SCALABLE NESTING SEI MESSAGES FOR SPECIFIED LAYERS

TECHNICAL FIELD

[0001] The present disclosure is generally related to video coding, and is specifically related to scalable nesting supplemental enhancement information (SEI) messages used to support encoding layers in multi-layer bitstreams.

BACKGROUND

[0002] The amount of video data needed to depict even a relatively short video can be substantial, which may result in difficulties when the data is to be streamed or otherwise communicated across a communications network with limited bandwidth capacity. Thus, video data is generally compressed before being communicated across modern day telecommunications networks. The size of a video could also be an issue when the video is stored on a storage device because memory resources may be limited. Video compression devices often use software and/or hardware at the source to code the video data prior to transmission or storage, thereby decreasing the quantity of data needed to represent digital video images. The compressed data is then received at the destination by a video decompression device that decodes the video data. With limited network resources and ever increasing demands of higher video quality, improved compression and decompression techniques that improve compression ratio with little to no sacrifice in image quality are desirable.

SUMMARY

[0003] In an embodiment, the disclosure includes a method implemented in a decoder, the method comprising: receiving, by a receiver of the decoder, a bitstream comprising one or more layers and a scalable nesting supplemental enhancement information (SEI) message in a current SEI network abstraction layer (NAL) unit, wherein the scalable nesting SEI message contains one or more scalable-nested SEI messages and one or more scalable nesting layer identifiers (Ids) that specify layer Id values of the layers to which the scalable-nested SEI messages apply except a layer Id value of the current SEI NAL unit; decoding, by a processor of the decoder, a coded picture from the one or more layers to produce a decoded picture; and forwarding, by the processor, the decoded picture for display as part of a decoded video sequence.

[0004] Some video coding systems employ SEI messages. An SEI message contains information that is not needed by the decoding process in order to determine the values of the samples in decoded pictures. For example, the SEI messages may contain parameters used to check a bitstream for conformance with standards. In some instances, a video is coded into a bitstream using multiple layers. Scalable nesting SEI messages may be employed to contain SEI messages for multiple layers. For example, a scalable nesting SEI message contains scalable-nested SEI messages. A problem may occur when a list of layer Ids is included in the scalable nesting SEI message to indicate the correspondence between the scalable-nested SEI messages and the layers. The scalable nesting SEI message is contained in a SEI NAL unit, which can be denoted as the current SEI NAL unit for clarity of discussion. The current SEI NAL unit is associated with a layer Id. The scalable nesting SEI message is always related to the layer associated with the current SEI NAL unit. This leads to redundant signaling as the layer Id of the current SEI NAL unit is included in the current SEI NAL unit and is also included in the scalable nesting SEI message contained in the current NAL unit.

[0005] The present example includes improvements to the scalable nesting SEI message for use in conjunction with specified layers. Specifically, the layer Id may be referred to as a NAL unit header layer Id (nuh_layer_id). The nuh_layer_id of the current SEI NAL unit is excluded from the scalable nesting SEI message. This can be done by employing a scalable nesting layer Id (layer_id[i]) syntax element that specifies the nuh_layer_id for each layer to which a scalable-nested SEI message applies. The scalable nesting layer_id[i] is constrained to include only nuh_layer_ids that are greater than the nuh_layer_id of the current SEI NAL unit. Hence, the nuh_layer_id of the current SEI NAL unit is omitted from the scalable nesting SEI message and inferred from the current SEI NAL unit. This improves the scalable nesting SEI message by omitting a redundant Id. For example, a loop that encodes/decodes the scalable nesting layer_id[i] executes one less time, which decreases processor resources during encoding and/or decoding. Further, this reduces the size of the encoded bitstream for each of the scalable nesting SEI messages in the bitstream. As a result, coding efficiency is increased, which reduces processor, memory, and/or network signaling resource usage at both the encoder and the decoder.

[0006] Optionally, in any of the preceding aspects, another implementation of the aspect provides, wherein the scalable nesting layer Ids are a scalable nesting layer_id[i] syntax element that specifies a nuh_layer_id value of an i-th layer to which the scalable-nested SEI messages apply when a scalable nesting all layers flag (all_layers_flag) is equal to zero.

[0007] Optionally, in any of the preceding aspects, another implementation of the aspect provides, wherein each value of scalable nesting layer_id[i] is greater than a nuh_layer_id value of the current SEI NAL unit.

[0008] Optionally, in any of the preceding aspects, another implementation of the aspect provides, further comprising deriving, by the processor, a number of nesting layers (nestingNumLayers) variable that specifies a number of layers to which the scalable-nested SEI messages apply.

[0009] Optionally, in any of the preceding aspects, another implementation of the aspect provides, further comprising deriving, by the processor, a list of nesting layer Ids (NestingLayerId[i]) that specifies a list of nuh_layer_id values of the layers to which the scalable-nested SEI messages apply for i in a range of zero to nestingNumLayers minus one, inclusive.

[0010] Optionally, in any of the preceding aspects, another implementation of the aspect provides, wherein the nestingNumLayers and the NestingLayerId[i] are derived as follows:

```
if( scalable nesting all_layers_flag ) {
        nestingNumLayers =
vps_max_layers_minus1 + 1 − GeneralLayerIdx[ nuh_layer_id ]
        for( i = 0; i < nestingNumLayers; i ++)
            NestingLayerId[ i ] = vps_layer_id[ GeneralLayerIdx[ nuh_layer_id ] + i ]
} else {
        nestingNumLayers = scalable nesting num_layers_minus1 + 1
        for( i = 0; i < nestingNumLayers; i ++)
            NestingLayerId[ i ] = ( i  = = 0 ) ? nuh_layer_id : scalable nesting layer_id[ i ]
}
```

where scalable nesting all_layers_flag specifies whether the scalable-nested SEI messages apply to all layers, vps_max_layers_minus1 plus one specifies a number of layers specified by a video parameter set (VPS), GeneralLayerIdx is a variable specifying a layer index of a layer with nuh_layer_id equal to vps_layer_id[i], vps_layer_id[i] specifies a nuh_layer_id value of an i-th layer, and scalable nesting num_layers_minus1 plus one specifies a number of layers to which the scalable-nested SEI messages apply.

[0011] Optionally, in any of the preceding aspects, another implementation of the aspect provides, wherein scalable nesting num_layers_minus1 is in a range of zero to vps_max_layers_minus1− GeneralLayerIdx[nuh_layer_id], inclusive, where nuh_layer_id is the nuh_layer_id of the current SEI NAL unit.

[0012] Optionally, in any of the preceding aspects, another implementation of the aspect provides, wherein the coded picture is decoded based on the scalable nested SEI messages.

[0013] Optionally, in any of the preceding aspects, another implementation of the aspect provides, wherein the scalable nesting SEI message includes at least one scalable-nested SEI message that applies to a layer of the current SEI NAL unit.

[0014] In an embodiment, the disclosure includes a method implemented in an encoder, the method comprising: encoding, by a processor of the encoder, a bitstream comprising one or more layers; encoding into the bitstream, by the processor, a scalable nesting SEI message in a current SEI NAL unit, wherein the scalable nesting SEI message contains one or more scalable-nested SEI messages and one or more scalable nesting layer Ids that specify layer Id values of the layers to which the scalable-nested SEI messages apply while applying a constraint to omit a layer Id value of the current SEI NAL unit; performing, by the processor, a set of bitstream conformance tests on the layers based on the scalable-nested SEI messages; and storing, by a memory coupled to the processor, the bitstream for communication toward a decoder.

[0015] Some video coding systems employ SEI messages. An SEI message contains information that is not needed by the decoding process in order to determine the values of the samples in decoded pictures. For example, the SEI messages may contain parameters used to check a bitstream for conformance with standards. In some instances, a video is coded into a bitstream using multiple layers. Scalable nesting SEI messages may be employed to contain SEI messages for multiple layers. For example, a scalable nesting SEI message contains scalable-nested SEI messages. A problem may occur when a list of layer Ids is included in the scalable nesting SEI message to indicate the correspondence between the scalable-nested SEI messages and the layers. The scalable nesting SEI message is contained in a SEI NAL unit, which can be denoted as the current SEI NAL unit for clarity of discussion. The current SEI NAL unit is associated with a layer Id. The scalable nesting SEI message is always related to the layer associated with the current SEI NAL unit. This leads to redundant signaling as the layer Id of the current SEI NAL unit is included in the current SEI NAL unit and is also included in the scalable nesting SEI message contained in the current NAL unit.

[0016] The present example includes improvements to the scalable nesting SEI message for use in conjunction with specified layers. Specifically, the layer Id may be referred to as a nuh_layer_id. The nuh_layer_id of the current SEI NAL unit is excluded from the scalable nesting SEI message. This can be done by employing a scalable nesting layer_id[i] syntax element that specifies the nuh_layer_id for each layer to which a scalable-nested SEI message applies. The scalable nesting layer_id[i] is constrained to include only nuh_layer_ids that are

greater than the nuh_layer_id of the current SEI NAL unit. Hence, the nuh_layer_id of the current SEI NAL unit is omitted from the scalable nesting SEI message and inferred from the current SEI NAL unit. This improves the scalable nesting SEI message by omitting a redundant Id. For example, a loop that encodes/decodes the scalable nesting layer_id[i] executes one less time, which decreases processor resources during encoding and/or decoding. Further, this reduces the size of the encoded bitstream for each of the scalable nesting SEI messages in the bitstream. As a result, coding efficiency is increased, which reduces processor, memory, and/or network signaling resource usage at both the encoder and the decoder.

[0017] Optionally, in any of the preceding aspects, another implementation of the aspect provides, wherein the scalable nesting layer Id is a scalable nesting layer_id[i] syntax element that specifies a nuh_layer_id value of an i-th layer to which the scalable-nested SEI messages apply when a scalable nesting all_layers_flag is equal to zero.

[0018] Optionally, in any of the preceding aspects, another implementation of the aspect provides, wherein each value of scalable nesting layer_id[i] is greater than a nuh_layer_id value of the current SEI NAL unit.

[0019] Optionally, in any of the preceding aspects, another implementation of the aspect provides, wherein performing the set of bitstream conformance tests further comprises derriving a number of nesting layers (nestingNumLayers) variable that specifies a number of layers to which the scalable-nested SEI messages apply.

[0020] Optionally, in any of the preceding aspects, another implementation of the aspect provides, wherein performing the set of bitstream conformance tests further comprises derriving a list of nesting layer Ids (NestingLayerId[i]) that specifies a list of nuh_layer_id values of the layers to which the scalable-nested SEI messages apply for i in a range of zero to nestingNumLayers minus one, inclusive.

[0021] Optionally, in any of the preceding aspects, another implementation of the aspect provides, wherein the nestingNumLayers and the NestingLayerId[i] are derived as follows:

```
if( scalable nesting all_layers_flag ) {
        nestingNumLayers =
vps_max_layers_minus1 + 1 − GeneralLayerIdx[ nuh_layer_id ]
        for( i = 0; i < nestingNumLayers; i ++)
            NestingLayerId[ i ] = vps_layer_id[ GeneralLayerIdx[ nuh_layer_id ] + i ]
} else {
        nestingNumLayers = scalable nesting num_layers_minus1 + 1
        for( i = 0; i < nestingNumLayers; i ++)
```

NestingLayerId[i] = (i = = 0) ? nuh_layer_id : scalable nesting layer_id[i]

}

where scalable nesting all_layers_flag specifies whether the scalable-nested SEI messages apply to all layers, vps_max_layers_minus1 plus one specifies a number of layers specified by a video parameter set (VPS), GeneralLayerIdx is a variable specifying a layer index of a layer with nuh_layer_id equal to vps_layer_id[i], vps_layer_id[i] specifies a nuh_layer_id value of an i-th layer, and scalable nesting num_layers_minus1 plus one specifies a number of layers to which the scalable-nested SEI messages apply.

[0022] Optionally, in any of the preceding aspects, another implementation of the aspect provides, wherein scalable nesting num_layers_minus1 is in a range of zero to vps_max_layers_minus1− GeneralLayerIdx[nuh_layer_id], inclusive, where nuh_layer_id is the nuh_layer_id of the current SEI NAL unit.

[0023] Optionally, in any of the preceding aspects, another implementation of the aspect provides, wherein the scalable nesting SEI message includes at least one scalable-nested SEI message that applies to a layer of the current SEI NAL unit.

[0024] In an embodiment, the disclosure includes a video coding device comprising: a processor, a receiver coupled to the processor, a memory coupled to the processor, and a transmitter coupled to the processor, wherein the processor, receiver, memory, and transmitter are configured to perform the method of any of the preceding aspects.

[0025] In an embodiment, the disclosure includes a non-transitory computer readable medium comprising a computer program product for use by a video coding device, the computer program product comprising computer executable instructions stored on the non-transitory computer readable medium such that when executed by a processor cause the video coding device to perform the method of any of the preceding aspects.

[0026] In an embodiment, the disclosure includes a decoder comprising: a receiving means for receiving a bitstream comprising one or more layers and a scalable nesting SEI message in a current SEI NAL unit, wherein the scalable nesting SEI message contains one or more scalable-nested SEI messages and one or more scalable nesting layer Ids that specify layer Id values of the layers to which the scalable-nested SEI messages apply except a layer Id value of the current SEI NAL unit; a decoding means for decoding a coded picture from the one or more layers to produce a decoded picture; and a forwarding means for forwarding the decoded picture for display as part of a decoded video sequence.

[0027] Some video coding systems employ SEI messages. An SEI message contains information that is not needed by the decoding process in order to determine the values of the

samples in decoded pictures. For example, the SEI messages may contain parameters used to check a bitstream for conformance with standards. In some instances, a video is coded into a bitstream using multiple layers. Scalable nesting SEI messages may be employed to contain SEI messages for multiple layers. For example, a scalable nesting SEI message contains scalable-nested SEI messages. A problem may occur when a list of layer Ids is included in the scalable nesting SEI message to indicate the correspondence between the scalable-nested SEI messages and the layers. The scalable nesting SEI message is contained in a SEI NAL unit, which can be denoted as the current SEI NAL unit for clarity of discussion. The current SEI NAL unit is associated with a layer Id. The scalable nesting SEI message is always related to the layer associated with the current SEI NAL unit. This leads to redundant signaling as the layer Id of the current SEI NAL unit is included in the current SEI NAL unit and is also included in the scalable nesting SEI message contained in the current NAL unit.

[0028] The present example includes improvements to the scalable nesting SEI message for use in conjunction with specified layers. Specifically, the layer Id may be referred to as a nuh_layer_id. The nuh_layer_id of the current SEI NAL unit is excluded from the scalable nesting SEI message. This can be done by employing a scalable nesting layer_id[i] syntax element that specifies the nuh_layer_id for each layer to which a scalable-nested SEI message applies. The scalable nesting layer_id[i] is constrained to include only nuh_layer_ids that are greater than the nuh_layer_id of the current SEI NAL unit. Hence, the nuh_layer_id of the current SEI NAL unit is omitted from the scalable nesting SEI message and inferred from the current SEI NAL unit. This improves the scalable nesting SEI message by omitting a redundant Id. For example, a loop that encodes/decodes the scalable nesting layer_id[i] executes one less time, which decreases processor resources during encoding and/or decoding. Further, this reduces the size of the encoded bitstream for each of the scalable nesting SEI messages in the bitstream. As a result, coding efficiency is increased, which reduces processor, memory, and/or network signaling resource usage at both the encoder and the decoder.

[0029] Optionally, in any of the preceding aspects, another implementation of the aspect provides, wherein the decoder is further configured to perform the method of any of the preceding aspects.

[0030] In an embodiment, the disclosure includes an encoder comprising: an encoding means for: encoding a bitstream comprising one or more layers; and encoding into the bitstream a scalable nesting SEI message in a current SEI NAL unit, wherein the scalable nesting SEI message contains one or more scalable-nested SEI messages and one or more scalable nesting layer Ids that specify layer Id values of the layers to which the scalable-nested SEI messages

apply while applying a constraint to omit a layer Id value of the current SEI NAL unit; a HRD means for performing a set of bitstream conformance tests on the layers based on the scalable-nested SEI messages; and a storing means for storing the bitstream for communication toward a decoder.

[0031] Some video coding systems employ SEI messages. An SEI message contains information that is not needed by the decoding process in order to determine the values of the samples in decoded pictures. For example, the SEI messages may contain parameters used to check a bitstream for conformance with standards. In some instances, a video is coded into a bitstream using multiple layers. Scalable nesting SEI messages may be employed to contain SEI messages for multiple layers. For example, a scalable nesting SEI message contains scalable-nested SEI messages. A problem may occur when a list of layer Ids is included in the scalable nesting SEI message to indicate the correspondence between the scalable-nested SEI messages and the layers. The scalable nesting SEI message is contained in a SEI NAL unit, which can be denoted as the current SEI NAL unit for clarity of discussion. The current SEI NAL unit is associated with a layer Id. The scalable nesting SEI message is always related to the layer associated with the current SEI NAL unit. This leads to redundant signaling as the layer Id of the current SEI NAL unit is included in the current SEI NAL unit and is also included in the scalable nesting SEI message contained in the current NAL unit.

[0032] The present example includes improvements to the scalable nesting SEI message for use in conjunction with specified layers. Specifically, the layer Id may be referred to as a nuh_layer_id. The nuh_layer_id of the current SEI NAL unit is excluded from the scalable nesting SEI message. This can be done by employing a scalable nesting layer_id[i] syntax element that specifies the nuh_layer_id for each layer to which a scalable-nested SEI message applies. The scalable nesting layer_id[i] is constrained to include only nuh_layer_ids that are greater than the nuh_layer_id of the current SEI NAL unit. Hence, the nuh_layer_id of the current SEI NAL unit is omitted from the scalable nesting SEI message and inferred from the current SEI NAL unit. This improves the scalable nesting SEI message by omitting a redundant Id. For example, a loop that encodes/decodes the scalable nesting layer_id[i] executes one less time, which decreases processor resources during encoding and/or decoding. Further, this reduces the size of the encoded bitstream for each of the scalable nesting SEI messages in the bitstream. As a result, coding efficiency is increased, which reduces processor, memory, and/or network signaling resource usage at both the encoder and the decoder.

[0033] Optionally, in any of the preceding aspects, another implementation of the aspect provides, wherein the encoder is further configured to perform the method of any of the preceding aspects.

[0034] For the purpose of clarity, any one of the foregoing embodiments may be combined with any one or more of the other foregoing embodiments to create a new embodiment within the scope of the present disclosure.

[0035] These and other features will be more clearly understood from the following detailed description taken in conjunction with the accompanying drawings and claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0036] For a more complete understanding of this disclosure, reference is now made to the following brief description, taken in connection with the accompanying drawings and detailed description, wherein like reference numerals represent like parts.

[0037] FIG. 1 is a flowchart of an example method of coding a video signal.

[0038] FIG. 2 is a schematic diagram of an example coding and decoding (codec) system for video coding.

[0039] FIG. 3 is a schematic diagram illustrating an example video encoder.

[0040] FIG. 4 is a schematic diagram illustrating an example video decoder.

[0041] FIG. 5 is a schematic diagram illustrating an example hypothetical reference decoder (HRD).

[0042] FIG. 6 is a schematic diagram illustrating an example multi-layer video sequence configured for inter-layer prediction.

[0043] FIG. 7 is a schematic diagram illustrating an example bitstream.

[0044] FIG. 8 is a schematic diagram of an example video coding device.

[0045] FIG. 9 is a flowchart of an example method of encoding a video sequence into a bitstream including scalable nesting SEI messages that apply to layers.

[0046] FIG. 10 is a flowchart of an example method of decoding a video sequence from a bitstream including scalable nesting SEI messages that apply to layers.

[0047] FIG. 11 is a schematic diagram of an example system for coding a video sequence using a bitstream including scalable nesting SEI messages that apply to layers.

DETAILED DESCRIPTION

[0048] It should be understood at the outset that although an illustrative implementation of one or more embodiments are provided below, the disclosed systems and/or methods may be implemented using any number of techniques, whether currently known or in existence. The disclosure should in no way be limited to the illustrative implementations, drawings, and techniques illustrated below, including the exemplary designs and implementations illustrated and described herein, but may be modified within the scope of the appended claims along with their full scope of equivalents.

[0049] The following terms are defined as follows unless used in a contrary context herein. Specifically, the following definitions are intended to provide additional clarity to the present disclosure. However, terms may be described differently in different contexts. Accordingly, the following definitions should be considered as a supplement and should not be considered to limit any other definitions of descriptions provided for such terms herein.

[0050] A bitstream may be a sequence of bits including video data that is compressed for transmission between an encoder and a decoder. An encoder may be a device that is configured to employ encoding processes to compress video data into a bitstream. A decoder may be a device that is configured to employ decoding processes to reconstruct video data from a bitstream for display. A picture may be an array of luma samples and/or an array of chroma samples that create a frame or a field thereof. A picture that is being encoded or decoded can be referred to as a current picture for clarity of discussion. A coded picture may be a coded representation of a picture comprising video coding layer (VCL) network abstraction layer (NAL) units with a particular value of NAL unit header layer identifier (nuh_layer_id) within an access unit (AU) and containing all coding tree units (CTUs) of the picture. A decoded picture may be a picture produced by applying a decoding process to a coded picture. An AU may be a set of coded pictures that are included in different layers and are associated with the same time for output from a decoded picture buffer (DPB). A NAL unit may be a syntax structure containing data in the form of a Raw Byte Sequence Payload (RBSP), an indication of the type of data, and interspersed as desired with emulation prevention bytes. A VCL NAL unit may be a NAL unit coded to contain video data, such as a coded slice of a picture. A non-VCL NAL unit may be a NAL unit that contains non-video data such as syntax and/or parameters that support decoding the video data, performance of conformance checking, or other operations. A layer may be a set of VCL NAL units that share a specified characteristic (e.g., a common resolution, frame rate, image size, etc.) as indicated by layer Id (identifier) and associated non-VCL NAL units. A NAL unit header layer identifier (nuh_layer_id) may be a syntax element

that specifies an identifier of a layer that includes a NAL unit. A video parameter set (VPS) may be a data unit that contains parameters related to an entire video. A coded video sequence may be a set of one or more coded pictures. A decoded video sequence may be a set of one or more decoded pictures.

[0051] A hypothetical reference decoder (HRD) may be a decoder model operating on an encoder that checks the variability of bitstreams produced by an encoding process to verify conformance with specified constraints. A bitstream conformance test may be a test to determine whether an encoded bitstream complies with a standard, such as Versatile Video Coding (VVC). HRD parameters are syntax elements that initialize and/or define operational conditions of an HRD. HRD parameters may be included in supplemental enhancement information (SEI) messages. A SEI message may be a syntax structure with specified semantics that conveys information that is not needed by the decoding process in order to determine the values of the samples in decoded pictures. A SEI NAL unit may be a NAL unit that contains one or more SEI messages. A specific SEI NAL unit may be referred to as a current SEI NAL unit. A scalable nesting SEI message may be a message that contains a plurality of SEI messages that correspond to one or more output layer sets (OLSs) or one or more layers. A buffering period (BP) SEI message may be a SEI message that contains HRD parameters for initializing an HRD to manage a coded picture buffer (CPB). A picture timing (PT) SEI message may be a SEI message that contains HRD parameters for managing delivery information for access units (AUs) at the CPB and/or a decoded picture buffer (DPB). A decoding unit information (DUI) SEI message may be a SEI message that contains HRD parameters for managing delivery information for DUs at the CPB and/or the DPB.

[0052] A scalable nesting SEI message may be a set of scalable-nested SEI messages. A scalable-nested SEI message may be a SEI message that is nested inside a scalable nesting SEI message. A scalable nesting layer id (layer_id[i]) may be a syntax element in a scalable nesting SEI message that specifies the nuh_layer_id value of an i-th layer to which the scalable-nested SEI messages apply. A scalable nesting all layers flag (all_layers_flag) may be a syntax element in a scalable nesting SEI message that specifies whether scalable-nested SEI messages in a scalable nesting SEI message apply to all layers. A scalable nesting number of layers minus one (num_layers_minus1) may be a syntax element in a scalable nesting SEI message that specifies the number of layers to which the scalable-nested SEI messages apply. A VPS maximum layers minus one (vps_max_layers_minus1) may be a syntax element in a VPS that specifies a number of layers specified by the VPS. A VPS layer Id (vps_layer_id[i]) may be a syntax element in a VPS that specifies a nuh_layer_id value of an i-th layer. A number of nesting

layers (nestingNumLayers) may be a variable that is set to the number of layers to which the scalable-nested SEI messages apply based on data in a bitstream. A list of nesting layer Ids (NestingLayerId[i]) may be a variable that is set to specify a list of nuh_layer_id values of the layers to which the scalable-nested SEI messages apply for each layer i based on data in a bitstream. A general layer index (GeneralLayerIdx) may be a variable set to specify a layer index of each layer with a nuh_layer_id based on data in a bitstream.

[0053] The following acronyms are used herein, Access Unit (AU), Coding Tree Block (CTB), Coding Tree Unit (CTU), Coding Unit (CU), Coded Layer Video Sequence (CLVS), Coded Layer Video Sequence Start (CLVSS), Coded Video Sequence (CVS), Coded Video Sequence Start (CVSS), Joint Video Experts Team (JVET), Motion Constrained Tile Set (MCTS), Maximum Transfer Unit (MTU), Network Abstraction Layer (NAL), Output Layer Set (OLS), Picture Order Count (POC), Random Access Point (RAP), Raw Byte Sequence Payload (RBSP), Sequence Parameter Set (SPS), Video Parameter Set (VPS), Versatile Video Coding (VVC).

[0054] Many video compression techniques can be employed to reduce the size of video files with minimal loss of data. For example, video compression techniques can include performing spatial (e.g., intra-picture) prediction and/or temporal (e.g., inter-picture) prediction to reduce or remove data redundancy in video sequences. For block-based video coding, a video slice (e.g., a video picture or a portion of a video picture) may be partitioned into video blocks, which may also be referred to as treeblocks, coding tree blocks (CTBs), coding tree units (CTUs), coding units (CUs), and/or coding nodes. Video blocks in an intra-coded (I) slice of a picture are coded using spatial prediction with respect to reference samples in neighboring blocks in the same picture. Video blocks in an inter-coded unidirectional prediction (P) or bidirectional prediction (B) slice of a picture may be coded by employing spatial prediction with respect to reference samples in neighboring blocks in the same picture or temporal prediction with respect to reference samples in other reference pictures. Pictures may be referred to as frames and/or images, and reference pictures may be referred to as reference frames and/or reference images. Spatial or temporal prediction results in a predictive block representing an image block. Residual data represents pixel differences between the original image block and the predictive block. Accordingly, an inter-coded block is encoded according to a motion vector that points to a block of reference samples forming the predictive block and the residual data indicating the difference between the coded block and the predictive block. An intra-coded block is encoded according to an intra-coding mode and the residual data. For further compression, the residual data may be transformed from the pixel domain to a transform domain. These result

in residual transform coefficients, which may be quantized. The quantized transform coefficients may initially be arranged in a two-dimensional array. The quantized transform coefficients may be scanned in order to produce a one-dimensional vector of transform coefficients. Entropy coding may be applied to achieve even more compression. Such video compression techniques are discussed in greater detail below.

[0055] To ensure an encoded video can be accurately decoded, video is encoded and decoded according to corresponding video coding standards. Video coding standards include International Telecommunication Union (ITU) Standardization Sector (ITU-T) H.261, International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) Motion Picture Experts Group (MPEG)-1 Part 2, ITU-T H.262 or ISO/IEC MPEG-2 Part 2, ITU-T H.263, ISO/IEC MPEG-4 Part 2, Advanced Video Coding (AVC), also known as ITU-T H.264 or ISO/IEC MPEG-4 Part 10, and High Efficiency Video Coding (HEVC), also known as ITU-T H.265 or MPEG-H Part 2. AVC includes extensions such as Scalable Video Coding (SVC), Multiview Video Coding (MVC) and Multiview Video Coding plus Depth (MVC+D), and three dimensional (3D) AVC (3D-AVC). HEVC includes extensions such as Scalable HEVC (SHVC), Multiview HEVC (MV-HEVC), and 3D HEVC (3D-HEVC). The joint video experts team (JVET) of ITU-T and ISO/IEC has begun developing a video coding standard referred to as Versatile Video Coding (VVC). VVC is included in a Working Draft (WD), which includes JVET-O2001-v14.

[0056] Some video coding systems employ supplemental enhancement information (SEI) messages. An SEI message contains information that is not needed by the decoding process in order to determine the values of the samples in decoded pictures. For example, the SEI messages may contain parameters used to check a bitstream for conformance with standards. In some instances, a video is coded into a bitstream using multiple layers. Scalable nesting SEI messages may be employed to contain SEI messages for multiple layers. For example, a scalable nesting SEI message contains scalable-nested SEI messages.

[0057] A first problem may occur when a list of layer identifiers (Ids) is included in the scalable nesting SEI message to indicate the correspondence between the scalable-nested SEI messages and the layers. The scalable nesting SEI message is contained in a SEI network abstraction layer (NAL) unit, which can be denoted as the current SEI NAL unit for clarity of discussion. The current SEI NAL unit is associated with a layer Id. The scalable nesting SEI message is always related to the layer associated with the current SEI NAL unit. This leads to redundant signaling as the layer Id of the current SEI NAL unit is included in the current SEI

NAL unit and is also included in the scalable nesting SEI message contained in the current NAL unit.

[0058] In a first example, disclosed herein are improvements to the scalable nesting SEI message for use in conjunction with specified layers. Specifically, the layer Id may be referred to as a NAL unit header layer Id (nuh_layer_id). The nuh_layer_id of the current SEI NAL unit is excluded from the scalable nesting SEI message. This can be done by employing a scalable nesting layer Id (layer_id[i]) syntax element that specifies the nuh_layer_id for each layer to which a scalable-nested SEI message applies. The scalable nesting layer_id[i] is constrained to include only nuh_layer_ids that are greater than the nuh_layer_id of the current SEI NAL unit. Hence, the nuh_layer_id of the current SEI NAL unit is omitted from the scalable nesting SEI message and inferred from the current SEI NAL unit. This improves the scalable nesting SEI message by omitting a redundant Id. For example, a loop that encodes/decodes the scalable nesting layer_id[i] executes one less time, which decreases processor resources during encoding and/or decoding. Further, this reduces the size of the encoded bitstream for each of the scalable nesting SEI messages in the bitstream. As a result, coding efficiency is increased, which reduces processor, memory, and/or network signaling resource usage at both the encoder and the decoder.

[0059] A second problem occurs when a flag is used to indicate that scalable-nested SEI messages in a scalable nesting SEI message apply to all layers in an access unit (AU). An AU is a set of coded pictures that are included in different layers and are associated with the same output time. This approach works when all AUs contain pictures for all layers. However, in some cases, pictures may be present for an output time in some layers and omitted in others. For example, a high framerate layer may include pictures at each AU, while a low framerate layer omits pictures in many of the AUs that are occupied by the higher framerate layer. The result is that the flag causes the SEI messages to only apply to layers that include pictures in the current AU. This causes conformance testing errors as a hypothetical reference decoder (HRD) may be unable to associate the scalable-nested SEI messages to the layers that omit pictures in a current AU. For example, such layers may contain pictures in later AUs after the current AU. In such a case, the HRD does not recognize the scalable-nested SEI messages as applying to such layers when the later AUs are tested for conformance. Accordingly, the HRD may be unable to correctly test such layers for conformance at later AUs when the flag for all layers is employed at an incomplete AU.

[0060] In a second example, disclosed herein is a scalable nesting all layers flag (all_layers_flag) to correct the abovementioned errors. The scalable nesting all_layers_flag can

be set to indicate that the scalable-nested SEI messages in a scalable nesting SEI message apply to all layers specified in a video parameter set (VPS). The VPS lists all layers. Hence, this approach allows the encoder to clearly indicate when the scalable-nested SEI messages are globally applied to all layers. The HRD conformance testing process can then check a bitstream for conformance without returning unpredictable errors at AUs that follow incomplete AUs that are associated with a scalable nesting SEI message. Hence, the scalable nesting all_layers_flag increases the functionality of an encoder and/or decoder by causing the bitstream to be correctly checked for conformance when incomplete AUs are present. Further, the scalable nesting all_layers_flag reduces the size of the encoded bitstream by omitting explicit layer signaling for scalable-nested SEI messages when such messages apply to all layers. As a result, coding efficiency is increased, which reduces processor, memory, and/or network signaling resource usage at both the encoder and the decoder.

[0061] FIG. 1 is a flowchart of an example operating method 100 of coding a video signal. Specifically, a video signal is encoded at an encoder. The encoding process compresses the video signal by employing various mechanisms to reduce the video file size. A smaller file size allows the compressed video file to be transmitted toward a user, while reducing associated bandwidth overhead. The decoder then decodes the compressed video file to reconstruct the original video signal for display to an end user. The decoding process generally mirrors the encoding process to allow the decoder to consistently reconstruct the video signal.

[0062] At step 101, the video signal is input into the encoder. For example, the video signal may be an uncompressed video file stored in memory. As another example, the video file may be captured by a video capture device, such as a video camera, and encoded to support live streaming of the video. The video file may include both an audio component and a video component. The video component contains a series of image frames that, when viewed in a sequence, gives the visual impression of motion. The frames contain pixels that are expressed in terms of light, referred to herein as luma components (or luma samples), and color, which is referred to as chroma components (or color samples). In some examples, the frames may also contain depth values to support three dimensional viewing.

[0063] At step 103, the video is partitioned into blocks. Partitioning includes subdividing the pixels in each frame into square and/or rectangular blocks for compression. For example, in High Efficiency Video Coding (HEVC) (also known as H.265 and MPEG-H Part 2) the frame can first be divided into coding tree units (CTUs), which are blocks of a predefined size (e.g., sixty-four pixels by sixty-four pixels). The CTUs contain both luma and chroma samples. Coding trees may be employed to divide the CTUs into blocks and then recursively subdivide

the blocks until configurations are achieved that support further encoding. For example, luma components of a frame may be subdivided until the individual blocks contain relatively homogenous lighting values. Further, chroma components of a frame may be subdivided until the individual blocks contain relatively homogenous color values. Accordingly, partitioning mechanisms vary depending on the content of the video frames.

[0064] At step 105, various compression mechanisms are employed to compress the image blocks partitioned at step 103. For example, inter-prediction and/or intra-prediction may be employed. Inter-prediction is designed to take advantage of the fact that objects in a common scene tend to appear in successive frames. Accordingly, a block depicting an object in a reference frame need not be repeatedly described in adjacent frames. Specifically, an object, such as a table, may remain in a constant position over multiple frames. Hence the table is described once and adjacent frames can refer back to the reference frame. Pattern matching mechanisms may be employed to match objects over multiple frames. Further, moving objects may be represented across multiple frames, for example due to object movement or camera movement. As a particular example, a video may show an automobile that moves across the screen over multiple frames. Motion vectors can be employed to describe such movement. A motion vector is a two-dimensional vector that provides an offset from the coordinates of an object in a frame to the coordinates of the object in a reference frame. As such, inter-prediction can encode an image block in a current frame as a set of motion vectors indicating an offset from a corresponding block in a reference frame.

[0065] Intra-prediction encodes blocks in a common frame. Intra-prediction takes advantage of the fact that luma and chroma components tend to cluster in a frame. For example, a patch of green in a portion of a tree tends to be positioned adjacent to similar patches of green. Intra-prediction employs multiple directional prediction modes (e.g., thirty-three in HEVC), a planar mode, and a direct current (DC) mode. The directional modes indicate that a current block is similar/the same as samples of a neighbor block in a corresponding direction. Planar mode indicates that a series of blocks along a row/column (e.g., a plane) can be interpolated based on neighbor blocks at the edges of the row. Planar mode, in effect, indicates a smooth transition of light/color across a row/column by employing a relatively constant slope in changing values. DC mode is employed for boundary smoothing and indicates that a block is similar/the same as an average value associated with samples of all the neighbor blocks associated with the angular directions of the directional prediction modes. Accordingly, intra-prediction blocks can represent image blocks as various relational prediction mode values instead of the actual values. Further, inter-prediction blocks can represent image blocks as motion vector values instead of

the actual values. In either case, the prediction blocks may not exactly represent the image blocks in some cases. Any differences are stored in residual blocks. Transforms may be applied to the residual blocks to further compress the file.

[0066] At step 107, various filtering techniques may be applied. In HEVC, the filters are applied according to an in-loop filtering scheme. The block based prediction discussed above may result in the creation of blocky images at the decoder. Further, the block based prediction scheme may encode a block and then reconstruct the encoded block for later use as a reference block. The in-loop filtering scheme iteratively applies noise suppression filters, de-blocking filters, adaptive loop filters, and sample adaptive offset (SAO) filters to the blocks/frames. These filters mitigate such blocking artifacts so that the encoded file can be accurately reconstructed. Further, these filters mitigate artifacts in the reconstructed reference blocks so that artifacts are less likely to create additional artifacts in subsequent blocks that are encoded based on the reconstructed reference blocks.

[0067] Once the video signal has been partitioned, compressed, and filtered, the resulting data is encoded in a bitstream at step 109. The bitstream includes the data discussed above as well as any signaling data desired to support proper video signal reconstruction at the decoder. For example, such data may include partition data, prediction data, residual blocks, and various flags providing coding instructions to the decoder. The bitstream may be stored in memory for transmission toward a decoder upon request. The bitstream may also be broadcast and/or multicast toward a plurality of decoders. The creation of the bitstream is an iterative process. Accordingly, steps 101, 103, 105, 107, and 109 may occur continuously and/or simultaneously over many frames and blocks. The order shown in FIG. 1 is presented for clarity and ease of discussion, and is not intended to limit the video coding process to a particular order.

[0068] The decoder receives the bitstream and begins the decoding process at step 111. Specifically, the decoder employs an entropy decoding scheme to convert the bitstream into corresponding syntax and video data. The decoder employs the syntax data from the bitstream to determine the partitions for the frames at step 111. The partitioning should match the results of block partitioning at step 103. Entropy encoding/decoding as employed in step 111 is now described. The encoder makes many choices during the compression process, such as selecting block partitioning schemes from several possible choices based on the spatial positioning of values in the input image(s). Signaling the exact choices may employ a large number of bins. As used herein, a bin is a binary value that is treated as a variable (e.g., a bit value that may vary depending on context). Entropy coding allows the encoder to discard any options that are clearly not viable for a particular case, leaving a set of allowable options. Each allowable option is then

assigned a code word. The length of the code words is based on the number of allowable options (e.g., one bin for two options, two bins for three to four options, etc.) The encoder then encodes the code word for the selected option. This scheme reduces the size of the code words as the code words are as big as desired to uniquely indicate a selection from a small sub-set of allowable options as opposed to uniquely indicating the selection from a potentially large set of all possible options. The decoder then decodes the selection by determining the set of allowable options in a similar manner to the encoder. By determining the set of allowable options, the decoder can read the code word and determine the selection made by the encoder.

[0069] At step 113, the decoder performs block decoding. Specifically, the decoder employs reverse transforms to generate residual blocks. Then the decoder employs the residual blocks and corresponding prediction blocks to reconstruct the image blocks according to the partitioning. The prediction blocks may include both intra-prediction blocks and inter-prediction blocks as generated at the encoder at step 105. The reconstructed image blocks are then positioned into frames of a reconstructed video signal according to the partitioning data determined at step 111. Syntax for step 113 may also be signaled in the bitstream via entropy coding as discussed above.

[0070] At step 115, filtering is performed on the frames of the reconstructed video signal in a manner similar to step 107 at the encoder. For example, noise suppression filters, de-blocking filters, adaptive loop filters, and SAO filters may be applied to the frames to remove blocking artifacts. Once the frames are filtered, the video signal can be output to a display at step 117 for viewing by an end user.

[0071] FIG. 2 is a schematic diagram of an example coding and decoding (codec) system 200 for video coding. Specifically, codec system 200 provides functionality to support the implementation of operating method 100. Codec system 200 is generalized to depict components employed in both an encoder and a decoder. Codec system 200 receives and partitions a video signal as discussed with respect to steps 101 and 103 in operating method 100, which results in a partitioned video signal 201. Codec system 200 then compresses the partitioned video signal 201 into a coded bitstream when acting as an encoder as discussed with respect to steps 105, 107, and 109 in method 100. When acting as a decoder, codec system 200 generates an output video signal from the bitstream as discussed with respect to steps 111, 113, 115, and 117 in operating method 100. The codec system 200 includes a general coder control component 211, a transform scaling and quantization component 213, an intra-picture estimation component 215, an intra-picture prediction component 217, a motion compensation component 219, a motion estimation component 221, a scaling and inverse transform component 229, a filter control

analysis component 227, an in-loop filters component 225, a decoded picture buffer component 223, and a header formatting and context adaptive binary arithmetic coding (CABAC) component 231. Such components are coupled as shown. In FIG. 2, black lines indicate movement of data to be encoded/decoded while dashed lines indicate movement of control data that controls the operation of other components. The components of codec system 200 may all be present in the encoder. The decoder may include a subset of the components of codec system 200. For example, the decoder may include the intra-picture prediction component 217, the motion compensation component 219, the scaling and inverse transform component 229, the in-loop filters component 225, and the decoded picture buffer component 223. These components are now described.

[0072] The partitioned video signal 201 is a captured video sequence that has been partitioned into blocks of pixels by a coding tree. A coding tree employs various split modes to subdivide a block of pixels into smaller blocks of pixels. These blocks can then be further subdivided into smaller blocks. The blocks may be referred to as nodes on the coding tree. Larger parent nodes are split into smaller child nodes. The number of times a node is subdivided is referred to as the depth of the node/coding tree. The divided blocks can be included in coding units (CUs) in some cases. For example, a CU can be a sub-portion of a CTU that contains a luma block, red difference chroma (Cr) block(s), and a blue difference chroma (Cb) block(s) along with corresponding syntax instructions for the CU. The split modes may include a binary tree (BT), triple tree (TT), and a quad tree (QT) employed to partition a node into two, three, or four child nodes, respectively, of varying shapes depending on the split modes employed. The partitioned video signal 201 is forwarded to the general coder control component 211, the transform scaling and quantization component 213, the intra-picture estimation component 215, the filter control analysis component 227, and the motion estimation component 221 for compression.

[0073] The general coder control component 211 is configured to make decisions related to coding of the images of the video sequence into the bitstream according to application constraints. For example, the general coder control component 211 manages optimization of bitrate/bitstream size versus reconstruction quality. Such decisions may be made based on storage space/bandwidth availability and image resolution requests. The general coder control component 211 also manages buffer utilization in light of transmission speed to mitigate buffer underrun and overrun issues. To manage these issues, the general coder control component 211 manages partitioning, prediction, and filtering by the other components. For example, the general coder control component 211 may dynamically increase compression complexity to

increase resolution and increase bandwidth usage or decrease compression complexity to decrease resolution and bandwidth usage. Hence, the general coder control component 211 controls the other components of codec system 200 to balance video signal reconstruction quality with bit rate concerns. The general coder control component 211 creates control data, which controls the operation of the other components. The control data is also forwarded to the header formatting and CABAC component 231 to be encoded in the bitstream to signal parameters for decoding at the decoder.

[0074] The partitioned video signal 201 is also sent to the motion estimation component 221 and the motion compensation component 219 for inter-prediction. A frame or slice of the partitioned video signal 201 may be divided into multiple video blocks. Motion estimation component 221 and the motion compensation component 219 perform inter-predictive coding of the received video block relative to one or more blocks in one or more reference frames to provide temporal prediction. Codec system 200 may perform multiple coding passes, e.g., to select an appropriate coding mode for each block of video data.

[0075] Motion estimation component 221 and motion compensation component 219 may be highly integrated, but are illustrated separately for conceptual purposes. Motion estimation, performed by motion estimation component 221, is the process of generating motion vectors, which estimate motion for video blocks. A motion vector, for example, may indicate the displacement of a coded object relative to a predictive block. A predictive block is a block that is found to closely match the block to be coded, in terms of pixel difference. A predictive block may also be referred to as a reference block. Such pixel difference may be determined by sum of absolute difference (SAD), sum of square difference (SSD), or other difference metrics. HEVC employs several coded objects including a CTU, coding tree blocks (CTBs), and CUs. For example, a CTU can be divided into CTBs, which can then be divided into CBs for inclusion in CUs. A CU can be encoded as a prediction unit (PU) containing prediction data and/or a transform unit (TU) containing transformed residual data for the CU. The motion estimation component 221 generates motion vectors, PUs, and TUs by using a rate-distortion analysis as part of a rate distortion optimization process. For example, the motion estimation component 221 may determine multiple reference blocks, multiple motion vectors, etc. for a current block/frame, and may select the reference blocks, motion vectors, etc. having the best rate-distortion characteristics. The best rate-distortion characteristics balance both quality of video reconstruction (e.g., amount of data loss by compression) with coding efficiency (e.g., size of the final encoding).

[0076] In some examples, codec system 200 may calculate values for sub-integer pixel positions of reference pictures stored in decoded picture buffer component 223. For example, video codec system 200 may interpolate values of one-quarter pixel positions, one-eighth pixel positions, or other fractional pixel positions of the reference picture. Therefore, motion estimation component 221 may perform a motion search relative to the full pixel positions and fractional pixel positions and output a motion vector with fractional pixel precision. The motion estimation component 221 calculates a motion vector for a PU of a video block in an inter-coded slice by comparing the position of the PU to the position of a predictive block of a reference picture. Motion estimation component 221 outputs the calculated motion vector as motion data to header formatting and CABAC component 231 for encoding and motion to the motion compensation component 219.

[0077] Motion compensation, performed by motion compensation component 219, may involve fetching or generating the predictive block based on the motion vector determined by motion estimation component 221. Again, motion estimation component 221 and motion compensation component 219 may be functionally integrated, in some examples. Upon receiving the motion vector for the PU of the current video block, motion compensation component 219 may locate the predictive block to which the motion vector points. A residual video block is then formed by subtracting pixel values of the predictive block from the pixel values of the current video block being coded, forming pixel difference values. In general, motion estimation component 221 performs motion estimation relative to luma components, and motion compensation component 219 uses motion vectors calculated based on the luma components for both chroma components and luma components. The predictive block and residual block are forwarded to transform scaling and quantization component 213.

[0078] The partitioned video signal 201 is also sent to intra-picture estimation component 215 and intra-picture prediction component 217. As with motion estimation component 221 and motion compensation component 219, intra-picture estimation component 215 and intra-picture prediction component 217 may be highly integrated, but are illustrated separately for conceptual purposes. The intra-picture estimation component 215 and intra-picture prediction component 217 intra-predict a current block relative to blocks in a current frame, as an alternative to the inter-prediction performed by motion estimation component 221 and motion compensation component 219 between frames, as described above. In particular, the intra-picture estimation component 215 determines an intra-prediction mode to use to encode a current block. In some examples, intra-picture estimation component 215 selects an appropriate intra-prediction mode to encode a current block from multiple tested intra-prediction modes. The selected intra-

prediction modes are then forwarded to the header formatting and CABAC component 231 for encoding.

[0079] For example, the intra-picture estimation component 215 calculates rate-distortion values using a rate-distortion analysis for the various tested intra-prediction modes, and selects the intra-prediction mode having the best rate-distortion characteristics among the tested modes. Rate-distortion analysis generally determines an amount of distortion (or error) between an encoded block and an original unencoded block that was encoded to produce the encoded block, as well as a bitrate (e.g., a number of bits) used to produce the encoded block. The intra-picture estimation component 215 calculates ratios from the distortions and rates for the various encoded blocks to determine which intra-prediction mode exhibits the best rate-distortion value for the block. In addition, intra-picture estimation component 215 may be configured to code depth blocks of a depth map using a depth modeling mode (DMM) based on rate-distortion optimization (RDO).

[0080] The intra-picture prediction component 217 may generate a residual block from the predictive block based on the selected intra-prediction modes determined by intra-picture estimation component 215 when implemented on an encoder or read the residual block from the bitstream when implemented on a decoder. The residual block includes the difference in values between the predictive block and the original block, represented as a matrix. The residual block is then forwarded to the transform scaling and quantization component 213. The intra-picture estimation component 215 and the intra-picture prediction component 217 may operate on both luma and chroma components.

[0081] The transform scaling and quantization component 213 is configured to further compress the residual block. The transform scaling and quantization component 213 applies a transform, such as a discrete cosine transform (DCT), a discrete sine transform (DST), or a conceptually similar transform, to the residual block, producing a video block comprising residual transform coefficient values. Wavelet transforms, integer transforms, sub-band transforms or other types of transforms could also be used. The transform may convert the residual information from a pixel value domain to a transform domain, such as a frequency domain. The transform scaling and quantization component 213 is also configured to scale the transformed residual information, for example based on frequency. Such scaling involves applying a scale factor to the residual information so that different frequency information is quantized at different granularities, which may affect final visual quality of the reconstructed video. The transform scaling and quantization component 213 is also configured to quantize the transform coefficients to further reduce bit rate. The quantization process may reduce the bit

depth associated with some or all of the coefficients. The degree of quantization may be modified by adjusting a quantization parameter. In some examples, the transform scaling and quantization component 213 may then perform a scan of the matrix including the quantized transform coefficients. The quantized transform coefficients are forwarded to the header formatting and CABAC component 231 to be encoded in the bitstream.

[0082] The scaling and inverse transform component 229 applies a reverse operation of the transform scaling and quantization component 213 to support motion estimation. The scaling and inverse transform component 229 applies inverse scaling, transformation, and/or quantization to reconstruct the residual block in the pixel domain, e.g., for later use as a reference block which may become a predictive block for another current block. The motion estimation component 221 and/or motion compensation component 219 may calculate a reference block by adding the residual block back to a corresponding predictive block for use in motion estimation of a later block/frame. Filters are applied to the reconstructed reference blocks to mitigate artifacts created during scaling, quantization, and transform. Such artifacts could otherwise cause inaccurate prediction (and create additional artifacts) when subsequent blocks are predicted.

[0083] The filter control analysis component 227 and the in-loop filters component 225 apply the filters to the residual blocks and/or to reconstructed image blocks. For example, the transformed residual block from the scaling and inverse transform component 229 may be combined with a corresponding prediction block from intra-picture prediction component 217 and/or motion compensation component 219 to reconstruct the original image block. The filters may then be applied to the reconstructed image block. In some examples, the filters may instead be applied to the residual blocks. As with other components in FIG. 2, the filter control analysis component 227 and the in-loop filters component 225 are highly integrated and may be implemented together, but are depicted separately for conceptual purposes. Filters applied to the reconstructed reference blocks are applied to particular spatial regions and include multiple parameters to adjust how such filters are applied. The filter control analysis component 227 analyzes the reconstructed reference blocks to determine where such filters should be applied and sets corresponding parameters. Such data is forwarded to the header formatting and CABAC component 231 as filter control data for encoding. The in-loop filters component 225 applies such filters based on the filter control data. The filters may include a deblocking filter, a noise suppression filter, a SAO filter, and an adaptive loop filter. Such filters may be applied in the spatial/pixel domain (e.g., on a reconstructed pixel block) or in the frequency domain, depending on the example.

[0084] When operating as an encoder, the filtered reconstructed image block, residual block, and/or prediction block are stored in the decoded picture buffer component 223 for later use in motion estimation as discussed above. When operating as a decoder, the decoded picture buffer component 223 stores and forwards the reconstructed and filtered blocks toward a display as part of an output video signal. The decoded picture buffer component 223 may be any memory device capable of storing prediction blocks, residual blocks, and/or reconstructed image blocks.

[0085] The header formatting and CABAC component 231 receives the data from the various components of codec system 200 and encodes such data into a coded bitstream for transmission toward a decoder. Specifically, the header formatting and CABAC component 231 generates various headers to encode control data, such as general control data and filter control data. Further, prediction data, including intra-prediction and motion data, as well as residual data in the form of quantized transform coefficient data are all encoded in the bitstream. The final bitstream includes all information desired by the decoder to reconstruct the original partitioned video signal 201. Such information may also include intra-prediction mode index tables (also referred to as codeword mapping tables), definitions of encoding contexts for various blocks, indications of most probable intra-prediction modes, an indication of partition information, etc. Such data may be encoded by employing entropy coding. For example, the information may be encoded by employing context adaptive variable length coding (CAVLC), CABAC, syntax-based context-adaptive binary arithmetic coding (SBAC), probability interval partitioning entropy (PIPE) coding, or another entropy coding technique. Following the entropy coding, the coded bitstream may be transmitted to another device (e.g., a video decoder) or archived for later transmission or retrieval.

[0086] FIG. 3 is a block diagram illustrating an example video encoder 300. Video encoder 300 may be employed to implement the encoding functions of codec system 200 and/or implement steps 101, 103, 105, 107, and/or 109 of operating method 100. Encoder 300 partitions an input video signal, resulting in a partitioned video signal 301, which is substantially similar to the partitioned video signal 201. The partitioned video signal 301 is then compressed and encoded into a bitstream by components of encoder 300.

[0087] Specifically, the partitioned video signal 301 is forwarded to an intra-picture prediction component 317 for intra-prediction. The intra-picture prediction component 317 may be substantially similar to intra-picture estimation component 215 and intra-picture prediction component 217. The partitioned video signal 301 is also forwarded to a motion compensation component 321 for inter-prediction based on reference blocks in a decoded picture buffer component 323. The motion compensation component 321 may be substantially similar to

motion estimation component 221 and motion compensation component 219. The prediction blocks and residual blocks from the intra-picture prediction component 317 and the motion compensation component 321 are forwarded to a transform and quantization component 313 for transform and quantization of the residual blocks. The transform and quantization component 313 may be substantially similar to the transform scaling and quantization component 213. The transformed and quantized residual blocks and the corresponding prediction blocks (along with associated control data) are forwarded to an entropy coding component 331 for coding into a bitstream. The entropy coding component 331 may be substantially similar to the header formatting and CABAC component 231.

[0088] The transformed and quantized residual blocks and/or the corresponding prediction blocks are also forwarded from the transform and quantization component 313 to an inverse transform and quantization component 329 for reconstruction into reference blocks for use by the motion compensation component 321. The inverse transform and quantization component 329 may be substantially similar to the scaling and inverse transform component 229. In-loop filters in an in-loop filters component 325 are also applied to the residual blocks and/or reconstructed reference blocks, depending on the example. The in-loop filters component 325 may be substantially similar to the filter control analysis component 227 and the in-loop filters component 225. The in-loop filters component 325 may include multiple filters as discussed with respect to in-loop filters component 225. The filtered blocks are then stored in a decoded picture buffer component 323 for use as reference blocks by the motion compensation component 321. The decoded picture buffer component 323 may be substantially similar to the decoded picture buffer component 223.

[0089] FIG. 4 is a block diagram illustrating an example video decoder 400. Video decoder 400 may be employed to implement the decoding functions of codec system 200 and/or implement steps 111, 113, 115, and/or 117 of operating method 100. Decoder 400 receives a bitstream, for example from an encoder 300, and generates a reconstructed output video signal based on the bitstream for display to an end user.

[0090] The bitstream is received by an entropy decoding component 433. The entropy decoding component 433 is configured to implement an entropy decoding scheme, such as CAVLC, CABAC, SBAC, PIPE coding, or other entropy coding techniques. For example, the entropy decoding component 433 may employ header information to provide a context to interpret additional data encoded as codewords in the bitstream. The decoded information includes any desired information to decode the video signal, such as general control data, filter control data, partition information, motion data, prediction data, and quantized transform

coefficients from residual blocks. The quantized transform coefficients are forwarded to an inverse transform and quantization component 429 for reconstruction into residual blocks. The inverse transform and quantization component 429 may be similar to inverse transform and quantization component 329.

[0091] The reconstructed residual blocks and/or prediction blocks are forwarded to intra-picture prediction component 417 for reconstruction into image blocks based on intra-prediction operations. The intra-picture prediction component 417 may be similar to intra-picture estimation component 215 and an intra-picture prediction component 217. Specifically, the intra-picture prediction component 417 employs prediction modes to locate a reference block in the frame and applies a residual block to the result to reconstruct intra-predicted image blocks. The reconstructed intra-predicted image blocks and/or the residual blocks and corresponding inter-prediction data are forwarded to a decoded picture buffer component 423 via an in-loop filters component 425, which may be substantially similar to decoded picture buffer component 223 and in-loop filters component 225, respectively. The in-loop filters component 425 filters the reconstructed image blocks, residual blocks and/or prediction blocks, and such information is stored in the decoded picture buffer component 423. Reconstructed image blocks from decoded picture buffer component 423 are forwarded to a motion compensation component 421 for inter-prediction. The motion compensation component 421 may be substantially similar to motion estimation component 221 and/or motion compensation component 219. Specifically, the motion compensation component 421 employs motion vectors from a reference block to generate a prediction block and applies a residual block to the result to reconstruct an image block. The resulting reconstructed blocks may also be forwarded via the in-loop filters component 425 to the decoded picture buffer component 423. The decoded picture buffer component 423 continues to store additional reconstructed image blocks, which can be reconstructed into frames via the partition information. Such frames may also be placed in a sequence. The sequence is output toward a display as a reconstructed output video signal.

[0092] FIG. 5 is a schematic diagram illustrating an example HRD 500. A HRD 500 may be employed in an encoder, such as codec system 200 and/or encoder 300. The HRD 500 may check the bitstream created at step 109 of method 100 before the bitstream is forwarded to a decoder, such as decoder 400. In some examples, the bitstream may be continuously forwarded through the HRD 500 as the bitstream is encoded. In the event that a portion of the bitstream fails to conform to associated constraints, the HRD 500 can indicate such failure to an encoder to cause the encoder to re-encode the corresponding section of the bitstream with different mechanisms.

[0093] The HRD 500 includes a hypothetical stream scheduler (HSS) 541. A HSS 541 is a component configured to perform a hypothetical delivery mechanism. The hypothetical delivery mechanism is used for checking the conformance of a bitstream or a decoder with regards to the timing and data flow of a bitstream 551 input into the HRD 500. For example, the HSS 541 may receive a bitstream 551 output from an encoder and manage the conformance testing process on the bitstream 551. In a particular example, the HSS 541 can control the rate that coded pictures move through the HRD 500 and verify that the bitstream 551 does not contain non-conforming data.

[0094] The HSS 541 may forward the bitstream 551 to a CPB 543 at a predefined rate. The HRD 500 may manage data in decoding units (DU) 553. A DU 553 is an Access Unit (AU) or a sub-set of an AU and associated non-video coding layer (VCL) network abstraction layer (NAL) units. Specifically, an AU contains one or more pictures associated with an output time. For example, an AU may contain a single picture in a single layer bitstream, and may contain a picture for each layer in a multi-layer bitstream. Each picture of an AU may be divided into slices that are each included in a corresponding VCL NAL unit. Hence, a DU 553 may contain one or more pictures, one or more slices of a picture, or combinations thereof. Also, parameters used to decode the AU, pictures, and/or slices can be included in non-VCL NAL units. As such, the DU 553 contains non-VCL NAL units that contain data needed to support decoding the VCL NAL units in the DU 553. The CPB 543 is a first-in first-out buffer in the HRD 500. The CPB 543 contains DUs 553 including video data in decoding order. The CPB 543 stores the video data for use during bitstream conformance verification.

[0095] The CPB 543 forwards the DUs 553 to a decoding process component 545. The decoding process component 545 is a component that conforms to the VVC standard. For example, the decoding process component 545 may emulate a decoder 400 employed by an end user. The decoding process component 545 decodes the DUs 553 at a rate that can be achieved by an example end user decoder. If the decoding process component 545 cannot decode the DUs 553 fast enough to prevent an overflow of the CPB 543, then the bitstream 551 does not conform to the standard and should be re-encoded.

[0096] The decoding process component 545 decodes the DUs 553, which creates decoded DUs 555. A decoded DU 555 contains a decoded picture. The decoded DUs 555 are forwarded to a DPB 547. The DPB 547 may be substantially similar to a decoded picture buffer component 223, 323, and/or 423. To support inter-prediction, pictures that are marked for use as reference pictures 556 that are obtained from the decoded DUs 555 are returned to the decoding process component 545 to support further decoding. The DPB 547 outputs the decoded video sequence

as a series of pictures 557. The pictures 557 are reconstructed pictures that generally mirror pictures encoded into the bitstream 551 by the encoder.

[0097] The pictures 557 are forwarded to an output cropping component 549. The output cropping component 549 is configured to apply a conformance cropping window to the pictures 557. This results in output cropped pictures 559. An output cropped picture 559 is a completely reconstructed picture. Accordingly, the output cropped picture 559 mimics what an end user would see upon decoding the bitstream 551. As such, the encoder can review the output cropped pictures 559 to ensure the encoding is satisfactory.

[0098] The HRD 500 is initialized based on HRD parameters in the bitstream 551. For example, the HRD 500 may read HRD parameters from a VPS, a SPS, and/or SEI messages. The HRD 500 may then perform conformance testing operations on the bitstream 551 based on the information in such HRD parameters. As a specific example, the HRD 500 may determine one or more CPB delivery schedules from the HRD parameters. A delivery schedule specifies timing for delivery of video data to and/or from a memory location, such as a CPB and/or a DPB. Hence, a CPB delivery schedule specifies timing for delivery of AUs, DUs 553, and/or pictures, to/from the CPB 543. It should be noted that the HRD 500 may employ DPB delivery schedules for the DPB 547 that are similar to the CPB delivery schedules.

[0099] Video may be coded into different layers and/or OLSs for use by decoders with varying levels of hardware capabilities as well as for varying network conditions. The CPB delivery schedules are selected to reflect these issues. Accordingly, higher layer sub-bitstreams are designated for optimal hardware and network conditions and hence higher layers may receive one or more CPB delivery schedules that employ a large amount of memory in the CPB 543 and short delays for transfers of the DUs 553 toward the DPB 547. Likewise, lower layer sub-bitstreams are designated for limited decoder hardware capabilities and/or poor network conditions. Hence, lower layers may receive one or more CPB delivery schedules that employ a small amount of memory in the CPB 543 and longer delays for transfers of the DUs 553 toward the DPB 547. The OLSs, layers, sublayers, or combinations thereof can then be tested according to the corresponding delivery schedule to ensure that the resulting sub-bitstream can be correctly decoded under the conditions that are expected for the sub-bitstream. Accordingly, the HRD parameters in the bitstream 551 can indicate the CPB delivery schedules as well as include sufficient data to allow the HRD 500 to determine the CPB delivery schedules and correlate the CPB delivery schedules to the corresponding OLSs, layers, and/or sublayers.

[00100] FIG. 6 is a schematic diagram illustrating an example multi-layer video sequence 600 configured for inter-layer prediction 621. The multi-layer video sequence 600 may be

encoded by an encoder, such as codec system 200 and/or encoder 300 and decoded by a decoder, such as codec system 200 and/or decoder 400, for example according to method 100. Further, the multi-layer video sequence 600 can be checked for standard conformance by a HRD, such as HRD 500. The multi-layer video sequence 600 is included to depict an example application for layers in a coded video sequence. A multi-layer video sequence 600 is any video sequence that employs a plurality of layers, such as layer N 631 and layer N+1 632.

[00101] In an example, the multi-layer video sequence 600 may employ inter-layer prediction 621. Inter-layer prediction 621 is applied between pictures 611, 612, 613, and 614 and pictures 615, 616, 617, and 618 in different layers. In the example shown, pictures 611, 612, 613, and 614 are part of layer N+1 632 and pictures 615, 616, 617, and 618 are part of layer N 631. A layer, such as layer N 631 and/or layer N+1 632, is a group of pictures that are all associated with a similar value of a characteristic, such as a similar size, quality, resolution, signal to noise ratio, capability, etc. A layer may be defined formally as a set of VCL NAL units and associated non-VCL NAL units. A VCL NAL unit is a NAL unit coded to contain video data, such as a coded slice of a picture. A non-VCL NAL unit is a NAL unit that contains non-video data such as syntax and/or parameters that support decoding the video data, performance of conformance checking, or other operations.

[00102] In the example shown, layer N+1 632 is associated with a larger image size than layer N 631. Accordingly, pictures 611, 612, 613, and 614 in layer N+1 632 have a larger picture size (e.g., larger height and width and hence more samples) than pictures 615, 616, 617, and 618 in layer N 631 in this example. However, such pictures can be separated between layer N+1 632 and layer N 631 by other characteristics. While only two layers, layer N+1 632 and layer N 631, are shown, a set of pictures can be separated into any number of layers based on associated characteristics. Layer N+1 632 and layer N 631 may also be denoted by a layer Id. A layer Id is an item of data that is associated with a picture and denotes the picture is part of an indicated layer. Accordingly, each picture 611-618 may be associated with a corresponding layer Id to indicate which layer N+1 632 or layer N 631 includes the corresponding picture. For example, a layer Id may include a NAL unit header layer identifier (nuh_layer_id), which is a syntax element that specifies an identifier of a layer that includes a NAL unit (e.g., that include slices and/or parameters of the pictures in a layer). A layer associated with a lower quality/bitstream size, such as layer N 631, is generally assigned a lower layer Id and is referred to as a lower layer. Further, a layer associated with a higher quality/bitstream size, such as layer N+1 632, is generally assigned a higher layer Id and is referred to as a higher layer.

[00103] Pictures 611-618 in different layers 631-632 are configured to be displayed in the alternative. As a specific example, a decoder may decode and display picture 615 at a current display time if a smaller picture is desired or the decoder may decode and display picture 611 at the current display time if a larger picture is desired. As such, pictures 611-614 at higher layer N+1 632 contain substantially the same image data as corresponding pictures 615-618 at lower layer N 631 (notwithstanding the difference in picture size). Specifically, picture 611 contains substantially the same image data as picture 615, picture 612 contains substantially the same image data as picture 616, etc.

[00104] Pictures 611-618 can be coded by reference to other pictures 611-618 in the same layer N 631 or N+1 632. Coding a picture in reference to another picture in the same layer results in inter-prediction 623. Inter-prediction 623 is depicted by solid line arrows. For example, picture 613 may be coded by employing inter-prediction 623 using one or two of pictures 611, 612, and/or 614 in layer N+1 632 as a reference, where one picture is referenced for unidirectional inter-prediction and/or two pictures are referenced for bidirectional inter-prediction. Further, picture 617 may be coded by employing inter-prediction 623 using one or two of pictures 615, 616, and/or 618 in layer N 631 as a reference, where one picture is referenced for unidirectional inter-prediction and/or two pictures are referenced for bidirectional inter-prediction. When a picture is used as a reference for another picture in the same layer when performing inter-prediction 623, the picture may be referred to as a reference picture. For example, picture 612 may be a reference picture used to code picture 613 according to inter-prediction 623. Inter-prediction 623 can also be referred to as intra-layer prediction in a multi-layer context. As such, inter-prediction 623 is a mechanism of coding samples of a current picture by reference to indicated samples in a reference picture that is different from the current picture where the reference picture and the current picture are in the same layer.

[00105] Pictures 611-618 can also be coded by reference to other pictures 611-618 in different layers. This process is known as inter-layer prediction 621, and is depicted by dashed arrows. Inter-layer prediction 621 is a mechanism of coding samples of a current picture by reference to indicated samples in a reference picture where the current picture and the reference picture are in different layers and hence have different layer IDs. For example, a picture in a lower layer N 631 can be used as a reference picture to code a corresponding picture at a higher layer N+1 632. As a specific example, picture 611 can be coded by reference to picture 615 according to inter-layer prediction 621. In such a case, the picture 615 is used as an inter-layer reference picture. An inter-layer reference picture is a reference picture used for inter-layer prediction 621. In most cases, inter-layer prediction 621 is constrained such that a current

picture, such as picture 611, can only use inter-layer reference picture(s) that are included in the same AU and that are at a lower layer, such as picture 615. When multiple layers (e.g., more than two) are available, inter-layer prediction 621 can encode/decode a current picture based on multiple inter-layer reference picture(s) at lower levels than the current picture.

[00106] A video encoder can employ a multi-layer video sequence 600 to encode pictures 611-618 via many different combinations and/or permutations of inter-prediction 623 and inter-layer prediction 621. For example, picture 615 may be coded according to intra-prediction. Pictures 616-618 can then be coded according to inter-prediction 623 by using picture 615 as a reference picture. Further, picture 611 may be coded according to inter-layer prediction 621 by using picture 615 as an inter-layer reference picture. Pictures 612-614 can then be coded according to inter-prediction 623 by using picture 611 as a reference picture. As such, a reference picture can serve as both a single layer reference picture and an inter-layer reference picture for different coding mechanisms. By coding higher layer N+1 632 pictures based on lower layer N 631 pictures, the higher layer N+1 632 can avoid employing intra-prediction, which has much lower coding efficiency than inter-prediction 623 and inter-layer prediction 621. As such, the poor coding efficiency of intra-prediction can be limited to the smallest/lowest quality pictures, and hence limited to coding the smallest amount of video data. The pictures used as reference pictures and/or inter-layer reference pictures can be indicated in entries of reference picture list(s) contained in a reference picture list structure.

[00107] The pictures 611-618 may also be included in access units (AUs) 627 and 628. An AU 627-628 is a set of coded pictures that are included in different layers and are associated with the same output time during decoding. Accordingly, coded pictures in the same AU 627-628 are scheduled for output from a DPB at a decoder at the same time. For example, pictures 614 and 618 are in the same AU 628. Pictures 613 and 617 are in a different AU 627 from pictures 614 and 618. Pictures 614 and 618 in the same AU 628 may be displayed in the alternative. For example, picture 618 may be displayed when a small picture size is desired and picture 614 may be displayed when a large picture size is desired. When the large picture size is desired, picture 614 is output and picture 618 is used only for inter-layer prediction 621. In this case, picture 618 is discarded without being output once inter-layer prediction 621 is complete. An AU 627 that includes a picture at all layers can be referred to as a complete AU 627. An AU 628 that does not include pictures at all layers can be referred to as an incomplete AU 628.

[00108] Layer N-1 633 is included to depict the scenario that an AU 628 is incomplete. Layer N-1 633 includes pictures 625 and 626, which are substantially the same as pictures 615 and

617, respectively. However, layer N-1 633 does not contain pictures that correspond to pictures 616 and 618. Accordingly, layer N-1 633 is substantially similar to layer N 631, but contains a lower frame rate. Accordingly, the AU 627 that contains pictures 611, 615, and 625 is complete. However, the AU 628 that contains pictures 612 and 616 is an incomplete AU 628 because this AU 628 is missing a picture at layer N-1 633.

[00109] Encoding incomplete AUs 628 can be useful as this allows the multi-layer video sequence 600 to be transmitted to different users at different frame rates. However, incomplete AUs 628 can create challenges when checking a coded video sequence for conformance. For example, SEI messages may be employed to transmit parameters that are used by a HRD, such as HRD 500, when checking a multi-layer video sequence 600 for conformance with standards. Some video coding systems allow SEI messages to apply to all layers in an AU 627-628. The problem is that such SEI messages may be persistent over multiple AUs 627-628. When a SEI message applies to all layers in an incomplete AU 628, such SEI messages are not applied to layer N-1 633 because layer N-1 633 is not included in the incomplete AU 628. Accordingly, the SEI message is not applied to layer N-1 633 in complete AUs 627 following the incomplete AU 628. This can cause unpredictable errors as the SEI messages are properly applied to layer N-1 633 when such messages happen to be associated with a complete AU 627 and are not applied to layer N-1 633 when such messages happen to be associated with an incomplete AU 628. This and other problems are addressed by the signaling scheme described below.

[00110] FIG. 7 is a schematic diagram illustrating an example bitstream 700. For example, the bitstream 700 can be generated by a codec system 200 and/or an encoder 300 for decoding by a codec system 200 and/or a decoder 400 according to method 100. Further, the bitstream 700 may include a multi-layer video sequence 600. In addition, the bitstream 700 may include various parameters to control the operation of a HRD, such as HRD 500. Based on such parameters, the HRD 500 can check the bitstream 700 for conformance with standards prior to transmission toward a decoder for decoding.

[00111] The bitstream 700 includes a VPS 711, one or more SPSs 713, a plurality of picture parameter sets (PPSs) 715, a plurality of slice headers 717, image data 720, and SEI messages 719. A VPS 711 contains data related to the entire bitstream 700. For example, the VPS 711 may contain data related OLSs, layers, and/or sublayers used in the bitstream 700. An SPS 713 contains sequence data common to all pictures in a coded video sequence contained in the bitstream 700. For example, each layer may contain one or more coded video sequences, and each coded video sequence may reference a SPS 713 for corresponding parameters. The parameters in a SPS 713 can include picture sizing, bit depth, coding tool parameters, bit rate

restrictions, etc. It should be noted that, while each sequence refers to a SPS 713, a single SPS 713 can contain data for multiple sequences in some examples. The PPS 715 contains parameters that apply to an entire picture. Hence, each picture in the video sequence may refer to a PPS 715. It should be noted that, while each picture refers to a PPS 715, a single PPS 715 can contain data for multiple pictures in some examples. For example, multiple similar pictures may be coded according to similar parameters. In such a case, a single PPS 715 may contain data for such similar pictures. The PPS 715 can indicate coding tools available for slices in corresponding pictures, quantization parameters, offsets, etc.

[00112] The slice header 717 contains parameters that are specific to each slice in a picture. Hence, there may be one slice header 717 per slice in the video sequence. The slice header 717 may contain slice type information, POCs, reference picture lists, prediction weights, tile entry points, deblocking parameters, etc. It should be noted that in some examples, a bitstream 700 may also include a picture header, which is a syntax structure that contains parameters that apply to all slices in a single picture. For this reason, a picture header and a slice header 717 may be used interchangeably in some contexts. For example, certain parameters may be moved between the slice header 717 and a picture header depending on whether such parameters are common to all slices in a picture.

[00113] The image data 720 contains video data encoded according to inter-prediction and/or intra-prediction as well as corresponding transformed and quantized residual data. For example, the image data 720 may include layers 723, pictures 725, and/or slices 727. A layer 723 is a set of VCL NAL units that share a specified characteristic (e.g., a common resolution, frame rate, image size, etc.) as indicated by a layer ID, such as a nuh_layer_id 729, and associated non-VCL NAL units. For example, a layer 723 may include a set of pictures 725 that share the same nuh_layer_id. A layer 723 may be substantially similar to layers 631, 632, and/or 633. A nuh_layer_id 729 is a syntax element that specifies an identifier of a layer 723 that includes at least one NAL unit. For example, the lowest quality layer 723, known as a base layer, may include the lowest value of nuh_layer_id 729 with increasing values of nuh_layer_id 729 for layers 723 of higher quality. Hence, a lower layer is a layer 723 with a smaller value of nuh_layer_id 729 and a higher layer is a layer 723 with a larger value of nuh_layer_id 729.

[00114] A picture 725 is an array of luma samples and/or an array of chroma samples that create a frame or a field thereof. For example, a picture 725 is a coded image that may be output for display or used to support coding of other picture(s) 725 for output. A picture 725 contains one or more slices 727. A slice 727 may be defined as an integer number of complete tiles or an integer number of consecutive complete coding tree unit (CTU) rows (e.g., within a tile) of a

picture 725 that are exclusively contained in a single NAL unit. The slices 727 are further divided into CTUs and/or coding tree blocks (CTBs). A CTU is a group of samples of a predefined size that can be partitioned by a coding tree. A CTB is a subset of a CTU and contains luma components or chroma components of the CTU. The CTUs /CTBs are further divided into coding blocks based on coding trees. The coding blocks can then be encoded/decoded according to prediction mechanisms.

[00115] A bitstream 700 can be coded as a sequence of NAL units. A NAL unit is a container for video data and/or supporting syntax. A NAL unit can be a VCL NAL unit or a non-VCL NAL unit. A VCL NAL unit is a NAL unit coded to contain video data, such as image data 720 and an associated slice header 717. A non-VCL NAL unit is a NAL unit that contains non-video data such as syntax and/or parameters that support decoding the video data, performance of conformance checking, or other operations. For example, a non-VCL NAL unit can contain a VPS 711, a SPS 713, a PPS 715, a SEI message 719, or other supporting syntax.

[00116] A SEI message 719 is a syntax structure with specified semantics that conveys information that is not needed by the decoding process in order to determine the values of the samples in decoded pictures. For example, the SEI messages 719 may contain data to support HRD processes or other supporting data that is not directly relevant to decoding the bitstream 700 at a decoder. A set of SEI messages 719 may be implemented as a scalable nesting SEI message 741. The scalable nesting SEI message 741 provides a mechanism to associate SEI messages 719 with specific layers 723. A scalable nesting SEI message 741 is a message that contains a plurality of scalable-nested SEI messages 742. A scalable-nested SEI message 742 is an SEI message 719 that corresponds to one or more OLSs or one or more layers 723. An OLS is a set of layers 723 where at least one of the layers 723 is an output layer. Accordingly, a scalable nesting SEI message 741 can be said to include a set of scalable-nested SEI messages 742 or said to include a set of SEI messages 719, depending on context. Further, a scalable nesting SEI message 741 contains a set of scalable-nested SEI messages 742 of the same type. SEI messages 719 may include a BP SEI message that contains HRD parameters for initializing an HRD to manage a CPB for testing corresponding OLSs and/or layers 723. SEI messages 719 may also include a PT SEI message that contains HRD parameters for managing delivery information for AUs at the CPB and/or the DPB for testing corresponding OLSs and/or layers 723. SEI messages 719 may also include a DUI SEI message that contains HRD parameters for managing delivery information for DUs at the CPB and/or the DPB for testing corresponding OLSs and/or layers 723.

[00117] As noted above, a SEI message 719 may be contained in a non-VCL NAL unit. For example, a scalable nesting SEI message 741 may be contained in a SEI NAL unit 743. A SEI NAL unit 743 is a NAL unit with type data indicating that the SEI NAL unit 743 contains SEI data. One of the benefits of a scalable nesting SEI message 741 is that the scalable nesting SEI message 741 allows a set of scalable-nested SEI messages 742 to be contained in a single SEI NAL unit 743. When scalable nesting SEI messages 741 are not employed, each SEI message 719 is contained in a separate SEI NAL unit 743. For clarity of discussion, a particular SEI NAL unit 743 can be referred to as the current SEI NAL unit 743.

[00118] One of the issues with scalable nesting SEI messages 741 is that an entire set of scalable-nested SEI messages 742 are contained in a single SEI NAL unit 743. However, the set of scalable-nested SEI messages 742 can relate to corresponding layers 723 in many ways. For example, each of the scalable-nested SEI messages 742 in a set can relate to all layers 723. In another example, each of the scalable-nested SEI messages 742 in a set can relate to one or more of the layers 723. Various flags can be employed to indicate the correlation between the scalable-nested SEI messages 742 and the layers 723. As noted above, one problem with scalable-nested SEI message 742 flags is that flags indicating the scalable-nested SEI messages 742 apply to all layers in an AU may cause errors when a video sequence includes incomplete AUs. Another problem is that the SEI NAL unit 743 that contains the scalable nesting SEI message 741 is always associated with a layer 723, for example the layer 723 with the lowest nuh_layer_id 729. When a scalable-nested SEI message 742 applies to the layer 723 associated with the SEI NAL unit 743, the association is signaled in both the SEI NAL unit 743 and in the scalable nesting SEI message 741. This is redundant and wastes bits.

[00119] The bitstream 700 includes various flags to signal the configuration of the SEI messages 719 while addressing the problems addressed above. For example, the scalable nesting SEI message 741 may include a scalable nesting all layers flag (all_layers_flag) 735. The scalable nesting all_layers_flag 735 syntax element specifies whether scalable-nested SEI messages 742 in a scalable nesting SEI message 741 apply to all layers 723, for example as specified in the VPS 711. By employing a scalable nesting all_layers_flag 735 that does not address AUs, the incomplete AU problem is avoided. Accordingly, when the scalable nesting all_layers_flag 735 is set, the scalable-nested SEI message 742 contained in the scalable nesting SEI message 741 apply to all layers 723 regardless of the number of layers 723 that are included in the AU to which the scalable nesting SEI message 741 is applied. In a specific example, the scalable nesting all_layers_flag 735 is set equal to one when specifying that the scalable-nested SEI messages 742 apply to all layers 723 that have a nuh_layer_id 729 greater than or equal to

a nuh_layer_id 729 of the current SEI NAL unit 743. Further, the scalable nesting all_layers_flag 735 can be set equal to zero when specifying that the scalable-nested SEI messages 742 might or might not apply to all layers 723 that have a nuh_layer_id 729 greater than or equal to a nuh_layer_id 729 of the current SEI NAL unit 743. Hence, the scalable nesting all_layers_flag 735 increases the functionality of an encoder and/or decoder by causing the bitstream 700 to be correctly checked for conformance when incomplete AUs are present. Further, the scalable nesting all_layers_flag 735 reduces the size of the encoded bitstream 700 by omitting explicit layer signaling for scalable-nested SEI messages 742 when such messages apply to all layers 723. As a result, coding efficiency is increased, which reduces processor, memory, and/or network signaling resource usage at both the encoder and the decoder.

[00120] To address the redundant signaling issue, the scalable nesting SEI message 741 may include a scalable nesting layer id (layer_id[i]) 739. The scalable nesting layer_id[i] 739 is a syntax element that specifies the nuh_layer_id 729 value of an i-th layer to which the scalable-nested SEI messages 742 apply. The scalable nesting layer_id[i] 739 may be employed when the scalable nesting all_layers_flag 735 is set to zero. As such, the layer_id[i] 739 can be used to correlate each of the scalable-nested SEI messages 742 to the corresponding layers 723. Further, the scalable nesting layer_id[i] 739 is configured to signal the nuh_layer_id 729 of all of the layers 723 that correspond to the scalable-nested SEI messages 742 except for the nuh_layer_id 729 of the layer 723 that corresponds to the SEI NAL unit 743. When a scalable-nested SEI message 742 is not associated with a nuh_layer_id 729 value in the scalable nesting layer_id[i] 739, the scalable-nested SEI message 742 can be inferred to be related to the layer 723 associated with the SEI NAL unit 743. For example, the SEI NAL unit 743 can be associated with the lowest value of nuh_layer_id 729. Hence, the scalable nesting layer_id[i] 739 may be constrained such that the value of the scalable nesting layer_id[i] 739 for each layer i must be greater than the nuh_layer_id 729 of the current SEI NAL unit 743. This ensures that the nuh_layer_id 729 of the current SEI NAL unit 743 is signaled in the current SEI NAL unit 743, but not repeated in the scalable nesting layer_id[i] 739 in the scalable nesting SEI message 741. This improves the scalable nesting SEI message 741 by omitting a redundant Id. For example, a loop that encodes/decodes the scalable nesting layer_id[i] 739 executes one less time, which decreases processor resources during encoding and/or decoding. Further, this reduces the size of the encoded bitstream 700 for each of the scalable nesting SEI messages 741 in the bitstream 700. As a result, coding efficiency is increased, which reduces processor, memory, and/or network signaling resource usage at both the encoder and the decoder.

[00121] A HRD and/or a decoder can use the scalable nesting layer_id[i] 739 and the scalable nesting all_layers_flag 735 to correlate the scalable-nested SEI messages 742 to the layers 723, for example based on other syntax elements in the bitstream 700. For example, the HRD and/or decoder can use a number of nesting layers (nestingNumLayers) variable and a list of nesting layer Ids (NestingLayerId[i]) variable. As a matter of nomenclature, syntax elements with underscores may be signaled in the bitstream 700 by the encoder, while syntax elements without underscores or spaces may be variables that are determined when reading data from the bitstream 700 (e.g., at the HRD and/or decoder). The nestingNumLayers variable specifies a number of layers 723 to which the scalable-nested SEI messages 742 apply based on data in a bitstream 700. The NestingLayerId[i] variable is a variable that is set to specify a list of nuh_layer_id values of the layers 723 to which the scalable-nested SEI messages 742 apply for each layer i in a range of zero to nestingNumLayers minus one, inclusive, based on data in a bitstream 700. The nestingNumLayers and the NestingLayerId[i] may be determined as follows:

```
if( scalable nesting all_layers_flag ) {
   nestingNumLayers = vps_max_layers_minus1 + 1 −  GeneralLayerIdx[ nuh_layer_id ]
   for( i = 0; i < nestingNumLayers; i ++)
         NestingLayerId[ i ] = vps_layer_id[ GeneralLayerIdx[ nuh_layer_id ] + i ]
} else {
    nestingNumLayers = scalable nesting num_layers_minus1 + 1
    for( i = 0; i < nestingNumLayers; i ++)
         NestingLayerId[ i ] = ( i  = = 0 ) ? nuh_layer_id : scalable nesting layer_id[ i ]
}
```

[00122] The preceding code relies on a VPS maximum layers minus one (vps_max_layers_minus1) 731. The vps_max_layers_minus1 731 is a syntax element carried in the VPS 711 that specifies a number of layers 723 in the bitstream 700. The preceding code also relies on a VPS layer Id (vps_layer_id[i]) 733. The vps_layer_id[i] 733 is a syntax element carried in the VPS 733 that specifies a nuh_layer_id 729 value of an i-th layer. Hence, the vps_layer_id[i] 733 specifies the nuh_layer_id 729 for every layer 723 such that the nuh_layer_ids 729 can be accessed by a layer index i.

[00123] The preceding code also relies on a general layer index (GeneralLayerIdx) variable. The GeneralLayerIdx is a variable determined by the HRD and/or decoder that specifies a layer index of a layer 723 with nuh_layer_id equal to vps_layer_id[i] 733. The preceding code also relies on a scalable nesting number of layers minus one (num_layers_minus1) 737 syntax element. The scalable nesting num_layers_minus1 737 is a syntax element carried in the

scalable nesting SEI message 741 that specifies the number of layers 723 to which the scalable-nested SEI messages 742 in the scalable nesting SEI message 741 apply. The scalable nesting num_layers_minus1 737 employs minus one format and hence includes one less than the actual value. For example, if a scalable nesting SEI message 741 includes scalable-nested SEI messages 742 that relate to five layers 723, then the scalable nesting num_layers_minus1 737 is set to a value of four. In some examples, the scalable nesting num_layers_minus1 737 is constrained to be in a range of zero to vps_max_layers_minus1 731 minus GeneralLayerIdx[nuh_layer_id], inclusive, where nuh_layer_id 729 is the nuh_layer_id 729 of the current SEI NAL unit 743.

[00124] The preceding information is now described in more detail herein below. Layered video coding is also referred to as scalable video coding or video coding with scalability. Scalability in video coding may be supported by using multi-layer coding techniques. A multi-layer bitstream comprises a base layer (BL) and one or more enhancement layers (ELs). Example of scalabilities includes spatial scalability, quality / signal to noise ratio (SNR) scalability, multi-view scalability, frame rate scalability, etc. When a multi-layer coding technique is used, a picture or a part thereof may be coded without using a reference picture (intra-prediction), may be coded by referencing reference pictures that are in the same layer (inter-prediction), and/or may be coded by referencing reference pictures that are in other layer(s) (inter-layer prediction). A reference picture used for inter-layer prediction of the current picture is referred to as an inter-layer reference picture (ILRP). FIG. 6 illustrates an example of multi-layer coding for spatial scalability in which pictures in different layers have different resolutions.

[00125] Some video coding families provide support for scalability in separated profile(s) from the profile(s) for single-layer coding. Scalable video coding (SVC) is a scalable extension of the advanced video coding (AVC) that provides support for spatial, temporal, and quality scalabilities. For SVC, a flag is signaled in each macroblock (MB) in EL pictures to indicate whether the EL MB is predicted using the collocated block from a lower layer. The prediction from the collocated block may include texture, motion vectors, and/or coding modes. Implementations of SVC may not directly reuse unmodified AVC implementations in their design. The SVC EL macroblock syntax and decoding process differs from the AVC syntax and decoding process.

[00126] Scalable HEVC (SHVC) is an extension of HEVC that provides support for spatial and quality scalabilities. Multiview HEVC (MV-HEVC) is an extension of HEVC that provides support for multi-view scalability. 3D HEVC (3D-HEVC) is an extension of HEVC that

provides support for 3D video coding that is more advanced and more efficient than MV-HEVC. Temporal scalability may be included as an integral part of a single-layer HEVC codec. In the multi-layer extension of HEVC, decoded pictures used for inter-layer prediction come only from the same AU and are treated as long-term reference pictures (LTRPs). Such pictures are assigned reference indices in the reference picture list(s) along with other temporal reference pictures in the current layer. Inter-layer prediction (ILP) is achieved at the prediction unit (PU) level by setting the value of the reference index to refer to the inter-layer reference picture(s) in the reference picture list(s). Spatial scalability resamples a reference picture or part thereof when an ILRP has a different spatial resolution than the current picture being encoded or decoded. Reference picture resampling can be realized at either picture level or coding block level.

[00127] VVC may also support layered video coding. A VVC bitstream can include multiple layers. The layers can be all independent from each other. For example, each layer can be coded without using inter-layer prediction. In this case, the layers are also referred to as simulcast layers. In some cases, some of the layers are coded using ILP. A flag in the VPS can indicate whether the layers are simulcast layers or whether some layers use ILP. When some layers use ILP, the layer dependency relationship among layers is also signaled in the VPS. Unlike SHVC and MV-HEVC, VVC may not specify OLSs. An OLS includes a specified set of layers, where one or more layers in the set of layers are specified to be output layers. An output layer is a layer of an OLS that is output. In some implementations of VVC, only one layer may be selected for decoding and output when the layers are simulcast layers. In some implementations of VVC, the entire bitstream including all layers is specified to be decoded when any layer uses ILP. Further, certain layers among the layers are specified to be output layers. The output layers may be indicated to be only the highest layer, all the layers, or the highest layer plus a set of indicated lower layers.

[00128] The preceding aspects contain certain problems. Some video coding systems specify that scalable nesting SEI messages for associating SEI messages with bitstream subsets correspond to various operation points, with specific layers, and/or sub-layers. A first problem occurs when nesting SEI messages for specific layers using the scalable nesting SEI message contained in an SEI NAL unit with a particular nuh_layer_id value of nuhLayerId. In this case, the applicable layer with nuh_layer_id equal to nuhLayerId is also explicitly signaled when the nested SEI messages do not apply to all layers. However, the applicable layer with nuh_layer_id equal to nuhLayerId is always an applicable layer. Thus the signaling of the nuh_layer_id of the SEI NAL unit is not necessary (e.g., redundant) and is a waste of bits.

[00129] A second problem occurs when the nested SEI messages apply to all layers. The semantics may state that the all_layers_flag is set equal to one to specify that the list nestingLayerIdList[0] includes all values of nuh_layer_id present in the current access unit that are greater than or equal to nuh_layer_id of the current SEI NAL unit, in increasing order of the values. However, the current AU may be an incomplete AU. This may occur when an AU does not have pictures present for all the layers. The persistent scope of a nested SEI message can include other AUs that contain pictures for more layers than the current AU. Consequently, layers that are missing in the current AU (the AU containing the scalable nesting SEI message) are not specified to be applicable. When some other AUs in the persistent scope of a nested SEI message do have pictures of the layers that are missing in the current AU, information carried in the nested SEI message is interpreted as not applicable to those pictures. Consequently, errors may occur. For example, if a frame packing SEI message is included in a nested SEI message, the rendered pictures may include artifacts resulting in a bad user experience.

[00130] In general, this disclosure describes approaches for scalable nesting of SEI messages for layers in multi-layer video bitstreams. The descriptions of the techniques are based on VVC. However, the techniques also apply to layered video coding based on other video codec specifications.

[00131] One or more of the abovementioned problems may be solved as follows. Specifically, this disclosure includes methods for efficiently nesting SEI messages for use with layers in conjunction with incomplete AUs as described above. First, the layer having layer Id equal to that of the SEI NAL unit containing the scalable nesting SEI message is inferred to be a layer to which the nested SEI messages apply instead of being explicitly signaled. Second, when the nested SEI messages apply to all layers, the nested SEI messages apply to all layers specified by the VPS, which are all the layers that may be present in the bitstream, instead of all layers present in the current AU.

[00132] An example implementation of the preceding mechanisms is as follows. An example scalable nesting SEI message syntax is as follows.

scalable_nesting(payloadSize) {	Descriptor
nesting_ols_flag	u(1)
if(nesting_ols_flag) {	
nesting_num_olss_minus1	ue(v)
for(i = 0; i <= nesting_num_olss_minus1; i++) {	

nesting_ols_idx_delta_minus1[i]	ue(v)
if(NumLayersInOls[NestingOlsIdx[i]] > 1) {	
nesting_num_ols_layers_minus1[i]	ue(v)
for(j = 0; j <= nesting_num_ols_layers_minus1[i]; j++)	
nesting_ols_layer_idx_delta_minus1[i][j]	ue(v)
}	
}	
} else {	
nesting_all_layers_flag	u(1)
if(!nesting_all_layers_flag) {	
nesting_num_layers_minus1	ue(v)
for(i = 1; i <= nesting_num_layers_minus1; i++)	
nesting_layer_id[i]	u(6)
}	
}	
nesting_num_seis_minus1	ue(v)
while(!byte_aligned( ))	
nesting_zero_bit /* equal to 0 */	u(1)
for(i = 0; i <= nesting_num_seis_minus1; i++)	
sei_message( )	
}	

[00133] In an alternative example, the syntax element nesting_layer_id[i] is a ue(v) coding of the layer index among layers specified by the VPS. In another alternative example, the syntax element nesting_layer_id[i] is a ue(v) coding of the delta of the layer index among layers specified by the VPS. In another alternative example, the syntax element nesting_layer_id[i] is a ue(v) coding of the delta of nuh_layer_id.

[00134] An example scalable nesting SEI message semantics is as follows. A scalable nesting SEI message provides a mechanism to associate SEI messages with specific layers in the context of specific OLSs or with specific layers not in the context of an OLS. A scalable

nesting SEI message contains one or more SEI messages. The SEI messages contained in the scalable nesting SEI message are also referred to as the scalable-nested SEI messages. Bitstream conformance may require that the following restrictions apply when SEI messages are contained in a scalable nesting SEI message.

[00135] An SEI message that has payloadType equal to one hundred thirty two (decoded picture hash) or one hundred thirty three (scalable nesting) should not be contained in a scalable nesting SEI message. When a scalable nesting SEI message contains a buffering period, picture timing, or decoding unit information SEI message, the scalable nesting SEI message should not contain any other SEI message with payloadType not equal to zero (buffering period), one (picture timing), or one hundred thirty (decoding unit information).

[00136] Bitstream conformance may also require that the following restrictions apply on the value of the nal_unit_type of the SEI NAL unit containing a scalable nesting SEI message. When a scalable nesting SEI message contains an SEI message that has payloadType equal to zero (buffering period), one (picture timing), one hundred thirty (decoding unit information), one hundred forty five (dependent RAP indication), or one hundred sixty eight (frame-field information), the SEI NAL unit containing the scalable nesting SEI message should have a nal_unit_type set equal to PREFIX_SEI_NUT. When a scalable nesting SEI message contains an SEI message that has payloadType equal to one hundred thirty two (decoded picture hash), the SEI NAL unit containing the scalable nesting SEI message should have a nal_unit_type set equal to SUFFIX_SEI_NUT.

[00137] A nesting_ols_flag may be set equal to one to specify that the scalable-nested SEI messages apply to specific layers in the context of specific OLSs. The nesting_ols_flag may be set equal to zero to specify that that the scalable-nested SEI messages generally apply (e.g., not in the context of an OLS) to specific layers.

[00138] Bitstream conformance may require that the following restrictions are applied to the value of nesting_ols_flag. When the scalable nesting SEI message contains an SEI message that has payloadType equal to zero (buffering period), one (picture timing), or one hundred thirty (decoding unit information), the value of nesting_ols_flag should be equal to one. When the scalable nesting SEI message contains an SEI message that has payloadType equal to a value in VclAssociatedSeiList, the value of nesting_ols_flag should be equal to zero.

[00139] A nesting_num_olss_minus1 plus one specifies the number of OLSs to which the scalable-nested SEI messages apply. The value of nesting_num_olss_minus1 should be in the range of zero to TotalNumOlss − 1, inclusive. The nesting_ols_idx_delta_minus1[i] is used to derive the variable NestingOlsIdx[i] that specifies the OLS index of the i-th OLS to which the

scalable-nested SEI messages apply when nesting_ols_flag is equal to one. The value of nesting_ols_idx_delta_minus1[i] should be in the range of zero to TotalNumOlss − 2, inclusive. The variable NestingOlsIdx[i] may be derived as follows:

```
if( i == 0 )
    NestingOlsIdx[ i ] = nesting_ols_idx_delta_minus1[ i ]
else
    NestingOlsIdx[ i ] = NestingOlsIdx[ i − 1 ] + nesting_ols_idx_delta_minus1[ i ] + 1
```

[00140] The nesting_num_ols_layers_minus1[i] plus one specifies the number of layers to which the scalable-nested SEI messages apply in the context of the NestingOlsIdx[i]-th OLS. The value of nesting_num_ols_layers_minus1[i] should be in the range of zero to NumLayersInOls[NestingOlsIdx[i]] − 1, inclusive.

[00141] The nesting_ols_layer_idx_delta_minus1[i][j] is used to derive the variable NestingOlsLayerIdx[i][j] that specifies the OLS layer index of the j-th layer to which the scalable-nested SEI messages apply in the context of the NestingOlsIdx[i]-th OLS when nesting_ols_flag is equal to one. The value of nesting_ols_layer_idx_delta_minus1[i] should be in the range of zero to NumLayersInOls[nestingOlsIdx[i]] − two, inclusive.

[00142] The variable NestingOlsLayerIdx[i][j] may be derived as follows:

```
if( j == 0 )
    NestingOlsLayerIdx[ i ][ j ] = nesting_ols_layer_idx_delta_minus1[ i ][ j ]
else
    NestingOlsLayerIdx[ i ][ j ] = NestingOlsLayerIdx[ i ][ j − 1 ] +
        nesting_ols_layer_idx_delta_minus1[ i ][ j ] + 1
```

[00143] The lowest value among all values of LayerIdInOls[NestingOlsIdx[i]][NestingOlsLayerIdx[i][0]] for i in the range of zero to nesting_num_olss_minus1, inclusive, should be equal to nuh_layer_id of the current SEI NAL unit (e.g., the SEI NAL unit containing the scalable nesting SEI message). The nesting_all_layers_flag may be set equal to one to specify that the scalable-nested SEI messages generally apply to all layers that have nuh_layer_id greater than or equal to the nuh_layer_id of the current SEI NAL unit. The nesting_all_layers_flag may be set equal to zero to specify that the scalable-nested SEI messages may or may not generally apply to all layers that have nuh_layer_id greater than or equal to the nuh_layer_id of the current SEI NAL unit.

[00144] The nesting_num_layers_minus1 plus one specifies the number of layers to which the scalable-nested SEI messages generally apply. The value of nesting_num_layers_minus1 should be in the range of zero to vps_max_layers_minus1 − GeneralLayerIdx[nuh_layer_id],

inclusive, where nuh_layer_id is the nuh_layer_id of the current SEI NAL unit. The nesting_layer_id[i] specifies the nuh_layer_id value of the i-th layer to which the scalable-nested SEI messages generally apply when nesting_all_layers_flag is equal to zero. The value of nesting_layer_id[i] should be greater than nuh_layer_id, where nuh_layer_id is the nuh_layer_id of the current SEI NAL unit.

[00145] When the nesting_ols_flag is equal to one, the variable NestingNumLayers, specifying the number of layer to which the scalable-nested SEI messages generally apply, and the list NestingLayerId[i] for i in the range of zero to NestingNumLayers − 1, inclusive, specifying the list of nuh_layer_id value of the layers to which the scalable-nested SEI messages generally apply, are derived as follows, where nuh_layer_id is the nuh_layer_id of the current SEI NAL unit:

```
if( nesting_all_layers_flag ) {
        NestingNumLayers =
vps_max_layers_minus1 + 1 − GeneralLayerIdx[ nuh_layer_id ]
        for( i = 0; i < NestingNumLayers; i ++)
            NestingLayerId[ i ] = vps_layer_id[ GeneralLayerIdx[ nuh_layer_id ] + i ]     (D-2)
} else {
        NestingNumLayers = nesting_num_layers_minus1 + 1
        for( i = 0; i < NestingNumLayers; i ++)
            NestingLayerId[ i ] = ( i  = = 0 ) ? nuh_layer_id : nesting_layer_id[ i ]
}
```

[00146] The nesting_num_seis_minus1 plus one specifies the number of scalable-nested SEI messages. The value of nesting_num_seis_minus1 should be in the range of zero to sixty three, inclusive. The nesting_zero_bit should be set equal to zero.

[00147] FIG. 8 is a schematic diagram of an example video coding device 800. The video coding device 800 is suitable for implementing the disclosed examples/embodiments as described herein. The video coding device 800 comprises downstream ports 820, upstream ports 850, and/or transceiver units (Tx/Rx) 810, including transmitters and/or receivers for communicating data upstream and/or downstream over a network. The video coding device 800 also includes a processor 830 including a logic unit and/or central processing unit (CPU) to process the data and a memory 832 for storing the data. The video coding device 800 may also comprise electrical, optical-to-electrical (OE) components, electrical-to-optical (EO) components, and/or wireless communication components coupled to the upstream ports 850 and/or downstream ports 820 for communication of data via electrical, optical, or wireless

communication networks. The video coding device 800 may also include input and/or output (I/O) devices 860 for communicating data to and from a user. The I/O devices 860 may include output devices such as a display for displaying video data, speakers for outputting audio data, etc. The I/O devices 860 may also include input devices, such as a keyboard, mouse, trackball, etc., and/or corresponding interfaces for interacting with such output devices.

[00148] The processor 830 is implemented by hardware and software. The processor 830 may be implemented as one or more CPU chips, cores (e.g., as a multi-core processor), field-programmable gate arrays (FPGAs), application specific integrated circuits (ASICs), and digital signal processors (DSPs). The processor 830 is in communication with the downstream ports 820, Tx/Rx 810, upstream ports 850, and memory 832. The processor 830 comprises a coding module 814. The coding module 814 implements the disclosed embodiments described herein, such as methods 100, 900, and 1000, which may employ a multi-layer video sequence 600 and/or a bitstream 700. The coding module 814 may also implement any other method/mechanism described herein. Further, the coding module 814 may implement a codec system 200, an encoder 300, a decoder 400, and/or a HRD 500. For example, the coding module 814 may be employed to implement a HRD. Further, the coding module 814 may be employed to encode scalable nesting SEI messages with corresponding flags to support clear and concise signaling of the correlation between the scalable-nested SEI messages in the scalable nesting SEI messages and corresponding layers. Accordingly, the coding module 814 may be configured to perform mechanisms to address one or more of the problems discussed above. Hence, coding module 814 causes the video coding device 800 to provide additional functionality and/or coding efficiency when coding video data. As such, the coding module 814 improves the functionality of the video coding device 800 as well as addresses problems that are specific to the video coding arts. Further, the coding module 814 effects a transformation of the video coding device 800 to a different state. Alternatively, the coding module 814 can be implemented as instructions stored in the memory 832 and executed by the processor 830 (e.g., as a computer program product stored on a non-transitory medium).

[00149] The memory 832 comprises one or more memory types such as disks, tape drives, solid-state drives, read only memory (ROM), random access memory (RAM), flash memory, ternary content-addressable memory (TCAM), static random-access memory (SRAM), etc. The memory 832 may be used as an over-flow data storage device, to store programs when such programs are selected for execution, and to store instructions and data that are read during program execution.

[00150] FIG. 9 is a flowchart of an example method 900 of encoding a video sequence into a bitstream, such as bitstream 700, including scalable nesting SEI messages that apply to layers. Method 900 may be employed by an encoder, such as a codec system 200, an encoder 300, and/or a video coding device 800 when performing method 100. Further, the method 900 may operate on a HRD 500 and hence may perform conformance tests on a multi-layer video sequence 600.

[00151] Method 900 may begin when an encoder receives a video sequence and determines to encode that video sequence into a multi-layer bitstream, for example based on user input. At step 901, the encoder encodes the video sequence into one or more layers and encodes the layers into a multi-layer bitstream. Hence, the bitstream comprises one or more layers. A layer may include a set of VCL NAL units with the same layer Id and associated non-VCL NAL units. For example, a layer may include a set of VCL NAL units that contain video data of encoded pictures as well as any parameter sets used to code such pictures. One or more of the layers may be output layers. Layers that are not an output layer are encoded to support reconstructing the output layer(s), but such supporting layers are not intended for output at a decoder. In this way, the encoder can encode various combinations of layers for transmission to a decoder upon request. The layer can be transmitted as desired to allow the decoder to obtain a different representation of the video sequence depending on network conditions, hardware capabilities, and/or user settings.

[00152] At step 903, the encoder encodes one or more scalable nesting SEI messages into the bitstream in a current SEI NAL unit. A SEI message is a syntax structure that contains data that is not used for decoding. For example, SEI messages may contain data to support conformance testing to ensure the bitstream conforms to standards. To support simplified signaling when used in conjunction with a multi-layer bitstream, the SEI messages are encoded as scalable nesting SEI messages. A scalable nesting SEI message contains one or more scalable-nested SEI messages. The scalable-nested SEI messages may each apply to one or more of OLSs and/or one or more of the layers. In order to support simplified signaling, a scalable nesting SEI message contains one or more scalable nesting layer Ids that specify layer Id values of the layers to which the scalable-nested SEI messages apply except a layer Id value of the current SEI NAL unit. This exception prevents redundant signaling of the layer Id of the current SEI NAL unit as the current SEI NAL unit also contains a corresponding layer Id. It should be noted that the scalable nesting SEI message omits the layer Id of the current SEI NAL unit even when the scalable nesting SEI message includes at least one scalable-nested SEI message that applies to the layer of the current SEI NAL unit.

[00153] In some examples, the scalable nesting layer Ids are a scalable nesting layer_id[i] syntax element that specifies a nuh_layer_id value of an i-th layer to which the scalable-nested SEI messages apply when a scalable nesting all_layers_flag is equal to zero. In some examples, each value of the scalable nesting layer_id[i] is constrained to be greater than a nuh_layer_id value of the current SEI NAL unit. It should be noted that syntax elements described herein as including underscores may be included in the bitstream, while syntax elements without spaces may be derived. The preceding constraints and/or requirements ensure that the bitstream conforms with, for example, VVC or some other standard, modified as indicated herein. However, the encoder may also be capable of operating in other modes where it is not so constrained, such as when operating under a different standard or a different version of the same standard.

[00154] At step 905, a HRD operating at the encoder can perform a set of bitstream conformance tests on the layers based on the scalable nesting SEI message. For example, the HRD can read the flags in a scalable nesting SEI message to determine how to interpret the scalable-nested SEI messages contained in the scalable nesting SEI message. The HRD can then read the scalable-nested SEI messages to determine how to check layers for conformance to standards. The HRD can then perform conformance tests on the layers based on the scalable-nested SEI messages and/or corresponding flags in the scalable nesting SEI message.

[00155] In some examples, the HRD can employ a nestingNumLayers variable and a NestingLayerId[i] variable to determine the correlations between the scalable-nested SEI messages and the layers. The nestingNumLayers variable specifies a number of layers to which the scalable-nested SEI messages apply. The NestingLayerId[i] variable specifies a list of nuh_layer_id values of the layers to which the scalable-nested SEI messages apply for i in a range of zero to nestingNumLayers minus one, inclusive. The nestingNumLayers and the NestingLayerId[i] can be derived by a HRD based on data from the bitstream. The HRD can then employ the nestingNumLayers variable and a NestingLayerId[i] variable as desired to support conformance testing. In some examples, the nestingNumLayers and the NestingLayerId[i] are derived as follows:

```
if( scalable nesting all_layers_flag ) {
   nestingNumLayers = vps_max_layers_minus1 + 1 −  GeneralLayerIdx[ nuh_layer_id ]
   for( i = 0; i < nestingNumLayers; i ++)
         NestingLayerId[ i ] = vps_layer_id[ GeneralLayerIdx[ nuh_layer_id ] + i ]
} else {
    nestingNumLayers = scalable nesting num_layers_minus1 + 1
    for( i = 0; i < nestingNumLayers; i ++)
         NestingLayerId[ i ] = ( i  = = 0 ) ? nuh_layer_id : scalable nesting layer_id[ i ]
}
```

where the scalable nesting all_layers_flag specifies whether the scalable-nested SEI messages apply to all layers, vps_max_layers_minus1 plus one specifies a number of layers specified by a VPS, GeneralLayerIdx is a variable specifying a layer index of a layer with nuh_layer_id equal to vps_layer_id[i], vps_layer_id[i] specifies a nuh_layer_id value of an i-th layer, and the scalable nesting num_layers_minus1 plus one specifies a number of layers to which the scalable-nested SEI messages apply. In some examples, the scalable nesting num_layers_minus1 is constrained to remain in a range of zero to vps_max_layers_minus1− GeneralLayerIdx[nuh_layer_id], inclusive, where nuh_layer_id is the nuh_layer_id of the current SEI NAL unit. It should be noted that syntax elements described herein as including underscores may be included in the bitstream, while syntax elements without spaces may be derived.

[00156] At step 907, the encoder can store the bitstream for communication toward a decoder upon request. The encoder can also transmit the bitstream toward the encoder as desired.

[00157] FIG. 10 is a flowchart of an example method 1000 of decoding a video sequence from a bitstream, such as bitstream 700, including scalable nesting SEI messages that apply to layers. Method 1000 may be employed by a decoder, such as a codec system 200, a decoder 400, and/or a video coding device 800 when performing method 100. Further, method 1000 may be employed on a multi-layer video sequence 600 that has been checked for conformance by a HRD, such as HRD 500.

[00158] Method 1000 may begin when a decoder begins receiving a bitstream of coded data representing a multi-layer video sequence, for example as a result of method 900. At step 1001, the decoder receives a bitstream comprising one or more layers. A layer may include a set of VCL NAL units with the same layer Id and associated non-VCL NAL units. For example, a layer may include a set of VCL NAL units that contain video data of encoded pictures as well as any parameter sets used to code such pictures. One or more of the layers may be output

layers. Layers that are not an output layer are decoded to support decoding the output layer(s), but such supporting layers are not output.

[00159] The bitstream also includes one or more scalable nesting SEI messages. A SEI message is a syntax structure that contains data that is not used for decoding. For example, SEI messages may contain data to support conformance testing to ensure the bitstream conforms to standards. To support simplified signaling when used in conjunction with a multi-layer bitstream, the SEI messages are coded in scalable nesting SEI messages. A scalable nesting SEI message includes one or more scalable-nested SEI messages. A specified scalable nesting SEI message may be included in a current SEI NAL unit, which is a non-VCL NAL unit that contains SEI message(s). The scalable-nested SEI messages may each apply to one or more of OLSs and/or one or more of the layers. In order to support simplified signaling, a scalable nesting SEI message contains one or more scalable nesting layer Ids that specify layer Id values of the layers to which the scalable-nested SEI messages apply except a layer Id value of the current SEI NAL unit. This exception prevents redundant signaling of the layer Id of the current SEI NAL unit as the current SEI NAL unit also contains a corresponding layer Id. It should be noted that the scalable nesting SEI message omits the layer Id of the current SEI NAL unit even when the scalable nesting SEI message includes at least one scalable-nested SEI message that applies to the layer of the current SEI NAL unit.

[00160] In some examples, the scalable nesting layer Ids are a scalable nesting layer_id[i] syntax element that specifies a nuh_layer_id value of an i-th layer to which the scalable-nested SEI messages apply when a scalable nesting all_layers_flag is equal to zero. In some examples, each value of the scalable nesting layer_id[i] is constrained to be greater than a nuh_layer_id value of the current SEI NAL unit.

[00161] In some examples, the decoder can employ a nestingNumLayers variable and a NestingLayerId[i] variable to determine the correlations between the scalable-nested SEI messages and the layers. The nestingNumLayers variable specifies a number of layers to which the scalable-nested SEI messages apply. The NestingLayerId[i] variable specifies a list of nuh_layer_id values of the layers to which the scalable-nested SEI messages apply for i in a range of zero to nestingNumLayers minus one, inclusive. The nestingNumLayers and the NestingLayerId[i] can be derived by a decoder based on data from the bitstream. The decoder can then employ the nestingNumLayers variable and a NestingLayerId[i] variable as desired to support decoding and/or display. In some examples, the nestingNumLayers and the NestingLayerId[i] are derived as follows:

```
if( scalable nesting all_layers_flag ) {
   nestingNumLayers = vps_max_layers_minus1 + 1 −  GeneralLayerIdx[ nuh_layer_id ]
   for( i = 0; i < nestingNumLayers; i ++)
         NestingLayerId[ i ] = vps_layer_id[ GeneralLayerIdx[ nuh_layer_id ] + i ]
} else {
    nestingNumLayers = scalable nesting num_layers_minus1 + 1
    for( i = 0; i < nestingNumLayers; i ++)
         NestingLayerId[ i ] = ( i  = = 0 ) ? nuh_layer_id : scalable nesting layer_id[ i ]
}
```

where the scalable nesting all_layers_flag specifies whether the scalable-nested SEI messages apply to all layers, vps_max_layers_minus1 plus one specifies a number of layers specified by a VPS, GeneralLayerIdx is a variable specifying a layer index of a layer with nuh_layer_id equal to vps_layer_id[i], vps_layer_id[i] specifies a nuh_layer_id value of an i-th layer, and the scalable nesting num_layers_minus1 plus one specifies a number of layers to which the scalable-nested SEI messages apply. In some examples, the scalable nesting num_layers_minus1 is constrained to remain in a range of zero to vps_max_layers_minus1− GeneralLayerIdx[nuh_layer_id], inclusive, where nuh_layer_id is the nuh_layer_id of the current SEI NAL unit.

[00162] In an embodiment, the video decoder expects the scalable nesting SEI message to omit the layer Id of the current SEI NAL unit as described above based on VVC or some other standard. If, however, the decoder determines that this condition is not true, the decoder may detect an error, signal an error, request that a revised bitstream (or a portion thereof) be resent, or take some other corrective measures to ensure that a conforming bitstream is received.

[00163] At step 1003, the decoder can decode a coded picture from the one or more layers to produce a decoded picture. For example, the presence of the scalable nesting SEI message can indicate that the bitstream has been checked by a HRD at the encoder and hence conforms to standards. Accordingly, the presence of the scalable nesting SEI message indicates the bitstream can be decoded. At step 1005, the decoder can forward the decoded picture for display as part of a decoded video sequence. For example, the decoded picture and/or video sequence can be displayed to a user on the display or screen of an electronic device (e.g., a smart phone, tablet, laptop, personal computer, etc.).

[00164] FIG. 11 is a schematic diagram of an example system 1100 for coding a video sequence using a bitstream including scalable nesting SEI messages that apply to layers. System 1100 may be implemented by an encoder and a decoder such as a codec system 200, an encoder

300, a decoder 400, and/or a video coding device 800. Further, the system 1100 may employ a HRD 500 to perform conformance tests on a multi-layer video sequence 600 and/or a bitstream 700. In addition, system 1100 may be employed when implementing method 100, 900, and/or 1000.

[00165] The system 1100 includes a video encoder 1102. The video encoder 1102 comprises an encoding module 1103 for encoding a bitstream comprising one or more layers. The encoding module 1103 is further for encoding into the bitstream a scalable nesting SEI message in a current SEI NAL unit, wherein the scalable nesting SEI message contains one or more scalable-nested SEI messages and one or more scalable nesting layer Id that specify layer Id values of the layers to which the scalable-nested SEI messages apply except a layer Id value of the current SEI NAL unit. The video encoder 1102 further comprises a HRD module 1105 for performing a set of bitstream conformance tests on the layers based on the scalable-nested SEI messages. The video encoder 1102 further comprises a storing module 1106 for storing the bitstream for communication toward a decoder. The video encoder 1102 further comprises a transmitting module 1107 for transmitting the bitstream toward a video decoder 1110. The video encoder 1102 may be further configured to perform any of the steps of method 900.

[00166] The system 1100 also includes a video decoder 1110. The video decoder 1110 comprises a receiving module 1111 for receiving a bitstream comprising one or more layers and a scalable nesting SEI message in a current SEI NAL unit, wherein the scalable nesting SEI message contains one or more scalable-nested SEI messages and one or more scalable nesting layer Id that specify layer Id values of the layers to which the scalable-nested SEI messages apply except a layer Id value of the current SEI NAL unit. The video decoder 1110 further comprises a decoding module 1113 for decoding a coded picture from the one or more layers to produce a decoded picture. The video decoder 1110 further comprises a forwarding module 1115 for forwarding the decoded picture for display as part of a decoded video sequence. The video decoder 1110 may be further configured to perform any of the steps of method 1000.

[00167] A first component is directly coupled to a second component when there are no intervening components, except for a line, a trace, or another medium between the first component and the second component. The first component is indirectly coupled to the second component when there are intervening components other than a line, a trace, or another medium between the first component and the second component. The term “coupled” and its variants include both directly coupled and indirectly coupled. The use of the term “about” means a range including ±10% of the subsequent number unless otherwise stated.

[00168] It should also be understood that the steps of the exemplary methods set forth herein are not necessarily required to be performed in the order described, and the order of the steps of such methods should be understood to be merely exemplary. Likewise, additional steps may be included in such methods, and certain steps may be omitted or combined, in methods consistent with various embodiments of the present disclosure.

[00169] While several embodiments have been provided in the present disclosure, it may be understood that the disclosed systems and methods might be embodied in many other specific forms without departing from the spirit or scope of the present disclosure. The present examples are to be considered as illustrative and not restrictive, and the intention is not to be limited to the details given herein. For example, the various elements or components may be combined or integrated in another system or certain features may be omitted, or not implemented.

[00170] In addition, techniques, systems, subsystems, and methods described and illustrated in the various embodiments as discrete or separate may be combined or integrated with other systems, components, techniques, or methods without departing from the scope of the present disclosure. Other examples of changes, substitutions, and alterations are ascertainable by one skilled in the art and may be made without departing from the spirit and scope disclosed herein.

CLAIMS

1. A method implemented in a decoder, the method comprising:

receiving, by a receiver of the decoder, a bitstream comprising one or more layers and a scalable nesting supplemental enhancement information (SEI) message in a current SEI network abstraction layer (NAL) unit, wherein the scalable nesting SEI message contains one or more scalable-nested SEI messages and one or more scalable nesting layer identifiers (Ids) that specify layer Id values of the layers to which the scalable-nested SEI messages apply;

decoding, by a processor of the decoder, a coded picture from the one or more layers to produce a decoded picture;

forwarding, by the processor, the decoded picture for display as part of a decoded video sequence;

deriving, by the processor, a number of nesting layers (nestingNumLayers) variable that specifies a number of layers to which the scalable-nested SEI messages apply; and

deriving, by the processor, a list of nesting layer Ids (NestingLayerId[i]) that specifies a list of nuh_layer_id values of the layers to which the scalable-nested SEI messages apply for i in a range of zero to nestingNumLayers minus one, inclusive;

wherein the scalable nesting layer Ids are a scalable nesting layer_id[i] syntax element that specifies a NAL unit header layer Id (nuh_layer_id) value of an i-th layer to which the scalable-nested SEI messages apply when a scalable nesting all layers flag (all_layers_flag) is equal to zero,

wherein the nestingNumLayers and the NestingLayerId[i] are derived as follows:

```
if( scalable nesting all_layers_flag ) {
   nestingNumLayers = vps_max_layers_minus1 + 1 −  GeneralLayerIdx[ nuh_layer_id ]
   for( i = 0; i < nestingNumLayers; i ++)
        NestingLayerId[ i ] = vps_layer_id[ GeneralLayerIdx[ nuh_layer_id ] + i ]
} else {
    nestingNumLayers = scalable nesting num_layers_minus1 + 1
    for( i = 0; i < nestingNumLayers; i ++)
        NestingLayerId[ i ] = ( i  = = 0 ) ? nuh_layer_id : scalable nesting layer_id[ i ]
}
```

where scalable nesting all_layers_flag specifies whether the scalable-nested SEI messages apply to all layers, vps_max_layers_minus1 plus one specifies a number of layers specified by a video parameter set (VPS), GeneralLayerIdx is a variable specifying a layer index of a layer with nuh_layer_id equal to vps_layer_id[i], vps_layer_id[i] specifies a nuh_layer_id value of an i-th layer, and scalable nesting num_layers_minus1 plus one specifies a number of layers to which the scalable-nested SEI messages apply.

2. The method of claim 1, wherein each value of scalable nesting layer_id[i] is greater than a nuh_layer_id value of the current SEI NAL unit.

3. The method of any of claims 1-2, wherein scalable nesting num_layers_minus1 is in a range of zero to vps_max_layers_minus1− GeneralLayerIdx[nuh_layer_id], inclusive, where nuh_layer_id is the nuh_layer_id of the current SEI NAL unit.

4. A method implemented in an encoder, the method comprising:

encoding, by a processor of the encoder, a bitstream comprising one or more layers;

encoding into the bitstream, by the processor, a scalable nesting supplemental enhancement information (SEI) message in a current SEI network abstraction layer (NAL) unit, wherein the scalable nesting SEI message contains one or more scalable-nested SEI messages and one or more scalable nesting layer identifiers (Ids) that specify layer Id values of the layers to which the scalable-nested SEI messages apply while applying a constraint to omit a layer Id value of the current SEI NAL unit;

performing, by the processor, a set of bitstream conformance tests on the layers based on the scalable-nested SEI messages; and

storing, by a memory coupled to the processor, the bitstream for communication toward a decoder,

wherein performing the set of bitstream conformance tests further comprises deriving a number of nesting layers (nestingNumLayers) variable that specifies a number of layers to which the scalable-nested SEI messages apply,

wherein performing the set of bitstream conformance tests further comprises deriving a list of nesting layer Ids (NestingLayerId[i]) that specifies a list of nuh_layer_id values of the layers to which the scalable-nested SEI messages apply for i in a range of zero to nestingNumLayers minus one, inclusive,

wherein the scalable nesting layer Id is a scalable nesting layer_id[i] syntax element that specifies a NAL unit header layer Id (nuh_layer_id) value of an i-th layer to which the scalable-nested SEI messages apply when a scalable nesting all layers flag (all_layers_flag) is equal to zero,

wherein the nestingNumLayers and the NestingLayerId[i] are derived as follows:

```
if( scalable nesting all_layers_flag ) {
   nestingNumLayers = vps_max_layers_minus1 + 1 −  GeneralLayerIdx[ nuh_layer_id ]
   for( i = 0; i < nestingNumLayers; i ++)
         NestingLayerId[ i ] = vps_layer_id[ GeneralLayerIdx[ nuh_layer_id ] + i ]
} else {
    nestingNumLayers = scalable nesting num_layers_minus1 + 1
    for( i = 0; i < nestingNumLayers; i ++)
         NestingLayerId[ i ] = ( i  = = 0 ) ? nuh_layer_id : scalable nesting layer_id[ i ]
}
```

where scalable nesting all_layers_flag specifies whether the scalable-nested SEI messages apply to all layers, vps_max_layers_minus1 plus one specifies a number of layers specified by a video parameter set (VPS), GeneralLayerIdx is a variable specifying a layer index of a layer with nuh_layer_id equal to vps_layer_id[i], vps_layer_id[i] specifies a nuh_layer_id value of an i-th layer, and scalable nesting num_layers_minus1 plus one specifies a number of layers to which the scalable-nested SEI messages apply.

5. The method of claim 4, wherein each value of scalable nesting layer_id[i] is greater than a nuh_layer_id value of the current SEI NAL unit.

6. A decoder comprising:

a receiving means for receiving a bitstream comprising one or more layers and a scalable nesting supplemental enhancement information (SEI) message in a current SEI network abstraction layer (NAL) unit, wherein the scalable nesting SEI message contains one or more scalable-nested SEI messages and one or more scalable nesting layer identifiers (Ids) that specify layer Id values of the layers to which the scalable-nested SEI messages apply except a layer Id value of the current SEI NAL unit;

a decoding means for decoding a coded picture from the one or more layers to produce a decoded picture; and

a forwarding means for forwarding the decoded picture for display as part of a decoded video sequence,

a deriving means for deriving a number of nesting layers (nestingNumLayers) variable that specifies a number of layers to which the scalable-nested SEI messages apply; and

a deriving means for deriving a list of nesting layer Ids (NestingLayerId[i]) that specifies a list of nuh_layer_id values of the layers to which the scalable-nested SEI messages apply for i in a range of zero to nestingNumLayers minus one, inclusive;

wherein the scalable nesting layer Ids are a scalable nesting layer_id[i] syntax element that specifies a NAL unit header layer Id (nuh_layer_id) value of an i-th layer to which the scalable-nested SEI messages apply when a scalable nesting all layers flag (all_layers_flag) is equal to zero,

wherein the nestingNumLayers and the NestingLayerId[i] are derived as follows:

```
if( scalable nesting all_layers_flag ) {
   nestingNumLayers = vps_max_layers_minus1 + 1 −  GeneralLayerIdx[ nuh_layer_id ]
   for( i = 0; i < nestingNumLayers; i ++)
        NestingLayerId[ i ] = vps_layer_id[ GeneralLayerIdx[ nuh_layer_id ] + i ]
} else {
    nestingNumLayers = scalable nesting num_layers_minus1 + 1
    for( i = 0; i < nestingNumLayers; i ++)
        NestingLayerId[ i ] = ( i  = = 0 ) ? nuh_layer_id : scalable nesting layer_id[ i ]
}
```

where scalable nesting all_layers_flag specifies whether the scalable-nested SEI messages apply to all layers, vps_max_layers_minus1 plus one specifies a number of layers specified by a video parameter set (VPS), GeneralLayerIdx is a variable specifying a layer index of a layer with nuh_layer_id equal to vps_layer_id[i], vps_layer_id[i] specifies a nuh_layer_id value of an i-th layer, and scalable nesting num_layers_minus1 plus one specifies a number of layers to which the scalable-nested SEI messages apply.

7. An encoder comprising:

an encoding means for:

encoding a bitstream comprising one or more layers; and

encoding into the bitstream a scalable nesting supplemental enhancement information (SEI) message in a current SEI network abstraction layer (NAL) unit, wherein the scalable nesting SEI message contains one or more scalable-nested SEI messages and one or more scalable nesting layer identifiers (Ids) that specify layer Id values of the layers to which the scalable-nested SEI messages apply while applying a constraint to omit a layer Id value of the current SEI NAL unit;

a hypothetical reference decoder (HRD) means for performing a set of bitstream conformance tests on the layers based on the scalable-nested SEI messages; and

a storing means for storing the bitstream for communication toward a decoder,

wherein performing the set of bitstream conformance tests further comprises deriving a number of nesting layers (nestingNumLayers) variable that specifies a number of layers to which the scalable-nested SEI messages apply,

wherein performing the set of bitstream conformance tests further comprises deriving a list of nesting layer Ids (NestingLayerId[i]) that specifies a list of nuh_layer_id values of the layers to which the scalable-nested SEI messages apply for i in a range of zero to nestingNumLayers minus one, inclusive,

wherein the scalable nesting layer Id is a scalable nesting layer_id[i] syntax element that specifies a NAL unit header layer Id (nuh_layer_id) value of an i-th layer to which the scalable-nested SEI messages apply when a scalable nesting all layers flag (all_layers_flag) is equal to zero,

wherein the nestingNumLayers and the NestingLayerId[i] are derived as follows:

```
if( scalable nesting all_layers_flag ) {
   nestingNumLayers = vps_max_layers_minus1 + 1 −  GeneralLayerIdx[ nuh_layer_id ]
   for( i = 0; i < nestingNumLayers; i ++)
         NestingLayerId[ i ] = vps_layer_id[ GeneralLayerIdx[ nuh_layer_id ] + i ]
} else {
    nestingNumLayers = scalable nesting num_layers_minus1 + 1
    for( i = 0; i < nestingNumLayers; i ++)
         NestingLayerId[ i ] = ( i  = = 0 ) ? nuh_layer_id : scalable nesting layer_id[ i ]
}
```

where scalable nesting all_layers_flag specifies whether the scalable-nested SEI messages apply to all layers, vps_max_layers_minus1 plus one specifies a number of layers specified by a video parameter set (VPS), GeneralLayerIdx is a variable specifying a layer index of a layer with nuh_layer_id equal to vps_layer_id[i], vps_layer_id[i] specifies a nuh_layer_id value of an i-th layer, and scalable nesting num_layers_minus1 plus one specifies a number of layers to which the scalable-nested SEI messages apply.