(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第5534528号
(P5534528)

(45)発行日　平成26年7月2日(2014.7.2)　(24)登録日　平成26年5月9日(2014.5.9)

(51)Int.Cl.　　F I
H03M 13/11 (2006.01)　　H03M 13/11
H03M 13/25 (2006.01)　　H03M 13/25

請求項の数 5　(全 45 頁)

(21)出願番号　特願2011-502100 (P2011-502100)
(86)(22)出願日　平成21年3月27日 (2009.3.27)
(65)公表番号　特表2011-528867 (P2011-528867A)
(43)公表日　平成23年11月24日 (2011.11.24)
(86)国際出願番号　PCT/US2009/038542
(87)国際公開番号　WO2009/120952
(87)国際公開日　平成21年10月1日 (2009.10.1)
審査請求日　平成24年3月19日 (2012.3.19)
(31)優先権主張番号　61/072,316
(32)優先日　平成20年3月28日 (2008.3.28)
(33)優先権主張国　米国 (US)
(31)優先権主張番号　08162031.2
(32)優先日　平成20年8月7日 (2008.8.7)
(33)優先権主張国　欧州特許庁 (EP)

(73)特許権者　501263810
トムソン　ライセンシング
Ｔｈｏｍｓｏｎ　Ｌｉｃｅｎｓｉｎｇ
フランス国，　９２１３０　イツシー　レ　ムーリノー，　ル　ジヤンヌ　ダルク，　１－５
１－５，　ｒｕｅ　Ｊｅａｎｎｅ　ｄ’Ａｒｃ，　９２１３０　ＩＳＳＹ　ＬＥＳ　ＭＯＵＬＩＮＥＡＵＸ，　Ｆｒａｎｃｅ
(74)代理人　100115864
弁理士　木越　力
(74)代理人　100121175
弁理士　石井　たかし
(74)代理人　100134094
弁理士　倉持　誠

最終頁に続く

(54)【発明の名称】信号を復号する装置および方法

(57)【特許請求の範囲】
【請求項１】
バイト・コード符号化プロセスを使用して符号化されたビットストリームを復号する方法であって、
バイト・コード符号化プロセスを使用して符号化された復調済みビットストリームを受信するステップと、
前記バイト・コード符号化プロセスに関連した復号プロセスに基づいて、前記復調済みビットストリームを複数のサブセットに配列するステップであって、該複数のサブセットの各々のサブセットが前記復調済みビットストリームの複数のビットを含み、前記復号プロセスが前記複数のビット間のパリティ関係を含む、該ステップと、
前記複数のサブセットの一つのサブセット中のシンボルのセットを特定するステップであって、該シンボルのセット中の各々のシンボルが前記複数のサブセットの前記サブセット中に少なくとも２つのビットを含み、前記シンボルのセットが前記復調済みビットストリームに対する変調フォーマットに関連する、該ステップと、
前記複数のサブセットの前記サブセット中の前記シンボルのセットを並べ替えるステップと、
前記シンボルのセット間のトレリス関係と、前記並べ替えたサブセット中の前記ビット間の前記パリティ関係とに基づいて、並べ替えたサブセットの各々を復号するステップと、
を含む、前記方法。

【請求項２】
前記パリティ関係が、パリティ関係のセットである、請求項１に記載の方法。
【請求項３】
前記復号ステップが、畳み込み復号プロセスを含む、請求項１に記載の方法。
【請求項４】
バイト・コード符号化プロセスを使用して符号化されたビットストリームを復号する装置であって、
バイト・コード符号化プロセスを使用して符号化された復調済みビットストリームを受信する手段と、
前記バイト・コード符号化プロセスに関連した復号プロセスに基づいて、前記復調済みビットストリームを複数のサブセットに配列する手段であって、該複数のサブセットの各々のサブセットが前記復調済みビットストリームの複数のビットを含み、前記復号プロセスが前記複数のビット間のパリティ関係を含む、該手段と、
前記複数のサブセットの一つのサブセット中のシンボルのセットを特定する手段であって、該シンボルのセット中の各々のシンボルが前記複数のサブセットの前記サブセット中に少なくとも２つのビットを含み、前記シンボルのセットが前記復調済みビットストリームに対する変調フォーマットに関連する、該手段と、
前記複数のサブセットの前記サブセット中の前記シンボルのセットを並べ替える手段と、
前記シンボルのセット間のトレリス関係と、前記並べ替えたサブセット中の前記ビット間の前記パリティ関係とに基づいて、並べ替えたサブセットの各々を復号する手段と、
を備える、前記装置。
【請求項５】
前記復号手段が、畳み込み復号プロセスを含む、請求項４に記載の装置。
【発明の詳細な説明】
【技術分野】
【０００１】
（関連出願の相互参照）
本願は、２００８年3月２８日出願の米国仮特許出願第６１／０７２３１６号の米国特許法１１９条に基づく特典、および２００８年８月７日出願の欧州特許出願ＥＰ０８１６２０３１号の米国特許法１１９条に基づく特典を請求するものである。
【０００２】
本開示は、一般に、ディジタル信号データ伝送システムの動作に関し、より詳細には、移動デバイス、ペデストリアン・デバイス（ｐｅｄｅｓｔｒｉａｎ　ｄｅｖｉｃｅ）、パーソナル・デバイスで使用される放送テレビジョン用データの復号に関する。
【背景技術】
【０００３】
最初に、以下に述べる本発明の様々な態様と関係がある可能性のある当技術分野の様々な態様を、読者に紹介しておく。ここで述べる内容は、本発明の様々な態様の理解を容易にする背景情報を読者に提供するのに役立つものと考えられるものである。つまり、以下の記述は、この点に鑑みて読まれるべきものであり、従来技術を承認するものではないことを理解されたい。
【０００４】
世界中のテレビジョン放送システムが、アナログの音声信号および映像信号を送出するものから、最新のディジタル通信システムに移行している。例えば、米国では、ＡＴＳＣ（Ａｄｖａｎｃｅｄ　Ｔｅｌｅｖｉｓｉｏｎ　Ｓｙｓｔｅｍｓ　Ｃｏｍｍｉｔｔｅｅ）が、「ＡＴＳＣ標準：ディジタル・テレビジョン標準Ａ／５３」（Ａ５３標準）という標準を開発している。Ａ５３標準は、ディジタル・テレビジョン放送用のデータの符号化および復号の方法を規定するものである。さらに、米国連邦通信委員会（ＦＣＣ）により、電磁スペクトルの一部がテレビジョン放送に割り当てられている。ＦＣＣは、この割当て部

分の中から、連続する６ＭＨｚのチャネルを、地上波（すなわちケーブルや衛星でない）ディジタル・テレビジョン放送の伝送を行う各放送業者に割り当てている。Ａ５３標準の符号化および変調のフォーマットに基づき、６ＭＨｚチャネルそれぞれのチャネル容量は毎秒約１９Ｍｂである。さらに、ＦＣＣは、６ＭＨｚチャネルを介した地上波ディジタル・テレビジョン・データの伝送は、Ａ５３標準に準拠していなければならないと定めている。

【０００５】

Ａ５３標準などのディジタル放送信号伝送標準は、ソース・データ（例えばディジタル音声データおよびディジタル映像データ）をどのように処理および変調して上記チャネルを介して伝送される信号にするかを規定している。この処理では、チャネルによって伝送信号に雑音およびマルチパス干渉が付加された場合でも、チャネルから信号を受信した受信側がソース・データを回復することができるように、ソース・データに冗長情報を付加する。ソース・データに冗長情報を付加することにより、ソース・データを伝送する有効データ・レートは低下するが、伝送信号からソース・データをうまく回復できる可能性は高くなる。

【０００６】

Ａ５３標準の開発プロセスでは、高精細度テレビジョン（ＨＤＴＶ）および固定受信に焦点を当てていた。このシステムは、既に市場に出回り始めていた大型の高解像度テレビジョン画面用の映像ビット・レートを最大限に高めるように設計されていた。ＡＴＳＣ　Ａ５３標準または旧来の符号化／伝送標準によって放送される伝送信号は、移動受信機では問題を生じる。

【０００７】

この点を踏まえ、ＡＴＳＣは、２００７年に、放送業者が自分のディジタル放送信号を使ってテレビジョン・コンテンツおよびテレビジョン・データを移動デバイスおよびハンドヘルド・デバイスに配信できるようにする標準を開発するプロセスを立ち上げることを発表した。旧来の伝送標準からの変更点としては、さらなるデータ冗長性を導入する追加符号化方式がある。この追加符号化は、旧来のＡ５３標準との後方互換性を維持しながら、移動デバイス、ハンドヘルド・デバイス、およびペデストリアン・デバイスの最新の受信機でより良好に実行されるように適応されている。その提案された変更により、既存の受信機器に悪影響を与えることなく、同じ無線周波（ＲＦ）チャネルで既存のＡＴＳＣサービスを動作させることも可能になっている。

【０００８】

しかし、現行のＡ５３標準の放送信号に導入された新たな符号化方式は何れも、ＡＴＳＣ　Ｍ／Ｈ受信機の復号プロセスをさらに複雑にするものである。例えば、旧来の動作で既に行われているトレリス復号およびリード・ソロモン復号に加えて、連結ブロック・コード化が追加されることにより、ブロック復号にしばしば使用される反復アポステリオリ型デコーダに関連する既知の順次入力ビット処理に基づく受信機における処理時間が大幅に増加することもある。復号処理の性能は、最適または所望の復号出力を得るのに必要な反復の回数と直接関係があることが多い。デコーダにおける反復回数、すなわち効率は、デコーダのアーキテクチャと当該デコーダ・アーキテクチャが入来メッセージ信号を処理できる速度の組合せによって制限されることになる。さらに、デコーダの処理時間を短縮するために使用される大きな参照テーブルを使用することにより、デコーダ・アーキテクチャの物理的サイズに関する非効率性が生じる。従って、復号アーキテクチャを改善することによって、復号プロセスおよびデコーダ効率を改善することが望ましい。

【発明の概要】

【０００９】

本発明の実施例の特徴によれば、バイト・コード符号化プロセスを用いて符号化された復調済みビットストリームを受信するステップと、前記復調済みビットストリームの一部分をビット・サブセットに配列するステップと、前記ビット・サブセットを並べ替えるステップと、前記並べ替えを行ったビット・サブセットおよび前記バイト・コード符号化プ

ロセスの性質に基づいて前記ビット・サブセットを復号するステップとを含む、ビットストリームを復号する方法について述べる。

【００１０】

本発明の実施例の別の特徴によれば、復調済みビットストリームを受信するステップと、前記復調済みビットストリームの一部分をビット・サブセットに配列するステップと、前記ビット・サブセット中のシンボルを特定するステップと、前記シンボルの以前の復号状態、および前記ビット・サブセット中の少なくとも２つのビットの間の関係に基づいて、前記シンボルの現在状態を復号するステップとを含む、ビットストリームを復号する方法について述べる。

【００１１】

本発明の実施例の別の特徴によれば、バイト・コード符号化プロセスを用いて符号化された復調済みビットストリームを受信する手段と、前記復調済みビットストリームの一部分をビット・サブセットにグループ化する手段と、前記ビット・サブセットを並べ替える手段と、前記並べ替えを行ったビット・サブセットおよび前記バイト・コード符号化プロセスの性質に基づいて前記ビット・サブセットを復号する手段とを備える装置について述べる。

【００１２】

本発明の実施例の別の特徴によれば、復調済みビットストリームを受信する手段と、前記復調済みビットストリームの一部分をビット・サブセットに配列する手段と、前記ビット・サブセット中のシンボルを特定する手段と、前記シンボルの以前の復号状態、および前記ビット・サブセット中の少なくとも２つのビットの間の関係に基づいて、前記シンボルの現在状態を復号する手段と、を備える装置について述べる。

【図面の簡単な説明】

【００１３】

【図１】本開示のエンコーダの実施例を示すブロック図である。

【図２】本開示の連結バイト・コード符号化回路の実施例を示すブロック図である。

【図３】本開示の受信機で使用されるデコーダの実施例を示すブロック図である。

【図４】本開示の連結バイト・コード復号回路の実施例を示すブロック図である。

【図５】本開示の要素バイト・コード・デコーダの実施例を示すブロック図である。

【図６】本開示の要素バイト・コード・デコーダの別の実施例を示すブロック図である。

【図７】本開示のデコーダのトレリス・ツリーの実施例を示す図である。

【図８】本開示のデコーダのトレリス・ツリーの別の実施例を示す図である。

【図９】本開示のデコーダのトレリス・ツリーのさらに別の実施例を示す図である。

【図１０】本開示のデコーダのトレリス・ツリーのまたさらに別の実施例を示す図である。

【図１１Ａ】本開示のデコーダのトレリス・ブロックの実施例を示すブロック図である。

【図１１Ｂ】本開示のデコーダのトレリス・ブロックの実施例を示すブロック図である。

【図１２Ａ】本開示のデコーダのアポステリオリ（ａ－ｐｏｓｔｅｒｉｏｒｉ）計算ブロックの実施例を示すブロック図である。

【図１２Ｂ】本開示のデコーダのアポステリオリ計算ブロックの実施例を示すブロック図である。

【図１３】本開示のバイト・コード・デコーダの実施例を示すブロック図である。

【図１４】本開示のバイト・コード・デコーダの別の実施例を示すブロック図である。

【図１５】本開示のデコーダのタナー・グラフの実施例を示す図である。

【図１６】本開示のバイト・コード・デコーダの別の実施例を示すブロック図である。

【図１７】本開示の復号プロセスの実施例を示す流れ図である。

【図１８】本開示の復号プロセスの別の実施例を示す流れ図である。

【００１４】

本開示の特徴および利点は、例示を目的として与える以下の説明を読めば、さらに明らかになるであろう。

【発明を実施するための形態】
【００１５】
以下、本開示の１つまたは複数の具体的な実施例を説明する。これらの実施例の説明を簡潔にするために、本明細書では、実際の実施態様の全ての特性について述べているわけではない。他のエンジニアリングまたは設計プロジェクトと同様に、これらの実際の実施態様の開発においても、システム関係の制約およびビジネス関係の制約の遵守など、実施態様によって変わる可能性のある開発者の目的を達成するためには、実施態様に特有の決定を多数行わなければならないことを理解されたい。それでも、こうした開発努力は、本開示の恩恵を受ける当業者にとっては、日常的な設計、作製、および製造業務であることも理解されたい。
【００１６】
以下は、テレビジョン放送信号、さらに具体的には、米国での使用のために規定された放送信号に関するシステムについて説明する。記載する実施例は、移動受信デバイス、ハンドヘルド受信デバイス、またはペデストリアン受信デバイスで使用することができる。使用されるデバイスの例としては、携帯電話機、インテリジェント電話機、携帯情報端末、ラップトップ・コンピュータ、携帯テレビジョン受信機などがあるが、これらに限定されるわけではない。その他のタイプの信号を送受信するために利用されるその他のシステムも、同様の構造およびプロセスを含むことがある。当業者なら、本明細書に記載する回路およびプロセスの実施例は、考えられる実施例をまとめた一例に過ぎないことを理解するであろう。一般に、Ａ５３標準などの放送および無線の標準に準拠する信号は、衛星リンク、同軸ケーブルまたは電話回線による伝送など、空気を介する以外の方法でも伝送することができることに留意されたい。従って、代替の実施例では、システムの構成要素を再配列する、または省略する、あるいは追加の構成要素を加えることもできる。例えば、小さな修正を加えることによって、記載のシステムを、世界の別の場所で使用されるサービスも含めて、その他の地上波放送サービス、衛星による映像および音声サービス、または電話データ・サービスに使用できるような構成にすることもできる。
【００１７】
以下に述べる実施例は、主として信号の送受信に関係している。特定の制御信号や電源接続など（ただしこれらに限定されない）、実施例の特定の特徴については、説明または図示をしていないが、当業者なら容易に突きとめることができる。これらの実施例は、マイクロプロセッサとプログラム・コードまたはカスタム集積回路を使用するなど、ハードウェアまたはソフトウェアあるいはその両者の任意の組合せを用いて実施することができることに留意されたい。また、これらの実施例の多くは、当該実施例の様々な要素間の動作および接続を反復することを含むことにも留意されたい。本明細書に記載の反復動作実施例の代わりに、またはそれに加えて、同一の要素を直列に接続して繰り返し利用するパイプライン・アーキテクチャを使用して、代替の実施例とすることができることもある。
【００１８】
図１は、エンコーダ１００の１実施例を示すブロック図である。エンコーダ１００は、Ａ５３伝送標準に関連するラギッド（ｒｕｇｇｅｄ）またはロバストなデータ・ストリームを符号化し、伝送するのに特に適している。エンコーダ１００は、ＭＰＥＧトランスポート・ストリーム・ソース１０２を備える。ＭＰＥＧトランスポート・ストリーム・ソース１０２は、いくつかの追加ブロックを含むＡＴＳＣ　Ｍ／Ｈブロック１１０に接続される。ＡＴＳＣ　Ｍ／Ｈブロック１１０に含まれるブロックは、入来データ・ストリームを処理し、移動デバイス、ペデストリアン・デバイス、およびハンドヘルド・デバイスでの受信および使用に適したラギッド・データ・ストリームを生成する。これらのブロックについては、後に詳細に述べる。ＡＴＳＣ　Ｍ／Ｈブロック１１０は、ｍｕｘ１３０に接続される。ｍｕｘ１３０は、旧来のＡＴＳＣ　Ａ５３のみの符号化で使用されるトランスポート・データ・コンテンツも受信する。ｍｕｘ１３０は、やはりいくつかの追加ブロックを含むＡＴＳＣ　Ａ５３レガシー・ブロック１５０に接続する。ＡＴＳＣ　Ａ５３レガシー・ブロック１５０内のブロックは、Ａ５３信号フォーマットの現行の放送信号を符号化

および伝送するために使用されるブロックを表すこともある。これらのブロックについても、後に詳細に述べる。制御装置１７０は、ｍｕｘ１３０およびＭＰＥＧトランスポート・ストリーム・ソース１０２に接続される。制御装置１７０は、エンコーダ１００内のその他のブロックに接続されることもある。

【００１９】

ＡＴＳＣ　Ｍ／Ｈブロック１１０内では、パケット・インタリーバ１１２が、パケット構成のデータ・ストリームを受信する。各パケットは、１８７バイトであり、パケット識別に使用される３バイトのヘッダを含む。パケット・インタリーバ１１２の出力は、ガロア体ＧＦ（２５６）直列連結ブロック・コーダ（ＳＣＢＣ）１１４に送られる。ＧＦ（２５６）ＳＣＢＣ１１４の出力部は、パケット・デインタリーバ１１６に接続される。パケット・デインタリーバ１１６の出力部は、トランスポート・ストリーム・ヘッダ修正器１１８に接続される。トランスポート・ストリーム・ヘッダ修正器１１８の出力部は、アプリオリ・トランスポート・パケット挿入器１２０に接続される。アプリオリ・トランスポート・パケット挿入器１２０の出力部は、ｍｕｘ１３０に接続される。

【００２０】

パケット・インタリーバ１１２は、通常は列として配列されるパケットの形で受信したデータを再配列して、各パケット列からバイトの縦列を作ってコードワードにする。パケット・インタリーバ１１２は、ある固定数の連続したパケットの各列から順番にバイトを取り出し、これらのバイトを縦列ごとに出力する。１実施例では、パケット・インタリーバ１１２は、１８７バイトのパケットを１２列読み込み、１２バイトのコードワードを１８７列出力する。パケット・インタリービングを行った結果として、全ての第１のバイトがグループ化され、全ての第２のバイトがグループ化され、以下同様にグループ化される。パケット・インタリーバ１１２に読み込まれただけのパケットがソース・フレームとなり、その数は、ＧＦ（２５６）ＳＣＢＣ１１４での処理に必要なソース・コードワードまたはシンボルの数と等しい。パケット・インタリーバ１１２の大きさは、使用されるメモリのタイプおよびサイズによって変わる可能性があることに留意されたい。

【００２１】

ＧＦ（２５６）ＳＣＢＣ１１４は、バイト・コード・エンコーダの１実施例である。特に、ＧＦ（２５６）ＳＣＢＣ１１４は、ガロア体（２５６）空間上で短い線形ブロック・コードを用いて実施される。使用することができる要素ブロック・コードはいくつかある。例えば、レート１／２バイト・コード・エンコーダは、以下の生成行列を使用することができる。

【００２２】

【数１】

$$G = (1 \quad 2) \qquad (1)$$

【００２３】

レート２／３バイト・コード・エンコーダは、以下の生成行列を使用することができる。

【００２４】

【数２】

$$G = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \end{pmatrix} \qquad (2)$$

【００２５】

この生成行列は、恒等行列と、上記の数式１および２の最後の列に値として示すｂ個の要素からなる列行列とを用いて形成される。その他のコード・レートを使用することもできる。

【００２６】

各要素コードの生成行列の係数は、ブロック・コード符号化と誤り訂正システム全体および変調プロセスとの関係に基づいて最適化されていることに留意されたい。その他の係数を使用することもできる。最適化では、特にＡＴＳＣ　８－ＶＳＢ変調におけるトレリス・コード化およびビット／シンボル・マッピングを考慮するが、これは、これらの特徴が、受信および復調プロセスで最初に遭遇する特徴であるからである。

【００２７】

ＧＦ（２５６）ＳＣＢＣ１１４は、単純なブロック・コード・エンコーダであっても、連結ブロック・コード・エンコーダであってもよい。連結ブロック・コード・エンコーダは、インタリービングやパンクチャリングなど、その他の機能を含むこともできる。連結ブロック・コード・エンコーダの１実施例については、以下でさらに詳細に述べる。ＧＦ（２５６）ＳＣＢＣ１１４は、複数の符号化レートで符号化を行うことができることもあり、また、レート・モード制御装置（図示せず）によって、レート・モードを切り換えることができることもある。好ましい実施例では、ＧＦ（２５６）ＳＣＢＣ１１４は、レート１／２要素コード、レート１２／２６連結コード、またはレート２４／２０８連結コードの１つを用いて入来データ・ストリームを符号化するように適合することもできる。

【００２８】

ＧＦ（２５６）ＳＣＢＣ１１４は、インタリーバ１１２から出力される列に沿ってバイトを符号化する。換言すれば、ＧＦ（２５６）ＳＣＢＣ１１４は、パケット・インタリーバ１１２の処理によってインタリーバ行列の第２の次元に従って符号化を行う。

【００２９】

パケット・デインタリーバ１１６は、ＧＦ（２５６）ＳＣＢＣ１１４が生成した符号化コードワード・ストリームを受信して、再構成した１８７バイト・パケットの列を出力する。パケット・デインタリーバ１１６は、オリジナル・メッセージまたはシステマティック・バイトおよびＧＦ（２５６）ＳＣＢＣ１１４の処理で追加された冗長または非システマティック・バイトの両方をそれぞれ含む列単位の順番で符号化コードワードを入力して、行単位の配列のバイトを出力する。このプロセスは、基本的には、パケット・インタリーバ１１２で述べたプロセスと逆のプロセスである。パケット・デインタリーバ１１６は、同じ列数のコードワードを入力し、各コードワードは、この時点では非システマティック・バイトの符号化セットを含む。出力における行の数は、符号化されたコードワードの長さに対応する。例えば、１２／２６コード・レートでは、２６行のパケットが出力されることになる。

【００３０】

ＭＰＥＧトランスポート・ストリーム・ヘッダ修正器１１８は、システマティック・パケットおよび非システマティック・パケットのグループを複数含むデインタリービングされた１８７バイト・パケットを受信し、各パケットに含まれる３バイト識別子ヘッダを修正する。この３バイトは、プログラム識別子（ＰＩＤ）、ならびに当該パケットに関する情報を搬送するために使用されるその他のいくつかのビットまたはビット・グループを含む。旧来の、またはＡ５３放送信号を受信することはできるがＡＴＳＣ　Ｍ／Ｈ符号化されたパケットの正しい復号および／または認識を行うことはできない受信機（例えば旧来の受信機）の最も効率的な動作を維持するために、ＡＴＳＣ　Ｍ／Ｈパケットの一部分のヘッダの特定のビットを修正してもよいことに留意されたい。

【００３１】

アプリオリ・トラッキング・パケット挿入器１２０は、ラギッド・データ・ストリーム中に所定のトラッキング・パケットを配置することができる。これらの所定のパケットとは、移動デバイス、ペデストリアン・デバイスまたはハンドヘルド・デバイスで使用される受信機など、ラギッド・データ・ストリームを受信することができる受信機が完全に、または大部分を知っている情報のパケットである。これらの所定のパケットは、受信機において、信号符号化および伝送の旧来または現行のＡ５３符号化部分の間に生じるトレリス状態の復号を補助するために使用される。これら所定のパケットは、受信機の等化器部分の収束も補助することができる。これら所定のパケットは、旧来の受信機の受信を改善

することを目的としたものではないが、潜在的に改善をもたらす可能性はあることに留意されたい。所定のトラッキング・パケットは、受信機に既知の疑似乱数生成器シーケンスなど、既知のトレーニング・シーケンス・プロセスを用いたいくつかの方法で生成することができる。

【００３２】

ＡＴＳＣ　Ｍ／Ｈブロック１１０で生成されるラギッドまたはロバストなデータ・ストリームは、ｍｕｘ１３０に供給される。ｍｕｘ１３０は、図示しないデータ・ソースから旧来のデータ・ストリームも受信する。実施例によっては、旧来のデータ・ストリームがＭＰＥＧトランスポート・ストリーム・ソース１０２から供給されることもあることに留意されたい。ｍｕｘ１３０は、入来するロバストＡＴＳＣ　Ｍ／Ｈデータ・ストリームおよび旧来のデータ・ストリームの一部を記憶するために、バッファ・メモリを備えることもできる。ｍｕｘ１３０は、ラギッドＡＴＳＣ　Ｍ／Ｈデータ・ストリームと旧来のデータ・ストリームの時間多重を生成し、この多重化ストリームをＡＴＳＣ　Ａ５３レガシー・ブロック１５０に供給する。

【００３３】

ｍｕｘ１３０は、制御装置１７０によって制御される。制御装置１７０は、マイクロプロセッサまたはマイクロコントローラとして埋め込まれた別個の回路とすることができる。あるいは、制御装置１７０は、ＭＰＥＧトランスポート・ストリーム・ソース１０２など、その他のブロックの１つに含めることもできる。また、制御装置１７０は、変調／伝送装置全体の動作を制御するために使用される別の制御装置に組み込むこともできる。

【００３４】

レガシーＡＴＳＣエンコーダ１５０は、多重化データ・ストリームとして与えられたラギッド・パケットおよびレガシー・パケットを、旧来のＡ５３標準に従って同じように符号化する。データ・ランダマイザ１５２は、パケットをランダム化し、当該パケットをリード・ソロモン・エンコーダ１５４に供給する。データのランダム化は、通常は、受信機に既知の乱数シーケンスでデータ・パケットを多重化することによって行われる。リード・ソロモン・エンコーダ１５４は、２０のパリティ・バイトを計算し、ランダム化データに連結して、２０７バイトを有するＲ－Ｓパケットを生成する。

【００３５】

畳み込みインタリーバ１５６は、データをさらに時間的にランダム化するために、Ｒ－Ｓパケットをインタリービングする。トレリス・エンコーダ１５８は、インタリービングされたパケットを符号化して、８２８個の３ビット・シンボルのブロックを生成する。Ａ５３標準では、１２個のトレリス・エンコーダを使用し、各トレリス・エンコーダは、インタリービングされたパケット中の２ビットごとに１つの３ビット・シンボルを生成する２／３レート・トレリス・エンコーダとすると指定している。その結果、トレリス・エンコーダ１５８は、デマルチプレクサ、１２個の並列の２／３レート・トレリス・エンコーダ、およびマルチプレクサを含む。畳み込みインタリーバ１５６のデータは、逆多重化されて１２個のトレリス・エンコーダに分配され、１２個のトレリス・エンコーダによって生成されたシンボルが多重化されて、シンボル・ストリームとなる。

【００３６】

同期挿入器１６０は、４つの既定のセグメント同期シンボルを、各８２８シンボル・ブロックの先頭に挿入して、８３２シンボル・セグメントを生成する。さらに、同期挿入器１６０は、生成される３１２個のセグメントごとに、８３２個のシンボルを含む同期フィールドを挿入する。詳細には、同期フィールドのシンボルは、３１２個のセグメントの前に位置する。

【００３７】

８－ＶＳＢ変調器１６２は、トレリス・エンコーダ１５８によって符号化されたデータ、セグメント付きシンボル、および同期フィールドを含む多重化シンボルを使用して、８－ＶＳＢ（残留側波帯）変調を用いて搬送波信号を変調する。詳細には、８－ＶＳＢ変調器１６２は、パルス振幅変調（ＰＡＭ）信号を生成する。ＰＡＭ信号の振幅は、特定の３

ビット・シンボルにそれぞれ対応する、８つの離散レベルのうちの１つとなる。ＰＡＭ信号は、図示しない回路を用いて、ディジタル信号フォーマットからアナログ信号フォーマットに変換され、正しい信号パルス形状を生成するためにフィルタリングされ、無線周波数にアップコンバートされる。

【００３８】

ｍｏｔｉｏｎ ｐｉｃｔｕｒｅ ｅｎｔｅｒｔａｉｎｍｅｎｔ ｇｒｏｕｐ（ＭＰＥＧ）トランスポート・ストリーム・ソース１０２やレガシー・コンテンツ用のソースなどの伝送ソースによって生成されたデータは、Ｉｎｔｅｒｎａｔｉｏｎａｌ Ｓｔａｎｄａｒｄｓ Ｏｒｇａｎｉｚａｔｉｏｎ／Ｉｎｔｅｒｎａｔｉｏｎａｌ Ｅｌｅｃｔｒｏｔｅｃｈｎｉｃａｌ Ｃｏｍｍｉｓｓｉｏｎ（ＩＳＯ／ＩＥＣ）１３８１８－２フォーマットと同等のＭＰＥＧ２フォーマットを用いて符号化されるソースである映像を含む。このデータは、Ｄｏｌｂｙ Ａｒｃ Ｃｏｎｓｉｓｔｅｎｃｙ ａｌｇｏｒｉｔｈｍ ＃３（ＡＣ－３）を用いて符号化されたソースである音声データも含む。Ａ５３標準では、プログラム・ガイド・データなど、その他のプログラム要素のメタデータを使用することもでき、これらのプログラム要素を、その他の方法を用いて符号化されたソースとすることもできる。さらに、Ａ５３標準では、標準精細度インタレース方式のテレビジョン画質や、順次走査方式の大画面高精細度テレビジョン画質など（ただしこれらに限定されない）、様々な映像品質レベルおよび表示フォーマットの映像を伝送することができる。

【００３９】

図２は、連結バイト・コード・エンコーダ２００の１実施例を示すブロック図である。連結バイト・コード・エンコーダ２００は、図１に示したＧＦ（２５６）ＳＣＢＣ１１４の代わりに使用することができ、１２／２６コード・レートを用いたデータ・ストリーム中のコードワードの符号化を可能にする。連結バイト・コード・エンコーダ２００は、パケットまたはコードワードを受信し、それらを第１の２／３レート・バイト・コード・エンコーダ２０２に供給する。第１の２／３レート・バイト・コード・エンコーダ２０２の出力は、インタリーバ２０４に供給される。インタリーバ２０４の出力は、第２の２／３レート・バイト・コード・エンコーダ２０６に供給される。第２の２／３レート・バイト・コード・エンコーダ２０６の出力は、バイト・パンクチャリング・ブロック２０８に供給される。パンクチャリング・ブロック２０８の出力は、データ・パケット化器２１０に供給される。データ・パケット化器２１０の出力は、さらなる処理（例えば図１で前述したレガシー伝送符号化など）に送ることもできる。

【００４０】

第１の２／３レート・バイト・コード・エンコーダ２０２は、コンテンツ・データ・パケットの１２バイトを受信し、その１２バイトから、第１のバイト・コード符号化ストリームを生成する。１２バイトのうちの２バイト、コンテンツ・データ・バイトM_AおよびM_Bごとに、第１のバイト・コード符号化ストリームは、バイトM_AおよびM_Bの複製と、数式２の生成行列を用いて計算した冗長バイトM_{AB}とを含む。実施例によっては、コンテンツ・データ・バイトM_AおよびM_Bは、データ生成器（例えば図１のデータ生成器１０２）によって生成される１つのコンテンツ・データ・パケットのバイトである。また、実施例によっては、第１の２／３レート・バイト・コード・エンコーダ２０２は、２つの異なるコンテンツ・データ・パケットＡおよびＢからコンテンツ・データ・バイトM_AおよびM_Bをそれぞれ選択する。コンテンツ・データ１２バイトごとに、１８バイトが、第１のバイト・コード符号化出力ストリームの一部として出力される。

【００４１】

第１のバイト・コード・エンコーダ２０２からのバイト・コード符号化ストリームを、インタリーバ２０４がインタリービングして、インタリーブされた１８バイトを含むインタリーブされたストリームを生成する。インタリーバ２０４、ならびに後述するその他のインタリーバは、当技術分野で既知の任意のインタリービング方法（例えば疑似乱数、行／列、コード最適化など）を使用することができる。さらに、インタリーバは、インタリーバのデータ長全体を記憶する記憶容量を有するメモリを備えることもできる。

【００４２】
インタリーブされたストリームは、第２の２／３レート・バイト・コード・エンコーダ２０６に供給される。第２の２／３レート・バイト・コード・エンコーダ２０６は、インタリーブされたストリーム中のインタリーブされた１８バイトのグループを符号化して、２７バイトのグループを含む第２のバイト・コード符号化ストリームを生成する。上述のように、インタリーバによって生成された２バイトM_AおよびM_Bごとに、第２の２／３レート・バイト・コード符号化ストリームは、２つのバイトM_AおよびM_Bの複製と、生成された冗長バイトM_{AB}とを有する。バイトM_Aは、データ生成器（例えば図１のデータ生成器１０２）によって生成されたコンテンツ・データのバイトの１つの複製であってもよいし、第１のバイト・コード・エンコーダ２０２によって冗長または非システマティック・バイトとして形成されたバイトであってもよいことは、明らかであろう。同様に、バイトM_Bも、コンテンツ・データのバイトの複製であってもよいし、第１のバイト・コード・エンコーダ２０２によって冗長または非システマティック・バイトとして形成されたバイトであってもよい。

【００４３】
バイト・パンクチャリング・ブロック２０８は、第２のバイト・コード符号化ストリーム中の２７バイトのグループから１バイトを除去して、２６バイトのグループを含むパンクチャリング済みストリームを生成する。バイト・パンクチャリングは、所与のコード化構造で供給され伝送されるバイト数を減少させることによってデータ効率を改善するために使用される。ただし、データ効率の改善と引き替えに、データ・ストリームから１つまたは複数の符号化バイトがなくなることにより、受信機の復号回路の性能が低下することになる。バイト・パンクチャリングを使用して、伝送フォーマットに好都合な符号化データのバイトまたはパケットのグループまたはブロックを生成することもできる。バイトまたはパケットの特定のグループ化に基づくコード化構造は、しばしばブロック・コードと呼ばれる。

【００４４】
パケット化器２１０は、パンクチャリング済みストリームのバイトを結合し、グループ化して、１８７バイトの離散パケットにする。バイト・コード・エンコーダ２００の構成要素によって生成されるラギッド・データ・ストリームは、１２／２６レート・データ・ストリームとなる。また、バイト・パンクチャリング・ブロック２０８を使用しない場合には、バイト・コード・エンコーダ２００は、１２／２７レート・データ・ストリームを生成することができる。

【００４５】
連結バイト・コード・エンコーダ６００に似た連結バイト・コード・エンコーダを利用して、上述の１２／２７レートおよび１２／２６レートのラギッド・データ・ストリーム以外のラギッド・データ・ストリームを生成することもできる。例えば、要素バイト・コード・エンコーダ、インタリーバ、およびパンクチャリング・ブロックを様々な組合せにすることにより、１７／２６レートや１２／５２レートなどのコード・レートを有するデータ・ストリームを生成することもできる。同様に、その他のタイプまたは配列のインタリーバまたはパンクチャリング・ブロックを、記載の実施例で使用したものの代わりに用いることもできる。

【００４６】
図３は、受信機で使用されるデコーダ３００の１実施例を示すブロック図である。デコーダ３００は、空気中の電磁波などの伝送媒体を介した信号の伝送によって悪影響を受けた信号を受信し、復号する回路および処理を含む。デコーダ３００は、ラギッド・データ・ストリームおよびレガシー・データ・ストリームの両方を復号することができる。例えば、デコーダ３００は、ＡＴＳＣ　Ｍ／Ｈ信号として伝送された信号を受信して復号することができる受信機に含めることができる。

【００４７】
デコーダ３００中では、入来信号は、初期処理の後で、等化器３１０に供給される。等

化器３１０は、トレリス・デコーダ３２０に接続され、トレリス・デコーダ３２０は、２つの出力を提供する。トレリス・デコーダ３２０の第１の出力は、フィードバックを提供するものであり、等化器３１０にフィードバック入力として接続される。トレリス・デコーダ３２０の第２の出力部は、畳み込みデインタリーバ３３０に接続される。畳み込みデインタリーバ３３０は、デランダマイザ（ｄｅ－ｒａｎｄｏｍｉｚｅｒ）３４０に接続される。デランダマイザ３４０の出力部は、バイト・コード・デコーダ３５０に接続され、このバイト・コード・デコーダ３５０も、２つの出力を提供する。バイト・コード・デコーダ３５０の第１の出力部は、畳み込みインタリーバ３６０およびランダマイザ３７０を介して、トレリス・デコーダ３２０にフィードバック入力として接続される。バイト・コード・デコーダ３５０の第２の出力部は、リード・ソロモン・デコーダ３８０に接続される。リード・ソロモン・デコーダ３８０の出力部は、データ・デコーダ３９０に接続される。ラギッド・ストリーム制御装置３９５は、バイト・コード・デコーダ３５０およびリード・ソロモン・デコーダ３８０に接続される。等化器３１０、トレリス・デコーダ３２０、畳み込みデインタリーバ、デランダマイザ３４０、リード・ソロモン・デコーダ３８０、およびデータ・デコーダ３９０は、ＡＴＳＣ　Ａ５３レガシー放送信号を受信するために使用される従来の受信機におけるそれらのブロックと同様に接続され、機能的に動作することに留意されたい。

【００４８】

受信機（図示せず）のフロントエンド処理（例えばアンテナ、チューナ、復調器、ＡＤ変換器）からの入力信号は、等化器３１０に供給される。等化器３１０は、受信信号を回復するために、受信信号を処理して伝送チャネル効果を完全または部分的に除去する。当業者には様々な除去または等価方法が周知であり、本明細書ではそれらについては述べない。等化器３１０は、フィードフォワード等化器（ＦＦＥ）部や判定帰還等化器（ＤＦＥ）部など、複数の処理回路部を含むことができる。等化器３１０は、同期除去、タイミング、およびアプリオリ・トレーニング・パケット処理用の回路を含むこともできる。

【００４９】

等化信号は、トレリス・デコーダ３２０に供給される。トレリス・デコーダ３２０は、１つの出力として、等化器３１０のＤＦＥ部に供給される判定値のセットを生成する。トレリス・デコーダ３２０は、やはり等化器３１０のＤＦＥ部に供給される中間判定値を生成することもできる。ＤＦＥ部は、トレリス・デコーダ３２０からの判定値ならびに中間判定値を使用して、等化器３１０のフィルタ・タップの値を調節する。調節済みのフィルタ・タップ値は、受信信号中の干渉および信号反射を打ち消すものである。この反復プロセスによって、等化器３１０は、トレリス・デコーダ３２０からのフィードバックも援用して、時間とともに変化する可能性のある信号伝送環境条件に対して動的に合わせることができる。反復プロセスは、ディジタル・テレビジョン放送信号の毎秒１９Ｍｂなど、信号の入来データ・レートと同様のレートで行うことができることに留意されたい。反復プロセスは、入来データ・レートより高いレートで行ってもよい。

【００５０】

トレリス・デコーダ３２０は、また、トレリス復号データ・ストリームを畳み込みデインタリーバ３３０に供給する。畳み込みデインタリーバ３３０は、エンコーダにおいて畳み込みバイト・インタリーバ１５６などによって実行されたインタリービングを逆にした方法で、データ・パケット内で構成されたデインタリービング済みのバイトを生成する。デインタリービング済みのパケットは、デランダマイザ３４０によってランダム化解除される。デランダマイザ３４０は、データ・ランダマイザ１５２などエンコーダのランダマイザで使用した複素数シーケンスの補数によってパケットを多重化することにより、エンコーダで付加されたランダム化コンテンツを除去する。ランダム化解除されたパケットは、バイト・コード・デコーダ３５０に供給される。バイト・コード・デコーダ３５０は、通常は、ラギッドまたはロバスト・データ・ストリームのパケットしか処理しない。そのため、ラギッド・データ・ストリームの一部でないパケットは、そのままバイト・コード・デコーダ３５０を通過して、リード・ソロモン・デコーダ３８０に渡される。デコーダ

３００の残りのブロックに供給されるトレリス復号ストリームは、データ・ストリームに含まれる情報の確率または信頼度の形態であってもよいことに留意されたい。

【００５１】

バイト・コード・デコーダ３５０およびトレリス・デコーダ３２０は、ターボ・デコーダと呼ばれる反復方式で動作して、ラギッド・データ・ストリームを復号する。詳細には、トレリス・デコーダ３２０は、デインタリービングおよびランダム化解除の後で、ラギッド・データ・ストリームに含まれるパケットの各バイトごとに、第１の軟判定ベクトルを、バイト・コード・デコーダ３５０に供給する。通常は、トレリス・デコーダ３２０は、確率値のベクトルとして軟判定を生成する。実施例によっては、ベクトル内の各確率値は、当該ベクトルと関連するバイトが有する可能性がある値と関連付けられる。また、実施例によっては、２／３レート・トレリス・デコーダは２ビット・シンボルを推定するので、システマティック・パケットに含まれる半ニブル（すなわち２ビット）ごとに確率値のベクトルを生成する。また、実施例によっては、トレリス・デコーダ３２０は、１バイトの４つの半ニブルに関連する４つの軟判定を組み合わせて、当該バイトが有する可能性のある値の確率のベクトルとなる１つの軟判定を生成する。これらの実施例では、当該バイトに対応する軟判定が、バイト・コード・デコーダ３５０に供給される。他の実施例では、バイト・コード・デコーダは、システマティック・パケットの１つのバイトに関連する軟判定を４つの軟判定ベクトルに分割し、４つの軟判定をそれぞれ当該バイトの半ニブルと関連付ける。

【００５２】

バイト・コード・デコーダ３５０は、ラギッド・データ・ストリームのパケットを構成するバイトに関連する軟判定ベクトルを使用して、当該パケットを構成するバイトの第１の推定値を生成する。バイト・コード・デコーダ３５０は、システマティック・パケットおよび非システマティック・パケットの両方を使用して、当該ラギッド・ストリームを構成するパケットの各バイトの第２の軟判定ベクトルを生成し、畳み込みインタリーバ３６０による再インタリービングおよびランダマイザ３７０による再ランダム化の後で、この第２の軟判定ベクトルをトレリス・デコーダ３２０に供給する。畳み込みインタリーバ３６０およびランダマイザ３７０は、図１に示した畳み込みバイト・インタリーバ１５６およびデータ・ランダマイザ１５２と同じように機能的に動作することに留意されたい。その後、トレリス・デコーダ３２０は、第２の軟判定ベクトルを使用して、第１の判定ベクトルの反復をさらに行い、これがバイト・コード・デコーダ３５０に供給される。トレリス・デコーダ３２０およびバイト・コード・デコーダ３５０は、トレリス・デコーダおよびバイト・コード・デコーダが生成した軟判定ベクトルが収束する、または所定回数の反復が行われるまで、このようにして反復を行う。その後、バイト・コード・デコーダ３５０は、システマティック・パケットの各バイトの軟判定ベクトルの確率値を使用して、システマティック・パケットの各バイトの硬判定を生成する。トレリス・デコーダ３２０は、最大アポステリオリ確率（ＭＡＰ）デコーダを用いて実施することができ、バイトの軟判定を基に動作することも、半ニブル（シンボル）の軟判定を基に動作することもできる。

【００５３】

ターボ復号では、通常、入来データ・レートより高い、ブロック間の判定データの受け渡しに関係する反復レートを利用することに留意されたい。可能な反復回数は、データ・レートと反復レートの比に制限される。その結果、実現可能な範囲で、ターボ・デコーダの反復レートが高い方が、一般に誤り訂正結果は良好である。１実施例では、入来データ・レートの８倍の反復レートを使用することができる。

【００５４】

図３に示すような軟入力軟出力バイト・コード・デコーダは、ベクトル復号機能を備えることができる。ベクトル復号では、システマティック・バイトおよび非システマティック・バイトを含めて、データのバイトをグループ化する。例えば、レート１／２バイト・コード符号化ストリームでは、１つのシステマティック・バイトと１つの非システマティ

ック・バイトをグループ化する。この２バイトで、６４０００を超える値を取り得る。ベクトル・デコーダは、この２バイトが取り得る値のそれぞれの確率を決定または推定して、確率マップを作成する。軟判定は、一部または全ての可能性の確率の重み付け、および可能なコードワードまでのユークリッド距離に基づいて行われる。硬判定は、ユークリッド距離の誤差がしきい値未満であるときに行うことができる。

【００５５】

バイト・コード・デコーダ３５０の硬出力（すなわちバイト・コード復号情報）は、リード・ソロモン・デコーダ３８０に供給される。リード・ソロモン・デコーダ３８０は、出力データを例えば２０７バイトのパケットの形態にする。リード・ソロモン・デコーダ３８０は、バイト・コード・デコーダ３５０によって生成される２０７バイトのシーケンスそれぞれを１つまたは複数のリード・ソロモン・コードワードと見なして、当該コードワードまたはパケット内の任意のバイトが伝送中のエラーによって破損しているかどうかを判定する。この判定は、当該コードワードの１組のシンドロームまたはエラー・パターンを計算し、評価することによって行うことが多い。破損が検出された場合には、リード・ソロモン・デコーダ３８０は、パリティ・バイト中に符号化された情報を使用して、破損したバイトの回復を試みる。その結果得られた誤り訂正データ・ストリームは、その後、伝送中のコンテンツのタイプに応じてデータ・ストリームを復号するデータ・デコーダ３９０に供給される。

【００５６】

データ・デコーダ３９０は、復号されたパケットのヘッダ中のＰＩＤなどの識別子を使用して、パケット内で搬送される情報のタイプ、およびその情報を復号する方法を決定する。ヘッダ中のＰＩＤは、データ・ストリームの一部として定期的に伝送し、受信機で更新することができるプログラム・マップ・テーブル（ＰＭＴ）中の情報と比較される。データ・デコーダ３９０は、認識されたタイプではないデータ・パケットのＰＩＤを有するパケットは全て無視する。

【００５７】

ラギッド・ストリーム制御装置３９５も、誤り訂正データ・ストリームを受信する。ラギッド・ストリーム制御装置３９５は、マイクロプロセッサまたはマイクロコントローラとして埋め込まれる別個の回路とすることができる。あるいは、ラギッド・ストリーム制御装置３９５は、バイト・コード・デコーダ３５０など、その他のブロックの１つに含めることもできる。また、ラギッド・ストリーム制御装置３９５は、受信装置全体の動作を制御するために使用される制御装置に組み込むこともできる。ラギッド・ストリーム制御装置３９５は、例えばラギッド・データ・ストリームに使用されるプリアンブルの形態のアプリオリ・トランスポート・パケットが存在するかどうかを判定することができる。アプリオリ・トランスポート・パケットが存在していれば、ラギッド・ストリーム制御装置３９５は、ラギッド・データ・ストリーム中の制御情報を識別し、復号する。

【００５８】

図３に示すようなデコーダは、単純なバイト・コード・エンコーダによる符号化も連結バイト・コード・エンコーダによる符号化も含めて、前述のバイト・コード・エンコーダによって符号化されているラギッド・データ・ストリームを復号することができる。図３のデコーダは、符号化ステップを１度しか行わない単純なバイト・コード・エンコーダまたは要素バイト・コード・エンコーダによって符号化されたラギッド・データ・ストリームを復号するものとして示してある。連結バイト・コード復号では、デインタリービング、デパンクチャリング（ｄｅ－ｐｕｎｃｔｕｒｉｎｇ）、再挿入などの中間処理に加えて、入来コードワードまたはバイトを複数の復号ステップで復号をすることを含む。

【００５９】

図４は、連結バイト・コード・デコーダ４００の１実施例を示すブロック図である。連結バイト・コード・デコーダ４００は、図３のバイト・コード・デコーダ３５０の代わりに使用することができる。連結バイト・コード・デコーダ４００は、ターボ・デコーダ構成で動作するように構成することができる。連結バイト・コード・デコーダ４００は、そ

の内部で、反復プロセスを使用してラギッド・データ・ストリーム中の連結バイト・コード符号化パケットを復号するターボ・デコーダとして動作する。連結バイト・コード・デコーダ４００は、レート１２／２６バイト・コード符号化信号ストリームを復号して、最初に符号化された２６バイトから１２バイトのデータを生成するように適合される。

【００６０】

通常は２６バイトの軟判定値を表す入来メッセージ・データ・ストリームは、バイト挿入ブロック４０２に供給される。バイト挿入ブロック４０２の出力部は、第１の２／３レート・バイト・コード・デコーダ４０４に接続される。第１の２／３レート・バイト・コード・デコーダ４０４は、２つの出力を提供する。第１の出力部は、パンクチャリング・ブロック４０６に接続され、パンクチャリング・ブロック４０６の出力部は、インタリーバを介してトレリス・デコーダにフィードバック入力として接続される。第１の２／３レート・バイト・コード・デコーダ４０４の第２の出力部は、デインタリーバ４０８に接続される。シンボル・デインタリーバ４０８の出力部は、第２の２／３レート・デコーダ４１０に接続され、第２の２／３レート・デコーダ４１０は、やはり２つの出力部を有する。第１の出力部は、インタリーバ４１２を介して第１の２／３レート・バイト・コード・デコーダ４０４にフィードバック入力として接続される。第２の出力部は、リード・ソロモン・デコーダなど、その他の処理ブロックに接続される。

【００６１】

バイト挿入ブロック４０２への２６バイト入力は、トレリス・デコーダによって生成される、データのシステマティック・バイトまたはシステマティック・パケットに関する第１の軟判定と、データの非システマティック・バイトまたは非システマティック・パケットに関する軟判定とを含む。データのシステマティック・バイトおよび非システマティック・バイトは、バイト・コード符号化されたパケットのバイトとすることができる。２／３レート・バイト・コード・デコーダは、２データ・バイトを復号するのに３バイト必要とする。しかし、元の連結符号化では、１バイト、好ましくは非システマティック・バイトを除去して、２７バイトから２６バイトにコードワードを短縮している。その結果として、符号化プロセスのパンクチャリングで除去されたバイトの代わりとなる１バイトが必要となる。さらに、トレリス・デコーダへの入力ストリームは、パンクチャリングされたバイトは含まないので、トレリス・デコーダは、データ・ストリーム中のパンクチャリングされたバイトに関するいかなる軟判定も生成しない。その結果として、パンクチャリングされたバイトの値の確率が等しいことを示す軟判定値が挿入される。バイト挿入ブロック４０２で挿入された軟判定値を含む第１の軟判定は、第１の２／３レート・バイト・コード・デコーダ４０４に供給される。第１の２／３レート・バイト・コード・デコーダ４０４は、第１の軟判定を使用して、システマティック・パケットおよび非システマティック・パケットのバイトの復号に基づく第２の軟判定を生成する。軟判定の生成では、例えば、１組のバイト・セットに、上記の数式（２）および（３）に示すバイト・コード符号化パケットを形成するために使用したb_1およびb_2要素の値の逆数をかける乗算を利用する。

【００６２】

第１の２／３レート・バイト・コード・デコーダの２７バイト軟出力は、パンクチャリング・ブロック４０６に供給される。この２７バイト軟出力は、第１の２／３レート・バイト・コード・デコーダで復号が行われた後の、システマティック・バイトおよび非システマティック・バイトの両方の軟判定値の更新済みセットを表す。パンクチャリング・ブロック４０６は、バイト・フォーマットをトレリス・デコーダが最初に処理した２６バイト・フォーマットに戻すために、以前に挿入された軟判定バイトを除去する。

【００６３】

システマティック・バイトのみを表す第１の２／３レート・バイト・コード・デコーダの１８バイト軟出力は、デインタリーバ４０８に供給される。デインタリーバ４０８は、２／３レート・バイト・コード符号化プロセスで行ったインタリービングを逆にした方法で、この１８バイトのデータをデインタリービングする。デインタリーバ４０８は、エン

コーダのインタリービング・マップを正確に反転する。デインタリービングされたバイトは、第２の２／３レート・バイト・コード・デコーダ４１０に供給される。第２の２／３レートバイト・コード・デコーダ４１０は、デインタリービングされた軟判定システマティック・バイトを使用して、上述したのと同様の方法で、軟判定バイトの２つの追加出力を生成する。１８バイト軟出力は、インタリーバ４１２に供給される。この１８バイト軟出力は、第１の２／３レートバイト・コード・デコーダ４０４における復号によって得られたシステマティック・バイトおよび非システマティック・バイトの両方の軟判定値の更新済みセットを含むフィードバック・メッセージを表す。インタリーバ４１２は、デインタリービング済みのバイトを第１の２／３レートバイト・コード・デコーダで使用したバイト・フォーマット中に配置するために、このデインタリービング済みのバイトを再インタリービングする。インタリーバ４１２は、図１のインタリーバ１０４など、エンコーダで使用するインタリーバと基本的には同じであり、１８バイトの再インタリービング済みのセットを第１の２／３レートバイト・コード・デコーダ４０４に供給する。１８バイトの再インタリービング済みのセットは、第１の２／３レートバイト・コード・デコーダ４０４が行う軟判定を改善するために使用される。

【００６４】

第２の２／３レートバイト・コード・デコーダ４１０の１２バイト出力は、システマティック・バイトを、１２／２６レート・バイト・コード符号化されたラギッド・データ・ストリームの完全に復号されたデータ出力として表す。好ましい実施例では、第２の２／３レートバイト・コード・デコーダ４１０によって生成された１２個のシステマティック出力バイトの軟判定が決定的なものである、または正確なデータ値として決定的であるとする所定のしきい値以内である場合には、第２の２／３レート・バイト・コード・デコーダ４１０は、これらの軟判定を使用して、これら１２個の出力バイトに関する硬判定を生成し、これら１２個の出力バイトを、リード・ソロモン・デコーダなど後続の処理ブロックに供給する。しかし、第２の２／３レートバイト・コード・デコーダによって生成された軟判定が決定的なものでない場合には、直前の反復中に形成され、フィードバックされた軟情報を用いて、上記と同様にさらなる反復を行う。この追加の軟情報は、各軟デコーダに対し、その次のデコーダから供給される。すなわち、トレリス・デコーダは、パンクチャリング・ブロック４０６を介して供給される第１の２／３レート・バイト・コード・デコーダ４０４からのフィードバック・メッセージ信号を使用し、第１の２／３レートバイト・コード・デコーダ４０４は、インタリーバ４１２を介して供給される第２の２／３レートバイト・コード・デコーダ４１０からのフィードバック・メッセージ信号を使用する。このような反復は、第２の２／３レートバイト・コード・デコーダ４１０が生成する軟判定が十分に収束するまで、または所定回数の反復が行われるまで続く。ターボ復号では、通常は、入来データ・レートより高い、ブロック間の判定データの受け渡しに関係する反復レートを利用する。

【００６５】

図５は、要素バイト・コード・デコーダ５００の１実施例を示すブロック図である。要素バイト・コード・デコーダ５００は、図３に示すバイト・コード・デコーダ３５０として使用することも、図４に示すデコーダ４０４および４１０として使用することもできる。バイト・コード・デコーダ５００は、２つの入力信号を受信する。第１の入力信号は、主にトレリス復号ブロックからの入来メッセージ信号である。第２の入力信号は、後続の復号ステージの出力からフィードバックされたフィードバック・メッセージ信号である。各入力部は、加算器５１０に接続される。加算器５１０の出力は、コア・ユニット５２０に供給される。コア・ユニット５２０内部で、加算器５１０の出力部は、エクストリンシック（ｅｘｔｒｉｎｓｉｃ）計算ブロック５２２および加算器５２４の一方の入力部に接続される。エクストリンシック計算ブロック５２２の出力部は、加算器５２４の第２の入力部に接続される。加算器５２４の出力は、コア・ユニット５２０の出力を表し、加算器５３０および加算器５４０の一方の入力となる。オリジナルの入力メッセージの反転信号が、加算器５３０の第２の入力部に供給される。フィードバック・メッセージの反転信号

が、加算器５４０の第２の入力部に供給される。加算器５３０および加算器５４０の出力は、要素デコーダ５００の出力を表すものであり、図４のデコーダ４０４およびデコーダ４１０について述べた出力と同様である。

【００６６】

加算器５１０は、入来コード化メッセージ信号を未コード化フィードバック・メッセージと結合する。加算機能は、通常は、特定のビット、より詳細には、メッセージ信号中の各ビットの特定のビット信頼度に基づいて信号を結合するものである。コア・ユニット５２０は、結合メッセージ信号を受信し、エクストリンシック計算ブロック５２２において、加算器５１０から供給されるイントリンシック（ｉｎｔｒｉｎｓｉｃ）信頼度情報を用いて入来ビットについてエクストリンシック情報を計算する。１実施例では、各入来ビット信頼度は、上述のように、ユークリッド距離技術および生成行列の逆転値を用いて処理する。エクストリンシック計算ユニット５２２で計算されるエクストリンシック情報を、加算器５２４が、加算器５１０から入力されるイントリンシック情報と結合または加算して、アポステリオリ情報を生成する。コア・ユニット５２０によって生成されたアポステリオリ情報は、オリジナルの信号メッセージ、フィードバック・メッセージ、および計算したエクストリンシック情報の組合せまたは合計を表す。

【００６７】

加算器５４０は、減算プロセスによって、入力部から供給されたフィードバック・メッセージ信号をアポステリオリ情報と結合する。好ましい実施例では、フィードバック・メッセージ信号は、加算器５４０に入力される前に反転される。その結果得られるメッセージ信号は、入力部からのオリジナルのコード化メッセージと、コア・ユニット５２０で計算され付加されたエクストリンシック情報とを表す。情報メッセージ信号は、通常は、メッセージ中のシステマティックおよび非システマティックな情報の確率または信頼度のセットとして、要素バイト・コード・デコーダ５００との間で搬送されることに留意されたい。１つの好ましい実施例では、エクストリンシック計算ブロック５２２は、対数計算を実行し、その結果得られる加算器５４０の出力における信頼度は、対数尤度比で表される。入来メッセージが、対数尤度比のセットの形態で受信されることもある。メッセージ中のビットに基づく出力メッセージの信頼度は、連結バイト・コード・デコーダ中の後続のステージに供給されるか、あるいは、反復バイト・コード復号プロセスの最終出力を表すものとして、さらに別のデコーダに供給される。

【００６８】

同様に、加算器５３０は、減算プロセスによって、入力メッセージ信号をアポステリオリ情報と結合する。好ましい実施例では、入力メッセージ信号は、加算器５３０に入力される前に反転される。その結果得られる信号は、フィードバック・メッセージ信号と、付加されたエクストリンシック情報とを表す。上記と同様に、情報は、確率または信頼度のセットとして搬送される。コード化メッセージの信頼度のセットは、連接デコーダ内のそれより前のバイト・コード・デコーダにフィードバック信号として供給されるか、あるいは、反復復号をさらに改善するためにトレリス・デコーダなどのそれより前の復号ブロックにフィードバック信号として供給される。

【００６９】

要素バイト・コード・デコーダ５００は、ビットごと、半ニブルごと、またはバイトごとに適用されるバイト・コード符号化に基づいて、入力メッセージ信号を反復処理することができる。前述のように、反復の回数は、ビット値の信頼度を改善することを考慮したものである。反復プロセスの数は、ハードウェアのサイズまたはハードウェアの速度によって制限されることがある。従って、アーキテクチャおよびプロセスはより効率的であることが望ましい。上述の逐次直接復号プロセスの代わりにガロア体空間に関連した性質を利用すれば、さらに改良した復号アーキテクチャを利用することができる。要素バイト・コードは、ガロア体ＧＦ（２５６）上で定義された短い線形ブロック・コードであることに留意されたい。ガロア体ＧＦ（２５６）は、ＧＦ（２）の拡大体であり、原始多項式によってさらに定義することができる。１つの好ましい原始多項式は、以下のようなもので

ある。

【００７０】

【数３】

$$P(x) = p_0 + p_1 x + .. + p_8 x^8 = 1 + x^2 + x^3 + x^4 + x^8, \text{ with } p_i \in GF(2). \quad (3)$$

【００７１】

伝送されるコード化メッセージは、以下のように定義することができる。

【００７２】

【数４】

$$c = mG, \quad (4)$$

ここで、Ｇは、ＧＦ（２５６）では、コード・レート１／２および２／３について数式（１）および（２）で記述したように定義することができる。ただし、生成行列は、ＧＦ（２）では、コード・レート１／２については、原始多項式に基づいて以下のように再定義することができる。

【００７３】

【数５】

$$G = [I \quad S] \quad (5)$$

【００７４】

数式５において、Ｉは、８×８の大きさの恒等行列であり、Ｓは、以下のようにＧＦ（２）で定義される部分行列である。

【００７５】

【数６】

$$S = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \quad (6)$$

【００７６】

同様に、コード・レート２／３については、生成行列は、ＧＦ（２）では原始多項式に基づいて以下のように定義することができる。

【００７７】

【数７】

$$G = \begin{bmatrix} I & 0 & S \\ 0 & I & S \end{bmatrix} \quad (7)$$

【００７８】

上記に定義した行列方程式は、ＧＦ（２）内のパリティ・チェック行列の生成をさらに可能にして、コード化メッセージからオリジナルのメッセージを回復することができることもある。パリティ・チェック式は、以下のように書くことができる。

【００７９】

【数８】

$$cH^T = 0 \qquad (8)$$

【００８０】

コード・レート１／２のパリティ・チェック行列は、以下のように定義することができる。

【００８１】

【数９】

$$H = \left[S^T \quad I \right] \qquad (9)$$

【００８２】

従って、Ｈは、以下のように行列形式で書くことができる。

【００８３】

【数１０】

$$H_{C=1/2} = \begin{bmatrix} 0&1&0&0&0&0&0&0 & 1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 & 0&1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 & 0&0&1&0&0&0&0&0 \\ 1&0&0&0&1&0&0&0 & 0&0&0&1&0&0&0&0 \\ 1&0&0&0&0&1&0&0 & 0&0&0&0&1&0&0&0 \\ 1&0&0&0&0&0&1&0 & 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1 & 0&0&0&0&0&0&1&0 \\ 1&0&0&0&0&0&0&0 & 0&0&0&0&0&0&0&1 \end{bmatrix} \qquad (10)$$

【００８４】

同様に、コード・レート２／３のパリティ・チェック行列は、以下のように定義することができる。

【００８５】

【数１１】

$$H = \left[S^T \quad S^T \quad I \right] \qquad (11)$$

【００８６】

従って、コード・レート２／３のＨは、以下のように書くことができる。

【００８７】

【数１２】

$$H_{C=2/3} = \left[\begin{array}{cccccccc|cccccccc|cccccccc} 0&1&0&0&0&0&0&0 & 0&1&0&0&0&0&0&0 & 1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 & 0&0&1&0&0&0&0&0 & 0&1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 & 0&0&0&1&0&0&0&0 & 0&0&1&0&0&0&0&0 \\ 1&0&0&0&1&0&0&0 & 1&0&0&0&1&0&0&0 & 0&0&0&1&0&0&0&0 \\ 1&0&0&0&0&1&0&0 & 1&0&0&0&0&1&0&0 & 0&0&0&0&1&0&0&0 \\ 1&0&0&0&0&0&1&0 & 1&0&0&0&0&0&1&0 & 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1 & 0&0&0&0&0&0&0&1 & 0&0&0&0&0&0&1&0 \\ 1&0&0&0&0&0&0&0 & 1&0&0&0&0&0&0&0 & 0&0&0&0&0&0&0&1 \end{array}\right] \qquad (12)$$

【００８８】

パリティ・チェック行列は、行列式による処理のためにビットを配列またはグループ化

することにより、受信機で直接、入来バイト・コード・コード化メッセージに適用することができる。さらに、パリティ・チェック行列は、パリティ・チェック方程式としばしば呼ばれる１組の方程式のセットに変形することができる。各方程式は、入来メッセージのうちの特定のビットしか含まない。これらの方程式は、メッセージのグループまたはサブセット中のビットの最終的な値または尤度を求めるために、１組の連立方程式として処理し、解くことができる。入来ビットを配列してパリティ・チェック方程式またはパリティ・チェック行列にするデコーダ、特にバイト・コード・デコーダについては、以下で述べる実施例によりさらに明らかになるであろう。

【００８９】

図６は、本開示の複数の特徴を使用する要素バイト・コード・デコーダ６００の別の実施例を示すブロック図である。要素バイト・コード・デコーダ６００は、図４のデコーダ４０４およびデコーダ４１０の代わりに使用することができる。２つの入力信号、すなわち入来メッセージ信号およびフィードバック・メッセージ信号が、入力として加算器６０５に供給される。加算器６０５の出力部は、入力ビット並べ替えブロック６１５に接続される。入力並べ替えブロック６１５の第１の出力部は、遅延バッファ６２０およびアルファ・トレリス・ブロック６２５に接続される。入力並べ替えブロック６１５の第２の出力部は、ベータ・トレリス・ブロック６３０に接続される。アルファ・トレリス・ブロック６２５の出力部は、メモリ・バッファ６３５に接続される。ベータ・トレリス・ブロック６３０の出力部は、メモリ・バッファ６４０に接続される。遅延バッファ６２０、メモリ・バッファ６３５、およびメモリ・バッファ６４０の出力部は全て、アポステリオリ・プロセッサ６４５に接続される。アポステリオリ・プロセッサ６４５の出力部は、出力並べ替えブロック６５０に接続される。出力並べ替えブロック６５０の出力部は、加算器６５５および加算器６６０それぞれの一方の入力部に接続される。コード化メッセージ信号は、遅延バッファ６１０にも接続される。遅延バッファ６１０の出力部は、反転され、加算器６５５のもう一方の入力部に接続される。フィードバック・メッセージ信号は、遅延バッファ６１２にも接続される。遅延バッファ６１２の出力部は、反転され、加算器６６０のもう一方の入力部に接続される。図５の要素バイト・コード・デコーダ５００について述べたのと同様に、加算器６５５および加算器６６０の出力は、要素デコーダ６００の出力を表す。

【００９０】

加算器６０５は、２つの入力、すなわち入力メッセージ信号およびフィードバック・メッセージ信号を受信し、これらの信号の信頼度を結合または加算する。入力並べ替えブロック６１５は、結合されたメッセージ信号をビット・サブセットにグループ化し、信号中のビットの順序を変更し、並べ替えた入力信号をアルファ・トレリス・ブロック６２５およびベータ・トレリス・ブロック６３０に供給する。並べ替えは、デコーダの効率を改善するために行うこともでき、後に詳細に述べる。選択されるサブセットのサイズは、用いるパリティ・チェックプロセスによって決まることがある。好ましい実施例では、サブセット・サイズは１６ビットであり、ベクトル$b=[b_0 b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 b_9 b_{10} b_{11} b_{12} b_{13} b_{14} b_{15}]$で特定される。入力並べ替えブロック６１５は、最初の並べ替えビットから最後の並べ替えビットまでの、順方向並べ替え出力を生成し、これがアルファ・トレリス・ブロック６２５およびバッファ６２０に供給される。バッファ６２０は、イントリンシック信号も表す順方向並べ替え出力のタイミング遅延を行なって、トレリス処理の後で再結合することができるようにする。入力並べ替えブロック６１５は、最後の並べ替えビットから最初の並べ替えビットまでの、逆順序信号も生成し、この信号をベータ・トレリス・ブロック６３０に供給する。

【００９１】

アルファ・トレリス・ブロック６２５およびベータ・トレリス・ブロック６３０は、トレリス・ツリーまたはトレリス図と呼ばれる１組の計算の生成処理に基づいて、順方向メトリック値および逆方向メトリック値を計算する。トレリス・ツリーまたはトレリス図の詳細な実施態様およびアルファ・トレリス・ブロック６２５およびベータ・トレリス・ブ

ロック６３０の動作については、後に詳細に述べる。順方向バッファ６３５および逆方向バッファ６４０は、アルファ・トレリス・ブロック６２５およびベータ・トレリス・ブロック６３０の出力を受信する。順方向バッファ６３５および逆方向バッファ６４０は、再結合のタイミング遅延を供給する。さらに、逆方向バッファ６４０は、ベータ・トレリス・ブロック６３０の出力のビット順序を逆転させる。

【００９２】

アルファ・トレリス・ブロック６２５およびベータ・トレリス・ブロック６３０からの順方向メトリック値および逆方向メトリック値は、アポステリオリ・ブロック６４５で、イントリンシック情報と結合される。アポステリオリ・ブロック６４５は、メッセージ・ストリーム中のビット・サブセットのアポステリオリ信頼度を計算する。出力並べ替えブロック６５０は、サブセットを元の順序に戻すために、サブセットのアポステリオリ信頼度情報を並べ替える。加算器６５５および６６０は、それぞれ減算のために反転されたフィードバック・メッセージと入力メッセージを、並べ替え済みのアポステリオリ・メッセージ信号と結合する。バッファ６１０および６１２は、出力されるアポステリオリ信頼度情報と適切に結合できるようにするために、入力メッセージおよびフィードバック・メッセージの必要なタイミング遅延を供給する。加算器６６０のフォワード出力および加算器６５５のコード化フィードバック出力は、フォワード出力値およびフィードバック出力値を表しており、図５の要素バイト・コード・デコーダ５００について述べた出力と機能的には同じであることに留意されたい。

【００９３】

要素バイト・コード・デコーダ６００は、ログ・マップ・デコーダを用い、メッセージ中の並べ替え済みビット・サブセットのトレリス復号に基づいて、入来メッセージを復号する。１つの可能なトレリス・アルゴリズムでは、ＢＣＪＲ（Ｂａｈｌ－Ｃｏｃｋｅ－Ｊｅｌｉｎｅｋ－Ｒａｖｉｖ）アルゴリズムを利用する。トレリス・マッピングを用いたログ・マップ復号を適用するために、前述のパリティ行列から形成した１組のパリティ方程式に基づいて、入来メッセージを等価なビット・トレリス・ツリーに変換する。コード・レート１／２では、番号をふった以下の方程式の組を使用することができる。

【００９４】

【数１３】

(1) $b_1+b_8=0$
(2) $b_2+b_9=0$
(3) $b_3+b_{10}=0$
(4) $b_0+b_4+b_{11}=0$
(5) $b_0+b_5+b_{12}=0$
(6) $b_0+b_6+b_{13}=0$
(7) $b_7+b_{14}=0$
(8) $b_0+b_{15}=0$

【００９５】

ビット・ベースのトレリス・アルゴリズムでは、形成したトレリス・ツリーを左から右に見ていく深さが増していく任意の特定の順序でビットを処理することができる。所与の深さでビットが選択されると、当該ビットを含む全てのパリティ・チェック方程式がアクティブになる。所与のパリティ・チェック方程式の全てのビットが処理されると、当該パリティ・チェック方程式はアクティブではなくなる。所与の深さにおけるパリティ・チェック方程式の現在状態を連結して、ｎビット状態を形成する。ここで、ｎは、アクティブなパリティ・チェック方程式の現在数である。アクティブなパリティ・チェック方程式の一部が理想的な状態にある可能性があるので、これらの方程式に対して１ビットだけでよい。全ての深さにわたるときの方程式の最大数ｎが可能な限り小さくなるようにビットを処理するビット順序を用いることで、効率的な復号が可能になる。好ましいビット・トレリス・ツリーを、図７に示す。

【００９６】
図７において、下側のｘ軸は、入力ビット並べ替えブロック６１５で実装されるビット順序を示す。順方向順序は、この軸上の左から右に示す順序であり、逆方向順序は、右から左に示す順序である。上側のｘ軸は、各ビット遷移または状態変化において使用される方程式の数を示す。実線は、次のビットの値または確率値が０であるときの、直前のビットから次のビットへの遷移を示す。破線は、次のビットの値または確率値が１であるときの、直前のビットから次のビットへの遷移を示す。
【００９７】
コード・レート２／３のビット・トレリス・ツリーも、サブセット・サイズを２４ビットとして、以下の番号付きパリティ・チェック方程式のセットを用いて、同様に形成することができる。
【００９８】
【数１４】

$$(1)\ b_1+b_9+b_{16}=0$$
$$(2)\ b_2+b_{10}+b_{17}=0$$
$$(3)\ b_3+b_{11}+b_{18}=0$$
$$(4)\ b_0+b_4+b_8+b_{12}+b_{19}=0$$
$$(5)\ b_0+b_5+b_8+b_{13}+b_{20}=0$$
$$(6)\ b_0+b_6+b_8+b_{14}+b_{21}=0$$
$$(7)\ b_7+b_{15}+b_{22}=0$$
$$(8)\ b_0+b_8+b_{23}=0$$

【００９９】
コード・レート２／３の好ましいトレリス・ツリーを、図８に示す。これは、図７に示すトレリス・ツリーと同様に説明することができる。
【０１００】
上述の復号プロセスに対して、代替のパリティ・チェック配列を使用することもできることに留意されたい。例えば、さらに小さなビット・サブセットを使用することもできる。その結果として、さらに小さなビット・サブセットであれば、それらのビットのうちの２つの間の関係を１つ使用するだけで復号することができる可能性もある。小さなビット・サブセットとパリティ関係の形態の単一の関係とを繰り返し復号してもよいし、サブセットにビットを加えたりサブセットからビットを除去したりして、単一の関係として別のパリティ方程式を後の復号ステップにおいて使用してもよい。
【０１０１】
要素バイト・コード・デコーダ６００では、その他の形でビットをグループ化することができることもある。例えば、１６ビットまたは２４ビットのサブセットを形成するグループ化に加えて、受信した伝送または変調シンボルとビットとの関係に基づいてビットをグループ化することによって第２のグループを形成することもできる。この追加のグループ化により、入来メッセージ中のビットに関連するイントリンシック情報が追加されるので、復号プロセスはさらに改善される。要素バイト・コード・デコーダは、メッセージの２ビットを含み、さらに４つの可能な値を表す各シンボルのコンスタレーション（ｃｏｎｓｔｅｌｌａｔｉｏｎ）値に基づいてメッセージ情報を受信し、処理する。１つのシンボル内の各ビットのコンスタレーション位置は、１つの信頼度値によって表される。従って、２ビットからなる１シンボルは、以下のように特定される４つの確率として表すことができる。
【０１０２】
【数１５】

$$P\left(x_{symbol}="00"\right),\ P\left(x_{symbol}="01"\right),\ P\left(x_{symbol}="10"\right),\ P\left(x_{symbol}="11"\right)$$

【０１０３】
ビットとシンボルの関係付けおよびマッピング、ならびに後のシンボル・メトリック値生成は、図３のトレリス・デコーダ３２０など、以前の処理ステージで行うことができることに留意されたい。シンボル・メトリック値およびシンボル・マップは、次いで、要素バイト・コード・デコーダ６００の入力部に供給される。ビット・シンボル・マッピングおよびメトリック値生成は、加算器ブロック６０５や入力並べ替えブロック６１０などにおいてバイト・コード・デコーダ６００の入力処理の一部として実行することもできるし、図示しない別個の処理ブロックにおいて実行することもできる。さらに、上述したメッセージにおけるビット確率と同様に、シンボル確率も、バイト・コード・デコーダ６００で維持し、処理し、そこから出力することができる。

【０１０４】
対応するシンボル・トレリスは、中間遷移を考慮して同一シンボルのビットの深さを単一の深さにまとめることによって形成される。シンボルs_nの２進表現が「$b_{2n}b_{2n+1}$」であるとすると、例えば、シンボルs_0＝３＝「１１」は、状態１を経由する状態０から状態３への遷移を引き起こし、シンボルs_0＝２＝「１０」は、状態１を経由する状態０から状態２への遷移を引き起こし、シンボルs_0＝１＝「０１」は、状態０を経由する状態０から状態１への遷移を引き起こし、シンボルs_0＝０＝「００」は、状態０を経由する状態０から状態０への遷移を引き起こす。このパターンが、全ての可能な状態のシンボル・トレリス全体が形成されるまで続く。

【０１０５】
図９は、コード・レート１／２のシンボル・ベースのトレリス復号の好ましいトレリス・ツリーを示す。図９において、下側軸は、並べ替え済みのシンボルS_0からS_7と、各シンボルに関連する対応するビットとを示す。上述のように、順方向順序は、左から右に示す順序であり、逆方向順序は、右から左に示す順序である。上側軸は、コード・レート１／２のビット・ベースのトレリス・ツリーに関して上述した、各シンボル遷移または状態変化で使用されるパリティ・チェック方程式の数を示す。実線は、次のシンボルの値または確率値が「０１」であるときの、直前のシンボルから次のシンボルへの遷移を示す。破線は、次のシンボルの値または確率値が「００」であるときの、直前のシンボルから次のシンボルへの遷移を示す。点線は、次のシンボルの値または確率値が「１０」であるときの、直前のシンボルから次のシンボルへの遷移を示す。最後に、一点鎖線は、次のシンボルの値または確率値が「１１」であるときの、直前のシンボルから次のシンボルへの遷移を示す。

【０１０６】
同様に、図１０は、２４ビットのビット・グループ・サブセットを使用する、コード・レート２／３のシンボル・ベースのトレリス復号の好ましいトレリス・ツリーを示す。下側軸は、並べ替え済みのシンボルS_0からS_{12}と、各シンボルに関連する対応するビットとを示す。上側軸は、コード・レート２／３のビット・ベースのトレリス・ツリーに関して上述した、各シンボル遷移または状態変化で使用されるパリティ・チェック方程式の数を示す。各線によって示す遷移は、図９と同様である。

【０１０７】
図１１Ａおよび図１１Ｂは、本開示の特徴を用いた要素バイト・コード・デコーダで使用されるトレリス・ブロック１１００の実施例を示すブロック図である。トレリス・ブロック１１００を用いて、図６のアルファ・トレリス・ブロック６２５またはベータ・トレリス・ブロック６３０の何れを実施することもできる。アルファ・トレリス・ブロック６２５は、後述するように、ベータ・トレリス・ブロック６３０とは異なる制御シーケンスを使用することができる。さらに、後述するように、各ブロックは、トレリス・ツリーの評価において、異なるメトリック計算を実施することもできる。

【０１０８】
トレリス・ブロック１１００において、トレリス・ブロック１１００に供給されたメッセージ信号は、ｉｎｐｕｔ＿ｓｅｌｅｃｔブロック１１１０に接続される。一連の加算器

１１２５ａ～ｐが、入力メッセージの選択部分（すなわちビットまたはシンボル）を、ｆｅｅｄｂａｃｋ＿ｓｅｌｅｃｔブロック１１２０を介して選択されたメトリック計算出力の一部分（すなわちビットまたはシンボル）と結合する。メトリック計算は、コア・プロセス・ユニット１１３０ａ～ｈおよび１１３５ａ～ｄで行われる。アルファ・トレリス・ブロック６２５およびベータ・トレリス・ブロック６３０で実施されるメトリック計算についても、後述する。

【０１０９】

コア・プロセス・ユニット１１３０ａ～ｈおよび１１３５ａ～ｄで計算された各メトリックは、ｏｕｔｐｕｔ＿ｍｕｘ１１４０に接続される。ｏｕｔｐｕｔ＿ｍｕｘ１１４０の出力は、Ｄ＿バッファ１１５０ａ～ｈでバッファリングされる。Ｄ＿バッファ１１５０ａ～ｈは、計算されたメトリック値を、ｆｅｅｄｂａｃｋ＿ｓｅｌｅｃｔブロック１１２０に供給する。

【０１１０】

ｉｎｐｕｔ＿ｓｅｌｅｃｔブロック１１１０、ｆｅｅｄｂａｃｋ＿ｓｅｌｅｃｔブロック１１２０、およびｏｕｔｐｕｔ＿ｍｕｘ１１４０によって実行される選択動作および多重化動作は、制御ブロック１１７０によって制御される。トレリス・ブロック１１００は、同期クロック信号およびストローブ信号も含み、非使用時に非活動化されたブロックのためのディセーブルｍｕｘ１１６０を備え、さらに制御ブロック１１７０に供給される復号コード・レートを選択するためのコード・レート・モード入力信号も含む。

【０１１１】

アルファ・トレリス・ブロック６２５によって生成される順方向メトリックは、トレリス・ツリーの左側から始まり、処理が進むにつれて右側に進む。アルファ・トレリス・ブロック６２５のトレリス・ツリーの順方向メトリック計算は、以下の式で表される。

【０１１２】

【数１６】

$$\alpha_{n+1}^{k} = \log\left(e^{\alpha_n^i + L_n^j}\right) \qquad (13)$$

【０１１３】

数式１３において、

【０１１４】

【数１７】

L_n^j

は、ｎ番目のシンボルのブランチｊのメトリックである。例えば、値「００」を有するシンボルｎのメトリック計算は、

【０１１５】

【数１８】

L_n^0

で表される。ｎ番目のシンボルの状態ｋのメトリックは、

【０１１６】

【数１９】

α_n^k

で表される。図９を参照すると、トレリス・ツリーの順方向メトリックは、以下のように評価することができる。

【０１１７】

【数２０】

$$\alpha_0^k = 0 \forall k = 0,1..3$$
$$\alpha_1^0 = \log\left(e^{\alpha_0^0 + L_0^0}\right)$$
$$\alpha_1^1 = \log\left(e^{\alpha_0^0 + L_0^1}\right)$$
$$\alpha_1^2 = \log\left(e^{\alpha_0^0 + L_0^2}\right)$$
$$\alpha_1^3 = \log\left(e^{\alpha_0^0 + L_0^3}\right)$$
$$\alpha_2{}^0 = \log\left(e^{\alpha_1^0 + L_1^0} + e^{\alpha_1^1 + L_1^1} + e^{\alpha_1^2 + L_1^2} + e^{\alpha_1^3 + L_1^3}\right) \quad (14)$$
$$\alpha_2{}^1 = \log\left(e^{\alpha_1^0 + L_1^1} + e^{\alpha_1^1 + L_1^0} + e^{\alpha_1^2 + L_1^3} + e^{\alpha_1^3 + L_1^2}\right)$$
$$\alpha_2{}^2 = \log\left(e^{\alpha_1^0 + L_1^2} + e^{\alpha_1^1 + L_1^3} + e^{\alpha_1^2 + L_1^0} + e^{\alpha_1^3 + L_1^1}\right)$$
$$\alpha_2{}^3 = \log\left(e^{\alpha_1^0 + L_1^3} + e^{\alpha_1^1 + L_1^2} + e^{\alpha_1^2 + L_1^1} + e^{\alpha_1^3 + L_1^0}\right)$$
$$\vdots$$
$$\alpha_{12}{}^0 = \log\left(e^{\alpha_{11}^0 + L_{11}^0} + e^{\alpha_{11}^1 + L_{11}^2} + e^{\alpha_{11}^2 + L_{11}^1} + e^{\alpha_{11}^3 + L_{11}^3}\right)$$

【０１１８】

一般に、コア・プロセス・ユニットで実施されるコア計算は、以下のように評価することができる。

【０１１９】

【数２１】

$$c = \log\left(e^a + e^b + e^c\right) \quad (15)$$

【０１２０】

数式１５のａ、ｂ、およびｃは指数係数を表し、コア・プロセス・ユニット１１３０ａ～ｈおよび１１３５ａ～ｄでは信号の標識として示される。計算速度を向上させるために、コア・プロセス・ユニット１１３０ａ～ｈおよび１１３５ａ～ｄは、以下のように計算を繰り返し実施する。

【０１２１】

【数２２】

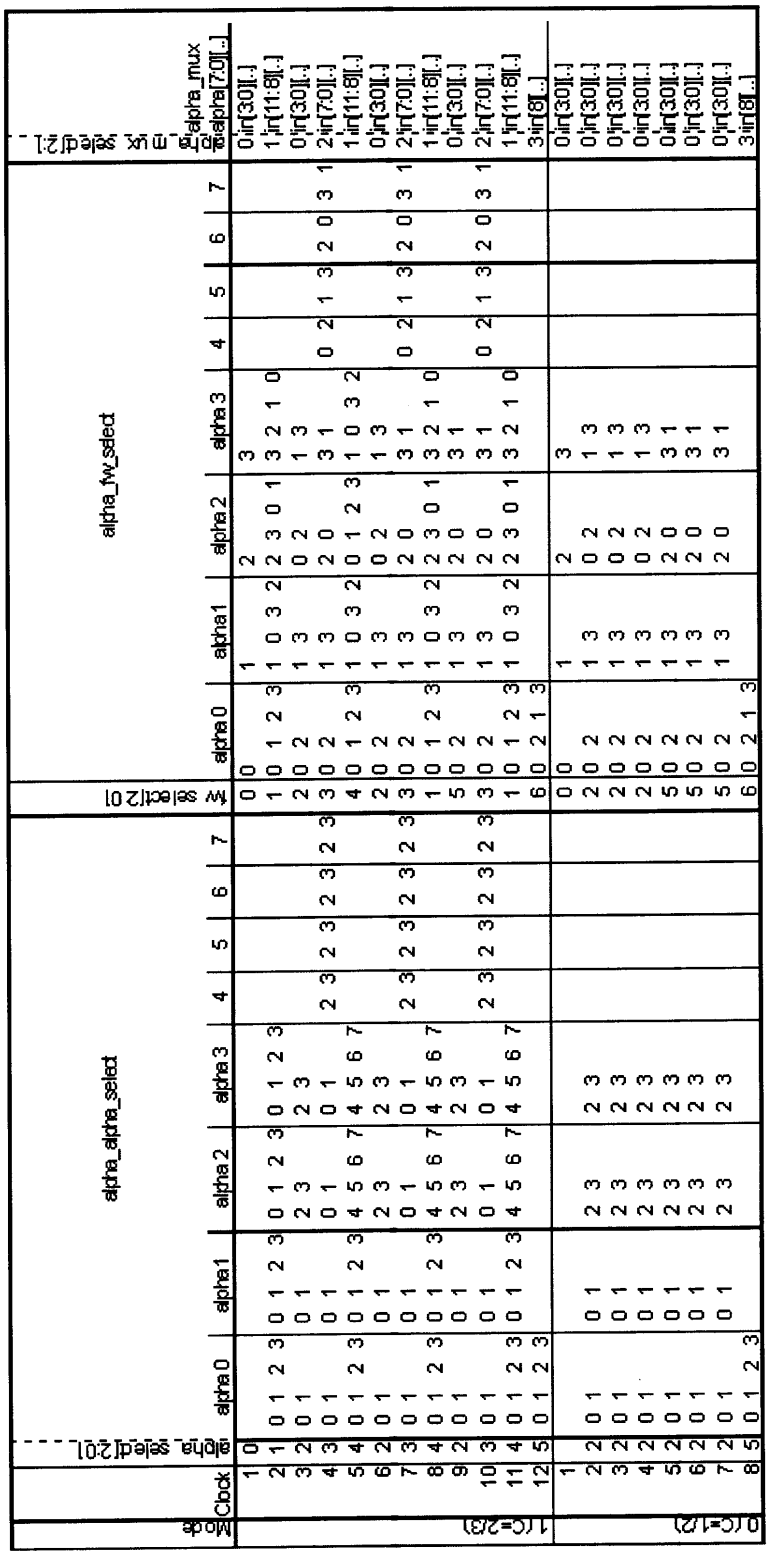
$$c = \log\left(e^{\log\left(e^a + e^b\right)} + e^c\right). \quad (16)$$

【０１２２】

上述のように、トレリス・ブロック１１００のアーキテクチャを共有することができるが、アルファ・トレリス・ブロック６２５は、ベータ・トレリス・ブロック６３０とは異なる制御シーケンスを実施することができる。制御シーケンスは、トレリス・コードおよび使用するトレリス図から導出することができる。好ましい実施例では、アルファ・トレリス・ブロック６２５の一部として使用されるｉｎｐｕｔ＿ｓｅｌｅｃｔブロック１１１０、ｆｅｅｄｂａｃｋ＿ｓｅｌｅｃｔブロック１１２０およびｏｕｔｐｕｔ＿ｍｕｘ１１４０の制御シーケンスは、以下の表１に示すように実施することができる。

【０１２３】

【表１】



【０１２４】

この表の各空きエントリに対して、ハードウェアに符号のない最大値を与えることができる。さらに、アルファ・トレリスで使用される制御シーケンスをプログラミングするコードの例示的な実施例は、以下のように表すことができる。

【０１２５】

【数２３】

```
Mode = 1 (C=2/3); Clock 2;
Alpha 0
CPU[0] Input A = alpha 0 + fw_modlogi 0
CPU[0] Input B = alpha 1 + fw_modlogi 1
CPU[1] Input A = alpha 2 + fw_modlogi 2
CPU[1] Input B = alpha 3 + fw_modlogi 3
(CPU[8] Input A = CPU[0]
 CPU[8] Input B = CPU[1])
Alpha 1
CPU[2] Input A = alpha 0 + fw_modlogi 1
CPU[2] Input B = alpha 1 + fw_modlogi 0
CPU[3] Input A = alpha 2 + fw_modlogi 3
CPU[3] Input B = alpha 3 + fw_modlogi 2
(CPU[9] Input A = CPU[2]
 CPU[9] Input B = CPU[3])
Alpha 2
CPU[4] Input A = alpha 0 + fw_modlogi 2
CPU[4] Input B = alpha 1 + fw_modlogi 3
CPU[5] Input A = alpha 2 + fw_modlogi 0
CPU[5] Input B = alpha 3 + fw_modlogi 1
(CPU[10] Input A = CPU[4]
 CPU[10] Input B = CPU[5])
Alpha 3
CPU[6] Input A = alpha 0 + fw_modlogi 3
CPU[6] Input B = alpha 1 + fw_modlogi 2
CPU[7] Input A = alpha 2 + fw_modlogi 1
CPU[7] Input B = alpha 3 + fw_modlogi 0
(CPU[11] Input A = CPU[6]
 CPU[11] Input B = CPU[7])

Alpha[7:0][..] = in[11:8][..] = CPU[11:8][..]
```

【０１２６】

アルファ・トレリス・ブロック６２５が生成する順方向メトリックとは異なり、ベータ・トレリス・ブロック６３０が生成する逆方向メトリックは、トレリス・ツリーの右側から始まり、処理が進むにつれて左側に進む。ベータ・トレリス・ブロック６３０のトレリス・ツリーの逆方向メトリック計算は、以下のように表される。

【０１２７】

【数２４】

$$\beta_n^k = \log\left(e^{\beta_{n+1}^i + L_n^j}\right) \qquad (17)$$

【０１２８】

数式１８において、

【０１２９】

【数２５】

L_n^j

は、ｎ番目のシンボルのブランチｊのメトリックである。上述のように、２進値「００」を有するシンボルｎのメトリックは、

【０１３０】

【数２６】

$$L_n^0$$

で表される。ｎ番目のシンボルの状態ｋのメトリックは、

【０１３１】

【数２７】

$$\beta_n^k$$

で表される。図９を再度参照すると、逆方向メトリック計算は、以下の式を用いて評価される。

【０１３２】

【数２８】

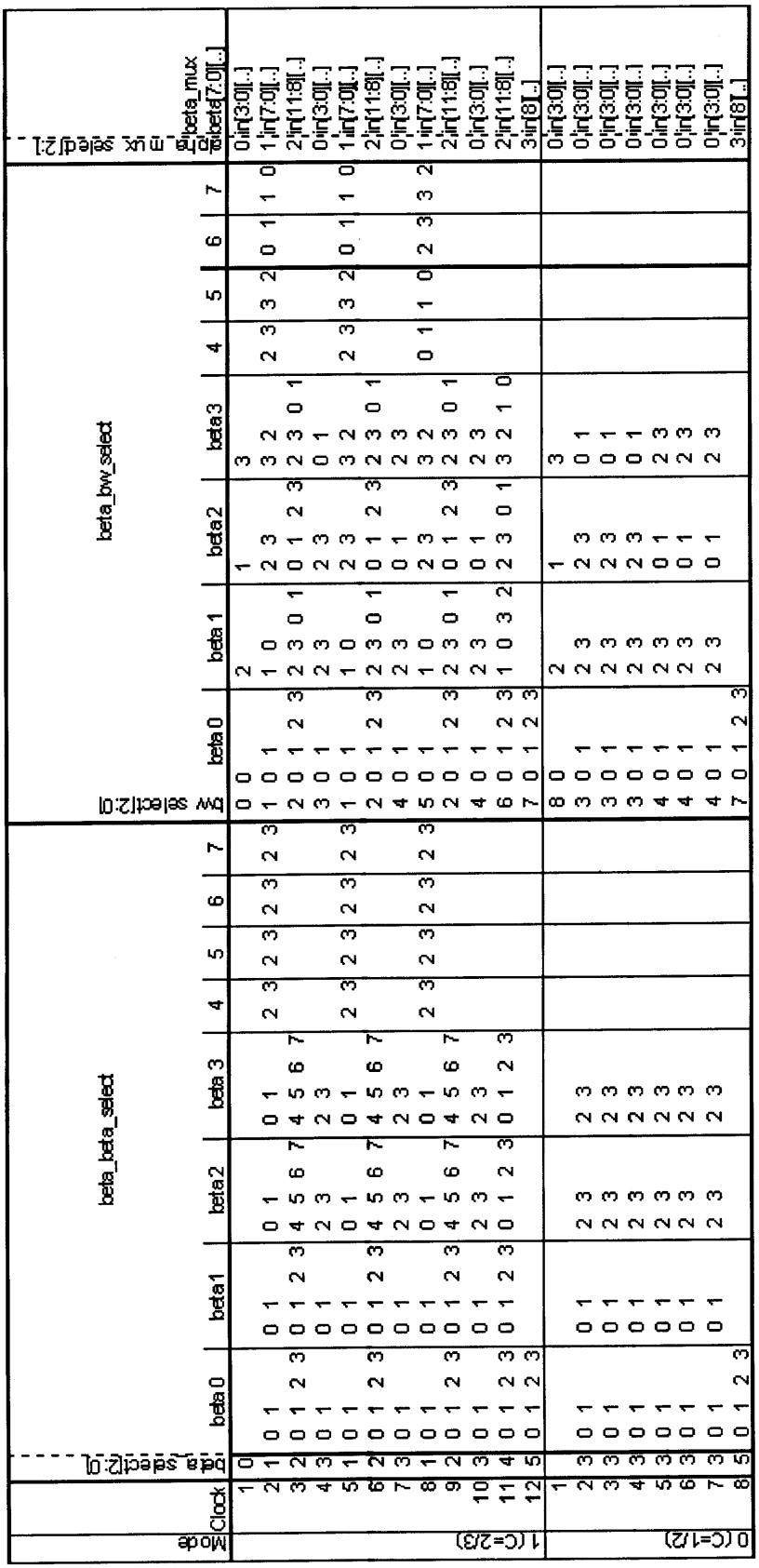
$$\begin{aligned}
&\beta_{12}^k = 0 \forall k = 0,1..3\\
&\beta_{11}^0 = \log\left(e^{\beta_{12}^0 + L_{11}^0}\right)\\
&\beta_{11}^1 = \log\left(e^{\beta_{12}^0 + L_{11}^2}\right)\\
&\beta_{11}^2 = \log\left(e^{\beta_{12}^0 + L_{11}^1}\right)\\
&\beta_{11}^3 = \log\left(e^{\beta_{12}^0 + L_{11}^3}\right)\\
&\beta_{10}{}^0 = \log\left(e^{\beta_{11}^0 + L_{10}^0} + e^{\beta_{11}^1 + L_{10}^1}\right)\\
&\beta_{10}{}^1 = \log\left(e^{\beta_{11}^0 + L_{10}^1} + e^{\beta_{11}^1 + L_{10}^0}\right)\\
&\beta_{10}{}^2 = \log\left(e^{\beta_{11}^0 + L_{10}^2} + e^{\beta_{11}^1 + L_{10}^3}\right)\\
&\beta_{10}{}^3 = \log\left(e^{\beta_{11}^0 + L_{10}^3} + e^{\beta_{11}^1 + L_{10}^2}\right)\\
&\beta_{10}{}^4 = \log\left(e^{\beta_{11}^2 + L_{10}^2} + e^{\beta_{11}^3 + L_{10}^3}\right)\\
&\beta_{10}{}^5 = \log\left(e^{\beta_{11}^2 + L_{10}^3} + e^{\beta_{11}^3 + L_{10}^2}\right)\\
&\beta_{10}{}^6 = \log\left(e^{\beta_{11}^2 + L_{10}^0} + e^{\beta_{11}^3 + L_{10}^1}\right)\\
&\beta_{10}{}^7 = \log\left(e^{\beta_{11}^2 + L_{10}^1} + e^{\beta_{11}^3 + L_{10}^0}\right)\\
&\vdots \qquad\qquad (18)\\
&\beta_0{}^0 = \log\left(e^{\alpha_1^0 + L_1^0} + e^{\alpha_1^1 + L_1^1} + e^{\alpha_1^2 + L_1^2} + e^{\alpha_1^3 + L_1^3}\right)
\end{aligned}$$

【０１３３】

好ましい実施例では、ベータ・トレリス・ブロック６３０の一部として使用されるｉｎｐｕｔ＿ｓｅｌｅｃｔブロック１１１０、ｆｅｅｄｂａｃｋ＿ｓｅｌｅｃｔブロック１１２０およびｏｕｔｐｕｔ＿ｍｕｘ１１４０の制御シーケンスは、以下の表２に示すように実施することができる。

【０１３４】

【表２】


Mode
Clock
beta_data_select
beta_beta_select
beta_bw_select
beta_mux
0 (C=1/2)
1 (C=2/3)

【０１３５】

図１２Ａおよび図１２Ｂは、本開示の特徴を用いた要素バイト・コード・デコーダで使用されるアポステリオリ・ブロック１２００の実施例を示すブロック図である。アポステリオリ・ブロック１２００において、メッセージ信号は、ｆｗｄ＿ｓｅｌｅｃｔブロック１２１０に接続される順方向メトリック、ｒｖｓ＿ｓｅｌｅｃｔブロック１２２０に接続される逆方向メトリック、およびｉｎｔｒｉｎｓｉｃ＿ｓｅｌｅｃｔブロック１２３０に接続されるイントリンシック・メトリックを表す。一連の加算器１２３５ａ～ｐおよび１２４０ａ～ｐが、順方向メトリック・メッセージ、逆方向メトリック・メッセージおよびイントリンシック・メッセージの選択部分（すなわちビットまたはシンボル）を結合する。この新たなメトリック計算は、コア・プロセス・ユニット１２５０ａ～ｈおよび１２５５ａ～ｄで行われる。

【０１３６】

コア・プロセス・ユニット１２５０ａ～ｈおよび１２５５ａ～ｄで計算された各メトリックは、ｐｏｓｔ＿ｍｕｘ１２６０に接続される。ｐｏｓｔ＿ｍｕｘ１２６０の出力は、Ｄ＿バッファ１２７０ａ～ｄでバッファリングされる。Ｄ＿バッファ１２７０ａ～ｄは、計算されたメトリック値を、出力としてアポステリオリ・ブロック１２００に供給する。

【０１３７】

ｆｗｄ＿ｓｅｌｅｃｔブロック１２１０、ｒｖｓ＿ｓｅｌｅｃｔブロック１２２０、ｉｎｔｒｉｎｓｉｃ＿ｓｅｌｅｃｔブロック１２３０およびｐｏｓｔ＿ｍｕｘ１２６０によって実行される選択動作および多重化動作は、制御ブロック１２８０によって制御される。アポステリオリ・ブロック１２００は、Ｄ＿バッファ１２７５ａ～ｂを介してさらに接続される同期クロック信号およびストローブ信号、ならびに制御ブロック１２８０に供給される復号コード・レートを選択するためのコード・レート・モード入力信号も含む。

【０１３８】

アポステリオリ・ブロック１２００は、以下の数式を用いてアポステリオリ・メトリックを計算する。

【０１３９】

【数２９】

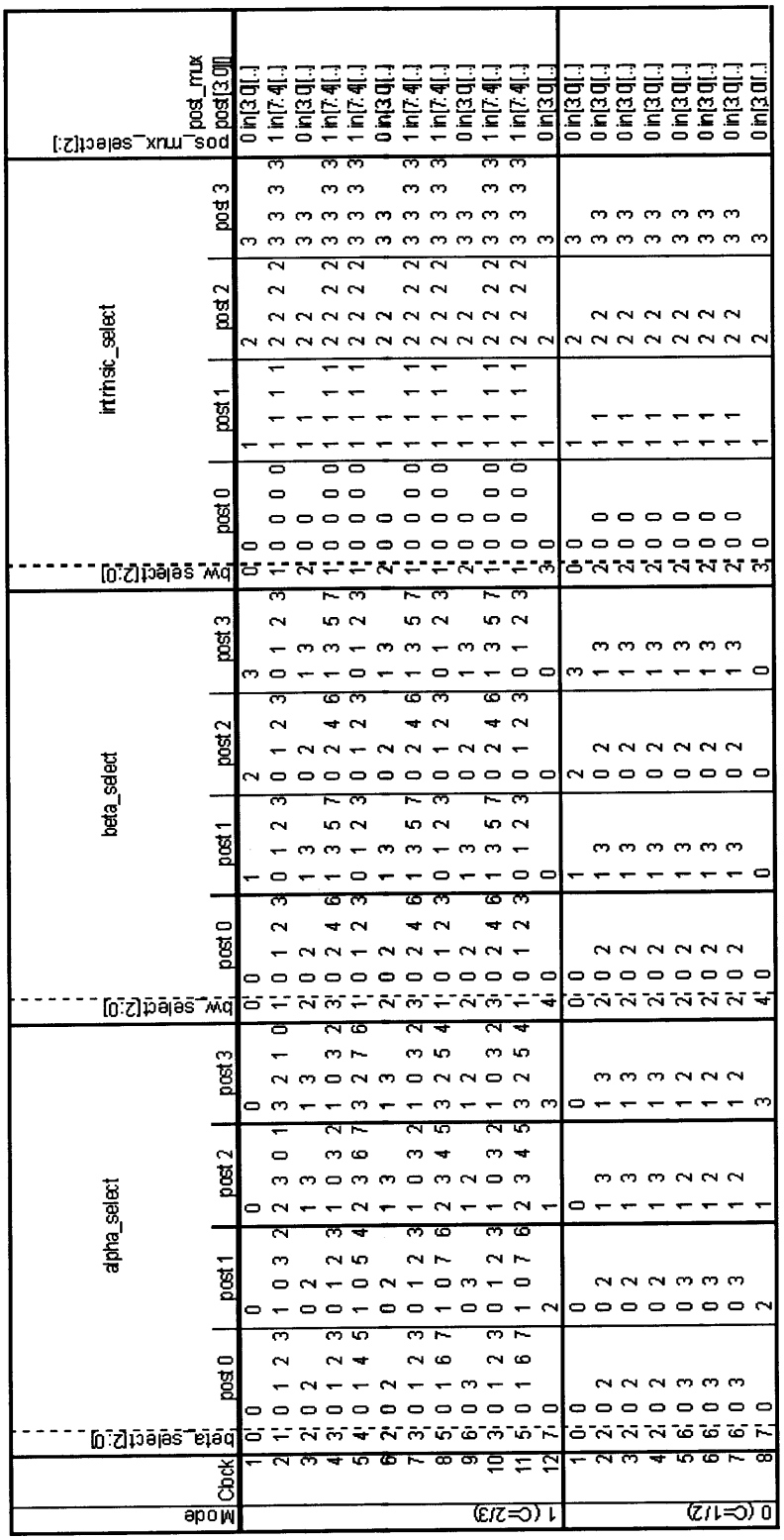
$$\gamma_0^0 = \log\left(e^{\alpha_0^0+\beta_1^0+L_0^0}\right)$$
$$\gamma_0^1 = \log\left(e^{\alpha_0^0+\beta_1^1+L_0^1}\right)$$
$$\gamma_0^2 = \log\left(e^{\alpha_0^0+\beta_1^2+L_0^2}\right)$$
$$\gamma_0^3 = \log\left(e^{\alpha_0^0+\beta_1^3+L_0^3}\right)$$
$$\vdots \qquad (19)$$
$$\gamma_{11}^0 = \log\left(e^{\alpha_{11}^0+\beta_{12}^0+L_{11}^0}\right)$$
$$\gamma_{11}^1 = \log\left(e^{\alpha_{11}^2+\beta_{12}^0+L_{11}^1}\right)$$
$$\gamma_{11}^2 = \log\left(e^{\alpha_{11}^1+\beta_{12}^0+L_{11}^2}\right)$$
$$\gamma_{11}^3 = \log\left(e^{\alpha_{11}^3+\beta_{12}^0+L_{11}^3}\right)$$

【０１４０】

好ましい実施例では、アポステリオリ・ブロック１２００の一部として使用されるｆｗｄ＿ｓｅｌｅｃｔブロック１２１０、ｒｖｓ＿ｓｅｌｅｃｔブロック１２２０、ｉｎｔｒｉｎｓｉｃ＿ｓｅｌｅｃｔブロック１２３０、およびｐｏｓｔ＿ｍｕｘ１２６０の制御シーケンスは、以下の表３に示すように実施することができる。

【０１４１】

【表３】



【０１４２】

図１３は、本開示の特徴を用いたバイト・コード・デコーダ１３００の実施例を示すブロック図である。バイト・コード・デコーダ１３００は、上述のバイト・コード符号化プロセスとパリティ行列の関係に基づいて代替の復号プロセスを利用するために、入来情報

のグループ化および並べ替えを利用する。入力コード化メッセージは、入力制御ブロック１３１０に供給される。入力制御ブロックの出力は、信号マルチプレクサ１３２０の一方の入力部に供給される。信号マルチプレクサ１３２０の出力部は、アポステリオリ・メモリ１３３０に接続される。アポステリオリ・メモリ１３３０の出力部は、加算器１３４０の一方の入力部に接続される。加算器の出力部は、遅延バッファ１３５０およびエクストリンシック計算プロセッサ１３６０に接続される。エクストリンシック計算プロセッサ１３６０の出力部は、エクストリンシック・メモリ１３７０に接続される。エクストリンシック・メモリ１３７０の出力部は、反転されて加算器１３４０の第２の入力部に接続され、加算器が減算機能を実施する。エクストリンシック計算プロセッサ１３６０および遅延バッファ１３５０の出力部は、加算器１３８０に接続される。加算器１３８０の出力部は、入力マルチプレクサ１３２０の第２の入力部に供給される。アポステリオリ・メモリ１３３０の出力は、反復復号プロセスの後で、バイト・コード・デコーダ１３００の出力となる。

【０１４３】

入力制御ブロック１３１０は、入来メッセージを受信し、入来メッセージの一部分をグループ化してビット・サブセットにする。このサブセットは、入力メッセージとなるが、これは通常は、当該サブセット内の各ビットの確率、例えばＬＬＲ値などからなる。また、入力制御ブロック１３１０は、ｍｕｘ１３２０への入力メッセージの送達の制御またはゲートも行い、ｍｕｘ１３２０の入力切替を制御することもできる。デコーダ１３００の反復動作は、ｍｕｘ１３２０によって制御される。ｍｕｘ１３２０は、初期復号の復号プロセス中の第１の反復時に入力メッセージ・ストリームを出力する、あるいは復号プロセス中の後の反復において加算器１３８０から供給される新たに計算されたフィードバック・メッセージを出力する。

【０１４４】

ｍｕｘ１３２０の選択出力は、アポステリオリ・メモリ１３３０に供給される。アポステリオリ・メモリ１３３０は、ｍｕｘ１３２０から受信した情報を記憶し、配列する。好ましい実施例では、アポステリオリ・メモリ１３３０は、復号プロセスの第１の反復中に、ビット信頼度として受信した入来ビット・サブセットを並べ替える。前述のように、並べ替えにより、パリティ・チェック行列を含む復号プロセスを改善することができることもある。アポステリオリ・メモリ１３３０は、並べ替えされた情報メッセージを、ビット・サブセットまたはビット信頼度として、結合器１３４０に供給する。

【０１４５】

結合器１３４０は、並べ替えされた情報メッセージを、エクストリンシック・メモリ１３７０から供給されるエクストリンシック情報メッセージと結合する。エクストリンシック・メモリ１３７０は、エクストリンシック計算ブロック１３６０で計算されたエクストリンシック情報メッセージのエクストリンシック情報値を記憶する。好ましい実施例では、結合器１３４０は、並べ替えされた情報メッセージ値からエクストリンシック情報メッセージ値を減算し、結合した情報メッセージ値をエクストリンシック計算ブロック１３６０に出力する。

【０１４６】

エクストリンシック計算ブロック１３６０は、結合器１３４０から受け取った結合情報メッセージを処理する。エクストリンシック計算ブロック１３６０の機能については、後にさらに述べる。結合器１３４０からの結合情報メッセージは、バッファ１３５０にも供給される。バッファ１３５０は、結合器１３８０におけるエクストリンシック計算ブロック１３６０の出力との再結合を適切に行うことができるようにするために、結合情報メッセージを遅延させるために使用される。結合器１３８０は、この遅延された結合情報メッセージを、エクストリンシック計算ブロック１３６０から出力されるエクストリンシック情報と結合する。結合器１３８０は、アポステリオリ情報メッセージ値を含む信号を出力し、この信号を、反復復号プロセスの中で使用されるようにｍｕｘ１３２０に供給する。

【０１４７】

要素復号プロセスおよび連結復号プロセスは両方とも、１つのパリティ・チェック行列を用いて記述することができ、バイト・コード・デコーダ１３００など単一のデコーダで処理することができることに留意されたい。連結復号プロセスでは、各要素復号プロセスを、それぞれのパリティ・チェック行列で定義することができる。両復号プロセス、従って両パリティ行列は、図４に示すようにインタリーバおよびデインタリーバを介して結合される。それらのパリティ・チェック行列を結合して、連結復号プロセスにおける両プロセスに使用できる単一のパリティ・チェック行列を形成することができる。インタリービング・ステップで、結合パリティ・チェック行列の非ゼロ要素の位置を規定する。

【０１４８】

別々に接続された要素デコーダのパリティ・チェック行列を結合することにより、バイト・コード・デコーダ１３００は、全ての可能なバイト・コード・コード・レート構成を復号することができる。入来メッセージの当該部分のＬＬＲ値はアポステリオリ・メモリに保存され、必要に応じて並べ替えられる。エクストリンシック情報は、アポステリオリ信頼度から減算される。その結果得られたイントリンシック情報が、バッファ１３５０およびエクストリンシック計算ブロック１３６０に送られる。計算したエクストリンシック情報は、エクストリンシック・メモリ１３７０に戻されて保存される。さらに、計算したエクストリンシック情報は、直前のイントリンシック情報にも付加され、アポステリオリ・メモリ１３３０に戻されて保存される。この計算プロセスは、各反復復号ステップについて繰り返される。反復復号は、情報メッセージのアポステリオリ信頼度値が信頼度しきい値を超えるまで、または割り当てられた復号時間が満了するまで、継続される。

【０１４９】

図１４は、本発明の特徴を用いたバイト・コード・デコーダ１４００の別の実施例を示すブロック図である。バイト・コード・デコーダ１４００は、低密度パリティ・チェック（ＬＤＰＣ）復号と呼ばれる特定のパリティ・チェック復号プロセスを実施するように構成することができる。

【０１５０】

情報メッセージのビットの確率を表す８つの入力ＬＬＲ値のシーケンスの形態をした入力メッセージが、パミュータ（ｐｅｒｍｕｔｅｒ）１４１０の入力部に供給される。パミュータ１４１０は、アプリオリな（すなわち第１の反復の）、またはアポステリオリのＬＬＲ値を順次受信する。パミュータ１４１０は、単純なシフト・レジスタであり、入来ＬＬＲ値を加算器１４２０ａ～ｈおよび１４４０ａ～ｃに直接接続することを可能にする。また、パミュータ１４１０により、入来値を右に１つ巡回桁送りすることが可能になることもある。パミュータ１４１０からの８つの出力部はそれぞれ、加算器１４２０ａ～ｈの一方の入力部に接続される。さらに、Ｌ３、Ｌ４およびＬ５で示されるパミュータ１４１０の３つの出力は、それぞれ第２のセットの加算器１４４０ａ～ｃの一方の入力部に供給される。

【０１５１】

計算したエクストリンシック情報は、エッジ先入れ先出し（ＦＩＦＯ）バッファ１４７０およびエッジＦＩＦＯバッファ１４７２を介して、さらに乗算器１４３０および１４５０によって符号が反転されて、加算器１４２０ａ～ｈおよび１４４０ａ～ｃの第２の入力部に接続される。計算したエクストリンシック情報は、入来値から減算される。その結果得られる、加算器１４２０ａ～ｈの出力におけるイントリンシック情報が、検査プロセス更新ユニット（ＣＰＵ）１４６５ａ～ｈおよびＦＩＦＯバッファ１４６０ａ～ｈに供給される。ＣＰＵ１４６５ａ～ｈの動作については、後に詳細に述べる。ＣＰＵ１４６５ａ～ｈからの計算したエクストリンシック情報は、加算器１４２０ａ～ｈおよび１４４０ａ～ｃに供給されるだけでなく、ＦＩＦＯバッファ１４６０ａ～ｈにバッファリングされたイントリンシック情報と結合される。その結果得られる新たなアポステリオリ情報であるビットのＬＬＲ値は、必要に応じてパミュータ１４９５で元の位置にシフトバックされ、この新たなアポステリオリＬＬＲ値が、図１３に示すアポステリオリ・メモリ１３３０などのメモリに戻されて保存される。

【０１５２】
ＣＰＵ１４６５ｄ、１４６５ｅおよび１４６５ｆは、２つのエクストリンシック情報値を計算することができる。ＣＰＵ１４６５ｄ、１４６５ｅおよび１４６５ｆは、第２の入力を、ｍｕｘ１４４５ａ～ｃを介して加算器１４４０ａ～ｃから受信する。ＦＩＦＯバッファ１４６２ａ～ｃは、直前に計算したエクストリンシック値を加算器１４８５ａ～ｅに供給し、ＣＰＵ１４６５ｄ、１４６５ｅおよび１４６５ｆから供給される１組の新たに計算されたエクストリンシック値と結合されるようにする。
【０１５３】
バイト・コード・デコーダ１４００で使用される復号プロセスは、２進ＬＤＰＣ復号プロセスと見なすことができる。各パリティ・チェック方程式は、低密度パリティ・チェックコードと解釈される。バイト・コード・デコーダ１４００の処理並列度は８である。すなわち、バイト・コード・デコーダ１４００は、８つの方程式を１組として１度に処理することができる。
【０１５４】
ＣＰＵ１４６５ａ～ｈは、ビット・ベースのＬＤＰＣアルゴリズムを使用してエクストリンシック情報を計算する。好ましい実施例では、ＬＤＰＣアルゴリズムの方程式は、以下のように表すことができる。
【０１５５】
【数３０】

$$R_{m,n} = 2\tanh^{-1}\left(\prod_{i\in C(m)\backslash n}\tanh\left(Q_{m,i}/2\right)\right) \stackrel{def}{\equiv} L\left(\sum_{i\in C(m)\backslash n}\oplus Q_{m,i}\right). \qquad (20)$$

【０１５６】
上記の方程式で、
【０１５７】
【数３１】

$$Q_{m,i_j}$$

は、ＣＰＵ１４６５ａ～ｈへの入力である、パリティ・チェック方程式ｍおよびビットi_jの計算したイントリンシック値を表す。
【０１５８】
【数３２】

$$R_{m,i_j}$$

は、ＣＰＵ１４６５ａ～ｈによって計算された新たなエクストリンシック値を表す。Ｃ（ｍ）＝｛ｎ：$H_{m,n}=1$｝は、検査方程式に接続される入力値の組を表す。ＬＤＰＣ復号プロセスでは、恒等行列Ｉおよび数式６の生成行列Ｓの両方に基づいてエクストリンシック値を計算する。恒等行列の復号中には、修正ＣＰＵユニットの第２の入力は使用されない。この入力は、ｍｕｘ１４４５ａ～ｃを介して最大の正のＬＬＲ値を供給することによって簡単にディセーブルされる。Ｓ行列の復号中には、イントリンシック情報を計算し、これを第２の入力部に供給する。
【０１５９】
Ｓ行列の第１列には、値が１である位置が複数ある。当該列の対応するアポステリオリ値は、当該列の各方程式の全てのエクストリンシック情報の和である。この加算は、加算器１４８５ｆおよびｍｕｘ１４９２を用いて行われる。ＣＰＵ１４６５ｈおよび加算器１４８０ｈの新たなアポステリオリ値を計算するために、ＣＰＵ１４６５ｄ、１４６５ｅおよび１４６５ｆでエクストリンシック値を計算し、加算器１４８０ｈからのアポステリオリ値を加算器１４８５の出力で結合する。
【０１６０】
上述のバイト・コード・デコーダ１４００では、その他のビットのグループ化が可能で

あることもある。例えば、２進ＬＤＰＣ復号プロセスで使用される８ビット以上のサブセットを形成するためのグループ化に加えて、当該サブセット内のビットをさらにグループ化してＬＤＰＣシンボルと呼ばれるシンボルを形成することにより、第２のグループを形成することもできる。この追加のグループ化により、ＬＤＰＣ復号プロセスを使用したときの復号時間が短縮されるので、復号プロセスはさらに改善される。

【０１６１】

ビットの代わりにＬＤＰＣシンボルを使用してエクストリンシック確率値およびアポステリオリ確率値を計算する復号アルゴリズムは、一般に、非２進ＬＤＰＣコードと呼ばれ、ガロア体空間の性質を用いて定義することができる。好ましい実施例では、ＧＦ（８）で、原始多項式は、以下のように定義することができる。

【０１６２】

【数３３】

$$p(x)=1+x+x^3 \qquad (21)$$

【０１６３】

受信したコード化メッセージは、システマティック・ビット情報および非システマティック・ビット情報を含む３つのメッセージ・ビットを含むシンボルとして表すことができる。

【０１６４】

その結果として、各受信シンボルは、以下の８つの確率として表すことができる。

【０１６５】

【数３４】

$$P(x_{symbol}="000"),P(x_{symbol}="001"),P(x_{symbol}="010"),P(x_{symbol}="011"),$$
$$P(x_{symbol}="100"),P(x_{symbol}="101"),P(x_{symbol}="110"),P(x_{symbol}="111")$$

【０１６６】

ＬＤＰＣシンボルを復号するために、以下のようにパリティ・チェック方程式を使用することができる。

【０１６７】

【数３５】

$$cH^T=3c_2+2c_1+c_0=0,\ c_i\in GF(8) \qquad (22)$$

【０１６８】

数式２２において、各シンボルｃ$_i$は、８つの異なるビット・コンスタレーションまたはビット確率を表す。方程式全体では、５１２（すなわち８×８×８）の異なる可能なビット・コンスタレーションを含むことになる。これらのコンスタレーション確率を入来情報に畳み込むことにより、新たなコンスタレーション確率を計算する。例えば、ｃ$_0$＝０に対して可能なビット・コンスタレーションは、以下のように表すことができる。

【０１６９】

【数３６】

$$r_{0,0}^0=\sum_{x':x'_0=0}\Pr ob\left[z_0\mid x'\right]\prod_{j\in N(0)\setminus 0}q_{0,j}^{x'_j} \qquad (23)$$
$$=q_{0,2}^0q_{0,1}^0+q_{0,2}^1q_1^4+q_{0,2}^2q_{0,1}^3+q_{0,2}^3q_{0,1}^7+q_{0,2}^5q_{0,1}^6+q_{0,2}^5q_{0,1}^2+q_{0,2}^6q_{0,1}^5+q_{0,2}^7q_{0,1}^1$$

数式２３において、値ｑ$^x{}_j$は、可能なコンスタレーション値のそれぞれについての現在の確率を表し、ｒ$^n{}_m$は、可能なコンスタレーション値のそれぞれについて新たに計算した確率を表す。

【０１７０】

【数３７】

$$r_{00}^{i}$$

のそれぞれに対して、２^{p}＝８の異なる可能性がある。

【０１７１】

【数３８】

$$r_{00,0}^{1} = \sum_{x':x_0'=1} \Pr ob\left[z_0 \mid x'\right] \prod_{j\in N(0)\setminus 0} q_{0j}^{x_j'} = q_{0,2}^{0}q_{0,1}^{5} + q_{0,2}^{1}q_{0,1}^{1} + q_{0,2}^{2}q_{0,1}^{6} + q_{0,2}^{3}q_{0,1}^{2} + q_{0,2}^{4}q_{0,1}^{3} + q_{0,2}^{5}q_{0,1}^{7} + q_{0,2}^{6}q_{0,1}^{0} + q_{0,2}^{7}q_{0,1}^{4} \quad (24)$$

$$\begin{aligned}
r_{0,0}^{2} &= q_{0,2}^{0}q_{0,1}^{1} + q_{0,2}^{1}q_{0,1}^{5} + q_{0,2}^{2}q_{0,1}^{2} + q_{0,2}^{3}q_{0,1}^{6} + q_{0,2}^{4}q_{0,1}^{7} + q_{0,2}^{5}q_{0,1}^{3} + q_{0,2}^{6}q_{0,1}^{4} + q_{0,2}^{7}q_{0,1}^{0} \\
r_{0,0}^{3} &= q_{0,2}^{0}q_{0,1}^{4} + q_{0,2}^{1}q_{0,1}^{0} + q_{0,2}^{2}q_{0,1}^{7} + q_{0,2}^{3}q_{0,1}^{3} + q_{0,2}^{4}q_{0,1}^{2} + q_{0,2}^{5}q_{0,1}^{6} + q_{0,2}^{6}q_{0,1}^{1} + q_{0,2}^{7}q_{0,1}^{5} \\
r_{0,0}^{4} &= q_{0,2}^{0}q_{0,1}^{2} + q_{0,2}^{1}q_{0,1}^{6} + q_{0,2}^{2}q_{0,1}^{1} + q_{0,2}^{3}q_{0,1}^{5} + q_{0,2}^{4}q_{0,1}^{4} + q_{0,2}^{5}q_{0,1}^{0} + q_{0,2}^{6}q_{0,1}^{7} + q_{0,2}^{7}q_{0,1}^{3} \\
r_{0,0}^{5} &= q_{0,2}^{0}q_{0,1}^{7} + q_{0,2}^{1}q_{0,1}^{3} + q_{0,2}^{2}q_{0,1}^{4} + q_{0,2}^{3}q_{0,1}^{0} + q_{0,2}^{4}q_{0,1}^{1} + q_{0,2}^{5}q_{0,1}^{5} + q_{0,2}^{6}q_{0,1}^{2} + q_{0,2}^{7}q_{0,1}^{6} \\
r_{0,0}^{6} &= q_{0,2}^{0}q_{0,1}^{3} + q_{0,2}^{1}q_{0,1}^{7} + q_{0,2}^{2}q_{0,1}^{0} + q_{0,2}^{3}q_{0,1}^{4} + q_{0,2}^{4}q_{0,1}^{5} + q_{0,2}^{5}q_{0,1}^{1} + q_{0,2}^{6}q_{0,1}^{6} + q_{0,2}^{7}q_{0,1}^{2} \\
r_{0,0}^{7} &= q_{0,2}^{0}q_{0,1}^{6} + q_{0,2}^{1}q_{0,1}^{2} + q_{0,2}^{2}q_{0,1}^{5} + q_{0,2}^{3}q_{0,1}^{1} + q_{0,2}^{4}q_{0,1}^{0} + q_{0,2}^{5}q_{0,1}^{4} + q_{0,2}^{6}q_{0,1}^{3} + q_{0,2}^{7}q_{0,1}^{7}
\end{aligned} \quad (25)$$

【０１７２】

図１５は、方程式２５をタナー・グラフで表したものである。タナー・グラフは、非２進ＬＤＰＣ復号に含まれる複雑な数学的相互作用を図形的に表現するためにしばしば使用されるものである。図１５は、ベクトル畳み込み関数による再配列した任意の値を含む値ｑ$^{x}{}_{j}$のそれぞれの初期処理、およびベクトル畳み込み後に得られる出力値ｒ$^{n}{}_{m}$を示す。

【０１７３】

畳み込み関数、特に複雑なベクトル畳み込み関数の実施には、大量の計算が必要となる。ベクトルで表されるｒ$^{n}{}_{m}$値およびｑ$^{x}{}_{j}$値に変換関数を適用すれば、単純化することができる。変換関数を適用することにより、変換後のｒ$^{n}{}_{m}$値およびｑ$^{x}{}_{j}$値のベクトル乗算を簡単にすることができる。ベクトル乗算後に、その結果得られた出力値に、逆変換関数を適用する。好ましい実施例では、フーリエ変換関数を使用することができ、これをさらに、より簡単な高速フーリエ変換（ＦＦＴ）としてハードウェアに実装することができる。ＦＦＴでは、乗算ステップではなく加算ステップで変換および逆変換を計算できるようにする、効率的なバタフライ演算技術をしばしば利用する。アダマール変換や離散コサイン変換など（ただしこれらに限定されない）、その他の変換関数を使用することもできることに留意されたい。

【０１７４】

図１６は、本発明の特徴を用いたバイト・コード・デコーダ１６００の別の実施例を示すブロック図である。バイト・コード・デコーダ１６００は、図１３のデコーダ１３００および図１４のデコーダ１４００について述べたのと同様の方法で、入来情報メッセージの一部分のグルーピングおよび並べ替えに関連する性質、ならびにバイト・コード符号化プロセスに関連するパリティ・チェック行列を利用する。

【０１７５】

バイト・コード・デコーダ１６００において、入力メッセージは、入力制御ブロック１６１０に供給される。入力制御ブロックの出力は、信号マルチプレクサ１６２０の一方の入力部に供給される。信号マルチプレクサ１６２０の出力は、アポステリオリ・メモリ１６３０に接続される。アポステリオリ・メモリ１６３０の出力部は、加算器１６３５の一方の入力部に接続される。加算器の出力部は、遅延バッファ１６４０および尤度計算プロ

セッサ１６４２に接続される。尤度計算プロセッサ１６４２の出力部は、直列に、パミュータ１６４４、高速フーリエ変換プロセッサ（ＦＦＴ）１６４６、乗算器１６４８、逆ＦＦＴ１６５０、パミュータ１６５２、および対数尤度計算プロセッサ１６５４に順次接続される。対数尤度計算プロセッサ１６５４の出力部は、エクストリンシック・メモリ１６７０および加算器１６６０の一方の入力部に接続される。エクストリンシック・メモリ１６７０の出力部は、加算器１６３５の第２の入力部に接続される。バッファ１６４０の出力部は、加算器１６６０の第２の入力部に接続される。加算器１６６０の出力部は、入力マルチプレクサ１６２０の第２の入力部に供給される。アポステリオリ・メモリ１６３０の出力は、反復復号プロセスの後で、バイト・コード・デコーダ１６００の出力となる。入力制御装置／挿入器１６１０、ｍｕｘ１６２０、アポステリオリ・メモリ１６３０、エクストリンシック・メモリ１６７０、加算器１６３５、バッファ１６４０および加算器１６６０が、バイト・コード・デコーダ１３００について述べたそれぞれ同じブロックと同様に接続され、同じ機能を有することは重要であり、必要な場合を除き、以下ではこれ以上説明しないことに留意されたい。

【０１７６】

尤度計算プロセッサ１６４２は、加算器１６３５からイントリンシック情報メッセージを受信し、イントリンシック情報のビットのＬＬＲ値を線形確率または尤度確率に変換する。尤度計算プロセッサ１６４２は、非２進ＬＤＰＣ復号プロセスのシンボル・グループ化の追加ビットに基づいてＬＤＰＣシンボルの尤度値を再計算することもできる。パミュータ１６４４は、尤度計算プロセッサからＬＤＰＣシンボルの尤度値を受信し、非２進ＬＤＰＣ復号プロセスに必要なＬＤＰＣシンボルの任意の並べ替えまたは再配列を行う。パミュータ１６４４は、ＬＤＰＣシンボルのどの組をＦＦＴ１６４６に供給するかをも制御する。

【０１７７】

ＦＦＴ１６４６、乗算器１６４８および逆ＦＦＴ１６５０は、非２進ＬＤＰＣアルゴリズムと併せて、上述の畳み込み関数を実行する。ＦＦＴ１６４６は、入来する時間領域ベースのＬＤＰＣシンボルを、等価な周波数領域シンボルに変換する。周波数領域ＬＤＰＣシンボルは、乗算器１６４８でベクトル乗算される。新たに計算された周波数領域ＬＤＰＣシンボルは、逆ＦＦＴ１６５０で変換されて、時間領域ＬＤＰＣシンボルに戻される。この新たに計算された時間領域ＬＤＰＣシンボルは、パミュータ１６５２に供給される。パミュータ１６５２は、パミュータ１６４４が実行した任意の並べ替えおよび再配列を元に戻す。

【０１７８】

対数尤度計算プロセッサ１６５４は、ＬＤＰＣシンボルに関連する線形尤度確率値をＬＬＲ値に変換する。対数尤度計算プロセッサ１６５４は、ＬＤＰＣシンボル内のビットのＬＬＲ値を再計算することもできる。ＬＤＰＣシンボルではなくビットのＬＬＲ値を変換することは、図３のトレリス・デコーダ３２０やリード・ソロモン・デコーダ３８０などのデコーダ内の以前または後続のブロックに新たなＬＬＲ値を供給するために必要になることがある。

【０１７９】

図１７は、本発明の特徴を用いたバイト・コード符号化信号を復号するプロセスの実施例を示す流れ図である。プロセス１７００は、主に要素バイト・コード・デコーダ１３００に関連して説明する。ただし、プロセス１７００は、要素バイト・コード・デコーダ６００などその他のバイト・コード・デコーダにも適用可能であり、デコーダ３００など複数のデコーダを含むさらに大きなプロセスの一部として使用することもできることに留意されたい。

【０１８０】

プロセス１７００は、最初に、ステップ１７１０で、システマティック・バイト・コード符号化データおよび非システマティック・バイト・コード符号化データのパケットまたはバイトを含むビットストリームとして、バイト・コード符号化メッセージを受信する。

受信したメッセージ・ビットストリームは、図１のＧＦ（２５６）ＳＣＢＣ１１４について述べたプロセスなど、線形ブロック符号化プロセスを用いた信号の符号化および伝送中に、バイト・コード符号化される。メッセージ・ビットストリームは、メッセージ中の各ビットのビット値を含む、あるいは好ましくは、ビット値の確率もしくは信頼度、またはシンボルなどのビット・グループの値の確率を含むことができる。

【０１８１】

次に、ステップ１７２０で、入来メッセージをサブセットに配列する。入来メッセージ中のビットをサブセットにする配列は、主に使用する復号プロセスに基づいて決定することができる。前述のように、バイト・コード復号では、通常、ビット連続評価プロセスを行う。復号プロセスは、反復評価によって改善することができる。バイト・コード符号化信号の復号は、さらにガロア体の性質に基づいてバイト・コード符号化プロセスに関連する性質を利用することにより、さらに改善することができる。好ましい実施例では、入来メッセージを、ＬＤＰＣパリティ・チェックプロセスを用いたパリティ・チェック行列の一部として処理される８ビットのサブセットに配列する。この配列は、パリティ・チェックプロセスにおける必要に応じて、サブセット中のビットの並べ替えまたは再配列をさらに含むこともあることに留意されたい。

【０１８２】

次に、ステップ１７３０で、選択したサブセットの性質およびバイト・コード符号化プロセスの性質に基づいて選択した復号プロセスを用いて、これらのサブセットを復号する。好ましい実施例では、原始多項式およびＧＦ（２）のパリティ行列の生成に基づいて、ＬＤＰＣパリティ・チェック復号プロセスが選択される。復号効率を向上させるために、生成行列値の逆行列を用いてＧＦ（２５６）空間で入来メッセージを復号するのではなく、ＧＦ（２）におけるＬＤＰＣパリティ・チェック復号プロセスを使用する。この復号ステップ１７３０は、サブセット中のメッセージ・ビットの現状態の信頼度のセットを計算することを含むこともある。メッセージ中のコード化冗長ビットについても確率を求め、これらを、復号ステップ１７３０で使用する別のパリティ・チェックプロセスなどのそれ以前の復号プロセス、またはそれ以前のトレリス復号プロセスにおいてフィードバック信号として使用できるようにする。

【０１８３】

次に、ステップ１７４０で、復号プロセスが適切にメッセージを決定し、復号したかどうかを判定する。この判定は、計算した確率を、所定のしきい値のセットと突き合わせて比較することを含む。ステップ１７４０で復号プロセスが完了していない場合には、プロセスはステップ１７３０に戻り、もう一度復号ステップを行う。ステップ１７４０で復号プロセスが完了している場合には、ステップ１７５０で、この最終値を復号プロセスから出力する。好ましい実施例では、最終値は、要素バイト・コード・デコーダ１３００の出力となる。最終値は、硬判定の値またはビット値として出力することもできるし、軟判定の確率値として出力することもできる。これらの出力を、別のバイト・コード・デコーダやリード・ソロモン・デコーダなど、さらに別のデコーダに供給することもできる。これらの出力は、トレリス復号をさらに改善するために、トレリス・デコーダなどの以前の復号ステージに戻すこともできる。ステップ１７５０は、出力メッセージ中のビットの適切な順序を回復するために、サブセット中のビットの並べ替えまたは再配列を含むこともできる。ステップ１７３０からステップ１７４０のプロセスは、復号が完了するまで、または復号ステップを完了するために割り当てられた特定の時間を超えるまで、反復することができることに留意されたい。

【０１８４】

図１８は、別の実施例として、本発明の特徴を用いたバイト・コード符号化信号を復号するプロセス１８００を示す流れ図である。プロセス１８００は、主に図６の要素バイト・コード・デコーダ６００に関連して説明する。ただし、プロセス１８００は、その他のバイト・コード・デコーダにも適用可能であり、図３のデコーダ３００など複数のデコーダを含むさらに大きなプロセスの一部として使用することもできることに留意されたい。

【０１８５】
プロセス１８００では、最初に、ステップ１８１０で、システマティック情報および非システマティック情報を含むビットストリームとして符号化メッセージを受信する。メッセージ・ビットストリームは、メッセージ中の各ビットのビット値を含む、あるいは好ましくは、ビット値の確率もしくは信頼度、またはシンボルなどのビット・グループの値の確率を含むことができる。メッセージは、受信したトレリス変調シンボル、およびシンボルとビットストリーム中のビットとの関係についての情報を含むこともできる。
【０１８６】
次に、ステップ１８２０で、入来メッセージ・ストリームをサブセットに配列する。前述のように、バイト・コード符号化メッセージなどのブロック符号化メッセージの復号では、通常は、ビット連続評価プロセスを行う。復号プロセスは、反復評価によってしばしば改善することができる。バイト・コード符号化信号の復号は、符号化プロセスに関連する性質を利用することにより、さらに改善することができる。好ましい実施例では、入来メッセージを、１６ビットのサブセットに配列し、ＢＣＪＲアルゴリズムなどの反復トレリス・ツリーに基づくアルゴリズムを用いて復号する。
【０１８７】
次に、ステップ１８３０で、識別されたシンボル関係に基づいてビット・サブセットをさらにグループ化する。固有の伝送シンボルの関係に基づくビットのグループ化により、復号プロセスをさらに改善するために使用することができる追加のイントリンシック情報が得られる。ステップ１８４０で、トレリス復号プロセスを最適化するために、シンボルと、その元になったビットとを並べ替える。可能なシンボル並べ替えは、前述の図９および図１０に示してある。
【０１８８】
次に、ステップ１８５０で、前述のＢＣＪＲアルゴリズムなどのシンボル・ベースのトレリス復号アルゴリズムを用い、さらに選択したサブセットの性質および符号化プロセスの性質に基づいて選択した１組のパリティ方程式のうちの１つまたは複数のパリティ方程式を用いて、サブセット中のシンボルを復号する。好ましい実施例では、パリティ・チェック方程式の組は、ＧＦ（２）のパリティ行列を用いて生成される。前述のように、復号中には、さらに小さく、場合によっては変化するビット・サブセットと、１つのパリティ方程式とを１度に使用することができることもある。復号ステップ１８５０は、サブセット中のメッセージ・ビットの１組の信頼度を計算することを含むこともある。メッセージ中のコード化冗長ビットについても確率を求め、これらを、それ以前の復号プロセスにおいてフィードバック信号として使用できるようにする。
【０１８９】
次に、ステップ１８６０で、復号プロセスが適切にメッセージを決定し、復号したかどうかを判定する。この判定は、計算した確率を、所定のしきい値のセットと突き合わせて比較することを含む。ステップ１８６０で復号プロセスが完了していない場合には、プロセスはステップ１８５０に戻り、もう一度復号ステップを行う。ステップ１８６０で復号プロセスが完了している場合には、ステップ１８７０で、この最終値を復号プロセスから出力する。好ましい実施例では、最終値は、要素バイト・コード・デコーダ６００の出力となる。最終値は、硬判定の値またはビット値として出力することもできるし、軟判定の確率値として出力することもできる。これらの出力を、別のバイト・コード・デコーダやリード・ソロモン・デコーダなど、さらに別のデコーダに供給することもできる。これらの出力を、図３のトレリス・デコーダ３２０などの以前の復号ステージに戻して、復号プロセスをさらに改善することもできる。ステップ１８７０は、出力メッセージ中のビットの適切な順序を回復するために、サブセット中のビットの並べ替えまたは再配列を含むこともできる。ステップ１８５０からステップ１８６０のプロセスは、復号プロセスが完了するまで反復することもできるし、割り当てられた特定の復号完了時間に基づいて打ち切ることもできる。
【０１９０】

主にシンボル・ベースのトレリス復号プロセスに関連してプロセス１８００について述べたが、プロセス１８００を使用して、ビット・ベースのトレリス復号プロセスを実施することもできる。ビット・ベースのトレリス復号プロセスを用いてプロセス１８００を実施する場合には、例えば、シンボルとビットの関係の特定に関するステップ１８３０を省略することができる。

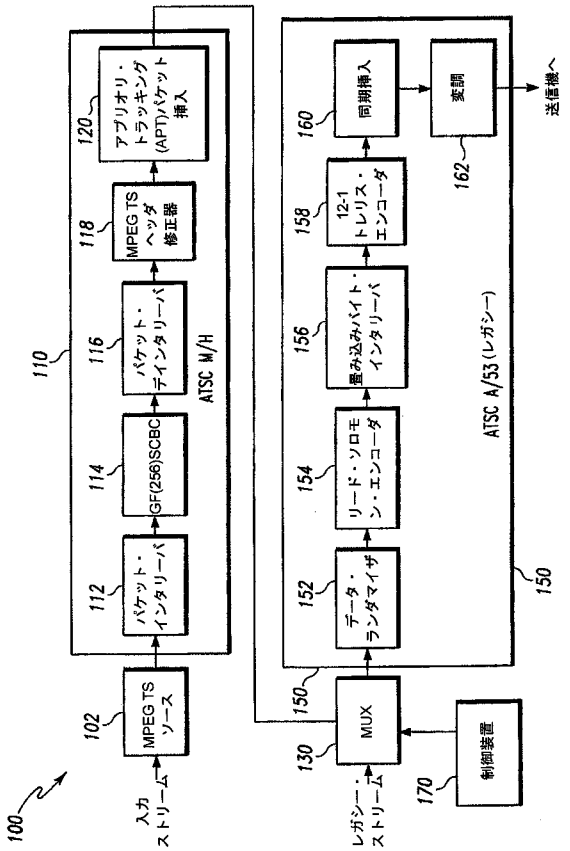
【０１９１】

本開示では、コード・レート１／２または２／３の要素コード・レートを有する特定の直列連結ブロック・コードについて述べている。本開示では、デコーダの効率および性能を向上させるために、非隣接のビットであってもよい入来ビットのグループ化を用いてコードを復号することができるという、特別な要素コード構造の性質を利用している。本開示では、２進ＬＤＰＣプロセス、非２進ＬＤＰＣプロセス、ビット・トレリス・マッピング・プロセス、および変調シンボル・トレリス・マッピング・プロセスなど、いくつかの可能なビット・グループ化の手法について述べている。

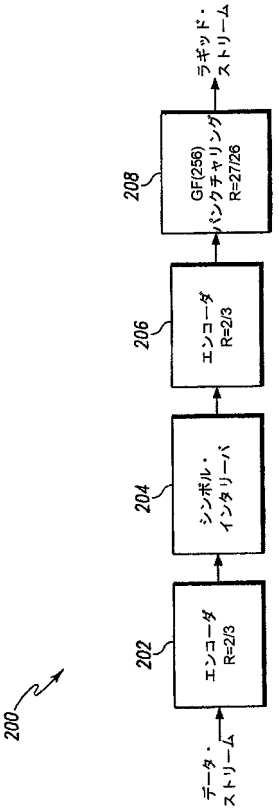
【０１９２】

実施例は、様々な修正を加えることが可能であり、また様々な代替形態を持つことも可能であるが、例示を目的として、具体的な実施例を図面に示し、本明細書で詳細に説明した。ただし、本開示は、開示した特定の形態に限定されるものではないことを理解されたい。また、本開示は、以下の添付の特許請求の範囲によって定義される本開示の趣旨および範囲に含まれる全ての修正形態、均等物、および代替物を含むものとする。

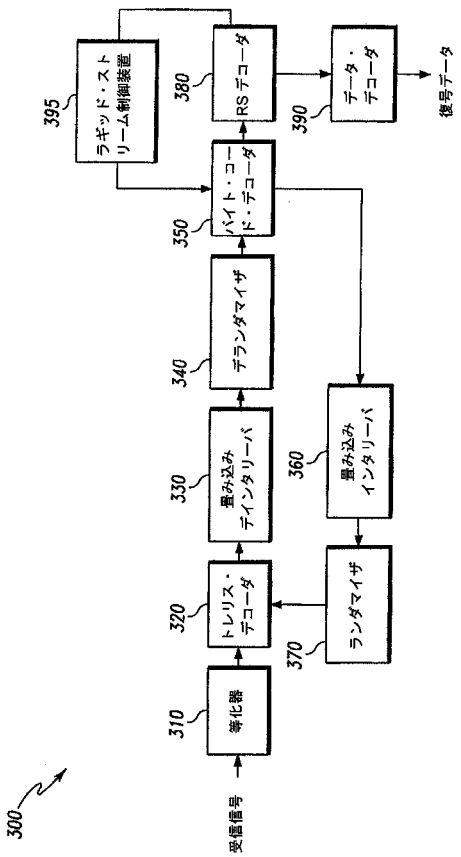
【図１】



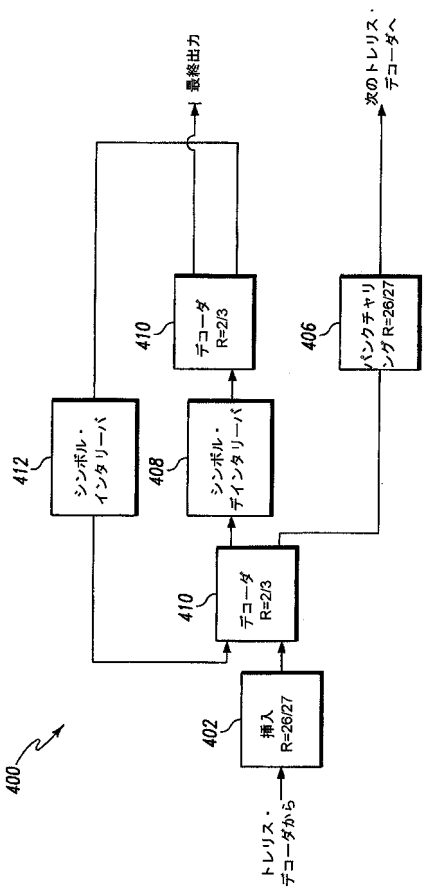
【図２】



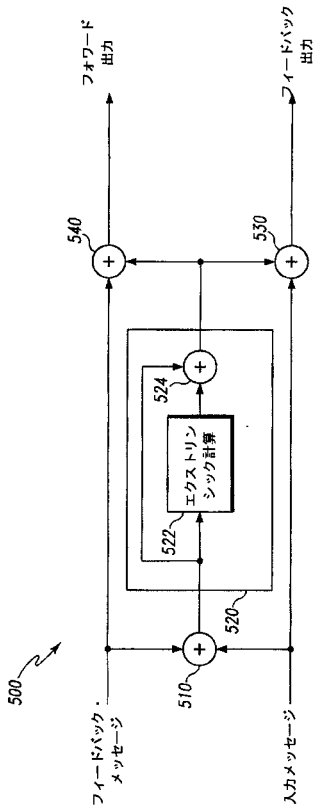
【図３】


300
受信信号
310
等化器
320
トレリス・デコーダ
330
畳み込みデインタリーバ
340
デランダマイザ
350
バイト・コード・デコーダ
360
畳み込みインタリーバ
370
ランダマイザ
380
RSデコーダ
390
データ・デコーダ
復号データ
395
ラギッド・ストリーム制御装置

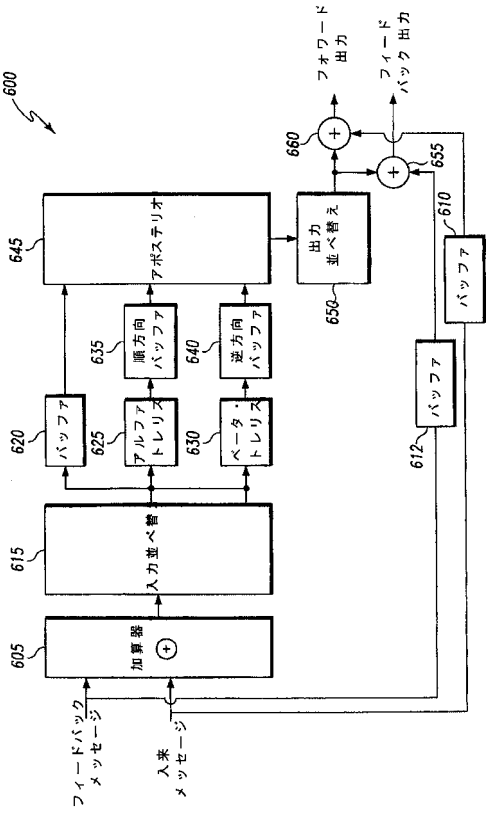
【図４】


400
トレリス・デコーダから
402
挿入 R=26/27
410
デコーダ R=2/3
408
シンボル・デインタリーバ
412
シンボル・インタリーバ
406
バンクチャリング R=26/27
最終出力
次のトレリス・デコーダへ

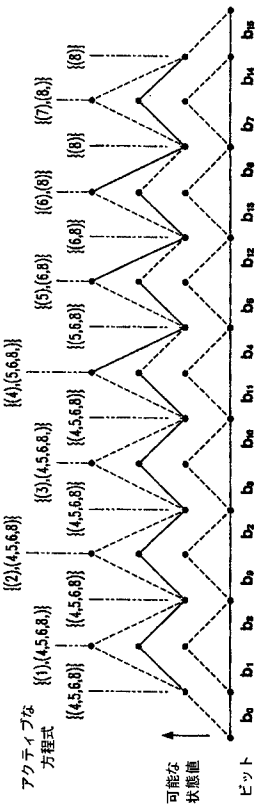
【図５】


500
フィードバック・メッセージ
入力メッセージ
510
520
522
エクストリンシック計算
524
540
530
フォワード出力
フィードバック出力

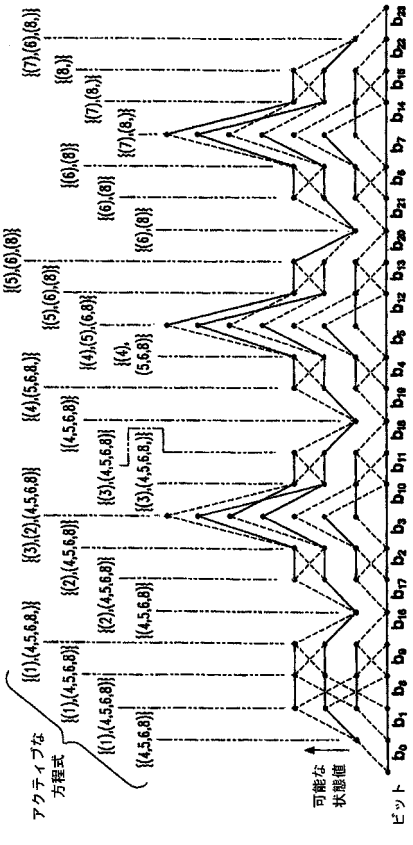
【図６】


600
フィードバック・メッセージ
入来メッセージ
605
加算器
615
入力並べ替え
620
バッファ
625
アルファ・トレリス
630
ベータ・トレリス
635
順方向バッファ
640
逆方向バッファ
645
アポステリオ
650
出力並べ替え
660
フォワード出力
655
フィードバック出力
610
バッファ
612
バッファ

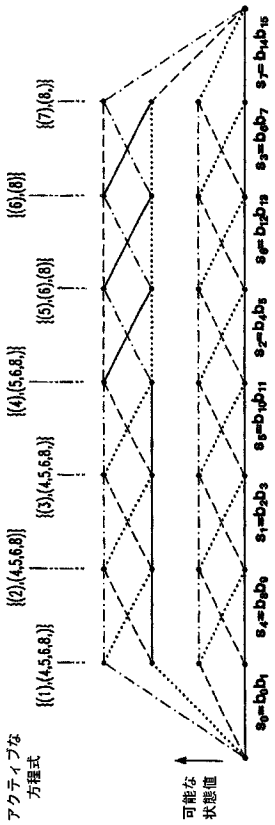
【図７】


アクティブな方程式
可能な状態値
ビット

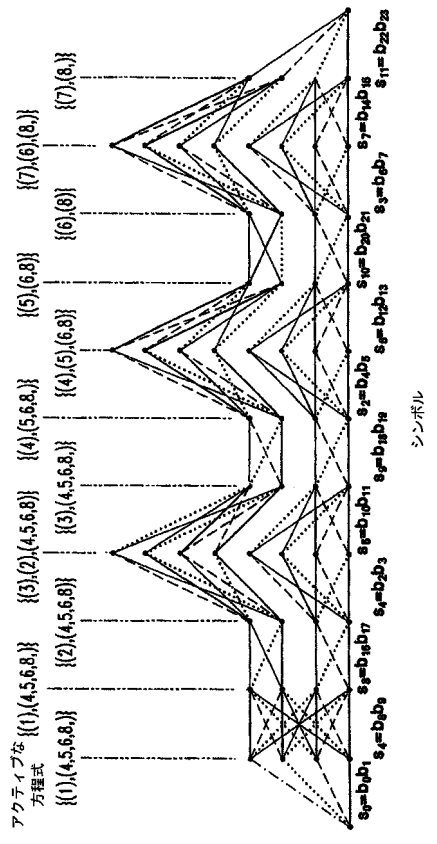
【図８】


アクティブな方程式
可能な状態値
ビット

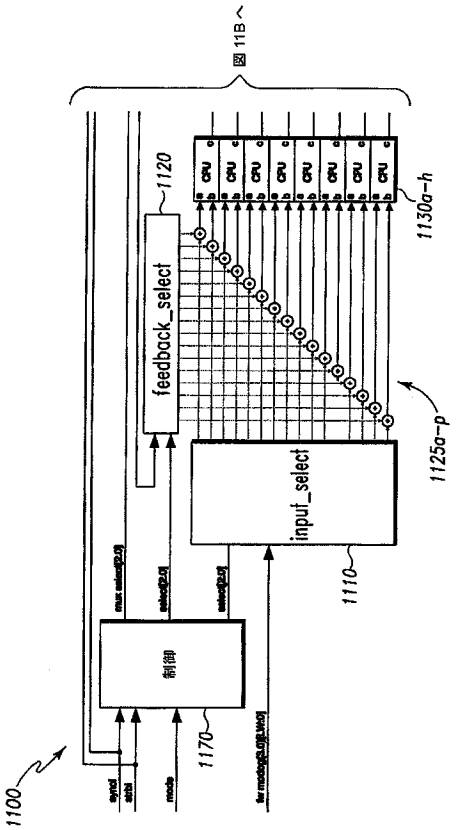
【図９】


アクティブな方程式
可能な状態値

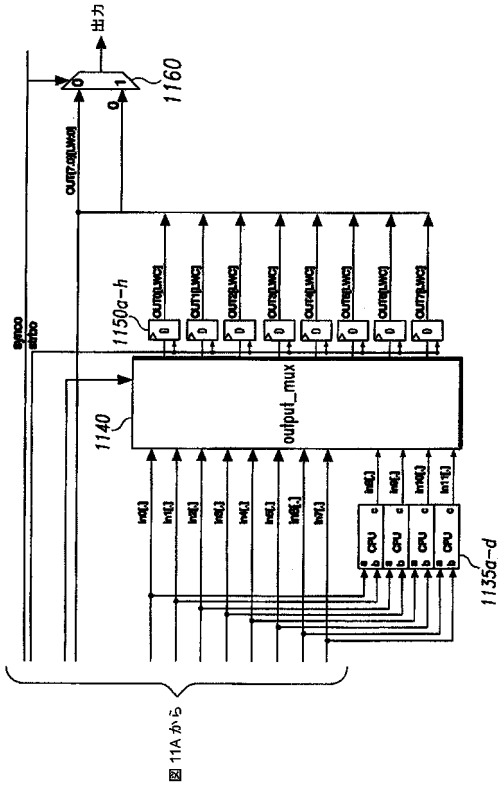
【図１０】


アクティブな方程式
シンボル

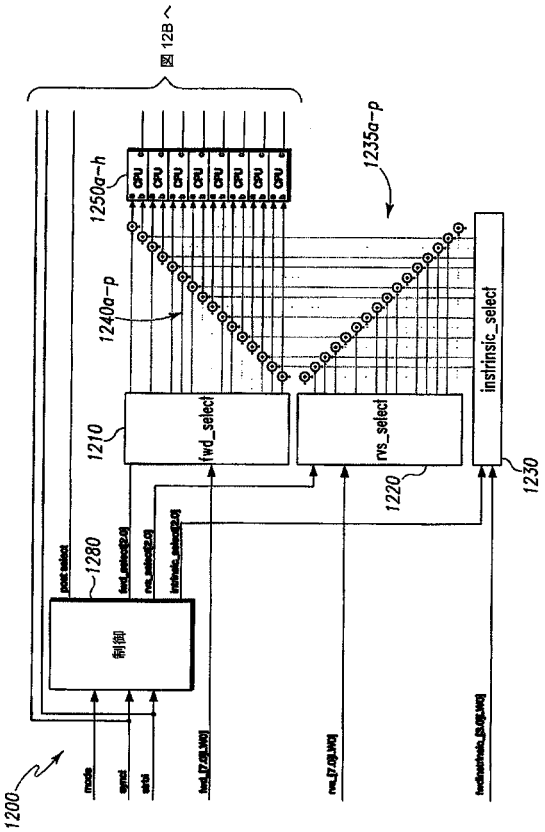
【図１１Ａ】



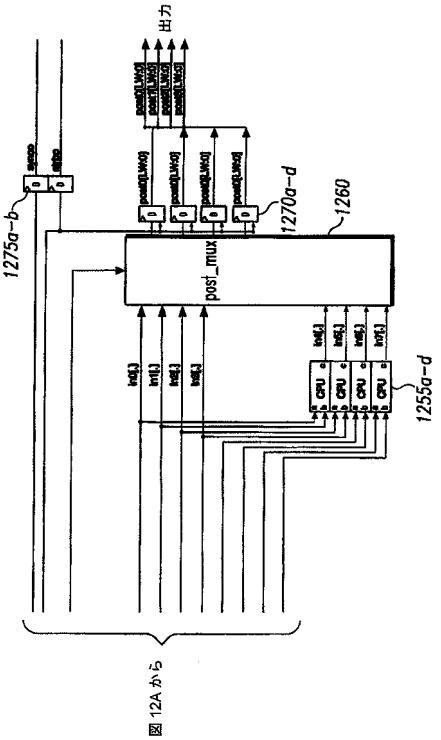
【図１１Ｂ】



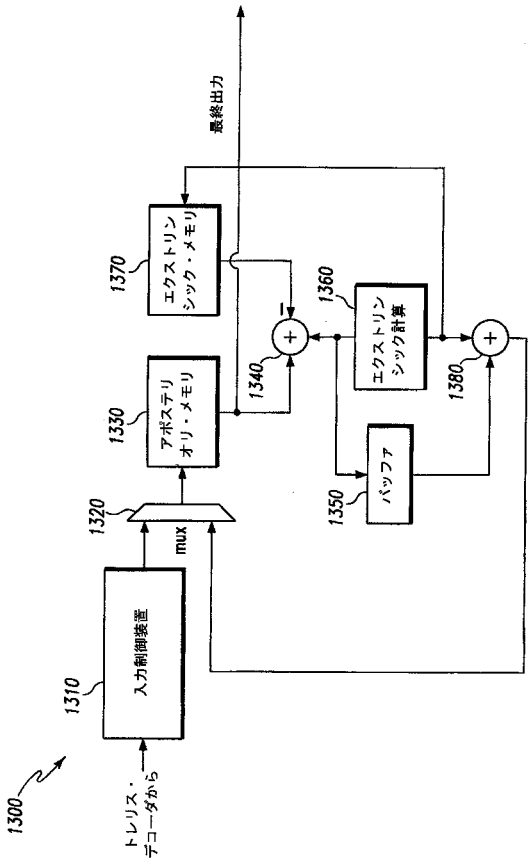
【図１２Ａ】



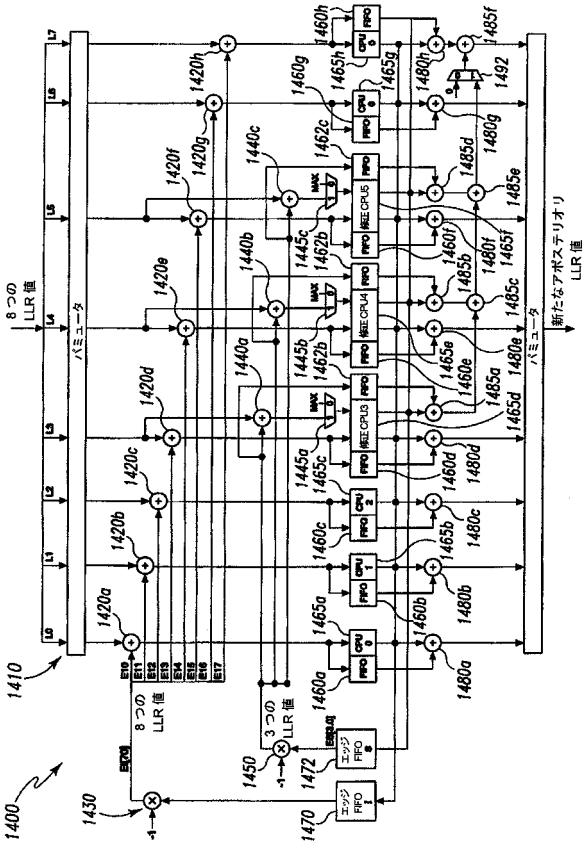
【図１２Ｂ】



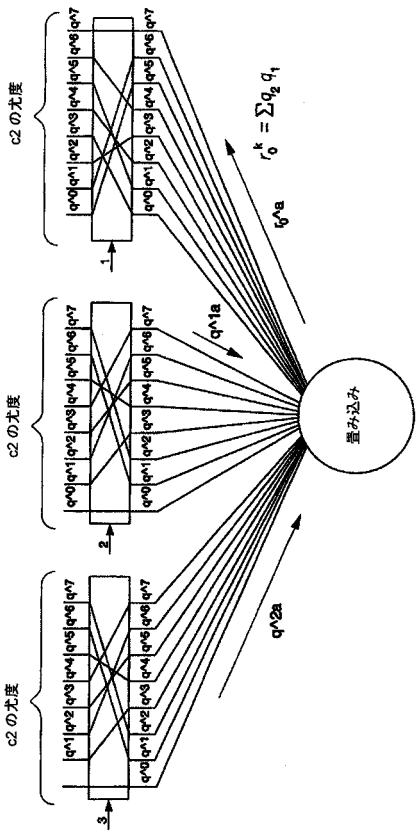
【図１３】



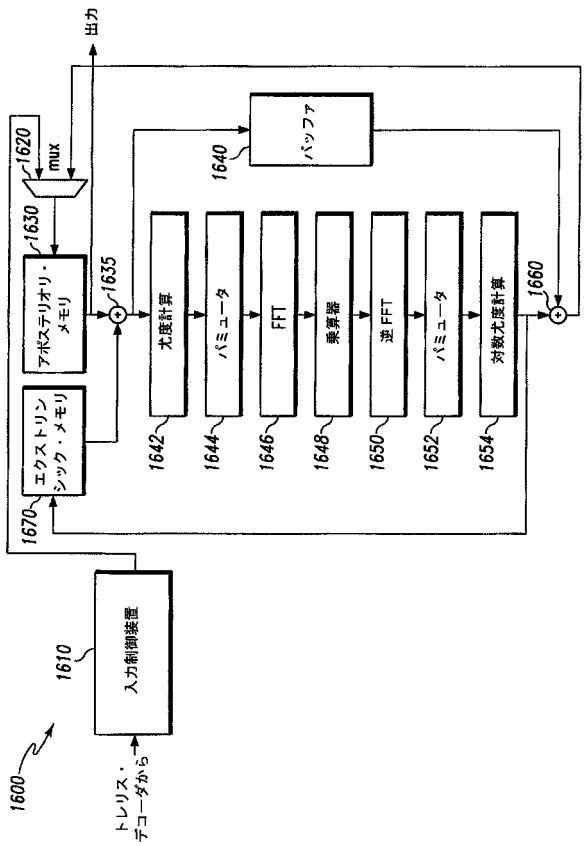
【図１４】



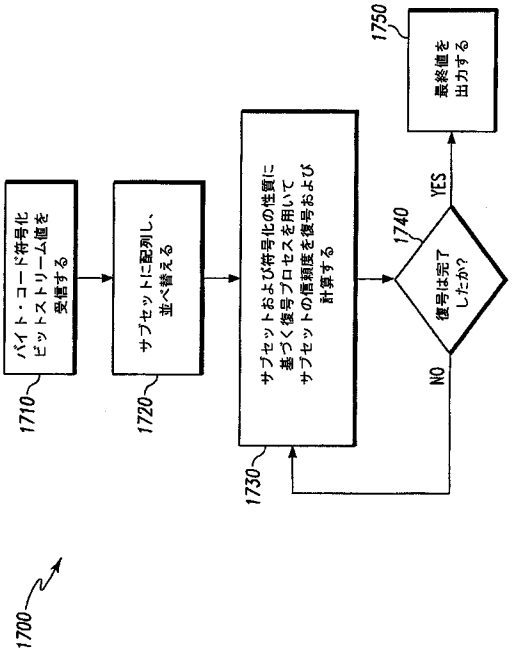
【図１５】



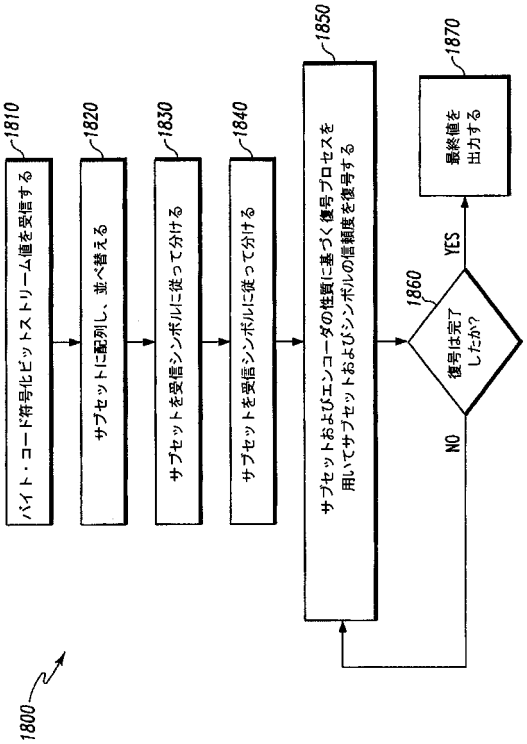
【図１６】



【図１７】


バイト・コード符号化ビットストリーム値を受信する
サブセットに配列し、並べ替える
サブセットおよび符号化の性質に基づく復号プロセスを用いてサブセットの信頼度を復号および計算する
復号は完了したか?
YES
NO
最終値を出力する
1700
1710
1720
1730
1740
1750

【図１８】


バイト・コード符号化ビットストリーム値を受信する
サブセットに配列し、並べ替える
サブセットを受信シンボルに従って分ける
サブセットを受信シンボルに従って分ける
サブセットおよびエンコーダの性質に基づく復号プロセスを用いてサブセットおよびシンボルの信頼度を復号する
復号は完了したか?
YES
NO
最終値を出力する
1800
1810
1820
1830
1840
1850
1860
1870

フロントページの続き

(74)代理人 100123629
弁理士 吹田 礼子
(72)発明者 ミユラー, シユテフアン
ドイツ国 ７８０４８ フイリンゲン－シユベニンゲン フオルツハイマー・シユトラツセ １５／１
(72)発明者 クラフチエンコ, アレクサンダー
ドイツ国 ７８０５６ フイリンゲン－シユベニンゲン ドイテンベルグリング １４６
(72)発明者 クリンケ, シユタニスロウ
ポーランド国 ７８０５２ タンハイム イム・ホールヴインケル ７
(72)発明者 ロプレスト, スコット
アメリカ合衆国 イリノイ州 シカゴ オークリー・ブールバード アパートメント ＃１ １３２９ エヌ

審査官 岡 裕之

(56)参考文献 特開２００７－２２８５８８（ＪＰ，Ａ）
特開２００７－００６４９４（ＪＰ，Ａ）
特表２０１０－５２７５６０（ＪＰ，Ａ）
特開２００３－２５８７７６（ＪＰ，Ａ）
米国特許第７２６６７４９（ＵＳ，Ｂ１）
Sheng Tong et al., On short forward error-correcting codes for wireless communication systems, Faculty of Engineering and Information Sciences Papers, University of Wollongong, ２００７年
Sheng Tong et al., On the Performance of Short Forward Error-Correcting Codes, IEEE Communications Letters, ２００７年１１月, Vol.11, No.11, pp.880-882
Young Ho Oh et al., A Recursive Trellis Decoder to A pproach The Shannon Capacity in ATSC DTV Receivers, Consumer Electronics, 2008. ICCE 2008. Digest of Technical Papers. International Conference on, ２００８年 1月１３日
ATSC Digital Television Standard Part 2 - RF/Transmission System Characteristics, Advanced Television Systems Committee, ２００７年 1月 3日
伊藤 修朗 外２名, 誤り訂正能力を付加した直流成分抑圧方式, テレビジョン学会技術報告, １９９５年１２月 8日, Vol.19, No.71, pp.59-66, VIR95-100
北神 正人 外２名, バイト誤り訂正符号における並列復号法, １９９５年電子情報通信学会基礎・境界ソサイエティ大会講演論文集, １９９５年 8月１５日, p.117, A-114
松岡 秀浩 外３名, ディジタル移動通信における連接符号化適応変調システムの伝送特性解析, 電子情報通信学会技術研究報告, １９９６年 2月２６日, Vol.95, No.535, pp.85-90, RCS95-155
Rong Zhou et al., Reliable transmission with low complexity Reed-Solomon block turbo codes, Wireless Communication Systems, 2004, 1st International Symposium on, ２００４年 9月２２日, pp.193-197

(58)調査した分野(Int.Cl., ＤＢ名)
Ｈ０３Ｍ １３／１１
Ｈ０３Ｍ １３／２５
ＩＥＥＥ Ｘｐｌｏｒｅ
ＣｉＮｉｉ