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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 May 2002 (16.05.2002)

PCT

(10) International Publication Number
WO 02/39591 A1

(51) International Patent Classification[7]: H03M 7/30

(21) International Application Number: PCT/US00/42018

(22) International Filing Date: 9 November 2000 (09.11.2000)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: REALTIME DATA LLC [US/US]; 206 East 63rd Street, New York, NY 10021 (US).

(72) Inventor: FALLON, James, J.; 11 Wampus Close, Armonk, NY 10504 (US).

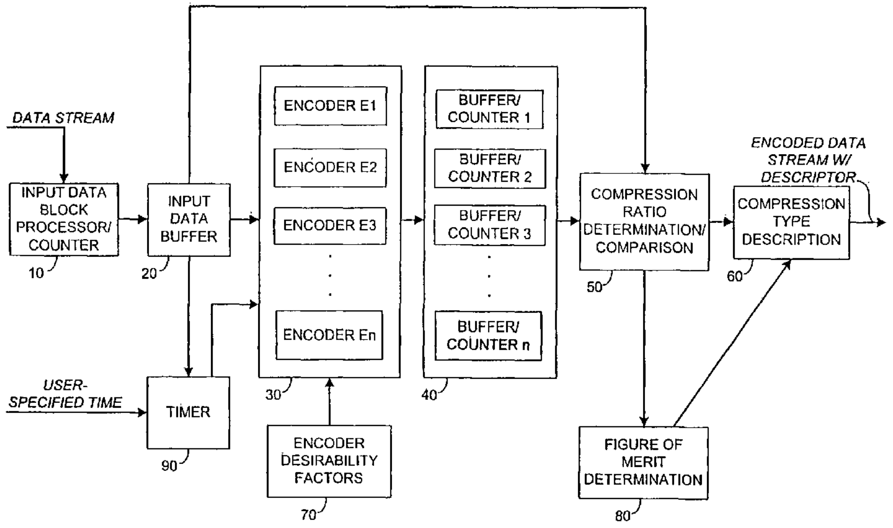
(74) Agents: DEROSA, Frank, V. et al.; F. Chau & Associates, LLP, 1900 Hempstead Turnpike, Suite 501, East Meadow, NY 11554 (US).

(81) Designated States *(national)*: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(54) Title: CONTENT INDEPENDENT DATA COMPRESSION METHOD AND SYSTEM


DATA STREAM
INPUT DATA BLOCK PROCESSOR/ COUNTER
10
INPUT DATA BUFFER
20
ENCODER E1
ENCODER E2
ENCODER E3
ENCODER En
30
BUFFER/ COUNTER 1
BUFFER/ COUNTER 2
BUFFER/ COUNTER 3
BUFFER/ COUNTER n
40
COMPRESSION RATIO DETERMINATION/ COMPARISON
50
ENCODED DATA STREAM W/ DESCRIPTOR
COMPRESSION TYPE DESCRIPTION
60
USER-SPECIFIED TIME
TIMER
90
ENCODER DESIRABILITY FACTORS
70
FIGURE OF MERIT DETERMINATION
80

(57) **Abstract:** Systems for providing content independent lossless data compression and decompression. Data block are stored in an input buffer (20) and is compressed by a plurality of encoders (30) configured to simultaneously or sequentially compress input. Ecah data block output from the encoders (30) is buffered (40) and counted. A processing unit (50) computes the compression ratio obtained by each of the encoders (30) and compares the compression ratios with a threshold or with a figure of merit, weighted from encoder desirability factor level. Encoded data block having highest compression ratio and meeting threshold level is selected. A processing unit (60) appends a compression description, to enable subsequent decompression and interpretation. If any compression ratios does not meet the threshold level, processing unit (50) will select buffered input block (20) for output and processing unit (60) will append a null descriptor. A timer (90) may be included to measure encoding time against a predefined time limit.

WO 02/39591 A1

CONTENT INDEPENDENT DATA COMPRESSION METHOD AND SYSTEM

BACKGROUND

1. Technical Field

The present invention relates generally to a data compression and decompression and, more particularly, to systems and methods for providing content independent lossless data compression and decompression.

2. Description of Related Art

Information may be represented in a variety of manners. Discrete information such as text and numbers are easily represented in digital data. This type of data representation is known as symbolic digital data. Symbolic digital data is thus an absolute representation of data such as a letter, figure, character, mark, machine code, or drawing,

Continuous information such as speech, music, audio, images and video, frequently exists in the natural world as analog information. As is well-known to those skilled in the art, recent advances in very large scale integration (VLSI) digital computer technology have enabled both discrete and analog information to be represented with digital data. Continuous information represented as digital data is often referred to as diffuse data. Diffuse digital data is thus a representation of data that is of low information density and is typically not easily recognizable to humans in its native form.

There are many advantages associated with digital data representation. For instance, digital data is more readily processed, stored, and transmitted due to its inherently high noise immunity. In addition, the inclusion of redundancy in digital data representation enables error detection and/or correction. Error detection and/or correction capabilities are dependent upon the amount and type of data redundancy, available error detection and correction processing, and extent of data corruption.

One outcome of digital data representation is the continuing need for increased capacity in data processing, storage, and transmittal. This is especially true for diffuse data where increases in fidelity and resolution create exponentially greater quantities of data. Data compression is widely used to reduce the amount of data required to process, transmit, or store a given quantity of information. In general, there are two types of data compression techniques that may be utilized either separately or jointly to encode/decode data: lossless and lossy data compression.

Lossy data compression techniques provide for an inexact representation of the original uncompressed data such that the decoded (or reconstructed) data differs from the original unencoded/uncompressed data. Lossy data compression is also known as irreversible or noisy compression. Entropy is defined as the quantity of information in a given set of data. Thus, one obvious advantage of lossy data compression is that the compression ratios can be larger than the entropy limit, all at the expense of information content. Many lossy data compression techniques seek to exploit various traits within the human senses to eliminate otherwise imperceptible data. For example, lossy data compression of visual imagery might seek to delete information content in excess of the display resolution or contrast ratio.

On the other hand, lossless data compression techniques provide an exact representation of the original uncompressed data. Simply stated, the decoded (or reconstructed) data is identical to the original unencoded/uncompressed data. Lossless data compression is also known as reversible or noiseless compression. Thus, lossless data compression has, as its current limit, a minimum representation defined by the entropy of a given data set.

There are various problems associated with the use of lossless compression techniques. One fundamental problem encountered with most lossless data compression techniques are their content sensitive behavior. This is often referred to as data dependency. Data dependency implies that the compression ratio achieved is highly contingent upon the content of the data being compressed. For example, database files often have large unused fields and high data redundancies, offering the opportunity to losslessly compress data at ratios of 5 to 1 or more. In contrast, concise software programs have little to no data redundancy and, typically, will not losslessly compress better than 2 to 1.

Another problem with lossless compression is that there are significant variations in the compression ratio obtained when using a single lossless data compression technique for data streams having different data content and data size. This process is known as natural variation.

A further problem is that negative compression may occur when certain data compression techniques act upon many types of highly compressed data. Highly compressed data appears random and many data compression techniques will substantially expand, not compress this type of data.

For a given application, there are many factors which govern the applicability of various data compression techniques. These factors include compression ratio, encoding and decoding processing requirements, encoding and decoding time delays, compatibility with existing standards, and implementation complexity and cost, along with the adaptability and robustness to variations in input data. A direct relationship exists in the current art between the compression ratio and the amount and complexity of processing required. One of the limiting factors in most existing prior art lossless data compression techniques is the rate at which the encoding and decoding processes are performed. Hardware and software implementation tradeoffs are often dictated by encoder and decoder complexity along with cost.

Another problem associated with lossless compression methods is determining the optimal compression technique for a given set of input data and intended application. To combat this problem, there are many conventional *content dependent* techniques which may be utilized. For instance, file type descriptors are typically appended to file names to describe the application programs that normally act upon the data contained within the file. In this manner data types, data structures, and formats within a given file may be ascertained. Fundamental problems with this *content dependent* technique are:

(1) The extremely large number of application programs, some of which do not possess published or documented file formats, data structures, or data type descriptors;

(2) The ability for any data compression supplier or consortium to acquire, store, and access the vast amounts of data required to identify known file descriptors and associated data types, data structures, and formats; and

(3) The rate at which new application programs are developed and the need to update file format data descriptions accordingly.

An alternative technique that approaches the problem of selecting an appropriate lossless data compression technique is disclosed in U.S. Patent No. 5,467,087 to Chu entitled "High Speed Lossless Data Compression System" ("Chu"). FIG. **1** illustrates an embodiment of this data compression and decompression technique. Data compression **1** comprises two phases, a data pre-compression phase **2** and a data compression phase **3**. Data decompression **4** of a compressed input data stream is also comprised of two phases, a data type retrieval phase **5** and a data decompression phase **6**. During the data compression process **1**, the data pre-compressor **2** accepts an uncompressed data stream, identifies the data type of the input stream, and generates a data type identification signal. The data compressor **3** selects a data compression method from a preselected set of methods to compress the input data stream, with the intention of producing the best available compression ratio for that particular data type.

There are several problems associated with the Chu method. One such problem is the need to unambiguously identify various data types. While these might include such common data types as ASCII, binary, or unicode, there, in fact, exists a broad universe of data types that fall outside the three most common data types. Examples of these alternate data types include: signed and unsigned integers of various lengths, differing types and precision of floating point numbers, pointers, other forms of character text, and a multitude of user defined data types. Additionally, data types may be interspersed or partially compressed, making data type recognition difficult and/or impractical. Another problem is that given a known data type, or mix of data types within a specific set or subset of input data, it may be difficult and/or impractical to predict which data encoding technique yields the highest compression ratio.

SUMMARY OF THE INVENTION

The present invention is directed to systems and methods for providing content independent lossless data compression and decompression. In one aspect of the present invention, a method for compressing data comprises the steps of:

receiving an input data stream comprising a plurality of disparate data types;

compressing the input data stream using each of a plurality of different encoders; and

generating an encoded data stream by selectively combining compressed data blocks output from each of the encoders based on compression ratios obtained by the encoders.

In another aspect of the invention, a method for providing content independent data compression comprises the steps of:

receiving as input a block of data from a stream of data;

encoding said input data block with a plurality of encoders to provide a plurality of encoded data blocks;

determining a compression ratio obtained for each of said encoders;

comparing each of said determined compression ratios with an *a priori* user specified compression threshold;

selecting for output said input data block and appending a null compression descriptor to said input data block, if all of said encoder compression ratios fall below said *a priori* specified compression threshold; and

selecting for output said encoded data block having the highest compression ratio and appending a corresponding compression type descriptor to said selected encoded data block, if at least one of said compression ratios meet said a priori specified compression threshold.

In another aspect of the present invention, a timer is preferably added to measure the time elapsed during the encoding process against an *a priori*-specified time limit. When the time limit expires, only the data output from those encoders that have completed the present encoding cycle are compared to determine the encoded data with the highest compression ratio. The time limit ensures that the real-time or pseudo real-time nature of the data encoding is preserved.

In another aspect of the present invention, the results from each encoder are buffered to allow additional encoders to be sequentially applied to the output of the previous encoder, yielding a more optimal lossless data compression ratio.

In another aspect of the present invention, a method for providing content independent lossless data decompression includes the steps of receiving as input a block of data from a stream of data, extracting an encoding type descriptor from the input data block, decoding the input data block with one or more of a plurality of available decoders in accordance with the extracted encoding type descriptor, and outputting the decoded data block. An input data block having a null descriptor type extracted therefrom is output without being decoded.

Advantageously, the present invention employs a plurality of encoders applying a plurality of compression techniques on an input data stream so as to achieve maximum compression in accordance with the real-time or pseudo real-time data rate constraint. Thus, the output bit rate is not fixed and the amount, if any, of permissible data quality degradation is not adaptable, but is user or data specified.

These and other aspects, features and advantages of the present invention will become apparent from the following detailed description of preferred embodiments, which is to be read in connection with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block/flow diagram of a content dependent high-speed lossless data compression and decompression system/method according to the prior art;

FIG. 2 is a block diagram of a content independent data compression system according to one embodiment of the present invention;

FIGs. 3a and 3b comprise a flow diagram of a data compression method according to one aspect of the present invention which illustrates the operation of the data compression system of FIG. 2;

FIG. 4 is a block diagram of a content independent data compression system according to another embodiment of the present invention having an enhanced metric for selecting an optimal encoding technique;

FIGs. 5a and 5b comprise a flow diagram of a data compression method according to another aspect of the present invention which illustrates the operation of the data compression system of FIG. 4;

FIG. 6 is a block diagram of a content independent data compression system according to another embodiment of the present invention having an *a priori* specified timer that provides real-time or pseudo real-time of output data;

FIGs. 7a and 7b comprise a flow diagram of a data compression method according to another aspect of the present invention which illustrates the operation of the data compression system of FIG. 6;

FIG. 8 is a block diagram of a content independent data compression system according to another embodiment having an *a priori* specified timer that provides real-time or pseudo real-time of output data and an enhanced metric for selecting an optimal encoding technique;

FIG. 9 is a block diagram of a content independent data compression system according to another embodiment of the present invention having an encoding architecture comprising a plurality of sets of serially-cascaded encoders;

FIGs. 10a and 10b comprise a flow diagram of a data compression method according to another aspect of the present invention which illustrates the operation of the data compression system of FIG. 9;

FIG. 11 is block diagram of a content independent data decompression system according to one embodiment of the present invention; and

FIG. 12 is a flow diagram of a data decompression method according to one aspect of the present invention which illustrates the operation of the data compression system of FIG. 11.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

The present invention is directed to systems and methods for providing content independent lossless data compression and decompression. In the following description, it is to be understood that system elements having equivalent or similar functionality are designated with the same reference numerals in the Figures. It is to be further understood that the present invention may be implemented in various forms of hardware, software, firmware, or a combination thereof. In particular, the system modules described herein are preferably implemented in software as an application program which is loaded into and executed by a general purpose computer having any suitable and preferred microprocessor architecture. Preferably, the present invention is implemented on a computer platform including hardware such as one or more central processing units (CPU), a random access memory (RAM), and input/output (I/O) interface(s). The computer platform also includes an operating system and microinstruction code. The various processes and functions described herein may be either part of the microinstruction code or application programs which are executed via the operating system. In addition, various other peripheral devices may be connected to the computer platform such as an additional data storage device and a printing device.

It is to be further understood that, because some of the constituent system components described herein are preferably implemented as software modules, the actual system connections shown in the Figures may differ depending upon the manner in which the systems are programmed. It is to be appreciated that special purpose microprocessors may be employed to implement the present invention. Given the teachings herein, one of ordinary skill in the related art will be able to contemplate these and similar implementations or configurations of the present invention.

Referring now to FIG. 2 a block diagram illustrates a content independent data compression system according to one embodiment of the present invention. The data compression system includes a counter module 10 which receives as input an uncompressed or compressed data stream. It is to be understood that the system processes the input data stream in data blocks that may range in size from individual bits through complete files or collections of multiple files. Additionally, the data block size may be fixed or variable. The counter module 10 counts the size of each input data block (i.e., the data block size is counted in bits, bytes, words, any convenient data multiple or metric, or any combination thereof).

An input data buffer 20, operatively connected to the counter module 10, may be provided for buffering the input data stream in order to output an uncompressed data stream in the event that, as discussed in further detail below, every encoder fails to achieve a level of compression that exceeds an *a priori* specified minimum compression ratio threshold. It is to be understood that the input data buffer 20 is not required for implementing the present invention.

An encoder module 30 is operatively connected to the buffer 20 and comprises a set of encoders E1, E2, E3 ... En. The encoder set E1, E2, E3 ... En may include any number "n" of those lossless encoding techniques currently well known within the art such as run length, Huffman, Lempel-Ziv Dictionary Compression, arithmetic coding, data compaction, and data null suppression. It is to be understood that the encoding techniques are selected based upon their ability to effectively encode different types of input data. It is to be appreciated that a full complement of encoders are preferably selected to provide a broad coverage of existing and future data types.

The encoder module 30 successively receives as input each of the buffered input data blocks (or unbuffered input data blocks from the counter module 10). Data compression is performed by the encoder module 30 wherein each of the encoders E1 .... En processes a given input data block and outputs a corresponding set of encoded data blocks. It is to be appreciated that the system affords a user the option to enable/disable any one or more of the encoders E1.... En prior to operation. As is understood by those skilled in the art, such feature allows the user to tailor the operation of the data compression system for specific applications. It is to be further appreciated that the encoding process may be performed either in parallel or sequentially. In particular, the encoders E1 through En of encoder module 30 may operate in parallel (i.e., simultaneously processing a given input data block by utilizing task multiplexing on a single central processor, via dedicated hardware, by executing on a plurality of processor or dedicated hardware systems, or any combination thereof). In addition, encoders E1 through En may operate sequentially on a given unbuffered or buffered input data block. This process is intended to eliminate the complexity and additional processing overhead associated with multiplexing concurrent encoding techniques on a single central processor and/or dedicated hardware, set of central processors and/or dedicated hardware, or any achievable combination. It is to be further appreciated that encoders of the identical type may be applied in parallel to enhance encoding speed. For instance, encoder E1 may comprise two parallel Huffman encoders for parallel processing of an input data block.

A buffer/counter module 40 is operatively connected to the encoding module 30 for buffering and counting the size of each of the encoded data blocks output from encoder module 30. Specifically, the buffer/counter 30 comprises a plurality of buffer/counters BC1, BC2, BC3 ....BCn, each operatively associated with a corresponding one of the encoders E1...En. A compression ratio module 50, operatively connected to the output buffer/counter 40, determines the compression ratio obtained for each of the enabled encoders E1...En by taking the ratio of the size of the input data block to the size of the output data block stored in the corresponding buffer/counters BC1 ... BCn. In addition, the compression ratio module 50 compares each compression ratio with an *a priori*-specified compression ratio threshold limit to determine if at least one of the encoded data blocks output from the enabled encoders E1...En achieves a compression that exceeds an *a priori*-specified threshold. As is understood by those skilled in the art, the threshold limit may be specified as any value inclusive of data expansion, no data compression or expansion, or any arbitrarily desired compression limit. A description module 60, operatively coupled to the compression ratio module 50, appends a corresponding compression type descriptor to each encoded data block which is selected for output so as to indicate the type of compression format of the encoded data block.

The operation of the data compression system of FIG. 2 will now be discussed in further detail with reference to the flow diagram of FIGs. 3a and 3b. A data stream comprising one or more data blocks is input into the data compression system and the first data block in the stream is received (step 300). As stated above, data compression is performed on a per data block basis. Accordingly, the first input data block in the input data stream is input into the counter module 10 which counts the size of the data block (step 302). The data block is then stored in the buffer 20 (step 304). The data block is then sent to the encoder module 30 and compressed by each (enabled) encoder E1 ... En (step 306). Upon completion of the encoding of the input data block, an encoded data block is output from each (enabled) encoder E1...En and maintained in a corresponding buffer (step 308), and the encoded data block size is counted (step 310).

Next, a compression ratio is calculated for each encoded data block by taking the ratio of the size of the input data block (as determined by the input counter 10) to the size of each encoded data block output from the enabled encoders (step 312). Each compression ratio is then compared with an *a priori*-specified compression ratio threshold (step 314). It is to be understood that the threshold limit may be specified as any value inclusive of data expansion, no data compression or expansion, or any arbitrarily desired compression limit. It is to be further understood that notwithstanding that the current limit for lossless data compression is the entropy limit (the present definition of information content) for the data, the present invention does not preclude the use of future developments in lossless data compression that may increase lossless data compression ratios beyond what is currently known within the art.

After the compression ratios are compared with the threshold, a determination is made as to whether the compression ratio of at least one of the encoded data blocks exceeds the threshold limit (step 316). If there are no encoded data blocks having a compression ratio that exceeds the compression ratio threshold limit (negative determination in step 316), then the original unencoded input data block is selected for output and a null data compression type descriptor is appended thereto (step 318). A null data compression type descriptor is defined as any recognizable data token or descriptor that indicates no data encoding has been applied to the input data block. Accordingly, the unencoded input data block with its corresponding null data compression type descriptor is then output for subsequent data processing, storage, or transmittal (step 320).

On the other hand, if one or more of the encoded data blocks possess a compression ratio greater than the compression ratio threshold limit (affirmative result in step 316), then the encoded data block having the greatest compression ratio is selected (step 322). An appropriate data compression type descriptor is then appended (step 324). A data compression type descriptor is defined as any recognizable data token or descriptor that indicates which data encoding technique has been applied to the data. It is to be understood that, since encoders of the identical type may be applied in parallel to enhance encoding speed (as discussed above), the data compression type descriptor identifies the corresponding encoding technique applied to the encoded data block, not necessarily the specific encoder. The encoded data block having the greatest compression ratio along with its corresponding data compression type descriptor is then output for subsequent data processing, storage, or transmittal (step 326).

After the encoded data block or the unencoded data input data block is output (steps 326 and 320), a determination is made as to whether the input data stream contains additional data blocks to be processed (step 328). If the input data stream includes additional data blocks (affirmative result in step 328), the next successive data block is received (step 330), its block size is counted (return to step 302) and the data compression process in repeated. This process is iterated for each data block in the input data stream. Once the final input data block is processed (negative result in step 328), data compression of the input data stream is finished (step 322).

Since a multitude of data types may be present within a given input data block, it is often difficult and/or impractical to predict the level of compression that will be achieved by a specific encoder. Consequently, by processing the input data blocks with a plurality of encoding techniques and comparing the compression results, content free data compression is advantageously achieved. It is to be appreciated that this approach is scalable through future generations of processors, dedicated hardware, and software. As processing capacity increases and costs reduce, the benefits provided by the present invention will continue to increase. It should again be noted that the present invention may employ any lossless data encoding technique.

Referring now to Fig. 4, a block diagram illustrates a content independent data compression system according to another embodiment of the present invention. The data compression system depicted in FIG. 4 is similar to the data compression system of FIG. 2 except that the embodiment of FIG. 4 includes an enhanced metric functionality for selecting an optimal encoding technique. In particular, each of the encoders E1...En in the encoder module 30 is tagged with a corresponding one of user-selected encoder desirability factors 70. Encoder desirability is defined as an *a priori* user specified factor that takes into account any number of user considerations including, but not limited to, compatibility of the encoded data with existing standards, data error robustness, or any other aggregation of factors that the user wishes to consider for a particular application. Each encoded data block output from the encoder module 30 has a corresponding desirability factor appended thereto. A figure of merit module 80, operatively coupled to the compression ratio module 50 and the descriptor module 60, is provided for calculating a figure of merit for each of the encoded data blocks which possess a compression ratio greater than the compression ratio threshold limit. The figure of merit for each encoded data block is comprised of a weighted average of the *a priori* user specified threshold and the corresponding encoder desirability factor. As discussed below in further detail with reference to FIGs. 5a and 5b, the figure of merit substitutes the *a priori* user compression threshold limit for selecting and outputting encoded data blocks.

The operation of the data compression system of Fig. 4 will now be discussed in further detail with reference to the flow diagram of FIGs. 5a and 5b. A data stream comprising one or more data blocks is input into the data compression system and the first data block in the stream is received (step 500). The size of the first data block is then determined by the counter module 10 (step 502). The data block is then stored in the buffer 20 (step 504). The data block is then sent to the encoder module 30 and compressed by each (enabled) encoder in the encoder set E1 ... En (step 506). Each encoded data block processed in the encoder module 30 is tagged with an encoder desirability factor which corresponds the particular encoding technique applied to the encoded data block (step 508). Upon completion of the encoding of the input data block, an encoded data block with its corresponding desirability factor is output from each (enabled) encoder E1...En and maintained in a corresponding buffer (step 510), and the encoded data block size is counted (step 512).

Next, a compression ratio obtained by each enabled encoder is calculated by taking the ratio of the size of the input data block (as determined by the input counter 10) to the size of the encoded data block output from each enabled encoder (step 514). Each compression ratio is then compared with an *a priori*-specified compression ratio threshold (step 516). A determination is made as to whether the compression ratio of at least one of the encoded data blocks exceeds the threshold limit (step 518). If there are no encoded data blocks having a compression ratio that exceeds the compression ratio threshold limit (negative determination in step 518), then the original unencoded input data block is selected for output and a null data compression type descriptor (as discussed above) is appended thereto (step 520). Accordingly, the original unencoded input data block with its corresponding null data compression type descriptor is then output for subsequent data processing, storage, or transmittal (step 522).

On the other hand, if one or more of the encoded data blocks possess a compression ratio greater than the compression ratio threshold limit (affirmative result in step 518), then a figure of merit is calculated for each encoded data block having a compression ratio which exceeds the compression ratio threshold limit (step 524). Again, the figure of merit for a given encoded data block is comprised of a weighted average of the *a priori* user specified threshold and the corresponding encoder desirability factor associated with the encoded data block. Next, the encoded data block having the greatest figure of merit is selected for output (step 526). An appropriate data compression type descriptor is then appended (step 528) to indicate the data encoding technique applied to the encoded data block. The encoded data block (which has the greatest figure of merit) along with its corresponding data compression type descriptor is then output for subsequent data processing, storage, or transmittal (step 530).

After the encoded data block or the unencoded input data block is output (steps 530 and 522), a determination is made as to whether the input data stream contains additional data blocks to be processed (step 532). If the input data stream includes additional data blocks (affirmative result in step 532), then the next successive data block is received (step 534), its block size is counted (return to step 502) and the data compression process is iterated for each successive data block in the input data stream. Once the final input data block is processed (negative result in step 532), data compression of the input data stream is finished (step 536).

Referring now to FIG. 6, a block diagram illustrates a data compression system according to another embodiment of the present invention. The data compression system depicted in FIG. 6 is similar to the data compression system discussed in detail above with reference to FIG. 2 except that the embodiment of FIG. 6 includes an *a priori* specified timer that provides real-time or pseudo real-time output data. In particular, an interval timer 90, operatively coupled to the encoder module 30, is preloaded with a user specified time value.

The role of the interval timer (as will be explained in greater detail below with reference to FIGs. 7a and 7b) is to limit the processing time for each input data block processed by the encoder module 30 so as to ensure that the real-time, pseudo real-time, or other time critical nature of the data compression processes is preserved.

The operation of the data compression system of Fig. 6 will now be discussed in further detail with reference to the flow diagram of FIGs. 7a and 7b. A data stream comprising one or more data blocks is input into the data compression system and the first data block in the data stream is received (step 700), and its size is determined by the counter module 10 (step 702). The data block is then stored in buffer 20 (step 704).

Next, concurrent with the completion of the receipt and counting of the first data block, the interval timer 90 is initialized (step 706) and starts counting towards a user-specified time limit. The input data block is then sent to the encoder module 30 wherein data compression of the data block by each (enabled) encoder E1 ... En commences (step 708). Next, a determination is made as to whether the user specified time expires before the completion of the encoding process (steps 710 and 712). If the encoding process is completed before or at the expiration of the timer, i.e., each encoder (E1 through En) completes its respective encoding process (negative result in step 710 and affirmative result in step 712), then an encoded data block is output from each (enabled) encoder E1...En and maintained in a corresponding buffer (step 714).

On the other hand, if the timer expires (affirmative result in 710), the encoding process is halted (step 716). Then, encoded data blocks from only those enabled encoders E1...En that have completed the encoding process are selected and maintained in buffers (step 718). It is to be appreciated that it is not necessary (or in some cases desirable) that some or all of the encoders complete the encoding process before the interval timer expires. Specifically, due to encoder data dependency and natural variation, it is possible that certain encoders may not operate quickly enough and, therefore, do not comply with the timing constraints of the end use. Accordingly, the time limit ensures that the real-time or pseudo real-time nature of the data encoding is preserved.

After the encoded data blocks are buffered (step 714 or 718), the size of each encoded data block is counted (step 720). Next, a compression ratio is calculated for each encoded data block by taking the ratio of the size of the input data block (as determined by the input counter 10) to the size of the encoded data block output from each enabled encoder (step 722). Each compression ratio is then compared with an *a priori*-specified compression ratio threshold (step 724). A determination is made as to whether the compression ratio of at least one of the encoded data blocks exceeds the threshold limit (step 726). If there are no encoded data blocks having a compression ratio that exceeds the compression ratio threshold limit (negative determination in step 726), then the original unencoded input data block is selected for output and a null data compression type descriptor is appended thereto (step 728). The original unencoded input data block with its corresponding null data compression type descriptor is then output for subsequent data processing, storage, or transmittal (step 730).

On the other hand, if one or more of the encoded data blocks possess a compression ratio greater than the compression ratio threshold limit (affirmative result in step 726), then the encoded data block having the greatest compression ratio is selected (step 732). An appropriate data compression type descriptor is then appended (step 734). The encoded data block having the greatest compression ratio along with its corresponding data compression type descriptor is then output for subsequent data processing, storage, or transmittal (step 736).

After the encoded data block or the unencoded input data block is output (steps 730 or 736), a determination is made as to whether the input data stream contains additional data blocks to be processed (step 738). If the input data stream includes additional data blocks (affirmative result in step 738), the next successive data block is received (step 740), its block size is counted (return to step 702) and the data compression process in repeated. This process is iterated for each data block in the input data stream, with each data block being processed within the user-specified time limit as discussed above. Once the final input data block is processed (negative result in step 738), data compression of the input data stream is complete (step 742).

Referring now to FIG. 8, a block diagram illustrates a content independent data compression system according to another embodiment of the present system. The data compression system of FIG. 8 incorporates all of the features discussed above in connection with the system embodiments of FIGs. 2, 4, and 6. For example, the system of FIG. 8 incorporates both the *a priori* specified timer for providing real-time or pseudo real-time of output data, as well as the enhanced metric for selecting an optimal encoding technique.

Based on the foregoing discussion, the operation of the system of FIG. 8 is understood by those skilled in the art.

Referring now to FIG. 9, a block diagram illustrates a data compression system according to a preferred embodiment of the present invention. The system of FIG. 9 contains many of the features of the previous embodiments discussed above. However, this embodiment advantageously includes a cascaded encoder module 30c having an encoding architecture comprising a plurality of sets of serially-cascaded encoders Em,n, where "m" refers to the encoding path (i.e., the encoder set) and where "n" refers to the number of encoders in the respective path. It is to be understood that each set of serially-cascaded encoders can include any number of disparate and/or similar encoders (i.e., n can be any value for a given path m).

The system of FIG. 9 also includes a output buffer module 40c which comprises a plurality of buffer/counters B/C m,n, each associated with a corresponding one of the encoders Em,n. In this embodiment, an input data block is sequentially applied to successive encoders (encoder stages) in the encoder path so as to increase the data compression ratio. For example, the output data block from a first encoder E1,1, is buffered and counted in B/C1,1, for subsequent processing by a second encoder E1,2. Advantageously, these parallel sets of sequential encoders are applied to the input data stream to effect content free lossless data compression. This embodiment provides for multi-stage sequential encoding of data with the maximum number of encoding steps subject to the available real-time, pseudo real-time, or other timing constraints.

As with each previously discussed embodiment, the encoders Em,n may include those lossless encoding techniques currently well known within the art, including: run length, Huffman, Lempel-Ziv Dictionary Compression, arithmetic coding, data compaction, and data null suppression. Encoding techniques are selected based upon their ability to effectively encode different types of input data. A full complement of encoders provides for broad coverage of existing and future data types. The input data blocks may be applied simultaneously to the encoder paths (i.e., the encoder paths may operate in parallel, utilizing task multiplexing on a single central processor, or via dedicated hardware, or by executing on a plurality of processor or dedicated hardware systems, or any combination thereof). In addition, an input data block may be sequentially applied to the encoder paths. Moreover, each serially-cascaded encoder path may comprise a fixed (predetermined) sequence of encoders or a random sequence of encoders. Advantageously, by simultaneously or sequentially processing input data blocks via a plurality of sets of serially-cascaded encoders, content free data compression is achieved.

The operation of the data compression system of FIG. 9 will now be discussed in further detail with reference to the flow diagram of FIGs. 10a and 10b. A data stream comprising one or more data blocks is input into the data compression system and the first data block in the data stream is received (step 100), and its size is determined by the counter module 10 (step 102). The data block is then stored in buffer 20 (step 104).

Next, concurrent with the completion of the receipt and counting of the first data block, the interval timer 90 is initialized (step 106) and starts counting towards a user-specified time limit. The input data block is then sent to the cascade encoder module 30C wherein the input data block is applied to the first encoder (i.e., first encoding stage) in each of the cascaded encoder paths E1,1 ... Em,1 (step 108). Next, a determination is made as to whether the user specified time expires before the completion of the first stage encoding process (steps 110 and 112). If the first stage encoding process is completed before the expiration of the timer, i.e., each encoder (E1,1 ... Em,1) completes its respective encoding process (negative result in step 110 and affirmative result in step 112), then an encoded data block is output from each encoder E1,1...Em,1 and maintained in a corresponding buffer (step 114). Then for each cascade encoder path, the output of the completed encoding stage is applied to the next successive encoding stage in the cascade path (step 116). This process (steps 110, 112, 114, and 116) is repeated until the earlier of the timer expiration (affirmative result in step 110) or the completion of encoding by each encoder stage in the serially-cascaded paths, at which time the encoding process is halted (step 118).

Then, for each cascade encoder path, the buffered encoded data block output by the last encoder stage that completes the encoding process before the expiration of the timer is selected for further processing (step 120). Advantageously, the interim stages of the multi-stage data encoding process are preserved. For example, the results of encoder E1,1 are preserved even after encoder E1,2 begins encoding the output of encoder E1,1. If the interval timer expires after encoder E1,1 completes its respective encoding process but before encoder E1,2 completes its respective encoding process, the encoded data block from encoder E1,1 is complete and is utilized for calculating the compression ratio for the corresponding encoder path. The incomplete encoded data block from encoder E1,2 is either discarded or ignored.

It is to be appreciated that it is not necessary (or in some cases desirable) that some or all of the encoders in the cascade encoder paths complete the encoding process before the interval timer expires. Specifically, due to encoder data dependency, natural variation and the sequential application of the cascaded encoders, it is possible that certain encoders may not operate quickly enough and therefore do not comply with the timing constraints of the end use. Accordingly, the time limit ensures that the real-time or pseudo real-time nature of the data encoding is preserved.

After the encoded data blocks are selected (step 120), the size of each encoded data block is counted (step 122). Next, a compression ratio is calculated for each encoded data block by taking the ratio of the size of the input data block (as determined by the input counter 10) to the size of the encoded data block output from each encoder (step 124). Each compression ratio is then compared with an *a priori*-specified compression ratio threshold (step 126). A determination is made as to whether the compression ratio of at least one of the encoded data blocks exceeds the threshold limit (step 128). If there are no encoded data blocks having a compression ratio that exceeds the compression ratio threshold limit (negative determination in step 128), then the original unencoded input data block is selected for output and a null data compression type descriptor is appended thereto (step 130). The original unencoded data block and its corresponding null data compression type descriptor is then output for subsequent data processing, storage, or transmittal (step 132).

On the other hand, if one or more of the encoded data blocks possess a compression ratio greater than the compression ratio threshold limit (affirmative result in step 128), then a figure of merit is calculated for each encoded data block having a compression ratio which exceeds the compression ratio threshold limit (step 134). Again, the figure of merit for a given encoded data block is comprised of a weighted average of the *a priori* user specified threshold and the corresponding encoder desirability factor associated with the encoded data block. Next, the encoded data block having the greatest figure of merit is selected (step 136). An appropriate data compression type descriptor is then appended (step 138) to indicate the data encoding technique applied to the encoded data block. For instance, the data type compression descriptor can indicate that the encoded data block was processed by either a single encoding type, a plurality of sequential encoding types, and a plurality of random encoding types. The encoded data block (which has the greatest figure of merit) along with its corresponding data compression type descriptor is then output for subsequent data processing, storage, or transmittal (step 140).

After the unencoded data block or the encoded data input data block is output (steps 132 and 140), a determination is made as to whether the input data stream contains additional data blocks to be processed (step 142). If the input data stream includes additional data blocks (affirmative result in step 142), then the next successive data block is received (step 144), its block size is counted (return to step 102) and the data compression process is iterated for each successive data block in the input data stream. Once the final input data block is processed (negative result in step 142), data compression of the input data stream is finished (step 146).

Referring now to FIG. 11, a block diagram illustrates a data decompression system according to one embodiment of the present invention. The data decompression system preferably includes an input buffer 1100 which receives as input an uncompressed or compressed data stream comprising one or more data blocks. The data blocks may range in size from individual bits through complete files or collections of multiple files. Additionally, the data block size may be fixed or variable. The input data buffer 1100 is preferably included (not required) to provide storage of input data for various hardware implementations. A descriptor extraction module 1102 receives the buffered (or unbuffered) input data block and then parses, lexically, syntactically, or otherwise analyzes the input data block using methods known by those skilled in the art to extract the data compression type descriptor associated with the data block. The data compression type descriptor may possess values corresponding to null (no encoding applied), a single applied encoding technique, or multiple encoding techniques applied in a specific or random order (in accordance with the data compression system embodiments and methods discussed above).

A decoder module 1104 includes a plurality of decoders D1...Dn for decoding the input data block using a decoder, set of decoders, or a sequential set of decoders corresponding to the extracted compression type descriptor. The decoders D1...Dn may include those lossless encoding techniques currently well known within the art, including: run length, Huffman, Lempel-Ziv Dictionary Compression, arithmetic coding, data compaction, and data null suppression. Decoding techniques are selected based upon their ability to effectively decode the various different types of encoded input data generated by the data compression systems described above or originating from any other desired source. As with the data compression systems discussed above, the decoder module 1104 may include multiple decoders of the same type applied in parallel so as to reduce the data decoding time.

The data decompression system also includes an output data buffer 1106 for buffering the decoded data block output from the decoder module 1104.

The operation of the data decompression system of FIG. 11 will be discussed in further detail with reference to the flow diagram of FIG. 12. A data stream comprising one or more data blocks of compressed or uncompressed data is input into the data decompression system and the first data block in the stream is received (step 1200) and maintained in the buffer (step 1202). As with the data compression systems discussed above, data decompression is performed on a per data block basis. The data compression type descriptor is then extracted from the input data block (step 1204). A determination is then made as to whether the data compression type descriptor is null (step 1206). If the data compression type descriptor is determined to be null (affirmative result in step 1206), then no decoding is applied to the input data block and the original undecoded data block is output (or maintained in the output buffer) (step 1208).

On the other hand, if the data compression type descriptor is determined to be any value other than null (negative result in step 1206), the corresponding decoder or decoders are then selected (step 1210) from the available set of decoders D1...Dn in the decoding module 1104. It is to be understood that the data compression type descriptor may mandate the application of: a single specific decoder, an ordered sequence of specific decoders, a random order of specific decoders, a class or family of decoders, a mandatory or optional application of parallel decoders, or any combination or permutation thereof. The input data block is then decoded using the selected decoders (step 1212), and output (or maintained in the output buffer 1106) for subsequent data processing, storage, or transmittal (step 1214). A determination is then made as to whether the input data stream contains additional data blocks to be processed (step 1216). If the input data stream includes additional data blocks (affirmative result in step 1216), the next successive data block is received (step 1220), and buffered (return to step 1202). Thereafter, the data decompression process is iterated for each data block in the input data stream. Once the final input data block is processed (negative result in step 1216), data decompression of the input data stream is finished (step 1218).

Although illustrative embodiments have been described herein with reference to the accompanying drawings, it is to be understood that the present invention is not limited to those precise embodiments, and that various other changes and modifications may be affected therein by one skilled in the art without departing from the scope or spirit of the invention. All such changes and modifications are intended to be included within the scope of the invention as defined by the appended claims.

WHAT IS CLAIMED IS:

1. A method for compressing data, comprising the steps of:
receiving an input data stream comprising a plurality of disparate data types;
compressing the input data stream using each of a plurality of different encoders; and
generating an encoded data stream by selectively combining compressed data blocks output from each of the encoders based on compression ratios obtained by the encoders.

2. The method of claim 1, wherein the step of generating the encoded data stream comprises tagging each compressed data block with a compression type descriptor.

3. The method of claim 1, wherein the step of generating the encoded data stream comprises combining uncompressed data blocks from the input data stream with the compressed data blocks and tagging each uncompressed data block with a null compression descriptor.

4. The method of claim 1, wherein the step of compressing the input data stream comprises compressing each data block in the input data stream using each of the encoders, and wherein the step of generating the encoded data stream comprises:
for each data block in the input stream,
determining a compression ratio obtained from each of the encoders;
selecting for output the input data block and appending a null compression descriptor to input the data block, if no compression ratio meets a predetermined threshold; and
selecting for output the encoded data block having the greatest compression ratio associated therewith that meets the predetermined threshold and appending a compression type descriptor to the selected encoded data block.

5. The method of claim 4, further comprising the step of applying a predetermined timing constraint to the compression process to provide real-time data compression of the input data stream.

6. The method of claim 5, wherein the step of applying a predetermined time constraint comprises the steps of:

initializing a timer with a user-specified time interval upon commencing compression of an input data block; and

terminating the encoding step upon the earlier of one of the expiration of the timer and the completion of the encoding of the input data block by all of the plurality of encoders, wherein the step of determining the compression ratios is only performed for the encoders that have completed encoding of the input data block before expiration of the timer.

7. The method of claim 1, wherein the method steps are tangibly embodied on a program storage device as program instructions that are readable by a machine and executable by the machine to perform the method steps.

8. A method for providing content independent data compression, comprising the steps of:

receiving as input a block of data from a stream of data;

encoding the input data block with a plurality of encoders to provide a plurality of encoded data blocks;

determining a compression ratio obtained for each of the encoders;

comparing each of the determined compression ratios with a predefined compression threshold;

selecting for output the input data block and appending a null compression descriptor to the input data block, if all of the encoder compression ratios fall below the predefined compression threshold; and

selecting for output the encoded data block having the highest compression ratio and appending a corresponding compression type descriptor to the selected encoded data block, if at least one of the compression ratios meets the predefined compression threshold.

9. The method of claim 8, further including the step of buffering the input data block.

10. The method of claim 8, wherein the size of the input data blocks are one of fixed, variable, and a combination thereof.

11. The method of claim 8, wherein the input data stream comprises one of compressed and uncompressed data blocks, and a combination thereof.

12. The method of claim 8, wherein the step of compressing the data block comprises simultaneously applying the data block to the plurality of encoders in parallel.

13. The method of claim 8, wherein the step of compressing the data block comprises sequentially applying the data block to a plurality of encoders.

14. The method of claim 8, further comprising the steps of:

initializing a timer with a user-specified time interval upon commencing the encoding of the input data block; and

terminating the encoding step upon the earlier of one of the expiration of the timer and the completion of the encoding of the input data block by all of the plurality of encoders, wherein the step of determining the compression ratios is only performed for the encoders that have completed encoding of the input data block before expiration of the timer.

15. The method of claim 8, wherein the method steps are tangibly embodied on a program storage device as program instructions that are readable by a machine and executable by the machine to perform the method steps.

16. A method for providing content independent data compression, comprising the steps of:

receiving as input a block of data from a stream of data;

compressing the input data block with a plurality of encoders and appending a corresponding encoder desirability factor to the encoded data block output from each of the encoders;

determining a compression ratio obtained by each of the encoders;

comparing each of the determined compression ratios with a predefined compression threshold;

selecting for output the input data block and appending a null compression descriptor to the input data block, if all of the encoder compression ratios fall below the predefined compression threshold;

calculating a figure of merit for each encoded data block having a compression ratio associated therewith that meets the predefined compression threshold, wherein the figure of merit comprises a weighted average of the predefined compression threshold and the corresponding encoder desirability factor; and

selecting for output the encoded data block having the highest figure of merit and appending a corresponding compression type descriptor to the selected encoded data block.

17. The method of claim 16, further comprising the steps of:

initializing a timer with a user-specified time interval upon commencing the encoding of the input data block; and

terminating the encoding step upon the earlier of one of the expiration of the timer and the completion of the encoding of the input data block by all of the plurality of encoders, wherein the step of determining the compression ratios is only performed for the encoders that have completed encoding of the input data block before expiration of the timer.

18. The method of claim 16, wherein the method steps are tangibly embodied on a program storage device as program instructions that are readable by a machine and executable by the machine to perform the method steps.

19. A method for providing content independent data compression, comprising the steps of:

receiving as input a block of data from a stream of data;

compressing the input data block with a plurality of encoders, wherein each encoder comprises a plurality of serially-cascaded encoders;

appending a corresponding encoder desirability factor to each of the encoded data blocks output from each of the encoders;

determining a data compression ratio obtained by each of the encoders;

comparing each of the determined compression ratios with a predefined compression threshold;

selecting for output the input data block and appending a null data type compression descriptor to the input data block, if all of the compression ratios fall below the predefined compression threshold

calculating a figure of merit for each encoded data block which meets the predefined compression threshold, wherein the figure of merit comprises a weighted average of the predefined compression threshold and the corresponding encoder desirability factor; and

selecting for output the encoded data block having the highest figure of merit and appending a corresponding compression type descriptor to the selected encoded data block.

20. The method of claim 19, wherein the corresponding compression type descriptor indicates one of a single encoding type descriptor, a plurality of sequential encoding types descriptor, and a plurality of random encoding types descriptor.

21. The method of claim 19, further comprising the steps of:

initializing a timer with a user-specified time interval upon commencing encoding of the input data block;

buffering the encoded data block output from each serially-cascaded encoder in the encoder;

terminating the encoding step upon the earlier of one of the expiration of the timer and the completion of the encoding of the input data block by all the encoders, wherein the step of determining the compression ratios for a given encoder is performed by determining the compression ratio for the corresponding serially-cascaded encoder that has last completed encoding of the input data block before expiration of the timer.

22. The method of claim 19, wherein the method steps are tangibly embodied on a program storage device as program instructions that are readable by a machine and executable by the machine to perform the method steps.

23. A method for providing content independent data decompression, comprising the steps of:

receiving as input a block of data from a stream of data, the data stream comprising one of at least one data block and a plurality of data blocks;

extracting an encoding type descriptor from the input data block;

decoding the input data block with one of a single and multiple decoders in accordance with the extracted encoding type descriptor; and

outputting the decoded data block.

24. The method of claim 23, further comprising the step of outputting the input data block without decoding if the extracted encoding type descriptor is a null encoding descriptor.

25. The method of claim 23, wherein the method steps are tangibly embodied on a program storage device as program instructions that are readable by a machine and executable by the machine to perform the method steps.

26. A content independent data compression system, comprising:

a plurality of encoders, wherein each encoder is adapted to receive and compress an input data block;

a first processing unit adapted to (i) compute a compression ratio of encoded data blocks, associated with the input data block, that are output from the encoders and to (ii) compare the compression ratios with a predetermined compression threshold; and

a second processing unit adapted to (i) append a null compression descriptor to the input data block and output the input data block with the appended null compression descriptor, if each compression ratio falls below the predetermined compression threshold and to (ii) append a compression type descriptor to an encoded data block having the highest compression ratio that meets the predefined compression threshold, and output the encoded data block with the appended compression type descriptor.

27. The system of claim 26, further comprising a buffer for buffering input data blocks.

28. The system of claim 26, wherein the encoders are connected in parallel and concurrently process the input data block.

29. The system of claim 26, wherein the encoders are implemented as one of a single digital signal processor and a plurality of digital signal processors.

30. The system of claim 26, further comprising a timer, operatively associated with the encoders, for terminating the encoding process upon the earlier of one of the expiration of the timer and the completion of the encoding of the input data block by all of the encoders, wherein the compression ratio of a given encoder is computed only if the given encoder has completed encoding of the input data block before expiration of the timer.

31. A content independent data compression system, comprising:

a plurality of encoders, wherein each encoder is adapted to receive and compress an input data block;

a first processing unit adapted to (i) compute a compression ratio of encoded data blocks, associated with the input data block, that are output from the encoders and to (ii) compare the compression ratios with a predetermined compression threshold;

a second processing unit adapted to append a corresponding encoder desirability factor to each of the encoded data blocks;

a third processing unit adapted to compute a figure of merit for each encoded data block which meets the predefined compression threshold, wherein the figure of merit comprises a weighted average of the predefined compression threshold and the corresponding encoder desirability factor; and

a fourth processing unit adapted to (i) append a null compression descriptor to the input data block and output the input data block with the appended null compression descriptor, if each compression ratio falls below the predetermined compression threshold and to (ii) append a compression type descriptor to an encoded data block having the highest figure of merit, and output the encoded data block with the appended compression type descriptor.

32. The system of claim 31, further comprising a timer, operatively associated with the encoders, for terminating the encoding process upon the earlier of one of the expiration of the timer and the completion of the encoding of the input data block by all of the plurality of encoders, wherein the compression ratio for a given encoder is computed only if the given encoder has completed encoding of the input data block before expiration of the timer.

33. A content independent data compression system, comprising:

a plurality of encoders, wherein each encoder is adapted to receive and compress an input data block and wherein each encoder comprises a plurality of serially-cascaded encoders;

a first processing unit adapted to (i) compute a compression ratio of encoded data blocks, associated with the input data block, that are output from the encoders and to (ii) compare the compression ratios with a predetermined compression threshold;

a second processing unit adapted to append a corresponding encoder desirability factor to each of the encoded data blocks;

a third processing unit adapted to compute a figure of merit for each encoded data block which meets the predefined compression threshold, wherein the figure of merit comprises a weighted average of the predefined compression threshold and the corresponding encoder desirability factor; and

a fourth processing unit adapted to (i) append a null compression descriptor to the input data block and output the input data block with the appended null compression descriptor, if each compression ratio falls below the predetermined compression threshold and to (ii) append a compression type descriptor to an encoded data block having the highest figure of merit, and output the encoded data block with the appended compression type descriptor.

34. The system of claim 33, further comprising:

a buffer adapted to store the encoded data block output from each serially-cascaded encoder comprising the encoders; and

a timer, operatively associated with the encoders, for terminating the encoding process upon the earlier of one of the expiration of the timer and the completion of the encoding of the input data block by all of the plurality of encoders, wherein the compression ratio for a given encoder is performed by determining the compression ratio for the corresponding serially-cascaded encoder that has last completed encoding of the input data block before expiration of the timer.

35. The system of claim 34, wherein the compression type descriptors include a null descriptor if no encoding is applied to an input data block, a single compression type descriptor if an input data block is encoded with a single encoder, an encoding sequence type descriptor if an input data block is encoded with a sequence of encoders, and a random encoding sequence type descriptor if an input data block is encoded with a plurality of encoding types employed in random sequence.

36. A content independent data decompression system, comprising:

a first processor adapted to extract a compression type descriptor from an input data block; and

a plurality of decoders adapted to decompress the input data block with one of a single and multiple decoders in accordance with the extracted compression type descriptor.

37. The system of claim 36, wherein the system outputs the input data block if the extracted compression type descriptor comprises a null compression descriptor.

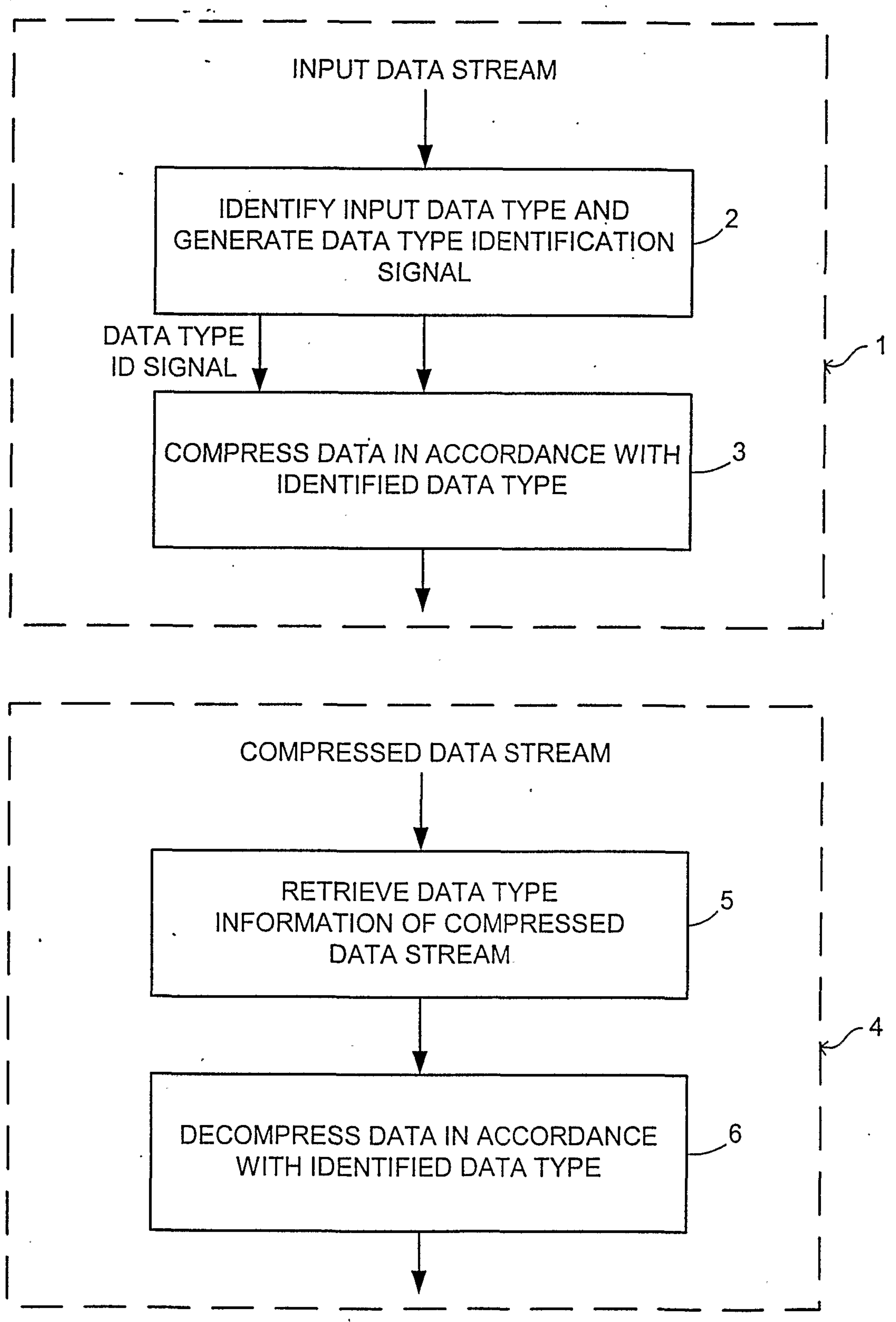

INPUT DATA STREAM
IDENTIFY INPUT DATA TYPE AND GENERATE DATA TYPE IDENTIFICATION SIGNAL
2
DATA TYPE ID SIGNAL
1
COMPRESS DATA IN ACCORDANCE WITH IDENTIFIED DATA TYPE
3
COMPRESSED DATA STREAM
RETRIEVE DATA TYPE INFORMATION OF COMPRESSED DATA STREAM
5
4
DECOMPRESS DATA IN ACCORDANCE WITH IDENTIFIED DATA TYPE
6

FIG. 1
PRIOR ART

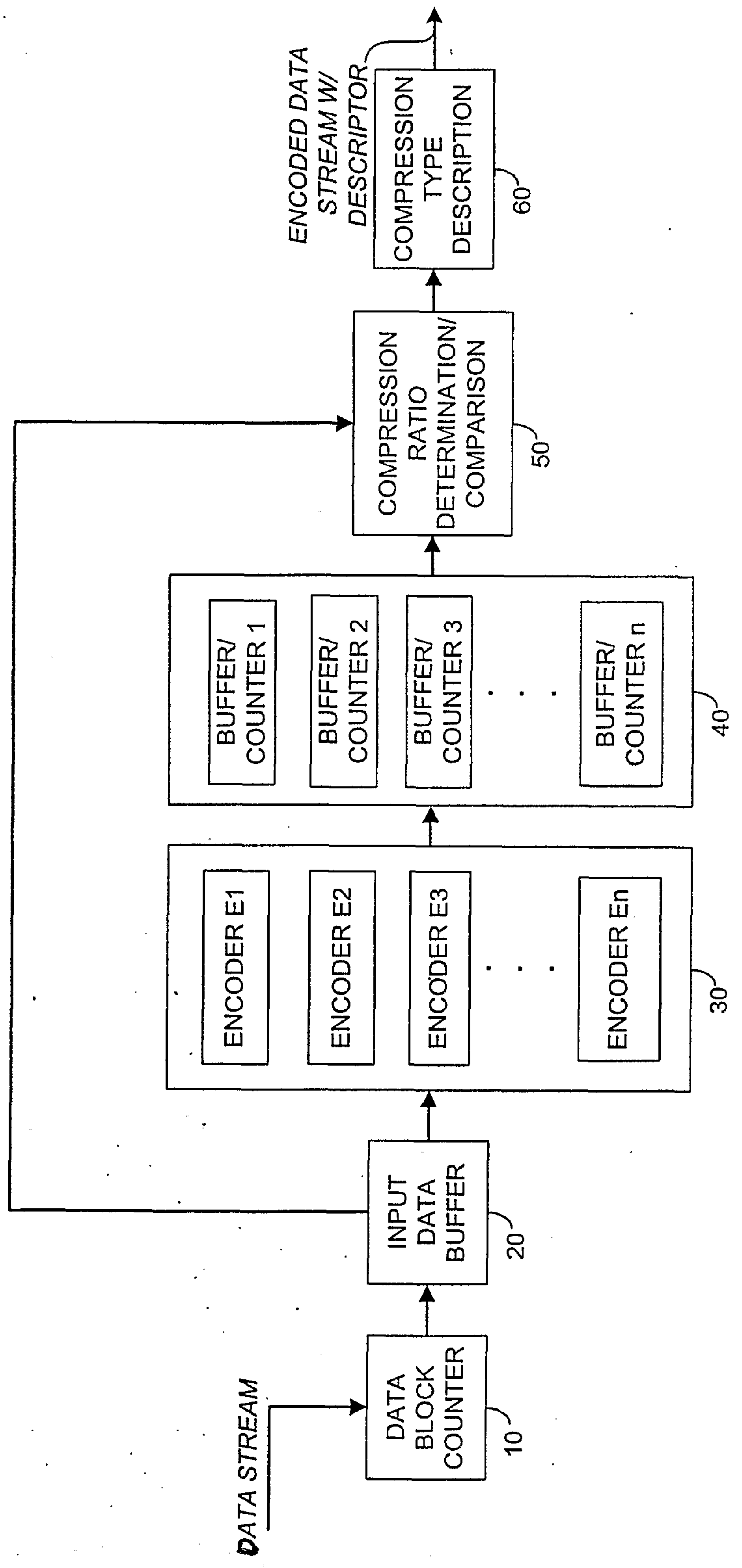

DATA STREAM
DATA BLOCK COUNTER
10
INPUT DATA BUFFER
20
ENCODER E1
ENCODER E2
ENCODER E3
ENCODER En
30
BUFFER/ COUNTER 1
BUFFER/ COUNTER 2
BUFFER/ COUNTER 3
BUFFER/ COUNTER n
40
COMPRESSION RATIO DETERMINATION/ COMPARISON
50
COMPRESSION TYPE DESCRIPTION
60
ENCODED DATA STREAM W/ DESCRIPTOR

FIG. 2

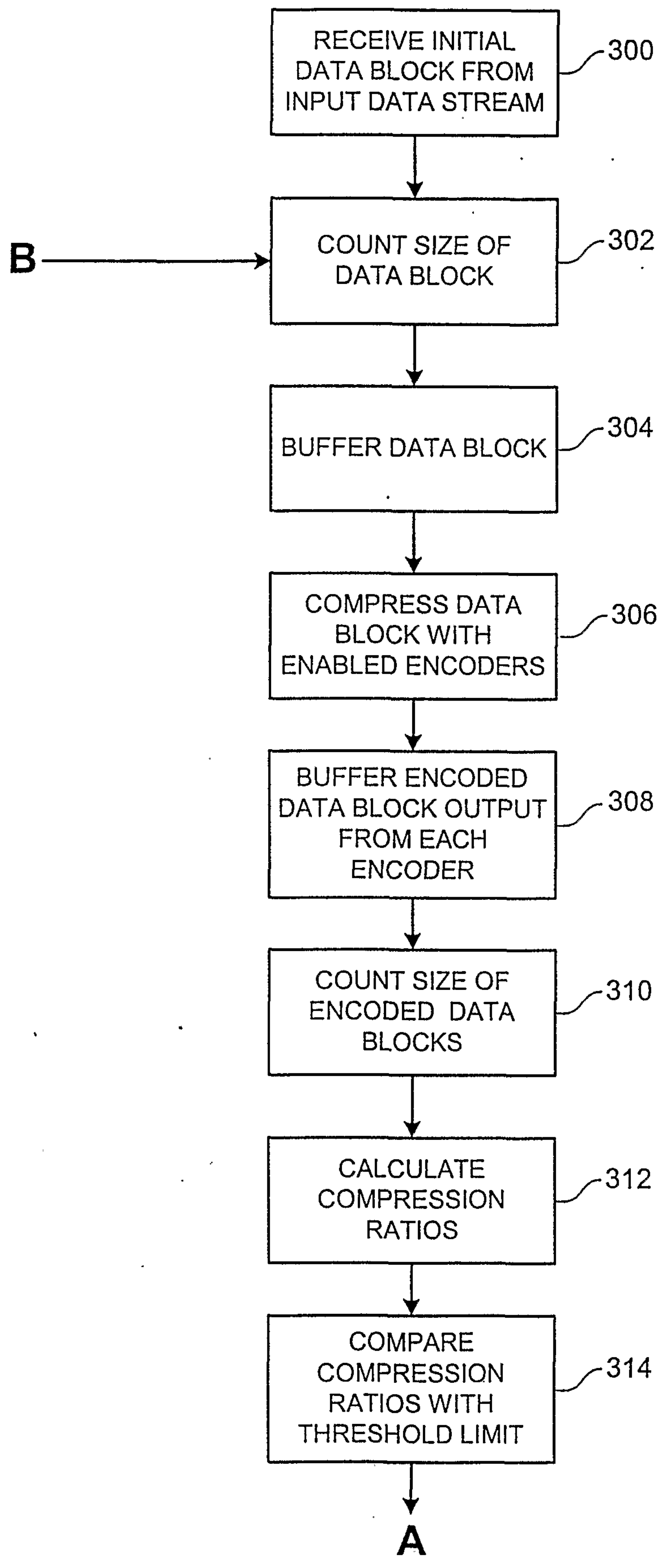

RECEIVE INITIAL DATA BLOCK FROM INPUT DATA STREAM
300
B
COUNT SIZE OF DATA BLOCK
302
BUFFER DATA BLOCK
304
COMPRESS DATA BLOCK WITH ENABLED ENCODERS
306
BUFFER ENCODED DATA BLOCK OUTPUT FROM EACH ENCODER
308
COUNT SIZE OF ENCODED DATA BLOCKS
310
CALCULATE COMPRESSION RATIOS
312
COMPARE COMPRESSION RATIOS WITH THRESHOLD LIMIT
314
A

FIG. 3a

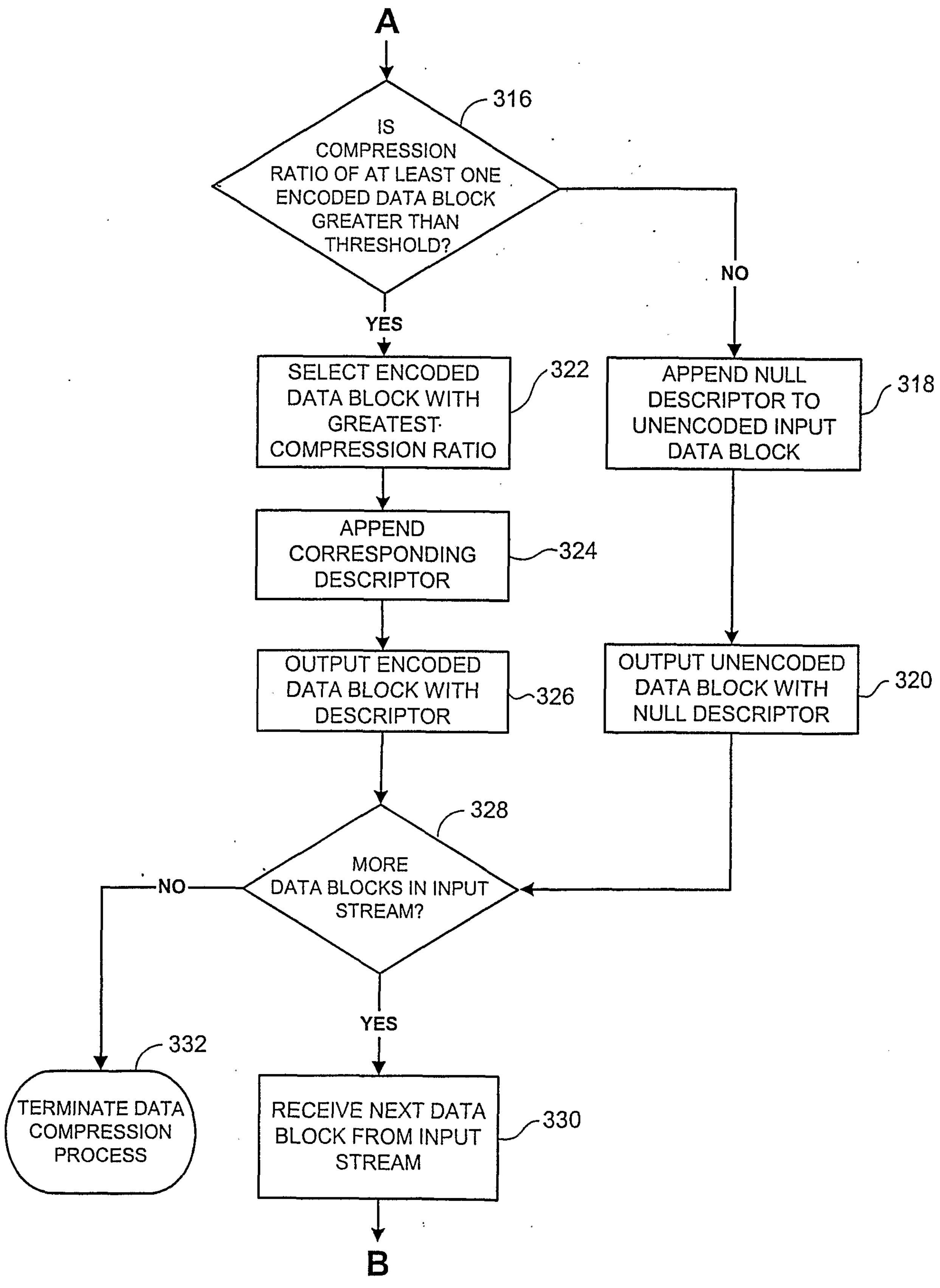

A
IS COMPRESSION RATIO OF AT LEAST ONE ENCODED DATA BLOCK GREATER THAN THRESHOLD?
316
NO
YES
SELECT ENCODED DATA BLOCK WITH GREATEST COMPRESSION RATIO
322
APPEND NULL DESCRIPTOR TO UNENCODED INPUT DATA BLOCK
318
APPEND CORRESPONDING DESCRIPTOR
324
OUTPUT ENCODED DATA BLOCK WITH DESCRIPTOR
326
OUTPUT UNENCODED DATA BLOCK WITH NULL DESCRIPTOR
320
MORE DATA BLOCKS IN INPUT STREAM?
328
NO
YES
332
TERMINATE DATA COMPRESSION PROCESS
RECEIVE NEXT DATA BLOCK FROM INPUT STREAM
330
B

FIG. 3b

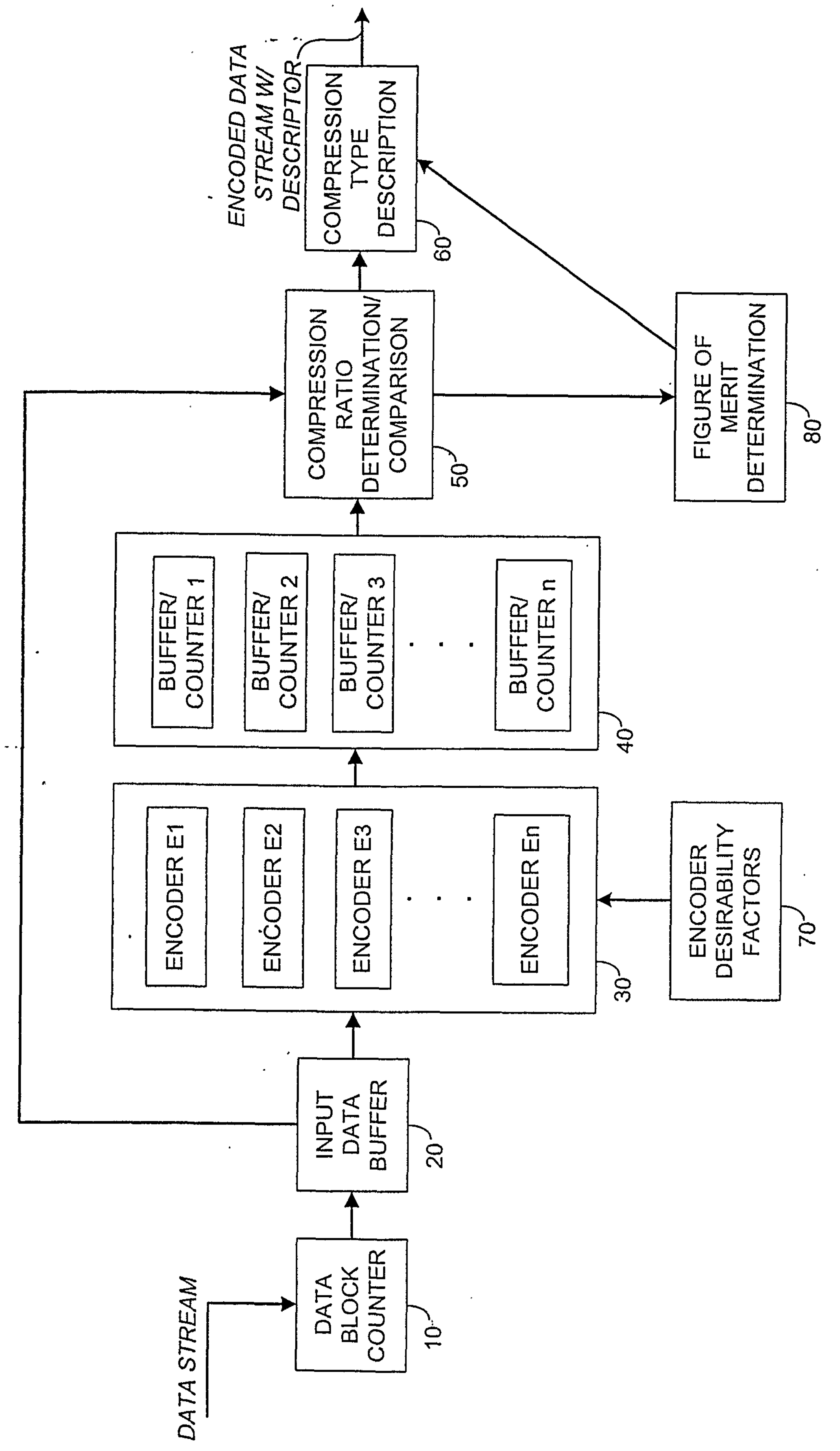

DATA STREAM
DATA BLOCK COUNTER
10
INPUT DATA BUFFER
20
ENCODER E1
ENCODER E2
ENCODER E3
ENCODER En
30
ENCODER DESIRABILITY FACTORS
70
BUFFER/ COUNTER 1
BUFFER/ COUNTER 2
BUFFER/ COUNTER 3
BUFFER/ COUNTER n
40
COMPRESSION RATIO DETERMINATION/ COMPARISON
50
FIGURE OF MERIT DETERMINATION
80
COMPRESSION TYPE DESCRIPTION
60
ENCODED DATA STREAM W/ DESCRIPTOR

FIG. 4

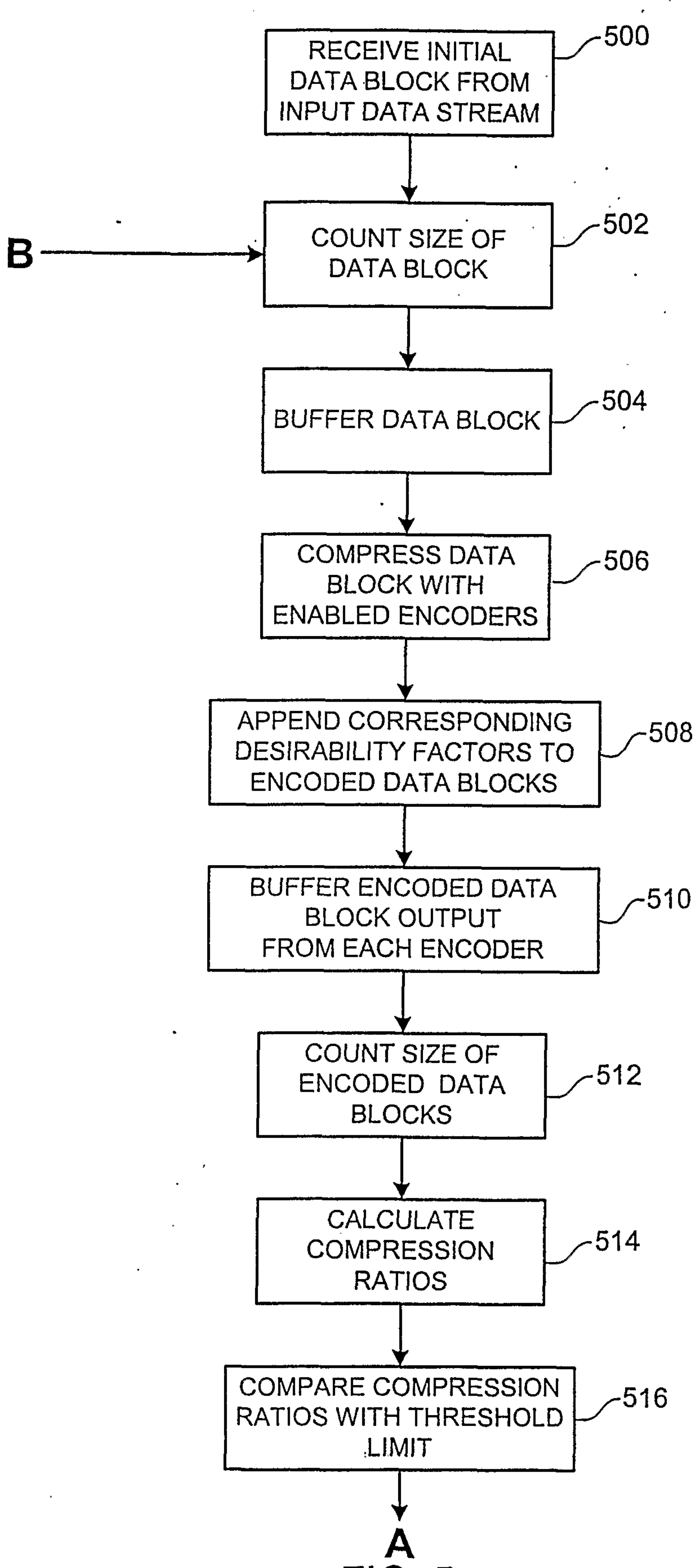

RECEIVE INITIAL DATA BLOCK FROM INPUT DATA STREAM
500
B
COUNT SIZE OF DATA BLOCK
502
BUFFER DATA BLOCK
504
COMPRESS DATA BLOCK WITH ENABLED ENCODERS
506
APPEND CORRESPONDING DESIRABILITY FACTORS TO ENCODED DATA BLOCKS
508
BUFFER ENCODED DATA BLOCK OUTPUT FROM EACH ENCODER
510
COUNT SIZE OF ENCODED DATA BLOCKS
512
CALCULATE COMPRESSION RATIOS
514
COMPARE COMPRESSION RATIOS WITH THRESHOLD LIMIT
516
A

FIG. 5a

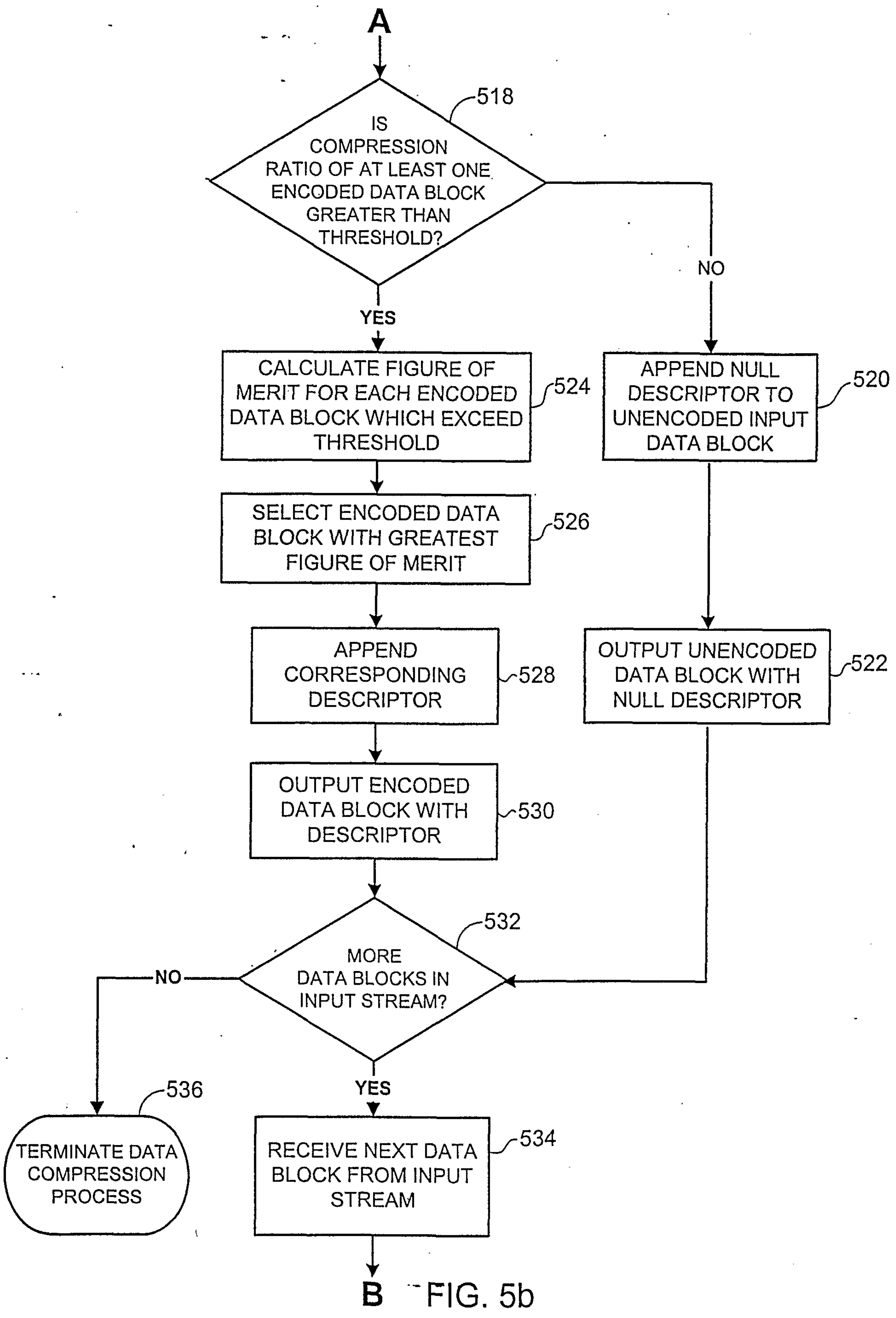

A
IS COMPRESSION RATIO OF AT LEAST ONE ENCODED DATA BLOCK GREATER THAN THRESHOLD?
518
NO
YES
CALCULATE FIGURE OF MERIT FOR EACH ENCODED DATA BLOCK WHICH EXCEED THRESHOLD
524
APPEND NULL DESCRIPTOR TO UNENCODED INPUT DATA BLOCK
520
SELECT ENCODED DATA BLOCK WITH GREATEST FIGURE OF MERIT
526
APPEND CORRESPONDING DESCRIPTOR
528
OUTPUT UNENCODED DATA BLOCK WITH NULL DESCRIPTOR
522
OUTPUT ENCODED DATA BLOCK WITH DESCRIPTOR
530
MORE DATA BLOCKS IN INPUT STREAM?
532
NO
YES
TERMINATE DATA COMPRESSION PROCESS
536
RECEIVE NEXT DATA BLOCK FROM INPUT STREAM
534
B

FIG. 5b

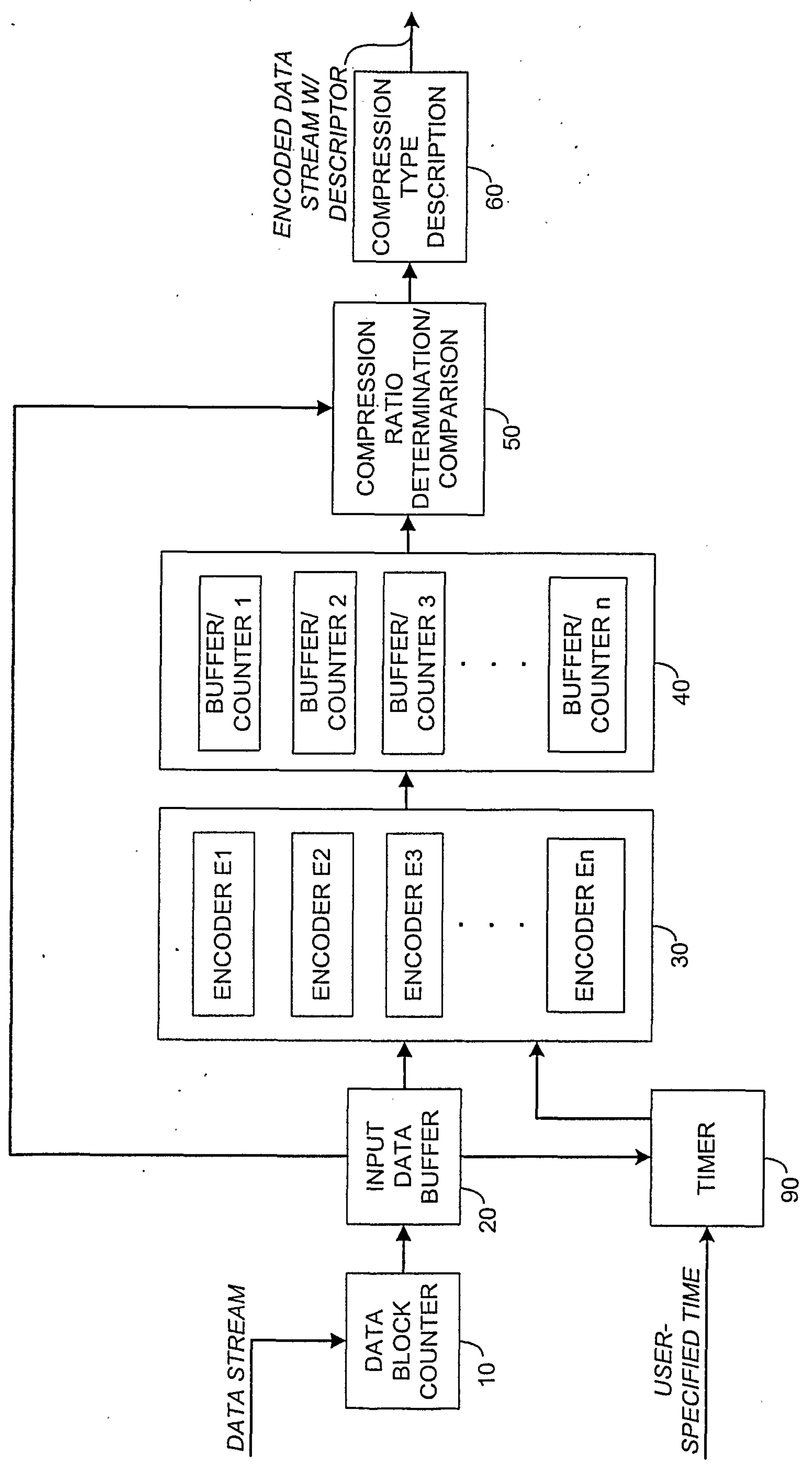

DATA STREAM
DATA BLOCK COUNTER
10
INPUT DATA BUFFER
20
USER-SPECIFIED TIME
TIMER
90
ENCODER E1
ENCODER E2
ENCODER E3
ENCODER En
30
BUFFER/ COUNTER 1
BUFFER/ COUNTER 2
BUFFER/ COUNTER 3
BUFFER/ COUNTER n
40
COMPRESSION RATIO DETERMINATION/ COMPARISON
50
COMPRESSION TYPE DESCRIPTION
60
ENCODED DATA STREAM W/ DESCRIPTOR

FIG. 6

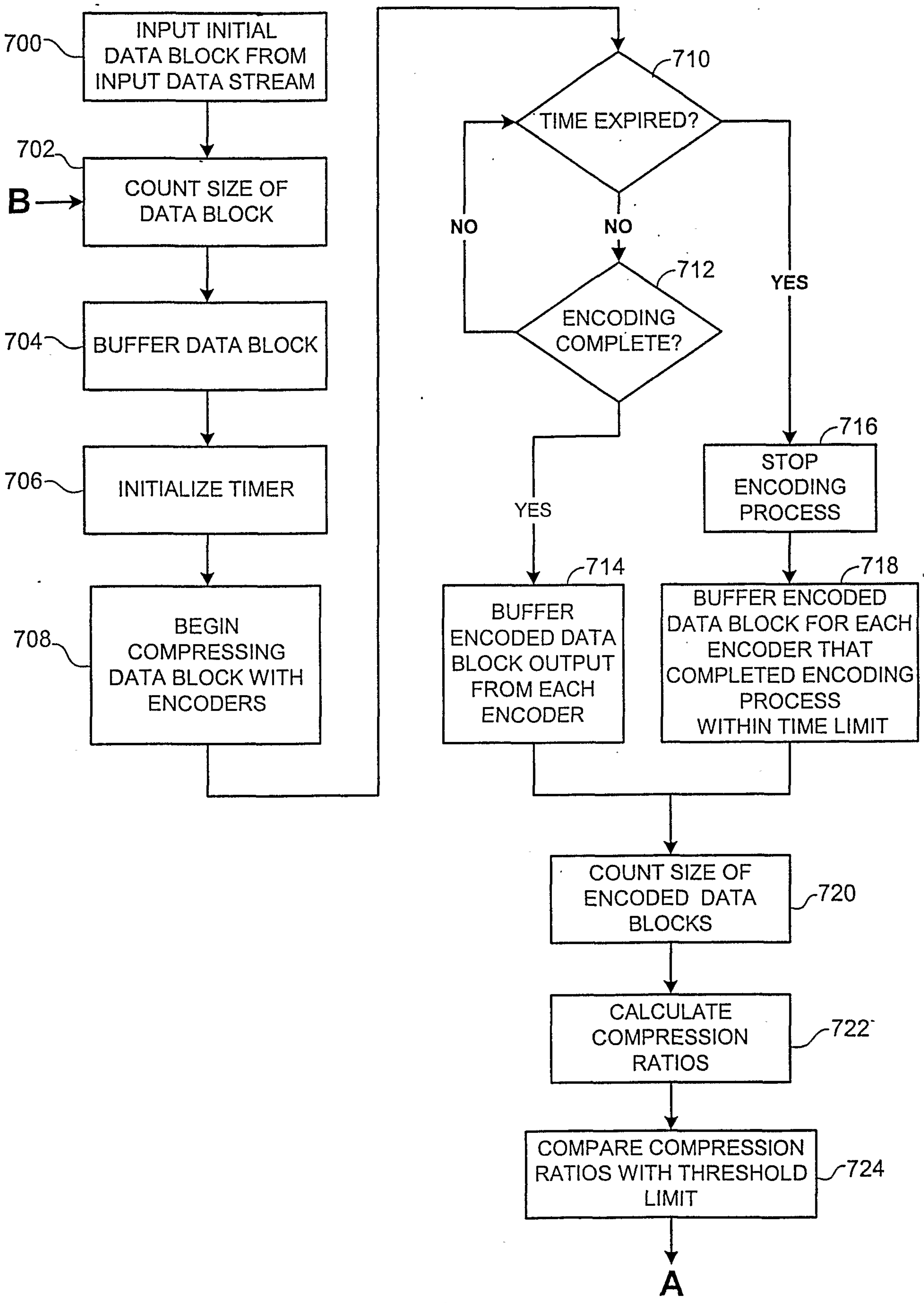

700
INPUT INITIAL DATA BLOCK FROM INPUT DATA STREAM
702
B
COUNT SIZE OF DATA BLOCK
704
BUFFER DATA BLOCK
706
INITIALIZE TIMER
708
BEGIN COMPRESSING DATA BLOCK WITH ENCODERS
710
TIME EXPIRED?
NO
NO
712
ENCODING COMPLETE?
YES
YES
716
STOP ENCODING PROCESS
714
BUFFER ENCODED DATA BLOCK OUTPUT FROM EACH ENCODER
718
BUFFER ENCODED DATA BLOCK FOR EACH ENCODER THAT COMPLETED ENCODING PROCESS WITHIN TIME LIMIT
COUNT SIZE OF ENCODED DATA BLOCKS
720
CALCULATE COMPRESSION RATIOS
722
COMPARE COMPRESSION RATIOS WITH THRESHOLD LIMIT
724
A

FIG. 7a

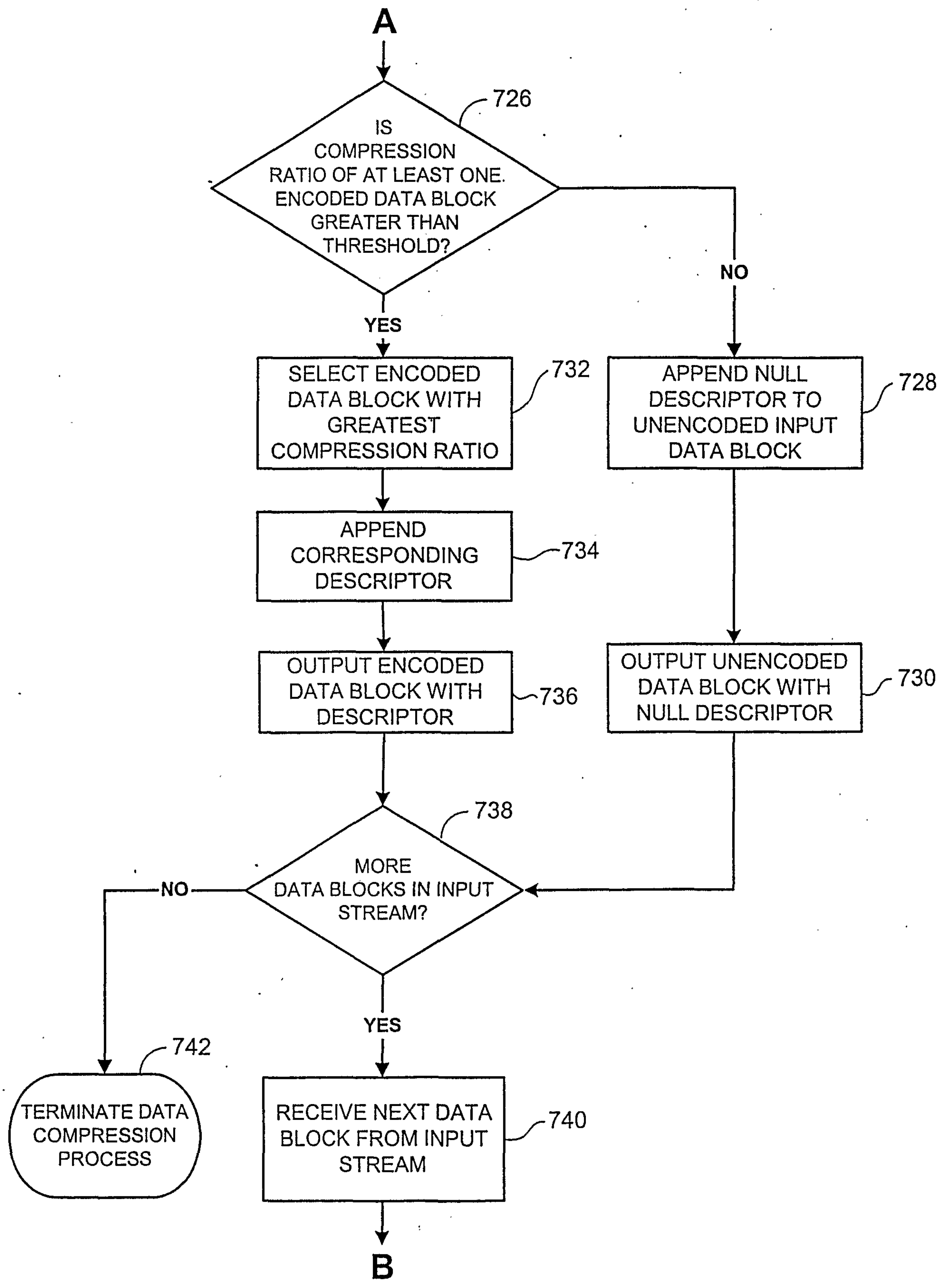

A
IS COMPRESSION RATIO OF AT LEAST ONE. ENCODED DATA BLOCK GREATER THAN THRESHOLD?
726
YES
NO
SELECT ENCODED DATA BLOCK WITH GREATEST COMPRESSION RATIO
732
APPEND NULL DESCRIPTOR TO UNENCODED INPUT DATA BLOCK
728
APPEND CORRESPONDING DESCRIPTOR
734
OUTPUT ENCODED DATA BLOCK WITH DESCRIPTOR
736
OUTPUT UNENCODED DATA BLOCK WITH NULL DESCRIPTOR
730
MORE DATA BLOCKS IN INPUT STREAM?
738
NO
YES
TERMINATE DATA COMPRESSION PROCESS
742
RECEIVE NEXT DATA BLOCK FROM INPUT STREAM
740
B

FIG. 7b

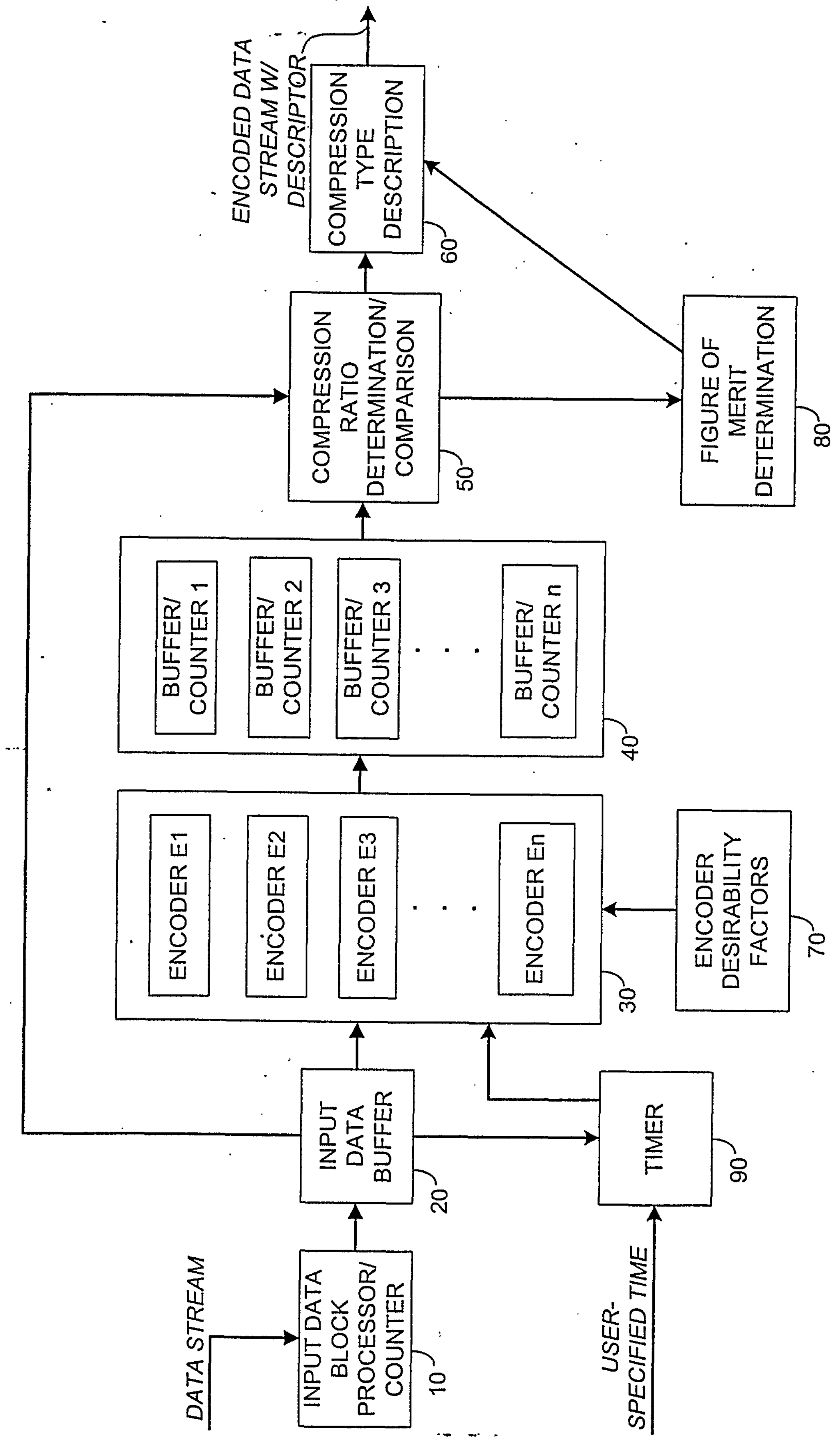

DATA STREAM
INPUT DATA BLOCK PROCESSOR/ COUNTER
10
INPUT DATA BUFFER
20
USER-SPECIFIED TIME
TIMER
90
ENCODER E1
ENCODER E2
ENCODER E3
ENCODER En
30
ENCODER DESIRABILITY FACTORS
70
BUFFER/ COUNTER 1
BUFFER/ COUNTER 2
BUFFER/ COUNTER 3
BUFFER/ COUNTER n
40
COMPRESSION RATIO DETERMINATION/ COMPARISON
50
FIGURE OF MERIT DETERMINATION
80
COMPRESSION TYPE DESCRIPTION
60
ENCODED DATA STREAM W/ DESCRIPTOR

FIG. 8

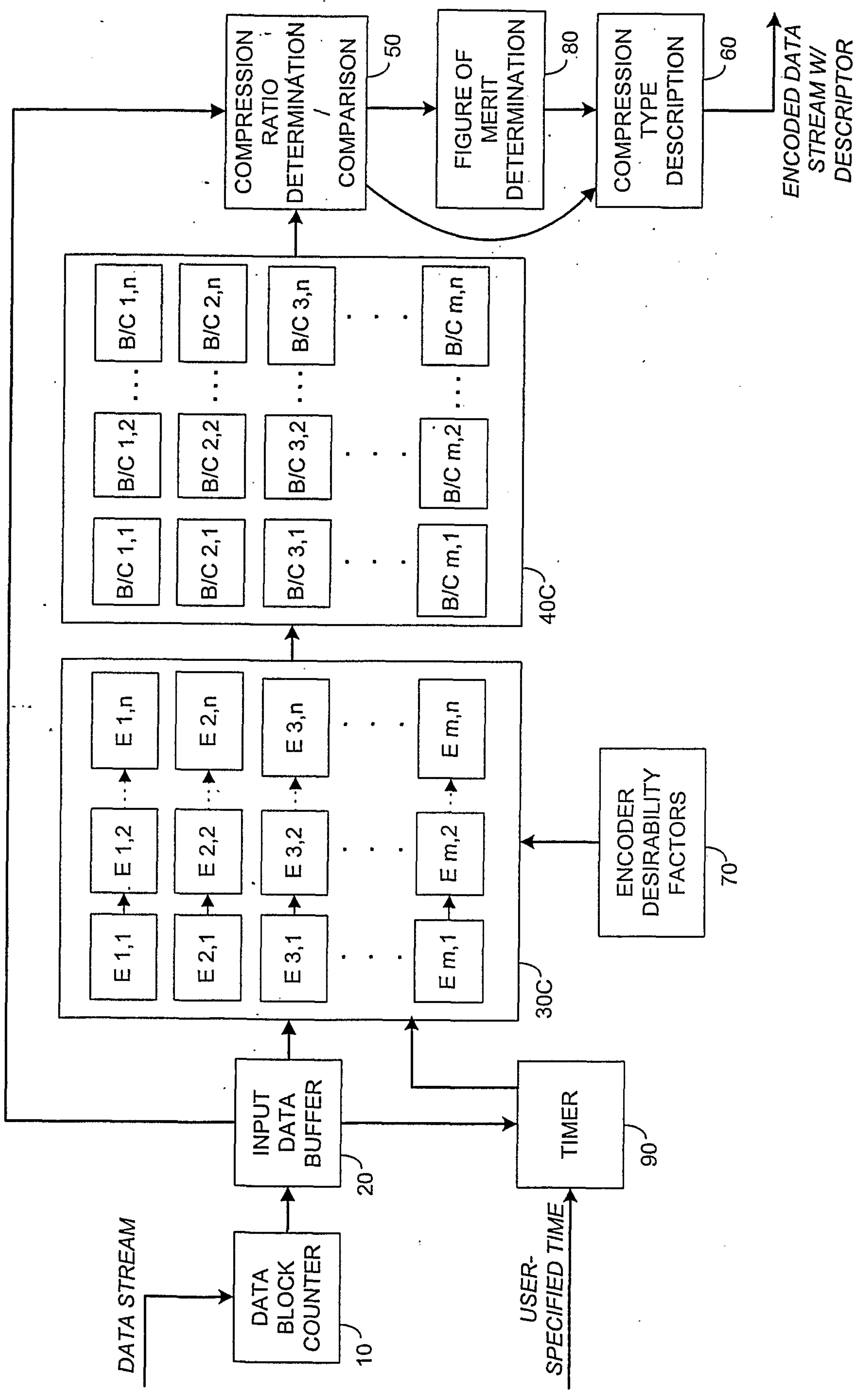

DATA STREAM
DATA BLOCK COUNTER
10
INPUT DATA BUFFER
20
USER-SPECIFIED TIME
TIMER
90
E 1,1
E 1,2
E 1,n
E 2,1
E 2,2
E 2,n
E 3,1
E 3,2
E 3,n
E m,1
E m,2
E m,n
30C
ENCODER DESIRABILITY FACTORS
70
B/C 1,1
B/C 1,2
B/C 1,n
B/C 2,1
B/C 2,2
B/C 2,n
B/C 3,1
B/C 3,2
B/C 3,n
B/C m,1
B/C m,2
B/C m,n
40C
COMPRESSION RATIO DETERMINATION / COMPARISON
50
FIGURE OF MERIT DETERMINATION
80
COMPRESSION TYPE DESCRIPTION
60
ENCODED DATA STREAM W/ DESCRIPTOR

FIG. 9

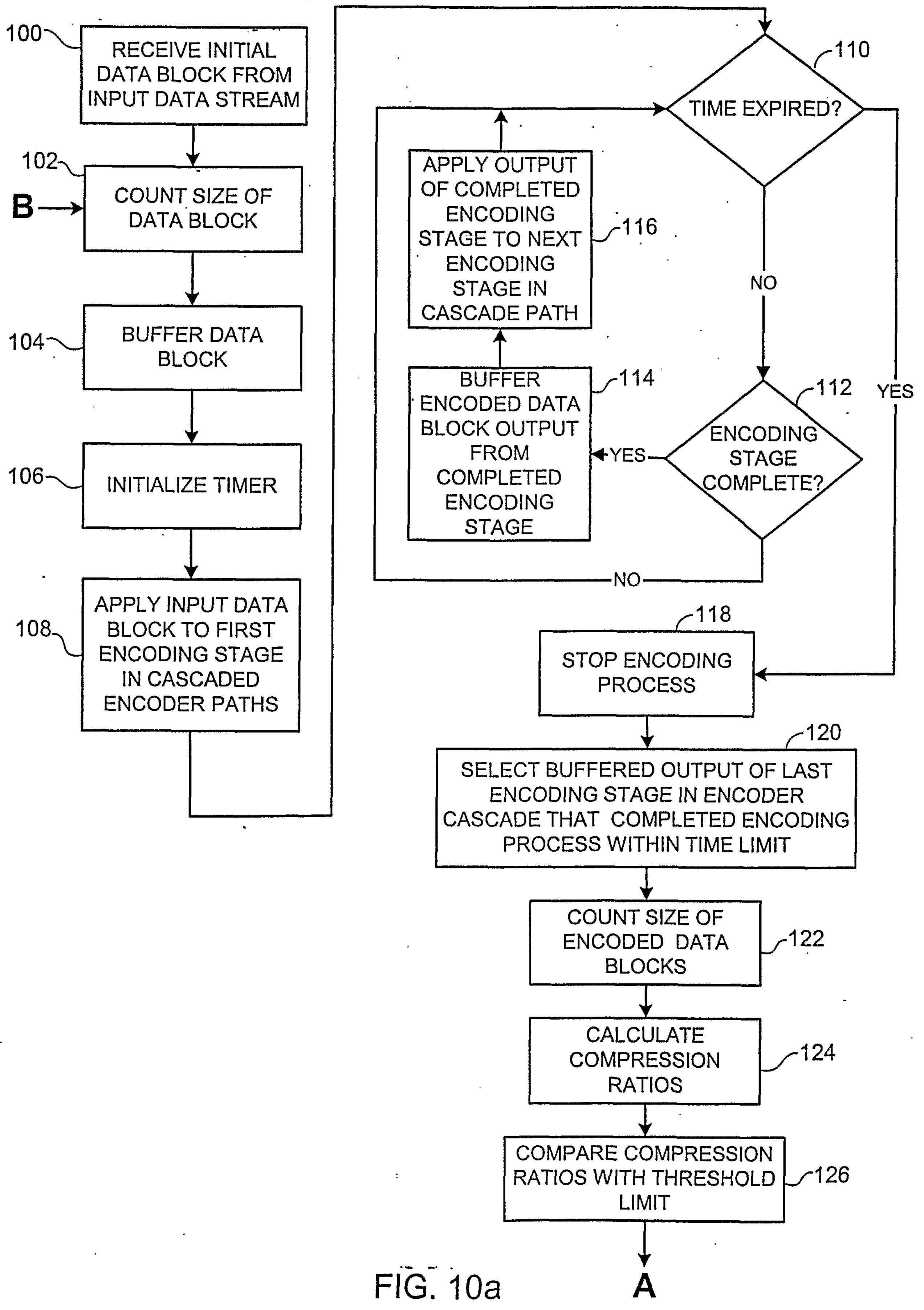

100
RECEIVE INITIAL DATA BLOCK FROM INPUT DATA STREAM
102
B
COUNT SIZE OF DATA BLOCK
104
BUFFER DATA BLOCK
106
INITIALIZE TIMER
108
APPLY INPUT DATA BLOCK TO FIRST ENCODING STAGE IN CASCADED ENCODER PATHS
110
TIME EXPIRED?
NO
YES
112
ENCODING STAGE COMPLETE?
YES
NO
114
BUFFER ENCODED DATA BLOCK OUTPUT FROM COMPLETED ENCODING STAGE
116
APPLY OUTPUT OF COMPLETED ENCODING STAGE TO NEXT ENCODING STAGE IN CASCADE PATH
118
STOP ENCODING PROCESS
120
SELECT BUFFERED OUTPUT OF LAST ENCODING STAGE IN ENCODER CASCADE THAT COMPLETED ENCODING PROCESS WITHIN TIME LIMIT
122
COUNT SIZE OF ENCODED DATA BLOCKS
124
CALCULATE COMPRESSION RATIOS
126
COMPARE COMPRESSION RATIOS WITH THRESHOLD LIMIT
A

FIG. 10a

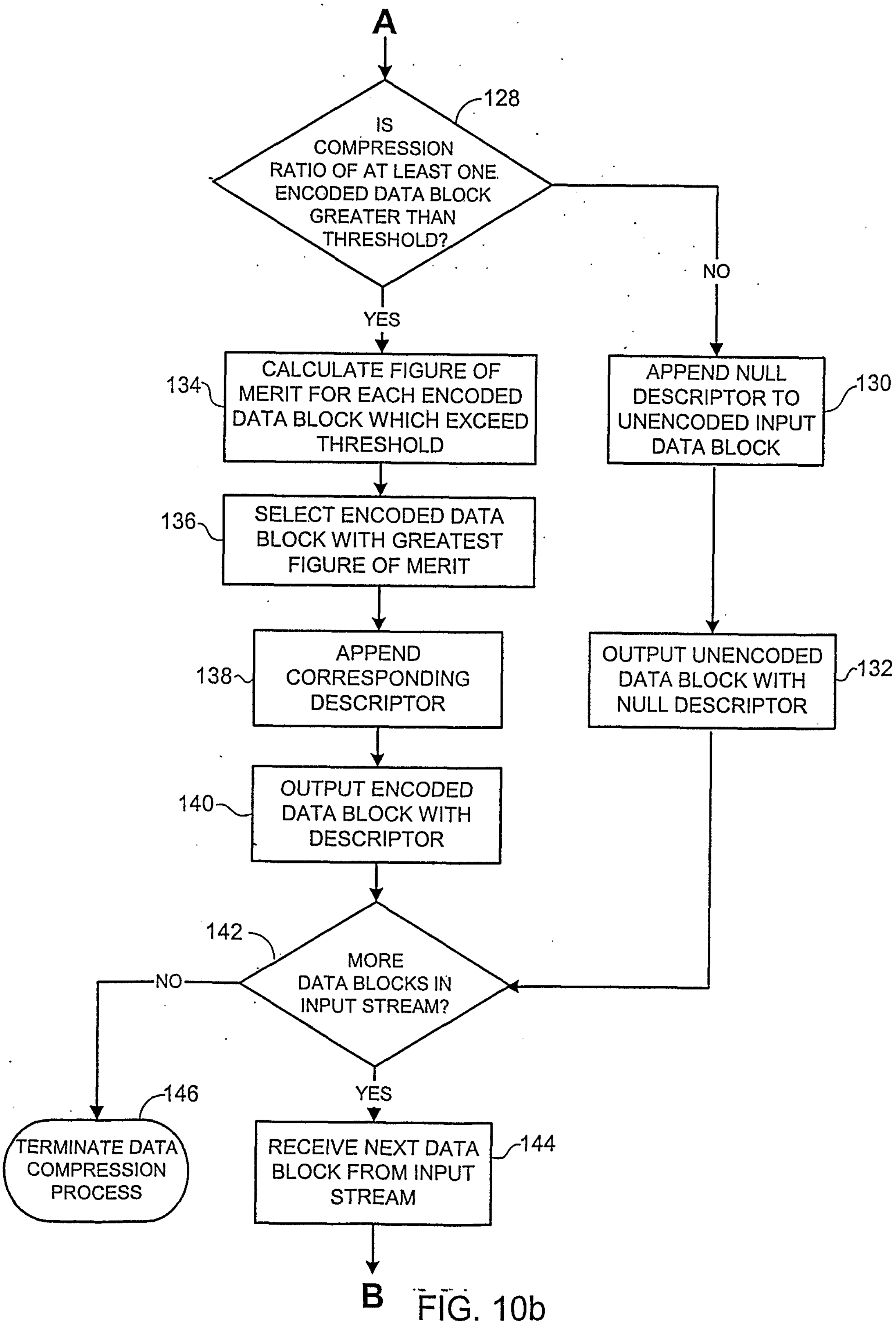
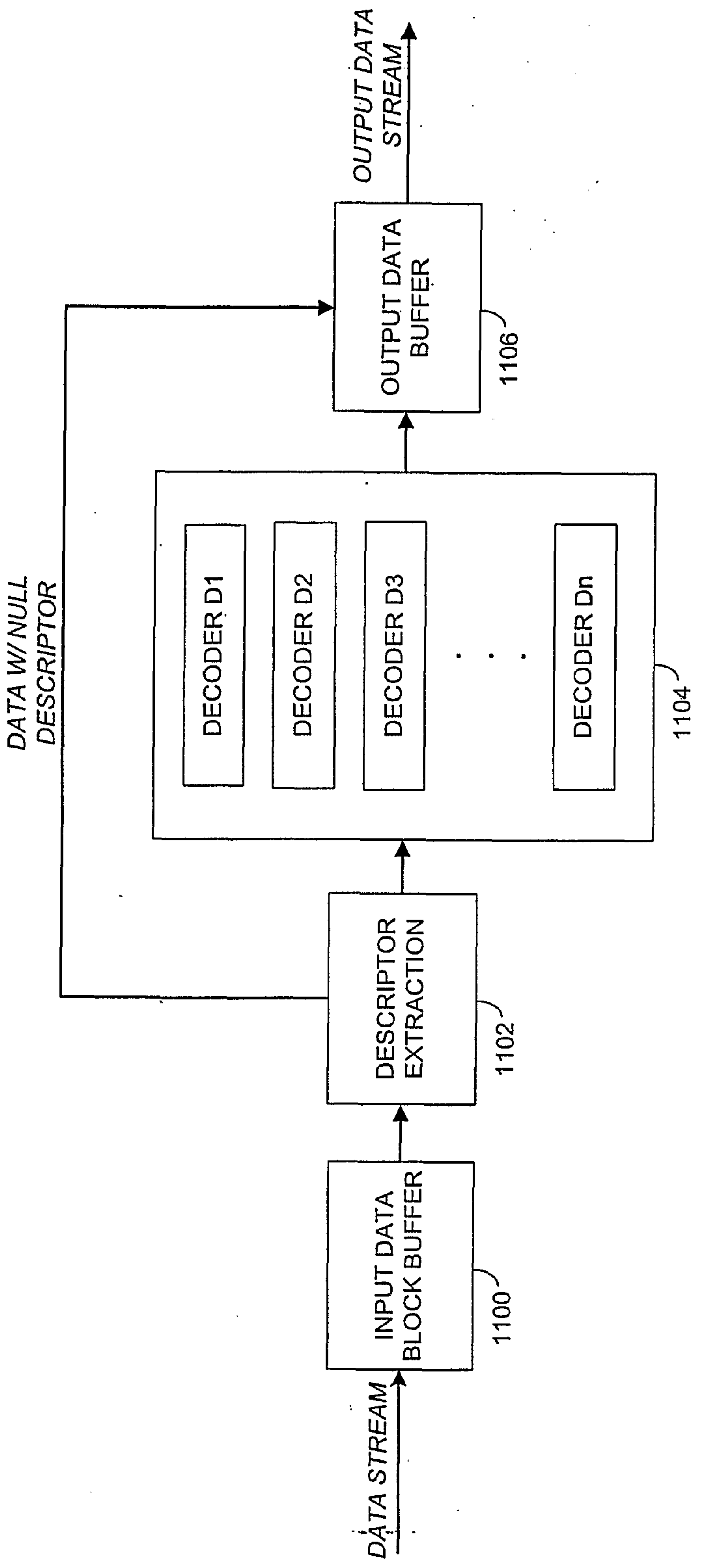
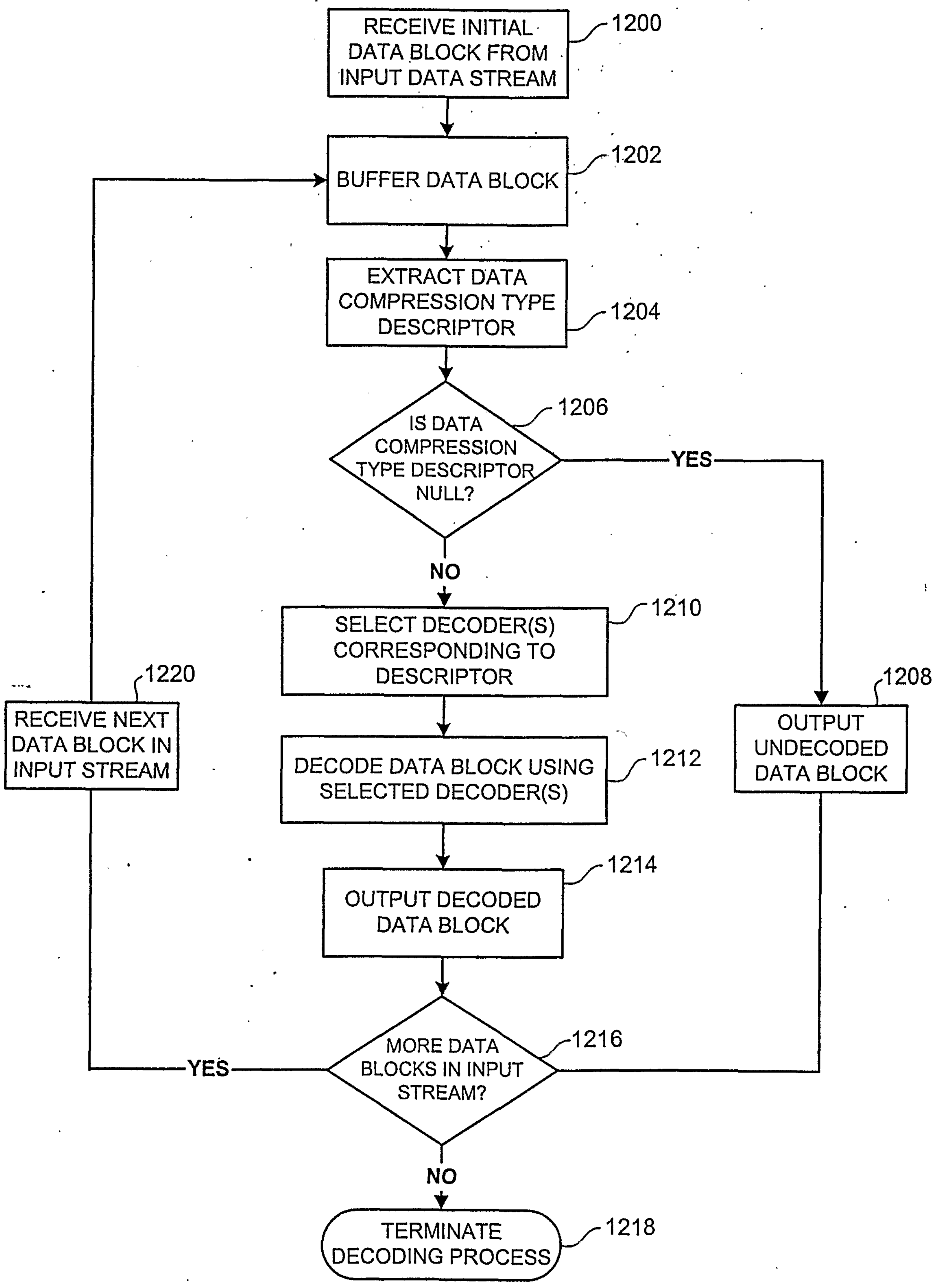

A
IS COMPRESSION RATIO OF AT LEAST ONE ENCODED DATA BLOCK GREATER THAN THRESHOLD?
128
NO
YES
CALCULATE FIGURE OF MERIT FOR EACH ENCODED DATA BLOCK WHICH EXCEED THRESHOLD
134
APPEND NULL DESCRIPTOR TO UNENCODED INPUT DATA BLOCK
130
SELECT ENCODED DATA BLOCK WITH GREATEST FIGURE OF MERIT
136
APPEND CORRESPONDING DESCRIPTOR
138
OUTPUT UNENCODED DATA BLOCK WITH NULL DESCRIPTOR
132
OUTPUT ENCODED DATA BLOCK WITH DESCRIPTOR
140
MORE DATA BLOCKS IN INPUT STREAM?
142
NO
YES
TERMINATE DATA COMPRESSION PROCESS
146
RECEIVE NEXT DATA BLOCK FROM INPUT STREAM
144
B

FIG. 10b



DATA STREAM
INPUT DATA BLOCK BUFFER
1100
DESCRIPTOR EXTRACTION
1102
DATA W/ NULL DESCRIPTOR
DECODER D1
DECODER D2
DECODER D3
DECODER Dn
1104
OUTPUT DATA BUFFER
1106
OUTPUT DATA STREAM

FIG. 11



RECEIVE INITIAL DATA BLOCK FROM INPUT DATA STREAM
1200
BUFFER DATA BLOCK
1202
EXTRACT DATA COMPRESSION TYPE DESCRIPTOR
1204
IS DATA COMPRESSION TYPE DESCRIPTOR NULL?
1206
YES
NO
SELECT DECODER(S) CORRESPONDING TO DESCRIPTOR
1210
1220
RECEIVE NEXT DATA BLOCK IN INPUT STREAM
1208
OUTPUT UNDECODED DATA BLOCK
DECODE DATA BLOCK USING SELECTED DECODER(S)
1212
OUTPUT DECODED DATA BLOCK
1214
MORE DATA BLOCKS IN INPUT STREAM?
1216
YES
NO
TERMINATE DECODING PROCESS
1218

FIG. 12

INTERNATIONAL SEARCH REPORT

Inte l Application No
PCT/US 00/42018

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H03M7/30

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 H04N

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, WPI Data, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 405 572 A (FUJITSU LTD) 2 January 1991 (1991-01-02)	1-4,7-9, 12,15, 23-29, 36,37
Y	page 5, line 22 -page 9, line 35	5,6,10, 11,13, 14,30
A		16-22, 31-35

-/--

[X] Further documents are listed in the continuation of box C. [X] 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 citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search 29 June 2001	Date of mailing of the international search report 31/07/2001
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Lombardi, G

INTERNATIONAL SEARCH REPORT

Inte Application No
PCT/US 00/42018

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
X	WO 97 48212 A (NOKIA TELECOMMUNICATIONS OY ;KARI HANNU H (FI)) 18 December 1997 (1997-12-18)	1-3,23, 25,36
Y	page 3, line 28 -page 8, line 22; figures 2A,2B	5,6,10, 11,13, 14,30
A		4,7-9, 12, 16-22, 24, 26-29, 31-35,37
X	EP 0 493 130 A (CANON KK) 1 July 1992 (1992-07-01)	1,2
A	page 11, line 35 -page 12, line 7 figure 16	8,9,19, 21,26, 27,33,34
E	US 6 195 024 B1 (FALLON JAMES J) 27 February 2001 (2001-02-27) the whole document	1-37
A	COENE W ET AL: "A FAST ROUTE FOR APPLICATION OF RATE-DISTORTION OPTIMAL QUANTIZATION IN AN MPEG VIDEO ENCODER" PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP),US,NEW YORK, IEEE, 16 September 1996 (1996-09-16), pages 825-828, XP000733350 ISBN: 0-7803-3259-8 the whole document	16-22, 31-35

INTERNATIONAL SEARCH REPORT

mation on patent family members

Inte | Application No
PCT/US 00/42018

Patent document cited in search report		Publication date	Patent family member(s)			Publication date
EP 0405572	A	02-01-1991	CA	2020084	A,C	30-12-1990
			DE	69014440	D	12-01-1995
			DE	69014440	T	11-05-1995
			JP	2046130	C	25-04-1996
			JP	3108824	A	09-05-1991
			JP	7073249	B	02-08-1995
			US	5091955	A	25-02-1992
WO 9748212	A	18-12-1997	FI	962381	A	08-12-1997
			AU	2965697	A	07-01-1998
			EP	0898825	A	03-03-1999
			JP	2000513519	T	10-10-2000
EP 0493130	A	01-07-1992	JP	4315369	A	06-11-1992
			JP	3082957	B	04-09-2000
			JP	4315370	A	06-11-1992
			JP	3143487	B	07-03-2001
			JP	4317264	A	09-11-1992
			JP	3015112	B	06-03-2000
			JP	4233373	A	21-08-1992
			DE	69126512	D	17-07-1997
			DE	69126512	T	06-11-1997
			DE	69132063	D	20-04-2000
			DE	69132063	T	14-09-2000
			EP	0763925	A	19-03-1997
			ES	2101730	T	16-07-1997
			ES	2143137	T	01-05-2000
			SG	72650	A	23-05-2000
			US	5838826	A	17-11-1998
US 6195024	B	27-02-2001	NONE			