(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 60 2004 003 610 T2** 2007.04.05

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 465 158 B1**
(21) Deutsches Aktenzeichen: **60 2004 003 610.2**
(96) Europäisches Aktenzeichen: **04 251 796.1**
(96) Europäischer Anmeldetag: **26.03.2004**
(97) Erstveröffentlichung durch das EPA: **06.10.2004**
(97) Veröffentlichungstag
der Patenterteilung beim EPA: **13.12.2006**
(47) Veröffentlichungstag im Patentblatt: **05.04.2007**

(51) Int Cl.[8]: ***G10L 19/14*** (2006.01)

(30) Unionspriorität:
402938 **01.04.2003** **US**

(84) Benannte Vertragsstaaten:
AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LI, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR

(73) Patentinhaber:
Digital Voice Systems, Inc., Westford, Mass., US

(74) Vertreter:
Hofstetter, Schurack & Skora, 81541 München

(72) Erfinder:
Hardwick, John C., Sudbury, MA 01776, US

(54) Bezeichnung: **Halbrätiger Vocoder**

Beschreibung

TECHNISCHES GEBIET

[0001] Diese Beschreibt betrifft im Allgemeinen das Codieren und/oder Decodieren von Sprache, Ton und anderen Audiosignalen.

HINTERGRUND

[0002] Die Sprachcodierung und -decodierung weisen eine große Anzahl von Anwendungen auf und wurden ausgedehnt untersucht. Im allgemeinen strebt eine Sprachcodierung, die auch als Sprachkomprimierung bekannt ist, danach, die Datenrate zu verringern, die erforderlich ist, um ein Sprachsignal darzustellen, ohne die Qualität oder Verständlichkeit der Sprache wesentlich zu verringern. Sprachkomprimierungsverfahren können durch einen Sprachcodierer implementiert werden, der auch als Stimmcodierer oder Vocoder bezeichnet werden kann.

[0003] Ein Sprachcodierer wird im allgemeinen als einen Codierer und einen Decodierer umfassend betrachtet. Der Codierer erzeugt einen komprimierten Strom von Bits aus einer digitalen Darstellung von Sprache, wie er z.B. am Ausgang eines Analog-Digital-Wandlers mit einem von einem Mikrophon erzeugten analogen Signal als Eingang erzeugt werden kann. Der Decodierer wandelt den komprimierten Bitstrom in eine digitale Darstellung von Sprache, die sich zur Wiedergabe eignet, durch einen Digital-Analog-Wandler und einen Lautsprecher um. In vielen Anwendungen sind der Codierer und der Decodierer physikalisch getrennt und der Bitstrom wird zwischen ihnen unter Verwendung eines Datenübertragungskanals übertragen.

[0004] Ein Schlüsselparameter eines Sprachcodierers ist das Ausmaß an Komprimierung, das der Codierer erreicht, welches durch die Bitrate des Stroms von Bits, der vom Codierer erzeugt wird, gemessen wird. Die Bitrate des Codierers ist im allgemeinen eine Funktion der gewünschten Wiedergabetreue (d.h. Sprachqualität) und der Art des verwendeten Sprachcodierers. Verschiedene Arten von Sprachcodierern wurden dazu ausgelegt, mit verschiedenen Bitraten zu arbeiten. In letzter Zeit haben Sprachcodierer mit niedriger bis mittlerer Rate, die unterhalb 10 kbps arbeiten, bezüglich eines breiten Bereichs von Mobilkommunikationsanwendungen (z.B. Mobilfernsprechwesen, Satellitenfernsprechwesen, Landmobilfunk und Fernsprechwesen beim Flug) Aufmerksamkeit erlangt. Diese Anwendungen erfordern typischerweise Sprache mit hoher Qualität und Unempfindlichkeit gegen Fehler, die durch akustisches Rauschen und Kanalrauschen verursacht werden (z.B. Bitfehler).

[0005] Sprache wird im Allgemeinen als nicht-stationäres Signal mit Signaleigenschaften, die sich über die Zeit ändern, betrachtet. Diese Änderung der Signaleigenschaften ist im Allgemeinen mit Änderungen verknüpft, die an den Eigenschaften des Stimmapparats einer Person gemacht werden, um verschiedene Töne zu erzeugen. Ein Ton wird typischerweise für einen gewissen kurzen Zeitraum, typischerweise 10–100 ms, gehalten, und dann wird der Stimmapparat wieder geändert, um den nächsten Ton zu erzeugen. Der Übergang zwischen Tönen kann langsam und kontinuierlich sein oder er kann schnell sein wie im Fall eines Sprach-"Beginns". Diese Änderung der Signaleigenschaften erhöht die Schwierigkeit der Codierung von Sprache mit niedrigeren Bitraten, da einige Töne von Natur aus schwieriger zu codieren sind als andere, und der Sprachcodierer alle Töne mit angemessener Wiedergabetreue codieren können muss, während die Fähigkeit bewahrt wird, sich an einen Übergang in den Eigenschaften der Sprachsignale anzupassen. Die Leistung eines Sprachcodierers mit niedriger bis mittlerer Bitrate kann verbessert werden, indem zugelassen wird, dass sich die Bitrate ändert. In Sprachcodierern mit variabler Bitrate wird zugelassen, dass die Bitrate für jedes Segment von Sprache zwischen zwei oder mehr Optionen in Abhängigkeit von verschiedenen Faktoren, wie z.B. Benutzereingabe, Systembelastung, Endgerätkonstruktion oder Signaleigenschaften, variiert.

[0006] Es gab mehrere Hauptmethoden zum Codieren von Sprache mit niedrigen bis mittleren Datenraten. Eine Methode, die um die lineare vorhersagende Codierung (LPC) basiert, versucht beispielsweise, jeden neuen Datenblock von Sprache aus vorherigen Abtastwerten unter Verwendung von Kurz- und Langzeit-Vorhersageeinrichtungen vorherzusagen. Der Vorhersagefehler wird typischerweise unter Verwendung von einer von mehreren Methoden, von welchen CELP und/oder Mehrfachimpuls zwei Beispiele sind, quantisiert. Der Vorteil des Verfahrens der linearen Vorhersage besteht darin, dass es eine gute Zeitauflösung aufweist, die für die Codierung von stimmlosen Tönen hilfreich ist. Insbesondere profitieren Verschlusslaute und Übergänge insofern davon, als sie nicht in der Zeit übermäßig verschwommen werden. Die lineare Vorhersage hat jedoch typischerweise eine Schwierigkeit für stimmhafte Töne, indem die codierte Sprache gewöhnlich aufgrund einer unzureichenden Periodizität im codierten Signal rau oder heiser klingt. Dieses Problem kann bei niedrigeren

Datenraten signifikanter sein, die typischerweise eine längere Datenblockgröße erfordern, und für die die Langzeit-Vorhersageeinrichtung bei der Wiederherstellung der Periodizität weniger wirksam ist.

[0007] Eine weitere führende Methode für die Sprachcodierung mit niedriger bis mittlerer Rate ist ein Sprachcodierer oder Vocoder auf Modellbasis. Ein Vocoder modelliert Sprache als Reaktion eines Systems auf eine Anregung über kurze Zeitintervalle. Beispiele von Vocodersystemen umfassen Vocoder mit linearer Vorhersage, wie z.B. MELP, homomorphe Vocoder, Kanalvocoder, Sinustransformationscodierer ("STC"), harmonische Vocoder und Mehrbandanregungs- ("MBE") Vocoder. In diesen Vocodern wird Sprache in kurze Segmente (typischerweise 10–40 ms) unterteilt, wobei jedes Segment durch einen Satz von Modellparametern charakterisiert wird. Diese Parameter stellen typischerweise einige Grundelemente jedes Sprachsegments dar, wie z.B. die Tonhöhe, den Sprachzustand und die Spektralhüllkurve des Segments. Ein Vocoder kann eine von einer Anzahl von bekannten Darstellungen für jeden dieser Parameter verwenden. Die Tonhöhe kann beispielsweise als Tonhöhenperiode, als Grundfrequenz oder Tonhöhenfrequenz (die das Inverse der Tonhöhenperiode ist) oder als Langzeit-Vorhersageverzögerung dargestellt werden. Ebenso kann der Sprachzustand durch eine oder mehrere Sprachmetriken, durch ein Sprachwahrscheinlichkeitsmaß oder durch einen Satz von Sprachentscheidungen dargestellt werden. Die Spektralhüllkurve wird häufig durch eine Allpolfilter-Reaktion dargestellt, kann jedoch auch durch einen Satz von Spektralamplituden oder andere Spektralmessungen dargestellt werden. Da sie ermöglichen, dass ein Sprachsegment nur unter Verwendung einer kleinen Anzahl von Parametern dargestellt wird, können Sprachcodierer auf Modellbasis wie z.B. Vocoder typischerweise mit mittleren bis niedrigen Datenraten arbeiten. Die Qualität eines Systems auf Modellbasis hängt jedoch von der Genauigkeit des zugrunde liegenden Modells ab. Folglich muss ein Modell mit hoher Wiedergabetreue verwendet werden, wenn diese Sprachcodierer eine hohe Sprachqualität erzielen sollen.

[0008] Der MBE-Vocoder ist ein harmonischer Vocoder auf der Basis des MBE-Sprachmodells, von dem gezeigt wurde, dass es in vielen Anwendungen gut arbeitet. Der MBE-Vocoder kombiniert eine harmonische Darstellung für stimmhafte Sprache mit einer flexiblen, frequenzabhängigen Sprachstruktur auf der Basis des MBE-Sprachmodells. Dies ermöglicht, dass der MBE-Vocoder natürliche klingende stimmlose Sprache erzeugt, und macht den MBE-Vocoder gegen die Anwesenheit von akustischem Hintergrundrauschen unempfindlicher. Diese Eigenschaften ermöglichen, dass der MBE-Vocoder Sprache mit höherer Qualität mit niedrigen bis mittleren Datenraten erzeugt, und haben zu seiner Verwendung in einer Anzahl von kommerziellen Mobilkommunikationsanwendungen geführt.

[0009] Das MBE-Sprachmodell stellt Segmente von Sprache unter Verwendung einer Grundfrequenz, die der Tonhöhe entspricht, eines Satzes von Sprachmetriken oder -entscheidungen und eines Satzes von Spektralamplituden, die dem Frequenzgang des Stimmapparats entsprechen, dar. Das MBE-Modell verallgemeinert die herkömmliche einzelne V/UV-Entscheidung pro Segment zu einem Satz von Entscheidungen, die jeweils den Sprachzustand innerhalb eines speziellen Frequenzbandes oder Frequenzbereichs darstellen. Jeder Datenblock wird dadurch in zumindest stimmhafte und stimmlose Frequenzbereiche unterteilt. Diese hinzugefügte Flexibilität im Sprachmodell ermöglicht, dass das MBE-Modell gemischten Sprachtönen, wie z.B. einigen stimmhaften Reibelauten, besser gerecht wird, ermöglicht eine genauere Darstellung von Sprache, die durch akustisches Hintergrundrauschen verstümmelt wurde, und verringert die Empfindlichkeit gegen einen Fehler in irgendeiner Entscheidung. Eine ausgedehnte Prüfung hat gezeigt, dass diese Verallgemeinerung zu verbesserter Sprachqualität und Verständlichkeit führt.

[0010] Vocoder auf MBE-Basis umfassen den IMBE™-Sprachcodierer, der in einer Anzahl von drahtlosen Kommunikationssystemen verwendet wurde, einschließlich des Mobilfunkstandards des APCO Project 25 ("P25"). Dieser P25-Vocoder-Standard besteht aus einem IMBE™-Vocoder mit 7200 bps, der 4400 bps von komprimierten Sprachdaten mit 2800 bps von Vorwärtsfehlerkontroll- (FEC) Daten kombiniert. Er ist im Dokument der Telecommunications Industry Association (TIA) TIA-102BABA mit dem Titel "APCO Project 25 Vocoder Description" dokumentiert.

[0011] Der Codierer eines Sprachcodierers auf MBE-Basis schätzt einen Satz von Modellparametern für jedes Sprachsegment oder jeden Datenblock ab. Die MBE-Modellparameter umfassen eine Grundfrequenz (den Kehrwert der Tonhöhenperiode); einen Satz von V/UV-Metriken oder -Entscheidungen, die den Sprachzustand charakterisieren; und einen Satz von Spektralamplituden, die die Spektralhüllkurve charakterisieren. Nach dem Abschätzen der MBE-Modellparameter für jedes Segment quantisiert der Codierer die Parameter, um einen Datenblock von Bits zu erzeugen. Der Codierer kann wahlweise diese Bits mit Fehler-Korrektur/Erkennungs-Codes (FEC) vor dem Verschachteln und Übertragen des resultierenden Bitstroms zu einem entsprechenden Decodierer schützen.

[0012] Der Decodierer in einem Vocoder auf MBE-Basis rekonstruiert die MBE-Modellparameter (Grundfrequenz, Sprachinformationen und Spektralamplituden) für jedes Segment von Sprache aus dem empfangenen Bitstrom. Als Teil dieser Rekonstruktion kann der Decodierer eine Entschachtelung und eine Fehlerprüfdecodierung durchführen, um Bitfehler zu korrigieren und/oder zu erkennen. Außerdem führt der Decodierer typischerweise eine Phasenregeneration durch, um eine synthetische Phaseninformation zu berechnen. In einem in der Vocoder-Beschreibung des APCO Project 25 festgelegten und in den US-Patenten 5 081 681 und 5 664 051 beschriebenen Verfahren wird eine Zufallsphasenregeneration verwendet, wobei das Ausmaß der Zufälligkeit von den Sprachentscheidungen abhängt.

[0013] Der Decodierer verwendet die rekonstruierten MBE-Modellparameter, um ein Sprachsignal zu synthetisieren, das wahrnehmbar der ursprünglichen Sprache in einem hohen Grad ähnelt. Normalerweise werden separate Signalkomponenten, die stimmhafter, stimmloser und wahlweise gepulster Sprache entsprechen, für jedes Segment synthetisiert und die resultierenden Komponenten werden dann zusammenaddiert, um das synthetische Sprachsignal zu bilden. Dieser Prozess wird für jedes Segment von Sprache wiederholt, um das vollständige Sprachsignal zu reproduzieren, das dann über einen D-A-Wandler und einen Lautsprecher ausgegeben werden kann. Die stimmlose Signalkomponente kann unter Verwendung eines Fenster-Überlappungs-Additions-Verfahrens zum Filtern eines Signals von weißem Rauschen synthetisiert werden. Die zeitlich veränderliche Spektralhüllkurve des Filters wird aus der Sequenz von rekonstruierten Spektralamplituden in Frequenzbereichen bestimmt, die als stimmlos bezeichnet werden, wobei andere Frequenzbereiche auf Null gesetzt werden.

[0014] Der Decodierer kann die stimmhafte Signalkomponente unter Verwendung von einem von mehreren Verfahren synthetisieren. In einem Verfahren, das in der Vocoder-Beschreibung des APCO Project 25 festgelegt ist, wird ein Satz von Oberwellenoszillatoren verwendet, wobei ein Oszillator jeder Oberwelle der Grundfrequenz zugeordnet ist, und die Beiträge von allen Oszillatoren werden summiert, um die stimmhafte Signalkomponente zu bilden.

[0015] Der IMBE™-Vocoder mit 7200 bps, der für das Mobilfunk-Kommunikationssystem des APCO Project 25 standardisiert ist, verwendet 144 Bits, um jeden Datenblock von 20 ms darzustellen. Diese Bits werden in 56 redundante FEC-Bits (die als Kombination von Golay- und Hamming-Codes angewendet werden), 1 Synchronisationsbit und 87 MBE-Parameterbits unterteilt. Die 87 MBE-Parameterbits bestehen aus 8 Bits zum Quantisieren der Grundfrequenz, 3–12 Bits zum Quantisieren der binären Stimmhaft/Stimmlos-Entscheidungen und 67–76 Bits zum Quantisieren der Spektralamplituden. Der resultierende Datenblock aus 144 Bits wird vom Codierer zum Decodierer übertragen. Der Decodierer führt eine Fehlerkorrekturdecodierung vor dem Rekonstruieren der MBE-Modellparameter aus den fehlerdecodierten Bits durch. Der Decodierer verwendet dann die rekonstruierten Modellparameter, um stimmhafte und stimmlose Signalkomponenten zu synthetisieren, die zusammenaddiert werden, um das decodierte Sprachsignal zu bilden.

[0016] EP-A-893791 offenbart die Korrektur der empfindlichsten Gruppe von codierten Bits mit z.B. einem Golay-Code.

ZUSAMMENFASSUNG

[0017] Gemäß der Erfindung werden ein Verfahren zum Codieren, wie in Anspruch 1 dargelegt, und Verfahren zum Decodieren, wie in den Ansprüchen 20 und 31 dargelegt, bereitgestellt.

[0018] In einem allgemeinen Aspekt umfasst das Codieren einer Sequenz von digitalen Sprachabtastwerten in einen Bitstrom das Unterteilen der digitalen Sprachabtastwerte in einen oder mehrere Datenblöcke, das Berechnen von Modellparametern für einen Datenblock und das Quantisieren der Modellparameter, um Tonhöhenbits, die Tonhöheninformationen übermitteln, Sprachbits, die Sprachinformationen übermitteln, und Verstärkungsbits, die Signalpegelinformationen übermitteln, zu erzeugen. Ein oder mehrere der Tonhöhenbits werden mit einem oder mehreren der Sprachbits und einem oder mehreren der Verstärkungsbits kombiniert, um ein erstes Parametercodewort zu erzeugen, das mit einem Fehlerprüfcode codiert wird, um ein erstes FEC-Codewort zu erzeugen. Das erste FEC-Codewort wird in einen Bitstrom für den Datenblock aufgenommen.

[0019] Implementierungen können eines oder mehrere der folgenden Merkmale umfassen. Das Berechnen der Modellparameter für den Datenblock kann das Berechnen eines Grundfrequenzparameters, einer oder mehreren von Sprachentscheidungen und eines Satzes von Spektralparametern umfassen. Die Parameter können unter Verwendung des Mehrbandanregungs-Sprachmodells berechnet werden.

[0020] Das Quantisieren der Modellparameter kann die Erzeugung von Tonhöhenbits durch Anwenden einer logarithmischen Funktion auf den Grundfrequenzparameter und das Erzeugen der Sprachbits durch gemeinsames Quantisieren von Sprachentscheidungen für den Datenblock umfassen. Die Sprachbits können einen Index in ein Sprachcodebuch darstellen und der Wert des Sprachcodebuchs kann für zwei oder mehr verschiedene Werte des Index gleich sein.

[0021] Das erste Parametercodewort kann zwölf Bits umfassen. Das erste Parametercodewort kann beispielsweise durch Kombinieren von vier der Tonhöhenbits, vier der Sprachbits und vier der Verstärkungsbits gebildet werden. Das erste Parametercodewort kann mit einem Golay-Fehlerprüfcode codiert werden.

[0022] Die Spektralparameter können einen Satz von logarithmischen Spektralamplituden umfassen und die Verstärkungsbits können zumindest teilweise durch Berechnen des Mittelwerts der logarithmischen Spektralamplituden erzeugt werden. Die logarithmischen Spektralamplituden können in Spektralbits quantisiert werden; und zumindest einige der Spektralbits können kombiniert werden, um ein zweites Parametercodewort zu erzeugen, das mit einem zweiten Fehlerprüfcode codiert wird, um ein zweites FEC-Codewort zu erzeugen, das in den Bitstrom für den Datenblock aufgenommen werden kann.

[0023] Die Tonhöhenbits, die Sprachbits, die Verstärkungsbits und die Spektralbits werden jeweils in wichtigere Bits und weniger wichtige Bits unterteilt. Die wichtigeren Tonhöhenbits, Sprachbits, Verstärkungsbits und Spektralbits werden in das erste Parametercodewort und das zweite Parametercodewort aufgenommen und mit Fehlerprüfcodes codiert. Die weniger wichtigen Tonhöhenbits, Sprachbits, Verstärkungsbits und Spektralbits werden in den Bitstrom für den Datenblock ohne Codierung mit Fehlerprüfcodes aufgenommen. In einer Implementierung sind 7 Tonhöhenbits vorhanden, die in 4 wichtigere Tonhöhenbits und 3 weniger wichtige Tonhöhenbits unterteilt werden, es sind 5 Sprachbits vorhanden, die in 4 wichtigere Sprachbits und 1 weniger wichtiges Sprachbit unterteilt werden, und es sind 5 Verstärkungsbits vorhanden, die in 4 wichtigere Verstärkungsbits und 1 weniger wichtiges Verstärkungsbit unterteilt werden. Der zweite Parametercode kann zwölf wichtigere Spektralbits umfassen, die mit einem Golay-Fehlerprüfcode codiert werden, um das zweite FEC-Codewort zu erzeugen.

[0024] Ein Modulationsschlüssel kann aus dem ersten Parametercodewort berechnet werden und eine Verwürfelungssequenz kann aus dem Modulationsschlüssel erzeugt werden. Die Verwürfelungssequenz kann mit dem zweiten FEC-Codewort kombiniert werden, um ein verwürfeltes zweites FEC-Codewort zu erzeugen, das in den Bitstrom für den Datenblock aufgenommen werden soll.

[0025] Bestimmte Tonsignale können erfasst werden. Wenn ein Tonsignal für einen Datenblock erfasst wird, werden Tonidentifikatorbits und Tonamplitudenbits in das erste Parametercodewort aufgenommen. Die Tonidentifikatorbits ermöglichen, dass die Bits für den Datenblock als einem Tonsignal entsprechend identifiziert werden. Wenn ein Tonsignal für einen Datenblock erfasst wird, können zusätzliche Tonindexbits, die Frequenzinformationen für das Tonsignal bestimmen, in den Bitstrom für den Datenblock aufgenommen werden. Die Tonidentifikatorbits können einem nicht zugelassenen Satz von Tonhöhenbits entsprechen, um zu ermöglichen, dass die Bits für den Datenblock als einem Tonsignal entsprechend identifiziert werden. In bestimmten Implementierungen umfasst das erste Parametercodewort sechs Tonidentifikatorbits und sechs Tonamplitudenbits, wenn ein Tonsignal für einen Datenblock erfasst wird.

[0026] In einem anderen allgemeinen Aspekt umfasst das Decodieren von digitalen Sprachabtastwerten von einem Bitstrom das Unterteilen des Bitstroms in einen oder mehrere Datenblöcke von Bits, das Gewinnen eines ersten FEC-Codeworts von einem Datenblock von Bits und Fehlerprüfdecodieren des ersten FEC-Codeworts, um ein erstes Parametercodewort zu erzeugen. Tonhöhenbits, Sprachbits und Verstärkungsbits werden vom ersten Parametercodewort gewonnen. Die gewonnenen Tonhöhenbits werden verwendet, um zumindest teilweise Tonhöheninformationen für den Datenblock zu rekonstruieren, die gewonnene Sprachbits werden verwendet, um zumindest teilweise Sprachinformationen für den Datenblock zu rekonstruieren, und die gewonnenen Verstärkungsbits werden verwendet, um zumindest teilweise Signalpegelinformationen für den Datenblock zu rekonstruieren. Die rekonstruierten Tonhöheninformationen, Sprachinformationen und Signalpegelinformationen für einen oder mehrere Datenblöcke werden verwendet, um digitale Sprachabtastwerte zu berechnen.

[0027] Implementierungen können ein oder mehrere der vorstehend angegebenen Merkmale und ein oder mehrere der folgenden Merkmale umfassen. Die Tonhöheninformationen für einen Datenblock können beispielsweise einen Grundfrequenzparameter umfassen und die Sprachinformationen für einen Datenblock können eine oder mehrere Sprachentscheidungen umfassen. Die Sprachentscheidungen für den Datenblock kön-

nen unter Verwendung der Sprachbits als Index in ein Sprachcodebuch rekonstruiert werden. Der Wert des Sprachcodebuchs kann für zwei oder mehr verschiedene Indizes gleich sein.

[0028] Spektralinformationen für einen Datenblock können auch rekonstruiert werden. Die Spektralinformationen für einen Datenblock können zumindest teilweise einen Satz von logarithmischen Spektralamplitudenparametern umfassen. Die Signalpegelinformationen können verwendet werden, um den Mittelwert der logarithmischen Spektralamplitudenparameter zu bestimmen. Das erste FEC-Codewort kann mit einem Golay-Decodierer decodiert werden. Vier Tonhöhenbits, vier Sprachbits und vier Verstärkungsbits können vom ersten Parametercodewort gewonnen werden. Ein Modulationsschlüssel kann aus dem ersten Parametercodewort erzeugt werden, eine Verwürfelungssequenz kann aus dem Modulationsschlüssel berechnet werden und ein zweites FEC-Codewort kann aus dem Datenblock von Bits gewonnen werden. Die Verwürfelungssequenz kann auf das zweite FEC-Codewort angewendet werden, um ein entwürfeltes zweites FEC-Codewort zu erzeugen, das fehlerprüfdecodiert werden kann, um ein zweites Parametercodewort zu erzeugen. Die Spektralinformationen für einen Datenblock können zumindest teilweise aus dem zweiten Parametercodewort rekonstruiert werden.

[0029] Eine Fehlermetrik kann aus der Fehlerprüfdecodierung des ersten FEC-Codeworts und aus der Fehlerprüfdecodierung des entwürfelten zweiten FEC-Codeworts berechnet werden und eine Datenblockfehlerverarbeitung kann angewendet werden, wenn die Fehlermetrik einen Schwellenwert übersteigt. Die Datenblockfehlerverarbeitung kann das Wiederholen der rekonstruierten Modellparameter von einem vorherigen Datenblock für den aktuellen Datenblock umfassen. Die Fehlermetrik kann die Summe der Anzahl von Fehlern, die durch die Fehlerprüfdecodierung des ersten FEC-Codeworts und durch die Fehlerprüfdecodierung des entwürfelten zweiten FEC-Codeworts korrigiert werden, verwenden.

[0030] In einem weiteren allgemeinen Aspekt umfasst das Decodieren von digitalen Signalabtastwerten aus einem Bitstrom das Unterteilen des Bitstroms in einen oder mehrere Datenblöcke von Bits, das Gewinnen eines ersten FEC-Codeworts von einem Datenblock von Bits, Fehlerprüfdecodieren des ersten FEC-Codeworts, um ein erstes Parametercodewort zu erzeugen, und Verwenden des ersten Parametercodeworts, um festzustellen, ob der Datenblock von Bits einem Tonsignal entspricht. Wenn der Datenblock von Bits als einem Tonsignal entsprechend festgestellt wird, werden Tonamplitudenbits aus dem ersten Parametercodewort gewonnen. Ansonsten werden Tonhöhenbits, Sprachbits und Verstärkungsbits aus dem ersten Codewort gewonnen, wenn der Datenblock von Bits als nicht einem Tonsignal entsprechend festgestellt wird. Entweder die Tonamplitudenbits oder die Tonhöhenbits, Sprachbits und Verstärkungsbits werden verwendet, um digitale Signalabtastwerte zu berechnen.

[0031] Implementierungen können ein oder mehrere der vorstehend angegebenen Merkmale und ein oder mehrere der folgenden Merkmale umfassen. Ein Modulationsschlüssel kann beispielsweise aus dem ersten Parametercodewort erzeugt werden und eine Verwürfelungssequenz kann aus dem Modulationsschlüssel berechnet werden. Die Verwürfelungssequenz kann auf ein zweites FEC-Codewort angewendet werden, das aus dem Datenblock von Bits gewonnen wird, um ein entwürfeltes zweites FEC-Codewort zu erzeugen, das fehlerprüfdecodiert werden kann, um ein zweites Parametercodewort zu erzeugen. Digitale Signalabtastwerte können unter Verwendung des zweiten Parametercodeworts berechnet werden.

[0032] Die Anzahl von Fehlern, die durch die Fehlerprüfdecodierung des ersten FEC-Codeworts und durch die Fehlerprüfdecodierung des entwürfelten zweiten FEC-Codeworts korrigiert werden, kann summiert werden, um eine Fehlermetrik zu berechnen. Eine Datenblockfehlenrerarbeitung kann angewendet werden, wenn die Fehlermetrik eine Schwelle übersteigt. Die Datenblockfehlerverarbeitung kann das Wiederholen des rekonstruierten Modellparameters von einem vorherigen Datenblock umfassen.

[0033] Zusätzliche Spektralbits können aus dem zweiten Parametercodewort gewonnen werden und verwendet werden, um die digitalen Signalabtastwerte zu rekonstruieren. Die Spektralbits umfassen Tonindexbits, wenn der Datenblock von Bits als einem Tonsignal entsprechend festgestellt wird. Der Datenblock von Bits kann als einem Tonsignal entsprechend festgestellt werden, wenn einige der Bits im ersten Parametercodewort gleich einem bekannten Tonidentifikatorwert sind, der einem nicht zugelassenen Wert der Tonhöhenbits entspricht. Die Tonindexbits können verwendet werden, um zu identifizieren, ob der Datenblock von Bits einem Signalfrequenzton, einem MFV-Ton, einem Knox-Ton oder einem Hörton entspricht.

[0034] Die Spektralbits können verwendet werden, um einen Satz von logarithmischen Spektralamplitudenparametern für den Datenblock zu rekonstruieren, und die Verstärkungsbits können verwendet werden, um den Mittelwert der logarithmischen Spektralamplitudenparameter zu bestimmen.

[0035] Das erste FEC-Codewort kann mit einem Golay-Decodierer decodiert werden. Vier Tonhöhenbits plus vier Sprachbits plus vier Verstärkungsbits können aus dem ersten Parametercodewort gewonnen werden. Die Sprachbits können als Index in ein Sprachcodebuch verwendet werden, um Sprachentscheidungen für den Datenblock zu rekonstruieren.

[0036] In einem weiteren allgemeinen Aspekt umfasst das Decodieren eines Datenblocks von Bits in Sprachabtastwerte das Bestimmen der Anzahl von Bits im Datenblock von Bits, das Gewinnen von Spektralbits aus dem Datenblock von Bits und das Verwenden von einem oder mehreren der Spektralbits, um einen Spektralcodebuchindex zu bilden, wobei der Index zumindest teilweise durch die Anzahl von Bits im Datenblock von Bits bestimmt ist. Spektralinformationen werden unter Verwendung des Spektralcodebuchindex rekonstruiert und Sprachabtastwerte werden unter Verwendung der rekonstruierten Spektralinformationen berechnet.

[0037] Implementierungen können eines oder mehrere der vorstehend angegebenen Merkmale und eines oder mehrere der folgenden Merkmale umfassen. Tonhöhenbits, Sprachbits und Verstärkungsbits können beispielsweise auch aus dem Datenblock von Bits gewonnen werden. Die Sprachbits können als Index in ein Sprachcodebuch verwendet werden, um Sprachinformationen zu rekonstruieren, die auch verwendet werden, um die Sprachabtastwerte zu berechnen. Der Datenblock von Bits kann als einem Tonsignal entsprechend festgestellt werden, wenn einige der Tonhöhenbits und einige der Sprachbits gleich einem bekannten Tonidentifikatorwert sind. Die Spektralinformationen können einen Satz von logarithmischen Spektralamplitudenparametern umfassen und die Verstärkungsbits können verwendet werden, um den Mittelwert der logarithmischen Spektralamplitudenparameter zu bestimmen. Die logarithmischen Spektralamplitudenparameter für einen Datenblock können unter Verwendung der gewonnenen Spektralbits für den Datenblock in Kombination mit den rekonstruierten logarithmischen Spektralamplitudenparametern von einem vorherigen Datenblock rekonstruiert werden. Der Mittelwert der logarithmischen Spektralamplitudenparameter für einen Datenblock kann aus den gewonnenen Verstärkungsbits für den Datenblock und aus dem Mittelwert der logarithmischen Spektralamplitudenparameter eines vorherigen Datenblocks ermittelt werden. In bestimmten Implementierungen kann der Datenblock von Bits 7 Tonhöhenbits, die die Grundfrequenz darstellen, 5 Sprachbits, die Sprachentscheidungen darstellen, und 5 Verstärkungsbits, die den Signalpegel darstellen, umfassen.

[0038] Die Verfahren können verwendet werden, um einen "Halbraten"-MBE-Vocoder bereitzustellen, der mit 3600 pbs arbeitet und im Wesentlichen dieselbe oder eine bessere Leistung bereitstellen kann als der Standard-"Vollraten"-Vocoder mit 7200 pbs des APCO Project 25, selbst wenn der neue Vocoder mit der halben Datenrate arbeitet. Die viel niedrigere Datenrate für den Halbraten-Vocoder kann einen viel besseren Kommunikationswirkungsgrad (d.h. die Menge an HF-Spektrum, die für die Übertragung erforderlich ist) im Vergleich zum Standard-Vollraten-Vocoder bereitstellen.

[0039] In der verwandten Anmeldung Nummer 10/353 974, eingereicht am 30. Januar 2003, mit dem Titel "Voice Transcoder" und veröffentlicht als US-A-2004153316, ist ein Verfahren zum Bereitstellen einer Interoperabilität zwischen verschiedenen MBE-Vocodern offenbart. Dieses Verfahren kann angewendet werden, um eine Interoperabilität zwischen einer aktuellen Anlage, die den Vollraten-Vocoder verwendet, und einer neueren Anlage, die den hierin beschriebenen Halbraten-Vocoder verwendet, bereitzustellen. Implementierungen der vorstehend erörterten Verfahren können ein Verfahren oder einen Prozess, ein System oder eine Vorrichtung oder eine Computersoftware auf einem für den Computer zugänglichen Medium umfassen. Weitere Merkmale sind aus der folgenden Beschreibung, einschließlich der Zeichnungen, und den Ansprüchen ersichtlich.

BESCHREIBUNG DER ZEICHNUNGEN

[0040] Fig. 1 ist ein Blockdiagramm einer Anwendung eines MBE-Vocoders.

[0041] Fig. 2 ist ein Blockdiagramm einer Implementierung eines Halbraten-MBE-Vocoders mit einem Codierer und einem Decodierer.

[0042] Fig. 3 ist ein Blockdiagramm eines MBE-Parameterabschätzers, wie er z.B. im Halbraten-MBE-Codierer von Fig. 2 verwendet werden kann.

[0043] Fig. 4 ist ein Blockdiagramm einer Implementierung eines MBE-Parameterquantisierers, wie er z.B. im Halbraten-MBE-Codierer von Fig. 2 verwendet werden kann.

[0044] Fig. 5 ist ein Blockdiagramm einer Implementierung eines Halbraten-MBE-Quantisierers für logarithmische Spektralamplituden des Halbraten-MBE-Codierers von Fig. 2.

[0045] Fig. 6 ist ein Blockdiagramm eines Spektralamplituden-Vorhersageabweichungsquantisierers des Halbraten-MBE-Codierers von Fig. 2.

AUSFÜHRLICHE BESCHREIBUNG

[0046] Fig. 1 zeigt ein Sprachcodierer- oder Vocodersystem **100**, das analoge Sprache oder irgendein anderes Signal von einem Mikrophon **105** abtastet. Ein Analog-Digital- ("A-D") Wandler **110** digitalisiert die abgetastete Sprache, um ein digitales Sprachsignal zu erzeugen. Die digitale Sprache wird durch eine MBE-Sprachcodiereinheit **115** verarbeitet, um einen digitalen Bitstrom **120** zu erzeugen, der zur Übertragung oder Speicherung geeignet ist. Typischerweise verarbeitet der Sprachcodierer das digitale Sprachsignal in kurzen Datenblöcken. Jeder Datenblock von digitalen Sprachabtastwerten erzeugt einen entsprechenden Datenblock von Bits im Bitstrom, der aus dem Codierer ausgegeben wird. In einer Implementierung ist die Datenblockgröße 20 ms in der Dauer und besteht aus 160 Abtastwerten mit einer Abtastrate von 8 kHz. Die Leistung kann in einigen Anwendungen durch Unterteilen von jedem Datenblock in zwei Unterdatenblöcke mit 10 ms erhöht werden.

[0047] Fig. 1 stellt auch einen empfangenen Bitstrom **125** dar, der in eine MBE-Sprachdecodiereinheit **130** eintritt, die jeden Datenblock von Bits verarbeitet, um einen entsprechenden Datenblock von synthetisierten Sprachabtastwerten zu erzeugen. Eine Digital-Analog- ("D-A") Wandlereinheit **135** wandelt dann die digitalen Sprachabtastwerte in ein analoges Signal um, das an eine Lautsprechereinheit **140** zur Umwandlung in ein akustisches Signal, das für das menschliche Hören geeignet ist, weitergeleitet werden kann.

[0048] Fig. 2 zeigt einen MBE-Vocoder, der eine MBE-Codiereinheit **200** umfasst, die eine Parameterabschätzeinheit **205** verwendet, um verallgemeinerte MBE-Modellparameter für jeden Datenblock abzuschätzen. Die Parameterabschätzeinheit **205** erfasst auch bestimmte Tonsignale und gibt Tondaten mit einem Sprach/Ton-Kennzeichen aus. Die Ausgaben für einen Datenblock werden dann entweder von der MBE-Parameterquantisierungseinheit **210**, um Sprachbits zu erzeugen, oder von einer Tonquantisierungseinheit **215**, um Tonbits zu erzeugen, in Abhängigkeit davon, ob ein Tonsignal für den Datenblock erfasst wurde, verarbeitet. Eine Auswahleinheit **220** wählt die geeigneten Bits (Tonbits, wenn ein Tonsignal erfasst wird, oder Sprachbits, wenn kein Tonsignal erfasst wird) aus und die ausgewählten Bits werden an eine FEC-Codiereinheit **225** ausgegeben, die die Quantisiererbits mit redundanten Vorwärtsfehlerkorrektur- ("FEC") Daten kombiniert, um das übertragene Bit für den Datenblock zu bilden. Das Hinzufügen von redundanten FEC-Daten ermöglicht, dass der Decodierer Bitfehler korrigiert und/oder erkennt, die durch Verschlechterung im Übertragungskanal verursacht werden. In bestimmten Implementierungen erfasst die Parameterabschätzeinheit **205** keine Tonsignale und die Tonquantisierungseinheit **215** und die Auswahleinheit **220** sind nicht vorgesehen.

[0049] In einer Implementierung wurde ein MBE-Vocoder mit 3600 bps, der sich gut zur Verwendung in der Funkanlage der nächsten Generation eignet, entwickelt. Diese Halbraten-Implementierung verwendet einen Datenblock mit 20 ms, der 72 Bits enthält, wobei die Bits in 23 FEC-Bits und 49 Sprach- oder Tonbits unterteilt werden. Die 23 FEC-Bits werden aus einem erweiterten [24,12]-Golay-Code und einem [23,12]-Golay-Code gebildet. Die FEC-Bits schützen die 24 empfindlichsten Bits des Datenblocks und können gewisse Bitfehlermuster in diesen geschützten Bits korrigieren und/oder erkennen. Die restlichen 25 Bits sind gegen Bitfehler weniger empfindlich und werden nicht geschützt. Die Sprachbits werden in 7 Bits zum Quantisieren der Grundfrequenz, 5 Bits zur Vektorquantisierung der Sprachentscheidungen über 8 Frequenzbänder und 37 Bits zum Quantisieren der Spektralamplituden unterteilt. Um die Fähigkeit zu steigern, Bitfehler in den empfindlichsten Bits zu erkennen, wird eine von den Daten abhängige Verwürfelung auf den [23,12]-Golay-Code innerhalb der FEC-Codiereinheit **225** angewendet. Eine pseudozufällige Verwürfelungssequenz wird aus einem Modulationsschlüssel auf der Basis der 12 Eingangsbits in den [24,12]-Golay-Code erzeugt. Ein Ausschließlich-ODER wird dann verwendet, um diese Verwürfelungssequenz mit den 23 Ausgangsbits aus dem [23,12]-Golay-Codierer zu kombinieren. Eine von den Daten abhängige Verwürfelung ist in den US-Patenten 5 870 405 und 5 517 511 beschrieben. Eine [4x18]-Zeilen-Spalten-Verschachtelungseinrichtung wird auch angewendet, um den Effekt von Blockfehlern zu verringern.

[0050] Fig. 2 zeigt auch ein Blockdiagramm einer MBE-Decodiereinheit **230**, die einen Datenblock von Bits verarbeitet, der von einem empfangenen Bitstrom erhalten wird, um ein digitales Ausgangssprachsignal zu erzeugen. Der MBE-Decodierer umfasst eine FEC-Decodiereinheit **235**, die Bitfehler im empfangenen Bitstrom korrigiert und/oder erkennt, um Sprach- oder Tonquantisiererbits zu erzeugen. Die FEC-Decodiereinheit umfasst typischerweise eine von den Daten abhängige Entwürfelung und Entschachtelung, wie erforderlich, um die durch den FEC-Codierer durchgeführten Schritte umzukehren. Die FEC-Decodiereinheit **235** kann wahlweise Weichentscheidungsbits verwenden, wobei jedes empfangene Bit unter Verwendung von mehr als zwei

möglichen Pegeln dargestellt wird, um die Fehlerprüfdecodierleistung zu verbessern. Die Quantisiererbits für den Datenblock werden durch die FEC-Decodiereinheit **235** ausgegeben und von einer Parameterrekonstruktionseinheit **240** verarbeitet, um die MBE-Modellparameter oder Tonparameter für den Datenblock durch Invertieren der durch den Codierer angewendeten Quantisierungsschritte zu rekonstruieren. Die resultierenden MBE- oder Tonparameter werden dann von einer Sprachsyntheseeinheit **245** verwendet, um ein synthetisches digitales Sprachsignal oder Tonsignal zu erzeugen, das das Ausgangssignal des Decodierers ist.

[0051] In der beschriebenen Implementierung invertiert die FEC-Decodiereinheit **235** die von den Daten abhängige Verwürfelungsoperation durch zuerst Decodieren des [24,12]-Golay-Codes, auf den keine Verwürfelung angewendet wird, und dann Verwenden der 12 Ausgangsbits aus dem [24,12]-Golay-Decodierer, um einen Modulationsschlüssel zu berechnen. Dieser Modulationsschlüssel wird dann verwendet, um eine Verwürfelungssequenz zu berechnen, die auf die 23 Eingangsbits vor der Decodierung des [23,12]-Golay-Codes angewendet wird. Unter der Annahme, dass der (24,12]-Golay-Code (der die wichtigsten Daten enthält) korrekt decodiert wird, wird dann die durch den Codierer angewendete Verwürfelungssequenz vollständig entfernt. Wenn jedoch der [24,12]-Golay-Code nicht korrekt decodiert wird, dann kann die durch den Codierer angewendete Verwürfelungssequenz nicht entfernt werden, was verursacht, dass viele Fehler durch den [23,12]-Golay-Decodierer gemeldet werden. Diese Eigenschaft wird vom FEC-Decodierer verwendet, um Datenblöcke zu erfassen, in denen die ersten 12 Bits falsch decodiert worden sein können.

[0052] Der FEC-Decodierer summiert die Anzahl von korrigierten Fehlern, die von beiden Golay-Decodierern gemeldet werden. Wenn diese Summe größer als oder gleich 6 ist, dann wird der Datenblock für ungültig erklärt und der aktuelle Datenblock von Bits wird während der Synthese nicht verwendet. Statt dessen führt die MBE-Syntheseeinheit **235** eine Datenblockwiederholung oder eine Stummschaltoperation nach drei aufeinander folgenden Datenblockwiederholungen durch. Während einer Datenblockwiederholung werden decodierte Parameter von einem vorherigen Datenblock für den aktuellen Datenblock verwendet. Ein "Komfortrausch"-Signal mit niedrigen Pegel wird während einer Stummschaltoperation ausgegeben.

[0053] In einer Implementierung sind der in Fig. 2 gezeigte Halbraten-Vocoder, die MBE-Parameterabschätzeinheit **205** und die MBE-Syntheseeinheit **235** im Allgemeinen dieselben wie entsprechende Einheiten im Vollraten-Vocoder von APCO 25 mit 7200 bps, der in der Vocoder-Beschreibung von APCO Project 25 (TIA-102BABA) beschrieben ist. Die gemeinsame Nutzung dieser Elemente zwischen dem Vollraten-Vocoder und dem Halbraten-Vocoder verringert den zum Implementieren beider Vocoder erforderlichen Speicher und verringert dadurch die Kosten der Implementierung beider Vocoder in derselben Anlage. Außerdem kann die Interoperabilität in dieser Implementierung unter Verwendung der MBE-Umcodierer-Verfahren verbessert werden, die in der gleichzeitig anhängigen, veröffentlichten Anmeldung US-A-2004153316 offenbart sind, die am 30. Januar 2003 eingereicht wurde und den Titel "Voice Transcoder" hat. Alternative Implementierungen können verschiedene Analyse- und Syntheseverfahren umfassen, um die Qualität zu verbessern, während sie mit dem hierin beschriebenen Halbraten-Bitstrom interoperierbar sind. Ein Sprachmodell mit drei Zuständen (stimmhaft, stimmlos oder gepulst) kann beispielsweise verwendet werden, um die Verzerrung für Verschlusslaut- und andere vorübergehend Töne zu verringern, während es unter Verwendung des Verfahrens, das in der gleichzeitig anhängigen US-Anmeldung 10/292 460 beschrieben ist, die am 13. November 2002 eingereicht wurde und den Titel "Interoperable Vocoder" hat, interoperierbar bleibt. Ebenso kann ein Sprachaktivitätsdetektor (VAD) hinzugefügt werden, um Sprache von Hintergrundrauschen zu unterscheiden, und/oder eine Rauschunterdrückung kann hinzugefügt werden, um die wahrgenommene Menge an Hintergrundrauschen zu verringern. Eine weitere alternative Implementierung tauscht verbesserte Tonhöhen- und Sprachabschätzungsverfahren aus, wie z.B. die in den US-Patenten 5 826 222 und 5 715 365 beschriebenen, um die Sprachqualität zu verbessern.

[0054] Fig. 3 zeigt einen MBE-Parameterabschätzer **300**, der eine Implementierung der MBE-Parameterabschätzeinheit **205** von Fig. 2 darstellt. Ein Hochpassfilter **305** filtert ein digitales Sprachsignal, um irgendeinen Gleichspannungspegel vom Signal zu entfernen. Als nächstes wird das gefilterte Signal durch eine Tonhöhenabschätzeinheit **310** verarbeitet, um eine anfängliche Tonhöhenabschätzung für jeden Datenblock von 20 ms zu bestimmen. Die gefilterte Sprache wird auch zu einer Ausschnittdarstellungs- und FFT-Einheit **315** geliefert, die die gefilterte Sprache mit einer Fensterfunktion, wie z.B. einem Hamming-Fenster mit 221 Punkten, multipliziert und eine FFT verwendet, um das Spektrum der ausschnittweise dargestellten Sprache zu berechnen.

[0055] Die anfängliche Tonhöhenabschätzung und das Spektrum werden dann durch einen Grundfrequenzabschätzer **320** weiter verarbeitet, um die Grundfrequenz f_0 und die zugehörige Anzahl von Oberwellen ($L = 0{,}4627/f_0$) für den Datenblock zu berechnen, wobei 0,4627 die typische Vocoderbandbreite, die durch die Abtastrate normiert wird, darstellt. Diese Parameter werden dann mit dem Spektrum durch einen Sprachentschei-

dungsgenerator **325**, der die Sprachmaße V_l berechnet, und einen Spektralamplitudengenerator **330**, der die Spektralamplituden M_l für jede Oberwelle $1 \leq l \leq L$ berechnet, weiter verarbeitet.

[0056] Das Spektrum kann wahlweise durch eine Tonerfassungseinheit **335**, die bestimmte Tonsignale erfasst, wie beispielsweise Einfrequenztöne, MFV-Töne und Hörtöne, weiter verarbeitet werden. Tonerfassungsverfahren sind gut bekannt und können durch Suchen nach Spitzen im Spektrum und Feststellen, dass ein Tonsignal vorliegt, wenn die Energie um eine oder mehrere aufgefundene Spitzen eine gewisse Schwelle (beispielsweise 99%) der Gesamtenergie im Spektrum übersteigt, durchgeführt werden. Die aus dem Tonerfassungselement ausgegebenen Tondaten umfassen typischerweise ein Sprach/Ton-Kennzeichen, einen Tonindex, um den Ton zu identifizieren, wenn das Sprach/Ton-Kennzeichen angibt, dass ein Tonsignal erfasst wurde, und die abgeschätzte Tonamplitude A_{TON}.

[0057] Die Ausgabe **340** der MBE-Parameterabschätzung umfasst die MBE-Parameter in Kombination mit beliebigen Tondaten.

[0058] Das in Fig. 3 gezeigte MBE-Parameterabschätzverfahren folgt eng dem Verfahren, das in der Vocoder-Beschreibung des APCO Project 25 beschrieben ist. Unterschiede umfassen, dass der Sprachentscheidungsgenerator **325** eine separate Sprachentscheidung vielmehr für jede Oberwelle im Halbraten-Vocoder als für jede Gruppe von drei oder mehr Oberwellen berechnet, und dass der Spektralamplitudengenerator **330** jede Spektralamplitude unabhängig von den Sprachentscheidungen berechnet, wie beispielsweise im US-Patent 5 754 974 beschrieben. Außerdem kann die wahlweise Tonerfassungseinheit **335** im Halbraten-Vocoder enthalten sein, um Tonsignale zur Übertragung durch den Vocoder unter Verwendung von speziellen Tondatenblöcken von Bits, die durch den Decodierer erkannt werden, zu erfassen.

[0059] Fig. 4 stellt ein MBE-Parameterquantisierungsverfahren **400** dar, das eine Implementierung der von der MBE-Parameterquantisierungseinheit **210** von Fig. 2 durchgeführten Quantisierung bildet. Zusätzliche Details hinsichtlich der Quantisierung sind im US-Patent 6 199 037 B1 und in der Vocoder-Beschreibung des APCO Project 25 zu finden. Das beschriebene MBE-Parameterquantisierungsverfahren wird typischerweise nur auf Sprachsignale angewendet, während erfasste Tonsignale unter Verwendung eines separaten Tonquantisierers quantisiert werden. MBE-Parameter **405** sind die Eingangssignale in das MBE-Parameterquantisierungsverfahren. Die MBE-Parameter **405** können unter Verwendung der durch Fig. 3 dargestellten Verfahren abgeschätzt werden. In einer Implementierung werden 42–49 Bits pro Datenblock verwendet, um die MBE-Modellparameter zu quantisieren, wie in Tabelle 1 gezeigt, wobei die Anzahl von Bits für jeden Datenblock im Bereich von 42–49 unter Verwendung eines wahlweisen Steuerparameters unabhängig ausgewählt werden kann.

Parameter	Bits pro Datenblock
Grundfrequenz	7
Sprachentscheidungen	5
Verstärkung	5
Spektralamplituden	25-32
Gesamtbits	42-49

Tabelle 1: MBE-Parameterbits

[0060] In dieser Implementierung wird die Grundfrequenz f_0 typischerweise zuerst unter Verwendung einer Grundfrequenzquantisierereinheit **410** quantisiert, die 7 Grundfrequenzbits b_{fund} ausgibt, die gemäß Gleichung [1] folgendermaßen berechnet werden können:

$b_{fund} = 0$ wenn $f_0 > 0{,}0503$
$b_{fund} = 119$ wenn $f_0 < 0{,}00811$
$b_{fund} = \lfloor -45{,}368 \cdot \log_2(f_0) \rfloor$ ansonsten [1]

[0061] Die Oberwellen-Sprachmaße D_l und die Spektralamplituden M_l für $1 \leq l \leq L$ werden als nächstes von Oberwellen auf Sprachbänder unter Verwendung einer Frequenzabbildungseinheit **415** abgebildet. In einer Implementierung werden 8 Sprachbänder verwendet, wobei das erste Sprachband Frequenzen [0, 500 Hz] ab-

deckt, das zweite Sprachband [500, 1000 Hz] abdeckt, ... und das letzte Sprachband die Frequenzen [3500, 4000 Hz] abdeckt. Das Ausgangssignal der Frequenzabbildungseinheit **415** ist die Sprachbandenergiemetrik $vener_k$ und die Sprachbandfehlermetrik lv_k für jedes Sprachband k im Bereich von $0 \leq k < 8$. Die Energiemetrik jedes Sprachbandes, $vener_k$, wird durch Summieren von $|M_l|^2$ über alle Oberwellen im k-ten Sprachband, d.h. für $b_k < l \leq b_{k+1}$, berechnet, wobei b_k gegeben ist durch:

$$b_k = (k - 0{,}25)/(16f_0)\rfloor \quad [2]$$

[0062] Die Sprachbandmetrik $verr_k$ wird durch Summieren von $D_l \cdot |M_l|^2$ über $b_k < l \leq b_{k+1}$ berechnet und die Sprachbandfehlermetrik lv_k wird dann aus $verr_k$ und $vener_k$ berechnet, wie in Gleichung [3] nachstehend gezeigt:

$$lv_k = \max[0{,}0,\min[1{,}0,\ 0{,}5\cdot(1{,}0 - \log_2(verr_k/(T_k \cdot vener_k)))]] \quad [3]$$

wobei max[x,y] das Maximum von x oder y zurückgibt und min[x,y] das Minimum von x oder y berechnet. Der Schwellenwert T_k wird gemäß $T_k = \Theta(k, 0{,}1309)$ aus der Schwellenfunktion $\Theta(k, \omega_0)$ berechnet, die in Gleichung [37] der Vocoder-Beschreibung des APCO Project 25 definiert ist.

[0063] Sobald die Sprachbandenergiemetriken $vener_k$ und die Sprachbandfehlermetriken lv_k für jedes Sprachband berechnet wurden, werden die Sprachentscheidungen für den Datenblock gemeinsam unter Verwendung einer gewichteten 5-Bit-Sprachband-Vektorquantisierereinheit **420** quantisiert, die in einer Implementierung den Sprachband-Untervektorquantisierer verwendet, der im US-Patent 6 199 037 B1 beschrieben ist. Die gewichtete Sprachband-Vektorquantisierereinheit **420** gibt die Sprachentscheidungsbits b_{vuv} aus, wobei b_{vuv} den Index des ausgewählten Kandidatenvektors $x_j(i)$ aus einem Sprachbandcodebuch bedeutet. Ein Sprachbandcodebuch mit 5 Bits (32 Elementen), das in einer Implementierung verwendet wird, ist in Tabelle 2 gezeigt.

Index: i	Kandidatenvektor: $x_j(i)$	Index: i	Kandidatenvektor: $x_j(i)$
0	0xFF	1	0xFF
2	0xFE	3	0xFE
4	0xFC	5	0xDF
6	0xEF	7	0xFB
8	0xF0	9	0xF8
10	0xE0	11	0xE1
12	0xC0	13	0xC0
14	0x80	15	0x80
16	0x00	17	0x00
18	0x00	19	0x00
20	0x00	21	0x00
22	0x00	23	0x00
24	0x00	25	0x00
26	0x00	27	0x00
28	0x00	29	0x00
30	0x00	31	0x00

Tabelle 2: 5-Bit-Sprachbandcodebuch

[0064] Man beachte, dass jeder Kandidatenvektor $x_j(i)$, der in Tabelle 2 gezeigt ist, als 8-Bit-Hexadezimalzahl dargestellt ist, wobei jedes Bit ein einzelnes Element eines 8-Element-Codebuchvektors darstellt und $x_j(i) = 1{,}0$, wenn das Bit, das 2^{7-j} entspricht, eine 1 ist, und $x_j(i) = 0{,}0$, wenn das Bit, das 2^{7-j} entspricht, eine 0 ist. Diese Schreibweise wird verwendet, um mit dem Sprachband-Untervektorquantisierer konsistent zu sein, der im US-Patent 6 199 037 B1 beschrieben ist.

[0065] Ein Merkmal des Halbraten-Vocoders besteht darin, dass er mehrere Kandidatenvektoren umfasst, die jeweils demselben Sprachzustand entsprechen. Die Indizes 16–31 in Tabelle 2 entsprechen beispielsweise alle dem ganz stimmlosen Zustand und die Indizes 0 und 1 entsprechen beide dem ganz stimmhaften Zustand. Dieses Merkmal stellt einen interoperierbaren Aufrüstungspfad für den Vocoder bereit, der alternative Implementierungen ermöglicht, die gepulste oder andere verbesserte Sprachzustände umfassen könnten. Anfänglich kann ein Codierer nur den niedrigstwertigen Index verwenden, sobald zwei oder mehr Indizes demselben Sprachzustand gleich sind. Ein aufgerüsteter Codierer kann jedoch die höherwertigen Indizes verwenden, um alternative zugehörige Sprachzustände darzustellen. Der anfängliche Decodierer würde entweder die niedrigsten oder höheren Indizes in denselben Sprachzustand decodieren (beispielsweise würden die Indizes 16–31 alle als ganz stimmlos decodiert werden), aber die aufgerüsteten Decodierer können diese Indizes in zugehörige, aber andere Sprachzustände für eine verbesserte Leistung decodieren.

[0066] Fig. 4 stellt auch die Verarbeitung der Spektralamplituden durch eine logarithmische Recheneinheit **425** dar, die die logarithmischen Spektralamplituden $\log_2(M_l)$ für $1 \le l \le L$ berechnet. Die ausgegebenen logarithmischen Spektralamplituden werden dann durch eine logarithmische Spektralamplituden-Quantisiereinheit **430** quantisiert, um ausgegebene logarithmische Spektralamplituden-Ausgangsbits zu erzeugen.

[0067] Fig. 5 zeigt ein Quantisierungsverfahren **500** für logarithmische Spektralamplituden, das eine Implementierung der durch die Quantisierungseinheit **430** von Fig. 4 durchgeführten Quantisierung bildet. Der

schattierte Abschnitt von Fig. 5, einschließlich der Elemente **525**–**550**, zeigt eine entsprechende Implementierung eines Rekonstruktionsverfahrens **555** für logarithmische Spektralamplituden, das innerhalb der Parameterrekonstruktionseinheit **240** von Fig. 2 implementiert werden kann, um die logarithmischen Spektralamplituden aus den von der FEC-Decodiereinheit **235** ausgegebenen Quantisiererbits zu rekonstruieren.

[0068] Mit Bezug auf Fig. 5 werden logarithmische Spektralamplituden für einen Datenblock (d.h. $\log_2(M_l)$ für $1 \le l \le L$) durch eine Mittelwertberechnungseinheit **505** verarbeitet, um den Mittelwert zu berechnen und von den logarithmischen Spektralamplituden zu entfernen. Der Mittelwert wird an eine Verstärkungsquantisierereinheit **515** ausgegeben, die die Verstärkung G(0) für den aktuellen Datenblock aus dem Mittelwert berechnet, wie in Gleichung [4] gezeigt:

$$G(0) = \text{Mittelwert}\{\log_2(M_l)\} + 0{,}5\log_2(L) \quad [4]$$

[0069] Die Differenzverstärkung Δ_G wird dann berechnet als:

$$\Delta_G = G(0) - 0{,}5 \cdot G(-1) \quad [5]$$

wobei G(–1) der Verstärkungsterm vom vorherigen Datenblock nach der Quantisierung und Rekonstruktion ist. Die Differenzverstärkung Δ_G wird dann unter Verwendung eines ungleichmäßigen 5-Bit-Quantisierers quantisiert, wie z.B. dem in Tabelle 3 gezeigten. Die aus dem Quantisierer ausgegebenen Verstärkungsbits werden als $b_{\text{Verstärkung}}$ bezeichnet.

Index: i	Differenzverstärkung: $\Delta_G(i)$	Index: i	Kandidatenvektor: $\Delta_G(i)$
0	-2,0	1	-0,67
2	0,2979	3	0,6637
4	1,0368	5	1,4381
6	1,8901	7	2,2280
8	2,4783	9	2,6676
10	2,7936	11	2,8933
12	3,0206	13	3,1386
14	3,2376	15	3,3226
16	3,4324	17	3,5719
18	3,6967	19	3,8149
20	3,9209	21	4,0225
22	4,1236	23	4,2283
24	4,3706	25	4,5437
26	4,7077	27	4,8489
28	5,0568	29	5,3265
30	5,7776	31	6,8745

Tabelle 3: 5-Bit-Differenzverstärkungscodebuch

[0070] Die Mittelwertberechnungseinheit **505** gibt logarithmische Spektralamplituden mit einem Mittelwert von Null an eine Subtraktionseinheit **510** aus, die vorhergesagte Amplituden subtrahiert, um einen Satz von Amp-

litudenvorhersageabweichungen zu erzeugen. Die Amplitudenvorhersageabweichungen werden in eine Quantisierungseinheit **520** eingegeben, die Amplitudenvorhersageabweichungs-Pararneterbits erzeugt.

[0071] Diese Amplitudenvorhersageabweichungs-Parameterbits werden auch in die Rekonstruktionseinheit **555**, die im schattierten Bereich von Fig. 5 dargestellt ist, eingespeist. Insbesondere berechnet die inverse Amplitudenvorhersageabweichungs-Quantisierungseinheit **525** rekonstruierte Amplitudenvorhersageabweichungen unter Verwendung der Eingangsbits und liefert die rekonstruierten Amplitudenvorhersageabweichungen zu einer Summiereinheit **530**, die sie zu den vorhergesagten Amplituden addiert, um rekonstruierte logarithmische Spektralamplituden mit einem Mittelwert von Null zu bilden, die im Datenblock-Speicherelement **535** gespeichert werden.

[0072] Die gespeicherten logarithmischen Spektralamplituden mit dem Mittelwert von Null von einem vorherigen Datenblock werden in Verbindung mit den rekonstruierten Grundfrequenzen für den aktuellen und früheren Datenblock durch die Berechnungseinheit **540** für vorhergesagte Amplituden verarbeitet und dann durch eine Skalierungseinheit **545** skaliert, um vorhergesagte Amplituden zu bilden, die an eine Differenzeinheit **510** und eine Summiereinheit **530** angelegt werden. Die Berechnungseinheit **540** für vorhergesagte Amplituden interpoliert typischerweise die rekonstruierten logarithmischen Spektralamplituden von einem vorherigen Datenblock auf der Basis des Verhältnisses der rekonstruierten Grundfrequenz vom aktuellen Datenblock zur rekonstruierten Grundfrequenz des vorherigen Datenblocks. Dieser Interpolation folgt eine Anwendung eines Skalierungsfaktors ρ, der normalerweise geringer ist als 1,0 (ρ = 0,65 ist typisch und in einigen Implementierungen kann ρ in Abhängigkeit von der Anzahl von Spektralamplituden im Datenblock variiert werden), durch die Skalierungseinheit **545**.

[0073] Außerdem wird der Mittelwert dann aus den Verstärkungsbits und aus dem gespeicherten Wert von G(–1) in einer Mittelwert-Rekonstruktionseinheit **550** rekonstruiert, die auch den rekonstruierten Mittelwert zu den rekonstruierten Amplitudenvorhersageabweichungen addiert, um rekonstruierte logarithmische Spektralamplituden **560** zu erzeugen.

[0074] In der in Fig. 5 gezeigten Implementierung nehmen die Quantisierungseinheit **520** und die inverse Quantisierungseinheit **525** einen wahlweisen Steuerparameter an, der ermöglicht, dass die Anzahl von Bits pro Datenblock innerhalb eines gewissen zulässigen Bereichs von Bits (beispielsweise 25–32 Bits pro Datenblock) ausgewählt wird. Typischerweise werden die Bits pro Datenblock unter Verwendung nur einer Teilmenge der zulässigen Quantisierungsvektoren in der Quantisierungseinheit **510** und der inversen Quantisierungseinheit **515** verändert, wie nachstehend weiter beschrieben. Dieser gleiche Steuerparameter kann in verschiedenen Weisen verwendet werden, um die Anzahl von Bits pro Datenblock über einen breiteren Bereich zu ändern, falls erforderlich. Dies kann beispielsweise auch durch Verringern der Anzahl von Bits vom Verstärkungsquantisierer durch Suchen nur der geraden Indizes 0, 2, 4, 6, ... 32 in Tabelle 3 durchgeführt werden. Dieses Verfahren kann auch auf die Grundfrequenz oder den Sprachquantisierer angewendet werden. Fig. 6 zeigt ein Amplitudenvorhersageabweichungs-Quantisierungsverfahren **600**, das eine Implementierung der von der Quantisierungseinheit **520** von Fig. 5 durchgeführten Quantisierung bildet. Zuerst unterteilt ein Blockteiler **605** die Amplitudenvorhersageabweichungen in vier Blöcke, wobei die Länge von jedem Block typischerweise durch die Anzahl von Oberwellen, L, bestimmt ist, wie in Tabelle 4 gezeigt. Blöcke mit niedrigerer Frequenz sind im Allgemeinen gleich oder kleiner in der Größe im Vergleich zu Blöcken mit höherer Frequenz, um die Leistung zu verbessern, indem die wahrnehmbar wichtigeren Niederfrequenzbereiche mehr betont werden. Jeder Block wird dann mit einer separaten diskreten Cosinustransformations- (DCT) Einheit **610** transformiert und die DCT-Koeffizienten werden in einen PRBA-Vektor mit acht Elementen (unter Verwendung der ersten zwei DCT-Koeffizienten jedes Blocks) und vier HOC-Vektoren (einen für jeden Block, der alle, bis auf die ersten zwei DCT-Koeffizienten bildet) durch eine PRBA- und HOC-Vektorbildungseinheit **615** unterteilt. Die Bildung des PRBA-Vektors verwendet die ersten zwei DCT-Koeffizienten für jeden Block, die folgendermaßen transformiert und beschaffen werden:

$$
\begin{aligned}
PRBA(0) &= Block_0(0) + 1.414 \cdot Block_0(0) \\
PRBA(1) &= Block_0(0) - 1.414 \cdot Block_0(0) \\
PRHA(2) &= Block_1(0) + 1.414 \cdot Block_1(0) \\
PRBA(3) &= Block_1(0) - 1.414 \cdot Block_1(0) \\
PRBA(4) &= Block_2(0) + 1.414 \cdot Block_2(0) \\
PRBA(5) &= Block_2(0) - 1.414 \cdot Block_2(0) \\
PRBA(6) &= Block_3(0) + 1.414 \cdot Block_3(0) \\
PRBA(7) &= Block_3(0) - 1.414 \cdot Block_3(0)
\end{aligned}
\qquad [6]
$$

wobei PRBA(n) das n-te Element des PRBA-Vektors ist und $Block_j(k)$ das k-te Element des j-ten Blocks ist.

L	$Block_0$	$Block_1$	$Block_2$	$Block_3$
9	2	2	2	3
10	2	2	3	3
11	2	3	3	3
12	2	3	3	4
13	3	3	3	4
14	3	3	4	4
15	3	3	4	5
16	3	4	4	5
17	3	4	5	5
18	4	4	5	5
19	4	4	5	6
20	4	4	6	6
21	4	5	6	6
22	4	5	6	7
23	5	5	6	7
24	5	5	7	7
25	5	6	7	7
26	5	6	7	8
27	5	6	8	8
28	6	6	8	8
29	6	6	8	9
30	6	7	8	9
31	6	7	9	9
32	6	7	9	10
33	7	7	9	10
34	7	8	9	10
35	7	8	10	10
36	7	8	10	11
37	8	8	10	11
38	8	9	10	11
39	8	9	11	11
40	8	9	11	12
41	8	9	11	13
42	8	9	12	13
43	8	10	12	13
44	9	10	12	13
45	9	10	12	14
46	9	10	13	14
47	9	11	13	14
48	10	11	13	14
49	10	11	13	15
50	10	11	14	15
51	10	12	14	15
52	10	12	14	16
53	11	12	14	16
54	11	12	15	16
55	11	12	15	17
56	11	13	15	17

Tabelle 4: Amplitudenvorhersageabweichungs-Blockgröße

[0075] Der PRBA-Vektor wird unter Verwendung einer Acht-Punkt-DCT, gefolgt von einer Teilvektor-Quantisiereinheit **620** weiter verarbeitet, um PRBA-Bits zu erzeugen. In einer Implementierung wird der erste PRBA-DCT-Koeffizient (als R_0 bezeichnet) ignoriert, da er mit dem separat quantisierten Verstärkungswert redundant ist. Alternativ kann dieser erste PRBA-DCT-Koeffizient anstelle der Verstärkung quantisiert werden, wie in der Vocoder-Beschreibung des APCO Project 25 beschrieben. Die letzten sieben PRBA-DCT-Koeffizienten [R_1-R_7] werden dann mit einem Teilvektor-Quantisierer quantisiert, der ein Codebuch mit neun Bits verwendet, um die drei Elemente [R_1-R_3] zu quantisieren, um PRBA-Quantisiererbits b_{PRBA13} zu erzeugen, und ein Codebuch mit sieben Bits wird verwendet, um die vier Elemente [R_4-R_7] zu quantisieren, um PRBA-Quantisiererbits b_{PRBA47} zu erzeugen. Diese 16 PRBA-Quantisiererbits (b_{PRBA13} und b_{PRBA47}) werden dann aus dem Quantisierer ausgegeben. Typische Teil-VQ-Codebücher, die verwendet werden, um den PRBA-Vektor zu quantisieren, sind im Anhang A gegeben.

[0076] Die vier HOC-Vektoren, die als HOC0, HOC1, HOC2 und HOC3 bezeichnet sind, werden dann unter

Verwendung von vier separaten Codebüchern **625** quantisiert. In einer Implementierung wird ein Codebuch mit fünf Bits für HOC0 verwendet, um HOCO-Quantisiererbits b_{HOC0} zu erzeugen; Codebücher mit vier Bits werden für HOC1 und HOC2 verwendet, um HOC1-Quantisiererbits b_{HOC1} und HOC2-Quantisiererbits b_{HOC2} zu erzeugen; und ein 3-Bit-Codebuch wird für HOC3 verwendet, um HOC3-Quantisiererbits b_{HOC3} zu erzeugen. Typische Codebücher, die verwendet werden, um die HOC-Vektoren in dieser Implementierung zu quantisieren, sind im Anhang B gezeigt. Man beachte, dass jeder HOC-Vektor in der Länge zwischen 0 und 15 Elementen variieren kann. Die Codebücher sind jedoch für ein Maximum von vier Elementen pro Vektor ausgelegt. Wenn ein HOC-Vektor weniger als vier Elemente aufweist, dann werden nur die ersten Elemente jedes Codebuchvektors vom Quantisierer verwendet. Wenn der HOC-Vektor alternativ mehr als vier Elemente aufweist, dann werden nur die ersten vier Elemente verwendet und alle anderen Elemente in diesem HOC-Vektor werden gleich Null gesetzt. Sobald alle HOC-Vektoren quantisiert sind, werden die 16 HOC-Quantisiererbits (b_{HOC0}, b_{HOC1}, b_{HOC2} und b_{HOC3}) vom Quantisierer ausgegeben.

[0077] In der in Fig. 6 gezeigten Implementierung nehmen die Vektorquantisiereinheiten **620** und/oder **625** einen wahlweisen Steuerparameter an, der ermöglicht, dass die Anzahl von Bits pro Datenblock, die verwendet werden, um die PRBA- und HOC-Vektoren zu quantisieren, innerhalb eines gewissen zulässigen Bereichs von Bits ausgewählt wird. Typischerweise werden die Bits pro Datenblock vom nominalen Wert von 32 unter Verwendung nur einer Teilmenge der zulässigen Quantisierungsvektoren in einem oder mehreren der vom Quantisierer verwendeten Codebücher reduziert. Wenn beispielsweise nur die geraden Kandidatenvektoren in einem Codebuch verwendet werden, dann ist das letzte Bit im Codebuchindex als Null bekannt, was ermöglicht, dass die Anzahl von Bits um Eins verringert wird. Dies kann auf jeden vierten Vektor erweitert werden, um zu ermöglichen, dass die Anzahl von Bits um zwei verringert wird.

[0078] Am Decodierer wird der Codebuchindex durch Anhängen der geeigneten Anzahl von "0"-Bits anstelle von irgendwelchen fehlenden Bits rekonstruiert, um zu ermöglichen, dass der quantisierte Codebuchvektor bestimmt wird. Diese Methode wird auf eines oder mehrere der HOC- und/oder PRBA-Codebücher angewendet, um die ausgewählte Anzahl von Bits für den Datenblock zu erhalten, wie in Tabelle 5 gezeigt, wobei die Anzahl von Amplitudenvorhersageabweichungs-Quantisiererbits typischerweise als Versatz von der Anzahl von Sprachbits im Datenblock bestimmt wird (d.h. die Anzahl von Sprachbits minus 17).

Amplitudenvorhersage-Abweichungs-Quantisierer-Bits pro Datenblock	PRBA $[R_1 - R_3]$	PRBA $[R_4 - R_7]$	HOC0	HOC1	HOC2	HOC3
32	9	7	5	4	4	3
31	9	7	5	4	4	2
30	9	7	5	4	4	1
29	9	7	5	4	3	1
28	9	7	5	3	3	1
27	9	7	4	3	3	1
26	9	6	4	3	3	1
25	8	6	4	3	3	1

Tabelle 5: Amplitudenvorhersageabweichungs-Quantisiererbits pro Datenblock

[0079] Mit Bezug auf Fig. 4 empfängt die Kombinationseinheit **435** Grundfrequenz- oder Tonhöhenbits b_{fund}, Sprachbits b_{vuv}, Verstärkungsbits b_{gain} und Spektralbits b_{PRBA13} b_{PRBA47} b_{HOC0}, b_{HOC1}, b_{HOC2} und b_{HOC} von den Quantisiereinheiten **410**, **420** und **430**. Typischerweise priorisiert die Kombinationseinheit **435** diese Eingangsbits, um Ausgangssprachbits zu erzeugen, so dass die ersten Sprachbits im Datenblock gegen Bitfehler empfindlicher sind, während die späteren Sprachbits im Datenblock gegen Bitfehler weniger empfindlich sind. Diese Priorisierung ermöglicht, dass ein FEC effizient auf die empfindlichsten Sprachbits angewendet wird, was zu einer verbesserten Sprachqualität und Unempfindlichkeit in den verschlechterten Kommunikationskanälen führt. In einer solchen Implementierung bestehen die ersten 12 Sprachbits in einem durch die Kombinationseinheit **435** ausgegebenen Datenblock aus den vier höchstwertigen Grundfrequenzbits, gefolgt von den ersten vier Sprachentscheidungsbits und den vier höchstwertigen Verstärkungsbits. Das resultierende Sprachdatenblockformat (d.h. die Reihenfolge der Ausgangssprachbits nach der Priorisierung durch die Kombinationseinheit **435**) ist in Tabelle 6 gezeigt.

Bitposition im Sprachdatenblock	Sprachbits
0-3	4 höchstwertige Bits von b_{fund}
4-7	4 höchstwertige Bits von b_{vuv}
8-11	4 höchstwertige Bits von b_{gain}
12-19	8 höchstwertige Bits von b_{PRBA13}
20-23	4 höchstwertige Bits von b_{PRBA47}
24-27	4 höchstwertige Bits von b_{HOC0}
28-30	3 höchstwertige Bits von b_{HOC1}
31-33	3 höchstwertige Bits von b_{HOC2}
34	1 höchstwertiges Bit von b_{HOC3}
35	1 niedrigstwertiges Bit von b_{vuv}
36	1 niedrigstwertiges Bit von b_{gain}
37-39	3 niedrigstwertige Bits von b_{fund}
40	1 niedrigstwertiges Bit von b_{PRBA13}
41-43	3 niedrigstwertige Bits von b_{PRBA47}
44	1 niedrigstwertiges Bit von b_{HOC0}
45	1 niedrigstwertiges Bit von b_{HOC1}
46	1 niedrigstwertiges Bit von b_{HOC2}
47-48	2 niedrigstwertige Bits von b_{HOC3}

Tabelle 6: Sprachdatenblockformat

[0080] Mit erneutem Bezug auf Fig. 2 kann der Codierer eine Tonquantisierungseinheit **215** umfassen, die einen Datenblock von Tonbits (d.h. einen Tondatenblock) ausgibt, wenn bestimmte Tonsignale (wie z.B. ein Einfrequenzton, Knox-Töne, ein MFV-Ton und/oder ein Hörton) im Codiereingangssignal erfasst werden. In einer Implementierung werden Tonbits erzeugt, wie in Tabelle 7 gezeigt, wenn die ersten 6 Bits lauter Einsen sind (Hexadezimalwert 0x3F), um zu ermöglichen, dass der Decodierer einen Tondatenblock von anderen Datenblöcken, die Sprachbits enthalten (d.h. Sprachdatenblöcke) eindeutig identifiziert. Diese eindeutige Unterscheidung ist aufgrund der Grenzen für den Wert von b_{fund}, die durch die Gleichung [1] auferlegt werden, möglich, die verhindern, dass der Tondatenblock-Identifikatorwert (0x3F) jemals für Sprachdatenblöcke auftritt, und da der Tondatenblockidentifikator dieselbe Position im Datenblock wie die vier höchstwertigen Tonhöhenbits b_{fund} überlappt, wie in Tabelle 6 gezeigt. Die sieben Tonamplitudenbits b_{TONAMP} werden aus der abgeschätzten Tonamplitude A_{TON} folgendermaßen berechnet:

$$B_{TONAMP} = \max[0, \min[127, 8{,}467 \cdot (\log_2(A_{TON}) + 1)]] \quad [4]$$

während der 8-Bit-Tonindex b_{TON}, der verwendet wird, um ein gegebenes Tonsignal darzustellen, im Anhang C gezeigt ist. Typischerweise wird der Tonindex b_{TON} mehrere Male innerhalb eines Tondatenblocks wiederholt, um die Unempfindlichkeit gegen Kanalfehler zu steigern. Dies ist in Tabelle 7 dargestellt, in der der Tonindex viermal innerhalb des Datenblocks von 49 Bits wiederholt wird.

Bitposition im Datenblock	Tonbits
0-5	0x3F
6-11	erste 6 höchstwertige Bits von b_{TONAMP}
12-19	b_{TON}
20-27	b_{TON}
28-35	b_{TON}
36-43	b_{TON}
44	7. niedrigstwertiges Bit von b_{TONAMP}
45-48	0

Tabelle 7: Tondatenblockformat

[0081] Obwohl die Verfahren weitgehend im Zusammenhang mit einem neuen Halbraten-MBE-Vocoder beschrieben sind, können die beschriebenen Verfahren leicht auf andere Systeme und/oder Vocoder angewendet werden. Andere Vocoder vom MBE-Typ können beispielsweise auch von den Verfahren profitieren, ungeachtet der Bitrate oder Datenblockgröße. Außerdem können die beschriebenen Verfahren auf viele andere Sprachcodiersysteme anwendbar sein, die ein anderes Sprachmodell mit alternativen Parametern verwenden (wie z.B. STC, MELP, MB-HTC, CELP, HVXC oder andere) oder die andere Verfahren zur Analyse, Quantisierung und/oder Synthese verwenden.

Anhang A: PRBA-Codebücher

Codebuchindex	PRBA13(0)	PRBA13(1)	PRBA13(2)
0	0.526055	-0.328567	-0.304727
1	0.441044	-0.303127	-0.201114
2	1.030896	-0.324730	-0.397204
3	0.839696	-0.351933	-0.224909
4	0.272958	-0.176118	-0.098893
5	0.221466	-0.160045	-0.061026
6	0.496555	-0.211499	0.047305
7	0.424376	-0.223752	0.069911
8	0.264531	-0.353355	-0.330505
9	0.273650	-0.253004	-0.250241
10	0.484531	-0.297627	-0.071051
11	0.410814	-0.224961	-0.084998
12	0.039519	-0.252904	-0.115128
13	0.017423	-0.296519	-0.045921
14	0.225113	-0.224371	0.037882
15	0.183424	-0.260492	0.050491
16	0.308704	-0.073205	-0.405880
17	0.213125	-0.101632	-0.333208
18	0.617735	-0.137299	-0.213670
19	0.514382	-0.126485	-0.170204
20	0.130009	-0.076955	-0.229303
21	0.061740	-0.108259	-0.203887
22	0.244473	-0.110094	-0.051689
23	0.230452	-0.076147	-0.028190
24	0.059837	-0.254595	-0.562704
25	0.011630	-0.135223	-0.432791
26	0.207077	-0.152248	-0.148391
27	0.158078	-0.128800	-0.122150
28	-0.265982	-0.144742	-0.199894
29	-0.356479	-0.204740	-0.156465
30	0.000324	-0.139549	-0.066471
31	0.001888	-0.170557	-0.025025
32	0.402913	-0.581478	-0.274626
33	0.191289	-0.540335	-0.193040
34	0.632914	-0.401410	-0.006636
35	0.471086	-0.463144	0.061489
36	0.044829	-0.438487	0.033433
37	0.015513	-0.539475	-0.006719
38	0.336218	-0.351311	0.214087
39	0.239967	-0.380836	0.157681
40	0.347609	-0.901619	-0.688432
41	0.064067	-0.826753	-0.492089
42	0.303089	-0.396757	-0.108446
43	0.235590	-0.446122	0.006437
44	-0.236964	-0.652532	-0.135520
45	-0.418285	-0.793014	-0.034730
46	-0.038262	-0.516984	0.273681
47	-0.037419	-0.958198	0.214749
48	0.061624	-0.238233	-0.237184
49	-0.013944	-0.235704	-0.204811

50	0.286428	-0.210542	-0.029587
51	0.257656	-0.261837	-0.056566
52	-0.235852	-0.310760	-0.165147
53	-0.334949	-0.385870	-0.197362
54	0.094870	-0.241144	0.059122
55	0.060177	-0.225884	0.031140
56	-0.301184	-0.306545	-0.446189
57	-0.293528	-0.504146	-0.429844
58	-0.055084	-0.379015	-0.125887
59	-0.115434	-0.375008	-0.059939
60	-0.777425	-0.592163	-0.107585
61	-0.950500	-0.893847	-0.181762
62	-0.259402	-0.396726	0.010357
63	-0.368905	-0.449026	0.038299
64	0.279719	-0.063196	-0.184628
65	0.255265	-0.067248	-0.121124
66	0.458433	-0.103777	0.010074
67	0.437231	-0.092496	-0.031028
68	0.082265	-0.028050	-0.041262
69	0.045920	-0.051719	-0.030155
70	0.271149	-0.043613	0.112085
71	0.246881	-0.065274	0.105436
72	0.056590	-0.117773	-0.142283
73	0.058824	-0.104418	-0.099608
74	0.213781	-0.111974	0.031269
75	0.187554	-0.070340	0.011834
76	-0.185701	-0.081106	-0.073803
77	-0.266112	-0.074133	-0.085370
78	-0.029368	-0.046490	0.124679
79	-0.017378	-0.102882	0.140482
80	0.114700	0.092738	-0.244271
81	0.072922	0.007863	-0.231476
82	0.270022	0.031819	-0.094208
83	0.254403	0.024805	-0.050389
84	-0.182905	0.021629	-0.168481
85	-0.225864	-0.010109	-0.130374
86	0.040089	0.013969	0.016028
87	0.001442	0.010551	0.032942
88	-0.287472	-0.036130	-0.296798
89	-0.332344	-0.108862	-0.342196
90	0.012700	0.022917	-0.052501
91	-0.040681	-0.001805	-0.050548
92	-0.718522	-0.061234	-0.278820
93	-0.879205	-0.213588	-0.303508
94	-0.234102	-0.065407	0.013686
95	-0.281223	-0.076139	0.046830
96	0.141967	-0.193679	-0.055697
97	0.100318	-0.161222	-0.063062
98	0.265859	-0.132747	0.078209
99	0.244805	-0.139776	0.122123
100	-0.121802	-0.179976	0.031732
101	-0.185318	-0.214011	0.018117
102	0.047014	-0.153961	0.218068
103	0.047305	-0.187402	0.282114

104	-0.027533	-0.415868	-0.333841
105	-0.125886	-0.334492	-0.290317
106	-0.030602	-0.190918	0.097454
107	-0.054936	-0.209948	0.158977
108	-0.507223	-0.295876	-0.217183
109	-0.581733	-0.403194	-0.208936
110	-0.299719	-0.289679	0.297101
111	-0.363169	-0.362718	0.436529
112	-0.124627	-0.042100	-0.157011
113	-0.161571	-0.092846	-0.183636
114	0.084520	-0.100217	-0.000901
115	0.055655	-0.136381	0.032764
116	-0.545087	-0.197713	-0.026888
117	-0.662772	-0.179815	0.026419
118	-0.165583	-0.148913	0.090382
119	-0.240772	-0.182830	0.105474
120	-0.576315	-0.359473	-0.456844
121	-0.713430	-0.554156	-0.476739
122	-0.275628	-0.223640	-0.051584
123	-0.359501	-0.230758	-0.027006
124	-1.282559	-0.284807	-0.233743
125	-1.060476	-0.399911	-0.562698
126	-0.871952	-0.272197	0.016126
127	-0.747922	-0.329404	0.276696
128	0.643086	0.046175	-0.660078
129	0.738204	-0.127844	-0.433708
130	1.158072	0.025571	-0.177856
131	0.974840	-0.009417	-0.112337
132	0.418014	0.032741	-0.124545
133	0.381422	-0.001557	-0.085504
134	0.768280	0.056085	0.095375
135	0.680004	0.052035	0.152318
136	0.473182	0.012560	-0.264221
137	0.345153	0.036627	-0.248756
138	0.746238	-0.025880	-0.106050
139	0.644319	-0.058256	-0.095133
140	0.185924	-0.022230	-0.070540
141	0.146068	-0.009550	-0.057871
142	0.338488	0.013022	0.069961
143	0.298969	0.047403	0.052598
144	0.346002	0.256253	-0.380261
145	0.313092	0.163821	-0.314004
146	0.719154	0.103108	-0.252648
147	0.621429	0.172423	-0.265180
148	0.240461	0.104684	-0.202582
149	0.206946	0.139642	-0.138016
150	0.359915	0.101273	-0.052997
151	0.318117	0.125888	-0.003486
152	0.150452	0.050219	-0.409155
153	0.188753	0.091894	-0.325733
154	0.334922	0.029098	-0.098587
155	0.324508	0.015809	-0.135408
156	-0.042506	0.038667	-0.208535
157	-0.083003	0.094758	-0.174054

158	0.094773	0.102653	-0.025701
159	0.063284	0.118703	-0.000071
160	0.355965	-0.139239	-0.191705
161	0.392742	-0.105496	-0.132103
162	0.663678	-0.204627	-0.031242
163	0.609381	-0.146914	0.079610
164	0.151855	-0.132843	-0.007125
165	0.146404	-0.161917	0.024842
166	0.400524	-0.135221	0.232289
167	0.324931	-0.116605	0.253458
168	0.169066	-0.215132	-0.185604
169	0.128681	-0.189394	-0.160279
170	0.356194	-0.116992	-0.038381
171	0.342866	-0.144687	0.020265
172	-0.065545	-0.202593	-0.043688
173	-0.124296	-0.260225	-0.035370
174	0.083224	-0.235149	0.153301
175	0.046256	-0.309608	0 190944
176	0.187385	-0.008168	-0.198575
177	0.190401	-0.018699	-0.136858
178	0.398009	-0.025700	-0.007458
179	0.346948	-0.022258	-0.020905
180	-0.047064	-0.085629	-0.080677
181	-0.067523	-0.128972	-0.119538
182	0.186086	-0.016828	0.070014
183	0.187364	0.017133	0.075949
184	-0.112669	-0.037433	-0.298944
185	-0.068276	-0.114504	-0.265795
186	0.147510	-0.040616	-0.013687
187	0.133084	-0.062849	-0.032637
188	-0.416571	-0.041544	-0.125088
189	-0.505337	-0.044193	-0.157651
190	-0.154132	-0.075106	0.050466
191	-0.148036	-0.059719	0.121516
192	0.490555	0.157659	-0.222208
193	0.436700	0.120500	-0.205869
194	0.754525	0.269323	0.045810
195	0.645077	0.271923	0.013942
196	0.237023	0.115337	-0.026429
197	0.204895	0.121020	-0.008541
198	0.383999	0.153963	0.171763
199	0.385026	0.222074	0.239731
200	0.198232	0.072972	-0.108179
201	0.147882	0.074743	-0.123341
202	0.390929	0.075205	0.081828
203	0.341623	0.089405	0.069389
204	-0.003381	0.159694	-0.016026
205	-0.043653	0.206860	-0.040729
206	0.135515	0.107824	0.179310
207	0.081086	0.119673	0.174282
208	0.192637	0.400335	-0.341906
209	0.171196	0.284921	-0.221516
210	0.377807	0.359087	-0.151523
211	0.411052	0.297925	-0.099774

212	-0.010060	0.261887	-0.149567
213	-0.107877	0.287756	-0.116982
214	0.158003	0.209727	0.077988
215	0.109710	0.232272	0.088135
216	0.000698	0.209353	-0.395208
217	-0.094015	0.230322	-0.279928
218	0.137355	0.230881	-0.124115
219	0.103058	0.166855	-0.100386
220	-0.305058	0.305422	-0.176026
221	-0.422049	0.337137	-0.293297
222	-0.121744	0.185124	0.048115
223	-0.171052	0.200312	0.052812
224	0.224091	-0.010673	-0.019727
225	0.200266	-0.020167	0.001798
226	0.382742	0.032362	0.161665
227	0.345631	-0.019705	0.164451
228	0.029431	0.045010	0.071518
229	0.031940	0.010876	0.087037
230	0.181935	0.039112	0.202316
231	0.181810	0.033189	0.253435
232	-0.008677	-0.066679	-0.144737
233	-0.021768	-0.021288	-0.125903
234	0.136766	0.000100	0.059449
235	0.135405	-0.020446	0.103793
236	-0.289115	0.039747	-0.012256
237	-0.338683	0.025909	-0.034058
238	-0.016515	0.048584	0.197981
239	-0.046790	0.011816	0.199964
240	0.094214	0.127422	-0.169936
241	0.048279	0.096189	-0.148153
242	0.217391	0.081732	0.013677
243	0.179656	0.084671	0.031434
244	-0.227367	0.118176	-0.039803
245	-0.327096	0.159747	-0.018931
246	0.000834	0.113118	0.125325
247	-0.014617	0.128924	0.163776
248	-0.254570	0.154329	-0.232018
249	-0.353068	0.124341	-0.174409
250	-0.061004	0.107744	0.037257
251	-0.100991	0.080302	0.062701
252	-0.927022	0.285660	-0.240549
253	-1.153224	0.277232	-0.322538
254	-0.569012	0.108135	0.172634
255	-0.555273	0.131461	0.325930
256	0.518847	0.065683	-0.132877
257	0.501324	-0.006585	-0.094884
258	1.066190	-0.150380	0.201791
259	0.858377	-0.166415	0.081686
260	0.320584	-0.031499	0.039534
261	0.311442	-0.075120	0.026013
262	0.625829	-0.019856	0.346041
263	0.525271	-0.003948	0.284868
264	0.312594	-0.075673	-0.066642
265	0.295732	-0.057895	-0.042207

266	0.550446	-0.029110	0.046850
267	0.465467	-0.068987	0.096167
268	0.122669	-0.051786	0.044283
269	0.079669	-0.044145	0.045805
270	0.238778	-0.031835	0.171694
271	0.200734	-0.072619	0.178726
272	0.342512	0.131270	-0.163021
273	0.294028	0.111759	-0.125793
274	0.589523	0.121808	-0.049372
275	0.550506	0.132318	0.017485
276	0.164280	0.047560	-0.058383
277	0.120110	0.049242	-0.052403
278	0.269181	0.035000	0.103494
279	0.297466	0.038517	0.139289
280	0.094549	-0.030880	-0.153376
281	0.080363	0.024359	-0.127578
282	0.281351	0.055178	0.000155
283	0.234900	0.039477	0.013957
284	-0.118161	0.011976	-0.034270
285	-0.157654	0.027765	-0.005010
286	0.102631	0.027283	0.099723
287	0.077285	0.052532	0.115583
288	0.329398	-0.278552	0.016316
289	0.305993	-0.267896	0.094952
290	0.775270	-0.394995	0.290748
291	0.583180	-0.252159	0.285391
292	0.192226	-0.182242	0.126859
293	0.185908	-0.245779	0.159940
294	0.346293	-0.250404	0.355682
295	0.354160	-0.364521	0.472337
296	0.134942	-0.313666	-0.115181
297	0.126077	-0.286568	-0.039927
298	0.405618	-0.211792	0.199095
299	0.312099	-0.213642	0.190972
300	-0.071392	-0.297366	0.081426
301	-0.165839	-0.301986	0.160640
302	0.147808	-0.290712	0.298198
303	0.063302	-0.310149	0.396302
304	0.141444	-0.081377	-0.076621
305	0.115936	-0.104440	-0.039885
306	0.367023	-0.087281	0.096390
307	0.330038	-0.117958	0.127050
308	0.002897	-0.062454	0.025151
309	-0.052404	-0.082200	0.041975
310	0.181553	-0.137004	0.230489
311	0.140768	-0.094604	0.265928
312	-0.101763	-0.209566	-0.135964
313	-0.159056	-0.191005	-0.095509
314	0.045016	-0.081562	0.075942
315	0.016808	-0.112482	0.068593
316	-0.408578	-0.132377	0.079163
317	-0.431534	-0.214646	0.157714
318	-0.096931	-0.101938	0.200304
319	-0.167867	-0.114851	0.262964

320	0.393882	0.086002	0.008961
321	0.338747	0.048405	-0.004187
322	0.877844	0.374373	0.171008
323	0.740790	0.324525	0.242248
324	0.200218	0.070150	0.085891
325	0.171760	0.090531	0.102579
326	0.314263	0.126417	0.322833
327	0.313523	0.065445	0.403855
328	0.164261	0.057745	-0.005490
329	0.122141	0.024122	0.009190
330	0.308248	0.078401	0.180577
331	0.251222	0.073868	0.160457
332	-0.047526	0.023725	0.086336
333	-0.091643	0.005539	0.093179
334	0.079339	0.044135	0.206697
335	0.104213	0.011277	0.240060
336	0.226607	0.186234	-0.056881
337	0.173281	0.158131	-0.059413
338	0.339400	0.214501	0.052905
339	0.309166	0.188181	0.058028
340	0.014442	0.194715	0.048945
341	-0.028793	0.194766	0.089078
342	0.069564	0.206743	0.193568
343	0.091532	0.202786	0.269680
344	-0.071196	0.135604	-0.103744
345	-0.118288	0.152837	-0.060151
346	0.146856	0.143174	0.061789
347	0.104379	0.143672	0.056797
348	-0.541832	0.250034	-0.017602
349	-0.641583	0.278411	-0.111909
350	-0.094447	0.159393	0.164848
351	-0.113612	0.120702	0.221656
352	0.204918	-0.078894	0.075524
353	0.161232	-0.090256	0.088701
354	0.378460	-0.033687	0.309964
355	0.311701	-0.049984	0.316881
356	0.019311	-0.050048	0.212387
357	0.002473	-0.062855	0.278462
358	0.151448	-0.090652	0.410031
359	0.162778	-0.071291	0.531252
360	-0.083704	-0.076839	-0.020798
361	-0.092832	-0.043492	0.029202
362	0.136844	-0.077791	0.186493
363	0.089536	-0.086826	0.184711
364	-0.270255	-0.058858	0.173048
365	-0.350416	-0.009219	0.273260
366	-0.105248	-0.205534	0.425159
367	-0.135030	-0.197464	0.623550
368	-0.051717	0.069756	-0.043829
369	-0.081050	0.056947	-0.000205
370	0.190388	0.016366	0.145922
371	0.142662	0.002575	0.159182
372	-0.352890	0.011117	0.091040
373	-0.367374	0.056547	0.147209

374	-0.003179	0.026570	0.282541
375	-0.069934	-0.005171	0.337678
376	-0.496181	0.026464	0.019432
377	-0.690384	0.069313	-0.004175
378	-0.146138	0.046372	0.161839
379	-0.197581	0.034093	0.241003
380	-0.989567	0.040993	0.049384
381	-1.151075	0.210556	0.237374
382	-0.335366	-0.058208	0.480168
383	-0.502419	-0.093761	0.675240
384	0.862548	0.264137	-0.294905
385	0.782668	0.251324	-0.122108
386	1.597797	0.463818	-0.133153
387	1.615756	0.060653	0.084764
388	0.435588	0.209832	0.095050
389	0.431013	0.165328	0.047909
390	1.248164	0.265923	0.488086
391	1.009933	0.345440	0.473702
392	0.477017	0.194237	-0.058012
393	0.401362	0.186915	-0.054137
394	1.202158	0.284782	-0.066531
395	1.064907	0.203766	0.046383
396	0.255848	0.133398	0.046049
397	0.218680	0.128833	0.065326
398	0.490817	0.182041	0.286583
399	0.440714	0.106576	0.301120
400	0.604263	0.522925	-0.238629
401	0.526329	0.377577	-0.198100
402	1.038632	0.606242	-0.121253
403	0.995283	0.552202	0.110700
404	0.262232	0.313664	-0.086909
405	0.230835	0.273385	-0.054268
406	0.548466	0.490721	0.278201
407	0.466984	0.355859	0.289160
408	0.367137	0.236160	-0.228114
409	0.309359	0.233843	-0.171325
410	0.465268	0.276569	0.010951
411	0.378124	0.250237	0.011131
412	0.061885	0.296810	-0.011420
413	0.000125	0.350029	-0.011277
414	0.163815	0.261191	0.175863
415	0.165132	0.308797	0.227800
416	0.461418	0.052075	-0.016543
417	0.472372	0.046962	0.045746
418	0.856406	0.136415	0.245074
419	0.834616	0.003254	0.372643
420	0.337869	0.036994	0.232513
421	0.267414	0.027593	0.252779
422	0.584983	0.113046	0.583119
423	0.475406	-0.024234	0.655070
424	0.264823	-0.029292	0.004270
425	0.246071	-0.019109	0.030048
426	0.477401	0.021039	0.155448
427	0.458453	-0.043959	0.187850

428	0.067059	-0.061227	0.126904
429	0.044608	-0.034575	0.150205
430	0.191304	-0.003810	0.316776
431	0.153078	0.029915	0.361303
432	0.320704	0.178950	-0.088835
433	0.300866	0.137645	-0.056893
434	0.553442	0.162339	0.131987
435	0.490083	0.123682	0.146163
436	0.118950	0.083109	0.034052
437	0.099344	0.066212	0.054329
438	0.228325	0.122445	0.309219
439	0.172093	0.135754	0.323361
440	0.064213	0.063405	-0.058243
441	0.011906	0.088795	-0.069678
442	0.194232	0.129185	0.125708
443	0.155182	0.174013	0.144099
444	-0.217068	0.112731	0.093497
445	-0.307590	0.171146	0.110735
446	-0.014897	0.138094	0.232455
447	-0.036936	0.170135	0.279166
448	0.681886	0.437121	0.078458
449	0.548559	0.376914	0.092485
450	1.259194	0.901494	0.256085
451	1.296139	0.607949	0.302184
452	0.319619	0.307231	0.099647
453	0.287232	0.359355	0.186844
454	0.751306	0.676688	0.499386
455	0.479609	0.553030	0.560447
456	0.276377	0.214032	-0.003661
457	0.238146	0.223595	0.028806
458	0.542688	0.266205	0.171393
459	0.460188	0.283979	0.158288
460	0.057385	0.309853	0.144517
461	-0.006881	0.348152	0.097310
462	0.244434	0.247298	0.322601
463	0.253992	0.335420	0.402241
464	0.354006	0.579776	-0.130176
465	0.267043	0.461976	-0.058178
466	0.534049	0.626549	0.046747
467	0.441835	0.468260	0.057556
468	0.110477	0.628795	0.102950
469	0.031409	0.489068	0.090605
470	0.229564	0.525640	0.325454
471	0.105570	0.582151	0.509738
472	0.005690	0.521474	-0.157885
473	0.104463	0.424022	-0.080647
474	0.223784	0.389860	0.060904
475	0.159806	0.340571	0.062061
476	-0.173976	0.573425	0.027383
477	-0.376008	0.587868	0.133042
478	-0.051773	0.348339	0.231923
479	-0.122571	0.473049	0.251159
480	0.324321	0.148510	0.116006
481	0.282263	0.121730	0.114016

482	0.690108	0.256346	0.418128
483	0.542523	0.294427	0.461973
484	0.056944	0.107667	0.281797
485	0.027844	0.106858	0.355071
486	0.160456	0.177656	0.528819
487	0.227537	0.177976	0.689465
488	0.111585	0.097896	0.109244
489	0.083994	0.133245	0.115789
490	0.208740	0.142084	0.208953
491	0.156072	0.143303	0.231368
492	-0.185830	0.214347	0.309774
493	-0.311053	0.240517	0.328512
494	-0.041749	0.090901	0.511373
495	-0.156164	0.098486	0.478020
496	0.151543	0.263073	-0.033471
497	0.126322	0.213004	-0.007014
498	0.245313	0.217564	0.120210
499	0.259136	0.225542	0.176601
500	-0.190632	0.260214	0.141755
501	-0.189271	0.331768	0.170606
502	0.054763	0.294766	0.357775
503	-0.033724	0.257645	0.365069
504	-0.184971	0.396532	0.057728
505	-0.293313	0.400259	0.001123
506	-0.015219	0.232287	0.177913
507	-0.022524	0.244724	0.240753
508	-0.520342	0.347950	0.249265
509	-0.671997	0.410782	0.153434
510	-0.253089	0.412356	0.489854
511	-0.410922	0.562454	0.543891

Codebuchindex	PRBA47(0)	PRBA47(1)	PRBA47(2)	PRBA47(3)
0	-0.103660	0.094597	-0.013149	0.081501
1	-0.170709	0.129958	-0.057316	0.112324
2	-0.095113	0.080892	-0.027554	0.003371
3	-0.154153	0.113437	-0.074522	0.003446
4	-0.109553	0.153519	0.006858	0.040930
5	-0.181931	0.217882	-0.019042	0.040049
6	-0.096246	0.144191	-0.024147	-0.035120
7	-0.174811	0.193357	-0.054261	-0.071700
8	-0.183241	-0.052840	0.117923	0.030960
9	-0.242634	0.009075	0.098007	0.091643
10	-0.143847	-0.028529	0.040171	-0.002812
11	-0.198809	0.006990	0.020668	0.026641
12	-0.233172	-0.028793	0.140130	-0.071927
13	-0.309313	0.056873	0.108262	-0.018930
14	-0.172782	-0.002037	0.048755	-0.087065
15	-0.242901	0.036076	0.015064	-0.064366

Tabelle A.1: PRBA13-Codebuch

16	0.077107	0.172685	0.159939	0.097456
17	0.024820	0.209676	0.087347	0.105204
18	0.085113	0.151639	0.084272	0.022747
19	0.047975	0.196695	0.038770	0.029953
20	0.113925	0.236813	0.176121	0.016635
21	0.009708	0.267969	0.127660	0.015872
22	0.114044	0.202311	0.096892	-0.043071
23	0.047219	0.260395	0.050952	-0.046996
24	-0.055095	0.034041	0.200464	0.039050
25	-0.061582	0.069566	0.113048	0.027511
26	-0.025469	0.040440	0.132777	-0.039098
27	-0.031388	0.064010	0.067559	-0.017117
28	-0.074386	0.086579	0.228232	-0.055461
29	-0.107352	0.120874	0.137364	-0.030252
30	-0.036897	0.089972	0.155831	-0.128475
31	-0.059070	0.097879	0.084489	-0.075821
32	-0.050865	-0.025167	-0.086636	0.011256
33	-0.051426	0.013301	-0.144665	0.038541
34	-0.073831	-0.028917	-0.142416	-0.025268
35	-0.083910	0.015004	-0.227113	-0.002808
36	-0.030840	-0.009326	-0.070517	-0.041304
37	-0.022018	0.029381	-0.124961	-0.031624
38	-0.064222	-0.014640	-0.108798	-0.092342
39	-0.038801	0.038133	-0.188992	-0.094221
40	-0.154059	-0.183932	-0.019894	0.082105
41	-0.188022	-0.113072	-0.117380	0.090911
42	-0.243301	-0.207086	-0.053735	-0.001975
43	-0.275931	-0.121035	-0.161261	0.004231
44	-0.118142	-0.157537	-0.036594	-0.008679
45	-0.153627	-0.111372	-0.103095	-0.009460
46	-0.173458	-0.180158	-0.057130	-0.103198
47	-0.208509	-0.127679	-0.149336	-0.109289
48	0.096310	0.047927	-0.024094	-0.057018
49	0.044289	0.075486	-0.008505	-0.067635
50	0.076751	0.025560	-0.066428	-0.102991
51	0.025215	0.090417	-0.058616	-0.114284
52	0.125980	0.070078	0.016282	-0.112355
53	0.070859	0.118988	0.001180	-0.116359
54	0.097520	0.059219	-0.026821	-0.172850
55	0.048226	0.145459	-0.050093	-0.188853
56	0.007242	-0.135796	0.147832	-0.034080
57	0.012843	-0.069616	0.077139	-0.047909
58	-0.050911	-0.116323	0.082521	-0.056362
59	-0.039630	-0.055678	0.036066	-0.067992
60	0.042694	-0.091527	0.150940	-0.124225
61	0.029225	-0.039401	0.071664	-0.113665
62	-0.025085	-0.099013	0.074622	-0.138674
63	-0.031220	-0.035717	0.020870	-0.143376
64	0.040638	0.087903	-0.049500	0.094607
65	0.026860	0.125924	-0.103449	0.140882
66	0.075166	0.110186	-0.115173	0.067330
67	0.036642	0.163193	-0.188762	0.103724
68	0.028179	0.095124	-0.053258	0.028900
69	0.002307	0.148211	-0.096037	0.046189

70	0.072227	0.137595	-0.095629	0.001339
71	0.033308	0.221480	-0.152201	0.012125
72	0.003458	-0.085112	0.041850	0.113836
73	-0.040610	-0.044880	0.029732	0.177011
74	0.011404	-0.054324	-0.012426	0.077815
75	-0.042413	-0.030930	-0.034844	0.122946
76	-0.002206	-0.045698	0.050651	0.054886
77	-0.041729	-0.016110	0.048005	0.102125
78	0.013963	-0.022204	0.001613	0.028997
79	-0.030218	-0.002052	-0.004365	0.065343
80	0.299049	0.046260	0.076320	0.070784
81	0.250160	0.098440	0.012590	0.137479
82	0.254170	0.095310	0.018749	0.004288
83	0.218892	0.145554	-0.035161	0.069784
84	0.303486	0.101424	0.135996	-0.013096
85	0.262919	0.165133	0.077237	0.071721
86	0.319358	0.170283	0.054554	-0.072210
87	0.272983	0.231181	-0.014471	0.011689
88	0.134116	-0.026693	0.161400	0.110292
89	0.100379	0.026517	0.086236	0.130478
90	0.144718	-0.000895	0.093767	0.044514
91	0.114943	0.022145	0.035871	0.069193
92	0.122051	0.011043	0.192803	0.022796
93	0.079482	0.026156	0.117725	0.056565
94	0.124641	0.027387	0.122956	-0.025369
95	0.090708	0.027357	0.064450	0.013058
96	0.159781	-0.055202	-0.090597	0.151598
97	0.084577	-0.037203	-0.126698	0.119739
98	0.192484	-0.100195	-0.162066	0.104148
99	0.114579	-0.046270	-0.219547	0.100067
100	0.153083	-0.010127	-0.086266	0.068648
101	0.088202	-0.010515	-0.102196	0.046281
102	0.164494	-0.057325	-0.132860	0.024093
103	0.109419	-0.013999	-0.169596	0.020412
104	0.039180	-0.209168	-0.035872	0.087949
105	0.012790	-0.177723	-0.129986	0.073364
106	0.045261	-0.256694	-0.088186	0.004212
107	-0.005314	-0.231202	-0.191671	-0.002628
108	0.037963	-0.153227	-0.045364	0.003322
109	0.030800	-0.126452	-0.114266	-0.010414
110	0.044125	-0.184146	-0.081400	-0.077341
111	0.029204	-0.157393	-0.172017	-0.089814
112	0.393519	-0.043228	-0.111365	-0.000740
113	0.289581	0.018928	-0.123140	0.000713
114	0.311229	-0.059735	-0.198982	-0.081664
115	0.258659	0.052505	-0.211913	-0.034928
116	0.300693	0.011381	-0.083545	-0.086683
117	0.214523	0.053878	-0.101199	-0.061018
118	0.253422	0.028496	-0.156752	-0.163342
119	0.199123	0.113877	-0.166220	-0.102584
120	0.249134	-0.165135	0.028917	0.051838
121	0.156434	-0.123708	0.017053	0.043043
122	0.214763	-0.101243	-0.005581	-0.020703
123	0.140554	-0.072067	-0.015063	-0.011165
124	0.241791	-0.152048	0.106403	-0.046857
125	0.142316	-0.131899	0.054076	-0.026485
126	0.206535	-0.086116	0.046640	-0.097615
127	0.129759	-0.081874	0.004693	-0.073169

Tabelle A.2: PRBA47-Codebuch

Anhang B: HOC-Codebücher

Codebuchindex	HOC0(0)	HOC0(1)	HOC0(2)	HOC0(3)
0	0.264108	0.045976	-0.200999	-0.122344
1	0.479006	0.227924	-0.016114	-0.006835
2	0.077297	0.080775	-0.068936	0.041733
3	0.185486	0.231840	0.182410	0.101613
4	-0.012442	0.223718	-0.277803	-0.034370
5	-0.059507	0.139621	-0.024708	-0.104205
6	-0.248676	0.255502	-0.134894	-0.058338
7	-0.055122	0.427253	0.025059	-0.045051
8	-0.058898	-0.061945	0.028030	-0.022242
9	0.084153	0.025327	0.066780	-0.180839
10	-0.193125	-0.082632	0.140899	-0.089559
11	0.000000	0.033758	0.276623	0.002493
12	-0.396582	-0.049543	-0.118100	-0.208305
13	-0.287112	0.096620	0.049650	-0.079312
14	-0.543760	0.171107	-0.062173	-0.010483
15	-0.353572	0.227440	0.230128	-0.032089
16	0.248579	-0.279824	-0.209589	0.070903
17	0.377604	-0.119639	0.008463	-0.005589
18	0.102127	-0.093666	-0.061325	0.052082
19	0.154134	-0.105724	0.099317	0.187972
20	-0.139232	-0.091146	-0.275479	-0.038435
21	-0.144169	0.034314	-0.030840	0.022207
22	-0.143985	0.079414	-0.194701	0.175312
23	-0.195329	0.087467	0.067711	0.186783
24	-0.123515	-0.377873	-0.209929	-0.212677
25	0.068698	-0.255933	0.120463	-0.095629
26	-0.106810	-0.319964	-0.089322	0.106947
27	-0.158605	-0.309606	0.190900	0.089340
28	-0.489162	-0.432784	-0.151215	-0.005786
29	-0.370883	-0.154342	-0.022545	0.114054
30	-0.742866	-0.204364	-0.123865	-0.038888
31	-0.573077	-0.115287	0.208879	-0.027698

Codebuchindex	HOC1(0)	HOC1(1)	HOC1(2)	HOC1(3)
0	-0.143886	0.235528	-0.116707	0.025541
1	-0.170182	-0.063822	-0.096934	0.109704
2	0.232915	0.269793	0.047064	-0.032761
3	0.153458	0.068130	-0.033513	0.126553
4	-0.440712	0.132952	0.081378	-0.013210
5	-0.480433	-0.249687	-0.012280	0.007112
6	-0.088001	0.167609	0.148323	-0.119892
7	-0.104628	0.102639	0.183560	0.121674
8	0.047408	-0.000908	-0.214196	-0.109372
9	0.113418	-0.240340	-0.121420	0.041117

Tabelle B.1: HOC0-Codebuch

10	0.385609	0.042913	-0.184584	-0.017851
11	0.453830	-0.180745	0.050455	0.030984
12	-0.155984	-0.144212	0.018226	-0.146356
13	-0.104028	-0.260377	0.146472	0.101389
14	0.012376	-0.000267	0.006657	-0.013941
15	0.165852	-0.103467	0.119713	-0.075455

Tabelle B.2: HOC1-Codebuch

Codebuchindex	HOC2(0)	HOC2(1)	HOC2(2)	HOC2(3)
0	0.182478	0.271794	-0.057639	0.026115
1	0.110795	0.092854	0.078125	-0.082726
2	0.057964	0.000833	0.176048	0.135404
3	-0.027315	0.098668	-0.065801	0.116421
4	-0.222796	0.062967	0.201740	-0.089975
5	-0.193571	0.309225	-0.014101	-0.034574
6	-0.389053	-0.181476	0.107682	0.050169
7	-0.345604	0.064900	-0.065014	0.065642
8	0.319393	-0.055491	-0.220727	-0.067499
9	0.460572	0.084686	0.048453	-0.011050
10	0.201623	-0.068994	-0.067101	0.108320
11	0.227528	-0.173900	0.092417	-0.066515
12	-0.016927	0.047757	-0.177686	-0.102163
13	-0.052553	-0.065689	0.019328	-0.033060
14	-0.144910	-0.238617	-0.195206	-0.063917
15	-0.024159	-0.338822	0.003581	0.060995

Tabelle B.3: HOC2-Codebuch

Codebuchindex	HOC3(0)	HOC3(1)	HOC3(2)	HOC3(3)
0	0.323968	0.008964	-0.063117	0.027909
1	0.010900	-0.004030	-0.125016	-0.080818
2	0.109969	0.256272	0.042470	0.000749
3	-0.135446	0.201769	-0.083426	0.093888
4	-0.441995	0.038159	0.022784	0.003943
5	-0.155951	0.032467	0.145309	-0.041725
6	-0.149182	-0.223356	-0.065793	0.075016
7	0.096949	-0.096400	0.083194	0.049306

Tabelle B.4: HOC3-Codebuch

Anhang C: MBE-Tonparameter

Tontyp	Frequenz-komponenten (Hz)	MBE-Modellparameter		
		Tonindex	Grund-frequenz (Hz)	Von Null verschiedene Oberwellen
Einzelton	156,25	5	156,25	1
Einzelton	187,5	6	187,5	1
...	...	...	...	...
Einzelton	375,0	12	375,0	1
Einzelton	406,3	13	203,13	2
...	...	...	...	...
Einzelton	781,25	25	390,63	2
Einzelton	812,50	26	270,83	3
...	...	...	...	...
Einzelton	1187,5	38	395,83	3
Einzelton	1218,75	39	304,69	4
...	...	...	...	...
Einzelton	1593,75	51	398,44	4
Einzelton	1625,0	52	325,0	5
...	...	...	...	...
Einzelton	2000,0	64	400,0	5
Einzelton	2031,25	65	338,54	6
...	...	...	...	...
Einzelton	2375,0	76	395,83	6
Einzelton	2406,25	77	343,75	7
...	...	...	...	...
Einzelton	2781,25	89	397,32	7
Einzelton	2812,5	90	351,56	8
...	...	...	...	...
Einzelton	3187,5	102	398,44	8
Einzelton	3218,75	103	357,64	9

...	...	...	...	...
Einzelton	3593,75	115	399,31	9
Einzelton	3625,0	116	362,5	10
...	...	...	...	...
Einzelton	3812,5	122	381,25	10
MFV-Ton	941,1336	128	78,50	12, 17
MFV-Ton	697,1209	129	173,48	4, 7
MFV-Ton	697,1336	130	70,0	10, 19
MFV-Ton	697,1477	131	87,0	8, 17
MFV-Ton	770,1209	132	109,95	7, 11
MFV-Ton	770,1336	133	191,68	4, 7
MFV-Ton	770,1477	134	70,17	11, 21
MFV-Ton	852,1209	135	71,06	12, 17
MFV-Ton	852,1336	136	121,58	7, 11
MFV-Ton	852,1477	137	212,0	4, 7
MFV-Ton	697,1633	138	116,41	6, 14
MFV-Ton	770,1633	139	96,15	8, 17
MFV-Ton	852,1633	140	71,0	12, 23
MFV-Ton	941,1633	141	234,26	4, 7
MFV-Ton	941,1209	142	134,38	7, 9
MFV-Ton	941,1477	143	134,35	7, 11
Knox-Ton	820,1162	144	68,33	12, 17
Knox-Ton	606,1052	145	150,89	4, 7
Knox-Ton	606,1162	146	67,82	9, 17
Knox-Ton	606,1297	147	86,50	7, 15
Knox-Ton	672,1052	148	95,79	7, 11
Knox-Ton	672,1162	149	166,92	4, 7
Knox-Ton	672,1297	150	67,70	10, 19
Knox-Ton	743,1052	151	74,74	10, 14
Knox-Ton	743,1162	152	105,90	7, 11
Knox-Ton	743,1297	153	92,78	8, 14
Knox-Ton	606,1430	154	101,55	6, 14

Knox-Ton	672,1430	155	84,02	8, 17
Knox-Ton	743,1430	156	67,83	11, 21
Knox-Ton	820,1430	157	102,30	8, 14
Knox-Ton	820,1052	158	117,0	7, 9
Knox-Ton	820,1297	159	117,49	7, 11
Hörton	350,440	160	87,78	4, 5
Hörton	440,480	161	70,83	6, 7
Hörton	480,630	162	122,0	4, 5
Hörton	350,490	163	70,0	5, 7

Patentansprüche

1. Verfahren zum Codieren einer Sequenz von digitalen Sprachabtastwerten in einen Bitstrom, wobei das Verfahren umfasst:
Unterteilen der digitalen Sprachabtastwerte in einen oder mehrere Datenblöcke;
Berechnen von Modellparametern für einen Datenblock;
Quantisieren der Modellparameter, um Tonhöhenbits, die Tonhöheninformationen übermitteln, Sprachbits, die Sprachinformationen übermitteln, und Verstärkungsbits, die Signalpegelinformationen übermitteln, zu erzeugen;
Kombinieren von einem oder mehreren der Tonhöhenbits mit einem oder mehreren der Sprachbits und einem oder mehreren der Verstärkungsbits, um ein erstes Parametercodewort zu erzeugen;
Codieren des ersten Parametercodeworts mit einem Fehlerprüfcode, um ein erstes FEC-Codewort zu erzeugen; und
Aufnehmen des ersten FEC-Codeworts in einen Bitstrom für den Datenblock.

2. Verfahren nach Anspruch 1, wobei das Berechnen der Modellparameter für den Datenblock das Berechnen eines Grundfrequenzparameters, einer oder mehrerer Sprachentscheidungen und eines Satzes von Spektralparametern umfasst.

3. Verfahren nach Anspruch 2, wobei das Berechnen der Modellparameter für einen Datenblock die Verwendung des Mehrbandanregungs-Sprachmodells umfasst.

4. Verfahren nach Anspruch 2 oder Anspruch 3, wobei das Quantisieren der Modellparameter das Erzeugen der Tonhöhenbits durch Anwenden einer logarithmischen Funktion auf den Grundfrequenzparameter umfasst.

5. Verfahren nach einem der Ansprüche 2 bis 4, wobei das Quantisieren der Modellparameter das Erzeugen von Sprachbits durch gemeinsames Quantisieren von Sprachentscheidungen für den Datenblock umfasst.

6. Verfahren nach Anspruch 5, wobei:
die Sprachbits einen Index in ein Sprachcodebuch darstellen, und
der Wert des Sprachcodebuchs für zwei oder mehr verschiedene Werte des Index gleich ist.

7. Verfahren nach einem der vorangehenden Ansprüche, wobei das erste Parametercodewort zwölf Bits umfasst.

8. Verfahren nach Anspruch 7, wobei das erste Parametercodewort durch Kombinieren von vier der Tonhöhenbits plus vier der Sprachbits plus vier der Verstärkungsbits gebildet wird.

9. Verfahren nach einem der vorangehenden Ansprüche, wobei das erste Parametercodewort mit einem Golay-Fehlerprüfcode codiert wird.

10. Verfahren nach einem der vorangehenden Ansprüche, wobei:

die Spektralparameter einen Satz von logarithmischen Spektralamplituden umfassen, und
die Verstärkungsbits zumindest teilweise durch Berechnen des Mittelswerts der logarithmischen Spektralamplituden erzeugt werden.

11. Verfahren nach Anspruch 10, welches ferner umfasst:
Quantisieren der logarithmischen Spektralamplituden in Spektralbits; und
Kombinieren einer Vielzahl der Spektralbits, um ein zweites Parametercodewort zu erzeugen; und
Codieren des zweiten Parametercodeworts mit einem zweiten Fehlerprüfcode, um ein zweites FEC-Codewort zu erzeugen,
wobei das zweite FEC-Codewort auch in den Bitstrom für den Datenblock aufgenommen wird.

12. Verfahren nach Anspruch 11, wobei:
die Tonhöhenbits, die Sprachbits, die Verstärkungsbits und die Spektralbits jeweils in mehrere wichtige Bits und weniger wichtige Bits unterteilt werden, wobei die wichtigeren Tonhöhenbits, Sprachbits, Verstärkungsbits und Spektralbits im ersten Parametercodewort und im zweiten Parametercodewort aufgenommen werden und mit Fehlerprüfcodes codiert werden, und
die weniger wichtigen Tonhöhenbits, Sprachbits, Verstärkungsbits und Spektralbits im Bitstrom für den Datenblock ohne Codierung mit Fehlerprüfcodes aufgenommen werden.

13. Verfahren nach Anspruch 12, wobei:
7 Tonhöhenbits vorhanden sind, die in 4 wichtigere Tonhöhenbits und 3 weniger wichtige Tonhöhenbits unterteilt werden,
5 Sprachbits vorhanden sind, die in 4 wichtigere Sprachbits und 1 weniger wichtiges Sprachbit unterteilt werden, und
5 Verstärkungsbits vorhanden sind, die in 4 wichtigere Verstärkungsbits und 1 weniger wichtiges Verstärkungsbit unterteilt werden.

14. Verfahren nach Anspruch 13, wobei der zweite Parametercode zwölf wichtigere Spektralbits umfasst, die mit einem Golay-Fehlerprüfcode codiert werden, um das zweite FEC-Codewort zu erzeugen.

15. Verfahren nach Anspruch 14, welches ferner umfasst:
Berechnen eines Modulationsschlüssels aus dem ersten Parametercodewort;
Erzeugen einer Verwürfelungssequenz aus dem Modulationsschlüssel;
Kombinieren der Verwürfelungssequenz mit dem zweiten FEC-Codewort, um ein verwürfeltes zweites FEC-Codewort zu erzeugen; und
Aufnehmen des verwürfelten zweiten FEC-Codeworts in den Bitstrom für den Datenblock.

16. Verfahren nach einem der vorangehenden Ansprüche, welches ferner umfasst:
Erfassen von bestimmten Tonsignalen; und
wenn ein Tonsignal für einen Datenblock erfasst wird, dann Aufnehmen von Tonidentifikatorbits und Tonamplitudenbits in das erste Parametercodewort, wobei die Tonidentifikatorbits ermöglichen, dass die Bits für den Datenblock als einem Tonsignal entsprechend identifiziert werden.

17. Verfahren nach Anspruch 16, wobei:
wenn ein Tonsignal für einen Datenblock erfasst wird, dann zusätzliche Tonindexbits in den Bitstrom für den Datenblock aufgenommen werden, und die Tonindexbits Frequenzinformationen für das Tonsignal festlegen.

18. Verfahren nach Anspruch 17, wobei die Tonidentifikatorbits einem nicht zugelassenen Satz von Tonhöhenbits entsprechen, um zu ermöglichen, dass die Bits für den Datenblock als einem Tonsignal entsprechend identifiziert werden.

19. Verfahren nach Anspruch 18, wobei das erste Parametercodewort sechs Tonidentifikatorbits und sechs Tonamplitudenbits umfasst, wenn ein Tonsignal für einen Datenblock erfasst wird.

20. Verfahren zum Decodieren von digitalen Sprachabtastwerten von einem Bitstrom, wobei das Verfahren umfasst:
Unterteilen des Bitstroms in einen oder mehrere Datenblöcke von Bits;
Gewinnen eines ersten FEC-Codeworts aus einem Datenblock von Bits;
Fehlerprüfdecodieren des ersten FEC-Codeworts, um ein erstes Parametercodewort zu erzeugen;
Gewinnen von Tonhöhenbits, Sprachbits und Verstärkungsbits aus dem ersten Parametercodewort;

Verwenden der gewonnenen Tonhöhenbits, um zumindest teilweise Tonhöheninformationen für den Datenblock zu rekonstruieren;
Verwenden der gewonnenen Sprachbits, um zumindest teilweise Sprachinformationen für den Datenblock zu rekonstruieren;
Verwenden der gewonnenen Verstärkungsbits, um zumindest teilweise Signalpegelinformationen für den Datenblock zu rekonstruieren; und
Verwenden der rekonstruierten Tonhöheninformationen, Sprachinformationen und Signalpegelinformationen für einen oder mehrere Datenblöcke, um digitale Sprachabtastwerte zu berechnen.

21. Verfahren nach Anspruch 20, wobei die Tonhöheninformationen für einen Datenblock einen Grundfrequenzparameter umfassen und die Sprachinformationen für einen Datenblock eine oder mehrere Sprachentscheidungen umfassen.

22. Verfahren nach Anspruch 21, wobei die Sprachentscheidungen für den Datenblock unter Verwendung der Sprachbits als Index in ein Sprachcodebuch rekonstruiert werden.

23. Verfahren nach Anspruch 22, wobei der Wert des Sprachcodebuchs für zwei oder mehr verschiedene Indizes gleich ist.

24. Verfahren nach einem der Ansprüche 20 bis 23, welches ferner das Rekonstruieren von Spektralinformationen für einen Datenblock umfasst.

25. Verfahren nach einem der Ansprüche 20 bis 24, wobei:
die Spektralinformationen für einen Datenblock zumindest teilweise einen Satz von logarithmischen Spektralamplitudenparametern umfassen, und die Signalpegelinformationen verwendet werden, um den Mittelwert der logarithmischen Spektralamplitudenparameter zu ermitteln.

26. Verfahren nach einem der Ansprüche 20 bis 25, wobei:
das erste FEC-Codewort mit einem Golay-Decodierer decodiert wird, und vier Tonhöhenbits plus vier Sprachbits plus vier Verstärkungsbits aus dem ersten Parametercodewort gewonnen werden.

27. Verfahren nach einem der Ansprüche 20 bis 26, welches ferner umfasst:
Erzeugen eines Modulationsschlüssels aus dem ersten Parametercodewort;
Berechnen einer Verwürfelungssequenz aus dem Modulationsschlüssel;
Gewinnen eines zweiten FEC-Codeworts aus dem Datenblock von Bits;
Anwenden der Verwürfelungssequenz auf das zweite FEC-Codewort, um ein entwürfeltes zweites FEC-Codewort zu erzeugen;
Fehlerprüfdecodieren des entwürfelten zweiten FEC-Codeworts, um ein zweites Parametercodewort zu erzeugen;
Berechnen einer Fehlermetrik aus der Fehlerprüfdecodierung des ersten FEC-Codeworts und aus der Fehlerprüfdecodierung des entwürfelten zweiten FEC-Codeworts; und
Anwenden einer Datenblock-Fehlerverarbeitung, wenn die Fehlermetrik einen Schwellenwert überschreitet.

28. Verfahren nach Anspruch 27, wobei die Datenblock-Fehlerverarbeitung das Wiederholen des rekonstruierten Modellparameters von einem vorherigen Datenblock für den aktuellen Datenblock umfasst.

29. Verfahren nach Anspruch 27 oder Anspruch 28, wobei die Fehlermetrik die Summe der Anzahl von durch die Fehlerprüfdecodierung des ersten FEC-Codeworts und durch die Fehlerprüfdecodierung des entwürfelten zweiten FEC-Codeworts korrigierten Fehlern verwendet.

30. Verfahren nach einem der Ansprüche 27 bis 29, wobei die Spektralinformationen für einen Datenblock zumindest teilweise aus dem zweiten Parametercodewort rekonstruiert werden.

31. Verfahren zum Decodieren von digitalen Signalabtastwerten von einem Bitstrom, wobei das Verfahren umfasst:
Unterteilen des Bitstroms in einen oder mehrere Datenblöcke von Bits;
Gewinnen eines ersten FEC-Codeworts aus einem Datenblock von Bits;
Fehlerprüfdecodieren des ersten FEC-Codeworts, um ein erstes Parametercodewort zu erzeugen;
Verwenden des ersten Parametercodeworts, um festzustellen, ob der Datenblock von Bits einem Tonsignal entspricht;

Gewinnen von Tonamplitudenbits aus dem ersten Parametercodewort, wenn festgestellt wird, dass der Rahmen von Bits einem Tonsignal entspricht, ansonsten Gewinnen von Tonhöhenbits, Sprachbits und Verstärkungsbits aus dem ersten Codewort, wenn festgestellt wird, dass der Rahmen von Bits nicht einem Tonsignal entspricht; und
Verwenden entweder der Tonamplitudenbits oder der Tonhöhenbits, Sprachbits und Verstärkungsbits, um digitale Signalabtastwerte zu berechnen.

32. Verfahren nach Anspruch 31, welches ferner umfasst:
Erzeugen eines Modulationsschlüssels aus dem ersten Parametercodewort;
Berechnen einer Verwürfelungssequenz aus dem Modulationsschlüssel;
Gewinnen eines zweiten FEC-Codeworts aus dem Datenblock von Bits;
Anwenden der Verwürfelungssequenz auf das zweite FEC-Codewort, um ein entwürfeltes zweites FEC-Codewort zu erzeugen;
Fehlerprüfdecodieren des entwürfelten zweiten FEC-Codeworts, um ein zweites Parametercodewort zu erzeugen; und
Berechnen von digitalen Signalabtastwerten unter Verwendung des zweiten Parametercodeworts.

33. Verfahren nach Anspruch 32, welches ferner umfasst:
Summieren der Anzahl von durch die Fehlerprüfdecodierung des ersten FEC-Codeworts und durch die Fehlerprüfdecodierung des entwürfelten zweiten FEC-Codeworts korrigierten Fehlern, um eine Fehlermetrik zu berechnen; und
Anwenden einer Datenblock-Fehlerverarbeitung, wenn die Fehlermetrik einen Schwellenwert überschreitet, wobei die Datenblock-Fehlerverarbeitung das Wiederholen des rekonstruierten Modellparameters von einem vorherigen Datenblock umfasst.

34. Verfahren nach Anspruch 32 oder Anspruch 33, wobei zusätzliche Spektralbits aus dem zweiten Parametercodewort gewonnen werden und verwendet werden, um die digitalen Signalabtastwerte zu rekonstruieren.

35. Verfahren nach einem der Ansprüche 31 bis 34, wobei die Spektralbits Tonindexbits umfassen, wenn festgestellt wird, dass der Datenblock von Bits einem Tonsignal entspricht.

36. Verfahren nach Anspruch 35, wobei festgestellt wird, dass der Datenblock von Bits einem Tonsignal entspricht, wenn einige der Bits im ersten Parametercodewort gleich einem bekannten Tonidentifikatorwert sind, der einem nicht zugelassenen Wert der Tonhöhenbits entspricht.

37. Verfahren nach Anspruch 35 oder Anspruch 36, wobei die Tonindexbits verwendet werden, um zu identifizieren, ob der Datenblock von Bits einem Signalfrequenzton, einem MFV-Ton, einem Knox-Ton oder einem Hörton entspricht.

38. Verfahren nach einem der Ansprüche 31 bis 37, wobei:
die Spektralbits verwendet werden, um einen Satz von logarithmischen Spektralamplitudenparametern für den Datenblock zu rekonstruieren, und die Verstärkungsbits verwendet werden, um den Mittelwert der logarithmischen Spektralamplitudenparameter zu ermitteln.

39. Verfahren nach einem der Ansprüche 31 bis 38, wobei die Sprachbits als Index in ein Sprachcodebuch verwendet werden, um Sprachentscheidungen für den Datenblock zu rekonstruieren.

40. Verfahren nach einem der Ansprüche 31 bis 39, wobei:
das erste FEC-Codewort mit einem Golay-Decodierer decodiert wird, und vier Tonhöhenbits plus vier Sprachbits plus vier Verstärkungsbits aus dem ersten Parametercodewort gewonnen werden.

Es folgen 6 Blatt Zeichnungen

Anhängende Zeichnungen

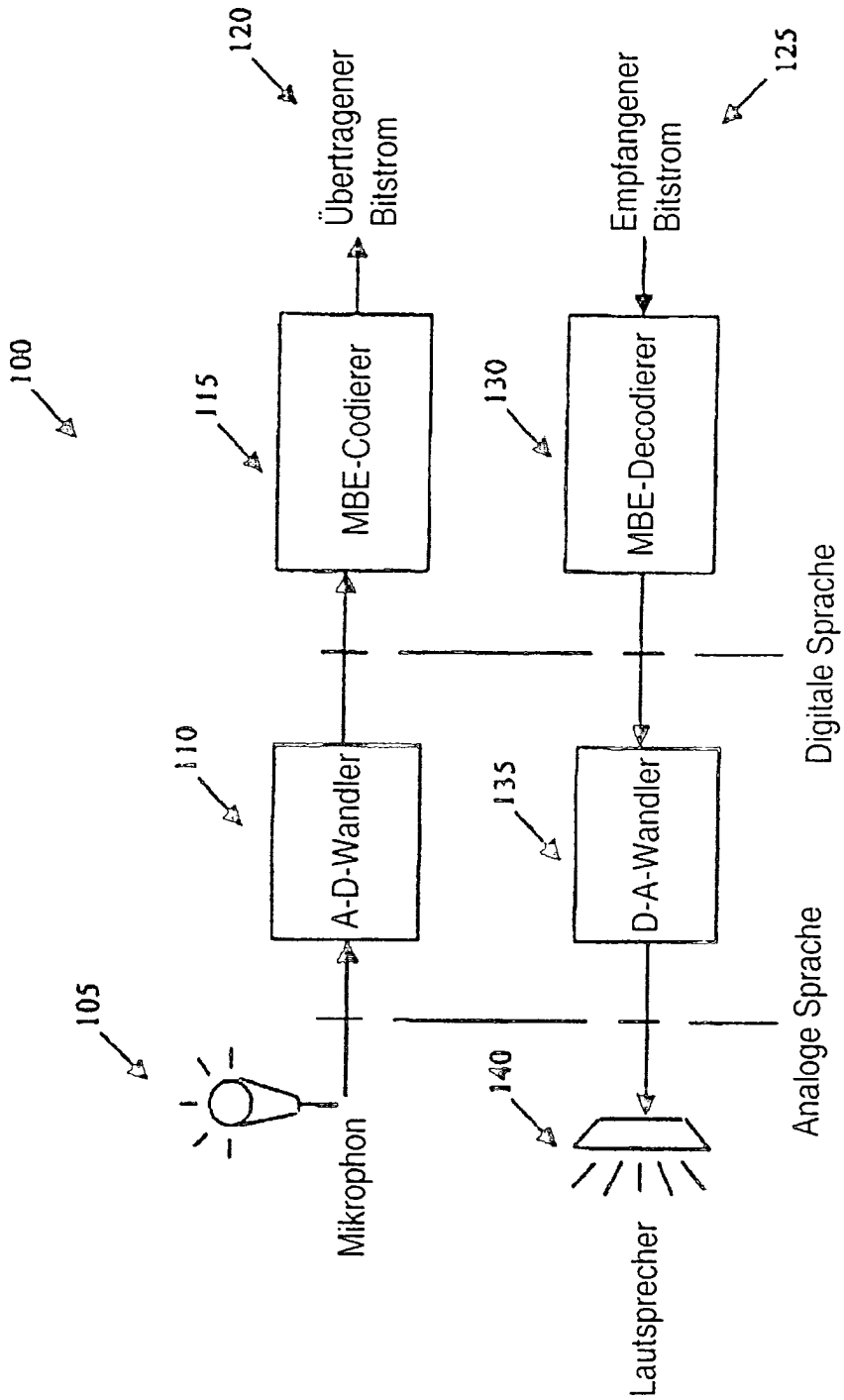

Fig. 1

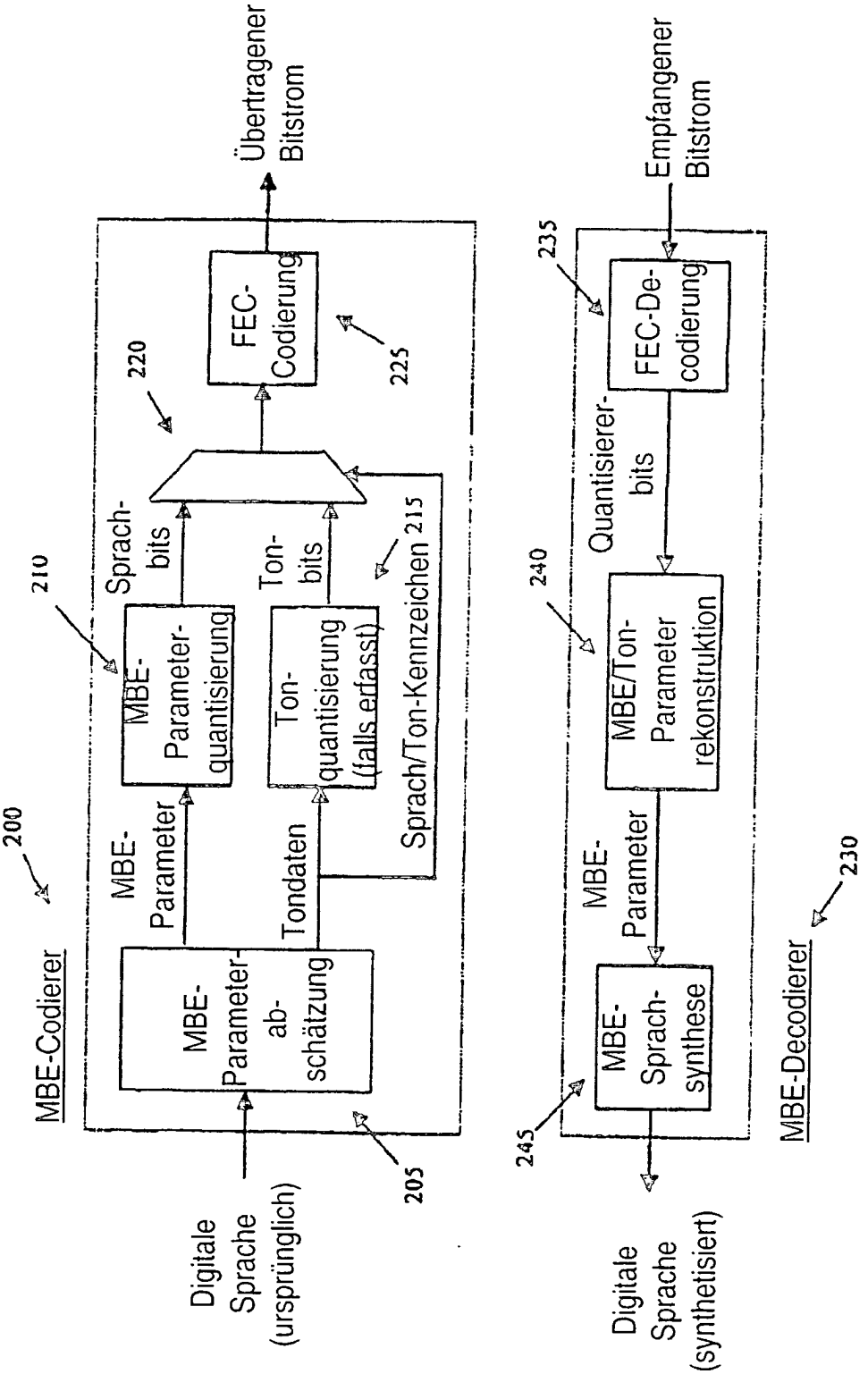
MBE-Codierer
200
210
MBE-Parameter-quantisierung
Ton-quantisierung (falls erfasst)
MBE-Parameter-ab-schätzung
MBE-Parameter
Tondaten
Sprach-bits
Ton-bits
Sprach/Ton-Kennzeichen 215
220
FEC-Codierung
225
Übertragener Bitstrom
Digitale Sprache (ursprünglich)
205
MBE-Decodierer
230
235
FEC-De-codierung
Quantisierer-bits
240
MBE/Ton-Parameter rekonstruktion
MBE-Parameter
MBE-Sprach-synthese
245
Empfangener Bitstrom
Digitale Sprache (synthetisiert)

Fig. 2

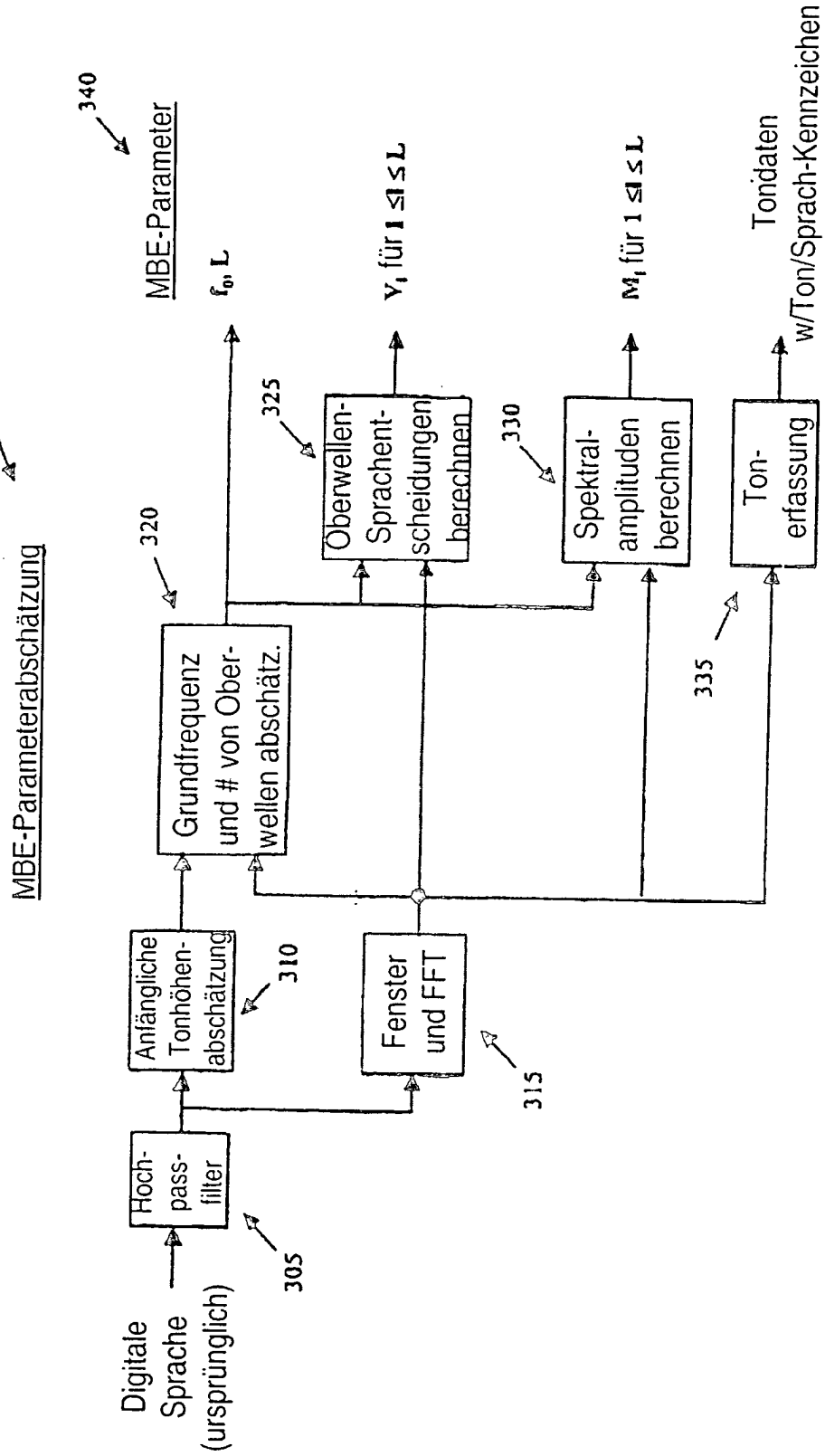
MBE-Parameterabschätzung
300
Digitale Sprache (ursprünglich)
Hoch-pass-filter
305
Anfängliche Tonhöhen-abschätzung
310
Fenster und FFT
315
Grundfrequenz und # von Ober-wellen abschätz.
320
Oberwellen-Sprachent-scheidungen berechnen
325
Spektral-amplituden berechnen
330
Ton-erfassung
335
MBE-Parameter
340
f_0, L
V_l für $l \leq L$
M_l für $l \leq L$
Tondaten w/Ton/Sprach-Kennzeichen

Fig. 3

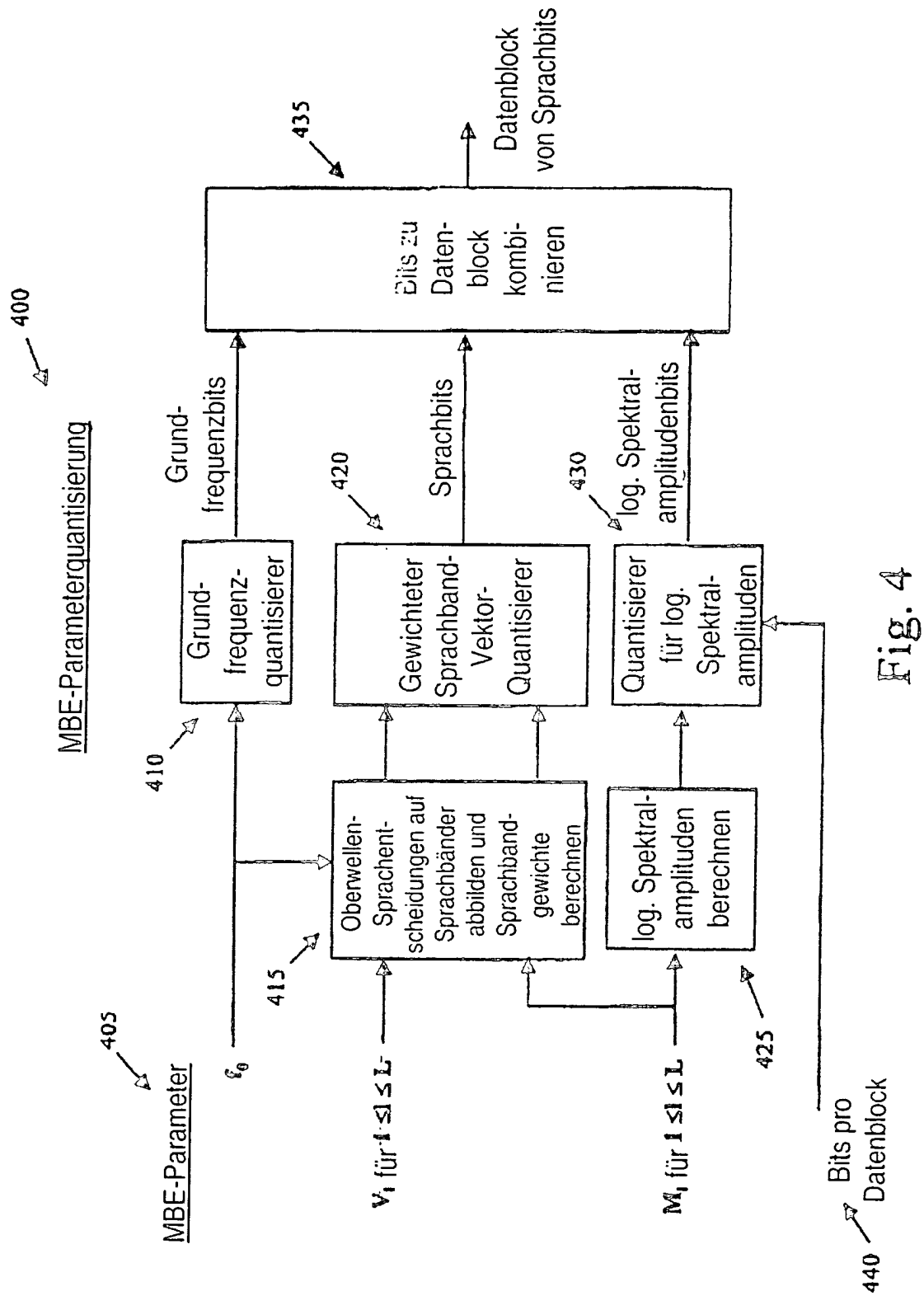
400
MBE-Parameterquantisierung
435
Bits zu Daten-block kombi-nieren
Datenblock von Sprachbits
Grund-frequenzbits
420
Sprachbits
430
log. Spektral-amplitudenbits
Grund-frequenz-quantisierer
Gewichteter Sprachband-Vektor-Quantisierer
Quantisierer für log. Spektral-amplituden
410
Oberwellen-Sprachent-scheidungen auf Sprachbänder abbilden und Sprachband-gewichte berechnen
log. Spektral-amplituden berechnen
415
425
405
MBE-Parameter
$\hat{\epsilon}_0$
$\hat{v}_l$ für $1 \le l \le L$
$\hat{M}_l$ für $1 \le l \le L$
Bits pro Datenblock
440

Fig. 4

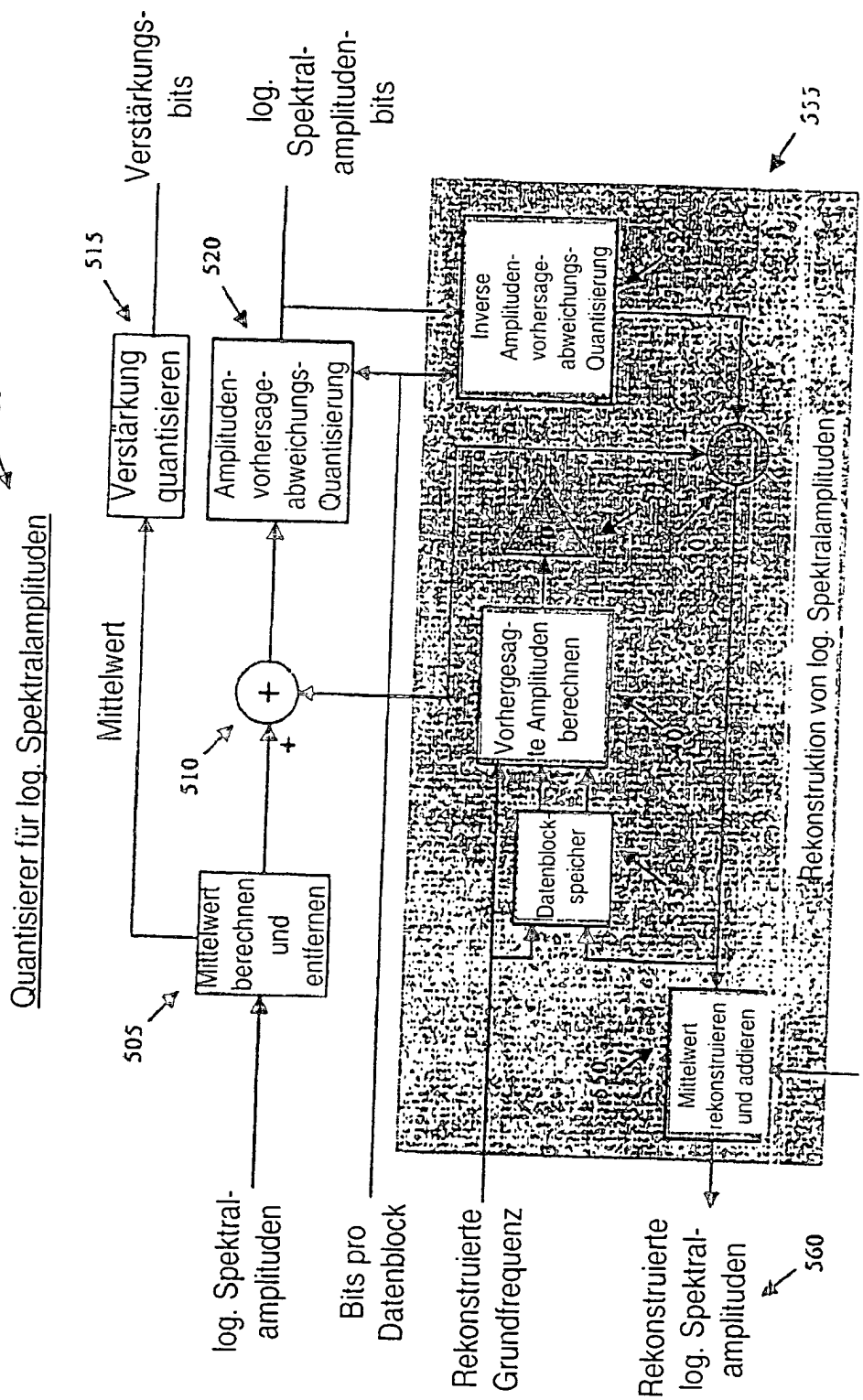
Quantisierer für log. Spektralamplituden
500
Verstärkungs-bits
log. Spektral-amplituden-bits
515
520
555
Verstärkung quantisieren
Amplituden-vorhersage-abweichungs-Quantisierung
Inverse Amplituden-vorhersage-abweichungs-Quantisierung
Mittelwert
510
+
+
Vorhergesagte Amplituden berechnen
Datenblock-speicher
Mittelwert berechnen und entfernen
Mittelwert rekonstruieren und addieren
Rekonstruktion von log. Spektralamplituden
505
560
log. Spektral-amplituden
Bits pro Datenblock
Rekonstruierte Grundfrequenz
Rekonstruierte log. Spektral-amplituden
Verstärkungs-bits

Fig. 5

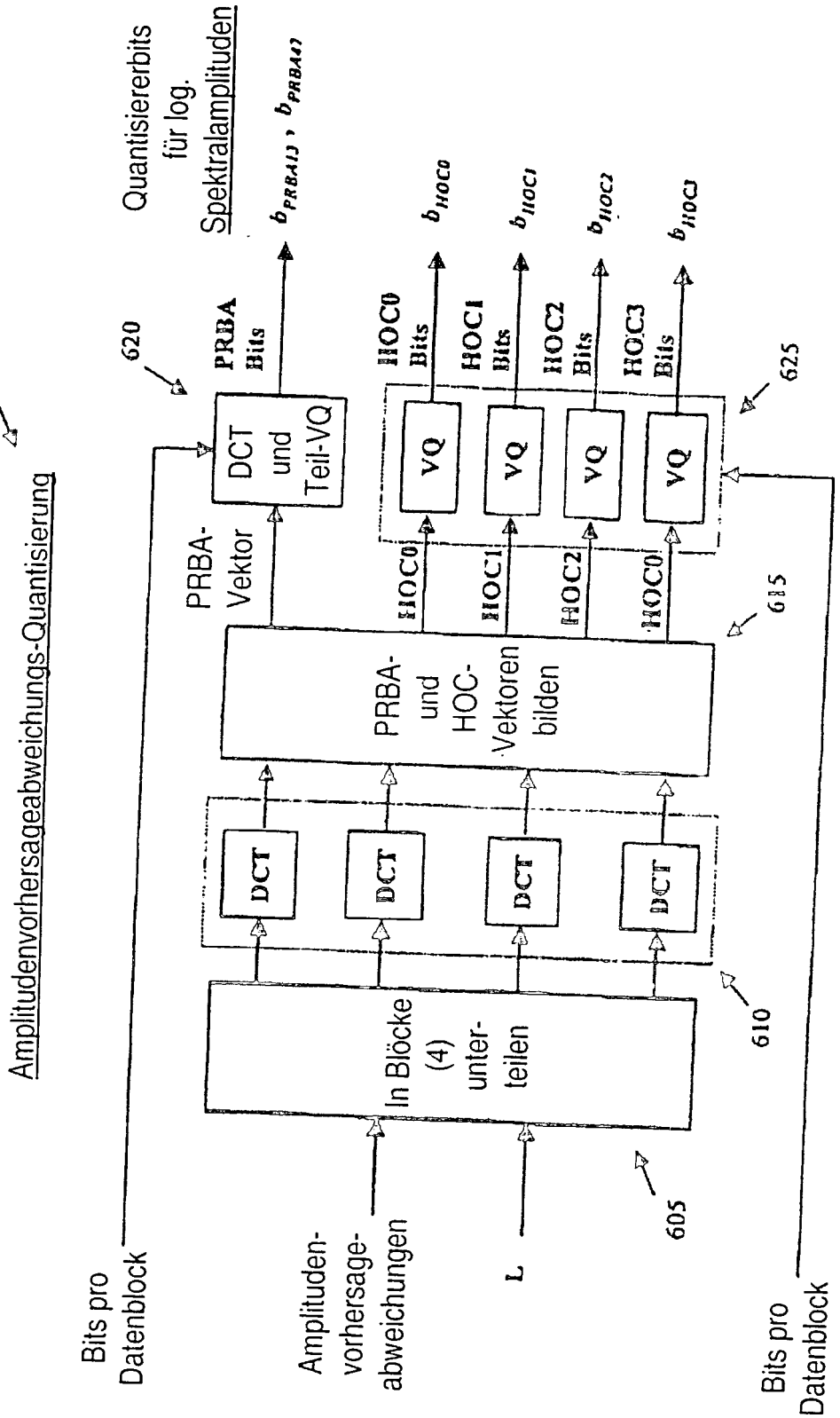
600
Amplitudenvorhersageabweichungs-Quantisierung
Quantisiererbits für log. Spektralamplituden
620
PRBA Bits
HOC0 Bits
HOC1 Bits
HOC2 Bits
HOC3 Bits
625
DCT und Teil-VQ
VQ
VQ
VQ
VQ
PRBA-Vektor
HOC0
HOC1
HOC2
HOC0
615
PRBA- und HOC-Vektoren bilden
DCT
DCT
DCT
DCT
In Blöcke (4) unter-teilen
610
605
Bits pro Datenblock
Amplituden-vorhersage-abweichungen
L
Bits pro Datenblock

Fig. 6