

(19)



(11)

EP 2 267 700 B1

(12)

EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention
of the grant of the patent:
21.12.2011 Bulletin 2011/51

(51) Int Cl.:
G10L 19/02 (2006.01)

(21) Application number: **09008170.4**

(22) Date of filing: **22.06.2009**

(54) **Apparatus and method for selecting quantisation bands in audio encoders**

Vorrichtung und Verfahren zum Auswählen von Quantisierungsbändern in Audiokodierern

Appareil et procédé de sélection de bandes de quantification dans des encodeurs audio

(84) Designated Contracting States:
**AT BE BG CH CY CZ DE DK EE ES FI FR GB GR
HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL
PT RO SE SI SK TR**

(43) Date of publication of application:
29.12.2010 Bulletin 2010/52

(73) Proprietor: **APT Licensing Limited**
Belfast
BT12 7FP (GB)

(72) Inventor: **Trainor, David**
Carryduff
Belfast, BT8 8B (GB)

(74) Representative: **Wallace, Alan Hutchinson et al**
FRKelly
4 Mount Charles
Belfast, Northern Ireland BT7 1NZ (GB)

(56) References cited:
WO-A-99/62189

EP 2 267 700 B1

Note: Within nine months of the publication of the mention of the grant of the European patent in the European Patent Bulletin, any person may give notice to the European Patent Office of opposition to that patent, in accordance with the Implementing Regulations. Notice of opposition shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).

DescriptionField of the Invention

5 **[0001]** The present invention relates to audio encoders, especially audio encoders for use in Bluetooth systems. The invention relates particularly to selecting quantisation bands by searching quantiser tables in audio encoders.

Background to the Invention

10 **[0002]** Bluetooth is a short range wireless communications technology that can create personal area networks (PANs) between a variety of fixed and mobile devices, such as mobile phones, laptops, digital cameras, games consoles and media players. It is estimated that Bluetooth currently enjoys an installed base of almost two billion devices worldwide. Specifications and standards for Bluetooth operation are created and maintained by the Bluetooth Special Interest Group (SIG).

15 **[0003]** Although Bluetooth is a ubiquitous wireless communications technology, there are a number of technical considerations when using it for the wireless transmission of high-quality stereo audio.

[0004] The transmission of uncompressed stereo audio at the same sample rates found on CDs requires a data rate of approximately 1.4 Mbps. This transmission rate cannot be met by Bluetooth devices that do not support Extended Data Rate (EDR) transmission rates. Even if a device supports EDR, power consumption and link reliability reasons argue against such a high transmission rate. Therefore, it is necessary to compress the audio so that it can be successfully transmitted at a lower rate. The Advanced Audio Distribution Profile (A2DP) defines a low-complexity audio compression codec called Sub Band Codec (SBC). SBC is fully specified by the Bluetooth standards documents, is free for commercial use and is the only mandatory audio compression codec for Bluetooth. Any Bluetooth device that needs to communicate high-quality audio using the A2DP must implement SBC. However, the A2DP allows the use of additional optional audio compression codecs. Therefore, a suitably equipped pair of Bluetooth devices may choose to communicate audio using a compression codec other than SBC. The decision as to which compression codec to use is made as part of an initial exchange of messages when the communicating devices first establish the Bluetooth link. This is a process known as pairing.

20 **[0005]** Accordingly, there is an opportunity to provide alternative audio compression codecs, with alternative audio compression and/or decompression techniques, that are suitable for use with Bluetooth and which improve upon conventional codecs. When implementing alternative codecs, there are technical issues that can be addressed to improve the performance of the codec.

25 **[0006]** For example, Bluetooth communications capability is typically added to a product, e.g. a mobile device, by integrating one of many single-chip Bluetooth solutions provided by companies such as Cambridge Silicon Radio, Broadcom and others. Audio compression and decompression functions are usually implemented in the form of embedded software executing on a general purpose processor or digital signal processor that is internal to the Bluetooth chip. The execution of the audio compression codec requires the processor to carry out a certain amount of work. This, in turn, causes the processor to draw a corresponding level of current from the battery powering the mobile device, which has a direct effect on how long the device can operate on a single charge of the battery.

30 **[0007]** WO 99/62189 A2 discloses a binary search procedure for determining the optimal quantization step size.

[0008] Accordingly, it would be advantageous to code the audio compression software so that its functions can be executed using the smallest number of processor instructions possible.

Summary of the Invention

35 **[0009]** Accordingly, a first aspect of the invention provides an apparatus for compressing an audio signal as claimed in claim 1.

[0010] A second aspect of the invention provides a method of selecting quantisation bands as claimed in claim 14.

[0011] A third aspect of the invention provides a computer program product as claimed in claim 15.

40 **[0012]** Preferred features are recited in the dependent claims.

[0013] The present invention enables quantiser tables to be searched using relatively few processor instructions for each search iteration. Fewer processor instructions equates to reduced current consumption and better battery life. Further advantageous aspects of the invention will become apparent to those ordinarily skilled in the art upon review of the following description of a preferred embodiment and with reference to the accompanying drawings.

55

Brief Description of the Drawings

[0014] An embodiment of the invention is now described by way of example and with reference to the accompanying

drawings in which:

Figure 1 is a block diagram of an audio compression encoder suitable for use with the present invention;

Figure 2 is a block diagram of an audio compression decoder suitable for use with the encoder of Figure 1 to provide an audio compression codec;

Figure 3 is a graph illustrating exemplary quantization bands that are suitable for use with the present invention;

Figure 4 is a block diagram of a processor architecture suitable for use in the implementation of the audio compression encoder of Figure 1 or the decoder of Figure 2;

Figure 5 is a first representation of registers and quantiser tables implemented by the architecture of Figure 4, illustrating an example of a table searching method for use in preferred embodiments of the invention and showing the state of the registers after initialisation;

Figure 6 is a second representation of the registers and quantiser tables of Figure 5, showing the state of the registers after a first instruction;

Figure 7 is a third representation of the registers and quantiser tables of Figure 5, showing the state of the registers after a second instruction;

Figure 8 is a fourth representation of the registers and quantiser tables of Figure 5, showing the state of the registers after a third instruction for a positive value in the register rMAC; and

Figure 9 is a fifth representation of the registers and quantiser tables of Figure 5, showing the state of the registers after the third instruction for a negative value in the register rMAC.

Detailed Description of the Drawings

[0015] Referring now to Figures 1 and 2 of the drawings, there is shown, by way of example, an audio compression codec comprising an audio encoder 10 and an audio decoder 50. The encoder 10 and decoder 50 are suitable for use as an audio compression codec in systems that support Bluetooth. The encoder 10 receives an input signal comprising audio samples, i.e. from an audio signal sampled in the time domain. The encoder 10 processes the input signal to produce encoded, or compressed, output samples. The processing performed by the encoder 10 compresses the input signal such that it, or more particularly a compressed representation of it, may be transmitted to the decoder 50 with lower bandwidth requirements than would be necessary for the uncompressed signal. The decoder 50 receives the encoded samples and decodes them to produce time domain audio samples representing the input audio signal received by the encoder. The correspondence between the audio samples produced by the decoder 50 and the audio samples received by the encoder 10 is dependent on the compression techniques used by the encoder 10. The encoder 10 and decoder 50 communicate by means of a communications link (not illustrated) which may, for example, comprises a wireless link such as a Bluetooth link. In such cases, the encoder 10 is co-operable with a transmitter (not shown) for transmitting the compressed output samples to the decoder 50, and the decoder 50 is co-operable with a receiver (not shown) for receiving the compressed output samples. The transmitter and receiver each process the samples in accordance with the communications technology, e.g. Bluetooth, that they are configured to support.

[0016] Referring in particular to Figure 1, in the illustrated embodiment, the encoder 10 processes the input stream 11 of audio samples in groups of multiple, in this case 4, samples per audio channel (i.e. left and right channels). Each successive group of samples is processed by an analysis filter bank 12 that splits the input audio stream into multiple, in this case 4, parallel sub-band sample streams 14, where the sample rate of each sub-band stream 14 is a fraction of, in this case $\frac{1}{4}$ of, that of the input audio stream 11. The respective data in each sub-band stream 14 represents a respective subset of audio content from the input stream 11 that falls into a respective frequency range covered by each sub-band. In the present example, each sub-band is of equal width and covers $\frac{1}{4}$ of the range of audible frequencies. For example, if the audio samples are arriving at a rate of 48 kHz, corresponding to a total audio bandwidth of 24 kHz, then each sub-band covers a 6 kHz range of frequencies. The first sub-band passes the audio content within the 0-6 kHz, the second 6-12 kHz, the third 12-18 kHz and the final sub-band passes 18-24 kHz.

[0017] Since the input audio stream has been transformed into 4 parallel streams, each at a sample rate $\frac{1}{4}$ that of the original, the overall data rate has not changed, but a form of time to frequency transformation has been carried out, since each sub-band stream represents a particularly frequency range. This allows the data in each sub-band to undergo

lossy compression independently and with different levels of loss or distortion in each band. Since the human ear is more sensitive to audio in some frequency ranges than others, additional levels of compression can be gained by more aggressively compressing the frequency bands to which the ear is relatively insensitive.

[0018] In the preferred embodiment, lossy compression of the data in each sub-band is carried out using differential coding and quantization. For each sub-band stream 14, a respective adaptive predictor filter 16 is provided and acts as a predictor, providing a predicted value for upcoming samples from the sub-band filter bank 12. A respective error calculator 18 is provided for each sub-band stream 14 and is arranged to receive, in respect of each sub-band sample of the respective sub-band stream 14, the actual sub-band value from the sub-band filter bank 12 and the corresponding predicted sub-band value from the predictive filter 16, and to calculate the difference between the predicted value and the actual value to produce a prediction error value. As a sub-band sample is created by the sub-band filter bank 12, the difference between its value and its predicted value is produced by the error calculator 18. If the predictive filter 16 is performing well then the magnitude of this difference, or prediction error value, is small compared to the sub-band sample. Since the prediction error tends to be small, it can be communicated using fewer bits than the corresponding sub-band sample and hence transmitting prediction errors offers a form of compression.

[0019] The levels of compression are increased by quantising the prediction error value so that the quantized error value can be represented using still fewer bits. To this end, the encoder 10 includes a respective quantizer 20 for each sub-band stream 14 arranged to receive the prediction error values and to produce respective quantized error values. It is the quantization operation that introduces the lossy aspect of the compression, since the quantization causes information to be discarded and is a process that can only be approximately inverted. Within the encoder 10, a respective inverse quantizer 22 is arranged to perform inverse quantization on the error values and the output is provided to the respective predictive filter 16. Inverse quantization is carried out to provide approximate prediction error samples to stimulate the predictive filters 16. Hence, the predictor filters 16 produce predictions of future samples emerging from the sub-band filter bank 12 based on a history of previous prediction error values.

[0020] In the illustrated embodiment, the same differential coding, quantization, inverse quantization and prediction techniques are applied independently to the respective data in each sub-band stream 14. However, the sophistication and accuracy of the respective prediction filters 16 and/or the levels of approximation employed by the respective quantizers 20 may differ from one sub-band to the next, for example based on the relative sensitivity of the human ear to audio frequencies in the respective sub-band.

[0021] The encoder 10 further includes a data packer 24 for concatenating the respective quantized prediction error values for each sub-band to provide a concatenated quantized error value in respect of each group of input audio samples. Each concatenated quantized error value provides a respective compressed output sample of the encoder 10. The data from the respective audio channels is interleaved and so the encoder 10 processes the respective channels alternately, e.g. each channel being processed alternately with the other, one groups of input samples at a time.

[0022] By way of example, when utilised in conjunction with Bluetooth transmission, the illustrated encoder 10 consumes 4 input 16-bit audio samples at a time and produces a single 16-bit compressed sample, hence compressing the input audio by a factor of 4.

[0023] It will be understood that in alternative embodiments, the encoder may be in general arranged to process audio samples in groups of at least one, but typically a plurality of, samples per audio channel. Further, in alternative embodiments, the encoder may be in general arranged to split the audio signal into any practicable number of sub-band sample streams, or even not to split the audio signal into sub-band streams. The configuration of the encoder, and correspondingly of the decoder, would be adapted accordingly as would be apparent to a skilled person.

[0024] It will be seen from the foregoing that the preferred encoder 10 of Figure 1 implements a Sub-band Adaptive Differential Pulse Code Modulation (SB-ADPCM) compression scheme employing lossy compression to reduce the required network transmission rate for an audio input signal. In the illustrated example, the encoder 10 achieves a fixed compression ratio of 4:1, thereby requiring only 25% of the network transmission rate associated with the communication of uncompressed audio.

[0025] Referring now to Figure 2, the corresponding decoder 50 is described. The decoder 50 must initially synchronise itself to the incoming data stream 51 that it receives from the encoder 10. In this embodiment, it is assumed that the encoder 10 and decoder 50 communicate via a Bluetooth link. In accordance with Bluetooth, data is transmitted in groups, or packets, of 8-bit bytes that do not necessarily correspond conveniently with the compressed output samples from the encoder 10. For example, the compressed samples output from the encoder 10 are 16 bits long in the present example and, for stereo transmission, are transmitted in channel interleaved fashion and so compressed samples for the left and right channels are sent alternately. Therefore, the decoder 50 needs to initially deduce from the sequence of bytes in a received Bluetooth-compatible packet whether a particular byte is the most or least significant byte of a compressed output sample from the encoder 10, and whether it belongs to the output sample for the left or the right channel. This creates four possibilities that need to be resolved and the decoder 50 accomplishes this by examining the incoming data stream 51 for synchronisation data. As is described in more detail hereinafter, the synchronisation data is preferably added by the quantization process during encoding. Once the synchronisation data has been found with

a high degree of reliability, it uniquely identifies the data from the left and right audio channels and which bytes are the most and least significant bytes of the compressed output samples.

[0026] Once synchronisation has been achieved, the compressed samples are unpacked such that the bits of each compressed sample are split into the respective quantized prediction error value for each sub-band. In the decoder 50 of Figure 2, the synchronisation and unpacking is performed by a synchronisation and unpacking module 52. The module 52 thus produces a respective data stream 53 for each sub-band, each data stream 53 comprising the respective quantized prediction error values. An adaptive inverse quantizer 54 is provided for each sub-band in order to perform inverse quantization of the quantized prediction error values. A respective adaptive predictor filter 56 is provided for each sub-band and is arranged to regenerate the same predicted sample values that were produced by corresponding predictor filters 16 at the encoder 10. For each sub-band, a respective adder 58 is provided for adding the inverse-quantized prediction error values produced by the inverse quantizers 54 to the respective predicted sample values produced by the predictor filters 56. By adding an inverse-quantized prediction error to the corresponding regenerated prediction sample value, a close approximation to the respective original sub-band sample of the respective sub-band stream 14 is obtained. In this embodiment, the form of the inverse quantization and prediction performed by the quantizer 54 and predictor filter 56 respectively is identical to that performed for the corresponding sub-band in the encoder 10.

[0027] The respective sub-band samples produced by the respective inverse quantizers 56 are input to a sub-band synthesis filter bank 60 in order to produce audio samples that are representative of the input audio samples received by the encoder 10. The sub-band filter bank 60 operates in the opposite sense to the filter bank 24 of the encoder 10. That is, instead of analysing an input stream and splitting it into multiple, in this case 4, lower-rate streams covering multiple, in this case 4, frequency ranges, the filter bank 60 takes the lower-rate streams and synthesises the original full-rate stream from it. Hence the consumption of a single compressed sample by the decoder 50 results in multiple, in this case 4, output audio samples being produced. The structure and individual filter specification of the synthesis filter bank 60 is similar in many respects to the analysis filter bank 12 in the encoder 10, but is not identical.

[0028] The codec 10, 50 is adaptive in two major ways. Firstly, the predictor filter in each sub-band adapts its coefficients based on a history of its input and previous output values. The adjustments are carried out to try to drive the prediction error towards zero and minimise it. The second area of adaptation is around the quantisation and inverse quantisation processes. The width or spacing of the quantisation levels is dynamically adjusted to quantise the signal with minimal distortion. Hence, if the quantiser input (prediction error) is small, the quantiser will tend to reduce the quantisation level spacing, so that small values can be quantised with little additional error. Similarly, if the quantiser input starts to get big, then the quantiser will increase the quantisation level spacing. Failure to do so would result in an inaccurate quantised representations of large input values, even if the outermost quantisation level is chosen.

[0029] During quantization of a prediction error value, each quantizer 20 is capable of selecting from a plurality of quantization bands, each quantization band covering a respective range of values that the prediction error value could take and being represented by a respective quantization output value. When the quantizer 20 selects a quantization band for a prediction error value, the respective quantization output value becomes the quantized prediction error value. An example is shown in Figure 3 where a continuous waveform 70 is quantized using eight quantization bands 72, the quantized output value 74 associated with each quantization band being, by way of example only, a 3-bit label. Prediction error values corresponding to points A and B on the waveform will, in this example, each be assigned to the same quantization band (with label 011) even though they have different absolute values. It is noted that, typically, the quantization bands are not equally sized. It is noted that the quantization output values may comprise fewer or more than 3 bits. Moreover, the number of bits making up the quantization output value may differ between sub-bands. For example, the quantizer may use a higher number of bits for the quantization output value (i.e. a higher resolution) for sub-bands in respect of which the human ear is more sensitive than for sub-bands in respect of which the human ear is relatively insensitive.

[0030] Data defining the quantization bands is stored in memory, typically in a quantiser table (illustrated in Figures 5 to 9). During use, the quantizer 20 performs a search of the quantiser table in order to determine the most appropriate quantisation band for the respective prediction error value. This is described in further detail below.

[0031] In order to formulate optimized algorithm coding techniques that result in a reduced number of processor instructions, it is necessary to take into account the detail of the underlying hardware architecture of the processor begin utilised to implement the audio compression or decompression. In particular, it is advantageous to exploit opportunities for parallelism, whereby several parallel operations can be triggered by a single processor instruction. Whilst there can be considerable variation in hardware architecture across different processors, Figure 4 shows the pertinent features of a typical hardware structure for a digital signal processor as found on a Bluetooth chip.

[0032] Referring now to Figure 4, there is shown, generally indicated as 80, a processor architecture that is suitable for use in implementing the encoder 10. The architecture 80 includes a processor 82 in the preferred form of a digital signal processor, although other data processors, or microprocessors, may alternatively be used. The processor 82 includes a plurality of data registers, shown in Figure 4 as register bank 84. These registers may comprise general purpose registers for containing data such as input operands and computed results of various arithmetic and logical

operations. Optionally, one or more dedicated registers or register banks may be provided for address generation. These are represented in Figure 4 by an address generator unit 86. The processor 82 further includes an arithmetic logic unit (ALU) 88 for performing logical and/or arithmetic computations on operands. A data memory interface unit 90 is also provided to allow the processor 82 to communicate with local memory device(s) 92, and in particular with data memory as is described in more detail hereinafter.

[0033] Signal processors typically have a set of peripherals, consisting of functions like timers, UARTs and special interfaces to get data on and off the processor in various formats. This is represented in Figure 4 by peripheral unit 85. The architecture 80 typically includes a memory management unit (MMU) 93 with which the processor 82 is in communication for handling accesses to memory 92. The MMU 93 can take any suitable conventional form. The memory device 92 includes a plurality of data memories 94, in this example two data memories labelled as Data Memory 1 and Data Memory 2 in Figure 4. The data memory interface unit 90 supports a respective data memory interface 91 between the processor 82 and each of the data memories 94. In addition, at least one program memory 96 is provided. The processor 82 is in communication with the program memory 96 by means of any suitable conventional program memory interface. The memory device 92 typically comprises one or more random access memory (RAM).

[0034] In use, the program memory 96 stores processor executable instructions that the processor 82 is required to implement. In this example, the instructions take the form of embedded executable computer software for performing one or more of the functions required to implement audio compression or decompression. In the preferred embodiment, one of the data memories 94, e.g. Data Memory 1, stores (when audio compression is being implemented) the prediction error values that are required to be quantized, while a separate data memory, e.g. Data Memory 2, stores data defining the quantisation bands, typically in the form of one or more quantiser tables.

[0035] The respective parallel data memory interfaces 91 allow a data value to be read from, or written to, each data memory Data Memory 1, Data Memory 2 concurrently. In conjunction with the ALU 88, this allows an arithmetic or logical computation to be performed in parallel with up to two memory accesses. Arranging the algorithmic computations to maximally exploit this parallelism facilitates keeping the processor instruction count low.

[0036] Typically, the address generator unit 86 has, in respect of each data memory 94, a plurality of address generators that can address the respective data memory. Each address generator comprises 3 hardware registers, namely an index (I) register, which holds the actual memory address to index the corresponding data memory 94; a modify (M) register, which contains a value by which the contents of the corresponding I register are automatically incremented by the processor architecture 80 after a memory access; and a length (L) register, which allows the set-up of modulo addressing, so that a range of addresses that are traversed by the processor hardware 80 in a circular or wrap-around fashion can be defined. These complex addressing schemes can be set up once in software and then operate under hardware control without further software programming. Therefore, mapping the memory accesses required to implement parts of the audio compression scheme to the addressing modes that can be implemented by these address generation registers greatly reduces the number of processor instructions required for general memory addressing.

[0037] Another important practical area of implementing the audio compression algorithm as software for Bluetooth systems is the numerical accuracy or precision with which the various signal processing functions in the algorithm are carried out. Typically, processors within Bluetooth chips are fixed-point processors, meaning that computation results need to be continually limited, scaled or rounded to remain compliant with a fixed numerical format adopted by the processor. This allows simpler and lower-powered processors to be created, but means that signal processing algorithms cannot be implemented in a mathematically-exact fashion. Fixed-point processors often possess a small number of extended-precision registers, which allow relatively short sequences of arithmetic and logical instructions to take place at higher levels of precision, with limits and rounding only being applied when results are moved from an extended-precision register. This can be advantageous for signal processing algorithms such as filters, where intermediate results are accumulated within an extended-precision register and precision loss is only incurred when the final filter output value is removed from that register. One of the primary attributes of preferred audio compression techniques is the maintenance of very high levels of audio quality, despite the effects of lossy compression. Hence, when adapting such algorithms for use with Bluetooth it is desirable to minimise the loss of mathematical precision caused by the fixed-point nature of the typical Bluetooth processor, so that such precision losses do not substantially affect the preservation of audio quality.

[0038] During the process that quantises prediction error values to produce compressed sub-band values the quantised value that most closely represents the higher-precision input value, i.e. the prediction error value being quantized, is determined. Successive bands or ranges of input values map to successive quantised values. An example of this mapping is shown in Figure 3, where 8 bands of input values map to 8 different quantised values.

[0039] In the encoder 10, for most sensible audio signals, the predictor 16 operation creates a situation whereby small errors are statistically more likely to be fed into the quantisation process than larger errors. Therefore, in order to reduce the average distortion and loss introduced by the quantisation process, it is advantageous to quantise small input values relatively finely and large input values relatively coarsely, i.e. the quantised output values represent small inputs relatively precisely and large inputs relatively inaccurately. The inaccurate representation of large inputs is acceptable because

they occur relatively infrequently in a statistical sense. Hence, for a particular sub-band, if the quantised output values are limited to N bits, the corresponding 2^N input bands are not of equal size or range. Input bands that cover small input values are of relatively small size. As the input value increase, bands become wider. This non-linear distribution of quantisation bands creates the situation where small-valued inputs are quantised more accurately than larger inputs. Figure 3 shows a widening of quantisation bands with increasing input size. The non-linear distribution of quantisation bands means that it is non-trivial to determine which band a particular high-precision input value falls into. There is an additional complication caused by the fact that the audio compression algorithm may adapt (i.e. change) the width of the quantisation bands dynamically, by linearly scaling the default (non-linearly distributed) input values that mark quantisation band boundaries. The value used for the scaling, called δ , may be based on an estimate of the short-term standard deviation of the input signal to the quantisation, and adapts the width of quantisation bands to best cover the current degree of variation seen in the input signal.

[0040] There is now described a preferred method of determining to which quantization band a high precision input value (in this example a prediction error value) should be assigned. It is noted that the following description only considers the case where the quantisation input value is positive. This is valid because, for an N -bit quantised output, the preferred quantisation process utilises 1 output bit to convey sign information about the value (i.e. whether it is positive or negative) and the remaining $N-1$ bits to convey magnitude information.

[0041] The quantisation of the magnitude is of primary interest, which is performed on the absolute value of the high-precision input value and results in an $N-1$ bit output. The default (non-linearly distributed) positive input values that mark quantisation band boundaries are stored in a look-up table (illustrated in Figures 5 to 9), or other suitable memory structure, which resides in processor memory (in this example Data Memory 2). The quantisation band boundaries values are sorted according to value within the table (e.g. increasing values in the table are stored at locations with increasing table indices (addresses)). In alternative embodiments, it is possible to use an arrangement whereby increasing table values are stored in decreasing memory addresses. This would still allow the three-instruction cycle for each bisection search iteration that is described hereinafter. To accommodate this and in contrast to the method described hereinafter, quantisation table address offsets would be subtracted, rather than added, to the base address of the quantiser table. The principal criterion for the search to work is that the quantisation table values are sorted (either ascending or descending).

[0042] This table is searched until the smallest table index, $I_{\min}$, is found for which the contents of the table at that index, when multiplied by the scaling value δ (if applicable), exceeds the high-precision input value. An alternative implementation would be to find the largest table index that gives a result less than or equal to the high-precision input value, assuming ascending quantisation table values are stored in ascending memory addresses.

[0043] The $N-1$ output bits for the quantised magnitude are then set to $I_{\min}$, i.e. the table indices serve in this example as the label, or quantized value, for the respective quantization band. With regard to the use of the notation $I_{\min}$ to represent, a table index, in this example it is assumed that the first (smallest) quantisation table value as index 0, the second value as index 1 and so on, up to the final value which will have index $(2^{N-1})-1$. The base of the table may be located at any convenient memory address. Using Figs 5-9 as an example, the table memory addresses run from 1000-1015 but the table indices run from 0 to 15. It is the table indices, not the physical memory addresses, that give the $N-1$ bits that form the absolute quantised value. This is concatenated with the single sign bit to give the (signed) N -bit quantised value.

[0044] The table searching process could be completed in a naive fashion by starting at the base of the table and incrementing the index by one until the table value multiplied by δ just exceeds the high-precision input value. However, this is inefficient, since many table entries may have to be tested incrementally before the correct one is found. Preferably, therefore, the search of the table is performed using a binary, or bisection, search technique, where the search procedure iteratively removes approximately half of the table indices that remain eligible as solutions. Therefore, the solution for a table containing 2^{N-1} values is obtained in $N-1$ iterations. Binary searching is not necessarily the fastest searching procedure but it is simple, easy to implement and is guaranteed to converge to the correct solution in all cases.

[0045] There is now described a preferred method of implementing binary searching in software that executes on processor architectures the same or similar to that of Figure 4. For the purposes of bisection searching, the key capabilities exhibited by the processor 82 are:

- The ability to carry out an unconditional arithmetic operation and two, or at least two, independent data memory accesses as a single processor instruction
- The ability to carry out a conditional arithmetic operation and one, or at least one, data memory access as a single processor instruction
- The ability to conditionally increment the value contained in an index register used to address data memories as a zero-overhead side effect of performing a memory access using that index register. As this feature is conditional, it is also possible to instruct the processor not to perform this automatic increment of the index register value as part of a memory access using any particular index register.

[0046] In the exemplary architecture of Figure 4, the MMU 93 and memory interface 90 are co-operable with one another to allow the two concurrent accesses to Data Memory 1 and Data Memory 2.

[0047] In order to implement binary searching efficiently on such processors, it is necessary to efficiently implement the basic iteration of the search procedure, as this iteration is carried out N-1 times to complete the search. Ideally, the basic search iteration should represent as few processor instructions and clock cycles as possible. The following description demonstrates how an iteration of bisection searching can be performed using just three consecutive processor instructions, representing a high degree of efficiency. Table 1 below introduces a nomenclature that describes the various processor memories and registers that hold key values for the bisection search iteration process.

TABLE 1

Label	Description
RThreshBase	A general-purpose processor register that holds the memory address of the base (bottom element) of the table of default quantisation band boundaries.
RThreshOffset	A general-purpose processor register that holds an address offset such that a particular element of the table of default quantisation band boundaries can be accessed by accessing the memory address given by rThreshBase + rThreshOffset.
rOldThreshOffset	A general-purpose processor register that holds the value of rThreshOffset from the previous iteration of the bisection search.
rThreshTableValue	A general-purpose processor register that holds a particular value read from the table of default quantisation band boundaries residing in processor memory (Data Memory 2 in the present example).
rDelta	A general-purpose processor register that holds the current value for the scaling factor δ .
rMAC	An extended-precision processor register that holds the output of multiply and multiply/accumulate operations without precision loss.
lBisOff	A dedicated index (I) processor register that holds the memory address of an element of a "bisection offset" table. An important capability for the purposes of bisection searching is the ability for the memory address held in the index register to be automatically incremented by 1 after a memory access using that index register. A description of the "bisection offset" table is given below.

[0048] With reference to Figure 4, the lBisOff register may be implemented in the address generator 86, while the remaining registers identified above may be implemented in the register bank 84.

[0049] The bisection offset table comprises data held in processor memory 92 (for example in Data Memory 1 of Figure 4) and is indexed, or addressed, using the dedicated index register lBisOff. The contents of the bisection offset table are descending powers of 2, with the first element containing the value 2^{N-3} , where N is the number of bits in the quantised output, including the sign bit. The final element of the table is a trailing value of zero, although it could be any dummy value (it is read during the final search iteration but never used). It is desirable to keep all the search iterations the same so that the three search instructions can be repeated for the required number of iterations, without having to do a special instruction sequence for the final iteration.

[0050] Hence, using the example $N=7$, the bisection offset table contains the following contents, listed from the first to last element: 16, 8, 4, 2, 1, 0

[0051] The purpose of the bisection offset table is to generate successively lower powers of 2 by using incrementing memory accesses to a table. This could alternatively be performed by the ALU 88 of the processor 82 by taking the previous power of 2 and generating the new power of 2 by a 1-bit right shift of a binary value. A problem with this approach is that the right shift is an arithmetic/logical operation and only one of these operations can be carried out by a single processor instruction. However, up to two memory accesses can be carried out by a single processor instruction, so implementing the descending power of 2 generation using a table lookup allows the bisection search iteration to complete using fewer consecutive instructions.

[0052] There is a small amount of initialisation code that is executed before the first iteration of the bisection search. The initialisation code causes the memory address (an address in Data Memory 1 in the example of Figure 4) of the absolute value of the high-precision input value to the quantisation process is calculated and stored by the processor 82.

[0053] In typical embodiments, this address value is stored in an index register in the address generator unit 86, because the processor does not support the particular double memory access format of instruction 1 in the 3-cycle bisection search iteration unless the high-precision input is read from memory via an index (I) register. However, in

alternative embodiments it may be stored in any convenient location, e.g. the register bank 84.

[0054] This allows the absolute input value to be re-read from memory into a processor register at any time during the iterative bisection search. The index register *lbisOff* is loaded with the address of the first element in the bisection offset table. Also, before the first search iteration starts, *rThreshBase* is loaded with the address of the first element in the table of default quantisation band boundaries, and both *rThreshOffset* and *rOldThreshOffset* are loaded with the value 2^{N-2} , where *N* is the number of bits in the quantised output, including the sign bit. With this initialisation complete, subsequent bisection search iterations are then executed using 3 processor instructions each. The parallel operations carried out during each instruction are outlined below in Table 2.

TABLE2

Instruction Number	Operations Executed
1	Read the absolute value of the high-precision input from memory and load the value into the extended-precision register <i>rMAC</i> . This operation counts as one memory access. Read the value of the element of the table of default quantisation band boundaries addressed by <i>rThreshBase</i> + <i>rThreshOffset</i> and load this value into the register <i>rThreshTableValue</i> . This operation counts as one memory access.
2	Multiply <i>rThreshTableValue</i> by <i>rDelta</i> , subtract this product from the value already held in <i>rMAC</i> and store the final result back into <i>rMAC</i> again. The multiplication result is stored without loss of precision since <i>rMAC</i> is an extended-precision register. This operation counts as one unconditional logical/arithmetic operation. Read the value of the bisection offset table indexed by <i>lbisOff</i> and load the result into <i>rThreshOffset</i> . Do not select the automatic increment of <i>lbisOff</i> to point to the next element in the bisection offset table. This operation counts as one memory access. Note that this sets <i>rThreshOffset</i> to the correct value for the next bisection search iteration.
3	Only if the value currently held in <i>rMAC</i> is positive, add the values of <i>rThreshBase</i> and <i>rOldThreshOffset</i> and store the result back into <i>rThreshBase</i> . This operation counts as one conditional logical/arithmetic operation. Unconditionally read the value of the bisection offset table indexed by <i>lbisOff</i> and load the result into <i>rOldThreshOffset</i> . Select the automatic increment of <i>lbisOff</i> to point to the next element in the bisection offset table. This operation counts as one memory access. Note that <i>rOldThreshOffset</i> does not receive its updated value until after the potential addition with <i>rThreshBase</i> in the parallel logical/arithmetic operation above has been computed. Therefore, if the parallel logical/arithmetic operation is executed, the addition uses the value of <i>rOldThreshOffset</i> before update. This is essential for correct operation. Note that the updates to the <i>rOldThreshOffset</i> register and the (conditional) updates to the <i>rThreshBase</i> register ensure that both registers have the correct values for the next bisection search iteration, which can now begin directly from the next instruction.

[0055] Accordingly, in preferred embodiments, the input value to the quantiser and the quantisation tables are stored in different data memories so that the first instruction in the bisection search iteration completes in a single clock cycle.

[0056] It is noted that the bisection offset table is successively read from simple incrementing addresses. In contrast, for the quantisation band boundary table, the bisection search process causes the table base and table offset register values to "jump" by relatively large values from one search iteration to the next.

[0057] A pictorial example of how search iterations unfold over the 3 processor instructions is now presented with reference to Figures 5 to 9. In this example, it is assumed by way of example only that *N*=5, where *N* is the number of bits in the quantised output, including the sign bit. Hence, the number of default quantisation boundary values in the table to be searched is $2^{N-1} = 16$ elements. The example shows how the very first bisection search iteration unfolds.

[0058] In Figures 5 to 9, the individual processor registers identified above are shown on the left and the regions of processor memory, with their addresses (which are exemplary only), on the right.

[0059] In preferred embodiments, the high-precision input value and the quantisation band boundary table are stored in respective data memories (Data Memory 1 and Data Memory 2 respectively in the illustrated example). The bisection offset table can be in either data memory, but is assumed to be in Data Memory 1 in the present example and the example shown in Figs 5-9.

[0060] Typically, the memory addressing of the processor 82 is based on a flat memory model and so, from a software programming point of view Data Memory 1 and Data Memory 2 are addressed as 2 different regions in one larger address

space. In other words, there is "continuity of addressing" of the 2 different memories. The data memories 94 may be separate from one another, e.g. physically and electrically independent from one another, as shown in Figure 4. Alternatively, they may be implemented as respective independently, or concurrently, accessible portions of the same memory (e.g. in a dual-port memory). Either arrangement allows two memory accesses to occur in the same processor clock cycle.

[0061] With reference to Figs 5-9, addresses below 1000 are in Data Memory 1 and addresses from 1000 up are in Data Memory 2. It is noted that the specific memory addresses shown in Figures 5-9 are given by way of illustration only and are not intended to correspond with actual memory addresses.

[0062] The values contained within registers or memory locations are contained within the respective boxes in the diagrams.

[0063] For the example under consideration, Figure 5 shows the state of the processor 82 after the initialisation steps described above.

[0064] The first processor instruction of the first bisection search iteration is now carried out, updating the state of the processor as shown in Figure 6. It is noted that the values of rMAC and rThreshTableValue have been updated as a direct consequence of this instruction. The first row of Table 2 explains how the relevant operations have been performed.

[0065] The second processor instruction of the first bisection search iteration is now carried out, updating the state of the processor as shown in Figure 7. It is noted that the values of rMAC and rThreshOffset have been updated as a direct consequence of this instruction. The second row of Table 2 explains how the relevant operations have been performed.

[0066] The third processor instruction of the first bisection search iteration is now carried out. Since this instruction behaves differently depending on whether the value in the register rMAC after the second instruction completes is positive or not, both cases are illustrated below. First, it is assumed that the value in rMAC is positive. This means that the desired value in the default quantisation band boundary table is in the upper half of the table and it is this table region that should be promoted to the second search iteration. Assuming this condition, the updated processor state after the third instruction is shown in Figure 8. The values of rThreshBase, lbisOff and rOldThreshOffset have been updated as a direct consequence of this instruction. The third row of Table 2 explains how the relevant operations have been performed. Note also that the updated value in rThreshBase means that the second bisection search iteration will search only the table elements from address 1008 and above, i.e. the upper section of the table.

[0067] Now it is assumed that the value in rMAC after the second instruction 2 is not positive. This means that the desired value in the default quantisation band boundary table is in the lower half of the table and it is this table region that should be promoted to the second search iteration. Assuming this condition, the updated processor state after the third instruction is shown in Figure 9. The values of lbisOff and rOldThreshOffset have been updated as a direct consequence of this instruction. The third row of Table explains how the relevant operations have been performed. Note also that the value in rThreshBase is unchanged. This ensures that the second bisection search iteration will search only the lower half of the table, from address 1000.

[0068] In the manner described above, subsequent bisection search iterations are carried out such that, with each iteration, an upper or lower section of the band boundary table is discarded until the boundary band that is most suited to the high precision input value is located.

[0069] The invention is not limited to the embodiments described herein, which may be modified or varied without departing from the scope of the invention.

Claims

1. An apparatus for compressing an audio signal, the apparatus comprising a processor and data memory, the processor including an arithmetic logic unit and a plurality of data registers and being programmable to implement a plurality of processor instructions, and wherein the processor is capable of performing, in respect of a single processor instruction, an unconditional arithmetic operation and two independent data memory accesses, or a conditional arithmetic operation and one data memory access, the processor further being capable of causing a value contained in at least one of said data registers that is used to address said data memory to be incremented, and wherein said apparatus further includes
 - a first set of data values stored in said data memory, said first set comprising data values each representing a respective quantisation band boundary and being sorted in said data memory according to their respective values, and
 - a second set of data values stored in said data memory, said second set comprising at least one data value that is a power of 2, the data values of said second set being sorted in said data memory in order of descending value, and wherein
 - a first of said data registers (rThreshBase) holds, in use, a memory address for a first data value of said first set, a second of said data registers (rThreshOffset) holds, in use, a current address offset value, a third of said data registers (rOldThreshOffset) holds, in use, a previous address offset value, a fourth of said data registers (rThreshTableValue) holds, in use, a data value obtained by said processor from said first data set, a fifth of said data registers

(rMAC) holds, in use, a data value being the result of one or more computations performed by said processor in respect of the data value held in said fourth register, and a sixth of said data registers (IbisOff) holds a memory address of a data value in said second set of data values,

and wherein, in respect of each of a plurality of search input values derived from said audio signal and stored in said data memory, said apparatus is arranged to perform a bisection search of said first set of data values to determine which of the data values in said first set is the best approximation of the respective input value, said bisection search being performed in one or more iterations wherein, in respect of each search iteration said processor is arranged to,

in execution of a first processor instruction, read the respective input value from data memory and cause the input value to be stored in said fifth data register, and read the data value of said first set whose address in said data memory corresponds to the address held in said first data register offset by the current address offset value held in said second data register, and cause said data value to be stored in said fourth data register, and

in execution of a second processor instruction, calculate the difference between the input value stored in said fifth data register and said data value of said first set that is stored in said fourth data register, or a derivative thereof, and cause the resultant value to be stored in said fifth data register, and read the data value of said second set whose address in said data memory corresponds to the address held in said sixth data register and cause the result to be stored in said second data register, and

in execution of a third processor instruction, adjust, only if said difference held in said fifth data register indicates that the next iteration of the bisection search should take place in an upper section of said first set of data values, the memory address in said first data register by the offset stored in said third data register and cause the resulting adjusted memory address to be stored in said first data register (rThreshBase), read the data value of said second set whose address in said data memory corresponds to the address held in said sixth data register and cause the result to be stored in said third data register, and cause the memory address held in data register and cause the result to be stored in said third data register, and cause the memory address held in said sixth data register to be incremented to point to the next data value in said second set.

2. An apparatus as claimed in claim 1, wherein a seventh of said data registers (rDelta) holds, in use, a current value of a quantisation band scaling factor, and wherein in execution of said second processor instruction said processor is arranged to calculate the difference between the input value stored in said fifth data register and said data value of said first set that is stored in said fourth data register multiplied by said scaling factor, and to cause the resultant value to be stored in said fifth data register.
3. An apparatus as claimed in claim 1 or 2, wherein, in execution of said second processor instruction, said processor is arranged to calculate said difference by subtracting said data value of said first set that is stored in said fourth data register, or a derivative thereof, from the input value stored in said fifth data register, and wherein, in execution of said third processor instruction, the processor is arranged to adjust the memory address in said first data register only if said difference is positive.
4. An apparatus as claimed in any preceding claim, wherein said fifth data register is an extended-precision processor register that holds the output of multiply and multiply/accumulate operations without precision loss.
5. An apparatus as claimed in any preceding claim, wherein said first set of data values are sorted in said data memory such that successive data values are associated with successive addresses in said data memory.
6. An apparatus as claimed in any preceding claim, wherein said data values in said first set are sorted and stored in said data memory such that increasing data values are stored at memory locations with increasing addresses.
7. An apparatus as claimed in any preceding claim, arranged to produce, in respect of each of said search input values, a quantised output value comprising N bits where N-1 bits represent the quantised value and the remaining bit indicates whether the N-1 bit quantised value is positive or negative.
8. An apparatus as claimed in claim 7, wherein said second data set comprises at least N-2 data values, the first data value in said second set taking the value 2^{N-3} the following N-3 data values being successive decreasing powers of 2.
9. An apparatus as claimed in claim 8, wherein said second data set comprises N-1 data values including a final dummy data value, for example set to zero.
10. An apparatus as claimed in any one of claims 7 to 9, wherein said second and third data registers are initialised

with the value 2^{N-2} .

11. An apparatus as claimed in any preceding claim, wherein said search input values are held in a separate data memory, or at least in a separately accessible part of a common data memory, to said first data set.
12. An apparatus as claimed in any one of claims 5 to 11, wherein each data value of said first set is associated with an index value and wherein multiple iterations of said bisection search are performed until the smallest index value ($I_{\min}$) is found for which the respective data value, when multiplied by said scaling value, if applicable, exceeds the search input value, or alternatively until the largest index is found for which the respective data value is less than or equal to the search input value.
13. An apparatus as claimed in any preceding claim, wherein, before a first iteration of said bisection search, the memory address of the first data value in said second data set is stored in said sixth data register.
14. A method of selecting quantisation bands for an apparatus as claimed in claim 1, the method comprising, in respect of each of a plurality of search input values derived from said audio signal and stored in said data memory, performing a bisection search of said first set of data values to determine which of the data values in said first set is the best approximation of the respective input value, said bisection search being performed in one or more iterations wherein, in respect of each search iteration said method includes,
 - reading, in execution of a first processor instruction, the respective input value from data memory and cause the input value to be stored in said fifth data register, reading the data value of said first set whose address in said data memory corresponds to the address held in said first data register offset by the current address offset value held in said second data register, and causing said data value to be stored in said fourth data register, and
 - calculating, in execution of a second processor instruction, the difference between the input value stored in said fifth data register and said data value of said first set that is stored in said fourth data register, or a derivative thereof, causing the resultant value to be stored in said fifth data register, and reading the data value of said second set whose address in said data memory corresponds to the address held in said sixth data register and cause the result to be stored in said second data register, and
 - in execution of a third processor instruction, adjusting, only if said difference held in said fifth data register indicates that the next iteration of the bisection search should take place in an upper section of said first set of data values, the memory address in said first data register by the offset stored in said third data register and causing the resulting adjusted memory address to be stored in said first data register ($rThreshBase$), reading the data value of said second set whose address in said data memory corresponds to the address held in said sixth data register and causing the result to be stored in said third data register, and causing the memory address held in said sixth data register to be incremented to point to the next data value in said second set.
15. A computer program product comprising computer program code stored on a computer usable medium and arranged to cause the processor of the apparatus of claim 1 to perform the method of claim 14.

Patentansprüche

1. Vorrichtung zum Komprimieren eines Audiosignals, wobei die Vorrichtung einen Prozessor und einen Datenspeicher umfasst, wobei der Prozessor eine arithmetische Logikeinheit und mehrere Datenregister beinhaltet und so programmiert werden kann, dass er mehrere Prozessorbefehle implementiert, und wobei der Prozessor in Bezug auf einen einzigen Prozessorbefehl eine unbedingte Rechenoperation und zwei unabhängige Datenspeicherzugriffe oder eine bedingte Rechenoperation und einen Datenspeicherzugriff ausführen kann, wobei der Prozessor ferner bewirken kann, dass ein in wenigstens einem der genannten Datenregister enthaltener Wert zum Adressieren des genannten Datenspeichers inkrementiert wird, und wobei die genannte Vorrichtung ferner Folgendes beinhaltet:

einen ersten Satz von in dem genannten Datenspeicher gespeicherten Datenwerten, wobei der genannte erste Satz Datenwerte umfasst, die jeweils eine jeweilige Quantisierungsbandgrenze repräsentieren und in dem genannten Datenspeicher nach ihren jeweiligen Werten sortiert werden, und

einen zweiten Satz von in dem genannten Datenspeicher gespeicherten Datenwerten, wobei der genannte zweite Satz wenigstens einen Datenwert umfasst, der eine Zweierpotenz ist, wobei die Datenwerte des genannten zweiten Satzes in dem genannten Datenspeicher in absteigender Wertefolge sortiert werden, und wobei ein erstes der genannten Datenregister ($rThreshBase$) beim Gebrauch eine Speicheradresse für einen ersten Datenwert des genannten ersten Satzes hält, ein zweites der genannten Datenregister ($rThreshOffset$) beim

Gebrauch einen aktuellen Adressenversatzwert hält, ein drittes der genannten Datenregister (rOldThreshOffset) beim Gebrauch einen vorherigen Adressenversatzwert hält, ein viertes der genannten Datenregister (rThreshTableValue) beim Gebrauch einen Datenwert hält, der von dem genannten Prozessor von dem genannten ersten Datensatz erhalten wird, ein fünftes der genannten Datenregister (rMAC) beim Gebrauch einen Datenwert hält, der das Ergebnis von einer oder mehreren Berechnungen ist, die von dem genannten Prozessor in Bezug auf den in dem genannten vierten Register gehaltenen Datenwert durchgeführt werden, und ein sechstes der genannten Datenregister (IbisOff) eine Speicheradresse eines Datenwertes in dem genannten zweiten Satz von Datenwerten hält,

und wobei die genannte Vorrichtung die Aufgabe hat, in Bezug auf jeden der mehreren Sucheingabewerte, die von dem genannten Audiosignal abgeleitet und in dem genannten Datenspeicher gespeichert sind, eine Bisektionssuche in dem genannten ersten Satz von Datenwerten durchzuführen, um zu ermitteln, welcher der Datenwerte in dem genannten ersten Satz die beste Annäherung an den jeweiligen Eingangswert ist, wobei die genannte Bisektionssuche mit einer oder mehreren Iterationen durchgeführt wird, wobei der genannte Prozessor in Bezug auf jede Suchiteration Folgendes durchführt:

Lesen, beim Ausführen eines ersten Prozessorbefehls, des jeweiligen Eingangswerts aus dem Datenspeicher und Bewirken, dass der Eingangswert in dem genannten fünften Datenregister gespeichert wird, und Lesen des Datenwertes des genannten ersten Satzes, dessen Adresse in dem genannten Datenspeicher der in dem genannten ersten Datenregister gehaltenen Adresse entspricht, um den in dem genannten zweiten Datenregister gehaltenen aktuellen Adressenversatzwert versetzt, und Bewirken, dass der genannte Datenwert in dem genannten vierten Datenregister gespeichert wird, und

Berechnen, beim Ausführen eines zweiten Prozessorbefehls, der Differenz zwischen dem in dem genannten fünften Datenregister gespeicherten Eingangswert und dem genannten Datenwert des genannten, in dem genannten vierten Datenregister gespeicherten ersten Satzes, oder eines Derivats davon, und Bewirken, dass der resultierende Wert in dem genannten fünften Datenregister gespeichert wird, und Lesen des Datenwertes des genannten ersten Satzes, dessen Adresse in dem genannten Datenspeicher der im genannten sechsten Datenregister gehaltenen Adresse entspricht, und Bewirken, dass das Ergebnis in dem genannten zweiten Datenregister gespeichert wird, und

Justieren beim Ausführen eines dritten Prozessorbefehls nur dann, wenn die genannte in dem genannten fünften Datenregister gehaltene Differenz anzeigt, dass die nächste Iteration der Bisektionssuche in einer oberen Sektion des genannten ersten Satzes von Datenwerten stattfinden soll, der Speicheradresse in dem genannten ersten Datenregister um den in dem genannten dritten Datenregister gespeicherten Versatz und Bewirken, dass die resultierende justierte Speicheradresse in dem genannten ersten Datenregister (rThreshBase) gespeichert wird, Lesen des Datenwertes des genannten zweiten Datensatzes, dessen Adresse in dem genannten Datenspeicher der in dem genannten sechsten Datenregister gehaltenen Adresse entspricht, und Bewirken, dass das Ergebnis in dem genannten dritten Datenregister gespeichert wird, und Bewirken, dass die im Datenregister gehaltene Speicheradresse und bewirken, dass das Ergebnis in dem genannten dritten Datenregister gespeichert wird [sic], und Bewirken, dass die in dem genannten sechsten Datenregister gehaltene Speicheradresse inkrementiert wird, so dass sie auf den nächsten Datenwert in dem genannten zweiten Datensatz zeigt.

2. Vorrichtung nach Anspruch 1, wobei ein siebtes der genannten Datenregister (rDelta) beim Gebrauch einen aktuellen Wert eines Quantisierungsband-Skalierungsfaktors hält und wobei der genannte Prozessor beim Ausführen des genannten zweiten Prozessorbefehls die Differenz zwischen dem in dem genannten fünften Datenregister gespeicherten Eingangswert und dem genannten Datenwert des genannten ersten Satzes, der in dem genannten vierten Datenregister gespeichert ist, multipliziert mit dem genannten Skalierungsfaktor, berechnet und bewirkt, dass der resultierende Wert in dem genannten fünften Datenregister gespeichert wird.
3. Vorrichtung nach Anspruch 1 oder 2, wobei der genannte Prozessor beim Ausführen des genannten zweiten Prozessorbefehls die genannte Differenz durch Subtrahieren des genannten Datenwertes des genannten ersten Satzes, der in dem genannten vierten Datenregister gespeichert ist, oder eines Derivats davon, von dem in dem genannten fünften Datenregister gespeicherten Eingangswert berechnet, und wobei der Prozessor beim Ausführen des genannten dritten Prozessorbefehls die Speicheradresse in dem genannten ersten Datenregister nur dann justiert, wenn die genannte Differenz positiv ist.
4. Vorrichtung nach einem der vorherigen Ansprüche, wobei das genannte fünfte Datenregister ein Prozessorregister mit erweiterter Präzision ist, das den Ausgang von Multiplizier- und Multiplizier/Akkumulier-Operationen ohne Präzisionsverlust hält.

5. Vorrichtung nach einem der vorherigen Ansprüche, wobei der genannte erste Satz von Datenwerten in dem genannten Datenspeicher so sortiert wird, dass aufeinander folgende Datenwerte mit aufeinander folgenden Adressen in dem genannten Datenspeicher assoziiert werden.
- 5 6. Vorrichtung nach einem der vorherigen Ansprüche, wobei die genannten Datenwerte in dem genannten ersten Satz in dem genannten Datenspeicher sortiert und gespeichert werden, so dass ansteigende Datenwerte an Speicherstellen mit ansteigenden Adressen gespeichert werden.
- 10 7. Vorrichtung nach einem der vorherigen Ansprüche zum Erzeugen, mit Bezug auf jeden der genannten Sucheingangswertes, eines quantisierten Ausgangswertes, der N Bits umfasst, wobei N-1 Bits den quantisierten Wert repräsentieren und das übrige Bit anzeigt, ob der quantisierte N-1 Bit Wert positiv oder negativ ist.
8. Vorrichtung nach Anspruch 7, wobei der genannte zweite Datensatz wenigstens N-2 Datenwerte umfasst, wobei der erste Datenwert in dem genannten zweiten Satz den Wert 2^{N-3} annimmt, wobei die nachfolgenden N-3 Datenwerte aufeinander folgende abnehmende Zweierpotenzen sind.
- 15 9. Vorrichtung nach Anspruch 8, wobei der genannte zweite Datensatz N-1 Datenwerte einschließlich eines letzten Dummy-Datenwerts enthält, der z.B. auf null gesetzt ist.
- 20 10. Vorrichtung nach einem der Ansprüche 7 bis 9, wobei das genannte zweite und dritte Datenregister mit dem Wert 2^{N-2} initialisiert werden.
11. Vorrichtung nach einem der vorherigen Ansprüche, wobei die genannten Sucheingangswerte in einem separaten Datenspeicher oder wenigstens in einem separat zugängigen Teil eines gemeinsamen Datenspeichers zu dem genannten ersten Datensatz gehalten werden.
- 25 12. Vorrichtung nach einem der Ansprüche 5 bis 11, wobei jeder Datenwert des genannten ersten Satzes mit einem Indexwert assoziiert ist und wobei mehrere Iterationen der genannten Bisektionssuche durchgeführt werden, bis der kleinste Indexwert ($I_{\min}$) gefunden wird, für den der jeweilige Datenwert bei Multiplikation mit dem genannten Skalierungswert, gegebenenfalls, den Sucheingangswert übersteigt, oder alternativ, bis der größte Index gefunden wird, für den der jeweilige Datenwert gleich oder kleiner als der Sucheingangswert ist.
- 30 13. Vorrichtung nach einem der vorherigen Ansprüche, wobei die Speicheradresse des ersten Datenwertes vor einer ersten Iteration der genannten Bisektionssuche in dem genannten zweiten Datensatz in dem genannten sechsten Datenregister gespeichert wird.
- 35 14. Verfahren zum Wählen von Quantisierungsbändern für eine Vorrichtung nach Anspruch 1, wobei das Verfahren in Bezug auf jeden der mehreren Sucheingangswerte, die von dem genannten Audiosignal abgeleitet und in dem genannten Datenspeicher gespeichert sind, das Ausführen einer Bisektionssuche in dem genannten ersten Satz nach Datenwerten beinhaltet, um zu ermitteln, welcher der Datenwerte in dem genannten ersten Satz die beste Annäherung an den jeweiligen Eingangswert ist, wobei die genannte Bisektionssuche in einer oder mehreren Iterationen durchgeführt wird, wobei das genannte Verfahren in Bezug auf jede Suchiteration Folgendes beinhaltet:

45 Lesen, beim Ausführen eines ersten Prozessorbefehls, des jeweiligen Eingangswertes aus dem Datenspeicher und Bewirken, dass der Eingangswert in dem genannten fünften Datenregister gespeichert wird, Lesen des Datenwertes des genannten ersten Satzes, dessen Adresse in dem genannten Datenspeicher der in dem genannten ersten Datenregister gehaltenen Adresse entspricht, um den in dem genannten zweiten Datenregister gehaltenen aktuellen Adressversatzwert versetzt, und Bewirken, dass der genannte Datenwert in dem genannten vierten Datenregister gespeichert wird, und

50 Berechnen, beim Ausführen eines zweiten Prozessorbefehls, der Differenz zwischen dem in dem genannten fünften Datenregister gespeicherten Eingangswert und dem genannten Datenwert des genannten ersten Satzes, der in dem genannten vierten Datenregister gespeichert ist, oder eines Derivats davon, Bewirken, dass der resultierende Wert in dem genannten fünften Datenregister gespeichert wird, und Lesen des Datenwertes des genannten zweiten Satzes, dessen Adresse in dem genannten Datenspeicher der im genannten sechsten Datenregister gehaltenen Adresse entspricht, und Bewirken, dass das Ergebnis in dem genannten zweiten Datenregister gespeichert wird, und

55 Justieren, beim Ausführen eines dritten Prozessorbefehls, nur dann, wenn die genannte in dem genannten fünften Datenregister gehaltene Differenz anzeigt, dass die nächste Iteration der Bisektionssuche in einer oberen

Sektion des genannten ersten Satzes von Datenwerten stattfinden soll, der Speicheradresse in dem genannten ersten Datenregister um den in dem genannten dritten Datenregister gespeicherten Versatz und Bewirken, dass die resultierende justierte Speicheradresse in dem genannten ersten Datenregister (rThreshBase) gespeichert wird, Lesen des Datenwertes des genannten zweiten Satzes, dessen Adresse in dem genannten Datenspeicher der in dem genannten sechsten Datenregister gehaltenen Adresse entspricht, und Bewirken, dass das Ergebnis in dem genannten dritten Datenregister gespeichert wird, und Bewirken, dass die in dem genannten sechsten Datenregister gehaltene Speicheradresse so inkrementiert wird, dass sie auf den nächsten Datenwert in dem genannten zweiten Satz zeigt.

15. Computerprogrammprodukt, das Computerprogrammcode umfasst, der auf einem von einem Computer benutzbaren Medium gespeichert ist und bewirken soll, dass der Prozessor der Vorrichtung nach Anspruch 1 das Verfahren nach Anspruch 14 ausführt.

Revendications

1. Un appareil de compression d'un signal audio, l'appareil comprenant un processeur et une mémoire de données, le processeur comprenant une unité logique arithmétique et une pluralité de registres de données et étant programmable de façon à mettre en oeuvre une pluralité d'instructions de processeur, et où le processeur est capable d'exécuter, par rapport à une instruction de processeur unique, une opération arithmétique inconditionnelle et deux accès à la mémoire de données indépendants, ou une opération arithmétique conditionnelle et un accès à la mémoire de données, le processeur étant capable en outre d'amener une valeur contenue dans au moins un desdits registres de données qui est utilisé pour consulter ladite mémoire de données à être incrémentée, et où ledit appareil comprend en outre
 - un premier ensemble de valeurs de données conservée en mémoire dans ladite mémoire de données, ledit premier ensemble comprenant des valeurs de données représentant chacune une limite de bande de quantification respective et étant triées dans ladite mémoire de données selon leurs valeurs respectives, et
 - un deuxième ensemble de valeurs de données conservée en mémoire dans ladite mémoire de données, ledit deuxième ensemble comprenant au moins une valeur de données qui est une puissance de 2, les valeurs de données dudit deuxième ensemble étant triées dans ladite mémoire de données dans un ordre de valeur décroissante, et où
 - un premier desdits registres de données (rThreshBase) possède, en utilisation, une adresse mémoire pour une première valeur de données dudit premier ensemble, un deuxième desdits registres de données (rThreshOffset) possède, en utilisation, une valeur de décalage d'adresse courante, un troisième desdits registres de données (rOldThreshOffset) possède, en utilisation, une valeur de décalage d'adresse antérieure, un quatrième desdits registres de données (rThreshTableValue) possède, en utilisation, une valeur de données obtenue par ledit processeur à partir dudit premier ensemble de données, un cinquième desdits registres de données (rMAC) possède, en utilisation, une valeur de données qui est le résultat d'un ou plusieurs calculs exécutés par ledit processeur par rapport à la valeur de données contenue dans ledit quatrième registre, et un sixième desdits registres de données (IbisOff) possède une adresse mémoire d'une valeur de données dans ledit deuxième ensemble de valeurs de données,
 - et où, par rapport à chaque valeur d'une pluralité de valeurs d'entrée de recherche dérivées dudit signal audio et conservées en mémoire dans ladite mémoire de données, ledit appareil est agencé de façon à exécuter une recherche de bisection dudit premier ensemble de valeurs de données de façon à déterminer laquelle des valeurs de données dudit premier ensemble est la meilleure approximation de la valeur d'entrée respective, ladite recherche de bisection étant exécutée dans une ou plusieurs itérations où, par rapport à chaque itération de recherche, ledit processeur est agencé de façon :

lors de l'exécution d'une première instruction de processeur, à lire la valeur d'entrée respective à partir d'une mémoire de données et à amener la valeur d'entrée à être placée en mémoire dans ledit cinquième registre de données, et à lire la valeur de données dudit premier ensemble dont l'adresse dans ladite mémoire de données correspond à l'adresse contenue dans ledit premier registre de données, décalée de la valeur de décalage d'adresse courante contenue dans ledit deuxième registre de données, et à amener ladite valeur de données à être placée en mémoire dans ledit quatrième registre de données, et

lors de l'exécution d'une deuxième instruction de processeur, à calculer la différence entre la valeur d'entrée conservée en mémoire dans ledit cinquième registre de données et ladite valeur de données dudit premier ensemble qui est conservée en mémoire dans ledit quatrième registre de données, ou une valeur dérivée de celle-ci, et à amener la valeur résultante à être placée en mémoire dans ledit cinquième registre de données,

et à lire la valeur de données dudit deuxième ensemble dont l'adresse dans ladite mémoire de données correspond à l'adresse contenue dans ledit sixième registre de données et à amener le résultat à être placé en mémoire dans ledit deuxième registre de données, et

lors de l'exécution d'une troisième instruction de processeur, à ajuster, uniquement si ladite différence contenue dans ledit cinquième registre de données indique que l'itération suivante de la recherche de bisection devrait se dérouler dans une partie supérieure dudit premier ensemble de valeurs de données, l'adresse mémoire dans ledit premier registre de données selon le décalage conservée en mémoire dans ledit troisième registre de données et à amener l'adresse mémoire ajustée résultante à être placée en mémoire dans ledit premier registre de données (rThreshBase), à lire la valeur de données dudit deuxième ensemble dont l'adresse dans ladite mémoire de données correspond à l'adresse contenue dans ledit sixième registre de données et à amener le résultat à être placé en mémoire dans ledit troisième registre de données, et à amener l'adresse mémoire contenue dans le registre de données et à amener le résultat à être placé en mémoire dans ledit troisième registre de données [sic], et à amener l'adresse mémoire contenue dans ledit sixième registre de données à être incrémentée de façon à pointer vers la valeur de données suivante dans ledit deuxième ensemble.

2. Un appareil selon la Revendication 1, où un septième desdits registres de données (rDelta) possède, en utilisation, une valeur courante d'un facteur d'échelle de bande de quantification, et où, lors de l'exécution de ladite deuxième instruction de processeur, ledit processeur est agencé de façon à calculer la différence entre la valeur d'entrée conservée en mémoire dans ledit cinquième registre de données et ladite valeur de données dudit premier ensemble qui est conservée en mémoire dans ledit quatrième registre de données multipliée par ledit facteur d'échelle, et à amener la valeur résultante à être placée en mémoire dans ledit cinquième registre de données.
3. Un appareil selon la Revendication 1 ou 2, où, lors de l'exécution de ladite deuxième instruction de processeur, ledit processeur est agencé de façon à calculer ladite différence en soustrayant ladite valeur de données dudit premier ensemble qui est conservée en mémoire dans ledit quatrième registre de données, ou une valeur dérivée de celle-ci, de la valeur d'entrée conservée en mémoire dans ledit cinquième registre de données, et où, lors de l'exécution de ladite troisième instruction de processeur, le processeur est agencé de façon à ajuster l'adresse mémoire dans ledit premier registre de données uniquement si ladite différence est positive.
4. Un appareil selon l'une quelconque des Revendications précédentes, où ledit cinquième registre de données est un registre de processeur à précision étendue qui possède le produit d'opérations de multiplication et de multiplication/accumulation sans perte de précision.
5. Un appareil selon l'une quelconque des Revendications précédentes, où ledit premier ensemble de valeurs de données est trié dans ladite mémoire de données de sorte que des valeurs de données successives sont associées à des adresses successives dans ladite mémoire de données.
6. Un appareil selon l'une quelconque des Revendications précédentes, où lesdites valeurs de données dans ledit premier ensemble sont triées et conservées en mémoire dans ladite mémoire de données de sorte que des valeurs de données croissantes sont conservées en mémoire dans des emplacements mémoire avec des adresses croissantes.
7. Un appareil selon l'une quelconque des Revendications précédentes, agencé de façon à produire, par rapport à chacune desdites valeurs d'entrée de recherche, une valeur de sortie quantifiée comprenant N bits où N-1 bits représentent la valeur quantifiée et le bit restant indique si la valeur quantifiée de N-1 bits est positive ou négative.
8. Un appareil selon la Revendication 7, où ledit deuxième ensemble de données comprend au moins N-2 valeurs de données, la première valeur de données dans ledit deuxième ensemble prenant la valeur 2^{N-3} , les valeurs de données N-3 suivantes étant des puissances décroissantes successives de 2.
9. Un appareil selon la Revendication 8, où ledit deuxième ensemble de données comprend N-1 valeurs de données comprenant une valeur de donnée fictive finale, par exemple définie sur zéro.
10. Un appareil selon l'une quelconque des Revendications 7 à 9, où les deuxième et troisième registres de données sont initialisés avec la valeur 2^{N-2} .
11. Un appareil selon l'une quelconque des Revendications précédentes, où lesdites valeurs d'entrée de recherche sont contenues dans une mémoire de données séparée, ou au moins dans une partie accessible séparément d'une

mémoire de données commune, dudit premier ensemble de données.

5 12. Un appareil selon l'une quelconque des Revendications 5 à 11, où chaque valeur de données dudit premier ensemble est associée à une valeur d'indice et où plusieurs itérations de ladite recherche de bisection sont exécutées jusqu'à ce que la valeur d'indice la plus petite ($I_{\min}$) soit trouvée pour laquelle la valeur de données respective, lorsqu'elle est multipliée par ladite valeur d'échelle, si applicable, dépasse la valeur d'entrée de recherche, ou sinon jusqu'à ce que l'indice le plus élevé soit trouvé pour lequel la valeur de données respective est inférieure ou égale à la valeur d'entrée de recherche.

10 13. Un appareil selon l'une quelconque des Revendications précédentes, où, avant une première itération de ladite recherche de bisection, l'adresse mémoire de la première valeur de données dans ledit deuxième ensemble de données est placée en mémoire dans ledit sixième registre de données.

15 14. Un procédé de sélection de bandes de quantification pour un appareil selon la Revendication 1, le procédé comprenant, par rapport à chaque valeur d'une pluralité de valeurs d'entrée de recherche dérivées dudit signal audio et conservées en mémoire dans ladite mémoire de données, l'exécution d'une recherche de bisection dudit premier ensemble de valeurs de données de façon à déterminer laquelle des valeurs de données dans ledit premier ensemble est la meilleure approximation de la valeur d'entrée respective, ladite recherche de bisection étant exécutée dans une ou plusieurs itérations où, par rapport à chaque itération de recherche, ledit procédé comprend :

20 la lecture, lors de l'exécution d'une première instruction de processeur, de la valeur d'entrée respective à partir d'une mémoire de données et le fait d'amener la valeur d'entrée à être placée en mémoire dans ledit cinquième registre de données, la lecture de la valeur de données dudit premier ensemble dont l'adresse dans ladite mémoire de données correspond à l'adresse contenue dans ledit premier registre de données décalée de la valeur de décalage d'adresse courante contenue dans ledit deuxième registre de données, et le fait d'amener ladite valeur de données à être placée en mémoire dans ledit quatrième registre de données, et
25 le calcul, lors de l'exécution d'une deuxième instruction de processeur, de la différence entre la valeur d'entrée conservée en mémoire dans ledit cinquième registre de données et ladite valeur de données dudit premier ensemble qui est conservée en mémoire dans ledit quatrième registre de données, ou une valeur dérivée de celle-ci, le fait d'amener la valeur résultante à être placée en mémoire dans ledit cinquième registre de données, et la lecture de la valeur de données dudit deuxième ensemble dont l'adresse dans ladite mémoire de données correspond à l'adresse contenue dans ledit sixième registre de données et le fait d'amener le résultat à être placé en mémoire dans ledit deuxième registre de données, et
30 lors de l'exécution d'une troisième instruction de processeur, l'ajustement, uniquement si ladite différence contenue dans ledit cinquième registre de données indique que l'itération suivante de la recherche de bisection devrait se dérouler dans une partie supérieure dudit premier ensemble de valeurs de données, de l'adresse mémoire dans ledit premier registre de données selon ledit décalage conservée en mémoire dans ledit troisième registre de données et le fait d'amener l'adresse mémoire ajustée résultante à être placée en mémoire dans ledit premier registre de données (rThreshBase), la lecture de la valeur de données dudit deuxième ensemble dont l'adresse dans ladite mémoire de données correspond à l'adresse contenue dans ledit sixième registre de données et le fait d'amener le résultat à être placé en mémoire dans ledit troisième registre de données, et le fait d'amener l'adresse mémoire contenue dans ledit sixième registre de données à être incrémentée de façon à pointer vers la valeur de données suivante dans ledit deuxième ensemble.

35 15. Un programme informatique contenant du code de programme informatique conservé en mémoire dans un support utilisable par un ordinateur et agencé de façon à amener le processeur de l'appareil selon la Revendication 1 à exécuter le procédé selon la Revendication 14.

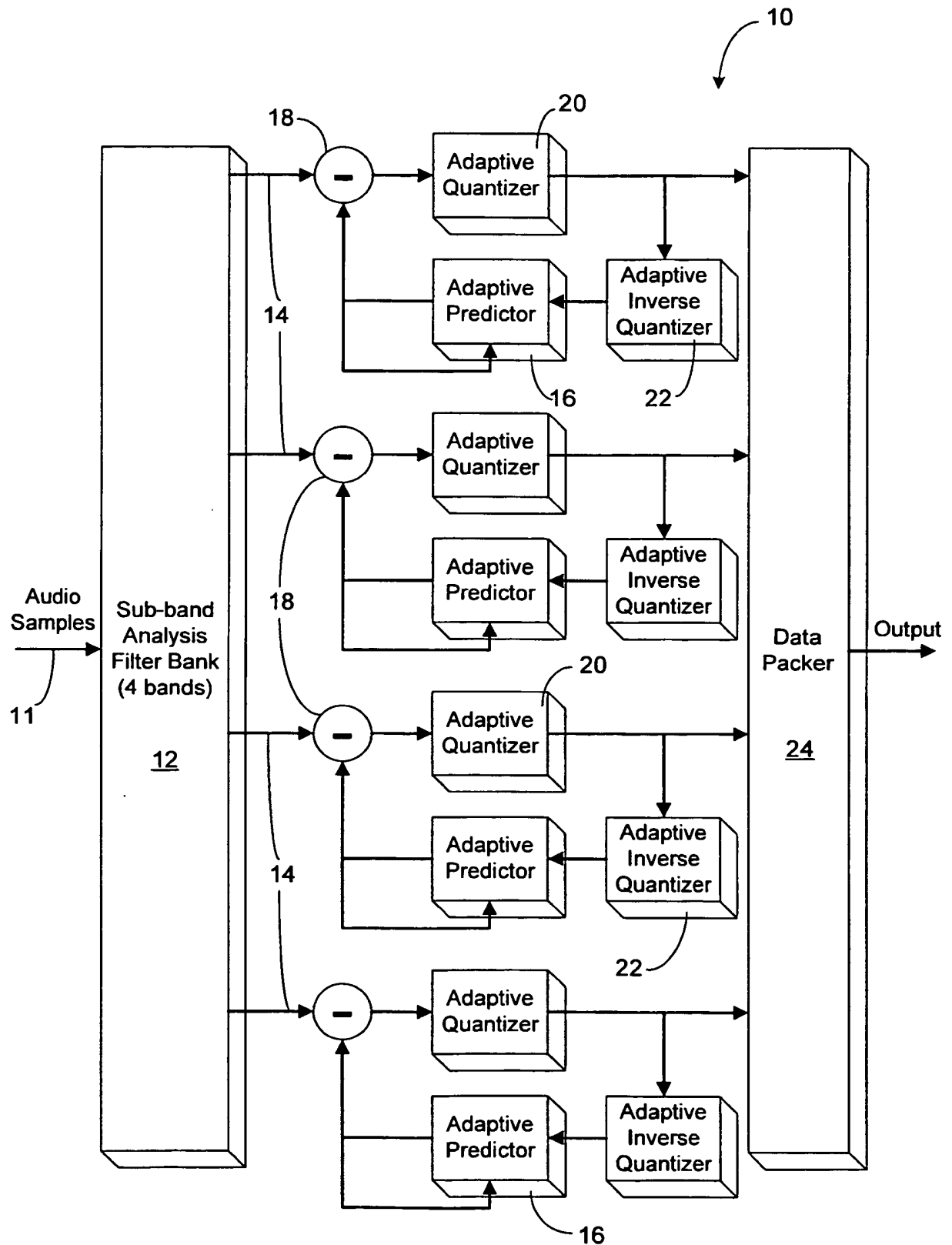


Fig. 1

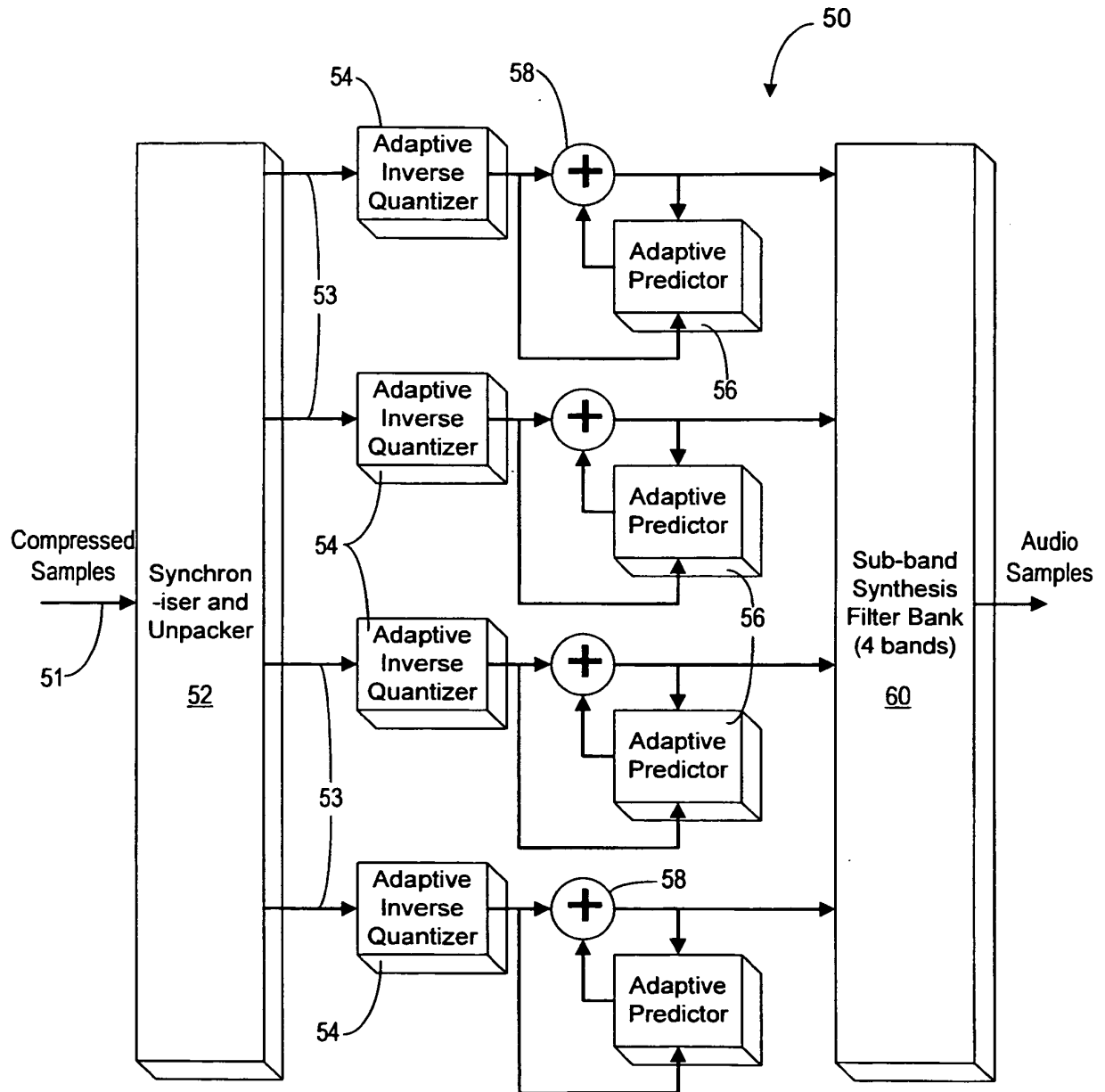


Fig. 2

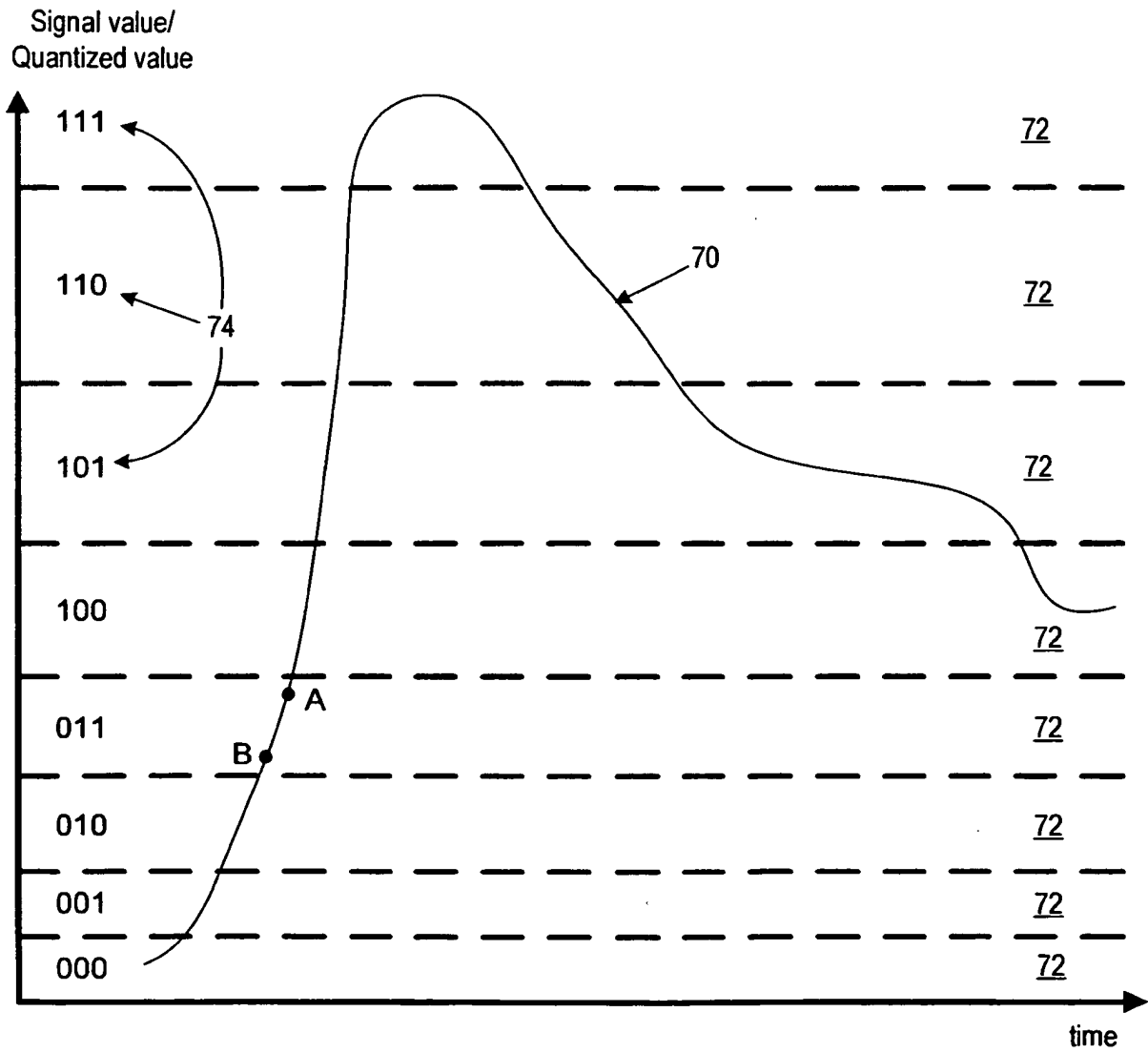


Fig. 3

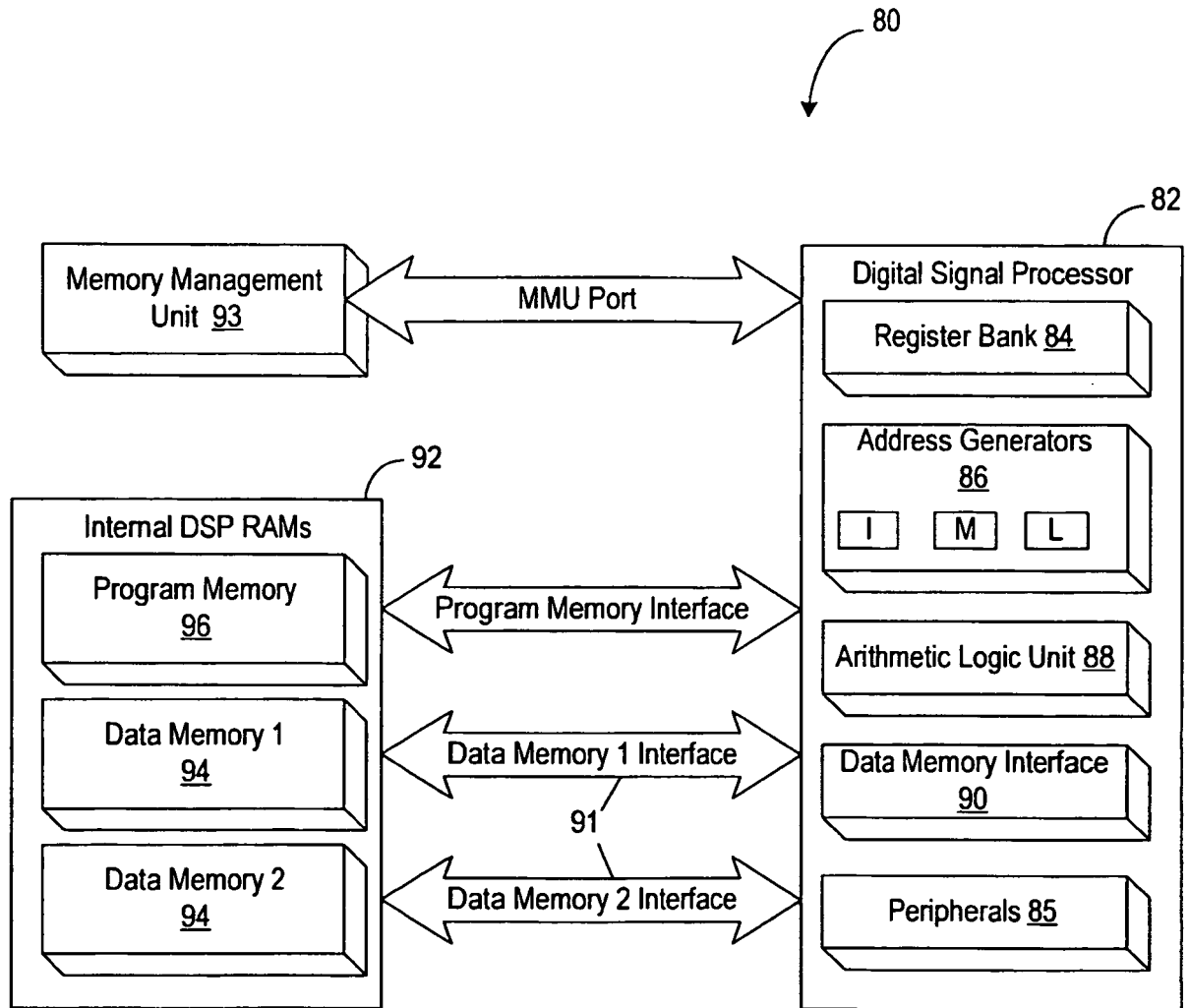


Fig. 4

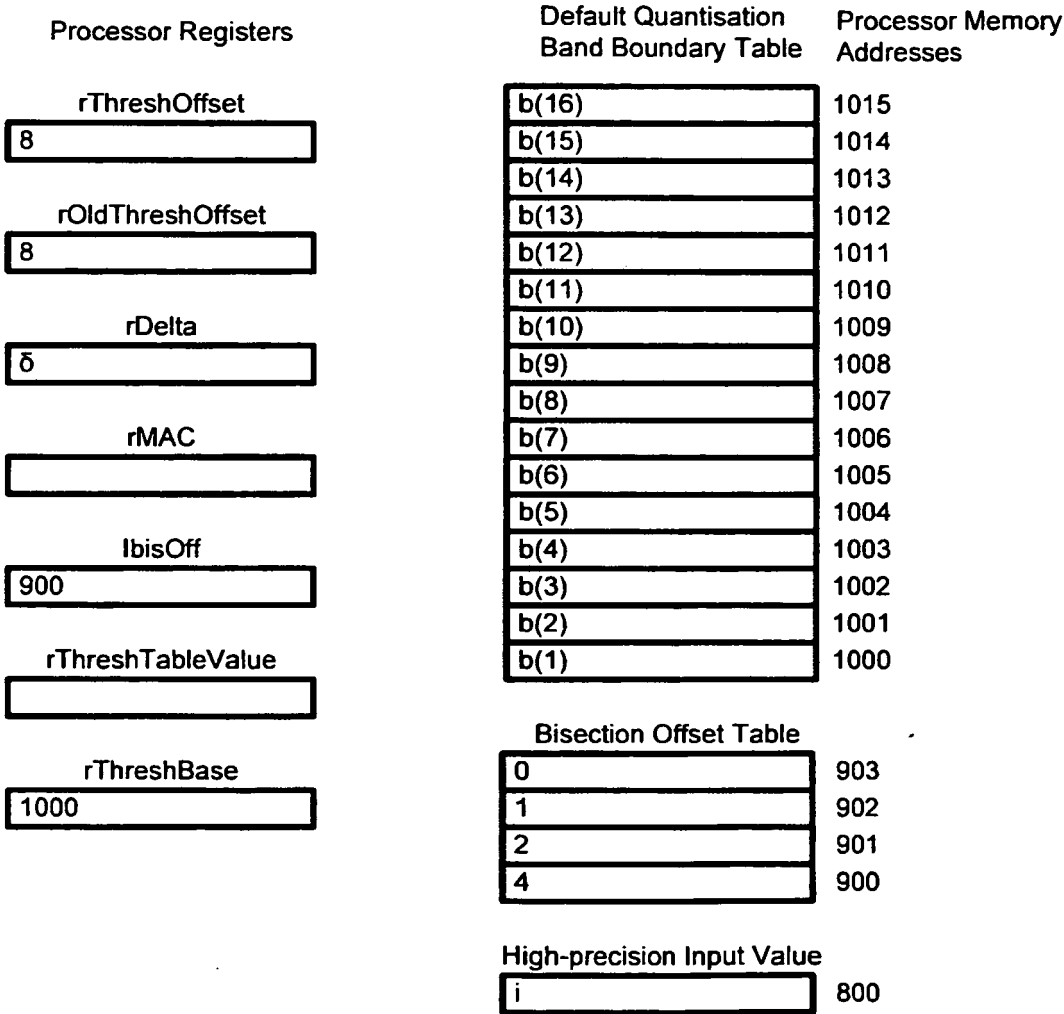
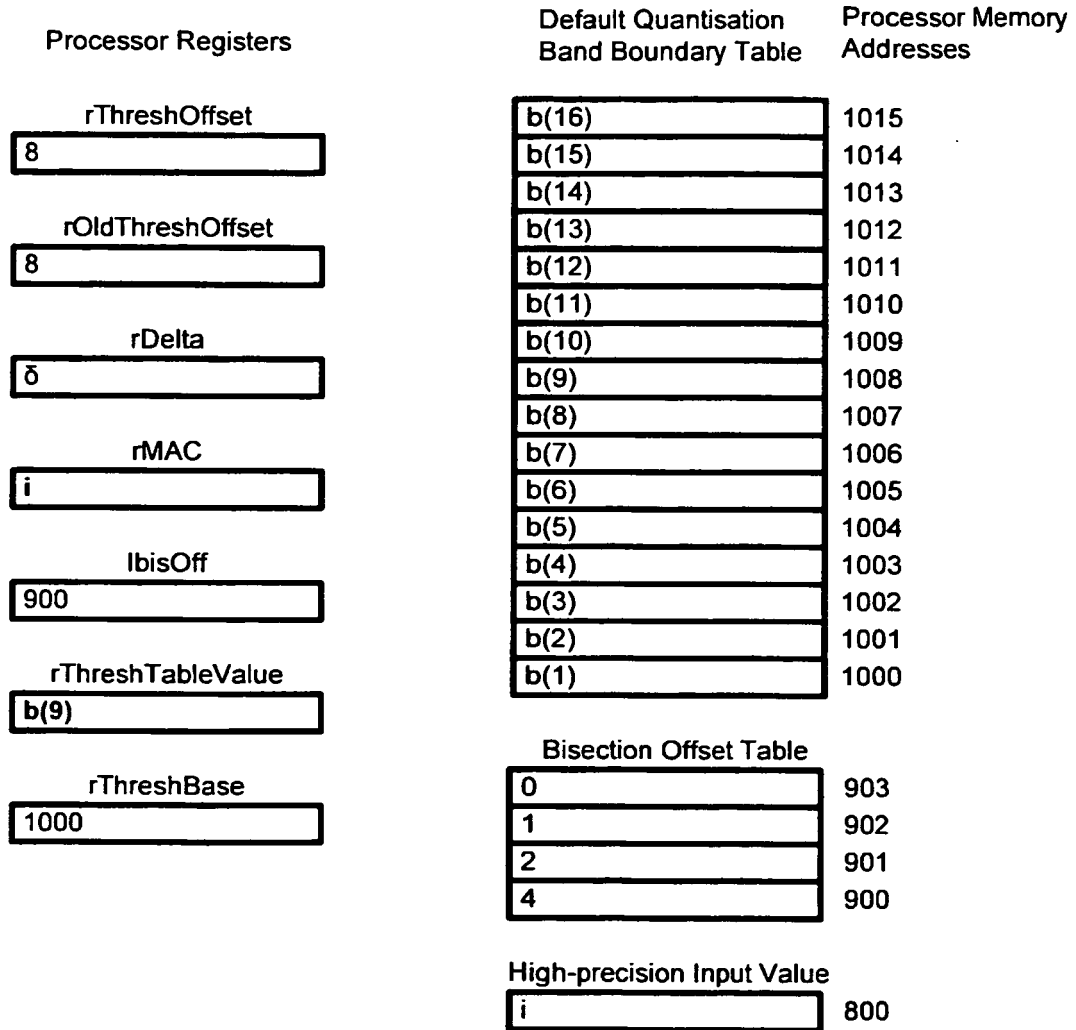


Fig. 5

Fig. 6

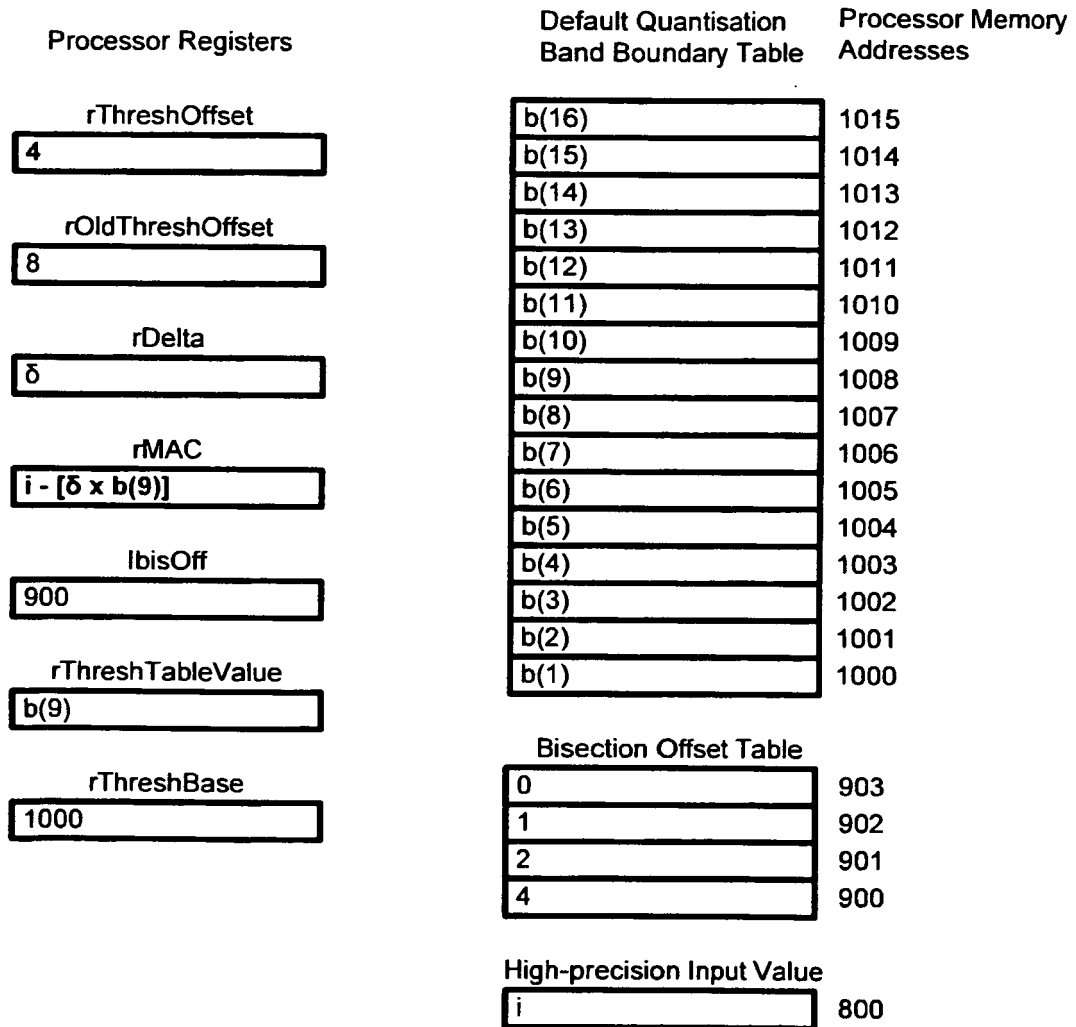


Fig. 7

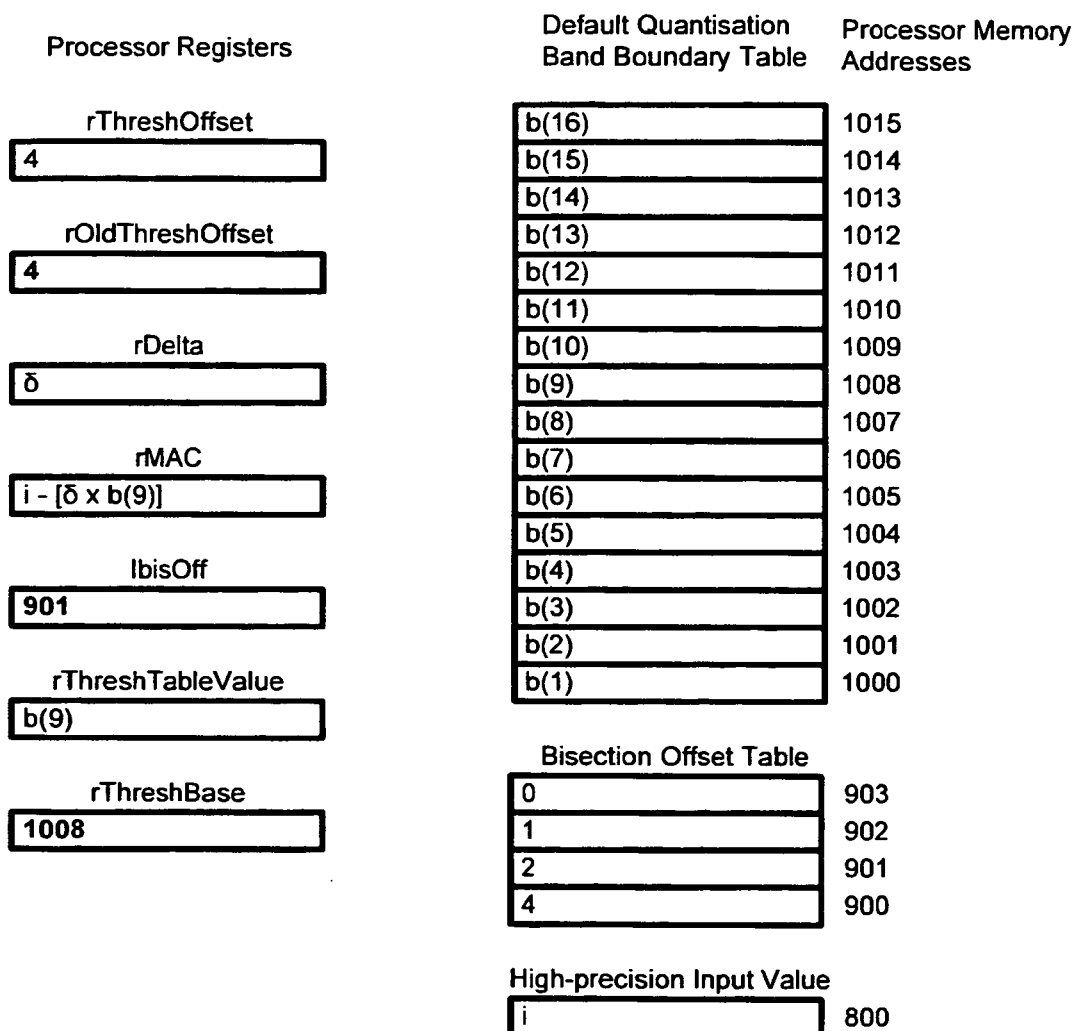


Fig. 8

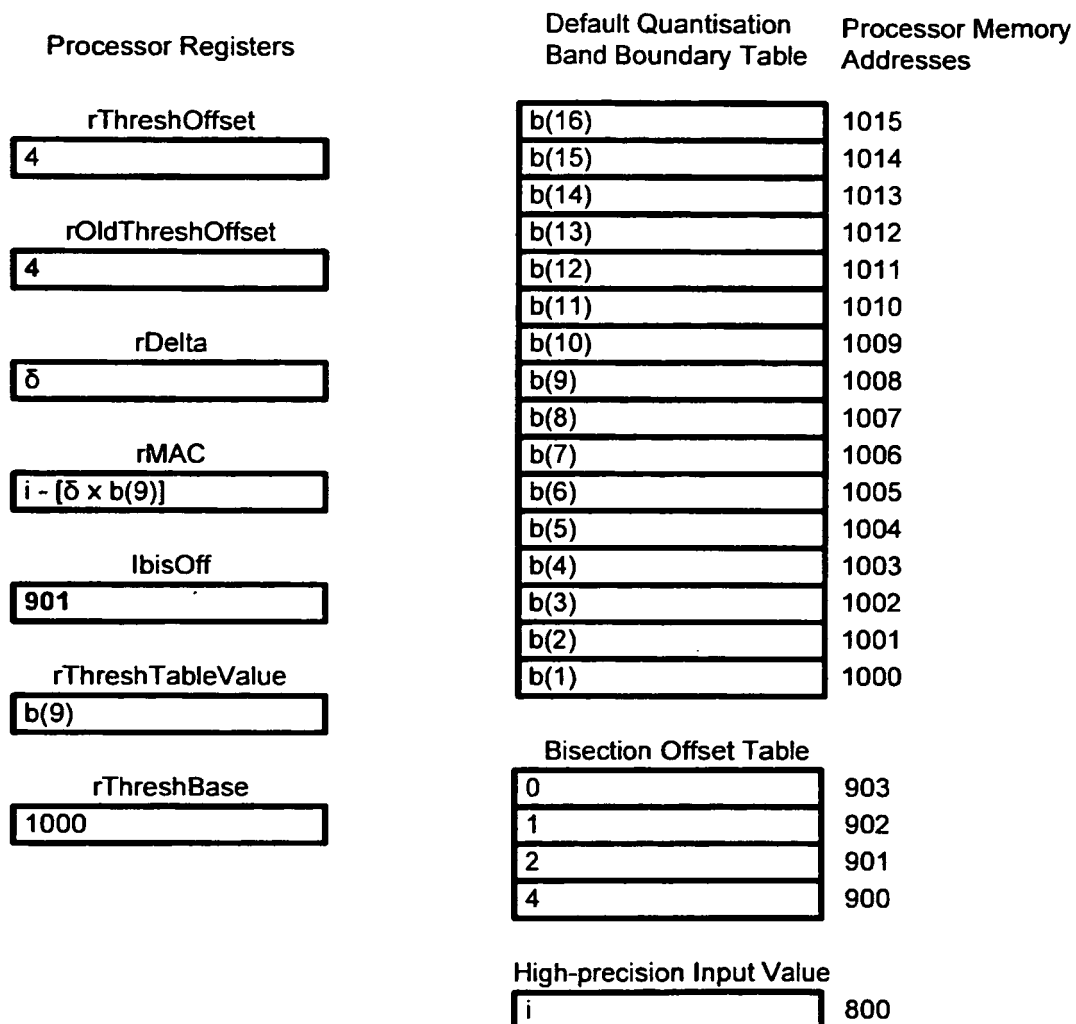


Fig. 9

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- WO 9962189 A2 [0007]