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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 April 2015 (16.04.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/052942 A1

(51) International Patent Classification:
H04N 19/70 (2014.01) *H04N 19/31* (2014.01)

(21) International Application Number:
PCT/JP2014/005206

(22) International Filing Date:
14 October 2014 (14.10.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

61/889,917	11 October 2013 (11.10.2013)	US
61/890,308	13 October 2013 (13.10.2013)	US
61/953,838	15 March 2014 (15.03.2014)	US

(71) Applicant: **SHARP KABUSHIKI KAISHA** [JP/JP]; 22-22, Nagaike-cho, Abeno-ku, Osaka-shi, Osaka, 5458522 (JP).

(72) Inventor: **DESHPANDE, Sachin G.**.

(74) Agent: **HARAKENZO WORLD PATENT & TRADEMARK**; Daiwa Minamimorimachi Building, 2-6, Tenjinbashi 2-chome Kita, Kita-ku, Osaka-shi, Osaka, 5300041 (JP).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

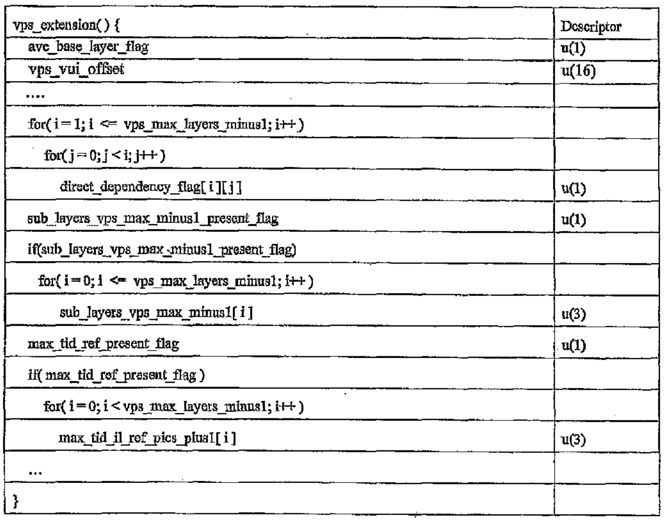
— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

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

Published:

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

(54) **Title**: SIGNALING INFORMATION FOR CODING



vps_extension( ) {	Descriptor
ave_base_layer_flag	u(1)
vps_vui_offset	u(16)
....	
for(i = 1; i <= vps_max_layers_minus1; i++)	
for(j = 0; j < i; j++)	
direct_dependency_flag[i][j]	u(1)
sub_layers_vps_max_minus1_present_flag	u(1)
if(sub_layers_vps_max_minus1_present_flag)	
for(i = 0; i <= vps_max_layers_minus1; i++)	
sub_layers_vps_max_minus1[i]	u(3)
max_tid_ref_present_flag	u(1)
if(max_tid_ref_present_flag)	
for(i = 0; i < vps_max_layers_minus1; i++)	
max_tid_il_ref_pics_plus1[i]	u(3)
...	
}	

FIG. 33

(57) **Abstract**: This invention relates to a method for decoding a video bitstream comprising the steps of: (a) receiving said video bitstream that includes a layer set, where said layer set identifies a plurality of different layers of said bitstream, where at least one of said plurality of different layers includes a plurality of temporal sub-layers; (b) receiving a video parameter set that includes information related to at least one layer of said video bitstream; (c) receiving a video parameter set extension referenced by said video parameter set that includes data regarding said plurality of different layers and said plurality of temporal sub-layers; (d) receiving a video parameter set temporal sub layers information present flag in said video parameter set extension indicating whether said information about plurality of temporal sub-layers are present.

Description

Title of Invention: SIGNALING INFORMATION FOR CODING

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] None.

Technical Field

[0002] The present disclosure relates generally to electronic devices.

Background Art

[0003] Electronic devices have become smaller and more powerful in order to meet consumer needs and to improve portability and convenience. Consumers have become dependent upon electronic devices and have come to expect increased functionality. Some examples of electronic devices include desktop computers, laptop computers, cellular phones, smart phones, media players, integrated circuits, etc.

[0004] Some electronic devices are used for processing and displaying digital media. For example, portable electronic devices now allow for digital media to be consumed at almost any location where a consumer may be. Furthermore, some electronic devices may provide download or streaming of digital media content for the use and enjoyment of a consumer.

[0005] The increasing popularity of digital media has presented several problems. For example, efficiently representing high-quality digital media for storage, transmittal and rapid playback presents several challenges. As can be observed from this discussion, systems and methods that represent digital media efficiently with improved performance may be beneficial.

[0006] The foregoing and other objectives, features, and advantages of the invention will be more readily understood upon consideration of the following detailed description of the invention, taken in conjunction with the accompanying drawings.

Summary of Invention

[0007] One embodiment of the present invention discloses a method for decoding a video bitstream comprising the steps of: (a) receiving said video bitstream that includes a layer set, where said layer set identifies a plurality of different layers of said bitstream, where at least one of said plurality of different layers includes a plurality of temporal sub-layers; (b) receiving a video parameter set that includes information related to at least one layer of said video bitstream; (c) receiving a video parameter set extension referenced by said video parameter set that includes data regarding said plurality of different layers and said plurality of temporal sub-layers; (d) receiving a video parameter set temporal sub layers information present flag in said video parameter set extension indicating whether said information about plurality of temporal sub-layers are present.

[0008] Another embodiment of the present invention discloses a method for decoding a video bitstream comprising the steps of: (a) receiving said video bitstream that includes a layer set, where said layer set identifies a plurality of different layers of said bitstream, where at least one of said plurality of different layers includes a plurality of temporal sub-layers; (b) receiving a video parameter set extension that includes data regarding said plurality of different layers and said plurality of sub-layers; (d) receiving for 0 to a maximum number of temporal sub-layers for a particular layer set (1) a bit rate present flag; (2) a picture rate present flag; (3) bit rate information (4) picture rate information.

[0009] Another embodiment of the present invention discloses a method for decoding a video bitstream comprising the steps of: (a) receiving said video bitstream that includes a plurality of different layers, where at least one of said plurality of different layers includes a plurality of temporal sub-layers; (b) receiving said video bitstream that includes a first slice as a portion of a first frame of one of said plurality of temporal sub-layers; (c) receiving said video bitstream that includes a second slice as a portion of a second frame of a different one of said plurality of temporal sub-layers; (d) receiving a first slice segment header that includes information related to said first slice of said video bitstream; (e) comparing a temporal sub layers maximum value from video parameter set with a temporal identifier of said second frame to determine whether to include said second slice as an active reference layer picture for said first slice that may be used for inter layer prediction for said first slice.

[0010] Another embodiment of the present invention discloses a method for decoding a video bitstream comprising the steps of: (a) receiving said video bitstream that includes a plurality of different layers, where at least one of said plurality of different layers includes a plurality of temporal sub-layers; (b) receiving said video bitstream that includes a first slice as a portion of a first frame of one of said plurality of temporal sub-layers; (c) receiving a first slice segment header that includes information related to said first slice of said video bitstream; (d) receiving a temporal identifier and nal unit type with said first slice segment header; (e) if said nal unit type is an IRAP picture then a TemporalId that is derived based upon said temporal identifier is equal to 0; (f) if said nal unit type is at least one of TSA and TSA_N then said TemporalId is not equal to 0; (g) if said nal unit type is at least one of STSA_R and STSA_N then said TemporalId is not equal to 0.

Brief Description of Drawings

[0011] [fig.1A]FIG. 1A is a block diagram illustrating an example of one or more electronic devices in which systems and methods for sending a message and buffering a bitstream may be implemented.

[fig.1B]FIG. 1B is another block diagram illustrating an example of one or more electronic devices in which systems and methods for sending a message and buffering a bitstream may be implemented.

[fig.2A]FIG. 2A is a block diagram illustrating one configuration of an encoder 604 on an electronic device.

[fig.2B]FIG. 2B is another block diagram illustrating one configuration of an encoder 604 on an electronic device.

[fig.3A]FIG. 3A is a block diagram illustrating one configuration of a decoder on an electronic device.

[fig.3B]FIG. 3B is another block diagram illustrating one configuration of a decoder on an electronic device.

[fig.4]FIG. 4 illustrates various components that may be utilized in a transmitting electronic device.

[fig.5]FIG. 5 is a block diagram illustrating various components that may be utilized in a receiving electronic device.

[fig.6]FIG. 6 is a block diagram illustrating one configuration of an electronic device in which systems and methods for sending a message may be implemented.

[fig.7]FIG. 7 is a block diagram illustrating one configuration of an electronic device in which systems and methods for buffering a bitstream may be implemented.

[fig.8A]FIG. 8A illustrates different NAL Unit header syntax.

[fig.8B]FIG. 8B illustrates different NAL Unit header syntax.

[fig.8C]FIG. 8C illustrates different NAL Unit header syntax.

[fig.9]FIG. 9 illustrates a general NAL Unit syntax.

[fig.10]FIG. 10 illustrates an existing video parameter set.

[fig.11]FIG. 11 illustrates existing scalability types.

[fig.12]FIG. 12 illustrates a base layer and enhancement layers.

[fig.13]FIG. 13 illustrates an exemplary picture having multiple slices.

[fig.14]FIG. 14 illustrates another exemplary picture having multiple slices.

[fig.15]FIG. 15 illustrates a picture with column and row boundaries.

[fig.16]FIG. 16 illustrates a picture with slices.

[fig.17]FIG. 17 illustrates an access unit with a base layer, enhancement layers, and tiles.

[fig.18A]FIG. 18A illustrates an exemplary slide segment header syntax.

[fig.18B]FIG. 18B illustrates an exemplary slide segment header syntax.

[fig.18C]FIG. 18C illustrates an exemplary slide segment header syntax.

[fig.18D]FIG. 18D illustrates an exemplary slide segment header syntax.

[fig.19]FIG. 19 illustrates a base layer and enhancement layers.

[fig.20A]FIG. 20A illustrates an exemplary vps extension syntax syntax.

[fig.20B]FIG. 20B illustrates an exemplary vps extension syntax syntax.

[fig.21]FIG. 21 illustrates an exemplary slice segment header syntax.

[fig.22]FIG. 22 illustrates an exemplary slice segment header syntax.

[fig.23]FIG. 23 illustrates an exemplary slice segment header syntax.

[fig.24]FIG. 24 illustrates an exemplary base layer and enhancement layer with permitted relationships.

[fig.25]FIG. 25 illustrates an exemplary slice segment header.

[fig.26A]FIG. 26A illustrates an exemplary vps extension syntax.

[fig.26B]FIG. 26B illustrates an exemplary vps extension syntax.

[fig.27]FIG. 27 illustrates an exemplary sequence parameter set syntax.

[fig.28]FIG. 28 illustrates an exemplary picture parameter set syntax.

[fig.29]FIG. 29 illustrates temporal sub-layers within a base layer and an enhancement layer.

[fig.30A]FIG. 30A illustrates an exemplary slice segment header syntax.

[fig.30B]FIG. 30B illustrates an exemplary slice segment header syntax.

[fig.30C]FIG. 30C illustrates an exemplary slice segment header syntax.

[fig.30D]FIG. 30D illustrates an exemplary slice segment header syntax.

[fig.31]FIG. 31 illustrates an exemplary vps_extension syntax.

[fig.32]FIG. 32 illustrates vps_max_sub_layers_minus1 signaling.

[fig.33]FIG. 33 illustrates an exemplary vps_extension syntax.

[fig.34]FIG. 34 illustrates vps_max_sub_layers_minus1 signaling.

[fig.35]FIG. 35 illustrates an exemplary vps_extension syntax.

[fig.36]FIG. 36 illustrates vps_max_sub_layers_minus1 signaling.

[fig.37]FIG. 37 illustrates an exemplary slice_segment_header syntax.

[fig.38]FIG. 38 illustrates an exemplary slice_segment_header syntax.

[fig.39]FIG. 39 illustrates an exemplary slice_segment_header syntax.

[fig.40]FIG. 40 illustrates an exemplary implementation for the layer_present_in_au_flag[i].

[fig.41]FIG. 41 illustrates an exemplary implementation for the layer_present_in_au_flag[i].

[fig.42]FIG. 42 illustrates an exemplary implementation for the layer_present_in_au_flag[i].

[fig.43]FIG. 43 illustrates an exemplary decoding process for inter-layer reference picture set.

[fig.44]FIG. 44 illustrates an exemplary decoding process for inter-layer reference picture set.

[fig.45]FIG. 45 illustrates an exemplary decoding process for inter-layer reference picture set.

[fig.46]FIG. 46 illustrates an exemplary decoding process for inter-layer reference picture set.

[fig.47]FIG. 47. Illustrates an exemplary slice segment header.

[fig.48A]FIG. 48A illustrates an exemplary vps extension syntax.

[fig.48B]FIG. 48B illustrates an exemplary vps extension syntax.

[fig.49]FIG. 49 illustrates an exemplary vps video usability information (VUI) syntax.

[fig.50]FIG. 50 illustrates an exemplary vps video usability information (VUI) syntax.

[fig.51]FIG. 51 illustrates temporal sub-layers within IRAP pictures and non-IRAP pictures.

[fig.52]FIG. 52 illustrates another temporal sub-layers within IRAP pictures and non-IRAP pictures.

[fig.53]FIG. 53 illustrates temporal sub-layers within IRAP pictures, TSA Pictures, STSA Pictures.

[fig.54]FIG. 54 illustrates another temporal sub-layers within IRAP pictures, TSA Pictures, STSA Pictures.

Description of Embodiments

[0012] FIG. 1A is a block diagram illustrating an example of one or more electronic devices 102 in which systems and methods for sending a message and buffering a bitstream may be implemented. In this example, electronic device A 102a and electronic device B 102b are illustrated. However, it should be noted that one or more of the features and functionality described in relation to electronic device A 102a and electronic device B 102b may be combined into a single electronic device in some configurations.

[0013] Electronic device A 102a includes an encoder 104. The encoder 104 includes a message generation module 108. Each of the elements included within electronic device A 102a (e.g., the encoder 104 and the message generation module 108) may be implemented in hardware, software or a combination of both.

[0014] Electronic device A 102a may obtain one or more input pictures 106. In some configurations, the input picture(s) 106 may be captured on electronic device A 102a using an image sensor, may be retrieved from memory and/or may be received from another electronic device.

[0015] The encoder 104 may encode the input picture(s) 106 to produce encoded data. For example, the encoder 104 may encode a series of input pictures 106 (e.g., video). In one configuration, the encoder 104 may be a HEVC encoder. The encoded data may be digital data (e.g., part of a bitstream 114). The encoder 104 may generate overhead signaling based on the input signal.

[0016] The message generation module 108 may generate one or more messages. For example, the message generation module 108 may generate one or more SEI messages or other messages. For a CPB that supports operation on a sub-picture level, the electronic device 102 may send sub-picture parameters, (e.g., CPB removal delay parameter). Specifically, the electronic device 102 (e.g., the encoder 104) may determine whether to include a common decoding unit CPB removal delay parameter in a picture timing SEI message. For example, the electronic device may set a flag (e.g., common_du_cpb_removal_delay_flag) to one when the encoder 104 is including a common decoding unit CPB removal delay parameter (e.g., common_du_cpb_removal_delay) in the picture timing SEI message. When the common decoding unit CPB removal delay parameter is included, the electronic device may generate the common decoding unit CPB removal delay parameter that is applicable to all decoding units in an access unit. In other words, rather than including a decoding unit CPB removal delay parameter for each decoding unit in an access unit, a common parameter may apply to all decoding units in the access unit with which the picture timing SEI message is associated.

[0017] In contrast, when the common decoding unit CPB removal delay parameter is not to be included in the picture timing SEI message, the electronic device 102 may generate a separate decoding unit CPB removal delay for each decoding unit in the access unit with which the picture timing SEI message is associated in some configurations, electronic device A 102a may send the message to electronic device B 102b as part of the bitstream 114. In some configurations electronic device A 102a may send the message to electronic device B 102b by a separate transmission 110. For example, the separate transmission may not be part of the bitstream 114. For instance, a picture timing SEI message or other message may be sent using some out-of-band mechanism. It should be noted that, in some configurations, the other message may include one or more of the features of a picture timing SEI message described above. Furthermore, the other message, in one or more aspects, may be utilized similarly to the SEI message described above.

[0018] The encoder 104 (and message generation module 108, for example) may produce a bitstream 114. The bitstream 114 may include encoded picture data based on the input picture(s) 106. In some configurations, the bitstream 114 may also include overhead data, such as a picture timing SEI message or other message, slice header(s), PPS(s), etc. As additional input pictures 106 are encoded, the bitstream 114 may include one or more encoded pictures. For instance, the bitstream 114 may include one or more encoded pictures with corresponding overhead data (e.g., a picture timing SEI message or other message).

[0019] The bitstream 114 may be provided to a decoder 112. In one example, the bitstream 114 may be transmitted to electronic device B 102b using a wired or wireless link. In some cases, this may be done over a network, such as the Internet or a Local Area Network (LAN). As illustrated in FIG. 1A, the decoder 112 may be implemented on electronic device B 102b separately from the encoder 104 on electronic device A 102a. However, it should be noted that the encoder 104 and decoder 112 may be implemented on the same electronic device in some configurations. In an implementation where the encoder 104 and decoder 112 are implemented on the same electronic device, for instance, the bitstream 114 may be provided over a bus to the decoder 112 or stored in memory for retrieval by the decoder 112.

[0020] The decoder 112 may be implemented in hardware, software or a combination of both. In one configuration, the decoder 112 may be a HEVC decoder. The decoder 112 may receive (e.g., obtain) the bitstream 114. The decoder 112 may generate one or more decoded pictures 118 based on the bitstream 114. The decoded picture(s) 118 may be displayed, played back, stored in memory and/or transmitted to another device, etc.

[0021] The decoder 112 may include a CPB 120. The CPB 120 may temporarily store encoded pictures. The CPB 120 may use parameters found in a picture timing SEI message to determine when to remove data. When the CPB 120 supports operation on a sub-picture level, individual decoding units may be removed rather than entire access units at one time. The decoder 112 may include a Decoded Picture Buffer (DPB) 122. Each decoded picture is placed in the DPB 122 for being referenced by the decoding process as well as for output and cropping. A decoded picture is removed from the DPB at the later of the DPB output time or the time that it becomes no longer needed for inter-prediction reference.

[0022] The decoder 112 may receive a message (e.g., picture timing SEI message or other message). The decoder 112 may also determine whether the received message includes a common decoding unit CPB removal delay parameter (e.g., common_du_cpb_removal_delay). This may include identifying a flag (e.g., common_du_cpb_removal_delay_flag) that is set when the common parameter is present in the picture timing SEI message. If the common parameter is present, the decoder 112 may determine the common decoding unit CPB removal delay parameter applicable to all decoding units in the access unit. If the common parameter is not present, the decoder 112 may determine a separate decoding unit CPB removal delay parameter for each decoding unit in the access unit. The decoder 112 may also remove decoding units from the CPB 120 using either the common decoding unit CPB removal delay parameter or the separate decoding unit CPB removal delay parameters.

[0023] The HRD described above may be one example of the decoder 112 illustrated in FIG. 1A. Thus, an electronic device 102 may operate in accordance with the HRD and CPB 120 and DPB 122 described above, in some configurations.

[0024] It should be noted that one or more of the elements or parts thereof included in the electronic device(s) 102 may be implemented in hardware. For example, one or more of these elements or parts thereof may be implemented as a chip, circuitry or hardware components, etc. It should also be noted that one or more of the functions or methods described herein may be implemented in and/or performed using hardware. For example, one or more of the methods described herein may be implemented in and/or realized using a chipset, an Application-Specific Integrated Circuit (ASIC), a Large-Scale Integrated circuit (LSI) or integrated circuit, etc.

[0025] FIG. 1B is a block diagram illustrating another example of an encoder 1908 and a decoder 1972. In this example, electronic device A 1902 and electronic device B 1970 are illustrated. However, it should be noted that the features and functionality described in relation to electronic device A 1902 and electronic device B 1970 may be combined into a single electronic device in some configurations.

[0026] Electronic device A 1902 includes the encoder 1908. The encoder 1908 may include a base layer encoder 1910 and an enhancement layer encoder 1920. The video encoder 1908 is suitable for scalable video coding and multi-view video coding, as described later. The encoder 1908 may be implemented in hardware, software or a combination of both. In one configuration, the encoder 1908 may be a high-efficiency video coding (HEVC) coder, including scalable and/or multi-view. Other coders may likewise be used. Electronic device A 1902 may obtain a source 1906. In some configurations, the source 1906 may be captured on electronic device A 1902 using an image sensor, retrieved from memory or received from another electronic device.

[0027] The encoder 1908 may code the source 1906 to produce a base layer bitstream 1934 and an enhancement layer bitstream 1936. For example, the encoder 1908 may code a series of pictures (e.g., video) in the source 1906. In particular, for scalable video encoding for SNR scalability also known as quality scalability the same source 1906 may be provided to the base layer and the enhancement layer encoder. In particular, for scalable video encoding for spatial scalability a downsampled source may be used for the base layer encoder. In particular, for multi-view encoding a different view source may be used for the base layer encoder and the enhancement layer encoder. The encoder 1908 may be similar to the encoder 1782 described later in connection with FIG. 2B.

[0028] The bitstreams 1934, 1936 may include coded picture data based on the source 1906. In some configurations, the bitstreams 1934, 1936 may also include overhead data, such as slice header information, PPS information, etc. As additional pictures in the source 1906 are coded, the bitstreams 1934, 1936 may include one or more coded pictures.

[0029] The bitstreams 1934, 1936 may be provided to the decoder 1972. The decoder 1972 may include a base layer decoder 1980 and an enhancement layer decoder 1990. The video decoder 1972 is suitable for scalable video decoding and multi-view video decoding. In one example, the bitstreams 1934, 1936 may be transmitted to electronic device B 1970 using a wired or wireless link. In some cases, this may be done over a network, such as the Internet or a Local Area Network (LAN). As illustrated in FIG. 1B, the decoder 1972 may be implemented on electronic device B 1970 separately from the encoder 1908 on electronic device A 1902. However, it should be noted that the encoder 1908 and decoder 1972 may be implemented on the same electronic device in some configurations. In an implementation where the encoder 1908 and decoder 1972 are implemented on the same electronic device, for instance, the bitstreams 1934, 1936 may be provided over a bus to the decoder 1972 or stored in memory for retrieval by the decoder 1972. The decoder 1972 may provide a decoded base layer 1992 and decoded enhancement layer picture(s) 1994 as output.

[0030] The decoder 1972 may be implemented in hardware, software or a combination of both. In one configuration, the decoder 1972 may be a high-efficiency video coding (HEVC) decoder, including scalable and/or multi-view. Other decoders may likewise be used. The decoder 1972 may be similar to the decoder 1812 described later in connection with FIG. 3B. Also, the base layer encoder and/or the enhancement layer encoder may each include a message generation module, such as that described in relation to FIG. 1A. Also, the base layer decoder and/or the enhancement layer decoder may include a coded picture buffer and/or a decoded picture buffer, such as that described in relation to FIG. 1A. In addition, the electronic devices of FIG. 1B may operate in accordance with the functions of the electronic devices of FIG. 1A, as applicable.

[0031] FIG. 2A is a block diagram illustrating one configuration of an encoder 604 on an electronic device 602. It should be noted that one or more of the elements illustrated as included within the electronic device 602 may be implemented in hardware, software or a combination of both. For example, the electronic device 602 includes an encoder 604, which may be implemented in hardware, software or a combination of both. For instance, the encoder 604 may be implemented as a circuit, integrated circuit, application-specific integrated circuit (ASIC), processor in electronic communication with memory with executable instructions, firmware, field-programmable gate array (FPGA), etc., or a combination thereof. In some configurations, the encoder 604 may be a HEVC coder.

[0032] The electronic device 602 may include a source 622. The source 622 may provide picture or image data (e.g., video) as one or more input pictures 606 to the encoder 604. Examples of the source 622 may include image sensors, memory, communication interfaces, network interfaces, wireless receivers, ports, etc.

[0033] One or more input pictures 606 may be provided to an intra-frame prediction module and reconstruction buffer 624. An input picture 606 may also be provided to a motion estimation and motion compensation module 646 and to a subtraction module 628.

[0034] The intra-frame prediction module and reconstruction buffer 624 may generate intra mode information 640 and an intra-signal 626 based on one or more input pictures 606 and reconstructed data 660. The motion estimation and motion compensation module 646 may generate inter mode information 648 and an inter signal 644 based on one or more input pictures 606 and a reference picture 678 from decoded picture buffer 676. In some configurations, the decoded picture buffer 676 may include data from one or more reference pictures in the decoded picture buffer 676.

[0035] The encoder 604 may select between the intra signal 626 and the inter signal 644 in accordance with a mode. The intra signal 626 may be used in order to exploit spatial characteristics within a picture in an intra-coding mode. The inter signal 644 may be used in order to exploit temporal characteristics between pictures in an inter coding mode. While in the intra coding mode, the intra signal 626 may be provided to the subtraction module 628 and the intra mode information 640 may be provided to an entropy coding module 642. While in the inter coding mode, the inter signal 644 may be provided to the subtraction module 628 and the inter mode information 648 may be provided to the entropy coding module 642.

[0036] Either the intra signal 626 or the inter signal 644 (depending on the mode) is subtracted from an input picture 606 at the subtraction module 628 in order to produce a prediction residual 630. The prediction residual 630 is provided to a transformation module 632. The transformation module 632 may compress the prediction residual 630 to produce a transformed signal 634 that is provided to a quantization module 636. The quantization module 636 quantizes the transformed signal 634 to produce transformed and quantized coefficients (TQCs) 638.

[0037] The TQCs 638 are provided to an entropy coding module 642 and an inverse quantization module 650. The inverse quantization module 650 performs inverse quantization on the TQCs 638 to produce an inverse quantized signal 652 that is provided to an inverse transformation module 654. The inverse transformation module 654 decompresses the inverse quantized signal 652 to produce a decompressed signal 656 that is provided to a reconstruction module 658.

[0038] The reconstruction module 658 may produce reconstructed data 660 based on the decompressed signal 656. For example, the reconstruction module 658 may reconstruct (modified) pictures. The reconstructed data 660 may be provided to a deblocking filter 662 and to the intra prediction module and reconstruction buffer 624. The deblocking filter 662 may produce a filtered signal 664 based on the reconstructed data 660.

[0039] The filtered signal 664 may be provided to a sample adaptive offset (SAO) module

666. The SAO module 666 may produce SAO information 668 that is provided to the entropy coding module 642 and an SAO signal 670 that is provided to an adaptive loop filter (ALF) 672. The ALF 672 produces an ALF signal 674 that is provided to the decoded picture buffer 676. The ALF signal 674 may include data from one or more pictures that may be used as reference pictures.

[0040] The entropy coding module 642 may code the TQCs 638 to produce bitstream A 614a (e.g., encoded picture data). For example, the entropy coding module 642 may code the TQCs 638 using Context-Adaptive Variable Length Coding (CAVLC) or Context-Adaptive Binary Arithmetic Coding (CABAC). In particular, the entropy coding module 642 may code the TQCs 638 based on one or more of intra mode information 640, inter mode information 648 and SAO information 668. Bitstream A 614a (e.g., encoded picture data) may be provided to a message generation module 608. The message generation module 608 may be configured similarly to the message generation module 108 described in connection with FIG. 1

[0041] For example, the message generation module 608 may generate a message (e.g., picture timing SEI message or other message) including sub-picture parameters. The sub-picture parameters may include one or more removal delays for decoding units (e.g., common_du_cpb_removal_delay or du_cpb_removal_delay[i]) and one or more NAL parameters (e.g., common_num_nalus_in_du_minus1 or num_nalus_in_du_minus1[i]). In some configurations, the message may be inserted into bitstream A 614a to produce bitstream B 614b. Thus, the message may be generated after the entire bitstream A 614a is generated (e.g., after most of bitstream B 614b is generated), for example. In other configurations, the message may not be inserted into bitstream A 614a (in which case bitstream B 614b may be the same as bitstream A 614a), but may be provided in a separate transmission 610.

[0042] In some configurations, the electronic device 602 sends the bitstream 614 to another electronic device. For example, the bitstream 614 may be provided to a communication interface, network interface, wireless transmitter, port, etc. For instance, the bitstream 614 may be transmitted to another electronic device via LAN, the Internet, a cellular phone base station, etc. The bitstream 614 may additionally or alternatively be stored in memory or other component on the electronic device 602.

[0043] FIG. 2B is a block diagram illustrating one configuration of a video encoder 1782 on an electronic device 1702. The video encoder 1782 may include an enhancement layer encoder 1706, a base layer encoder 1709, a resolution upscaling block 1770 and an output interface 1780. The video encoder of FIG. 2B, for example, is suitable for scalable video coding and multi-view video coding, as described herein.

[0044] The enhancement layer encoder 1706 may include a video input 1781 that receives an input picture 1704. The output of the video input 1781 may be provided to an adder/

subtractor 1783 that receives an output of a prediction selection 1750. The output of the adder/subtractor 1783 may be provided to a transform and quantize block 1752. The output of the transform and quantize block 1752 may be provided to an entropy encoding 1748 block and a scaling and inverse transform block 1772. After entropy encoding 1748 is performed, the output of the entropy encoding block 1748 may be provided to the output interface 1780. The output interface 1780 may output both the encoded base layer video bitstream 1707 and the encoded enhancement layer video bitstream 1710.

[0045] The output of the scaling and inverse transform block 1772 may be provided to an adder 1779. The adder 1779 may also receive the output of the prediction selection 1750. The output of the adder 1779 may be provided to a deblocking block 1751. The output of the deblocking block 1751 may be provided to a reference buffer 1794. An output of the reference buffer 1794 may be provided to a motion compensation block 1754. The output of the motion compensation block 1754 may be provided to the prediction selection 1750. An output of the reference buffer 1794 may also be provided to an intra predictor 1756. The output of the intra predictor 1756 may be provided to the prediction selection 1750. The prediction selection 1750 may also receive an output of the resolution upscaling block 1770.

[0046] The base layer encoder 1709 may include a video input 1762 that receives a downsampled input picture, or other image content suitable for combing with another image, or an alternative view input picture or the same input picture 1703 (i.e., the same as the input picture 1704 received by the enhancement layer encoder 1706). The output of the video input 1762 may be provided to an encoding prediction loop 1764. Entropy encoding 1766 may be provided on the output of the encoding prediction loop 1764. The output of the encoding prediction loop 1764 may also be provided to a reference buffer 1768. The reference buffer 1768 may provide feedback to the encoding prediction loop 1764. The output of the reference buffer 1768 may also be provided to the resolution upscaling block 1770. Once entropy encoding 1766 has been performed, the output may be provided to the output interface 1780. The encoded base layer video bitstream 1707 and/or the encoded enhancement layer video bitstream 1710 may be provided to one or more message generation modules, as desired.

[0047] FIG. 3A is a block diagram illustrating one configuration of a decoder 712 on an electronic device 702. The decoder 712 may be included in an electronic device 702. For example, the decoder 712 may be a HEVC decoder. The decoder 712 and one or more of the elements illustrated as included in the decoder 712 may be implemented in hardware, software or a combination of both. The decoder 712 may receive a bitstream 714 (e.g., one or more encoded pictures and overhead data included in the bitstream 714) for decoding. In some configurations, the received bitstream 714 may include received overhead data, such as a message (e.g., picture timing SEI message or other message), slice header, PPS, etc. In some configurations, the decoder 712 may additionally receive a separate transmission 710. The separate transmission 710 may include a message (e.g., a picture timing SEI message or other message). For example, a picture timing SEI message or other message may be received in a separate transmission 710 instead of in the bitstream 714. However, it should be noted that the separate transmission 710 may be optional and may not be utilized in some configurations.

[0048] The decoder 712 includes a CPB 720. The CPB 720 may be configured similarly to the CPB 120 described in connection with FIG. 1 above. The decoder 712 may receive a message (e.g., picture timing SEI message or other message) with sub-picture parameters and remove and decode decoding units in an access unit based on the sub-picture parameters. It should be noted that one or more access units may be included in the bitstream and may include one or more of encoded picture data and overhead data.

[0049] The Coded Picture Buffer (CPB) 720 may provide encoded picture data to an entropy decoding module 701. The encoded picture data may be entropy decoded by an entropy decoding module 701, thereby producing a motion information signal 703 and quantized, scaled and/or transformed coefficients 705.

[0050] The motion information signal 703 may be combined with a portion of a reference frame signal 798 from a decoded picture buffer 709 at a motion compensation module 780, which may produce an inter-frame prediction signal 782. The quantized, descaled and/or transformed coefficients 705 may be inverse quantized, scaled and inverse transformed by an inverse module 707, thereby producing a decoded residual signal 784. The decoded residual signal 784 may be added to a prediction signal 792 to produce a combined signal 786. The prediction signal 792 may be a signal selected from either the inter-frame prediction signal 782 produced by the motion compensation module 780 or an intra-frame prediction signal 790 produced by an intra-frame prediction module 788. In some configurations, this signal selection may be based on (e.g., controlled by) the bitstream 714.

[0051] The intra-frame prediction signal 790 may be predicted from previously decoded information from the combined signal 786 (in the current frame, for example). The combined signal 786 may also be filtered by a de-blocking filter 794. The resulting filtered signal 796 may be written to decoded picture buffer 709. The resulting filtered signal 796 may include a decoded picture. The decoded picture buffer 709 may provide a decoded picture which may be outputted 718. In some cases 709 may be a considered as frame memory.

[0052] FIG. 3B is a block diagram illustrating one configuration of a video decoder 1812 on an electronic device 1802. The video decoder 1812 may include an enhancement layer decoder 1815 and a base layer decoder 1813. The video decoder 812 may also include an interface 1889 and resolution upscaling 1870. The video decoder of FIG. 3B, for example, is suitable for scalable video coding and multi-view video encoded, as described herein.

[0053] The interface 1889 may receive an encoded video stream 1885. The encoded video stream 1885 may consist of base layer encoded video stream and enhancement layer encoded video stream. These two streams may be sent separately or together. The interface 1889 may provide some or all of the encoded video stream 1885 to an entropy decoding block 1886 in the base layer decoder 1813. The output of the entropy decoding block 1886 may be provided to a decoding prediction loop 1887. The output of the decoding prediction loop 1887 may be provided to a reference buffer 1888. The reference buffer may provide feedback to the decoding prediction loop 1887. The reference buffer 1888 may also output the decoded base layer video stream 1884.

[0054] The interface 1889 may also provide some or all of the encoded video stream 1885 to an entropy decoding block 1890 in the enhancement layer decoder 1815. The output of the entropy decoding block 1890 may be provided to an inverse quantization block 1891. The output of the inverse quantization block 1891 may be provided to an adder 1892. The adder 1892 may add the output of the inverse quantization block 1891 and the output of a prediction selection block 1895. The output of the adder 1892 may be provided to a deblocking block 1893. The output of the deblocking block 1893 may be provided to a reference buffer 1894. The reference buffer 1894 may output the decoded enhancement layer video stream 1882. The output of the reference buffer 1894 may also be provided to an intra predictor 1897. The enhancement layer decoder 1815 may include motion compensation 1896. The motion compensation 1896 may be performed after the resolution upscaling 1870. The prediction selection block 1895 may receive the output of the intra predictor 1897 and the output of the motion compensation 1896. Also, the decoder may include one or more coded picture buffers, as desired, such as together with the interface 1889.

[0055] FIG. 4 illustrates various components that may be utilized in a transmitting electronic device 802. One or more of the electronic devices 102, 602, 702 described herein may be implemented in accordance with the transmitting electronic device 802 illustrated in FIG. 4.

[0056] The transmitting electronic device 802 includes a processor 817 that controls operation of the electronic device 802. The processor 817 may also be referred to as a CPU. Memory 811, which may include both read-only memory (ROM), random access memory (RAM) or any type of device that may store information, provides instructions 813a (e.g., executable instructions) and data 815a to the processor 817. A portion of the memory 811 may also include non-volatile random access memory (NVRAM).

The memory 811 may be in electronic communication with the processor 817.

[0057] Instructions 813b and data 815b may also reside in the processor 817. Instructions 813b and/or data 815b loaded into the processor 817 may also include instructions 813a and/or data 815a from memory 811 that were loaded for execution or processing by the processor 817. The instructions 813b may be executed by the processor 817 to implement the systems and methods disclosed herein. For example, the instructions 813b may be executable to perform one or more of the methods 200, 300, 400, 500 described above.

[0058] The transmitting electronic device 802 may include one or more communication interfaces 819 for communicating with other electronic devices (e.g., receiving electronic device). The communication interfaces 819 may be based on wired communication technology, wireless communication technology, or both. Examples of a communication interface 819 include a serial port, a parallel port, a Universal Serial Bus (USB), an Ethernet adapter, an IEEE 1394 bus interface, a small computer system interface (SCSI) bus interface, an infrared (IR) communication port, a Bluetooth wireless communication adapter, a wireless transceiver in accordance with 3rd Generation Partnership Project (3GPP) specifications and so forth.

[0059] The transmitting electronic device 802 may include one or more output devices 823 and one or more input devices 821. Examples of output devices 823 include a speaker, printer, etc. One type of output device that may be included in an electronic device 802 is a display device 825. Display devices 825 used with configurations disclosed herein may utilize any suitable image projection technology, such as a cathode ray tube (CRT), liquid crystal display (LCD), light-emitting diode (LED), gas plasma, electroluminescence or the like. A display controller 827 may be provided for converting data stored in the memory 811 into text, graphics, and/or moving images (as appropriate) shown on the display 825. Examples of input devices 821 include a keyboard, mouse, microphone, remote control device, button, joystick, trackball, touchpad, touchscreen, lightpen, etc.

[0060] The various components of the transmitting electronic device 802 are coupled together by a bus system 829, which may include a power bus, a control signal bus and a status signal bus, in addition to a data bus. However, for the sake of clarity, the various buses are illustrated in FIG. 4 as the bus system 829. The transmitting electronic device 802 illustrated in FIG. 4 is a functional block diagram rather than a listing of specific components.

[0061] FIG. 5 is a block diagram illustrating various components that may be utilized in a receiving electronic device 902. One or more of the electronic devices 102, 602, 702 described herein may be implemented in accordance with the receiving electronic device 902 illustrated in FIG. 5.

[0062] The receiving electronic device 902 includes a processor 917 that controls operation of the electronic device 902. The processor 917 may also be referred to as a CPU. Memory 911, which may include both read-only memory (ROM), random access memory (RAM) or any type of device that may store information, provides instructions 913a (e.g., executable instructions) and data 915a to the processor 917. A portion of the memory 911 may also include non-volatile random access memory (NVRAM). The memory 911 may be in electronic communication with the processor 917.

[0063] Instructions 913b and data 915b may also reside in the processor 917. Instructions 913b and/or data 915b loaded into the processor 917 may also include instructions 913a and/or data 915a from memory 911 that were loaded for execution or processing by the processor 917. The instructions 913b may be executed by the processor 917 to implement the systems and methods disclosed herein. For example, the instructions 913b may be executable to perform one or more of the methods 200, 300, 400, 500 described above.

[0064] The receiving electronic device 902 may include one or more communication interfaces 919 for communicating with other electronic devices (e.g., a transmitting electronic device). The communication interface 919 may be based on wired communication technology, wireless communication technology, or both. Examples of a communication interface 919 include a serial port, a parallel port, a Universal Serial Bus (USB), an Ethernet adapter, an IEEE 1394 bus interface, a small computer system interface (SCSI) bus interface, an infrared (IR) communication port, a Bluetooth wireless communication adapter, a wireless transceiver in accordance with 3rd Generation Partnership Project (3GPP) specifications and so forth.

[0065] The receiving electronic device 902 may include one or more output devices 923 and one or more input devices 921. Examples of output devices 923 include a speaker, printer, etc. One type of output device that may be included in an electronic device 902 is a display device 925. Display devices 925 used with configurations disclosed herein may utilize any suitable image projection technology, such as a cathode ray tube (CRT), liquid crystal display (LCD), light-emitting diode (LED), gas plasma, electroluminescence or the like. A display controller 927 may be provided for converting data stored in the memory 911 into text, graphics, and/or moving images (as appropriate) shown on the display 925. Examples of input devices 921 include a keyboard, mouse, microphone, remote control device, button, joystick, trackball, touchpad, touchscreen, lightpen, etc.

[0066] The various components of the receiving electronic device 902 are coupled together by a bus system 929, which may include a power bus, a control signal bus and a status signal bus, in addition to a data bus. However, for the sake of clarity, the various buses are illustrated in FIG. 5 as the bus system 929. The receiving electronic device 902 illustrated in FIG. 5 is a functional block diagram rather than a listing of specific components.

[0067] FIG. 6 is a block diagram illustrating one configuration of an electronic device 1002 in which systems and methods for sending a message may be implemented. The electronic device 1002 includes encoding means 1031 and transmitting means 1033. The encoding means 1031 and transmitting means 1033 may generate a bitstream 1014. FIG. 4 above illustrates one example of a concrete apparatus structure of FIG. 6. A DSP may be realized by software.

[0068] FIG. 7 is a block diagram illustrating one configuration of an electronic device 1102 in which systems and methods for buffering a bitstream 1114 may be implemented. The electronic device 1102 may include receiving means 1135 and decoding means 1137. The receiving means 1135 and decoding means 1137 may receive a bitstream 1114. FIG. 5 above illustrates one example of a concrete apparatus structure of FIG. 7. A DSP may be realized by software.

[0069] The decoding process for reference picture set (RPS) may be invoked. Reference picture set is a set of reference pictures associated with a picture, consisting of all reference pictures that are prior to the associated picture in decoding order, that may be used for inter prediction of the associated picture or any picture following the associated picture in decoding order.

[0070] The bitstream of the video may include a syntax structure that is placed into logical data packets generally referred to as Network Abstraction Layer (NAL) units. Each NAL unit includes a NAL unit header, such as a two-byte NAL unit header (e.g., 16 bits), to identify the purpose of the associated data payload. For example, each coded slice (and/or picture) may be coded in one or more slice (and/or picture) NAL units. Other NAL units may be included for other categories of data, such as for example, supplemental enhancement information, coded slice of temporal sub-layer access (TSA) picture, coded slice of step-wise temporal sub-layer access (STSA) picture, coded slice a non-TSA, non-STSA trailing picture, coded slice of broken link access picture, coded slice of instantaneous decoded refresh picture, coded slice of clean random access picture, coded slice of decodable leading picture, coded slice of tagged for discard picture, video parameter set, sequence parameter set, picture parameter set, access unit delimiter, end of sequence, end of bitstream, filler data, and/or sequence enhancement information message. Table (1) illustrates one example of NAL unit codes and NAL unit type classes. Other NAL unit types may be included, as desired. It should also be understood that the NAL unit type values for the NAL units shown in the Table (1) may be reshuffled and reassigned. Also additional NAL unit types may be added. Also some NAL unit types may be removed.

[0071] An intra random access point (IRAP) picture is a coded picture for which each video coding layer NAL unit has nal_unit_type in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive as shown in Table (1). An IRAP picture contains only Intra coded (I) slices. An instantaneous decoding refresh (IDR) picture is an IRAP picture for which each video coding layer NAL unit has nal_unit_type equal to IDR_W_RADL or IDR_N_LP as shown in Table 14). An instantaneous decoding refresh(IDR) picture contains only I slices, and may be the first picture in the bitstream in decoding order, or may appear later in the bitstream. Each IDR picture is the first picture of a coded video sequence (CVS) in decoding order. A broken link access (BLA) picture is an IRAP picture for which each video coding layer NAL unit has nal_unit_type equal to BLA_W_LP, BLA_W_RADL, or BLA_N_LP as shown in Table (1). A BLA picture contains only I slices, and may be the first picture in the bitstream in decoding order, or may appear later in the bitstream. Each BLA picture begins a new coded video sequence, and has the same effect on the decoding process as an IDR picture. However, a BLA picture contains syntax elements that specify a non-empty reference picture set.

nal_unit_type	Name of nal_unit_type	Content of NAL unit and raw byte sequence payload (RBSP) syntax structure	NAL unit type class
0 1	TRAIL_N TRAIL_R	Coded slice segment of a non-TSA, non-STSA trailing picture slice_segment_layer_rbsp( )	Video Coding Layer (VCL)
2 3	TSA_N TSA_R	Coded slice segment of a temporal sub-layer access (TSA) picture slice_segment_layer_rbsp( )	VCL
4 5	STSA_N STSA_R	Coded slice segment of an Step-wise Temporal sub-layer access (STSA) picture slice_segment_layer_rbsp( )	VCL
6 7	RADL_N RADL_R	Coded slice segment of a random access decodable leading (RADL) picture slice_segment_layer_rbsp( )	VCL
8 9	RASL_N RASL_R	Coded slice segment of a random access skipped leading (RASL) picture slice_segment_layer_rbsp( )	VCL
10 12 14	RSV_VCL_N10 RSV_VCL_N12 RSV_VCL_N14	Reserved non-IRAP sub-layer non-reference VCL NAL unit types	VCL
11 13 15	RSV_VCL_R11 RSV_VCL_R13 RSV_VCL_R15	Reserved non-IRAP sub-layer reference VCL NAL unit types	VCL
16 17 18	BLA_W_LP BLA_W_RADL BLA_N_LP	Coded slice segment of a broken link access (BLA) picture slice_segment_layer_rbsp( )	VCL
19 20	IDR_W_RADL IDR_N_LP	Coded slice segment of an instantaneous decoding refresh (IDR) picture slice_segment_layer_rbsp( )	VCL

21	CRA_NUT	Coded slice segment of a clean random access (CRA) picture slice_segment_layer_rbsp( )	VCL
22 23	RSV_IRAP_VCL22 RSV_IRAP_VCL23	Reserved IRAP VCL NAL unit types	VCL
24..31	RSV_VCL24.. RSV_VCL31	Reserved non-IRAP VCL NAL unit types	VCL
32	VPS_NUT	Video parameter set video_parameter_set_rbsp( )	non-video coding layer (non-VCL)
33	SPS_NUT	Sequence parameter set seq_parameter_set_rbsp( )	non-VCL
34	PPS_NUT	Picture parameter set pic_parameter_set_rbsp( )	non-VCL
35	AUD_NUT	Access unit delimiter access_unit_delimiter_rbsp( )	non-VCL
36	EOS_NUT	End of sequence end_of_seq_rbsp( )	non-VCL
37	EOB_NUT	End of bitstream end_of_bitstream_rbsp( )	non-VCL
38	FD_NUT	Filler data filler_data_rbsp( )	non-VCL
39 40	PREFIX_SEI_NUT SUFFIX_SEI_NUT	Supplemental enhancement information sei_rbsp( )	non-VCL
41..47	RSV_NVCL41.. RSV_NVCL47	Reserved	non-VCL
48..63	UNSPEC48.. UNSPEC63	Unspecified	non-VCL

Table (1)

[0072] Referring to Table (2), the NAL unit header syntax may include two bytes of data, namely, 16 bits. The first bit is a "forbidden_zero_bit" which is always set to zero at the start of a NAL unit. The next six bits is a "nal_unit_type" which specifies the type of raw byte sequence payloads ("RBSP") data structure contained in the NAL unit as shown in Table (1). The next 6 bits is a "nuh_layer_id" which specify the indentifier of the layer. In some cases these six bits may be specified as "nuh_reserved_zero_6bits" instead. The nuh_reserved_zero_6bits may be equal to 0 in the base specification of the standard. In a scalable video coding and/or syntax extensions nuh_layer_id may specify that this particular NAL unit belongs to the layer identified by the value of these 6 bits. The next syntax element is "nuh_temporal_id_plus1". The nuh_temporal_id_plus1 minus 1 may specify a temporal identifier for the NAL unit. The variable temporal identifier TemporalId may be specified as TemporalId = nuh_temporal_id_plus1 - 1. The temporal identifier TemporalId is used to identify a temporal sub-layer. The variable HighestTid identifies the highest temporal sub-layer to be decoded.

nal_unit_header( ) {	**Descrip tor**
forbidden_zero_bit	f(1)
nal_unit_type	u(6)
nuh_layer_id	u(6)
nuh_temporal_id_plus1	u(3)
}	

Table (2)

[0073] Referring to FIG. 8A, as previously described the NAL unit header syntax may include two bytes of data, namely, 16 bits. The first bit is a "forbidden_zero_bit" which is always set to zero at the start of a NAL unit. The next six bits is a "nal_unit_type" which specifies the type of raw byte sequence payloads ("RBSP") data structure contained in the NAL unit. The next 6 bits is a "nuh_reserved_zero_6bits". The nuh_reserved_zero_6bits may be equal to 0 in the base specification of the standard. Other values of nuh_reserved_zero_6bits may be specified as desired. Decoders may ignore (i.e., remove from the bitstream and discard) all NAL units with values of nuh_reserved_zero_6bits not equal to 0 when handling a stream based on the base specification of the standard. In a scalable or other extension nuh_reserved_zero_6bits may specify other values, to signal scalable video coding and/or syntax extensions. In some cases syntax element nuh_reserved_zero_6bits may be called reserved_zero_6bits. In some cases the syntax element nuh_reserved_zero_6bits may be called as layer_id_plus1 or layer_id, as illustrated in FIG. 8B and FIG. 8C. In this case the element layer_id will be layer_id_plus1 minus 1. In this case it may be used to signal information related to layer of scalable coded video. The next syntax element is "nuh_temporal_id_plus1". nuh_temporal_id_plus1 minus 1 may specify a temporal identifier for the NAL unit. The variable temporal identifier TemporalId may be specified as TemporalId = nuh_temporal_id_plus1 - 1.

[0074] Referring to FIG. 9, a general NAL unit syntax structure is illustrated. The NAL unit header two byte syntax of FIG. 8 is included in the reference to nal_unit_header() of FIG. 9. The remainder of the NAL unit syntax primarily relates to the RBSP.

[0075] One existing technique for using the "nuh_reserved_zero_6bits" is to signal scalable video coding information by partitioning the 6 bits of the nuh_reserved_zero_6bits into distinct bit fields, namely, one or more of a dependency ID, a quality ID, a view ID, and a depth flag, each of which refers to the identification of a different layer of the scalable coded video. Accordingly, the 6 bits indicate what layer of the scalable encoding technique this particular NAL unit belongs to. Then in a data payload, such as a video parameter set ("VPS") extension syntax ("scalability_type") as illustrated in FIG. 10, the information about the layer is defined. The VPS extension syntax of FIG. 10 includes 4 bits for scalability type (syntax element scalability_type) which specifies the scalability types in use in the coded video sequence and the dimensions signaled through layer_id_plus1 (or layer_id) in the NAL unit header. When the scalability type is equal to 0, the coded video sequence conforms to the base specification, thus layer_id_plus1 of all NAL units is equal to 0 and there are no NAL units belonging to an enhancement layer or view. Higher values of the scalability type are interpreted as illustrated in FIG. 11.

[0076] The layer_id_dim_len[i] specifies the length, in bits, of the i-th scalability dimension ID. The sum of the values layer_id_dim_len[i] for all i values in the range of 0 to 7 is less than or equal to 6. The vps_extension_byte_alignment_reserved_zero_bit is zero. The vps_layer_id[i] specifies the value of layer_id of the i-th layer to which the following layer dependency information applies. The num_direct_ref_layers[i] specifies the number of layers the i-th layer directly depends on. The ref_layer_id[i][j] identifies the j-th layer the i-th layer directly depends on.

[0077] In this manner, the existing technique signals the scalability identifiers in the NAL unit and in the video parameter set to allocate the bits among the scalability types listed in FIG. 11. Then for each scalability type, FIG. 11 defines how many dimensions are supported. For example, scalability type 1 has 2 dimensions (i.e., spatial and quality). For each of the dimensions, the layer_id_dim_len[i] defines the number of bits allocated to each of these two dimensions, where the total sum of all the values of layer_id_dim_len[i] is less than or equal to 6, which is the number of bits in the nuh_reserved_zero_6bits of the NAL unit header. Thus, in combination the technique identifies which types of scalability is in use and how the 6 bits of the NAL unit header are allocated among the scalability.

[0078] As previously described, scalable video coding is a technique of encoding a video bitstream that also contains one or more subset bitstreams. A subset video bitstream may be derived by dropping packets from the larger video to reduce the bandwidth required for the subset bitstream. The subset bitstream may represent a lower spatial resolution (smaller screen), lower temporal resolution (lower frame rate), or lower quality video signal. For example, a video bitstream may include 5 subset bitstreams, where each of the subset bitstreams adds additional content to a base bitstream. Hannuksela, et al., "Test Model for Scalable Extensions of High Efficiency Video Coding (HEVC)" JCTVC-L0453, Shanghai, October 2012, is hereby incorporated by reference herein in its entirety. Chen, et al., "SHVC Draft Text 1," JCTVC-L1008, Geneva, March, 2013, is hereby incorporated by reference herein in its entirety. J. Chen, J. Boyce, Y. Ye, M Hannuksela, SHVC Draft 3, JCTVC-N1008, Vienna, August 2013; and Y. Chen, Y.-K. Wang, A. K. Ramasubromanian, MV-HEVC/SHVC HLS: Cross-layer POC Alignment, JCTVC-N0244, Vienna, July 2013; each of which is incorporated by reference herein in its entirety.

[0079] As previously described, multi-view video coding is a technique of encoding a video bitstream that also contains one or more other bitstreams representative of alternative views. For example, the multiple views may be a pair of views for stereoscopic video. For example, the multiple views may represent multiple views of the same scene from different viewpoints. The multiple views generally contain a large amount of inter-view statistical dependencies, since the images are of the same scene from different viewpoints. Therefore, combined temporal and inter-view prediction may achieve efficient multi-view encoding. For example, a frame may be efficiently predicted not only from temporally related frames, but also from the frames of neighboring viewpoints. Hannuksela, et al., "Common specification text for scalable and multi-view extensions," JCTVC-L0452, Geneva, January 2013, is hereby incorporated by reference herein in its entirety. Tech, et. al. "MV-HEVC Draft Text 3 (ISO/IEC 23008-2:201x/PDAM2)," JCT3V-C1004_d3, Geneva, January 2013, is hereby incorporated by reference herein in its entirety. G. Tech, K. Wegner, Y. Chen, M. Hannuksela, J. Boyce, "MV-HEVC Draft Text 5 (ISO/IEC 203008-2:201x/PDAM2), JCTVC-E1004, Vienna, August 2013, is hereby incorporated by reference herein in its entirety.

[0080] Chen, et al., "SHVC Draft Text 1," JCTVC-L1008, Geneva, January 2013; Hannuksela, et al. "Test Model for Scalable Extensions of High Efficiency Video Coding (HEVC)," JCTVC-L0453-spec-text, Shanghai, October 2012; and Hannuksela, "Draft Text for Multiview Extension of High Efficiency Video Coding (HEVC)," JCTVC-L0452-spec-text-r1, Shanghai, October 2012; each of which is incorporated by reference herein in its entirety, each have an output order decoded picture buffer (DPB) which operates based on using sps_max_num_reorder_pics[HighestTid], sps_max_latency_increase_plus1[HighestTid] and sps_max_dec_pic_buffering[HighestTid] syntax elements for the output and removal of pictures 0 from the DPB. This information is signaled in the video parameter set for the base layer, which provides buffering information for the video content including the enhancement layers, if any.

[0081] Referring to FIG. 12, when coding scalable high efficiency coding ("SVHC") the base layer may include one or more SPS and may also include one or more PPS. Also, each enhancement layer may include one or more SPS and may also include one or more PPS. In FIG. 12 SPS+ indicates one or more SPS and PPS+ indicates one or more PPS being signaled for a particular base or enhancement layer. In this manner, for a video bitstream having both a base layer and one or more enhancement layers, the collective number of SPS and PPS data sets becomes significant together with the required bandwidth to transmit such data, which tends to be limited in many applications. With such bandwidth limitations, it is desirable to limit the data that needs to be transmitted, and locate the data in the bitstream in an effective manner. Each layer may have one SPS and/or PPS that is activate at any particular time, and may select a different active SPS and/or PPS, as desired.

[0082] An input picture may comprise a plurality of coded tree blocks (e.g., generally referred to herein as blocks) may be partitioned into one or several slices. The values of the samples in the area of the picture that a slice represents may be properly decoded without the use of data from other slices provided that the reference pictures used at the encoder and the decoder are the same and that de-blocking filtering does not use information across slice boundaries. Therefore, entropy decoding and block reconstruction for a slice does not depend on other slices. In particular, the entropy coding state may be reset at the start of each slice. The data in other slices may be marked as unavailable when defining neighborhood availability for both entropy decoding and reconstruction. The slices may be entropy decoded and reconstructed in parallel. No intra prediction and motion-vector prediction is preferably allowed across the boundary of a slice. In contrast, de-blocking filtering may use information across slice boundaries.

[0083] FIG. 13 illustrates an exemplary video picture 2090 comprising eleven blocks in the horizontal direction and nine blocks in the vertical direction (nine exemplary blocks labeled 2091-2099). FIG. 13 illustrates three exemplary slices: a first slice denoted "SLICE #0" 2080, a second slice denoted "SLICE #1" 2081 and a third slice denoted "SLICE #2" 2082. The decoder may decode and reconstruct the three slices 2080, 2081, 2082 in parallel. Each of the slices may be transmitted in scan line order in a sequential manner. At the beginning of the decoding/reconstruction process for each slice, context models are initialized or reset and blocks in other slices are marked as unavailable for both entropy decoding and block reconstruction. The context model generally represents the state of the entropy encoder and/or decoder. Thus, for a block, for example, the block labeled 2093, in "SLICE #1," blocks (for example, blocks labeled 2091 and 2092) in "SLICE #0" may not be used for context model selection or reconstruction. Whereas, for a block, for example, the block labeled 2095, in "SLICE

#1," other blocks (for example, blocks labeled 2093 and 2094) in "SLICE #1" may be used for context model selection or reconstruction. Therefore, entropy decoding and block reconstruction proceeds serially within a slice. Unless slices are defined using a flexible block ordering (FMO), blocks within a slice are processed in the order of a raster scan.

[0084] Flexible block ordering defines a slice group to modify how a picture is partitioned into slices. The blocks in a slice group are defined by a block-to-slice-group map, which is signaled by the content of the picture parameter set and additional information in the slice headers. The block-to-slice-group map consists of a slice-group identification number for each block in the picture. The slice-group identification number specifies to which slice group the associated block belongs. Each slice group may be partitioned into one or more slices, wherein a slice is a sequence of blocks within the same slice group that is processed in the order of a raster scan within the set of blocks of a particular slice group. Entropy decoding and block reconstruction proceeds serially within a slice group.

[0085] FIG. 14 depicts an exemplary block allocation into three slice groups: a first slice group denoted "SLICE GROUP #0" 2083, a second slice group denoted "SLICE GROUP #1" 2084 and a third slice group denoted "SLICE GROUP #2" 2085. These slice groups 2083, 2084, 2085 may be associated with two foreground regions and a background region, respectively, in the picture 2090.

[0086] The arrangement of slices, as illustrated in FIG. 14, may be limited to defining each slice between a pair of blocks in the image scan order, also known as raster scan or a raster scan order. This arrangement of scan order slices is computationally efficient but does not tend to lend itself to the highly efficient parallel encoding and decoding. Moreover, this scan order definition of slices also does not tend to group smaller localized regions of the image together that are likely to have common characteristics highly suitable for coding efficiency. The arrangement of slices 2083, 2084, 2085, as illustrated in FIG. 14, is highly flexible in its arrangement but does not tend to lend itself to high efficient parallel encoding or decoding. Moreover, this highly flexible definition of slices is computationally complex to implement in a decoder.

[0087] Referring to FIG. 15, a tile technique divides an image into a set of rectangular (inclusive of square) regions. The blocks (alternatively referred to as largest coding units or coded treeblocks in some systems) within each of the tiles are encoded and decoded in a raster scan order. The arrangement of tiles are likewise encoded and decoded in a raster scan order. Accordingly, there may be any suitable number of column boundaries (e.g., 0 or more) and there may be any suitable number of row boundaries (e.g., 0 or more). Thus, the frame may define one or more slices, such as the one slice illustrated in FIG. 15. In some embodiments, blocks located in different tiles are not available for intra-prediction, motion compensation, entropy coding context selection or other processes that rely on neighboring block information.

[0088] Referring to FIG. 16, the tile technique is shown dividing an image into a set of three rectangular columns. The blocks (alternatively referred to as largest coding units or coded treeblocks in some systems) within each of the tiles are encoded and decoded in a raster scan order. The tiles are likewise encoded and decoded in a raster scan order. One or more slices may be defined in the scan order of the tiles. Each of the slices are independently decodable. For example, slice 1 may be defined as including blocks 1-9, slice 2 may be defined as including blocks 10-28, and slice 3 may be defined as including blocks 29-126 which spans three tiles. The use of tiles facilitates coding efficiency by processing data in more localized regions of a frame.

[0089] Referring to FIG. 17, the base layer and the enhancement layers may each include tiles which each collectively form a picture or a portion thereof. The coded pictures from the base layer and one or more enhancement layers may collectively form an access unit. The access unit may be defined as a set of NAL units that are associated with each other according to a specified classification rule, are consecutive in decoding order, and/or contain the VCL NAL units of all coded pictures associated with the same output time (picture order count or otherwise) and their associated non-VCL NAL units. The VCL NAL is the video coding layer of the network abstraction layer. Similarly, the coded picture may be defined as a coded representation of a picture comprising VCL NAL units with a particular value of nuh_layer_id within an access unit and containing all coding tree units of the picture. Additional descriptions are described in B. Bros, W-J. Han, J-R. Ohm, G. J. Sullivan, and T. Wiegand, "High efficiency video coding (HEVC) text specification draft 10," JCTVC-L1003, Geneva, January 2013; J. Chen, J. Boyce, Y. Ye, M.M. Hannuksela, "SHVC Draft Text 2," JCTVC-M1008, Incheon, May 2013; G. Tech, K. Wegner, Y. Chen, M. Hannuksela, J. Boyce, "MV-HEVC Draft Text 4 (ISO/IEC 23008-2:201x/PDAM2)," JCTVC-D1004, Incheon, May 2013; each of which is incorporated by reference herein in its entirety.

[0090] Referring to FIGS. 18A-18D, each slice may include a slice segment header. In some cases a slice segment header may be called slice header. Within the slice segment header there includes syntax elements that are used for inter-layer prediction. This inter-layer prediction defines what other layers the slice may depend upon. In other words this inter-layer prediction defines what other layers the slice may use as its reference layers. The reference layers may be used for sample prediction and / or for motion filed prediction. Referring to FIG. 19 by way of example, enhancement layer 3 may depend upon enhancement layer 2, and base layer 0. This dependency relationship may be expressed in the form of a list, such as, [2, 0].

[0091] The NumDirectRefLayers for a layer may be derived based upon a direct_dependency_flag[i][j] that when equal to 0 specifies that the layer with index j is not a direct reference layer for the layer with index i. The direct_dependency_flag[i][j] equal to 1 specifies that the layer with index j may be a direct reference layer for the layer with index i. When the direct_dependency_flag[i][j] is not present for i and j in the range of 0 to vps_max_layers_minus1, it is inferred to be equal to 0.

[0092] The direct_dep_type_len_minus2 plus 2 specifies the number of bits of the direct_dependency_type[i][j] syntax element. In bitstreams conforming to this version of this Specification the value of direct_dep_type_len_minus2 shall be equal 0. Although the value of direct_dep_type_len_minus2 shall be equal to 0 in this version of this Specification, decoders shall allow other values of direct_dep_type_len_minus2 in the range of 0 to 30, inclusive, to appear in the syntax.

[0093] The direct_dependency_type[i][j] is used to derive the variables NumSamplePredRefLayers[i], NumMotionPredRefLayers[i], SamplePredEnabledFlag[i][j], and MotionPredEnabledFlag[i][j]. direct_dependency_type[i][j] shall be in the range of 0 to 2, inclusive, in bitstreams conforming to this version of this Specification. Although the value of direct_dependency_type[i][j] shall be in the range of 0 to 2, inclusive, in this version of this Specification, decoders shall allow values of direct_dependency_type[i][j] in the range of 3 to 2^{32}-2, inclusive, to appear in the syntax.

[0094] The variables NumSamplePredRefLayers[i], NumMotionPredRefLayers[i], SamplePredEnabledFlag[i][j], MotionPredEnabledFlag[i][j], NumDirectRefLayers[i], DirectRefLayerIdx[i][j], RefLayerId[i][j], MotionPredRefLayerId[i][j], and SamplePredRefLayerId[i][j] are derived as follows:

```
      for( i = 0; i < 64; i++ ) {
         NumSamplePredRefLayers[ i ] = 0
         NumMotionPredRefLayers[ i ] = 0
         NumDirectRefLayers[ i ] = 0
         for( j = 0; j < 64; j++ ) {
               SamplePredEnabledFlag[ i ][ j ] = 0
               MotionPredEnabledFlag[ i ][ j ] = 0
               RefLayerId[ i ][ j ] = 0
               SamplePredRefLayerId[ i ][ j ] = 0
               MotionPredRefLayerId[ i ][ j ] = 0
         }
}
   for( i = 1; i  <=  vps_max_layers_minus1; i++ ) {
      iNuhLId = layer_id_in_nuh[ i ]
      for( j = 0; j < i; j++ )
         if( direct_dependency_flag[ i ][ j ] ) {
                    DirectRefLayerIdx[ iNuhLid ][ layer_id_in_nuh[ j ] ] =
NumDirectRefLayers[ iNuhLId ]
                    RefLayerId[ iNuhLId ][ NumDirectRefLayers[ iNuhLId ]++ ] =
layer_id_in_nuh[ j ]
                          SamplePredEnabledFlag[ iNuhLId ][ j ] =
( ( direct_dependency_type[ i ][ j ] + 1 ) & 1 )
                          NumSamplePredRefLayers[ iNuhLId ] +=
SamplePredEnabledFlag[ iNuhLId ][ j ]
                          MotionPredEnabledFlag[ iNuhLId ][ j ] =
( ( ( direct_dependency_type[ i ][ j ] + 1 ) & 2 ) >> 1 )
                          NumMotionPredRefLayers[ iNuhLId ] +=
MotionPredEnabledFlag[ iNuhLId ][ j ]
                    }
}
for( i = 1, mIdx = 0, sIdx = 0; i <= vps_max_layers_minus1; i++ ) {
               iNuhLId = layer_id_in_nuh[ i ]
               for( j = 0, j < i; j++ ) {
                  if( MotionPredEnabledFlag[ iNuhLId ][ j ] )
                        MotionPredRefLayerId[ iNuhLId ][ mIdx++ ] =
layer_id_in_nuh[ j ]
                  if( SamplePredEnabledFlag[ INuhLid ][ j ] )
                        SamplePredRefLayerId[ iNuhLid ][ sIdx++ ] =
layer_id_in_nuh[ j ]
               }
}
```

[0095] The direct_dependency_flag[i][j], direct_dep_type_len_minus2, direct_dependency_type[i][j] are included in the vps_extension syntax illustrated in FIG. 20A and FIG. 20B, which is included by reference in the VPS syntax which provides syntax for the coded video sequence.

[0096] It is typically desirable to reduce the number of referenced layers that need to be signaled within the bitstream, and other syntax elements within the slice segment header may be used to effectuate such a reduction. The other syntax elements may include inter_layer_pred_enabled_flag, num_inter_layer_ref_pics_minus1, and/or inter_layer_pred_layer_idc[i]. These syntax elements may be signaled in slice segment header.

[0097] The inter_layer_pred_enabled_flag equal to 1 specifies that inter-layer prediction may be used in decoding of the current picture. The inter_layer_pred_enabled_flag equal to 0 specifies that inter-layer prediction is not used in decoding of the current picture. When not present, the value of inter_layer_pred_enabled_flag is inferred to be equal to 0.

[0098] The num_inter_layer_ref_pics_minus1 plus 1 specifies the number of pictures that may be used in decoding of the current picture for inter-layer prediction. The length of the num_inter_layer_ref_pics_minus1 syntax element is Ceil(Log2(NumDirectRefLayers[nuh_layer_id])) bits. The value of num_inter_layer_ref_pics_minus1 shall be in the range of 0 to NumDirectRefLayers[nuh_layer_id] - 1, inclusive.

[0099] The variable NumActiveRefLayerPics is derived as follows:

```
if( nuh_layer_id  = =  0  | |  NumDirectRefLayers[ nuh_layer_id ]  = =  0
| |   !inter_layer_pred_enabled_flag )
    NumActiveRefLayerPics = 0
else if( max_one_active_ref_layer_flag  | |
NumDirectRefLayers[ nuh_layer_id ]  = = 1 )
    NumActiveRefLayerPics = 1
else
    NumActiveRefLayerPics = num_inter_layer_ref_pics_minus1 + 1
```

All slices of a coded picture shall have the same value of NumActiveRefLayerPics.

[0100] The inter_layer_pred_layer_idc[i] specifies the variable, RefPicLayerId[i], representing the nuh_layer_id of the i-th picture that may be used by the current picture for inter-layer prediction. The length of the syntax element inter_layer_pred_layer_idc[i] is Ceil(Log2(NumDirectRefLayers[nuh_layer_id])) bits. The value of inter_layer_pred_layer_idc[i] may be in the range of 0 to NumDirectRefLayers[nuh_layer_id] - 1, inclusive. When not present, the value of inter_layer_pred_layer_idc[i] is inferred to be equal to 0.

[0101] By way of example, the system may signal various syntax elements especially the direct_dependency_flag[i][j] in VPS which results in the inter-layer reference picture set for layer 3 to be [2, 0],. Then the system may refine further the inter-layer reference picture set with the use of the additional syntax elements for example syntax elements in slice segment header as [2], may refine further the inter-layer reference picture set with the use of the additional syntax elements as [0], or may refine further the inter-layer reference picture set with the use of the additional syntax elements as [ ] which is the null set. However, depending on the design of the encoder, the reference picture set of [2, 0] may be signaled as [2, 0]..

[0102] Referring to FIG. 21, the slice segment header may be modified to include a comparison between the number of direct reference layers for a particular layer (NumDirectRefLayers[num_layer_id] in the syntax) and the number of active reference layers for the same particular layer (NumActiveRefLayerPics in the syntax). In particular, this may be signaled as "if(NumActiveRefLayerPics!=NumDirectRefLayers[nuh_layer_id])". Thus, if both of these indicate the same number of layers, then there is no need to signal inter_layer_pred_layer_idc[i] in the bitstream, but may rather determine/ infer such values based on other syntax elements already signaled.

[0103] Referring to FIG. 22, the slice segment header signalling may be modified in a similar manner to FIG. 21 to infer the values for the inter_layer_pred_layer_idc[i] by not signalling them.

[0104] If NumActiveRefLayerPics is equal to NumDirectRefLayers[nuh_layer_id], then the value of inter_layer_pred_layer_idc[i] may be inferred as follows.

```
for( i = 0; i < NumActiveRefLayerPics; i++)
        inter_layer_pred_layer_idc[ i ] = i;
```

[0105] When not present and when NumActiveRefLayerPics is not equal to NumDirectRefLayers[nuh_layer_id], the value of inter_layer_pred_layer_idc[i] is inferred to be equal to 0.

[0106] When i is greater than 0, inter_layer_pred_layer_idc[i] may be greater than inter_layer_pred_layer_idc[i - 1].

[0107] The variables RefPicLayerId[i] for each value of i in the range of 0 to NumActiveRefLayerPics - 1, inclusive, NumActiveMotionPredRefLayers, and ActiveMotionPredRefLayerId[j] for each value of j in the range of 0 to NumActiveMotionPredRefLayers - 1, inclusive, maybe derived as follows:

```
        for( i = 0, j = 0; i < NumActiveRefLayerPics; i++)
                RefPicLayerId[ i ] =
RefLayerId[ nuh_layer_id ][ inter_layer_pred_layer_idc[ i ] ]

        if( MotionPredEnabledFlag[ nuh_layer_id ][ inter_layer_pred_layer_idc[ i ] ]
)
                        ActiveMotionPredRefLayerId[ j++ ] =
RefLayerId[ nuh_layer_id ][ inter_layer_pred_layer_idc[ i ] ]
        }
        NumActiveMotionPredRefLayers = j
```

[0108] All slices of a picture may have the same value of inter_layer_pred_layer_idc[i] for each value of i in the range of 0 to NumActiveRefLayerPics - 1, inclusive.

[0109] The max_tid_il_ref_pics_plus1[i] is signaled in VPS extension. max_tid_il_ref_pics_plus1[i] equal to 0 specifies that within the CVS non-IRAP pictures with nuh_layer_id equal to layer_id_in_nuh[i] are not used as reference for inter-layer prediction. max_tid_il_ref_pics_plus1[i] greater than 0 specifies that within the CVS pictures with nuh_layer_id equal to layer_id_in_nuh[i] and TemporalId greater than max_tid_il_ref_pics_plus1[i] - 1 are not used as reference for inter-layer prediction. When not present, max_tid_il_ref_pics_plus1[i] is unspecified.

[0110] It may be a requirement of bitstream conformance that for each value of i in the range of 0 to NumActiveRefLayerPics - 1, inclusive, either of the following two conditions may be true:

The value of max_tid_il_ref_pics_plus1[LayerIdxInVps[RefPicLayerId[i]]] is greater than TemporalId.

The values of max_tid_il_ref_pics_plus1[LayerIdxInVps[RefPicLayerId[i]]] and TemporalId are both equal to 0 and the picture in the current access unit with nuh_layer_id equal to RefPicLayerId[i] is an IRAP picture.

[0111] In another embodiment It may be a requirement of bitstream conformance that for each value of i in the range of 0 to NumActiveRefLayerPics - 1, inclusive, either of the following two conditions may be true:

The value of max_tid_il_ref_pics_plus1[LayerIdxInVps[RefPicLayerId[i]]] is greater than TemporalId of the picture in the current access unit with nuh_layer_id equal to RefPicLayerId[i].

The values of max_tid_il_ref_pics_plus1[LayerIdxInVps[RefPicLayerId[i]]] is equal to 0 and the picture in the current access unit with nuh_layer_id equal to RefPicLayerId[i] is an IRAP picture.

[0112] It may be a requirement of bitstream conformance that for each value of i in the range of 0 to NumActiveRefLayerPics - 1, inclusive, the value of SamplePredEnabledFlag[nuh_layer_id] [RefPicLayerId[i]] or MotionPredEnabledFlag[nuh_layer_id] [RefPicLayerId[i]] shall be equal to 1.

[0113] Referring to FIG. 23, another embodiment for signaling slice segment header is illustrated.

[0114] For the embodiment illustrated in FIG. 23, an inter_layer_pred_layer_mask[i] equal to 1 specifies that layer RefLayerId[nuh_layer_id][i], may be used by the current picture for inter-layer prediction. The inter_layer_pred_layer_mask[i] equal to 0 specifies that layer RefLayerId[nuh_layer_id][i], is not used by the current picture for inter-layer prediction.

[0115] When not present the value of inter_layer_pred_layer_mask [i] is inferred to be equal to 0.

[0116] The variables RefPicLayerId[i] for each value of i in the range of 0 to NumActiveRefLayerPics - 1, inclusive, NumActiveMotionPredRefLayers, and ActiveMotionPredRefLayerId[j] for each value of j in the range of 0 to NumActiveMotionPredRefLayers - 1, inclusive, are derived as follows:

```
for( i = 0, j = 0, k=0; i < NumDirectRefLayers[ nuh_layer_id ]; i++)
        if(inter_layer_pred_layer_mask[ i ])
                RefPicLayerId[ k++ ] = RefLayerId[ nuh_layer_id ][ i ]
        if( MotionPredEnabledFlag[ nuh_layer_id ][ i ] )
                ActiveMotionPredRefLayerId[ j++ ]=
RefLayerId[ nuh_layer_id ][ i ]
}
NumActiveMotionPredRefLayers = j
```

[0117] All slices of a picture may have the same value of inter_layer_pred_layer_mask[i] for each value of i in the range of 0 to NumDirectRefLayers[nuh_layer_id] - 1, inclusive.

[0118] It may be a requirement of bitstream conformance that for each value of i in the range of 0 to NumActiveRefLayerPics - 1, inclusive, either of the following two conditions shall be true:

The value of max_tid_il_ref_pics_plus1[LayerIdxInVps[RefPicLayerId[i]]] is greater than TemporalId.

The values of max_tid_il_ref_pics_plus1[LayerIdxInVps[RefPicLayerId[i]]] and TemporalId are both equal to 0 and the picture in the current access unit with nuh_layer_id equal to RefPicLayerId[i] is an IRAP picture.

[0119] It may be a requirement of bitstream conformance that for each value of i in the range of 0 to NumActiveRefLayerPics - 1, inclusive, the value of SamplePredEnabledFlag[nuh_layer_id] [RefPicLayerId[i]] or MotionPredEnabledFlag[nuh_layer_id] [RefPicLayerId[i]] may be equal to 1.

[0120] It is shown in FIG. 23 that the inter_layer_pred_layer_mask[i] may be signed with u(1) which uses 1 bit, and FIG. 22 which signals inter_layer_pred_layer_idc[i] may be signed with u(v) which may use multiple bits. In an embodiment inter_layer_pred_layer_mask[i] is signaled instead of intra_layer_pred_idc[i]

[0121] Referring to FIG. 24, it is desirable to define profiles where the complexity of the system is reduced by limiting the permitted referencing interrelationships between the different layers (e.g., base layer and/ enhancement layers). In general, the syntax structure permits one layer to reference multiple other layers, which results in a relatively high decoder complexity and also high encoder complexity. If desired, a modified syntax structure may be used for profiles of a reduced complexity where the syntax structure permits one layer to reference at most only one other layer. This limitation on the syntax structure may be signaled by setting a max_one_active_ref_layer_flag being set to 1.

[0122] The max_one_active_ref_layer_flag is signaled in VPS extension. max_one_active_ref_layer_flag equal to 1 specifies that at most one picture is used for inter-layer prediction for each picture in the CVS. max_one_active_ref_layer_flag equal to 0 specifies that more than one picture may be used for inter-layer prediction for each picture in the CVS.

[0123] The layer_id_in_nuh[i] is signaled in VPS extension. layer_id_in_nuh[i] specifies the value of the nuh_layer_id syntax element in VCL NAL units of the i-th layer. For i in a range from 0 to vps_max_layers_minus1, inclusive, when not present, the value of layer_id_in_nuh[i] is inferred to be equal to i. When i is greater than 0, layer_id_in_nuh[i] shall be greater than layer_id_in_nuh[i - 1].

[0124] A bitstream constraint may be included in the case where only one direct reference layer for a layer is used or at most one picture is used for inter-layer prediction for each picture in CVS, such as follows:

In one choice, it may a requirement of the bitstream conformance that if NumDirectRefLayers[layer_id_in_nuh[i]] is equal to 1 for each layer i=1,...vps_max_layers_minus1 then max_one_active_ref_layer_flag is equal to 1.

```
In another choice,
Let
for(i=1;i<=vps_max_layers_minus1,i++)
        for(j=0,NumDirDepFlags[i]=0;j<i;j++)
                NumDirDepFlags[i]+=direct_dependency_flag[i][j];
```

It may be a requirement of the bitstream conformance that if NumDirDepFlags[i] is equal to 1 for each layer i=1,...vps_max_layers_minus1 then max_one_active_ref_layer_flag is equal to 1.

[0125] In another embodiment, it is desirable to not support the ability to signal an inter-layer reference picture from different direct dependent layers for each picture when max_one_active_ref_layer_flag is set equal to 1. This embodiment results in lower complexity for decoding an output layer set. In this embodiment the bitstream constraint proposed below related to NumDirectRefLayers being equal to 1 may be required to be obeyed:

In one choice, it is a requirement of the bitstream conformance that if max_one_active_ref_layer_flag is equal to 1 then NumDirectRefLayers[layer_id_in_nuh[i]] is equal to 1 for each layer i=1,...vps_max_layers_minus1.

```
In another choice,
let
for(i=1;i<=vps_max_layers_minus1,i++)
        for(j=0,NumDirDepFlags[i]=0;j<i;j++)
                NumDirDepFlags[i]+=direct_dependency_flag[i][j];
```

It may be a requirement of the bitstream conformance that if max_one_active_ref_layer_flag is equal to 1 then NumDirDepFlags[i] is equal to 1 for i=1,...vps_max_layers_minus1.

[0126] Another embodiment may include a gating flag controlled in a parameter set (e.g. pps, sps, and/or vps) to conditionally signal selected syntax elements in the slice header related to inter-layer prediction signalling.

[0127] Referring to FIG. 25, for example, the syntax elements inter_layer_pred_enabled_flag, num_inter_layer_ref_pics_minus1, and/or inter_layer_pred_layer_idc[i] are signaled in slice segment header only if a ilp_slice_signaling_enabled_flag is equal to 1. Thus ilp_slice_signaling_enabled_flag is a gating flag.

[0128] Referring to FIG. 26A, and FIG. 26B the ilp_slice_signaling_enabled_flag may be signaled in a parameter set such as in video parameter set. Referring to FIG. 27, the ilp_slice_signaling_enabled_flag may be signaled in a parameter set such as in sequence parameter set. Referring to FIG. 28, the ilp_slice_signaling_enabled_flag may be signaled in a parameter set such as in the picture parameter set. The ilp_slice_signaling_enabled_flag may be signaled in another location of the bitstream, as desired. In each of these parameters sets the ilp_slice_signaling_enabled_flag may be sent in any location different than that shown in that illustrated.

[0129] The ilp_slice_signaling_enabled_flag equal to 1 specifies that inter_layer_pred_enabled_flag, num_inter_layer_ref_pics_minus1, inter_layer_pred_layer_idc[i] are present in the slice segment headers. ilp_slice_signaling_enabled_flag equal to 0 specifies that inter_layer_pred_enabled_flag, num_inter_layer_ref_pics_minus1, inter_layer_pred_layer_idc[i] are not present in the slice segment header.

In some embodiments ilp_slice_signaling_enabled_flag may be instead called ilp_slice_signaling_present_flag.

[0130] When ilp_slice_signaling_enabled_flag is equal to 1 inter_layer_pred_enabled_flag, num_inter_layer_ref_pics_minus1, inter_layer_pred_layer_idc[i] and NumActiveRefLayersPics values are inferred as follows:

NumActiveRefLayerPics is inferred as follows:

NumActiveRefLayerPics = NumDirectRefLayers[nuh_layer_id]

inter_layer_pred_layer_idc[i] is inferred as follows:

```
for( i = 0; i < NumActiveRefLayerPics; i++)
    inter_layer_pred_layer_idc[ i ] = i;
```

num_inter_layer_ref_pics_minus1 is inferred to be equal to NumDirectRefLayers[nuh_layer_id] -1.

inter_layer_pred_enabled_flag is inferred to be equal to 1.

[0131] In another embodiment one or more of the syntax elements may be signaled using a known fixed number of bits instead of u(v) instead of ue(v). For example they could be signaled using u(8) or u(16) or u(32) or u(64), etc.

[0132] In another embodiment one or more of these syntax element could be signaled with ue(v) or some other coding scheme instead of fixed number of bits such as u(v) coding.

[0133] In another embodiment the names of various syntax elements and their semantics may be altered by adding a plus1 or plus2 or by subtracting a minus1 or a minus2 compared to the described syntax and semantics.

[0134] In yet another embodiment various syntax elements may be signaled per picture anywhere in the bitstream. For example they may be signaled in slice segment header, pps/ sps/ vps/ or any other parameter set or other normative part of the bitstream.

[0135] Referring to FIG. 29, the video may include temporal sub-layer support specified by a temporal identifier in the NAL unit header, which indicates a level in a hierarchical temporal prediction structure. The number of decoded temporal sublayers can be adjusted during the decoding process of one coded video sequence. Different layers may have different number of sub-layers. For example, in FIG. 29 the base layer may include 3 temporal sub-layers, namely, TemporalId 0, TemporalId 1, TemporalId 2. For example, the enhancement layer 1 may include 4 temporal sub-layers, namely, TemporalId 0, TemporalId 1, TemporalId 2, and TemporalId 3. The access unit may be defined as a set of NAL units that are associated with each other according to a specified classification rule, are consecutive in decoding order, and/or contain the VCL NAL units of all coded pictures associated with the same output time (picture order count or otherwise) and their associated non-VCL NAL units.

In FIG. 29 base layer has a lower overall frame rate compared to the enhancement layer 1. For example the frame rate of the base layer may be 30 Hz or 30 frames per second. The frame rate of the enhancement layer 1 may be 60 Hz or 60 frames per second. In FIG. 29 at some output times an access unit may contain a coded picture of base layer and a coded picture of enhancement layer 1 (e.g. access unit Y in FIG. 29). In FIG. 29 at some output times an access unit may contain only a coded picture of enhancement layer 1 (e.g. access unit X in FIG. 29).

[0136] As previously described, the dependency of one layer on one or more other layers may be signaled in the VPS for a sequence. In addition at each slice within a respective layer, the slice segment header syntax permits a further refinement of this dependency by removing one or more of the dependencies for the respective slice. For example, the layer dependency in the VPS may indicate that layer 3 is dependent on layer 2 and base layer 0. For example, a slice in layer 3 may further modify this dependency to remove the dependency on layer 2.

[0137] Referring to FIGS. 30A-30D, a slice segment header (slice_segment_header), includes a syntax structure that facilitates the identification of dependencies, a portion of which is excerpted below.

if(nuh_layer_id > 0 && all_ref_layers_active_flag && NumDirectRefLayers[nuh_layer_id] > 0) {	
inter_layer_pred_enabled_flag	u(1)
if(inter_layer_pred_enabled_flag && NumDirectRefLayers[nuh_layer_id] > 1) {	
if(!max_one_active_ref_layer_flag)	
num_inter_layer_ref_pics_minus1	u(v)
if(NumActiveRefLayerPics != NumDirectRefLayers[nuh_layer_id])	
for(i = 0; i < NumActiveRefLayerPics; i++)	
inter_layer_pred_layer_idc[i]	u(v)
}	
}	

[0138] In an example case a base layer has coded pictures at a rate of 30 hertz and an enhancement layer has coded pictures at a rate of 60 hertz, where every other coded picture of the enhancement layer are not aligned with the coded pictures of the base layer. This scenarios is similar to the FIG. 29. Also, it is noted that in general each coded picture of the enhancement layer may not include a corresponding coded picture in the base layer. In some cases, there may be some corresponding coded pictures in the base layer with coded pictures of the enhancement layer. Unfortunately, this syntax structure does not permit discrimination between the case where a coded picture of the base layer is not present in an access unit in the original bitstream (e.g. access unit X in FIG. 29) and the case where a coded picture of the base layer was present in an access unit in the original bitstream but has been lost during transmission. In this manner, the decoder does not know if the coded picture of the base layer has been lost (i.e. a lost picture) or whether there was no coded picture of the base layer in the first place (i.e. a non-existing base layer picture).

[0139] It was determined that even with the syntax illustrated in FIGS. 30A-30D, there are conditions where the system can not signal the removal of a layer in the slice segment header. Under such conditions the decoder is not able to distinguish between the case that an AU had no coded picture for a direct reference layer of a current layer due to that picture not existing in the bitstream (due to the reference layer having different frame rate) versus the case that the coded picture for the direct reference layer of a current layer was lost during transmission. The particular conditions include three conditions, namely, when max_one_active_ref_layer_flag is equal to 1, NumDirectRefLayers[nuh_layer_id] is equal to 1, and/ or all_ref_layers_active_flag is equal to 1. For each of these conditions a "No reference picture" would be inferred during the decoding process for the inter-layer reference picture set even when base layer (i.e. reference layer) did not have a picture in the original bitstream. This is incorrect and no-optimal behavior. In some cases in this scenario an unavailable reference picture would be regenerated for such a "no reference picture" and would be used as the base layer (i.e. reference layer) picture thus resulting in incorrect operation.

[0140] To alleviate this limitation, it was determined that it is desirable to signal the maximum number of temporal sub-layers for each layer in the SHVC and/or MV-HEVC. This signaling may be achieved in any suitable manner. A first technique for signaling the maximum number of temporal sub-layers for each layer is by always explicitly signaling the maximum number for each layer. A second technique for signaling the maximum number of temporal sub-layers for each layer is signaled conditioned on a presence flag. In a third technique for signaling the maximum number of temporal sub-layers for each layer is coded predictively with respect to the maximum number of temporal sub-layers for the previous layer by conditioning them on a presence flag. Also, the semantics of the slice segment header syntax elements num_inter_layer_ref_pics_minus1 and inter_layer_pred_layer_idc[i] and the derivation of NumActiveRefLayerPics may be modified based upon the signaling of the temporal sub-layer information for each layer. Additionally, or alternatively a layer_present_in_au_flag[i] may be signaled for NumActiveRefLayerPics in the slice segment header, to similarly disambiguate between lost picture case and non-existing picture case.

[0141] In HEVC (JCTVC-L1003), SHVC (JCTVC-N1008) and MV-HEVC (JCT3V-E1004) it is required that:

-The value of TemporalId shall be the same for all VCL NAL units of an access unit.

-The value of TemporalId of an access unit is the value of the TemporalId of the VCL NAL units of the access unit.

[0142] Referring to FIG. 31, a modified vps_expension() syntax may include explicitly signaling the maximum number temporal sub-layers that may be present for each layer, as opposed to the bitstream as a whole. In this manner, two different layers may each have a different maximum number of temporal sublayers. In particular the sub_layers_vps_max_minus1[i] plus 1 specifies the maximum number of temporal sub-layers that may be present in the CVS for layer with nuh_layer_id equal to layer_id_in_nuh[i]. The value of sub_layers_vps_max_minus1[i] shall be in the range of 0 to vps_max_sub_layers_minus1 inclusive. When not present sub_layers_vps_max_minus1[i] shall be equal to vps_max_sub_layers_minus1. Alternatively, the value of sub_layers_vps_max_minus1[i] may be in the range of 0 to 6 inclusive. Alternatively, the value of sub_layers_vps_max_minus1[i] may only be signaled for the enhancement layers in the VPS extension as illustrated in FIG. 32.

[0143] Referring to FIG. 33, a modified vps_expension() syntax may include signaling the maximum number for each layer conditioned on a presence flag. In this manner, two different layers may each have a different maximum number of temporal sublayers. In particular the sub_layers_vps_max_minus1_present_flag equal to 1 specifies that the syntax elements sub_layers_vps_max_minus1[i] are present. The sub_layers_vps_max_minus1_present_flag equal to 0 specifies that the syntax elements sub_layers_vps_max_minus1[i] are not present. The sub_layers_vps_max_minus1[i] plus 1 specifies the maximum number of temporal sub-layers that may be present in the CVS for layer with nuh_layer_id equal to layer_id_in_nuh[i]. The value of sub_layers_vps_max_minus1[i] shall be in the range of 0 to vps_max_sub_layers_minus1 inclusive. When not present sub_layers_vps_max_minus1[i] shall be equal to vps_max_sub_layers_minus1. Alternatively, the value of sub_layers_vps_max_minus1[i] may be in the range of 0 to 6 inclusive. Alternatively, the value of sub_layers_vps_max_minus1[i] may only be signaled for the enhancement layers in the VPS extension as illustrated in FIG. 34.

Referring to FIG. 35, a modified vps_expension() syntax may include signaling the maximum number of temporal sub-layers for each layer by coding them predictively with respect to the maximum number of temporal sub-layers for the previous layer by conditioning them on a presence flag. In this manner, two different layers may each have a different maximum number of temporal sublayers. In particular the sub_layers_vps_max_minus1_predict_flag[i] equal to 1 specifies that sub_layers_vps_max_minus1[i] is inferred to be equal to sub_layers_vps_max_minus1 [i - 1]. The sub_layers_vps_max_minus1_predict_flag[i] equal to 0 specifies that sub_layers_vps_max_minus1[i] is explicitly signalled. The value of sub_layers_vps_max_minus1_predict_flag[0] is inferred to be equal to 0. The sub_layers_vps_max_minus1[i] plus 1 specifies the maximum number of temporal sub-layers that may be present in the CVS for layer with nuh_layer_id equal to layer_id_in_nuh[i]. The value of sub_layers_vps_max_minus1[i] shall be in the range of 1 to vps_max_sub_layers_minus1 inclusive. When sub_layers_vps_max_minus1_predict_flag [i] is equal to 1, sub_layers_vps_max_minus1[i] is inferred to be equal to sub_layers_vps_max_minus1[i - 1]. The value of sub_layers_vps_max_minus1 [0] is inferred to be equal to vps_max_sub_layers_minus1. Alternatively, the value of sub_layers_vps_max_minus1[i] may be in the range of 0 to 6 inclusive. Alternatively, the value of sub_layers_vps_max_minus1[i] may only be signaled for the enhancement layers in the VPS extension as illustrated in FIG. 36.

[0144] The slice segment headers may be modified, such as described below, in such a manner that the derivation of the NumActiveRefLayerPics accounts for the occurrence of one of the aforementioned three conditions so as to reduce the ambiguity using the signaled information about the maximum number of temporal sub-layers that may be present for each layer.

[0145] The inter_layer_pred_enabled_flag equal to 1 specifies that inter-layer prediction may be used in decoding of the current picture. The inter_layer_pred_enabled_flag equal to 0 specifies that inter-layer prediction is not used in decoding of the current picture. The num_inter_layer_ref_pics_minus1 plus 1 specifies the number of pictures that may be used in decoding of the current picture for inter-layer prediction. The length of the num_inter_layer_ref_pics_minus1 syntax element is Ceil(Log2(NumDirectRefLayers[nuh_layer_id])) bits. The value of num_inter_layer_ref_pics_minus1 shall be in the range of 0 to NumDirectRefLayers[nuh_layer_id] - 1, inclusive. The variable NumActiveRefLayerPics is derived as follows:

```
if( nuh_layer_id  = =  0  | |  NumDirectRefLayers[ nuh_layer_id ]  = =  0 )
        NumActiveRefLayerPics = 0
else if( all_ref_layers_active_flag ){
        NumActiveRefLayerPics = NumDirectRefLayers[ nuh_layer_id ]
        for( i = 0; i < NumDirectRefLayers[ nuh_layer_id ]; i++) {
                if( sub_layers_vps_max_minus1[ LayerIdxInVps[ RefLayer[ n
        uh_layer_id ][ i ] ] ] < TemporalId )
                NumActiveRefLayerPics = NumActiveRefLayerPics - 1
        }
}
else if( !inter_layer_pred_enabled_flag )
        NumActiveRefLayerPics = 0
else if( max_one_active_ref_layer_flag  | |
    NumDirectRefLayers[ nuh_layer_id ]  = = 1 ) {
        if( sub_layers_vps_max_minus1[ LayerIdxInVps[ RefLayer[ nuh_lay
er_id ][ 0 ] ] ] < TemporalId )
            NumActiveRefLayerPics = 0
        else
            NumActiveRefLayerPics = 1
}
else
        NumActiveRefLayerPics = num_inter_layer_ref_pics_minus1 + 1
```

[0146] All slices of a coded picture shall have the same value of NumActiveRefLayerPics. The inter_layer_pred_layer_idc[i] specifies the variable, RefPicLayerId[i], representing the nuh_layer_id of the i-th picture that may be used by the current picture for inter-layer prediction. The length of the syntax element inter_layer_pred_layer_idc[i] is Ceil(Log2(NumDirectRefLayers[nuh_layer_id])) bits. The value of inter_layer_pred_layer_idc[i] shall be in the range of 0 to NumDirectRefLayers[nuh_layer_id] - 1, inclusive. When not present, the value of inter_layer_pred_layer_idc[i] is inferred as follows:

```
for( i = 0, j = 0; i < NumDirectRefLayers[ nuh_layer_id ]; i++) {
        if( sub_layers_vps_max_minus1[ LayerIdxInVps[ RefLayer[ nuh_layer_id ][ i ] ] ] >= TemporalId )
                inter_layer_pred_layer_idc[ j++ ] = i;
}
```

In a variant embodiment when not present, the value of inter_layer_pred_layer_idc[i] is inferred as follows:

```
for( i = 0, j = 0; i < NumDirectRefLayers[ nuh_layer_id ]; i++) {
        if( sub_layers_vps_max_minus1[ LayerIdxInVps[ RefLayer[ nuh_layer_id ][ i ] ] ] < TemporalId )
                inter_layer_pred_layer_idc[ j++ ] = i;
}
```

[0147] When i is greater than 0, inter_layer_pred_layer_idc[i] shall be greater than inter_layer_pred_layer_idc[i - 1]. The variables RefPicLayerId[i] for all values of i in the range of 0 to NumActiveRefLayerPics - 1, inclusive, are derived as follows:

```
for( i = 0, j = 0; i < NumActiveRefLayerPics; i++)
        RefPicLayerId[ i ] =
    RefLayerId[ nuh_layer_id ][ inter_layer_pred_layer_idc[ i ] ]
```

[0148] All slices of a picture shall have the same value of inter_layer_pred_layer_idc[i] for each value of i in the range of 0 to NumActiveRefLayerPics - 1, inclusive. It is a requirement of bitstream conformance that for each value of i in the range of 0 to NumActiveRefLayerPics - 1, inclusive, either of the following two conditions shall be true:

(1) The value of max_tid_il_ref_pics_plus1[LayerIdxInVps[RefPicLayerId[i]]] is greater than TemporalId.

(2) The values of max_tid_il_ref_pics_plus1[LayerIdxInVps[RefPicLayerId[i]]] and TemporalId are both equal to 0 and the picture in the current access unit with nuh_layer_id equal to RefPicLayerId[i] is an IRAP picture.

[0149] In another embodiment the names of various syntax elements and their semantics may be altered by adding a plus1 or plus2 or by subtracting a minus1 or a minus2 compared to the described syntax and semantics.

[0150] In another embodiment some of the conditions in the if statements may be altered by adding a plus1 or plus2 or by subtracting a minus1 or a minus2 compared to the described syntax.

[0151] Referring to FIG. 37, an additional signaling technique involves signaling a layer_present_in_au_flag[i]. The layer_present_in_au_flag[i] equal to 1 specifies that a picture with nuh_layer_id equal to RefPicLayerId[i] is present in the current access unit. The layer_present_in_au_flag[i] equal to 0 specifies that a picture with nuh_layer_id equal to RefPicLayerId[i] is not present in the current access unit. When not present layer_present_in_au_flag[i] is inferred to be equal to 1.

[0152] Referring to FIG. 38, an additional signaling technique involves signaling the layer_present_in_au_flag[i]. The layer_present_in_au_flag[i] equal to 1 specifies that a picture with nuh_layer_id equal to RefLayerId[nuh_layer_id][i] is present in the current access unit. The layer_present_in_au_flag[i] equal to 0 specifies that a picture with nuh_layer_id equal to RefLayerId[nuh_layer_id][i] is not present in the current access unit. When not present layer_present_in_au_flag[i] is inferred to be equal to 1.

[0153] Referring to FIG. 39, an additional signaling technique involves signaling the layer_present_in_au_flag[i]. The layer_present_in_au_flag[i] equal to 1 specifies that a picture with nuh_layer_id equal to layer_id_in_nuh[i] is present in the current access unit. layer_present_in_au_flag[i] equal to 0 specifies that a picture with nuh_layer_id equal to layer_id_in_nuh[i] is not present in the current access unit. When not present layer_present_in_au_flag[i] is inferred to be equal to 1.

[0154] If desired, the flags layer_present_in_au_flag[i] may be only signaled in FIG. 37, FIG. 38, and/or FIG. 39 if one or more of the following conditions are met.

[0155] The first condition is that if only one active reference layer can be used for each layer (i.e. max_one_active_ref_layer_flag is equal to 1).

[0156] The second condition is that the number of direct reference layers for a layer as signaled by direct dependency relationship between layers (e.g. by direct_dependency_flag[i][j]) is equal to 1 (i.e. NumDirectRefLayers[nuh_layer_id] is equal to 1).

[0157] The third condition is that all the direct reference layers for a layer as signaled by direct dependency relationship between layers (e.g. by direct_dependency_flag[i][j]) is equal to 1 are active reference layers for the coded picture of the layer (e.g. all_ref_layers_active_flag is equal to 1).

[0158] The three variants shown in FIG. 40, FIG. 41, and FIG. 42 for the above three conditions corresponds respectively to FIG. 37, FIG. 38, and FIG. 39.

[0159] Referring to FIG. 43, the decoding process for the inter-layer reference picture set may be modified. The outputs of this process are updated lists of inter-layer reference pictures RefPicSetInterLayer0 and RefPicSetInterLayer1 and the variables NumActiveRefLayerPics0 and NumActiveRefLayerPics1. The variable currLayerId is set equal to nuh_layer_id of the current decoded pictures. The lists RefPicSetInterLayer0 and RefPicSetInterLayer1 are first emptied, NumActiveRefLayerPics0 and NumActiveRefLayerPics1 are set equal to 0 followed by steps as illustrated in FIG. 43. There shall be no entry equal to "no reference picture" in RefPicSetInterLayer0 or RefPicSetInterLayer1. The RefPicSetInterLayer1 is always empty since the value of ViewId[i] is equal to zero for all layers. If the current picture is a RADL picture, there shall be no entry in the RefPicSetInterLayer0 or RefPicSetInterLayer1 that is a RASL picture. An access unit may contain both RASL and RADL pictures.

[0160] Referring to FIG. 44, the decoding process for the inter-layer reference picture set may be modified. The outputs of this process are updated lists of inter-layer reference pictures RefPicSetInterLayer0 and RefPicSetInterLayer1 and the variables NumActiveRefLayerPics0 and NumActiveRefLayerPics1. The variable currLayerId is set equal to nuh_layer_id of the current decoded picture. The lists RefPicSetInterLayer0 and RefPicSetInterLayer1 are first emptied, NumActiveRefLayerPics0 and NumActiveRefLayerPics1 are set equal to 0 followed by steps as illustrated in FIG. 44. There shall be no entry equal to "no reference picture" in RefPicSetInterLayer0 or RefPicSetInterLayer1. The RefPicSetInterLayer1 is always empty since the value of ViewId[i] is equal to zero for all layers. If the current picture is a RADL picture, there shall be no entry in the RefPicSetInterLayer0 or RefPicSetInterLayer1 that is a RASL picture. An access unit may contain both RASL and RADL pictures.

[0161] Referring to FIG. 45, the decoding process for the inter-layer reference picture set may be modified. The outputs of this process are updated lists of inter-layer reference pictures RefPicSetInterLayer0 and RefPicSetInterLayer1 and the variables NumActiveRefLayerPics0 and NumActiveRefLayerPics1. The variable currLayerId is set equal to nuh_layer_id of the current decoded picture. The lists RefPicSetInterLayer0 and RefPicSetInterLayer1 are first emptied, NumActiveRefLayerPics0 and NumActiveRefLayerPics1 are set equal to 0 followed by steps as illustrated in FIG. 45. There shall be no entry equal to "no reference picture" in RefPicSetInterLayer0 or RefPicSetInterLayer1. The RefPicSetInterLayer1 is always empty since the value of ViewId[i] is equal to zero for all layers. If the current picture is a RADL picture, there shall be no entry in the RefPicSetInterLayer0 or RefPicSetInterLayer1 that is a RASL picture. An access unit may contain both RASL and RADL pictures.

[0162] Referring to FIG. 46, the decoding process for the inter-layer reference picture set may be modified. The outputs of this process are updated lists of inter-layer reference pictures RefPicSetInterLayer0 and RefPicSetInterLayer1 and the variables NumActiveRefLayerPics0 and NumActiveRefLayerPics1. The variable currLayerId is set equal to nuh_layer_id of the current decoded picture. The lists RefPicSetInterLayer0 and RefPicSetInterLayer1 are first emptied, NumActiveRefLayerPics0 and NumActiveRefLayerPics1 are set equal to 0 followed by steps as illustrated in FIG. 46. There shall be no entry equal to "no reference picture" in RefPicSetInterLayer0 or RefPicSetInterLayer1. The RefPicSetInterLayer1 is always empty since the value of ViewId[i] is equal to zero for all layers. If the current picture is a RADL picture, there shall be no entry in the RefPicSetInterLayer0 or RefPicSetInterLayer1 that is a RASL picture. An access unit may contain both RASL and RADL pictures.

[0163] In an alternative embodiment the syntax for signaling inter-layer prediction information in slice segment header may be modified as shown in Figure 47. In this case the syntax elements inter_layer_pred_enabled_flag, num_inter_layer_ref_pics_minus1 and inter_layer_pred_layer_idc[i] would be always signaled even when one or more of the conditions as follows are true: when max_one_active_ref_layer_flag is equal to 1, and / or NumDirectRefLayers[nuh_layer_id] is equal to 1, and/ or all_ref_layers_active_flag is equal to 1

In this case the ambiguity about a lost reference layer picture versus non-existing reference layer picture is removed. In this case the following may apply.

[0164] The inter_layer_pred_enabled_flag equal to 1 specifies that inter-layer prediction may be used in decoding of the current picture. The inter_layer_pred_enabled_flag equal to 0 specifies that inter-layer prediction is not used in decoding of the current picture. The num_inter_layer_ref_pics_minus1 plus 1 specifies the number of pictures that may be used in decoding of the current picture for inter-layer prediction. The length of the num_inter_layer_ref_pics_minus1 syntax element is Ceil(Log2(NumDirectRefLayers[nuh_layer_id])) bits. The value of num_inter_layer_ref_pics_minus1 shall be in the range of 0 to NumDirectRefLayers[nuh_layer_id] - 1, inclusive. The variable NumActiveRefLayerPics is derived as follows:

```
if( nuh_layer_id  = =  0  | |  NumDirectRefLayers[ nuh_layer_id ]  = =  0 )
        NumActiveRefLayerPics = 0
else
        NumActiveRefLayerPics = num_inter_layer_ref_pics_minus1 + 1
```

[0165] All slices of a coded picture shall have the same value of NumActiveRefLayerPics. The inter_layer_pred_layer_idc[i] specifies the variable, RefPicLayerId[i], representing the nuh_layer_id of the i-th picture that may be used by the current picture for inter-layer prediction. The length of the syntax element inter_layer_pred_layer_idc[i] is Ceil(Log2(NumDirectRefLayers[nuh_layer_id])) bits. The value of inter_layer_pred_layer_idc[i] shall be in the range of 0 to NumDirectRefLayers[nuh_layer_id] - 1, inclusive. When i is greater than 0, inter_layer_pred_layer_idc[i] shall be greater than inter_layer_pred_layer_idc[i - 1]. The variables RefPicLayerId[i] for all values of i in the range of 0 to NumActiveRefLayerPics - 1, inclusive, are derived as follows:

```
for( i = 0, j = 0; i < NumActiveRefLayerPics; i++)
            RefPicLayerId[ i ] =
    RefLayerId[ nuh_layer_id ][ inter_layer_pred_layer_idc[ i ] ]
```

All slices of a picture shall have the same value of inter_layer_pred_layer_idc[i] for each value of i in the range of 0 to NumActiveRefLayerPics - 1, inclusive. It is a requirement of bitstream conformance that for each value of i in the range of 0 to NumActiveRefLayerPics - 1, inclusive, either of the following two conditions shall be true:

(1) The value of max_tid_il_ref_pics_plus1[LayerIdxInVps[RefPicLayerId[i]]] is greater than TemporalId.

(2) The values of max_tid_il_ref_pics_plus1[LayerIdxInVps[RefPicLayerId[i]]] and TemporalId are both equal to 0 and the picture in the current access unit with nuh_layer_id equal to RefPicLayerId[i] is an IRAP picture.

[0166] The NumDirectRefLayers for a layer may be derived based upon a direct_dependency_flag[i][j] that when equal to 0 specifies that the layer with index j is not a direct reference layer for the layer with index i. The direct_dependency_flag[i][j] equal to 1 specifies that the layer with index j may be a direct reference layer for the layer with index i. When direct_dependency_flag[i][j] is not present for i and j in the range of 0 to vps_max_layers_minus1, it is inferred to be equal to 0.

[0167] The variables NumDirectRefLayers[i], RefLayerId[i][j] SamplePredEnabledFlag[i][j], MotionPredEnabledFlag[i][j] and DirectRefLayerIdx[i][j] may be derived as follows:

```
for( i = 0; i  <=  vps_max_layers_minus1; i++ ) {
    iNuhLId = layer_id_in_nuh[ i ]
    NumDirectRefLayers[ iNuhLId ] = 0
        for( j = 0; j < i; j++ )
            if( direct_dependency_flag[ i ][ j ] ) {
                RefLayerId[ iNuhLId ][ NumDirectRefLayers[ iNuhLId ]++ ] =
layer_id_in_nuh[ j ]
                SamplePredEnabledFlag[ iNuhLId ][ j ] =
( ( direct_dependency_type[ i ][ j ] + 1 ) & 1 )
                MotionPredEnabledFlag[ iNuhLId ][ j ] =
( ( ( direct_dependency_type[ i ][ j ] + 1 ) & 2 ) >> 1 )
                DirectRefLayerIdx[ iNuhLid ][ layer_id_in_nuh[ j ] ] =
NumDirectRefLayers[ iNuhLId ] – 1          }
}
```

[0168] The direct_dependency_type[i][j] indicates the type of dependency between the layer with nuh_layer_id equal layer_id_in_nuh[i] and the layer with nuh_layer_id equal to layer_id_in_nuh[j]. direct_dependency_type[i][j] equal to 0 indicates that the layer with nuh_layer_id equal to layer_id_in_nuh[j] is used for inter-layer sample prediction but not for inter-layer motion prediction of the layer with nuh_layer_id equal layer_id_in_nuh[i]. direct_dependency_type[i][j] equal to 1 indicates that the layer with nuh_layer_id equal to layer_id_in_nuh[j] is used for inter-layer motion prediction but not for inter-layer sample prediction of the layer with nuh_layer_id equal layer_id_in_nuh[i]. direct_dependency_type[i][j] equal to 2 indicates that the layer with nuh_layer_id equal to layer_id_in_nuh[j] is used for both inter-layer sample motion prediction and inter-layer motion prediction of the layer with nuh_layer_id equal layer_id_in_nuh[i]. Although the value of direct_dependency_type[i][j] shall be in the range of 0 to 2, inclusive, in this version of this Specification, decoders shall allow values of direct_dependency_type[i][j] in the range of 3 to 2^{32} - 2, inclusive, to appear in the syntax.

[0169] The direct_dependency_flag[i][j], direct_dep_type_len_minus2, direct_dependency_type[i][j] are included in the vps_extension syntax illustrated in FIG. 48A and FIG. 48B, which is included by reference in the VPS syntax which provides syntax for the coded video sequence.

[0170] It is typically desirable to reduce the number of referenced layers that need to be signaled within the bitstream, and other syntax elements within the slice segment header may be used to effectuate such a reduction. The other syntax elements may include inter_layer_pred_enabled_flag, num_inter_layer_ref_pics_minus1, and/or inter_layer_pred_layer_idc[i]. These syntax elements may be signaled in slice segment header.

[0171] The inter_layer_pred_enabled_flag equal to 1 specifies that inter-layer prediction may be used in decoding of the current picture. The inter_layer_pred_enabled_flag equal to 0 specifies that inter-layer prediction is not used in decoding of the current picture. When not present, the value of inter_layer_pred_enabled_flag is inferred to be equal to 0.

[0172] The num_inter_layer_ref_pics_minus1 plus 1 specifies the number of pictures that may be used in decoding of the current picture for inter-layer prediction. The length of the num_inter_layer_ref_pics_minus1 syntax element is Ceil(Log2(NumDirectRefLayers[nuh_layer_id])) bits. The value of num_inter_layer_ref_pics_minus1 shall be in the range of 0 to NumDirectRefLayers[nuh_layer_id] - 1, inclusive.

[0173] The variable NumActiveRefLayerPics is derived as follows:

```
    if( nuh_layer_id = = 0 | | NumDirectRefLayers[ nuh_layer_id ] = = 0
| | !inter_layer_pred_enabled_flag )
        NumActiveRefLayerPics = 0
else if( max_one_active_ref_layer_flag | | NumDirectRefLayers[ nuh_layer_id ] = = 1 )
        NumActiveRefLayerPics = 1
else
        NumActiveRefLayerPics = num_inter_layer_ref_pics_minus1 + 1
```

All slices of a coded picture shall have the same value of NumActiveRefLayerPics.

[0174] The inter_layer_pred_layer_idc[i] specifies the variable, RefPicLayerId[i], representing the nuh_layer_id of the i-th picture that may be used by the current picture for inter-layer prediction. The length of the syntax element inter_layer_pred_layer_idc[i] is Ceil(Log2(NumDirectRefLayers[nuh_layer_id])) bits. The value of inter_layer_pred_layer_idc[i] may be in the range of 0 to NumDirectRefLayers[nuh_layer_id] - 1, inclusive. When not present, the value of inter_layer_pred_layer_idc[i] is inferred to be equal to 0.

[0175] By way of example, the system may signal various syntax elements especially the direct_dependency_flag[i][j] in VPS which results in the inter-layer reference picture set for layer 3 to be [2, 0],. Then the system may refine further the inter-layer reference picture set with the use of the additional syntax elements for example syntax elements in slice segment header as [2], may refine further the inter-layer reference picture set with the use of the additional syntax elements as [0], or may refine further the inter-layer reference picture set with the use of the additional syntax elements as [ ] which is the null set. However, depending on the design of the encoder, the reference picture set of [2, 0] may be signaled as [2, 0].

[0176] In FIG. 48B the vps_vui_present_flag equal to 1 specifies that the vps_vui( ) syntax structure is present in the VPS. vps_vui_present_flag equal to 0 specifies that the vps_vui( ) syntax structure is not present in the VPS.
vps_vui_alignment_bit_equal_to_one may be equal to 1.

[0177] VPS VUI includes syntax elements which indicate inter-layer prediction restrictions. Essentially depending on spatial segmentation tools used a delay in units of slices, tiles, wavefront coded tree block (CTB) rows with respect to the collocated spatial segment in the reference layer may be signaled. Also based on flag a delay in units of CTBs may be signaled. These inter-layer decoding delay signaling can help parallel decoding of layers, where for a dependent layer instead of waiting for each reference layer to be decoded completely in its entirety before starting its own decoding, the decoding could be started after the indicated delay for each reference layer.

[0178] FIG. 49 shows part of an exemplary VPS Video Usability Information (VUI) syntax. This may correspond to the vps_vui() structure in FIG. 48B and exemplary vps extension syntax.

[0179] FIG. 50 shows part of another exemplary VPS Video Usability Information (VUI) syntax with some differences in syntax compared to FIG. 49. This may correspond to the vps_vui() structure in FIG. 48B and exemplary vps extension syntax.

[0180] VPS VUI includes syntax elements related to bit rate and picture rate information for the video.

[0181] In SHVC different layers may have different frame rates. As a result a layer with a higher frame rate may have a higher value of maximum temporal sub-layers compared to a layer with a lower frame-rate. The j-th subset of a layer set is the output of the sub-bitstream extraction process when it is invoked with the layer set, j, and the layer identifier list associated with the layer set as inputs. In mixed frame rate case for certain layer sets the maximum number of temporal sub-layers in the layer set may be less than vps_max_sub_layers_minus1. In this case some of the (vps_max_sub_layers_minus1 + 1) subsets of such a layer set will be identical. It is wasteful to signal bitrate and picture information for these identical subsets. Information regarding maximum number of temporal sub-layers for a layer (sub_layers_vps_max_minus1) is already signalled in VPS.

[0182] Modification of signaling bit rate and picture rate information in VPS VUI as shown in FIG. 50 has benefits in not wasting bits to send information for identical subsets. In FIG. 50 the bit rate and picture rate information (including bit_rate_present_flag[i][j], pic_rate_present_flag[i][j], avg_bit_rate[i][j], max_bit_Rate[i][j], constant_pic_rate_idc[i][j], avg_pic_rate[i][j]) is signalled only up to the maximum temporal sub-layers in the corresponding layer set. Thus it is preferable to signal the bit rate and picture rate information for subsets only up to the maximum temporal sub-layers in the corresponding layer set.

[0183] The variable MaxSlLayersetMinus1[i] is derived as follows :

```
for( i = 0; i <= vps_number_layer_sets_minus1; i++ ) {
        for( k = 0, MaxSlLayersetMinus1[ i ]=0; k < NumLayersInIdList[ i ]; k++ ) {
                MaxSlLayersetMinus1[ i ] =Max(MaxSlLayersetMinus1[ i ], sub_layers_vps_max_minus1[ LayerIdxInVps[ LayerSetLayerIdList[ i ][ k ] ] ]);
        }
}
```

[0184] In another embodiment the variable MaxSlLayersetMinus1[i] is derived as follows :

```
for( i = 0; i <= vps_number_layer_sets_minus1; i++ ) {
        mSlMinus1 = 0
        for( k = 0; k < NumLayersInIdList[ i ]; k++ ) {
        lId = LayerSetLayerIdList[ i ][ k ]
        mSlMinus1 =Max(mSlMinus1,
sub_layers_vps_max_minus1[ LayerIdxInVps[ lId ] ]);
        }
MaxSlLayersetMinus1[ i ] = mSlMinus1
}
```

[0185] Then the derived MaxSlLayersetMinus1[i] is used such that the j the index for subsets ranges from 0 to MaxSlLayersetMinus1[i], inclusive instead of from 0 to vps_max_sub_layers_minus1, inclusive.

[0186] bit_rate_present_vps_flag equal to 1 specifies that the syntax element bit_rate_present_flag[i][j] is present. bit_rate_present_vps_flag equal to 0 specifies that the syntax element bit_rate_present_flag[i][j] is not present.

[0187] pic_rate_present_vps_flag equal to 1 specifies that the syntax element pic_rate_present_flag[i][j] is present. pic_rate_present_vps_flag equal to 0 specifies that the syntax element pic_rate_present_flag[i][j] is not present.

[0188] bit_rate_present_flag[i][j] equal to 1 specifies that the bit rate information for the j-th subset of the i-th layer set is present. bit_rate_present_flag[i] equal to 0 specifies that the bit rate information for the j-th subset of the i-th layer set is not present. The j-th subset of a layer set is the output of the sub-bitstream extraction process when it is invoked with the layer set, j, and the layer identifier list associated with the layer set as inputs. When not present, the value of bit_rate_present_flag[i][j] is inferred to be equal to 0.

[0189] pic_rate_present_flag[i][j] equal to 1 specifies that picture rate information for the j-th subset of the i-th layer set is present. pic_rate_present_flag[i][j] equal to 0 specifies that picture rate information for the j-th subset of the i-th layer set is not present. When not present, the value of pic_rate_present_flag[i][j] is inferred to be equal to 0.

[0190] avg_bit_rate[i][j] indicates the average bit rate of the j-th subset of the i-th layer set, in bits per second. The value is given by BitRateBPS(avg_bit_rate[i][j]) with the function BitRateBPS( ) being specified as follows:

$$\text{BitRateBPS}(x) = (x \,\&\, (2^{14} - 1)) * 10^{(2 + (x >> 14))}$$

[0191] The average bit rate is derived according to the access unit removal time specified in clause F.13. In the following, bTotal is the number of bits in all NAL units of the j-th subset of the i-th layer set, t_1 is the removal time (in seconds) of the first access unit to which the VPS applies, and t_2 is the removal time (in seconds) of the last access unit (in decoding order) to which the VPS applies. With x specifying the value of avg_bit_rate[i][j], the following applies:

If t_1 is not equal to t_2, the following condition shall be true:

$$(x \& (2^{14} - 1)) == \mathrm{Round}(bTotal \div ((t_2 - t_1) * 10^{(2 + (x >> 14))}))$$

Otherwise (t_1 is equal to t_2), the following condition shall be true:

$$(x \& (2^{14} - 1)) == 0$$

[0192] max_bit_rate_layer[i][j] indicates an upper bound for the bit rate of the j-th subset of the i-th layer set in any one-second time window of access unit removal time as specified in clause F.13. The upper bound for the bit rate in bits per second is given by BitRateBPS(max_bit_rate_layer[i][j]). The bit rate values are derived according to the access unit removal time specified in clause F.13. In the following, t_1 is any point in time (in seconds), t_2 is set equal to

$$t_1 + 1 \div 100$$

, and bTotal is the number of bits in all NAL units of access units with a removal time greater than or equal to t_1 and less than t_2. With x specifying the value of max_bit_rate_layer[i][j], the following condition shall be obeyed for all values of t_1:

$$(x \& (2^{14} - 1)) >= bTotal \div ((t_2 - t_1) * 10^{(2 + (x >> 14))})$$

[0193] constant_pic_rate_idc[i][j] indicates whether the picture rate of the j-th subset of the i-th layer set is constant. In the following, a temporal segment tSeg is any set of two or more consecutive access units, in decoding order, of the j-th subset of the i-th layer set, auTotal(tSeg) is the number of access units in the temporal segment tSeg, t_1(tSeg) is the removal time (in seconds) of the first access unit (in decoding order) of the temporal segment tSeg, t_2(tSeg) is the removal time (in seconds) of the last access unit (in decoding order) of the temporal segment tSeg, and avgPicRate(tSeg) is the average picture rate in the temporal segment tSeg, and is specified as follows:

$$avgPicRate(tSeg) == \mathrm{Round}(auTotal(tSeg) * 256 \div (t_2(tSeg) - t_1(tSeg)))$$

[0194] If the j-th subset of the i-th layer set only contains one or two access units or the value of avgPicRate(tSeg) is constant over all the temporal segments, the picture rate is constant; otherwise, the picture rate is not constant.

[0195] constant_pic_rate_idc[i][j] equal to 0 indicates that the picture rate of the j-th subset of the i-th layer set is not constant. constant_pic_rate_idc[i][j] equal to 1 indicates that the picture rate of the j-th subset of the i-th layer set is constant. constant_pic_rate_idc[i][j] equal to 2 indicates that the picture rate of the j-th subset of the i-th layer set may or may not be constant. The value of constant_pic_rate_idc[i

][j] shall be in the range of 0 to 2, inclusive.

[0196] avg_pic_rate[i] indicates the average picture rate, in units of picture per 256 seconds, of the j-th subset of the layer set. With auTotal being the number of access units in the j-th subset of the i-th layer set, t_1 being the removal time (in seconds) of the first access unit to which the VPS applies, and t_2 being the removal time (in seconds) of the last access unit (in decoding order) to which the VPS applies, the following applies:

If t_1 is not equal to t_2, the following condition shall be true:

avg_pic_rate[i] = = Round(auTotal * 256 ÷ ($t_2 - t_1$))

Otherwise (t_1 is equal to t_2), the following condition shall be true:

avg_pic_rate[i] = = 0

[0197] Currently in JCTVC-P1008 and JCT3V-G1004 in DPB Size Semantics the variable MaxSubLayersInLayerSetMinus1[i] is derived as follows:

```
for( i = 1; i < NumOutputLayerSets; i++ ) {
    maxSLMinus1 = 0
    optLsIdx = LayerSetIdxForOutputLayerSet[ i ]
    for( k = 0; k < NumLayersInIdList[ optLsIdx ]; k++ ) {
        lId = LayerSetLayerIdList[ optLsIdx ][ k ]
        maxSLMinus1 =Max( maxSLMinus1,
sub_layers_vps_max_minus1[ LayerIdxInVps[ lId ] ] )
    }
    MaxSubLayersInLayerSetMinus1[ i ] = maxSLMinus1
}
```

[0198] In some embodiment the above derivation and the proposed derivation of MaxSlLayersetMinus1[i] may be combined with derivation of MaxSubLayersInLayerSetMinus1[i] as follows.

The variable MaxSlLayersetMinus1[i] is derived as follows :

```
for( i = 0; i <= vps_number_layer_sets_minus1; i++ ) {
        mSlMinus1 = 0
        for( k = 0; k < NumLayersInIdList[ i ]; k++ ) {
         lId = LayerSetLayerIdList[ i ][ k ]
         mSlMinus1 =Max(mSlMinus1,
    sub_layers_vps_max_minus1[ LayerIdxInVps[ lId ] ]);
        }
        MaxSlLayersetMinus1[ i ] = mSlMinus1
}
for( i = 1; i < NumOutputLayerSets; i++ ) {
    MaxSubLayersInLayerSetMinus1[ i ] =
MaxSlLayersetMinus1[ LayerSetIdxForOutputLayerSet[ i ] ]
}
```

[0199] In yet another embodiment the variable MaxSlLayersetMinus1[LayerSetIdxForOutputLayerSet[i]]may be directly used in place of variable MaxSubLayersInLayerSetMinus1[i].

[0200] Thus the dpb_size may be signaled as follows

dpb_size( ) {	
for(i = 1; i < NumOutputLayerSets; i++) {	
sub_layer_flag_info_present_flag[i]	u(1)
for(j = 0; j <= MaxSlLayersetMinus1[LayerSetIdxForOutputLayerSet[i]]; j++) {	
if(j > 0 && sub_layer_flag_info_present_flag[i])	
sub_layer_dpb_info_present_flag[i][j]	u(1)
if(sub_layer_dpb_info_present_flag[i][j]) {	
for(k = 0; k < NumSubDpbs[LayerSetIdxForOutputLayerSet[i]]; k++)	
max_vps_dec_pic_buffering_minus1[i][k][j]	ue(v)
max_vps_num_reorder_pics[i][j]	ue(v)
if(NumSubDpbs[LayerSetIdxForOutputLayerSet[i]] != NumLayersInIdList[LayerSetIdxForOutputLayerSet[i]])	
for(k = 0; k < NumLayersInIdList[LayerSetIdxForOutputLayerSet[i]]; k++)	
max_vps_layer_dec_pic_buff_minus1[i][k][j]	ue(v)
max_vps_latency_increase_plus1[i][j]	ue(v)
}	
}	
}	
}	

[0201] The semantics of various parameter using MaxSubLayersInLayerSetMinus1[i] may be changed to directly use MaxSlLayersetMinus1[LayerSetIdxForOutputLayerSet[i]].

[0202] sub_layer_flag_info_present_flag[i] equal to 1 specifies that

sub_layer_dpb_info_present_flag[i][j] is present for i in the range of 1 to MaxSlLayersetMinus1[LayerSetIdxForOutputLayerSet[i]], inclusive.
sub_layer_flag_info_present_flag[i] equal to 0 specifies that, for each value of j greater than 0, sub_layer_dpb_info_present_flag[i][j] is not present and the value is inferred to be equal to 0.

[0203] sub_layer_dpb_info_present_flag[i][j] equal to 1 specifies that max_vps_dec_pic_buffering_minus1[i][k][j] is present for k in the range of 0 to NumSubDpbs[LayerSetIdxForOutputLayerSet[i]] - 1, inclusive, for the j-th sub-layer, and max_vps_num_reorder_pics[i][j] and max_vps_latency_increase_plus1[i][j] are present for the j-th sub-layer. sub_layer_dpb_info_present_flag[i][j] equal to 0 specifies that the values of max_vps_dec_pic_buffering_minus1[i][k][j] are equal to max_vps_dec_pic_buffering_minus1[i][k][j - 1] for k in the range of 0 to NumSubDpbs[LayerSetIdxForOutputLayerSet[i]] - 1, inclusive, and that the values max_vps_num_reorder_pics[i][j] and max_vps_latency_increase_plus1[i][j] are set equal to max_vps_num_reorder_pics[i][j - 1] and max_vps_latency_increase_plus1[i][j - 1], respectively. The value of sub_layer_dpb_info_present_flag[i][0] for any possible value of i is inferred to be equal to 1. When not present, the value of sub_layer_dpb_info_present_flag[i][j] for j greater than 0 and any possible value of i, is inferred to be equal to be equal to 0.

[0204] max_vps_dec_pic_buffering_minus1[i][k][j] plus 1 specifies the maximum required size of the k-th sub-DPB for the CVS in the i-th output layer set in units of picture storage buffers when HighestTid is equal to j. When j is greater than 0, max_vps_dec_pic_buffering_minus1[i][k][j] shall be greater than or equal to max_vps_dec_pic_buffering_minus1[i][k][j - 1]. When max_vps_dec_pic_buffering_minus1[i][k][j] is not present for j in the range of 1 to MaxSlLayersetMinus1[LayerSetIdxForOutputLayerSet[i]], inclusive, it is inferred to be equal to max_vps_dec_pic_buffering_minus1[i][k][j - 1].

[0205] max_vps_layer_dec_pic_buff_minus1[i][k][j] plus 1 specifies the maximum number of decoded pictures, of the k-th layer for the CVS in the i-th output layer set, that need to be stored in the DPB when HighestTid is equal to j. When j is greater than 0, max_vps_layer_dec_pic_buff_minus1[i][k][j] shall be greater than or equal to max_vps_layer_dec_pic_buff_minus1[i][k][j - 1]. When max_vps_layer_dec_pic_buff_minus1[i][k][j] is not present for j in the range of 0 to MaxSlLayersetMinus1[LayerSetIdxForOutputLayerSet[i]], inclusive, it is inferred to be equal to max_vps_layer_dec_pic_buff_minus1[i][k][j - 1].

[0206] max_vps_num_reorder_pics[i][j] specifies, when HighestTid is equal to j, the maximum allowed number of access units containing a picture with PicOutputFlag equal to 1 that can precede any access unit auA that contains a picture with PicOutputFlag equal to 1 in the i-th output layer set in the CVS in decoding order and follow the access unit auA that contains a picture with PicOutputFlag equal to 1 in output order. When max_vps_num_reorder_pics[i][j] is not present for j in the range of 1 to MaxSlLayersetMinus1[LayerSetIdxForOutputLayerSet[i]], inclusive, due to sub_layer_dpb_info_present_flag[i][j] being equal to 0, it is inferred to be equal to max_vps_num_reorder_pics[i][j - 1].

[0207] max_vps_latency_increase_plus1[i][j] not equal to 0 is used to compute the value of VpsMaxLatencyPictures[i][j], which, when HighestTid is equal to j, specifies the maximum number of access units containing a picture with PicOutputFlag equal to 1 in the i-th output layer set that can precede any access unit auA that contains a picture with PicOutputFlag equal to 1 in the CVS in output order and follow the access unit auA that contains a picture with PicOutputFlag equal to 1 in decoding order. When max_vps_latency_increase_plus1[i][j] is not present for j in the range of 1 to MaxSlLayersetMinus1[LayerSetIdxForOutputLayerSet[i]], inclusive, due to sub_layer_dpb_info_present_flag[i][j] being equal to 0, it is inferred to be equal to max_vps_latency_increase_plus1[i][j - 1].

[0208] When max_vps_latency_increase_plus1[i][j] is not equal to 0, the value of VpsMaxLatencyPictures[i][j] is specified as follows:

$$\mathrm{VpsMaxLatencyPictures}[\,i\,][\,j\,] = \mathrm{max\_vps\_num\_reorder\_pics}[\,i\,][\,j\,] + \mathrm{max\_vps\_latency\_increase\_plus1}[\,i\,][\,j\,] - 1$$

[0209] When max_vps_latency_increase_plus1[i][j] is equal to 0, no corresponding limit is expressed. The value of max_vps_latency_increase_plus1[i][j] shall be in the range of 0 to 2^{32} - 2, inclusive.

[0210] In another embodiment max_vps_layer_dec_pic_buff_minus1[i][k][j] plus 1 specifies the maximum number of decoded pictures, of the k-th layer for the CVS in the i-th output layer set, that need to be stored in the DPB when HighestTid is equal to j. When j is greater than 0, max_vps_layer_dec_pic_buff_minus1[i][k][j] shall be greater than or equal to max_vps_layer_dec_pic_buff_minus1[i][k][j - 1]. When max_vps_layer_dec_pic_buff_minus1[i][k][j] is not present for j in the range of 0 to MaxSubLayersInLayerSetMinus1[i], inclusive, it is inferred to be equal to max_vps_layer_dec_pic_buff_minus1[i][k][j - 1].

[0211] In HEVC (JCTVC-L1003), SHVC (JCTVC-N1008) and MV-HEVC (JCT3V-E1004) it is required that the value of TemporalId shall be the same for all VCL NAL units of an access unit. The value of TemporalId of an access unit is the value of the TemporalId of the VCL NAL units of the access unit.

[0212] For HEVC an access unit is defined as a set of NAL units that are associated with each other according to a specified classification rule, are consecutive in decoding order, and contain exactly one coded picture.

[0213] In SHVC and MV-HEVC an access unit is defined as a set of NAL units that are associated with each other according to a specified classification rule, are consecutive in decoding order, and contain the VCL NAL units of all coded pictures associated with the same output time and their associated non-VCL NAL units.

[0214] In SHVC and MV-HEVC IRAP pictures are allowed to be cross-layer non-aligned. This is helpful in supporting different IRAP frequency for different layers. It also allows flexible placement of IRAP pictures in any layer without requiring an IRAP picture to be coded in the same access unit for other layers. However in HEVC, SHVC and MV-HEVC if nal_unit_type is in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive, i.e. the coded slice segment belongs to an IRAP picture, TemporalId shall be equal to 0.

[0215] Thus although in SHVC and MV-HEVC an IRAP picture could be flexibly coded in any layer in an access unit without requiring an IRAP picture in other layers in the same access unit, it is still currently required that when an IRAP picture is coded in any layer in an access unit then all the other layers in the same access unit must have coded pictures with TemporalId equal to 0. It is asserted that this puts unnecessary restrictions on the flexibility of coding structures that can be supported. For example following scenario is currently not supported in SHVC and MV-HEVC.

[0216] If a particular layer (e.g. base layer) is coded with an all intra configuration where each coded picture is an IRAP picture then all the collocated pictures in those access units for all the other layers must be coded with TemporalId equal to 0 (either as IRAP pictures or as non-IRAP pictures with TemporalId equal to 0) which means that the temporal sub-layering could not be used for those pictures. This limitation is shown in FIG. 51. Thus with current SHVC and MV-HEVC specification the coding configuration can only be similar to as shown in FIG. 51 where all the coded pictures of base layer are IRAP pictures. In this case all the coded pictures in the same AU for enhancement layer 1 must be coded with TemporalId equal to 0.

[0217] Changes in the TemporalID alignment to support more flexible coding structure are described below. The described changes allow the a more flexible coding structure to be supported in SHVC and MV-HEVC. Thus with the changes described below the coding structure as shown in FIG. 52 is supported. In FIG. 52 coding structure the base layer consists of coded pictures which are all IRAP pictures and thus have a TemporalId equal to 0. But the enhancement layer 1 pictures in the same AU can be coded with TemporalId different than TemporalId 0. Thus the Enhancement layer 1 picture can have a TemporalId 1 in the same AU where base layer picture is an IRAP picture and has a TemporalId equal to 0.

[0218] The changes to achieve this flexibility in SHVC and MV-HEVC are described next.

[0219] Non-intra random access point (Non-IRAP) access unit is defined as an 'access unit' in which the 'coded picture' is not an 'IRAP picture'.

[0220] Non-intra random access point (Non-IRAP) picture is defined as a coded 'picture' for which each 'VCL NAL unit' has nal_unit_type with a VCL NAL unit type value other than any value in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive.

[0221] It can be noted that a non-IRAP picture is a picture which is not a BLA picture, a CRA picture or an IDR picture.

[0222] The nuh_temporal_id_plus1 minus 1 specifies a temporal identifier for the NAL unit. The value of nuh_temporal_id_plus1 shall not be equal to 0.

[0223] The variable TemporalId may be specified as TemporalId = nuh_temporal_id_plus1 - 1.

[0224] If nal_unit_type is in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive, i.e. the coded slice segment belongs to an IRAP picture, TemporalId shall be equal to 0. Otherwise, when nal_unit_type is equal to TSA_R, TSA_N, STSA_R, or STSA_N, TemporalId shall not be equal to 0.

[0225] The value of TemporalId shall be the same for all VCL NAL units of all non-IRAP coded pictures in an access unit. If in an access unit all VCL NAL units have a nal_unit_type in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive, i.e. the coded slice segments belongs to an IRAP picture, the value of Temporal ID of the access unit is 0. Otherwise the value of TemporalId of an access unit is the value of the TemporalId of the VCL NAL units of non-IRAP coded pictures in the access unit.

[0226] The value of TemporalId for non-VCL NAL units is constrained as follows:

If nal_unit_type is equal to VPS_NUT or SPS_NUT, TemporalId shall be equal to 0 and the TemporalId of the access unit containing the NAL unit shall be equal to 0.

Otherwise if nal_unit_type is equal to EOS_NUT or EOB_NUT, TemporalId shall be equal to 0.

Otherwise, if nal_unit_type is equal to AUD_NUT or FD_NUT, TemporalId shall be equal to the TemporalId of the access unit containing the NAL unit.

Otherwise, TemporalId shall be greater than or equal to the TemporalId of the access unit containing the NAL unit.

[0227] It can be noted that When the NAL unit is a non-VCL NAL unit, the value of TemporalId is equal to the minimum value of the TemporalId values of all access units to which the non-VCL NAL unit applies. When nal_unit_type is equal to PPS_NUT, TemporalId may be greater than or equal to the TemporalId of the containing access unit, as all PPSs may be included in the beginning of a bitstream, wherein the first coded picture has TemporalId equal to 0. When nal_unit_type is equal to PREFIX_SEI_NUT or SUFFIX_SEI_NUT, TemporalId may be greater than or equal to the TemporalId of the containing access unit, as an SEI NAL unit may contain information, e.g. in a buffering period SEI message or a picture timing SEI message, that applies to a bitstream subset that includes access units for which the TemporalId values are greater than the TemporalId of the access unit containing the SEI NAL unit.

[0228] In a variant embodiment the value of TemporalId shall be the same for all VCL NAL units with nal_unit_type equal to any value except values in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive in an access unit. If in an access unit all VCL NAL units have a nal_unit_type in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive, i.e. the coded slice segment belongs to an IRAP picture, the value of Temporal ID of the access unit is 0. Otherwise the value of TemporalId of an access unit is the value of the TemporalId of the VCL NAL units of non-IRAP coded pictures in the access unit.

[0229] In another variant embodiment the value of TemporalId shall be the same for all VCL NAL units with nal_unit_type equal to any value except values in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive in an access unit. The value of TemporalId of an access unit is the value of the highest TemporalId of the VCL NAL units in the access unit.

[0230] In a further variant embodiment the value of TemporalId shall be the same for all VCL NAL units of all non-IRAP coded pictures in an access unit. The value of TemporalId of an access unit is the value of the highest TemporalId of the VCL NAL units in the access unit.

[0231] As mentioned previously in HEVC (JCTVC-L1003), SHVC (JCTVC-N1008) and MV-HEVC (JCT3V-E1004) it is required that the value of TemporalId shall be the same for all VCL NAL units of an access unit.

[0232] Also in HEVC, SHVC, and MV-HEVC if nal_unit_type is in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive, i.e. the coded slice segment belongs to an IRAP picture, TemporalId shall be equal to 0.

[0233] It is also required that when nal_unit_type is equal to TSA_R, TSA_N, STSA_R, or STSA_N, TemporalId shall not be equal to 0.

[0234] Also in HEVC, SHVC, and MV-HEVC there are also further restrictions as follows:

When one picture picA of a layer layerA has nal_unit_type equal to TSA_N or TSA_R, each picture in the same access unit as picA in a direct or indirect reference layer of layerA shall have nal_unit_type equal to TSA_N or TSA_R.

When one picture picA of a layer layerA has nal_unit_type equal to STSA_N or STSA_R, each picture in the same access unit as picA in a direct or indirect reference layer of layerA shall have nal_unit_type equal to STSA_N or STSA_R.

[0235] Thus with all the current restrictions in HEVC, SHVC, and MV-HEVC a layer could not code a TSA or STSA picture when any other picture in the same access unit is an IRAP picture. Also a TSA or STSA picture must be coded in this case in direct and indirect reference layers of a layer. This current limitation is shown in FIG. 53 which results in a less flexibility in coding structure. In FIG. 53 enhancement layer 1 is using base layer as its direct reference layer. When a TSA picture is coded in enhancement layer1, a TSA picture must be coded in the same access unit in the base layer. Similarly when a STSA picture is coded in enhancement layer1, a STSA picture must be coded in the same access unit in the base layer. This limits flexibility.

[0236] In a more flexible scenario if an IDR picture could be coded in one of the direct or indirect reference layers and TSA or STSA picture could be coded in other layer(s) then temporal layer upswitching at that access unit would still be supported. FIG. 54 shows such a flexible coding structure. In coding structure in FIG. 54 when a TSA picture is coded in enhancement layer 1, a TSA picture could be coded in the same access unit in the base layer similar to FIG. 53. This scenario is not shown in FIG. 54 but is supported. Additionally as shown in FIG. 54 at output time t2 when a TSA picture is coded in enhancement layer1, an IDR picture (or in a variant embodiment an IRAP picture) could be coded in the same access unit in the base layer. Similarly as shown in FIG. 54 at output time t3 when a STSA picture is coded in enhancement layer1, an IDR picture (or in a variant embodiment an IRAP picture) could be coded in the same access unit in the base layer. Additionally in coding structure in FIG. 54 when a STSA picture is coded in enhancement layer 1, a STSA picture could be coded in the same access unit in the base layer similar to FIG. 53. This scenario is not shown in FIG. 54 but is supported. The overall flexibility shown in FIG. 54 is currently disallowed by SHVC and MV-HEVC.

[0237] Changes to the alignment of TSA and STSA pictures to support more flexible coding structure are described next. These changes allow example coding structure shown in FIG. 54 and other similar flexbile coding structure when using TSA and STSA pictures.

[0238] nal_unit_type specifies the type of RBSP data structure contained in the NAL unit as specified in Table (1).

[0239] When one picture picA of a layer layerA has nal_unit_type equal to TSA_N or TSA_R, each picture in the same access unit as picA in a direct or indirect reference layer of layerA shall have nal_unit_type equal to TSA_N or TSA_R or IDR_W_RADL or IDR_N_LP.

[0240] When one picture picA of a layer layerA has nal_unit_type equal to STSA_N or STSA_R, each picture in the same access unit as picA in a direct or indirect reference layer of layerA shall have nal_unit_type equal to STSA_N or STSA_R or IDR_W_RADL or IDR_N_LP.

[0241] In a variant embodiment: nal_unit_type specifies the type of RBSP data structure contained in the NAL unit as specified in Table (1).

[0242] When one picture picA of a layer layerA has nal_unit_type equal to TSA_N or

TSA_R, each picture in the same access unit as picA in a direct or indirect reference layer of layerA shall have nal_unit_type equal to TSA_N or TSA_R or IDR_N_LP.

[0243] When one picture picA of a layer layerA has nal_unit_type equal to STSA_N or STSA_R, each picture in the same access unit as picA in a direct or indirect reference layer of layerA shall have nal_unit_type equal to STSA_N or STSA_R or IDR_N_LP.

[0244] In a variant embodiment: nal_unit_type specifies the type of RBSP data structure contained in the NAL unit as specified in Table (1).

[0245] When one picture picA of a layer layerA has nal_unit_type equal to TSA_N or TSA_R, each picture in the same access unit as picA in a direct or indirect reference layer of layerA shall have nal_unit_type equal to TSA_N or TSA_R or IDR_W_RADL or IDR_N_LP or BLA_W_LP or BLA_W_RADL or BLA_N_LP.

[0246] When one picture picA of a layer layerA has nal_unit_type equal to STSA_N or STSA_R, each picture in the same access unit as picA in a direct or indirect reference layer of layerA shall have nal_unit_type equal to STSA_N or STSA_R or IDR_W_RADL or IDR_N_LP or BLA_W_LP or BLA_W_RADL or BLA_N_LP.

[0247] In a variant embodiment: nal_unit_type specifies the type of RBSP data structure contained in the NAL unit as specified in Table (1).

[0248] When one picture picA of a layer layerA has nal_unit_type equal to TSA_N or TSA_R, each picture in the same access unit as picA in a direct or indirect reference layer of layerA shall have nal_unit_type equal to TSA_N or TSA_R or IDR_W_RADL or IDR_N_LP or BLA_W_LP or BLA_W_RADL or BLA_N_LP or CRA_NUT.

[0249] When one picture picA of a layer layerA has nal_unit_type equal to STSA_N or STSA_R, each picture in the same access unit as picA in a direct or indirect reference layer of layerA shall have nal_unit_type equal to STSA_N or STSA_R or IDR_W_RADL or IDR_N_LP or BLA_W_LP or BLA_W_RADL or BLA_N_LP or CRA_NUT.

[0250] In a variant embodiment: nal_unit_type specifies the type of RBSP data structure contained in the NAL unit as specified in Table (1).

[0251] When one picture picA of a layer layerA has nal_unit_type equal to TSA_N or TSA_R, each picture in the same access unit as picA in a direct or indirect reference layer of layerA shall have nal_unit_type equal to TSA_N or TSA_R or or nal_unit_type is in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive.

[0252] When one picture picA of a layer layerA has nal_unit_type equal to STSA_N or STSA_R, each picture in the same access unit as picA in a direct or indirect reference layer of layerA shall have nal_unit_type equal to STSA_N or STSA_R or nal_unit_type is in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive.

[0253] nuh_layer_id specifies the identifier of the layer.

[0254] When nal_unit_type is equal to AUD_NUT, the value of nuh_layer_id shall be equal to the minimum of the nuh_layer_id values of all VCL NAL units in the access unit.

[0255] When nal_unit_type is equal to VPS_NUT, the value of nuh_layer_id shall be equal to 0. Decoder shall ignore NAL units with nal_unit_type equal to VPS_NUT and nuh_layer_id greater than 0.

[0256] nuh_temporal_id_plus1 minus 1 specifies a temporal identifier for the NAL unit. The value of nuh_temporal_id_plus1 shall not be equal to 0.

[0257] The variable TemporalId is specified as follows:

$$\text{TemporalId} = \text{nuh\_temporal\_id\_plus1} - 1 \qquad (7\text{-}1)$$

If nal_unit_type is in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive, i.e. the coded slice segment belongs to an IRAP picture, TemporalId shall be equal to 0. Otherwise, when nal_unit_type is equal to TSA_R, TSA_N, STSA_R, or STSA_N, TemporalId shall not be equal to 0.

The value of TemporalId shall be the same for all VCL NAL units of all non-IRAP coded pictures in an access unit. If in an access unit all VCL NAL units have a nal_unit_type in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive, i.e. the coded slice segment belongs to an IRAP picture, the value of Temporal ID of the access unit is 0. Otherwise the value of TemporalId of an access unit is the value of the TemporalId of the VCL NAL units of non-IRAP coded pictures in the access unit.

[0258] The value of TemporalId for non-VCL NAL units is constrained as follows:

If nal_unit_type is equal to VPS_NUT or SPS_NUT, TemporalId shall be equal to 0 and the TemporalId of the access unit containing the NAL unit shall be equal to 0.

Otherwise if nal_unit_type is equal to EOS_NUT or EOB_NUT, TemporalId shall be equal to 0.

Otherwise, if nal_unit_type is equal to AUD_NUT or FD_NUT, TemporalId shall be equal to the TemporalId of the access unit containing the NAL unit.

Otherwise, TemporalId shall be greater than or equal to the TemporalId of the access unit containing the NAL unit.

When the NAL unit is a non-VCL NAL unit, the value of TemporalId is equal to the minimum value of the TemporalId values of all access units to which the non-VCL NAL unit applies. When nal_unit_type is equal to PPS_NUT, TemporalId may be greater than or equal to the TemporalId of the containing access unit, as all PPSs may be included in the beginning of a bitstream, wherein the first coded picture has TemporalId equal to 0. When nal_unit_type is equal to PREFIX_SEI_NUT or SUFFIX_SEI_NUT, TemporalId may be greater than or equal to the TemporalId of the containing access unit, as an SEI NAL unit may contain information, e.g. in a buffering period SEI message or a picture timing SEI message, that applies to a bitstream subset that includes access units for which the TemporalId values are greater than the

TemporalId of the access unit containing the SEI NAL unit.

[0259] It is to be understood that any of the features, whether indicated as shall or necessary, may be omitted as desired. In addition, the features may be combined in different combinations, as desired.

[0260] The term "computer-readable medium" refers to any available medium that can be accessed by a computer or a processor. The term "computer-readable medium," as used herein, may denote a computer- and/or processor-readable medium that is non-transitory and tangible. By way of example, and not limitation, a computer-readable or processor-readable medium may comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium that can be used to carry or store desired program code in the form of instructions or data structures and that can be accessed by a computer or processor. Disk and disc, as used herein, includes compact disc (CD), laser disc, optical disc, digital versatile disc (DVD), floppy disk and Blu-ray (registered trademark) disc where disks usually reproduce data magnetically, while discs reproduce data optically with lasers.

[0261] It should be noted that one or more of the methods described herein may be implemented in and/or performed using hardware. For example, one or more of the methods or approaches described herein may be implemented in and/or realized using a chipset, an ASIC, a large-scale integrated circuit (LSI) or integrated circuit, etc.

[0262] Each of the methods disclosed herein comprises one or more steps or actions for achieving the described method. The method steps and/or actions may be interchanged with one another and/or combined into a single step without departing from the scope of the claims. In other words, unless a specific order of steps or actions is required for proper operation of the method that is being described, the order and/or use of specific steps and/or actions may be modified without departing from the scope of the claims.

[0263] It is to be understood that the claims are not limited to the precise configuration and components illustrated above. Various modifications, changes and variations may be made in the arrangement, operation and details of the systems, methods, and apparatus described herein without departing from the scope of the claims.

Claims

[Claim 1] A method for decoding a video bitstream comprising the steps of:
(a) receiving said video bitstream that includes a layer set, where said layer set identifies a plurality of different layers of said bitstream, where at least one of said plurality of different layers includes a plurality of temporal sub-layers;
(b) receiving a video parameter set that includes information related to at least one layer of said video bitstream;
(c) receiving a video parameter set extension referenced by said video parameter set that includes data regarding said plurality of different layers and said plurality of temporal sub-layers;
(d) receiving a video parameter set temporal sub layers information present flag in said video parameter set extension indicating whether said information about plurality of temporal sub-layers are present.

[Claim 2] The method of claim 1 wherein said information about plurality of temporal sub-layers indicates a maximum value minus one of said plurality of temporal sub-layers that may be present for said plurality of different layers.

[Claim 3] The method of claim 2 wherein said video parameter set sub layers present flag equal to 1 indicates said presence of information about said plurality of temporal sub-layers being present.

[Claim 4] The method of claim 3 wherein said video parameter set sub layers present flag equal to 0 indicates said presence of information about said plurality of temporal sub-layers not being present.

[Claim 5] The method of claim 4 wherein said syntax elements sub_layers_vps_max_minus1[i] is present when said video parameter set sub layers present flag is said equal to 1.

[Claim 6] The method of claim 5 wherein said syntax elements sub_layers_vps_max_minus1[i] is not present when said video parameter set sub layers present flag is said equal to 0.

[Claim 7] The method of claim 2 wherein said information about plurality of temporal sub-layers indicates a maximum value minus one of said plurality of temporal sub-layers that may be present for said plurality of different layers is indicated by the syntax element sub_layers_vps_max_minus1[i] for layer with nuh_layer_id equal to layer_id_in_nuh[i].

[Claim 8] A method for decoding a video bitstream comprising the steps of:

(a) receiving said video bitstream that includes a layer set, where said layer set identifies a plurality of different layers of said bitstream, where at least one of said plurality of different layers includes a plurality of temporal sub-layers;
(b) receiving a video parameter set extension that includes data regarding said plurality of different layers and said plurality of sub-layers;
(d) receiving for 0 to a maximum number of temporal sub-layers for a particular layer set (1) a bit rate present flag; (2) a picture rate present flag; (3) bit rate information (4) picture rate information.

[Claim 9] The method of claim 8 wherein said maximum number of temporal sub-layers for said particular layer set is less than or equal to a number of temporal sub-layers that are capable of being present in said video for said layer set.

[Claim 10] The method of claim 9 wherein said video said data regarding said plurality of different layers and said plurality of temporal sub-layers is included in a video parameter set extension.

[Claim 11] The method of claim 10 wherein a first one of said layers of said layer set has a first number of temporal sub-layers that may be present, wherein a second one of said layers of said layer set has a second number of temporal sub-layers that may be present, where said first number of temporal sub-layers is different than said second number of temporal sub-layers, and said maximum number of temporal sub-layers for said layer set is the greater of said first number of temporal sub-layers and said second number of temporal sub-layers.

[Claim 12] The method of claim 11 wherein said maximum number is said maximum number minus 1.

[Claim 13] The method of claim 10 wherein said maximum number of temporal sub-layers is said maximum number of temporal sub-layers minus 1.

[Claim 14] The method of claim 8 said maximum number of temporal sub-layers is said maximum number of temporal sub-layers minus 1.

[Claim 15] The method of claim 8 where said receiving (1) a bit rate present flag; (2) a picture rate present flag; (3) bit rate information (4) picture rate information does not include receiving information for temporal sub-layers from maximum number of temporal sub-layers for a particular layer set+1 to maximum number of temporal sub layers that may be present in the bitstream.

[Claim 16] A method for decoding a video bitstream comprising the steps of:

(a) receiving said video bitstream that includes a plurality of different layers, where at least one of said plurality of different layers includes a plurality of temporal sub-layers;
(b) receiving said video bitstream that includes a first slice as a portion of a first frame of one of said plurality of temporal sub-layers;
(c) receiving said video bitstream that includes a second slice as a portion of a second frame of a different one of said plurality of temporal sub-layers;
(d) receiving a first slice segment header that includes information related to said first slice of said video bitstream;
(e) comparing a temporal sub layers maximum value from video parameter set with a temporal identifier of said second frame to determine whether to include said second slice as an active reference layer picture for said first slice that may be used for inter layer prediction for said first slice.

[Claim 17] The method of claim 16 wherein said comparing is based upon a temporal sub layers maximum value from video parameter set.

[Claim 18] The method of claim 17 wherein said comparing is based upon said sub layer video parameter set maximum minus 1.

[Claim 19] The method of claim 18 wherein a total number of said active reference layer pictures for said first slice is determined.

[Claim 20] The method of claim 19 wherein said total number of said active reference layer pictures is NumActiveRefLayerPics.

[Claim 21] The method of claim 16 wherein said second slice may be used as an active reference layer picture for said first slice that may be used for inter layer prediction for said first slice if a temporal sub layers maximum value from video parameter set is greater than or equal to a temporal identifier of said second frame.

[Claim 22] The method of claim 16 wherein said second slice may not be used as an active reference layer picture for said first slice if a temporal sub layers maximum value from video parameter set is less than a temporal identifier of said second frame.

[Claim 23] A method for decoding a video bitstream comprising the steps of:
(a) receiving said video bitstream that includes a plurality of different layers, where at least one of said plurality of different layers includes a plurality of temporal sub-layers;
(b) receiving said video bitstream that includes a first slice as a portion of a first frame of one of said plurality of temporal sub-layers;

(c) receiving a first slice segment header that includes information related to said first slice of said video bitstream;
(d) receiving a temporal identifier and nal unit type with said first slice segment header;
(e) if said nal unit type is an IRAP picture then a TemporalId that is derived based upon said temporal identifier is equal to 0;
(f) if said nal unit type is at least one of TSA and TSA_N then said TemporalId is not equal to 0;
(g) if said nal unit type is at least one of STSA_R and STSA_N then said TemporalId is not equal to 0.

[Fig. 1A]

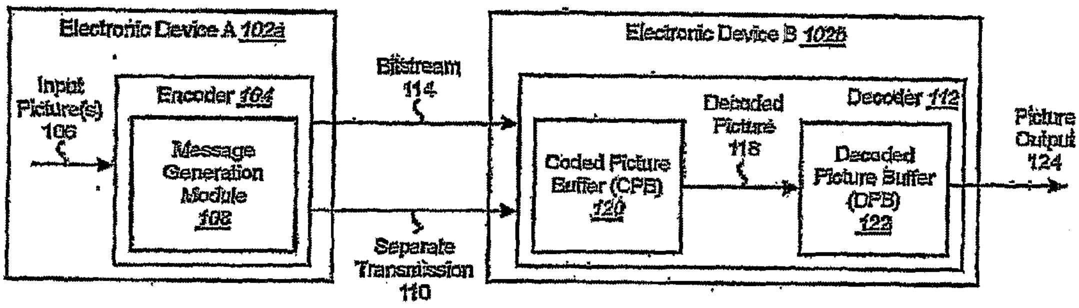

Electronic Device A 102a
Input Picture(s) 106
Encoder 104
Message Generation Module 108
Bitstream 114
Separate Transmission 110
Electronic Device B 102b
Decoder 112
Coded Picture Buffer (CPB) 120
Decoded Picture 118
Decoded Picture Buffer (DPB) 122
Picture Output 124

FIG. 1A

[Fig. 1B]

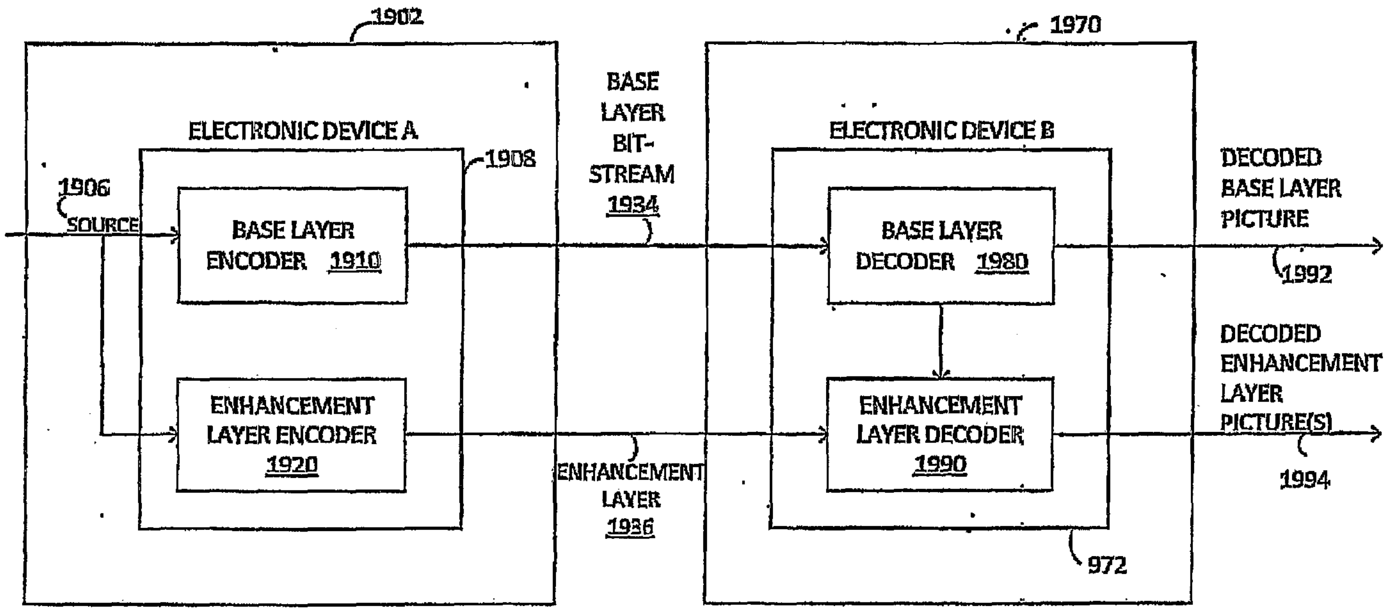

1902
ELECTRONIC DEVICE A
1906
SOURCE
1908
BASE LAYER ENCODER 1910
ENHANCEMENT LAYER ENCODER 1920
BASE LAYER BIT-STREAM 1934
ENHANCEMENT LAYER 1936
1970
ELECTRONIC DEVICE B
BASE LAYER DECODER 1980
ENHANCEMENT LAYER DECODER 1990
972
DECODED BASE LAYER PICTURE
1992
DECODED ENHANCEMENT LAYER PICTURE(S)
1994

FIG. 1B

[Fig. 2A]

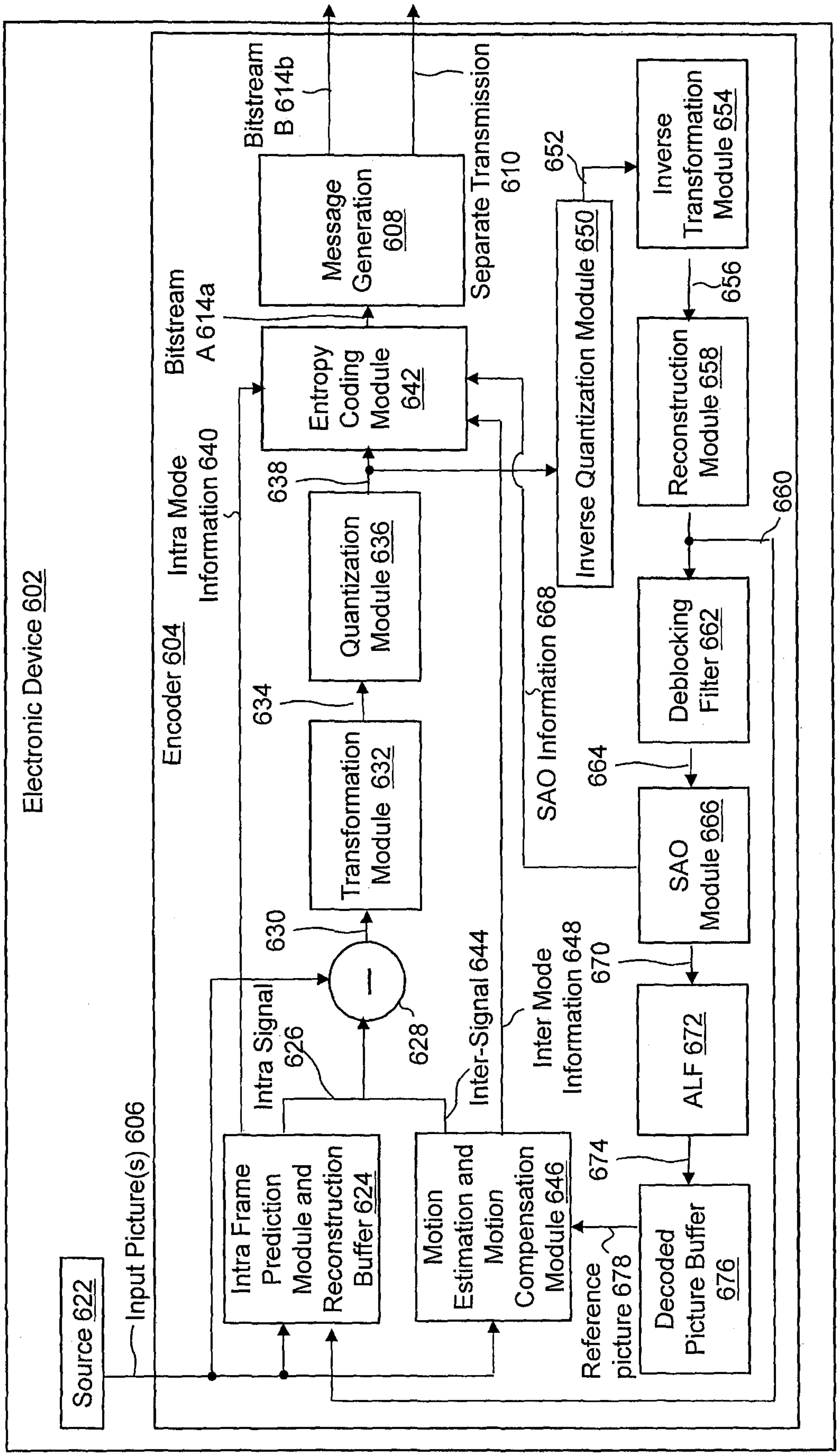

Electronic Device 602
Encoder 604
Source 622
Input Picture(s) 606
Intra Frame Prediction Module and Reconstruction Buffer 624
Motion Estimation and Motion Compensation Module 646
Intra Signal 626
Inter-Signal 644
628
630
Transformation Module 632
634
Quantization Module 636
638
Intra Mode Information 640
Entropy Coding Module 642
Bitstream A 614a
Message Generation 608
Bitstream B 614b
Separate Transmission 610
Inverse Quantization Module 650
652
Inverse Transformation Module 654
656
Reconstruction Module 658
660
Deblocking Filter 662
664
SAO Module 666
SAO Information 668
Inter Mode Information 648
670
ALF 672
674
Decoded Picture Buffer 676
Reference picture 678

FIG. 2A

[Fig. 2B]

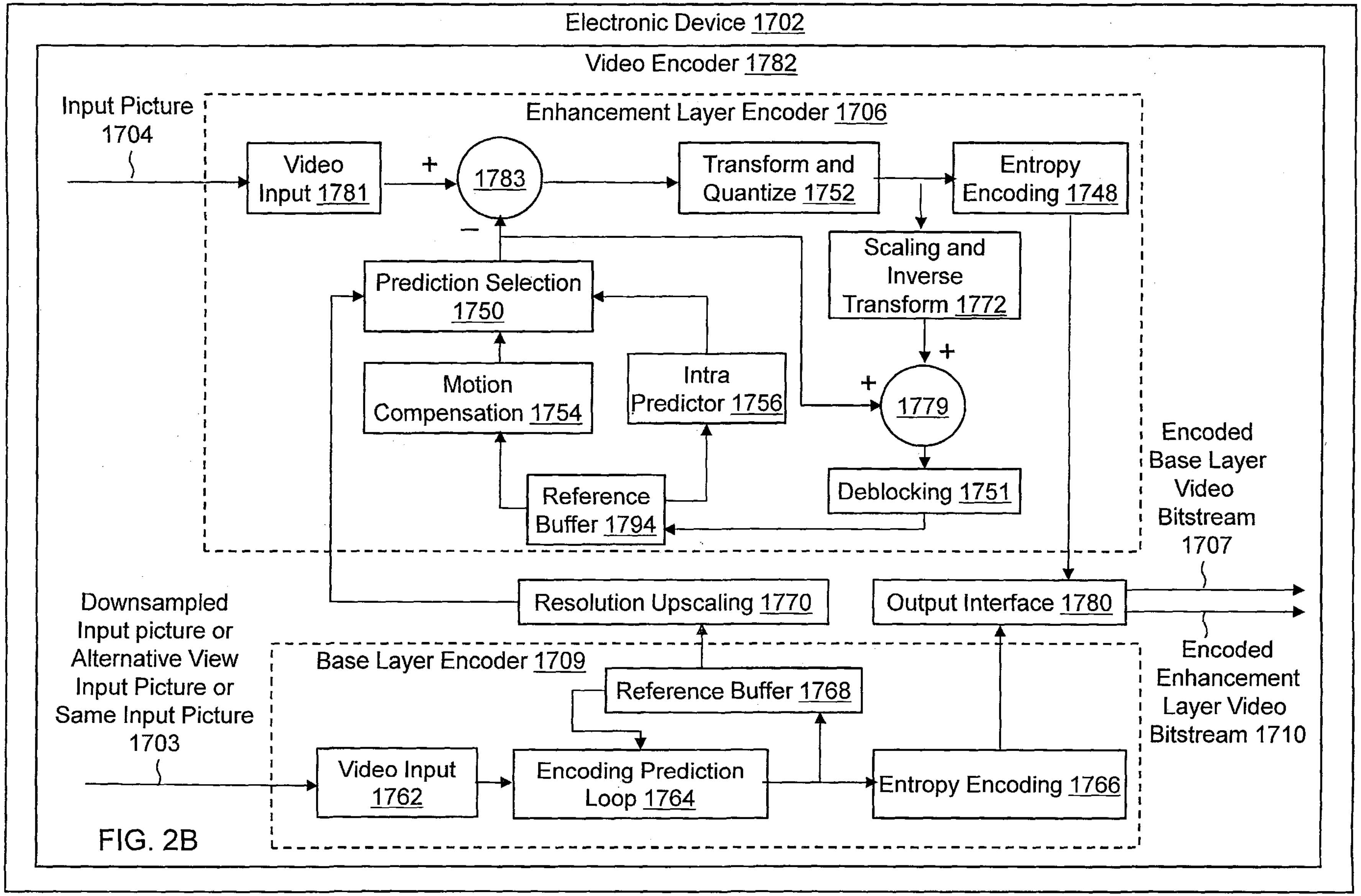

Electronic Device 1702
Video Encoder 1782
Input Picture 1704
Enhancement Layer Encoder 1706
Video Input 1781
+
1783
−
Transform and Quantize 1752
Entropy Encoding 1748
Scaling and Inverse Transform 1772
Prediction Selection 1750
Motion Compensation 1754
Intra Predictor 1756
+
+
1779
Deblocking 1751
Reference Buffer 1794
Encoded Base Layer Video Bitstream 1707
Resolution Upscaling 1770
Output Interface 1780
Downsampled Input picture or Alternative View Input Picture or Same Input Picture 1703
Base Layer Encoder 1709
Reference Buffer 1768
Video Input 1762
Encoding Prediction Loop 1764
Entropy Encoding 1766
Encoded Enhancement Layer Video Bitstream 1710

FIG. 2B

[Fig. 3A]

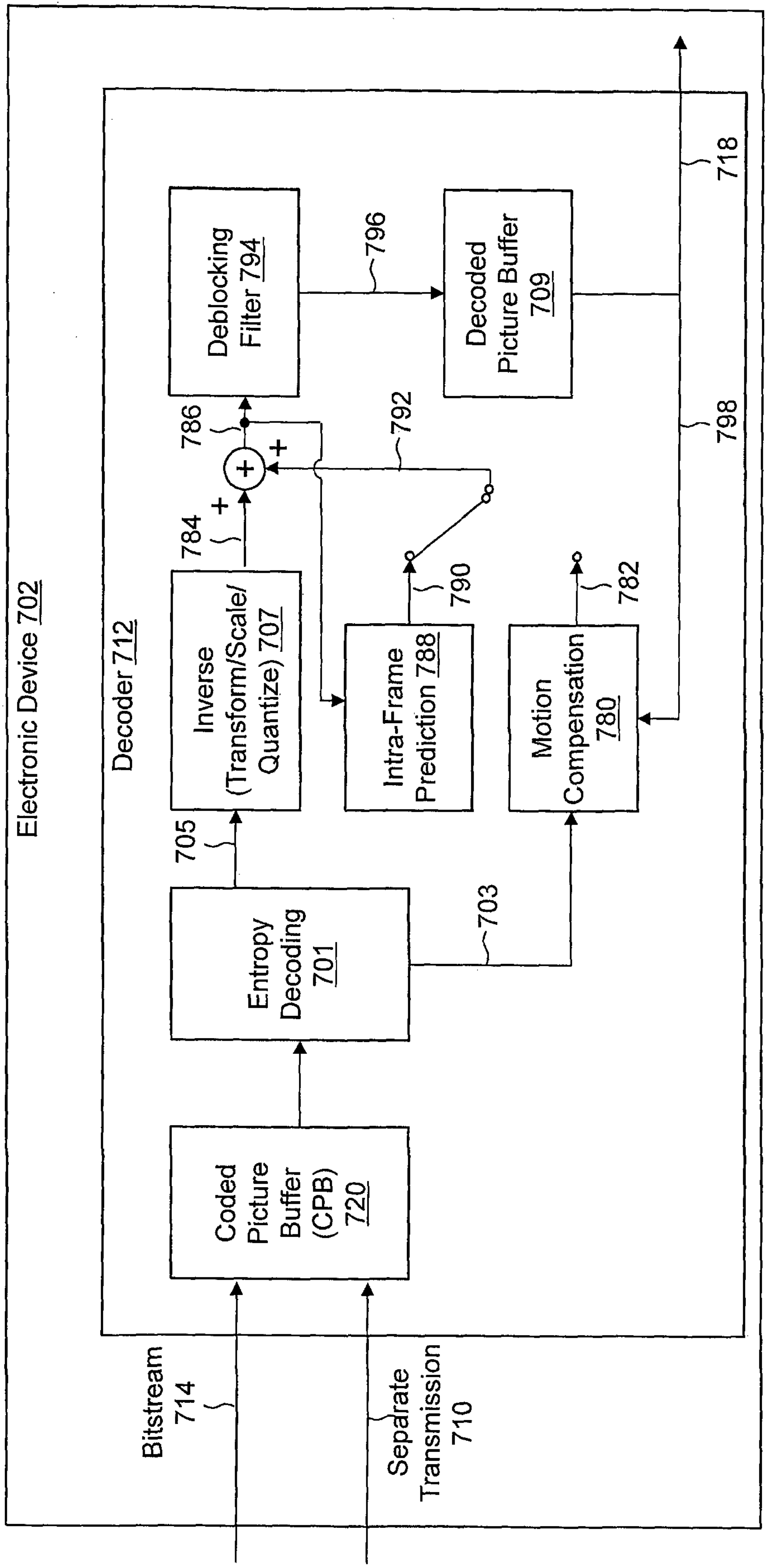

Electronic Device 702
Decoder 712
Coded Picture Buffer (CPB) 720
Entropy Decoding 701
705
Inverse (Transform/Scale/ Quantize) 707
784
+
+
786
Deblocking Filter 794
796
Decoded Picture Buffer 709
718
798
792
Intra-Frame Prediction 788
790
Motion Compensation 780
782
703
Bitstream 714
Separate Transmission 710

FIG. 3A

[Fig. 3B]

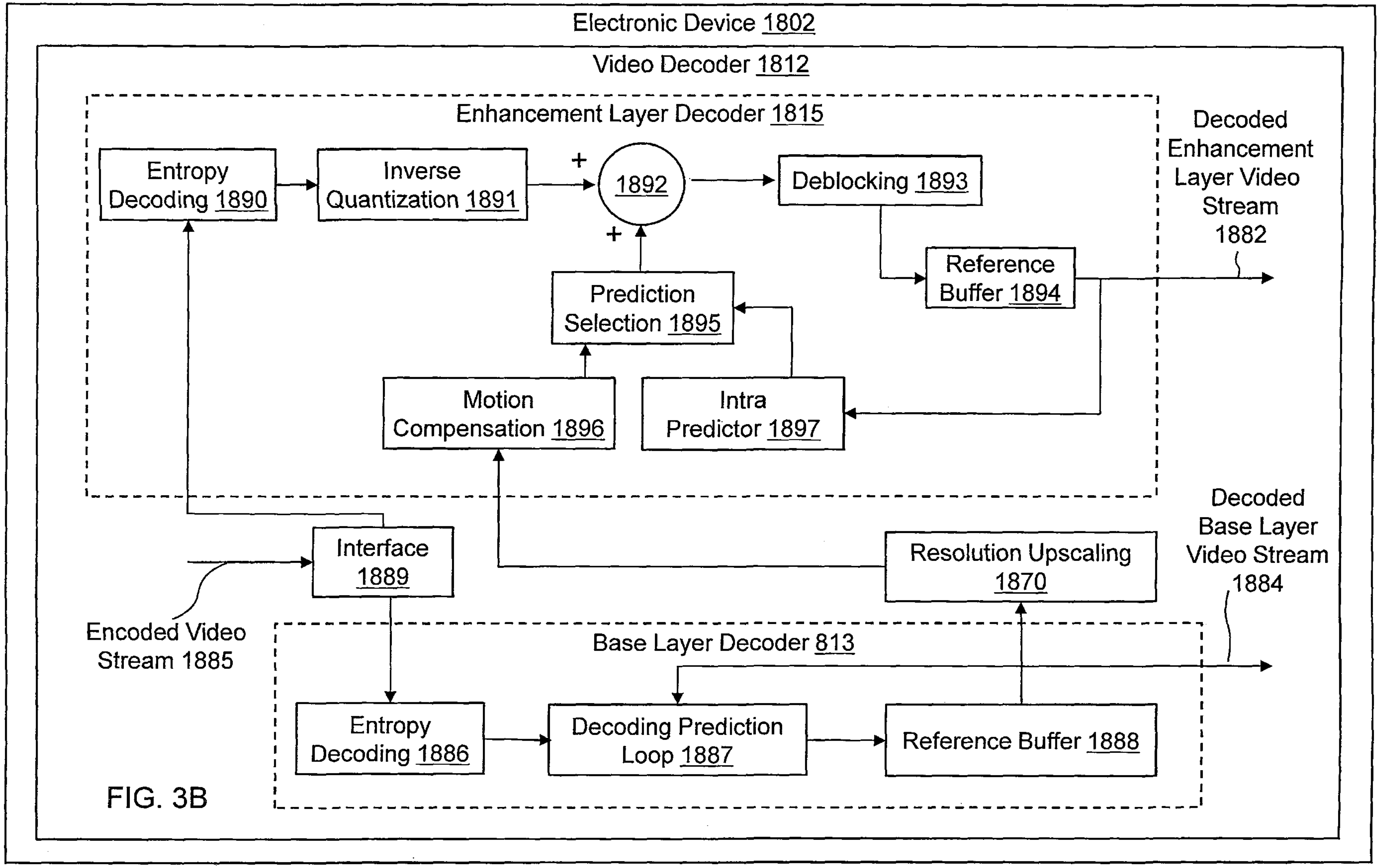

Electronic Device 1802
Video Decoder 1812
Enhancement Layer Decoder 1815
Entropy Decoding 1890
Inverse Quantization 1891
+
1892
+
Deblocking 1893
Reference Buffer 1894
Decoded Enhancement Layer Video Stream 1882
Prediction Selection 1895
Motion Compensation 1896
Intra Predictor 1897
Interface 1889
Encoded Video Stream 1885
Resolution Upscaling 1870
Decoded Base Layer Video Stream 1884
Base Layer Decoder 813
Entropy Decoding 1886
Decoding Prediction Loop 1887
Reference Buffer 1888

FIG. 3B

[Fig. 4]

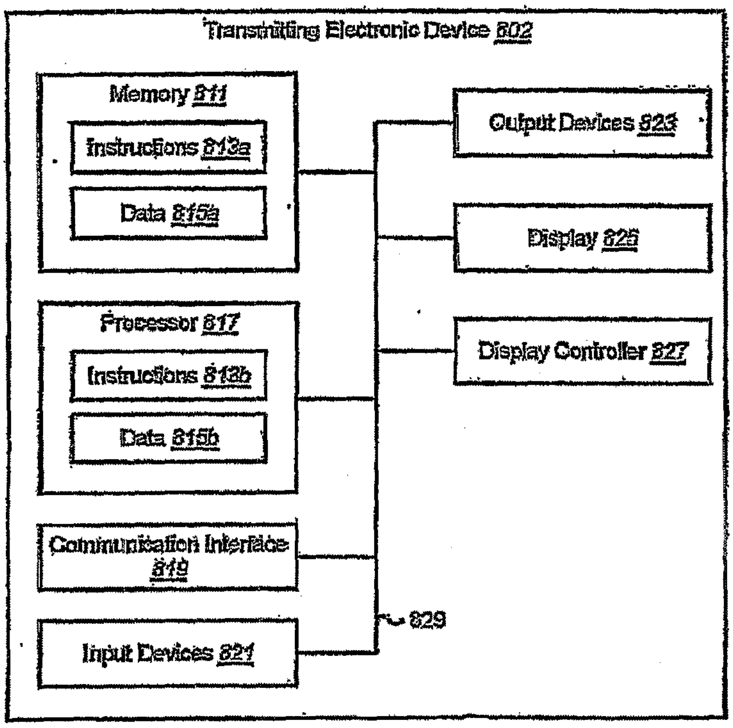


FIG. 4

[Fig. 5]

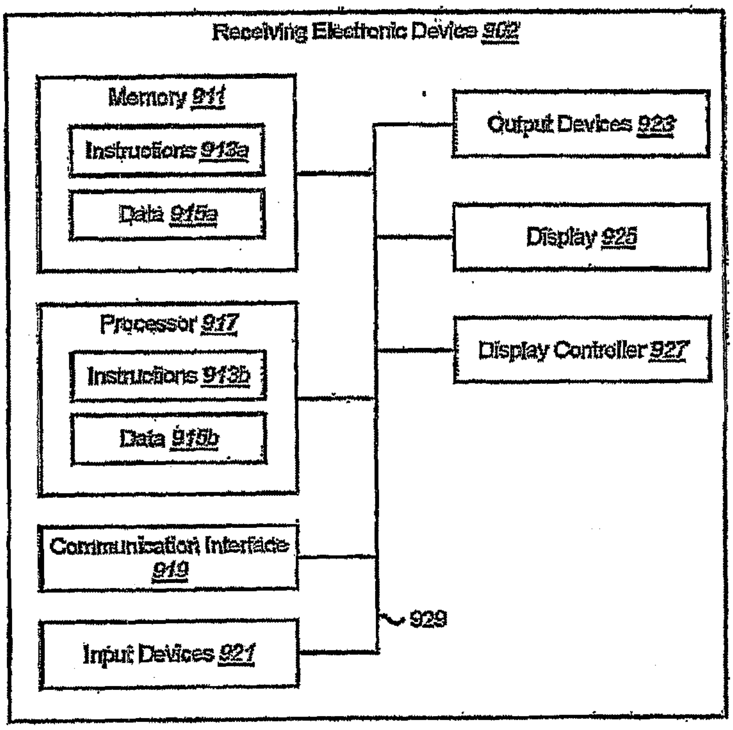


FIG. 5

[Fig. 6]

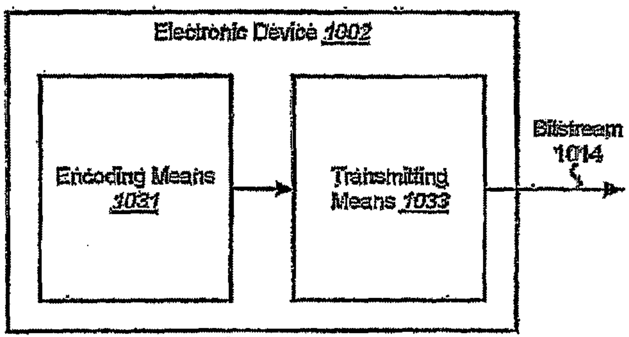


FIG. 6

[Fig. 7]

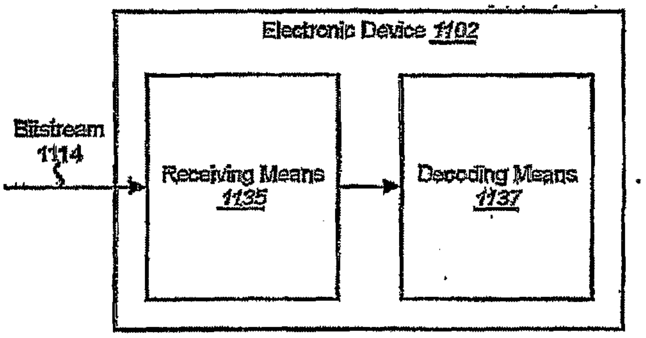


FIG. 7

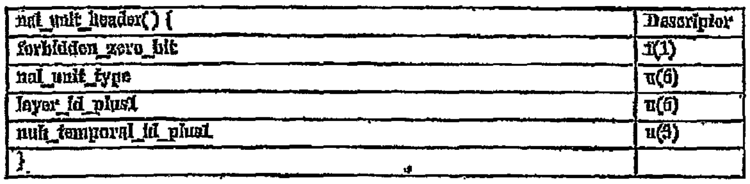
[Fig. 8A]

nal_unit_header( ) {	Descriptor
forbidden_zero_bit	f(1)
nal_unit_type	u(6)
nuh_reserved_zero_6bits	u(6)
nuh_temporal_id_plus1	u(3)
}	

NAL UNIT HEADER SYNTAX

FIG. 8A

[Fig. 8B]



nal_unit_header( ) {	Descriptor
forbidden_zero_bit	f(1)
nal_unit_type	u(6)
layer_id_plus1	u(6)
nuh_temporal_id_plus1	u(3)
}	

NAL UNIT HEADER SYNTAX

FIG. 8B

[Fig. 8C]

nal_unit_header() {	Descriptor
forbidden_zero_bit	f(1)
nal_unit_type	u(6)
layer_id	u(6)
nuh_temporal_id_plus1	u(3)
}	

NAL UNIT HEADER SYNTAX

FIG. 8C

[Fig. 9]

nal_unit(NumBytesInNALunit) {	Descriptor
nal_unit_header()	
NumBytesInRBSP = 0	
for(i = 2; i < NumBytesInNALunit; i++) {	
if(i + 2 < NumBytesInNALunit && next_bits(24) = = 0x000003) {	
rbsp_byte[NumBytesInRBSP++]	b(8)
rbsp_byte[NumBytesInRBSP++]	b(8)
i += 2	
emulation_prevention_three_byte /* equal to 0x03 */	f(8)
} else	
rbsp_byte[NumBytesInRBSP++]	b(8)
}	
}	

GENERAL NAL UNIT SYNTAX

FIG. 9

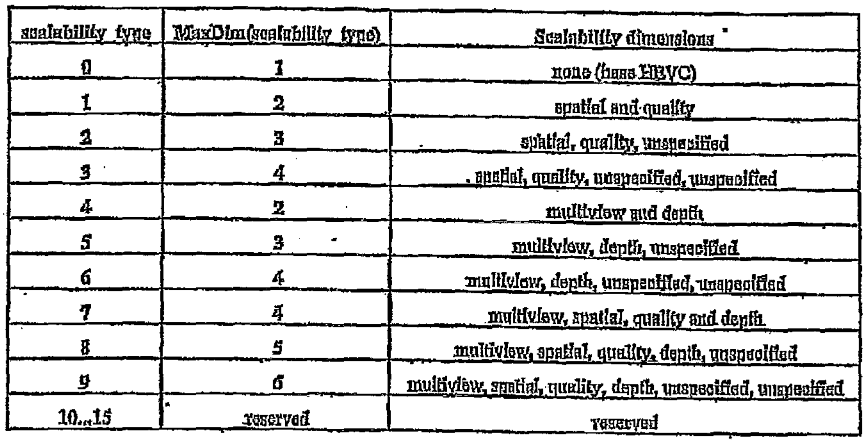
[Fig. 10]

vps_extension( ) {	Descriptor
while(!byte_aligned( ))	
vps_extension_byte_alignment_reserved_zero_bit	u(1)
// scalability type and layer_id partitioning method	
scalability_type	u(4)
for(i = 0; i < MaxDim(scalability_type); i++)	
layer_id_dim_len[i]	u(3)
// layer specific information	
for(i = 0; i <= max_num_layers_minus1; i++) {	
vps_layer_id[i]	u(6)
// layer dependency	
num_direct_ref_layers[i]	u(6)
for(j = 0; j < num_direct_ref_layers[i]; j++)	
ref_layer_id[i][j]	u(6)
}	
}	

Existing video parameter set extension syntax

FIG. 10

[Fig. 11]



scalability type	MaxDim(scalability type)	Scalability dimensions
0	1	none (base HEVC)
1	2	spatial and quality
2	3	spatial, quality, unspecified
3	4	spatial, quality, unspecified, unspecified
4	2	multiview and depth
5	3	multiview, depth, unspecified
6	4	multiview, depth, unspecified, unspecified
7	4	multiview, spatial, quality and depth
8	5	multiview, spatial, quality, depth, unspecified
9	6	multiview, spatial, quality, depth, unspecified, unspecified
10...15	reserved	reserved

Existing Scalability Types

FIG. 11

[Fig. 12]

BASE LAYER
SPS+
PPS+

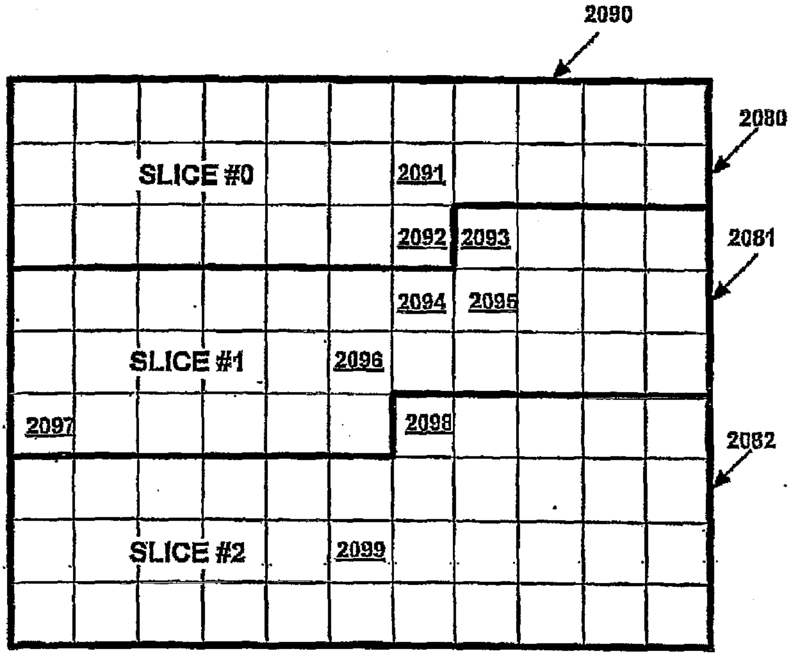
ENHANCEMENT LAYER 0
SPS+
PPS+

ENHANCEMENT LAYER 1
SPS+
PPS+

ENHANCEMENT LAYER 2
SPS+
PPS+

FIG. 12

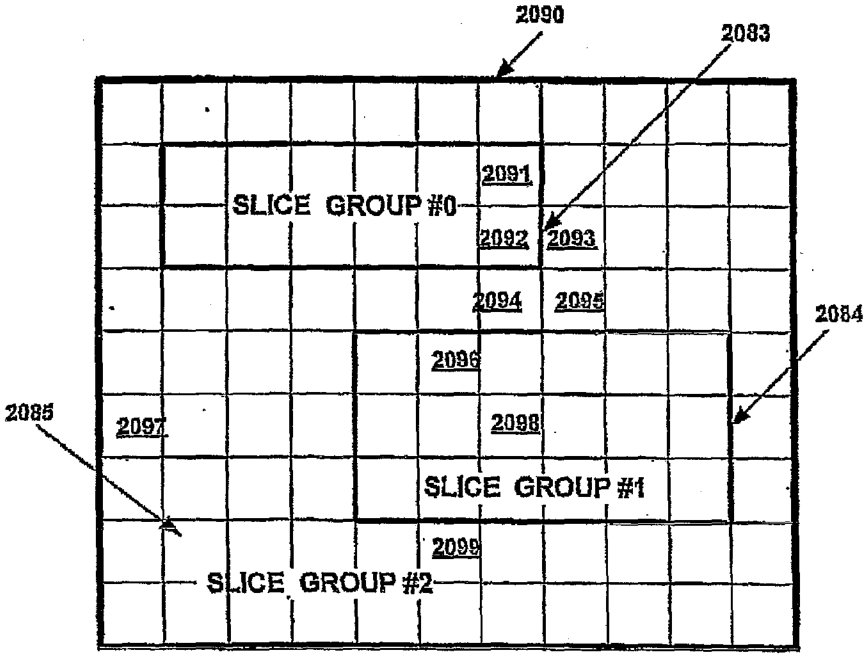
[Fig. 13]



PRIOR ART

FIG. 13

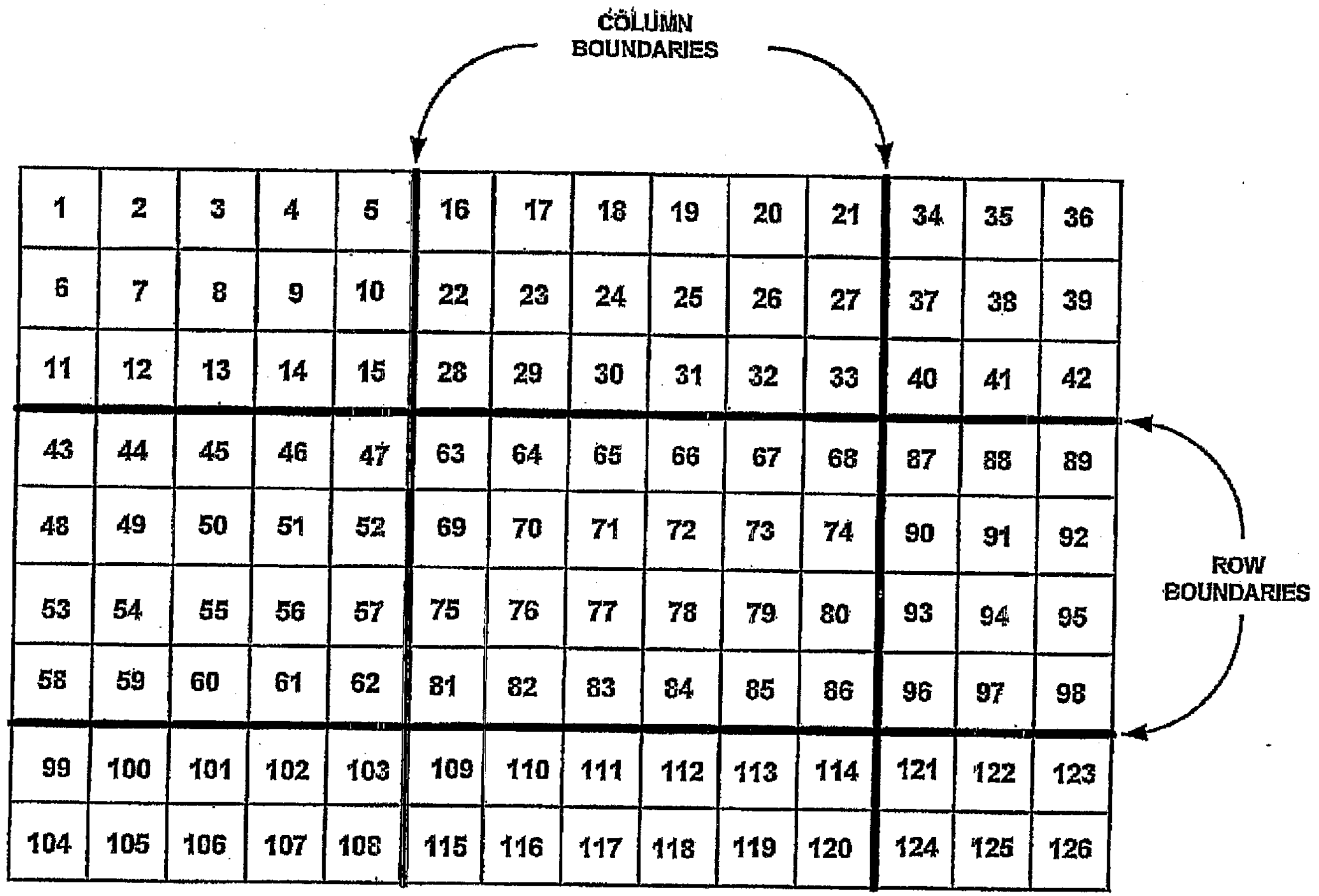
[Fig. 14]



PRIOR ART

FIG. 14

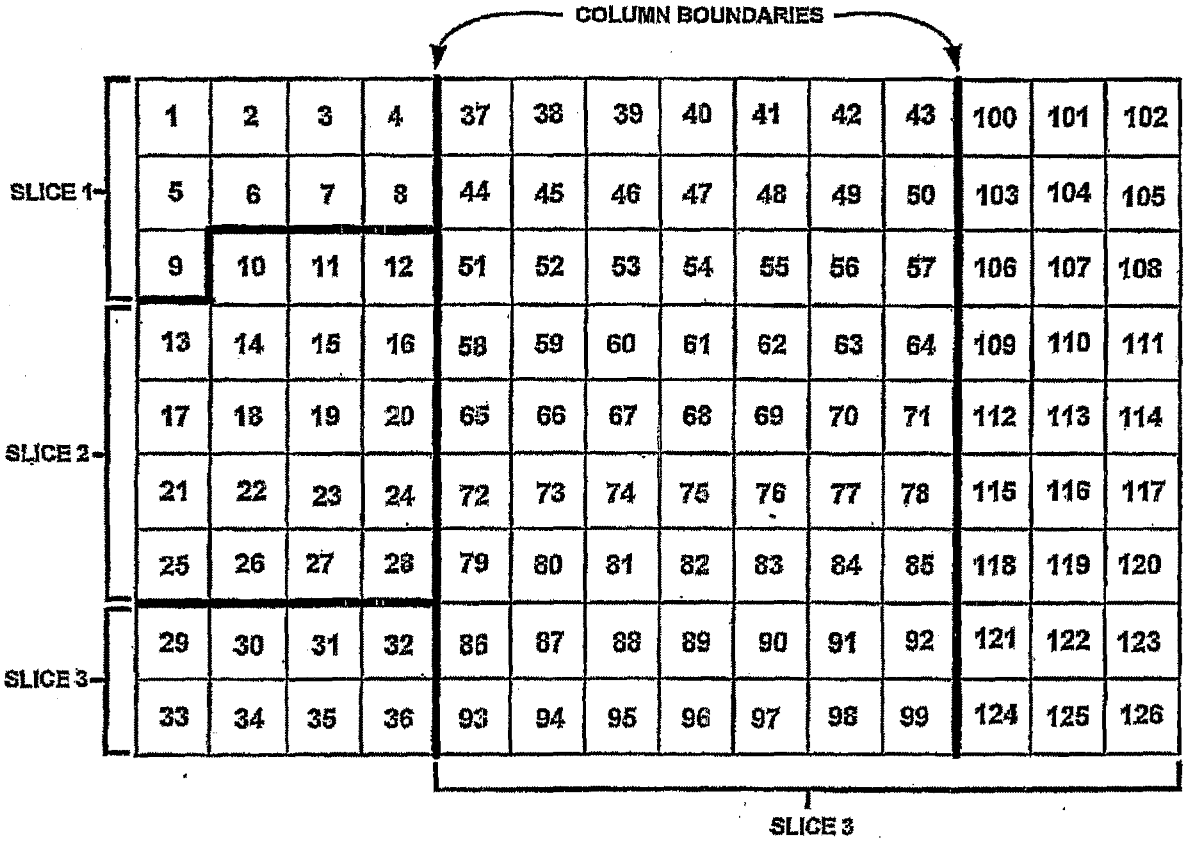
[Fig. 15]



1	2	3	4	5	16	17	18	19	20	21	34	35	36
6	7	8	9	10	22	23	24	25	26	27	37	38	39
11	12	13	14	15	28	29	30	31	32	33	40	41	42
43	44	45	46	47	63	64	65	66	67	68	87	88	89
48	49	50	51	52	69	70	71	72	73	74	90	91	92
53	54	55	56	57	75	76	77	78	79	80	93	94	95
58	59	60	61	62	81	82	83	84	85	86	96	97	98
99	100	101	102	103	109	110	111	112	113	114	121	122	123
104	105	106	107	108	115	116	117	118	119	120	124	125	126

FIG. 15

[Fig. 16]



1	2	3	4	37	38	39	40	41	42	43	100	101	102
5	6	7	8	44	45	46	47	48	49	50	103	104	105
9	10	11	12	51	52	53	54	55	56	57	106	107	108
13	14	15	16	58	59	60	61	62	63	64	109	110	111
17	18	19	20	65	66	67	68	69	70	71	112	113	114
21	22	23	24	72	73	74	75	76	77	78	115	116	117
25	26	27	28	79	80	81	82	83	84	85	118	119	120
29	30	31	32	86	87	88	89	90	91	92	121	122	123
33	34	35	36	93	94	95	96	97	98	99	124	125	126

FIG. 16

[Fig. 17]

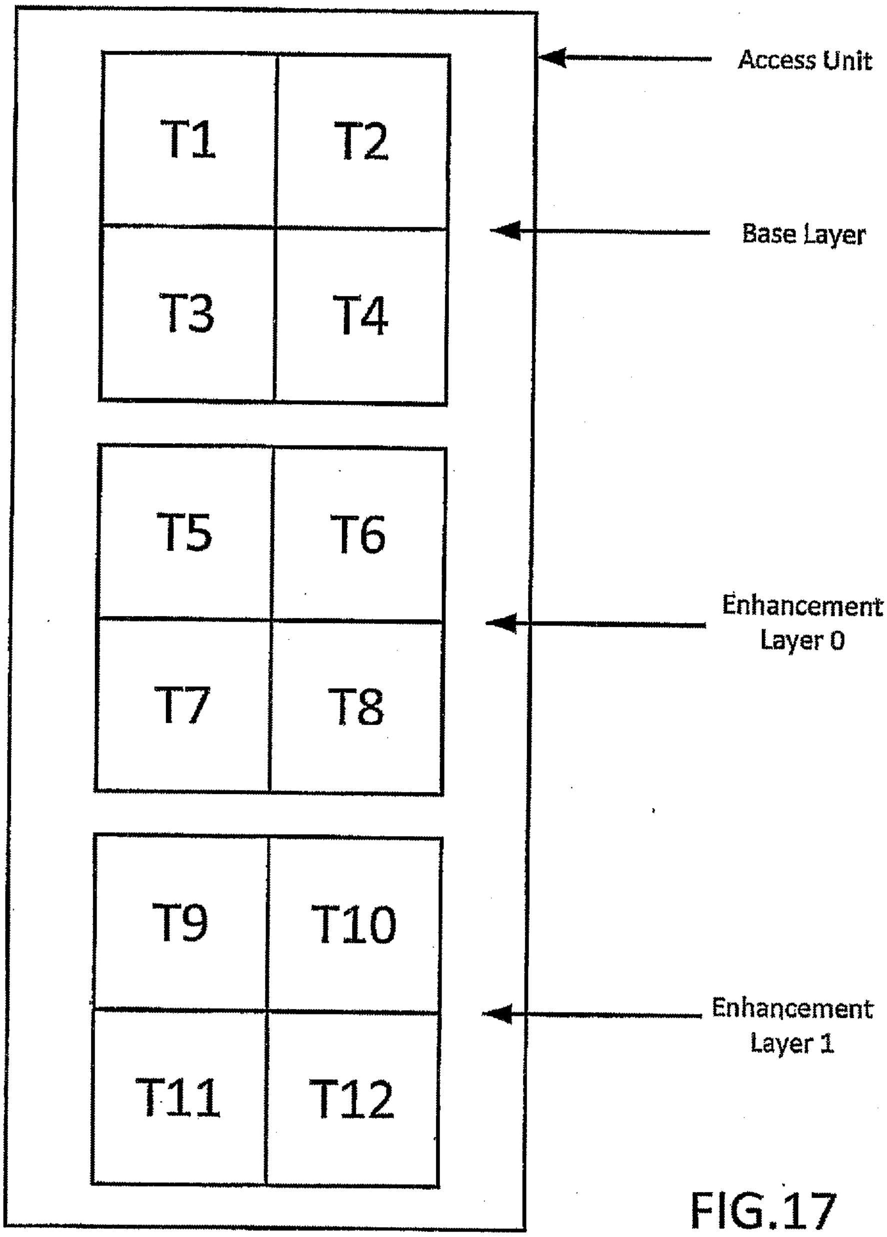

T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
T11
T12
Access Unit
Base Layer
Enhancement Layer 0
Enhancement Layer 1

FIG.17

[Fig. 18A]

slice_segment_header() {	Descriptor
first_slice_segment_in_pic_flag	u(1)
if(nal_unit_type >= BLA_W_LP && nal_unit_type <= RSV_IRAP_VCL23)	
no_output_of_prior_pics_flag	u(1)
slice_pic_parameter_set_id	ue(v)
if(!first_slice_segment_in_pic_flag) {	
if(dependent_slice_segments_enabled_flag)	
dependent_slice_segment_flag	u(1)
slice_segment_address	u(v)
}	
if(!dependent_slice_segment_flag) {	
if(num_extra_slice_header_bits > 0)	
discardable_flag	u(1)
for(i = 1; i < num_extra_slice_header_bits ; i++)	
slice_reserved_flag[i]	u(1)
slice_type	ue(v)
if(output_flag_present_flag)	
pic_output_flag	u(1)
if(separate_colour_plane_flag == 1)	
colour_plane_id	u(2)
if(nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP) {	
slice_pic_order_cnt_lsb	u(v)
short_term_ref_pic_set_sps_flag	u(1)
if(!short_term_ref_pic_set_sps_flag)	
short_term_ref_pic_set(num_short_term_ref_pic_sets)	
else if(num_short_term_ref_pic_sets > 1)	
short_term_ref_pic_set_idx	u(v)
if(long_term_ref_pics_present_flag) {	
if(num_long_term_ref_pics_sps > 0)	
num_long_term_sps	ue(v)
num_long_term_pics	ue(v)
for(i = 0; i < num_long_term_sps + num_long_term_pics; i++) {	
if(i < num_long_term_sps) {	
if(num_long_term_ref_pics_sps > 1)	
lt_idx_sps[i]	u(v)
} else {	
poc_lsb_lt[i]	u(v)
used_by_curr_pic_lt_flag[i]	u(1)
}	
delta_poc_msb_present_flag[i]	u(1)

FIG. 18 A

[Fig. 18B]

if(delta_poc_msb_present_flag[i])	
delta_poc_msb_cycle_lt[i]	ue(v)
}	
}	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
}	
if(nuh_layer_id > 0 && NumDirectRefLayers[nuh_layer_id] > 0) {	
inter_layer_pred_enabled_flag	u(1)
if(inter_layer_pred_enabled_flag && NumDirectRefLayers[nuh_layer_id] > 1) {	
if(lmax_one_active_ref_layer_flag)	
num_inter_layer_ref_pics_minus1	u(v)
for(i = 0; i < NumActiveRefLayerPics; i++)	
Inter_layer_pred_layer_idc[i]	u(v)
}	
}	
if(NumSamplePredRefLayers[nuh_layer_id] > 0 && NumActiveRefLayerPics > 0)	
inter_layer_sample_pred_only_flag	u(1)
if(sample_adaptive_offset_enabled_flag) {	
slice_sao_luma_flag	u(1)
slice_sao_chroma_flag	u(1)
}	
if(slice_type = = P || slice_type = = B) {	
num_ref_idx_active_override_flag	u(1)
if(num_ref_idx_active_override_flag) {	
num_ref_idx_l0_active_minus1	ue(v)
if(slice_type = = B)	
num_ref_idx_l1_active_minus1	ue(v)
}	
if(lists_modification_present_flag && NumPocTotalCurr > 1)	
ref_pic_lists_modification( )	
if(slice_type = = B)	
mvd_l1_zero_flag	u(1)
if(cabac_init_present_flag)	
cabac_init_flag	u(1)
if(slice_temporal_mvp_enabled_flag) {	
if(nuh_layer_id > 0 && NumActiveMotionPredRefLayers > 0)	
alt_collocated_indication_flag	u(1)
if(alt_collocated_indication_flag)	

FIG. 18B

[Fig. 18C]

if(NumActiveMotionPredRefLayers > 1)	
collocated_ref_layer_idx	ue(v)
else {	
if(slice_type = = B)	
collocated_from_l0_flag	u(1)
if((collocated_from_l0_flag && num_ref_idx_l0_active_minus1 > 0) || (!collocated_from_l0_flag && num_ref_idx_l1_active_minus1 > 0))	
collocated_ref_idx	ue(v)
}	
}	
if((weighted_pred_flag && slice_type = = P) || (weighted_bipred_flag && slice_type = = B))	
pred_weight_table( )	
five_minus_max_num_merge_cand	ue(v)
}	
slice_qp_delta	se(v)
if(pps_slice_chroma_qp_offsets_present_flag) {	
slice_cb_qp_offset	se(v)
slice_cr_qp_offset	se(v)
}	
if(deblocking_filter_override_enabled_flag)	
deblocking_filter_override_flag	u(1)
if(deblocking_filter_override_flag) {	
slice_deblocking_filter_disabled_flag	u(1)
if(!slice_deblocking_filter_disabled_flag) {	
slice_beta_offset_div2	se(v)
slice_tc_offset_div2	se(v)
}	
}	
if(pps_loop_filter_across_slices_enabled_flag && (slice_sao_luma_flag || slice_sao_chroma_flag || !slice_deblocking_filter_disabled_flag))	
slice_loop_filter_across_slices_enabled_flag	u(1)
}	
if(tiles_enabled_flag || entropy_coding_sync_enabled_flag) {	
num_entry_point_offsets	ue(v)
if(num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset_minus1[i]	u(v)
}	

FIG. 18 C

[Fig. 18D]

}	
if(slice_segment_header_extension_present_flag) {	
slice_segment_header_extension_length	ue(v)
for(i = 0; i < slice_segment_header_extension_length; i++)	
slice_segment_header_extension_data_byte[i]	u(8)
}	
byte_alignment( )	
}	

FIG. 18 D

[Fig. 19]

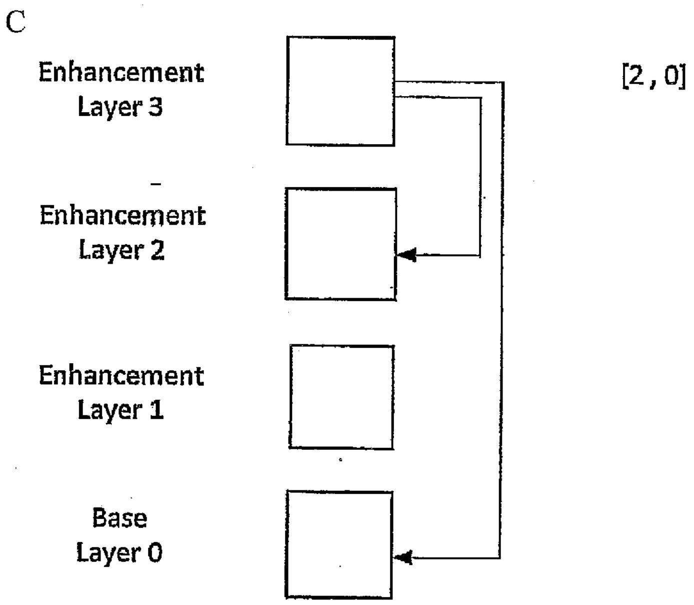

C
Enhancement Layer 3
Enhancement Layer 2
Enhancement Layer 1
Base Layer 0
[2 , 0]

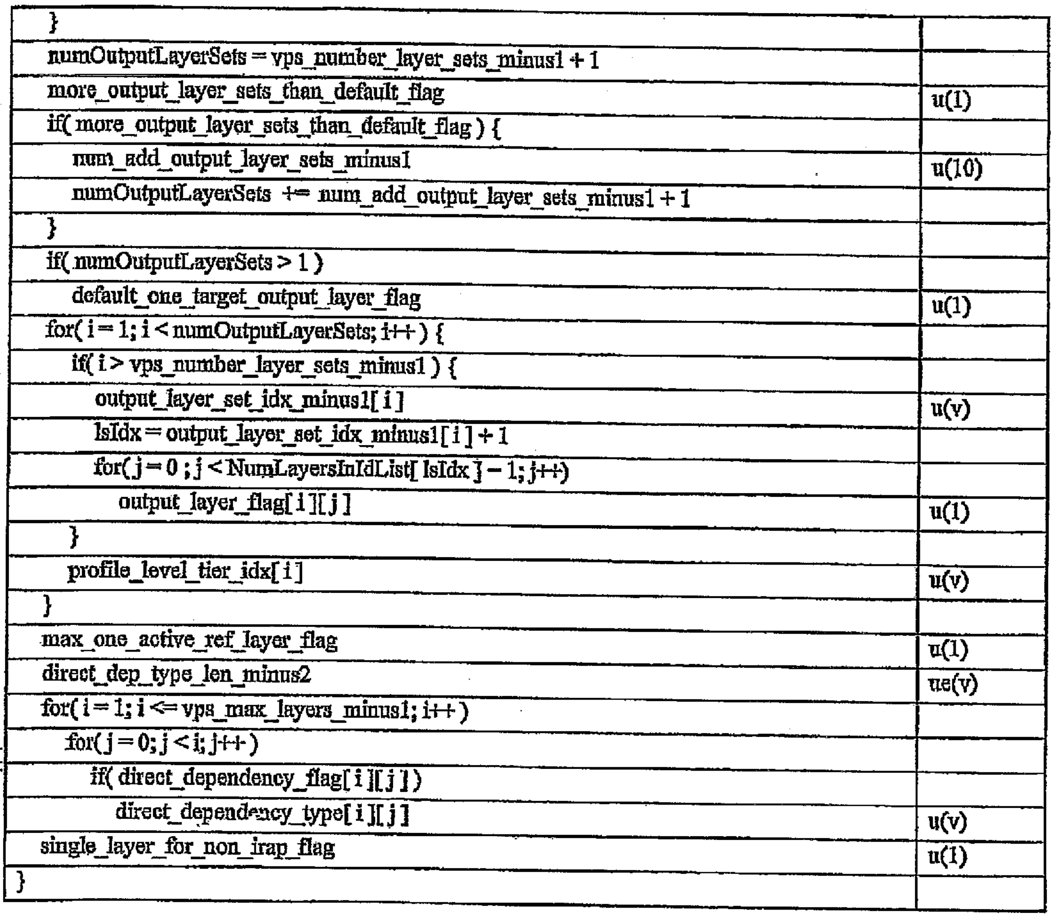
FIG.19

[Fig. 20A]

vps_extension( ) {	Descriptor
while(!byte_aligned( ))	
vps_extension_byte_alignment_reserved_one_bit	u(1)
avc_base_layer_flag	u(1)
splitting_flag	u(1)
for(i = 0, NumScalabilityTypes = 0; i < 16; i++) {	
scalability_mask[i]	u(1)
NumScalabilityTypes += scalability_mask[i]	
}	
for(j = 0; j < (NumScalabilityTypes – splitting_flag); j++)	
dimension_id_len_minus1[j]	u(3)
vps_nuh_layer_id_present_flag	u(1)
for(i = 0; i <= vps_max_layers_minus1; i++) {	
if(vps_nuh_layer_id_present_flag && i > 0)	
layer_id_in_nuh[i]	u(6)
if(!splitting_flag)	
for(j = 0; j < NumScalabilityTypes; j++)	
dimension_id[i][j]	u(v)
}	
for(i = 1; i <= vps_max_layers_minus1; i++)	
for(j = 0; j < i; j++)	
direct_dependency_flag[i][j]	u(1)
for(i = 0; i< vps_max_layers_minus1; i++)	
max_tid_il_ref_pics_plus1[i]	u(3)
vps_number_layer_sets_minus1	u(10)
vps_num_profile_tier_level_minus1	u(6)
for(i = 1; i <= vps_num_profile_tier_level_minus1; i ++) {	
vps_profile_present_flag[i]	u(1)
if(!vps_profile_present_flag[i])	
profile_ref_minus1[i]	u(6)
profile_tier_level(vps_profile_present_flag[i], vps_max_sub_layers_minus1)	

FIG. 20A

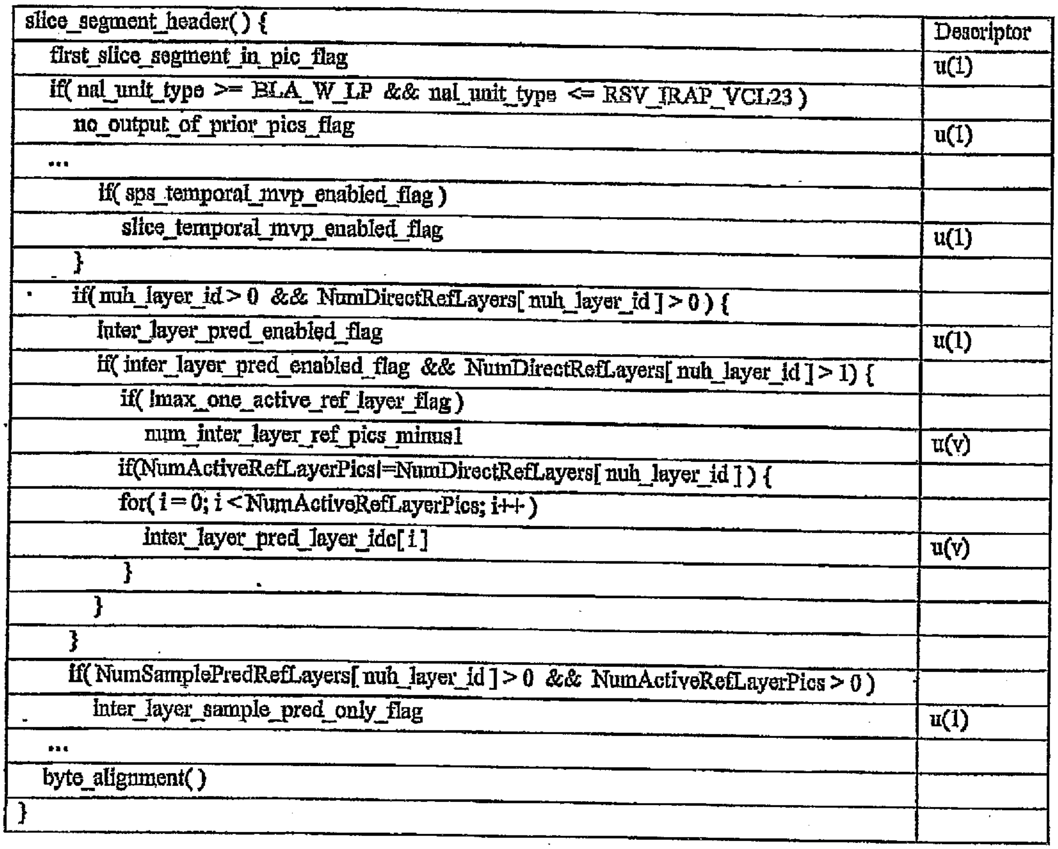
[Fig. 20B]



}	
numOutputLayerSets = vps_number_layer_sets_minus1 + 1	
more_output_layer_sets_than_default_flag	u(1)
if(more_output_layer_sets_than_default_flag) {	
num_add_output_layer_sets_minus1	u(10)
numOutputLayerSets += num_add_output_layer_sets_minus1 + 1	
}	
if(numOutputLayerSets > 1)	
default_one_target_output_layer_flag	u(1)
for(i = 1; i < numOutputLayerSets; i++) {	
if(i > vps_number_layer_sets_minus1) {	
output_layer_set_idx_minus1[i]	u(v)
lsIdx = output_layer_set_idx_minus1[i] + 1	
for(j = 0 ; j < NumLayersInIdList[lsIdx] − 1; j++)	
output_layer_flag[i][j]	u(1)
}	
profile_level_tier_idx[i]	u(v)
}	
max_one_active_ref_layer_flag	u(1)
direct_dep_type_len_minus2	ue(v)
for(i = 1; i <= vps_max_layers_minus1; i++)	
for(j = 0; j < i; j++)	
if(direct_dependency_flag[i][j])	
direct_dependency_type[i][j]	u(v)
single_layer_for_non_irap_flag	u(1)
}	

FIG. 20 B

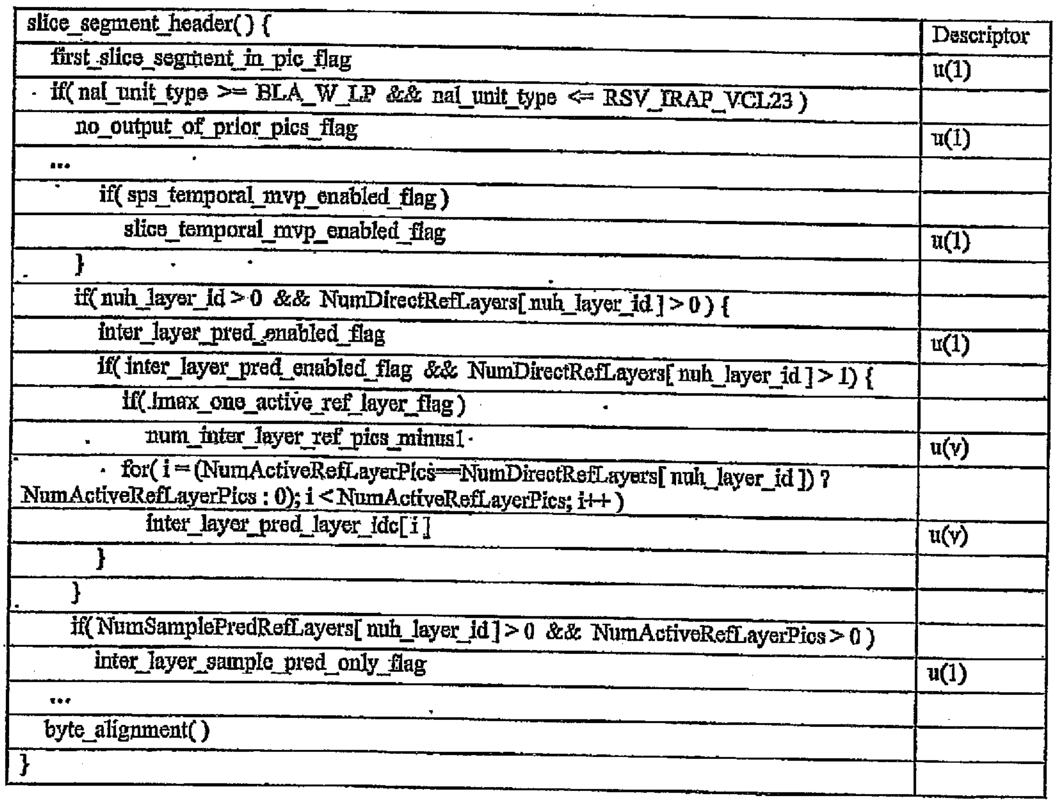
[Fig. 21]



slice_segment_header( ) {	Descriptor
first_slice_segment_in_pic_flag	u(1)
if(nal_unit_type >= BLA_W_LP && nal_unit_type <= RSV_IRAP_VCL23)	
no_output_of_prior_pics_flag	u(1)
...	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
}	
if(nuh_layer_id > 0 && NumDirectRefLayers[nuh_layer_id] > 0) {	
inter_layer_pred_enabled_flag	u(1)
if(inter_layer_pred_enabled_flag && NumDirectRefLayers[nuh_layer_id] > 1) {	
if(!max_one_active_ref_layer_flag)	
num_inter_layer_ref_pics_minus1	u(v)
if(NumActiveRefLayerPics!=NumDirectRefLayers[nuh_layer_id]) {	
for(i = 0; i < NumActiveRefLayerPics; i++)	
inter_layer_pred_layer_idc[i]	u(v)
}	
}	
}	
if(NumSamplePredRefLayers[nuh_layer_id] > 0 && NumActiveRefLayerPics > 0)	
inter_layer_sample_pred_only_flag	u(1)
...	
byte_alignment( )	
}	

FIG. 21

[Fig. 22]



slice_segment_header( ) {	Descriptor
first_slice_segment_in_pic_flag	u(1)
if(nal_unit_type >= BLA_W_LP && nal_unit_type <= RSV_IRAP_VCL23)	
no_output_of_prior_pics_flag	u(1)
...	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
}	
if(nuh_layer_id > 0 && NumDirectRefLayers[nuh_layer_id] > 0) {	
inter_layer_pred_enabled_flag	u(1)
if(inter_layer_pred_enabled_flag && NumDirectRefLayers[nuh_layer_id] > 1) {	
if(!max_one_active_ref_layer_flag)	
num_inter_layer_ref_pics_minus1	u(v)
for(i = (NumActiveRefLayerPics==NumDirectRefLayers[nuh_layer_id]) ? NumActiveRefLayerPics : 0); i < NumActiveRefLayerPics; i++)	
inter_layer_pred_layer_idc[i]	u(v)
}	
}	
if(NumSamplePredRefLayers[nuh_layer_id] > 0 && NumActiveRefLayerPics > 0)	
inter_layer_sample_pred_only_flag	u(1)
...	
byte_alignment( )	
}	

FIG. 22

[Fig. 23]

slice_segment_header( ) {	Descriptor
first_slice_segment_in_pic_flag	u(1)
If(nal_unit_type >= BLA_W_LP && nal_unit_type <= RSV_IRAP_VCL23)	
no_output_of_prior_pics_flag	u(1)
...	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
}	
if(nuh_layer_id > 0 && NumDirectRefLayers[nuh_layer_id] > 0) {	
inter_layer_pred_enabled_flag	u(1)
if(inter_layer_pred_enabled_flag && NumDirectRefLayers[nuh_layer_id] > 1) {	
if(!max_one_active_ref_layer_flag)	
num_inter_layer_ref_pics_minus1	u(v)
for(i = 0; i < NumDirectRefLayers[nuh_layer_id]; i++)	
inter_layer_pred_layer_mask[i]	u(1)
}	
}	
if(NumSamplePredRefLayers[nuh_layer_id] > 0 && NumActiveRefLayerPics > 0)	
inter_layer_sample_pred_only_flag	u(1)
...	
byte_alignment( )	
}	

FIG. 23

[Fig. 24]

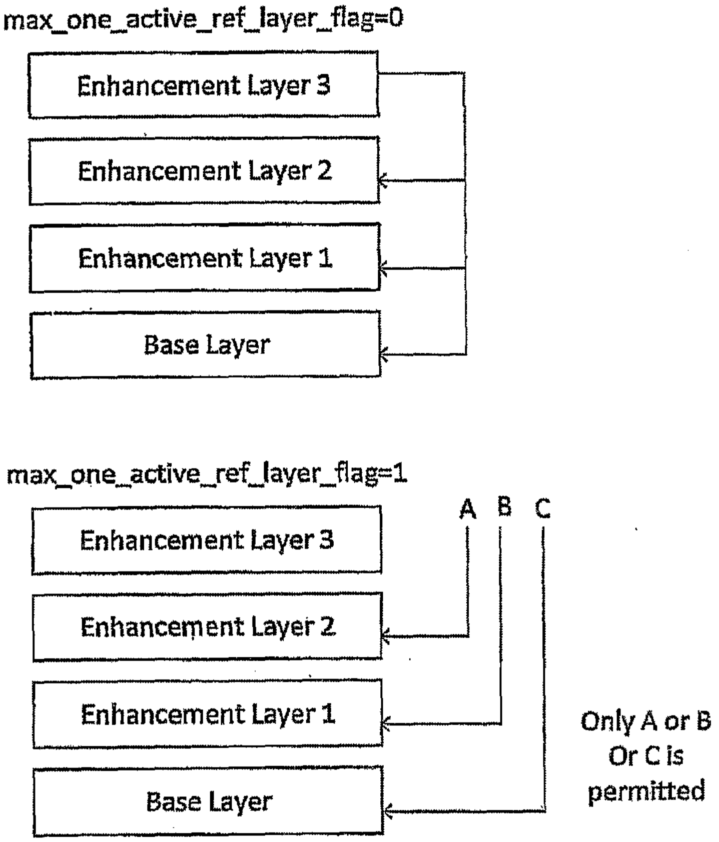

max_one_active_ref_layer_flag=0
Enhancement Layer 3
Enhancement Layer 2
Enhancement Layer 1
Base Layer
max_one_active_ref_layer_flag=1
A
B
C
Enhancement Layer 3
Enhancement Layer 2
Enhancement Layer 1
Base Layer
Only A or B
Or C is
permitted

FIG. 24

[Fig. 25]

slice_segment_header( ) {	Descriptor
first_slice_segment_in_pic_flag	u(1)
if(nal_unit_type >= BLA_W_LP && nal_unit_type <= RSV_IRAP_VCL23)	
no_output_of_prior_pics_flag	u(1)
...	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
}	
if(ilp_slice_signaling_enabled_flag) {	
if(nuh_layer_id > 0 && NumDirectRefLayers[nuh_layer_id] > 0) {	
inter_layer_pred_enabled_flag	u(1)
if(inter_layer_pred_enabled_flag && NumDirectRefLayers[nuh_layer_id] > 1) {	
if(!max_one_active_ref_layer_flag)	
num_inter_layer_ref_pics_minus1	u(v)
for(i = 0; i < NumActiveRefLayerPics; i++)	
inter_layer_pred_layer_idc[i]	u(v)
}	
}	
}	
if(NumSamplePredRefLayers[nuh_layer_id] > 0 && NumActiveRefLayerPics > 0)	
inter_layer_sample_pred_only_flag	u(1)
...	
byte_alignment( )	
}	

FIG. 25

[Fig. 26A]

vps_extension( ) {	Descriptor
while(!byte_aligned( ))	
vps_extension_byte_alignment_reserved_one_bit	u(1)
avc_base_layer_flag	u(1)
splitting_flag	u(1)
for(i = 0, NumScalabilityTypes = 0; i < 16; i++) {	
scalability_mask[i]	u(1)
NumScalabilityTypes += scalability_mask[i]	
}	
for(j = 0; j < (NumScalabilityTypes – splitting_flag); j++)	
dimension_id_len_minus1[j]	u(3)
vps_nuh_layer_id_present_flag	u(1)
for(i = 1; i <= vps_max_layers_minus1; i++) {	
if(vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)
if(!splitting_flag)	
for(j = 0; j < NumScalabilityTypes; j++)	
dimension_id[i][j]	u(v)
}	
for(i = 1; i <= vps_max_layers_minus1; i++)	
for(j = 0; j < i; j++)	
direct_dependency_flag[i][j]	u(1)
for(i = 0; i< vps_max_layers_minus1; i++)	
max_tid_il_ref_pics_plus1[i]	u(3)
if(nuh_layer_id>0)	
ilp_slice_signaling_enabled_flag	u(1)
.....	

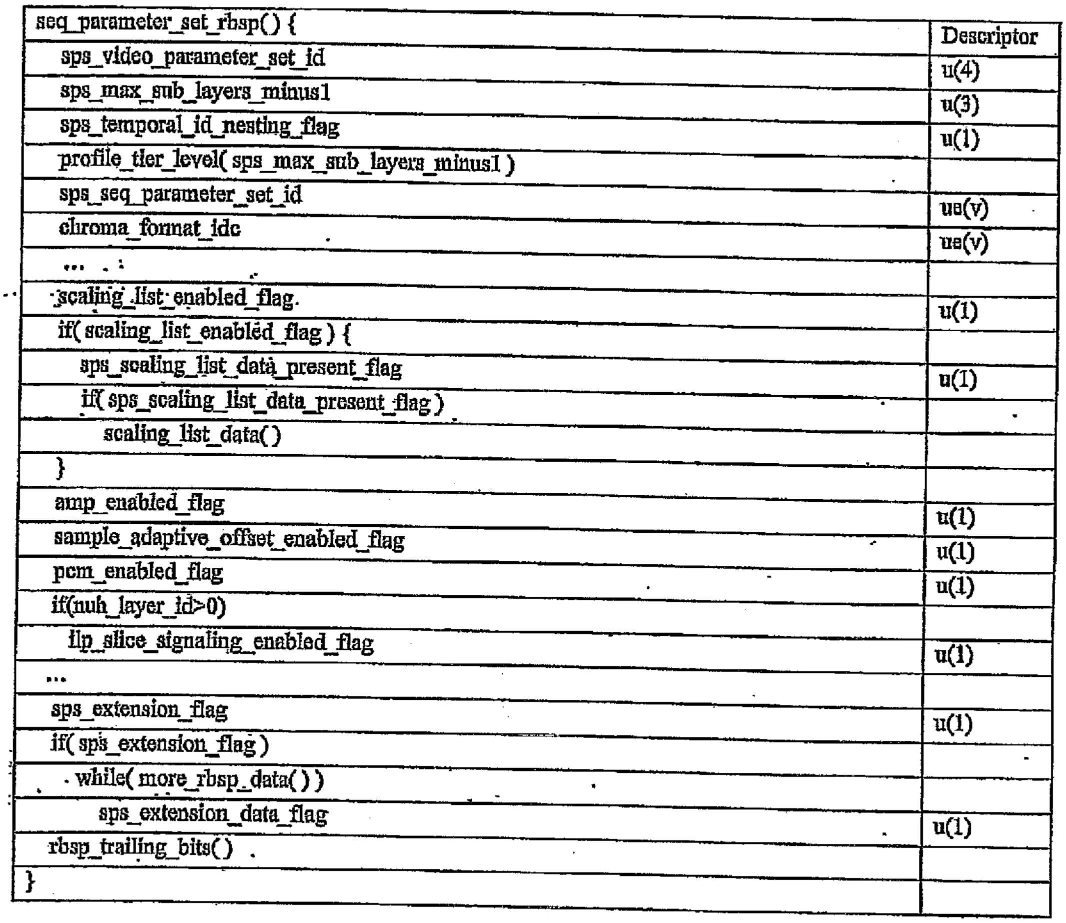
FIG.26A

[Fig. 26B]

vps_extension( ) {	Descriptor
while(!byte_aligned( ))	
vps_extension_byte_alignment_reserved_one_bit	u(1)
avc_base_layer_flag	u(1)
splitting_flag	u(1)
for(i = 0, NumScalabilityTypes = 0; i < 16; i++) {	
scalability_mask[i]	u(1)
NumScalabilityTypes += scalability_mask[i]	
}	
for(j = 0; j < (NumScalabilityTypes − splitting_flag); j++)	
dimension_id_len_minus1[j]	u(3)
vps_nuh_layer_id_present_flag	u(1)
for(i = 1; i <= vps_max_layers_minus1; i++) {	
if(vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)
if(!splitting_flag)	
for(j = 0; j < NumScalabilityTypes; j++)	
dimension_id[i][j]	u(v)
}	
for(i = 1; i <= vps_max_layers_minus1; i++)	
for(j = 0; j < i; j++)	
direct_dependency_flag[i][j]	u(1)
for(i = 0; i< vps_max_layers_minus1; i++)	
max_tid_il_ref_pics_plus1[i]	u(3)
ilp_slice_signaling_enabled_flag	u(1)
.....	

FIG.26B

[Fig. 27]



seq_parameter_set_rbsp( ) {	Descriptor
sps_video_parameter_set_id	u(4)
sps_max_sub_layers_minus1	u(3)
sps_temporal_id_nesting_flag	u(1)
profile_tier_level(sps_max_sub_layers_minus1)	
sps_seq_parameter_set_id	ue(v)
chroma_format_idc	ue(v)
...	
scaling_list_enabled_flag	u(1)
if(scaling_list_enabled_flag) {	
sps_scaling_list_data_present_flag	u(1)
if(sps_scaling_list_data_present_flag)	
scaling_list_data( )	
}	
amp_enabled_flag	u(1)
sample_adaptive_offset_enabled_flag	u(1)
pcm_enabled_flag	u(1)
if(nuh_layer_id>0)	
ilp_slice_signaling_enabled_flag	u(1)
...	
sps_extension_flag	u(1)
if(sps_extension_flag)	
while(more_rbsp_data( ))	
sps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

FIG. 27

[Fig. 28]

pic_parameter_set_rbsp() {	Descriptor
pps_pic_parameter_set_id	ue(v)
pps_seq_parameter_set_id	ue(v)
dependent_slice_segments_enabled_flag	u(1)
output_flag_present_flag	u(1)
num_extra_slice_header_bits	u(3)
sign_data_hiding_enabled_flag	u(1)
cabac_init_present_flag	u(1)
num_ref_idx_l0_default_active_minus1	ue(v)
num_ref_idx_l1_default_active_minus1	ue(v)
init_qp_minus26	se(v)
pps_loop_filter_across_slices_enabled_flag	u(1)
....	
if(nuh_layer_id>0)	
ilp_slice_signaling_enabled_flag	u(1)
....	
pps_extension_flag	u(1)
if(pps_extension_flag)	
while(more_rbsp_data())	
pps_extension_data_flag	u(1)
rbsp_trailing_bits()	
}	

FIG. 28

[Fig. 29]

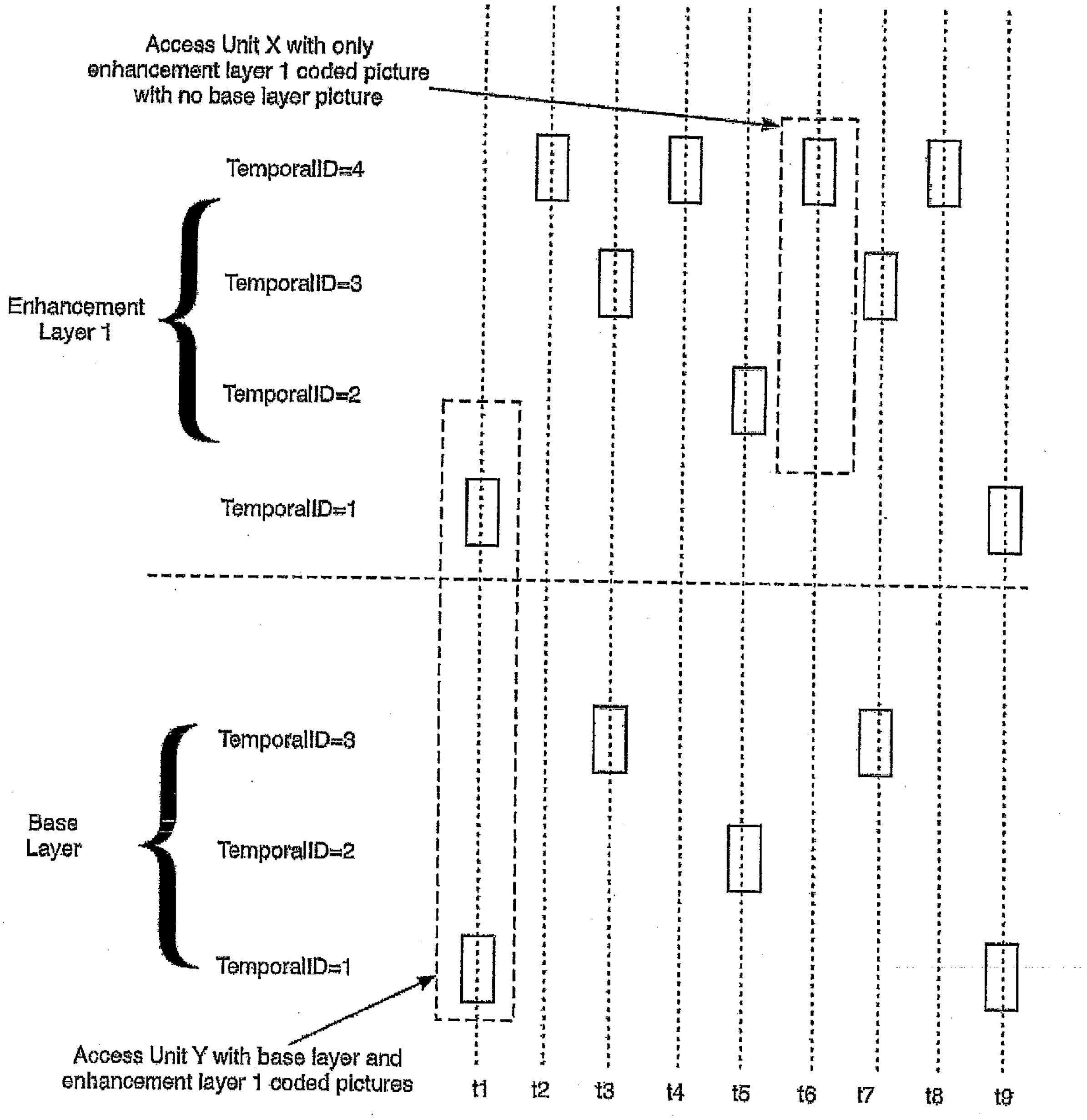

Access Unit X with only enhancement layer 1 coded picture with no base layer picture
TemporalID=4
TemporalID=3
Enhancement Layer 1
TemporalID=2
TemporalID=1
TemporalID=3
Base Layer
TemporalID=2
TemporalID=1
Access Unit Y with base layer and enhancement layer 1 coded pictures
t1
t2
t3
t4
t5
t6
t7
t8
t9

FIG. 29

[Fig. 30A]

slice_segment_header( ) {	Descriptor
first_slice_segment_in_pic_flag	u(1)
if(nal_unit_type >= BLA_W_LP && nal_unit_type <= RSV_IRAP_VCL23)	
no_output_of_prior_pics_flag	u(1)
slice_pic_parameter_set_id	ue(v)
if(!first_slice_segment_in_pic_flag) {	
if(dependent_slice_segments_enabled_flag)	
dependent_slice_segment_flag	u(1)
slice_segment_address	u(v)
}	
if(!dependent_slice_segment_flag) {	
i = 0	
if(num_extra_slice_header_bits > i) {	
i++	
poc_reset_flag	u(1)
}	
if(num_extra_slice_header_bits > i) {	
i++	
discardable_flag	u(1)
}	
for(~~i=1~~; i < num_extra_slice_header_bits; i++)	
slice_reserved_flag[i]	u(1)
slice_type	ue(v)
if(output_flag_present_flag)	
pic_output_flag	u(1)
if(separate_colour_plane_flag = = 1)	
colour_plane_id	u(2)

if(nuh_layer_id > 0 ||
(nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP)) {	

slice_pic_order_cnt_lsb	u(v)
if(nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP) {	
short_term_ref_pic_set_sps_flag	u(1)
if(!short_term_ref_pic_set_sps_flag)	
short_term_ref_pic_set(num_short_term_ref_pic_sets)	
else if(num_short_term_ref_pic_sets > 1)	
short_term_ref_pic_set_idx	u(v)

FIG. 30A

[Fig. 30B]

if(long_term_ref_pics_present_flag) {	
if(num_long_term_ref_pics_sps > 0)	
num_long_term_sps	ue(v)
num_long_term_pics	ue(v)
for(i = 0; i < num_long_term_sps + num_long_term_pics; i++) {	
if(i < num_long_term_sps) {	
if(num_long_term_ref_pics_sps > 1)	
lt_idx_sps[i]	u(v)
} else {	
poc_lsb_lt[i]	u(v)
used_by_curr_pic_lt_flag[i]	u(1)
}	
delta_poc_msb_present_flag[i]	u(1)
if(delta_poc_msb_present_flag[i])	
delta_poc_msb_cycle_lt[i]	ue(v)
}	
}	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
}	

if(nuh_layer_id > 0 && all_ref_layers_active_flag &&
        NumDirectRefLayers[nuh_layer_id] > 0) {	

inter_layer_pred_enabled_flag	u(1)
if(inter_layer_pred_enabled_flag && NumDirectRefLayers[nuh_layer_id] > 1) {	
if(!max_one_active_ref_layer_flag)	
num_inter_layer_ref_pics_minus1	u(v)
if(NumActiveRefLayerPics != NumDirectRefLayers[nuh_layer_id])	
for(i = 0; i < NumActiveRefLayerPics; i++)	
inter_layer_pred_layer_idc[i]	u(v)
}	
}	
if(sample_adaptive_offset_enabled_flag) {	
slice_sao_luma_flag	u(1)
slice_sao_chroma_flag	u(1)
}	

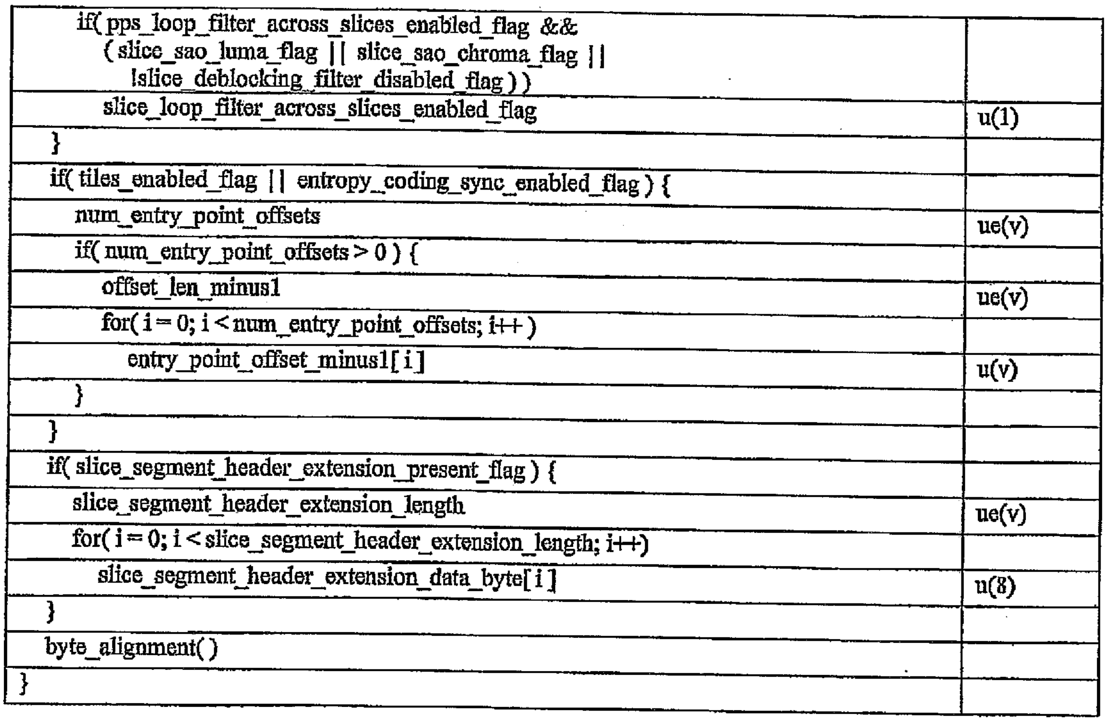
FIG. 30B

[Fig. 30C]

if(slice_type == P || slice_type == B) {	
num_ref_idx_active_override_flag	u(1)
if(num_ref_idx_active_override_flag) {	
num_ref_idx_l0_active_minus1	ue(v)
if(slice_type == B)	
num_ref_idx_l1_active_minus1	ue(v)
}	
if(lists_modification_present_flag && NumPicTotalCurr > 1)	
ref_pic_lists_modification( )	
if(slice_type == B)	
mvd_l1_zero_flag	u(1)
if(cabac_init_present_flag)	
cabac_init_flag	u(1)
if(slice_temporal_mvp_enabled_flag) {	
if(slice_type == B)	
collocated_from_l0_flag	u(1)
if((collocated_from_l0_flag && num_ref_idx_l0_active_minus1 > 0) || (!collocated_from_l0_flag && num_ref_idx_l1_active_minus1 > 0))	
collocated_ref_idx	ue(v)
}	
if((weighted_pred_flag && slice_type == P) || (weighted_bipred_flag && slice_type == B))	
pred_weight_table( )	
five_minus_max_num_merge_cand	ue(v)
}	
slice_qp_delta	se(v)
if(pps_slice_chroma_qp_offsets_present_flag) {	
slice_cb_qp_offset	se(v)
slice_cr_qp_offset	se(v)
}	
if(deblocking_filter_override_enabled_flag)	
deblocking_filter_override_flag	u(1)
if(deblocking_filter_override_flag) {	
slice_deblocking_filter_disabled_flag	u(1)
if(!slice_deblocking_filter_disabled_flag) {	
slice_beta_offset_div2	se(v)
slice_tc_offset_div2	se(v)
}	
}	

FIG. 30C

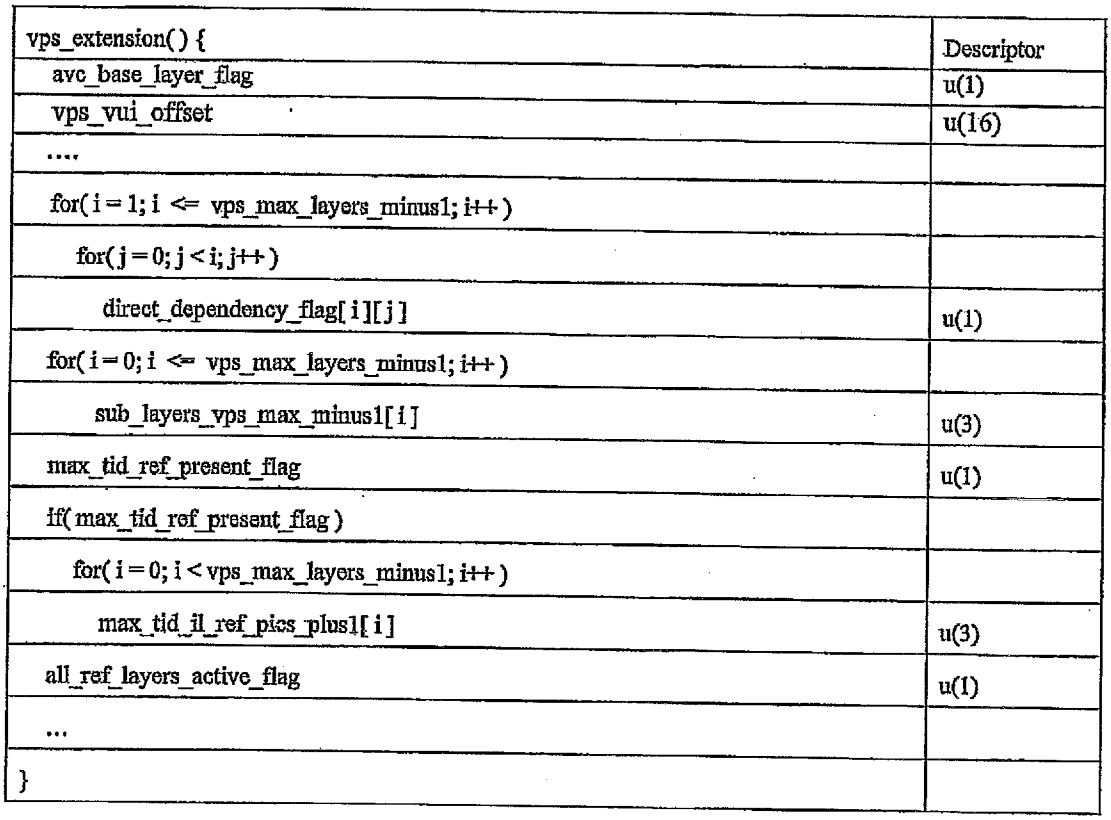
[Fig. 30D]



if(pps_loop_filter_across_slices_enabled_flag && (slice_sao_luma_flag || slice_sao_chroma_flag || !slice_deblocking_filter_disabled_flag))	
slice_loop_filter_across_slices_enabled_flag	u(1)
}	
if(tiles_enabled_flag || entropy_coding_sync_enabled_flag) {	
num_entry_point_offsets	ue(v)
if(num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset_minus1[i]	u(v)
}	
}	
if(slice_segment_header_extension_present_flag) {	
slice_segment_header_extension_length	ue(v)
for(i = 0; i < slice_segment_header_extension_length; i++)	
slice_segment_header_extension_data_byte[i]	u(8)
}	
byte_alignment( )	
}	

FIG. 30D

[Fig. 31]



vps_extension( ) {	Descriptor
avc_base_layer_flag	u(1)
vps_vui_offset	u(16)
....	
for(i = 1; i <= vps_max_layers_minus1; i++)	
for(j = 0; j < i; j++)	
direct_dependency_flag[i][j]	u(1)
for(i = 0; i <= vps_max_layers_minus1; i++)	
sub_layers_vps_max_minus1[i]	u(3)
max_tid_ref_present_flag	u(1)
if(max_tid_ref_present_flag)	
for(i = 0; i < vps_max_layers_minus1; i++)	
max_tid_il_ref_pics_plus1[i]	u(3)
all_ref_layers_active_flag	u(1)
...	
}	

FIG. 31

[Fig. 32]

...	
direct_dependency_flag[i][j]	u(1)
for(i = 1; i <= vps_max_layers_minus1; i++)	
sub_layers_vps_max_minus1[i]	u(3)
max_tid_ref_present_flag	u(1)
...	

FIG. 32

[Fig. 33]

vps_extension( ) {	Descriptor
avc_base_layer_flag	u(1)
vps_vui_offset	u(16)
....	
for(i = 1; i <= vps_max_layers_minus1; i++)	
for(j = 0; j < i; j++)	
direct_dependency_flag[i][j]	u(1)
sub_layers_vps_max_minus1_present_flag	u(1)
if(sub_layers_vps_max_minus1_present_flag)	
for(i = 0; i <= vps_max_layers_minus1; i++)	
sub_layers_vps_max_minus1[i]	u(3)
max_tid_ref_present_flag	u(1)
if(max_tid_ref_present_flag)	
for(i = 0; i < vps_max_layers_minus1; i++)	
max_tid_il_ref_pics_plus1[i]	u(3)
...	
}	

FIG. 33

[Fig. 34]

...	
direct_dependency_flag[i][j]	u(1)
sub_layers_vps_max_minus1_present_flag	u(1)
if(sub_layers_vps_max_minus1_present_flag)	
for(i = 1; i <= vps_max_layers_minus1; i++)	
sub_layers_vps_max_minus1[i]	u(3)
max_tid_ref_present_flag	u(1)
...	

FIG. 34

[Fig. 35]

vps_extension( ) {	Descriptor
avc_base_layer_flag	u(1)
vps_vui_offset	u(16)
....	
for(i = 1; i <= vps_max_layers_minus1; i++)	
for(j = 0; j < i; j++)	
direct_dependency_flag[i][j]	u(1)
for(i = 0; i <= vps_max_layers_minus1; i++) {	
sub_layers_vps_max_minus1_predict_flag[i]	u(1)
if(!sub_layers_vps_max_minus1_predict_flag[i])	
sub_layers_vps_max_minus1[i]	u(3)
}	
max_tid_ref_present_flag	u(1)
if(max_tid_ref_present_flag)	
for(i = 0; i < vps_max_layers_minus1; i++)	
max_tid_il_ref_pics_plus1[i]	u(3)
all_ref_layers_active_flag	u(1)
...	
}	

FIG. 35

[Fig. 36]

...	
direct_dependency_flag[i][j]	u(1)
for(i = 1; i <= vps_max_layers_minus1; i++) {	
sub_layers_vps_max_minus1_predict_flag[i]	u(1)
if(!sub_layers_vps_max_minus1_predict_flag[i])	
sub_layers_vps_max_minus1[i]	u(3)
}	
max_tid_ref_present_flag	u(1)
...	

FIG. 36

[Fig. 37]

slice_segment_header( ) {	Descriptor
first_slice_segment_in_pic_flag	u(1)
....	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
}	

if(nuh_layer_id > 0 && all_ref_layers_active_flag &&
NumDirectRefLayers[nuh_layer_id] > 0) {	

inter_layer_pred_enabled_flag	u(1)
if(inter_layer_pred_enabled_flag && NumDirectRefLayers[nuh_layer_id] > 1) {	
if(!max_one_active_ref_layer_flag)	
num_inter_layer_ref_pics_minus1	u(v)
if(NumActiveRefLayerPics != NumDirectRefLayers[nuh_layer_id])	
for(i = 0; i < NumActiveRefLayerPics; i++)	
inter_layer_pred_layer_idc[i]	u(v)
}	
}	
if(nuh_layer_id > 0) {	
for(i = 0; i < NumActiveRefLayerPics; i++)	
layer_present_in_au_flag[i]	u(1)
}	
if(sample_adaptive_offset_enabled_flag) {	
slice_sao_luma_flag	u(1)
slice_sao_chroma_flag	u(1)
}	
...	
}	

FIG. 37

[Fig. 38]

slice_segment_header( ) {	Descriptor
first_slice_segment_in_pic_flag	u(1)
....	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
}	
if(nuh_layer_id > 0 && all_ref_layers_active_flag && NumDirectRefLayers[nuh_layer_id] > 0) {	
inter_layer_pred_enabled_flag	u(1)
if(inter_layer_pred_enabled_flag && NumDirectRefLayers[nuh_layer_id] > 1) {	
if(!max_one_active_ref_layer_flag)	
num_inter_layer_ref_pics_minus1	u(v)
if(NumActiveRefLayerPics != NumDirectRefLayers[nuh_layer_id])	
for(i = 0; i < NumActiveRefLayerPics; i++)	
inter_layer_pred_layer_idc[i]	u(v)
}	
}	
if(nuh_layer_id > 0) {	
for(i = 0; i < NumDirectRefLayers[nuh_layer_id]; i++)	
layer_present_in_au_flag[i]	u(1)
}	
if(sample_adaptive_offset_enabled_flag) {	
slice_sao_luma_flag	u(1)
slice_sao_chroma_flag	u(1)
}	
...	
}	

FIG. 38

[Fig. 39]

slice_segment_header( ) {	Descriptor
first_slice_segment_in_pic_flag	u(1)
....	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
}	
if(nuh_layer_id > 0 && all_ref_layers_active_flag && NumDirectRefLayers[nuh_layer_id] > 0) {	
inter_layer_pred_enabled_flag	u(1)
if(inter_layer_pred_enabled_flag && NumDirectRefLayers[nuh_layer_id] > 1) {	
if(!max_one_active_ref_layer_flag)	
num_inter_layer_ref_pics_minus1	u(v)
if(NumActiveRefLayerPics != NumDirectRefLayers[nuh_layer_id])	
for(i = 0; i < NumActiveRefLayerPics; i++)	
inter_layer_pred_layer_idc[i]	u(v)
}	
}	
if(nuh_layer_id > 0) {	
for(i = 0; i < LayerIdxInVps[nuh_layer_id]; i++)	
layer_present_in_au_flag[i]	u(1)
}	
if(sample_adaptive_offset_enabled_flag) {	
slice_sao_luma_flag	u(1)
slice_sao_chroma_flag	u(1)
}	
...	
}	

FIG. 39

[Fig. 40]

if(nuh_layer_id > 0 && (all_ref_layers_active_flag || max_one_active_ref_layer_flag || (NumDirectRefLayers[nuh_layer_id] == 1)) {	
for(i = 0; i < NumActiveRefLayerPics; i++)	
layer_present_in_au_flag[i]	u(1)
}	

FIG. 40

[Fig. 41]

if(nuh_layer_id > 0 && (all_ref_layers_active_flag || max_one_active_ref_layer_flag || (NumDirectRefLayers[nuh_layer_id] == 1)) {	
for(i = 0; i < NumDirectRefLayers[nuh_layer_id]; i++)	
layer_present_in_au_flag[i]	u(1)
}	

FIG. 41

[Fig. 42]

if(nuh_layer_id > 0 && (all_ref_layers_active_flag || max_one_active_ref_layer_flag || (NumDirectRefLayers[nuh_layer_id] == 1)) {	
for(i = 0; i < LayerIdxInVps[nuh_layer_id]; i++)	
layer_present_in_au_flag[i]	u(1)
}	

FIG. 42

[Fig. 43]

```
for( i = 0; i < NumActiveRefLayerPics; i++ ) {
    if(layer_present_in_au_flag[ i ]) {
    if( there is a picture picX in the DPB that is in the same access unit as the current picture and has
        nuh_layer_id equal to RefPicLayerId[ i ] ) {
        an interlayer reference picture rsPic is derived by invoking the subclause H.8.1.4 with picX and
             RefPicLayerId[ i ] given as inputs
        if( ( ViewId[ nuh_layer_id ] <= ViewId[ 0 ]  &&
                                        ViewId[ nuh_layer_id ] <= ViewId[ RefPicLayerId[ i ] ] ) ||
            ( ViewId[ nuh_layer_id ] >= ViewId[ 0 ]  &&
                                        ViewId[ nuh_layer_id ] >= ViewId[ RefPicLayerId[ i ] ] ) ) {
            RefPicSetInterLayer0[ NumActiveRefLayerPics0 ] = rsPic
            RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] is marked as "used for long-term reference"
        } else {
            RefPicSetInterLayer1[ NumActiveRefLayerPics1 ] = rsPic
            RefPicSetInterLayer1[ NumActiveRefLayerPics1++ ] is marked as "used for long-term reference"
        }
    } else
        RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] = "no reference picture"
    }
}
```

FIG. 43

[Fig. 44]

```
for( i = 0; i < NumActiveRefLayerPics; i++ ) {
if(layer_present_in_au_flag[ LayerIdxInVps[ RefPicLayerId[ i ] ] ]) {
    if( there is a picture picX in the DPB that is in the same access unit as the current picture and has
        nuh_layer_id equal to RefPicLayerId[ i ] ) {
        an interlayer reference picture rsPic is derived by invoking the subclause H.8.1.4 with picX and
             RefPicLayerId[ i ] given as inputs
        if( ( ViewId[ nuh_layer_id ] <= ViewId[ 0 ]  &&
                                        ViewId[ nuh_layer_id ] <= ViewId[ RefPicLayerId[ i ] ] ) ||
            ( ViewId[ nuh_layer_id ] >= ViewId[ 0 ]  &&
                                        ViewId[ nuh_layer_id ] >= ViewId[ RefPicLayerId[ i ] ] ) ) {
            RefPicSetInterLayer0[ NumActiveRefLayerPics0 ] = rsPic
            RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] is marked as "used for long-term reference"
        } else {
            RefPicSetInterLayer1[ NumActiveRefLayerPics1 ] = rsPic
            RefPicSetInterLayer1[ NumActiveRefLayerPics1++ ] is marked as "used for long-term reference"
        }
    } else
        RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] = "no reference picture"
    }
}
```

FIG. 44

[Fig. 45]

```
for( i = 0; i < NumActiveRefLayerPics; i++ ) {
if(layer_present_in_au_flag[ LayerIdxInVps[ RefLayerId[ nuh_layer_id ][ inter_layer_pred_layer_idc[ i ] ] ] ]) {
          if( there is a picture picX in the DPB that is in the same access unit as the current picture and has
                    nuh_layer_id equal to RefPicLayerId[ i ] ) {
                    an interlayer reference picture rsPic is derived by invoking the subclause H.8.1.4 with picX and
                               RefPicLayerId[ i ] given as inputs
                    if( ( ViewId[ nuh_layer_id ] <= ViewId[ 0 ]  &&
                                                                      ViewId[ nuh_layer_id ] <=
ViewId[ RefPicLayerId[ i ] ] ) ||
                              ( ViewId[ nuh_layer_id ] >= ViewId[ 0 ]  &&
                                                                      ViewId[ nuh_layer_id ] >=
ViewId[ RefPicLayerId[ i ] ] ) ) {
                              RefPicSetInterLayer0[ NumActiveRefLayerPics0 ] = rsPic
                              RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] is marked as "used for long-term
reference"
                    } else {
                              RefPicSetInterLayer1[ NumActiveRefLayerPics1 ] = rsPic
                              RefPicSetInterLayer1[ NumActiveRefLayerPics1++ ] is marked as "used for long-term
reference"
                    }
          } else
                    RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] = "no reference picture"
          }
}
```

FIG. 45

[Fig. 46]

```
for( i = 0; i < NumActiveRefLayerPics; i++ ) {
if(layer_present_in_au_flag[ LayerIdxInVps[ RefLayerId[ nuh_layer_id ] [ i ] ] ]) {
   if( there is a picture picX in the DPB that is in the same access unit as the current picture and has
      nuh_layer_id equal to RefPicLayerId[ i ] ) {
      an interlayer reference picture rsPic is derived by invoking the subclause H.8.1.4 with picX and
         RefPicLayerId[ i ] given as inputs
      if( ( ViewId[ nuh_layer_id ] <= ViewId[ 0 ]  &&
                                ViewId[ nuh_layer_id ] <= ViewId[ RefPicLayerId[ i ] ] ) ||
         ( ViewId[ nuh_layer_id ] >= ViewId[ 0 ]  &&
                                ViewId[ nuh_layer_id ] >= ViewId[ RefPicLayerId[ i ] ] ) ) {
         RefPicSetInterLayer0[ NumActiveRefLayerPics0 ] = rsPic
         RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] is marked as "used for long-term reference"
      } else {
         RefPicSetInterLayer1[ NumActiveRefLayerPics1 ] = rsPic
         RefPicSetInterLayer1[ NumActiveRefLayerPics1++ ] is marked as "used for long-term reference"
      }
   } else
      RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] = "no reference picture"
   }
}
```

FIG. 46

[Fig. 47]

slice_segment_header() {	Descriptor
first_slice_segment_in_pic_flag	u(1)
....	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
}	
if(nuh_layer_id > 0 && NumDirectRefLayers[nuh_layer_id] > 0) {	
inter_layer_pred_enabled_flag	u(1)
if(inter_layer_pred_enabled_flag) {	
num_inter_layer_ref_pics_minus1	u(v)
for(i = 0; i < num_inter_layer_ref_pics_minus1; i++)	
inter_layer_pred_layer_idc[i]	u(v)
}	
}	
if(sample_adaptive_offset_enabled_flag) {	
slice_sao_luma_flag	u(1)
slice_sao_chroma_flag	u(1)
}	
...	
}	

FIG. 47

[Fig. 48A]

vps_extension( ) {	Descriptor
avc_base_layer_flag	u(1)
vps_vui_offset	u(16)
splitting_flag	u(1)
for(i = 0, NumScalabilityTypes = 0; i < 16; i++) {	
scalability_mask_flag[i]	u(1)
NumScalabilityTypes += scalability_mask_flag[i]	
}	
for(j = 0; j < (NumScalabilityTypes − splitting_flag); j++)	
dimension_id_len_minus1[j]	u(3)
vps_nuh_layer_id_present_flag	u(1)
for(i = 1; i <= vps_max_layers_minus1; i++) {	
if(vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)
if(!splitting_flag)	
for(j = 0; j < NumScalabilityTypes; j++)	
dimension_id[i][j]	u(v)
}	
if(NumViews > 1)	
view_id_len_minus1	u(4)
for(i = 0; i < NumViews; i++)	
view_id_val[i]	u(v)
for(i = 1; i <= vps_max_layers_minus1; i++)	
for(j = 0; j < i; j++)	
direct_dependency_flag[i][j]	u(1)
max_tid_ref_present_flag	u(1)
if(max_tid_ref_present_flag)	
for(i = 0; i < vps_max_layers_minus1; i++)	
max_tid_il_ref_pics_plus1[i]	u(3)
all_ref_layers_active_flag	u(1)
vps_number_layer_sets_minus1	u(10)
vps_num_profile_tier_level_minus1	u(6)
for(i = 1; i <= vps_num_profile_tier_level_minus1; i ++) {	
vps_profile_present_flag[i]	u(1)
if(!vps_profile_present_flag[i])	
profile_ref_minus1[i]	u(6)
profile_tier_level(vps_profile_present_flag[i], vps_max_sub_layers_minus1)	
}	

FIG. 48A

[Fig. 48B]

numOutputLayerSets = vps_number_layer_sets_minus1 + 1	
more_output_layer_sets_than_default_flag	u(1)
if(more_output_layer_sets_than_default_flag) {	
num_add_output_layer_sets_minus1	u(10)
numOutputLayerSets += num_add_output_layer_sets_minus1 + 1	
}	
if(numOutputLayerSets > 1)	
default_one_target_output_layer_flag	u(1)
for(i = 1; i < numOutputLayerSets; i++) {	
if(i > vps_number_layer_sets_minus1) {	
output_layer_set_idx_minus1[i]	u(v)
lsIdx = output_layer_set_idx_minus1[i] + 1	
for(j = 0 ; j < NumLayersInIdList[lsIdx] − 1; j++)	
output_layer_flag[i][j]	u(1)
}	
profile_level_tier_idx[i]	u(v)
}	
rep_format_idx_present_flag	u(1)
if(rep_format_idx_present_flag)	
vps_num_rep_formats_minus1	u(4)
for(i = 0; i <= vps_num_rep_formats_minus1; i++)	
rep_format( )	
if(rep_format_idx_present_flag)	
for(i = 1; i <= vps_max_layers_minus1; i++)	
if(vps_num_rep_formats_minus1 > 0)	
vps_rep_format_idx[i]	u(4)
max_one_active_ref_layer_flag	u(1)
cross_layer_irap_aligned_flag	u(1)
direct_dep_type_len_minus2	ue(v)
for(i = 1; i <= vps_max_layers_minus1; i++)	
for(j = 0; j < i; j++)	
if(direct_dependency_flag[i][j])	
direct_dependency_type[i][j]	u(v)
single_layer_for_non_irap_flag	u(1)
vps_vui_present_flag	u(1)
if(vps_vui_present_flag) {	
while(!byte_aligned( ))	
vps_vui_alignment_bit_equal_to_one	u(1)
vps_vui( )	
}	
}	

FIG. 48B

[Fig. 49]

vps_vui( ){	Descriptor
...	
bit_rate_present_vps_flag	u(1)
pic_rate_present_vps_flag	u(1)
if(bit_rate_present_vps_flag || pic_rate_present_vps_flag)	
for(i = 0; i <= vps_number_layer_sets_minus1; i++)	
for(j = 0; j <= vps_max_sub_layers_minus1; j++) {	
if(bit_rate_present_vps_flag)	
bit_rate_present_flag[i][j]	u(1)
if(pic_rate_present_vps_flag)	
pic_rate_present_flag[i][j]	u(1)
if(bit_rate_present_flag[i][j]) {	
avg_bit_rate[i][j]	u(16)
max_bit_rate[i][j]	u(16)
}	
if(pic_rate_present_flag[i][j]) {	
constant_pic_rate_idc[i][j]	u(2)
avg_pic_rate[i][j]	u(16)
}	
}	
...	
}	

FIG. 49

[Fig. 50]

vps_vui( ){	Descriptor
...	
bit_rate_present_vps_flag	u(1)
pic_rate_present_vps_flag	u(1)
if(bit_rate_present_vps_flag || pic_rate_present_vps_flag)	
for(i = 0; i <= vps_number_layer_sets_minus1; i++)	
for(j = 0; j <= MaxSlLayerSetMinus1[i]; j++) {	
if(bit_rate_present_vps_flag)	
bit_rate_present_flag[i][j]	u(1)
if(pic_rate_present_vps_flag)	
pic_rate_present_flag[i][j]	u(1)
if(bit_rate_present_flag[i][j]) {	
avg_bit_rate[i][j]	u(16)
max_bit_rate[i][j]	u(16)
}	
if(pic_rate_present_flag[i][j]) {	
constant_pic_rate_idc[i][j]	u(2)
avg_pic_rate[i][j]	u(16)
}	
}	
...	
}	

FIG. 50

[Fig. 51]

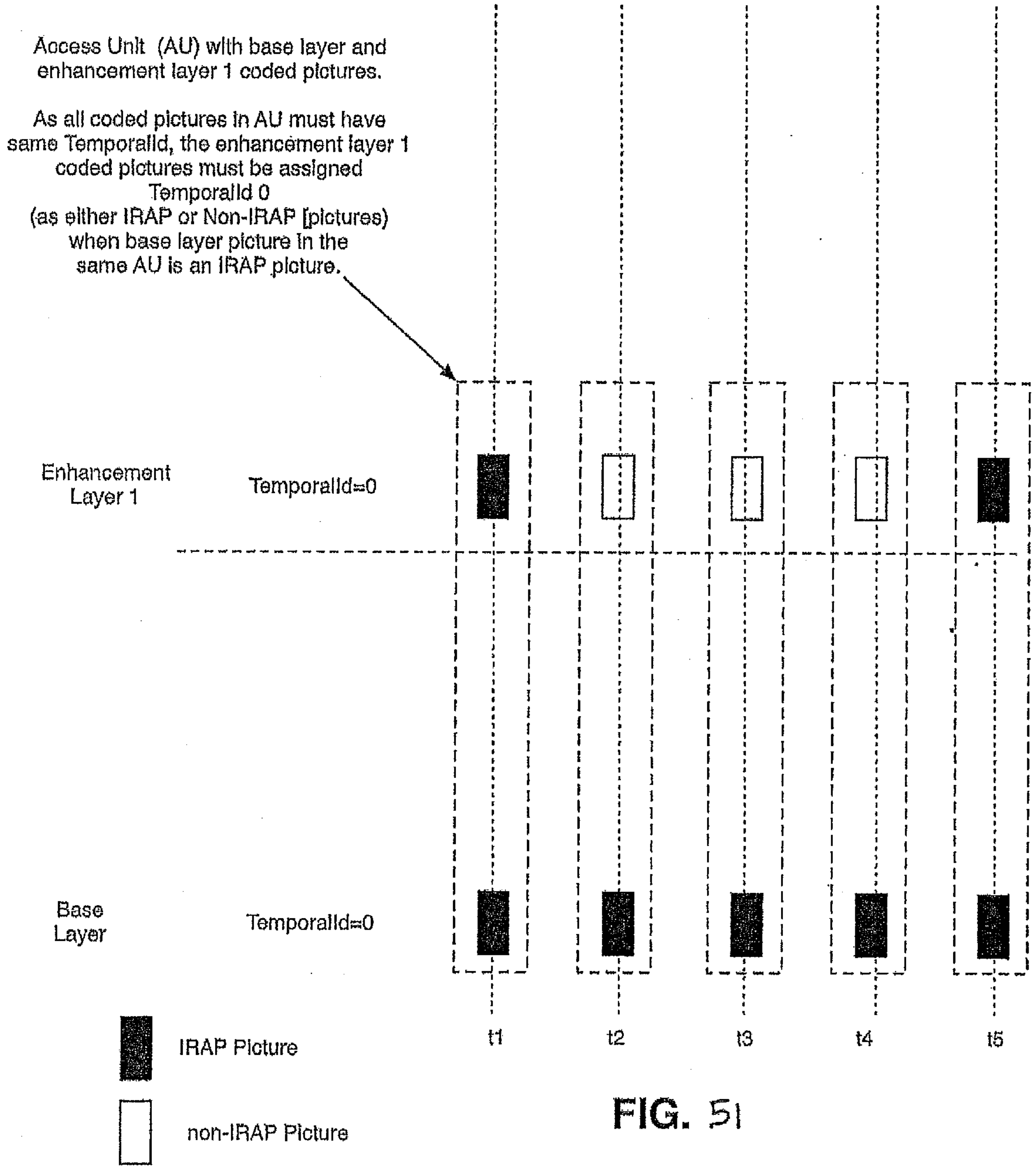

Access Unit (AU) with base layer and enhancement layer 1 coded pictures.
As all coded pictures in AU must have same TemporalId, the enhancement layer 1 coded pictures must be assigned TemporalId 0 (as either IRAP or Non-IRAP [pictures) when base layer picture in the same AU is an IRAP picture.
Enhancement Layer 1
TemporalId=0
Base Layer
TemporalId=0
t1
t2
t3
t4
t5
IRAP Picture
non-IRAP Picture

FIG. 51

[Fig. 52]

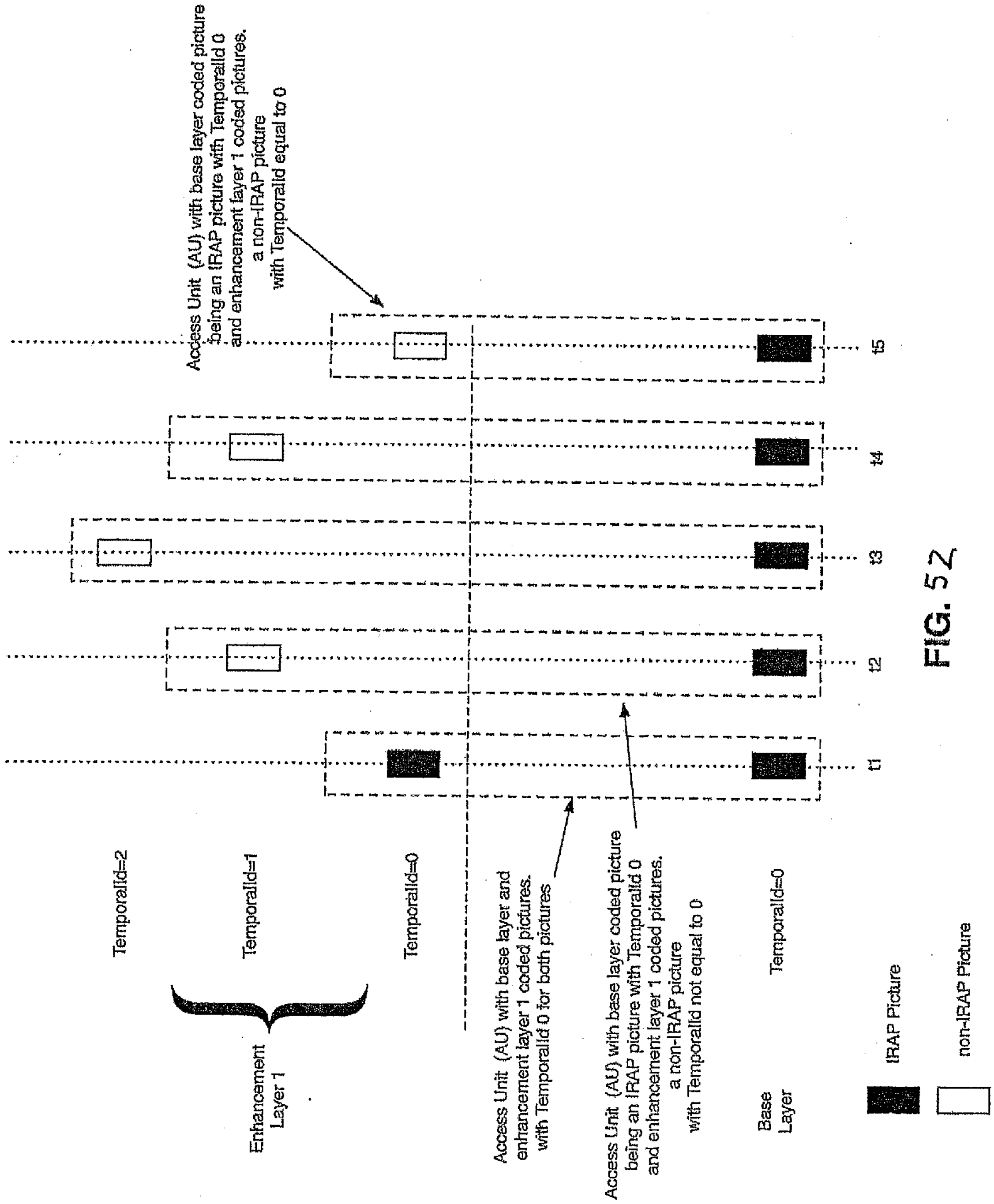

Access Unit (AU) with base layer coded picture being an IRAP picture with TemporalId 0 and enhancement layer 1 coded pictures. a non-IRAP picture with TemporalId equal to 0
TemporalId=2
TemporalId=1
TemporalId=0
Enhancement Layer 1
Access Unit (AU) with base layer and enhancement layer 1 coded pictures. with TemporalId 0 for both pictures
Access Unit (AU) with base layer coded picture being an IRAP picture with TemporalId 0 and enhancement layer 1 coded pictures. a non-IRAP picture with TemporalId not equal to 0
TemporalId=0
Base Layer
IRAP Picture
non-IRAP Picture
t1
t2
t3
t4
t5

FIG. 52

[Fig. 53]

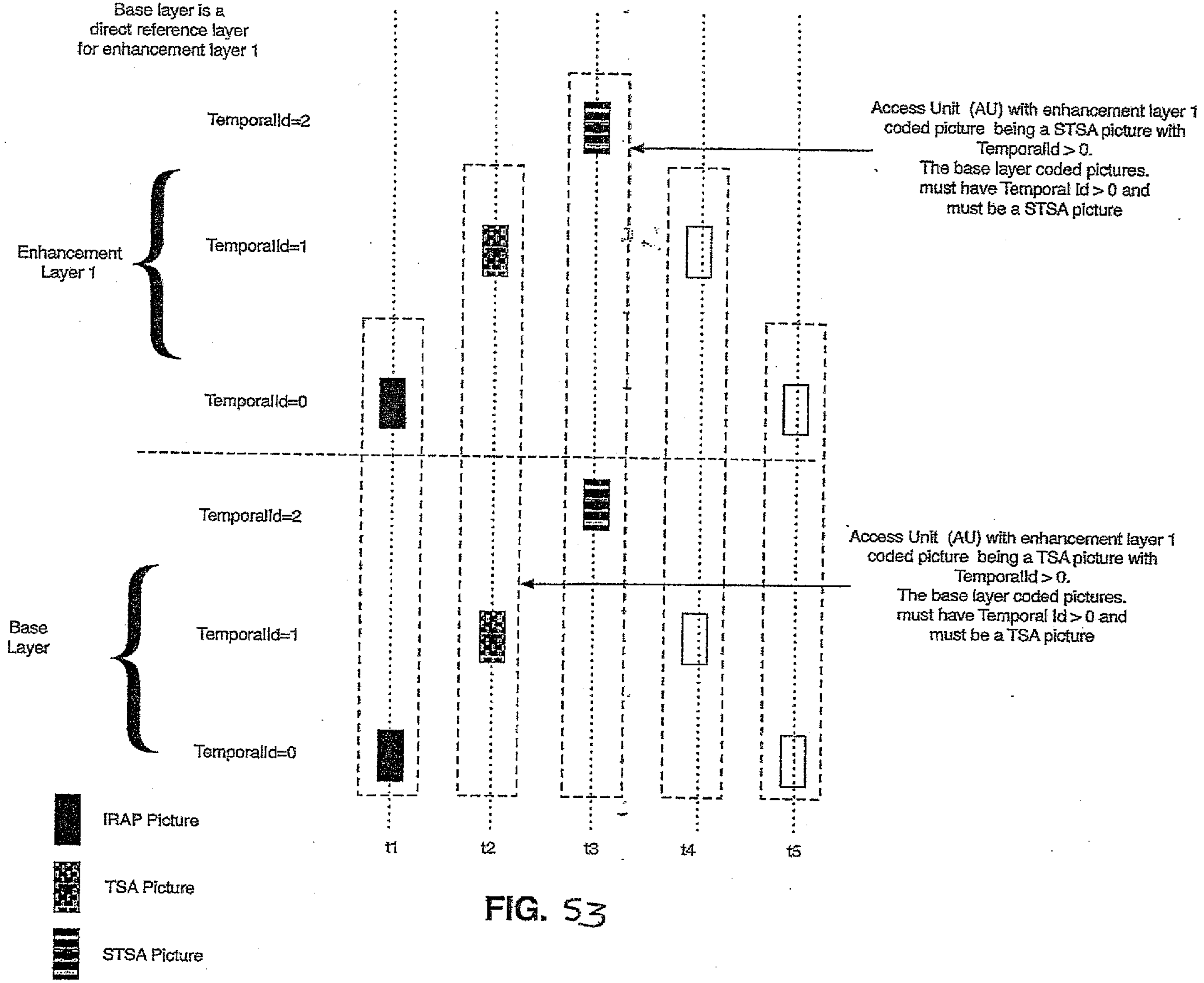

Base layer is a
direct reference layer
for enhancement layer 1
Enhancement
Layer 1
TemporalId=2
TemporalId=1
TemporalId=0
Base
Layer
TemporalId=2
TemporalId=1
TemporalId=0
IRAP Picture
TSA Picture
STSA Picture
t1
t2
t3
t4
t5
Access Unit (AU) with enhancement layer 1 coded picture being a STSA picture with TemporalId > 0.
The base layer coded pictures. must have Temporal Id > 0 and must be a STSA picture
Access Unit (AU) with enhancement layer 1 coded picture being a TSA picture with TemporalId > 0.
The base layer coded pictures. must have Temporal Id > 0 and must be a TSA picture

FIG. 53

[Fig. 54]

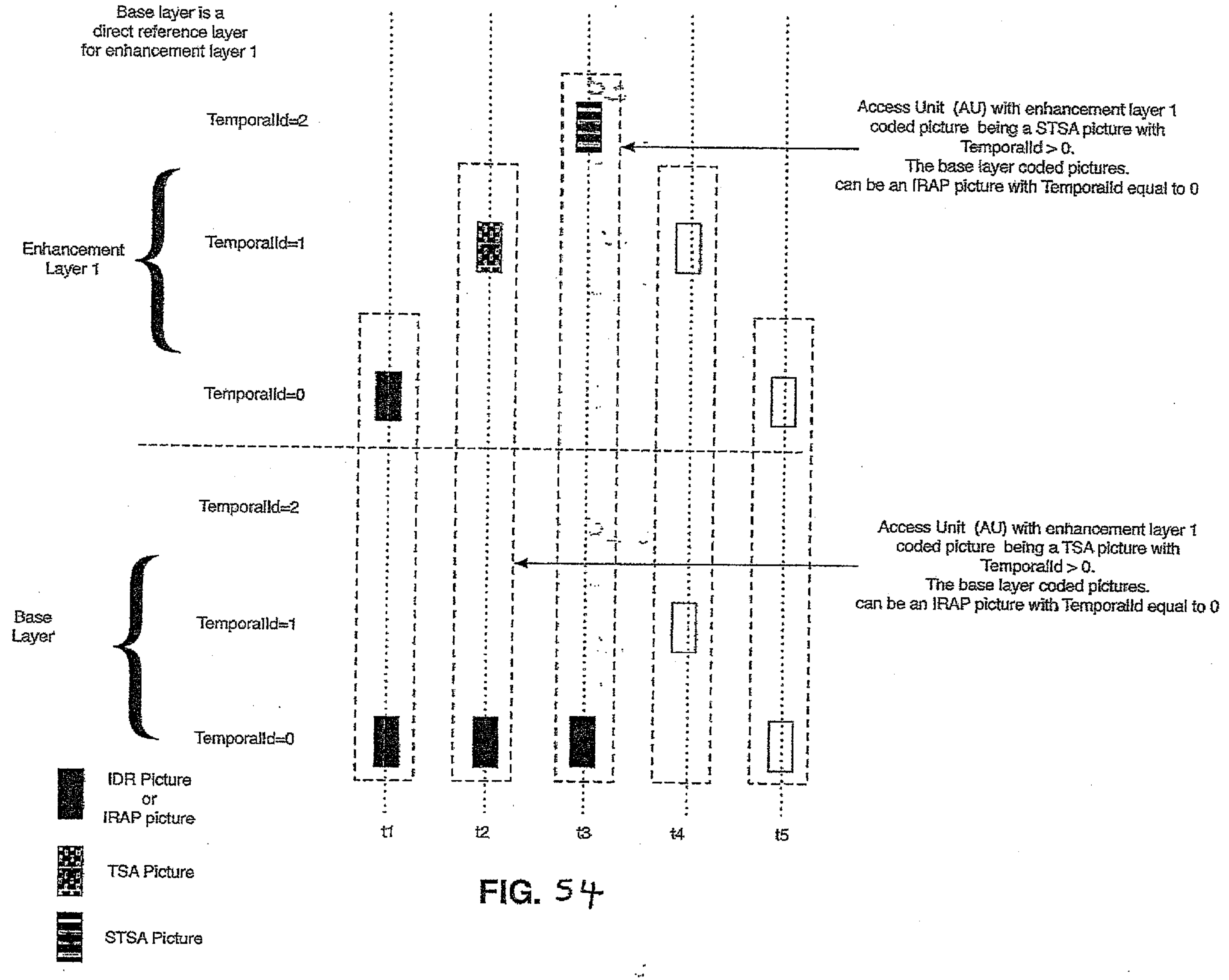

Base layer is a
direct reference layer
for enhancement layer 1
Enhancement
Layer 1
TemporalId=2
TemporalId=1
TemporalId=0
Base
Layer
TemporalId=2
TemporalId=1
TemporalId=0
Access Unit (AU) with enhancement layer 1
coded picture being a STSA picture with
TemporalId > 0.
The base layer coded pictures.
can be an IRAP picture with TemporalId equal to 0
Access Unit (AU) with enhancement layer 1
coded picture being a TSA picture with
TemporalId > 0.
The base layer coded pictures.
can be an IRAP picture with TemporalId equal to 0
t1
t2
t3
t4
t5
IDR Picture
or
IRAP picture
TSA Picture
STSA Picture

FIG. 54

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2014/005206

A. CLASSIFICATION OF SUBJECT MATTER

Int.Cl. H04N19/70(2014.01)i, H04N19/31(2014.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Int.Cl. H04N19/00-19/98

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Published examined utility model applications of Japan 1922-1996
Published unexamined utility model applications of Japan 1971-2014
Registered utility model specifications of Japan 1996-2014
Published registered utility model applications of Japan 1994-2014

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X A	Sachin Deshpande, "On Signaling DPB Parameters in VPS", Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 JCTVC-N0197, ITU-T, 2013.08.02, p.1-6	1-7 8-23
X	Jianle Chen et al., "High efficiency video coding (HEVC) scalable extension draft 3", Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 JCTVC-N1008_v3, ITU-T, 2013.09.16, p.8-10	8-15

☑ Further documents are listed in the continuation of Box C. □ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search 19.12.2014	Date of mailing of the international search report 13.01.2015
Name and mailing address of the ISA/JP **Japan Patent Office** 3-4-3, Kasumigaseki, Chiyoda-ku, Tokyo 100-8915, Japan	Authorized officer 5C 2948 Sunao HASEGAWA Telephone No. +81-3-3581-1101 Ext. 3541

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	Sachin Deshpande, "Comments On SHVC and MV-HEVC", Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 JCTVC-N0195, ITU-T, 2013.08.02, p.1-9	16-22
X	David Flynn et al., "High Efficiency Video Coding (HEVC) Range Extensions text specification: Draft 4", Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 JCTVC-N1005_v1, ITU-T, 2013.08.08, p.55-59	23

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2014/005206

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. □ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. □ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. □ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

See extra sheet.

1. □ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2. ☑ As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of additional fees.
3. □ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. □ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

□ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.

□ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.

□ No protest accompanied the payment of additional search fees.

Claims 1-7, claims 8-15, claims 16-22 and claim 23 have a common technical feature that the decoding method comprises the step of receiving a video bitstream that includes a plurality of different layers, where at least one of said plurality of different layers includes a plurality of temporal sub-layers.

However, this technical feature is not a "special technical feature"(STF) because it is not new and inventive and so makes no contribution over the prior art such as SHVC.

And there is no other technical relationship among those inventions involving one or more of the same or corresponding technical features that make a contribution over the prior art.

Therefore, these groups of inventions are not so linked as to form a single general inventive concept.

In all, there are 4 inventions listed below claimed in this application.

(Invention 1) Claims 1-7

A decoding method comprising the step of receiving a video parameter set temporal sub layers information present flag in said video parameter set extension indicating whether said information about plurality of temporal sub-layers are present.

(Invention 2) Claims 8-15

A decoding method comprising the step of receiving for 0 to a maximum number of temporal sub-layers for a particular layer set (1) a bit rate present flag; (2) a picture rate present flag; (3) bit rate information (4) picture rate information.

(Invention 3) Claims 16-22

A decoding method comprising the step of determining whether to include the second slice as an active reference layer picture for the first slice that may be used for inter layer prediction for the first slice.

(Invention 4) Claim 23

A decoding method comprising the step of receiving a temporal identifier and nal unit type with the first slice segment header.