. In some examples, to perform quantization, video encoder **200** may perform a bitwise right-shift of the value to be quantized.

[0058] Following quantization, video encoder **200** may scan the transform coefficients, producing a one-dimensional vector from the two-dimensional matrix including the quantized transform coefficients. The scan may be designed to place higher energy (and therefore lower frequency) transform coefficients at the front of the vector and to place lower energy (and therefore higher frequency) transform coefficients at the back of the vector. In some examples, video encoder **200** may utilize a predefined scan order to scan the quantized transform coefficients to produce a serialized vector, and then entropy encode the quantized transform coefficients of the vector. In other examples, video encoder **200** may perform an adaptive scan. After scanning the quantized transform coefficients to form the one-dimensional vector, video encoder **200** may entropy encode the one-dimensional vector, e.g., according to context-adaptive binary arithmetic coding (CABAC). Video encoder **200** may also entropy encode values for syntax elements describing metadata associated with the encoded video data for use by video decoder **300** in decoding the video data.

[0059] To perform CABAC, video encoder **200** may assign a context within a context model to a symbol to be transmitted. The context may relate to, for example, whether neighboring values of the symbol are zero-valued or not. The probability determination may be based on a context assigned to the symbol.

[0060] Video encoder **200** may further generate syntax data, such as block-based syntax data, picture-based syntax data, and sequence-based syntax data, to video decoder **300**, e.g., in a picture header, a block header, a slice header, or other syntax data, such as a sequence parameter set (SPS), picture parameter set (PPS), or video parameter set (VPS). Video decoder **300** may likewise decode such syntax data to determine how to decode corresponding video data.

[0061] In this manner, video encoder **200** may generate a bitstream including encoded video data, e.g., syntax elements describing partitioning of a picture into blocks (e.g., CUs) and prediction and/or residual information for the blocks. Ultimately, video decoder **300** may receive the bitstream and decode the encoded video data.

[0062] In general, video decoder **300** performs a reciprocal process to that performed by video encoder **200** to decode the encoded video data of the bitstream. For example, video decoder **300** may decode values for syntax elements of the bitstream using CABAC in a manner substantially similar to, albeit reciprocal to, the CABAC encoding process of video encoder **200**. The syntax elements may define partitioning information for partitioning of a picture into CTUs, and partitioning of each CTU according to a corresponding partition structure, such as a QTBT structure, to define CUs of the CTU. The syntax elements may further define prediction and residual information for blocks (e.g., CUs) of video data.

[0063] The residual information may be represented by, for example, quantized transform coefficients. Video decoder **300** may inverse quantize and inverse transform the quantized transform coefficients of a block to reproduce a residual block for the block. Video decoder **300** uses a signaled prediction mode (intra- or inter-prediction) and related prediction information (e.g., motion information for inter-prediction) to form a prediction block for the block. Video decoder **300** may then combine the prediction block and the residual block (on a sample-by-sample basis) to reproduce the original block. Video decoder **300** may perform additional processing, such as performing a deblocking process to reduce visual artifacts along boundaries of the block.

[0064] This disclosure may generally refer to "signaling" certain information, such as syntax elements. The term "signaling" may generally refer to the communication of values for syntax elements and/or other data used to decode encoded video data. That is, video encoder **200** may signal values for syntax elements in the bitstream. In general,

signaling refers to generating a value in the bitstream. As noted above, source device **102** may transport the bitstream to destination device **116** substantially in real time, or not in real time, such as might occur when storing syntax elements to storage device **112** for later retrieval by destination device **116**.

[0065] In accordance with the techniques of this disclosure, as will be explained in more detail below, video encoder **200** and video decoder **300** may be configured to use the probability initialization techniques of this disclosure. For example, video encoder **200** and video decoder **300** may be configured to receive an independently codable unit of video data, determine a first initial probability state for a first probability using a first probability initialization process, determine a second initial probability state for a second probability using a second probability initialization process, wherein the second initial probability state is different from the first initial probability state, and perform arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state.

Binary Arithmetic Coding

[0066] In many modern video coding standards, a video sequence is first converted into data elements (or syntax elements) with spatial-temporal redundancy removed. The syntax elements are then losslessly converted into binary representations (or bitstreams) USING entropy coding. Entropy coding may include binary arithmetic coding (e.g., CABAC) in recent video coding standards, such as HEVC and VVC.

[0067] Example binary arithmetic coding processes may include three main stages: binarization, adaptive probability estimation, and arithmetic coding. In binarization, a video coder converts each non-binary syntax element to be coded into a string of binary data symbols (or bins). In adaptive probability estimation, a video coder estimates a probability distribution (e.g., the probability of a bin being 0 or 1) for each bin, no matter whether the bin is a binary syntax element, or the bin is one element of a binary string converted from a non-binary syntax element. The video coder classifies the distributions into two categories: (1) a stationary and uniform distribution (e.g., always probability p=0.5) and (2) a time-varying or non-uniform distribution. The video coder assigns a bin with a Category (2) distribution (e.g., time-varying or non-uniform) a probability model (or context model) tracking the real-time distribution of that bin based on values of previous bins and other context statistics.

[0068] In arithmetic coding, the video coder codes the bin with a Category (1) distribution (e.g., stationary and uniform) in the bypass mode (e.g., with a fixed probability model), which is a low-complexity mode that is able to be processed using highly parallel processing techniques. The video coder codes the bin with Category (2) distribution in regular mode, where the bin value and its probability estimated by the associated context model are used.

[0069] When used in binary arithmetic coders for video coding, the probability described above (theoretically real valued and ranging from 0 to 1) is digitized, and therefore is usually referred to as a probability state. For example, in HEVC, the probability has 7-bit precision, corresponding to 128 probability states. The probability state in HEVC represents the real probability in the logarithmic domain. In VVC, the probability estimate of a certain bin is the average of two probabilities tracked in the associated context model of the bin and updated at a rapid and a slow rate, respectively (also called short and long windows). The probability updated at a rapid rate has 10-bit precision, corresponding to 1024 probability states. The other probability, updated at a slow rate, has 14-bit precision, corresponding to 16384 probability states. Unlike HEVC, VVC adopted a linear mapping between probability state and probability.

Context Model Initialization

[0070] In HEVC and VVC, a video bitstream includes multiple independently decodable units (e.g., slices), implying that at the beginning of such a unit, the probability states of all context models are reset to some predefined values. Typically, without any prior knowledge of the statistical nature of the source content, each context model should assume a uniform distribution (p=0.5). However, to bridge the learning phase of the adaptive probability estimation and to enable a preadaptation at different coding conditions, it has been found to be beneficial to determine a more appropriate initial probability state (e.g., initialization process) than to use an equiprobable state for each probability model.

[0071] The CABAC process in HEVC has a quantization-parameter (QP) dependent initialization process invoked at the beginning of each slice. Given the initial value of the luma QP for the slice (SliceQPy), the initial probability state of a certain context model, denoted as InitProbState, is generated by equations (1) to (3) below:

$$m = SlopeIdx * 5 - 45 \tag{1}$$

$$n = (OffsetIdx << 3) - 16 \tag{2}$$

$$InitProbState = Clip3(1, 126, ((m * SliceQP_Y) >> 4) + n), \tag{3}$$

where SlopeIdx and OffsetIdx (both integers ranging from 0 to 15 inclusive) are the initialization parameters predefined and stored for each context model. Eq. (3) means that InitProbState is modeled by a linear function of SliceQPY, with the slope approximately m>>4 (e.g., the value of m bitwise right shifted by 4), and the intersection n at SliceQPY=0. The mapping from SlopeIdx to slope and from OffsetIdx to intersection can be found in Tables 1 and 2, respectively in the HEVC standard.

[0072] In other words, a context model does not store initial probability state directly. Instead, a video coder may store, for a context model, two initialization parameters jointly determining a linear function that, at the beginning of each slice, uses SliceQPy as the argument to derive the probability state. The value of SlopeIdx and OffsetIdx, both having 4-bit precision, are packed into a single 8-bit initialization value, of which the high and the low nibble is SlopeIdx and OffsetIdx, respectively.

[0073] The CABAC process used in VVC uses basically the same techniques to derive InitProbState as in HEVC, except that in Eq. (3) the clipping ranges from 0 to 127. However, after deriving the value InitProbState, which represents the probability in the logarithmic domain, the initialization process in VVC uses one more step to convert InitProbState to a probability state that represents the probability in the linear domain, in order to be used in the

arithmetic coding engine of VVC. The conversion (or mapping) is implemented by using a look-up table (LUT), as below:

[0074] 1. Use InitProbState as the search index to find the corresponding probability_state value in the LUT.

[0075] 2.—The probability state in lower precision (10-bit), denoted as ProbabilityStateL, is derived in Eq. (4).

$$ProbabilityStateL = \text{probability\_state} >> 5 \quad (4)$$

[0076] The probability state in higher precision (14-bit), denoted as ProbabilityStateH, is derived in Eq. (5).

$$ProbabilityStateH = \text{probability\_state} >> 1 \quad (5)$$

[0077] The CABAC process of VVC and the Enhanced Compression Model version 11.0 (ECM 11.0) (shown in Eqs. (6)-(8) below) share a similar initialization model for state probabilities. The P0 and P1 probability states correspond to short and long windows and share the same initial probability state, as shown below.

$$\text{m\_state[0]} = InitProbState << 8;$$

$$\text{m\_state[1]} = InitProbState << 8,$$

where m_state[0] is used to derive the initial probability state for the P0 probability (short window), and m_state[1] is used to derive the initial probability state of the P1 probability (long window).

[0078] The initial state, InitProbState is trained with other CABAC context parameters combinedly.

[0079] The CABAC process in ECM 11.0 has a slightly different model, as given below:

$$m = SlopeIdx - 4 \quad (6)$$

$$n = (OffsetIdx * 18) + 1 \quad (7)$$

$$InitProbState = Clip3(1, 127, ((m * (SliceQP_Y - 16)) >> 1) + n) \quad (8)$$

EXAMPLES

[0080] In general, this disclosure describes techniques for a context initialization process that sets the separate initial probability states of each context model used in a binary arithmetic coder. Video encoder **200** and video decoder **300** may invoke the context initialization process at the beginning of entropy encoding or entropy decoding an independently decodable unit of video data, such as a slice of video data. The techniques of this disclosure are suitable for use with binary arithmetic coders where the probability state represents the real probability in the linear domain, such as the CABAC adopted in Versatile Video Coding (VVC) or other video coding standards and video coding codecs.

[0081] More specifically, this disclosure describes techniques where, instead of using the same starting point (e.g., probability initialization) for each of the probability states (e.g., probability states P0 and P1 corresponding to long and short windows), video encoder **200** and video decoder **300** may be configured to use a different initialization or starting point for each probability state. In one example, such initialization may be state specific. For example, video encoder **200** and video decoder **300** may be configured to use one initialization for the first probability state (e.g., P0) and use another initialization for the second probability state (e.g., P1).

[0082] In another example, video encoder **200** and video decoder **300** may be configured to use up to two probability initializations. Video encoder **200** may be configured to signal a syntax element (e.g., a flag) which indicates which probability initialization is used for a specific probability state.

[0083] Similarly, video encoder **200** and video decoder **300** may be configured to store a CABAC parameter(s), including probabilities, from previously coded pictures. Video encoder **200** and video decoder **300** may be configured to use the stored parameters for the initialization of a probability state for the current picture (e.g., temporal CABAC tool in ECM). In such case, the probability of each probability state may be stored from previously coded pictures and may be reused to initialize each probability state of the current picture.

[0084] In one implementation example, the initial state for each probability state (e.g., P0 and P1) may be trained and used separately.

$$\text{m\_state[0]} = \text{InitProbState\_P0} << 8 \text{ (short window);}$$

$$\text{m\_state[1]} = \text{InitProbState\_P1} << 8 \text{ (long window);}$$

[0085] The separate initial states, InitProbState_P0 and InitProbState_P1, may be trained with other CABAC context parameters combinedly.

[0086] In view of the above, in one example of the disclosure, video encoder **200** and video decoder **300** may be configured to receive an independently codable unit of video data (e.g., a slice), determine a first initial probability state for a first probability using a first probability initialization process, and determine a second initial probability state for a second probability using a second probability initialization process. In this example, the second initial probability state is different from the first initial probability state. Video encoder **200** and video decoder **300** may be further configured to perform arithmetic coding (e.g., CABAC) of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state.

[0087] In one example, to perform arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state, video encoder **200** and video decoder **300** are configured to determine the first probability using the first initial probability state, wherein the first probability is a short window probability (P0), determine the second probability using the second initial probability state, wherein the second probability is a long window probability (P1), and perform arithmetic coding of syntax elements in

the independently codable unit of video data using the short window probability (P0) and the long window probability (P1).

[0088] In one example, to determine the first initial probability state for the first probability using the first probability initialization process, video encoder **200** and video decoder **300** may determine the first initial probability state for the first probability using the first probability initialization process and parameters stored from a previous picture of video data. Likewise, to determine the second initial probability state for the second probability using the second probability initialization process video encoder **200** and video decoder **300** may determine the second initial probability state for the second probability using the second probability initialization process and parameters stored from a previous picture of video data.

[0089] In another example of the disclosure, video encoder **200** and video decoder **300** may be configured to receive an independently codable unit of video data (e.g., a slice), receive a first syntax element indicating which of a plurality of probability initialization processes is to be used for a first initial probability state for a first probability, and receive a second syntax element indicating which of a plurality of probability initialization processes is to be used for a second initial probability state for a second probability. Video encoder **200** and video decoder **300** may further determine the first initial probability state for the first probability according to a first probability initialization process indicated by the first syntax element, determine the second initial probability state for the second probability according to second probability initialization process indicated by the second syntax element, and perform arithmetic coding (e.g., CABAC) of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state.

[0090] In one example, the first probability initialization process and the second probability initialization process are the same. In another example, the first probability initialization process and the second probability initialization process are different.

[0091] FIG. **2** is a block diagram illustrating an example video encoder **200** that may perform the techniques of this disclosure. FIG. **2** is provided for purposes of explanation and should not be considered limiting of the techniques as broadly exemplified and described in this disclosure. For purposes of explanation, this disclosure describes video encoder **200** according to the techniques of VVC and HEVC. However, the techniques of this disclosure may be performed by video encoding devices that are configured to other video coding standards and video coding formats, such as AV1 and successors to the AV1 video coding format.

[0092] In the example of FIG. **2**, video encoder **200** includes video data memory **230**, mode selection unit **202**, residual generation unit **204**, transform processing unit **206**, quantization unit **208**, inverse quantization unit **210**, inverse transform processing unit **212**, reconstruction unit **214**, filter unit **216**, decoded picture buffer (DPB) **218**, and entropy encoding unit **220**. Any or all of video data memory **230**, mode selection unit **202**, residual generation unit **204**, transform processing unit **206**, quantization unit **208**, inverse quantization unit **210**, inverse transform processing unit **212**, reconstruction unit **214**, filter unit **216**, DPB **218**, and entropy encoding unit **220** may be implemented in one or more processors or in processing circuitry. For instance, the units of video encoder **200** may be implemented as one or more circuits or logic elements as part of hardware circuitry, or as part of a processor, ASIC, or FPGA. Moreover, video encoder **200** may include additional or alternative processors or processing circuitry to perform these and other functions.

[0093] Video data memory **230** is an example of a memory system that may store video data to be encoded by the components of video encoder **200**. Video encoder **200** may receive the video data stored in video data memory **230** from, for example, video source **104** (FIG. **1**). DPB **218** is an example of a memory system that may act as a reference picture memory that stores reference video data for use in prediction of subsequent video data by video encoder **200**. Video data memory **230** and DPB **218** may each be formed by any of a variety of one or more memory devices or memory units, such as dynamic random access memory (DRAM), including synchronous DRAM (SDRAM), magnetoresistive RAM (MRAM), resistive RAM (RRAM), or other types of memory devices. Video data memory **230** and DPB **218** may be provided by the same memory device or separate memory devices. In various examples, video data memory **230** may be on-chip with other components of video encoder **200**, as illustrated, or off-chip relative to those components.

[0094] In this disclosure, reference to video data memory **230** should not be interpreted as being limited to memory internal to video encoder **200**, unless specifically described as such, or memory external to video encoder **200**, unless specifically described as such. Rather, reference to video data memory **230** should be understood as reference memory that stores video data that video encoder **200** receives for encoding (e.g., video data for a current block that is to be encoded). Memory **106** of FIG. **1** may also provide temporary storage of outputs from the various units of video encoder **200**.

[0095] The various units of FIG. **2** are illustrated to assist with understanding the operations performed by video encoder **200**. The units may be implemented as fixed-function circuits, programmable circuits, or a combination thereof. Fixed-function circuits refer to circuits that provide particular functionality, and are preset on the operations that can be performed. Programmable circuits refer to circuits that can be programmed to perform various tasks, and provide flexible functionality in the operations that can be performed. For instance, programmable circuits may execute software or firmware that cause the programmable circuits to operate in the manner defined by instructions of the software or firmware. Fixed-function circuits may execute software instructions (e.g., to receive parameters or output parameters), but the types of operations that the fixed-function circuits perform are generally immutable. In some examples, one or more of the units may be distinct circuit blocks (fixed-function or programmable), and in some examples, one or more of the units may be integrated circuits.

[0096] Video encoder **200** may include arithmetic logic units (ALUs), elementary function units (EFUs), digital circuits, analog circuits, and/or programmable cores, formed from programmable circuits. In examples where the operations of video encoder **200** are performed using software executed by the programmable circuits, memory **106** (FIG. **1**) may store the instructions (e.g., object code) of the

software that video encoder **200** receives and executes, or another memory within video encoder **200** (not shown) may store such instructions.

[0097] Video data memory **230** is configured to store received video data. Video encoder **200** may retrieve a picture of the video data from video data memory **230** and provide the video data to residual generation unit **204** and mode selection unit **202**. Video data in video data memory **230** may be raw video data that is to be encoded.

[0098] Mode selection unit **202** includes a motion estimation unit **222**, a motion compensation unit **224**, and an intra-prediction unit **226**. Mode selection unit **202** may include additional functional units to perform video prediction in accordance with other prediction modes. As examples, mode selection unit **202** may include a palette unit, an intra-block copy unit (which may be part of motion estimation unit **222** and/or motion compensation unit **224**), an affine unit, a linear model (LM) unit, or the like.

[0099] Mode selection unit **202** generally coordinates multiple encoding passes to test combinations of encoding parameters and resulting rate-distortion values for such combinations. The encoding parameters may include partitioning of CTUs into CUs, prediction modes for the CUS, transform types for residual data of the CUs, quantization parameters for residual data of the CUs, and so on. Mode selection unit **202** may ultimately select the combination of encoding parameters having rate-distortion values that are better than the other tested combinations.

[0100] Video encoder **200** may partition a picture retrieved from video data memory **230** into a series of CTUs, and encapsulate one or more CTUs within a slice. Mode selection unit **202** may partition a CTU of the picture in accordance with a tree structure, such as the MTT structure, QTBT structure. superblock structure, or the quad-tree structure described above. As described above, video encoder **200** may form one or more CUs from partitioning a CTU according to the tree structure. Such a CU may also be referred to generally as a "video block" or "block."

[0101] In general, mode selection unit **202** also controls the components thereof (e.g., motion estimation unit **222**, motion compensation unit **224**, and intra-prediction unit **226**) to generate a prediction block for a current block (e.g., a current CU, or in HEVC, the overlapping portion of a PU and a TU). For inter-prediction of a current block, motion estimation unit **222** may perform a motion search to identify one or more closely matching reference blocks in one or more reference pictures (e.g., one or more previously coded pictures stored in DPB **218**). In particular, motion estimation unit **222** may calculate a value representative of how similar a potential reference block is to the current block, e.g., according to sum of absolute difference (SAD), sum of squared differences (SSD), mean absolute difference (MAD), mean squared differences (MSD), or the like. Motion estimation unit **222** may generally perform these calculations using sample-by-sample differences between the current block and the reference block being considered. Motion estimation unit **222** may identify a reference block having a lowest value resulting from these calculations, indicating a reference block that most closely matches the current block.

[0102] Motion estimation unit **222** may form one or more motion vectors (MVs) that defines the positions of the reference blocks in the reference pictures relative to the position of the current block in a current picture. Motion estimation unit **222** may then provide the motion vectors to motion compensation unit **224**. For example, for uni-directional inter-prediction, motion estimation unit **222** may provide a single motion vector, whereas for bi-directional inter-prediction, motion estimation unit **222** may provide two motion vectors. Motion compensation unit **224** may then generate a prediction block using the motion vectors. For example, motion compensation unit **224** may retrieve data of the reference block using the motion vector. As another example, if the motion vector has fractional sample precision, motion compensation unit **224** may interpolate values for the prediction block according to one or more interpolation filters. Moreover, for bi-directional inter-prediction, motion compensation unit **224** may retrieve data for two reference blocks identified by respective motion vectors and combine the retrieved data, e.g., through sample-by-sample averaging or weighted averaging.

[0103] When operating according to the AV1 video coding format, motion estimation unit **222** and motion compensation unit **224** may be configured to encode coding blocks of video data (e.g., both luma and chroma coding blocks) using translational motion compensation, affine motion compensation, overlapped block motion compensation (OBMC), and/or compound inter-intra prediction.

[0104] As another example, for intra-prediction, or intra-prediction coding, intra-prediction unit **226** may generate the prediction block from samples neighboring the current block. For example, for directional modes, intra-prediction unit **226** may generally mathematically combine values of neighboring samples and populate these calculated values in the defined direction across the current block to produce the prediction block. As another example, for DC mode, intra-prediction unit **226** may calculate an average of the neighboring samples to the current block and generate the prediction block to include this resulting average for each sample of the prediction block.

[0105] When operating according to the AV1 video coding format, intra-prediction unit **226** may be configured to encode coding blocks of video data (e.g., both luma and chroma coding blocks) using directional intra prediction, non-directional intra prediction, recursive filter intra prediction, chroma-from-luma (CFL) prediction, intra block copy (IBC), and/or color palette mode. Mode selection unit **202** may include additional functional units to perform video prediction in accordance with other prediction modes.

[0106] Mode selection unit **202** provides the prediction block to residual generation unit **204**. Residual generation unit **204** receives a raw, unencoded version of the current block from video data memory **230** and the prediction block from mode selection unit **202**. Residual generation unit **204** calculates sample-by-sample differences between the current block and the prediction block. The resulting sample-by-sample differences define a residual block for the current block. In some examples, residual generation unit **204** may also determine differences between sample values in the residual block to generate a residual block using residual differential pulse code modulation (RDPCM). In some examples, residual generation unit **204** may be formed using one or more subtractor circuits that perform binary subtraction.

[0107] In examples where mode selection unit **202** partitions CUs into PUs, each PU may be associated with a luma prediction unit and corresponding chroma prediction units. Video encoder **200** and video decoder **300** may support PUs

having various sizes. As indicated above, the size of a CU may refer to the size of the luma coding block of the CU and the size of a PU may refer to the size of a luma prediction unit of the PU. Assuming that the size of a particular CU is 2N×2N, video encoder **200** may support PU sizes of 2N×2N or N×N for intra prediction, and symmetric PU sizes of 2N×2N, 2N×N, N×2N, N×N, or similar for inter prediction. Video encoder **200** and video decoder **300** may also support asymmetric partitioning for PU sizes of 2NxnU, 2NxnD, nLx2N, and nRx2N for inter prediction.

[0108] In examples where mode selection unit **202** does not further partition a CU into PUs, each CU may be associated with a luma coding block and corresponding chroma coding blocks. As above, the size of a CU may refer to the size of the luma coding block of the CU. The video encoder **200** and video decoder **300** may support CU sizes of 2N×2N, 2N×N, or N×2N.

[0109] For other video coding techniques such as an intra-block copy mode coding, an affine-mode coding, and linear model (LM) mode coding, as some examples, mode selection unit **202**, via respective units associated with the coding techniques, generates a prediction block for the current block being encoded. In some examples, such as palette mode coding, mode selection unit **202** may not generate a prediction block, and instead generate syntax elements that indicate the manner in which to reconstruct the block based on a selected palette. In such modes, mode selection unit **202** may provide these syntax elements to entropy encoding unit **220** to be encoded.

[0110] As described above, residual generation unit **204** receives the video data for the current block and the corresponding prediction block. Residual generation unit **204** then generates a residual block for the current block. To generate the residual block, residual generation unit **204** calculates sample-by-sample differences between the prediction block and the current block.

[0111] Transform processing unit **206** applies one or more transforms to the residual block to generate a block of transform coefficients (referred to herein as a "transform coefficient block"). Transform processing unit **206** may apply various transforms to a residual block to form the transform coefficient block. For example, transform processing unit **206** may apply a discrete cosine transform (DCT), a directional transform, a Karhunen-Loeve transform (KLT), or a conceptually similar transform to a residual block. In some examples, transform processing unit **206** may perform multiple transforms to a residual block, e.g., a primary transform and a secondary transform, such as a rotational transform. In some examples, transform processing unit **206** does not apply transforms to a residual block.

[0112] When operating according to AV1, transform processing unit **206** may apply one or more transforms to the residual block to generate a block of transform coefficients (referred to herein as a "transform coefficient block"). Transform processing unit **206** may apply various transforms to a residual block to form the transform coefficient block. For example, transform processing unit **206** may apply a horizontal/vertical transform combination that may include a discrete cosine transform (DCT), an asymmetric discrete sine transform (ADST), a flipped ADST (e.g., an ADST in reverse order), and an identity transform (IDTX). When using an identity transform, the transform is skipped in one of the vertical or horizontal directions. In some examples, transform processing may be skipped.

[0113] Quantization unit **208** may quantize the transform coefficients in a transform coefficient block, to produce a quantized transform coefficient block. Quantization unit **208** may quantize transform coefficients of a transform coefficient block according to a quantization parameter (QP) value associated with the current block. Video encoder **200** (e.g., via mode selection unit **202**) may adjust the degree of quantization applied to the transform coefficient blocks associated with the current block by adjusting the QP value associated with the CU. Quantization may introduce loss of information, and thus, quantized transform coefficients may have lower precision than the original transform coefficients produced by transform processing unit **206**.

[0114] Inverse quantization unit **210** and inverse transform processing unit **212** may apply inverse quantization and inverse transforms to a quantized transform coefficient block, respectively, to reconstruct a residual block from the transform coefficient block. Reconstruction unit **214** may produce a reconstructed block corresponding to the current block (albeit potentially with some degree of distortion) based on the reconstructed residual block and a prediction block generated by mode selection unit **202**. For example, reconstruction unit **214** may add samples of the reconstructed residual block to corresponding samples from the prediction block generated by mode selection unit **202** to produce the reconstructed block.

[0115] Filter unit **216** may perform one or more filter operations on reconstructed blocks. For example, filter unit **216** may perform deblocking operations to reduce blockiness artifacts along edges of CUs. Operations of filter unit **216** may be skipped, in some examples.

[0116] When operating according to AV1, filter unit **216** may perform one or more filter operations on reconstructed blocks. For example, filter unit **216** may perform deblocking operations to reduce blockiness artifacts along edges of CUs. In other examples, filter unit **216** may apply a constrained directional enhancement filter (CDEF), which may be applied after deblocking, and may include the application of non-separable, non-linear, low-pass directional filters based on estimated edge directions. Filter unit **216** may also include a loop restoration filter, which is applied after CDEF, and may include a separable symmetric normalized Wiener filter or a dual self-guided filter.

[0117] Video encoder **200** stores reconstructed blocks in DPB **218**. For instance, in examples where operations of filter unit **216** are not performed, reconstruction unit **214** may store reconstructed blocks to DPB **218**. In examples where operations of filter unit **216** are performed, filter unit **216** may store the filtered reconstructed blocks to DPB **218**. Motion estimation unit **222** and motion compensation unit **224** may retrieve a reference picture from DPB **218**, formed from the reconstructed (and potentially filtered) blocks, to inter-predict blocks of subsequently encoded pictures. In addition, intra-prediction unit **226** may use reconstructed blocks in DPB **218** of a current picture to intra-predict other blocks in the current picture.

[0118] In general, entropy encoding unit **220** may entropy encode syntax elements received from other functional components of video encoder **200**. For example, entropy encoding unit **220** may entropy encode quantized transform coefficient blocks from quantization unit **208**. As another example, entropy encoding unit **220** may entropy encode prediction syntax elements (e.g., motion information for inter-prediction or intra-mode information for intra-predic-

tion) from mode selection unit **202**. Entropy encoding unit **220** may perform one or more entropy encoding operations on the syntax elements, which are another example of video data, to generate entropy-encoded data. For example, entropy encoding unit **220** may perform a context-adaptive variable length coding (CAVLC) operation, a CABAC operation, a variable-to-variable (V2V) length coding operation, a syntax-based context-adaptive binary arithmetic coding (SBAC) operation, a Probability Interval Partitioning Entropy (PIPE) coding operation, an Exponential-Golomb encoding operation, or another type of entropy encoding operation on the data. In some examples, entropy encoding unit **220** may operate in bypass mode where syntax elements are not entropy encoded.

[0119] Video encoder **200** may output a bitstream that includes the entropy encoded syntax elements needed to reconstruct blocks of a slice or picture. In particular, entropy encoding unit **220** may output the bitstream.

[0120] In accordance with AV1, entropy encoding unit **220** may be configured as a symbol-to-symbol adaptive multi-symbol arithmetic coder. A syntax element in AV1 includes an alphabet of N elements, and a context (e.g., probability model) includes a set of N probabilities. Entropy encoding unit **220** may store the probabilities as n-bit (e.g., 15-bit) cumulative distribution functions (CDFs). Entropy encoding unit **220** may perform recursive scaling, with an update factor based on the alphabet size, to update the contexts.

[0121] In accordance with the techniques of this disclosure, as was described above, entropy encoding unit **220** may be configured to determine a first initial probability state for a first probability using a first probability initialization process, determine a second initial probability state for a second probability using a second probability initialization process, wherein the second initial probability state is different from the first initial probability state, and perform arithmetic coding of syntax elements in an independently codable unit of video data using the first initial probability state and the second initial probability state. In one example, the independently codable unit of video data is a slice of video data. In one example, the arithmetic coding is context adaptive binary arithmetic coding (CABAC).

[0122] In one example, to perform arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state, entropy encoding unit **220** may determine the first probability using the first initial probability state, wherein the first probability is a short window probability (P0), determine the second probability using the second initial probability state, wherein the second probability is a long window probability (P1), and perform arithmetic coding of syntax elements in the independently codable unit of video data using the short window probability (P0) and the long window probability (P1).

[0123] In another example, to determine the first initial probability state for the first probability using the first probability initialization process, entropy encoding unit **220** may determine the first initial probability state for the first probability using the first probability initialization process and parameters stored from a previous picture of video data. Likewise, to determine the second initial probability state for the second probability using the second probability initialization process, entropy encoding unit **220** may determine the second initial probability state for the second probability using the second probability initialization process and the parameters stored from the previous picture of video data.

[0124] The operations described above are described with respect to a block. Such description should be understood as being operations for a luma coding block and/or chroma coding blocks. As described above, in some examples, the luma coding block and chroma coding blocks are luma and chroma components of a CU. In some examples, the luma coding block and the chroma coding blocks are luma and chroma components of a PU.

[0125] In some examples, operations performed with respect to a luma coding block need not be repeated for the chroma coding blocks. As one example, operations to identify a motion vector (MV) and reference picture for a luma coding block need not be repeated for identifying a MV and reference picture for the chroma blocks. Rather, the MV for the luma coding block may be scaled to determine the MV for the chroma blocks, and the reference picture may be the same. As another example, the intra-prediction process may be the same for the luma coding block and the chroma coding blocks.

[0126] Video encoder **200** represents an example of a device configured to encode video data including a memory configured to store video data, and one or more processing units implemented in circuitry and configured to use one or more of the probability estimation techniques of this disclosure.

[0127] FIG. **3** is a block diagram illustrating an example video decoder **300** that may perform the techniques of this disclosure. FIG. **3** is provided for purposes of explanation and is not limiting on the techniques as broadly exemplified and described in this disclosure. For purposes of explanation, this disclosure describes video decoder **300** according to the techniques of VVC and HEVC. However, the techniques of this disclosure may be performed by video coding devices that are configured to other video coding standards.

[0128] In the example of FIG. **3**, video decoder **300** includes coded picture buffer (CPB) memory **320**, entropy decoding unit **302**, prediction processing unit **304**, inverse quantization unit **306**, inverse transform processing unit **308**, reconstruction unit **310**, filter unit **312**, and DPB **314**. Any or all of CPB memory **320**, entropy decoding unit **302**, prediction processing unit **304**, inverse quantization unit **306**, inverse transform processing unit **308**, reconstruction unit **310**, filter unit **312**, and DPB **314** may be implemented in one or more processors or in processing circuitry. For instance, the units of video decoder **300** may be implemented as one or more circuits or logic elements as part of hardware circuitry, or as part of a processor, ASIC, or FPGA. Moreover, video decoder **300** may include additional or alternative processors or processing circuitry to perform these and other functions.

[0129] Prediction processing unit **304** includes motion compensation unit **316** and intra-prediction unit **318**. Prediction processing unit **304** may include additional units to perform prediction in accordance with other prediction modes. As examples, prediction processing unit **304** may include a palette unit, an intra-block copy unit (which may form part of motion compensation unit **316**), an affine unit, a linear model (LM) unit, or the like. In other examples, video decoder **300** may include more, fewer, or different functional components.

[0130] When operating according to AV1, motion compensation unit **316** may be configured to decode coding blocks of video data (e.g., both luma and chroma coding blocks) using translational motion compensation, affine motion compensation, OBMC, and/or compound inter-intra prediction, as described above. Intra-prediction unit **318** may be configured to decode coding blocks of video data (e.g., both luma and chroma coding blocks) using directional intra prediction, non-directional intra prediction, recursive filter intra prediction, CFL, IBC, and/or color palette mode, as described above.

[0131] CPB memory **320** is an example of a memory system that may store video data, such as an encoded video bitstream, to be decoded by the components of video decoder **300**. The video data stored in CPB memory **320** may be obtained, for example, from computer-readable medium **110** (FIG. **1**). CPB memory **320** may include a CPB that stores encoded video data (e.g., syntax elements) from an encoded video bitstream. Also, CPB memory **320** may store video data other than syntax elements of a coded picture, such as temporary data representing outputs from the various units of video decoder **300**. DPB **314** is an example of a memory system that generally stores decoded pictures, which video decoder **300** may output and/or use as reference video data when decoding subsequent data or pictures of the encoded video bitstream. CPB memory **320** and DPB **314** may each be formed by any of a variety of memory devices or memory units, such as DRAM, including SDRAM, MRAM, RRAM, or other types of memory devices. CPB memory **320** and DPB **314** may be provided by the same memory device or separate memory devices. In various examples, CPB memory **320** may be on-chip with other components of video decoder **300**, or off-chip relative to those components.

[0132] Additionally or alternatively, in some examples, video decoder **300** may retrieve coded video data from memory **120** (FIG. **1**). That is, memory **120** may store data as discussed above with CPB memory **320**. Likewise, memory **120** may store instructions to be executed by video decoder **300**, when some or all of the functionality of video decoder **300** is implemented in software to be executed by processing circuitry of video decoder **300**.

[0133] The various units shown in FIG. **3** are illustrated to assist with understanding the operations performed by video decoder **300**. The units may be implemented as fixed-function circuits, programmable circuits, or a combination thereof. Similar to FIG. **2**, fixed-function circuits refer to circuits that provide particular functionality, and are preset on the operations that can be performed. Programmable circuits refer to circuits that can be programmed to perform various tasks, and provide flexible functionality in the operations that can be performed. For instance, programmable circuits may execute software or firmware that cause the programmable circuits to operate in the manner defined by instructions of the software or firmware. Fixed-function circuits may execute software instructions (e.g., to receive parameters or output parameters), but the types of operations that the fixed-function circuits perform are generally immutable. In some examples, one or more of the units may be distinct circuit blocks (fixed-function or programmable), and in some examples, one or more of the units may be integrated circuits.

[0134] Video decoder **300** may include ALUs, EFUs, digital circuits, analog circuits, and/or programmable cores formed from programmable circuits. In examples where the operations of video decoder **300** are performed by software executing on the programmable circuits, on-chip or off-chip memory may store instructions (e.g., object code) of the software that video decoder **300** receives and executes.

[0135] Entropy decoding unit **302** may receive encoded video data from the CPB and entropy decode the video data to reproduce syntax elements. Prediction processing unit **304**, inverse quantization unit **306**, inverse transform processing unit **308**, reconstruction unit **310**, and filter unit **312** may generate decoded video data based on the syntax elements extracted from the bitstream.

[0136] In general, video decoder **300** reconstructs a picture on a block-by-block basis. Video decoder **300** may perform a reconstruction operation on each block individually (where the block currently being reconstructed, i.e., decoded, may be referred to as a "current block").

[0137] Entropy decoding unit **302** may entropy decode syntax elements defining quantized transform coefficients of a quantized transform coefficient block, as well as transform information, such as a quantization parameter (QP) and/or transform mode indication(s). Inverse quantization unit **306** may use the QP associated with the quantized transform coefficient block to determine a degree of quantization and, likewise, a degree of inverse quantization for inverse quantization unit **306** to apply. Inverse quantization unit **306** may, for example, perform a bitwise left-shift operation to inverse quantize the quantized transform coefficients. Inverse quantization unit **306** may thereby form a transform coefficient block including transform coefficients.

[0138] In accordance with the techniques of this disclosure, as was described above, entropy decoding unit **302** may be configured to determine a first initial probability state for a first probability using a first probability initialization process, determine a second initial probability state for a second probability using a second probability initialization process, wherein the second initial probability state is different from the first initial probability state, and perform arithmetic coding of syntax elements in an independently codable unit of video data using the first initial probability state and the second initial probability state. In one example, the independently codable unit of video data is a slice of video data. In one example, the arithmetic coding is context adaptive binary arithmetic coding (CABAC).

[0139] In one example, to perform arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state, entropy decoding unit **302** may determine the first probability using the first initial probability state, wherein the first probability is a short window probability (P0), determine the second probability using the second initial probability state, wherein the second probability is a long window probability (P1), and perform arithmetic coding of syntax elements in the independently codable unit of video data using the short window probability (P0) and the long window probability (P1).

[0140] In another example, to determine the first initial probability state for the first probability using the first probability initialization process, entropy decoding unit **302** may determine the first initial probability state for the first probability using the first probability initialization process and parameters stored from a previous picture of video data. Likewise, to determine the second initial probability state

for the second probability using the second probability initialization process, entropy encoding unit **220** may determine the second initial probability state for the second probability using the second probability initialization process and the parameters stored from the previous picture of video data.

[0141] After inverse quantization unit **306** forms the transform coefficient block, inverse transform processing unit **308** may apply one or more inverse transforms to the transform coefficient block to generate a residual block associated with the current block. For example, inverse transform processing unit **308** may apply an inverse DCT, an inverse integer transform, an inverse Karhunen-Loeve transform (KLT), an inverse rotational transform, an inverse directional transform, or another inverse transform to the transform coefficient block.

[0142] Furthermore, prediction processing unit **304** generates a prediction block according to prediction information syntax elements that were entropy decoded by entropy decoding unit **302**. For example, if the prediction information syntax elements indicate that the current block is inter-predicted, motion compensation unit **316** may generate the prediction block. In this case, the prediction information syntax elements may indicate a reference picture in DPB **314** from which to retrieve a reference block, as well as a motion vector identifying a location of the reference block in the reference picture relative to the location of the current block in the current picture. Motion compensation unit **316** may generally perform the inter-prediction process in a manner that is substantially similar to that described with respect to motion compensation unit **224** (FIG. **2**).

[0143] As another example, if the prediction information syntax elements indicate that the current block is intra-predicted, intra-prediction unit **318** may generate the prediction block according to an intra-prediction mode indicated by the prediction information syntax elements. Again, intra-prediction unit **318** may generally perform the intra-prediction process in a manner that is substantially similar to that described with respect to intra-prediction unit **226** (FIG. **2**). Intra-prediction unit **318** may retrieve data of neighboring samples to the current block from DPB **314**.

[0144] Reconstruction unit **310** may reconstruct the current block using the prediction block and the residual block. For example, reconstruction unit **310** may add samples of the residual block to corresponding samples of the prediction block to reconstruct the current block.

[0145] Filter unit **312** may perform one or more filter operations on reconstructed blocks. For example, filter unit **312** may perform deblocking operations to reduce blockiness artifacts along edges of the reconstructed blocks. Operations of filter unit **312** are not necessarily performed in all examples.

[0146] Video decoder **300** may store the reconstructed blocks in DPB **314**. For instance, in examples where operations of filter unit **312** are not performed, reconstruction unit **310** may store reconstructed blocks to DPB **314**. In examples where operations of filter unit **312** are performed, filter unit **312** may store the filtered reconstructed blocks to DPB **314**. As discussed above, DPB **314** may provide reference information, such as samples of a current picture for intra-prediction and previously decoded pictures for subsequent motion compensation, to prediction processing unit **304**. Moreover, video decoder **300** may output decoded pictures (e.g., decoded video) from DPB **314** for subsequent presentation on a display device, such as display device **118** of FIG. **1**.

[0147] In this manner, video decoder **300** represents an example of a video decoding device including a memory configured to store video data, and one or more processing units implemented in circuitry and configured to use one or more of the probability estimation techniques of this disclosure.

[0148] FIG. **4** is a flowchart illustrating an example method for encoding a current block in accordance with the techniques of this disclosure. The current block may be or include a current CU. Although described with respect to video encoder **200** (FIGS. **1** and **2**), it should be understood that other devices may be configured to perform a method similar to that of FIG. **4**.

[0149] In this example, video encoder **200** initially predicts the current block (**400**). For example, video encoder **200** may form a prediction block for the current block. Video encoder **200** may then calculate a residual block for the current block (**402**). To calculate the residual block, video encoder **200** may calculate a difference between the original, unencoded block and the prediction block for the current block. Video encoder **200** may then transform the residual block and quantize transform coefficients of the residual block (**404**). Next, video encoder **200** may scan the quantized transform coefficients of the residual block (**406**). During the scan, or following the scan, video encoder **200** may entropy encode the transform coefficients (**408**). For example, video encoder **200** may encode the transform coefficients using CAVLC or CABAC. Video encoder **200** may then output the entropy encoded data of the block (**410**).

[0150] FIG. **5** is a flowchart illustrating an example method for decoding a current block of video data in accordance with the techniques of this disclosure. The current block may be or include a current CU. Although described with respect to video decoder **300** (FIGS. **1** and **3**), it should be understood that other devices may be configured to perform a method similar to that of FIG. **5**.

[0151] Video decoder **300** may receive entropy encoded data for the current block, such as entropy encoded prediction information and entropy encoded data for transform coefficients of a residual block corresponding to the current block (**500**). Video decoder **300** may entropy decode the entropy encoded data to determine prediction information for the current block and to reproduce transform coefficients of the residual block (**502**). Video decoder **300** may predict the current block (**504**), e.g., using an intra- or inter-prediction mode as indicated by the prediction information for the current block, to calculate a prediction block for the current block. Video decoder **300** may then inverse scan the reproduced transform coefficients (**506**), to create a block of quantized transform coefficients. Video decoder **300** may then inverse quantize the transform coefficients and apply an inverse transform to the transform coefficients to produce a residual block (**508**). Video decoder **300** may ultimately decode the current block by combining the prediction block and the residual block (**510**).

[0152] FIG. **6** is a flowchart illustrating an example method for coding video data in accordance with the techniques of this disclosure. The techniques of FIG. **6** may be performed by one or more components of video encoder **200** and video decoder **300**, including entropy encoding unit **220** (FIG. **2**) and entropy decoding unit **302** (FIG. **3**).

[0153] In one example, video encoder **200** and video decoder **300** may be configured to receive an independently codable unit of video data (**600**), determine a first initial probability state for a first probability using a first probability initialization process (**602**), and determine a second initial probability state for a second probability using a second probability initialization process, wherein the second initial probability state is different from the first initial probability state (**604**). Video encoder **200** and video decoder **300** may then perform arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state (**606**).

[0154] In one example, to perform arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state, video encoder **200** and video decoder **300** may determine the first probability using the first initial probability state, wherein the first probability is a short window probability (P0), determine the second probability using the second initial probability state, wherein the second probability is a long window probability (P1), and perform arithmetic coding of syntax elements in the independently codable unit of video data using the short window probability (P0) and the long window probability (P1).

[0155] In one example, the independently codable unit of video data is a slice of video data.

[0156] In one example, the arithmetic coding is context adaptive binary arithmetic coding (CABAC).

[0157] In another example, to determine the first initial probability state for the first probability using the first probability initialization process, video encoder **200** and video decoder **300** may determine the first initial probability state for the first probability using the first probability initialization process and parameters stored from a previous picture of video data. Likewise, to determine the second initial probability state for the second probability using the second probability initialization process, video encoder **200** and video decoder **300** may determine the second initial probability state for the second probability using the second probability initialization process and the parameters stored from the previous picture of video data.

[0158] The following numbered clauses illustrate one or more aspects of the devices and techniques described in this disclosure.

[0159] Aspect 1A. A method of coding video data, the method comprising: receiving an independently codable unit of video data; determining a first initial probability state for a first probability using a first probability initialization process; determining a second initial probability state for a second probability using a second probability initialization process, wherein the second probability initialization process is different from the first probability initialization process; and performing arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state.

[0160] Aspect 2A. The method of Aspect 1A, wherein the independently codable unit of video data is a slice of video data.

[0161] Aspect 3A. The method of any of Aspects 1A-2A, wherein the arithmetic coding is context adaptive binary arithmetic coding (CABAC).

[0162] Aspect 4A. The method of any of Aspects 1A-3A, wherein determining the first initial probability state for the first probability using the first probability initialization process comprises: determining the first initial probability state for the first probability using the first probability initialization process and parameters stored from a previous picture of video data.

[0163] Aspect 5A. The method of any of Aspects 1A-4A, wherein determining the second initial probability state for the second probability using the second probability initialization process comprises: determining the second initial probability state for the second probability using the second probability initialization process and parameters stored from a previous picture of video data.

[0164] Aspect 6A. A method of coding video data, the method comprising: receiving an independently codable unit of video data; receiving a first syntax element indicating which of a plurality of probability initialization processes is to be used for a first initial probability state for a first probability; receiving a second syntax element indicating which of a plurality of probability initialization processes is to be used for a second initial probability state for a second probability; determining the first initial probability state for the first probability according to a first probability initialization process indicated by the first syntax element; determining the second initial probability state for the second probability according to second probability initialization process indicated by the second syntax element; and performing arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state.

[0165] Aspect 7A. The method of Aspect 6A, wherein the first probability initialization process and the second probability initialization process are the same.

[0166] Aspect 8A. The method of Aspect 6A, wherein the first probability initialization process and the second probability initialization process are different.

[0167] Aspect 9A. The method of any of Aspects 6A-8A, wherein the independently codable unit of video data is a slice of video data.

[0168] Aspect 10A. The method of any of Aspects 6A-9A, wherein the arithmetic coding is context adaptive binary arithmetic coding (CABAC).

[0169] Aspect 11A. The method of any of Aspects 6A-10A, wherein determining the first initial probability state for the first probability using the first probability initialization process comprises: determining the first initial probability state for the first probability using the first probability initialization process and parameters stored from a previous picture of video data.

[0170] Aspect 12A. The method of any of Aspects 6A-11A, wherein determining the second initial probability state for the second probability using the second probability initialization process comprises: determining the second initial probability state for the second probability using the second probability initialization process and parameters stored from a previous picture of video data.

[0171] Aspect 13A. The method of any of Aspects 1A-12A, wherein coding comprises decoding.

[0172] Aspect 14A. The method of any of Aspects 1A-12A, wherein coding comprises encoding.

[0173] Aspect 15A. A device for coding video data, the device comprising one or more means for performing the method of any of Aspects 1A-14A.

[0174] Aspect 16A. The device of Aspect 15A, wherein the one or more means comprise one or more processors implemented in circuitry.

[0175] Aspect 17A. The device of any of Aspects 15A and 16A, further comprising a memory to store the video data.

[0176] Aspect 18A. The device of any of Aspects 15A-17A, further comprising a display configured to display decoded video data.

[0177] Aspect 19A. The device of any of Aspects 15A-18A, wherein the device comprises one or more of a camera, a computer, a mobile device, a broadcast receiver device, or a set-top box.

[0178] Aspect 20A. The device of any of Aspects 15A-19A, wherein the device comprises a video decoder.

[0179] Aspect 21A. The device of any of Aspects 15A-20A, wherein the device comprises a video encoder.

[0180] Aspect 22A. A computer-readable storage medium having stored thereon instructions that, when executed, cause one or more processors to perform the method of any of Aspects 1A-14A.

[0181] Aspect 1B. A method of coding video data, the method comprising: receiving an independently codable unit of video data; determining a first initial probability state for a first probability using a first probability initialization process; determining a second initial probability state for a second probability using a second probability initialization process, wherein the second initial probability state is different from the first initial probability state; and performing arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state.

[0182] Aspect 2B. The method of Aspect 1B, wherein performing arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state comprises: determining the first probability using the first initial probability state, wherein the first probability is a short window probability (P0); determining the second probability using the second initial probability state, wherein the second probability is a long window probability (P1); and performing arithmetic coding of syntax elements in the independently codable unit of video data using the short window probability (P0) and the long window probability (P1).

[0183] Aspect 3B. The method of any of Aspects 1B-2B, wherein the independently codable unit of video data is a slice of video data.

[0184] Aspect 4B. The method of any of Aspects 1B-3B, wherein the arithmetic coding is context adaptive binary arithmetic coding (CABAC).

[0185] Aspect 5B. The method of any of Aspects 1B-4B, wherein determining the first initial probability state for the first probability using the first probability initialization process comprises: determining the first initial probability state for the first probability using the first probability initialization process and parameters stored from a previous picture of video data; and wherein determining the second initial probability state for the second probability using the second probability initialization process comprises: determining the second initial probability state for the second probability using the second probability initialization process and the parameters stored from the previous picture of video data.

[0186] Aspect 6B. The method of any of Aspects 1B-5B, wherein coding is encoding and wherein performing arithmetic coding of syntax elements comprises performing arithmetic encoding of syntax elements.

[0187] Aspect 7B. The method of any of Aspects 1B-5B, wherein coding is decoding and wherein performing arithmetic coding of syntax elements comprises performing arithmetic decoding of syntax elements.

[0188] Aspect 8B. An apparatus configured to code video data, the apparatus comprising: a memory; and processing circuitry in communication with the memory, the processing circuitry configured to: receive an independently codable unit of video data; determine a first initial probability state for a first probability using a first probability initialization process; determine a second initial probability state for a second probability using a second probability initialization process, wherein the second initial probability state is different from the first initial probability state; and perform arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state.

[0189] Aspect 9B. The apparatus of Aspect 8B, wherein to perform arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state, the processing circuitry is further configured to: determine the first probability using the first initial probability state, wherein the first probability is a short window probability (P0); determine the second probability using the second initial probability state, wherein the second probability is a long window probability (P1); and perform arithmetic coding of syntax elements in the independently codable unit of video data using the short window probability (P0) and the long window probability (P1).

[0190] Aspect 10B. The apparatus of any of Aspects 8B-9B, wherein the independently codable unit of video data is a slice of video data.

[0191] Aspect 11B. The apparatus of any of Aspects 8B-10B, wherein the arithmetic coding is context adaptive binary arithmetic coding (CABAC).

[0192] Aspect 12B. The apparatus of any of Aspects 8B-11B, wherein to determine the first initial probability state for the first probability using the first probability initialization process, the processing circuitry is further configured to: determine the first initial probability state for the first probability using the first probability initialization process and parameters stored from a previous picture of video data; and wherein to determine the second initial probability state for the second probability using the second probability initialization process, the processing circuitry is further configured to: determine the second initial probability state for the second probability using the second probability initialization process and the parameters stored from the previous picture of video data.

[0193] Aspect 13B. The apparatus of any of Aspects 8B-12B, wherein the apparatus is configured to encode video data, and wherein to perform arithmetic coding of syntax elements, the processing circuitry is configured to perform arithmetic encoding of syntax elements.

[0194] Aspect 14B. The apparatus of any of Aspects 8B-12B, wherein the apparatus is configured to decode video data, and wherein to perform arithmetic coding of syntax elements, the processing circuitry is configured to perform arithmetic decoding of syntax elements.

[0195] Aspect 15B. A non-transitory computer-readable storage medium storing instructions that, when executed,

cause one or more processors of a device configured to code video data to: receive an independently codable unit of video data; determine a first initial probability state for a first probability using a first probability initialization process; determine a second initial probability state for a second probability using a second probability initialization process, wherein the second initial probability state is different from the first initial probability state; and perform arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state.

[0196] Aspect 16B. The non-transitory computer-readable storage medium of Aspect 15B, wherein to perform arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state, the instructions further cause the one or more processors to: determine the first probability using the first initial probability state, wherein the first probability is a short window probability (P0); determine the second probability using the second initial probability state, wherein the second probability is a long window probability (P1); and perform arithmetic coding of syntax elements in the independently codable unit of video data using the short window probability (P0) and the long window probability (P1).

[0197] Aspect 17B. The non-transitory computer-readable storage medium of any of Aspects 15B-16B, wherein the independently codable unit of video data is a slice of video data, and wherein the arithmetic coding is context adaptive binary arithmetic coding (CABAC).

[0198] Aspect 18B. The non-transitory computer-readable storage medium of any of Aspects 15B-17B, wherein to determine the first initial probability state for the first probability using the first probability initialization process, the instructions further cause the one or more processors to: determine the first initial probability state for the first probability using the first probability initialization process and parameters stored from a previous picture of video data; and wherein to determine the second initial probability state for the second probability using the second probability initialization process, the instructions further cause the one or more processors to: determine the second initial probability state for the second probability using the second probability initialization process and the parameters stored from the previous picture of video data.

[0199] Aspect 19B. The non-transitory computer-readable storage medium of any of Aspects 15B-18B, wherein the instructions cause the one or more processors to encode video data, and wherein to perform arithmetic coding of syntax elements, the instructions further cause the one or more processors to perform arithmetic encoding of syntax elements.

[0200] Aspect 20B. The non-transitory computer-readable storage medium of any of Aspects 15B-18B, wherein the instructions cause the one or more processors to decode video data, and wherein to perform arithmetic coding of syntax elements, the instructions further cause the one or more processors to perform arithmetic decoding of syntax elements.

[0201] It is to be recognized that depending on the example, certain acts or events of any of the techniques described herein can be performed in a different sequence, may be added, merged, or left out altogether (e.g., not all described acts or events are necessary for the practice of the techniques). Moreover, in certain examples, acts or events may be performed concurrently, e.g., through multi-threaded processing, interrupt processing, or multiple processors, rather than sequentially.

[0202] In one or more examples, the functions described may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored on or transmitted over as one or more instructions or code on a computer-readable medium and executed by a hardware-based processing unit. Computer-readable media may include computer-readable storage media, which corresponds to a tangible medium such as data storage media, or communication media including any medium that facilitates transfer of a computer program from one place to another, e.g., according to a communication protocol. In this manner, computer-readable media generally may correspond to (1) tangible computer-readable storage media which is non-transitory or (2) a communication medium such as a signal or carrier wave. Data storage media may be any available media that can be accessed by one or more computers or one or more processors to retrieve instructions, code and/or data structures for implementation of the techniques described in this disclosure. A computer program product may include a computer-readable medium.

[0203] By way of example, and not limitation, such computer-readable storage media may include one or more of RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage, or other magnetic storage devices, flash memory, or any other medium that can be used to store desired program code in the form of instructions or data structures and that can be accessed by a computer. Also, any connection is properly termed a computer-readable medium. For example, if instructions are transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared, radio, and microwave, then the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio, and microwave are included in the definition of medium. It should be understood, however, that computer-readable storage media and data storage media do not include connections, carrier waves, signals, or other transitory media, but are instead directed to non-transitory, tangible storage media. Disk and disc, as used herein, includes compact disc (CD), laser disc, optical disc, digital versatile disc (DVD), floppy disk and Blu-ray disc, where disks usually reproduce data magnetically, while discs reproduce data optically with lasers. Combinations of the above should also be included within the scope of computer-readable media.

[0204] Instructions may be executed by one or more processors, such as one or more DSPs, general purpose microprocessors, ASICs, FPGAs, or other equivalent integrated or discrete logic circuitry. Accordingly, the terms "processor" and "processing circuitry," as used herein may refer to any of the foregoing structures or any other structure suitable for implementation of the techniques described herein. In addition, in some aspects, the functionality described herein may be provided within dedicated hardware and/or software modules configured for encoding and decoding, or incorporated in a combined codec. Also, the techniques could be fully implemented in one or more circuits or logic elements.

[0205] The techniques of this disclosure may be implemented in a wide variety of devices or apparatuses, includ-

ing a wireless handset, an integrated circuit (IC) or a set of ICs (e.g., a chip set). Various components, modules, or units are described in this disclosure to emphasize functional aspects of devices configured to perform the disclosed techniques, but do not necessarily require realization by different hardware units. Rather, as described above, various units may be combined in a codec hardware unit or provided by a collection of interoperative hardware units, including one or more processors as described above, in conjunction with suitable software and/or firmware.

[0206] Various examples have been described. These and other examples are within the scope of the following claims.

What is claimed is:

1. A method of coding video data, the method comprising:

receiving an independently codable unit of video data;

determining a first initial probability state for a first probability using a first probability initialization process;

determining a second initial probability state for a second probability using a second probability initialization process, wherein the second initial probability state is different from the first initial probability state; and

performing arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state.

2. The method of claim **1**, wherein performing arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state comprises:

determining the first probability using the first initial probability state, wherein the first probability is a short window probability (P0);

determining the second probability using the second initial probability state, wherein the second probability is a long window probability (P1); and

performing arithmetic coding of syntax elements in the independently codable unit of video data using the short window probability (P0) and the long window probability (P1).

3. The method of claim **1**, wherein the independently codable unit of video data is a slice of video data.

4. The method of claim **1**, wherein the arithmetic coding is context adaptive binary arithmetic coding (CABAC).

5. The method of claim **1**, wherein determining the first initial probability state for the first probability using the first probability initialization process comprises:

determining the first initial probability state for the first probability using the first probability initialization process and parameters stored from a previous picture of video data; and

wherein determining the second initial probability state for the second probability using the second probability initialization process comprises:

determining the second initial probability state for the second probability using the second probability initialization process and the parameters stored from the previous picture of video data.

6. The method of claim **1**, wherein coding is encoding and wherein performing arithmetic coding of syntax elements comprises performing arithmetic encoding of syntax elements.

7. The method of claim **1**, wherein coding is decoding and wherein performing arithmetic coding of syntax elements comprises performing arithmetic decoding of syntax elements.

8. An apparatus configured to code video data, the apparatus comprising:

a memory; and

processing circuitry in communication with the memory, the processing circuitry configured to:

receive an independently codable unit of video data;

determine a first initial probability state for a first probability using a first probability initialization process;

determine a second initial probability state for a second probability using a second probability initialization process, wherein the second initial probability state is different from the first initial probability state; and

perform arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state.

9. The apparatus of claim **8**, wherein to perform arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state, the processing circuitry is further configured to:

determine the first probability using the first initial probability state, wherein the first probability is a short window probability (P0);

determine the second probability using the second initial probability state, wherein the second probability is a long window probability (P1); and

perform arithmetic coding of syntax elements in the independently codable unit of video data using the short window probability (P0) and the long window probability (P1).

10. The apparatus of claim **8**, wherein the independently codable unit of video data is a slice of video data.

11. The apparatus of claim **8**, wherein the arithmetic coding is context adaptive binary arithmetic coding (CABAC).

12. The apparatus of claim **8**, wherein to determine the first initial probability state for the first probability using the first probability initialization process, the processing circuitry is further configured to:

determine the first initial probability state for the first probability using the first probability initialization process and parameters stored from a previous picture of video data; and

wherein to determine the second initial probability state for the second probability using the second probability initialization process, the processing circuitry is further configured to:

determine the second initial probability state for the second probability using the second probability initialization process and the parameters stored from the previous picture of video data.

13. The apparatus of claim **8**, wherein the apparatus is configured to encode video data, and wherein to perform arithmetic coding of syntax elements, the processing circuitry is configured to perform arithmetic encoding of syntax elements.

14. The apparatus of claim **8**, wherein the apparatus is configured to decode video data, and wherein to perform

arithmetic coding of syntax elements, the processing circuitry is configured to perform arithmetic decoding of syntax elements.

15. A non-transitory computer-readable storage medium storing instructions that, when executed, cause one or more processors of a device configured to code video data to:

receive an independently codable unit of video data;

determine a first initial probability state for a first probability using a first probability initialization process;

determine a second initial probability state for a second probability using a second probability initialization process, wherein the second initial probability state is different from the first initial probability state; and

perform arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state.

16. The non-transitory computer-readable storage medium of claim **15**, wherein to perform arithmetic coding of syntax elements in the independently codable unit of video data using the first initial probability state and the second initial probability state, the instructions further cause the one or more processors to:

determine the first probability using the first initial probability state, wherein the first probability is a short window probability (P0);

determine the second probability using the second initial probability state, wherein the second probability is a long window probability (P1); and

perform arithmetic coding of syntax elements in the independently codable unit of video data using the short window probability (P0) and the long window probability (P1).

17. The non-transitory computer-readable storage medium of claim **15**, wherein the independently codable unit of video data is a slice of video data, and wherein the arithmetic coding is context adaptive binary arithmetic coding (CABAC).

18. The non-transitory computer-readable storage medium of claim **15**, wherein to determine the first initial probability state for the first probability using the first probability initialization process, the instructions further cause the one or more processors to:

determine the first initial probability state for the first probability using the first probability initialization process and parameters stored from a previous picture of video data; and

wherein to determine the second initial probability state for the second probability using the second probability initialization process, the instructions further cause the one or more processors to:

determine the second initial probability state for the second probability using the second probability initialization process and the parameters stored from the previous picture of video data.

19. The non-transitory computer-readable storage medium of claim **15**, wherein the instructions cause the one or more processors to encode video data, and wherein to perform arithmetic coding of syntax elements, the instructions further cause the one or more processors to perform arithmetic encoding of syntax elements.

20. The non-transitory computer-readable storage medium of claim **15**, wherein the instructions cause the one or more processors to decode video data, and wherein to perform arithmetic coding of syntax elements, the instructions further cause the one or more processors to perform arithmetic decoding of syntax elements.

* * * * *