(54) Title: METHOD AND APPARATUS FOR VARIABLE LENGTH CODING OF BIT PLANES

(57) Abstract: A method and apparatus for variable length coding is described. A method comprises receiving a set of data having a set of significant values, identifying a set of positions of the set of significant values within the set of data without branching, for each of the set of positions, encoding a run of non-significant values preceding each of the set of positions.
Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM),
European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE,
ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, SE, SI,
SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN,
GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— with international search report
— before the expiration of the time limit for amending the
claims and to be republished in the event of receipt of
amendments

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.
BACKGROUND OF THE INVENTION

Field of the Invention

[0001] The invention relates to the field of data processing. More specifically, the invention relates to encoding.

Background of the Invention

[0002] In a traditional communications system, an encoder compresses a signal into a bit rate that is less than, and close to, a channel capacity. An example signal is a video signal. A video encoder compresses the video signal for transmission over a channel. A decoder receives the compressed video signal and reconstructs the video signal using all bits received from the channel. In such a model, two basic assumptions are made: 1) the encoder knows the channel capacity, and 2) the decoder is able to decode all bits received from the channel fast enough to reconstruct the video. These two basic assumptions are challenges in Internet streaming video applications.

[0003] In MPEG-2 and MPEG-4, several layered scalability techniques, namely, SNR scalability, temporal scalability, and spatial scalability, have been included. In such a layered scalable coding technique, a video sequence is coded into a base layer and an enhancement layer. The enhancement layer enhances the quality of video when received and decoded.

[0004] In MPEG-4 Fine-Granularity Scalability (FGS), the enhancement layer is encoded using bit-plane coding. An image is composed of a two-dimensional array of picture elements called pixels. Pixel values may be represented in base-2 using bits. Bits in a pixel value vary in their significance. The first bit is the 1’s place, the second is the
2's place, the third is the 4's place, etc. In most cases 8 bits describe pixel value so the
8th bit is 128's place. An image or a video may be divided into a two-dimensional
array represented by value bits at a particular bit position in the pixel value. The two-
dimensional array is called a bit-plane because a single bit represents each pixel. The
approach to compressing a bit-plane is to encode the run length of the number of zeros
between each one.

[0005] In run length encoding, non-zero values are located first. The 0's between non-
zero values are then counted and the number of 0's along with the non-zero data is
encoded. In some implementations, each value is individually tested with a conditional
operation. Each time a non-zero value is extracted, a mispredicted branch results because
of the conditional operation. Branch misprediction decreases pipeline efficiency and
slows down a processor. The slowdown of the processor becomes more pronounced as the
processor's pipeline length increases.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] The invention may best be understood by referring to the following description
and accompanying drawings that are used to illustrate embodiments of the invention. In
the drawings:

[0007] Figure 1 is a block diagram illustrating one embodiment of a computer system
in accordance with one embodiment of the invention.

[0008] Figure 2A is a flowchart for bit plane extraction according to one embodiment
of the invention.

[0009] Figure 2B is a flow chart for variable length coding of a bit plane according to
one embodiment of the invention.
[0010] Figure 3 is a flow chart for bit plane extraction according to another embodiment of the invention.

[0011] Figure 4 is a flow chart for another embodiment of variable length coding according to one embodiment of the invention.

[0012] Figure 5 is a block diagram illustrating exemplary extraction of a selected bit plane into a buffer as described in block 403 of Figure 4 according to one embodiment of the invention.

[0013] Figure 6 is a flow chart for locating the MSB with shift and compare operations according to one embodiment of the invention.

[0014] Figure 7 is an exemplary diagram illustrating components for locating the MSB of a register with daisy-chaining according to one embodiment of the invention.

[0015] Figure 8 is a flowchart for locating the MSB in a register with one or more look-up tables according to one embodiment of the invention.

DETAILED DESCRIPTION OF THE DRAWINGS

[0016] In the following description, numerous specific details are set forth to provide a thorough understanding of the invention. However, it is understood that the invention may be practiced without these specific details. In other instances, well-known circuits, structures and techniques have not been shown in detail in order not to obscure the invention.

[0017] Figure 1 is a block diagram illustrating one embodiment of a computer system 100 in accordance with one embodiment of the invention. The computer system 100 comprises a processor 110, a storage device 120, and a bus 115. The processor 110 is coupled to the storage device 120 by the bus 115. In addition, a number of user input/output devices 140 (e.g., keyboard, mouse) are also coupled to the bus 115. The
processor 110 represents a central processing unit of any type of architecture, such as CISC, RISC, VLIW, or hybrid architecture. Furthermore, the processor 110 could be implemented on one or more chips. The bus 115 represents one or more buses (e.g., AGP, PCI, ISA, X-Bus, VESA, etc.) and bridges. While this embodiment is described in relation to a single processor computer system, the invention could be implemented in a multi-processor computer system.

[0018] In addition to other devices, one or more of a video input/output card 135 may optionally be coupled to the bus 115. The storage device 120 represents one or more machine-readable media. Thus, a machine-readable medium includes any mechanism that provides (i.e., stores and/or transmits) information in a form readable by a machine (e.g., a computer). For example, a machine-readable medium includes read only memory (ROM); random access memory (RAM); magnetic disk storage media; optical storage media; flash memory devices; electrical, optical, acoustical or other form of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.), etc.

[0019] The video input/output card 135 represents one or more devices for digitizing images, capturing images, capturing video, transmitting video, etc.

[0020] The system 100 also includes a variable length coding (VLC) module 180. The VLC module 180 may be different types of hardware (such as digital logic) executing the processing described herein (which is described in more detail below). Although the VLC module 180 is illustrated as coupled with the processor 110 via the bus 115, the VLC module 180 may be implemented as circuitry and/or instructions within the processor in alternative embodiments of the invention. The VLC module 180 may be included on the video input/output card 135 as circuitry and/or instructions in other embodiments. Alternative embodiments of the present invention may implement the VLC module as code in the storage device 120.
[0021] Figure 2A is a flowchart for bit plane extraction according to one embodiment of the invention. At block 201, the VLC module 180 initializes POSITION [0] to a boundary value (e.g., -1). Also at block 201, the VLC module 180 initializes a loop counter ("LOOP_COUNTER") to 0 and initializes a bit counter ("BIT_COUNTER") to 1. At block 203, the VLC module 180 sets POSITION [BIT_COUNTER] to LOOP_COUNTER. At block 205, a flag is set to the result of a logic operation on BIT_PLANE [LOOP_COUNTER]. An example of the logic operation would be the following: ((BIT_PLANE [LOOP_COUNTER] & BIT_MASK) || 0).

[0022] At block 207, the VLC module 180 increments BIT_COUNTER with the flag. At block 209, the VLC module 180 determines if it has reached the end of the bit plane. If the VLC module 180 determines that is has not reached the end of the bit plane, then at block 211 the VLC module 180 increments LOOP_COUNTER. From block 211 control flows to block 203. If at block 209 the VLC module 180 determines that it has reached the end of the bit plane, then the VLC module 180 sets POSITION [BIT_COUNTER] to LOOP_COUNTER at block 213. From block 213 control flows to block 215.

[0023] Figure 2B is a flow chart for variable length coding of a bit plane according to one embodiment of the invention. At block 215, the VLC module 180 sets LOOP_COUNTER to 1. At block 217, the VLC module 180 sets a variable indicating a run of bits ("RUN") to the result of the following operation:


[0024] At block 219, the VLC module 180 increments a variable indicating current location within the bit plane ("ADDRESS_COUNTER") with RUN. At block 221, the VLC module 180 calls an encoding procedure and passes RUN, sign bits for the bit plane ("SIGN_BIT"), and ADDRESS_COUNTER as parameters. The encoding procedure encodes the bits of the bit plane up to the position indicated by ADDRESS_COUNTER.
and, when necessary, the corresponding sign bit(s). In one embodiment, an array of values (represented by individual bits or multiple bits) indicate whether it is necessary to encode the sign bit(s).

[0025] At block 225, the VLC module 180 increments ADDRESS_COUNTER. At block 227, the VLC module 180 determines if it has reached the end of the bit plane. For this illustration, a significant bit is a non-zero value bit. Alternative embodiments of the invention may encode runs of non-zero value bits and track positions of zero value bits as significant bits. In another embodiment, runs of certain value represented by multiple bits (e.g., a byte) may be encoded while positions of other values represented by multiple bits are tracked. If the VLC module 180 determines that it has not reached the last position with a significant bit in the bit plane, then at block 229 the VLC module 180 increments LOOP_COUNTER. From block 229 control flows to block 217. If at block 227 the VLC module 180 determines that it has reached the final position in the bit plane with a significant bit, then at block 231 the VLC module 180 sets RUN to the result of the following operation:

\[ \text{POSITION [LOOP_COUNTER]} - \text{POSITION [LOOP_COUNTER - 1]} \]

[0026] At block 233, the VLC module 180 increments ADDRESS_COUNTER with RUN. At block 235, the VLC module 180 calls the encoding procedure and passes RUN, SIGN_BIT[ADDRESS_COUNTER], and an end of plane (EOP) bit as parameters. The encoding procedure encodes the bits of the bit plane after the last significant bit up to the EOP bit and, when necessary, the corresponding sign bit(s).

[0027] The described invention enables variable length coding without non-loop related conditional branches. Removing non-loop related conditional branches dramatically reduces the number of mispredicted branches that occur while performing variable length coding. Reducing mispredicted branches in variable length coding increases the speed of
variable length coding and increases efficiency of an associated processor, such as processor 110.

[0028] Figure 3 is a flow chart for bit plane extraction according to another embodiment of the invention. At block 301, the VLC module 180 initializes POSITION[0], BIT_COUNTER and LOOP_COUNTER. The VLC module 180 initializes POSITION[0] to a boundary value (e.g., -1). The VLC module 180 initializes BIT_COUNTER to 1 and LOOP_COUNTER to 0 as in block 201 of Figure 2A. At block 303, the VLC module 180 sets a flag to the result of a logic operation on BIT_PLANE [LOOP_COUNTER]. An example logic operation would be the following:

(BIT_PLANE [LOOP_COUNTER] & BIT_MASK) || 0). Block 303 is similar to block 205 of Figure 2A.

[0029] At block 305, the VLC module 180 sets POSITION[BIT_COUNTER] to LOOP_COUNTER if the flag has been set to 1. Block 305 may be implemented with a predicate instruction or a conditional move instruction. At block 307, the VLC module 180 increments BIT_COUNTER with the flag. At block 309, the VLC module 180 determines if it has reached the end of the bit plane. If the VLC module 180 determines that it has not reached the end of the bit plane, then at block 311 the VLC module 180 increments LOOP_COUNTER. Control flows from block 311 to block 303. If at block 309 the VLC module 180 determines that it has reached the end of the bit plane, then at block 313 the VLC module 180 sets POSITION[BIT_COUNTER] to LOOP_COUNTER. Control flows from block 313 to block 215 of Figure 2B.

[0030] Figures 2A – 2B and Figure 3 illustrate embodiments of the invention at a source code level. Alternative embodiments of the invention may be implemented at a lower level.
[0031] Figure 4 is a flow chart for another embodiment of variable length coding according to one embodiment of the invention. At block 403, the VLC module 180 loads a selected bit plane into a buffer. At block 405, the VLC module 180 finds the position of the most significant bit (MSB) of the buffer and sets a variable ADDRESS_COUNTER to the location of the MSB. At block 407, the VLC module 180 sets RUN to ADDRESS_COUNTER. At block 409, the VLC module 180 encodes bits up to the position indicated by RUN in the buffer and their corresponding sign bits, which may be stored in another buffer. At block 413, the VLC module 180 shifts the buffer by RUN+1 bits. At block 415, the VLC module 180 finds the position of a new MSB in the buffer and sets RUN to the position. At block 417, the VLC module 180 increments ADDRESS_COUNTER with RUN+1. At block 419, the VLC module 180 determines if it has reached the end of the bit plane. If the VLC module 180 determines that it has not reached the end of the bit plane, then control flows to block 409. If at block 419 the VLC module 180 determines that it has reached the end of the bit plane, then at block 421 the VLC module is done performing variable length coding of the selected bit plane.

[0032] Figure 5 is a block diagram illustrating exemplary extraction of a selected bit plane into a buffer as described in block 403 of Figure 4 according to one embodiment of the invention. In Figure 5, masking bits are stored in a buffer 501. The masking bits in the buffer 501 are set in accordance with a certain bit plane. Data bits from a data stream ("source data") are stored in a buffer 503. In the example illustrated in Figure 5, the first bit in each byte of the source data corresponds to a particular bit plane. If this particular bit plane is a desired bit plane, then the first bit of each byte in the buffer 501 is set to 1. Each masking bit in the buffer 501 selects a corresponding data bit in the buffer 503. The selected bits are stored in a destination buffer 505 as the 4 rightmost bits of the destination buffer 505. All remaining locations of the destination buffer 505 that have not been set are
then set to 0. The buffers 501, 503, and 505 may be implemented as one or a combination of RAM, optical storage, magnetic storage, a register, etc.

[0033] In one embodiment, a bit plane is extracted from a source data in multiple passes. The masking bits of the buffer 501 are used to select corresponding bits from the source data in the multiple passes. Various embodiments of the invention may extract bits with shift and logic operations differently. In one embodiment, the destination buffer is shifted and the selected bits of the current pass are extracted into the right most positions of the destination buffer. In another embodiment, the selected bits of a current pass are concatenated with the previously extracted bits into another buffer using shift and logic operations.

[0034] The process repeats itself until the destination buffer has been filled with bits from the desired bit plane, or an alternative limit has been reached. For example, one embodiment may store two desired bit planes in a buffer. In such an embodiment, the bit extraction process would repeat itself until it as filled the segment of the buffer allocated for the first desired bit plane. In another embodiment, the multiple bit planes may be extracted from a data stream. Multiple masking bits may be set. The data bits corresponding to a first masking bit may be extracted into a first destination buffer while the data bits corresponding to a second masking bit may be extracted into a second destination buffer.

[0035] In addition, Figure 5 illustrates consistent positions for masking bits corresponding to a selected bit plane. In an alternative embodiment, the bits for a given bit plane may not be located at the same position throughout a data stream. Hence, locations of masking bits may vary with the locations of bits of a selected bit plane.

[0036] Figure 6 is a flow chart for locating the MSB with shift and compare operations according to one embodiment of the invention. At block 601, a bit is shifted out of a
buffer hosting a bit plane. At block 603, a variable indicating the MSB's position ("MSB_POSITION") is incremented. At block 605, the shifted bit is compared. A zero bit causes control to flow to block 601. A one bit causes the MSB_POSITION to be returned at block 607.

[0037] Figure 7 is an exemplary diagram illustrating components for locating the MSB of a register with daisy-chaining according to one embodiment of the invention. In Figure 7, a bit plane is stored in a temp register 701. The temp register 701 is coupled with set of inverters 703. The set of inverters are coupled with a set of logic 705. The set of logic 705 is coupled with an adder 707. The bit plane that is stored in the temp register 701 is driven into the set of inverters 703. The set of inverters 703 flip all of the bits in the bit plane. The flipped bits of the bit plane then flow into the set of logic 705. The set of logic 705 in Figure 7 cause all of the bits after the leftmost one bit to be set to zero (0). The modified bits of the bit plane flow from the set of logic 705 to the adder 709. The adder 707 adds the modified bit plane bits in an adder tree and generates a result, which is the MSB_POSITION.

The generated result indicates the position of the MSB with 0 corresponding to the leftmost position. While Figure 7 illustrates inverters and AND gates, it should be understood that such operations are exemplary (e.g., alternative embodiments may perform different logical operations and implement different logic for daisy chaining).

[0038] Figure 8 is a flowchart for locating the MSB in a register with one or more look-up tables according to one embodiment of the invention. At block 841, a look-up table is loaded. At block 843, a word (X-bits) of N bits of data into an x-bit register is loaded. At block 845, a value corresponding to the word in the loaded look-up table is looked-up. At block 847, the next look-up table is loaded in accordance with the value. In one embodiment, while a corresponding zero value in the look-up table causes the MSB-
finding module 405/415 to load the same look-up table, the look-up table may indicate a
handle for a different look-up table when the corresponding value is not zero. At block
849, the next word of the selected bit plane is loaded. The operations of block 845, block
847, and block 849 are repeated until the selected bit plane has been processed (i.e.,
operations are repeated N/x-1 times). After (N/X-1) iterations, block 853 returns the
MSB_POSITIONS based on the sum of the values from the table lookups.

[0039] Locating a MSB in a buffer with look-up tables may be implemented in a
variety of ways. Different implementations of locating the MSB with look-up tables relate
to desired results with factors such as memory limitations, speed, etc. Various
implementations of locating the MSB in a buffer may differ in relation to which words are
loaded first, the most significant or the least significant, the size of the data, and/or the size
of the words.

[0040] While the flow diagrams in the Figures show a particular order of operations
performed by certain embodiments of the invention, it should be understood that such
order is exemplary (e.g., alternative embodiments may perform certain of the operations in
a different order, combine certain of the operations, perform certain of the operations in
parallel, etc.). For example, while the flow diagrams in the Figures show finding MSB of a
particular buffer and encoding the symbol one after another, an alternative embodiment is
to find multiple MSBs and encode the symbols simultaneously.

[0041] Various applications that use variable length coding may be optimized with the
described invention. Variable length coding for image compression, audio compression,
and streaming video are a few examples of applications that may benefit from the
described invention. Reducing mispredicted branches from a processor’s pipeline also
increases the overall efficiency of the processor since cycles of the processor may be used
for other tasks.
While the invention has been described in terms of several embodiments, those skilled in the art will recognize that the invention is not limited to the embodiments described. The method and apparatus of the invention may be practiced with modification and alteration within the spirit and scope of the appended claims. The description is thus to be regarded as illustrative instead of limiting on the invention.
CLAIMS

What is claimed is:

1. A method comprising:
   receiving a set of data having a set of significant values;
   identifying a set of positions of the set of significant values within the set of data;
   for each of the set of positions, encoding a run of non-significant values preceding each of the set of positions.

2. The method of claim 1 further comprising encoding a second run of non-significant values with an ending symbol.

3. The method of claim 1 further comprising encoding a set of signs corresponding to the set of significant values.

4. The method of claim 1 wherein the set of positions are identified without non-loop related conditional branches.

5. The method of claim 4 wherein identifying the set of positions without non-loop related conditional branches comprises:
   performing a logic operation of each value in the set of data; and
   tracking the set of positions with a result of the logic operation.

6. The method of claim 4 wherein the identifying the set of positions without non-loop related conditional branches comprises:
performing a logic operation of each value in the set of data;
setting a flag to a result of the logic operation; and
tracking the set of positions using a conditional instruction with the flag.

7. A method comprising:
  loading a set of data into a buffer, the set of data having a set of one or more
    significant bits;
  identifying a position of a most significant bit (MSB) in the buffer without non-
    loop related conditional branching;
  encoding a run of non-significant bits preceding the position; and
  shifting the run of non-significant bits and the first MSB out of the buffer.

8. The method of claim 7 further comprising encoding a second run of significant
    values with an ending symbol.

9. The method of claim 7 further comprising encoding a sign bit corresponding to the
    MSB at the position in a second buffer.

10. The method of claim 7 wherein loading a set of data into the buffer comprises:
    loading a set of source data into a first buffer, the set of source data including the
        set of data;
    loading a mask for a bit-plane into a second buffer;
    selecting the set of data from the set of source data with the mask; and
    extracting the set of data into the buffer.
11. The method of claim 7 wherein identifying the MSB comprises:
   shifting each bit of the set of data out of the buffer;
   comparing each shifted bit with a compare bit to determine if each bit is one of the
   set of significant bits;
   tracking each of the set of significant bits when detected with the comparing; and
   incrementing a counter for each shifted bit.

12. The method of claim 7 wherein the MSB is identified with one or more look-up
    tables.

13. The method of claim 7 further comprising
    identifying multiple positions of MSBs in the buffer without non-loop related
    conditional branching; and
    encoding runs of non-significant bits preceding the positions simultaneously.

14. An apparatus comprising:
    a buffer to host a set of data having a set of one or more significant bits; and
    a variable length coding (VLC) module coupled with the memory, the VLC
    module to identify the set of significant bits' positions and to encode each
    run of non-significant bits with respect to the set of significant bits' positions.

15. The apparatus of claim 14 further comprising the VLC module to encode a second
    run of non-significant bits with an ending symbol.
16. The apparatus of claim 14 further comprising the VLC module to encode a set of sign bits corresponding to the set of significant bits.

17. The apparatus of claim 14 further comprising a set of one or more look-up tables for the VLC module to identify the set of significant bits' positions.

18. The apparatus of claim 14 wherein the VLC module to identify the set of significant bits' positions comprises

to load the set of data into a memory,

to flip all bits of the set of data,

to set all bits after the significant bit to non-significant bits, and

to add the significant bits in the memory in an adder tree.

19. A machine-readable medium that provides instructions, which when executed by a set of processors of one or more processors, cause said set of processors to perform operations comprising:

receiving a set of data having a set of significant values;

identifying a set of positions of the set of significant values within the set of data;

for each of the set of positions, encoding a run of non-significant values preceding each of the set of positions.

20. The machine-readable medium of claim 19 further comprising encoding a second run of non-significant values with an ending symbol.
21. The machine-readable medium of claim 19 further comprising encoding a set of signs corresponding to the set of significant values.

22. The machine-readable medium of claim 19 wherein the set of positions are identified without non-loop related conditional branches.

23. The machine-readable medium of claim 22 wherein identifying the set of positions without non-loop related conditional branches comprises:
   performing a logic operation of each value in the set of data; and
   tracking the set of positions with a result of the logic operation.

24. The machine-readable medium of claim 22 wherein the identifying the set of positions without non-loop related conditional branches comprises:
   performing a logic operation of each value in the set of data;
   setting a flag to a result of the logic operation; and
   tracking the set of positions using a conditional instruction with the flag.

25. A machine-readable medium that provides instructions, which when executed by a set of processors of one or more processors, cause said set of processors to perform operations comprising:
   loading a set of data into a buffer, the set of data having a set of one or more significant bits;
   identifying a position of a most significant bit (MSB) in the buffer without non-loop related conditional branching;
encoding a run of non-significant bits preceding the position; and
shifting the run of non-significant bits and the first MSB out of the buffer.

26. The machine-readable medium of claim 25 further comprising encoding a second run of significant values with an ending symbol.

27. The machine-readable medium of claim 25 further comprising encoding a sign bit corresponding to the MSB at the position in a second buffer.

28. The machine readable medium of claim 25 wherein loading a set of data into the buffer comprises:

loading a set of source data into a first buffer, the set of source data including the set of data;

loading a mask for a bit-plane into a second buffer;

selecting the set of data from the set of source data with the mask; and

extracting the set of data into the buffer.

29. The machine-readable medium of claim 25 wherein identifying the MSB comprises:

loading the set of data into a memory;

flipping all bits of the set of data;

setting all bits after the first zero bit to zero; and

adding the significant bits in the second memory in an adder tree, the significant bits being one bits.
30. The machine-readable medium of claim 25 wherein the MSB is identified with one or more look-up tables.
FIG. 1
INITIALIZE POSITION[0] = BOUNDARY VALUE, BIT_COUNTER = 1, AND LOOP_COUNTER = 0

SET POSITION[BIT_COUNTER] = LOOP_COUNTER

SET FLAG TO RESULT OF LOGIC OPERATION ON BIT_PLANE[LOOP_COUNTER]

INCREMENT BIT COUNTER WITH FLAG VALUE

END OF PLANE?

INCREMENT LOOP_COUNTER

POSITION[BIT_COUNTER] = LOOP_COUNTER

FIG. 2A
A

LOOP_COUNTER = 1

SET RUN = POSITION[LOOP_COUNTER] - POSITION[LOOP_COUNTER - 1] - 1

INCREMENT ADDRESS_COUNTER WITH RUN

INCREMENT ADDRESS_COUNTER

ENCODER RUN AND SIGN_BIT[ADDRESS_COUNTER]

INCREMENT ADDRESS_COUNTER

END OF PLANE?

SET RUN = POSITION[LOOP_COUNTER] - POSITION[LOOP_COUNTER - 1] - 1

INCREMENT ADDRESS_COUNTER WITH RUN

ENCODE RUN, SIGN_BIT [ADDRESS_COUNTER], AND EOP

FIG. 2B
INITIALIZE POSITION[0] = BOUNDARY VALUE, BIT_COUNTER = 1, AND LOOP_COUNTER = 0

SET FLAG TO RESULT OF LOGIC OPERATION ON BIT_PLANE[LOOP_COUNTER]

SET POSITION[BIT_COUNTER] = LOOP_COUNTER IF FLAG == 1

INCREMENT BIT_COUNTER WITH FLAG VALUE

INCREMENT LOOP_COUNTER

END OF PLANE?

POSITION[BIT_COUNTER] = LOOP_COUNTER

FIG. 3
LOAD DATA FROM SELECTED BIT PLANE INTO A BUFFER

FIND MSB OF BUFFER AND SET ADDRESS_COUNTER TO LOCATION OF MSB

SET RUN TO ADDRESS_COUNTER

ENCODE RUN AND SIGN_BIT[ADDRESS_COUNTER]

SHIFT BUFFER BY RUN+1 BITS

FIND NEW MSB OF BUFFER AND SET RUN TO LOCATION OF MSB

INCREMENT ADDRESS_COUNTER WITH RUN+1

END OF PLANE?

FIG. 4
FIRST PASS

BUFFER 501
00000001 00000001 00000001 00000001

BUFFER 503
10100011 11010011 00101000 01101111

DESTINATION BUFFER 505
00000000 00000000 00000000 00001101

FIG. 5
LOAD LOOK-UP TABLE

LOAD THE FIRST WORD (X-BIT) OUT OF THE DATA (N-BIT) INTO X-BIT REGISTER

LOOK-UP VALUE OF WORD IN LOADED LOOK-UP TABLE

LOAD NEXT TABLE BASED ON VALUE

LOAD NEXT WORD

REPEAT (N/X-1)-TIMES

AFTER (N/X-1) REPEATS

RETURN MSB_POSITION

FIG. 8
INTERNATIONAL SEARCH REPORT

A. CLASSIFICATION OF SUBJECT MATTER

IPC 7 H03M7/46 H03M5/14 G11B20/14 H04N7/26 H04N1/41
G06F9/38 H03M7/40

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)

IPC 7 H03M G11B H04N G06F

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 practical, search terms used)

EPO-Internal, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>WO 00 59116 A (MICROSOFT CORP) 5 October 2000 (2000-10-05) the whole document page 3, line 24 -page 4, line 12 page 10, line 9 -page 11, line 27 page 17 -page 26, line 20 figures 2,5,6</td>
<td>1-30</td>
</tr>
<tr>
<td>A</td>
<td>EP 0 724 215 A (SUN MICROSYSTEMS INC) 31 July 1996 (1996-07-31) column 2, line 4 -column 3, line 12 column 6, line 5 -column 7, line 4</td>
<td>1-30</td>
</tr>
</tbody>
</table>

X Further documents are listed in the continuation of box C.

Patent family members are listed in annex.

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

Date of the actual completion of the international search 23 June 2003

Date of mailing of the international search report 01/07/2003

Name and mailing address of the ISA European Patent Office, P.B. 5816 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2046, Fax. 31 651 80 774, P.O. Box (+31-70) 3400 GA Rijswijk

Authorized officer Bareil-Faucheux, C

Form PCT/ISA/210 (second ed. 2002) (July 1999)
<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patent document cited in search report</td>
<td>Publication date</td>
<td>Patent family member(s)</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AU 3772300 A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AU 3916900 A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AU 3922200 A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EP 1166565 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EP 1188244 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP 2002540711 T</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP 2002540740 T</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 0059231 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 0059232 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 0059116 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP 8263288 A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 5748935 A</td>
</tr>
</tbody>
</table>

Form PCT/ISA/20 (patent family search) (July 1992)