PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : H03M 7/48	A1	(11) International Publication Number: WO 00/36754
		(43) International Publication Date: 22 June 2000 (22.06.00)

(21) International Application Number: PCT/US99/29109

(22) International Filing Date: 7 December 1999 (07.12.99)

(30) Priority Data:
09/211,531 14 December 1998 (14.12.98) US

(71) Applicant: MICROSOFT CORPORATION [US/US]; Building 4, One Microsoft Way, Redmond, WA 98052-6399 (US).

(72) Inventors: CHEN, Wei-ge; 24635 S.E. 37th Street, Issaquah, WA 98029 (US). LEE, Ming-Chieh; 5558 166th Place S.E., Bellevue, WA 98006 (US).

(74) Agent: WIGHT, Stephen, A.; Klarquist, Sparkman, Campbell Leigh & Whinston, LLP, Suite 1600, One World Trade Center, 121 S.W. Salmon Street, Portland, OR 97204 (US).

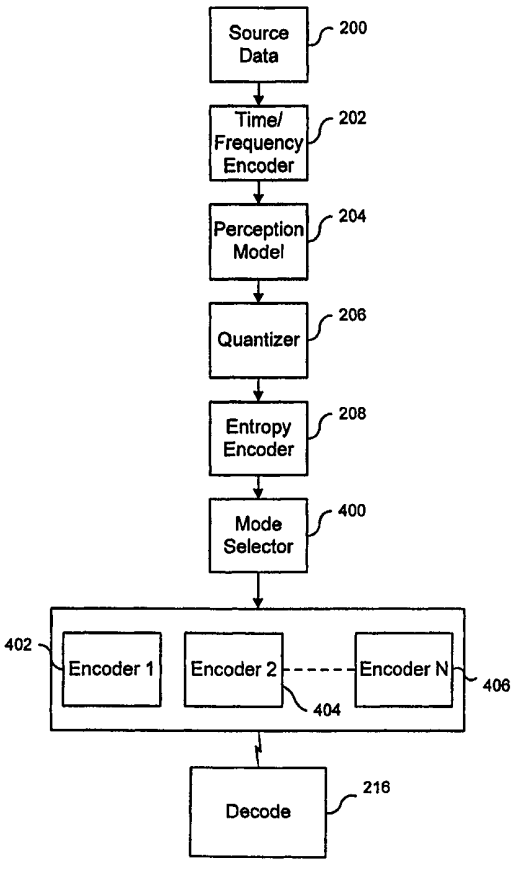
(81) Designated States: JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: ENTROPY CODE MODE SWITCHING FOR FREQUENCY-DOMAIN AUDIO CODING

(57) Abstract

A frequency-domain audio coder selects among different entropy coding modes according to characteristics of an input stream. In particular, the input stream is partitioned into frequency ranges according to some statistical criteria derived from a statistical analysis of typical or actual input to be encoded. Each range is assigned an entropy encoder optimized to encode that range's type of data. During encoding and decoding, a mode selector applies the correct entropy method to the different frequency ranges. Partition boundaries can be decided in advance, allowing the decoder to implicitly know which decoding method to apply to encoded data. Or, adaptive arrangements may be used, in which boundaries are flagged in the output stream by indicating a change in encoding mode for subsequent data. For example, one can create a partition boundary which separates out primarily zero quantized frequency coefficients, from primarily non-zero quantized coefficients, and then apply a coder optimized for such data. An overall more efficient process is achieved by basing coding methods according to the properties of the input data. In practice, the number of partitions and frequency ranges will vary according to the type of data to be encoded and decoded.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

Entropy Code Mode Switching for Frequency-domain Audio Coding

Field Of The Invention

The invention generally relates to frequency domain audio coding, and more specifically relates to entropy coding methods used in frequency domain audio encoders and decoders.

Background

In a typical audio coding environment, data is formatted, if necessary (e.g., from an analog format) into a long sequence of symbols which is input to an encoder. The input data is encoded by an encoder, transmitted over a communication channel (or simply stored), and decoded by a decoder. During encoding, the input is pre-processed, sampled, converted, compressed or otherwise manipulated into a form for transmission or storage. After transmission or storage, the decoder attempts to reconstruct the original input.

Audio coding techniques can be categorized into two classes, namely the time-domain techniques and frequency-domain ones. Time-domain techniques, e.g., ADPCM, LPC, operate directly in the time domain while the frequency-domain techniques transform the audio signals into the frequency domain where compression is performed. Frequency-domain codecs (compressors/decompressors) can be further separated into either sub-band or transform coders, although the distinction between the two is not always clear. That is, sub-band coders typically use bandpass filters to divide an input signal into a small number (e.g., four) of sub-bands, whereas transform coders typically have many sub-bands (and therefore a correspondingly large number of transform coefficients).

Processing an audio signal in the frequency domain is motivated by both classical signal processing theories and the human psychoaoustics model. Psychoacoustics take advantage of known properties of the listener in order to reduce information content. For example, the inner ear, specifically the basilar membrane, behaves like a spectral analyzer and transforms the audio signal into spectral data before further neural processing proceeds. Frequency-domain audio codecs often take advantage of auditory masking that is occurring in the human hearing system by modifying an original signal to eliminate information redundancies. Since human ears are incapable of perceiving these modifications, one can achieve efficient compression without distortion.

Masking analysis is usually conducted in conjunction with quantization so that quantization noise can be conveniently "masked." In modern audio coding techniques, the quantized spectral data are usually further compressed by applying entropy coding, e.g., Huffman coding. Compression is required because communication channels usually have limited available capacity or bandwidth. It is frequently necessary to reduce the information content of input data in order to allow it to be reliably transmitted, if at all, over the communication channel.

Tremendous effort has been invested in developing lossless and lossy compression techniques for reducing the size of data to transmit or store. One popular lossless technique is Huffman encoding, which is a particular form of entropy encoding. Entropy coding assigns code words to different input sequences, and stores all input sequences in a code book. The complexity of entropy encoding depends on the number *m* of possible values an input sequence X may take. For small *m*, there are few possible input combinations, and therefore the code book for the messages can be very small (e.g., only a few bits are needed to unambiguously represent all possible input sequences). For digital applications, the code alphabet is most likely a series of binary digits {0, 1}, and code word lengths are measured in bits.

If it is known that input is composed of symbols having equal probability of occurring, an optimal encoding is to use equal length code words. But, it is not typical that an input stream has equal probability of receiving any particular message. In practice, certain messages are more likely than others, and entropy encoders take advantage of such data correlation to minimize the average length of code words among expected inputs. Traditionally, however, fixed length input sequences are assigned variable length codes (or conversely, variable length sequences are assigned fixed length codes).

Summary

The invention relates to a method for selecting an entropy coding mode for frequency-domain audio coding. In particular, a given input stream representing audio input is partitioned into frequency ranges according to some statistical criteria derived from a statistical analysis of typical or actual input to be encoded. Each range is assigned an entropy encoder optimized to encode that range's type of data. During encoding and decoding, a mode selector applies the correct entropy method to the different frequency ranges. Partition boundaries can be decided in advance, allowing the decoder to implicitly know which decoding method to apply to encoded data. Or,

a forward adaptive arrangement may be used, in which boundaries are flagged in the output stream by indicating a change in encoding mode for subsequent data.

For natural sounds, such as speech and music, information content is concentrated in the low frequency range. This means that, statistically, the lower frequencies will have more non-zero energy values (after quantization), while the higher frequency range will have more zero values to reflect the lack of content in the higher frequencies. This statistical analysis can be used to define one or more partition boundaries separating lower and higher frequency ranges. For example, a single partition can be defined such that the lower 1/4 of the frequency components are below the partition. Alternatively, one can set the partition so that approximately one-half of the critical bands are in each defined frequency band. (Critical bands are frequency ranges of non-uniform width that correspond to the human auditory system's sensitivity to particular frequencies.) The result of such a division is to define two frequency ranges, in which one contains predominately non-zero frequency coefficients, while the other contains predominately zero frequency coefficients. Advance knowledge that the ranges containing predominately zero and non-zero values can be encoded with encoders optimized for such zero and non-zero values.

In one implementation, the range containing predominately zero values is encoded with a multi-level run-length encoder (RLE), i.e., an encoder that statistically correlates sequences of zero values with one or more non-zero symbols and assigns variable length code words to arbitrarily long input sequences of such zero and non-zero values. Similarly, the range containing mostly non-zero values is encoded with a variable-to-variable entropy encoder, where a variable length code word is assigned to arbitrarily long input sequences of quantization symbols. An overall more efficient process is achieved by basing coding methods according to the properties of the input data. In practice, the number of partitions and frequency ranges will vary according to the type of data to be encoded and decoded.

Brief Description of the Drawings

FIG. 1 is a block diagram of a computer system that may be used to implement frequency domain audio coding and decoding that employs entropy code mode switching.

FIG. 2 is a flow chart showing encoding and decoding audio data in a frequency domain audio coder.

FIG. 3 illustrates a frequency range divided according to audio

FIG. 4 illustrates an that employs entropy coding mode switching.

FIG. 5 is a flowchart showing creation of a code book having variable length entries for variable length symbol groupings.

FIGS. 6-12 illustrate creation of a code book pursuant to FIG. 5 for an alphabet {A, B, C}.

FIG. 13 illustrates a spectral threshold grid used in encoding audio sequences having repeating spectral coefficients.

FIG. 14 illustrates implementing the FIG. 2 entropy encoder.

Detailed Description

Exemplary Operating Environment

FIG. 1 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. While the invention will be described in the general context of computer-executable instructions of a computer program that runs on a personal computer, those skilled in the art will recognize that the invention also may be implemented in combination with other program modules. Generally, program modules include routines, programs, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers, and the like. The illustrated embodiment of the invention also is practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. But, some embodiments of the invention can be practiced on stand alone computers. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

With reference to FIG. 1, an exemplary system for implementing the invention includes a computer 20, including a processing unit 21, a system memory 22, and a system bus 23 that couples various system components including the system memory to the processing unit 21. The processing unit may be any of various commercially available processors, including Intel x86, Pentium and compatible microprocessors from Intel and others, the Alpha processor by Digital, and the PowerPC from IBM and Motorola. Dual microprocessors and other multi-processor architectures also can be used as the processing unit 21.

The system bus may be any of several types of bus structure including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of conventional bus architectures such as PCI, AGP, VESA, Microchannel, ISA and EISA, to name a few. The system memory includes read only memory (ROM) 24 and random access memory (RAM) 25. A basic input/output system (BIOS), containing the basic routines that help to transfer information between elements within the computer 20, such as during start-up, is stored in ROM 24.

The computer 20 further includes a hard disk drive 27, a magnetic disk drive 28, e.g., to read from or write to a removable disk 29, and an optical disk drive 30, e.g., for reading a CD-ROM disk 31 or to read from or write to other optical media. The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic disk drive interface 33, and an optical drive interface 34, respectively. The drives and their associated computer-readable media provide nonvolatile storage of data, data structures, computer-executable instructions, etc. for the computer 20. Although the description of computer-readable media above refers to a hard disk, a removable magnetic disk and a CD, it should be appreciated by those skilled in the art that other types of media which are readable by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, and the like, may also be used in the exemplary operating environment.

A number of program modules may be stored in the drives and RAM 25, including an operating system 35, one or more application programs (e.g., Internet browser software) 36, other program modules 37, and program data 38.

A user may enter commands and information into the computer 20 through a keyboard 40 and pointing device, such as a mouse 42. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 21 through a serial port interface 46 that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, game port or a universal serial bus (USB). A monitor 47 or other type of display device is also connected to the system bus 23 via an interface, such as a video adapter 48. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers and printers.

The computer 20 is expected to operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 49. The remote computer 49 may be a web server, a router, a peer device

or other common network node, and typically includes many or all of the elements described relative to the computer 20, although only a memory storage device 50 has been illustrated in FIG. 1. The computer 20 can contact the remote computer 49 over an Internet connection established through a Gateway 55 (e.g., a router, dedicated-line, or other network link), a modem 54 link, or by an intra-office local area network (LAN) 51 or wide area network (WAN) 52. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

In accordance with the practices of persons skilled in the art of computer programming, the present invention is described below with reference to acts and symbolic representations of operations that are performed by the computer 20, unless indicated otherwise. Such acts and operations are sometimes referred to as being computer-executed. It will be appreciated that the acts and symbolically represented operations include the manipulation by the processing unit 21 of electrical signals representing data bits which causes a resulting transformation or reduction of the electrical signal representation, and the maintenance of data bits at memory locations in the memory system (including the system memory 22, hard drive 27, floppy disks 29, and CD-ROM 31) to thereby reconfigure or otherwise alter the computer system's operation, as well as other processing of signals. The memory locations where data bits are maintained are physical locations that have particular electrical, magnetic, or optical properties corresponding to the data bits.

FIG. 2 shows a transmission model for transmitting audio data over a channel 210. The source of the transmission may be a live broadcast, stored data, or information retrieved over wired / wireless communication link (e.g., a LAN or the Internet). It is presumed that the channel 210 is of limited bandwidth, and therefore compression of source data 200 is desirable before data can be reliably sent over the channel. Note that although this discussion focuses on transmission of audio data, the invention applies to transfer of other data, such as audio visual information having embedded audio data (e.g., multiplexed within an MPEG data stream), or other data sources having compressible data patterns (e.g., coherent data).

As illustrated, source data 200 is input to a time / frequency transform encoder 202 such as a filter bank or discrete-cosine type transform. Transform encoder 202 is designed so as to convert a continuous or sampled time-domain input, such as an audio data source, into multiple frequency bands of predetermined (although perhaps differing) bandwidth. These bands can then be analyzed with respect to a human auditory perception model 204 (for example, a psychoacoustic

model) in order to determine components of the signal that may be safely reduced without audible impact. For example, it is well known that certain frequencies are inaudible when certain other sounds or frequencies are present in the input signal (simultaneous masking). Consequently, such inaudible signals can be safely removed from the input signal. Use of human auditory models is well known, e.g., the MPEG 1, 2 and 4 standards. (Note that such models may be combined into a quantization 206 operation.)

After performing the time/frequency transformation 202, frequency coefficients within each range are quantized 206 to convert each coefficient (amplitude levels) to a value taken from a finite set of possible values, where each value has a size based on the bits allocated to representing the frequency range. The quantizer may be a conventional uniform or non-uniform quantizer, such as a midriser or midtreader quantizer with (or without) memory. The general quantization goal is identifying an optimum bit allocation for representing the input signal data, i.e., to distribute usage of available encoding bits to ensure encoding the (acoustically) significant portions of the source data. Various quantization methods, such as quantization step size prediction to meet a desired bit rate (assuming constant bit rate) can be used. After the source 200 has been quantized, the resultant data is then entropy encoded 208 (see discussion for FIGS. 6-13).

The entropy encoded output is transmitted over the communication channel 210 (or stored for later transmission). The receiving end 216 then implements a reverse-encoding process, i.e., a series of steps to undo the encoding of the source data 200. That is, encoded data is received over the channel 210 as input to an entropy decoder 212 which performs a reverse code book look-up to convert the encoded output into an approximation of the original quantization output for the input symbol series 200. This approximate data is then processed by a de-quantizer 214 and a time / frequency transform decoder 218 to reverse the original coding operations, resulting in a reconstructed data 220 that is similar to the original source data 200. It should be noted that the reconstructed data 220 only approximates the original source data 200 since applying steps 204-208 is a lossy process.

One possible implementation for this transmission model is a client application program wanting to process, display or play real-time data as it is retrieved over a network link from a server / serving application. For example, the client can use a streaming delivery system that provides adaptive bandwidth reservation. (One such streaming format is the Microsoft Advanced Streaming Format.) A streaming environment contrasts traditional networking programs by allowing data delivery to be

optimized for particular retrieval needs, such as line speed constraints. A distinguishing feature of streaming data is that data can be viewed progressively in real time as a client receives it. Note that it is intended that processed data can be stored for later retrieval by a client, and that such retrieval can be performed in a non-streaming format (e.g., by a small playback appliance).

The streaming format defines the structure of synchronized object data streams, and allows any object, e.g., audio and video data objects, scripts, ActiveX controls, and HTML documents, to be placed into a data stream. An Application Programming Interface (one such API is the Microsoft Audio Compression Manager) is provided to facilitate application support for the streaming format. Transmission of streaming format data over the communication channel 210 requires that the source information be converted into a form suitable for the network. But, unlike traditional packets which contain routing information and data, streaming packets contain a prioritized mix of data from different objects within the stream, so that the bandwidth can be first allocated to higher priority objects. On the receiving end 216, the objects within the prioritized data stream are reconstructed for use by the receiver.

Because data is probably being used as it is received, streaming content is susceptible to transmission delays. If data does not arrive reliably, or if transmission speed falls below an acceptable minimum, the data might become unusable (e.g., playback of a video sequence may fail). Consequently, bandwidth intensive data (such as audio feeds) needs significant compression to ensure its bandwidth requirements can be met by the communication channel 210. As the degree of lossy compression necessarily impacts the quality of the reproduced signal, a server should provide selectable encodings for different client network connection speeds (or use an adaptive feedback system to discern real-time throughput).

A particularly effective method for encoding the frequency coefficients source data 200 to ensure reliable transmission over the communication channel 210 is entropy encoding. As discussed below, one can capitalize on the data coherency by applying different encoding methods optimized for different parts of the input data. Entropy encoding is effective when symbols have non-uniform probability distribution. Entropy coding methods that group many input symbols, such as the variable-to-variable and RLE coders discussed below, are good at capitalizing on data coherency. Using different encoding methods for different frequency ranges allows for more-optimal encoding when the encoders are tailored to probability distributions for each such range.

FIG. 3 illustrates a time domain signal that has been converted to the frequency domain. Along the X axis is a range 300 of frequencies from zero 302 through a maximum frequency 304. A partition 306 has been defined within the range 300, where the partition is determined according to statistical analysis of an expected input stream (e.g., statistical information obtained while training an entropy code book, or by adaptive analysis of the actual input), and this statistical model is applied against actual input 308 for encoding.

One approach to setting a partition is, as discussed above, is placing a certain percentage of frequencies or critical bands below the boundary.

An alternate method is to collect basic statistics, such as the probability of zeros and non-zeros according to the probability distributions for each frequency. Inspection of each frequency's statistics shows a gradual change across frequencies, and a partition boundary can be selected so that distributions within each partition are similar. Note that the frequency partition is sensitive to the sampling rate of the input and the expected bit rate. The sampling rate and the bit rate determine the stochastic property of the quantized frequency coefficients, and this property is basically responsible for determining partition placement.

A more optimal method is to adaptively locate a partition by performing an exhaustive search to determine an "optimal" boundary location. That is, an optimal solution is to try every frequency (or subset thereof) as a partition location, perform entropy coding, and track which boundary potential position yielded a minimum number of bits for encoding. Although computationally more intensive, if computation costs are at issue, the compression benefits of an exhaustive search (or near exhaustive if frequency subsets are used) can outweigh costs when multiple partitions are used.

By separating out the frequency spectrum 300 into separate frequency sub-ranges 310, 312, an encoder can apply different encoding schemes that have been optimized to encode the different frequency ranges. This contrasts previous methods, such as entropy encoding schemes that substituted different entropy coding tables according to characteristics of data to be encoded. Such prior methods are limited by the flexibility of their single entropy encoding algorithm, by the inability of an encoding table to account for different kinds of input data, and by the overhead associated with identifying when different tables should be used. A method optimized for one type of data can not be efficiently applied to a different type of data.

In the illustrated embodiment, the selected dividing criteria for the range 300 is the probability ***C(F)*** (Y-axis) that a particular spectral event is a run of

coefficients at or near a particular intensity (e.g., zero). As with code book generation, the probability of receiving zero value data can be pre-computed with respect to exemplary input. As illustrated, the input signal 308 has high probability of being zero after the indicated partition 306. (The position of the partition divider 306 was chosen so that 80% or 90% of the input beyond the divider would be at or near zero.)

It is assumed that, at a minimum, an input signal 308 is divided into two ranges, each range having data characteristics best-suited to compression by different encoding methods. In the illustrated embodiment, one range has primarily zero values, while the other has primarily non-zero values. Thus, two encoders are used, each optimized for the type of data within its corresponding range. While the illustrated implementation partitions the frequency coefficients into two ranges, more than two ranges can be defined, each having its own optimized encoder, or different ranges can share similar characteristics and thus utilize the same encoder.

For encoding the mostly non-zero range 310, an entropy coder such as that discussed for FIGS. 6-13 may be used. As discussed below, the FIGS. 6-13 coding method is particularly well suited to encoding non-zero auditory input data. For the mostly zero-value range 312, an encoder optimized for such data is used. In the illustrated embodiment, a run length encoder is used as it is optimized for encoding data that has a predominate value (e.g., zero). FIG. 13 illustrates one RLE-based entropy encoder that can efficiently encode the mostly zero valued range 312.

FIG. 4 illustrates a transmission model for transmitting audio data over a channel (see FIG. 2), in which multiple entropy encoding / decoding methods are used to manipulate input data 200. It is known that the source audio data 200 will have values within some frequency range. As discussed above for FIG. 2, source data 200 may be converted 202 into the frequency domain, reduced according to psycho-acoustic models 204, and quantized 206. Since quantization may produce significant numbers of near zero output values, an entropy encoder 208 can be optimized to encode this quantization output.

After quantization, the spectral coefficients for the quantized data tend to track the information content of typical audio data. Analysis of the quantization coefficients shows they are most likely non-zero at lower frequency ranges, and mostly zero coefficients at higher frequencies. Therefore, for frequency partitions located at certain frequency positions, a mode selector 400 can determines which encoder to according to the frequency range being encoded.

Determining placement of the partition can be based on a statistical analysis identifying which of several known entropy encoders will achieve better

coding efficiency for different sub-ranges. In one configuration, analysis is performed in advance of encoding or decoding with respect to exemplary input. This allows for pre-determination of partition locations, and corresponding encoders for each sub-range, so that no overhead needs to be introduced to flag changes in applicable coders.

Alternatively, statistical analysis may be performed on current data (in real time or off-line). In this configuration, although the encoders / decoders are known in advance, a flag needs to be embedded into the encoded data stream to indicate changes in applicable coders. As discussed above, different potential partition locations can be tried until a certain degree of coding efficiency is achieved for each sub-range. Receipt by a decoder of the flag indicates the end of a sub-range, and the value of the flag indicates which decoder to use for successive data.

Although inserting markers adds some overhead to the encoding / decoding process, such markers represent an improvement over prior-art encoding methods. For example, compare illustrated embodiments with traditional (see, e.g., MPEG 1, 2, and 4) entropy encoding of audio data. A traditional system uses a single entropy encoder for all data, where different code books are associated with each of many critical bands in the input data's frequency range (usually 24 or more bands, depending on the sampling rate). At each critical band transition, assuming 24 bands, a 2 bit (or longer) flag is required to indicate which of 24 code books are to be used to encode the band's data. (5 bits are required to track 24 states, but this flag can itself be encoded into effectively fewer bits.) This sharply contrasts the illustrated embodiments which either require no flag at all, or which uses flags, but is more efficient over prior methods unless the number of sub-ranges becomes comparable to the number of critical bands, and the number of encoding methods approaches the number of tables. That is, in every encoding using critical bands, there will be 24 sub-ranges requiring a 2-5 bit flag to indicate which encoding table to use. In contrast, illustrated embodiments may only have 2 or three sub-ranges, thus much less overhead.

As shown, there are *N* pre-defined encoders 402-406, each optimized to encode a frequency range having data with some predominate characteristic. This does not mean that there are necessarily N distinct input ranges, as different frequency ranges may have similar statistical characteristics for its data, and hence use the same encoder. In the illustrated example, there are only two ranges (one partition), corresponding to low (mostly non-zero coefficients) and high (mostly zero coefficients) frequency ranges. Hence, the mostly zero data past the partition is encoded with an

RLE type encoder (see, e.g., FIG. 13), and the data before the partition is encoded with a variable-to-variable entropy-type entropy encoder.

In the general case, however, once statistical information is available for a particular input, different encoders may be selected according to whichever encoder is best able to compress an input. For example, encoding methods, such as traditional Huffman encoding, vector Huffman variants, RLE encoding, etc., can be optimized and their code books trained for input having certain characteristics such as high spectral values, low spectral values, mixed or alternating spectral values, or some other desired / probable feature. In contrast with prior use of a single encoder for all input, illustrated configurations match different encoding methods according to a best match between a statistical profile for an input and the statistical profile for data on which an encoder code book was trained.

After determining which encoder 402-406 to use, processing continues as discussed with respect FIG. 2 for transmitting data to a receiver 216 for decoding. Note that an inverse mode selector is not shown. A mode switcher is necessary (e.g., as part of the FIG. 2 decoder 212) to properly select an appropriate decoder to reverse the work of the mode selector 400. However, as discussed above, range divider locations can be determined in advance, thus leaving their identification implied during decoding. Or, for dynamic adaptive encoding / decoding, embedded flags may be used to trigger decoder selection. Using flags is equivalent to using a mode selector, and the mode selector can be designed to operate for both pre-determined and adaptively located partitions.

FIG. 5 is a flowchart showing a preferred method for generating an entropy encoder's code book for input having a high probability of non-zero frequency coefficients. In particular, and in contrast with prior art techniques, FIG. 5 illustrated creating a code having variable length code assignments for variable length symbol groupings. (Prior art techniques either require fixed-length codes or fixed-length blocks of input.) Preferred implementations overcome the resource requirements of large dimension vector encoding, and the inapplicability of coding into words of equal lengths, by providing an entropy based variable-to-variable code, where variable length code words are used to encode variable length X sequences. Resource requirements can be arbitrarily capped by setting a fixed maximum code book size. This code book is created as follows.

Let y_i represent each source symbol group $\{x_j\}$, for $1 <= j <= N_i$, having probability P_i of occurring within the input stream, and each group is assigned a corresponding code word having L_i bits. Assuming that each x_j is drawn from a fixed

alphabet of predetermined size, the objective is to minimize the equation L = $\sum_i \frac{L_i * P_i}{N_i}$.

Instead of finding a general solution to the problem, the problem is separated into two different tasks. The first task is identification of a (sub-optimal) grouping of a set of input symbols $\{x_i\}$ through an empirical approach described below. The second task is assigning a entropy-type code for the grouped symbols $\{y_i\}$. Note that it is known that if the source is not coherent (i.e., the input is independent or without memory), any grouping that has the same configuration of $\{N_i\}$ can achieve the same coding efficiency. In this situation, the first task becomes inconsequential.

To perform the first task, an initial trivial symbol grouping 500 is prepared, such as $\{y_i\} = \{x_i\}$. This initial configuration assumes that an exemplary input stream is being used to train creation of the code book. It is understood that a computer may be programmed with software constructions such as data structures to track receipt of each symbol from an input. Such data structures may be implemented as a binary-type tree structure, hash table, or some combination of the two. Other equivalent structures may also be used.

After determining the trivial grouping, the probability of occurrence for each y_i is computed 502. Such probability is determined with respect to any exemplary input used to train code book generation. As further symbols are added to the symbol data structure, the probabilities are dynamically adjusted.

Next, the most probable grouping y_i is identified 504 (denoted as y_{mp}). If 506 the highest probability symbol is a grouping of previously lower probability symbols, then the grouping is split 508 into its constituent symbols, and processing restarted from step 502. (Although symbols may be combined, the group retains memory of all symbols therein so that symbols can be extracted.)

If the symbol is not a grouping, then processing continues with step 510, in which the most probable grouping is then tentatively extended with single symbol extensions x_i's. Preferably y_{mp} is extended with each symbol from the X alphabet. However, a predictor can be used to only generate an extension set containing only probable extensions, if the alphabet is very large and it is known many extensions are unlikely. For example, such a predictor may be based on semantic or contextual meaning, so that very improbable extensions can be ignored a priori.

The probability for each tentative expansion of y_{mp} is then computed 512, and only the most probable extension retained 514. The rest of the lower probability extensions are collapsed together 516 as a combined grouping and stored in

code book with a special symbol (event) to indicate a combined grouping. This wild-card symbol represents any arbitrary symbol grouping having y_{mp} as a prefix, but with an extension (suffix) different from the most probable extension. That is, if $y_{mp} + x_{mp}$ is the most probable root and extension, then the other less probable extensions are represented as $y_{mp}*$, $* \neq x_{mp}$. (Note that this discussion presumes, for clarity, serial processing of single-symbol extensions; however, parallel execution of multiple symbol extensions is contemplated.)

It is understood by one skilled in the art that applying single symbol extensions, and keeping only one most probable grouping, are restrictions imposed for clarity of discussion. It is further understood that although discussion focuses on serial processing, code book construction may be paralleled.

Code book construction is completed by repeating 518 steps 502-516 until all possible extensions have been made, or the number of the code book entries reaches a predetermined limit. That is, repeating computing probabilities for each current y_i 502, where the code book set {Y} now includes $y_{mp} + x_{mp}$, and respectively choosing 504 and grouping the most and least likely extensions. The effect of repeatedly applying the above operations is to automatically collect symbol groupings having high correlation, so that inter-group correlation is minimized. This minimizes the numerator of L, while simultaneously maximizing the length of the most probable y_i so that the denominator of L is maximized.

FIGS. 6-13 illustrate creation of a code book pursuant to FIG. 5 for an exemplary alphabet {A, B, C}. For this discussion, the code book is defined with respect to an exemplary input stream "A A A B B A A C A B A B B A B". As discussed above, one or more exemplary inputs may be used to generate a code book that is then used by encoders and decoders to process arbitrary inputs. For clarity, the code book is presented as a tree structure, although it may in fact be implemented as a linear table, hash table, database, etc. As illustrated, the tree is oriented left-to-right, where the left column (e.g., "A" and "X0") represents the top row of a tree-type structure, and successively indented rows represent the "children" of the previous row's node (e.g., in a top-down tree for FIG. 7, top node "A" is a first-row parent node for a second-row middle-child node "B".).

In preparing the code cook, the general rule is to pick the most probable leaf node, expand it, re-compute probabilities to determine the most probable leaf-node, and then compact the remaining sibling nodes into a single Xn node (n=0..N, tracking each time nodes have been combined). If it turns out that the most probable node is a group node, then the group is split, probabilities recalculated, and the most

probable member node retained (i.e., the remaining group members are re-grouped). Processing cycles until a stop state is reached, such as a code book having predetermined size.

FIG. 6 shows an initial grouping for the input stream "A A A B B A A - C A B A B B A B". An initial parsing of the input gives probabilities of occurrence of A = 8/15, B = 6/15, and C = 1/15. This initial trivial grouping can be created based on different criteria, the simplest being having a first-level node for every character in the alphabet. However, if the input alphabet is large, the trivial grouping may be limited to some subset of symbols having highest probability, where the remaining symbols are combined into an X grouping. FIG. 6 illustrates this technique by starting with only two initial groups, group A 600 having probability 8/15, and group X0 602 having probability 7/15, where X0 represents all remaining low probability symbols in the alphabet, e.g., B and C.

After preparing an initial trivial grouping, the leaf-node having highest probability is selected for extension (see also FIG. 5 discussion regarding processing sequence). Hence, as shown in FIG. 7, group A 600 is tentatively expanded by each character in the alphabet (or one may limit the expansion to some subset thereof as described for creating the initial grouping). Probabilities are then recomputed with respect to the input stream "A A A B B A A C A B A B B A B" to determine values for the tentative extensions A 606, B 608, and C 610. The result is nine parsing groups, where "A A" appears 2/9, "A B" appears 4/9, and "A C" appears 0/9. Therefore, the most probable extension "A B" is retained and the other extensions collapsed into X1 = A,C. Note that although this discussion repeatedly recalculates all probabilities, a more efficient approach is to retain probabilities and symbol associations for each node within the node, and only computing information as necessary.

FIG. 8 shows the collapse into X1 612 for FIG. 7. Processing repeats with identification of the node having highest probability, e.g., node B 608 at probability 4/9.

As shown in FIG. 9, this node 608 is tentatively extended with symbols A 614, B 616, C 618, and as discussed above, the tentative grouping with highest probability is retained. After recalculating probabilities, the result is eight parsing groups in which the symbol sequence "A B A" 614 appears once, "A B B" 616 appears once, and "A B C" 618 does not appear at all. Since tentative extensions A 614 and B 616 have the same probability of occurrence, a rule needs to be defined to choose which symbol to retain. For this discussion, whenever there are equal probabilities, the highest row node (e.g., the left-most child node in a top-down tree) is

retained. Similarly, when there is a conflict between tree rows, the left-most row's node (e.g., the node closest to the root of a top-down tree) is retained.

Therefore, as shown in FIG. 10, node A 614 (FIG. 9) is retained and nodes B 616 and C 618 are combined into node X2=B,C 620, having combined probability of 1/8 + 0/8. Now, the next step is to expand the node currently having highest probability with respect to the input stream. As shown, nodes X1=A,C 612 and X0=B,C 602 have the same probability of occurrence (3/8). As discussed above, a default rule is used so that the highest node in the tree (X0 602) is extended. (Although it is only necessary to be consistent, it is also preferable to expand higher level nodes since this may increase coding efficiency by increasing the number of long code words.)

However, X0 602 is a combined node, so it must be split instead of extended. FIG. 11 illustrates the result of splitting node X0 into its constituent symbols B 622 and C 624. Recalculating probabilities indicates that symbol sequences "A B A" appears 1/8, "A B X2" appears 1/8, "A X1" appears 3/8, "B" 422 appears 2/8, and "C" appears 1/8. Since this is a split operation, the split node having highest probability, e.g., node B 622, is retained, and the remaining node(s) re-combined back into X0 602.

FIG. 12 shows the result of retaining high-probability node B 622. Note that grouping X0 602 now only represents a single symbol "C". After revising probabilities, the node having highest probability must be identified and split or extended. As shown, symbol sequence "A B A" appears 1/8, "A B X2" appears 1/8, "A X1" appears 3/8, "B" appears 2/8, and "X0" appears 1/8. Therefore node X1 612, as a combined node, must be split.

Splitting proceeds as discussed above, and processing the input cycles as discussed above in conjunction with FIG. 5, where highest probability nodes are extended or split until a stop state is reached (e.g., the code book reaches a maximum size). Once the code book has reached a stop state, it is available for encoding data to transmit over a communication channel.

FIG. 13 illustrates a threshold grid that can be used to modify the FIG. 5 method of code book generation. As discussed for FIGS. 3 and 4, encoding becomes more efficient when encoders can be tailored to process certain portions of the input data. In particular, when it is known that the encoding method will produce a significant number of repeating values, an entropy coder can be combined with RLE-type encoding to increase coding efficiency for data containing the repeated value.

In the illustrated embodiments, quantization of the input data introduces zero or near-zero spectral coefficients for significant portions of the frequency ranges for the input data. Consequently, rather than applying the same entropy coder used for the mostly non-zero data (e.g., the encoder and code book of FIG. 5), instead a RLE-based entropy coder is used.

To construct a code book for a RLE-based entropy encoder, let the absolute values of the non-zero spectral samples form an integer set $L_i = \{1, 2, 3, ..., L_n\}$ where L_n stands for any value that is greater than or equal to L_n. Let the run length of zero spectral samples in an input stream form another set $R_j = \{1, 2, 3, ..., R_m\}$ with R_m stands for any zero runs with length longer than or equal to R_m. Using this notation, we can represent an input spectrum with a string of input symbols defined as (R_i, L_j), which corresponds to R_i zero spectral samples followed by L_j (i.e., symbols encoded with the entropy encoder).

As described above for FIG. 5 et seq., the first step in constructing a code book is to collect the probability of all input events. Here, the input is adjusted with respect to defined thresholds, and therefore probability is determined for (R_i, L_j) for all $1 <= i <= n$ and $1 <= j <= m$. These probabilities are pictorially presented in FIG. 13, in which darker squares (e.g., 806, 808) correspond to events having higher probability, and lighter squares (e.g., 810, 812) have low or near zero probability. All high-probability input configurations are collectively referenced as range 800, and all low probability configurations as range 802. All low probability combinations are excluded from the code book. A probability threshold 804 is defined such that any value below the divider is set to zero and excluded from the code book. Remaining above-threshold configurations are assigned a entropy-type code having length inversely proportional to its probability. For quantized audio data, high amplitude inputs have low probability. Consequently, they fall below the threshold and are excluded from the code book (however, they can be escaped and placed in the encoded bit stream).

In order to interleave entropy coding output with use of a secondary encoder, a special entropy code book code is reserved to demark excluded events (e.g., RLE encoded data). At encoding time, spectral samples (input symbols) can be are compared to the list of possible events and if a match is found (e.g., if using a variable to variable encoder, in the tree, table, hash structure or equivalent used to represent the code book), the corresponding entropy-type code is output followed by a sign bit.

If a match is not found, the escape code is sent followed by necessary information to identify the event, i.e., information for the RLE encoding of the data. In the case of an input spectrum ending with N zeros, either an explicit (special) ending signal is needed or a special event such as (N, 1) suffices because the decoder is aware the total number of samples and able to stop decoding when that limit is exceeded.

For decoding, a threshold grid is not required, as the grid is used to cull code book entries. Decoding methods disclosed herein can be used along with a FIG. 13 code book generated as described.

FIG. 14 shows one method for implementing the entropy encoder 208 of FIG. 2 through application of a code book derived according to FIG. 5 to quantized data. (Note that the variable-to-variable encoding method is generally applicable for encoding other types of data.) As illustrated, the quantized data is received 900 as input to the entropy encoder of FIG. 2. It is understood that the input is in some form of discrete signals or data packets, and that for simplicity of discussion, all input is simply assumed to be a long series of discrete symbols. The received input 900 is scanned 902 in order to locate a corresponding code book key in the code book of FIG. 5. Such scanning corresponds to a data look-up, and depending on how the data structure used to implement the code book, the exact method of look-up will vary.

Note that there are various techniques available for storing and manipulating an encoder's code book. For example, one structure for a variable to variable code book is traversal and storage of a N-ary (e.g., binary, tertiary, etc.) tree, where symbol groupings guide a traversal of the tree structure. The path to a leaf node of the tree represents the end of a recognized symbol sequence, where a entropy-type code is associated with the sequence. (Note that the code cook may be implemented as a table, where a table entry contains the entire input sequence, e.g., the path to the node.) Nodes can be coded in software as a structure, class definition, or other structure allowing storage of a symbol or symbols associated with the node, and association of a corresponding entropy-type code 906.

Or, for the RLE encoder, its code book can be stored as a two-dimensional grid in permanent storage, where data retrieval is performed by identifying two indices. Thus, one can retrieve table entries by specification of a run-length and a particular symbol value. A decoding table can be implemented as a Huffman tree. Another code book implementation includes Rice-Golomb structures, and their equivalents.

Although not explicitly illustrated, as discussed with respect to FIG. 2, decoding operates as an inverse operation of encoding, where the encoded data 908 is looked up 906 in a decoding code book, in order to produce an approximation of the original input frequency coefficients 900.

Having described and illustrated the principles of my invention with reference to an illustrated embodiment, it will be recognized that the illustrated embodiment can be modified in arrangement and detail without departing from such principles. Accordingly, we claim as the invention all such modifications as may come within the scope and spirit of the following claims and equivalents thereto.

What is claimed is:

1. A method of encoding a sequence of audio data frequency coefficients with two or more different entropy encoders, the method comprising:

partitioning a frequency range for the sequence of audio data frequency coefficients into at least first and second sub-ranges, such partitioning made according to statistical analysis identifying which entropy encoder will achieve better coding efficiency for each sub-range;

encoding the first sub-range with a first entropy encoder; and

encoding the second sub-range with a second entropy encoder.

2. A method according to claim 1, wherein the statistical analysis is performed with respect to the sequence of audio data frequency coefficients.

3. A method according to claim 2, wherein statistical analysis is directed towards identifying at least two frequency ranges, a first frequency range having primarily non-zero spectral coefficients, and a second frequency range having repeating spectral coefficient intensities at or near a fixed value.

4. A method according to claim 2, wherein a run-length entropy encoder is used to encode data for the second frequency range.

5. A method according to claim 4, wherein a variable-to-variable entropy encoder is used to assign variable length entropy codes to arbitrarily long sequences of frequency coefficients.

6. A method according to claim 3, wherein the fixed value is zero, resulting in the second frequency range comprising primarily near-zero values.

7. A method according to claim 3, wherein the statistical analysis is performed in real time.

8. A method according to claim 1, wherein the statistical analysis is performed with respect to an exemplary sequence of audio data frequency coefficients.

9. A method according to claim 8, wherein statistical analysis is directed towards identifying at least two frequency ranges, a first range having primarily non-zero spectral coefficients, and a second range having repeating spectral coefficient intensities at or near a fixed value.

10. A method according to claim 8, wherein a run-length entropy encoder is used to encode data for the second frequency range.

11. A method according to claim 10, wherein a variable-to-variable entropy encoder is used to assign variable length entropy codes to arbitrarily long sequences of frequency coefficients.

12. A method according to claim 9, wherein the fixed value is zero, resulting in the second frequency range comprising primarily near-zero values.

13. A method according to claim 1, further comprising preparing a first code book for the first entropy encoder, and preparing a second code book for the second entropy encoder.

14. A computer readable medium having encoded thereon instructions for directing a computer to perform the steps of claim 1.

15. A method according to claim 1, wherein each encoder uses a code book generated according to frequency coefficients sharing a similar statistical profile.

16. A method of decoding a sequence of encoded audio frequency coefficients with two or more different entropy encoders, the method comprising:

receiving a coded audio input sequence having a frequency range partitioned into at least a first and a second sub-range, each sub-range having an associated entropy decoder; and

decoding data in each sub-range with the associated decoder.

17. A method of decoding according to claim 16, wherein range partitions are predetermined prior to encoding, and selection of an appropriate associated decoder is automatic according each frequency range being decoded.

18. A method of decoding according to claim 16, in which range partitions are not predetermined, and wherein a decoder-selection flag is embedded within the encoded audio frequency coefficients, such flag identifying a decoder to apply to subsequently received data.

19. A method according to claim 16, wherein the second entropy encoder is a run-length entropy encoder.

20. A computer readable medium having encoded thereon instructions for directing a computer to perform the steps of claim 16.

21. A method of entropy encoding a sequence of audio data frequency coefficients audio data symbols with at least two different entropy encoders, such data symbols having a minimum and a maximum amplitude, the method comprising:

preparing a first code book for the first entropy encoder;

preparing a second code book for the second entropy encoder according to optimizing for encoding repeating spectral coefficient intensities at or near a fixed value;

partitioning a frequency range for the sequence of audio data frequency coefficients into at least first and second sub-ranges, such partitioning made according to a statistical analysis identifying which entropy encoder will achieve better coding efficiency for each sub-range;

encoding each sub-range with an appropriate entropy encoder.

22. A method according to claim 21, wherein the second entropy encoder is a run-length entropy encoder.

23. A computer readable medium having encoded thereon instructions for directing a computer to perform the steps of claim 21.

24. A method of encoding a sequence of time-domain audio data symbols with two or more different entropy encoders into an output data stream, each encoder optimized for data sharing a particular value characteristic such as being non-zero, or being at or near a fixed intensity value, the method comprising:

converting the time-domain data symbols into frequency domain data;

reducing the frequency domain data according to a psychoacoustic model;

quantizing the reduced frequency domain data;

selecting an entropy encoder to apply to the quantized data; and

encoding the quantized data with the selected entropy encoder.

25. A method of encoding an arbitrarily long series of audio input symbols having spectral coefficients within a frequency range, the method comprising:

calculating the probability of spectral coefficients having near zero values;

partitioning the frequency range into a first and a second sub-range to group input symbols having primarily non-zero coefficients into the first sub-range, and input symbols having primarily zero coefficients into the second sub-range;

encoding the first range with a variable-to-variable entropy encoder; and

encoding the second range with a run-length entropy encoder.

26. A system for encoding an input signal with multiple encoding methods according to characteristics of the input signal, the system comprising:

an input for receiving a time-domain audio input signal;

a signal transformer for converting the time-domain audio signal to a frequency-domain audio signal;

a quantizer for converting the frequency-domain signal into quantized symbols; and

a mode selector for selecting an entropy encoder for the quantized symbols.

27. A system for encoding an input signal with multiple encoding methods according to characteristics of the input signal, the system comprising:

means for receiving a time-domain audio input signal;

means for converting the time-domain audio signal to a frequency-domain audio signal;

means for quantizing the frequency-domain signal into quantized symbols; and

means for selecting an entropy encoder for the quantized symbols.

28. A system according to claim 27, in which there are at least two different entropy encoders, the system further comprising:

means for partitioning the frequency domain signal into sub-ranges according to a probability that data within each such sub-range shares a certain characteristic; and

means for identifying which of the at least two different entropy encoders to apply to each sub-range.

FIG. 1

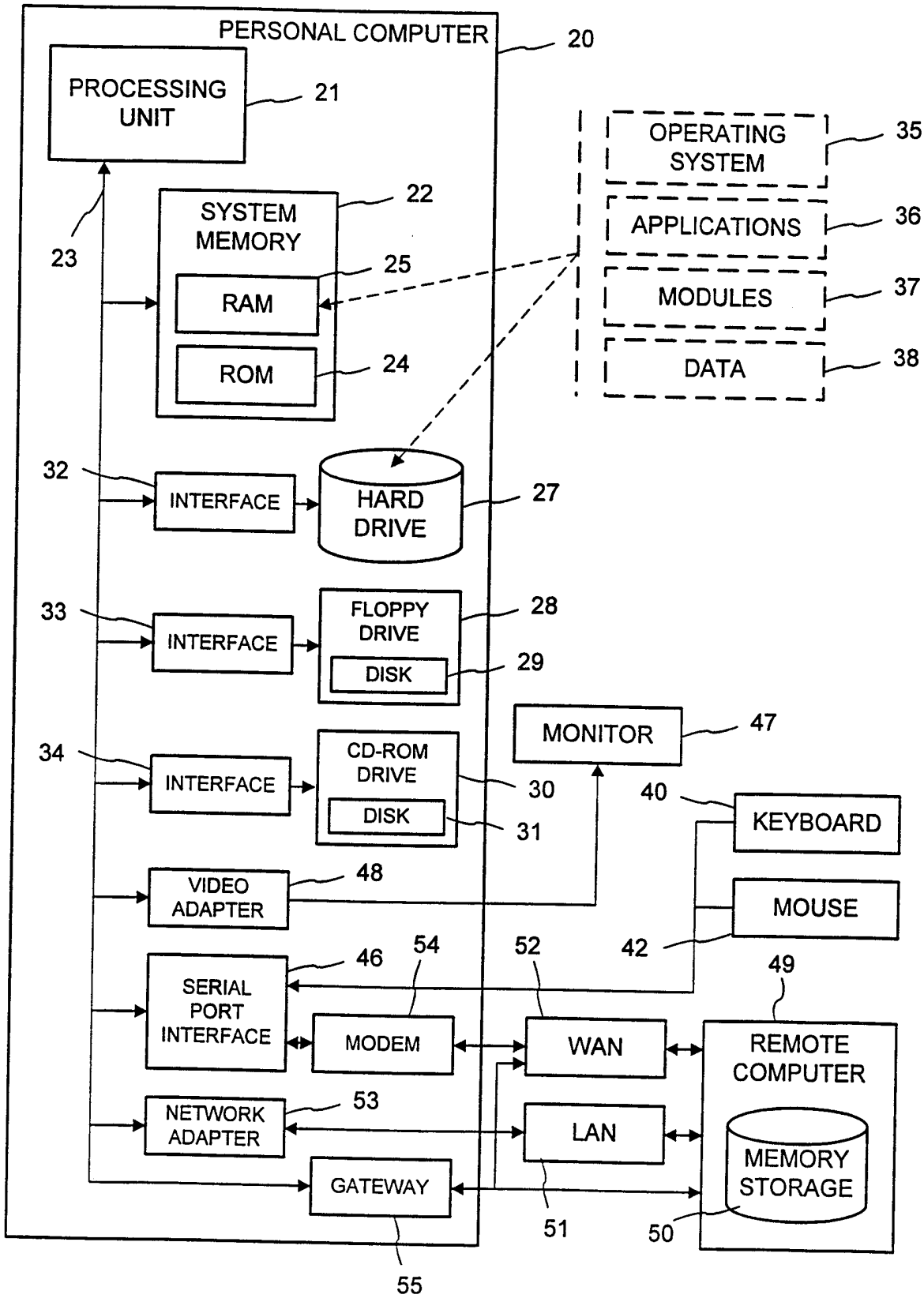

PERSONAL COMPUTER
20
PROCESSING UNIT
21
SYSTEM MEMORY
22
23
RAM
25
ROM
24
OPERATING SYSTEM
35
APPLICATIONS
36
MODULES
37
DATA
38
32
INTERFACE
HARD DRIVE
27
33
INTERFACE
FLOPPY DRIVE
28
DISK
29
34
INTERFACE
CD-ROM DRIVE
30
DISK
31
MONITOR
47
40
KEYBOARD
MOUSE
42
VIDEO ADAPTER
48
SERIAL PORT INTERFACE
46
54
52
49
MODEM
WAN
REMOTE COMPUTER
NETWORK ADAPTER
53
LAN
MEMORY STORAGE
GATEWAY
51
50
55

FIG. 2

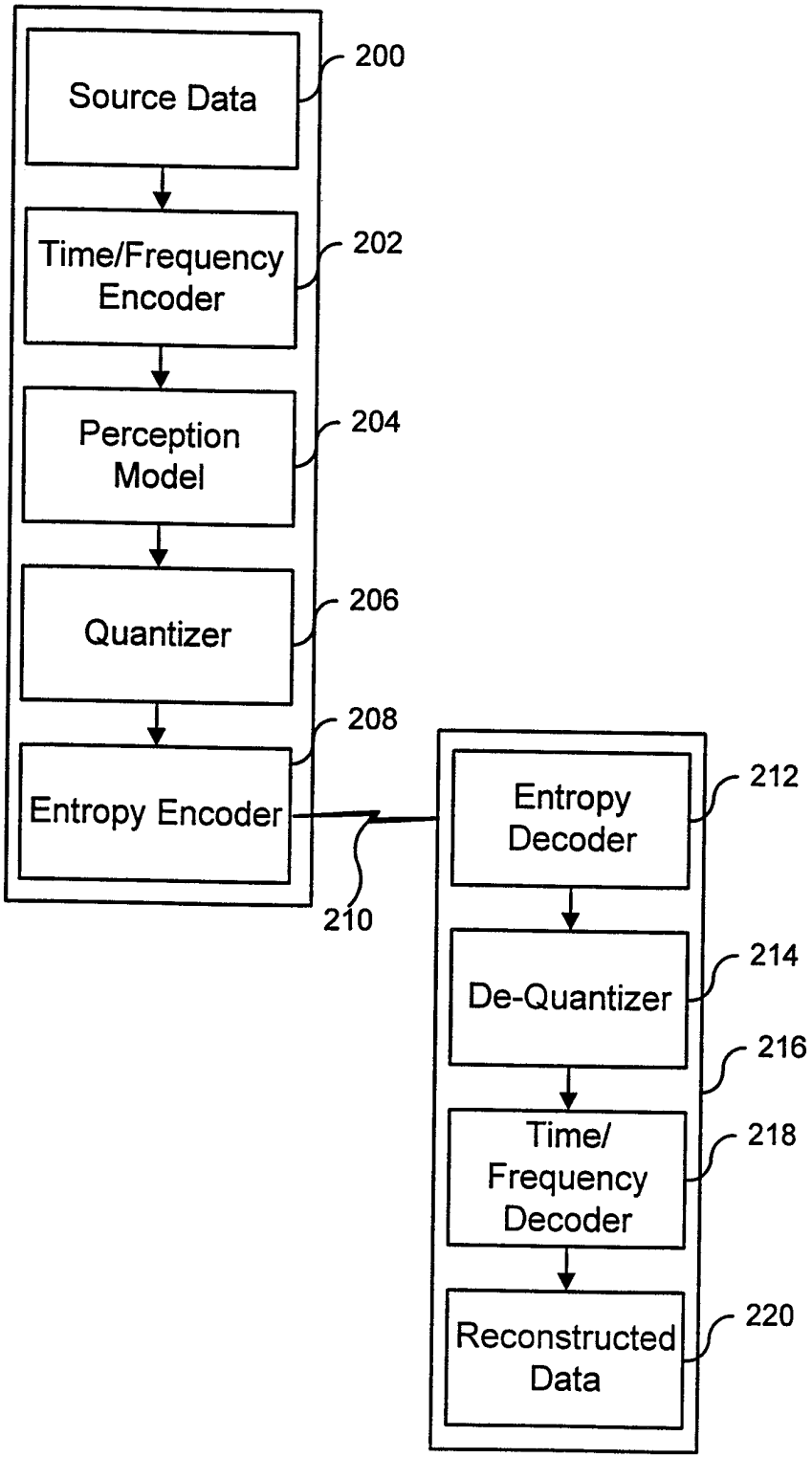

Source Data
200
Time/Frequency Encoder
202
Perception Model
204
Quantizer
206
Entropy Encoder
208
210
Entropy Decoder
212
De-Quantizer
214
216
Time/ Frequency Decoder
218
Reconstructed Data
220

FIG. 3

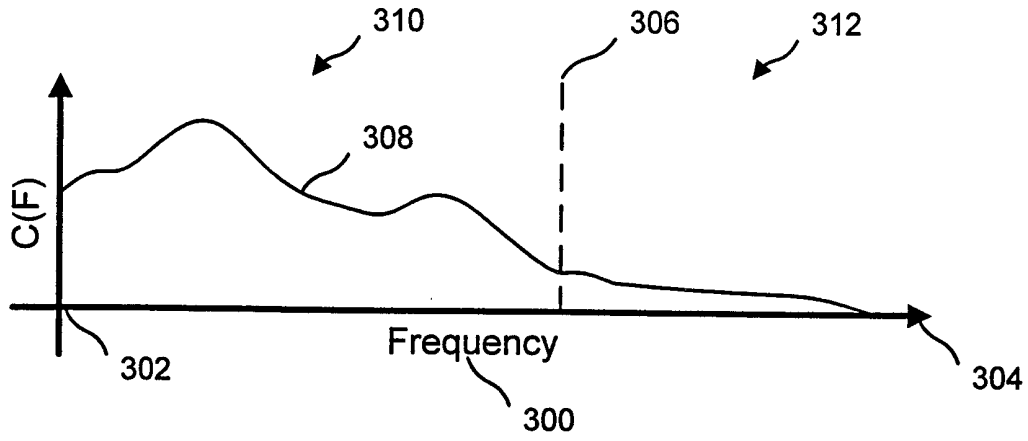


FIG. 4

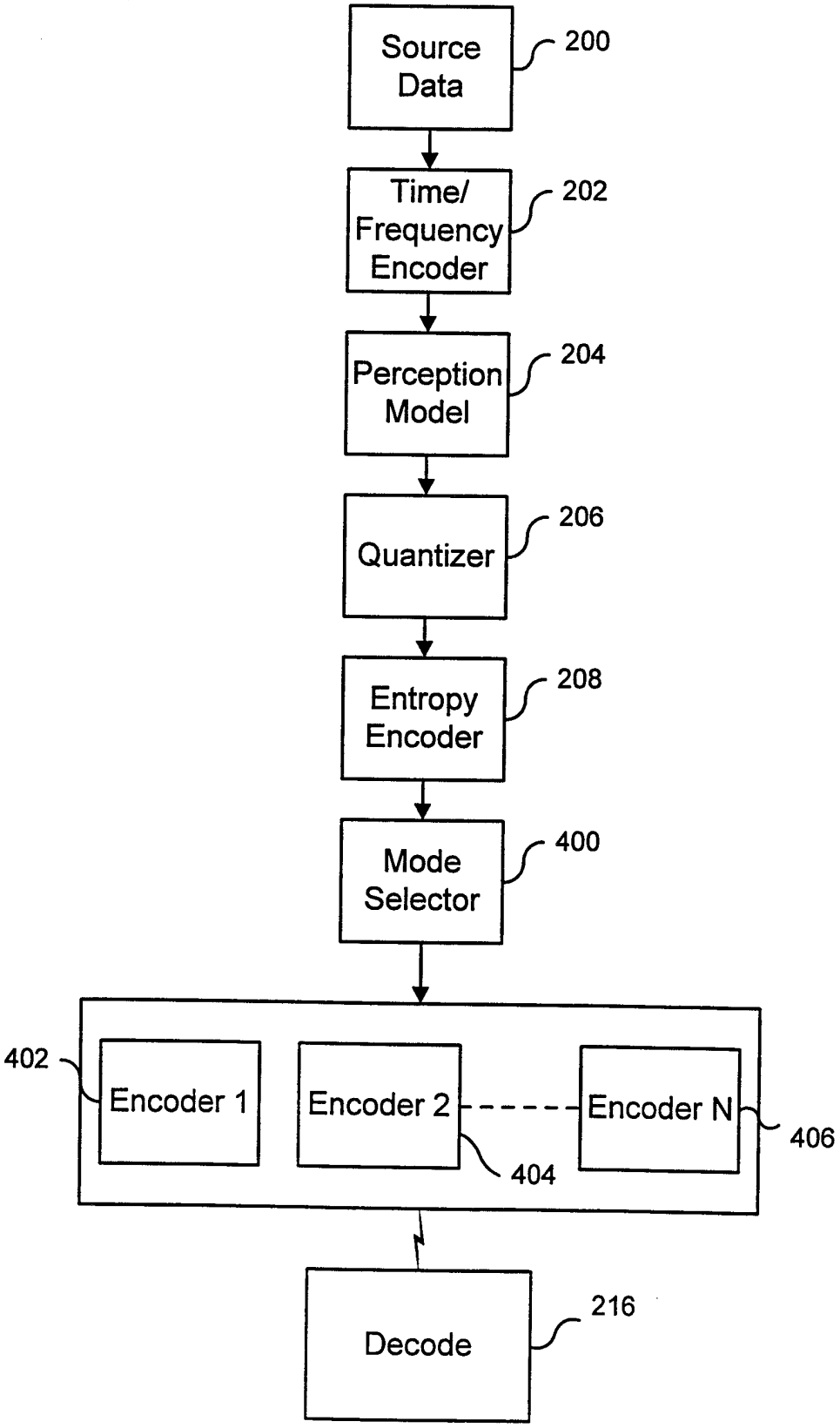

Source Data
200
Time/ Frequency Encoder
202
Perception Model
204
Quantizer
206
Entropy Encoder
208
Mode Selector
400
402
Encoder 1
Encoder 2
Encoder N
406
404
Decode
216

FIG. 5

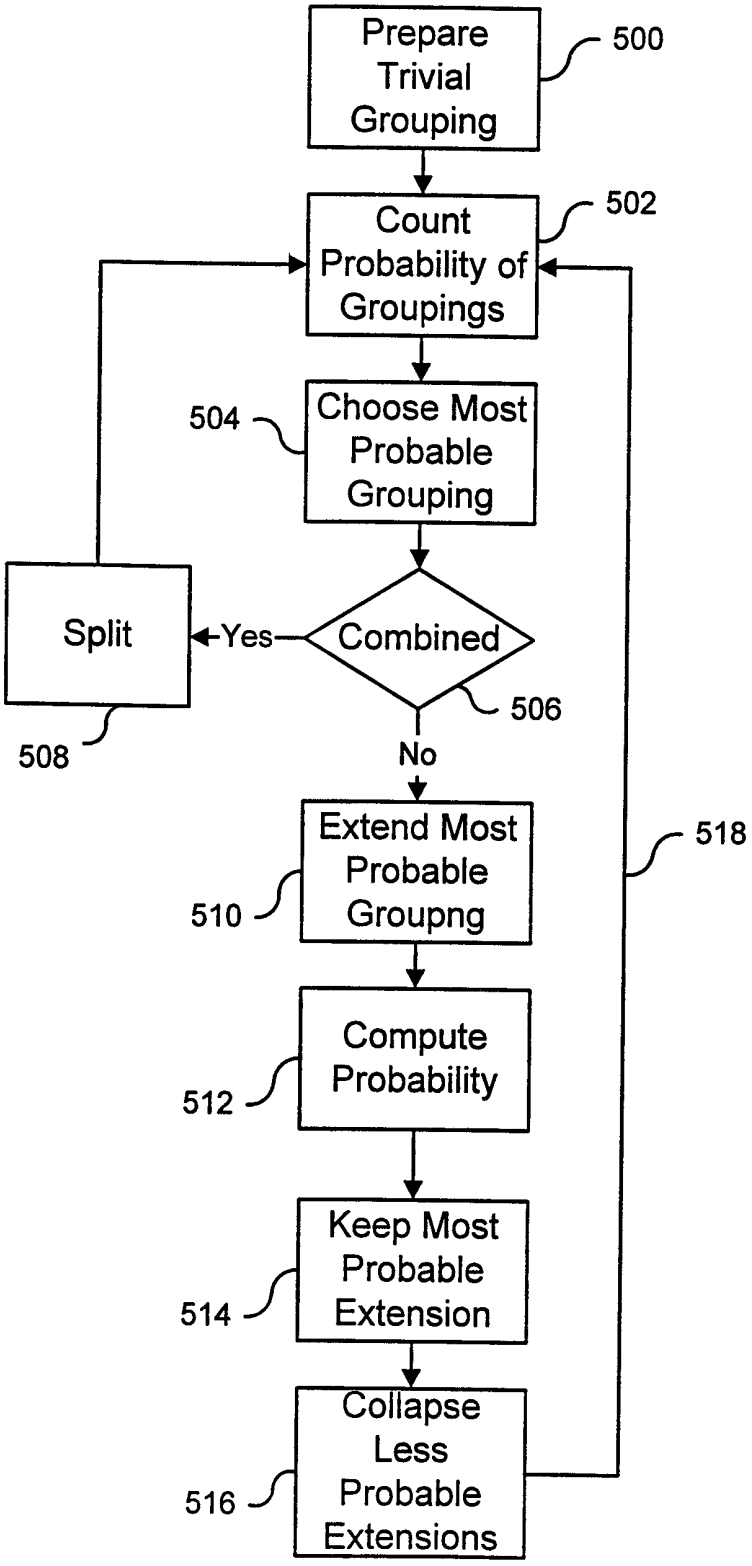
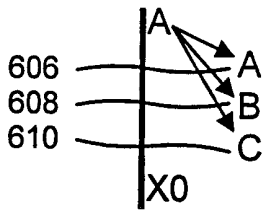

Prepare Trivial Grouping
500
Count Probability of Groupings
502
Choose Most Probable Grouping
504
Combined
Yes
Split
506
508
No
Extend Most Probable Groupng
510
518
Compute Probability
512
Keep Most Probable Extension
514
Collapse Less Probable Extensions
516

FIG. 6

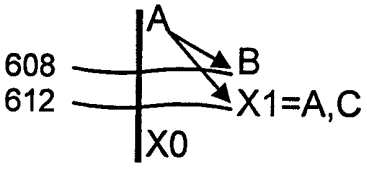
8/15 Expand
7/15

FIG. 7



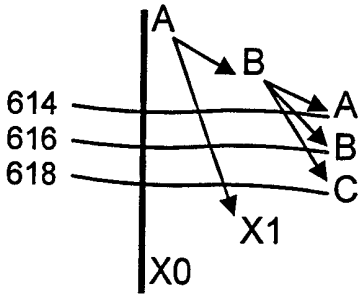
2/9
4/9 Keep
0/9

FIG. 8



4/9 Expand
2/9
3/9

FIG. 9



1/8 Keep
1/8
0/8
3/8
3/8

FIG. 10

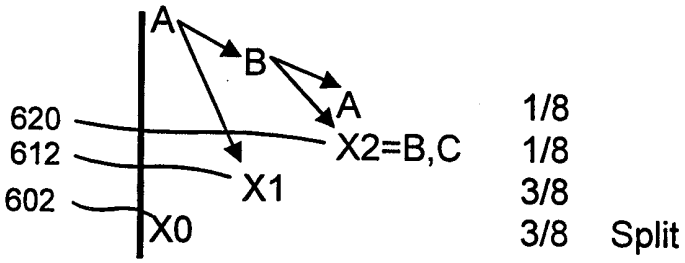


FIG. 11

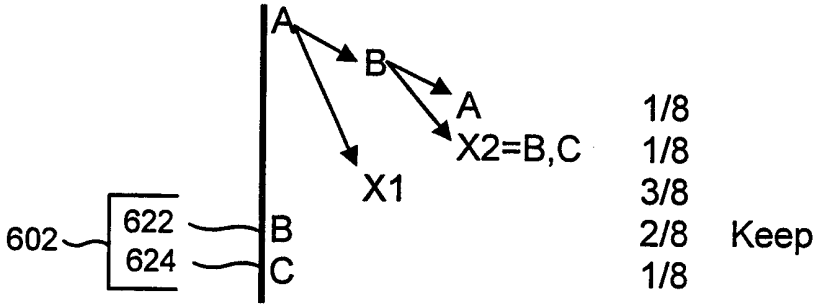


FIG. 12

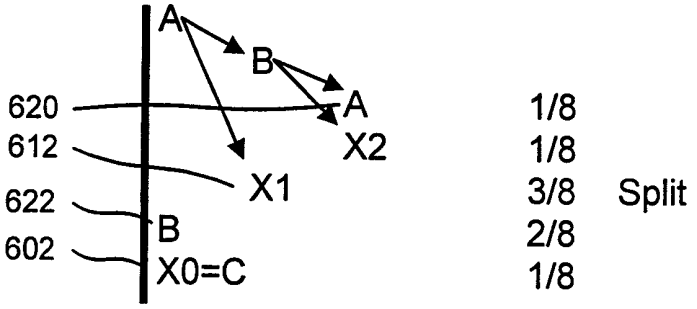


FIG. 13

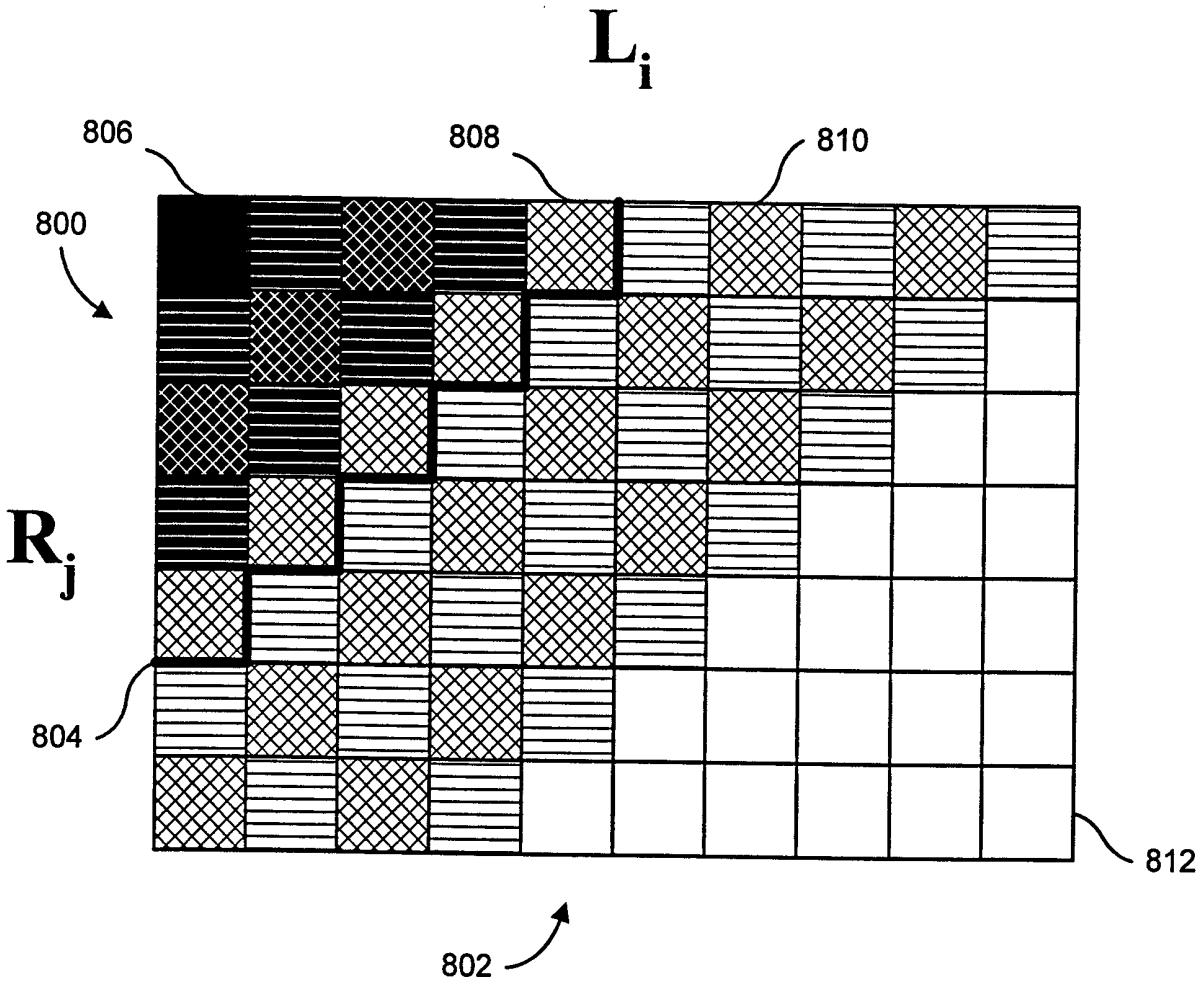
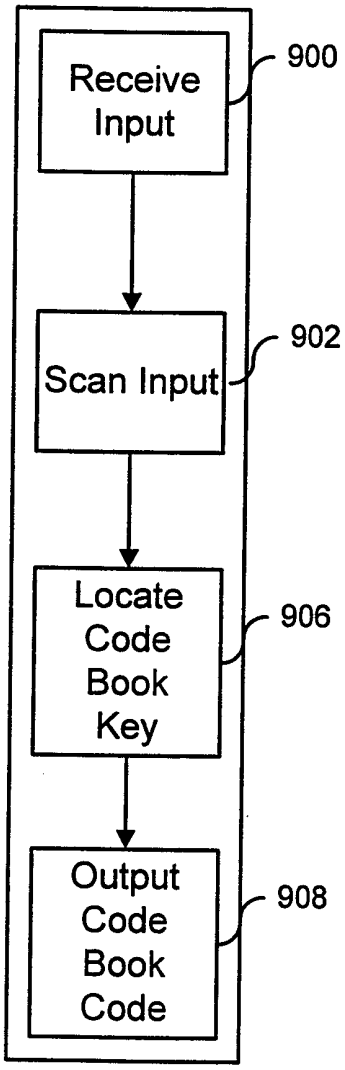


FIG. 14


Receive Input
900
Scan Input
902
Locate Code Book Key
906
Output Code Book Code
908

INTERNATIONAL SEARCH REPORT

Inter nal Application No
PCT/US 99/29109

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H03M7/48

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

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 INSPEC PAJ

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 612 156 A (FRAUNHOFER GES FORSCHUNG) 24 August 1994 (1994-08-24) page 2, line 21 -page 2, line 51 page 6, line 46 -page 7, line 28 claims 1,2,10-12	1,2, 13-16, 18,20, 21,23, 26-28
A	---	3-12,17, 19,22, 24,25
A	TEWFIK A H ET AL: "ENHANCED WAVELET BASED AUDIO CODER" PROCEEDINGS OF THE ASILOMAR CONFERENCE,US,NEW YORK, IEEE, 1993, pages 896-900, XP000438425 the whole document	1-28

	-/--	

[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	Date of mailing of the international search report
8 May 2000	17/05/2000

Name and mailing address of the ISA	Authorized officer
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	Fassnacht, C

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Inter ional Application No
PCT/US 99/29109

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 644 305 A (INOUE SADAYUKI ET AL) 1 July 1997 (1997-07-01) column 7, line 3 -column 7, line 54	1-28
P,X	US 5 884 269 A (CHENES PIERRE ET AL) 16 March 1999 (1999-03-16) column 1, line 60 -column 3, line 3 column 12, line 57 -column 13, line 33	1,2, 13-16, 18,20, 21,23, 26-28

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

information on patent family members

Inte onal Application No
PCT/US 99/29109

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0612156 A	24-08-1994	DE 3912605 A	25-10-1990
		AT 140571 T	15-08-1996
		AT 144090 T	15-10-1996
		WO 9013182 A	01-11-1990
		DE 59010419 D	22-08-1996
		DE 59010538 D	14-11-1996
		DK 393526 T	18-11-1996
		EP 0393526 A	24-10-1990
		EP 0717503 A	19-06-1996
		ES 2088918 T	01-10-1996
		GR 3021283 T	31-01-1997
		JP 2739377 B	15-04-1998
		JP 4504936 T	27-08-1992
		KR 136572 B	15-05-1998
		NO 913931 A	07-10-1991
		US 5579430 A	26-11-1996
US 5644305 A	01-07-1997	JP 8018459 A	19-01-1996
		US 5751232 A	12-05-1998
US 5884269 A	16-03-1999	NONE	