

(51) International Patent Classification:
H04N 21/6336 (2011.01)(21) International Application Number:
PCT/SG2012/000333(22) International Filing Date:
12 September 2012 (12.09.2012)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicants (for all designated States except US):
AGENCY FOR SCIENCE, TECHNOLOGY AND RESEARCH [SG/SG]; 1 Fusionopolis Way, #20-10 Connexis, Singapore 138632 (SG). **HUAWEI INTERNATIONAL PTE. LTD.** [CN/CN]; No. 2222, Xin Jinqiao Road, Pudong, Shanghai 201206 (CN).

(72) Inventors; and

(75) Inventors/Applicants (for US only): **YEN, Kai** [SG/SG]; c/o PKM, Institute for Infocomm Research, 1 Fusionopolis Way, #21-01 Connexis, South Tower, Singapore 138632 (SG). **SUN, Sumei** [SG/SG]; c/o PKM, Institute for Infocomm Research, 1 Fusionopolis Way, #21-01 Connexis, South Tower, Singapore 138632 (SG). **SETHAKASET, Ubolthip** [TH/SG]; c/o PKM, Institute for Infocomm Research, 1 Fusionopolis Way, #21-01 Connexis, South Tower, Singapore 138632 (SG). **TAN, Peng Hui** [SG/SG]; c/o PKM, Institute for Infocomm Research, 1 Fusionopolis Way, #21-01 Connexis, South Tower, Singapore 138632 (SG). **LI, Zhengguo** [CN/SG]; c/o PKM, Institute for Infocomm Research, 1 Fusionopolis Way, #21-01 Connexis, South Tower, Singapore 138632 (SG). **ZHENG, Jinghong** [CN/SG]; c/o PKM, Institute for Infocomm Research, 1 Fusionopolis Way, #21-01 Connexis, South Tower, Singapore 138632 (SG).pore 138632 (SG). **WEI, Yuejun** [CN/CN]; c/o Huawei International Pte. Ltd., No. 2222, Xin Jinqiao Road, Pudong, Shanghai 201206 (CN). **WU, Kedi** [CN/CN]; c/o Huawei International Pte. Ltd., No. 2222, Xin Jinqiao Road, Pudong, Shanghai 201206 (CN).(74) Agent: **VIERING, JENTSCHURA & PARTNER LLP**; P.O. Box 1088, Rochor Post Office, Rochor Road, Singapore 911833 (SG).

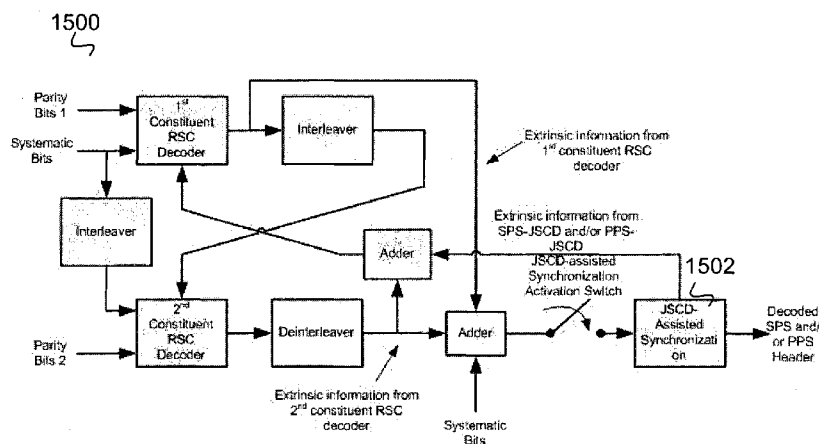
(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE,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

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

Published:

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

(54) Title: A DECODER AND METHOD OF DECODING AN ENCODED HEADER OF A VIDEO DATA PACKET



(57) Abstract: The present invention is a decoder including an input unit configured to receive an encoded header of a video data packet; and a circuit configured to decode the encoded header using joint source-channel decoding (JSCD). A method of decoding an encoded header of a video data packet is also disclosed.

A Decoder and Method of Decoding an Encoded Header of a Video Data Packet

Technical Field

[0001] Various embodiments generally relate to the field of decoders and methods of decoding an encoded video sequence, in particular, an encoded header of a video data packet.

Background

[0002] The H.264 video format has a very broad application range that covers all forms of digital compressed video from low bit-rate Internet streaming applications to HDTV broadcast and Digital Cinema applications. Correct decoding of an associated H.264 video sequence is important to ensure high quality video playback.

[0003] Both headers are encoded by the H.264 encoder according to a parsing process as specified by the H.264 standard. The headers are then transmitted to the receiver along with the payload video information.

[0004] The Sequence Parameter Set (SPS) header contains a varying number of decoding parameters which are essential to the correct decoding of the associated H.264 video sequence. Likewise, the Picture Parameter Set (PPS) header contains important parameters to ensure a successful decoding of a sequence or subset of coded frames of an H.264 encoded video.

[0005] Before the H.264 payload video sequence can be decoded to a format suitable for viewing at the receiver, the decoding parameters as specified in both the SPS and PPS headers must first be extracted by the H.264 decoder. The H.264 decoder follows substantially or exactly the same parsing process as the H.264 encoder. If there are no errors in the headers, then the parsing process is always successful and the correct decoding parameters are extracted. The H.264 decoder then continues to decode the rest of the payload video sequence based on these extracted parameters.

[0006] On the other hand, any errors in the headers may give two possible outcomes. Firstly, bit errors may disrupt the parsing process. When this happens, the H.264 decoder is unable to extract all the necessary decoding parameters. Hence, the associated payload video sequence cannot be decoded even if the sequence may be received without any errors. Secondly, the parsing process may be successful but any undetected bit errors in the headers may lead to incorrect decoding parameters being extracted. When this happens, the received payload video sequence may be decoded in a manner that does not mirror the encoding process. This usually gives a degradation of the playback video quality.

[0007] Hence it is very important that both the SPS and PPS headers are detected correctly at the receiver front end (i.e. the physical layer) before it reaches the H.264 decoder at the application layer.

[0008] A source decoder is unable to decode the video sequence if the received SPS and PPS headers at a receiver are erroneous.

[0009] Conventionally, the headers are protected only by channel codes. These channel codes correct a high number of bit errors at the expense of a larger transmission bandwidth. Better error correction capabilities are accomplished at the expense of a larger transmission bandwidth.

[0010] It has been proposed to use Joint Source and Channel Decoding (JSCD) to decode the H.264 video sequence. For example, Integrated JSCD (IJSCD) scheme is used whereby the channel decoder and the entropy decoder are merged into a single unit. Due to the merging of both decoders into a single unit, iterative processing is not possible for IJSCD.

[0011] In another example, a H.264 decoder is modified in order to serve two roles. One role is the usual source decoder, while the other serves as an error detection decoder. This role can be fulfilled due to the fact that the H.264 decoder has the capability of detecting erroneously estimated video bitstream, based on the source syntax and semantics. This way, iterative process is possible. Conventionally, the channel decoder selects the best sequence and pass this sequence to the source decoder.

[0012] In all the proposals, it has always been assumed that the parameters required for the JSCD to work are readily available at the receiver. For example, the information of these parameters are transmitted as side information. Such techniques incur additional overhead and moreover the current systems are also not designed for such considerations.

[0013] Thus, there is a need to provide a decoder and a method of decoding a video sequence that would enhance the error correction without suffering from any bandwidth expansion at the physical layer; thereby seeking to address at least the above mentioned problems.

Summary of the Invention

[0014] In a first aspect, the present invention relates to a decoder including an input unit configured to receive an encoded header of a video data packet; and a circuit configured to decode the encoded header using joint source-channel decoding (JSCD). In one embodiment, the decoder further includes a synchronizer configured to determine a presence of a complete network abstract layer (NAL) unit in the video data packet, wherein the complete NAL unit comprises a NAL unit sandwiched between two prefixes, and to determine the location of the encoded header in the video data packet and/or the length of the encoded header from the video data packet.

[0015] According to a second aspect, the present invention relates to a receiver including a decoder in accordance to various embodiments.

[0016] In a third aspect, the present invention relates to a method of decoding an encoded header of a video data packet, the method including receiving an encoded header of a video data packet; and decoding the encoded header using joint source-channel decoding (JSCD).

Brief Description of the Drawings

[0017] In the drawings, like reference characters generally refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis

instead generally being placed upon illustrating the principles of the invention. The dimensions of the various features/elements may be arbitrarily expanded or reduced for clarity. In the following description, various embodiments of the invention are described with reference to the following drawings, in which:

[0018] FIG. 1 shows a schematic block diagram of an overview of a system comprising JSCD;

[0019] FIG. 2 shows a schematic block diagram of a decoder, in accordance to various embodiments;

[0020] FIG. 3 shows a frame structure of a NAL unit in a H.264 encoded video sequence;

[0021] FIG. 4 shows a block diagram of a SPS-JSCD, in accordance to various embodiments;

[0022] FIG. 5 shows an exemplary trellis, in accordance to various embodiments;

[0023] FIG. 6 shows another exemplary trellis, in accordance to various embodiments;

[0024] FIG. 7 shows a block diagram of a VLC SPS-JSCD, in accordance to various embodiments;

[0025] FIG. 8 shows an exemplary transformed trellis of FIG. 6, in accordance to various embodiments;

[0026] FIG. 9 shows a schematic block diagram of a receiver configuration with only the SPS-JSCD and perfect synchronization with notation $RC_{SPS-JSCD}(N_{inner}, N_{outer})$, in accordance to various embodiments;

[0027] FIG. 10 shows a schematic block diagram of a receiver configuration with only the PPS-JSCD and perfect synchronization with notation $RC_{PPS-JSCD}(N_{inner}, N_{outer})$, in accordance to various embodiments;

[0028] FIG. 11 shows a schematic block diagram of a receiver configuration with the SPS-JSCD and PPS-JSCD and perfect synchronization with notation $RC_{JSCDs}(N_{inner}, N_{outer})$, in accordance to various embodiments;

[0029] FIG. 12 shows a frame structure of a complete NAL unit;

[0030] FIG. 13 shows a synchronization pattern, where $N_{RBSP} = 24, 32, 40, 48, 56, \dots, 104$, in accordance to various embodiments;

[0031] FIG. 14 shows a flow diagram of the synchronization algorithm, in accordance to various embodiments;

[0032] FIG. 15 shows a schematic block diagram of a receiver configuration with the JSCD-assisted synchronization with notation $RC_{\text{SYNC}}(N_{\text{inner}}, N_{\text{outer}})$, in accordance to various embodiments;

[0033] FIG. 16 shows a flow diagram of a method of decoding an encoded header of a video data packet, in accordance to various embodiments;

[0034] FIG. 17 shows a block diagram of a transmitter model, in accordance to various embodiments;

[0035] FIG. 18 shows a frame structure including complete SPS and PPS NAL units, in accordance to various embodiments;

[0036] FIG. 19 shows a frame structure of a SPS header NAL unit, in accordance to various embodiments;

[0037] FIG. 20 shows the basis SPS header BER with $RC_{\text{SPS-JSCD}}(N_{\text{inner}}, 1)$ for a rate-1/3 turbo code with a packet size of 1024;

[0038] FIG. 21 shows the basis SPS header HER with $RC_{\text{SPS-JSCD}}(N_{\text{inner}}, 1)$ for a rate-1/3 turbo code with a packet size of 1024;

[0039] FIG. 22 shows Y PSNR with $RC_{\text{SPS-JSCD}}(8, 1)$ for a rate-1/3 turbo code with a packet size of 1024;

[0040] FIG. 23 shows the basis SPS header BER with $RC_{\text{SPS-JSCD}}(N_{\text{inner}}, 1)$ for a rate-3/4 turbo code with a packet size of 1025;

[0041] FIG. 24 shows the basis SPS header HER with $RC_{\text{SPS-JSCD}}(N_{\text{inner}}, 1)$ for a rate-3/4 turbo code with a packet size of 1025;

[0042] FIG. 25 shows Y PSNR with $RC_{\text{SPS-JSCD}}(8, 1)$ for a rate-3/4 turbo code with a packet size of 1025;

[0043] FIG. 26 shows the basis SPS header BER with $RC_{\text{SPS-JSCD}}(1, N_{\text{outer}})$ for a rate-1/3 turbo code with a packet size of 1024;

[0044] FIG. 27 shows the basis SPS header HER with $RC_{\text{SPS-JSCD}}(1, N_{\text{outer}})$ for a rate-1/3 turbo code with a packet size of 1024;

[0045] FIG. 28 shows Y PSNR with $RC_{SPS-JSCD} (1,8)$ for a rate-1/3 turbo code with a packet size of 1024;

[0046] FIG. 29 shows the basis SPS header BER with $RC_{SPS-JSCD} (2, N_{outer})$ for a rate-1/3 turbo code with a packet size of 1024;

[0047] FIG. 30 shows the basis SPS header HER with $RC_{SPS-JSCD} (2, N_{outer})$ for a rate-1/3 turbo code with a packet size of 1024;

[0048] FIG. 31 shows the basis SPS header BER with $RC_{SPS-JSCD} (4, N_{outer})$ and $RC_{SPS-JSCD} (6, N_{outer})$ for a rate-1/3 turbo code with a packet size of 1024;

[0049] FIG. 32 shows the basis SPS header HER with $RC_{SPS-JSCD} (4, N_{outer})$ and $RC_{SPS-JSCD} (6, N_{outer})$ for a rate-1/3 turbo code with a packet size of 1024;

[0050] FIG. 33 shows Y PSNR with $RC_{SPS-JSCD} (2,8)$, $RC_{SPS-JSCD} (4,6)$, and $RC_{SPS-JSCD} (6,4)$ for a rate-1/3 turbo code with a packet size of 1024;

[0051] FIG. 34 shows the basis PPS header BER with $RC_{PPS-JSCD} (6, N_{outer})$ for a rate-1/3 turbo code with a packet size of 1024;

[0052] FIG. 35 shows the basis PPS header HER with $RC_{PPS-JSCD} (6, N_{outer})$ for a rate-1/3 turbo code with a packet size of 1024;

[0053] FIG. 36 shows Y PSNR with $RC_{PPS-JSCD} (6,4)$ for a rate-1/3 turbo code with a packet size of 1024;

[0054] FIG. 37 shows the basis SPS header BER with $RC_{JSCDs} (6, N_{outer})$ for a rate-1/3 turbo code with a packet size of 1024;

[0055] FIG. 38 shows the basis PPS header BER with $RC_{JSCDs} (6, N_{outer})$ for a rate-1/3 turbo code with a packet size of 1024;

[0056] FIG. 39 shows Y PSNR with $RC_{JSCDs} (6,4)$ for a rate-1/3 turbo code with a packet size of 1024;

[0057] FIG. 40 shows the basis SPS header BER with $RC_{SYNC} (6, N_{outer})$ for a rate-1/3 turbo code with a packet size of 1024 and $M = 1$;

[0058] FIG. 41 shows the basis PPS header BER with $RC_{SYNC} (6, N_{outer})$ for a rate-1/3 turbo code with a packet size of 1024 and $M = 1$;

[0059] FIG. 42 shows the basis SPS header BER with $RC_{SYNC} (6, N_{outer})$ for a rate-1/3 turbo code with a packet size of 1024 and $M = 4$;

[0060] FIG. 43 shows the basis PPS header BER with $RC_{\text{SYNC}}(6, N_{\text{outer}})$ for a rate-1/3 turbo code with a packet size of 1024 and $M = 4$;

[0061] FIG. 44 shows Y PSNR with $RC_{\text{SYNC}}(6, 4)$ for a rate-1/3 turbo code with a packet size of 1024 and $M = 4$;

[0062] FIG. 45 shows the basis SPS header BER with $RC_{\text{SYNC}}(6, N_{\text{outer}})$ for a rate-1/3 turbo code with a packet size of 1024 and $M = 6$;

[0063] FIG. 46 shows the basis PPS header BER with $RC_{\text{SYNC}}(6, N_{\text{outer}})$ for a rate-1/3 turbo code with a packet size of 1024 and $M = 6$;

[0064] FIG. 47 shows the basis SPS header BER with $RC_{\text{SYNC}}(6, N_{\text{outer}})$ for a rate-1/3 turbo code with a packet size of 1024 and Threshold = 52.0;

[0065] FIG. 48 shows the basis PPS header BER with $RC_{\text{SYNC}}(6, N_{\text{outer}})$ for a rate-1/3 turbo code with a packet size of 1024 and Threshold = 52.0;

[0066] FIG. 49 shows the packet BER with $RC_{\text{SYNC}}(6, N_{\text{outer}})$ for a rate-1/3 turbo code with a packet size of 1024 without any headers in the packet.

Detailed Description

[0067] The following detailed description refers to the accompanying drawings that show, by way of illustration, specific details and embodiments in which the invention may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention. Other embodiments may be utilized and structural, and logical changes may be made without departing from the scope of the invention. The various embodiments are not necessarily mutually exclusive, as some embodiments can be combined with one or more other embodiments to form new embodiments.

[0068] In order that the invention may be readily understood and put into practical effect, particular embodiments will now be described by way of examples and not limitations, and with reference to the figures.

[0069] In joint source-channel decoding (JSCD), the decoding is performed by exploiting the properties of the channel code and source code jointly. FIG. 1 shows an overview

system 100 of video transmission using JSCD. In FIG. 1, the system 100 includes a transmitter 102 which includes a source encoder 104 to encode a video sequence X 106. The source-encoded video sequence X' 108 is subsequently passed to a channel encoder 110 which outputs a transmitting encoded sequence Y₁ 112 for transmission via a channel 114. The channel 114 may be a noisy channel and may introduce errors to the encoded sequence Y' 116, which is received by a receiver 118. The receiver 118 includes a channel decoder 120 to decode the encoded sequence Y' 116. A decoded header is obtained from the channel-decoded sequence Y₂ 122 using JSCD 124. The JSCD-decoded sequence Z' 126 including the decoded header, which contains important parameters for correct decoding of the associated sequence Y' 116 at the receiver 118, is passed to a source decoder 128 to obtain the decoded video sequence Z 130.

[0070] Various embodiments provide Joint Source and Channel Decoding (JSCD) for Sequence Parameter Set (SPS) headers (SPS-JSCD).

[0071] Various embodiments provide JSCD for Picture Parameter Set (PPS) Headers (PPS-JSCD).

[0072] The SPS-JSCD and PPS-JSCD are able to reduce the probability of bit and header errors without any increased in the transmission bandwidth. Reduction of the header error probability increases the likelihood of a source decoder able to decode the associated video sequence.

[0073] Successful implementation of both JSCDs at a receiver requires the knowledge of two parameters; namely the location of the headers in a received packet as well as its length. If the location of the headers in the received packet is unknown, then the JSCDs would not know where to begin decoding the respective headers. If the length of the headers is unknown, then the JSCDs would not be able to construct an accurate trellis for reliable decoding. This results in erroneous decoding whose undesirable effects may be propagated during the course of the iterative decoding. Both of these parameters may not be deterministic at the receiver and may have to be acquired by some means.

[0074] In various embodiments, a practical synchronization algorithm is provided to acquire the parameters, which are necessary for the SPS-JSCD and PPS-JSCD to decode correctly, directly from the received packet. The synchronization algorithm is aided by

the JSCDs in accordance to various embodiments; thereby enabling the problem of acquiring the necessary decoding parameters with a high probability of success for the JSCDs to be addressed. The synchronization algorithm in accordance to various embodiments make use of these parameters to subsequently reduce significantly the number of bit errors in the respective headers and therefore do not incur additional overhead or bandwidth required to transmit the parameters, in contrast to transmitting the parameters as side information. As a result, the performance is greatly improved, which approaches nearly the same level of performance achieved with error-free headers. These parameters may be directly acquired from the received packet at the physical layer without the need of any side information.

[0075] In other embodiments, the SPS-JSCD and PPS-JSCD may still function without the synchronization as long as the abovementioned two decoding parameters are made known to the SPS-JSCD and PPS-JSCD, for example, as side information.

[0076] Various embodiments also provide JSCD for H.264 SPS and PPS headers with synchronization. There are currently no existing literature that addresses the synchronization of the SPS or PPS headers, or in general, synchronization related to a H.264 encoded video sequence.

[0077] In a first aspect, a decoder 200 is provided as shown in FIG. 2. In FIG. 2, the decoder 200 includes an input unit 202 configured to receive an encoded header of a video data packet; and a circuit 204 configured to decode the encoded header using joint source-channel decoding (JSCD).

[0078] As used herein, the term “decoder” as in the decoder 200 generally refers to a device which detects and corrects errors present in the information to be decoded. For example, such errors may be introduced by a noisy environment (e.g. a noisy or degrading transmission channel) or may be due to loss of information. The term “decode” may be similarly defined as the function of the decoder 200 and may interchangeably be referred to as “translate” or “convert”. It should be appreciated that the decoder 200 is different from a decoder such as a source decoder or a channel decoder, which performs the reverse operation of an encoder such as a source encoder or a channel encoder, by underdoing the encoding so that the original information can be retrieved.

[0079] The term “information” may be interchangeably referred to as bit pattern, bit sequence, video data, video sequence, data, data sequence or data packet.

[0080] For example, the decoder 200 may be the JSCD 124 of FIG. 1.

[0081] In one embodiment, the decoder 200 may be a trellis-based decoder. The term “trellis” is used to describe schemes by a state-transition (trellis) diagram similar to trellis diagrams of binary convolutional codes. A “trellis” may be interchangeably referred to as “branch”.

[0082] The term “using” may interchangeably refer to “invoking”, “executing”, or “performing”.

[0083] In the context of various embodiments, a “circuit” may be understood as any kind of a logic implementing entity, which may be special purpose circuitry or a processor executing software stored in a memory, firmware, or any combination thereof. Thus, a “circuit” may be a hard-wired logic circuit or a programmable logic circuit such as a programmable processor, e.g. a microprocessor (e.g. a Complex Instruction Set Computer (CISC) processor or a Reduced Instruction Set Computer (RISC) processor). A “circuit” may also be a processor executing software, e.g. any kind of computer program, e.g. a computer program using a virtual machine code such as e.g. Java or e.g. digital signal processing algorithm. Any other kind of implementation of the respective functions which are described may also be understood as a “circuit” in accordance with an alternative aspect of this disclosure.

[0084] The phrase “joint source-channel decoding (JSCD)” refers to decoding using a single code instead of source decoding (decompression) followed by channel decoding (error correction). When operating under a delay-constraint on a time-varying channel, it is generally not optimal to regard the source decoder and channel decoder separately. The JSCD encompasses all decoding techniques where the source and channel decoders are not entirely separated.

[0085] As used herein, the phrase “encoded header of a video data packet” may mean an encoded video data packet which contains a header; or a header, which is encoded, being contained in a video data packet which may also be encoded. For example, the “encoded header” may contain errors such that by merely performing a reverse operation of the

encoder on the encoded header would not retrieve the original header. For example, the video data packet may be the encoded video sequence Y' 116 of FIG. 1.

[0086] In various embodiments, the encoded header may be an encoded sequence parameter set (SPS) header or an encoded picture parameter set (PPS) header.

[0087] The SPS header is the header in a sequence parameter set, which applies to a series of consecutive coded video pictures called a coded video sequence. The PPS header is the header in a picture parameter set, which applies to the decoding of one or more individual pictures within a coded video sequence.

[0088] A "parameter set" contains information that is expected to rarely change and offers the decoding to a large number of video data packets. For example, the video data packets may be Video Coding Layer (VCL) network abstraction layer (NAL) units of a H.264 video sequence.

[0089] The H.264 encoded video sequence is made up of a series of NAL units. NAL units can be further sub-divided into two categories; Video Coding Layer (VCL) NAL units and non-VCL NAL units. VCL NAL units basically contain the coded video data. Therefore, SPS and PPS headers are considered as non-VCL NAL units. Regardless of the category, NAL units follows the structure as shown in FIG. 3.

[0090] In FIG. 3, the H.264 encoded video sequence 300 is made up of a plurality of NAL units, for example, the NAL unit 302. The NAL unit 302 begins with a prefix 304, which is 4 bytes long. Basically, it consists of the zero_byte (0x00) and the start_code_prefix_one_3bytes (0x000001), in accordance to a byte stream format specified in the H.264 standard, where 0x is the hexadecimal notation. The H.264 standard makes use of this prefix 304 to indicate the start of an NAL unit 302 and it is primarily used for synchronization by the H.264 decoder at the application layer.

[0091] The 4-byte prefix 304 is then followed by the NAL unit identifier (or syntax) 306, which are listed in Table 1.

[0092] Table 1

NAL unit syntax	Descriptor	Remarks
forbidden_zero_bit	f(1)	Equals to zero
nal_ref_idc	u(2)	Not equals to zero if the NAL unit contains a SPS, SPS extension, subset SPS, PPS, etc, as specified in the H.264 standard. In the joint model (JM) codec, nal_ref_idc = 11 in binary, if it is not equals to zero.
nal_unit_type	u(5)	Specifies the type of RBSP contained in the NAL unit. For the SPS header, nal_unit_type = 7 or 00111 in binary. For the PPS header, nal_unit_type = 8 or 01000.

[0093] The NAL unit syntax 306 uniquely identifies the type of NAL unit. More specifically, for the SPS header, the NAL unit syntax 306 has the following bit sequence : 0 1 1 0 0 1 1 1 = 0x67. On the other hand, the NAL unit syntax 306 for the PPS header has the following bit sequence : 0 1 1 0 1 0 0 0 = 0x68.

[0094] Finally, the last part of the NAL unit 302 constitutes the Raw Byte Sequence Payload (RBSP) 308, which is terminated by a sequence of trailing bits so that the number of bits in the RBSP 308 is in multiples of a byte. This NAL unit 302 is followed by a next NAL unit (if any) having the same structure as shown in FIG. 3.

[0095] Both the SPS header and the PPS header constitute the RBSP portion of a non-VCL NAL unit. For a successful parsing of parameters, both the H.264 encoder and decoder, for example, the encoders 104, 110 and the decoders 120, 124, 128 of FIG. 1, follow the same data syntax as stipulated in the H.264 standard. The syntax that can be found in a SPS header are listed in Table 2.

[0096] Table 2

Table Row No.	seq_parameter_set_data(){	Descriptor
2.1	profile_idc	u(8)

2.2	constraint_set0_flag	u(1)
2.3	constraint_set1_flag	u(1)
2.4	constraint_set2_flag	u(1)
2.5	constraint_set3_flag	u(1)
2.6	constraint_set4_flag	u(1)
2.7	constraint_set5_flag	u(1)
2.8	reserved_zero_2bits	u(2)
2.9	level_idc	u(8)
2.10	seq_parameter_set_id	ue(v)
2.11	if (profile_idc == 100 profile_idc == 110 profile_idc == 122 profile_idc == 244 profile_idc == 44 profile_idc == 83 profile_idc == 86 profile_idc == 118 profile_idc == 128){	
2.12	chroma_format_idc	ue(v)
2.13	if (chroma_format_idc == 3)	
2.14	separate_colour_plane_flag	u(1)
2.15	bit_depth_luma_minus8	ue(v)
2.16	bit_depth_chroma_minus8	ue(v)
2.17	qpprime_y_zero_transform_bypass_flag	u(1)
2.18	seq_scaling_matrix_present_flag	u(1)
2.19	if (seq_scaling_matrix_present_flag)	
2.20	for (i = 0; i < ((chroma_format_idc != 3) ? 8 : 12); i++) {	
2.21	seq_scaling_list_present_flag[i]	u(1)
2.22	if (seq_scaling_list_present_flag[i])	
2.23	if (i < 6)	
2.24	scaling_list(ScalingList4x4[i], 16, UseDefaultScalingMatrix4x4Flag[i])	
2.25	Else	

2.26	scaling_list(ScalingList4x4[i], 16, UseDefaultScalingMatrix8x8Flag[i - 6])	
2.27	}	
2.28	}	
2.29	log2_max_frame_num_minus4	ue(v)
2.30	pic_order_cnt_type	ue(v)
2.31	if (pic_order_cnt_type == 0)	
2.32	log2_max_pic_order_cnt_lsb_minus4	ue(v)
2.33	else if (pic_order_cnt_type == 1){	
2.34	delta_pic_order_always_zero_flag	u(1)
2.35	offset_for_non_ref_pic	se(v)
2.36	offset_for_top_to_bottom_field	se(v)
2.37	num_ref_frames_in_pic_order_cnt_cycle	ue(v)
2.38	for (i = 0; i < num_ref_frames_in_pic_order_cnt_cycle; i++)	
2.39	offset_for_ref_frame[i]	se(v)
2.40	}	
2.41	max_num_ref_frames	ue(v)
2.42	gaps_in_frame_num_value_allowed_flag	u(1)
2.43	pic_width_in_mbs_minus1	ue(v)
2.44	pic_height_in_map_units_minus1	ue(v)
2.45	frame_mbs_only_flag	u(1)
2.46	if (!frame_mbs_only_flag)	
2.47	mb_adaptive_frame_field_flag	u(1)
2.48	direct_8x8_inference_flag	u(1)
2.49	frame_cropping_flag	u(1)
2.50	if (frame_cropping_flag){	
2.51	frame_crop_left_offset	ue(v)
2.52	frame_crop_right_offset	ue(v)

2.53	frame_crop_top_offset	ue(v)
2.54	frame_crop_bottom_offset	ue(v)
2.55	}	
2.56	vui_parameters_present_flag	u(1)
2.57	if(vui_parameters_present_flag)	
2.58	vui_parameters()	
2.59	}	
2.60	trailing_bits()	

[0097] The syntax for the PPS header are listed in Table 3.

[0098] Table 3

Table Row No.	pic_parameter_set_rbsp(){	Descriptor
3.1	pic_parameter_set_id	ue(v)
3.2	seq_parameter_set_id	ue(v)
3.3	entropy_coding_mode_flag	u(1)
3.4	bottom_field_pic_order_in_frame_present_flag	u(1)
3.5	num_slice_groups_minus1	ue(v)
3.6	if(num_slice_groups_minus1 > 0) {	
3.7	slice_group_map_type	ue(v)
3.8	if(slice_group_map_type == 0)	
3.9	for (iGroup = 0; iGroup <= num_slice_groups_minus1; iGroup++)	
3.10	run_length_minus1[iGroup]	ue(v)
3.11	else if (slice_group_map_type == 2)	
3.12	for (iGroup = 0; iGroup <= num_slice_groups_minus1; iGroup++) {	
3.13	top_left[iGroup]	ue(v)
3.14	bottom_right[iGroup]	ue(v)

3.15	}	
3.16	else if (slice_group_map_type == 3 slice_group_map_type == 4 slice_group_map_type == 5) {	
3.17	slice_group_change_direction_flag	u(1)
3.18	slice_group_change_rate_minus1	ue(v)
3.19	} else if (slice_group_map_type == 6) {	
3.20	pic_size_in_map_units_minus1	ue(v)
3.21	for (i = 0; i <= pic_size_in_map_units_minus1; i++)	
3.22	slice_group_id[i]	u(v)
3.23	}	
3.24	}	
3.25	num_ref_idx_10_default_active_minus1	ue(v)
3.26	num_ref_idx_11_default_active_minus1	ue(v)
3.27	weighted_red_flag	u(1)
3.28	weighted_bired_idc	u(2)
3.29	pic_init_qp_minus26	se(v)
3.30	pic_init_qs_minus26	se(v)
3.31	chroma_qp_index_offset	se(v)
3.32	deblocking_filter_control_present_flag	u(1)
3.33	constrained_intra_pred_flag	u(1)
3.34	redundant_pic_cnt_present_flag	u(1)
3.35	if (more_rbsp_data()) {	
3.36	transform_8x8_mode_flag	u(1)
3.37	pic_scaling_matrix_present_flag	u(1)
3.38	if (pic_scaling_matrix_present_flag)	
3.39	for (i = 0; i < 6 + ((chroma_format_idc != 3) ? 2:6) *	

	transform_8x8_mode_flag; i++) {	
3.40	pic_scaling_list_present_flag[i]	u(1)
3.41	if (pic_scaling_list_present_flag[i])	
3.42	if (i < 6)	
3.43	scaling_list(ScalingList4x4[i], 16, UseDefaultScalingMatrix4x4Flag[i])	
3.44	else	
3.45	scaling_list(ScalingList8x8[i-6], 64, UseDefaultScalingMatrix8x8Flag[i-6])	
3.46	}	
3.47	second_chroma_qp_index_offset	se(v)
3.48	}	
3.49	rbsp_trailing_bits()	
3.50	}	

[0099] Detailed explanation of the syntax of both headers can be found in the H.264 standardization document. The nomenclature for the descriptors given in the 2nd column of both tables is as follows:-

- u(n): unsigned integer using n bits.
- ue(v): unsigned integer Exp-Golomb-coded syntax element with the left bit first.
- se(v): signed integer Exp-Golomb-coded syntax element with the left bit first.

[00100] From both Tables 2 and 3, it can be seen that the syntax of the SPS and PPS headers can either be represented by a fixed length code (FLC) or a variable-length code (VLC). The VLC is of the Exp-Golomb type. Examples of the Exp-Golomb codes are listed in Table 4.

[00101] Table 4

Value	Bit sequence	Value	Bit sequence
0	1	9	0001010
1	010	10	0001011
2	011	11	0001100
3	00100	12	0001101
4	00101	13	0001110
5	00110	14	0001111
6	00111	15	000010000
7	0001000	16	000010001
8	0001001	17	000010010

[00102] In various embodiments, the encoded header or a part thereof may include a variable length code (VLC).

[00103] A VLC header includes a code of variable (or non-fixed) length.

[00104] In various embodiments, Tables 2 and 3 show that both the SPS and PPS headers are made up of several if statements. Because of these if statements, the number of syntax found in a transmitted SPS or PPS header is a varying term, which depends on whether the conditions of the if statements are true or false. The receiver, for example, the receiver 118 of FIG. 1, has no knowledge of how many syntax or the type of syntax (i.e. the descriptors) are there in the received headers. For decoding and error correction purposes, both the SPS-JSCD and PPS-JSCD require such information. The lack of this knowledge presents a challenging proposition for the design of the JSCD.

[00105] To illustrate, suppose the receiver receives 6 bits corresponding to the SPS header. These 6 bits can represent several possibilities. For example, the SPS header may consists of one u(1) syntax and one ue(v) syntax of 5-bit length. Or it may consists of two ue(v) syntax, each having a 3-bit length. It may also consists of 3 u(1) syntax (not necessarily in a contiguous fashion) and 1 3-bit ue(v) syntax and so forth. Therefore, for effective decoding and error correction, the SPS-JSCD and PPS-JSCD deduce how many

syntax are there in their respective received SPS and PPS header and what is the descriptor type of these syntax.

[00106] In various embodiments, when decoding the encoded header using joint source-channel decoding (JSCD), the circuit 204 may be configured to determine a code from the received encoded header or a part thereof, and using the code to generate a bit sequence and a probability value. The probability value indicates the likelihood that the generated bit sequence is a correctly decoded header or a part thereof of the video data packet.

[00107] As used herein, the term “determine” may refer but is not limited to “compute”, “verify”, “check”, “evaluate”, “establish”, or “assess”.

[00108] The term “code” generally refers to a rule for converting a piece of information. A “code” may be represented by a bit pattern or may be formed by a trellis.

[00109] The term “generate” may refer but is not limited to “formulate”, “derive”, “create”, or “obtain”.

[00110] In the context of various embodiments, the phrase “correctly decoded header” means the substantially original header. It should be appreciated and understood that the term “substantially” may include “exactly” and “similar” which is to an extent that it may be perceived as being “exact”. For illustration purposes only and not as a limiting example, the term “substantially” may be quantified as a variance of +/- 5% from the exact or actual. For example, the phrase “A is (at least) substantially the same as B” may encompass embodiments where A is exactly the same as B, or where A may be within a variance of +/- 5%, for example of a value, of B, or vice versa.

[00111] In various embodiments, when determining the code, the circuit 204 may be configured to generate a plurality of codes and respective probability values from the received encoded header or the part thereof and to select from the plurality of codes the code based on the respective probability values.

[00112] For example, the circuit 204 may be configured to select from the plurality of codes the code having the largest probability value, which is having the highest likelihood.

[00113] In various embodiments, the circuit 204 may operate based on a mode of posterior distribution.

[00114] In one embodiment, the circuit 204 may operate based on maximum *a posteriori* probability (MAP). For example, the circuit 204 may operate based on BCJR MAP or logMAP. In this example, the encoded header may include bit indexes; and the decoder 200 may further include a syntax converter configured to convert the bit indexes into respective syntax indexes which the circuit 204 uses to determine the code. Detail of the operation of the circuit 204 is described later on.

[00115] In various embodiments, when the encoded header is an encoded SPS header, the decoder 200 may further include a sub-decoder configured to decode another part of the encoded SPS header comprising a fixed length code of more than 1 bit. For example, the fixed length code may include 25 bits.

[00116] In various embodiments, the decoder 200 may further include a demultiplexer configured to divide the encoded header received by the input unit 202 into the part comprising the variable length code for the circuit 204 and the other part comprising the fixed length code for the sub-decoder.

[00117] As used herein, the term “divide” may be interchangeably referred to as “separate” or “split”.

[00118] In various embodiments, the decoder 200 may further include a multiplexer configured to multiplex the bit sequence generated by the circuit 204 and a decoded output from the sub-decoder.

[00119] As used herein, the term “multiplex” may be interchangeably referred to as “join”, “combine”, or “append”.

[00120] In various embodiments, the decoder 200 may further include a convolutional decoding unit configured to generate extrinsic information for the circuit 204.

[00121] In the context of various embodiments, the phrase “extrinsic information” may be referred to as “refinement factor”. The extrinsic information may be used to update *the a priori* probabilities. Generally, extrinsic information is introduced to identify the component of the generated reliability value which depends on redundant information

introduced by the considered constituent code. For example, a natural reliability value, in the binary case, may be the logarithmic likelihood ratio (LLR).

[00122] For example, the convolutional decoding unit may include a turbo decoder or a recursive systematic convolutional (RSC) decoder.

[00123] In various embodiments, the convolutional decoding unit may further be configured to receive side information from a transmitter sending the video data packet, the side information indicating the location of the encoded header in the video data packet, the length of the encoded header, or both.

[00124] For example, the transmitter may be the transmitter 102 of FIG. 1.

[00125] As used herein, the “side information” is not part of the main information. In one example, the side information may be transmitted through the same channel of the main information. In another example, the side information may be transmitted via a different channel from the one the main information is transmitted.

[00126] The term “indicating” includes, but not be limited by, specifying, showing, implying, revealing, notifying, publishing, or registering.

[00127] In various embodiments, the decoder 200 may further include a feedback path coupled between an input of the convolutional decoding unit and a combination of extrinsic information of the circuit 204 and the extrinsic information of the convolution decoding unit, wherein the feedback path is configured to provide *a priori* information to the convolutional decoding unit for a subsequent iteration of decoding.

[00128] In one embodiment, the decoder 200 may further include a summation unit configured to combine the extrinsic information of the circuit 204 and the extrinsic information of the convolution decoder to provide the *a priori* information.

[00129] In the context of various embodiments, the phrase “*a priori* information” refers to information that is known beforehand prior to the start of decoding. The “*a priori* information” may be a parameter or latent variable.

[00130] Various embodiments provides simplifying the algorithm of the SPS-JSCD and PPS-JSCD by making assumptions on the respective header syntax, as follow.

SPS-JSCD

[00131] SPS-JSCD uses a trellis-based decoding. The shape of the trellis is dependent on the SPS header syntax structure. It is important to note that removing the following assumptions increases the complexity of the SPS-JSCD in accordance to various embodiments and does not alter the general principles in any way.

1. The profile adopted is Baseline. Hence the `profile_idc == 66`. Thus all the syntax following the `if (profile_idc)` statement (i.e., Table Row Nos. 2.11 to 2.28 of Table 2) are not considered. Nevertheless, the SPS-JSCD in accordance to various embodiments still attempts to decode the received `profile_idc` in order to simulate the fact that the SPS-JSCD does not know which profile was transmitted. Hence, some decoding errors in the `profile_idc` is to be expected, i.e., some other profiles may be detected instead. On the other hand, if the Baseline profile assumption is removed, then the SPS-JSCD in accordance to various embodiments has to take into account the syntax within the `if (profile_idc)` statement (i.e., Table Row Nos. 2.11 to 2.28 of Table 2) if the conditions are met. This may be done in the following manner. First and foremost, the `profile_idc` is decoded. If the decoded `profile_idc` satisfies the conditions of the `if (profile_idc)` statement, the SPS-JSCD proceeds to decode the syntax within the `if (profile_idc)` statement.
2. The `pic_order_cnt_type` is not equal to 1. Hence the syntax under the `if (pic_order_cnt_type == 1)` statement (i.e., Table Row Nos. 2.33 to 2.40 of Table 2) are not considered. This assumption is invoked in order to simplify the SPS-JSCD in accordance to various embodiments. However, these syntax may be easily incorporated into the SPS-JSCD at the expense of an increased complexity.
3. For simple video sequences, Video Usability Information (VUI) is usually not invoked. Hence it is assumed that no VUI is transmitted (i.e., Table Row Nos. 2.57 to 2.58 of Table 2) and that the `vui_parameters_present_flag` is always zero.

However, the VUI parameters may still be taken into account by the SPS-JSCD, again, at the expense of additional complexity.

4. The size of the video is assumed to be small, such as a QCIF format (176x144 pixels) up to a maximum size of 208x208 pixels. This assumption is to limit the maximum possible values of some of the decoding parameters, for example the syntax `pic_width_mbs_minus1`, so as to simplify the SPS-JSCD in accordance to various embodiments. Decoding a larger size video may be easily implemented by increasing the maximum possible values of the relevant decoding parameters at the expense of increased complexity.

[00132] Based on the above assumptions, the structure of the SPS header is simplified and this is reflected in Table 5, where the third column shows the number of bits that may be used to represent the syntax.

[00133] Table 5

<code>seq_parameter_set_data(){</code>	Descriptor	Number of bits (Min/Max values)
<code>profile_idc</code>	<code>u(8)</code>	8
<code>constraint_set0_flag</code>	<code>u(1)</code>	1
<code>constraint_set1_flag</code>	<code>u(1)</code>	1
<code>constraint_set2_flag</code>	<code>u(1)</code>	1
<code>constraint_set3_flag</code>	<code>u(1)</code>	1
<code>constraint_set4_flag</code>	<code>u(1)</code>	1
<code>constraint_set5_flag</code>	<code>u(1)</code>	1
<code>reserved_zero_2bits</code>	<code>u(2)</code>	2
<code>level_idc</code>	<code>u(8)</code>	8
<code>seq_parameter_set_id</code>	<code>ue(v)</code>	1
<code>log2_max_frame_num_minus4</code>	<code>ue(v)</code>	1,3,5,7 (0/12)

pic_order_cnt_type	ue(v)	1,3
if (pic_order_cnt_type == 0)		
log2_max_pic_order_cnt_lsb_minus4	ue(v)	1,3,5,7 (0/12)
max_num_ref_frames	ue(v)	1,3,5,7,9 (0/17)
gaps_in_frame_num_value_allowed_flag	u(1)	1
pic_width_in_mbs_minus1	ue(v)	1,3,5,7 (0/12)
pic_height_in_map_units_minus1	ue(v)	1,3,5,7 (0/12)
frame_mbs_only_flag	u(1)	1
if (!frame_mbs_only_flag)		
mb_adaptive_frame_field_flag	u(1)	1
direct_8x8_inference_flag	u(1)	1
frame_cropping_flag	u(1)	1
if (frame_cropping_flag){		
frame_crop_left_offset	ue(v)	1,3,5,7 (0/12)
frame_crop_right_offset	ue(v)	1,3,5,7 (0/12)
frame_crop_top_offset	ue(v)	1,3,5,7 (0/12)
frame_crop_bottom_offset	ue(v)	1,3,5,7 (0/12)
}		
vui_parameters_present_flag	u(1)	1
}		
trailing_bits()		1,2,3,4,5,6,7,8

[00134] The minimum and maximum values of the syntax associated with the ue(v) descriptor (i.e., the VLC syntax) are also given. These limits are important as they define the search space boundaries when decoding these syntax. Comparing with Table 2, it is observed that Table Row Nos. 2.11 to 2.28, Table Row Nos. 2.33 to 2.40, and Table Row Nos. 2.57 to 2.58 are removed. Also, because of assumption 4, the maximum number of bits used to represent pic_width_in_mbs_minus1, pic_height_in_map_units_minus1, frame_crop_left_offset, frame_crop_right_offset, frame_crop_top_offset, and

frame_crop_bottom_offset is limited to 7, or a maximum value of 12. Increasing this limit to a higher value does not affect the general algorithm of the SPS-JSCD in accordance to various embodiments. The search space needs only to be expanded according to the maximum value. Henceforth, in various embodiments, the design of the SPS-JSCD may be based on this simplified Table 5.

[00135] According to the simplified SPS data syntax in Table 5, the syntax from profile_idc to level_idc are all represented by FLC. This means that the total number of bits allocated for these syntax is a constant number, which is 24 bits in total. On the other hand, the syntax from seq_parameter_set_id to the end of the header are made up of either 1-bit FLC or VLC. Therefore, the total number of bits representing all of these syntax is a varying term. Furthermore, the number of syntax found in the header is also a varying term due to the three *if* statements as seen in Table 5. Because of this, the SPS-JSCD 400 in accordance to various embodiments may actually be subdivided into two sub-decoders as shown in FIG. 4. In FIG. 4, the SPS-JSCD 400 receives a SPS header 402 which is divided by a demultiplexer 404 (DeMUX) for feeding into the two sub-decoders 406, 408. The outputs of the respective sub-decoders are then combined using a multiplexer 410 (MUX) to give a decoded SPS header 412.

[00136] The first sub-decoder, referred to as the FLC SPS-JSCD 406 in accordance to various embodiments, handles the first 24 bits corresponding to the first 9 FLC syntax (from profile_idc to level_idc). However in the joint model (JM) codec, the seq_parameter_set_id syntax is always fixed to the value 0, which translates to a bit '1' (because it is an Exp-Golomb code). Hence this syntax is considered as a FLC and is decoded by the FLC SPS-JSCD 406. Because of this, the FLC SPS-JSCD 406 now handles the first 25 bits of the SPS header, whose corresponding syntax are shown in Table 6.

[00137] Table 6

Syntax	Descriptor	Remarks
profile_idc	u(8)	10 options; (44, 66, 77, 88, 100, 110, 122,

		244, 118, 128)
constraint_set0_flag	u(1)	Equals to 0 in JM
constraint_set1_flag	u(1)	Equals to 0 in JM
constraint_set2_flag	u(1)	Equals to 0 in JM
constraint_set3_flag	u(1)	Equals to 0 in JM
constraint_set4_flag	u(1)	Equals to 0 in JM
constraint_set5_flag	u(1)	Equals to 0 in JM
reserved_zero_2bits	u(2)	Equals to 00 in JM
level_idc	u(8)	16 options; (9, 10, 11, 12, 13, 20, 21, 22, 30, 31, 32, 40, 41, 42, 50, 51)
seq_parameter_set_id	ue(v)	0 to 31, inclusive. Equals to 0 in JM

[00138] The second sub-decoder, referred to as the VLC SPS-JSCD 408 in accordance to various embodiments, handles the remaining bits, which is made up of a combination of 1-bit FLC and VLC. Unlike the FLC SPS-JSCD 406, the number of syntax to be decoded by the VLC SPS-JSCD 408 is not a constant. However, the number of bits is fixed, which may be obtained by subtracting 25 bits (to be allocated to FLC SPS-JSCD 406) from the total number of bits in the SPS header 402.

[00139] FLC SPS-JSCD

[00140] As mentioned previously, the first 25 bits of the received SPS header are passed to the FLC SPS-JSCD in accordance to various embodiments. The index of these bits is denoted as $t = 0, \dots, 24$. Although there are exactly 10 syntax to be decoded by the FLC SPS-JSCD, as shown in Table 6, in essence only 2 of these syntax can be decoded and corrected by the FLC SPS-JSCD, namely the `profile_idc` and the `level_idc`. This is because all the other syntax are either predetermined by the encoder (i.e. *a priori* information) or they are 1-bit boolean FLC, which cannot be corrected. In this case, the JM encoder actually fixed the values of all these syntax such that this *a priori* information

may be made used of to aid the decoding. Thus, in actuality the FLC SPS-JSCD only decodes the `profile_idc` and the `level_idc`.

[00141] There are ten possible values for `profile_idc` and 16 possible values for `level_idc`, as listed in Table 7 and Table 8, respectively.

[00142] Table 7

Index	profile_idc	Description	8-bit sequence
I ₁	44	CAVLC 4:4:4 Intra	0 0 1 0 1 1 0 0
I ₂	66	Baseline	0 1 0 0 0 0 1 0
I ₃	77	Main	0 1 0 0 1 1 0 1
I ₄	88	Extended	0 1 0 1 1 0 0 0
I ₅	100	High	0 1 1 0 0 1 0 0
I ₆	110	High 10 or High 10 Intra	0 1 1 0 1 1 1 0
I ₇	122	High 4:2:2 Predictive or High 4:2:2 Intra	0 1 1 1 1 0 1 0
I ₈	244	High 4:4:4 or High 4:4:4 Intra	1 1 1 1 0 1 0 0
I ₉	118	Multiview High	0 1 1 1 0 1 1 0
I ₁₀	128	Stereo High	1 0 0 0 0 0 0 0

[00143] Table 8

Index	level_idc	Description	8-bit sequence
I ₁	9	Supports QCIF format and below with 380160 samples/sec	0 0 0 0 1 0 0 1
I ₂	10	Supports QCIF format and below with 380160 samples/sec	0 0 0 0 1 0 1 0
I ₃	11	Supports CIF and below. 768000 samples/sec	0 0 0 0 1 0 1 1
I ₄	12	Supports CIF and below. 1536000 samples/sec	0 0 0 0 1 1 0 0
I ₅	13	Supports CIF and below. 3041280 samples/sec	0 0 0 0 1 1 0 1
I ₆	20	Supports CIF and below. 3041280 samples/sec	0 0 0 1 0 1 0 0
I ₇	21	Supports HHR formats. Enables Interlace	0 0 0 1 0 1 0 1

		support. 5068800 samples/sec	
I_8	22	Supports SD/4CIF formats. Enables Interlace support. 5184000 samples/sec	0 0 0 1 0 1 1 0
I_9	30	Supports SD/4CIF formats. Enables Interlace support. 10368000 samples/sec	0 0 0 1 1 1 1 0
I_{10}	31	Supports 720p HD format. Enables Interlace support. 27648000 samples/sec	0 0 0 1 1 1 1 1
I_{11}	32	Supports SXGA format. Enables Interlace support. 55296000 samples/sec	0 0 1 0 0 0 0 0
I_{12}	40	Supports 2Kx1K format. Enables Interlace support. 62914560 samples/sec	0 0 1 0 1 0 0 0
I_{13}	41	Supports 2Kx1K format. Enables Interlace support. 62914560 samples/sec	0 0 1 0 1 0 0 1
I_{14}	42	Supports 2Kx1K format. Frame coding only. 125829120 samples/sec	0 0 1 0 1 0 1 0
I_{15}	50	Supports 3672x1536 format. Frame coding only. 150994944 samples/sec	0 0 1 1 0 0 1 0
I_{16}	51	Supports 4096x2304 format. Frame coding only. 251658240 samples/sec	0 0 1 1 0 0 1 1

[00144] Their corresponding binary sequences are also shown in the last column of the Tables 7 and 8. This 8-bit sequence is derived based on the conventional decimal to binary conversion. Notice that for the level_idc, the first two MSB are always zero. Hence, this may be used as *a priori* knowledge and it is not necessary to decode them. For generality the profile_idc and level_idc are referred to as a FLC_N , where $N = 8$ for profile_idc and $N = 6$ for level_idc, respectively. The N -bit sequence of the I_k -th FLC_N is denoted as $I_k = [I_k(0)...I_k(N-1)]$, where the index $k = 1, \dots, 10$ for profile_idc and $k = 1, \dots, 16$ for level_idc. For example, according to Table 7, the bit sequence of I_7 of the profile_idc is [0 1 1 1 1 0 1 0].

[00145] The soft inputs of the received SPS header to the SPS-JSCD are typically represented in terms of the log-likelihood ratio (LLR), of which the absolute value gives the sign of the bit while the amplitude gives the probability of a correct decision. For example, the LLR of the bit u_k is given by

$$L(u_k) = \ln \left(\frac{P(u_k = 0)}{P(u_k = 1)} \right) \quad (1)$$

[00146] Suppose the received soft inputs for FLC_N is denoted by $y = [L(y_0) \dots L(y_{N-1})]$, where $L(y_N)$ is given in Equation (1). The LLR for each I_k -th FLC_N, given the received soft inputs, is then given by

$$L(I_k|y) = \sum_{n=0}^{N-1} \frac{1}{2} [1 - 2I_k(n)] L(y_n) \quad (2)$$

[00147] The one corresponding to the largest $L(I_k|y)$ is the decoded profile_idc and level_idc. However, the soft outputs are obtained in order to enable further iterative processing. The soft output corresponding to bit u_n of FLC_N, where $n = 0, \dots, N-1$, is given by

$$L(u_n) = \frac{\ln \left(\sum_{I_k(n)=0} e^{L(I_k|y)} \right)}{\ln \left(\sum_{I_k(n)=1} e^{L(I_k|y)} \right)} \quad (3)$$

[00148] The extrinsic soft information of the bit u_n is then given by

$$L_e(u_n) = L(u_n) - L(y_n) \quad (4)$$

[00149] There is a total of 14 of these extrinsic soft information, which corresponds to the 8 bits of the profile_idc ($t = 0, \dots, 7$) and the 6 bits of the level_idc ($t = 18, \dots, 23$). As mentioned previously, the remaining 11 bits ($t = 8, \dots, 17, 24$) are either fixed by the JM codec or fixed by the nature of the bit sequences (i.e., the 2 MSB (most significant bit) of level_idc). Hence their extrinsic soft information is given by

$$L_e(u_t) = (1 - 2b_t)E_{\max} \quad (5)$$

where b_t is the fixed value of the considered bit and $E_{\max}$ is the maximum extrinsic soft information. Ideally, $E_{\max} = \infty$ or a very large value. However, because of the risk of overflow during simulation, $E_{\max}$ is set to a much lower value. Usually, $E_{\max} = 100$ is suffice. On the other hand, if the values of these syntax are not fixed by the encoder, then their extrinsic information is zero because these syntax cannot be decoded by the FLC SPS-JSCD.

[00150] In summary, the output of the FLC SPS-JSCD in accordance to various embodiments may comprise or consist of the extrinsic information corresponding to the first 25 bits, as given in Equation (4) and Equation (5), as well as the equivalent decoded binary values, which are derived from the decoded profile_idc and level_idc as well as the *a priori* knowledge set by the JM encoder.

[00151] VLC SPS-JSCD

[00152] According to Table 5, the number of syntax that are handled by the VLC SPS-JSCD ranged from 11 to a maximum of 17. Note that the sequence of trailing bits is considered as 1 syntax because the VLC SPS-JSCD decodes the trailing bits. These syntax are shown in Table 9.

[00153] Table 9

Syntax for VLC SPS-JSCD	Descriptor	Number of bits (Min/Max values)
log2_max_frame_num_minus4	ue(v)	1,3,5,7 (0/12)
pic_order_cnt_type	ue(v)	1,3
if (pic_order_cnt_type == 0)		
log2_max_pic_order_cnt_lsb_minus4	ue(v)	1,3,5,7 (0/12)
max_num_ref_frames	ue(v)	1,3,5,7,9 (0/17)
gaps_in_frame_num_value_allowed_flag	u(1)	1
pic_width_in_mbs_minus1	ue(v)	1,3,5,7 (0/12)
pic_height_in_map_units_minus1	ue(v)	1,3,5,7 (0/12)
frame_mbs_only_flag	u(1)	1
if (!frame_mbs_only_flag)		
mb_adaptive_frame_field_flag	u(1)	1
direct_8x8_inference_flag	u(1)	1
frame_cropping_flag	u(1)	1
if (frame_cropping_flag){		
frame_crop_left_offset	ue(v)	1,3,5,7 (0/12)
frame_crop_right_offset	ue(v)	1,3,5,7 (0/12)
frame_crop_top_offset	ue(v)	1,3,5,7 (0/12)
frame_crop_bottom_offset	ue(v)	1,3,5,7 (0/12)
}		
vui_parameters_present_flag	u(1)	1
trailing_bits()		

[00154] As mentioned previously, the receiver, for example, the receiver 118 of FIG. 1, has no knowledge on the number of syntax that are in the SPS header. Hence for a trellis-based decoding algorithm, the lack of this knowledge presents a problem as the shape of the trellis is unknown.

[00155] To illustrate this problem with an example, the first three syntax of Table 9 are consider, where `log2_max_frame_num_minus4` and `pic_order_cnt_type` are the first two syntax. The third syntax may either be `max_num_ref_frames`, if `pic_order_cnt_type` is equal to 2 (recall from assumption 2 that `pic_order_cnt_type=1` is not considered here), or it may be `log2_max_pic_order_cnt_lsb_minus4` if `pic_order_cnt_type` is equal to 0.

[00156] The trellis for both of these cases are shown in FIG. 5 and FIG. 6, respectively. Note that each branch in the trellis may contain more than 1 parallel branch, which are not drawn here in order not to clutter the trellis. For instance, the first syntax is of an Exp-Golomb type. Hence from bit 0 to bit 5, there are 4 parallel branches because there are 4 possible bit sequence combinations having a length of 5 bits, as tabulated in Table 4.

[00157] It is obvious to see that both sets of trellis (FIGs. 5 and 6) are very different, due to the *if* statement. The first syntax is common to both cases and so the trellis for the first syntax is the same. For second syntax (i.e., `pic_order_cnt_type`), its value may be 0 or 2. The former is represented by 1 bit while the latter is represented by 3 bits. It is observed that the trellis for the second syntax is slightly different. When `pic_order_cnt_type=0`, the next syntax is `log2_max_pic_order_cnt_lsb`, which may range from 1 bit to 7 bits. However, if `pic_order_cnt_type=2`, `log2_max_pic_order_cnt_lsb` is not considered, instead `max_num_ref_frames` is the next syntax, which may range from 1 bit to 9 bits. It is observed that now the trellis of both cases begins to diverge. It is not difficult to imagine that if the remaining syntax are considered, two totally different trellis are derived. Thus it is not possible to decode the sequence using a trellis if the number of syntax and the description of the syntax are not known. Analogously, one can think of the trellis of a convolutional code of constraint length 3 and 4. Both have totally different trellis and it is not possible to use the trellis corresponding to the constraint length 3 code to decode codes encoded with constraint length 4 and vice versa. The decoder needs to have prior knowledge so that the correct trellis is used.

[00158] In order to solve this problem, a scenario-based VLC SPS-JSCD, for example, the VLC SPS-JSCD 408 of FIG. 4, is considered, whereby each scenario takes care of a specific *if* statement condition combination. Since there are 3 *if* statements in Table 9,

there are 8 of such scenarios. These scenarios are listed in Table 10 to Table 17, together with the number of syntax corresponding to the scenario as well as the total number of bits in the SPS header.

[00159] Table 10

Scenario 1	Descriptor	Number of bits (Min/Max values)
log2_max_frame_num_minus4	ue(v)	1,3,5,7 (0/12)
pic_order_cnt_type	ue(v)	3 (2/2)
max_num_ref_frames	ue(v)	1,3,5,7,9 (0/17)
gaps_in_frame_num_value_allowed_flag	u(1)	1 (0/1)
pic_width_in_mbs_minus1	ue(v)	1,3,5,7 (0/12)
pic_height_in_map_units_minus1	ue(v)	1,3,5,7 (0/12)
frame_mbs_only_flag	u(1)	1 (1/1)
direct_8x8_inference_flag	u(1)	1 (0/1)
frame_cropping_flag	u(1)	1 (0/0)
vui_parameters_present_flag	u(1)	1 (0/0)
trailing_bits()		
Number of syntax for VLC SPS-JSCD: 11; Number of bits in SPS header: 40 – 64.		

[00160] Table 11

Scenario 2	Descriptor	Number of bits (Min/Max values)
log2_max_frame_num_minus4	ue(v)	1,3,5,7 (0/12)
pic_order_cnt_type	ue(v)	1 (0/0)
log2_max_pic_order_cnt_lsb_minus4	ue(v)	1,3,5,7 (0/12)
max_num_ref_frames	ue(v)	1,3,5,7,9 (0/17)

gaps_in_frame_num_value_allowed_flag	u(1)	1 (0/1)
pic_width_in_mbs_minus1	ue(v)	1,3,5,7 (0/12)
pic_height_in_map_units_minus1	ue(v)	1,3,5,7 (0/12)
frame_mbs_only_flag	u(1)	1 (1/1)
direct_8x8_inference_flag	u(1)	1 (0/1)
frame_cropping_flag	u(1)	1 (0/0)
vui_parameters_present_flag	u(1)	1 (0/0)
trailing_bits()		
Number of syntax for VLC SPS-JSCD: 12; Number of bits in SPS header: 40 – 72.		

[00161] Table 12

Scenario 3	Descriptor	Number of bits (Min/Max values)
log2_max_frame_num_minus4	ue(v)	1,3,5,7 (0/12)
pic_order_cnt_type	ue(v)	3 (2/2)
max_num_ref_frames	ue(v)	1,3,5,7,9 (0/17)
gaps_in_frame_num_value_allowed_flag	u(1)	1 (0/1)
pic_width_in_mbs_minus1	ue(v)	1,3,5,7 (0/12)
pic_height_in_map_units_minus1	ue(v)	1,3,5,7 (0/12)
frame_mbs_only_flag	u(1)	1 (0/0)
mb_adaptive_frame_field_flag	u(1)	1 (0/1)
direct_8x8_inference_flag	u(1)	1 (0/1)
frame_cropping_flag	u(1)	1 (0/0)
vui_parameters_present_flag	u(1)	1 (0/0)
trailing_bits()		

Number of syntax for VLC SPS-JSCD: 12;
 Number of bits in SPS header: 40 – 72.

[00162] Table 13

Scenario 4	Descriptor	Number of bits (Min/Max values)
log2_max_frame_num_minus4	ue(v)	1,3,5,7 (0/12)
pic_order_cnt_type	ue(v)	3 (2/2)
max_num_ref_frames	ue(v)	1,3,5,7,9 (0/17)
gaps_in_frame_num_value_allowed_flag	u(1)	1 (0/1)
pic_width_in_mbs_minus1	ue(v)	1,3,5,7 (0/12)
pic_height_in_map_units_minus1	ue(v)	1,3,5,7 (0/12)
frame_mbs_only_flag	u(1)	1 (1/1)
direct_8x8_inference_flag	u(1)	1 (0/1)
frame_cropping_flag	u(1)	1 (1/1)
frame_crop_left_offset	ue(v)	1,3,5,7 (0/12)
frame_crop_right_offset	ue(v)	1,3,5,7 (0/12)
frame_crop_top_offset	ue(v)	1,3,5,7 (0/12)
frame_crop_bottom_offset	ue(v)	1,3,5,7 (0/12)
vui_parameters_present_flag	u(1)	1 (0/0)
trailing_bits()		
Number of syntax for VLC SPS-JSCD: 15; Number of bits in SPS header: 48 – 96.		

[00163] Table 14

Scenario 5	Descriptor	Number of bits (Min/Max values)
log2_max_frame_num_minus4	ue(v)	1,3,5,7 (0/12)
pic_order_cnt_type	ue(v)	1 (0/0)
log2_max_pic_order_cnt_lsb_minus4	ue(v)	1,3,5,7 (0/12)
max_num_ref_frames	ue(v)	1,3,5,7,9 (0/17)
gaps_in_frame_num_value_allowed_flag	u(1)	1 (0/1)
pic_width_in_mbs_minus1	ue(v)	1,3,5,7 (0/12)
pic_height_in_map_units_minus1	ue(v)	1,3,5,7 (0/12)
frame_mbs_only_flag	u(1)	1 (0/0)
mb_adaptive_frame_field_flag	u(1)	1 (0/1)
direct_8x8_inference_flag	u(1)	1 (0/1)
frame_cropping_flag	u(1)	1 (0/0)
vui_parameters_present_flag	u(1)	1 (0/0)
trailing_bits()		
Number of syntax for VLC SPS-JSCD: 13; Number of bits in SPS header: 40 – 72.		

[00164] Table 15

Scenario 6	Descriptor	Number of bits (Min/Max values)
log2_max_frame_num_minus4	ue(v)	1,3,5,7 (0/12)
pic_order_cnt_type	ue(v)	1 (0/0)
log2_max_pic_order_cnt_lsb_minus4	ue(v)	1,3,5,7 (0/12)
max_num_ref_frames	ue(v)	1,3,5,7,9 (0/17)
gaps_in_frame_num_value_allowed_flag	u(1)	1 (0/1)

pic_width_in_mbs_minus1	ue(v)	1,3,5,7 (0/12)
pic_height_in_map_units_minus1	ue(v)	1,3,5,7 (0/12)
frame_mbs_only_flag	u(1)	1 (1/1)
direct_8x8_inference_flag	u(1)	1 (0/1)
frame_cropping_flag	u(1)	1 (1/1)
frame_crop_left_offset	ue(v)	1,3,5,7 (0/12)
frame_crop_right_offset	ue(v)	1,3,5,7 (0/12)
frame_crop_top_offset	ue(v)	1,3,5,7 (0/12)
frame_crop_bottom_offset	ue(v)	1,3,5,7 (0/12)
vui_parameters_present_flag	u(1)	1 (0/0)
trailing_bits()		
Number of syntax for VLC SPS-JSCD: 16; Number of bits in SPS header: 48 – 104.		

[00165] Table 16

Scenario 7	Descriptor	Number of bits (Min/Max values)
log2_max_frame_num_minus4	ue(v)	1,3,5,7 (0/12)
pic_order_cnt_type	ue(v)	3 (2/2)
max_num_ref_frames	ue(v)	1,3,5,7,9 (0/17)
gaps_in_frame_num_value_allowed_flag	u(1)	1 (0/1)
pic_width_in_mbs_minus1	ue(v)	1,3,5,7 (0/12)
pic_height_in_map_units_minus1	ue(v)	1,3,5,7 (0/12)
frame_mbs_only_flag	u(1)	1 (0/0)
mb_adaptive_frame_field_flag	u(1)	1 (0/1)
direct_8x8_inference_flag	u(1)	1 (0/1)
frame_cropping_flag	u(1)	1 (1/1)

frame_crop_left_offset	ue(v)	1,3,5,7 (0/12)
frame_crop_right_offset	ue(v)	1,3,5,7 (0/12)
frame_crop_top_offset	ue(v)	1,3,5,7 (0/12)
frame_crop_bottom_offset	ue(v)	1,3,5,7 (0/12)
vui_parameters_present_flag	u(1)	1 (0/0)
trailing_bits()		
Number of syntax for VLC SPS-JSCD: 16; Number of bits in SPS header: 48 – 96.		

[00166] Table 17

Scenario 8	Descriptor	Number of bits (Min/Max values)
log2_max_frame_num_minus4	ue(v)	1,3,5,7 (0/12)
pic_order_cnt_type	ue(v)	1 (0/0)
log2_max_pic_order_cnt_lsb_minus4	ue(v)	1,3,5,7 (0/12)
max_num_ref_frames	ue(v)	1,3,5,7,9 (0/17)
gaps_in_frame_num_value_allowed_flag	u(1)	1 (0/1)
pic_width_in_mbs_minus1	ue(v)	1,3,5,7 (0/12)
pic_height_in_map_units_minus1	ue(v)	1,3,5,7 (0/12)
frame_mbs_only_flag	u(1)	1 (0/0)
mb_adaptive_frame_field_flag	u(1)	1 (0/1)
direct_8x8_inference_flag	u(1)	1 (0/1)
frame_cropping_flag	u(1)	1 (1/1)
frame_crop_left_offset	ue(v)	1,3,5,7 (0/12)
frame_crop_right_offset	ue(v)	1,3,5,7 (0/12)
frame_crop_top_offset	ue(v)	1,3,5,7 (0/12)
frame_crop_bottom_offset	ue(v)	1,3,5,7 (0/12)

vui_parameters_present_flag	u(1)	1 (0/0)
trailing_bits()		
Number of syntax for VLC SPS-JSCD: 17; Number of bits in SPS header: 48 – 104.		

[00167] This total number of bits includes the first 25 bits that are assigned to the FLC SPS-JSCD 406. Note again that the total number of bits in the SPS header is a multiple of 8. In Tables 10 to 17, under the (Min/Max values) column, (x/x) denotes that x is the one and only possible value for the associated syntax.

[00168] Each scenario has a fixed number of syntax and the description of these syntax is fixed as well. Hence each scenario has its own specific trellis, and may be decoded using the BCJR MAP or LogMAP algorithm, which is understood and appreciated to be, for example, the typical BCJR MAP or LogMAP algorithm.

[00169] As an example, the LogMAP algorithm is used in the simulation. More explicitly, the VLC SPS-JSCD (e.g., the VLC SPS-JSCD 408 of FIG. 4) has 8 parallel LogMAP decoders performing the decoding on each of these 8 scenario-specific trellis as shown in FIG. 7. In FIG. 7, an input 700 is fed into 8 parallel LogMAP decoders 702, the outputs of which enters an unit 704 for selecting the decoder with the metric from the 8 parallel LogMAP decoders 702 to provide an output 706.

[00170] The trellis having the highest metric after the decoding is deemed as the desired trellis and the output 706 from this trellis decoding is used. The output 706 may comprise or consist of the decoded SPS header pertaining to the winning scenario as well as the extrinsic information.

[00171] It may be useful to modify the trellis so that the decoding can be performed more efficiently. In conventional trellis terminology, the y-axis of the trellis usually symbolizes the state. The total number of states ultimately determines the complexity of the decoder. Hence it is often desirable to keep the number of states as small as possible. Upon examining the trellis of FIG. 5 and FIG. 6 closely, it is observed that the y-axis

represents the bit index, which is equivalent to the state. Thus, the complexity of the SPS-JSCD trellis is directly proportional to the total number of bits. Furthermore, many 'empty' nodes in the trellis (i.e., nodes without any branches) may be observed. This is an inefficient way of representing a trellis. It is possible to reduce the complexity and improve the efficiency by following a transformation rule. Based on this transformation rule, the bit index n in the y-axis is transformed to a state v according to

$$v = n - xl_{\min} \quad (6)$$

where x denotes the current syntax index of the x-axis and $l_{\min}$ is the minimum length of the VLC, which is 1 in this case. Using this transformation rule, the transformed trellis of FIG. 6 is shown in FIG. 8, where the y-axis is changed from the bit index to the state.

[00172] The transformed trellis (in FIG. 8) is now more compact, which makes the decoding more efficient. More importantly, all branches end at an even-number state, unlike the original trellis whereby the branches may either end at an odd or an even number bit index. This is due to the fact that all the syntax, except for the trailing bits, have an odd number of bits and by applying the transformation rule of Equation (6), the resulting state is always an even number. This is advantageous because now the decoding needs only to keep track of the even-number states, which in turn reduces the complexity of the decoder. On the other hand, the number of trailing bits can be even or odd. But because it is the last syntax, all branches terminate at the same state, which the decoder needs to track in any case.

[00173] Based on this transformed trellis, the LogMAP algorithm is performed as follows. In the LogMAP algorithm for the SPS-JSCD, the forward and backward recursive calculation may be employed. The forward recursive calculation of the $\alpha_k(s)$ LLR values is given by

$$\alpha_k(s) = \log \left[\sum_{s',m} \exp(\alpha_{k-1}(s') + \gamma_{k,m}(s',s)) \right] \quad (7)$$

where k is the current syntax index, as denoted by the x-axis, and s' and s denote the previous state and current state, respectively. The index m is required here because for a given syntax index k , the state transition from s' to s may contain several branches and m refers to the m -th branch.

[00174] Similarly, the backward recursive calculation of the $\beta_{k-1}(s)$ LLR values is given by

$$\beta_{k-1}(s') = \log \left[\sum_{s,m} \exp(\beta_k(s) + \gamma_{k,m}(s',s)) \right] \quad (8)$$

[00175] The initialization is such that $\alpha_0(0) = 0$ and $\beta_N(\bar{s}) = 0$, where N is the total number of syntax and $\bar{s}$ is the termination state, while all other values are set to a value close to negative infinity.

[00176] For both Equation (7) and Equation (8), the summation may be simplified by using the following Jacobian algorithm

$$\log[\exp(x_1) + \exp(x_2)] = \max(x_1, x_2) + \log[1 + \exp(-|x_1 - x_2|)] \quad (9)$$

[00177] Finally, the transition probability LLR values $\gamma_{k,m}(s',s)$ in Equation (7) and Equation (8) may be calculated from the received LLR values obtained from the channel as well as the *a priori* information, if any.

[00178] Let the total number of branches linking the previous state s' to the current state s be M . In this case, $\gamma_k(s',s)$ for the m -th branch having a length n_m (i.e., the number of bits) is calculated as

$$\gamma_{k,m}(s',s) = \sum_{n=1}^{n_m} [L(y_n|x_n) + L_\alpha(u_n)] + R_m \quad (10)$$

where $L(y_n|x_n)$ is the conditional LLR of the received soft bit y_n given the considered bit x_n of that branch, $L_\alpha(u_n)$ is the *a priori* LLR information of the considered bit, and R_m is the so-called bias.

[00179] The bias is essential when calculating the transition probabilities involving VLC and is derived as

$$R_m = \log \frac{2^{-n_m}}{\sum_{m=1}^M 2^{-n_m}} \quad (11)$$

[00180] From Equation (11), it is observed that the bias is different for different length codes. However, if all the codes are of a fixed length, then the bias is the same for all codes and it may be ignored.

[00181] The final LLR value of the n -th bit, given the received SPS header y is given by

$$L(u_n|y) = \log \frac{\sum_{(s',s), m \Rightarrow u_k=0} \exp(\alpha_{k-1}(s') + \gamma_{k,m}(s',s) + \beta_k(s))}{\sum_{(s',s), m \Rightarrow u_k=1} \exp(\alpha_{k-1}(s') + \gamma_{k,m}(s',s) + \beta_k(s))} \quad (12)$$

[00182] The extrinsic information is then obtained by subtracting the inputs from $L(u_n|y)$, i.e.

$$L_e(u_n) = L(u_n|y) - L(y_n) - L_\alpha(u_n) \quad (13)$$

[00183] It is noted that the LogMAP decoder is only able to decode and possibly correct any bit errors occurring in the VLC-based syntax because these codes possess a certain amount of redundancies. It is unable to detect bit errors for the FLC boolean-based syntax such as `direct_8x8_inference_flag`.

[00184] When it comes to selecting the best scenario out of the 8, one way of doing it is to compare the sum of $L(u_n|y)$ for each scenario and then select the one having the highest value. While this method does work, it is noticed that sometimes the wrong scenario may be chosen. This is because the SPS header is essentially not an error correction code. Hence even with the different scenarios, the decoded outputs from these scenarios are quite similar.

[00185] For example, if scenario 2 (see Table 11) and scenario 5 (see Table 14), the first 7 syntax are the same, which constitutes more than half of the total syntax. Hence output LLR values of these first 7 syntax from the two scenarios are almost similar. The same phenomenon may be observed between scenario 2 and scenario 6 (see Table 15) as well as between other scenarios. Thus, when comparing the accumulated sum of the LLR outputs, the margin of error is considerably (very) small.

[00186] Most of the scenarios have a different number of syntax. But the total number of bits considered is always fixed. Hence according to the transformation rule of Equation (6), the trellis of these scenarios have a different termination state. A better distinction may be achieved if the LLR value of the different scenarios at the termination state are compared. This selection criteria is used in the simulations.

[00187] The block diagram of the receiver configuration with an SPS-JSCD 900 is shown in FIG. 9, where perfect synchronization is assumed, i.e., no synchronization is required. The SPS-JSCD 900 has prior knowledge of the exact location of the SPS header in the packet. The SPS-JSCD 900 is invoked after the turbo decoder 902 has completed its decoding procedure, as demonstrated by the SPS-JSCD activation switch 904 shown in FIG. 9.

[00188] The number of iterations the turbo decoder took to complete its decoding is denoted by N_{inner} . Each turbo decoder iteration is defined as an *inner* iteration. The input to the SPS-JSCD may comprise or consist of the LLR sum of the received systematic soft bits from the channel as well as the extrinsic information 906, 908 from both the 1st and 2nd constituent RSC decoders 910, 912 of the turbo decoder 902.

[00189] After the SPS-JSCD 900 has finished its decoding, the extrinsic information from its output 914, as given in Equation (13), is added to the extrinsic information 908 obtained from the 2nd constituent RSC decoder 912 by an adder 916 and the resultant output from the summation 918 is passed to the 1st constituent RSC decoder 910. The rationale behind this is to improve the *a priori* information going into the 1st constituent RSC decoder 910 during the next set of inner iterations. Each time the SPS-JSCD 900 is activated and new extrinsic information from the SPS-JSCD 914 is feedback to the turbo decoder 902, this is counted as one *outer* iteration. The number of outer iterations is denoted by N_{outer} . Note that the same SPS-JSCD extrinsic information 914 may be used for each and every inner iteration of the turbo decoding. This extrinsic information 914 is initialized to zero at the 0th outer iteration. The performance and complexity of the receiver 920 is highly dependent on the two parameters N_{inner} and N_{outer} . As such, a receiver configuration with the designation $RC_{SPS-JSCD}(N_{inner}, N_{outer})$ is denoted, where the subscript SPS-JSCD specifies that the receiver configuration (RC) only has a SPS-JSCD 900 with perfect synchronization.

PPS-JSCD

[00190] PPS-JSCD decodes the PPS header based on its syntax structure. The PPS-JSCD uses a trellis-based decoding, similar to VLC SPS-JSCD.

[00191] Similar assumptions may be made for the PPS header syntax to simplify the PPS-JSCD in accordance to various embodiments.

1. The syntax indicated by Table Row Nos. 3.6 to 3.24 in Table 3 corresponds to the case when a picture frame is represented by more than one slice. It is assumed that there is only one slice per frame. Hence, those syntax in Table Row Nos. 3.6 to 3.24 is not used and they are considered in the PPS-JSCD.
2. The syntax indicated by Table Row Nos. 3.35 to 3.48 in Table 3 are applicable to a more advanced video encoding process. Since basic video encoding is used,

these syntax are not included in the PPS-JSCD in accordance to various embodiments.

[00192] Hence, after taking these two assumptions into account, the simplified H.264 PPS header data syntax is shown in Table 18.

[00193] Table 18

pic_parameter_set_rbsp(){	Descriptor	Remarks
pic_parameter_set_id	ue(v)	Equals to either 0,1 or 2 in JM encoder (1-3 bits)
seq_parameter_set_id	ue(v)	Equals to 0 in JM encoder (1 bit)
entropy_coding_mode_flag	u(1)	
bottom_field_pic_order_in_frame_present_flag	u(1)	
num_slice_groups_minus1	ue(v)	Equals to 0 from assumption 1 (1 bit)
num_ref_idx_10_default_active_minus1	ue(v)	0 to 31 (1-11 bits)
num_ref_idx_11_default_active_minus1	ue(v)	0 to 31 (1-11 bits)
weighted_red_flag	u(1)	
weighted_bired_idc	u(2)	0 to 2
pic_init_qp_minus26	se(v)	Equals to 0 in JM encoder (1 bit)
pic_init_qs_minus26	se(v)	Equals to 0 in JM encoder (1 bit)
chroma_qp_index_offset	se(v)	-12 to 12 (1-9 bits)
deblocking_filter_control_present_flag	u(1)	

constrained_intra_pred_flag	u(1)	
redundant_pic_cnt_present_flag	u(1)	
rbsp_trailing_bits()		(1-8 bits)
}		

[00194] The PPS-JSCD in accordance to various embodiments is designed based on this simplified PPS header data syntax. In a way, these assumptions may be considered as *a priori* information, where the JSCD has an idea beforehand of what type of video it is processing and so adapting its decoding approach accordingly.

[00195] Looking at the PPS header syntax of Table 18, it comprises a combination of FLC, as denoted by u(n), and VLC, as denoted by ue(v) or se(v). This arrangement is similar, but not identical, to any one of the SPS header scenarios syntax given in Tables 10 to 17. Hence the same trellis-based decoding algorithm employed for the VLC SPS-JSCD 408 of FIG. 4, as discussed previously, may also be used to decode the PPS header.

[00196] The trellis for the PPS-JSCD may be very different from the VLC SPS-JSCD 408 due to the different FLC/VLC arrangement. But computation wise, it is exactly the same as that of the VLC SPS-JSCD 408. Hence, all the equations for calculating the branch metrics and the extrinsic information given previously under the VLC SPS-JSCD 408 framework are applicable to the PPS-JSCD.

[00197] Lastly, from Table 18 it is noted that there are no *if* statements in the PPS header, unlike that of the SPS header. So basically there is only one scenario for the PPS header, which makes the PPS-JSCD less complex.

[00198] If the receiver, for example, the receiver 118 of FIG. 1, has prior knowledge about the exact location of the PPS header, then the PPS-JSCD is suffice to decode the header. In this case, the receiver configuration is no different from that of the receiver configuration with only the SPS-JSCD and perfect synchronization, as shown in FIG. 9.

[00199] The receiver configuration 1000 with only the PPS-JSCD 1002 with perfect synchronization is shown in FIG. 10. Following the previous notation, this configuration is referred to as $RC_{PPS-JSCD} (N_{inner}, N_{outer})$.

[00200] It is possible to have both SPS and PPS headers residing within a packet. In such situations, both the SPS-JSCD and PPS-JSCD, for example, the SPS-JSCD 900 of FIG. 9 and the PPS-JSCD 1002 of FIG. 10, may be activated. Again, it is assumed that the receiver has prior knowledge about the exact locations of the SPS and PPS headers. Hence, both JSCDs 1100 may be activated without any synchronization. The receiver configuration 1102 is shown in FIG. 11 with the notation $RC_{JSCDs}(N_{inner}, N_{outer})$.

[00201] In various embodiments, the decoder 200 of FIG. 2 may further include a synchronizer configured to determine a presence of a complete network abstract layer (NAL) unit in the video data packet, wherein the complete NAL unit comprises a NAL unit sandwiched between two prefixes, and to determine the location of the encoded header in the video data packet and/or the length of the encoded header from the video data packet.

[00202] FIG. 12 shows a complete NAL unit 1200 which includes two prefixes 1202, 1204 sandwiching a NAL unit identifier 1206 followed by a raw byte sequence payload (RBSP) 1208. The prefix 1202 and part of the NAL unit identifier 1206 make a front synchronization word (FSW) 1210. A remainder part of the NAL unit identifier 1206 is referred to as header type (HT) 1212. The prefix 1204 makes a rear synchronization word (RSW) 1214.

[00203] For example, the prefix 1202, the NAL unit identifier 1206 and the RBSP 1208 may be the prefix 304, the NAL unit identifier 306 and the RBSP 308 of FIG. 3, respectively.

[00204] In various embodiments, the synchronizer may include a correlation unit configured to generate a plurality of synchronization patterns, to correlate each of the synchronization patterns with soft information of the video data packet to obtain respective correlation metrics and to compare each of the respective correlation metrics against a predefined threshold.

[00205] In various embodiments, the correlation unit may be configured to correlate the each of the synchronization patterns with the soft information of the video data packet by sliding the each of the synchronization patterns across the video data packet to search

for a similar pattern in the packet. The correlation unit may be configured to perform sliding in a byte-by-byte format.

[00206] In various embodiments, the synchronizer may further include a sorting unit configured to sort the each of the plurality of synchronization patterns into different groups based on header types; and a selection unit configured to determine for each group, a synchronization pattern that most likely comprises a header corresponding to the header type of that group.

[00207] In various embodiments, the selection unit may further be configured to utilize the circuit 204 (FIG. 2) to decode the respective encoded headers of the sorted synchronization patterns for each group.

[00208] Synchronization is achieved with the aid of predefined synchronization words, which are normally inserted periodically into the transmitted bitstream. At a receiver, for example, the receiver 118 of FIG. 1, a synchronizer searches for these synchronization words in each period in order to establish synchronization. The prefixes, for example, the prefix 304 of FIG. 3, found in a H.264 encoded video sequence, for example, the H.264 encoded video sequence 300 of FIG. 3, may be considered as synchronization words, even though they are not meant for synchronization at the physical layer.

[00209] Various embodiments address the possibility of exploiting these synchronization words or prefixes so that synchronization may be performed at the physical layer.

[00210] Most received packets do not contain the SPS and PPS headers. The SPS-JSCD and PPS-JSCD must be conscious of the presence or absence of headers in the received packet. The headers, if present, usually constitutes only a small portion in the received packet. The location of these headers is usually not fixed, due to the varying length of the encoded source. The SPS-JSCD and PPS-JSCD must have knowledge of the header location in order to commence decoding. The trellis-based decoding of the VLC SPS-JSCD and PPS-JSCD must have knowledge of the length of the header in order to decode properly. The header length is usually not fixed, again due to the varying length of the encoded source.

[00211] There are two objectives that the synchronization algorithm needs to achieve. Firstly the synchronizer needs to detect the presence or absence of a complete NAL unit 1200 (FIG. 12) within a received packet. The complete NAL unit 1200 as one having the prefixes 1202, 1204 on both ends. The prefix 1202 at the front end belongs to the considered NAL unit while the prefix 1204 at the rear end belongs to the next NAL unit. Detecting the presence of a complete NAL unit 1200 reveals the location of the NAL unit within the received packet as well as the number of bits it carries, which are the two most critical parameters for the JSCD. In this case, the JSCD is invoked. If no complete NAL unit is found, the synchronization is deemed unsuccessful and the JSCD is not activated.

[00212] The second objective for the synchronizer is to determine if the complete NAL unit contains the SPS header or the PPS header. If it does, then its RBSP is pushed to the SPS-JSCD or the PPS-JSCD, respectively.

[00213] Key assumptions are made for the synchronizer:-

1. It is assumed that no VCL NAL unit may be complete. This is a valid assumption because a VCL NAL unit typically spans over many bits such that it is not likely to find a complete VCL NAL unit within a received packet.
2. If by any chance the packet size is large enough to accommodate a complete VCL NAL unit, the maximum allowable length for a complete NAL unit (e.g., the complete NAL unit 1200 of FIG. 12) have to be limited. This is borne from the fact that the length of the SPS or PPS header is usually much shorter than that of the VCL RBSP. Based on the simplified SPS header model as given in Table 5, the maximum length of the SPS header is only 104 bits and the RBSP of the VCL NAL unit is much longer than that. The maximum length of the PPS header is even shorter, spanning at most 48 bits, as given in Table 18. Thus, given this assumption, the synchronizer ignores any NAL unit whose RBSP length exceeds beyond the limit (which is 104 bits) and only non-VCL NAL units are detected, which contain either the SPS or PPS header.

[00214] The synchronization algorithm may be sub-divided into 3 phases; mainly correlation, sorting, and selection.

[00215] Correlation Phase:-

[00216] As mentioned previously, the synchronizer first needs to determine the presence of a complete NAL unit (e.g., the complete NAL unit 1200 of FIG. 12). This is achieved by identifying the two prefixes 1202, 1204 at the front and rear end of the NAL unit. The NAL unit syntax for the SPS header and the PPS header have the following bit sequence : 0 1 1 0 0 1 1 1 = 0x67 and 0 1 1 0 1 0 0 0 = 0x68, respectively. It is noted that the first 4 MSB of both headers are identical. So it may be considered that the first 4 bits of the NAL unit syntax is part of the front end synchronization word. As such, the front end synchronization word consists of the 32-bit prefix plus the 4-bit NAL unit syntax. This is referred to as the Front Synchronization Word (FSW) 1210 (FIG. 12) having a hexadecimal format 0x000000016. The rear end synchronization word still retains the 32-bit prefix structure and it is referred to as the Rear Synchronization Word (RSW) 1214 (FIG. 12).

[00217] The number of bits between the start of the FSW and the start of the RSW constitute the total number of bits in the NAL unit. Subtracting the length of the FSW 1210, i.e., 36 bits, from this number gives the length of the RBSP 1208 plus the 4 bits of the nal_unit_type, which identifies the type of header. This 4-bit sequence is referred to as the header type (HT) 1212 (FIG. 12). Unfortunately, the number of bits in the RBSP 1208 is an unknown variable. However, based on the simplified SPS header model as given in Table 5, the RBSP length of the SPS header is ranged from a minimum of 40 bits to a maximum of 104 bits, in multiples of 8. In addition, the simplified PPS header model given in Table 18 shows that the RBSP length of the PPS header ranges from 24 bits to 48 bits, again in multiples of 8. Thus a total of 11 unique synchronization patterns, where each pattern takes on a particular header length, are created as shown in FIG. 13.

[00218] The nomenclatures of FIG. 13 with respect to the FSW 1210, the HT 1212 and the RSW 1214 of FIG. 12 are as follows:

$$N_{\text{RBSP}} 1300 - \text{Length of the RBSP}$$

N_{NAL} 1302 – Length of the NAL unit

N_{pattern} 1304 – Length of the synchronization pattern.

[00219] The different synchronization patterns are distinguished with a superscript k , i.e.

N_{RBSP}^k , N_{NAL}^k , and N_{pattern}^k , where $k = 1, \dots, 11$.

[00220] Each of these 11 synchronization patterns slides across the received packet in order to search for a similar pattern in the packet. Sliding is performed in a byte-by-byte format. Similar to the conventional synchronization methods, the search is based on correlating the pattern with the soft information of the received sequence and then comparing the obtained correlation metric against a predefined threshold. The correlation is only done on the FSW 1210 and RSW 1214, but not the HT 1212. The main purpose of the predefined threshold is to ensure that when the received packet does not contain any header NAL unit, the correlation metrics of all synchronization patterns fall below the threshold and the synchronization becomes unsuccessful. Invoking the JSCDs when no SPS or PPS header is present have a undesirable effect on the packet performance, a phenomenon that should be avoided.

[00221] The input to the synchronizer corresponds to the Log-Likelihood Ratio (LLR) output of the turbo decoder, for example, similar to the turbo decoder 902 of FIG. 9. For a more efficient comparison, the LLR output is converted to its equivalent soft information using the following probability equations.

$$P(u_n = 0) = \frac{\exp\left(\frac{L(u_n)}{2.0}\right)}{\exp\left(\frac{L(u_n)}{2.0}\right) + \exp\left(-\frac{L(u_n)}{2.0}\right)} \quad (14)$$

$$P(u_n = 1) = \frac{\exp\left(-\frac{L(u_n)}{2.0}\right)}{\exp\left(\frac{L(u_n)}{2.0}\right) + \exp\left(-\frac{L(u_n)}{2.0}\right)} \quad (15)$$

where $L(u_n)$ is the LLR value of bit u_n . The reason why the LLR values are converted to their corresponding soft information is because the LLR values are usually very large. This results in large correlation metrics, which presents a problem in choosing a suitable threshold value due to its wide range. By using the soft information, the correlation metric is limited to a maximum value of $36+32=68$, which makes selecting the threshold value more manageable.

[00222] The correlation metric $C_{n,k}$ is obtained using the following equation.

$$C_{n,k} = \sum_{i=0}^{35} P(u_{n+1} = FSW(b_i)) + \sum_{i=0}^{31} P(u_{n+N_{NAK}^k+i} = RSW(b_i)) \quad (16)$$

where $FSW(b_i)$ and $RSW(b_i)$ represent the i th bit in the FSW 1210 and RSW 1214, respectively, and n denotes the bit position in the packet. Since the pattern slides across the packet in a byte-by-byte format, n only takes on the values 0, 8, 16, ..., $N_{packet} - N_{pattern}^k$, where N_{packet} is the received packet size. For each slide across the packet, the correlation metric for the FSW 1210 is fixed for all synchronization patterns. Only the correlation metric of the RSW 1214 is different for different patterns, depending on the length of the pattern. This property is exploited in order to reduce the complexity in terms of the number of addition operations. Complexity may be reduced further by first calculating all the possible correlation metrics for the FSW 1210 and the RSW 1214 across the packet. After that, it is just a matter of adding the appropriate FSW and RSW correlation metrics to obtain the correlation metric for the pattern.

[00223] The total number of correlation metrics obtained for the all the synchronization patterns is given by

$$\sum_{k=1}^{11} \left\lceil \frac{N_{packet} - N_{pattern}^k}{8} \right\rceil + 1 \quad (17)$$

[00224] Patterns whose correlation metric exceeds the threshold are placed in a so-called sorting list to kick-start the sorting phase. Each of these patterns in the sorting list corresponds to a different position in the packet and/or RBSP length. If the sorting list is empty, i.e. no correlation metric has exceeded the threshold, then the synchronization algorithm is terminated and the subsequent sorting and selection phases do not commence. This means that the synchronization is unable to find any headers in the packet.

[00225] Sorting Phase:-

[00226] The next step is to sort these patterns by the type of headers, i.e. SPS or PPS header. To do this, the *a priori* information regarding the headers is used. Recall that the length of the SPS header ranges from 40 bits to 104 bits, while the length of the PPS header ranges from 24 bits to 48 bits. This means that all patterns with RBSP length greater than 48 bits belong to the SPS header. Likewise all patterns with RBSP length less than 40 bits belong to the PPS header. The only ambiguity here is for patterns having RBSP length of 40 and 48, since these patterns may belong to either the SPS header or the PPS header. In order to solve this ambiguity problem, the 4-bit HT portion of these ambiguous patterns is correlated against “0 1 1 1” of the SPS header type and “1 0 0 0” of the PPS header type. The higher correlation value is the decided type. At the end of the sorting phase, there are two shortlists of patterns, one for the SPS header and one for the PPS header. The selection phase next determines which pattern in each shortlist is most likely to contain the respective headers.

[00227] JSCD-assisted Selection Phase:-

[00228] Now with the patterns sorted into the SPS header type and the PPS header type, the winning pattern of each type, i.e., the pattern that most likely contains the respective headers is to be determined. Normally, the synchronization pattern that corresponds to the highest correlation metric that exceeds the threshold is deemed to be the winner. However, in some simulation examples (to be discussed later), selecting this synchronization pattern does not give the best performance. This is because the correct

synchronization pattern may not have the highest correlation metric. The performance by selecting the best M synchronization patterns having the highest correlation metric for each header type may be improved. If there are fewer than M synchronization patterns of the header type, then only these few synchronization patterns are considered. In order to select the best synchronization pattern of each header type, but not necessarily having the highest correlation metric, the corresponding SPS-JSCD and the PPS-JSCD are made use of to aid the decision.

[00229] Recall that in the SPS-JSCD in accordance to various embodiments, a scenario-based trellis is used to select the best SPS header scenario based on the highest trellis branch metric. Typically, the scenario that corresponds to the actual SPS header gives the highest branch metric. Similarly, since these M synchronization patterns may correspond to a different length of the N_{RBSP} , their corresponding RBSP may also be considered as different scenarios. Hence by applying the same principle, the RBSP that corresponds to the highest trellis branch metric amongst the M SPS-JSCD outputs may be the decoded SPS header. The associated extrinsic output from this best RBSP sequence may be fed back to the turbo decoder, for example, similar to the turbo decoder 902 of FIG. 9 for further iterative processing. The same PPS-JSCD assisted selection procedure is also used for the patterns corresponding to the PPS header.

[00230] The value of M is to be chosen wisely. If M is too low, then the possibility of detecting the correct synchronization pattern is reduced. Hence, intuitively, a higher M value is desirable. However, the SPS header and PPS header are not error correcting codes, so given a multitude of synchronization patterns possibilities, it is also not easy for the SPS-JSCD and PPS-JSCD to efficiently distinguish the best synchronization pattern.

[00231] A flow diagram depicting the synchronization algorithm is shown in FIG. 14. This synchronization algorithm also works for cases when only one type of header is in the packet or no headers are in the packet. When only one type of header is present, then ideally the sorting phase sorts all the patterns into the shortlist corresponding to that header type. If no headers are present, then ideally the correlation metric of all patterns falls below the threshold and the JSCDs is not activated. Of course, in practice, some irregularities from the ideal case may be expected.

[00232] In FIG. 14, the correlation algorithm 1400 starts with converting LLR to soft information 1402, followed by obtaining correlation metric for all synchronization patterns 1404 and removing all patterns with correlation metric below or less than the threshold 1406. The correlation algorithm 1400 then checks if any pattern is left at 1408. If yes, the sorting algorithm 1410 starts. If no, the correlation algorithm 1400 ends.

[00233] The sorting algorithm 1410 starts with checking whether all patterns have been sorted at 1412. If yes, the JSCD-assisted selection algorithm 1414 starts. If no, the sorting algorithm 1410 checks if N_{RBSP} of the pattern is more than 48 bits at 1416. If yes, the sorting algorithm 1410 places the pattern in the SPS header shortlist at 1418 and proceeds back to 1412. If no, the sorting algorithm 1410 checks if N_{RBSP} of the pattern is less than 40 bits at 1420. If yes, the sorting algorithm 1410 places the pattern in the PPS header shortlist at 1422 and proceeds back to 1412. If no, the sorting algorithm 1410 correlates HT at 1424 and then check if SPS HT is more than PPS HT at 1426. If yes, the sorting algorithm 1410 places the pattern in the SPS header shortlist at 1428 and proceeds back to 1412. If no, the sorting algorithm 1410 places the pattern in the PPS header shortlist at 1430 and proceeds back to 1412.

[00234] The JSCD-assisted selection algorithm 1414 starts with selecting the top M patterns having the highest correlation metric at 1432, followed by decoding the RBSP of the patterns using JSCD and noting the branch metric at 1434, and subsequently providing feedback on the extrinsic information having the highest branch metric at 1436. The JSCD-assisted selection algorithm 1414 is for both SPS-JSCD and PPS-JSCD.

[00235] The receiver configuration with synchronization 1500 is shown in FIG. 15 and follows the notation $RC_{SYNC}(N_{inner}, N_{outer})$. The JSCD 1502 is specified as a general term in FIG. 15 in this case because it is not known which JSCD is activated. For example, both the SPS and PPS headers may be present in the packet but if the synchronization is only able to detect an SPS header after the sorting phase, only the SPS-JSCD is activated. Likewise, if only the SPS header is present in the packet but the synchronization somehow detected the presence of a PPS header after the sorting phase, the PPS-JSCD is activated.

[00236] In a second aspect, a receiver including a decoder in accordance to various embodiments is provided. For example, the receiver may be the receiver 118 of FIG. 1, the receiver 920 of FIG. 9, the receiver 1000 of FIG. 10, the receiver 1102 of FIG. 11, or the receiver 1500 of FIG. 15.

[00237] In a third aspect, a method of decoding an encoded header of a video data packet is provided as shown in FIG. 16. In FIG. 16, the method 1600 includes receiving an encoded header of a video data packet 1602; and decoding the encoded header using joint source-channel decoding (JSCD) 1604.

[00238] The terms “decoding”, “encoded header”, “using”, and “JSCD” may be as defined above.

[00239] In various embodiments, the video sequence may be a H.264 video sequence.

[00240] In various embodiments, the step of decoding the encoded header using joint source-channel decoding (JSCD) 1604 may include constructing a plurality of trellis.

[00241] As used herein, the term “constructing” may refer to “building”, “forming”, “evaluating”, or “computing”.

[00242] In one embodiment, the step of decoding the encoded header using joint source-channel decoding (JSCD) 1604 may include determining a code from the received encoded header or a part thereof, and generating a bit sequence and a probability value by using the code, the probability value indicating the likelihood that the generated bit sequence is a correctly decoded header or a part thereof of the video data packet.

[00243] The terms “determining”, “code”, “generating”, “correctly decoded header” may be as defined above.

[00244] In this embodiment, the step of determining the code may include generating a plurality of codes and respective probability values from the received encoded header or the part thereof and selecting from the plurality of codes the code based on the respective probability values. The step of selecting the code may include selecting from the plurality of codes the code having the largest probability value.

[00245] The step of generating the bit sequence and the probability value may include performing a process operating on a mode of posterior distribution, or on maximum *a posteriori* probability (MAP), or on BCJR MAP or logMAP.

[00246] In various embodiments, the method 1600 may further include converting the bit indexes of the encoded header into respective syntax indexes for determining the code. The method 1600 may further include transforming the bit indexes into states such that each state has an even number of bits.

[00247] The step of generating the probability value may include determining the likelihood that each bit of the bit sequence corresponds to a respective bit of the correctly decoded header or the part thereof.

[00248] In various embodiments, when the encoded header is an encoded SPS header, the method 1600 may further include decoding another part of the encoded SPS header comprising a fixed length code of more than 1 bit.

[00249] For example, the fixed length code may be 25 bits.

[00250] In various embodiments, the method 1600 may further include dividing the encoded header received by the input unit into the part comprising the variable length code and the other part comprising the fixed length code.

[00251] The method 1600 may further include multiplexing the generated bit sequence and an output obtained from decoding the other part of the encoded SPS header comprising the fixed length code of more than 1 bit.

[00252] The terms “dividing” and “multiplexing” may be as defined above.

[00253] In various embodiments, the method 1600 may further include generating extrinsic information for decoding the encoded header.

[00254] The method 1600 may further include receiving side information from a transmitter sending the video data packet, the side information indicating the location of the encoded header in the video data packet, the length of the encoded header, or both.

[00255] The phrases “extrinsic information” and “side information” are as defined above.

[00256] In various embodiments, the method 1600 may further include determining a presence of a complete network abstract layer (NAL) unit in the video data packet, wherein the complete NAL unit comprises the NAL unit sandwiched between two prefixes; and determining the location of the encoded header in the video data packet and/or the length of the encoded header from the video data packet.

[00257] The step of determining the location of the encoded header in the video data packet and/or the length of the encoded header from the video data packet may include generating a plurality of synchronization patterns; correlating each of the synchronization patterns with soft information of the video data packet to obtain respective correlation metrics; and comparing each of the respective correlation metrics against a predefined threshold.

[00258] The phrases “complete NAL unit”, “correlating”, “soft information” may be defined as above.

[00259] The step of correlating the each of the synchronization patterns with the soft information of the video data packet may including sliding the each of the synchronization patterns across the video data packet to search for a similar pattern in the packet. Sliding may be in a byte-by-byte format.

[00260] In various embodiments, the method 1600 may further include sorting the each of the plurality of synchronization patterns into different groups based on header types; and determining for each group, a synchronization pattern that most likely comprises a header corresponding to the header type of that group.

[00261] The step of determining the synchronization pattern that most likely comprises the header corresponding to the header type may include decoding the respective encoded headers of the sorted synchronization patterns for each group.

[00262] In various embodiments, the method 1600 may include providing *a priori* information for a subsequent iteration of decoding by utilizing a feedback of a combination of extrinsic information obtained from decoding the encoded header and the generated extrinsic information for decoding the encoded header.

[00263] The following non-limiting examples are described with respect to various embodiments of the invention, as follows.

Simulation Setup

[00264] The transmitter model employed in the simulation is shown in FIG. 17. In FIG. 17, a JM H.264 encoder 1700 is coupled to a SPS+PPS header complete NAL unit extraction 1702 which is in turn coupled to a SPS+PPS header complete NAL unit

insertion 1704. The SPS+PPS header complete NAL unit insertion 1704 also receives from a random data generator 1706. The output of the SPS+PPS header complete NAL unit insertion 1704 is fed into a turbo encoder 1708. BPSK modulation 1710 is used such that the encoded signal (i.e., received signal 1712) is transmitted through an additive white Gaussian noise (AWGN) channel 1714.

[00265] The test video sequence used is the foreman_qcif.yuv, which is encoded by the JM encoder to generate the corresponding H.264 video sequence. Specifically, the generated H.264 video sequence has a Baseline profile with a frame pattern IPPPPIPPPPIP..... There are a total of 300 frames. The complete SPS and PPS header NAL unit are extracted from the encoded H.264 video sequence. These two NAL units are contiguous in the foreman test video sequence and so their complete NAL unit overlaps, as shown in FIG. 18. In FIG. 18, two prefixes 1800, 1802 sandwiches a SPS NAL identifier 1804, and a SPS header 1806, denoted as the complete SPS NAL unit 1808. The prefix 1802 further sandwiches a PPS NAL identifier 1810 and a PPS header 1812 with another prefix 1814, denoted as the complete PPS NAL unit 1816.

[00266] FIG. 19 shows a SPS header NAL unit 1900. Based on the test video sequence, the total number of bits in the SPS header, including the 8 trailing bits required to round the total number of bits in the header to a multiple of 8 (byte) is 64. This excludes the 32-bit sync word 1902, 1904 and the 8 bits needed to represent the NAL unit syntax 1906. The N-bit SPS header 1908 is a 64-bit SPS header being referred to as the *basis* SPS header from hereafter. Table 19 shows the bit pattern of the basis SPS header.

[00267] Table 19

Syntax	Bit Sequence	Value
profile_idc	01000010	66 (Baseline)
constraint_set0_flag	0	0 (Fixed in JM encoder)
constraint_set1_flag	0	0 (Fixed in JM encoder)
constraint_set2_flag	0	0 (Fixed in JM encoder)
constraint_set3_flag	0	0 (Fixed in JM encoder)

reserved_zero_4bits	0000	0
level_idc	00101000	40
seq_parameter_set_id	1	0 (Fixed in JM encoder)
log2_max_frame_num_minus4	1	0
pic_order_cnt_type	1	0
log2_max_pic_order_cnt_lsb_minus4	00111	6
max_num_ref_frames	00110	5
gaps_in_frame_num_value_allowed_flag	0	0
pic_width_in_mbs_minus1	0001011	10
pic_height_in_map_units_minus1	0001001	8
frame_mbs_only_flag	1	1
direct_8x8_inference_flag	1	1
frame_cropping_flag	0	0
vui_parameters_present_flag	0	0
trailing_bits	10000000	

[00268] Likewise, the total number of bits in the PPS header, including the trailing bits is 32. This excludes the 32-bit sync word and the 8-bit NAL unit syntax. This 32-bit PPS header is referred to as the *basis* PPS header from hereafter. Table 20 shows the bit pattern of the basis PPS header used in the simulations.

[00269] Table 20

Syntax	Bit Sequence	Value
pic_parameter_set_id	1	0
seq_parameter_set_id	1	0 (Fixed in JM encoder)
entropy_coding_mode_flag	0	0
bottom_field_pic_order_in_frame_present_flag	0	0
num_slice_groups_minus1	1	0 (Fixed in JM encoder)

num_ref_idx_10_default_active_minus1	00101	4
num_ref_idx_11_default_active_minus1	00101	4
weighted_red_flag	0	0
weighted_bired_idc	00	0
pic_init_qp_minus26	1	0 (Fixed in JM encoder)
pic_init_qs_minus26	1	0 (Fixed in JM encoder)
chroma_qp_index_offset	1	0
deblocking_filter_control_present_flag	0	0
constrained_intra_pred_flag	0	0
redundant_pic_cnt_present_flag	0	0
trailing_bits	10000000	

[00270] At the start of every simulation run, a sequence of random bits is generated. This sequence of bits is referred to as a packet and the number of bits in this packet is referred to as the packet size. The two contiguous NAL units are inserted in a random position for each run in order to test the efficiency of the synchronization, thus overriding the previously generated random bits. The same NAL units is used for every run during the simulation. If perfect synchronization is assumed, then the SPS-JSCD and/or PPS-JSCD is assumed to have perfect knowledge of the location of their respective headers. This sequence of bits, with the NAL units inserted, is then turbo encoded (i.e., by the turbo encoder 1708 of FIG. 17) according to the 3GPP TS 25.212 standards. For simplicity sake, it is assumed that the modulation is binary phase shift keying (BPSK), i.e., the BPSK modulation 1710 of FIG. 17. The channel is a purely AWGN channel 1714 (FIG. 17), whereby the noise values is randomly generated for each run of the simulation. For every simulation run, only the packet embedded with the two NAL units is transmitted, since only the performance concerning the SPS header or the PPS header are observed. The remaining video sequence are not transmitted. The received signal 1712 is then passed forward to the receiver for processing.

Performance of SPS-JSCD with perfect synchronization

[00271] The BER and HER of the basis SPS header based on $RC_{SPS-JSCD}(N_{inner}, 1)$ are shown in FIG. 20 and FIG. 21, respectively, for a rate-1/3 turbo code having a packet size of 1024. In other words, the SPS-JSCD is only activated once while the number of inner iterations of the turbo decoder is varied. The performance is compared against that without any SPS-JSCD. From FIG. 20, a consistent performance improvement is observed when the SPS-JSCD is employed for any value of N_{inner} . With $N_{inner} = 8$, the inclusion of the SPS-JSCD improved the BER performance by about 0.1 dB as compared to one without any SPS-JSCD at a BER of about 2×10^{-4} . On the other hand, referring to FIG. 21, the corresponding HER does not improve as much. The reason for this may be because there are still certain bits in the SPS header that the SPS-JSCD cannot correct, as mentioned previously. Any errors in these bits cause a header error, regardless of how many other bits the SPS-JSCD has corrected. Because of this, the corresponding PSNR, as shown in FIG. 22, has only improved slightly when comparing between with and without the SPS-JSCD at $N_{inner} = 8$. The achieved PSNR is still about 5-6 dB below the perfect SPS header case.

[00272] For a higher rate turbo code, such as a rate-3/4 code, the performance gain that may be gleaned from $RC_{SPS-JSCD}(N_{inner}, 1)$ increases. For example, FIG. 23 shows that BER of the basis SPS header using RC1 for a rate-3/4 turbo code, in which the performance improvement with SPS-JSCD is about 0.2 dB as compared to one without SPS-JSCD at a BER of about 10^{-4} . Similarly, it is observed that the corresponding HER, as shown in FIG. 24, do not improve as much. Nevertheless, the improvement is more than that with a rate-1/3 turbo code. Hence, in terms of the PSNR, a slightly more improvement is observed when the SPS-JSCD is employed, as shown in FIG. 25. The achievable PSNR is now about 2-3 dB off the perfect SPS header case.

[00273] As mentioned previously, the limited performance improvement from the SPS-JSCD is because there are some bits that the SPS-JSCD cannot correct. These bits can only be corrected by the turbo decoder 1708 (FIG. 17). With $N_{outer} = 1$, there is no feedback from the SPS-JSCD. So for $N_{outer} > 1$, the turbo decoder 1708 exploits the extrinsic information generated by the SPS-JSCD and then attempt to improve the error

correction capability further. The BER and HER of the basis SPS header based on $RC_{SPS-JSCD}(1, N_{outer})$ are shown in FIG. 26 and FIG. 27, respectively, for a rate-1/3 turbo code having a packet size of 1024, where $N_{outer} = 1, \dots, 8$. $RC_{SPS-JSCD}(1, N_{outer})$ has improved the performance further. Now, the achievable gain with $N_{outer} = 8$ is about 0.2 dB at a BER of 2×10^{-4} as compared to only 0.1 dB for $N_{outer} = 1$ at the same BER (see FIG. 20). More importantly, the corresponding HER achieves the same amount of gain, as portrayed in FIG. 27, unlike that of $N_{outer} = 1$. The reason for this is because with the aid of the feedback from the SPS-JSCD, the bits that cannot be corrected by the SPS-JSCD have the opportunity to be correct by the turbo decoder 1708 in the next set of inner iterations. The probability of being corrected increases with every iteration. This is observed by comparing the HER of FIG. 21 ($N_{outer} = 1$) and FIG. 27 ($N_{outer} = 8$). This improvement of the HER has a great impact on the PSNR, as depicted in FIG. 28. It is observed that the achievable PSNR using $RC_{SPS-JSCD}(1, 8)$ is now very close to the error-free SPS header bound, with at most 1 dB loss at the E_b/N_0 of 0-0.1 dB.

[00274] The previous examples were based on $N_{inner} = 1$. The error correction capability may be improved further by increasing the number of inner iterations. The BER simulation results for $RC_{SPS-JSCD}(2, N_{outer})$ is shown in FIG. 29 for a rate-1/3 turbo code having a packet size of 1024 bits, where $N_{outer} = 1, \dots, 8$. The performance without SPS-JSCD with $N_{inner} = 16$ is also shown (dashed line) for comparison sake. Comparing this with $RC_{SPS-JSCD}(2, 8)$ at a BER of about 2×10^{-4} , it is observed that there is a gain of about 0.25 dB. The corresponding HER for $RC_{SPS-JSCD}(2, N_{outer})$ is shown in FIG. 30.

[00275] The BER and HER simulation results for $RC_{SPS-JSCD}(4, N_{outer})$ and $RC_{SPS-JSCD}(6, N_{outer})$ for various values of N_{outer} are shown in FIG. 31 and FIG. 32, respectively. It is observed that apart from the first iteration and to some extent the second iteration, both $RC_{SPS-JSCD}(4, N_{outer})$ and $RC_{SPS-JSCD}(6, N_{outer})$ have approximately similar performance as the number of outer iterations increases.

[00276] Last but not least, a comparison of the achievable Y PSNR between $RC_{SPS-JSCD}(2, 8)$, $RC_{SPS-JSCD}(4, 6)$, and $RC_{SPS-JSCD}(6, 4)$ is shown in FIG. 33. The Y PSNR obtained from the various combinations of $RC_{SPS-JSCD}(N_{inner}, N_{outer})$ are very similar and they are close to the error-free case.

Performance of PPS-JSCD with perfect synchronization

[00277] The BER simulation results of the basis PPS header for $RC_{PPS-JSCD}(6, N_{outer})$ is shown in FIG. 34, based on a rate-1/3 turbo code having a packet size of 1024 bits, where $N_{outer} = 1, \dots, 4$. The corresponding BER without the aid of the PPS-JSCD is also shown for comparison. From FIG. 34, it is observed that with the PPS-JSCD, a gain of about 0.1 dB may be achieved at a BER of $10e^{-4}$. For the SPS header case as shown previously, a gain of more than 0.2 dB may be observed with a SPS-JSCD. There are two reasons why the PPS-JSCD achieved a smaller gain as compared to the SPS-JSCD. Firstly, there are greater redundancies found in the SPS header as compared to the PPS header. For instance, the amount of *a priori* information for the PPS header, in terms of the number of values fixed by the JM encoder, is less than that of the SPS header. Also, redundancies in the SPS header may be derived from at least 4 VLC syntax (depending on the scenario) in addition to the FLC syntax `profile_idc` and `level_idc`. These are the syntax where error corrections may be made by the SPS-JSCD. On the other hand, the PPS header contains at most 4 VLCs. Hence the number of error corrections that may be made by the PPS-JSCD is limited. Secondly, the basis PPS header, which is 32 bits long, is considerably shorter than the basis SPS header, which is 64 bits long. The effect of a shorter header is that the extrinsic information feedback from the JSCD to the turbo decoder is diminished. Hence, the gain that may be obtained from subsequent iterations is not as great as when a longer header is considered.

[00278] Likewise, a similar gain in the basis PPS header HER is observed, as illustrated in FIG. 35, when the PPS-JSCD is invoked. Naturally, due to the poorer performance relative to the SPS-JSCD, the corresponding PSNR, as shown in FIG. 36, do not improve as much as what have been seen for the SPS header previously. In FIG. 36, it is assumed that the SPS header is error-free. So only an error in the basis PPS header gives a total loss of video. Because the basis PPS header has a shorter length, its HER is generally lower. Hence even without the aid of any JSCD, the achievable PSNR with the basis PPS header is higher than that with the basis SPS header. The inclusion of the PPS-

JSCD improves the PSNR by about 2 dB, and this is still about 1.5 dB away from the error-free case.

Performance of SPS-JSCD & PPS-JSCD with perfect synchronization

[00279] Previously, the performance of a standalone SPS-JSCD or PPS-JSCD in accordance to various embodiments is shown. Now the performance when both JSCDs are activated to decode both headers is presented. FIG. 37 shows the BER of the basis SPS header when both SPS-JSCD and PPS-JSCD are activated. As a benchmark, the performance is compared against that when the PPS-JSCD is not used. A very slight improvement in the BER is observed for $N_{outer} > 1$. This improvement is wholly due to the extrinsic information feedback from the PPS-JSCD. Obviously when $N_{outer} = 1$, there are no feedback from the PPS-JSCD yet and hence no gain is seen, as evident from FIG. 37.

[00280] FIG. 38 shows the BER of the basis PPS header when both SPS-JSCD and PPS-JSCD are activated. This time, the performance is compared against that when the SPS-JSCD is not used. In the case of the basis PPS header, a larger performance improvement of about 0.05 dB is observed. Clearly, the extrinsic information feedback from the SPS-JSCD is more beneficial to the performance of the PPS-JSCD. This is of course due to the fact that the SPS-JSCD itself has a better decoding performance as compared to the PPS-JSCD.

[00281] In the context of the PSNR, a decent improvement of about 3 dB is observed when the SPS-JSCD and PPS-JSCD are invoked, as seen in FIG. 39. This is about 1.5 dB away from the error-free header case.

Performance of SPS-JSCD & PPS-JSCD assisted synchronization

[00282] So far, the examples presented were based on the assumption of perfect synchronization. In other words, the decoding parameters required by the JSCDs were perfectly known. Now the example when the knowledge of these parameters is acquired through the JSCD-assisted synchronization in accordance to various embodiments is presented. The achievable performance if M is set to 1 is investigated. In this case, the

synchronization pattern having the highest correlation metric that exceeds the threshold is the winner. The results for the basis SPS header BER based on $RC_{\text{SYNC}}(6, N_{\text{outer}})$ is shown in FIG. 40. The threshold is set to 50.0. The outcomes are compared against those with perfect synchronization, i.e. $RC_{\text{JSCDs}}(6, N_{\text{outer}})$. A degradation in performance of about 0.15 dB is observed when compared with the perfect synchronization case. However, this performance is still better than one without any JSCDs. Similar amount of degradation is also observed for the basis PPS header as displayed in FIG. 41.

[00283] Now what happens when M is increased to 4. In this case, the best pattern is selected based on the branch metric of the SPS-JSCD and the PPS-JSCD, instead of the correlation metric. The threshold is again set to 50.0. The results for the basis SPS header BER is shown in FIG. 42. Comparing with $M = 1$, it is observed from the figures that the performance have improved when M is increased to 4. The gap between $RC_{\text{SYNC}}(6, N_{\text{outer}})$ and $RC_{\text{JSCDs}}(6, N_{\text{outer}})$ is now just about 0.05dB. Similar improvement is also observed for the basis PPS header as demonstrated in FIG. 43. This shows that the JSCD based on its branch metric is a good tool to sort out the best synchronization pattern. In terms of the PSNR, there is only a very slight degradation between $RC_{\text{SYNC}}(6, 4)$ and $RC_{\text{JSCDs}}(6, 4)$, as shown in FIG. 44.

[00284] However, increasing the value of M does not mean that performance is improved further. A performance comparison between $M = 4$ and $M = 6$ is illustrated in FIG. 45 for the basis SPS header and in FIG. 46 for the basis PPS header. It is clearly observed that the performance with $M = 4$ is always better than that with $M = 6$. This shows that increasing M does not necessarily lead to an improvement in the performance. Furthermore, increasing M increases the complexity as there are more patterns for the JSCDs to decode.

[00285] If every received packet contains both the SPS and PPS headers, then the threshold is not required because both JSCDs must be activated every time. However, in practical situations, this is not the case. In fact, complete NAL units carrying the SPS header and/or the PPS header occurs infrequently. Hence in most cases, the SPS-JSCD and PPS-JSCD are not be invoked. False activation of the JSCDs naturally lead to a degradation in the performance. The threshold is there to prevent such false activation.

Ideally, the threshold value is set such that when there are no complete NAL units, be it the SPS header or the PPS header, then the correlation metrics corresponding to all synchronization patterns fall below the threshold. So a high threshold is desirable. On the other hand, if the threshold value is set too high, all the correlation metrics may also fall below the threshold if there are really complete NAL units in the packet carrying the headers. The following set of examples investigates the impact of the threshold value when the complete NAL units are present in the packet and when the complete NAL units are not present in the packet.

[00286] First of all, the threshold is increased from 50.0 and 52.0 and compare both performances. The basis SPS header BER and basis PPS header BER are illustrated in FIG. 47 and FIG. 48, respectively. Increasing the threshold to 52.0 does not impact the performance of both headers much. In fact, some minor improvements are observed. So it is concluded that if the headers are present in the packet, similar performance may be gleaned with a threshold of 50.0 or 52.0.

[00287] Now, what happens when no headers are present in the packet. FIG. 49 shows the packet BER performance when no SPS and PPS headers are transmitted. The performance is compared between a threshold of 50.0 and 52.0. When the threshold is set to a lower value of 50.0, a degradation is seen especially at the lower BER region. This means that sometimes, the correlation metrics of the patterns is above the threshold, even though there are no headers present. The subsequent activation of the JSCDs causes a degradation in the performance. On the other hand, the performance when the threshold is set to 52.0 is the same as a turbo-coded packet. So most, if not all, of the time, the correlation metric of the patterns falls below the threshold and the performance is not affected.

[00288] The examples show that the SPS-JSCD and the PPS-JSCD are able to reduce the error probability of the headers. The margin of improvement may be adjusted by controlling the number of inner and outer iterations. The decoding parameters required by the JSCDs may be acquired by the synchronization algorithm and the resultant performance is not too far away from the perfect synchronization case.

[00289] In the context of various embodiments, the term “about” as applied to a numeric value encompasses the exact value and a variance of +/- 5% of the value.

[00290] While the invention has been particularly shown and described with reference to specific embodiments, it should be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention as defined by the appended claims. The scope of the invention is thus indicated by the appended claims and all changes which come within the meaning and range of equivalency of the claims are therefore intended to be embraced.

CLAIMS

1. A decoder comprising:
an input unit configured to receive an encoded header of a video data packet; and
a circuit configured to decode the encoded header using joint source-channel decoding (JSCD).
2. The decoder of claim 1, wherein the encoded header is an encoded sequence parameter set (SPS) header or an encoded picture parameter set (PPS) header.
3. The decoder of claim 1, wherein the encoded header or a part thereof comprises a variable length code.
4. The decoder of claim 1, wherein the decoder is a trellis-based decoder.
5. The decoder of claim 1, wherein when decoding the encoded header using joint source-channel decoding (JSCD), the circuit is configured to determine a code from the received encoded header or a part thereof, and using the code to generate a bit sequence and a probability value, the probability value indicating the likelihood that the generated bit sequence is a correctly decoded header or a part thereof of the video data packet.
6. The decoder of claim 5, wherein when determining the code, the circuit is configured to generate a plurality of codes and respective probability values from the received encoded header or the part thereof and to select from the plurality of codes the code based on the respective probability values.
7. The decoder of claim 6, wherein the circuit is configured to select from the plurality of codes the code having the largest probability value.

8. The decoder of claim 5, wherein the circuit operates based on a mode of posterior distribution.
9. The decoder of claim 5, wherein the circuit operates based on maximum a posteriori probability (MAP).
10. The decoder of claim 9, wherein the circuit operates based on BCJR MAP or logMAP.
11. The decoder of claim 10, wherein the encoded header comprises bit indexes; and wherein the decoder further comprises a syntax converter configured to convert the bit indexes into respective syntax indexes which the circuit uses to determine the code.
12. The decoder of claim 3, wherein when the encoded header is an encoded SPS header, the decoder further comprises a sub-decoder configured to decode another part of the encoded SPS header comprising a fixed length code of more than 1 bit.
13. The decoder of claim 12, wherein the fixed length code comprises 25 bits.
14. The decoder of claim 12, further comprising a demultiplexer configured to divide the encoded header received by the input unit into the part comprising the variable length code for the circuit and the other part comprising the fixed length code for the sub-decoder.
15. The decoder of claim 12, further comprising a multiplexer configured to multiplex the bit sequence generated by the circuit and a decoded output from the sub-decoder.
16. The decoder of claim 1, further comprising a convolutional decoding unit configured to generate extrinsic information for the circuit.

17. The decoder of claim 16, wherein the convolutional decoding unit is further configured to receive side information from a transmitter sending the video data packet, the side information indicating the location of the encoded header in the video data packet, the length of the encoded header, or both.

18. The decoder of claim 1, further comprising a synchronizer configured to determine a presence of a complete network abstract layer (NAL) unit in the video data packet, wherein the complete NAL unit comprises a NAL unit sandwiched between two prefixes, and to determine the location of the encoded header in the video data packet and/or the length of the encoded header from the video data packet.

19. The decoder of claim 18, wherein the synchronizer comprises a correlation unit configured to generate a plurality of synchronization patterns, to correlate each of the synchronization patterns with soft information of the video data packet to obtain respective correlation metrics and to compare each of the respective correlation metrics against a predefined threshold.

20. The decoder of claim 19, wherein the correlation unit is configured to correlate the each of the synchronization patterns with the soft information of the video data packet by sliding the each of the synchronization patterns across the video data packet to search for a similar pattern in the packet.

21. The decoder of claim 20, wherein the correlation unit is configured to perform sliding in a byte-by-byte format.

22. The decoder of claim 19, wherein the synchronizer further comprises
a sorting unit configured to sort the each of the plurality of synchronization patterns into different groups based on header types; and
a selection unit configured to determine for each group, a synchronization pattern that most likely comprises a header corresponding to the header type of that group.

23. The decoder of claim 22, wherein the selection unit is further configured to utilize the circuit to decode the respective encoded headers of the sorted synchronization patterns for each group.
24. The decoder of claim 16, wherein the convolutional decoding unit comprises a turbo decoder.
25. The decoder of claim 16, wherein the convolutional decoding unit comprises a recursive systematic convolutional (RSC) decoder.
26. The decoder of claim 16, further comprising a feedback path coupled between an input of the convolutional decoding unit and a combination of extrinsic information of the circuit and the extrinsic information of the convolution decoding unit, wherein the feedback path is configured to provide a priori information to the convolutional decoding unit for a subsequent iteration of decoding.
27. The decoder of claim 26, further comprising a summation unit configured to combine the extrinsic information of the circuit and the extrinsic information of the convolution decoder to provide the a priori information.
28. A receiver comprising a decoder of claim 1.
29. A method of decoding an encoded header of a video data packet, the method comprising:
receiving an encoded header of a video data packet; and
decoding the encoded header using joint source-channel decoding (JSCD).
30. The method of claim 29, wherein the video sequence is a H.264 video sequence.

31. The method of claim 30, wherein the encoded header is an encoded sequence parameter set (SPS) header or an encoded picture parameter set (PPS) header.
32. The method of claim 29, wherein the encoded header or the part thereof comprises a variable length code.
33. The method of claim 29, wherein decoding the encoded header using joint source-channel decoding (JSCD) comprises constructing a plurality of trellis.
34. The method of claim 29, wherein decoding the encoded header using joint source-channel decoding (JSCD) comprises determining a code from the received encoded header or a part thereof, and generating a bit sequence and a probability value by using the code, the probability value indicating the likelihood that the generated bit sequence is a correctly decoded header or a part thereof of the video data packet.
35. The method of claim 34, wherein determining the code comprises generating a plurality of codes and respective probability values from the received encoded header or the part thereof and selecting from the plurality of codes the code based on the respective probability values.
36. The method of claim 35, wherein selecting the code comprises selecting from the plurality of codes the code having the largest probability value.
37. The method of claim 34, wherein generating the bit sequence and the probability value comprises performing a process operating on a mode of posterior distribution.
38. The method of claim 34, wherein generating the bit sequence and the probability value comprises performing a process operating on maximum a posteriori probability (MAP).

39. The method of claim 34, wherein generating the bit sequence and the probability value comprises performing a process operating on BCJR MAP or logMAP.
40. The method of claim 39, further comprising converting the bit indexes of the encoded header into respective syntax indexes for determining the code.
41. The method of claim 40, further comprising transforming the bit indexes into states such that each state has an even number of bits.
42. The method of claim 34, wherein generating the probability value comprises determining the likelihood that each bit of the bit sequence corresponds to a respective bit of the correctly decoded header or the part thereof.
43. The method of claim 32, wherein when the encoded header is an encoded SPS header, the method further comprises decoding another part of the encoded SPS header comprising a fixed length code of more than 1 bit.
44. The method of claim 43, wherein the fixed length code comprises 25 bits.
45. The method of claim 43, further comprising dividing the encoded header received by the input unit into the part comprising the variable length code and the other part comprising the fixed length code.
46. The method of claim 43, further comprising multiplexing the generated bit sequence and an output obtained from decoding the other part of the encoded SPS header comprising the fixed length code of more than 1 bit.
47. The method of claim 29, further comprising generating extrinsic information for decoding the encoded header.

48. The method of claim 47, further comprising receiving side information from a transmitter sending the video data packet, the side information indicating the location of the encoded header in the video data packet, the length of the encoded header, or both.

49. The method of claim 29, further comprising:

determining a presence of a complete network abstract layer (NAL) unit in the video data packet, wherein the complete NAL unit comprises the NAL unit sandwiched between two prefixes; and

determining the location of the encoded header in the video data packet and/or the length of the encoded header from the video data packet.

50. The method of claim 49, wherein determining the location of the encoded header in the video data packet and/or the length of the encoded header from the video data packet comprises

generating a plurality of synchronization patterns;

correlating each of the synchronization patterns with soft information of the video data packet to obtain respective correlation metrics; and

comparing each of the respective correlation metrics against a predefined threshold.

51. The method of claim 50, wherein correlating the each of the synchronization patterns with the soft information of the video data packet comprises sliding the each of the synchronization patterns across the video data packet to search for a similar pattern in the packet.

52. The method of claim 51, wherein sliding the each of the synchronization patterns across the video data packet comprises sliding in a byte-by-byte format.

53. The method of claim 49, further comprising:

sorting the each of the plurality of synchronization patterns into different groups based on header types; and

determining for each group, a synchronization pattern that most likely comprises a header corresponding to the header type of that group.

54. The method of claim 53, wherein determining the synchronization pattern that most likely comprises the header corresponding to the header type comprises decoding the respective encoded headers of the sorted synchronization patterns for each group.

55. The method of claim 49, further comprising providing a priori information for a subsequent iteration of decoding by utilizing a feedback of a combination of extrinsic information obtained from decoding the encoded header and the generated extrinsic information for decoding the encoded header.

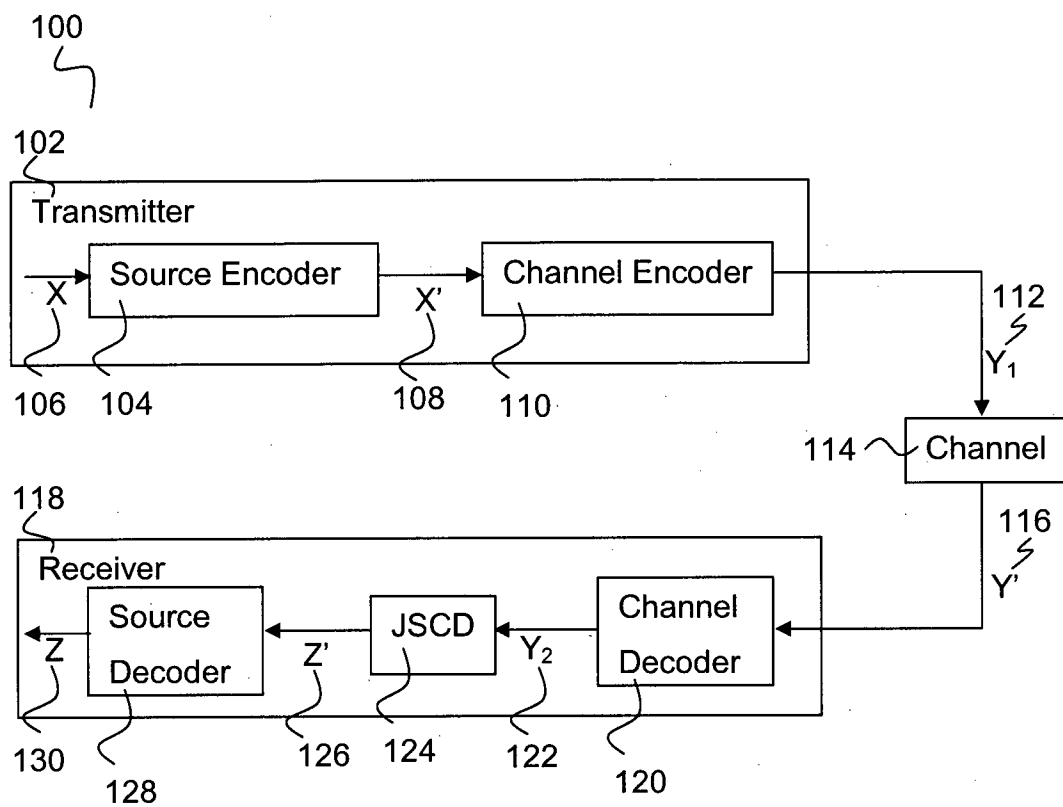


FIG. 1

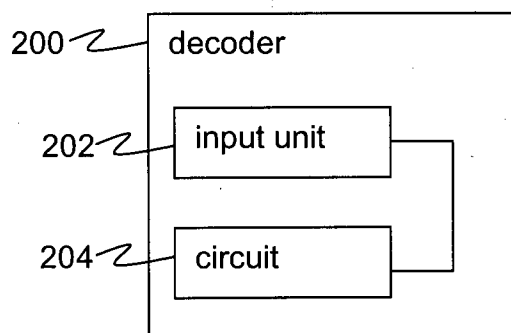


FIG. 2

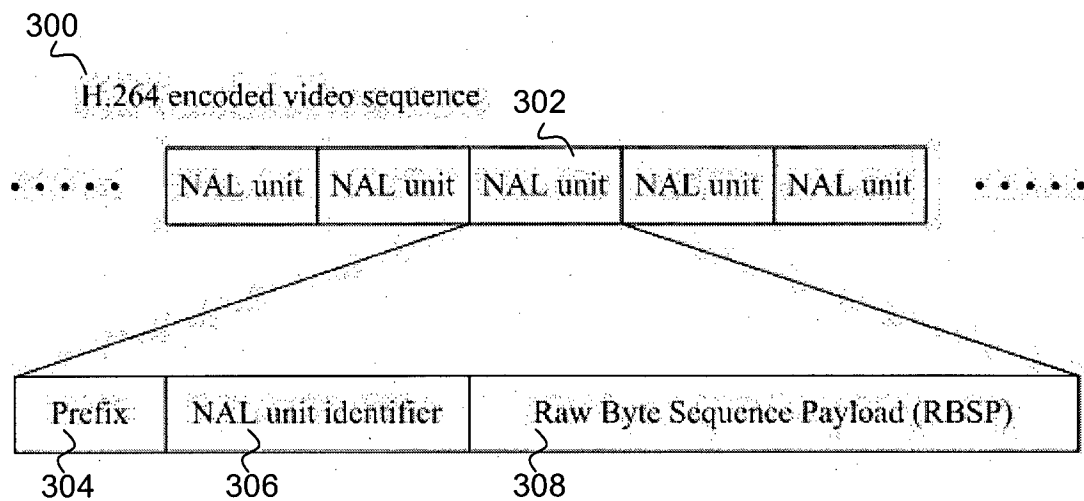


FIG. 3

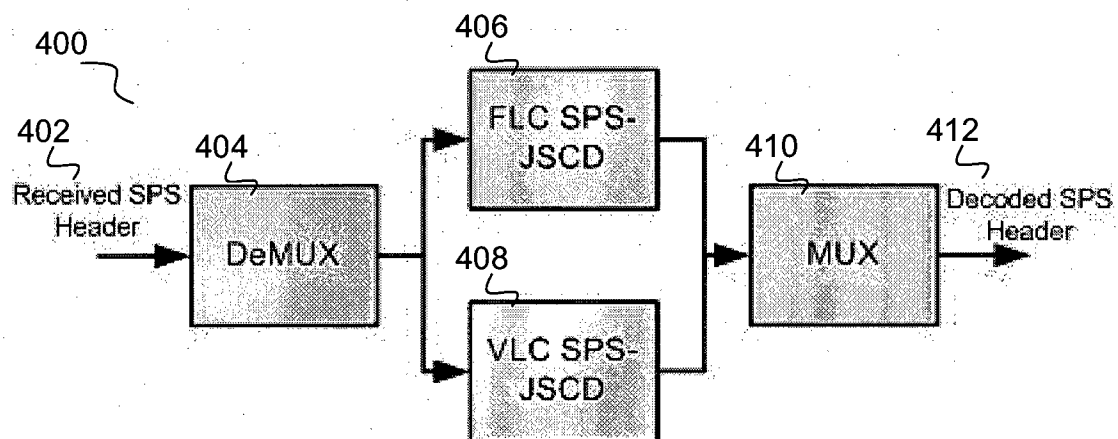


FIG. 4

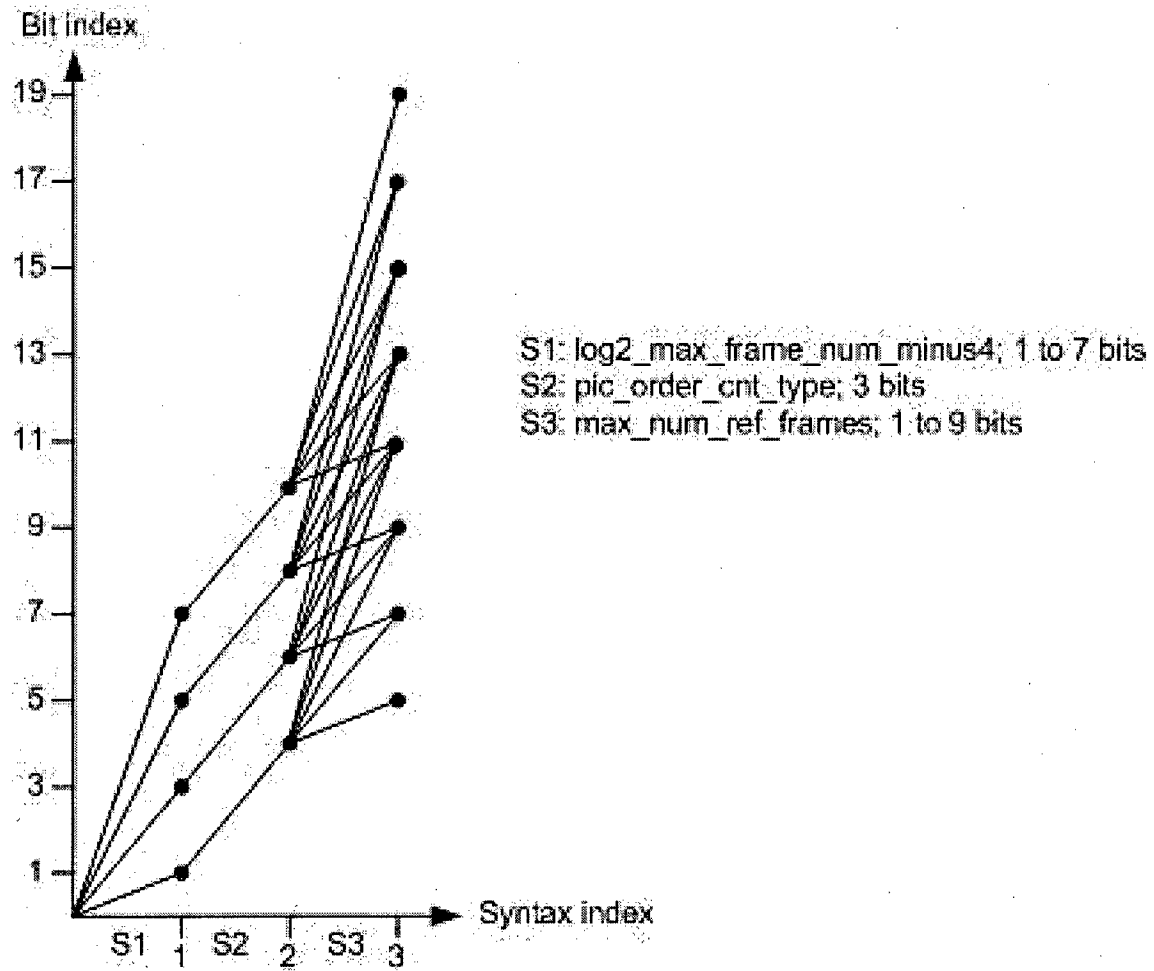


FIG. 5

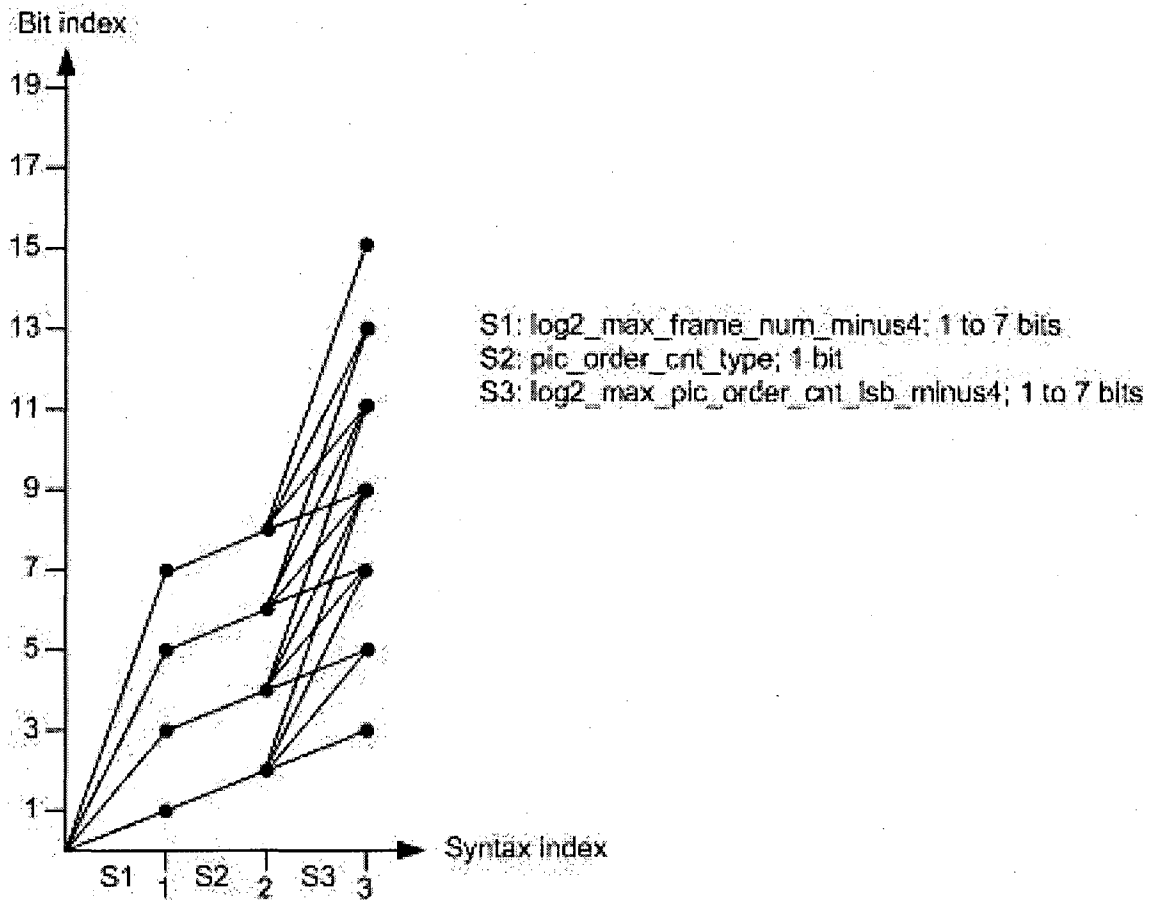


FIG. 6

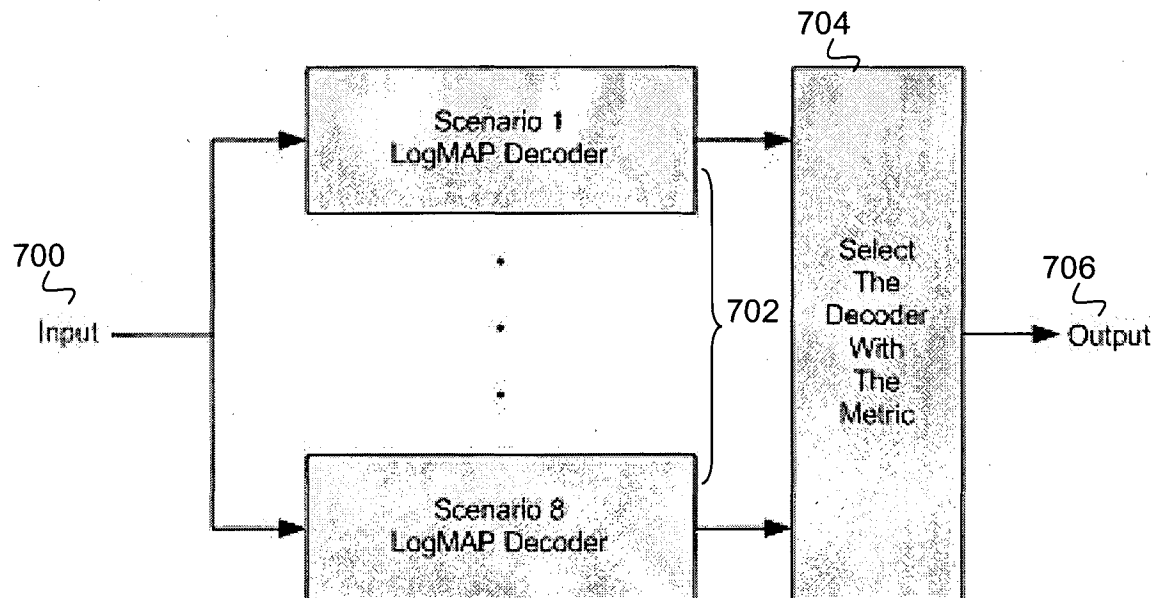


FIG. 7

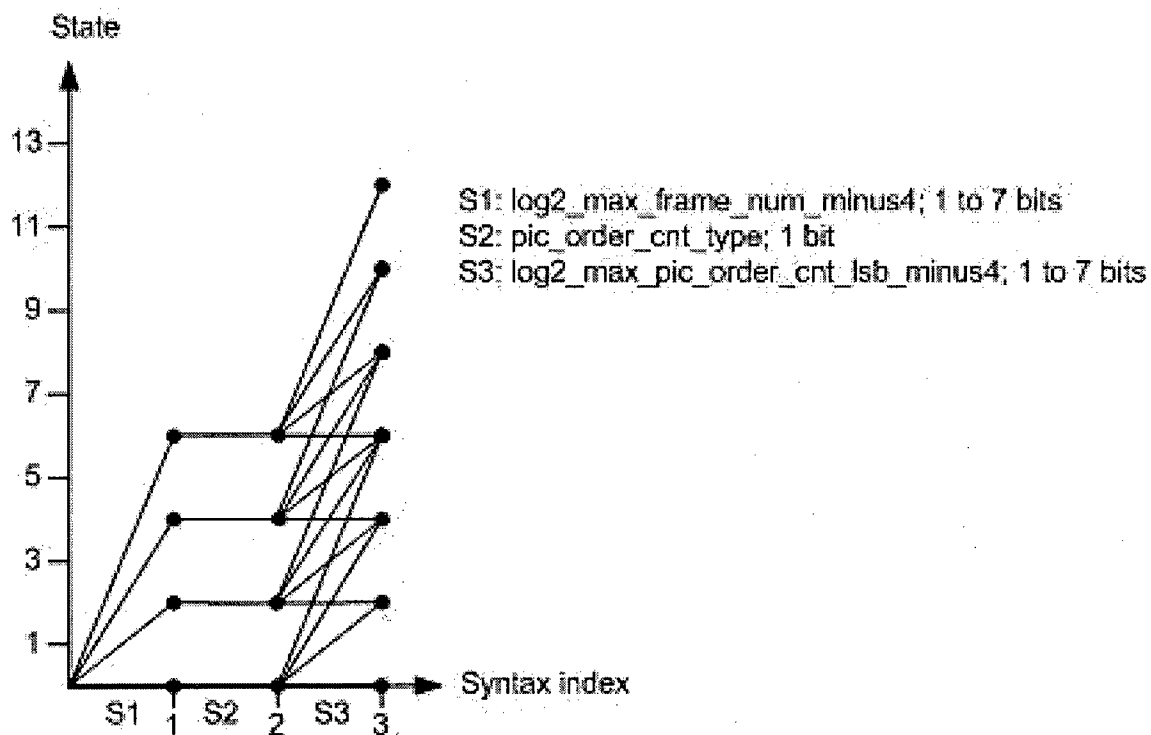


FIG. 8

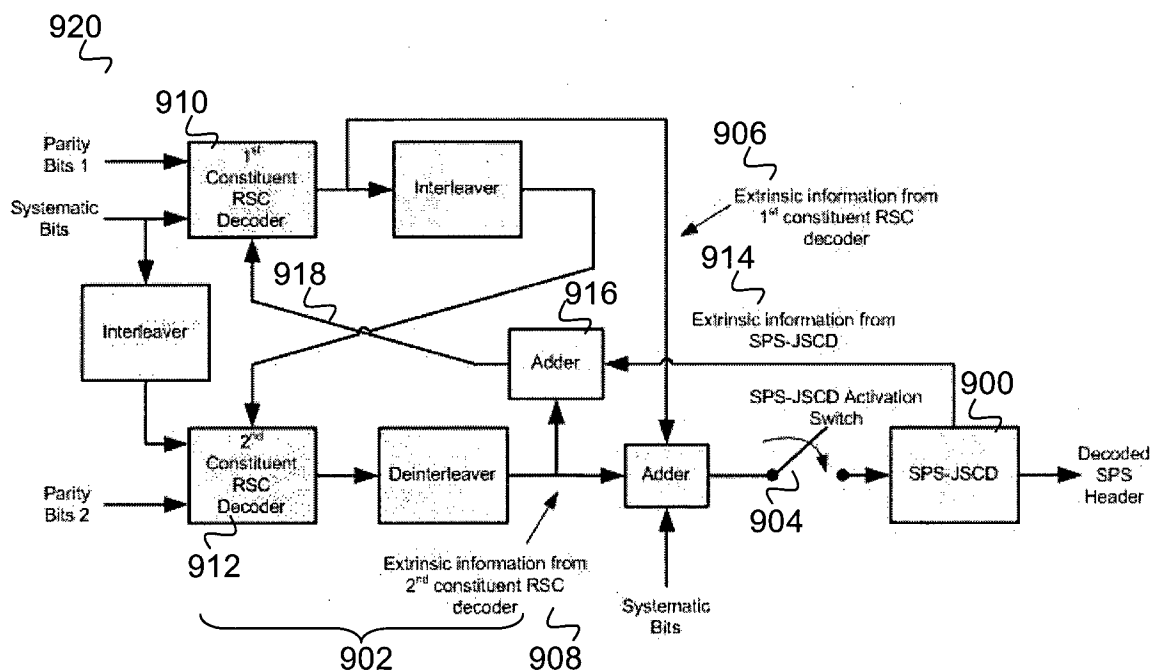


FIG. 9

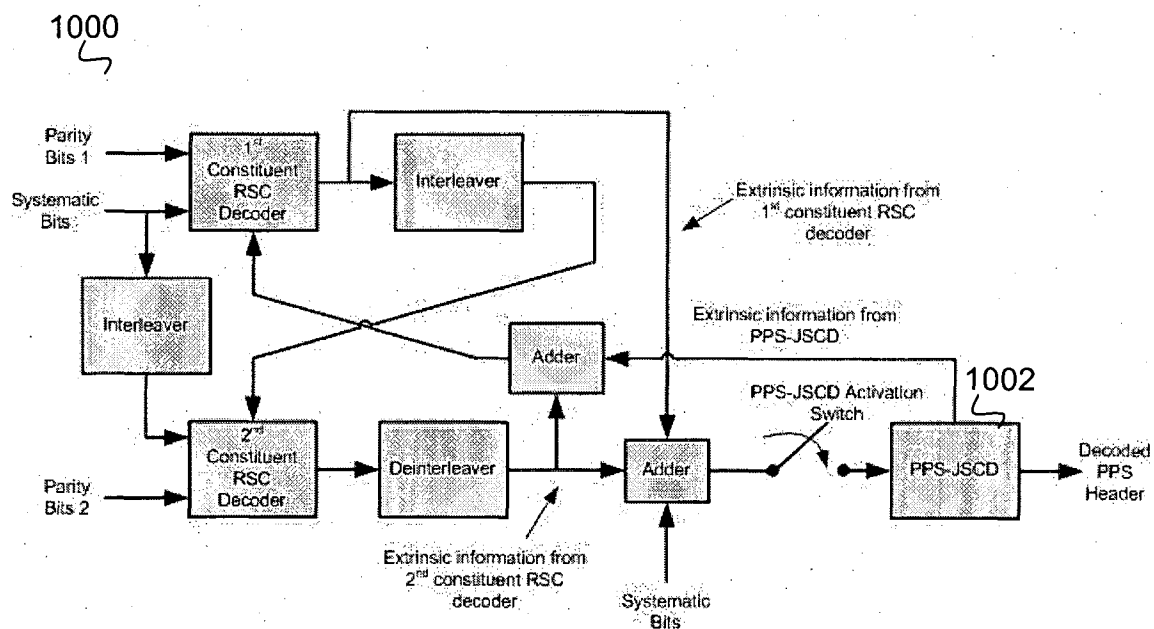


FIG. 10

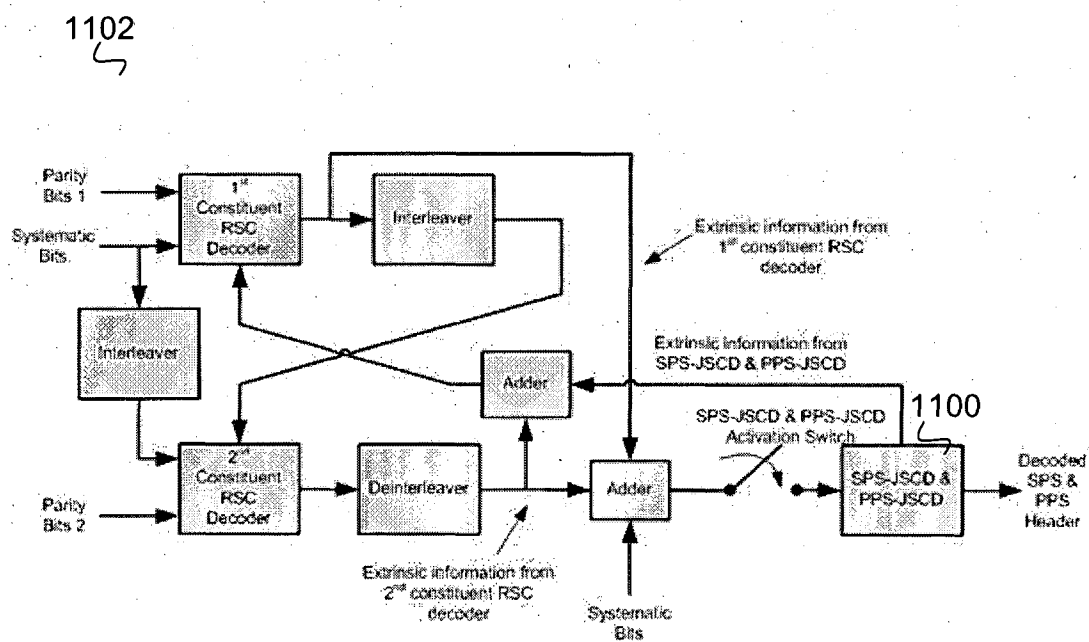


FIG. 11

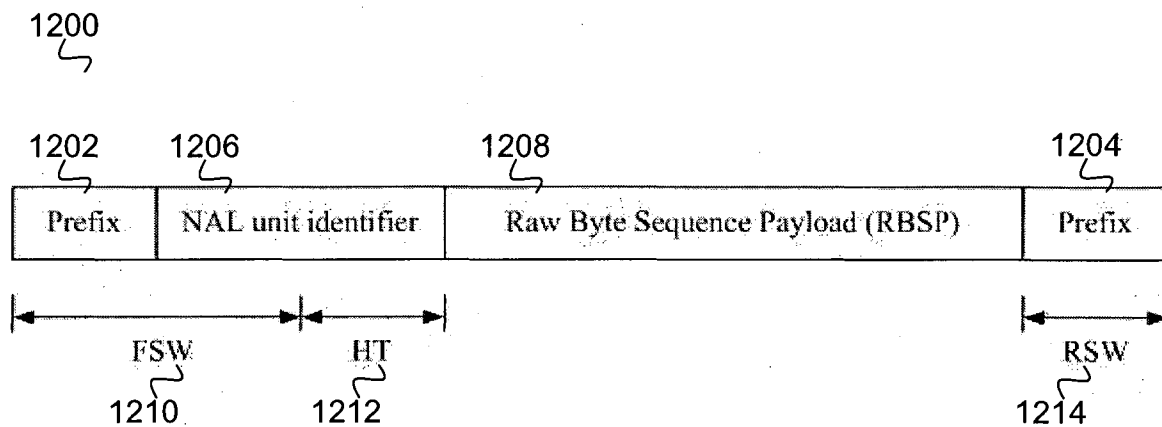


FIG. 12

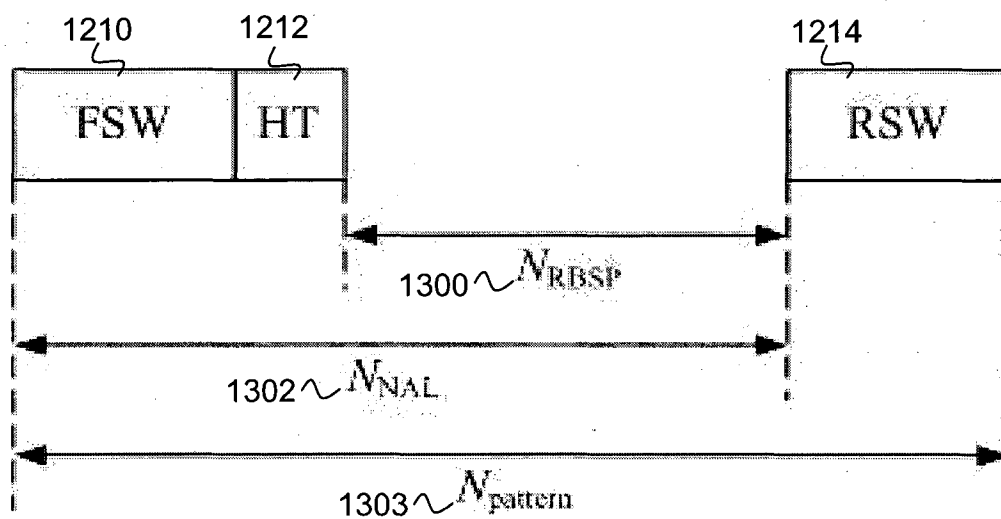


FIG. 13

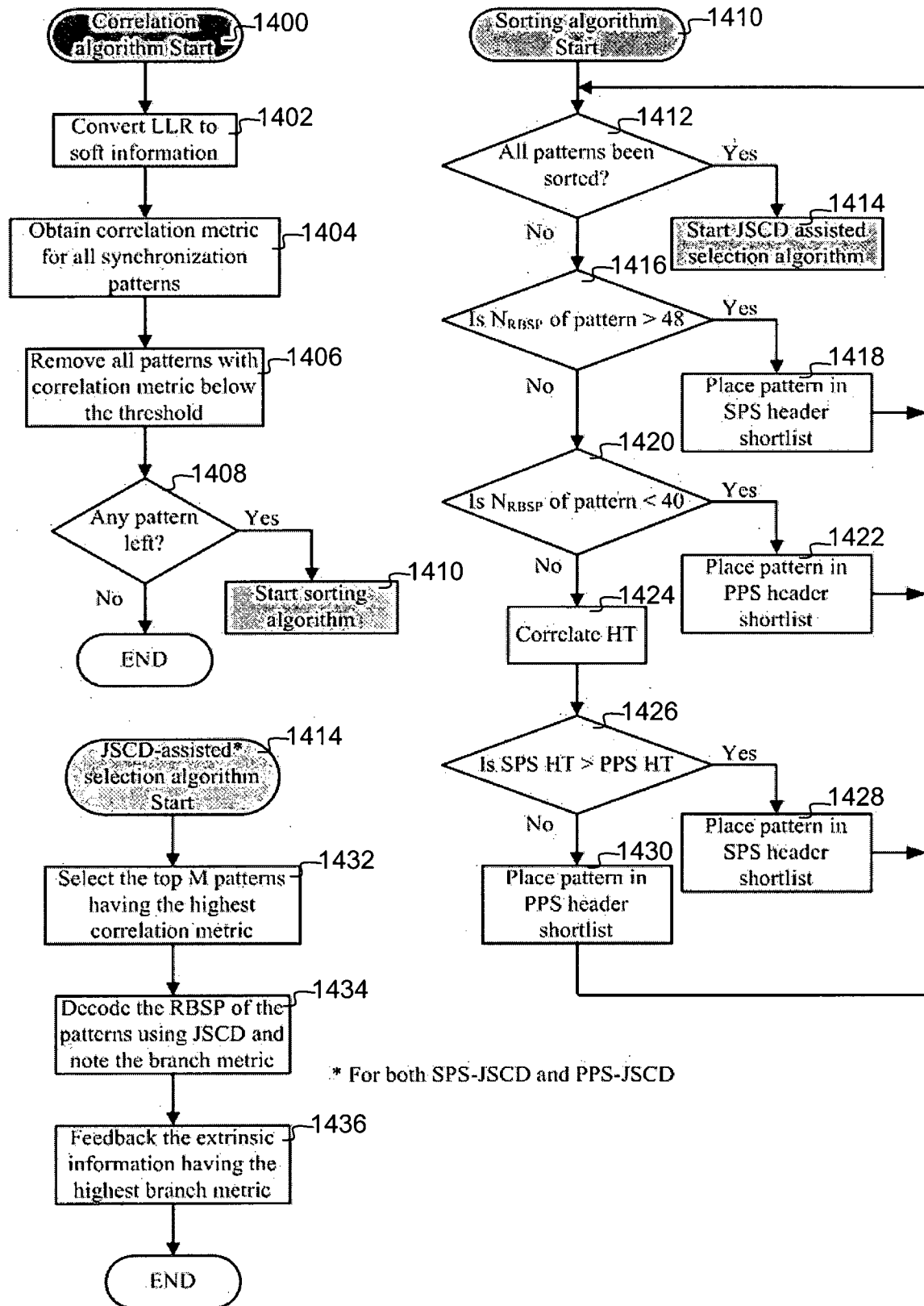


FIG. 14

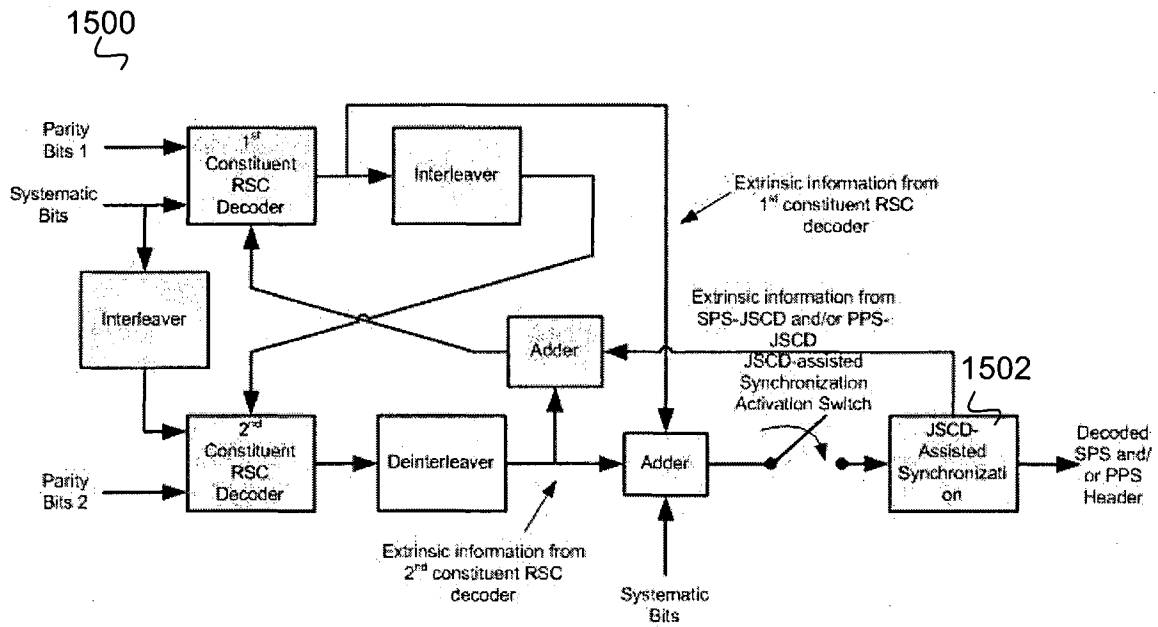


FIG. 15

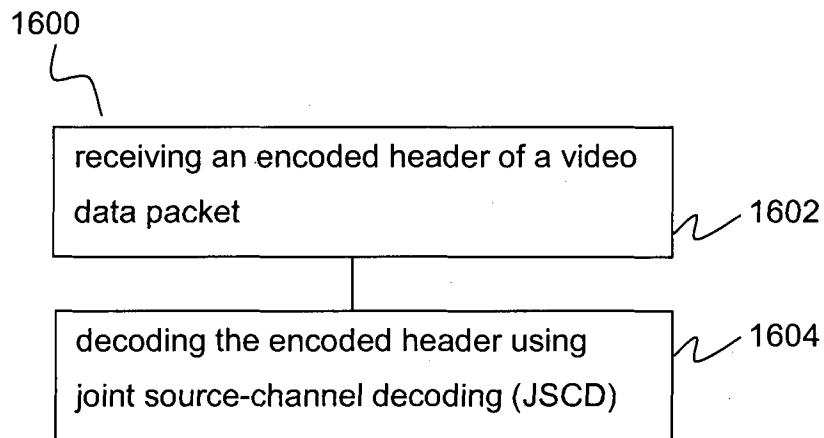


FIG. 16

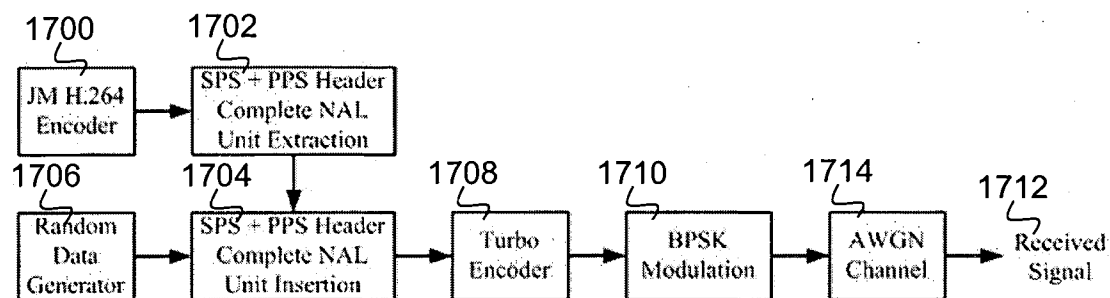


FIG. 17

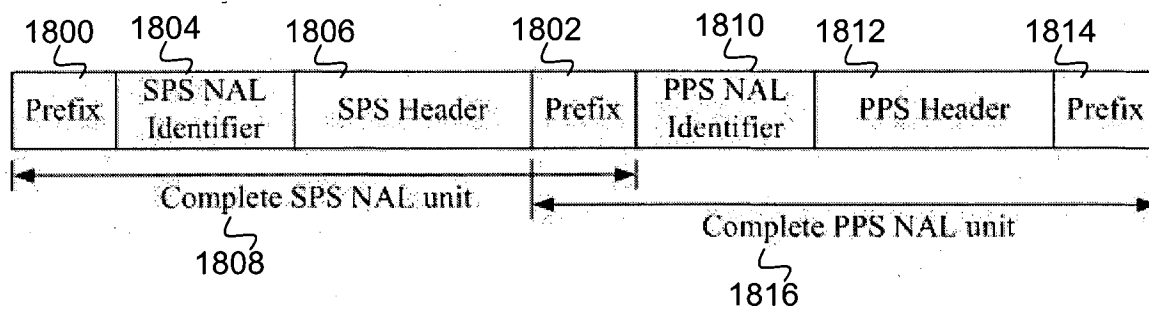


FIG. 18

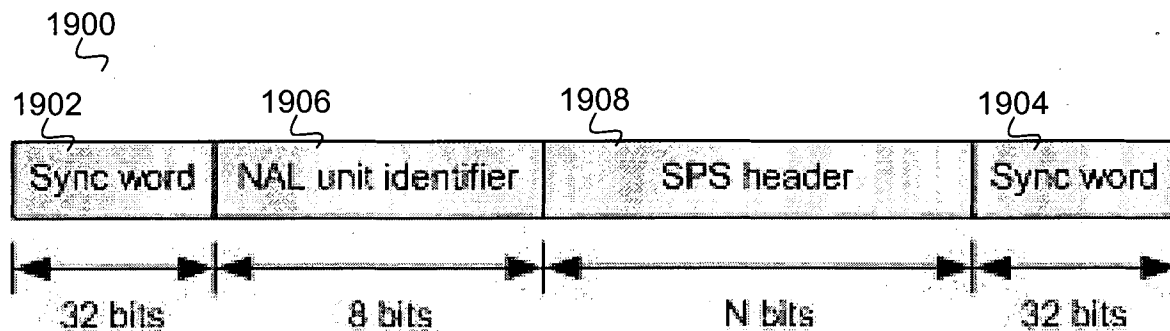


FIG. 19

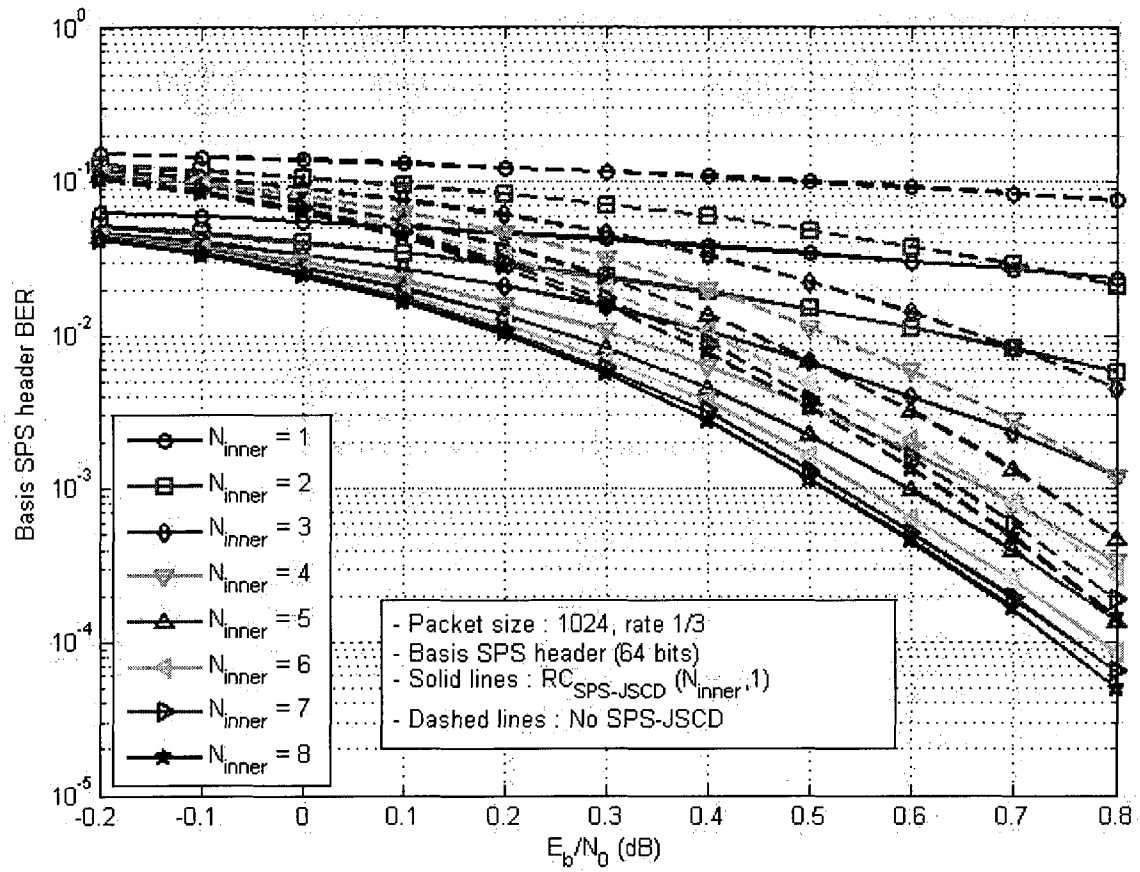


FIG. 20

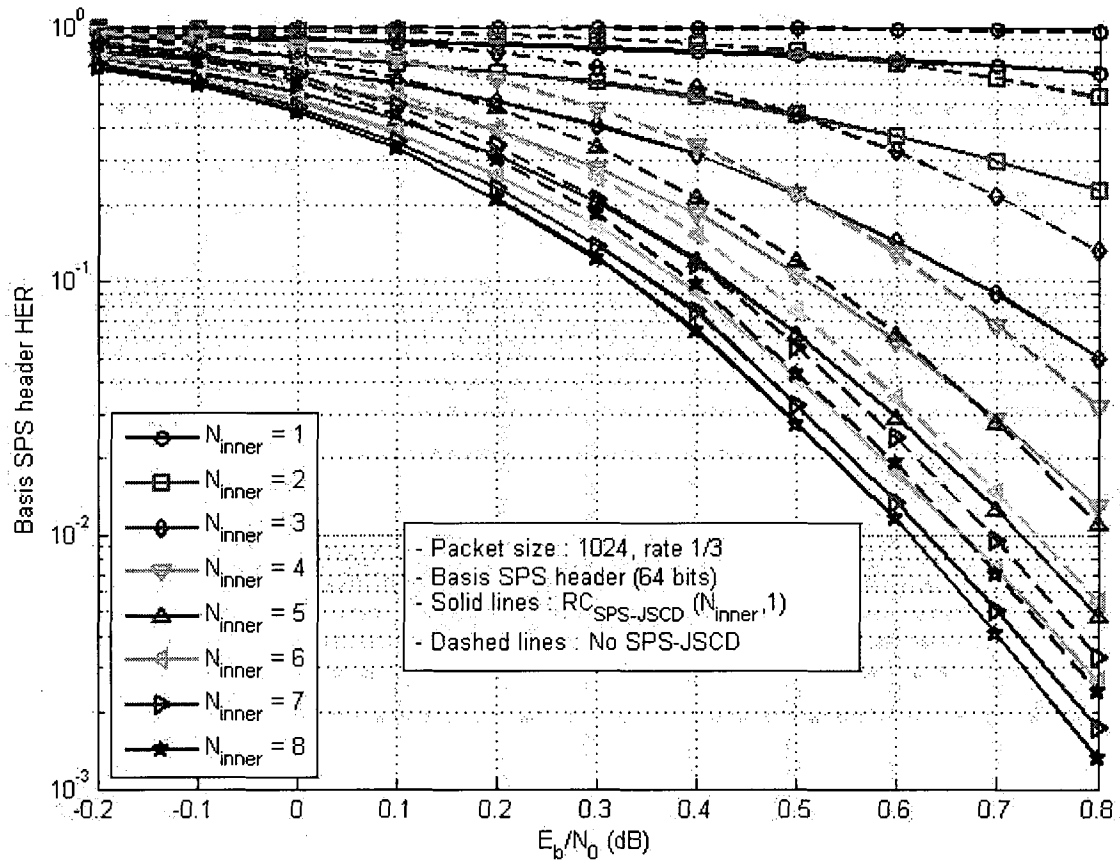


FIG. 21

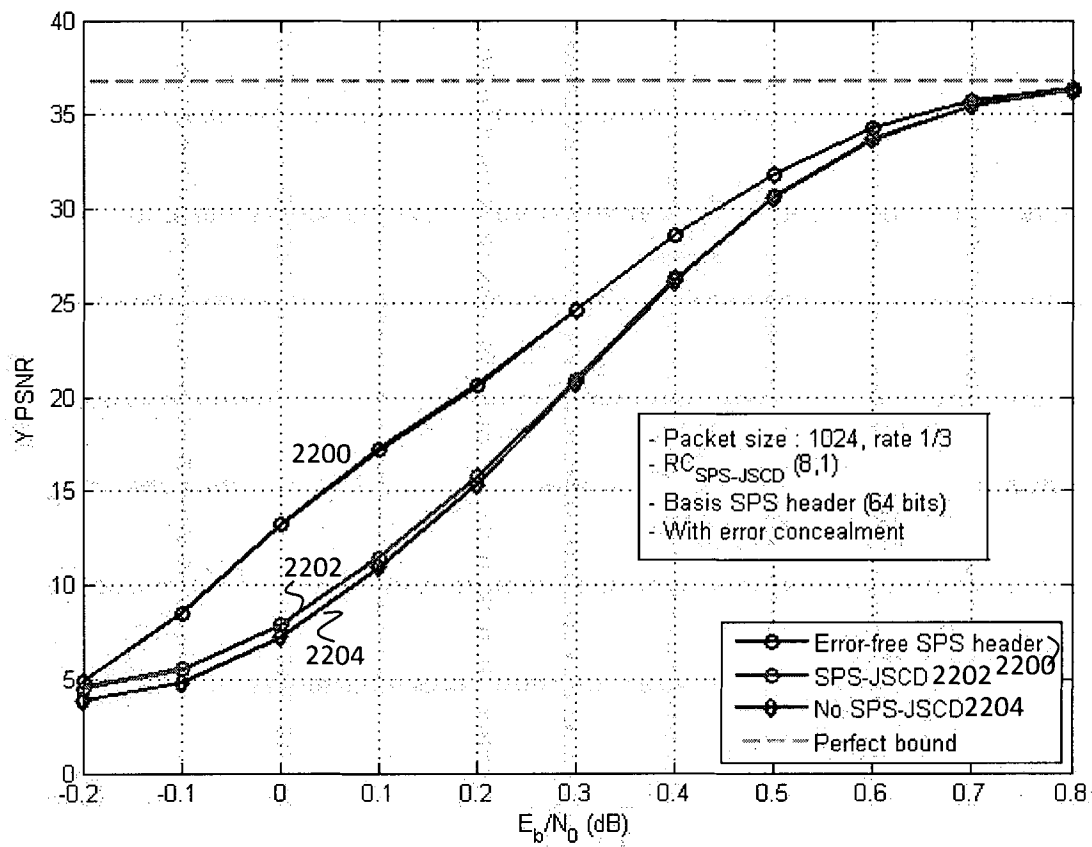


FIG. 22

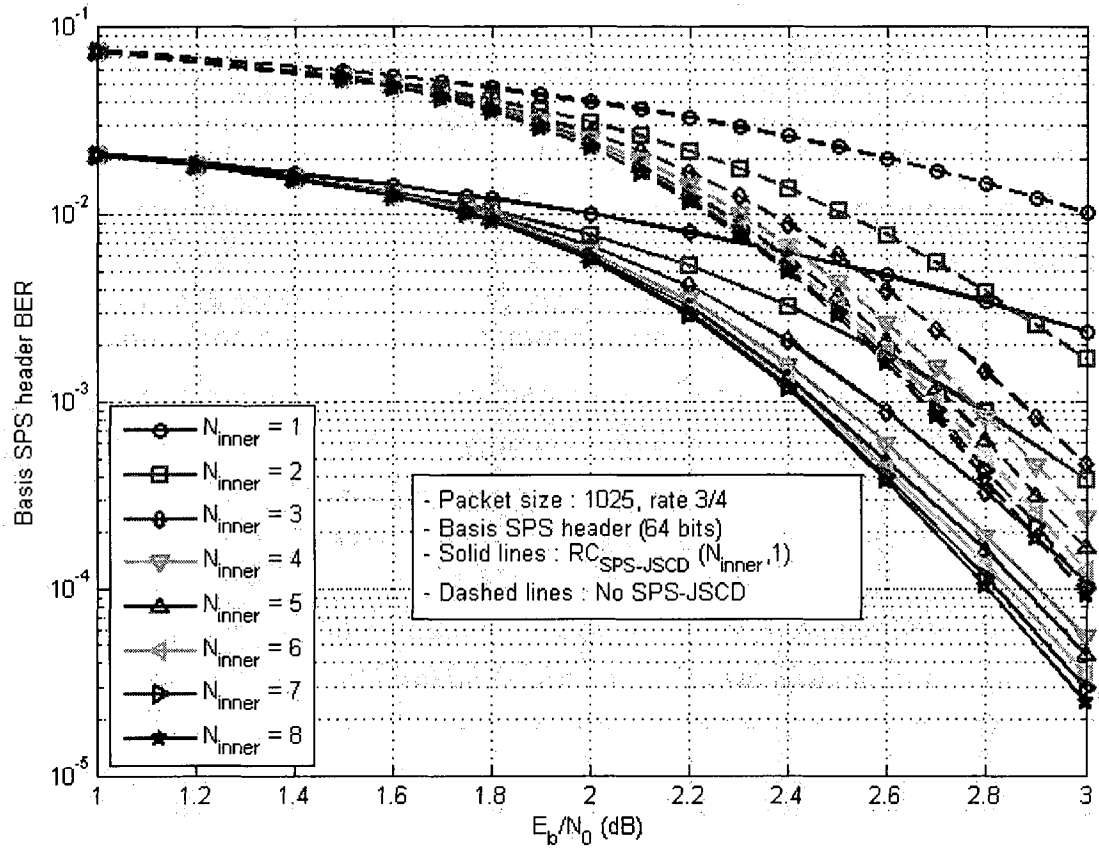


FIG. 23

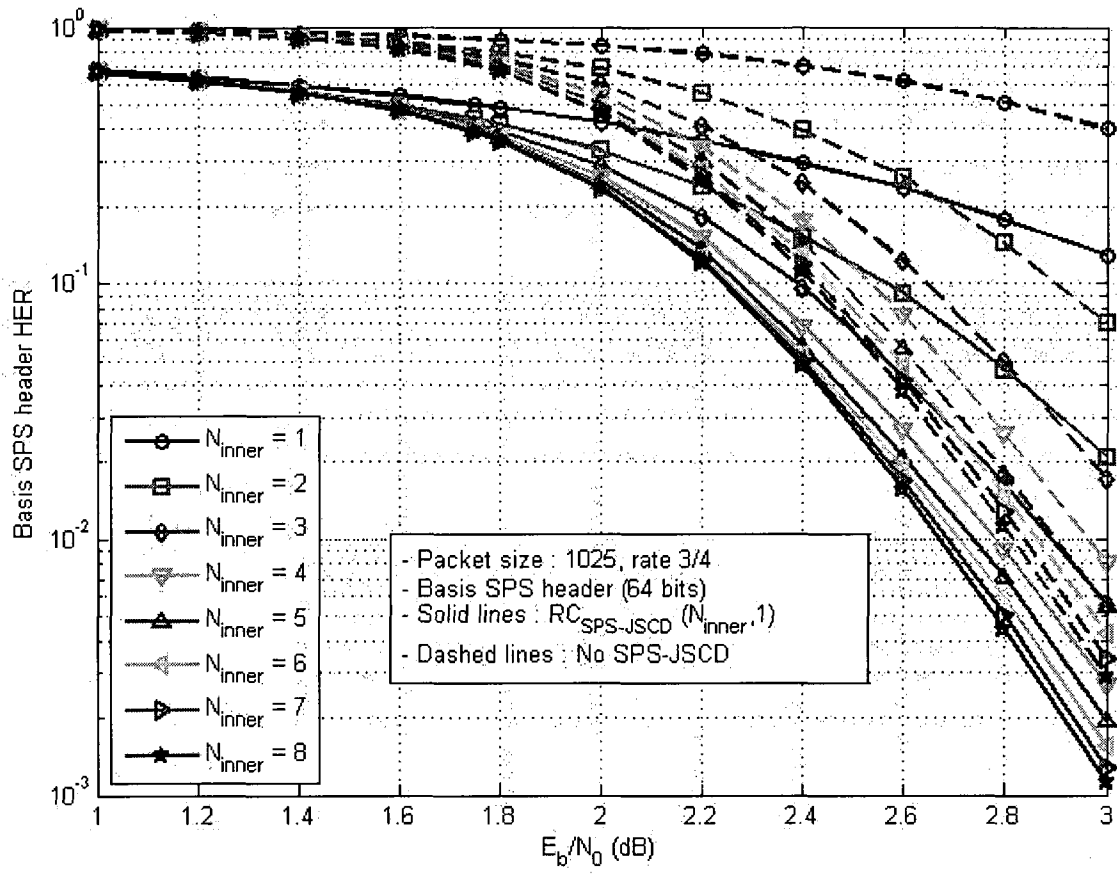


FIG. 24

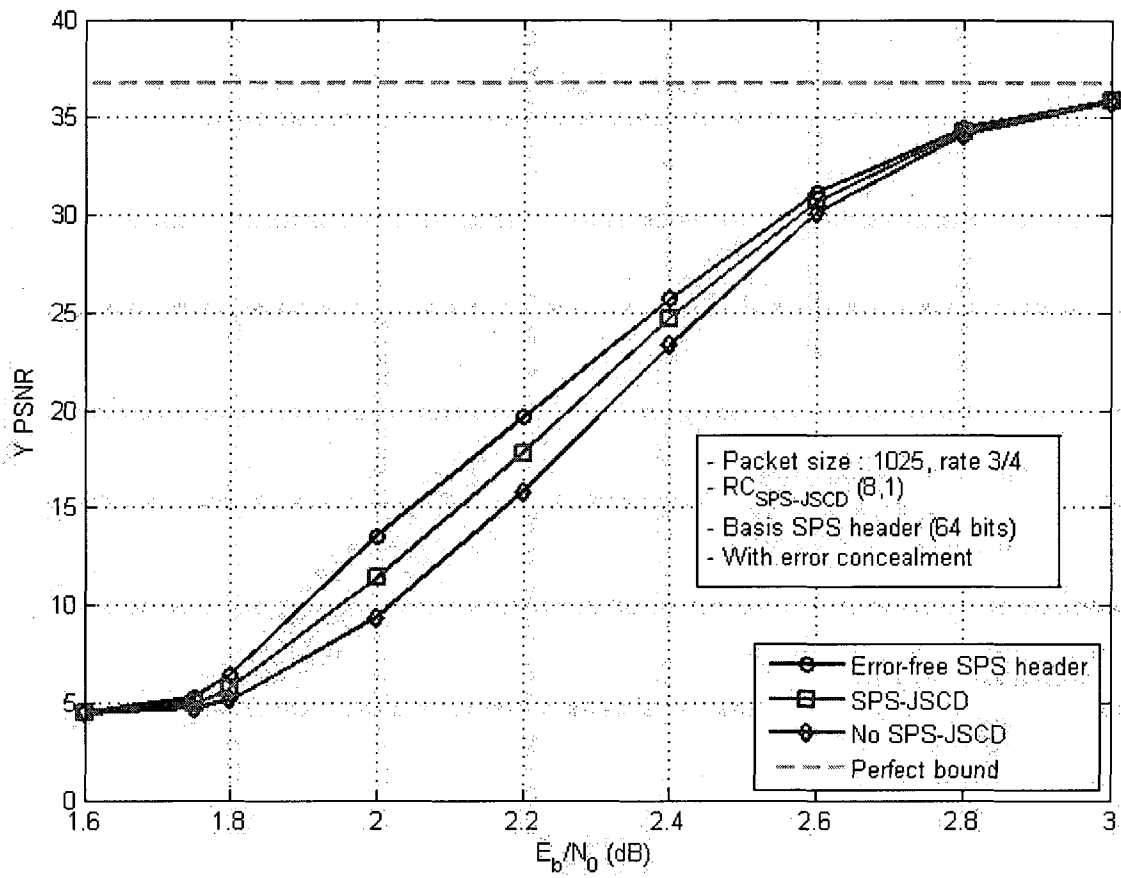


FIG. 25

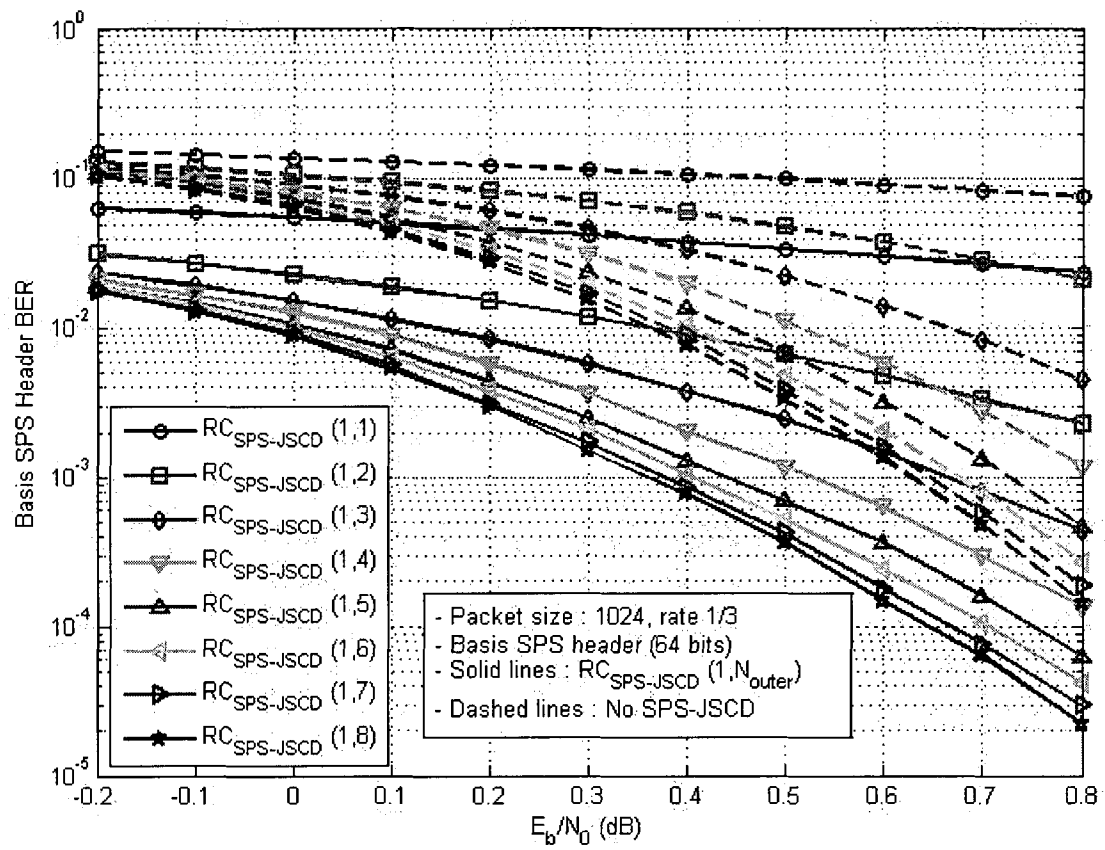


FIG. 26

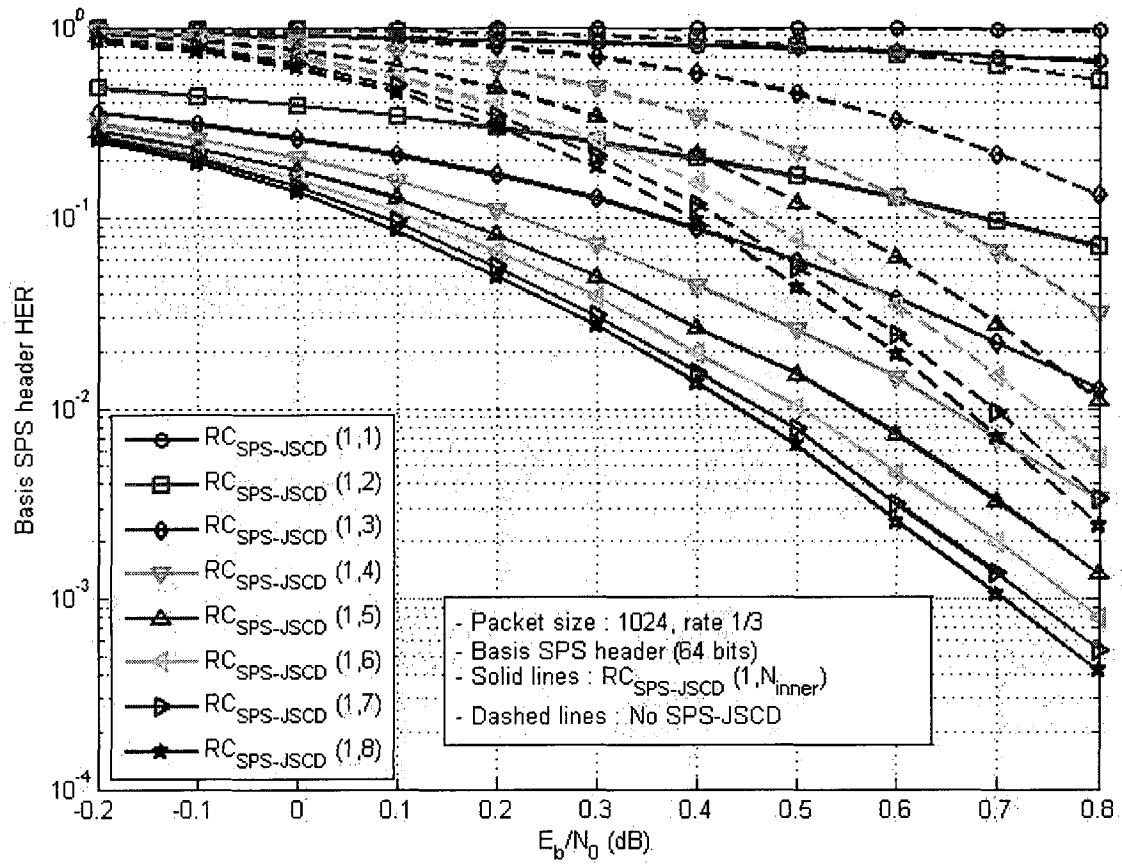


FIG. 27

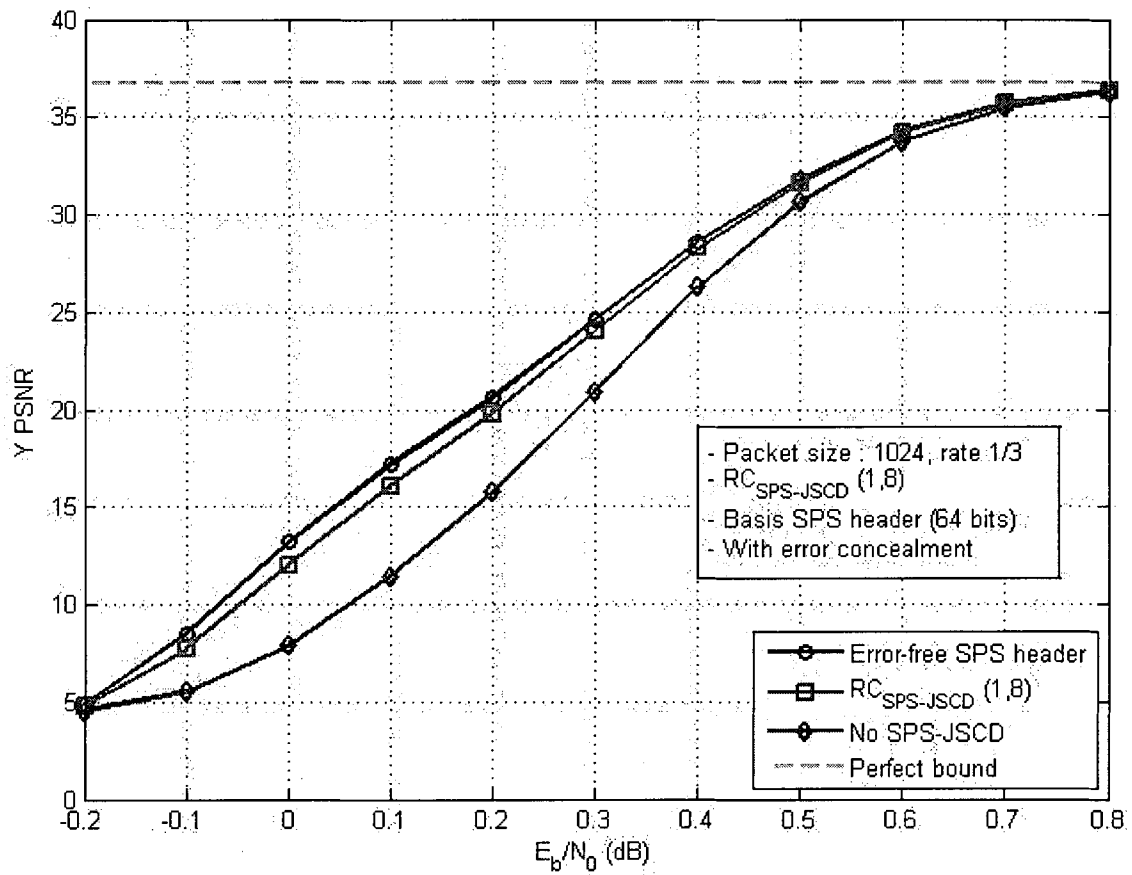


FIG. 28

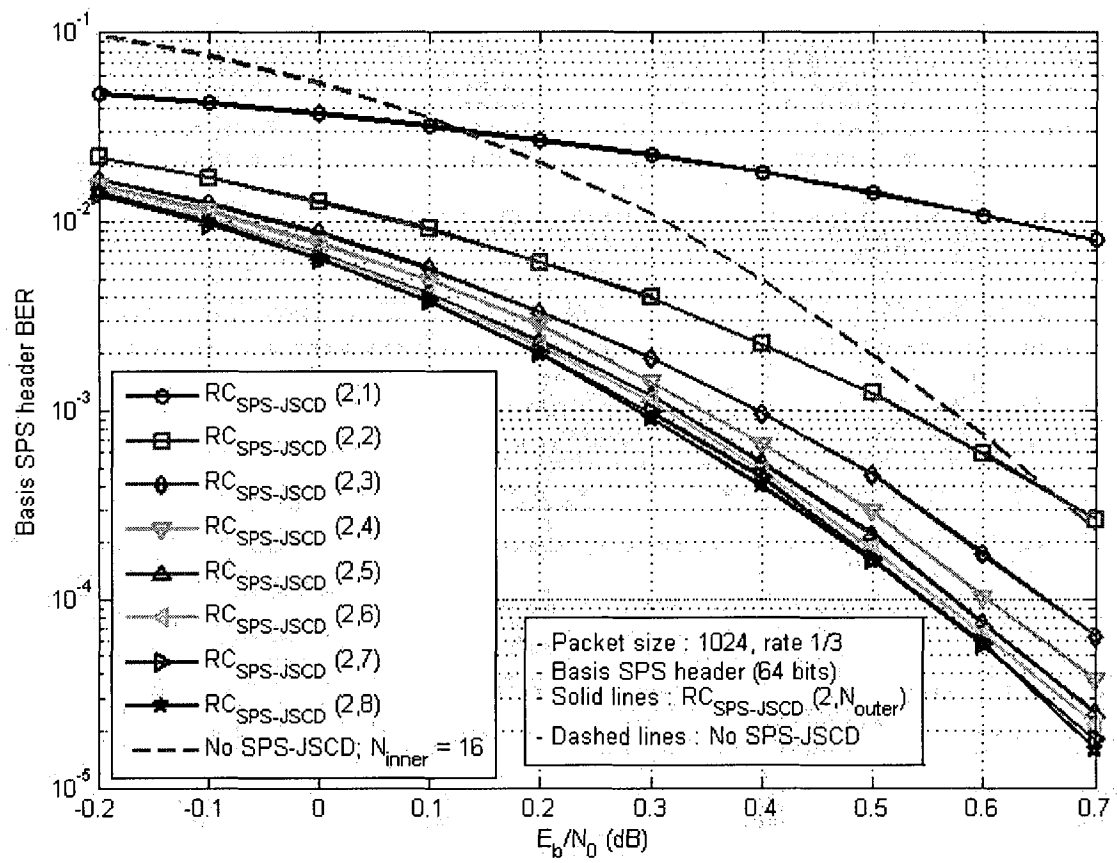


FIG. 29

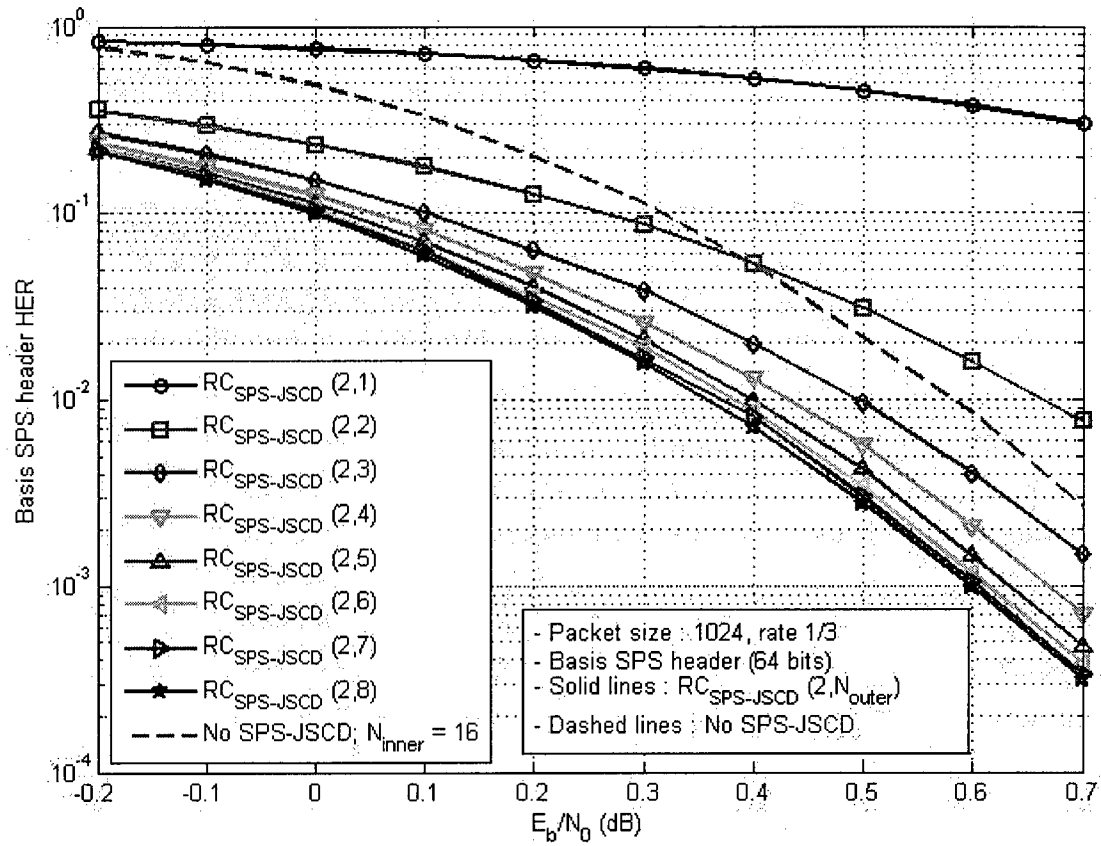


FIG. 30

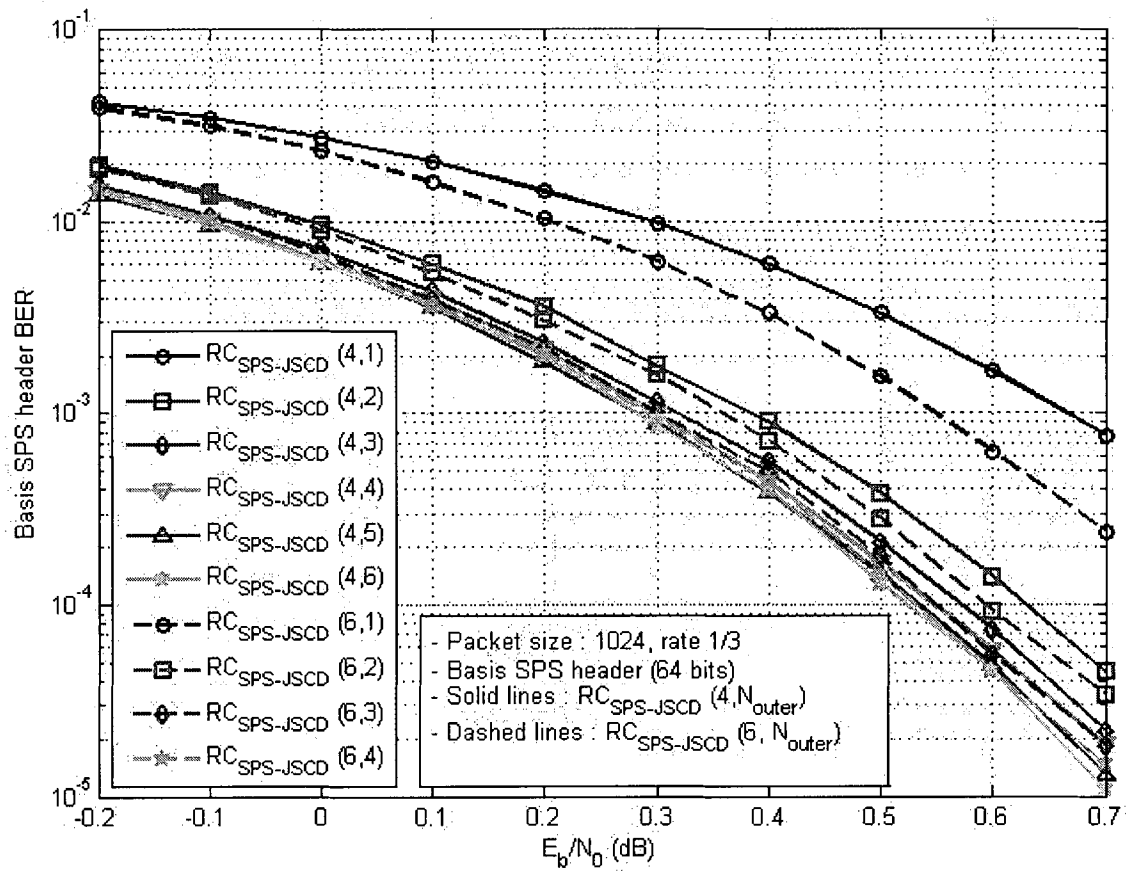


FIG. 31

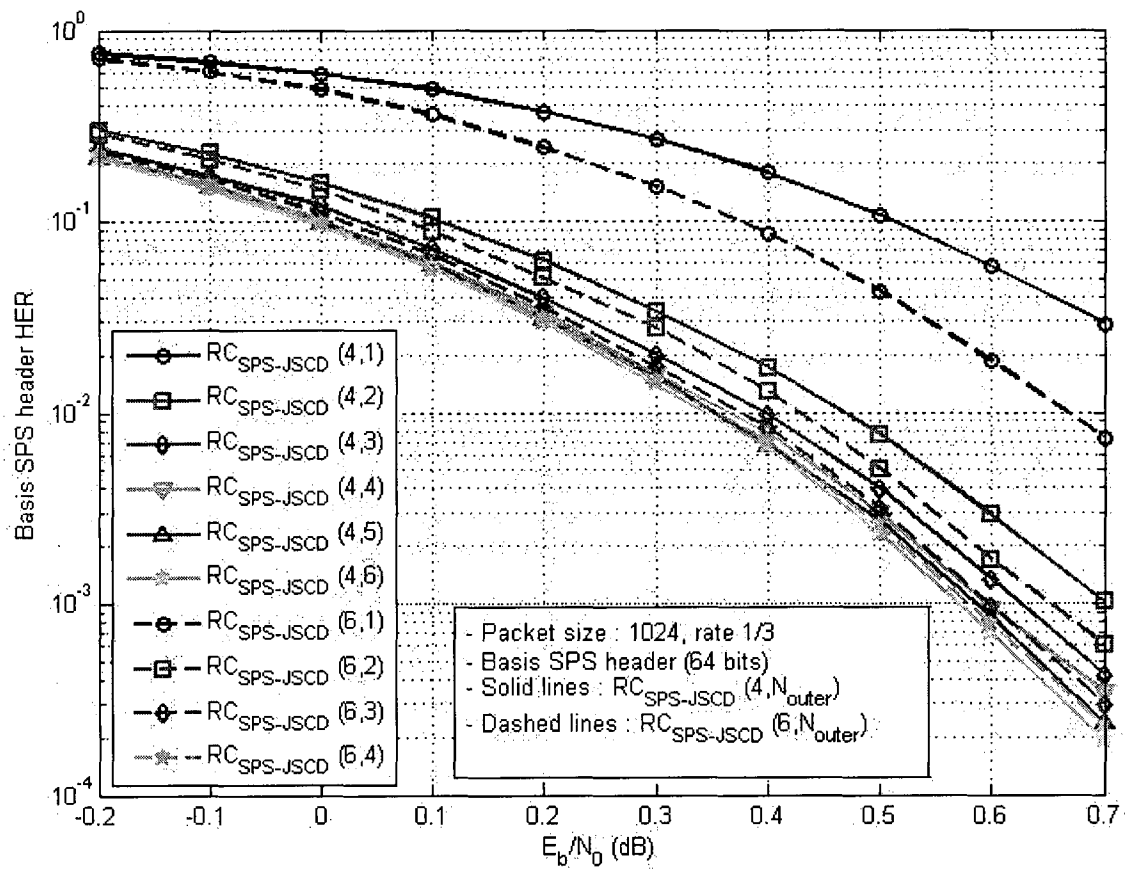


FIG. 32

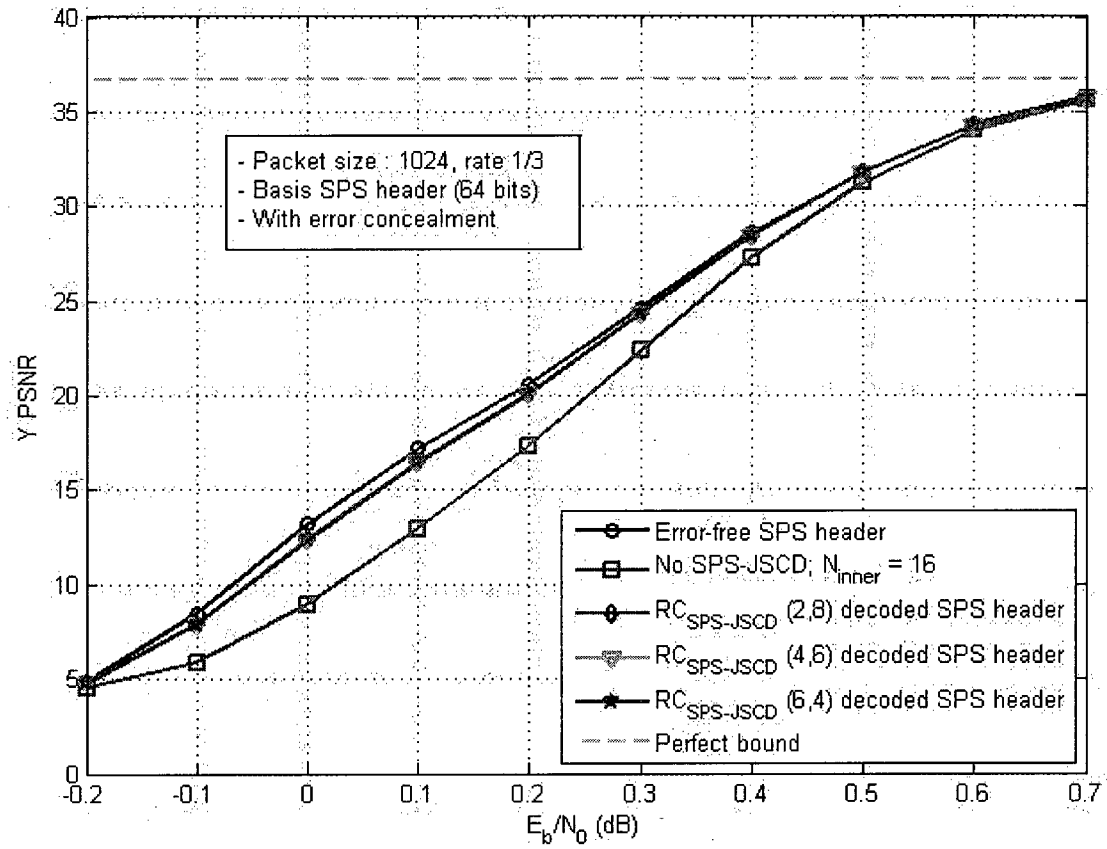


FIG. 33

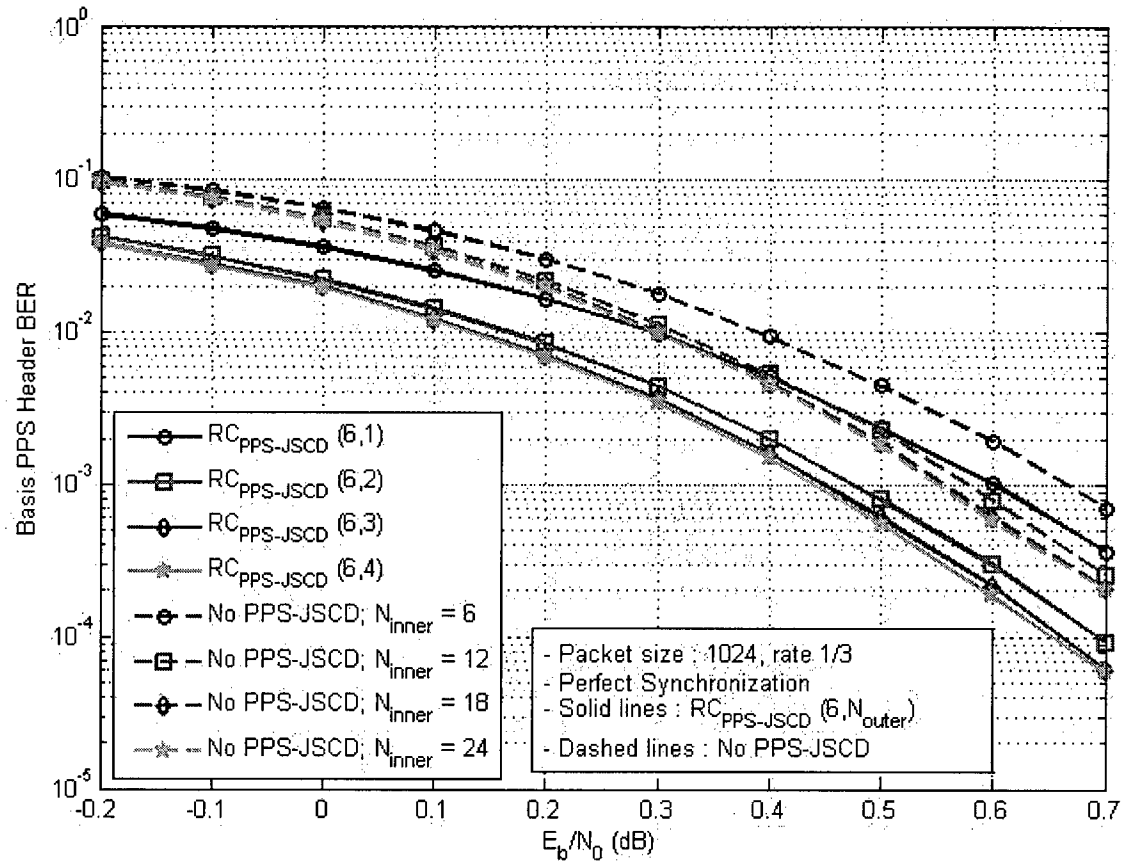


FIG. 34

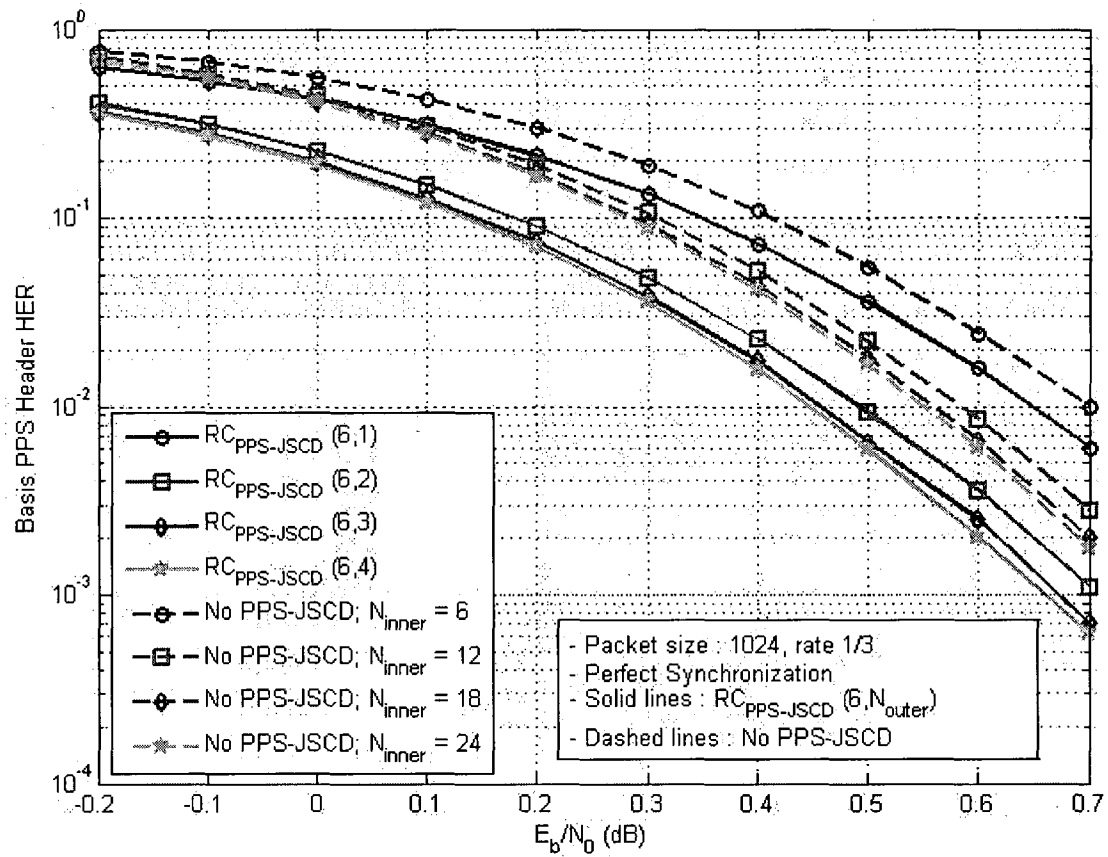


FIG. 35

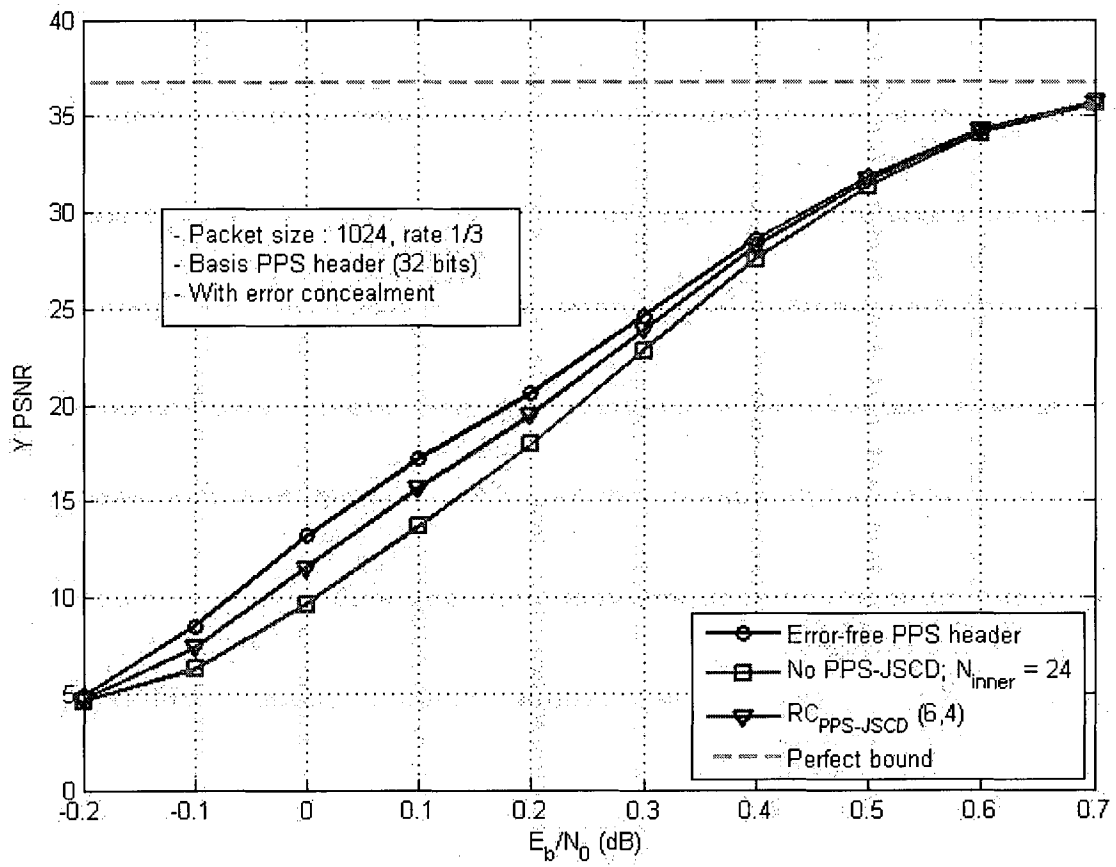


FIG. 36

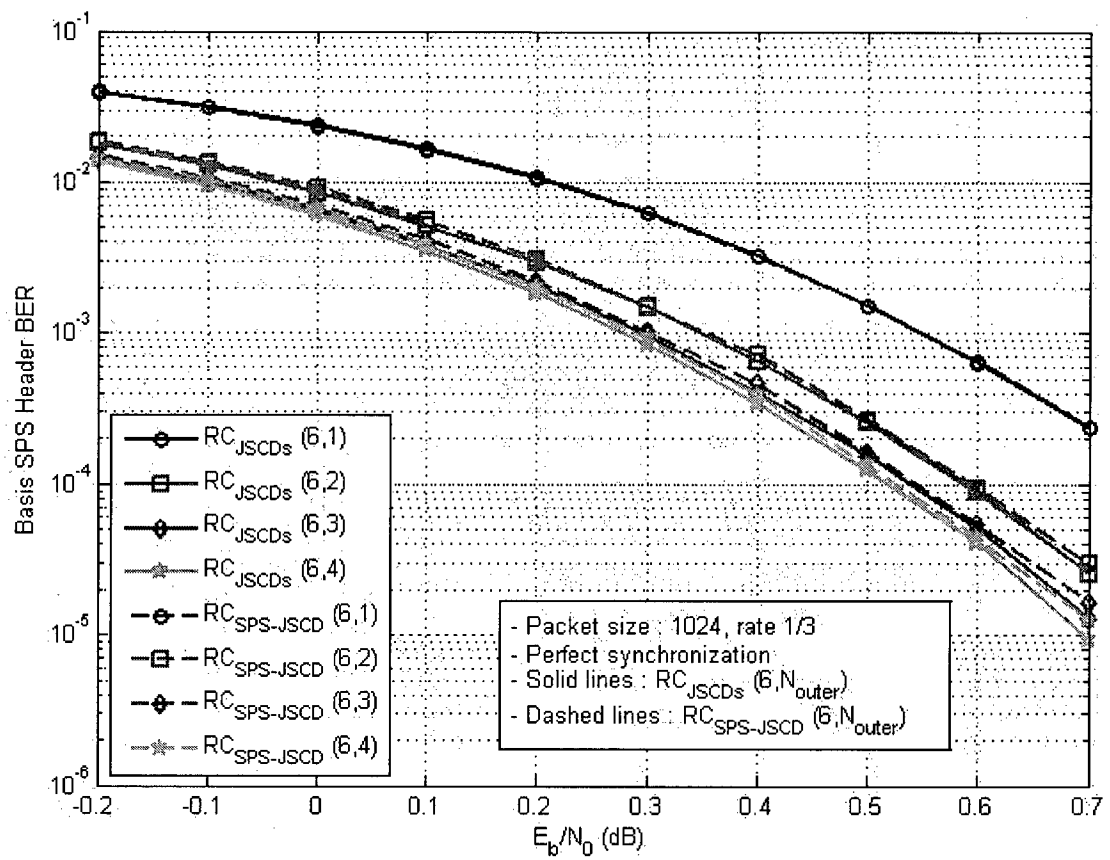


FIG. 37

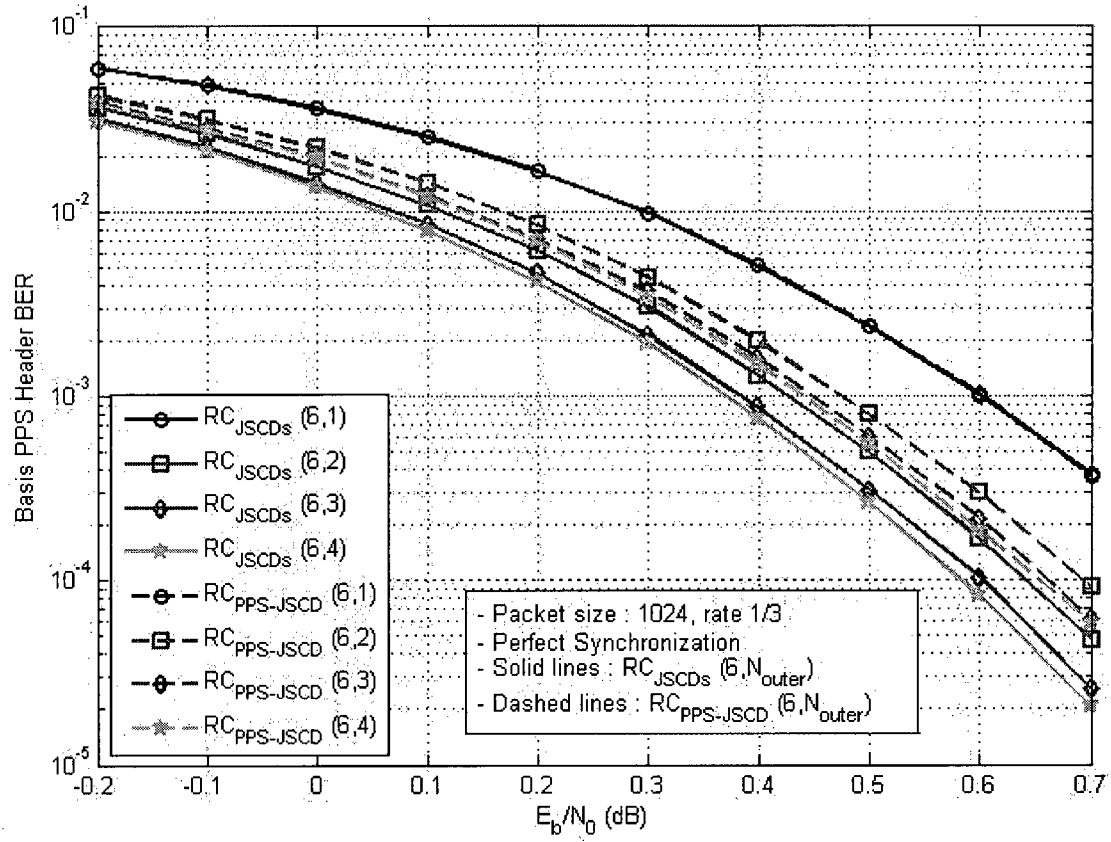


FIG. 38

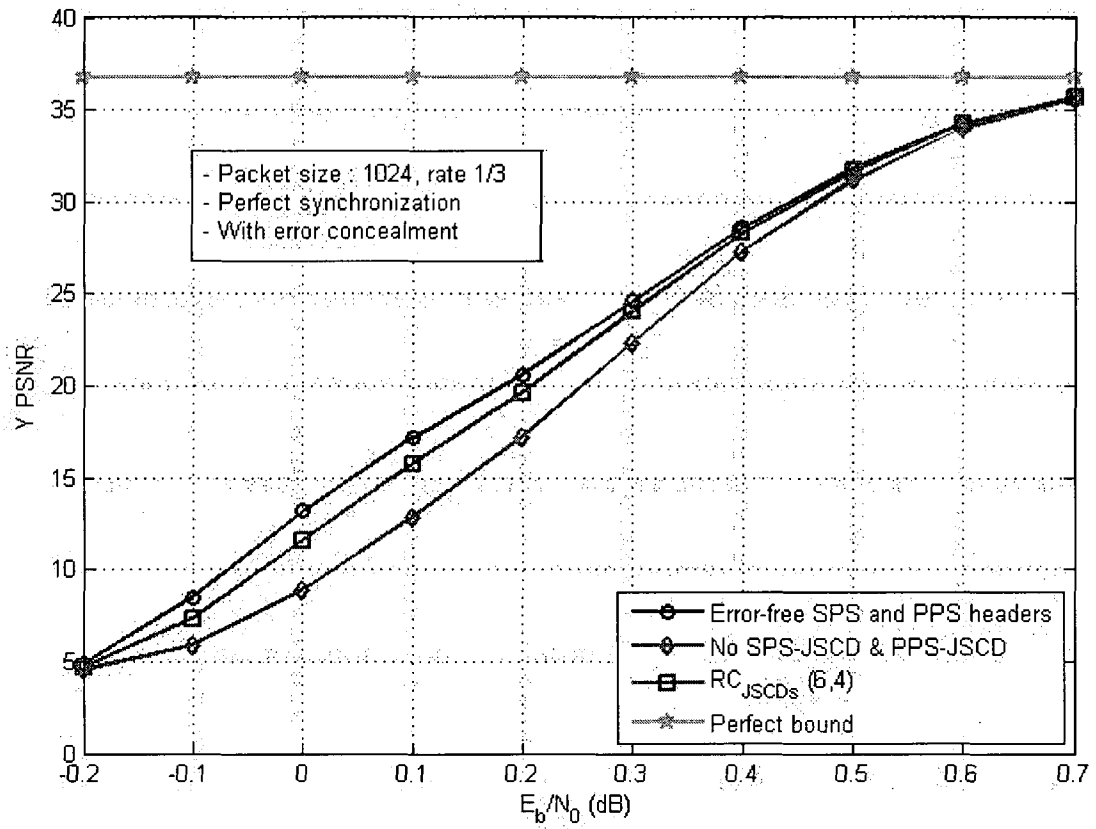


FIG. 39

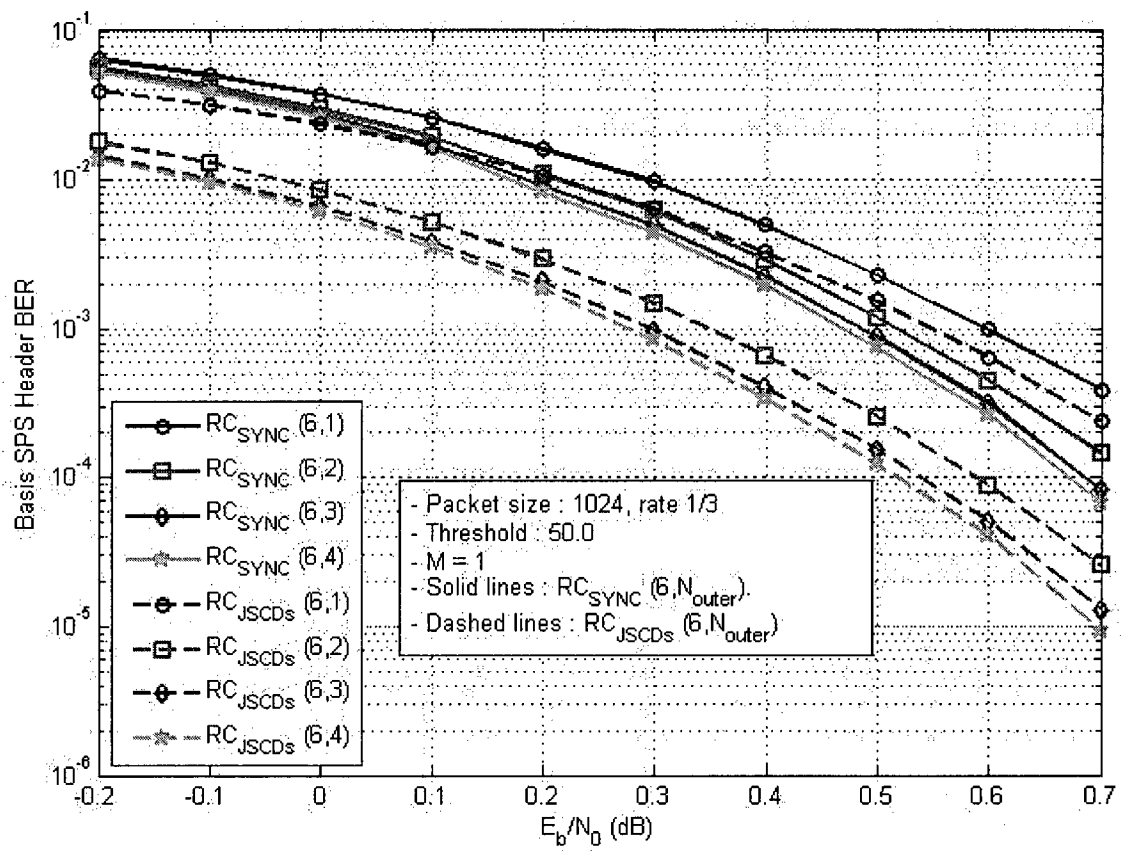


FIG. 40

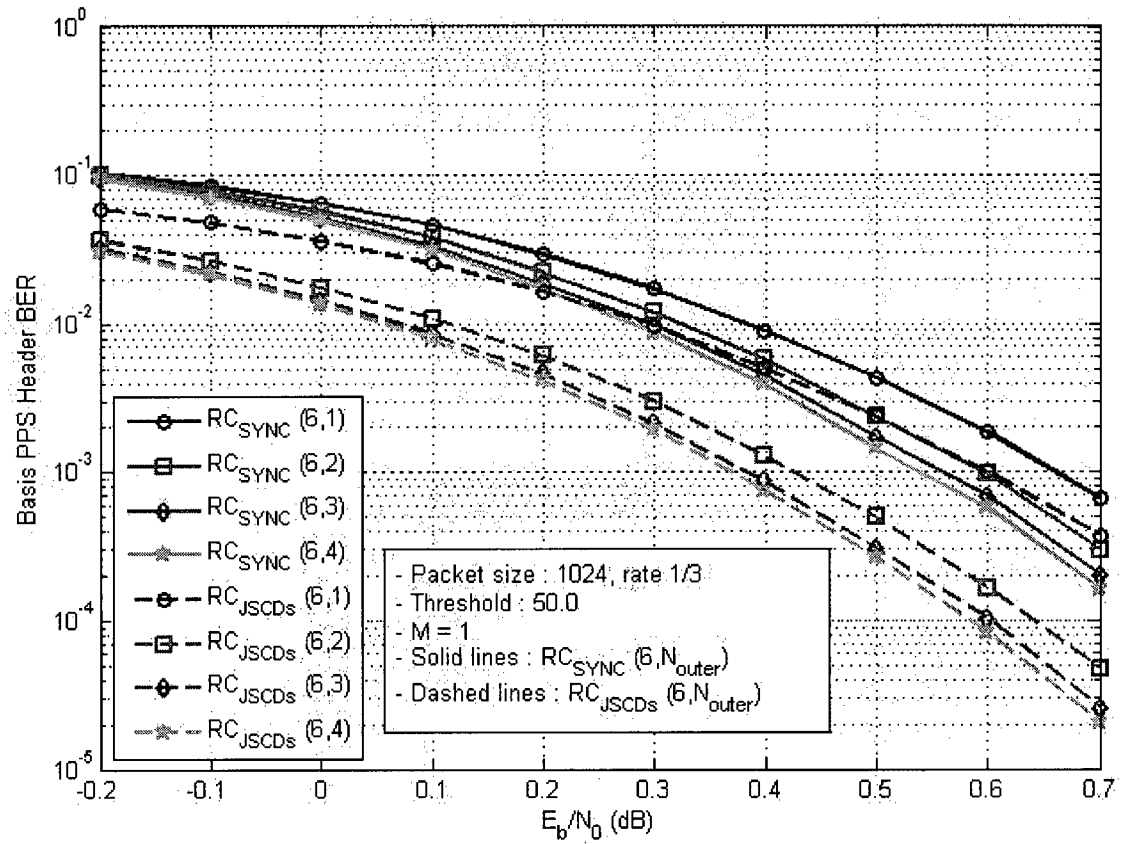


FIG. 41

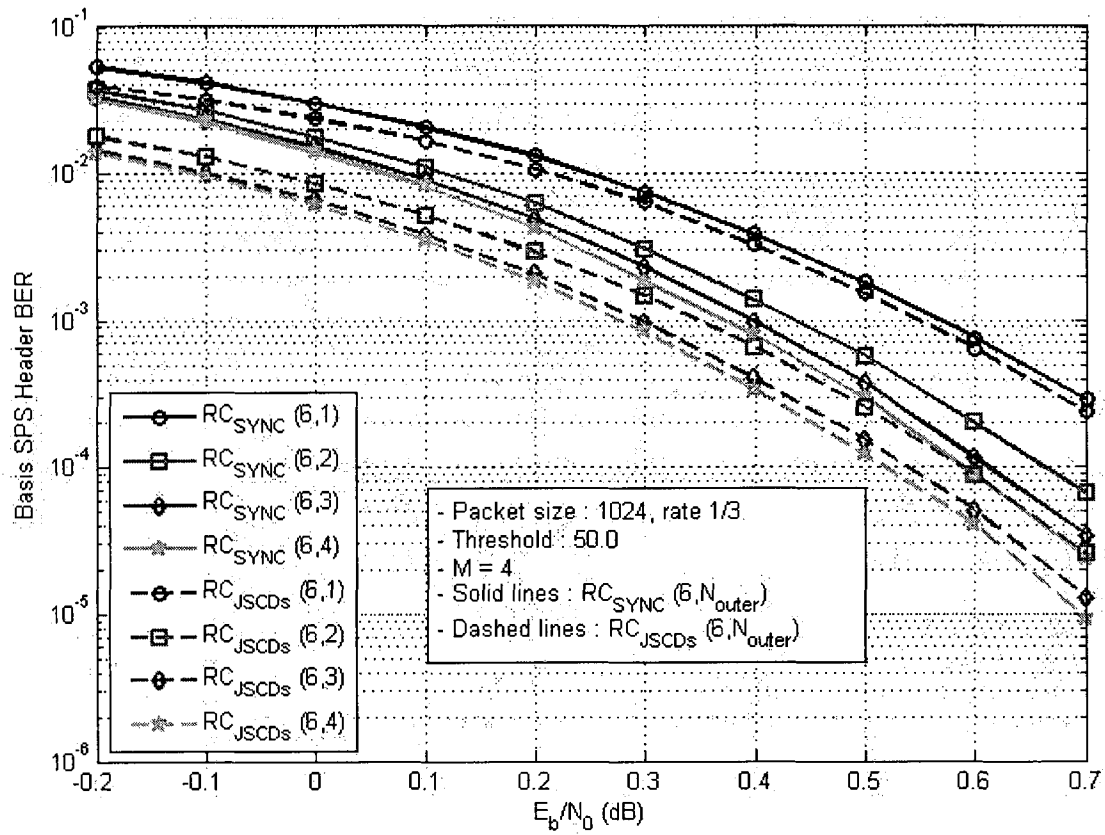


FIG. 42

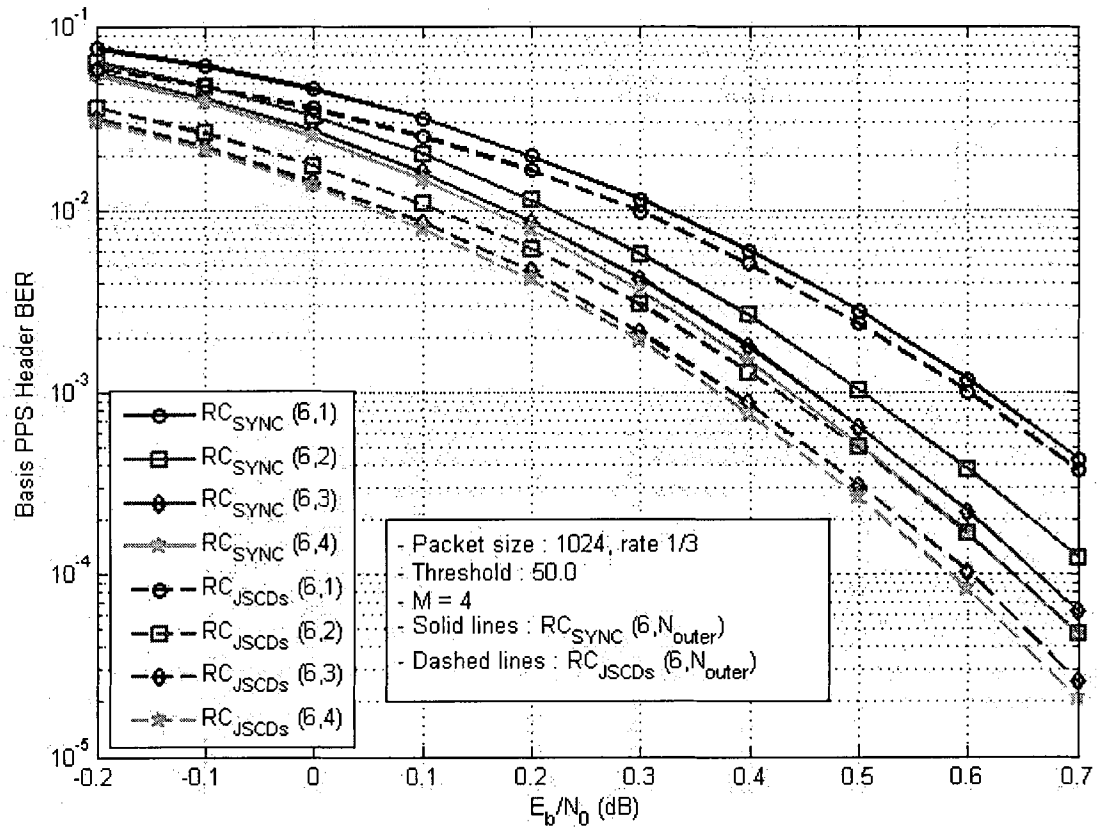


FIG. 43

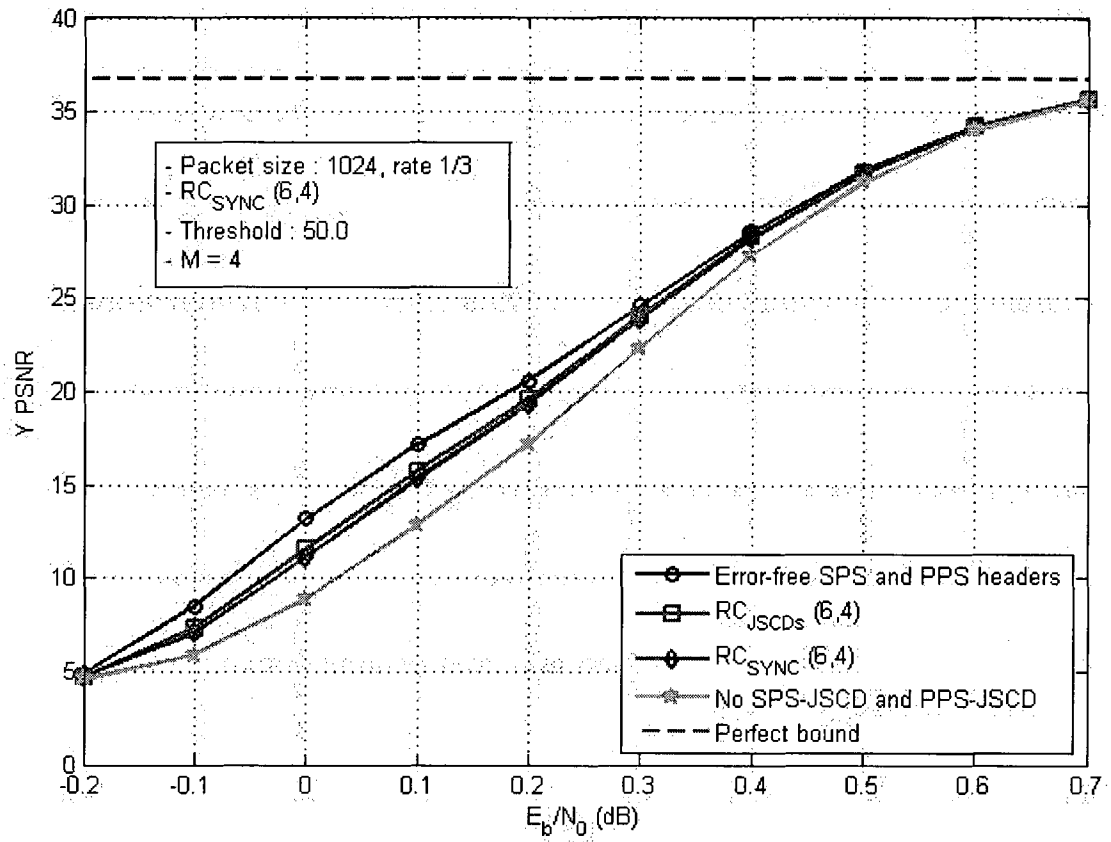


FIG. 44

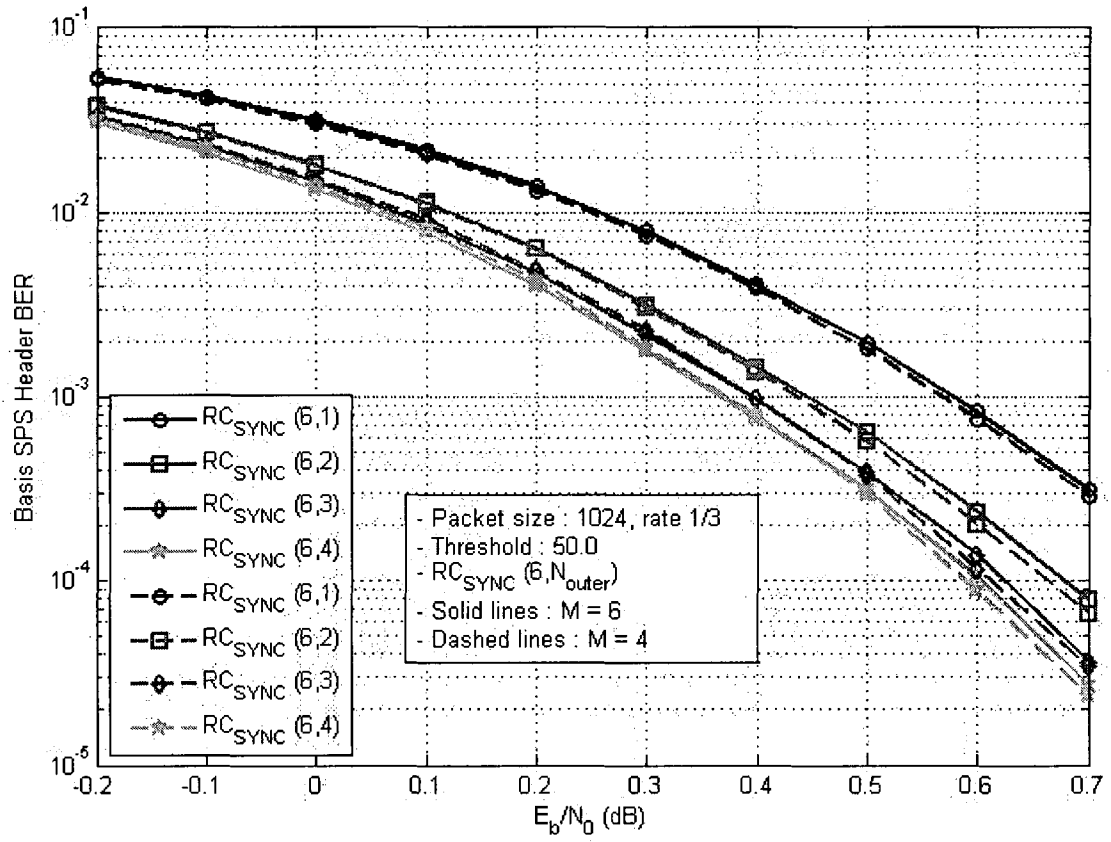


FIG. 45

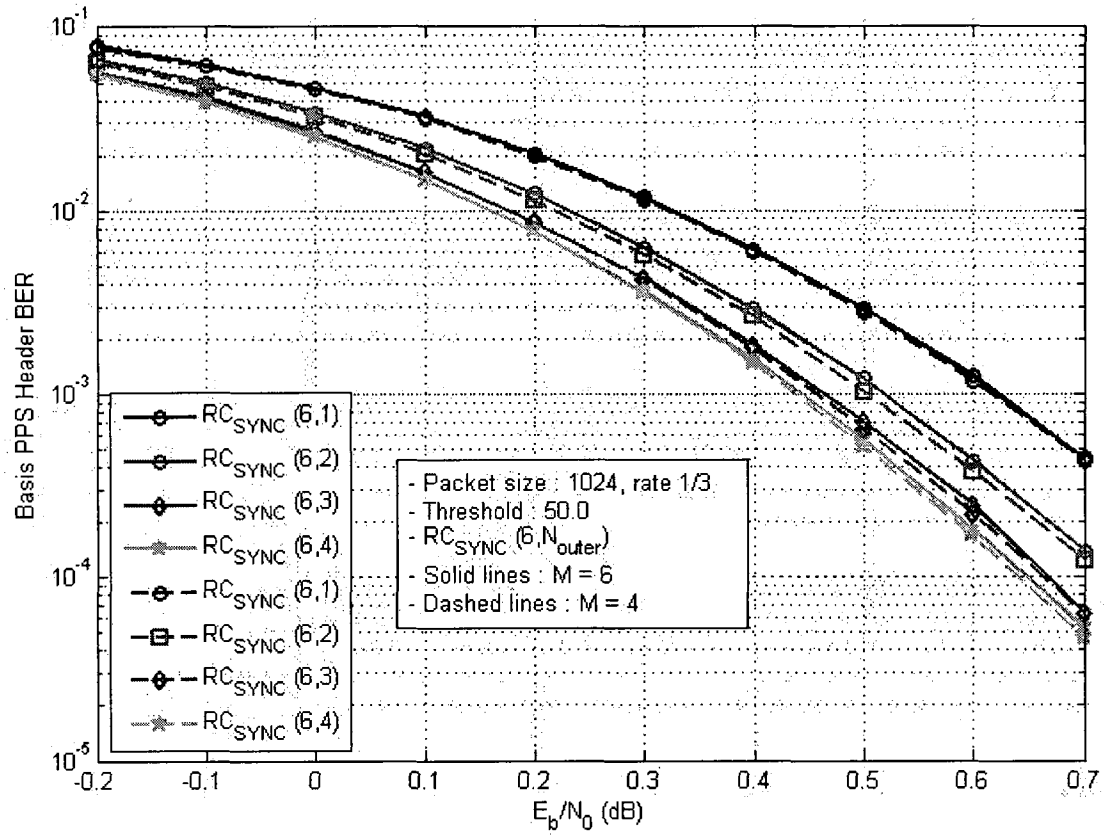


FIG. 46

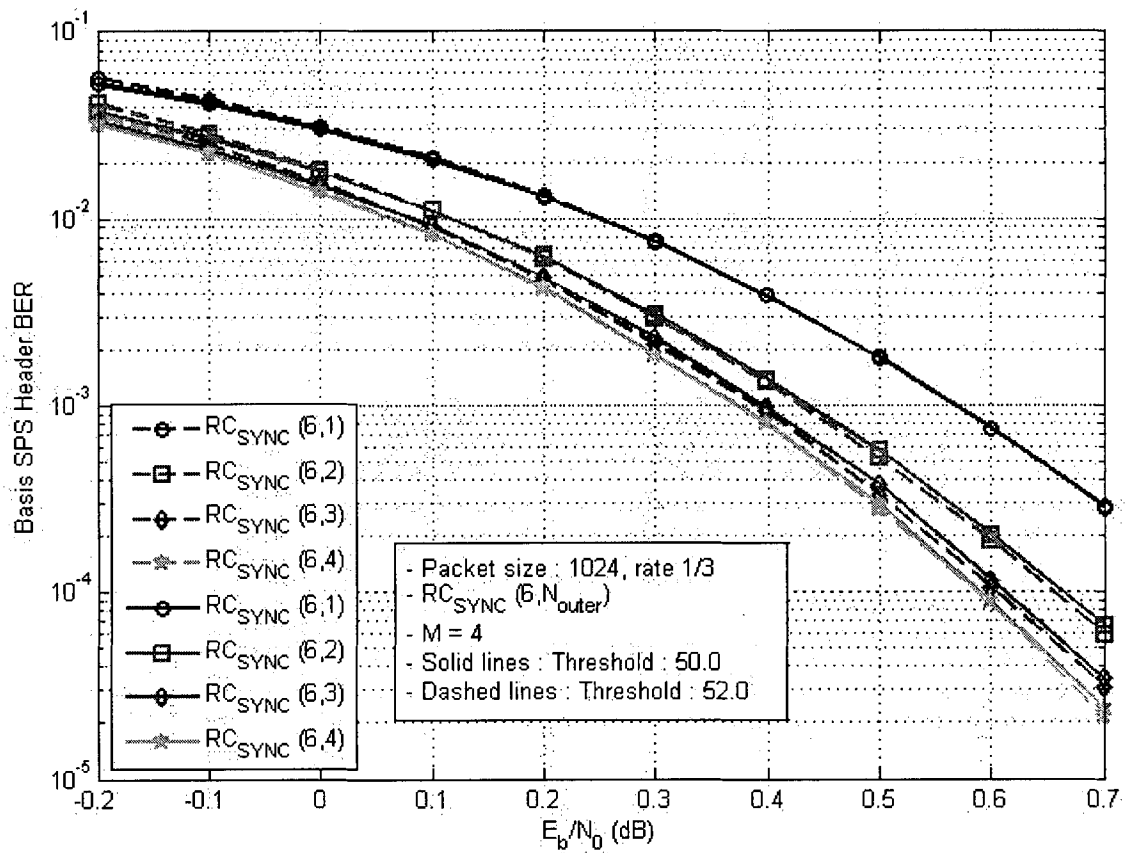


FIG. 47

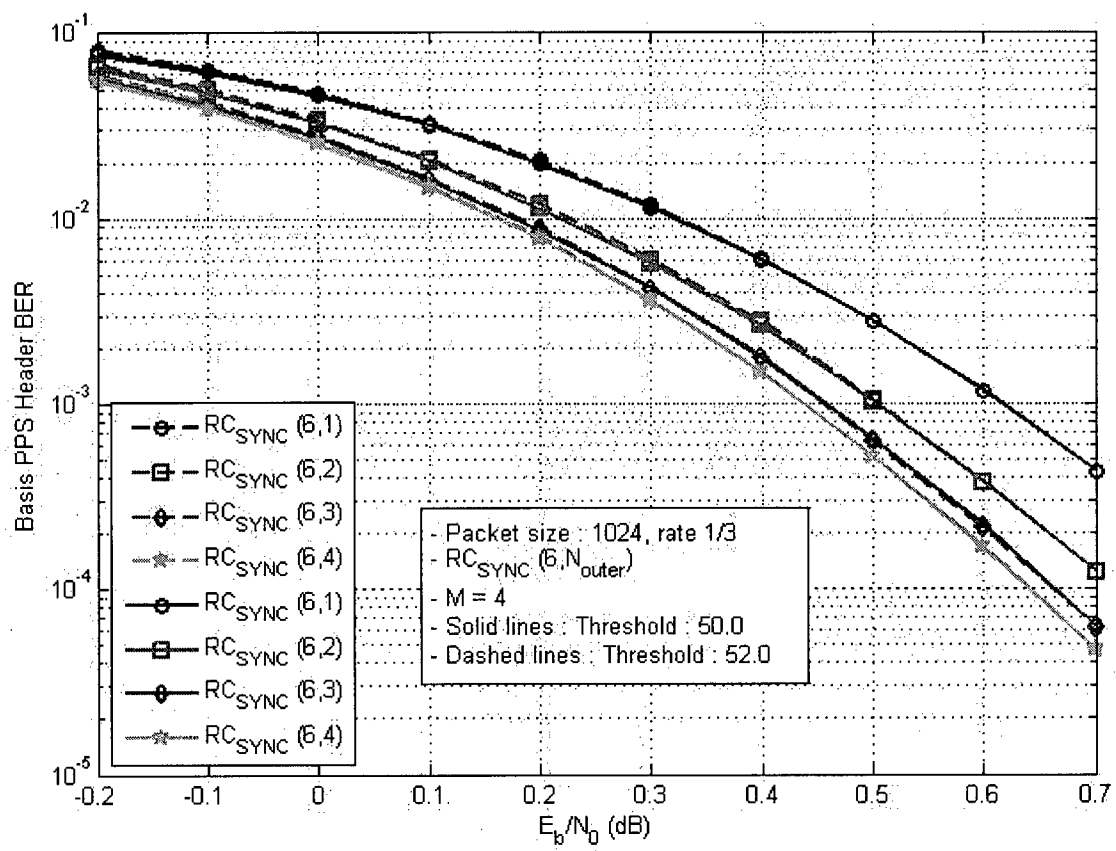


FIG. 48

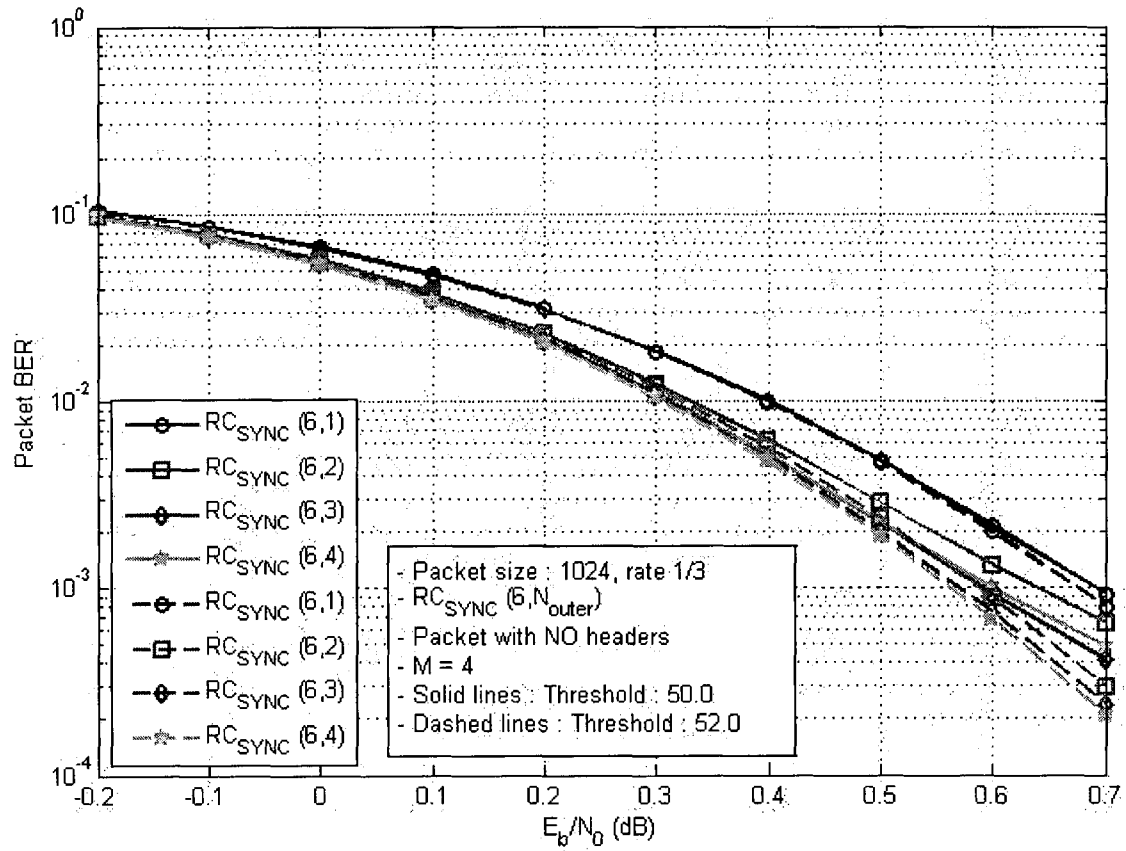


FIG. 49

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SG2012/000333

A. CLASSIFICATION OF SUBJECT MATTER

H04N 21/6336 (2011.01)

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

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

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

TXTUS1, TXTUS2, TXTUS3, TXTUS4, TXTEP1, TXTGB1, TXTWO1:

joint source channel decoder, header decoder and encoder, source decoder, channel decoder, sequence parameter set, picture parameter set, trellis with other similar terms and synonyms.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	Documents are listed in the continuation of Box C	



Further documents are listed in the continuation of Box C



See patent family annex

* Special categories of cited documents:	
"A" document defining the general state of the art which is not considered to be of particular relevance	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"E" earlier application or patent but published on or after the international filing date	"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O" document referring to an oral disclosure, use, exhibition or other means	"&" document member of the same patent family
"P" document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search
28 November 2012

Date of mailing of the international search report
28 November 2012

Name and mailing address of the ISA/AU

AUSTRALIAN PATENT OFFICE
PO BOX 200, WODEN ACT 2606, AUSTRALIA
Email address: pct@ipaustalia.gov.au
Facsimile No.: +61 2 6283 7999

Authorised officer

Boris Cetinich
AUSTRALIAN PATENT OFFICE
(ISO 9001 Quality Certified Service)
Telephone No. 0399359619

INTERNATIONAL SEARCH REPORT		International application No.
C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		PCT/SG2012/000333
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X Y	US 2011/0083032 A1 (MARIN et al.) 07 April 2011 paragraphs 213-217 and figures 15-16 as above	1, 3, 28-30 and 32 2, 9, 16, 18, 24, 31, 28, 47 and 49
X Y	KIEFFER M. et al. - "Joint Protocol and Channel Decoding: an Overview" - Future Network and MobileSummit 2010 Conference Proceedings Paul Cunningham and Miriam Cunningham (Eds), IIMC - International Information Management Corporation, 2010 sections 1-2 as above	1, 3-4, 28-30 and 32-33 2, 9, 16, 18, 24, 31, 28, 47 and 49
X Y	MÉRIAUX F. et al. - "ROBUST IP AND UDP-LITE HEADER RECOVERY FOR PACKETIZED MULTIMEDIA TRANSMISSION" - International Conference on Acoustics, Speech, and Signal Processing - Dallas, Texas : United States (1-21 December 2010) section 1 as above	1, 4-5, 28-30, 33-34 and 42 2, 9, 16, 18, 24, 31, 28, 47 and 49
X Y	MARIN C. et al. - "Robust MAC-Lite and Soft Header Recovery for Packetized Multimedia Transmission" - IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. 58, NO. 3, March 2010 sections I and VII as above	1, 3-4, 28-30 and 32-33 2, 9, 16, 18, 24, 31, 28, 47 and 49
X Y	HU R. et al. - "ROBUST CRITICAL DATA RECOVERY FOR MPEG-4 AAC ENCODED BITSTREAMS" - International Conference on Acoustics, Speech, and Signal Processing - Dallas, Texas : United States (1-21 December 2010) section 1 as above	1, 3-4, 28-30 and 32-33 2, 9, 16, 18, 24, 31, 28, 47 and 49
Y	WENGER S. - "H.264/AVC Over IP" - IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 13, NO. 7, JULY 2003 pages 650 and 651-652	2, 18, 31 and 49
Y	CHEN Q. et al. - "Joint Source-Channel Decoding for MPEG-4 Video Transmission Over Wireless Channels" - IEEE Journal Selected Areas in Communications - VOL. 21, NO. 10, DECEMBER 2003 pages 1-3	9,16, 24, 38 and 47
Form PCT/ISA/210 (fifth sheet) (July 2009)		

INTERNATIONAL SEARCH REPORT Information on patent family members		International application No. PCT/SG2012/000333	
This Annex lists known patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.			
Patent Document/s Cited in Search Report		Patent Family Member/s	
Publication Number	Publication Date	Publication Number	Publication Date
US 2011/0083032 A1	07 Apr 2011	CN 102017497 A	13 Apr 2011
		EP 2117149 A1	11 Nov 2009
		EP 2117149 B1	26 Oct 2011
		JP 2011520372 A	14 Jul 2011
		KR 20110030423 A	23 Mar 2011
		US 2011083032 A1	07 Apr 2011
		WO 2009135882 A1	12 Nov 2009
End of Annex			
Due to data integration issues this family listing may not include 10 digit Australian applications filed since May 2001. Form PCT/ISA/210 (Family Annex)(July 2009)			