



(86) Date de dépôt PCT/PCT Filing Date: 2012/11/07
(87) Date publication PCT/PCT Publication Date: 2013/05/16
(85) Entrée phase nationale/National Entry: 2014/05/07
(86) N° demande PCT/PCT Application No.: US 2012/063920
(87) N° publication PCT/PCT Publication No.: 2013/070746
(30) Priorités/Priorities: 2011/11/08 (US61/557,275);
2012/04/16 (US61/624,714); 2012/11/06 (US13/670,176)

(51) Cl.Int./Int.Cl. *H04N 19/40* (2014.01),
H04N 19/597 (2014.01), *H04N 21/643* (2011.01)
(71) Demandeur/Applicant:
SONY CORPORATION, JP
(72) Inventeurs/Inventors:
HAQUE, MUNSI, US;
TABATABAI, ALI, US
(74) Agent: GOWLING LAFLEUR HENDERSON LLP

(54) Titre : SYSTEME DE CODAGE VIDEO ET SON PROCEDE DE FONCTIONNEMENT
(54) Title: VIDEO CODING SYSTEM AND METHOD OF OPERATION THEREOF

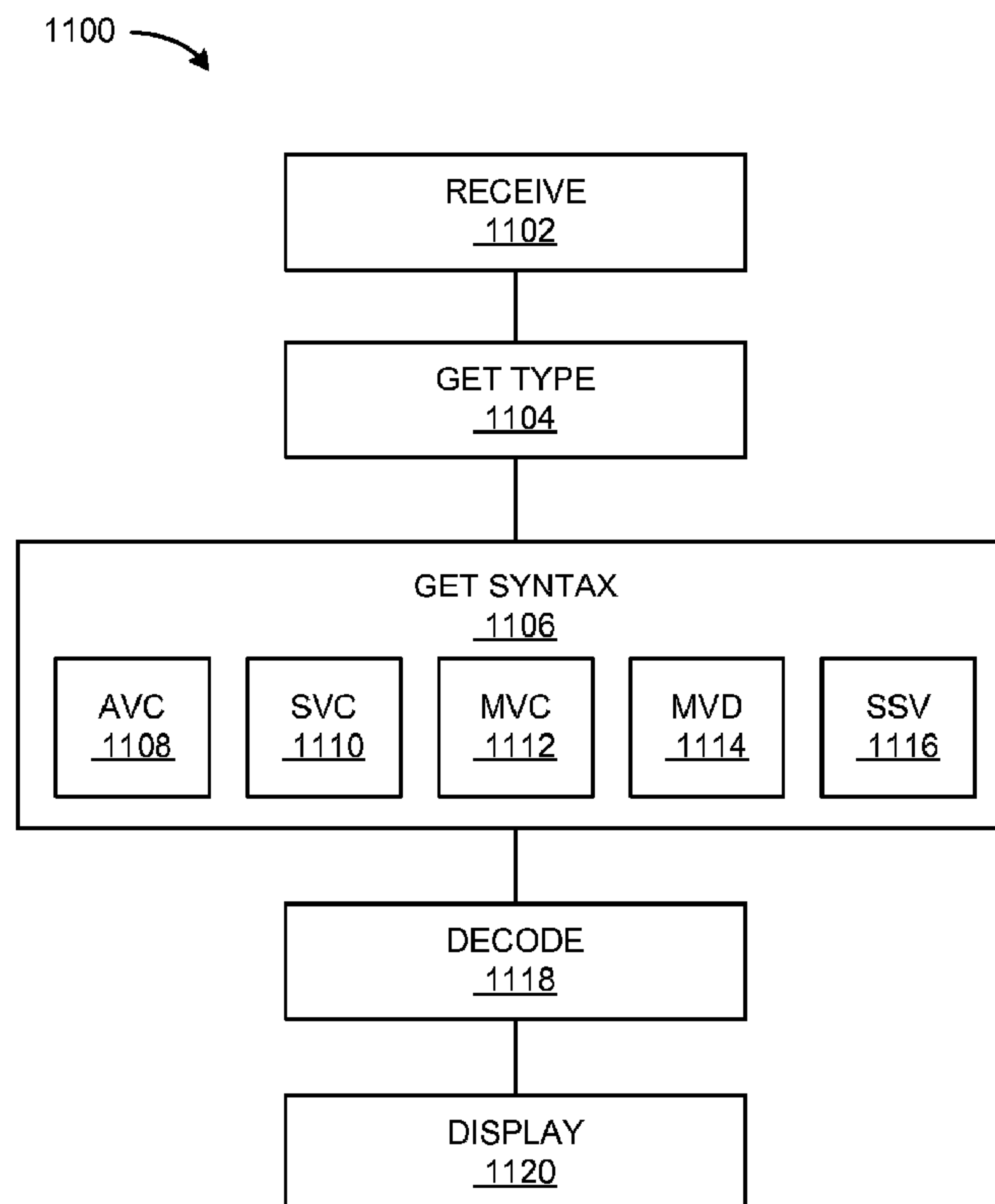


FIG. 11

(57) Abrégé/Abstract:

A method (1200) of operation of a video coding system (100) includes: receiving a video bitstream (110); identifying a syntax type (132) of the video bitstream (110); extracting a video syntax (114) from the video bitstream (110) for the syntax type (132); and forming a video stream (112) based on the video syntax (114) for displaying on a device (102).



- (51) International Patent Classification:
H04N 7/26 (2006.01) H04N 13/00 (2006.01)

(21) International Application Number:
PCT/US2012/063920

(22) International Filing Date:
7 November 2012 (07.11.2012)

(25) Filing Language:
English

(26) Publication Language:
English

(30) Priority Data:
61/557,275 8 November 2011 (08.11.2011) US
61/624,714 16 April 2012 (16.04.2012) US
13/670,176 6 November 2012 (06.11.2012) US

(71) Applicant: SONY CORPORATION [JP/JP]; 1-7-1 Konan, Minato-ku, Tokyo 108-0075 (JP).

(72) Inventors; and

(71) Applicants (for US only): HAQUE, Munsif [US/US]; 5702 Pogliana Court, San Jose, California 95138 (US).
- TABATABAI, Ali [US/US]; 10265 E. Estates Dr., Cupertino, California 95014 (US).

(74) Agents: FUJII, Harold T. et al.; Sony Corporation of America, 16530 Via Esprillo, MZ7190, San Diego, California 92127 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ,

[Continued on next page]

(54) Title: VIDEO CODING SYSTEM AND METHOD OF OPERATION THEREOF

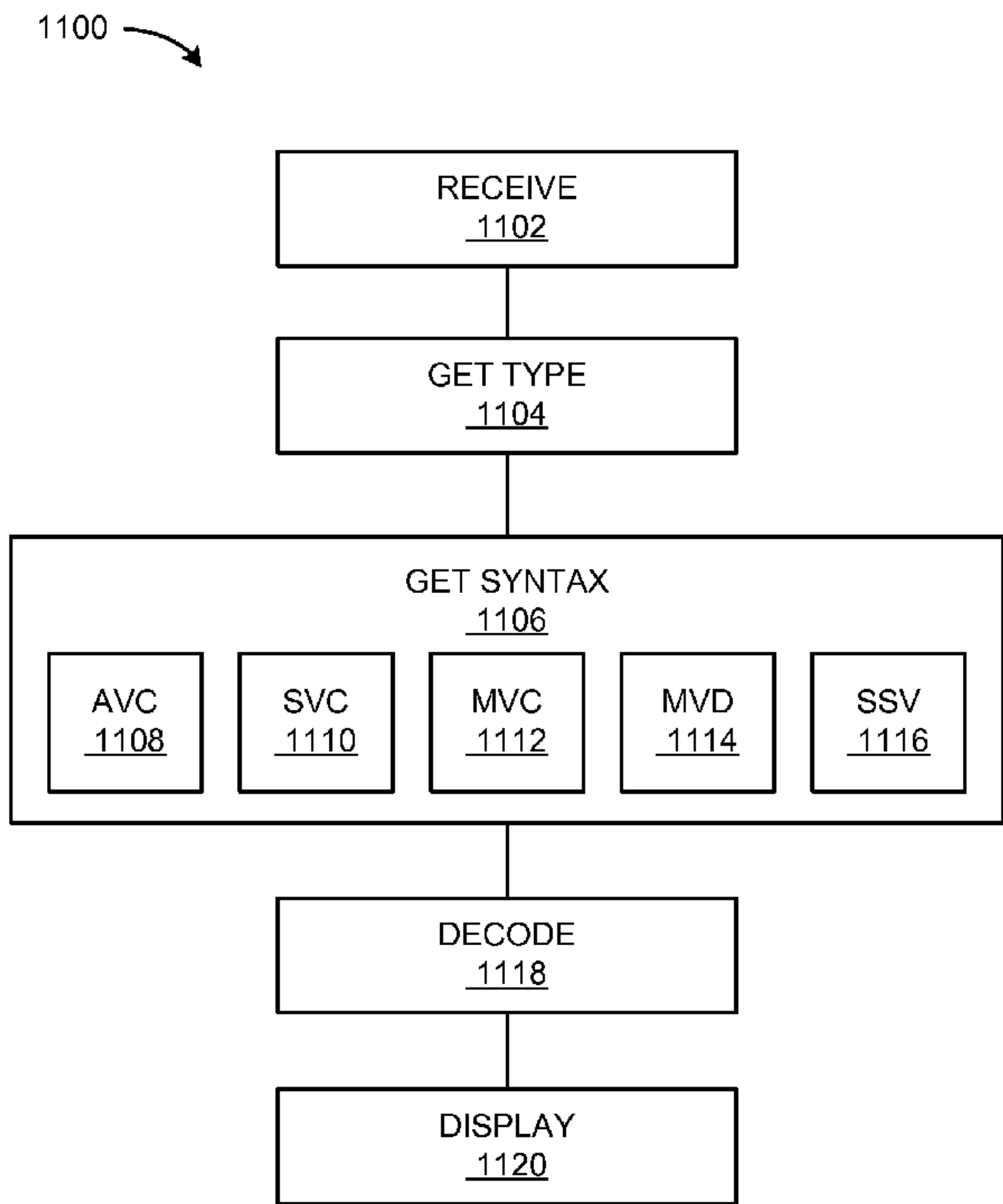


FIG. 11

(57) Abstract: A method (1200) of operation of a video coding system (100) includes: receiving a video bitstream (110); identifying a syntax type (132) of the video bitstream (110); extracting a video syntax (114) from the video bitstream (110) for the syntax type (132); and forming a video stream (112) based on the video syntax (114) for displaying on a device (102).

WO 2013/070746 A2

WO 2013/070746 A2

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

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

Published:

— without international search report and to be republished upon receipt of that report (Rule 48.2(g))
- Declarations under Rule 4.17:**
- as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))

VIDEO CODING SYSTEM AND METHOD OF OPERATION THEREOF

CROSS-REFERENCE TO RELATED APPLICATION

This application claims the benefit of U.S. Provisional Patent Application serial number 61/557,275 filed November 8, 2011, U.S. Provisional Patent Application serial number 61/624,714 filed April 16, 2012, and U.S. Patent Application serial number 13/670,176 filed November 6, 2012 and the subject matter thereof is incorporated herein by reference in its entirety.

TECHNICAL FIELD

The present invention relates generally to video systems, and more particularly to a system for video coding.

BACKGROUND ART

The deployment of high quality video to smart phones, high definition televisions, automotive information systems, and other video devices with screens has grown tremendously in recent years. The wide variety of information devices supporting video content requires multiple types of video content to be provided to devices with different size, quality, and connectivity capabilities.

Video has evolved from two dimensional single view video to multiview video with high-resolution three dimensional imagery. In order to make the transfer of video more efficient, different video coding and compression schemes have tried to get the best picture from the least amount of data. The Moving Pictures Experts Group (MPEG) developed standards to allow good video quality based on a standardized data sequence and algorithm. The H.264 (MPEG4 Part 10) / Advanced Video Coding design was an improvement in coding efficiency typically by a factor of two over the prior MPEG-2 format. The quality of the video is dependent upon the manipulation and compression of the data in the video. The video can be modified to accommodate the varying bandwidths used to send the video to the display devices with different resolutions and feature sets. However, distributing larger,

higher quality video, or more complex video functionality requires additional bandwidth and improved video compression.

Thus, a need still remains for a video coding system that can deliver good picture quality and features across a wide range of device with different sizes, resolutions, and connectivity. In view of the increasing demand for providing video on the growing spectrum of intelligent devices, it is increasingly critical that answers be found to these problems. In view of the ever-increasing commercial competitive pressures, along with growing consumer expectations and the diminishing opportunities for meaningful product differentiation in the marketplace, it is critical that answers be found for these problems. Additionally, the need to save costs, improve efficiencies and performance, and meet competitive pressures, adds an even greater urgency to the critical necessity for finding answers to these problems.

Solutions to these problems have long been sought but prior developments have not taught or suggested any solutions and, thus, solutions to these problems have long eluded those skilled in the art.

DISCLOSURE OF THE INVENTION

The present invention provides a method of operation of a video coding system including: receiving a video bitstream; identifying a syntax type of the video bitstream; extracting a video syntax from the video bitstream for the syntax type; and forming a video stream based on the video syntax for displaying on a device.

The present invention provides a video coding system, including: a receive module for receiving a video bitstream; a get type module, coupled to the receive module, for identifying a syntax type from the video bitstream; a get syntax module, coupled to the get type module, for extracting a video syntax from the video bitstream for the syntax type; and a decode module, coupled to the get syntax module, for forming a video stream based on the video syntax and the video bitstream for displaying on a device.

Certain embodiments of the invention have other aspects in addition to or in place of those mentioned above. The aspects will become apparent to those skilled in the art from a reading of the following detailed description when taken with reference to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a video coding system in an embodiment of the present invention.

FIG. 2 is an example of an Advanced Video Coding (AVC) Video Usability Information (VUI) syntax.

FIG. 3 is an example of a Scalable Video Coding (SVC) VUI syntax.

FIG. 4 is an example of a SVC VUI syntax extension.

FIG. 5 is an example of a Multiview Video Coding (MVC) VUI syntax.

FIG. 6 is an example of a MVC VUI syntax extension.

FIG. 7 is an example of a Multiview Video plus Depth (MVD) VUI syntax.

FIG. 8 is an example of a MVD VUI syntax extension.

FIG. 9 is an example of a Stereoscopic Video (SSV) VUI syntax extension.

FIG. 10 is a functional block diagram of the video coding system.

FIG. 11 is a control flow of the video coding system.

FIG. 12 is a flow chart of a method of operation of the video coding system in a further embodiment of the present invention.

BEST MODE FOR CARRYING OUT THE INVENTION

The following embodiments are described in sufficient detail to enable those skilled in the art to make and use the invention. It is to be understood that other embodiments would be evident based on the present disclosure, and that process or mechanical changes may be made without departing from the scope of the present invention.

In the following description, numerous specific details are given to provide a thorough understanding of the invention. However, it will be apparent that the invention may be practiced without these specific details. In order to avoid obscuring the present invention, some well-known circuits, system configurations, and process steps are not disclosed in detail.

Likewise, the drawings showing embodiments of the system are semi-diagrammatic and not to scale and, particularly, some of the dimensions are for the clarity of presentation and are shown greatly exaggerated in the drawing FIGs. Where multiple embodiments are disclosed and described, having some features in common, for clarity and ease of

illustration, description, and comprehension thereof, similar and like features one to another will ordinarily be described with like reference numerals.

The term “syntax” means the set of elements describing a data structure. The term “module” referred to herein can include software, hardware, or a combination thereof in the present invention in accordance with the context used.

Referring now to FIG. 1, therein is shown a block diagram of a video coding system 100 in an embodiment of the present invention. A video encoder 102 can receive a video content 108 and send a video bitstream 110 to a video decoder 104 for decoding and display on a display interface 120.

The video encoder 102 can receive and encode the video content 108. The video encoder 102 is a unit for encoding the video content 108 into a different form. The video content 108 is defined as a visual representation of a scene of objects.

Encoding is defined as computationally modifying the video content 108 to a different form. For example, encoding can compress the video content 108 into the video bitstream 110 to reduce the amount of data needed to transmit the video bitstream 110.

In another example, the video content 108 can be encoded by being compressed, visually enhanced, separated into one or more views, changed in resolution, changed in aspect ratio, or a combination thereof. In another illustrative example, the video content 108 can be encoded according to High-Efficiency Video Coding (HEVC)/H.265

The video encoder 102 can encode the video content 108 to form the video bitstream 110. The video bitstream 110 is defined a sequence of bits representing information associated with the video content 108. For example, the video bitstream 110 can be a bit sequence representing a compression instance of the video content 108.

The video encoder 102 can receive the video content 108 for a scene in a variety of ways. For example, the video content 108 representing objects in the real-world can be captured with a video camera, multiple cameras, generated with a computer, provided as a file, or a combination thereof.

The video content 108 can support a variety of video features. For example, the video content 108 can include single view video, multiview video, stereoscopic video, or a combination thereof. In a further example, the video content 108 can be multiview video of four or more cameras for supporting three-dimensional (3D) video viewing without 3D glasses.

The video encoder 102 can encode the video content 108 using a video syntax 114 to generate the video bitstream 110. The video syntax 114 is defined as a set of information elements that describe a coding methodology for encoding and decoding the video content 108. The video bitstream 110 is compliant with the video syntax 114, such as the High-Efficiency Video Coding/H.265 standard, and can include a HEVC video bitstream, an Ultra High Definition video bitstream, or a combination thereof.

The video bitstream 110 can include information representing the imagery of the video content 108 and the associated control information related to the encoding of the video content 108. For example, the video bitstream 110 can include an instance of the video syntax 114 and an instance of the video content 108.

The video coding system 100 can include the video decoder 104 for decoding the video bitstream 110. The video decoder 104 is defined as a unit for receiving the video bitstream 110 and modifying the video bitstream 110 to form a video stream 112.

The video decoder 104 can decode the video bitstream 110 to form the video stream 112 using the video syntax 114. Decoding is defined as computationally modifying the video bitstream 110 to form the video stream 112. For example, decoding can decompress the video bitstream 110 to form the video stream 112 formatted for displaying on a smart phone display.

The video stream 112 is defined as a computationally modified version of the video content 108. For example, the video stream 112 can include a modified instance of the video content 108 with different properties. The video stream 112 can include cropped decoded pictures from the video content 108.

In a further example, the video stream 112 can have a different resolution, a different aspect ratio, a different frame rate, different stereoscopic views, different view order, or a combination thereof than the video content 108. The video stream 112 can have different visual properties including different color parameters, color planes, contrast, hue, or a combination thereof.

The video coding system 100 can include a display processor 118. The display processor 118 can receive the video stream 112 from the video decoder 104 for display on the display interface 120. The display interface 120 is a unit that can present a visual representation of the video stream 112. For example, the display interface 120 can include a smart phone display, a digital projector, a DVD player display, or a combination thereof.

The video encoder 102 can send the video bitstream 110 to the video decoder 104 over a communication path 106. The communication path 106 can be a variety of networks.

For example, the communication path 106 can include wireless communication, wired communication, optical, ultrasonic, or the combination thereof. Satellite communication, cellular communication, Bluetooth, Infrared Data Association standard (IrDA), wireless fidelity (WiFi), and worldwide interoperability for microwave access (WiMAX) are examples of wireless communication that can be included in the communication path 106. Ethernet, digital subscriber line (DSL), fiber to the home (FTTH), and plain old telephone service (POTS) are examples of wired communication that can be included in the communication path 106.

The video coding system 100 can employ a variety of video coding standards. For example, the video coding system 100 can encode and decode video information using the High Efficiency Video Coding / H.265 working draft version. The HEVC draft version is described in documents that are hereby included by reference. The documents incorporated by reference include:

B. Boss, W. Han, J Ohm, G. Sullivan, T. Wiegand, "WD4 Working Draft 4 of High-Efficiency Video Coding", JCTVC-F803 d1, July 2011 (Torino).

M. Haque, A. Tabatabai, T. Suzuki, "On VUI syntax parameters", JCTVC-F289, July 2011 (Torino).

M. Haque, K. Sato, A. Tabatabai, T. Suzuki, "HEVC VUI Parameters with Extension Hooks", JCTVC-J0270, July 2012 (Stockholm).

M. Haque, K. Sato, A. Tabatabai, T. Suzuki, "Simplifications of HRD parameters for Temporal Scalability", JCTVC-J0272, July 2012 (Stockholm).

M. Haque, K. Sato, A. Tabatabai, T. Suzuki, "A simple ordering issue for VUI parameters syntax", JCTVC-J0273, July 2012 (Stockholm).

B. Boss, W. Han, J Ohm, G. Sullivan, T. Wiegand, "High-Efficiency Video Coding (HEVC) text specification draft 8", JCTVC-J1003 d7, July 2012 (Stockholm).

The video bitstream 110 can include a variety of video types as indicated by a syntax type 132. The syntax type 132 is defined as an indicator of the video coding used to encode and decode the video bitstream 110. For example, the video content 108 can include the syntax type 132 for advanced video coding 122, scalable video coding 124, multiview video coding 126, multiview video plus depth video 128, and stereoscopic video 130.

Advanced video coding and scalable video coding can be used to encode single view based video to form the video bitstream 110. The single view-based video can include the video content 108 generate from a single camera.

Multiview video coding, multiview video plus depth, and stereoscopic video can be used to encode the video content 108 having two more views. For example, multiview video can include the video content 108 from multiple cameras.

The video syntax 114 can include an entry identifier 134. The entry identifier 134 is a value for differentiating between multiple coded video sequences. The coded video sequences can include instances of the video content 108 having a different bit-rate, frame-rate, resolution, or scalable layers for a single view video, multiview video, or stereoscopic video.

The video syntax 114 can include an entry count 136 for identifying the number of entries associated with each frame in the video content 108. The entry count 136 is the maximum number of entries represented in the video content 108.

The video syntax 114 can include an iteration identifier 138. The iteration identifier 138 is a value to differentiate between individual iterations of the video content 108.

The video syntax 114 can include an iteration count 140. The iteration count 140 is a value indicating the maximum number of iterations of the video content 108.

For scalable video coding, the term iteration count can be used to indicate the number of information entries tied to different scalable video layers in the case of scalable video coding. For multiview video coding, the iteration count can be used to indicate the number of operation points tied to the number of views of the video content 108.

For example, in scalable video coding, the video content 108 can be encoded to include a base layer with additional enhancement layers to form multi-layer instances of the video bitstream 110. The base layer can have the lowest resolution, frame-rate, or quality.

The enhancement layers can include gradual refinements with additional left-over information used to increase the quality of the video. The scalable video layer extension can include a new baseline standard of HEVC that can be extended to cover scalable video coding.

The video syntax 114 can include an operation identifier 142. The operation identifier 142 is a value to differentiate between individual operation points of the video content 108. The operation points are information entries present for multiview video coding, such as timing information, network abstraction layer (NAL) hypothetical

referenced decoder (HRD) parameters, video coding layer (VCL) HRD parameters, a pic_struct_present_flag element, or a combination thereof.

The video syntax 114 can include an operation count 144. The operation count 144 is a value indicating the maximum number of operations of the video content 108.

5 The operation points are tied to generation of coded video sequences from various views, such as views generated by different cameras, for multiview and 3D video. For multiview video coding, an operation point is associated with a subset of the video bitstream 110 having a target output view and the other views dependent on the target output view. The other views are dependent on the target output view if they are derived using a sub-
10 bitstream extraction process. More than one operation point may be associated with the same subset of the video bitstream 110. For example, decoding an operation point refers to the decoding of the subset of the video bitstream corresponding to the operation point and subsequent output of the target output views as a portion of the video stream 112 for display on the device 102.

15 The video syntax 114 can include a view identifier 146. The view identifier 146 is a value to differentiate between individual views of the video content 108.

The video syntax 114 can include a view count 148. The view count 148 is a value indicating the maximum number of views of the video content 108.

For example, a single view can be a video generated by a single camera. Multiview
20 video can be generated by multiple cameras situated at different positions and distances from the objects being viewed in a scene.

The video content 108 can include a variety of video properties. For example, the video content 108 can be high resolution video, such as Ultra High Definition video. The video content 108 can have a resolution of 3840 x 2160 or higher, including resolutions of
25 7680 x 4320, 8K x 2K, 4K x 2K, or a combination thereof. Although the video content 108 supports high resolution video, it is understood that the video content 108 can also support lower resolutions, such as high definition (HD) video. The video syntax 114 can support the resolution of the video content 108.

The video content 108 can support a variety of frame rates including 24 frames per
30 second (fps), 25 fps, 50 fps, 60 fps, and 120 fps. Although individual frame rates are described, it is understood that the video content 108 can support fixed and variable rational frame rates of zero frames per second and higher. The video syntax 114 can support the frame rate of the video content 108.

Referring now to FIG. 2, therein is shown an example of an Advanced Video Coding (AVC) Video Usability Information (VUI) syntax 202. The AVC VUI syntax 202 describes configuration elements of the video syntax 114 of FIG. 1 for HEVC.

The AVC VUI syntax 202 includes elements as described in the AVC VUI syntax table of FIG. 2. The elements of the AVC VUI syntax 202 are arranged in a hierarchical structure as described in the AVC VUI syntax table of FIG. 2.

The AVC VUI syntax 202 includes a variety of elements to support the processing of Video Usability Information for HEVC. Processing is defined as modifying video information based on the video syntax 114. For example, processing can include encoding or decoding the video content 108 of FIG. 1 and the video bitstream 110 of FIG. 1 respectively.

The AVC VUI syntax 202 includes an AVC VUI syntax header 204, such as a vui_parameters element. The AVC VUI syntax header 204 is a descriptor for identifying the AVC VUI syntax 202. The AVC VUI syntax 202 is used to encode and decode the video bitstream 110 for AVC.

The AVC VUI syntax can include a coding unit 206, such as a max_bits_per_cu_denom element, to indicate the maximum number of bits per coding unit. The coding unit 206 is a rectangular area of one image of the video content 108 used for compression of the video bitstream 110. The max_bits_per_cu_denom message can replaced the max_bits_per_mb_denom messages in the AVC VUI.

It has been discovered that encoding and decoding the video content 108 using the AVC VUI syntax 202 can reduce the size of the video bitstream 110 and reduce the need for video buffering. Reducing the size of the video bitstream 110 increases functionality and increases the performance of display of the video stream 120 of FIG. 1.

Referring now to FIG. 3, therein is shown an example of a Scalable Video Coding (SVC) VUI syntax 302. The SVC VUI syntax 302 enables an instance of the video bitstream 110 of FIG. 1 to be used at different frame rates, spatial resolutions, or quality levels.

The SVC VUI syntax 302 includes elements as described in the SVC VUI syntax table of FIG. 3. The elements of the SVC VUI syntax 302 are arranged in a hierarchical structure as described in the table of FIG. 3.

The SVC VUI syntax 302 includes a SVC VUI syntax header 304, such as a svc_vui_parameters_extensions element. The SVC VUI syntax header 304 is a descriptor

for identifying the SVC VUI syntax 302. The SVC VUI syntax 302 is used to encode and decode the video bitstream 110 for SVC.

The SVC VUI syntax 302 can include the coding unit 206 of FIG. 2, such as a max_bits_per_cu_denom element to indicate the maximum number of bits per coding unit.

5 The max_bits_per_cu_denom message can replaced the max_bits_per_mb_denom messages in the ADC VUI.

The SVC VUI syntax 302 can include the entry identifier 134, such as the element [i]. The SVC VUI syntax 302 can include the entry count 136, such as vui_ext_num_entries_minus1 element, for identifying the number of entries associated with each frame in the video content 108 of FIG. 1. The entry count 136 indicates the number of
10 entries minus 1 to map the entry count 136 from 0 to the number of entries minus 1.

It has been discovered that the SVC VUI syntax 302 enables video scalability by including the vui_ext_dependency_id element, the vui_ext_quality_id element, and the vui_temporal_id element for each entry defined by the vui_ext_num_entries_minus1
15 element. Spatial scalability, temporal scalability, and quality scalability can be implemented based on the value of the elements for each entry.

It has been discovered that encoding and decoding the video content 108 using the SVC VUI syntax 302 can reduce the size of the video bitstream 110 and reduce the need for video buffering. Reducing the size of the video bitstream 110 increases functionality and
20 increases the performance of display of the video stream 112 of FIG. 1.

Referring now to FIG. 4, therein is shown an example of a SVC VUI syntax extension 402. The SVC VUI syntax extension 402 includes descriptive video information for Advanced Video Coding and Scalable Video Coding for HEVC.

The SVC VUI syntax extension 402 includes elements as described in the SVC VUI syntax extension table of FIG. 4. The elements of the SVC VUI syntax extension 402 are
25 arranged in a hierarchical structure as described in the SVC VUI syntax extension table of FIG. 4.

The SVC VUI syntax extension 402 includes a SVC VUI syntax extension header 404, such as a vui_parameters element. The SVC VUI syntax extension header 404 is a
30 descriptor for identifying the SVC VUI syntax extension 402. The SVC VUI syntax extension 402 is used to encode and decode the video bitstream 110 of FIG. 1 for SVC.

The SVC VUI syntax extension 402 can include the type indicator 406, such as a svc_mvc_flag element, for identifying the type of coding used for the video bitstream 110.

For example, the type indicator 406 can represent the type of coding using 0 to indicate AVC and 1 to indicate SVC.

The SVC VUI syntax extension 402 can include the entry count 136 of FIG. 1, such as num_entries_minus1 element, for identifying the number of entries associated with each frame in the video content 108 of FIG. 1. The entry count 136 indicates the number of entries minus 1 to map the entry count 136 from 0 to the number of entries minus 1.

For example, the entry count 136 can represent the number of entries associated with a stereoscopic instance of the video content 108. The entry count 136 can have a value of 1 to indicate that two images are associated with each frame and a value of 0 to represent the video bitstream 110 with only a single image per frame.

The SVC VUI syntax extension 402 can include a temporal identifier 410, such as a temporal_id element, to indicate the maximum number of temporal layers in the video content 108. The SVC VUI syntax extension 402 can include a dependency identifier 412, such as a dependency_id element, to indicate the spatial dependency between images. The SVC VUI syntax extension 402 can include a quality identifier 414, such as a quality_id element, to indicate a quality level identifier.

The dependency_id element and the quality_id element can be concatenated together to indicate the maximum value of DQID, data quality identification, for each subset of coded video sequences in the SVC VUI syntax extension 402 for HEVC. The maximum value of DQID is calculated by adding the dependency_id element and the quality_id element.

It has been discovered that encoding and decoding the video bitstream 110 using the SVC VUI syntax extension 402 increases video display quality, scalability, and reliability. Identifying and linking multiple images using the temporal_id, dependency_id, and quality_id defines the relationship between images to increase the quality of video display.

It has been discovered that encoding and decoding the video content 108 using the SVC VUI syntax extension 402 can reduce the size of the video bitstream 110 and reduce the need for video buffering. Reducing the size of the video bitstream 110 increases functionality and increases the performance of display of the video stream 120 of FIG. 1.

Referring now to FIG. 5, therein is shown an example of a Multiview Video Coding (MVC) VUI syntax 502. The MVC VUI syntax 502 includes descriptive information for encoding and decoding the video content 108 of FIG. 1 having multiview video information.

The MVC VUI syntax 502 includes elements as described in the MVC VUI syntax table of FIG. 5. The elements of the MVC VUI syntax 502 are arranged in a hierarchical structure as described in the MVC VUI syntax table of FIG. 5.

The MVC VUI syntax 502 includes a MVC VUI syntax header 504, such as a
5 mvc_vui_parameters_extension element. The MVC VUI syntax header 504 is a descriptor for identifying the MVC VUI syntax 502 for HEVC. The MVC VUI syntax 502 is used to encode and decode the video bitstream 110 of FIG. 1 for MVC.

Multiview video coding is for enabling efficient encoding and decoding of multiple video sequences within a single compressed instance of the video bitstream 110. MVC can
10 be used to encode stereoscopic video, as well as other types of three-dimensional (3D) video.

The MVC VUI syntax 502 can include the operation count 144 of FIG. 1, such as a vui_mvc_num_ops_minus1 element to identify the total number of operations in the video bitstream 110. The vui_mvc_num_ops_minus1 specifies the number of operation points for
15 information entries present for multiview video coding, such as timing information, NAL HRD parameters, VCL HRD parameters, a pic_struct_present_flag element, or a combination thereof. The MVC VUI syntax 502 can include the operation identifier 142 of FIG. 1, such as the counter [i].

It has been discovered that encoding and decoding the video content 108 using the
20 MVC VUI syntax 502 can reduce the size of the video bitstream 110 and reduce the need for video buffering. Reducing the size of the video bitstream 110 increases functionality and increases the performance of display of the video stream 112 of FIG. 1.

Referring now to FIG. 6, therein is shown an example of a MVC VUI syntax extension 602. The MVC VUI syntax extension 602 is a combination of Advanced Video
25 Coding, Scalable Video Coding, and Multiview Video Coding elements.

The MVC VUI syntax extension 602 includes elements as described in the MVC VUI syntax extension table of FIG. 6. The elements of the MVC VUI syntax extension 602 are arranged in a hierarchical structure as described in the MVC VUI syntax extension table of FIG. 6.

The MVC VUI syntax extension 602 includes a MVC extension header 604, such as
30 a vui_parameters element. The MVC VUI syntax extension 602 is a descriptor for identifying the MVC VUI syntax extension 602 for HEVC. The MVC VUI syntax

extension 602 is used to encode and decode the video bitstream 110 of FIG. 1 for AVC, SVC, and MVC video.

The MVC VUI syntax extension 602 can include the type indicator 406 of FIG. 4, such as a `svc_mvc_flag` element, for identifying the type of coding used for the video
5 bitstream 110. For example, the type indicator 406 can represent the type of coding using a value of 0 to indicate AVC, 1 to indicate SVC, and 2 to indicate MVC.

The MVC VUI syntax extension 602 can include the iteration identifier 138 for differentiating between multiple coded video sequences. The MVC VUI syntax extension 602 can include the iteration count 140, such as `num_iterations_minus1` element, for
10 identifying the number of iterations associated with each frame in the video content 108 of FIG. 1. Each iteration can represent one of multiple scalable video layer extensions. The iteration count 140 indicates the number of iterations minus 1 to map the range of iterations from 0 to the number of iterations minus 1.

For SVC video, the `num_iterations_minus1` element indicates multiple iterations for
15 multiple scalable video layer extensions. For MVC video, the `num_iterations_minus1` element indicates multiple operation points for multi-view video.

The MVC VUI syntax extension 602 can include the view identifier 146, such as a `view_id` element. The view identifier 146 is a value identifying a view within a multiview configuration for displaying the video content 108.

It has been discovered that encoding and decoding the video bitstream 110 using the
20 MVC VUI syntax extension 602 increases video display quality, scalability, and reliability. Identifying and linking multiple images from multiple views using the `temporal_id`, `dependency_id`, and `quality_id` defines the relationship between images to increase the quality of video display.

It has been discovered that encoding and decoding the video content 108 using the
25 MVC VUI syntax extension 602 can reduce the size of the video bitstream 110 and reduce the need for video buffering. Reducing the size of the video bitstream 110 increases functionality and increases the performance of display of the video stream 112 of FIG. 1.

Referring now to FIG. 7, therein is shown an example of a Multiview Video plus
30 Depth (MVD) VUI syntax 702. The MVD VUI syntax 702 includes descriptive information for encoding and decoding the video content 108 of FIG. 1 having three-dimensional video (3DV) information and scalable video coding information.

The MVD VUI syntax 702 includes elements as described in the MVD VUI syntax table of FIG. 7. The elements of the MVD VUI syntax 702 are arranged in a hierarchical structure as described in the MVD VUI syntax extension table of FIG. 7.

The MVD VUI syntax 702 includes a MVD header 704, such as a
5 mvd_vui_parameters_extension element. The MVD header 704 is a descriptor for identifying the MVD VUI syntax 702 for HEVC. The MVD VUI syntax 702 is used to encode and decode the video bitstream 110 of FIG. 1.

The MVD VUI syntax 702 can include the operation count 144 of FIG. 1, such as a
10 vui_mvd_num_ops_minus1 element to identify the total number of operations in the video bitstream 110. The MVD VUI syntax 702 can include the operation identifier 142 of FIG. 1, such as the counter [i].

The MVD VUI syntax 702 can include the view count 148, such as a
15 vui_mvd_num_target_output_views_minus1 element to identify views in a multiview configuration. The MVD VUI syntax 702 can include the view identifier 146, such as a vui_mvd_view_id element.

It has been discovered that the MVD VUI syntax 702 provides increased
functionality and improved performance by enabling displaying the video stream 112 of
FIG. 1 in a multiview configuration having more than one view displayed simultaneously.
By identifying the view identifier 146 for a view in a multiview configuration of the view
20 count 148 views, the MVD VUI syntax 702 enables multiview functionality with reduced overhead.

It has been discovered that encoding and decoding the video content 108 using the
MVD VUI syntax 702 can reduce the size of the video bitstream 110 and reduce the need
for video buffering. Reducing the size of the video bitstream 110 increases functionality
25 and increases the performance of display of the video stream 112 of FIG. 1.

Referring now to FIG. 8, therein is shown an example of a MVD VUI syntax
extension 802. The MVD VUI syntax extension 802 is a combination of Advanced Video
Coding, Scalable Video Coding, Multiview Video Coding, and Multiview Video plus Depth
elements.

30 The MVD VUI syntax extension 802 includes elements as described in the MVD
VUI syntax extension table of FIG. 8. The elements of the MVD VUI syntax extension
802 are arranged in a hierarchical structure as described in the MVD VUI syntax extension
table of FIG. 8.

The MVD VUI syntax extension 802 includes a MVD extension header 804, such as a vui_parameters element. The MVD extension header 804 is a descriptor for identifying the MVD VUI syntax extension 802 for HEVC. The MVD VUI syntax extension 802 is used to encode and decode the video bitstream 110 of FIG. 1 for AVC, SVC, MVC, and MVD video.

The MVD VUI syntax extension 802 can include the type indicator 406 of FIG. 4, such as a svc_mvc_flag, element, for identifying the type of coding used for the video bitstream 110. For example, the type indicator 406 can represent the type of coding using a value of 0 to indicate AVC, 1 to indicate SVC, 2 to indicate MVC, and 3 to indicate MVD.

The MVD VUI syntax extension 802 can include the iteration identifier 138 of FIG. 1. The MVD VUI syntax extension 802 can include the iteration count 140 of FIG. 1, such as num_iterations_minus1 element, for identifying the number of iterations associated with the video bitstream 110. The num_iterations_minus1 element can be a replacement for other elements in other coding syntaxes, such as the vui_ext_num_entries_minus1 for SVC, the vui_mvc_num_ops_minus1 for MVC, and the vui_mvd_num_ops_minus1 for MVD.

The iteration count 140 can encode the number of iterations minus 1 to map the range of iterations from 0 to the number of iterations minus 1. For example, for MVD video, the iteration count 140 indicates multiple operation points for multi-view and depth video.

The MVD VUI syntax extension 802 can include the view count 148 of FIG. 1, such as a num_target_output_views_minus1 element, to identify views per iteration in the multiview configuration. The MVD VUI syntax extension 802 can include the view identifier 146 of FIG. 1, such as a view_id element, for identifying each view in the multiview video information.

It has been discovered that encoding and decoding the video bitstream 110 using the MVD VUI syntax extension 802 increases video display quality, scalability, and reliability. Identifying and linking multiple images from multiple views using the temporal_id, dependency_id, and quality_id defines the relationship between images to increase the quality of video display.

It has been discovered that encoding and decoding the video content 108 of FIG. 1 using the MVC VUI syntax extension 602 of FIG. 6 can reduce the size of the video bitstream 110 and reduce the need for video buffering. Reducing the size of the video

bitstream 110 increases functionality and increases the performance of display of the video stream 112 of FIG. 1.

Referring now to FIG. 9, therein is shown an example of a Stereoscopic Video (SSV) VUI syntax extension 902. The SSV VUI syntax extension 902 is a combination of
5 Advanced Video Coding, Scalable Video Coding, Multiview Video Coding, and Stereoscopic Video elements. The SSV VUI syntax extension 902 can be used to encode and decode left and right stereoscopic view video.

The SSV VUI syntax extension 902 includes elements as described in the SSV VUI syntax extension table of FIG. 9. The elements of the SSV VUI syntax extension 902 are
10 arranged in a hierarchical structure as described in the SSV VUI syntax extension table of FIG. 9.

The SSV VUI syntax extension 902 includes a SSV extension header 904, such as a vui_parameters element. The SSV extension header 904 is a descriptor for identifying the SSV VUI syntax extension 902 for HEVC. The SSV VUI syntax extension 902 is used to
15 encode and decode the video bitstream 110 of FIG. 1 for SSV video.

The SSV VUI syntax extension 902 can include the type indicator 406 of FIG. 4, such as a svc_mvc_flag element, for identifying the type of coding used for the video bitstream 110. For example, the type indicator 406 can represent the type of coding using a value of 0 to indicate AVC and a value of 1 to indicate SSV.

The SSV VUI syntax extension 902 can include a first context indicator 906, such as a param_one_id element, and a second context indicator 908, such as a param_two_id element. The terms first and second are used to differentiate between context indicators and do not imply any ordering, ranking, importance, or other property.
20

The first context indicator 906 can include different information depending on the type of video coding being performed. For example, the param_one_id element can
25 represent a dependency_id element for SVC and a left_view_id for SSV.

The second context indicator 908 can include different types of information depending on the type of video coding being performed. For example, the param_two_id element can represent a quality_id element for SVC and a right_view_id for SSV.

It has been discovered that encoding and decoding the video bitstream 110 using the SSV VUI syntax extension 902 increases video display quality, scalability, and reliability for stereoscopic video. Identifying scalability factors for stereoscopic video using the first
30

context indicator 906 and the second context indicator 908 increases the quality of the video bitstream 110.

It has been discovered that encoding and decoding the video content 108 of FIG. 1 using the SSV VUI syntax extension 902 can reduce the size of the video bitstream 110 and reduce the need for video buffering. Reducing the size of the video bitstream 110 increases functionality and increases the performance of display of the video stream 112 of FIG. 1.

Referring now to FIG. 10, therein is shown a functional block diagram of the video coding system 100. The video coding system 100 can include the first device 102, the second device 104 and the communication path 106.

The first device 102 can communicate with the second device 104 over the communication path 106. The first device 102 can send information in a first device transmission 1032 over the communication path 106 to the second device 104. The second device 104 can send information in a second device transmission 1034 over the communication path 106 to the first device 102.

For illustrative purposes, the video coding system 100 is shown with the first device 102 as a client device, although it is understood that the video coding system 100 can have the first device 102 as a different type of device. For example, the first device can be a server. In a further example, the first device 102 can be the video encoder 102, the video decoder 104, or a combination thereof.

Also for illustrative purposes, the video coding system 100 is shown with the second device 104 as a server, although it is understood that the video coding system 100 can have the second device 104 as a different type of device. For example, the second device 104 can be a client device. In a further example, the second device 104 can be the video encoder 102, the video decoder 104, or a combination thereof.

For brevity of description in this embodiment of the present invention, the first device 102 will be described as a client device, such as a video camera, smart phone, or a combination thereof. The present invention is not limited to this selection for the type of devices. The selection is an example of the present invention.

The first device 102 can include a first control unit 1008. The first control unit 1008 can include a first control interface 1014. The first control unit 1008 can execute a first software 1012 to provide the intelligence of the video coding system 100.

The first control unit 1008 can be implemented in a number of different manners. For example, the first control unit 1008 can be a processor, an embedded processor, a

microprocessor, a hardware control logic, a hardware finite state machine (FSM), a digital signal processor (DSP), or a combination thereof.

The first control interface 1014 can be used for communication between the first control unit 1008 and other functional units in the first device 102. The first control
5 interface 1014 can also be used for communication that is external to the first device 102.

The first control interface 1014 can receive information from the other functional units or from external sources, or can transmit information to the other functional units or to external destinations. The external sources and the external destinations refer to sources and destinations external to the first device 102.

10 The first control interface 1014 can be implemented in different ways and can include different implementations depending on which functional units or external units are being interfaced with the first control interface 1014. For example, the first control interface 1014 can be implemented with electrical circuitry, microelectromechanical systems (MEMS), optical circuitry, wireless circuitry, wireline circuitry, or a combination
15 thereof.

The first device 102 can include a first storage unit 1004. The first storage unit 1004 can store the first software 1012. The first storage unit 1004 can also store the relevant information, such as images, syntax information, video, maps, profiles, display preferences, sensor data, or any combination thereof.

20 The first storage unit 1004 can be a volatile memory, a nonvolatile memory, an internal memory, an external memory, or a combination thereof. For example, the first storage unit 1004 can be a nonvolatile storage such as non-volatile random access memory (NVRAM), Flash memory, disk storage, or a volatile storage such as static random access memory (SRAM).

25 The first storage unit 1004 can include a first storage interface 1018. The first storage interface 1018 can be used for communication between the first storage unit 1004 and other functional units in the first device 102. The first storage interface 1018 can also be used for communication that is external to the first device 102.

The first device 102 can include a first imaging unit 1006. The first imaging unit
30 1006 can capture the video content 108 from the real world. The first imaging unit 1006 can include a digital camera, an video camera, an optical sensor, or any combination thereof.

The first imaging unit 1006 can include a first imaging interface 1016. The first imaging interface 1016 can be used for communication between the first imaging unit 1006 and other functional units in the first device 102.

The first imaging interface 1016 can receive information from the other functional units or from external sources, or can transmit information to the other functional units or to external destinations. The external sources and the external destinations refer to sources and destinations external to the first device 102.

The first imaging interface 1016 can include different implementations depending on which functional units or external units are being interfaced with the first imaging unit 1006. The first imaging interface 1016 can be implemented with technologies and techniques similar to the implementation of the first control interface 1014.

The first storage interface 1018 can receive information from the other functional units or from external sources, or can transmit information to the other functional units or to external destinations. The external sources and the external destinations refer to sources and destinations external to the first device 102.

The first storage interface 1018 can include different implementations depending on which functional units or external units are being interfaced with the first storage unit 1004. The first storage interface 1018 can be implemented with technologies and techniques similar to the implementation of the first control interface 1014.

The first device 102 can include a first communication unit 1010. The first communication unit 1010 can be for enabling external communication to and from the first device 102. For example, the first communication unit 1010 can permit the first device 102 to communicate with the second device 104, an attachment, such as a peripheral device or a computer desktop, and the communication path 106.

The first communication unit 1010 can also function as a communication hub allowing the first device 102 to function as part of the communication path 106 and not limited to be an end point or terminal unit to the communication path 106. The first communication unit 1010 can include active and passive components, such as microelectronics or an antenna, for interaction with the communication path 106.

The first communication unit 1010 can include a first communication interface 1020. The first communication interface 1020 can be used for communication between the first communication unit 1010 and other functional units in the first device 102. The first

communication interface 1020 can receive information from the other functional units or can transmit information to the other functional units.

The first communication interface 1020 can include different implementations depending on which functional units are being interfaced with the first communication unit 1010. The first communication interface 1020 can be implemented with technologies and techniques similar to the implementation of the first control interface 1014.

The first device 102 can include a first user interface 1002. The first user interface 1002 allows a user (not shown) to interface and interact with the first device 102. The first user interface 1002 can include a first user input (not shown). The first user input can include touch screen, gestures, motion detection, buttons, sliders, knobs, virtual buttons, voice recognition controls, or any combination thereof.

The first user interface 1002 can include the first display interface 120. The first display interface 120 can allow the user to interact with the first user interface 1002. The first display interface 120 can include a display, a video screen, a speaker, or any combination thereof.

The first control unit 1008 can operate with the first user interface 1002 to display video information generated by the video coding system 100 on the first display interface 120. The first control unit 1008 can also execute the first software 1012 for the other functions of the video coding system 100, including receiving video information from the first storage unit 1004 for display on the first display interface 120. The first control unit 1008 can further execute the first software 1012 for interaction with the communication path 106 via the first communication unit 1010.

For illustrative purposes, the first device 102 can be partitioned having the first user interface 1002, the first storage unit 1004, the first control unit 1008, and the first communication unit 1010, although it is understood that the first device 102 can have a different partition. For example, the first software 1012 can be partitioned differently such that some or all of its function can be in the first control unit 1008 and the first communication unit 1010. Also, the first device 102 can include other functional units not shown in FIG. 10 for clarity.

The video coding system 100 can include the second device 104. The second device 104 can be optimized for implementing the present invention in a multiple device embodiment with the first device 102. The second device 104 can provide the additional or higher performance processing power compared to the first device 102.

The second device 104 can include a second control unit 1048. The second control unit 1048 can include a second control interface 1054. The second control unit 1048 can execute a second software 1052 to provide the intelligence of the video coding system 100.

The second control unit 1048 can be implemented in a number of different manners. For example, the second control unit 1048 can be a processor, an embedded processor, a microprocessor, a hardware control logic, a hardware finite state machine (FSM), a digital signal processor (DSP), or a combination thereof.

The second control interface 1054 can be used for communication between the second control unit 1048 and other functional units in the second device 104. The second control interface 1054 can also be used for communication that is external to the second device 104.

The second control interface 1054 can receive information from the other functional units or from external sources, or can transmit information to the other functional units or to external destinations. The external sources and the external destinations refer to sources and destinations external to the second device 104.

The second control interface 1054 can be implemented in different ways and can include different implementations depending on which functional units or external units are being interfaced with the second control interface 1054. For example, the second control interface 1054 can be implemented with electrical circuitry, microelectromechanical systems (MEMS), optical circuitry, wireless circuitry, wireline circuitry, or a combination thereof.

The second device 104 can include a second storage unit 1044. The second storage unit 1044 can store the second software 1052. The second storage unit 1044 can also store the relevant information, such as images, syntax information, video, maps, profiles, display preferences, sensor data, or any combination thereof.

The second storage unit 1044 can be a volatile memory, a nonvolatile memory, an internal memory, an external memory, or a combination thereof. For example, the second storage unit 1044 can be a nonvolatile storage such as non-volatile random access memory (NVRAM), Flash memory, disk storage, or a volatile storage such as static random access memory (SRAM).

The second storage unit 1044 can include a second storage interface 1058. The second storage interface 1058 can be used for communication between the second storage

unit 1044 and other functional units in the second device 104. The second storage interface 1058 can also be used for communication that is external to the second device 104.

The second storage interface 1058 can receive information from the other functional units or from external sources, or can transmit information to the other functional units or to external destinations. The external sources and the external destinations refer to sources and destinations external to the second device 104.

The second storage interface 1058 can include different implementations depending on which functional units or external units are being interfaced with the second storage unit 1044. The second storage interface 1058 can be implemented with technologies and techniques similar to the implementation of the second control interface 1054.

The second device 104 can include a second imaging unit 1046. The second imaging unit 1046 can capture the video content 108 of FIG. 1 from the real world. The first imaging unit 1006 can include a digital camera, an video camera, an optical sensor, or any combination thereof.

The second imaging unit 1046 can include a second imaging interface 1056. The second imaging interface 1056 can be used for communication between the second imaging unit 1046 and other functional units in the second device 104.

The second imaging interface 1056 can receive information from the other functional units or from external sources, or can transmit information to the other functional units or to external destinations. The external sources and the external destinations refer to sources and destinations external to the second device 104.

The second imaging interface 1056 can include different implementations depending on which functional units or external units are being interfaced with the second imaging unit 1046. The second imaging interface 1056 can be implemented with technologies and techniques similar to the implementation of the first control interface 1014.

The second device 104 can include a second communication unit 1050. The second communication unit 1050 can enable external communication to and from the second device 104. For example, the second communication unit 1050 can permit the second device 104 to communicate with the first device 102, an attachment, such as a peripheral device or a computer desktop, and the communication path 106.

The second communication unit 1050 can also function as a communication hub allowing the second device 104 to function as part of the communication path 106 and not limited to be an end point or terminal unit to the communication path 106. The second

communication unit 1050 can include active and passive components, such as microelectronics or an antenna, for interaction with the communication path 106.

The second communication unit 1050 can include a second communication interface 1060. The second communication interface 1060 can be used for communication between the second communication unit 1050 and other functional units in the second device 104. The second communication interface 1060 can receive information from the other functional units or can transmit information to the other functional units.

The second communication interface 1060 can include different implementations depending on which functional units are being interfaced with the second communication unit 1050. The second communication interface 1060 can be implemented with technologies and techniques similar to the implementation of the second control interface 1054.

The second device 104 can include a second user interface 1042. The second user interface 1042 allows a user (not shown) to interface and interact with the second device 104. The second user interface 1042 can include a second user input (not shown). The second user input can include touch screen, gestures, motion detection, buttons, sliders, knobs, virtual buttons, voice recognition controls, or any combination thereof.

The second user interface 1042 can include a second display interface 1043. The second display interface 1043 can allow the user to interact with the second user interface 1042. The second display interface 1043 can include a display, a video screen, a speaker, or any combination thereof.

The second control unit 1048 can operate with the second user interface 1042 to display information generated by the video coding system 100 on the second display interface 1043. The second control unit 1048 can also execute the second software 1052 for the other functions of the video coding system 100, including receiving display information from the second storage unit 1044 for display on the second display interface 1043. The second control unit 1048 can further execute the second software 1052 for interaction with the communication path 106 via the second communication unit 1050.

For illustrative purposes, the second device 104 can be partitioned having the second user interface 1042, the second storage unit 1044, the second control unit 1048, and the second communication unit 1050, although it is understood that the second device 104 can have a different partition. For example, the second software 1052 can be partitioned differently such that some or all of its function can be in the second control unit 1048 and

the second communication unit 1050. Also, the second device 104 can include other functional units not shown in FIG. 10 for clarity.

The first communication unit 1010 can couple with the communication path 106 to send information to the second device 104 in the first device transmission 1032. The second device 104 can receive information in the second communication unit 1050 from the first device transmission 1032 of the communication path 106.

The second communication unit 1050 can couple with the communication path 106 to send video information to the first device 102 in the second device transmission 1034. The first device 102 can receive video information in the first communication unit 1010 from the second device transmission 1034 of the communication path 106. The video coding system 100 can be executed by the first control unit 1008, the second control unit 1048, or a combination thereof.

The functional units in the first device 102 can work individually and independently of the other functional units. For illustrative purposes, the video coding system 100 is described by operation of the first device 102. It is understood that the first device 102 can operate any of the modules and functions of the video coding system 100. For example, the first device 102 can be described to operate the first control unit 1008.

The functional units in the second device 104 can work individually and independently of the other functional units. For illustrative purposes, the video coding system 100 can be described by operation of the second device 104. It is understood that the second device 104 can operate any of the modules and functions of the video coding system 100. For example, the second device 104 is described to operate the second control unit 1048.

For illustrative purposes, the video coding system 100 is described by operation of the first device 102 and the second device 104. It is understood that the first device 102 and the second device 104 can operate any of the modules and functions of the video coding system 100. For example, the first device 102 is described to operate the first control unit 1008, although it is understood that the second device 104 can also operate the first control unit 1008.

Referring now to FIG. 11, therein is shown a control flow 1100 of the video coding system 100 of FIG. 1. The control flow 1100 describes decoding the video bitstream 110 of FIG. 1 by receiving the video bitstream 110, extracting the video syntax 114 of FIG. 1, decoding the video bitstream 110, and displaying the video stream 112 of FIG. 1.

The video coding system 100 can include a receive module 1102. The receive module 1102 can receive the video bitstream 110 encoded by the video encoder 102 of FIG. 1.

The video bitstream 110 can be received in a variety of ways. For example, the video bitstream 110 can be received from the video encoder 102 of FIG. 1, as a pre-encoded video file (not shown), in a digital message (not shown) over the communication path 106 of FIG. 1, or a combination thereof.

The video coding system 100 can include a get type module 1104. The get type module 1104 can identify the type of video coding used to encode and decode the video bitstream 110 by extracting the syntax type 132 of FIG. 1.

The get type module 1104 can detect the syntax type 132 in a variety of ways. The get type module 1104 can determine the syntax type 132 by parsing the type indicator 406 of FIG. 4, such as the `svc_mvc_flag` element, from the video bitstream 110. In another example, the get type module 1104 can extract the syntax type 132 from the video syntax 114 extracting the type indicator 406 from the video bitstream 110 using a demultiplexer (not shown) to separate the video syntax 114 from the video image data of the video bitstream 110.

In an illustrative example, if the `svc_mvc_flag` has a value of 0, then the type indicator 406 is set to AVC. If the `svc_mvc_flag` has a value of 1, then the type indicator 406 is set to SVC. If the `svc_mvc_flag` element has a value of 2, then the type indicator 406 is set to MVC.

If the `svc_mvc_flag` element has a value of 3, then the type indicator 406 is set to MVD. If the `svc_mvc_flag` element has a value of 4, then the type indicator 406 is set to SSV. The syntax type 132 is assigned the value of the type indicator 406 extracted from the video bitstream 110.

The video coding system 100 can include a get syntax module 1106. The get syntax module 1106 can identify and extract the video syntax 114 embedded within the video bitstream 110.

For example, the video syntax 114 can be extracted by searching the video bitstream 110 for video usability information headers indicating the presence of the video syntax 114. In yet another example, the video syntax 114 can be extracted from the video bitstream 110 using a demultiplexer (not shown) to separate the video syntax 114 from the video image data of the video bitstream 110. In yet another example, the video syntax 114 can be

extracted from the video bitstream 110 by extracting a sequence parameter set Raw Byte Sequence Payload (RBSP) syntax. The sequence parameter set RBSP is a syntax structure containing an integer number of bytes encapsulated in a network abstraction layer unit. The RBSP can be either empty or have the form of a string of data bits containing syntax elements followed by a RBSP stop bit and followed by zero or more additional bits equal to 0.

In another example, if the video bitstream 110 is received in a file, then the video syntax 114 can be detected by examining the file extension of the file containing the video bitstream 110. In yet another example, if the video bitstream 110 is received as a digital message over the communication path 106 of FIG. 1, then the video syntax 114 can be provided as a portion of the structure of the digital message.

The get syntax module 1106 can extract the individual elements of the video syntax 114 based on the syntax type 132. The get syntax module 1106 can include an AVC module 1108, a SVC module 1110, a MVC module 1112, a MVD module 1114, and a SSV module 1116 to extract the elements of the video syntax 114 based on the syntax type 132.

If the syntax type 132 indicates AVC coding, then the control flow can pass to the AVC module 1108. The AVC module 1108 can extract the AVC VUI syntax 202 of FIG. 2 from the video syntax 114. The elements of the AVC VUI syntax 202 can be extracted from the video syntax 114 according to the definition of the elements of the AVC VUI syntax 202 in the table of FIG. 2.

It has been discovered that using the AVC VUI syntax 202 increases reliability and reduces overhead by encoding and decoding the video content 108 of FIG. 1 according to the reduced data footprint of the video usability information of the AVC VUI syntax 202. Reducing the amount of data required to define the video bitstream 110 increases reliability and reduces data overhead.

If the syntax type 132 indicates SVC coding, then the control flow can pass to the SVC module 1110. The SVC module 1110 can extract the SVC VUI syntax extension 402 of FIG. 4 from the video syntax 114. The elements of the SVC VUI syntax extension 402 can be extracted from the video syntax 114 according to the definition of the elements of the SVC VUI syntax extension 402 in the table of FIG. 4.

It has been discovered that using the SVC VUI syntax extension 402 increases reliability and reduces overhead by encoding and decoding the video content 108 according to the reduced data footprint of the video usability information of the SVC VUI syntax

extension 402. Reducing the amount of data required to define the video bitstream 110 increases reliability and reduces data overhead.

If the syntax type 132 indicates MVC coding, then the control flow can pass to the MVC module 1112. The MVC module 1112 can extract the MVC VUI syntax extension 602 of FIG. 6 from the video syntax 114. The elements of the MVC VUI syntax extension 602 can be extracted from the video syntax 114 according to the definition of the elements of the MVC VUI syntax extension 602 in the table of FIG. 6.

It has been discovered that using the MVC VUI syntax extension 602 increases reliability and reduces overhead by encoding and decoding the video content 108 according to the reduced data footprint of the video usability information of the MVC VUI syntax extension 602. Reducing the amount of data required to define the video bitstream 110 increases reliability and reduces data overhead for multiview video plus depth coding.

If the syntax type 132 indicates MVD coding, then the control flow can pass to the MVD module 1114. The MVD module 1114 can extract the MVD VUI syntax extension 802 of FIG. 8 from the video syntax 114. The elements of the MVD VUI syntax extension 802 can be extracted from the video syntax 114 according to the definition of the elements of the MVD VUI syntax 802 in the table of FIG. 8.

It has been discovered that using the MVD VUI syntax extension 802 increases reliability and reduces overhead by encoding and decoding the video content 108 according to the reduced data footprint of the video usability information of the MVD VUI syntax extension 802. Reducing the amount of data required to define the video bitstream 110 increases reliability and reduces data overhead for MVD coding.

If the syntax type 132 indicates SSV coding, then the control flow can pass to the SSV module 1116. The SSV module 1116 can extract the SSV VUI syntax extension 902 of FIG. 9 from the video syntax 114. The elements of the SSV VUI syntax extension 902 can be extracted from the video syntax 114 according to the definition of the elements of the SSV VUI syntax extension 902 in the table of FIG. 9.

It has been discovered that using the SSV VUI syntax extension 902 increases reliability and reduces overhead by encoding and decoding the video content 108 according to the reduced data footprint of the video usability information of the SSV VUI syntax extension 902. Reducing the amount of data required to define the video bitstream 110 increases reliability and reduces data overhead for stereoscopic video.

The video coding system 100 can include a decode module 1118. The decode module 1118 can decode the video bitstream 110 using the elements of the video syntax 114 for the extracted instance of the syntax type 132 to form the video stream 112.

The decode module 1118 can decode the video bitstream 110 using the syntax type 132 to determine the type of video coding used to form the video bitstream 110. If the syntax type 132 indicates advanced video coding, then the decode module 1118 can decode the video bitstream 110 using the AVC VUI syntax 202.

If the syntax type 132 indicates scalable video coding, then the decode module 1118 can decode the video bitstream 110 using the SVC VUI syntax extension 402. The SVC VUI syntax extension 402 can include an array of scalability elements having an array size as indicated by the entry count 136. For example, the SVC VUI syntax extension 402 can include an array of `temporal_id[i]`, `dependency_id[i]`, and `quality_id[i]` where `[i]` has a maximum value of the entry count 136.

If the syntax type 132 indicates multiview video coding, then the decode module 1118 can decode the video bitstream 110 using the MVC VUI syntax extension 602. If the syntax type 132 indicates MVC, then the MVC VUI syntax extension 602 can include an array of the `view_id[i][j]`, where `[i]` has a maximum value of the entry count 136 and `[j]` has a maximum value of the view count 148 of FIG. 1.

If the syntax type 132 indicates multiview video coding plus depth, then the decode module 1118 can decode the video bitstream 110 using the MVD VUI syntax extension 802. If the syntax type 132 indicates multiview video coding plus depth, then the decode module 1118 can decode the video bitstream 110 using the MVD VUI syntax extension 802. If the syntax type 132 indicates MVD, then the MVD VUI syntax extension 802 can include an array of the `view_id[i][j]`, where `[i]` has a maximum value of the entry count 136 and `[j]` has a maximum value of the view count 148.

If the syntax type 132 indicates SSV coding, then the decode module 1118 can decode the video bitstream 110 using the SSV VUI syntax extension 902. The SSV VUI syntax extension 902 can include an array of scalability elements having an array size as indicated by the entry count 136. For example, the SSV VUI syntax extension 902 can include an array of `temporal_id[i]`, `param_one_id[i]`, and `param_two_id[i]` where `[i]` has a maximum value of the entry count 136.

The video coding system 100 can include a display module 1120. The display module 1120 can receive the video stream 112 from the decode module 1118 and display on the display interface 120 of FIG. 1.

5 The physical transformation from the optical images of physical objects of the video content 108 to displaying the video stream 112 on the pixel elements of the display interface 120 of the video decoder 104 of FIG. 1 results in physical changes to the pixel elements of the display interface 120 in the physical world, such as the change of electrical state the pixel element, is based on the operation of the video coding system 100. As the changes in the physical world occurs, such as the motion of the objects captured in the video content
10 108, the movement itself creates additional information, such as the updates to the video content 108, that are converted back into changes in the pixel elements of the display interface 120 for continued operation of the video coding system 100.

The first software 1012 of FIG. 10 of the first device 102 can include the video coding system 100. For example, the first software 1012 can include the receive module
15 1102, the get type module 1104, the get syntax module 1106, the decode module 1118, and the display module 1120.

The first control unit 1008 of FIG. 10 can execute the first software 1012 for the receive module 1102 to receive the video bitstream 110. The first control unit 1008 can execute the first software 1012 for the get type module 1104 to determine the syntax type
20 132 for the video bitstream 110. The first control unit 1008 can execute the first software 1012 for the get syntax module 1106 to identify and extract the video syntax 114 from the video bitstream 110. The first control unit 1008 can execute the first software 1012 for the decode module 1118 to form the video stream 112. The first control unit 1008 can execute the first software 1012 for the display module 1120 to display the video stream 112.

25 The second software 1052 of FIG. 10 of the second device 104 can include the video coding system 100. For example, the second software 1052 can include the receive module 1102, the get type module 1104, the get syntax module 1106, and the decode module 1118.

The second control unit 1048 of FIG. 10 can execute the second software 1052 for the receive module 1102 to receive the video bitstream 110. The second control unit 1048
30 can execute the second software 1052 for the get type module 1104 to determine the syntax type 132 for the video bitstream 110. The second control unit 1048 can execute the second software 1052 for the get syntax module 1106 to identify and extract the video syntax 114 from the video bitstream 110. The second control unit 1048 can execute the second

software 1052 for the decode module 1118 to form the video stream 112 of FIG. 1. The second control unit 1048 can execute the second software for the display module 1120 to display the video stream 112.

The video coding system 100 can be partitioned between the first software 1012 and the second software 1052. For example, the second software 1052 can include the get syntax module 1106, the decode module 1118, and the display module 1120. The second control unit 1048 can execute modules partitioned on the second software 1052 as previously described.

The first software 1012 can include the receive module 1102 and the get type module 1104. Depending on the size of the first storage unit 1004 of FIG. 10, the first software 1012 can include additional modules of the video coding system 100. The first control unit 1008 can execute the modules partitioned on the first software 1012 as previously described.

The first control unit 1008 can operate the first communication unit 1010 of FIG. 10 to send the video bitstream 110 to the second device 104. The first control unit 1008 can operate the first software 1012 to operate the first imaging unit 1006 of FIG. 10. The second communication unit 1050 of FIG. 10 can send the video stream 112 to the first device 102 over the communication path 106.

The video coding system 100 describes the module functions or order as an example. The modules can be partitioned differently. For example, the get type module 1104, the get syntax module 1106, and the decode module 1118 can be combined. Each of the modules can operate individually and independently of the other modules.

Furthermore, data generated in one module can be used by another module without being directly coupled to each other. For example, the get syntax module 1106 can receive the video bitstream 110 from the receive module 1102.

The modules can be implemented in a variety of ways. The receive module 1102, the get type module 1104, the get syntax module 1106, the decode module 1118, and the display module 1120 can be implemented in as hardware accelerators (not shown) within the first control unit 1008 or the second control unit 1048, or can be implemented in as hardware accelerators (not shown) in the first device 102 or the second device 104 outside of the first control unit 1008 or the second control unit 1048.

Referring now to FIG. 12, therein is shown a flow chart of a method 1200 of operation of the video coding system 100 of FIG. 1 in a further embodiment of the present

invention. The method 1200 includes: receiving a video bitstream in a block 1202; identifying a syntax type of the video bitstream in a block 1204; extracting a video syntax from the video bitstream for the syntax type in a block 1206; and forming a video stream based on the video syntax for displaying on a device in a block 1208.

5 It has been discovered that the present invention thus has numerous aspects. The present invention valuably supports and services the historical trend of reducing costs, simplifying systems, and increasing performance. These and other valuable aspects of the present invention consequently further the state of the technology to at least the next level.

Thus, it has been discovered that the video coding system of the present invention
10 furnishes important and heretofore unknown and unavailable solutions, capabilities, and functional aspects for efficiently coding and decoding video content for high definition applications. The resulting processes and configurations are straightforward, cost-effective, uncomplicated, highly versatile and effective, can be surprisingly and unobviously implemented by adapting known technologies, and are thus readily suited for efficiently and
15 economically manufacturing video coding devices fully compatible with conventional manufacturing processes and technologies. The resulting processes and configurations are straightforward, cost-effective, uncomplicated, highly versatile, accurate, sensitive, and effective, and can be implemented by adapting known components for ready, efficient, and economical manufacturing, application, and utilization.

20 While the invention has been described in conjunction with a specific best mode, it is to be understood that many alternatives, modifications, and variations will be apparent to those skilled in the art in light of the foregoing description. Accordingly, it is intended to embrace all such alternatives, modifications, and variations that fall within the scope of the included claims. All matters hithertofore set forth herein or shown in the accompanying
25 drawings are to be interpreted in an illustrative and non-limiting sense.

CLAIM

What is claimed is:

1. A method (1200) of operation of a video coding system (100) comprising:
receiving a video bitstream (110);
5 identifying a syntax type (132) of the video bitstream (110);
extracting a video syntax (114) from the video bitstream (110) for the syntax type
(132); and
forming a video stream (112) based on the video syntax (114) for displaying on a
device (102).
- 10 2. The method (1200) as claimed in claim 1 wherein extracting the video syntax
(114) includes identifying the syntax type (132) for the video bitstream (110) for a scalable
video coding video usability information syntax extension (402).
3. The method (1200) as claimed in claim 1 wherein extracting the video syntax
(114) includes identifying the syntax type (132) for the video bitstream (110) for a
15 multiview video coding video usability information syntax extension (602).
4. The method (1200) as claimed in claim 1 wherein extracting the video syntax
(114) includes identifying the syntax type (132) for the video bitstream (110) for an
multiview video plus depth video usability information syntax extension (802).
5. The method (1200) as claimed in claim 1 wherein extracting the video syntax
20 (114) includes identifying the syntax type (132) for the video bitstream (110) for a
stereoscopic video usability information syntax extension (902).
6. A video coding system (100) comprising:
a receive module (1102) for receiving a video bitstream (110);
a get type module (1104), coupled to the receive module (1102), for identifying a
25 syntax type (132) from the video bitstream (110);
a get syntax module (1106), coupled to the get type module (1104), for extracting a
video syntax (114) from the video bitstream (110) for the syntax type (132);
and
a decode module (1118), coupled to the get syntax module (1106), for forming a
30 video stream (112) based on the video syntax (114) and the video bitstream
(110) for displaying on a device (102).

7. The system (100) as claimed in claim 6 wherein the decode module (1118) is for identifying the syntax type (132) for the video bitstream (110) for a scalable video coding video usability information syntax extension (402).

8. The system (100) as claimed in claim 6 wherein the decode module (1118) is
5 for identifying the syntax type (132) for the video bitstream (110) for a multiview video coding video usability information syntax extension (602).

9. The system (100) as claimed in claim 6 wherein the decode module (1118) is for identifying the syntax type (132) for the video bitstream (110) for a multiview video plus depth video usability information syntax extension (802).

10. The system (100) as claimed in claim 6 wherein the decode module (1118) is
10 for identifying the syntax type (132) for the video bitstream (110) for a stereoscopic video usability information syntax extension (902).

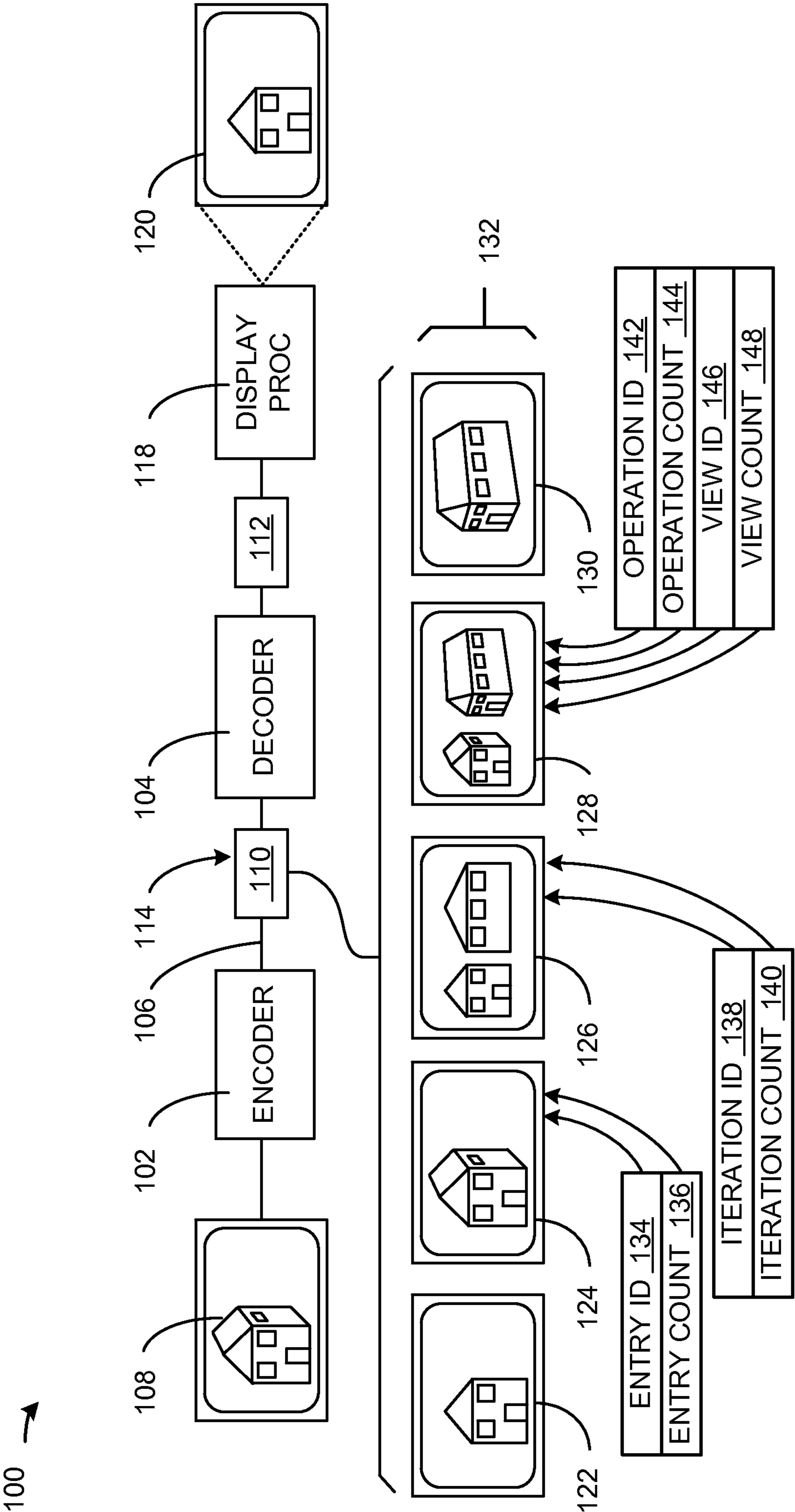


FIG. 1

2/12

202 →
204 →

vui_parameters() { // start of HEVC VUI syntaxes	Desc.
aspect_ratio_info_present_flag	u(1)
if(aspect_ratio_info_present_flag) {	
aspect_ratio_idc	u(8)
if(aspect_ratio_idc == Extended_SAR) {	
sar_width	u(16)
sar_height } }	u(16)
overscan_info_present_flag	u(1)
if(overscan_info_present_flag)	
overscan_appropriate_flag	u(1)
video_signal_type_present_flag	u(1)
if(video_signal_type_present_flag) {	
video_format	u(3)
video_full_range_flag	u(1)
colour_description_present_flag	u(1)
if(colour_description_present_flag) {	
colour_primaries	u(8)
transfer_characteristics	u(8)
matrix_coefficients } }	u(8)
chroma_loc_info_present_flag	u(1)
if(chroma_loc_info_present_flag) chroma_sample_loc_type	ue(v)
timing_info_present_flag	u(1)
if(timing_info_present_flag) {	
num_units_in_tick	u(32)
time_scale	u(32)
fixed_frame_rate_flag }	u(1)
nal_hrd_parameters_present_flag	u(1)
if(nal_hrd_parameters_present_flag) hrd_parameters()	
vcl_hrd_parameters_present_flag	u(1)
if(vcl_hrd_parameters_present_flag) hrd_parameters()	
if(nal_hrd_parameters_present_flag vcl_hrd_parameters_present_flag)	
low_delay_hrd_flag	u(1)
bitstream_restriction_flag	u(1)
if(bitstream_restriction_flag) {	
motion_vectors_over_pic_boundaries_flag	u(1)
max_bytes_per_pic_denom	ue(v)
max_bits_per_cu_denom	ue(v)
log2_max_mv_length_horizontal	ue(v)
log2_max_mv_length_vertical	ue(v)
num_reorder_frames	ue(v)
max_dec_frame_buffering } }	ue(v)

206 →

FIG. 2

3/12

302
304
136
134

svc_vui_parameters_extension() {	Descriptor
vui_ext_num_entries_minus1	ue(v)
for(i = 0; i <= vui_ext_num_entries_minus1; i++) {	
vui_ext_dependency_id[i]	u(3)
vui_ext_quality_id[i]	u(4)
vui_ext_temporal_id[i] // common to MVC VUI	u(3)
vui_ext_timing_info_present_flag[i]	u(1)
if(vui_ext_timing_info_present_flag[i]) {	
vui_ext_num_units_in_tick[i]	u(32)
vui_ext_time_scale[i]	u(32)
vui_ext_fixed_frame_rate_flag[i]	u(1)
}	
vui_ext_nal_hrd_parameters_present_flag[i]	u(1)
if(vui_ext_nal_hrd_parameters_present_flag[i])	
hrd_parameters()	
vui_ext_vcl_hrd_parameters_present_flag[i]	u(1)
if(vui_ext_vcl_hrd_parameters_present_flag[i])	
hrd_parameters()	
if(vui_ext_nal_hrd_parameters_present_flag[i] vui_ext_vcl_hrd_parameters_present_flag[i])	
vui_ext_low_delay_hrd_flag[i]	u(1)
}	
}	

FIG. 3

4/12

402			
404		vui_parameters() { // start of HEVC VUI syntaxes (AVC+ SVC)	Desc
		 // edited	
406		svc_mvc_flag // 0-AVC, 1- SVC, 2-MVC, 3- reserved	u(2)
		num_entries_minus1 // 0 (AVC); >=0 (SVC)	ue(v)
		for(i = 0; i <= num_entries_minus1; i++) { // i=0 (AVC); i>=0 (SVC)	
		if(svc_mvc_flag == 1) { // loop for SVC only	
410		temporal_id[i]	u(3)
412		dependency_id[i]	u(3)
414		quality_id[i]	u(4)
		}	
		timing_info_present_flag[i]	u(1)
		if(timing_info_present_flag[i]) {	
		num_units_in_tick[i]	u(32)
		time_scale[i]	u(32)
		fixed_frame_rate_flag[i]	u(1)
		}	
		nal_hrd_parameters_present_flag[i]	u(1)
		if(nal_hrd_parameters_present_flag[i])	
		hrd_parameters()	
		vcl_hrd_parameters_present_flag[i]	u(1)
		if(vcl_hrd_parameters_present_flag[i])	
		hrd_parameters()	
		if(nal_hrd_parameters_present_flag[i] vcl_hrd_parameters_present_flag[i])	
		low_delay_hrd_flag[i]	u(1)
		} // iterative loop ends for SVC video layer extensions	
		} // end of HEVC VUI syntaxes (AVC+SVC)	

FIG. 4

5/12

502 →

504 →

	Descriptor
mvc_vui_parameters_extension() {	
vui_mvc_num_ops_minus1	ue(v)
for(i = 0; i <= vui_mvc_num_ops_minus1; i++) {	
vui_mvc_temporal_id[i] // common with SVC VUI	u(3)
vui_mvc_num_target_output_views_minus1[i]	ue(v)
for(j = 0; j <= vui_mvc_num_target_output_views_minus1[i]; j++)	
vui_mvc_view_id[i][j]	ue(v)
vui_mvc_timing_info_present_flag[i]	u(1)
if(vui_mvc_timing_info_present_flag[i]) {	
vui_mvc_num_units_in_tick[i]	u(32)
vui_mvc_time_scale[i]	u(32)
vui_mvc_fixed_frame_rate_flag[i]	u(1)
}	
vui_mvc_nal_hrd_parameters_present_flag[i]	u(1)
if(vui_mvc_nal_hrd_parameters_present_flag[i])	
hrd_parameters()	
vui_mvc_vcl_hrd_parameters_present_flag[i]	u(1)
if(vui_mvc_vcl_hrd_parameters_present_flag[i])	
hrd_parameters()	
if(vui_mvc_nal_hrd_parameters_present_flag[i] vui_mvc_vcl_hrd_parameters_present_flag[i])	
vui_mvc_low_delay_hrd_flag[i]	u(1)
}	
}	

FIG. 5

6/12

602	→	vui_parameters() { // start of HEVC VUI syntaxes (AVC+ SVC+MVC)	Desc.
604	→	 // edited	
		svc_mvc_flag // 0-AVC, 1- SVC, 2-MVC, 3- reserved	u(2)
138	→	num_iterations_minus1	ue(v)
140	→	for(i = 0; i <= num_iterations_minus1; i++) { // start iterative loop	
		if(svc_mvc_flag > 0) { // for SVC_MVC only	
		temporal_id[i] // common syntax between SVC and MVC VUIs	u(3)
		if(svc_mvc_flag == 1) { // SVC only: svc_mvc_flag =1	
		dependency_id[i]	u(3)
		quality_id[i]	u(4)
		}	
		else { // MVC only: svc_mvc_flag =2	
		num_target_output_views_minus1[i]	ue(v)
		for(j = 0; j <= num_target_output_views_minus1[i]; j++)	
146	→	view_id[i][j]	ue(v)
		}	
		} // end SVC_MVC only	
		timing_info_present_flag[i]	u(1)
		if(timing_info_present_flag[i]) {	
		num_units_in_tick[i]	u(32)
		time_scale[i]	u(32)
		fixed_frame_rate_flag[i]	u(1)
		}	
		nal_hrd_parameters_present_flag[i]	u(1)
		if(nal_hrd_parameters_present_flag[i])	
		hrd_parameters()	
		vcl_hrd_parameters_present_flag[i]	u(1)
		if(vcl_hrd_parameters_present_flag[i])	
		hrd_parameters()	
		if(nal_hrd_parameters_present_flag[i] vcl_hrd_parameters_present_flag[i])	
		low_delay_hrd_flag[i]	u(1)
		} // end iterative loop	
		}	

FIG. 6

7/12

702 →

704	mvd_vui_parameters_extension() { Descriptor	Desc.
144	vui_mvd_num_ops_minus1	ue(v)
	for(i = 0; i <= vui_mvd_num_ops_minus1; i++) {	
	vui_mvd_temporal_id[i] // common with SVC VUI	u(3)
148	vui_mvd_num_target_output_views_minus1 i	ue(v)
	for(j = 0; j <= vui_mvd_num_target_output_views_minus1[i]; j++)	
146	vui_mvd_view_id[i][j]	ue(v)
	vui_mvd_timing_info_present_flag[i]	u(1)
	if(vui_mvd_timing_info_present_flag[i]) {	
	vui_mvd_num_units_in_tick[i]	u(32)
	vui_mvd_time_scale[i]	u(32)
	vui_mvd_fixed_frame_rate_flag[i]	u(1)
	}	
	vui_mvd_nal_hrd_parameters_present_flag[i]	u(1)
	if(vui_mvd_nal_hrd_parameters_present_flag[i])	
	Hrd_parameters()	
	vui_mvd_vcl_hrd_parameters_present_flag[i]	u(1)
	if(vui_mvd_vcl_hrd_parameters_present_flag[i])	
	Hrd_parameters()	
	if(vui_mvd_nal_hrd_parameters_present_flag[i] vui_mvd_vcl_hrd_parameters_present_flag[i])	
	vui_mvd_low_delay_hrd_flag[i]	u(1)
	}	
	video_src_info_present_flag	u(1)
	if(video_src_info_present_flag) {	
	pic_width_in_mbs_minus1	ue(v)
	pic_height_in_mbs_minus1	ue(v)
	aspect_ratio_info_present_flag	u(1)
	if (aspect_ratio_info_present_flag) {	
	aspect_ratio_idc	u(8)
	if (aspect_ratio_idc == Extended_SAR) {	
	sar_width	u(16)
	sar_height	u(16)
	}}}}	

FIG. 7

8/12

802 →

vui_parameters() { // start of HEVC VUI syntaxes (AVC+SVC+MVC+MVD)	Desc.
// edited	
svc_mvc_flag // O-AVC, 1- SVC, 2-MVC, 3-MVD	u(2)
num_iterations_minus1	ue(V)
for(i = 0; i <= num iterations minus1; i++) { // start iterative loop	
if(svc mvc flag > 0) { // for SVC MVC only	
temporal_id[i] // common syntax between SVC and MVC VUIs	u(3)
if(svc mvc flag == 1) { // SVC only: svc mvc flag =1	
dependency_id[i]	u(3)
quality_id[i]	u(4)
else { // MVC / MVD: svc_mvc_flag =2, 3	
num_target_output_views_minus1[i]	ue(V)
for(j = 0; j <= num target output views minus1[i]; j++)	
view_id[i][j]	ue(V)
}} // end SVC_MVC only	
timing_info_present_flag[i]	u(1)
if(timing info present flag[i]) {	
num_units_in_tick[i]	u(32)
time_scale[i]	u(32)
fixed_frame_rate_flag[i]	u(1)
nal_hrd_parameters_present_flag1 i l	u(1)
if(nal hrd parameters present flag[i])	
hrd_parameters()	
vc1_hrd_parameters_present_flag[i]	u(1)
if(vc1 hrd parameters present flag i)	
hrd_parameters()	
if(nal_hrd_parameters_present_flag[i] vc1_hrd_parameters_present_flag[i])	
low_delay_hrd_flag[i] // end iterative loop	u(1)
if(svc mvc flag == 3) { // MVD	
video_src_info_present_flag	u(1)
if(video_src_info_present_flag) {	
pic_width_in_mbs_minus1	ue(V)
pic_height_in_mbs_minus1	ue(V)
aspect_ratio_info_present_flag	u(1)
if (aspect_ratio_info_present_flag) {	
aspect_ratio_idc	u(8)
if (aspect_ratio_idc == Extended SAR) {	
sar_width	u(16)
sar_height	u(16)
}}}} // end of MVD	

FIG. 8

9/12

902 →

904 →

vui_parameters() { // start of HEVC VUI syntaxes (AVC+ SVC+SSV)	Desc.
..... // edited	
svc_mvc_flag // 0-AVC, 1- SVC/ SSV	u(1)
num_iterations_minus1	ue(v)
for(i = 0; i <= num_iterations_minus1; i++) { // start iterative loop	
if(svc_mvc_flag) { // for SVC_SSV only	
temporal_id[i]	u(3)
param_one_id[i] // dependency_id for SVC, left_view_id for SSV	u(3)/ue(v)
param_two_id[i] // quality_id for SVC, right_view_id for SSV	u(4)/ue(v)
} // end SVC_SSV only	
timing_info_present_flag[i]	u(1)
if(timing_info_present_flag[i]) {	
num_units_in_tick[i]	u(32)
time_scale[i]	u(32)
fixed_frame_rate_flag[i]	u(1)
}	
nal_hrd_parameters_present_flag[i]	u(1)
if(nal_hrd_parameters_present_flag[i])	
hrd_parameters()	
vcl_hrd_parameters_present_flag[i]	u(1)
if(vcl_hrd_parameters_present_flag[i])	
hrd_parameters()	
if(nal_hrd_parameters_present_flag[i] vcl_hrd_parameters_present_flag[i])	
low_delay_hrd_flag[i]	u(1)
} // end iterative loop	
}	

906 →

908 →

FIG. 9

100 →

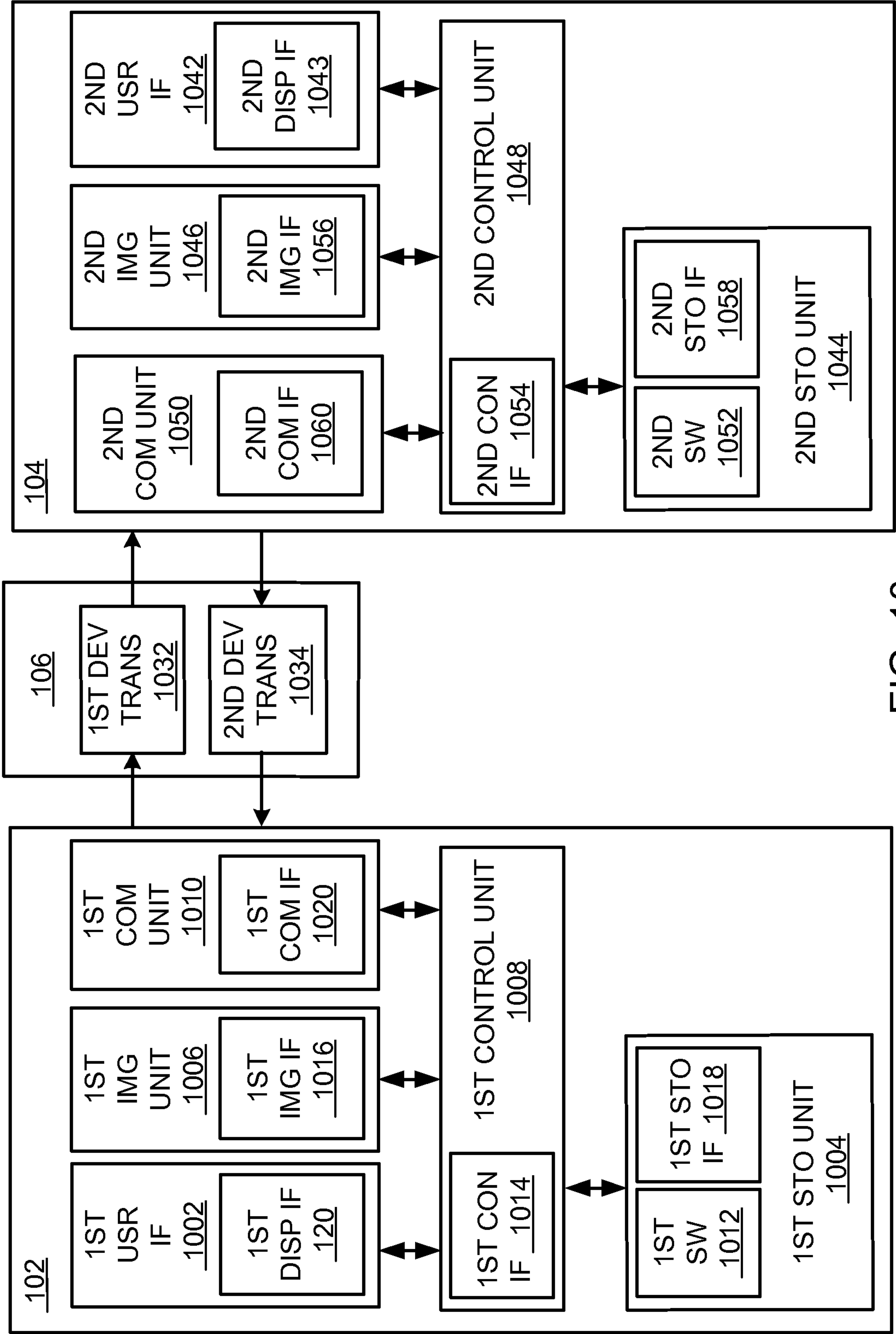


FIG. 10

11/12

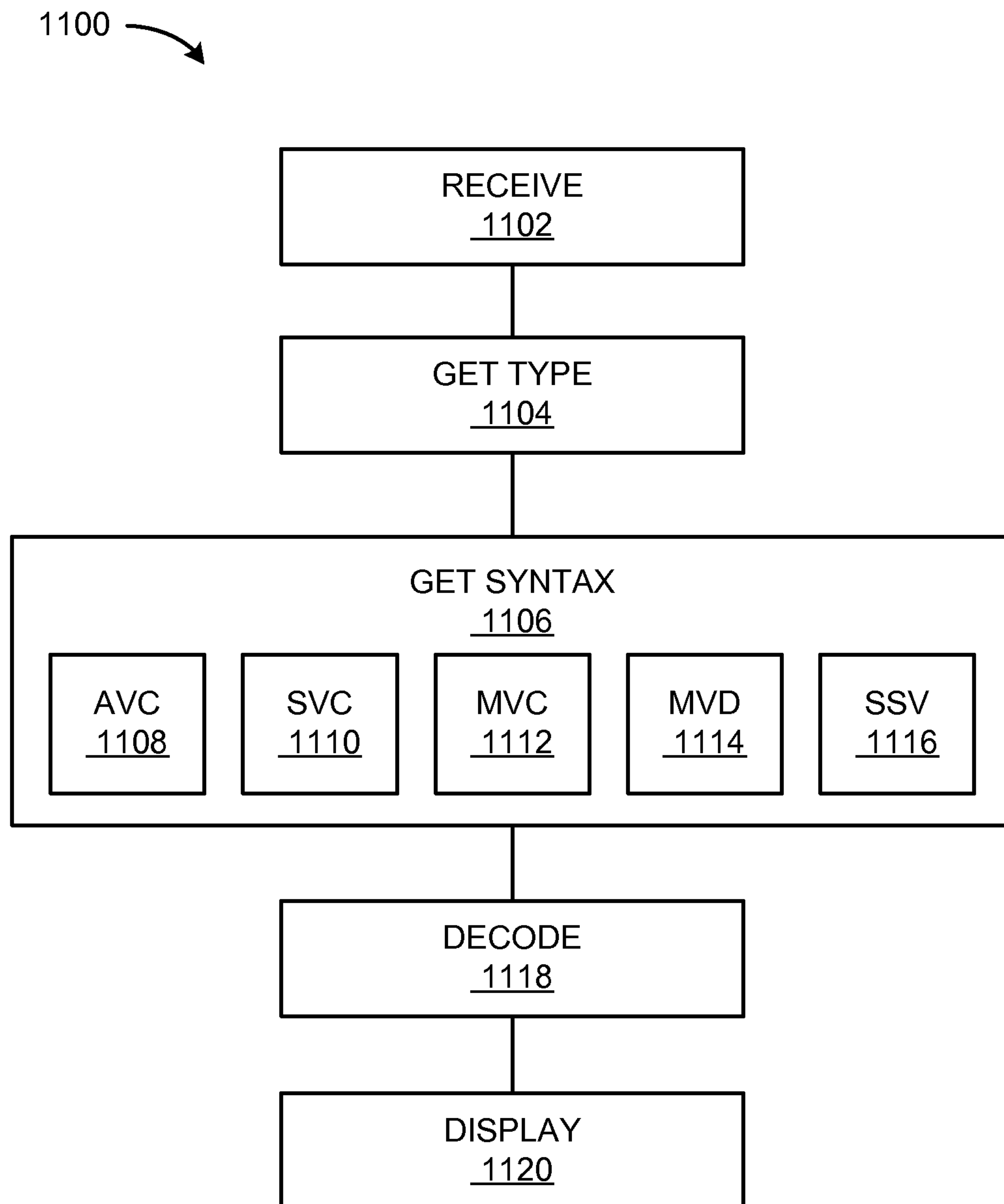


FIG. 11

12/12

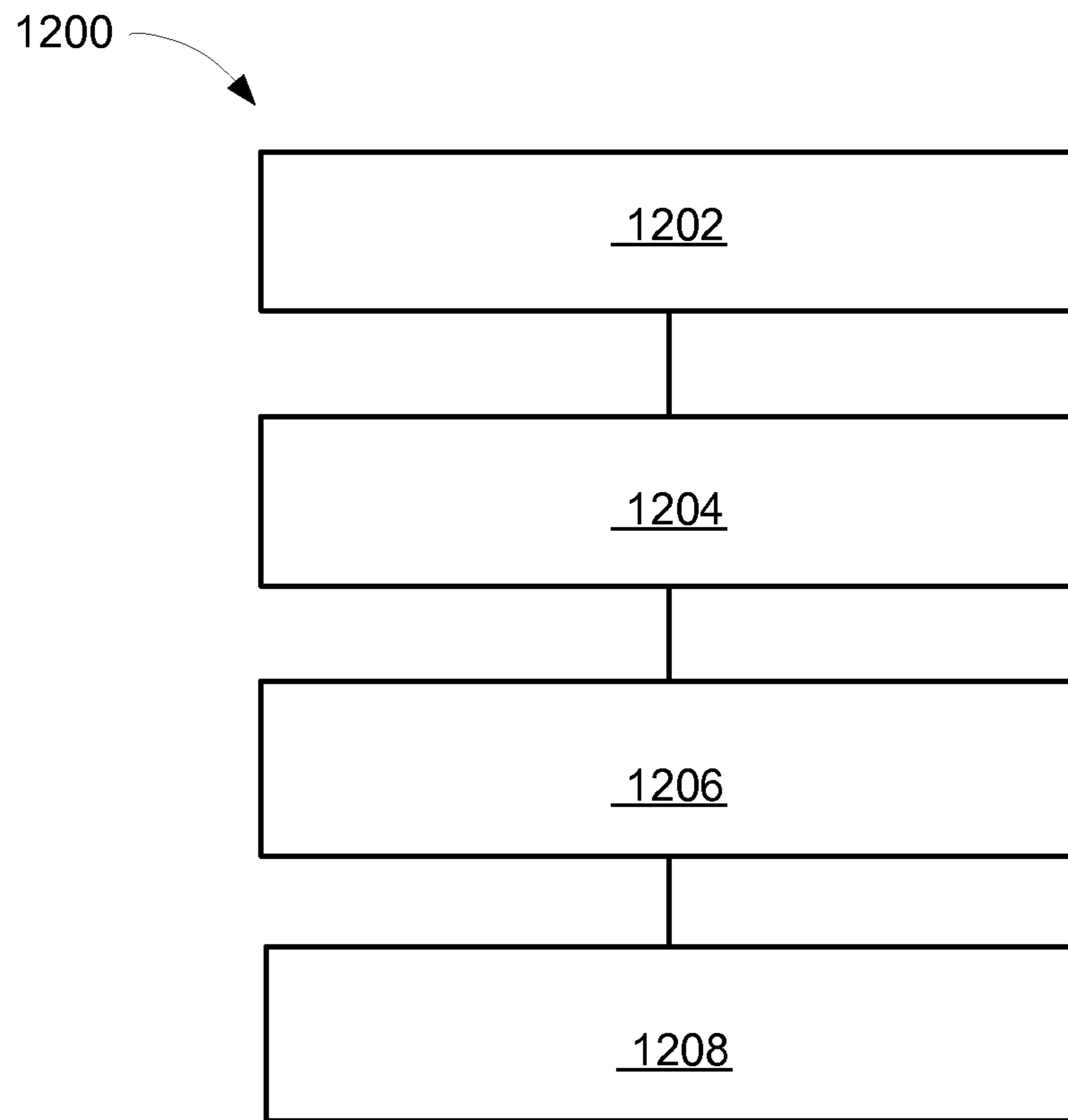


FIG. 12

1100 

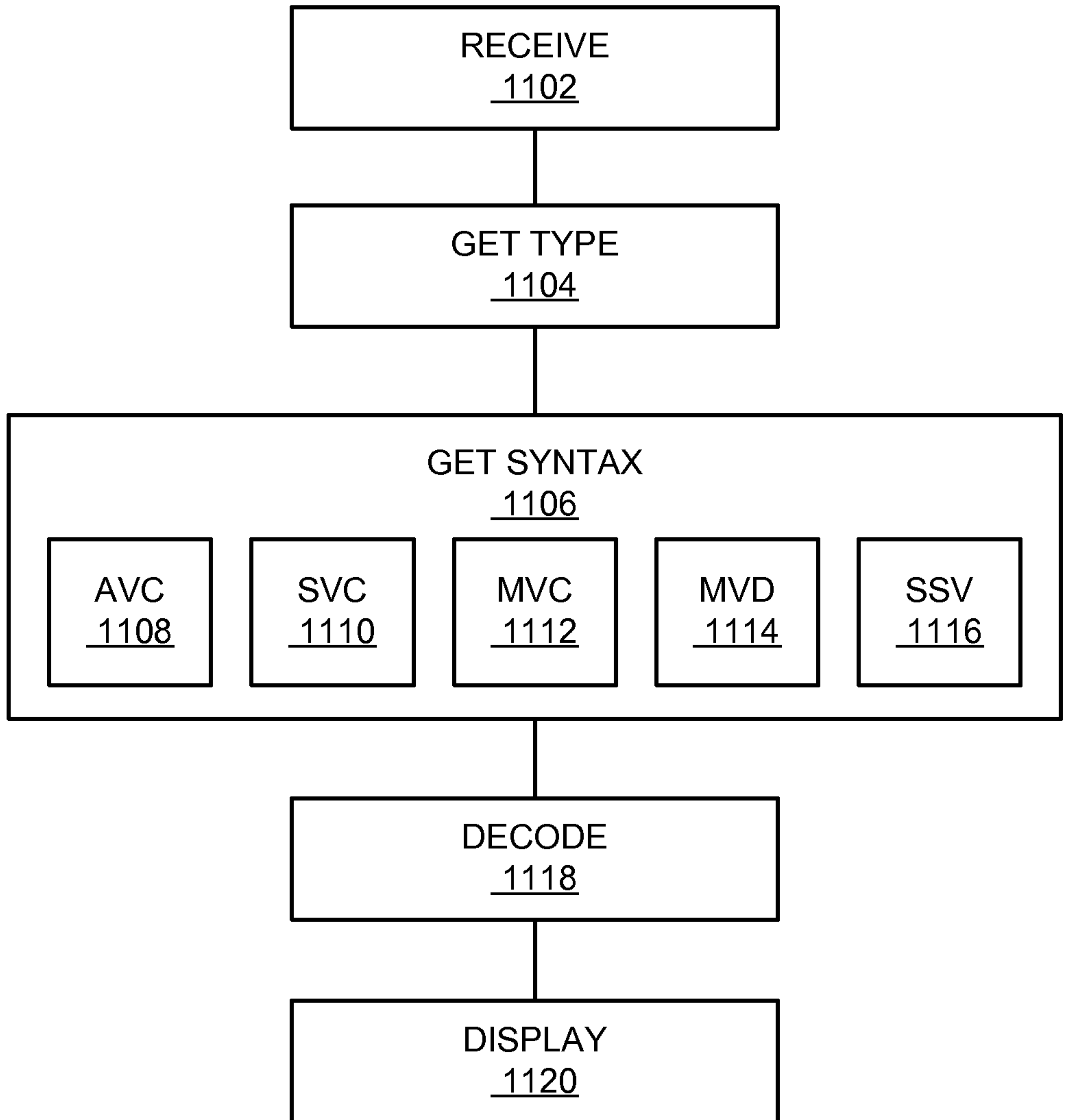


FIG. 11