



- (51) **International Patent Classification:**
H04N 7/50 (2006.01) **G06T 9/00** (2006.01)
H04N 7/28 (2006.01)
- (21) **International Application Number:**
PCT/AU2012/000789
- (22) **International Filing Date:**
29 June 2012 (29.06.2012)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
2011203174 29 June 2011 (29.06.2011) AU
- (71) **Applicant (for all designated States except US):** **CANON KABUSHIKI KAISHA** [JP/JP]; 30-2, Shimomaruko 3-chome, Ohta-ku, Tokyo, 146 (JP).
- (72) **Inventors; and**
- (75) **Inventors/Applicants (for US only):** **ROSEWARNE, Christopher James** [AU/AU]; 5/78 Donnison Street, Gosford, NSW 2250 (AU). **WILSON, Rowena** [AU/AU]; 51/22 Taranto Road, Marsfield NSW 2122 (AU).
- (74) **Agent:** **SPRUSON & FERGUSON**; GPO BOX 3898, Sydney, New South Wales 2001 (AU).
- (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, 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, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.
- (84) **Designated States (unless otherwise indicated, for every kind of regional protection available):** ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ,

[Continued on next page]

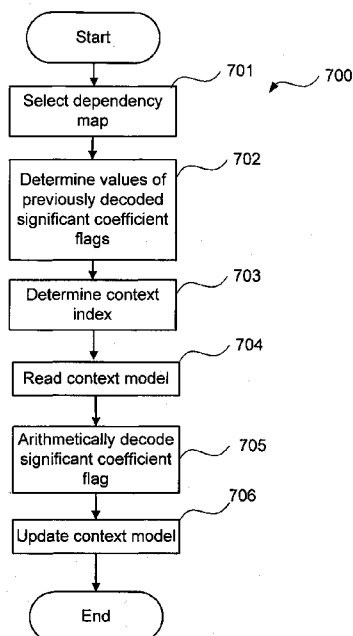
(54) **Title:** METHOD, APPARATUS AND SYSTEM FOR DETERMINING THE VALUE OF A COEFFICIENT FLAG

Fig. 7

(57) **Abstract:** A method of determining the value of a first significant coefficient flag (e.g., 802) in a video encoder for a first residual coefficient of a transform unit (e.g., 800), is disclosed. A dependency map for the transform unit (800) is selected from a set of predetermined dependency maps based on a scan pattern for the transform unit (800). The selected dependency map excludes a second significant coefficient flag (e.g., 805) located adjacent to the first significant coefficient flag (802). The determination of the value of the first coefficient is independent of determination of the value of the excluded second coefficient. A context index is determined for the first significant coefficient flag (802) by using each significant coefficient flag identified by the dependency map and compensating for the excluded second significant coefficient flag (805) using at least one significant coefficient flag of the dependency map. The value of the first significant coefficient flag of the transform unit (800) is determined using a context value selected according to the determined context index.



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))*

- 1 -

METHOD, APPARATUS AND SYSTEM FOR DETERMINING THE VALUE OF A COEFFICIENT FLAG

FIELD OF INVENTION

The present invention relates generally to digital video signal processing and, in particular, to a method, apparatus and system for determining the value of a significant
5 coefficient flag for a residual coefficient of a transform unit. The present invention also relates to a computer program product including a computer readable medium having recorded thereon a computer program for determining the value of a significant coefficient flag for a residual coefficient of a transform unit.

BACKGROUND

10 Many applications for video coding currently exist, including applications for transmission and storage of video data. Many video coding standards have also been developed and are currently in development. Recent developments in video coding standardisation have led to the formation of a group called the "Joint Collaborative Team on Video Coding" (JCT-VC). The Joint Collaborative Team on Video Coding (JCT-VC)
15 includes members of the Video Coding Experts Group (VCEG) of the Telecommunication Standardisation Sector (ITU-T) of the International Telecommunication Union (ITU) and members of the Moving Picture Experts Group (MPEG) of the International Organisations for Standardisation/International Electrotechnical Commission (ISO/IEC).

The JCT-VC has the goal of producing a new video coding standard to significantly
20 outperform a presently existing video coding standard, known as "H.264/MPEG-4 AVC". The H.264/MPEG-4 AVC standard is itself a large improvement on previous video coding standards, such as MPEG-4 and ITU-T H.263. The new video coding standard under development has been named "high efficiency video coding (HEVC)". The JCT-VC is also considering implementation challenges arising from technology proposed for high efficiency
25 video coding (HEVC) that create difficulties when scaling implementations of the standard to operate at high resolutions or high frame rates.

One area of the H.264/MPEG-4 AVC video coding standard that presents difficulties for implementation is efficiently entropy coding syntax elements used to represent video data. The H.264/MPEG-4 AVC standard introduced a coding scheme known as the context
30 adaptive binary arithmetic coding (CABAC) scheme, which provides significant coding

- 2 -

benefits over an earlier coding scheme known as the context adaptive variable length coding (CAVLC) scheme.

A reduction in bitstream size of 12-15% is achieved for many bitstreams using the context adaptive binary arithmetic coding (CABAC) scheme compared to the context adaptive variable length coding (CAVLC) scheme, with higher reductions possible for some bitstreams. Such a size reduction is achieved at a cost of increased computational complexity in performing the arithmetic coding and increased length of timing paths in digital logic used to implement the context adaptive binary arithmetic coding (CABAC) scheme.

A binary arithmetic coding algorithm (BAC) scheme, which is a variant upon the context adaptive binary arithmetic coding (CABAC) scheme is defined within the high efficiency video coding (HEVC) standard under development. In the high efficiency video coding (HEVC) standard under development, when binary arithmetic coding (BAC) is enabled, each syntax element is expressed as a sequence of bins, where the bins are selected from a set of available bins. Creating such a sequence of bins from a syntax element is known as “binarising” the syntax element.

The set of available bins is known as a “context model”. Each bin also has an associated value, indicating the likely bin value, known as a “most probable symbol” (‘valMPS’). As each bin is encoded or decoded, an associated probability level for arithmetically encoding or decoding the bin, known as a probability state, is updated. An instance of the context model exists at both an encoder and decoder and, provided no errors in transmission occur, the probability state and most probable symbol value (valMPS) for each bin is synchronised between the encoder and decoder. For each bin in a binarised syntax element, a symbol is coded in the bitstream by an arithmetic encoder, using the probability state associated with the bin. Symbol values of ‘0’ and ‘1’ indicate whether the bin value is equal to the most probable symbol value (valMPS) or not equal to the most probable symbol value (valMPS), respectively.

When implementing binary arithmetic coding (BAC) the binary arithmetic decoding algorithm has a feedback dependency loop. The feedback dependency loop is between a context selector, for determining which context from the context model to use for a current bin, a context modeller, for updating context information of each bin, and an arithmetic coder. The context selector uses values of previously decoded bins to determine the context for the current bin. In an encoder, the process of context selection is referred to as “binarisation”. In a decoder, the process of context selection is referred to as “inverse binarisation”. The

- 3 -

context modeller updates the most probable symbol value (valMPS) and the probability state for the current bin based on the received symbol value. The length of the feedback dependency loop limits bin throughput achievable in hardware implementations of binary arithmetic coding (BAC) for a given clock frequency. The limitation in bin throughput is relevant when an implementation is required to provide higher bin throughput. Such higher bin throughput is required to support video formats offering higher frame rates and higher resolutions, such as ultra high definition television (UHDTV) or super hi-vision (SHV) and wide quad high definition (WQHD).

SUMMARY OF THE INVENTION

It is an object of the present invention to substantially overcome, or at least ameliorate, one or more disadvantages of existing arrangements.

According to one aspect of the present disclosure there is provided a method of determining a value of a first significant coefficient flag in a video decoder for a first residual coefficient of a transform unit, the method comprising:

selecting a dependency map for use on a main region of the transform unit, the selected dependency map being selected from a set of predetermined dependency maps based on a scan pattern for the transform unit and excluding a second significant coefficient flag located adjacent to the first significant coefficient flag, wherein the determination of the value of the first significant coefficient flag is independent of determination of the value of the excluded second significant coefficient flag;

determining a context index for the first significant coefficient flag by using each significant coefficient flag identified by the dependency map; and

determining the value of the first significant coefficient flag of the transform unit, using a context value selected according to the determined context index.

According to another aspect of the present disclosure there is provided a system for determining a value of a first significant coefficient flag in a video decoder for a first residual coefficient of a transform unit, the system comprising:

a memory comprising data and a computer program;

a processor coupled to the memory for executing the program, the program comprising instructions for:

selecting a dependency map for use on a main region of the transform unit, the selected dependency map being selected from a set of predetermined dependency maps based on a scan pattern for the transform unit and excluding a second significant

- 4 -

coefficient flag located adjacent to the first significant coefficient flag, wherein the determination of the value of the first significant coefficient flag is independent of determination of the value of the excluded second significant coefficient flag;

determining a context index for the first significant coefficient flag by using
5 each significant coefficient flag identified by the dependency map; and
determining the value of the first significant coefficient flag of the transform unit,
using a context value selected according to the determined context index.

According to still another aspect of the present disclosure there is provided an apparatus for determining a value of a first significant coefficient flag in a video decoder for a
10 first residual coefficient of a transform unit, the apparatus comprising:

means for selecting a dependency map for use on a main region of the transform unit,
the selected dependency map being selected from a set of predetermined dependency maps
based on a scan pattern for the transform unit and excluding a second significant coefficient
flag located adjacent to the first significant coefficient flag, wherein the determination of the
15 value of the first significant coefficient flag is independent of determination of the value of
the excluded second significant coefficient flag;

means for determining a context index for the first significant coefficient flag by using
each significant coefficient flag identified by the dependency map; and

20 means for determining the value of the first significant coefficient flag of the
transform unit, using a context value selected according to the determined context index.

According to still another aspect of the present disclosure there is provided a computer
readable medium having a computer program stored thereon for determining a value of a first
significant coefficient flag in a video decoder for a first residual coefficient of a transform
unit, the program comprising:

25 code for selecting a dependency map for use on a main region of the transform unit,
the selected dependency map being selected from a set of predetermined dependency maps
based on a scan pattern for the transform unit and excluding a second significant coefficient
flag located adjacent to the first significant coefficient flag, wherein the determination of the
value of the first significant coefficient flag is independent of determination of the value of
30 the excluded second significant coefficient flag;

code for determining a context index for the first significant coefficient flag by using
each significant coefficient flag identified by the dependency map; and

- 5 -

code for determining the value of the first significant coefficient flag of the transform unit, using a context value selected according to the determined context index.

According to still another aspect of the present disclosure there is provided a method of encoding a value of a first significant coefficient flag in a video encoder, for a first residual coefficient of a transform unit, the method comprising:

selecting a dependency map for use on a main region of the transform unit, the selected dependency map being selected from a set of predetermined dependency maps based on scan pattern for the transform unit and excluding a second significant coefficient flag located adjacent to the first coefficient flag, wherein a determination of the value of the first significant coefficient flag is independent of a determination of a value of the excluded second significant coefficient flag;

determining a context index for the first significant coefficient flag by using each significant coefficient flag identified by the dependency map; and

determining the value of the first significant coefficient flag of the transform unit, using a context value selected according to the determined context index.

According to still another aspect of the present disclosure there is provided a system for encoding a value of a first significant coefficient flag in a video encoder, for a first residual coefficient of a transform unit, the system comprising:

a memory comprising data and a computer program;

a processor coupled to the memory for executing the program, the program comprising instructions for:

selecting a dependency map for use on a main region of the transform unit, the selected dependency map being selected from a set of predetermined dependency maps based on scan pattern for the transform unit and excluding a second significant coefficient flag located adjacent to the first coefficient flag, wherein a determination of the value of the first significant coefficient flag is independent of a determination of a value of the excluded second significant coefficient flag;

determining a context index for the first significant coefficient flag by using each significant coefficient flag identified by the dependency map; and

determining the value of the first significant coefficient flag of the transform unit, using a context value selected according to the determined context index.

- 6 -

According to still another aspect of the present disclosure there is provided an apparatus for encoding a value of a first significant coefficient flag in a video encoder, for a first residual coefficient of a transform unit, the apparatus comprising:

means for selecting a dependency map for use on a main region of the transform unit, the selected dependency map being selected from a set of predetermined dependency maps based on scan pattern for the transform unit and excluding a second significant coefficient flag located adjacent to the first coefficient flag, wherein a determination of the value of the first significant coefficient flag is independent of a determination of a value of the excluded second significant coefficient flag;

means for determining a context index for the first significant coefficient flag by using each significant coefficient flag identified by the dependency map; and

means for determining the value of the first significant coefficient flag of the transform unit, using a context value selected according to the determined context index.

According to still another aspect of the present disclosure there is provided a computer readable medium having a computer program stored thereon for encoding a value of a first significant coefficient flag in a video encoder, for a first residual coefficient of a transform unit, the program comprising:

code for selecting a dependency map for use on a main region of the transform unit, the selected dependency map being selected from a set of predetermined dependency maps based on scan pattern for the transform unit and excluding a second significant coefficient flag located adjacent to the first coefficient flag, wherein a determination of the value of the first significant coefficient flag is independent of a determination of a value of the excluded second significant coefficient flag;

code for determining a context index for the first significant coefficient flag by using each significant coefficient flag identified by the dependency map; and

code for determining the value of the first significant coefficient flag of the transform unit, using a context value selected according to the determined context index.

According to one aspect of the present disclosure there is provided a method of determining the value of a first significant coefficient flag in a video encoder for a first residual coefficient of a transform unit, the method comprising:

selecting a dependency map for the transform unit from a set of predetermined dependency maps based on a scan pattern for the transform unit, the selected dependency map excluding a second significant coefficient flag located adjacent to the first significant

- 7 -

coefficient flag, wherein the determination of the value of the first significant coefficient flag is independent of determination of the value of the excluded second significant coefficient flag;

determining a context index for the first significant coefficient flag by using each significant coefficient flag identified by the dependency map and compensating for the excluded second significant coefficient flag using at least one significant coefficient flag of the dependency map; and

determining the value of the first significant coefficient flag of the transform unit, using a context value selected according to the determined context index.

According to another aspect of the present disclosure there is provided a system for determining the value of a first significant coefficient flag in a video encoder for a first residual coefficient of a transform unit, the system comprising:

a memory comprising data and a computer program;

a processor coupled to the memory for executing the program, the program comprising instructions for:

selecting a dependency map for the transform unit from a set of predetermined dependency maps based on a scan pattern for the transform unit, the selected dependency map excluding a second significant coefficient flag located adjacent to the

first significant coefficient flag, wherein the determination of the value of the first coefficient is independent of determination of the value of the excluded second coefficient;

determining a context index for the first significant coefficient flag by using each significant coefficient flag identified by the dependency map and compensating for the excluded second significant coefficient flag using at least one significant coefficient flag of the dependency map; and

determining the value of the first significant coefficient flag of the transform unit, using a context value selected according to the determined context index.

According to still another aspect of the present disclosure there is provided an apparatus for determining the value of a first significant coefficient flag in a video encoder for a first residual coefficient of a transform unit, the apparatus comprising:

means for selecting a dependency map for the transform unit from a set of predetermined dependency maps based on a scan pattern for the transform unit, the selected

- 8 -

dependency map excluding a second significant coefficient flag located adjacent to the first significant coefficient flag, wherein the determination of the value of the first coefficient is independent of determination of the value of the excluded second coefficient;

means for determining a context index for the first significant coefficient flag by using each significant coefficient flag identified by the dependency map and compensating for the excluded second significant coefficient flag using at least one significant coefficient flag of the dependency map; and

means for determining the value of the first significant coefficient flag of the transform unit, using a context value selected according to the determined context index.

According to still another aspect of the present disclosure there is provided a computer readable medium having a computer program stored thereon for determining the value of a first significant coefficient flag in a video encoder for a first residual coefficient of a transform unit, the program comprising:

code for selecting a dependency map for the transform unit from a set of predetermined dependency maps based on a scan pattern for the transform unit, the selected dependency map excluding a second significant coefficient flag located adjacent to the first significant coefficient flag, wherein the determination of the value of the first coefficient is independent of determination of the value of the excluded second coefficient;

code for determining a context index for the first significant coefficient flag by using each significant coefficient flag identified by the dependency map and compensating for the excluded second significant coefficient flag using at least one significant coefficient flag of the dependency map; and

code for determining the value of the first significant coefficient flag of the transform unit, using a context value selected according to the determined context index.

Other aspects of the invention are also disclosed.

BRIEF DESCRIPTION OF THE DRAWINGS

One or more embodiments of the present invention will now be described with reference to the following drawings, in which:

Fig. 1 is a schematic block diagram showing functional modules of a video encoder;

Fig. 2 is a schematic block diagram showing functional modules of a video decoder;

Figs. 3A and 3B form a schematic block diagram of a general purpose computer system upon which the encoder and decoder of Figs. 1 and 2, respectively, may be practiced;

- 9 -

Fig. 4 is a schematic block diagram showing a video entropy encoder used in the video encoder of Fig. 1;

Fig. 5 is a flow diagram showing a method encoding significant coefficient flags as performed by the entropy encoder of Fig. 4;

5 Fig. 6 is a schematic block diagram showing a video entropy decoder used in the video decoder of Fig. 2;

Fig. 7 is a flow diagram showing a method of decoding a bitstream as performed by the entropy decoder of Fig. 6;

10 Fig. 8A shows a transform unit where a horizontal scan is performed on the transform unit;

Fig. 8B shows a transform unit where a vertical scan is performed on the transform unit;

Fig. 9A shows an 8x8 transform unit;

15 Fig. 9B shows a particular point in time during a zig-zag scan of the transform unit of Fig. 9A by the entropy decoder of Fig. 6;

Fig. 9C shows another point in time during the zig-zag scan of the transform unit of Fig. 9A by the entropy decoder of Fig. 6;

Fig. 9D shows a particular point in time during a diagonal scan from the lower-left to the upper-right of the transform unit of Fig. 9A by the entropy decoder of Fig. 6;

20 Fig. 10A shows a transform unit where a horizontal scan is performed on the transform unit;

Fig. 10B shows a transform unit where a vertical scan is performed on the transform unit;

25 Fig. 11A shows a transform unit where the dependency map for a significant coefficient flag near the left edge region of the transform unit is wrapped around to the right side of the transform unit and shifted up by one row;

Fig. 11B shows the transform unit where the dependency map for a significant coefficient flag near the top edge region of the transform unit is wrapped around to the bottom side of the transform unit and shifted left by one column;

30 Fig. 12A shows a particular point in time during a zig-zag scan of a transform unit by the entropy decoder 202 of Fig. 6 when decoding the bitstream;

Fig. 12B shows another particular point in time during the zig-zag scan of the transform unit of Fig. 12A by the entropy decoder 202 of Fig. 6 when decoding the bitstream;

- 10 -

Fig. 13A shows a transform unit;

Fig. 13B shows a particular point in time during a reverse zig-zag scan of the transform unit of Fig. 13A by the entropy decoder of Fig. 6; and

Fig. 14 is a flow diagram showing a method of determining the value of a significant
5 coefficient flag.

DETAILED DESCRIPTION

Where reference is made in any one or more of the accompanying drawings to steps and/or features, which have the same reference numerals, those steps and/or features have for the purposes of this description the same function(s) or operation(s), unless the contrary
10 intention appears.

In a video encoder or video decoder, as separate context information is available for each bin, context selection for bins provides a means to improve coding efficiency. In particular, coding efficiency may be improved by selecting a particular bin such that statistical properties from previous instances of the bin, where the associated context information was
15 used, correlate with statistical properties of a current instance of the bin. Such context selection frequently utilises spatially local information to determine the optimal context.

In binary arithmetic coding (BAC), each frame of video data is divided into hierarchical sets of coding units (CUs). A top level of each coding unit is referred to as a largest coding unit (LCU). The largest coding unit (LCU) has a maximum size, with edge
20 dimensions being a power of two and having equal width and height. Each frame is decomposed into an array of largest coding units (LCUs). Each largest coding unit (LCU) may be subdivided into four coding units (CUs), each having half the width and height of a parent largest coding unit (LCU). Each of the coding units (CUs) may be further subdivided into four coding units (CUs). Such a subdivision process may be applied recursively, enabling
25 coding units (CUs) to be defined down to a minimum supported size. The hierarchy of coding units (CUs) forms a coding unit (CU) tree.

At leaf nodes of the coding unit (CU) tree, residual coefficient data may be encoded into a bitstream. Each "residual coefficient" is a number representing image characteristics within a transform unit in the frequency (DCT) domain and occupying a unique location
30 within the transform unit. In this instance, a transform unit (TU) tree is created where the spatial region occupied by the coding unit (CU) is further subdivided into individual "transform units (TUs)". A transform unit is a block of residual data samples that may be transformed between the spatial and the frequency domains. In the frequency domain, the

- 11 -

transform unit (TU) encodes the residual data samples as residual coefficient data. Transform units are sized in powers of two (2), ranging from 4x4 data samples to 32x32 data samples for a "Luma" channel and 2x2 to 16x16 data samples for a "Chroma" channel. The leaf nodes of the transform unit (TU) tree may contain either a transform unit (TU) or nothing at all, in the case where no residual coefficient data is required.

Modified discrete cosine transforms (DCTs) or modified discrete sine transforms (DSTs) may be used to implement the residual transform. Implementations of the residual transform are configured to support each required transform unit (TU) size. In a video encoder, the residual coefficients from the residual transform are scaled and quantised. The scaling and quantisation reduces the magnitude of the coefficients, reducing the size of the data coded into the bitstream at the cost of reducing the image quality.

Fig. 1 is a schematic block diagram showing functional modules of a video encoder 100. Fig. 2 is a schematic block diagram showing functional modules of a video decoder 200. The video encoder 100 and video decoder 200 may be implemented using a general-purpose computer system 300, as shown in Figs. 3A and 3B.

As seen in Fig. 3A, the computer system 300 includes: a computer module 301; input devices such as a keyboard 302, a mouse pointer device 303, a scanner 326, a camera 327, and a microphone 380; and output devices including a printer 315, a display device 314 and loudspeakers 317. An external Modulator-Demodulator (Modem) transceiver device 316 may be used by the computer module 301 for communicating to and from a communications network 320 via a connection 321. The communications network 320 may be a wide-area network (WAN), such as the Internet, a cellular telecommunications network, or a private WAN. Where the connection 321 is a telephone line, the modem 316 may be a traditional "dial-up" modem. Alternatively, where the connection 321 is a high capacity (e.g., cable) connection, the modem 316 may be a broadband modem. A wireless modem may also be used for wireless connection to the communications network 320.

The computer module 301 typically includes at least one processor unit 305, and a memory unit 306. For example, the memory unit 306 may have semiconductor random access memory (RAM) and semiconductor read only memory (ROM). The computer module 301 also includes an number of input/output (I/O) interfaces including: an audio-video interface 307 that couples to the video display 314, loudspeakers 317 and microphone 380; an I/O interface 313 that couples to the keyboard 302, mouse 303, scanner 326, camera 327 and optionally a joystick or other human interface device (not

- 12 -

illustrated); and an interface 308 for the external modem 316 and printer 315. In some implementations, the modem 316 may be incorporated within the computer module 301, for example within the interface 308. The computer module 301 also has a local network interface 311, which permits coupling of the computer system 300 via a connection 323 to a local-area communications network 322, known as a Local Area Network (LAN). As illustrated in Fig. 3A, the local communications network 322 may also couple to the wide network 320 via a connection 324, which would typically include a so-called "firewall" device or device of similar functionality. The local network interface 311 may comprise an EthernetTM circuit card, a BluetoothTM wireless arrangement or an IEEE 802.11 wireless arrangement; however, numerous other types of interfaces may be practiced for the interface 311.

The I/O interfaces 308 and 313 may afford either or both of serial and parallel connectivity, the former typically being implemented according to the Universal Serial Bus (USB) standards and having corresponding USB connectors (not illustrated). Storage devices 309 are provided and typically include a hard disk drive (HDD) 310. Other storage devices such as a floppy disk drive and a magnetic tape drive (not illustrated) may also be used. An optical disk drive 312 is typically provided to act as a non-volatile source of data. Portable memory devices, such optical disks (e.g., CD-ROM, DVD, Blu-ray DiscTM), USB-RAM, portable, external hard drives, and floppy disks, for example, may be used as appropriate sources of data to the system 300.

The components 305 to 313 of the computer module 301 typically communicate via an interconnected bus 304 and in a manner that results in a conventional mode of operation of the computer system 300 known to those in the relevant art. For example, the processor 305 is coupled to the system bus 304 using a connection 318. Likewise, the memory 306 and optical disk drive 312 are coupled to the system bus 304 by connections 319. Examples of computers on which the described arrangements can be practised include IBM-PC's and compatibles, Sun Sparcstations, Apple MacTM or alike computer systems.

The encoder 100 and the decoder 200, as well as methods described below, may be implemented using the computer system 300 wherein the encoder 100, the decoder 200 and the processes of Figs. 5, 7 and 14, to be described, may be implemented as one or more software application programs 333 executable within the computer system 300. In particular, the encoder 100, the decoder 200 and the steps of the described methods are effected by instructions 331 (see Fig. 3B) in the software 333 that are carried out within the computer

- 13 -

system 300. The software instructions 331 may be formed as one or more code modules, each for performing one or more particular tasks. The software may also be divided into two separate parts, in which a first part and the corresponding code modules performs the described methods and a second part and the corresponding code modules manage a user interface between the first part and the user.

The software may be stored in a computer readable medium, including the storage devices described below, for example. The software is loaded into the computer system 300 from the computer readable medium, and then executed by the computer system 300. A computer readable medium having such software or computer program recorded on the computer readable medium is a computer program product. The use of the computer program product in the computer system 300 preferably effects an advantageous apparatus for implementing the encoder 100, the decoder 200 and the described methods.

The software 333 is typically stored in the HDD 310 or the memory 306. The software is loaded into the computer system 300 from a computer readable medium, and executed by the computer system 300. Thus, for example, the software 333 may be stored on an optically readable disk storage medium (e.g., CD-ROM) 325 that is read by the optical disk drive 312.

In some instances, the application programs 333 may be supplied to the user encoded on one or more CD-ROMs 325 and read via the corresponding drive 312, or alternatively may be read by the user from the networks 320 or 322. Still further, the software can also be loaded into the computer system 300 from other computer readable media. Computer readable storage media refers to any non-transitory tangible storage medium that provides recorded instructions and/or data to the computer system 300 for execution and/or processing. Examples of such storage media include floppy disks, magnetic tape, CD-ROM, DVD, Blu-ray Disc, a hard disk drive, a ROM or integrated circuit, USB memory, a magneto-optical disk, or a computer readable card such as a PCMCIA card and the like, whether or not such devices are internal or external of the computer module 301. Examples of transitory or non-tangible computer readable transmission media that may also participate in the provision of software, application programs, instructions and/or data to the computer module 301 include radio or infra-red transmission channels as well as a network connection to another computer or networked device, and the Internet or Intranets including e-mail transmissions and information recorded on Websites and the like.

- 14 -

The second part of the application programs 333 and the corresponding code modules mentioned above may be executed to implement one or more graphical user interfaces (GUIs) to be rendered or otherwise represented upon the display 314. Through manipulation of typically the keyboard 302 and the mouse 303, a user of the computer system 300 and the application may manipulate the interface in a functionally adaptable manner to provide controlling commands and/or input to the applications associated with the GUI(s). Other forms of functionally adaptable user interfaces may also be implemented, such as an audio interface utilizing speech prompts output via the loudspeakers 317 and user voice commands input via the microphone 380.

Fig. 3B is a detailed schematic block diagram of the processor 305 and a "memory" 334. The memory 334 represents a logical aggregation of all the memory modules (including the HDD 309 and semiconductor memory 306) that can be accessed by the computer module 301 in Fig. 3A.

When the computer module 301 is initially powered up, a power-on self-test (POST) program 350 executes. The POST program 350 is typically stored in a ROM 349 of the semiconductor memory 306 of Fig. 3A. A hardware device such as the ROM 349 storing software is sometimes referred to as firmware. The POST program 350 examines hardware within the computer module 301 to ensure proper functioning and typically checks the processor 305, the memory 334 (309, 306), and a basic input-output systems software (BIOS) module 351, also typically stored in the ROM 349, for correct operation. Once the POST program 350 has run successfully, the BIOS 351 activates the hard disk drive 310 of Fig. 3A. Activation of the hard disk drive 310 causes a bootstrap loader program 352 that is resident on the hard disk drive 310 to execute via the processor 305. This loads an operating system 353 into the RAM memory 306, upon which the operating system 353 commences operation. The operating system 353 is a system level application, executable by the processor 305, to fulfil various high level functions, including processor management, memory management, device management, storage management, software application interface, and generic user interface.

The operating system 353 manages the memory 334 (309, 306) to ensure that each process or application running on the computer module 301 has sufficient memory in which to execute without colliding with memory allocated to another process. Furthermore, the different types of memory available in the system 300 of Fig. 3A must be used properly so that each process can run effectively. Accordingly, the aggregated memory 334 is not

- 15 -

intended to illustrate how particular segments of memory are allocated (unless otherwise stated), but rather to provide a general view of the memory accessible by the computer system 300 and how such is used.

As shown in Fig. 3B, the processor 305 includes a number of functional modules including a control unit 339, an arithmetic logic unit (ALU) 340, and a local or internal memory 348, sometimes called a cache memory. The cache memory 348 typically includes a number of storage registers 344- 346 in a register section. One or more internal busses 341 functionally interconnect these functional modules. The processor 305 typically also has one or more interfaces 342 for communicating with external devices via the system bus 304, using a connection 318. The memory 334 is coupled to the bus 304 using a connection 319.

The application program 333 includes a sequence of instructions 331 that may include conditional branch and loop instructions. The program 333 may also include data 332 which is used in execution of the program 333. The instructions 331 and the data 332 are stored in memory locations 328, 329, 330 and 335, 336, 337, respectively. Depending upon the relative size of the instructions 331 and the memory locations 328-330, a particular instruction may be stored in a single memory location as depicted by the instruction shown in the memory location 330. Alternately, an instruction may be segmented into a number of parts each of which is stored in a separate memory location, as depicted by the instruction segments shown in the memory locations 328 and 329.

In general, the processor 305 is given a set of instructions which are executed therein. The processor 305 waits for a subsequent input, to which the processor 305 reacts to by executing another set of instructions. Each input may be provided from one or more of a number of sources, including data generated by one or more of the input devices 302, 303, data received from an external source across one of the networks 320, 302, data retrieved from one of the storage devices 306, 309 or data retrieved from a storage medium 325 inserted into the corresponding reader 312, all depicted in Fig. 3A. The execution of a set of the instructions may in some cases result in output of data. Execution may also involve storing data or variables to the memory 334.

The encoder 100, the decoder 200 and the described methods use input variables 354, which are stored in the memory 334 in corresponding memory locations 355, 356, 357. The encoder 100, the decoder 200 and the described methods produce output variables 361, which are stored in the memory 334 in corresponding memory locations 362, 363, 364. Intermediate variables 358 may be stored in memory locations 359, 360, 366 and 367.

- 16 -

Referring to the processor 305 of Fig. 3B, the registers 344, 345, 346, the arithmetic logic unit (ALU) 340, and the control unit 339 work together to perform sequences of micro-operations needed to perform "fetch, decode, and execute" cycles for every instruction in the instruction set making up the program 333. Each fetch, decode, and execute cycle comprises:

5 (a) a fetch operation, which fetches or reads an instruction 331 from a memory location 328, 329, 330;

(b) a decode operation in which the control unit 339 determines which instruction has been fetched; and

10 (c) an execute operation in which the control unit 339 and/or the ALU 340 execute the instruction.

Thereafter, a further fetch, decode, and execute cycle for the next instruction may be executed. Similarly, a store cycle may be performed by which the control unit 339 stores or writes a value to a memory location 332.

Each step or sub-process in the processes of Figs. 5, 7 and 14 is associated with one or
15 more segments of the program 333 and is performed by the register section 344, 345, 347, the ALU 340, and the control unit 339 in the processor 305 working together to perform the fetch, decode, and execute cycles for every instruction in the instruction set for the noted segments of the program 333.

The encoder 100, the decoder 200 and the described methods may alternatively be
20 implemented in dedicated hardware such as one or more integrated circuits performing the functions or sub functions of the described methods. Such dedicated hardware may include graphic processors, digital signal processors, or one or more microprocessors and associated memories.

As described above, the video encoder 100 may be implemented as one or more
25 software code modules of the software application program 333 resident on the hard disk drive 305 and being controlled in its execution by the processor 305. In particular the video encoder 100 comprises modules 102 to 112 and 114 which may each be implemented as one or more software code modules of the software application program 333.

Although the video encoder 100 is an example of a high efficiency video coding
30 (HEVC) video decoding pipeline, processing stages performed by the modules 102 to 112 and 114 are common to other video codecs such as VC-1 or H.264/MPEG-4 AVC. The video encoder 100 receives unencoded frame data 101 as a series of frames consisting of luminance and chrominance samples. The video encoder 100 divides each frame of the frame data 101

- 17 -

into one or more slices which contain an integer number of coding units. Each coding unit is encoded sequentially by further dividing the coding unit into two-dimensional arrays of data samples known as blocks. The video encoder 100 operates by outputting from a multiplexer module 110 a prediction for each array of data samples. A difference module 115 outputs the difference between a prediction and a corresponding array of data samples received from the frame data 101, known as residual data samples. The prediction from the multiplexer module 110 will be described in more detail below.

The residual data samples from the difference module 115 are received by a transform module 102, which converts the difference from a spatial representation to a frequency domain representation to create transform coefficients. For the H.264/MPEG-4 AVC video coding standard, the conversion to the frequency domain representation is implemented using a modified discrete cosine transform (DCT), modified to be implemented using shifts and additions. The transform coefficients are then input to a scale and quantise module 103 where the coefficients are scaled and quantised to produce residual coefficients. The scale and quantisation process results in a loss of precision. The residual coefficients are taken as input to an inverse scaling module 105 which reverses the scaling performed by the scale and quantise module 103 to produce rescaled versions of the transform coefficients. The residual coefficients are also taken as input to an entropy encoder module 104 which encodes the residual coefficients in an output bitstream 113. Due to the loss of precision resulting from the scale and quantise module 103, the rescaled transform coefficients are not identical to the original transform coefficients. The rescaled transform coefficients from the inverse scaling module 105 are then output to an inverse transform module 106. The inverse transform module 106 performs an inverse transform from the frequency domain to the spatial domain to produce a spatial-domain representation of the rescaled transform coefficients identical to a spatial domain representation that is produced at a decoder.

A motion estimation module 107 produces motion vectors by comparing the frame data 101 with previous frame data stored in a frame buffer module 112 configured within the memory 306. The motion vectors are then input to a motion compensation module 108 which produces inter-predicted reference samples by filtering samples stored in the frame buffer module 112, taking into account a spatial offset derived from the motion vectors. The motion vectors are also passed as syntax elements to an entropy encoder module 104 for coding in the output bitstream 113. An intra-frame prediction module 109 produces intra-predicted

- 18 -

reference samples using samples obtained from a summation module 114 of the output of the multiplexor module 110 and the output from the inverse transform module 106.

Coding units may be coded using intra-prediction or inter-prediction methods. The decision as to whether to use intra-prediction or inter-prediction is made according to a rate-distortion trade-off between desired bit-rate of the resulting bitstream and the amount of image quality distortion introduced by either the intra-prediction or inter-prediction method. The multiplexor module 110 selects either the intra-predicted reference samples from the intra-frame prediction module 109 or the inter-predicted reference samples from the motion compensation block 108, depending on the current prediction mode. The summation module 114 produces a sum that is input to a deblocking filter module 111. The deblocking filter module 111 performs filtering along block boundaries, producing deblocked samples that are written to the frame buffer module 112 configured within the memory 306. The frame buffer module 112 is a buffer with sufficient capacity to hold data from multiple past frames for future reference.

In the video encoder 100, the residual data samples within one transform unit (TU) are determined by finding the difference between data samples of the input frame data 101 and a prediction of the data samples of the input frame data 101. The difference provides a spatial representation of the transform unit (TU) coefficients. The prediction is provided by an array of data samples known as a prediction unit (PU). Each prediction unit (PU) contains a prediction of the input frame data derived by applying an intra-prediction or an inter-prediction process. Several methods may be used for coding prediction units (PUs) within a coding unit (CU). A single prediction unit (PU) may occupy an entire area of the coding unit (CU), or the coding unit (CU) may be split into two equal-sized rectangular prediction units (PUs), either horizontally or vertically. Additionally, the coding units (CU) may be split into four equal-sized square prediction units (PUs).

As the spatial representation of the transform unit is a two-dimensional array of residual data samples, as described in detail below, a frequency domain representation resulting from the transform is also a two-dimensional array of residual coefficients. The spectral characteristics of a typical transform unit (TU) are such that the frequency domain representation is more compact than the spatial representation. Further, the predominance of lower-frequency spectral information typical in a transform unit (TU) results in a clustering of larger-valued residual coefficients towards the upper-left of the transform unit (TU), where low-frequency residual coefficients are represented.

- 19 -

The two-dimensional frequency domain representation of a transform unit (TU) is converted to a one-dimensional list by scanning the two-dimensional list in a particular order, known as a scan order. In HEVC reference model version 3.0, the scan operation starts at an upper-left sample of the transform unit (TU) and progresses until a last significant coefficient is reached (i.e. the last coefficient having a nonzero value). Scan operations having this property are known as 'forward scans'. Residual coefficients in the one-dimensional list having a nonzero magnitude are known as "significant coefficients". In the HEVC reference software version 3.0, the location of the last significant coefficient is signalled by encoding co-ordinates of the coefficient in the transform unit (TU). A list of significant coefficient flags indicating the significance of each residual coefficient prior to the last significant coefficient is coded into the bitstream.

The clustering of larger-valued residual coefficients towards the upper-left of the transform unit (TU) results in most residual coefficients earlier in the residual coefficient list being significant, whereas few significant residual coefficients are found later in the residual coefficient list.

As described in detail below, an alternative scan ordering scheme known as 'backward scan' commences scanning at the position of a last significant coefficient flag and progresses back to the upper-left location within the transform unit (TU). Backward scans have the property that residual coefficients at lower list indices tend to have lower magnitudes and fewer significant residual coefficients than residual coefficients at higher list indices.

As described above, the video encoder 100 also comprises an entropy encoder module 104 implementing an entropy encoding method. The entropy encoder module 104 produces syntax elements from incoming residual coefficient data (or residual coefficients) received from the scale and quantise module 103. The entropy encoder module 104 outputs bitstream data 113 and will be described in more detail below. In accordance with the H.264/MPEG-4 AVC video coding standard, the bitstream data 113 is delineated into network abstraction layer (NAL) units. Each slice is contained in one NAL unit.

There are several alternatives for the entropy encoding method implemented in the entropy encoder module 104. The alternatives available depend on which video coding method the encoder 100 is performing. For example, the high efficiency video coding (HEVC) standard currently under development supports use of low complexity entropy coding (LCEC), a variant of context adaptive variable length coding (CAVLC) in 'low complexity' configurations. The high efficiency video coding (HEVC) standard currently

- 20 -

under development also supports binary arithmetic coding (BAC), a variant of context adaptive binary arithmetic coding (CABAC) in 'high efficiency' configurations.

For a given video coding method being performed by the encoder 100 one of the supported entropy coding methods is selected according to the configuration of the encoder 100. Further, in encoding the coding units from each frame, the entropy encoder module 104 writes the bitstream data 113 such that each frame has one or more slices per frame, with each slice containing image data for part of the frame. Producing one slice per frame reduces overhead associated with delineating each slice boundary. However, dividing the frame into multiple slices is also possible.

The video decoder 200 may be implemented as one or more software code modules of the software application program 333 resident on the hard disk drive 305 and being controlled in its execution by the processor 305. In particular the video decoder 200 comprises modules 202 to 208 and 210 which may each be implemented as one or more software code modules of the software application program 333. Although the video decoder 200 is described with reference to a high efficiency video coding (HEVC) video decoding pipeline, processing stages performed by the modules 202 to 208 and 209 are common to other video codecs that employ entropy coding, such as H.264/MPEG-4 AVC, MPEG-2 and VC-1.

An encoded bitstream, such as the bitstream 113, is received by the video decoder 200. The encoded bitstream 113 may be read from memory 306, the hard disk drive 310, a CD-ROM, a Blu-ray disk or other computer readable storage medium. Alternatively the encoded bitstream 113 may be received from an external source such as a server connected to the communications network 320 or a radio-frequency receiver. The encoded bitstream 113 comprises encoded syntax elements representing frame data to be decoded.

The encoded bitstream 113 is input to an entropy decoder module 202 which extracts the syntax elements from the encoded bitstream 113 and passes the values of the syntax elements to other blocks in the video decoder 200. There may be multiple entropy decoding methods implemented in the entropy decoder module 202. For example, the H.264/MPEG-4 AVC standard supports context adaptive binary arithmetic coding (CABAC) and context adaptive variable length coding (CAVLC) entropy decoding methods. Further, high efficiency video coding (HEVC) supports the low complexity entropy coding (LCEC) and binary arithmetic coding (BAC) entropy decoding methods. Syntax element data representing residual coefficient data is passed to an inverse scale and transform module 203 and syntax element data representing motion vector information is passed to a motion compensation

- 21 -

module 204. The inverse scale and transform module 203 performs inverse scaling on the residual coefficient data to create reconstructed transform coefficients. The module 203 then performs an inverse transform to convert the reconstructed transform coefficients from a frequency domain representation to a spatial domain representation, producing residual samples. For the H.264/MPEG-4 AVC standard, the inverse transform is a modified inverse discrete cosine transform (IDCT), which uses only shifts and adds for reduced complexity.

The motion compensation module 204 uses the motion vector data from entropy decoder module 202, combined with previous frame data from a frame buffer block 208, configured within the memory 306, to produce inter-predicted reference samples for a prediction unit (PU). When a syntax element indicates that the current coding unit was coded using intra-prediction, an intra-frame prediction module 205 produces intra-predicted reference samples for the prediction unit (PU) using samples spatially neighbouring the prediction unit (PU) that are obtained from the sum from a summation module 210. A multiplexor module 206 selects intra-predicted reference samples or inter-predicted reference samples for the prediction unit (PU) depending on the current prediction mode, which is indicated by a syntax element in the bitstream data 113. The array of samples output from the multiplexor module 206 is added to the residual samples from the inverse scale and transform module 203 by the summation module 210 to produce a sum which is then input to a deblocking filter module 207 and the intra-frame prediction module 205. The deblocking filter module 207 performs filtering along data block boundaries to smooth artefacts visible along the data block boundaries. The output of the deblocking filter module 207 is written to the frame buffer module 208 configured within the memory 306. The frame buffer module 208 provides sufficient storage to hold multiple decoded frames for future reference. Decoded frames 209 are also output from the frame buffer module 208.

In order to decode each significant coefficient flag, a suitable context must be selected within the entropy decoder 202, as will be described below. In HEVC, the method for selecting the context is dependent on the scan order for the transform unit, the location of the significant coefficient flag within the transform unit (TU) and the size of the transform unit (TU). For transform units (TUs) sized 16x16 coefficients and above, values of neighbouring significant coefficient flags within the transform unit are also used for context selection. The context selection scheme for significant coefficient flags divides the transform unit (TU) into four regions, each having a dedicated set of contexts. The four regions will be described with reference to Fig. 9A.

- 22 -

As seen in Fig. 9A, a transform unit (TU) 900 is divided into four regions 901, 902, 903 and 904, each having a dedicated set of contexts for encoding or decoding significant coefficient flag values.

Corner region 901, occupying the upper left 2x2 array of the transform unit (TU) 900
5 has one context dedicated to each location in the 2x2 array.

Top-edge region 902, indicated by upward cross-hatch and occupying the remaining locations in the uppermost row of the transform unit (TU), has a further dedicated set of contexts.

Left-edge region 903, indicated by downward cross-hatch and occupying the
10 remaining locations in the leftmost column, has another dedicated set of contexts.

Main region 904, occupying the remainder of the transform unit (TU) has yet another dedicated set of contexts.

For each of the dedicated sets of contexts, it is necessary to determine which particular context should be used to encode or decode a particular significant coefficient flag. The
15 determination occurs by referencing the neighbouring significant coefficient flags to the particular significant coefficient flag that have already been decoded from the encoded bitstream 113. The set of referenced neighbouring significant coefficient flags has a pattern relative to the particular significant coefficient flag, referred to below as a 'dependency map'. An index, known as the context index, into each dedicated set of contexts may be derived by
20 summing the values of the significant coefficient flags falling within the dependency map. Summing the values of the significant coefficient flags falling within the dependency map indicates whether the set of dedicated contexts has a size equal to the number of neighbouring significant coefficient flags in the dependency map. In one implementation, the range of values of the significant coefficient flags may be restricted to a smaller value, reducing the
25 number of contexts in the set of dedicated contexts. Other implementations having large dependency maps may reduce the number of contexts in the set of dedicated contexts by using an integer division or a logical shift of the sum of the neighbouring significant coefficient values to produce the context index. Other methods of determining the context index based upon values of the significant coefficient flags within the dependency map are also possible.
30 For example, implementations may also restrict the range of the output after the summation to restrict the range of the context index, reducing the number of contexts in the context model.

An optimisation to improve parallelism of decoding the list of significant coefficient flags at specific locations within a transform unit (TU), has been proposed for the high

- 23 -

efficiency video coding (HEVC) standard under development, via proposal JCTVC-E330. In accordance with the proposal, along a top-most row and left-most column of the transform unit (TU), there are locations where a zig-zag scan changes direction. At the locations where the zig-zag scan changes direction, a significant coefficient flag decoded immediately prior in the zig-zag scan order will be the neighbouring significant coefficient flag above or to the left of a significant coefficient flag, defined as a current significant coefficient flag, with reference to the method 700 of Fig. 7. Significant coefficient flags in either of the locations form part of the dependency map for determining the context index for the current significant coefficient flag.

In contrast, for coefficients in the main region 904 of the transform unit (TU) 900, a decoded significant coefficient flag immediately prior in the zig-zag scan order is located either one location below-left or above-right of the current significant coefficient flag location. Significant coefficient flags located in either of the below-left or above-right positions are not part of the dependency map for determining the context index for the current significant coefficient flag. The JCTVC-E330 proposal for the high efficiency video coding (HEVC) standard under development provides an optimisation that improves the parallelism of the zig-zag scan along the top-edge region 902 and left-edge region 903. The optimisation enables pipelined decoding of significant coefficient flags and therefore enables a higher throughput of significant coefficient flags to be achieved. In accordance with the proposal, the significant coefficient flag that was decoded immediately to the left of the current significant coefficient flag, when decoding significant coefficient flags from the top-edge region 902, or immediately above the current significant coefficient flag when decoding significant coefficient flags from the left-edge region 903, is omitted from the dependency map used to determine the context index for the current significant coefficient flag.

As described above, determining the correct context to access within the entropy decoder 202 requires use of previously decoded values of significant coefficient flags within the dependency map. The previously decoded values may be used to determine a context index for determining the correct context. The context consists of a probability state and valMPS. The probability state and valMPS values may then be provided for arithmetic decoding, where the probability state and valMPS values are used to perform the arithmetic decoding. As each significant coefficient value has been decoded, the decoded value is available for use in determining the required context index for subsequent significant

- 24 -

coefficient values. As each of the significant coefficient flags is decoded, sufficient data becomes available to determine the required context for a future significant coefficient flag.

As described below, a dependency map may be selected, such that the value of the significant coefficient flag that is temporally adjacent and preceding the current significant coefficient flag is not required to determine the context index for the current significant coefficient flag. The selection of the dependency map enables a software implementation or an interleaved hardware implementation of significant coefficient flag decoding, such that the utilisation of the arithmetic decoding pipeline stages are maximised by avoiding dead-time while waiting for a context index to be determined.

The throughput of decoding the significant coefficient flags within a transform unit is limited by the ability to decode the value of the significant coefficient flags upon which each successive significant coefficient flag depends. In order to decode the significant coefficient flags in the transform unit it is necessary to select the correct context for each significant coefficient flag prior to decoding the bin associated with the significant coefficient flag. Probabilistic information held in each context represents statistical knowledge of the video sequence that the binary arithmetic coding (BAC) algorithm acquires during operation. The selection of one context from a set of available contexts for each significant coefficient flag benefits from a dependency on the values of neighbouring significant coefficient flag values, due to a statistical correlation between neighbouring significant coefficient flags in the transform unit.

Fig. 4 shows the entropy encoder 104 in accordance with one implementation. The entropy encoder 104 may be implemented as one or more software code modules of the software application program 333 resident on the hard disk drive 305 and being controlled in its execution by the processor 305. In particular, as seen in Fig. 4, the entropy encoder 104 comprises modules 403, 404 and 406 which may each be implemented as one or more software code modules of the software application program 333. Alternatively, the modules 403, 404 and 406 of the entropy encoder 104 may be implemented in dedicated hardware.

Fig. 5 is a flow diagram showing a method 500 of encoding syntax elements into a bitstream 113, as performed by the modules 403, 404 and 406 of the entropy encoder 104 of Fig. 4.

The method 500 is performed for each significant coefficient flag associated with each residual coefficient within a transform unit. The method 500 will be described in relation to encoding the value for one of the significant coefficient flags in the transform unit.

- 25 -

The method 500 begins at step 501, where syntax elements 402 are received by the entropy encoder 104 and input to a binariser module 403. The binariser module 403 binarises each syntax element into a sequence of bins with the values of each bin being determined based on the value of each syntax element. The syntax elements 402 received at step 501 include residual coefficients and a list of significant coefficient flags indicating the significance of each of the residual coefficients. In particular, each residual coefficient has an associated significant coefficient flag indicating whether the particular coefficient has a nonzero value.

At step 501, the binariser module 403 selects a dependency map from a set of predetermined dependency maps 405 for use in encoding the syntax elements 402. The dependency map may be selected, for example, from the memory 306 or hard disk drive 310. Alternatively, the dependency map may be configured in hardware logic. The dependency map is selected from the predetermined dependency maps 405 depending on a scan pattern 401 to be performed by the binariser module 403 on each transform unit formed from the residual coefficients or the location of the current significant coefficient flag, or a combination of the two. As an example, Fig. 8A shows a transform unit 800 comprising an 8x8 block of residual coefficients. A horizontal scan pattern (as represented by the arrow in Fig. 8A) is performed by the entropy decoder 202 on the transform unit 800. Each residual coefficient of the transform unit 800 has an associated significant coefficient flag. Accordingly, each block of the transform unit 800 will be referred to below with reference to the associated "significant coefficient flag" rather than the residual coefficient that each block represents.

The specific significant coefficient flags of the transform unit 800 are differently highlighted in Fig. 8A to indicate different roles in the determination of a context for significant coefficient flag 802. In particular, when the scan pattern of the transform unit 800 is the horizontal scan pattern, as seen in Fig. 8A, values of the significant coefficient flags 801, 809, 820 and 821 indicated by dots in Fig. 8A, may be used to determine the context for the significant coefficient flag 802. The significant coefficient flags 801, 809, 820 and 821 indicated by dots in Fig. 8A represent the dependency map for determining the context for the significant coefficient flag 802 of the transform unit 800. Although the method 500 will be described with reference to the horizontal scan pattern and the dependency map of Fig. 8A, any of the scan patterns described below and the associated dependency maps may be selected at step 501.

- 26 -

At the next step 502, the binariser module 403 accesses previously determined significant coefficient flag values 407 for use in determining a context index for the significant coefficient flag 802. The values of the predetermined significant coefficient flags 407 are determined from the residual coefficients provided by the scale and quantise module 103. As described above, the context index is used to read the probability state and most probable symbol value (valMPS) for the significant coefficient flag from a context model module 404. The previously determined significant coefficient flag values may be accessed from the memory 306 or the hard disk drive 310.

At step 503, the binariser module 403 determines the context index for the significant coefficient flag based on the previously determined significant coefficient flag values. In one implementation, the context index may be determined by summing together the values of the significant coefficient flags contained within the syntax elements 402 and located within the dependency map. Implementations may also restrict the range of the output of the summation to restrict the range of the context index, reducing the number of contexts in the context model module 404.

At the next step 504, the binariser module 403 uses the context index determined for the significant coefficient flag to read the context model module 404 and determine the context for the significant coefficient flag, where the context for the significant coefficient flag includes a most probable symbol (valMPS) value and a probability level indicating the probability of the significant coefficient flag being equal to the least probable symbol (LPS). In particular, the binariser module 403 uses the list of significant coefficient flags within the syntax elements 402 to determine the context index for the significant coefficient flag. Where the location of a particular significant coefficient flag to be determined is such that the locations of any of the significant coefficient flags in the dependency map fall outside a transform unit (TU) boundary area, the locations of the significant coefficient flags used to determine the particular significant coefficient flag falling outside the transform unit (TU) boundary area are not used to determine the context index for the bin at step 503. Alternative implementations may include the significant coefficient flags in the dependency map falling outside the transform unit (TU) boundary in determining the context index by predicting values for the significant coefficient flags using the values of significant coefficient flags located within the transform unit (TU). The determination of the context index by predicting values for the significant coefficient flags will be described in more detail with reference to Figs. 9B, 9C and 9D below.

- 27 -

At step 505, the binary arithmetic encoder module 406 arithmetically encodes the significant coefficient flag into the encoded bitstream 113 using the context information provided by the context model 404.

At step 506, the context model module 404 updates the context information stored within the context model module 603 at the context index associated with the significant coefficient flag based on the determined value for the significant coefficient flag. The context update includes an adjustment of the probability level associated with the context index and may also include an adjustment of the most probable symbol (valMPS) associated with the context index.

Fig. 6 is a schematic block diagram showing the entropy decoder 202 of Fig. 2 in accordance with one implementation. The entropy decoder 202 may be implemented as one or more software code modules of the software application program 333 resident on the hard disk drive 305 and being controlled in its execution by the processor 305. In particular the entropy decoder 202 comprises modules 602, 603 and 605 which may each be implemented as one or more software code modules of the software application program 333. Alternatively, the modules 602, 603 and 605 may be implemented in dedicated hardware.

Fig. 7 is a flow diagram showing a method 700 of decoding a bitstream, as performed by the modules 602, 603 and 605 of the entropy decoder 202 of Fig. 6. The entropy decoder 202 decodes an encoded bitstream, such as the encoded bitstream 113, produced by the entropy encoder 104.

The method 700 is performed for each significant coefficient flag associated with each residual coefficient within the transform unit. The method 700 will be described in relation to determining the value for one of the significant coefficient flags, hereby referred to as the current significant coefficient flag.

The method 700 begins at step 701, where the encoded bitstream 113 is received by the entropy decoder 202 and the inverse binariser module 605 selects a dependency map for use in decoding the bitstream 113 from a set of predetermined dependency maps 604. The dependency map may be selected from the predetermined dependency maps 604 based on the scan pattern, the location of the current significant coefficient flag, or a combination of the two. Again, the dependency map may be selected, for example, from the memory 306 or hard disk drive 310. Alternatively, the dependency map may be configured in hardware logic.

The dependency map is selected at step 701 depending on a scan pattern 606 to be performed by the decoder 202 on a transform unit formed from residual coefficients within

- 28 -

the bitstream 113. As described above, in the example of Fig. 8A, the scan pattern used to encode the bitstream 113 was the horizontal scan pattern and the selected dependency map is the dependency map shown by significant coefficient flags 801, 809, 820 and 821 in Fig. 8A.

At the next step 702, the inverse binariser module 605 accesses values of previously
5 decoded significant coefficient flags 607 for use in determining a context index for the current significant coefficient flag (e.g., current significant coefficient flag 802) associated with a residual coefficient in the bitstream 113. The context index is used to read a probability state and most probable symbol value (valMPS) from a context model module 603 for the current significant coefficient flag. The values of the previously decoded significant coefficient flags
10 may be accessed from the memory 306 or the hard disk drive 310. Then at step 703, the inverse binariser module 605 determines the context index for the current significant coefficient flag based on the previously decoded significant coefficient flags.

At the next step 704, the inverse binariser module 605 uses the context index
15 determined for the current significant coefficient flag to read the context model module 603 and determine a context for the current significant coefficient flag, where the context for the current significant coefficient flag includes a most probable symbol (valMPS) value and a probability level indicating the probability of the corresponding significant coefficient flag being equal to the least probable symbol (LPS).

Then at step 705, the context for the current significant coefficient flag determined at
20 step 704 is provided to the binary arithmetic decoder 602. The binary arithmetic decoder 602 uses the context for the current significant coefficient flag and the contents of the encoded bitstream 113 to arithmetically decode a decoded significant coefficient flag 601 associated with one of the residual coefficients in the bitstream 113 to determine a value for the current significant coefficient flag.

25 The value of the decoded significant coefficient flag 601 may be stored within the memory 306 or hard disk drive 310. The values of each decoded significant coefficient flag 601 is used to decode associated residual coefficients from the bitstream 113.

At step 706, the value of the decoded significant coefficient flag 601 determined at
30 step 705 is used to update the context for the context index associated with the decoded significant coefficient flag 601 within the context model module 603.

For the entropy decoder 202 of Fig. 6, the context model module 603 and the binary arithmetic decoder module 602 may be used to improve the overall throughput of the video decoder 200. In one implementation, as described below, the context for significant

- 29 -

coefficient flags associated with both “luma” residual coefficients and “chroma” residual coefficients are determined when decoding the bitstream 113. The determination of a context for a significant coefficient flag for forward vertical scans and forward horizontal scans of a transform unit, in order to perform a residual coefficient inverse binarisation, in accordance with the method 700, is dependent on neighbouring significant coefficient flags, as described in detail below.

The scan pattern of a transform unit defines the order in which values of the significant coefficient flags of the transform unit are arithmetically encoded by the video encoder 100 and arithmetically decoded by the video decoder 200. Depending on the scan pattern and the location of the specific significant coefficient flag, the specific significant coefficient flag may have a “nearest previous neighbouring” significant coefficient flag associated with the specific significant coefficient flag. The nearest previous neighbouring significant coefficient flag is defined as a significant coefficient flag that immediately precedes the specific significant coefficient flag in the scan order and is spatially located adjacent to the specific significant coefficient flag.

In the example of Fig. 8A, when the scan pattern of the transform unit 800 is the horizontal scan pattern, as seen in Fig. 8A, values of the significant coefficient flags in the dependency map for significant coefficient flag 802 (e.g., 801, 809, 820 and 821) indicated by dots in Fig. 8A, may be used to determine the context index for the significant coefficient flag 802 (which is shown shaded in Fig. 8A), without knowledge of the value of the nearest previous neighbouring significant coefficient flag 805 (i.e., as indicated by stripes in Fig. 8A). The context index for the significant coefficient flag 802 may be determined by summing the values of the significant coefficient flags within the dependency map for the significant coefficient flag 802.

In the example of Fig. 8A, the significant coefficient flags within the dependency map for significant coefficient flag 802 are significant coefficient flags 809, 801, 820, 821. In other examples, the dependency map for the significant coefficient flag 802 may consist of different or additional significant coefficient flags and the determination of the context index for significant coefficient flag 802 may be determined via a method other than the summing of the values of the significant coefficient flags within the dependency map.

Determining the context for significant coefficient flags 802 without using the value of previous nearest neighbouring significant coefficient flags (e.g., 805) enables the inverse binariser 605 to provide the context index for significant coefficient flags 802 to the context

- 30 -

model module 603 before the value of the previous nearest neighbouring significant coefficient flag 815 has been determined by the entropy decoder 202 of Fig. 6.

Similarly, Fig. 8B shows the transform unit 800 where a vertical scan pattern is performed on the transform unit 800 by the video decoder 202. When the scan pattern is the vertical scan, the significant coefficient flags (e.g., 813) indicated by dots may be used to determine the context for the significant coefficient flag 816 without knowledge of a previous nearest neighbouring significant coefficient flag 814. Again, the significant coefficient flags (e.g., 813) indicated by dots in Fig. 8B represent a dependency map for determining the value of the significant coefficient flag 816.

In one example of the determination of the value of the significant coefficient flag 816 in Fig. 8B, the value of the significant coefficient flags in the dependency map for significant coefficient flag 816 have the following values:

- (i) significant flag 815 has the value '0',
- (ii) significant coefficient flag 818 has the value '1',
- (iii) significant coefficient flag 819 has the value '1'; and
- (iv) significant coefficient flag 813 has the value '1'.

The value of the significant coefficient flag 814 has not been determined. In this example, the context index for significant coefficient flag 816 is determined by summing the values of the significant coefficient flags within the dependency map for significant coefficient flag 816 (i.e. significant coefficient flags 815, 818, 819 and 813). Thus, in this example, the context index for significant coefficient flag 816 is equal to $0 + 1 + 1 + 1 = 3$. The context index may be used to determine the context for significant coefficient flag 816.

Determining the context for significant coefficient flags 816 without using the value of previous nearest neighbouring significant coefficient flags (e.g., 814) enables the inverse binariser 605 to provide the context index for significant coefficient flags 816 to the context model module 603 before the value of the previous nearest neighbouring significant coefficient flag 814 has been determined by the entropy decoder 202 of Fig. 6.

The context model module 603 provides an associated probability state and most probable symbol value (valMPS) to the binary arithmetic decoder 602. The binary arithmetic decoder 602 uses the provided probability state and the most probable symbol value (valMPS) to decode one bin from the bitstream 113 and to provide an updated probability state and most probable symbol value (valMPS) to the context model module 603. The decoded bin is provided to the inverse binariser 605.

- 31 -

As described above, the context for significant coefficient flags 802 and 816 may be determined by the binary arithmetic decoder 602 and provided to the context model module 603 of the entropy decoder 202, without depending on the previous nearest neighbouring significant coefficient flags 805 and 814. Accordingly, the context for the significant coefficient flags 802 and 816 may be provided to the context model module 603, while the binary arithmetic decoder 602 is decoding the bin corresponding to the previous nearest neighbouring significant coefficient flags 805 and 814. Using the context model module 603 and the binary arithmetic decoder 602 as described above with reference to Figs. 8A and 8B, improves the overall throughput of the entropy decoder 202.

In one implementation of the entropy decoder 202, the dependency map used to determine the context index for significant coefficient flags associated with luma residual coefficients omits the nearest previous neighbouring significant coefficient flag, as described above. However, in such an implementation, the dependency map used to determine the context index for significant coefficient flags associated with chroma residual coefficients includes the nearest previous neighbouring significant coefficient flag.

In yet another implementation of the entropy decoder 202, the dependency map used to determine the context index for significant coefficient flags associated with luma residual coefficients includes the nearest previous neighbouring significant coefficient flag, as described above. However, in such an implementation, the dependency map used to determine the context index for significant coefficient flags associated with chroma residual coefficients omits the nearest previous neighbouring significant coefficient flag.

In still another implementation, the significant coefficient flags 807 and 817, as seen in Figs. 8A and 8B, are included in the dependency map used in the determination of the context index for the significant coefficient flags 802 and 816, respectively. Accordingly, the number of significant coefficient flags included in the dependency map associated with each of the significant coefficient flags 802 and 816 is increased by one significant coefficient flag.

In still another implementation of the entropy decoder 202, the values of significant coefficient flags 809 and 818 may be assigned to the values of the significant coefficient flags 802 and 814, respectively. Assigning the values of the significant coefficient flags 809 and 818 to the values of the significant coefficient flags 802 and 814 has the effect of allowing the significant coefficient flags 805 and 814 to take on the value of a mirrored significant coefficient flag for determining the value of the significant coefficient flags 802 and 816. For example, the significant coefficient flags 805 and 814 may be temporarily assigned the value

- 32 -

of mirroring significant coefficient flags 809 and 818, respectively, for the purposes of compensating for the excluded coefficient flags 805 and 814 when determining the context index for significant coefficient flags 802 and 816. In such an implementation, the mirroring significant coefficient flags 809 and 818 may be assigned a double weighting during
5 determination of the context index for the significant coefficient flags 802 and 816.

In still another implementation of the entropy decoder 202, estimated values may be determined by the inverse binariser module 605 for the significant coefficient flags 802 and 816. The estimated values may be used in the determination of the significant coefficient flags 802 and 816. The estimated values for the significant coefficient flags 802 and 816 may
10 be determined based upon the actual values of significant coefficient flags neighbouring the significant coefficient flags 802 and 816, respectively. The estimated values may be determined based on a logical OR of values of the significant coefficient flags that are spatially located directly to the left and directly above the significant coefficient flags 805 and 814, respectively. For example, the estimated value of the significant coefficient flag 805
15 may be determined based on a logical OR of value of the significant coefficient flag 820 spatially located directly above the significant coefficient flag 805 and the significant coefficient flag 821 spatially located directly to the left of the significant coefficient flag 805. The estimated value of the significant coefficient flag 814 may be determined based on a logical OR of value of the significant coefficient flag 813 spatially located directly above the
20 significant coefficient flag 814 and the significant coefficient flag 818 spatially located directly to the left of the significant coefficient flag 814.

Fig. 9A shows part of a transform unit 900. In one implementation of the entropy decoder 202, described with reference to Figs. 9B and 9C, a single, unified set of contexts may be used for arithmetically decoding significant coefficient flags in left-edge region 902,
25 top-edge region 903 and main region 904 of the transform unit (TU) 900.

The unified set of contexts are stored in the context model module 603 configured within the memory 306 or the hard disk drive 310. Reducing the number of contexts used in the context model module 603 reduces the memory requirements and therefore the area of the entropy decoder 202 of the video decoder 200.

30 Fig. 9B shows a particular point in time during a zig-zag scan 905 of the transform unit 900 by the entropy decoder 202 when decoding the bitstream 113. Similarly, Fig. 9C shows another point in time during the zig-zag scan 905 of the transform unit 900 by the entropy decoder 202 when decoding the bitstream 113. Fig. 9D shows a particular point in

- 33 -

time during a diagonal scan of the transform unit 900 by the entropy decoder 202 when decoding the bitstream 113.

In the implementation of Figs. 9B and 9C, context modelling within the top and left edge regions (e.g., 902, 903) of the transform unit 900 and the main region (e.g., 904) of the transform unit 900, may be unified by selecting the context that will most accurately model a current significant coefficient flag (e.g., 906, 910). Selecting the context that will most accurately model a current significant coefficient flag improves context selection by approximately harmonising the context index selection for dependency maps for significant coefficient flags falling within the top edge, left edge and main regions of the transform unit 900.

The harmonised dependency maps result in a similar selection of contexts for significant coefficient flags regardless of which of the regions of the transform unit 900 that the current significant coefficient map happens to fall within. Such harmonisation may be achieved by configuring the entropy decoder 202 to predict values of significant coefficients falling outside a boundary of the transform unit 900. Furthermore, excluding an immediately prior decoded significant coefficient flag (e.g., 908) prevents any delay occurring in the selection of contexts by the inverse binariser 605 for a significant coefficient flag 906.

In the transform unit 900 shown in Fig. 9B, a dependency map for the significant coefficient flag 906 includes a significant coefficient flag 907 and predicted significant coefficient flag 909, indicated by downward cross-hatch, and a predicted significant coefficient flag 908, indicated by upward cross-hatch. The predicted significant coefficient flag 908 immediately precedes the significant coefficient flag 906 in the zig-zag scan 905. The dependency map for the significant coefficient flag 906 may be used to determine a context for the significant coefficient flag 906. The use of the predicted significant coefficient flag 908 enables the context for the significant coefficient flag 906 to be determined while the significant coefficient flag preceding the significant coefficient flag 906 in the zig-zag scan 905 is being decoded by the binary arithmetic decoder 602. The value of the predicted significant coefficient flag 908 may be determined by duplicating the value of the significant coefficient flag 907. The values of the predicted significant coefficient flag 909 may be determined by duplicating the values of the significant coefficient flag 907 and the predicted significant coefficient flag 908. The context index of the significant coefficient flag 906 may be determined by the summation of the significant coefficient flag 907, the predicted significant coefficient flag 908 and the predicted significant coefficient flags 909.

- 34 -

In the transform unit 900 shown in Fig. 9C, a dependency map for a significant coefficient flag 910 includes both of significant coefficient flag 911 and predicted significant coefficient flag 912 indicated by a downward cross-hatch. The significant coefficient flags 911 located adjacent to the significant coefficient flag 910 does not immediately precede the significant coefficient flag 910 in the zig-zag scan 905. The dependency map for the significant coefficient flag 910 may be used to determine a context index for the significant coefficient flag 910. The value of each of the predicted significant coefficient flags 912 may be determined by duplicating a diagonally adjacent value from the significant coefficient flags 911. The context index may then be calculated by the summation of the values of each of the significant coefficient flags 911 and each of the predicted significant coefficient flags 912. The context index is used to select one context from the context model 603.

In each of the transform units depicted in Figs. 9B and 9C, the steps described above are applicable if the zig-zag scan 905 is replaced with other scan patterns, such as the vertical or the horizontal scan patterns.

In the transform unit 900 shown in Fig. 9D, a dependency map for a significant coefficient flag 914 includes significant coefficient flag 915 and predicted significant coefficient flag 916. The transform unit 900 shown in Fig. 9D uses a diagonal scan pattern 913, from the lower-left to the upper-right of the transform unit 900. The diagonal scan pattern 913 scans the transform unit 900 in the order depicted by the solid grey lines. Discontinuities in the diagonal scan pattern 913 are indicated by dotted grey lines. The dependency map for the significant coefficient flag 914 may be used to determine a context index for the significant coefficient flag 914. The value of each of the predicted significant coefficient flags 916 may be determined by duplicating the diagonally adjacent values of each of the significant coefficient flags 915. The context index may then be calculated by the summation of the values of each of the significant coefficient flags 915 and each of the predicted significant coefficient flags 916. The context index may then be used to select one context from the context model 603.

In the transform unit 900 shown in Fig. 9D, the steps described above for determining a context index for significant coefficient flag 914 are also applicable when a diagonal scan pattern, from the upper-right to the lower-left of the transform unit 900 is applied.

Fig. 10A shows the transform unit 800 of Fig. 8A where a horizontal scan is performed by the entropy decoder 202 on the transform unit 800. However, in contrast to the example of Fig. 8A, in the example of Fig. 10A, a spatial layout of the significant coefficient

- 35 -

flags (e.g., 1001), indicated by dots in Fig. 10A, and which are used to determine the context for the significant coefficient flag 802, is stretched along the backwards direction of the scan pattern as shown in Fig. 10A. Accordingly, the dependency map associated with the significant coefficient flag 802 is stretched along the backwards direction of the scan pattern.

5 Similarly, Fig. 10B shows the transform unit 800 of Fig. 8B where a vertical scan is performed by the entropy decoder 202 on the transform unit 800. However, in contrast to the example of Fig. 8B, in the example of Fig. 10B, a spatial layout of the significant coefficient flags (e.g., 1003) (i.e., indicated by dots) used to determine the context for the significant coefficient flag 816 is stretched along the backwards direction of the scan pattern.
10 Accordingly, the dependency map associated with the significant coefficient flag 816 for the vertical scan of Fig. 8B is stretched along the backwards direction of the scan pattern.

For a horizontal scan, as shown in Fig. 10A, five significant coefficient flags (e.g., 1001) (i.e., indicated by dots in Fig. 10A) represent the dependency map for the significant coefficient flag 802 and may be used to determine the context for the significant coefficient
15 flag 802. The five significant coefficient flags may be selected based on close spatial proximity to the significant coefficient flag 802 and alignment of the significant five significant coefficient flags (e.g., 1001) (i.e., indicated by dots) along the backwards direction of the scan.

Similarly, for a vertical scan, as shown in Fig. 10B, five significant coefficient flags
20 (e.g., 1003) (i.e., as indicated by dots in Fig. 10B) represent the dependency map for the significant coefficient flag 816 and may be used to determine the context for the significant coefficient flag 816. The five significant coefficient flags are selected based on close spatial proximity to the significant coefficient flag 816 and alignment of the five significant coefficient flags (e.g., 1001) (i.e., indicated by dots) along the backwards direction of the
25 scan.

Altering the dependencies on neighbouring significant coefficient flags (i.e., the dependency map) for context index determination according to a selected scan pattern for the significant coefficient flags and the location of the significant coefficient flag within the transform unit, enables improvement of the context selection in the decoding of the significant
30 coefficient flags using the video decoder 200 and the encoding of significant coefficient flags using the video encoder 100.

The throughput of decoding the significant coefficient flags within a transform unit may be achieved by "pipelining" decoding of the significant coefficient flags such that the

- 36 -

context index for one significant coefficient flag can be determined, before all of the values of the significant coefficient flags within a dependency map for the transform unit have been determined.

In one implementation described with reference to Figs. 11A and 11B, context modelling between the top edge region 902, the left edge region 903 and the main region 904 of the transform unit 900, may be improved by selecting the context that will most accurately model current significant coefficient flags 1105 and 1106 seen in Figs. 11A and 11B, respectively. When the current significant coefficient flag 1105 falls close to a boundary of the transform unit (TU) 900, a dependency map containing significant coefficient flag 1101 references locations outside the transform unit (TU) 900 boundary. Similarly, when the current significant coefficient flag 1106 falls close to a boundary of the transform unit (TU) 900, a dependency map containing significant coefficient flag 1103 normally references locations outside the boundary of the transform unit (TU) 900.

In the implementation described with reference to Figs. 11A and 11B, context selection may be improved by wrapping around the dependency maps along an opposing side of the transform unit (TU) 900. As shown in Fig. 11A, the dependency map containing significant coefficient flags 1110, 1111 and 1112 is wrapped around, along the backwards direction of the horizontal scan, to an opposing side of the transform unit 900 compared to the significant coefficient flag 1105. Similarly, in Fig. 11B, the dependency map containing significant coefficient flags 1113, 1114 and 1115, is wrapped around, along the backwards direction of the vertical scan, to an opposing edge of the transform unit 900 compared to the significant coefficient flag 1106. Such wrapping around may be implemented using a shift operation where dependent significant coefficient flags (e.g., 1101) falling outside the left edge region 903 of the transform unit (TU) 900 are wrapped around to a right side of the transform unit (TU) 900 and shifted up by one row as seen in Fig. 11A. Similarly, dependent significant coefficient flags (e.g., 1103) falling outside the top edge region 902 of the transform unit (TU) 900 are wrapped around to a bottom side of the transform unit (TU) 900 and shifted left by one column as seen in Fig. 11B. The shifting operation maintains causality of the entropy decoder 200, as the dependency map for a significant coefficient flag (e.g., 1105 and 1106) cannot reference any significant coefficient flags that have not yet been decoded. In all cases, significant coefficient flags 1102 and 1104 (i.e., as indicated by stripes) are not used in the dependency maps, to provide for the determination of the context index for

- 37 -

significant coefficient flags 1105 and 1106 to be pipelined with the determination of the value of significant coefficients 1102 and 1104 by the arithmetic decoder 602.

Fig. 12A shows a particular point in time during a zig-zag scan of a transform unit 1200 by the entropy decoder 202 when decoding the bitstream 113. Similarly, Fig. 12B shows another point in time during the zig-zag scan of the transform unit 1200 by the entropy decoder 202 when decoding the bitstream 113. As shown in Figs. 12A and 12B, when the zig-zag scan is applied to significant coefficient flags in the transform unit (TU) 1200, the scan direction changes each time an upper-most row or left-most column, respectively, is reached.

As described above, performance of the binary arithmetic coding algorithm may be increased by improving the selection of the dependency map for the determination of the context index for a significant coefficient flag, while not using the nearest previous neighbouring significant coefficient flag (e.g., 1205 or 1204) in the selection of a context index for a current significant coefficient flag (e.g., 1202 or 1206).

However, in another implementation, instead of omitting the nearest previous neighbouring significant coefficient flags 1205 or 1204 from the dependency map used in the determination of the context index for the significant coefficient flags 1202 or 1206, respectively, an estimated value for the nearest previous neighbouring significant coefficient flags 1205 or 1204 may be predicted from other significant coefficient flag values that are available to compensate for the exclusion of flags. In particular, as seen in Fig. 12A, the value of a nearest, previously decoded, neighbouring significant coefficient flag 1201 along a top edge of the transform unit 1200 may be applied to the significant coefficient flag 1205. In the example of Fig. 12A, the dependency map for the significant coefficient flag 1202 is the significant coefficient flag 1201.

Similarly, as seen in Fig. 12B, the value of a nearest, previously decoded, neighbouring significant coefficient flag 1203 along a left edge of the transform unit 1200 may be applied to the significant coefficient flag 1204. Again, in the example of Fig. 12B, the dependency map for the significant coefficient flag 1206 is the significant coefficient flag 1203.

In the implementation described with reference to Figs. 12A and 12B, the assigned value of the significant coefficient flags 1205 and 1204 may be used in the determination of the context index for the significant coefficient flags 1202 and 1206, respectively, instead of a determined value of the respective significant coefficient flags 1205 and 1204. In the

- 38 -

entropy encoder 104, the determined value of the respective significant coefficient flags 1205 and 1204 is provided by values of predetermined significant coefficient flags 407. In the entropy decoder 202, the determined value of the respective significant coefficient flags 1205 and 1204 are provided by the value of the decoded significant coefficient flag 601 when the binary arithmetic decoder 602 completes a binary arithmetic decoding operation. In this instance, the next nearest available neighbouring significant coefficient flag 1201 and 1203 contribute twice in the determination of the context index for the current significant coefficient flags 1202 and 1206, respectively, and the actual value of the nearest previous neighbouring significant coefficient flags 1205 and 1204 do not contribute to the context selection process.

In one example, at a particular point in time for the implementation described with reference to Fig. 12A, the value of the significant coefficient flag 1201 has been previously determined to be logical '1' through a binary arithmetic decoding operation performed by the binary arithmetic decoder 602. In accordance with the example, at the particular point in time, the actual value of the significant coefficient flag 1205 has not yet been determined. The inverse binariser 605 determines the dependency map for the determination of the context index for the significant coefficient flag 1202 to consist of the significant coefficient flag 1201 and the estimated value for the nearest previous neighbouring significant coefficient flag 1205. The inverse binariser 605 determines the estimated value for the nearest previous neighbouring significant coefficient flag 1205 to be equal to the value of the significant coefficient flag 1201. Thus, the estimated value for the nearest previous neighbouring significant coefficient flag 1205 is logical '1'. The inverse binariser 605 uses the dependency map to determine the context index for the significant coefficient flag 1202. In the present example, the inverse binariser 605 sums the values of the significant coefficient flags within the dependency map (i.e. significant coefficient flag 1201 and estimated significant coefficient flag 1205) to determine the context index for the significant coefficient flag 1202. Thus the value of the context index for the significant coefficient flag 1202 is equal to $1 + 1 = 2$.

Fig. 13A shows a transform unit 1300. In one implementation of the entropy decoder 202, described with reference to Fig. 13B, two sets of contexts may be used for arithmetically decoding significant coefficient flags (e.g., 1301 and 1302) in various regions of the transform unit (TU) 1300. As seen in Fig. 13A, the transform unit 1300 includes a top-left region comprising significant coefficient flags (e.g., 1301) which are shaded in Fig. 13A. The

- 39 -

transform unit 1300 includes a main region (shown as unshaded) comprising significant coefficient flags (e.g., 1302).

Fig. 13B shows a particular point in time during a reverse zig-zag scan 1303 of the transform unit 1300 by the entropy decoder 202 when decoding the bitstream 113. The reverse zig-zag scan begins from a last significant coefficient 1303 and progresses towards the upper-left corner of transform unit 1300. A dependency map for the significant coefficient flag 1306 includes significant coefficient flag 1305, significant coefficient flags 1308 and a predicted significant coefficient flag 1307. The dependency map for the significant coefficient flag 1306 necessarily differs from the dependency map for significant coefficient flags 802, 816 due to the use of a reverse scan and the need to reference temporality preceding significant coefficient flags in order to determine the context index. The predicted significant coefficient flag 1307 is located adjacent to the significant coefficient flag 1306.

A context index for significant coefficient flag 1306 may be determined prior to decoding all the values of the significant coefficient flags in the dependency map with reference to the predicted significant coefficient flag 1307. The predicted significant coefficient flag 1307 may be determined by determining the summation of each of the values of the significant coefficient flags indicated by dots (e.g., 1308) and comparing a result of the summation with a threshold. If the threshold is exceeded, the predicted significant coefficient flag 1307 is assigned a value of '1'. Otherwise, the predicted significant coefficient flag 1307 is assigned a value of '0'. Alternative implementations may determine the value of the predicted significant coefficient flag 1307 by duplicating one of the significant coefficient flags indicated by dots (e.g., 1308), or by applying a logical operator such as an OR or an AND to a subset of the significant coefficient flags indicated by dots (e.g., 1308). Application of an OR operator introduces a bias towards incrementing the context index for the significant coefficient flag 1306 if any one or more of the significant coefficient flags 1308 is a logical '1'. Application of an AND operator introduces bias towards not incrementing the context index for the significant coefficient flag 1306 if any one or more of the significant coefficient flags 1308 is a logical '0'.

The context index for the significant coefficient flag 1306 may be determined by determining the summation of significant coefficient flag 1305, significant coefficient flags indicated by dots (e.g., 1308) and predicted significant coefficient flag 1307. The use of predicted significant coefficient flag 1307 enables the determination of the context index to

- 40 -

occur prior to the value of the significant coefficient flag in the same location as predicted significant coefficient flag 1307 being determined by the binary arithmetic decoder 602.

Determining the context index for significant coefficient flag 1306 as described above is also applicable when other scan patterns, such as the vertical scan or the horizontal scan, are used on the transform unit 1300.

A method 1400 of determining the value of a significant coefficient flag for a first residual coefficient of a transform unit will now be described with reference to Fig. 14. The method 1400 may be implemented by the modules 602, 603 and 605 of the entropy decoder 202 of Fig. 6. The transform unit (TU) is configured for encoding residual coefficients of a portion of a frame of video data, for example, as represented by the bitstream 113.

The method 1400 will be described with reference to the transform unit 800 and the horizontal scan pattern shown in Fig. 8A. The method 1400 will also be described with reference to the dependency map represented by the significant coefficient flags 801, 809, 820 and 821 of Fig. 8A. However, any of the dependency maps, scan patterns, significant coefficient flag locations within the transform unit and transform unit sizes described above are applicable to the method 1400.

The method 1400 begins at step 1401, where the inverse binariser module 605, under execution of the processor 305, performs the step of selecting a dependency map for a first significant coefficient flag, associated with a residual coefficient within the transform unit, from a set of predetermined dependency maps based on a scan pattern for the transform unit 800. As described above, the dependency map may be selected from memory 306 or the hard disk drive 310. Alternatively, the dependency map may be implemented in hardware. In the present example, for the transform unit 800, the inverse binariser module 605 selects the dependency map seen in Fig. 8A for a horizontal forward scan. Alternatively, the inverse binariser module 605 may select the dependency map seen in Fig. 8B for a vertical forward scan or any of the other dependency maps and associated scan patterns described above.

In each instance, the dependency map selected at step 1401 is excluding a significant coefficient flag located adjacent to the first significant coefficient flag is being determined. For example, the dependency map represented by the significant coefficient flags 801, 809, 820 and 821 shown in Fig. 8A excludes the significant coefficient flag 805, which are located adjacent to the significant coefficient flag. Similarly, the other dependency maps described above exclude a significant coefficient flag located adjacent to the first significant coefficient flag for which the value is being determined.

- 41 -

The determination of the value of the first significant coefficient flag at step 1401 is independent of determination of the value of the excluded coefficient. As described above, the significant coefficient flag 802 may be determined without knowledge of the value of significant coefficient flag 805.

5 At the next step 1403, the inverse binariser module 605 performs the step of determining a context index for the current significant coefficient flag 802 by using the value of each significant coefficient flag identified by the dependency map (i.e., significant coefficients 809, 801, 820 and 821) and compensating for the omitted significant coefficient flag 805 using the value of at least one significant coefficient flag of the dependency map (i.e., at least one of
10 significant coefficients 809, 801, 820 and 821). As described above, the context index for the significant coefficient flag 802 may be determined by summing the values of the significant coefficient flags 809, 801, 820 and 821 within the dependency map for the significant coefficient flag 802.

At step 1405, the binary arithmetic decoder 602 receives the context value from the
15 context model module 603 configured within the memory 603 or hard disk drive 610 using the context index determined at step 1403. Also at step 1405, the binary arithmetic decoder 602 performs the step of determining the value of the first significant coefficient flag using the context selected according to the determined context index and the contents of the bitstream 113. Also at step 1405, the context model module 603 updates the context value
20 associated with the context index determined at step 1403 according to the value of the significant coefficient flag determined previously in step 1405.

Industrial Applicability

The arrangements described are applicable to the computer and data processing industries and particularly for the digital signal processing.

25 The foregoing describes only some embodiments of the present invention, and modifications and/or changes can be made thereto without departing from the scope and spirit of the invention, the embodiments being illustrative and not restrictive.

In the context of this specification, the word "comprising" means "including principally but not necessarily solely" or "having" or "including", and not "consisting only of". Variations of the word "comprising", such as "comprise" and "comprises" have
30 correspondingly varied meanings.

Claims:

1. A method of determining a value of a first significant coefficient flag in a video
5 decoder for a first residual coefficient of a transform unit, the method comprising:
selecting a dependency map for use on a main region of the transform unit, the
selected dependency map being selected from a set of predetermined dependency maps based
on a scan pattern for the transform unit and excluding a second significant coefficient flag
located adjacent to the first significant coefficient flag, wherein the determination of the value
10 of the first significant coefficient flag is independent of determination of the value of the
excluded second significant coefficient flag;
determining a context index for the first significant coefficient flag by using each
significant coefficient flag identified by the dependency map; and
determining the value of the first significant coefficient flag of the transform unit,
15 using a context value selected according to the determined context index.
2. A method according to claim 1, wherein the scan pattern for the transform unit is a
horizontal scan pattern.
- 20 3. A method according to claim 2, wherein the horizontal scan pattern is a reverse scan.
4. A method according to claim 1, wherein the scan pattern for the transform unit is a
vertical scan pattern.
- 25 5. A method according to claim 4, wherein the vertical scan pattern is a reverse scan.
6. A method according to claim 1, wherein the scan pattern is a reverse scan.
7. A method according to claim 1, wherein the second coefficient flag is a nearest
30 previous neighbouring coefficient flag.
8. A system for determining a value of a first significant coefficient flag in a video
decoder for a first residual coefficient of a transform unit, the system comprising:

- 43 -

a memory comprising data and a computer program;

a processor coupled to the memory for executing the program, the program comprising instructions for:

selecting a dependency map for use on a main region of the transform unit, the
5 selected dependency map being selected from a set of predetermined dependency
maps based on a scan pattern for the transform unit and excluding a second significant
coefficient flag located adjacent to the first significant coefficient flag, wherein the
determination of the value of the first significant coefficient flag is independent of
determination of the value of the excluded second significant coefficient flag;

10 determining a context index for the first significant coefficient flag by using
each significant coefficient flag identified by the dependency map; and

determining the value of the first significant coefficient flag of the transform
unit, using a context value selected according to the determined context index.

15 9. An apparatus for determining a value of a first significant coefficient flag in a video
decoder for a first residual coefficient of a transform unit, the apparatus comprising:

means for selecting a dependency map for use on a main region of the transform unit,
the selected dependency map being selected from a set of predetermined dependency maps
based on a scan pattern for the transform unit and excluding a second significant coefficient
20 flag located adjacent to the first significant coefficient flag, wherein the determination of the
value of the first significant coefficient flag is independent of determination of the value of
the excluded second significant coefficient flag;

means for determining a context index for the first significant coefficient flag by using
each significant coefficient flag identified by the dependency map; and

25 means for determining the value of the first significant coefficient flag of the
transform unit, using a context value selected according to the determined context index.

10. A computer readable medium having a computer program stored thereon for
determining a value of a first significant coefficient flag in a video decoder for a first residual
30 coefficient of a transform unit, the program comprising:

code for selecting a dependency map for use on a main region of the transform unit,
the selected dependency map being selected from a set of predetermined dependency maps
based on a scan pattern for the transform unit and excluding a second significant coefficient

- 44 -

flag located adjacent to the first significant coefficient flag, wherein the determination of the value of the first significant coefficient flag is independent of determination of the value of the excluded second significant coefficient flag;

code for determining a context index for the first significant coefficient flag by using
5 each significant coefficient flag identified by the dependency map; and

code for determining the value of the first significant coefficient flag of the transform unit, using a context value selected according to the determined context index.

11. A method of encoding a value of a first significant coefficient flag in a video encoder,
10 for a first residual coefficient of a transform unit, the method comprising:

selecting a dependency map for use on a main region of the transform unit, the selected dependency map being selected from a set of predetermined dependency maps based on scan pattern for the transform unit and excluding a second significant coefficient flag located adjacent to the first coefficient flag, wherein a determination of the value of the first
15 significant coefficient flag is independent of a determination of a value of the excluded second significant coefficient flag;

determining a context index for the first significant coefficient flag by using each significant coefficient flag identified by the dependency map; and

determining the value of the first significant coefficient flag of the transform unit,
20 using a context value selected according to the determined context index.

12. A method according to claim 1, further comprising compensating for the excluded second significant coefficient flag using at least one significant coefficient flag of the dependency map.
25

13. A system for encoding a value of a first significant coefficient flag in a video encoder, for a first residual coefficient of a transform unit, the system comprising:

a memory comprising data and a computer program;

a processor coupled to the memory for executing the program, the program comprising
30 instructions for:

selecting a dependency map for use on a main region of the transform unit, the selected dependency map being selected from a set of predetermined dependency

- 45 -

maps based on scan pattern for the transform unit and excluding a second significant coefficient flag located adjacent to the first coefficient flag, wherein a determination of the value of the first significant coefficient flag is independent of a determination of a value of the excluded second significant coefficient flag;

5 determining a context index for the first significant coefficient flag by using each significant coefficient flag identified by the dependency map; and

 determining the value of the first significant coefficient flag of the transform unit, using a context value selected according to the determined context index.

10 14. An apparatus for encoding a value of a first significant coefficient flag in a video encoder, for a first residual coefficient of a transform unit, the apparatus comprising:

 means for selecting a dependency map for use on a main region of the transform unit, the selected dependency map being selected from a set of predetermined dependency maps based on scan pattern for the transform unit and excluding a second significant coefficient
15 flag located adjacent to the first coefficient flag, wherein a determination of the value of the first significant coefficient flag is independent of a determination of a value of the excluded second significant coefficient flag;

 means for determining a context index for the first significant coefficient flag by using each significant coefficient flag identified by the dependency map; and

20 means for determining the value of the first significant coefficient flag of the transform unit, using a context value selected according to the determined context index.

15. A computer readable medium having a computer program stored thereon for encoding a value of a first significant coefficient flag in a video encoder, for a first residual coefficient
25 of a transform unit, the program comprising:

 code for selecting a dependency map for use on a main region of the transform unit, the selected dependency map being selected from a set of predetermined dependency maps based on scan pattern for the transform unit and excluding a second significant coefficient flag located adjacent to the first coefficient flag, wherein a determination of the value of the
30 first significant coefficient flag is independent of a determination of a value of the excluded second significant coefficient flag;

 code for determining a context index for the first significant coefficient flag by using each significant coefficient flag identified by the dependency map; and

- 46 -

code for determining the value of the first significant coefficient flag of the transform unit, using a context value selected according to the determined context index.

16. A method of determining the value of a first significant coefficient flag in a video encoder for a first residual coefficient of a transform unit, the method comprising:

selecting a dependency map for the transform unit from a set of predetermined dependency maps based on a scan pattern for the transform unit, the selected dependency map excluding a second significant coefficient flag located adjacent to the first significant coefficient flag, wherein the determination of the value of the first significant coefficient flag is independent of determination of the value of the excluded second significant coefficient flag;

determining a context index for the first significant coefficient flag by using each significant coefficient flag identified by the dependency map and compensating for the excluded second significant coefficient flag using at least one significant coefficient flag of the dependency map; and

determining the value of the first significant coefficient flag of the transform unit, using a context value selected according to the determined context index.

17. The method according to claim 16, wherein the dependency map is also selected based on the location of the first significant coefficient flag.

18. A system for determining the value of a first significant coefficient flag in a video encoder for a first residual coefficient of a transform unit, the system comprising:

a memory comprising data and a computer program;

a processor coupled to the memory for executing the program, the program comprising instructions for:

selecting a dependency map for the transform unit from a set of predetermined dependency maps based on a scan pattern for the transform unit, the selected dependency map excluding a second significant coefficient flag located adjacent to the first significant coefficient flag, wherein the determination of the value of the first coefficient is independent of determination of the value of the excluded second coefficient;

- 47 -

determining a context index for the first significant coefficient flag by using each significant coefficient flag identified by the dependency map and compensating for the excluded second significant coefficient flag using at least one significant coefficient flag of the dependency map; and

5 determining the value of the first significant coefficient flag of the transform unit, using a context value selected according to the determined context index.

19. An apparatus for determining the value of a first significant coefficient flag in a video encoder for a first residual coefficient of a transform unit, the apparatus comprising:

10 means for selecting a dependency map for the transform unit from a set of predetermined dependency maps based on a scan pattern for the transform unit, the selected dependency map excluding a second significant coefficient flag located adjacent to the first significant coefficient flag, wherein the determination of the value of the first coefficient is independent of determination of the value of the excluded second coefficient;

15 means for determining a context index for the first significant coefficient flag by using each significant coefficient flag identified by the dependency map and compensating for the excluded second significant coefficient flag using at least one significant coefficient flag of the dependency map; and

20 means for determining the value of the first significant coefficient flag of the transform unit, using a context value selected according to the determined context index.

20. A computer readable medium having a computer program stored thereon for determining the value of a first significant coefficient flag in a video encoder for a first residual coefficient of a transform unit, the program comprising:

25 code for selecting a dependency map for the transform unit from a set of predetermined dependency maps based on a scan pattern for the transform unit, the selected dependency map excluding a second significant coefficient flag located adjacent to the first significant coefficient flag, wherein the determination of the value of the first coefficient is independent of determination of the value of the excluded second coefficient;

30 code for determining a context index for the first significant coefficient flag by using each significant coefficient flag identified by the dependency map and compensating for the

- 48 -

excluded second significant coefficient flag using at least one significant coefficient flag of the dependency map; and

code for determining the value of the first significant coefficient flag of the transform unit, using a context value selected according to the determined context index.

- 1 / 17 -

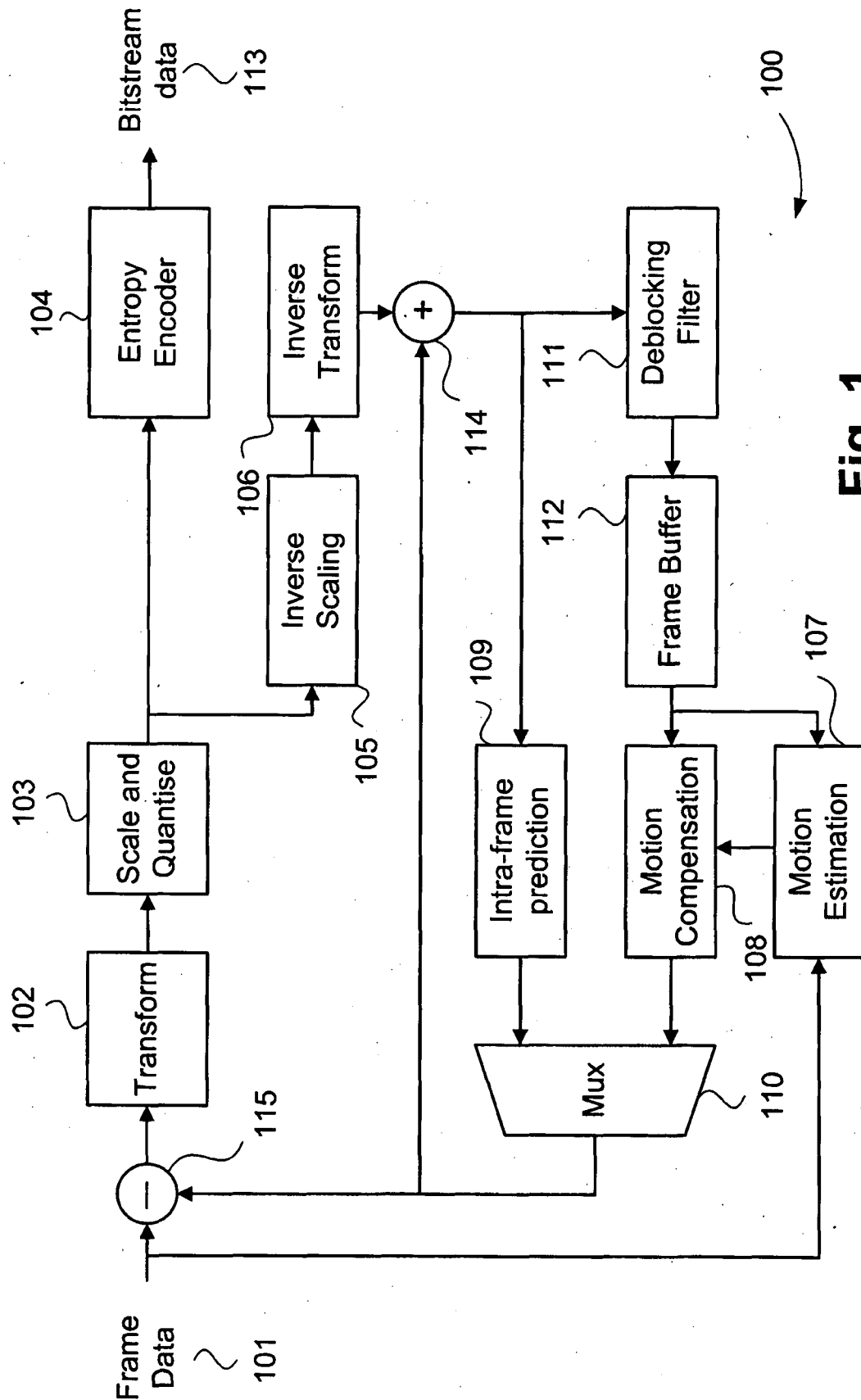
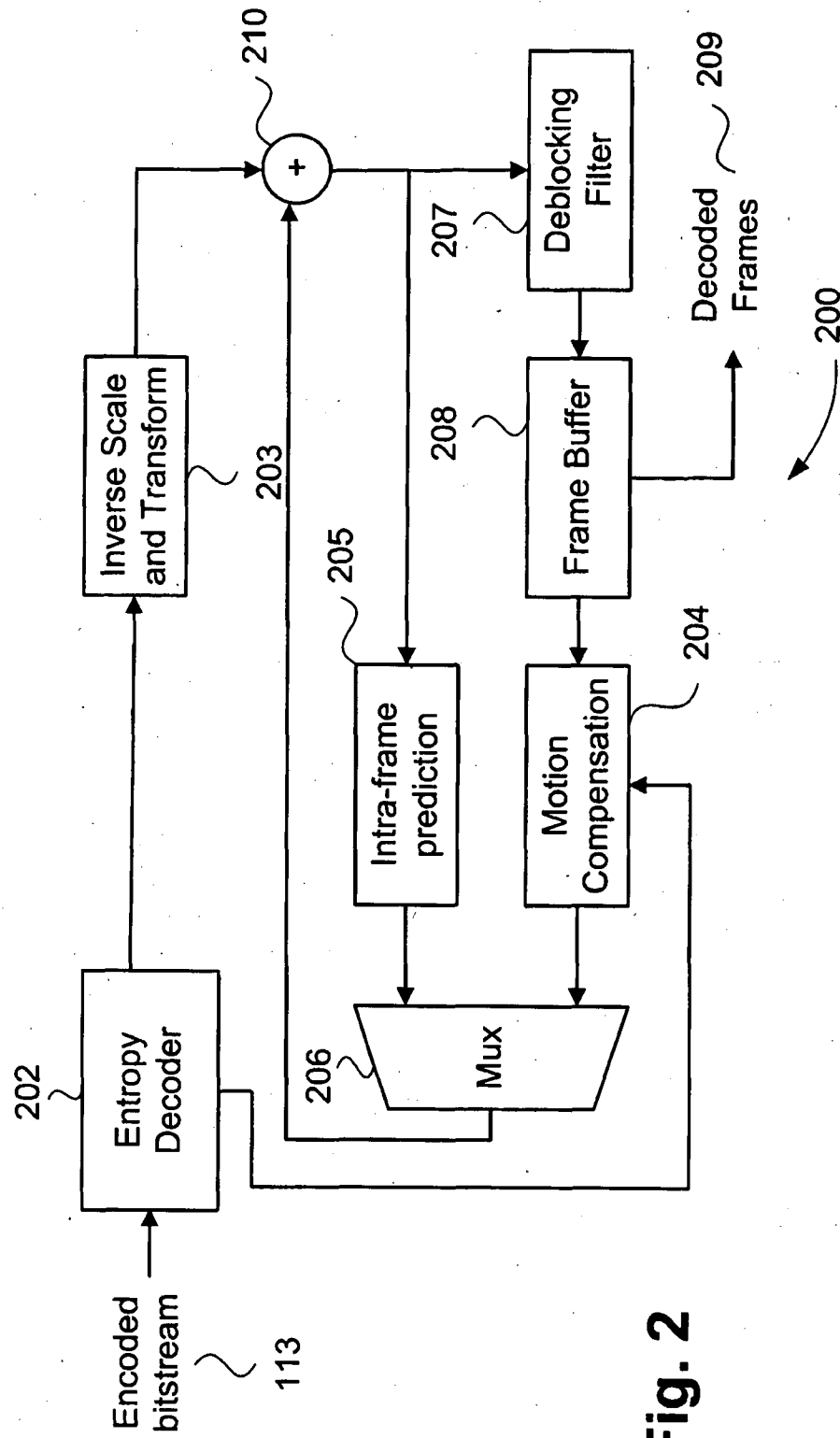


Fig. 1

- 2 / 17 -

**Fig. 2**

- 3 / 17 -

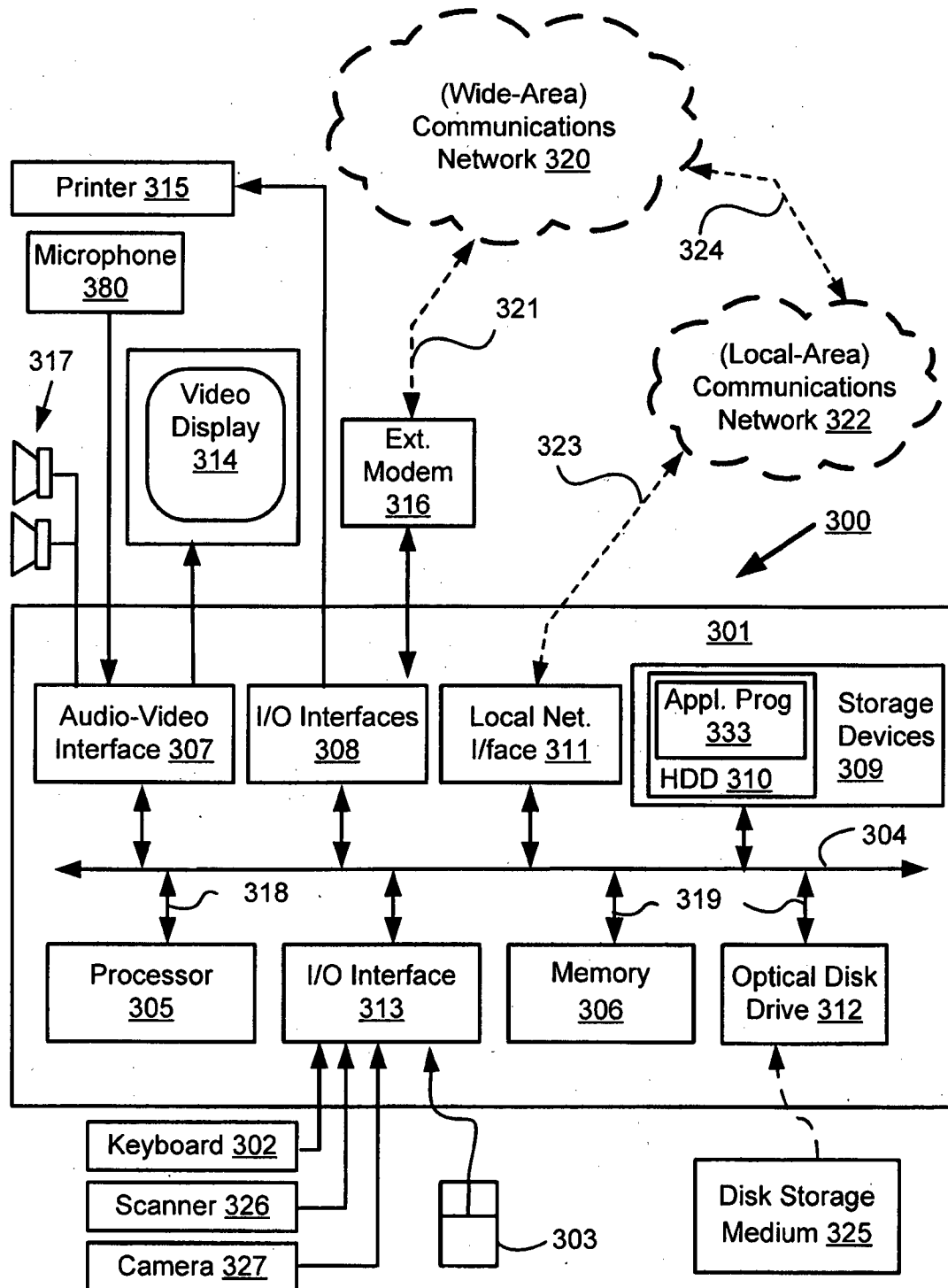


Fig. 3A

- 4 / 17 -

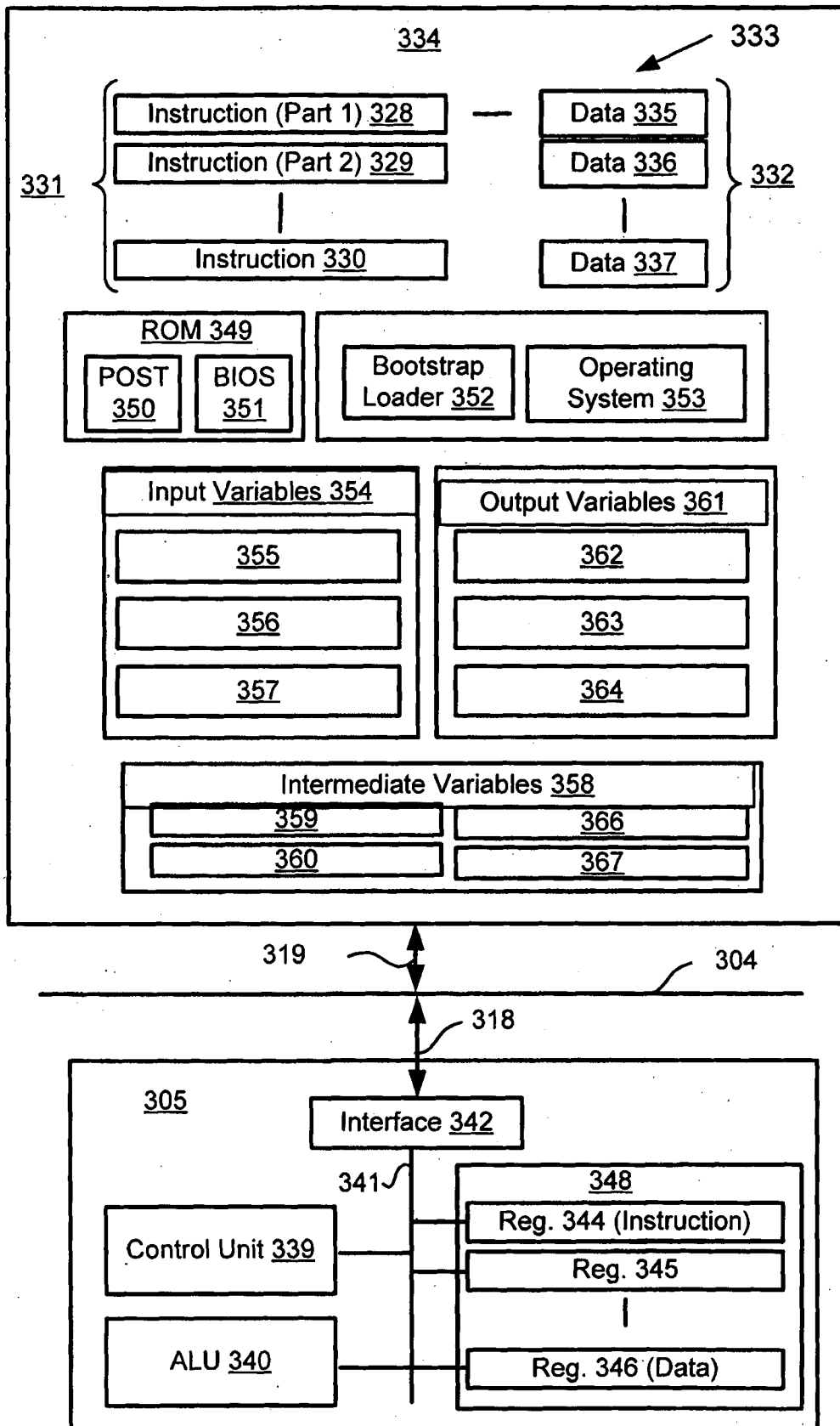
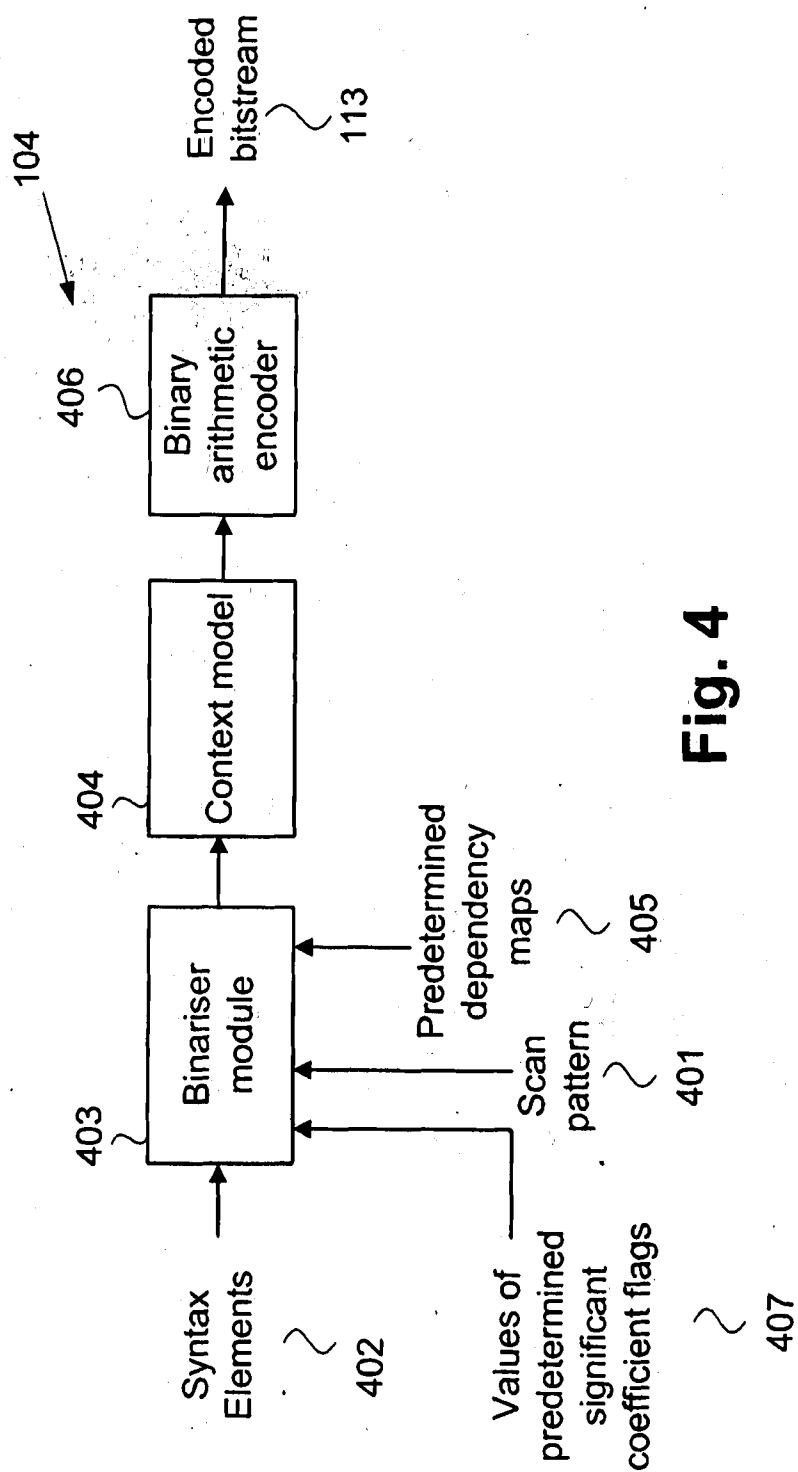
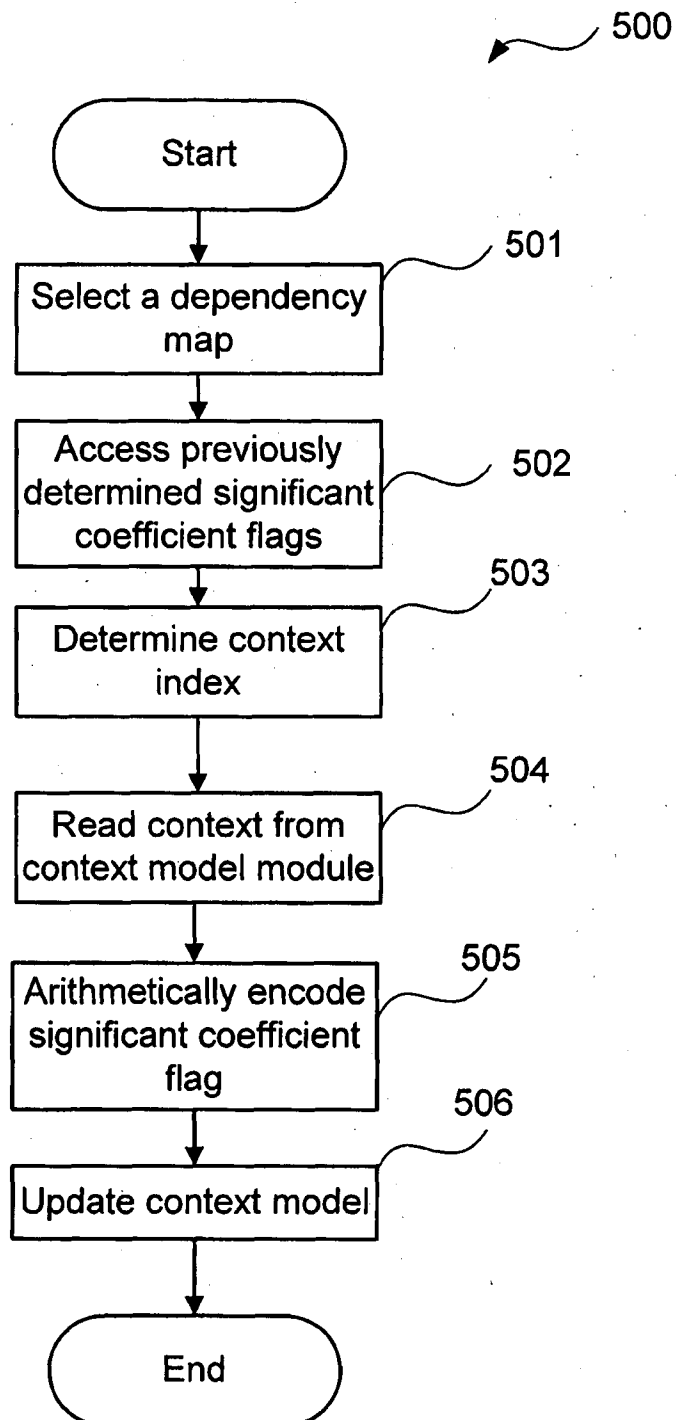


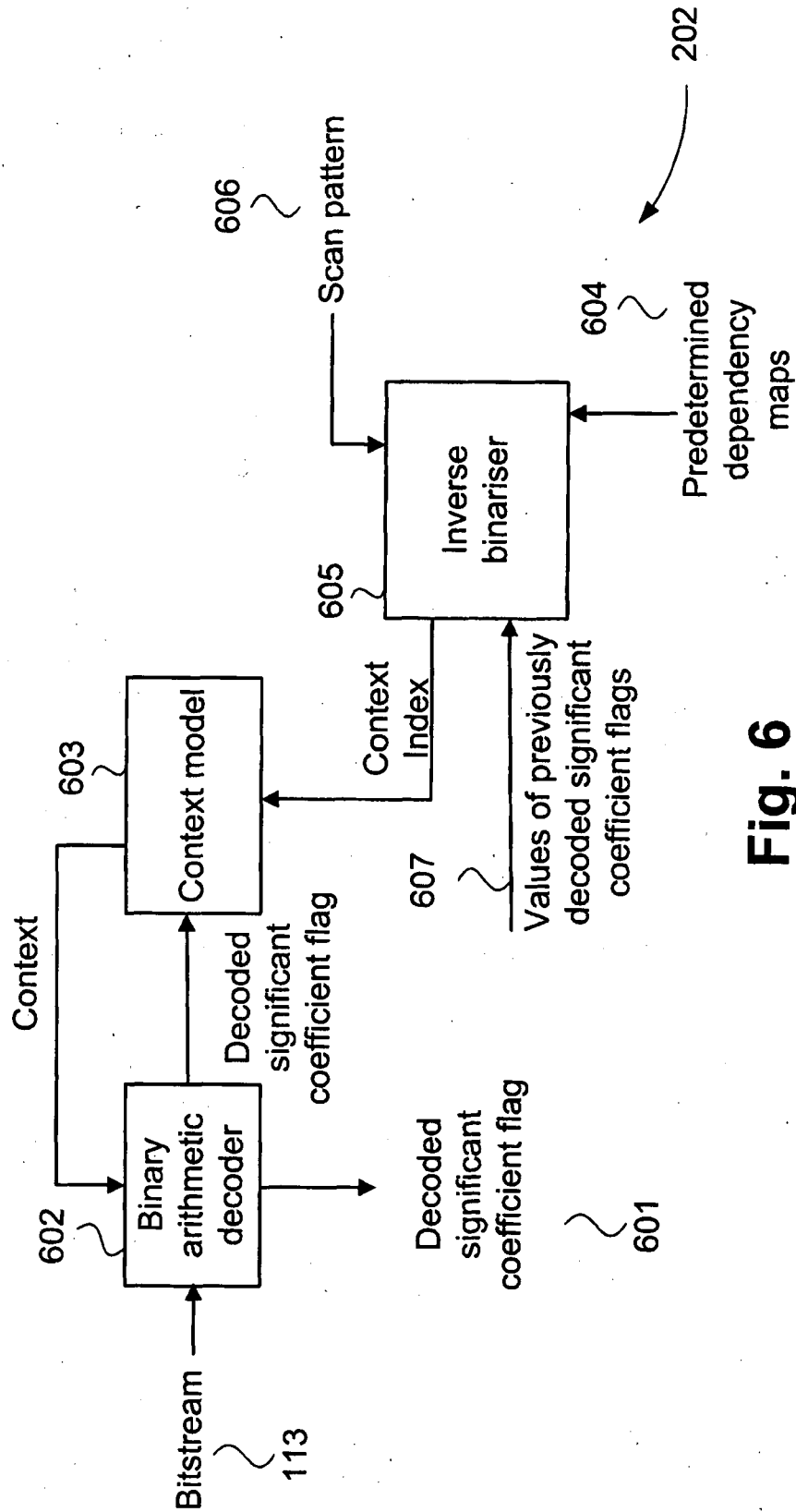
Fig. 3B



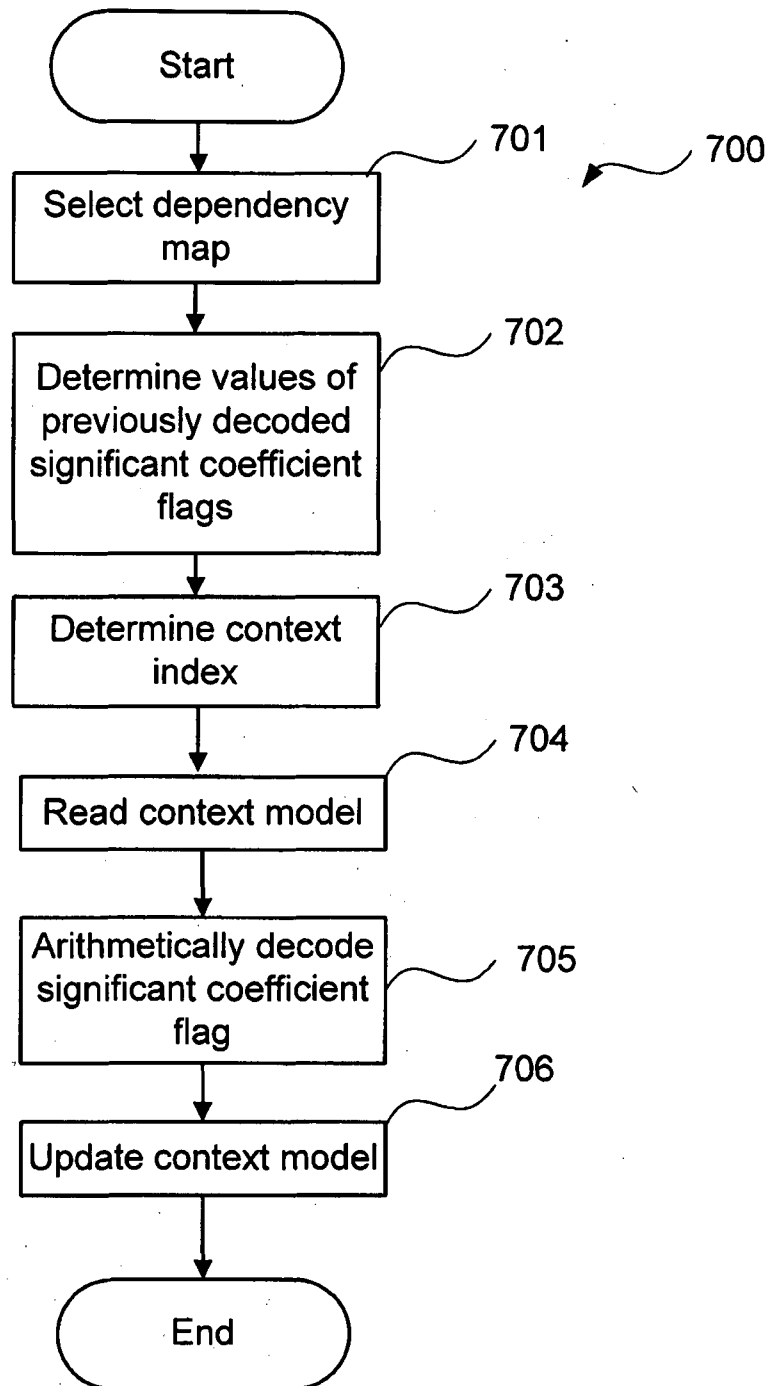
- 6 / 17 -

**Fig. 5**

- 7 / 17 -

**Fig. 6**

- 8 / 17 -

**Fig. 7**

- 9 / 17 -

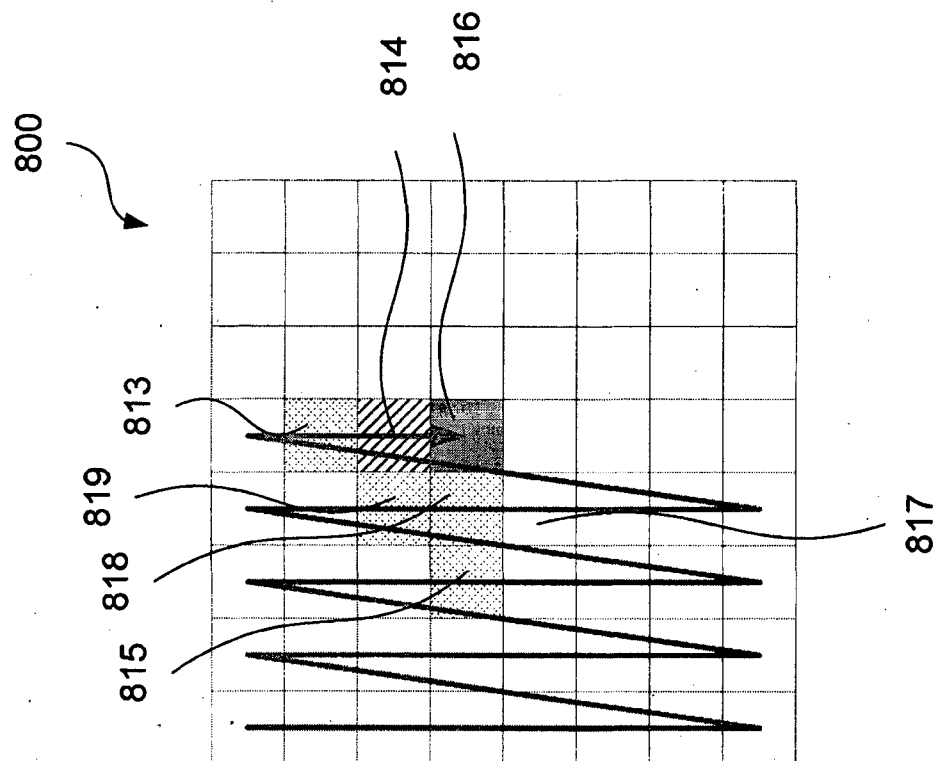


Fig. 8B

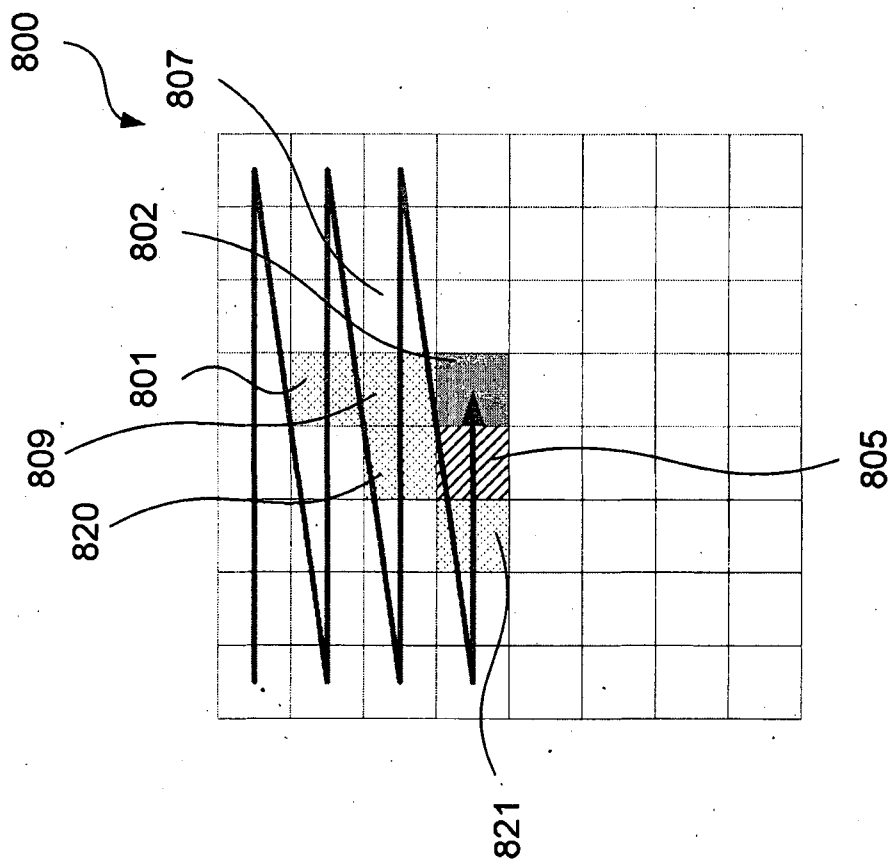


Fig. 8A

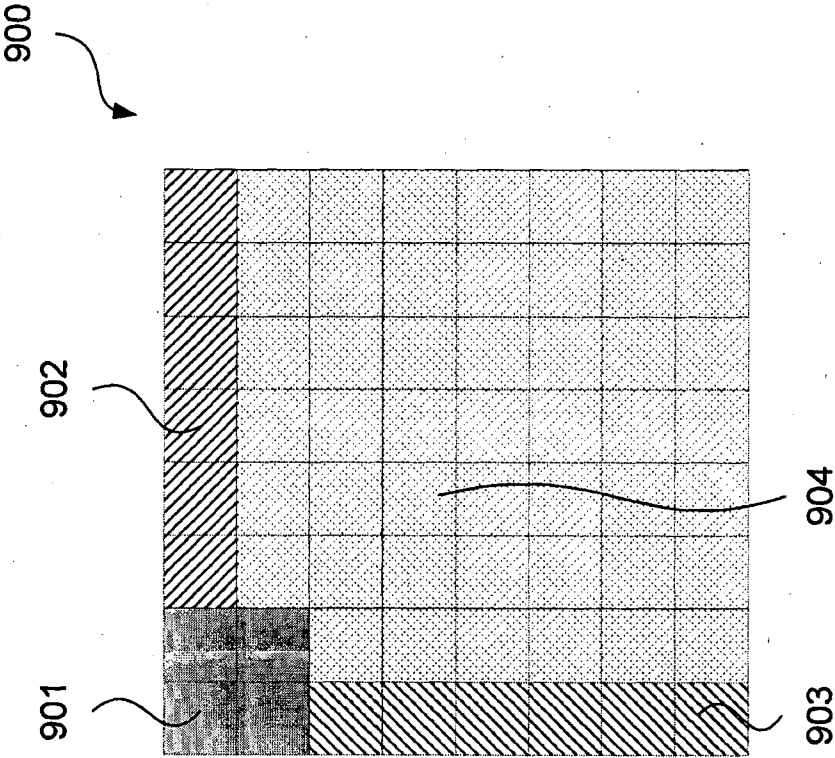


Fig. 9A

- 11 / 17 -

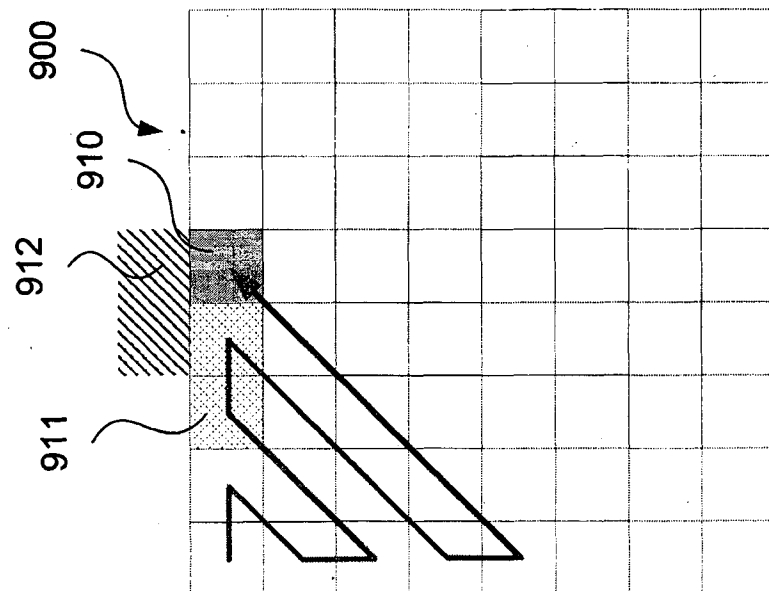


Fig. 9B

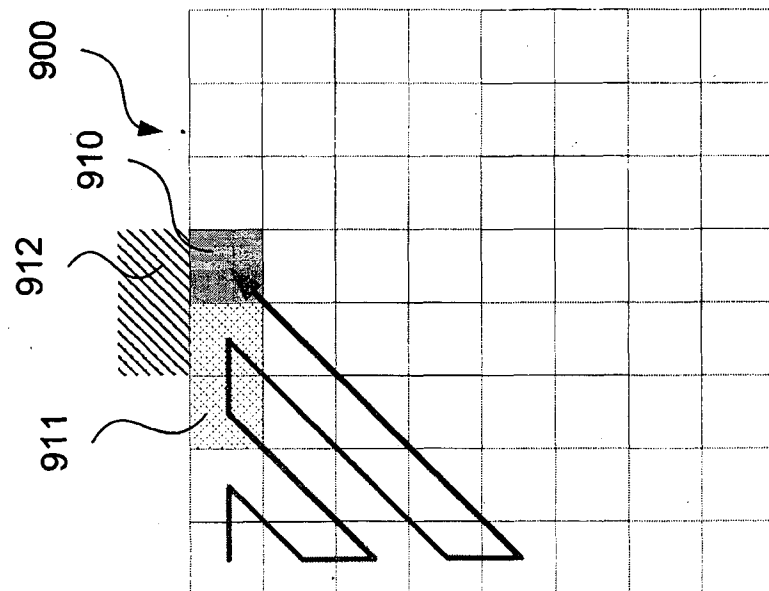


Fig. 9C

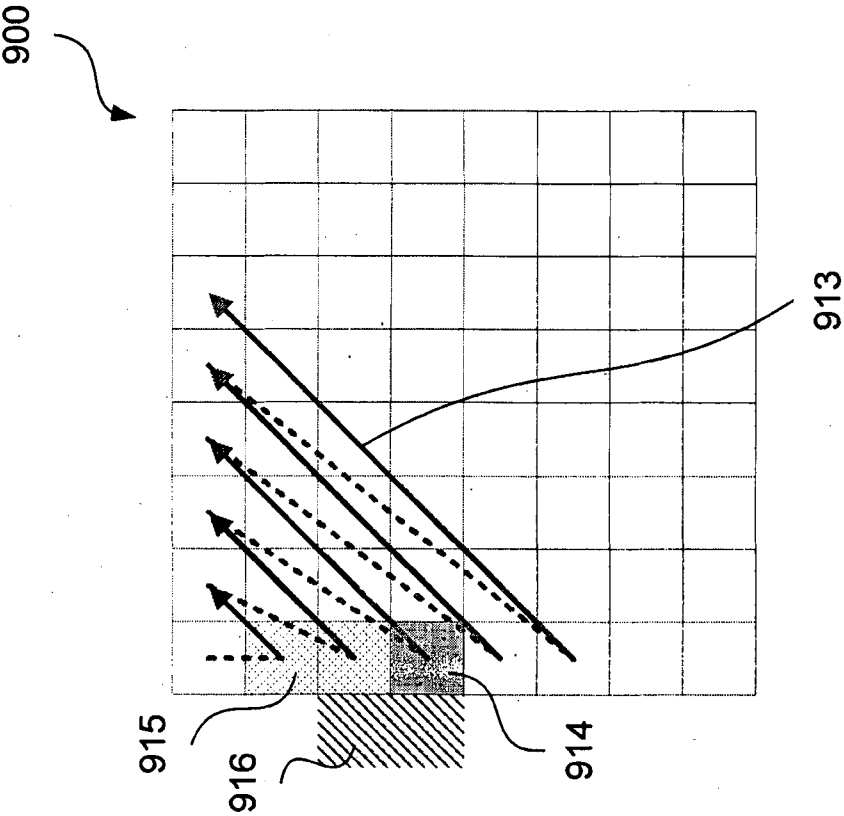


Fig. 9D

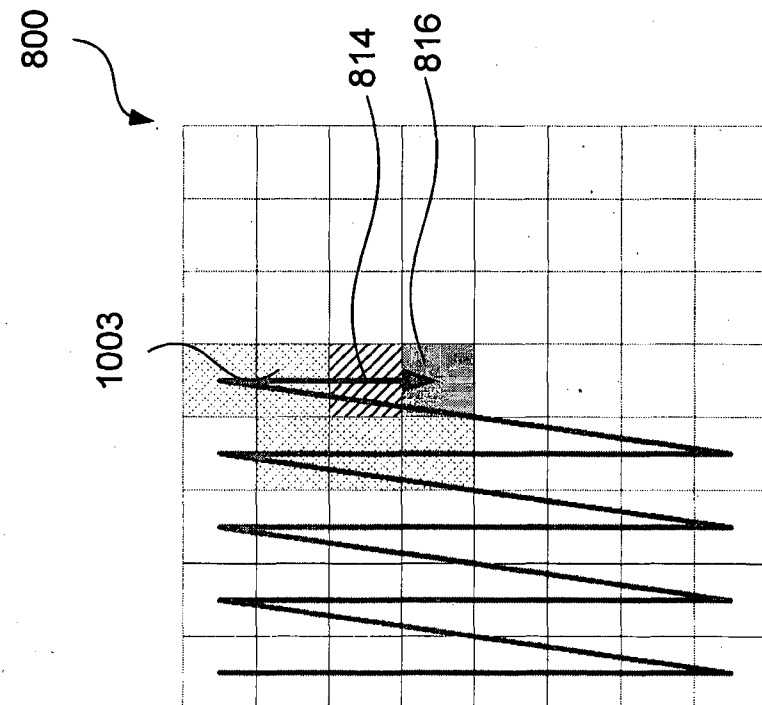


Fig. 10A

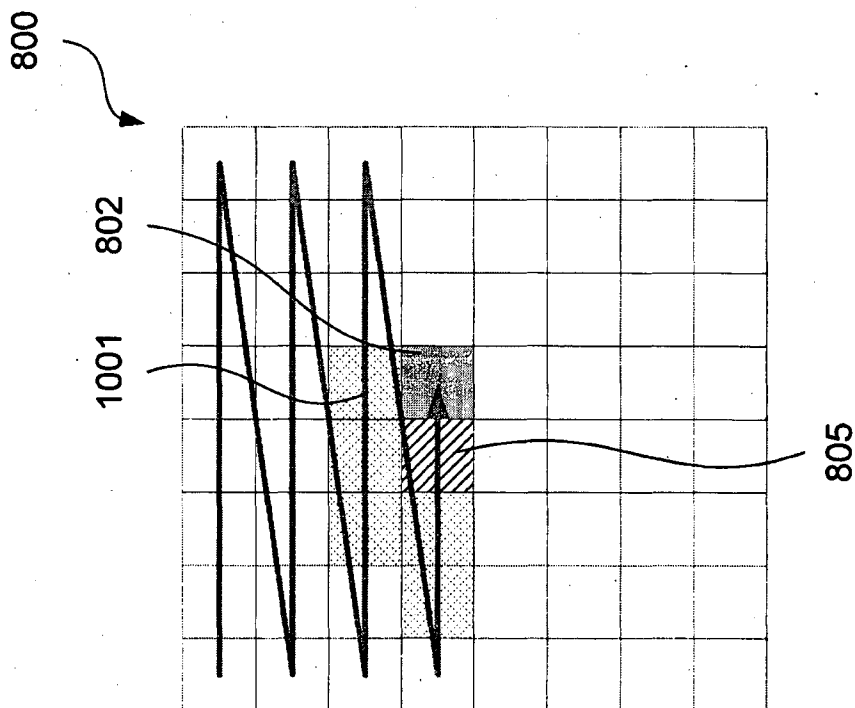


Fig. 10B

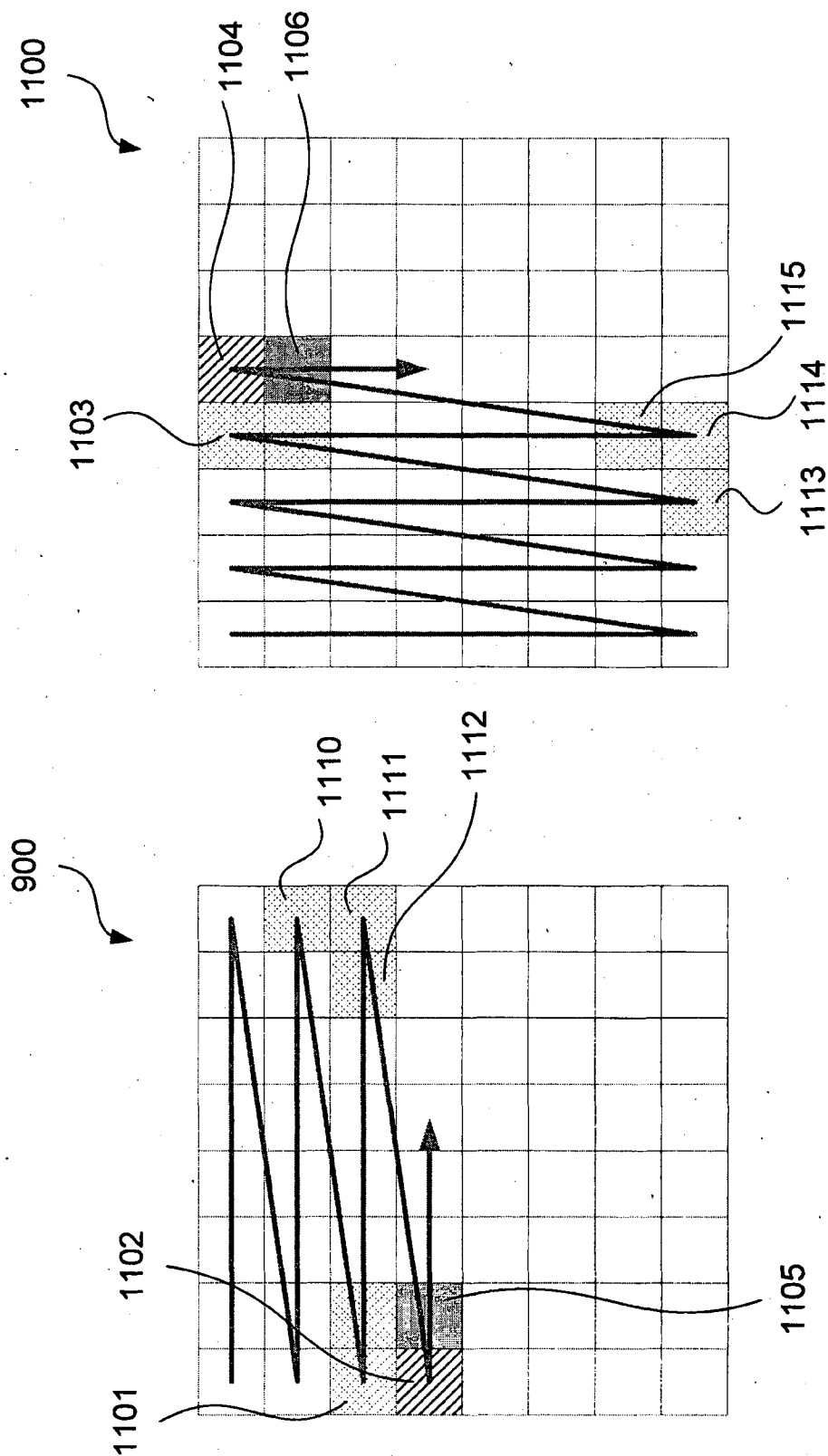


Fig. 11A

Fig. 11B

- 15 / 17 -

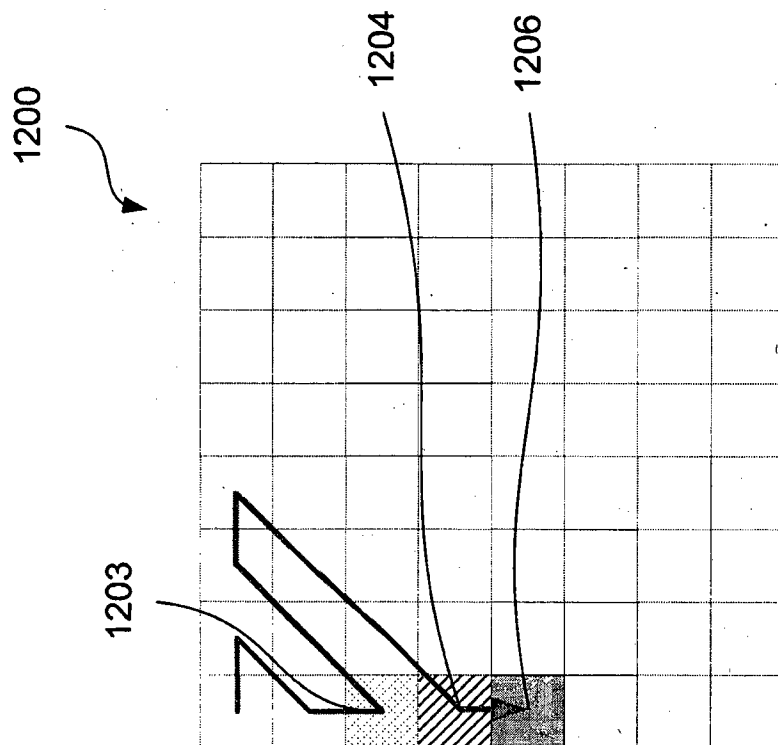


Fig. 12A

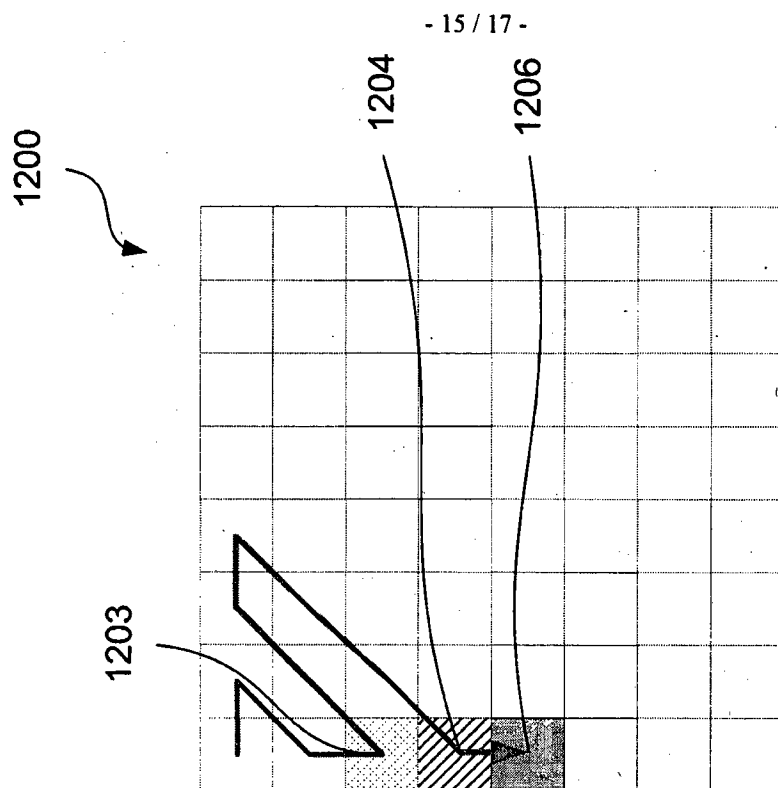


Fig. 12B

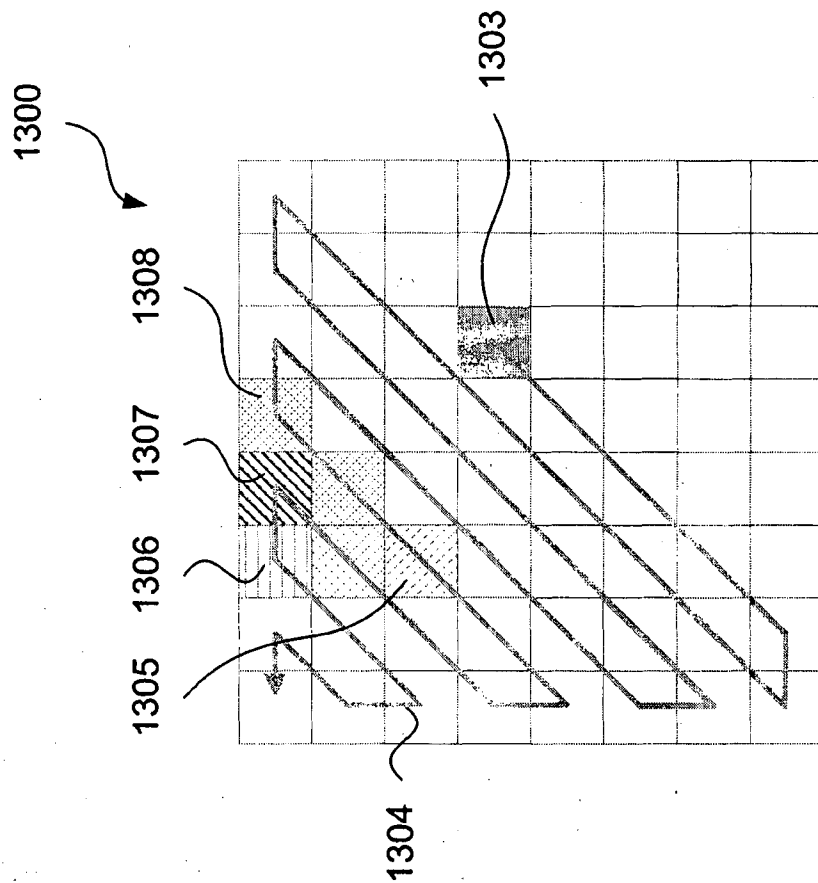


Fig. 13B

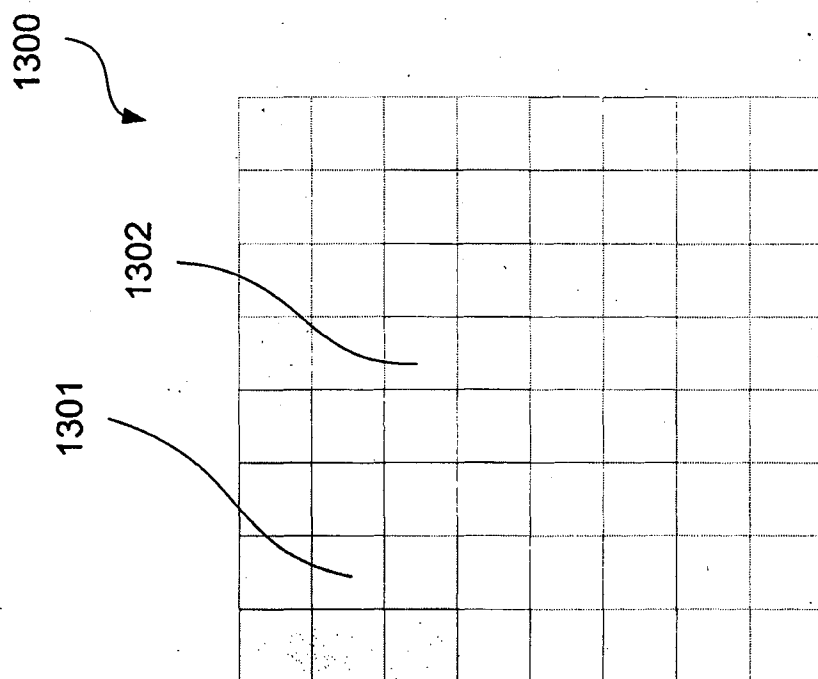
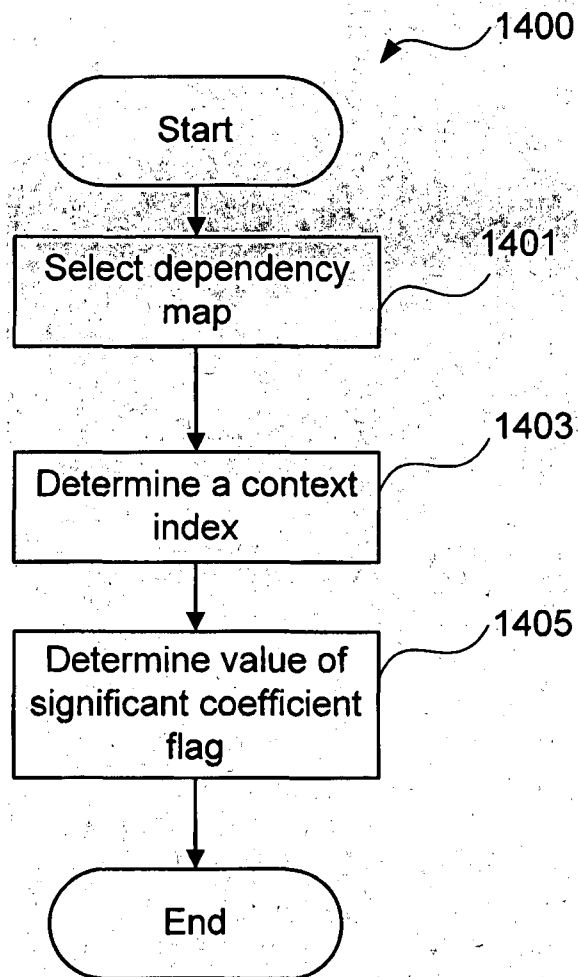


Fig. 13A

- 17 / 17 -

**Fig. 14**

INTERNATIONAL SEARCH REPORT

International application No.
PCT/AU2012/000789

A. CLASSIFICATION OF SUBJECT MATTER

H04N 7/50 (OCT 2005) H04N 7/28 (OCT 2005) G06T 9/00 (OCT 2005)

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)

EPODOC, WPI (abstracts); GOOGLE keywords: VIDEO, DECODER, DEPENDENCY MAP, FLAG, TRANSFORM, CONTEXT, SCAN and similar terms and phrases.

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 26 July 2012	Date of mailing of the international search report 30 July 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	Authorized officer Bayer Mitrovic AUSTRALIAN PATENT OFFICE (ISO 9001 Quality Certified Service) Telephone No. 0262832164

INTERNATIONAL SEARCH REPORT		International application No.
C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		PCT/AU2012/000789
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2010/0097250 A1 (DEMIRCIN ET AL) 22 April 2010 Whole document, especially abstract, paragraphs [0005], [0020]-[0028], [0033], [0038]-[0041], [0054], Fig.3	1-20
Y	WO 2006/123310 A2 (KONINKLIJKE PHILIPS ELECTRONICS N.V.) 23 November 2006 Whole document, especially abstract	1-20

Form PCT/ISA/210 (fifth sheet) (July 2009)

INTERNATIONAL SEARCH REPORT Information on patent family members		International application No. PCT/AU2012/000789	
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 2010/0097250 A1	22 Apr 2010	US 2010097250 A1	22 Apr 2010
		US 7932843 B2	26 Apr 2011
		US 2010097248 A1	22 Apr 2010
		US 8068043 B2	29 Nov 2011
		US 2010098155 A1	22 Apr 2010
WO 2006/123310 A2	23 Nov 2006	WO 2006123310 A2	23 Nov 2006
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)			