(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2014-99944
(P2014-99944A)
(43)公開日　平成26年5月29日(2014.5.29)

| (51)Int.Cl. | F I | テーマコード（参考） |
|---|---|---|
| ***H03M 13/19 (2006.01)*** | H03M 13/19 | 5J065 |

審査請求 有　請求項の数 6　OL　（全 21 頁）

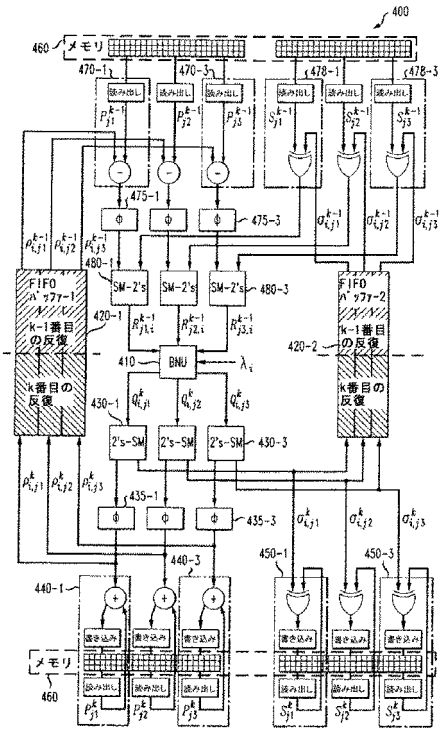
(21)出願番号　特願2014-44504(P2014-44504)
(22)出願日　平成26年3月7日(2014.3.7)
(62)分割の表示　特願2007-198612(P2007-198612)の分割
原出願日　平成19年7月31日(2007.7.31)
(31)優先権主張番号　11/496,121
(32)優先日　平成18年7月31日(2006.7.31)
(33)優先権主張国　米国(US)

(71)出願人　500587067
アギア　システムズ　インコーポレーテッド
アメリカ合衆国．１８１０９　ペンシルヴァニア，アレンタウン，アメリカン　パークウェイ　エヌイー　１１１０
(74)代理人　100094112
弁理士　岡部　讓
(74)代理人　100106183
弁理士　吉澤　弘司
(72)発明者　エリック　エフ．　ハラッシュ
アメリカ合衆国　１８０１７　ペンシルヴァニア　ベツレヘム，　バーバリー　ストリート　５１０５

最終頁に続く

(54)【発明の名称】ハードウェア共用および直列和積アーキテクチャを用いる低密度パリティ検査復号の方法および装置

(57)【要約】
【課題】相互に接続されたビット・ノードおよび検査ノードを有する２部グラフを用いて記述されることが可能な符号を復号する方法および装置を提供する。
【解決手段】検査ノードｊからビット・ノードｉへの、検査ノードからビット・ノードへのメッセージの絶対値が計算され、この絶対値は、ビット・ノードｉおよび検査ノードｊについての、ビット・ノードから検査ノードへのメッセージの変換された絶対値を除く、検査ノードｊに接続された複数のビット・ノードについての、ビット・ノードから検査ノードへのメッセージの変換された絶対値の和に基づく。検査ノードｊからビット・ノードｉへの、検査ノードからビット・ノードへのメッセージの正負符号を計算することも、ビット・ノードから検査ノードへのメッセージの正負符号の積$S_j$と、ビット・ノードｉおよび検査ノードｊについての、ビット・ノードから検査ノードへのメッセージの正負符号とを掛け合わせることにより、可能である。
【選択図】図４



【特許請求の範囲】
【請求項１】
相互に接続されたビット・ノードおよび検査ノードを有する２部グラフを用いて記述されることが可能な符号を復号する方法であって、
検査ノードｊからビット・ノードｉへの、検査ノードからビット・ノードへのメッセージの絶対値を計算するステップであって、前記絶対値は、ビット・ノードｉおよび検査ノードｊについての、ビット・ノードから検査ノードへのメッセージの変換された絶対値を除く、前記検査ノードｊに接続された複数のビット・ノードについての、ビット・ノードから検査ノードへの前記メッセージの変換された絶対値の和に基づき、前記変換された絶対値の各々がビット・ノードから検査ノードへのメッセージの対応する絶対値を当てはめられる非線形関数の結果を含むステップと、
前記検査ノードｊに接続された複数のビット・ノードの中の、ビット・ノードから検査ノードへのメッセージの正負符号の積$S_j$と、ビット・ノードｉおよび検査ノードｊについての、ビット・ノードから検査ノードへの前記メッセージの符号とを掛け合わせることにより、検査ノードｊからビット・ノードｉへの、検査ノードからビット・ノードへの前記メッセージの正負符号を計算し、１番目のビット・ノード更新ユニットに接続される複数の並列の検査ノード更新ユニットが前記検査ノードからビット・ノードへのメッセージを計算するステップとを含む方法。
【請求項２】
相互に接続されたビット・ノードおよび検査ノードを有する２部グラフを用いて記述されることが可能な符号を復号する復号器であって、
１番目のビット・ノード更新ユニットと、
前記ビット・ノード更新ユニットに接続されて、１つまたは複数の変換された絶対値に基づいて、複数の、検査ノードからビット・ノードへのメッセージを計算する、複数の並列の検査ノード更新ユニットと、を備え、前記変換された絶対値の各々がビット・ノードから検査ノードへのメッセージの対応する絶対値を当てはめられる非線形関数の結果を含む復号器。
【請求項３】
ビット・ノードｉと検査ノードｊの間の前記ビット・ノードから検査ノードへのメッセージ$Q_{i,j}$の前記変換された絶対値$\rho_{i,j}$が、

$$r_{i,j} = f\left(\left|Q_{i,j}\right|\right),$$

但し、記号｜｜は絶対値であることを示す記号である、
として計算される、請求項１に記載の方法。
【請求項４】
前記非線形関数が、-log tanh(x/2)および

$$\log\frac{e^x+1}{e^x-1}$$

の内の、いずれか又は両方を含む、請求項１に記載の方法。
【請求項５】
ビット・ノードｉと検査ノードｊの間の前記ビット・ノードから検査ノードへのメッセージ$Q_{i,j}$の前記変換された絶対値$\rho_{i,j}$が、

$$r_{i,j} = f\left(\left|Q_{i,j}\right|\right),$$

但し、記号｜｜は絶対値であることを示す記号である、
として計算される、請求項２に記載の復号器。
【請求項６】
前記非線形関数が、-log tanh(x/2)および

$$\log\frac{e^x+1}{e^x-1}$$

の内の、いずれか又は両方を含む、請求項２に記載の復号器。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、主に符号化および復号の技術に関し、特に、符号化された信号の復号に用いられる復号器のアルゴリズムおよびアーキテクチャに関する。

【背景技術】

【０００２】

符号化は、デジタル情報の伝送において、ビットおよびパケットのエラー確率を低減するために広く用いられている。多くの用途において、この目的のために畳み込み符号が用いられている。畳み込み符号は、符号化されるべき入力ビットによって状態遷移が決定される状態機械により定義される。畳み込み符号、または畳み込み符号をベースとする符号を復号するアルゴリズムとして最も一般的な２つが、ビタビ・アルゴリズムおよび最大事後確率（ｍａｘｉｍｕｍ　ａ－ｐｏｓｔｅｒｉｏｒｉ　ｐｒｏｂａｂｉｌｉｔｙ）（ＭＡＰ）アルゴリズムである。

【０００３】

より強力なエラー訂正能力を必要とする用途では、ターボ符号またはＬＤＰＣ符号が使用されることが多い。連接符号（ターボ符号など）の復号は、ある符号トレリスの復号の結果を次の符号トレリスの復号の入力とし、その後の符号トレリスについても同様にすることが必要である。ターボ符号の復号アルゴリズムの反復性は、実装に関して重大な課題を提示する。

【０００４】

ＬＤＰＣ符号は、前途有望な次世代エラー訂正符号であり、符号化利得に関してはターボ符号をしのぐ可能性がある。ＬＤＰＣ符号の並列または直列の復号に関しては、多くの技術が提案されている。たとえば、Ａ．Ｊ．Ｂｌａｎｋｓｂｙ、Ｃ．Ｊ．Ｈｏｗｌａｎｄ、「Ａ　６９０－ｍＷ　１－Ｇｂ／ｓ　１０２４－ｂ，Ｒａｔｅ－１／２　Ｌｏｗ－Ｄｅｎｓｉｔｙ　Ｐａｒｉｔｙ－Ｃｈｅｃｋ　Ｄｅｃｏｄｅｒ」、ＩＥＥＥ　Ｊ．Ｓｏｌｉｄ－Ｓｔａｔｅ　Ｃｉｒｃｕｉｔｓ、Ｖｏｌ．３７、４０４～４１２頁（２００２年３月）、および米国特許第６，５３９，３６７号（Ｂｌａｎｋｓｂｙら）には、低密度パリティ検査（Ｌｏｗ　Ｄｅｎｓｉｔｙ　Ｐａｒｉｔｙ　Ｃｈｅｃｋ）（ＬＤＰＣ）符号のブロック並列復号についての記述がある。

【０００５】

ＬＤＰＣ復号器の直列実装は、一部のハードウェアを共用できるので、並列の復号方法と比較して、必要な計算素子の数を少なくすることが可能であり、また、必要なルーティング機構を簡単にすることが可能である。しかしながら、直列方法は、２部グラフのエッジに沿うすべてのメッセージを保存する必要があるために、一般に追加メモリを必要とし、また、直列実装は、符号のブロックを復号するために、より多くのクロック・サイクルを費やすので、スループットが制限される。一般に、２部グラフは、ＬＤＰＣ符号で用いられるパリティ検査行列のグラフィカル表現である。典型的な必要メモリ量は、符号内のエッジの数の２倍に比例し、サイクル数は、一般に、符号内のビット・ノードおよび検査ノードの数の和に比例する。

【０００６】

たとえば、Ｚｉｎｉｎｇ　ＷｕとＧｒｅｇｏｒｙ　Ｂｕｒｄは、ＬＤＰＣ復号器がチャネル検出器と連結される通信システムのための直列アーキテクチャを提案している。この開示されたアーキテクチャは、従来の直列実装より必要メモリ量が少ない。Ｚ．Ｗｕ、Ｇ．Ｂｕｒｄ、「Ｅｑｕａｔｉｏｎ　Ｂａｓｅｄ　ＬＤＰＣ　Ｄｅｃｏｄｅｒ　ｆｏｒ　Ｉｎｔｅｒｓｙｍｂｏｌ　Ｉｎｔｅｒｆｅｒｅｎｃｅ　Ｃｈａｎｎｅｌｓ」、ＩＥＥＥ　Ｐ

roc. of Acoustics, Speech, and Signal Processing 2005（ICASSP '05）、Vol. 5, 757～60頁、（2005年3月18～23日）を参照されたい。D. E. Hocevarは、やはり従来の直列実装より必要メモリ量が少ないスタンドアロンLDPC復号器のための直列アーキテクチャを提案している。D. E. Hocevar、「LDPC Code Construction With Flexible Hardware Implementation」、IEEE Int'l Conf. on Comm.（ICC）, Anchorage, AK、2708～2712頁（2003年5月）を参照されたい。

【0007】

必要メモリ量または一反復当たりのサイクル数（またはその両方）、ならびにビット・エラー・レート性能において、並列および直列実装の両方に対して、さらなる改善を示す直列LDPC復号アーキテクチャが、スタンドアロンLDPC復号器についても、連結LDPC復号器についても、依然として必要とされている。

【先行技術文献】

【特許文献】

【0008】

【特許文献1】米国特許第6, 539, 367号（Blanksbyら）

【非特許文献】

【0009】

【非特許文献1】A. J. Blanksby、C. J. Howland、「A 690-mW 1-Gb/s 1024-b, Rate-1/2 Low-Density Parity-Check Decoder」、IEEE J. Solid-State Circuits、Vol. 37、404～412頁（2002年3月）

【非特許文献2】Z. Wu、G. Burd、「Equation Based LDPC Decoder for Intersymbol Interference Channels」、IEEE Proc. of Acoustics, Speech, and Signal Processing 2005（ICASSP '05）、Vol. 5, 757～60頁、（2005年3月18～23日）

【非特許文献3】D. E. Hocevar、「LDPC Code Construction With Flexible Hardware Implementation」、IEEE Int'l Conf. on Comm.（ICC）, Anchorage, AK、2708～2712頁（2003年5月）

【非特許文献4】E. Yeoら、「VLSI Architectures for Iterative Decoders in Magnetic Recording Channels」、IEEE Trans. On Magnetics、Vol. 37、No. 2、748～755頁（2001年3月）

【発明の概要】

【課題を解決するための手段】

【0010】

主として、相互に接続されたビット・ノードおよび検査ノードを有する2部グラフを用いて記述されることが可能な符号（LDPC符号など）を復号する方法および装置を提供する。2部グラフは、たとえば、パリティ検査行列のグラフィカル表現であることが可能である。本発明の一態様によれば、検査ノードjからビット・ノードiへの、検査ノードからビット・ノードへのメッセージの絶対値が計算され、この絶対値は、ビット・ノードiおよび検査ノードjについての、ビット・ノードから検査ノードへのメッセージの変換された絶対値を除く、検査ノードjに接続された複数のビット・ノードについての、ビット・ノードから検査ノードへのメッセージの変換された絶対値の和に基づく。検査ノードjからビット・ノードiへの、検査ノードからビット・ノードへのメッセージの正負符号を計算することも、検査ノードjに接続された複数のビット・ノードの中の、ビット・ノードから検査ノードへのメッセージの正負符号の積$S_j$と、ビット・ノードiおよび検査

ノードｊについての、ビット・ノードから検査ノードへのメッセージの正負符号とを掛け合わせることにより、可能である。

【００１１】

本発明の連結実施形態では、検査ノードｊからビット・ノードｉへの、検査ノードからビット・ノードへのメッセージの絶対値が計算され、この絶対値は、ビット・ノードｉについてのアプリオリ情報値の変換された絶対値を除く、検査ノードｊに接続された複数のビット・ノードについてのアプリオリ情報値の変換された絶対値の和に基づく。検査ノードｊからビット・ノードｉへの、検査ノードからビット・ノードへのメッセージの正負符号を計算することも、検査ノードｊに接続された複数のビット・ノードについてのアプリオリ情報値の正負符号の積$S_j$と、ビット・ノードｉについてのアプリオリ情報値の正負符号とを掛け合わせることにより、可能である。これらの計算は、軟出力検出器に連結されたＬＤＰＣ復号器によって実行されることが可能である。アプリオリ情報値は、軟出力検出器によって生成されるか、軟出力検出器によって与えられる軟出力に基づいて計算される。

【００１２】

本発明のさらなる態様によれば、相互に接続されたビット・ノードおよび検査ノードを有する２部グラフを用いて記述されることが可能な符号を復号する復号器アーキテクチャが開示される。本開示の復号器は、スタンドアロン形式で実装されるか、軟出力検出器と連結されることが可能である。本開示の復号器は、ビット・ノード更新ユニットと、ビット・ノード更新ユニットに接続された、複数の並列の、検査ノード更新ユニットとを備えて、複数の、検査ノードからビット・ノードへのメッセージを計算する。

【００１３】

スタンドアロン実施形態では、検査ノード更新ユニットは、ビット・ノードから検査ノードへのメッセージの、ある１つの変換された絶対値を除く、複数の、ビット・ノードから検査ノードへのメッセージの変換された絶対値の和に基づいて、検査ノードからビット・ノードへのメッセージの絶対値を計算する。連結実施形態では、検査ノード更新ユニットは、ビット・ノードのアプリオリ情報値の、ある１つの変換された絶対値を除く、複数のビット・ノードについてのアプリオリ情報値の変換された絶対値の和に基づいて、検査ノードからビット・ノードへのメッセージの絶対値を計算する。

【００１４】

メモリに接続される、いくつかの回路ユニットが開示される。それらの回路ユニットは、加算器、減算器、またはＸＯＲゲートのようなデジタル論理素子と、読み出し回路または読み出し／書き込み回路からなる。本開示の回路ユニットは、スタンドアロン実施形態における、複数の、ビット・ノードから検査ノードへのメッセージの変換された絶対値および正負符号に対して、ならびに、連結実施形態におけるアプリオリ情報値の変換された絶対値および正負符号に対して、様々な動作を実行する。

【００１５】

以下の詳細説明および図面を参照することにより、本発明、ならびに本発明のさらなる特徴および利点について、より深い理解が得られるであろう。

【図面の簡単な説明】

【００１６】

【図１】ＬＤＰＣ行列Ｈの一般構造を示す図である。

【図２】ＬＤＰＣ符号の例示的な２部グラフ表現を示す図である。

【図３】例示的なハードウェア共用ＬＤＰＣ復号器アーキテクチャのブロック図である。

【図４】本発明の機能を組み込んだＬＤＰＣ復号器のブロック図である。

【図５】ＬＤＰＣ復号器が軟入力軟出力（ＳＩＳＯ）検出器と連結された、典型的な通信システムのブロック図である。

【図６】ＳＩＳＯ検出器と連結されたＬＤＰＣ復号器のブロック図である。

【発明を実施するための形態】

【００１７】

本発明は、ハードウェア共用および直列和積（ｓｕｍ－ｐｒｏｄｕｃｔ）アーキテクチャを用いるＬＤＰＣ復号の方法および装置を提供する。本開示のＬＤＰＣ復号器は、従来の実装に比べて少ないメモリおよび少ないクロック・サイクルを用いる和積復号アルゴリズムを実行する。

【００１８】

低密度パリティ検査符号

以下の、ＬＤＰＣ符号およびＬＤＰＣ復号の背景説明は、参照によって本明細書に組み込まれている、Ａ．Ｊ．Ｂｌａｎｋｓｂｙ、Ｃ．Ｊ．Ｈｏｗｌａｎｄ、「Ａ　６９０－ｍＷ　１－Ｇｂ／ｓ　１０２４－ｂ，Ｒａｔｅ－１／２　Ｌｏｗ－Ｄｅｎｓｉｔｙ　Ｐａｒｉｔｙ－Ｃｈｅｃｋ　Ｄｅｃｏｄｅｒ」、ＩＥＥＥ　Ｊ．Ｓｏｌｉｄ－Ｓｔａｔｅ　Ｃｉｒｃｕｉｔｓ、Ｖｏｌ．３７、４０４～４１２頁（２００２年３月）に基づく。詳細については、Ｂｌａｎｋｓｂｙ、Ｈｏｗｌａｎｄの論文の全文を参照されたい。

【００１９】

ＬＤＰＣ符号の行列表現

ＬＤＰＣ符号は、線形ブロック符号である。すべての符号語$x \in C_x$の集合は、次のように、パリティ検査行列Ｈのヌル空間を張る。

$$Hx^T = 0 \quad \forall x \in C_x \qquad (1)$$

ＬＤＰＣ符号のパリティ検査行列は、２値の疎行列である。

【００２０】

図１は、ＬＤＰＣ行列Ｈの一般構造１００を示す。図１に示すように、パリティ検査行列Ｈの各行は、パリティ検査に対応し、集合要素$h_{ji}$は、データ・ビットｉがパリティ検査ｊに関与することを示す。ｎビットのブロックには、ｍ個の冗長パリティ・ビットがある。符号レートは、次式で与えられる。

$$r = (n-m)/n \qquad (2)$$

【００２１】

パリティ検査行列Ｈの行および列の集合要素は、行および列の所望の重みプロファイルを満たすように選択され、行および列の重みは、それぞれ、所与の行および列にある集合要素の数として定義される。正則ＬＤＰＣ符号では、すべての行が一様の重みを有し、すべての列も同様である。行および列が一様の重みでない場合、そのＬＤＰＣ符号は、非正則であると言われる。

【００２２】

ＬＤＰＣ符号のグラフ表現

ＬＤＰＣ符号は、２部グラフでも表現可能であり、その場合は、一方の集合のノードがパリティ検査制約を表し、他方の集合がデータ・ビットを表す。図２は、ＬＤＰＣ符号の例示的な２部グラフ表現２００である。パリティ検査行列は、グラフの接続行列であり、その場合、Ｈのエントリ$h_{ji}$がセットされていれば（非ゼロであれば）、Ｈの列ｉに対応するビット・ノードｉが、Ｈの行ｊに対応する検査ノードｊに接続される。

【００２３】

ＬＤＰＣ符号の復号に用いられるアルゴリズムは、和積（ｓｕｍ－ｐｒｏｄｕｃｔ）アルゴリズムとして知られる。このアルゴリズムで良好な復号性能を得るためには、ＬＤＰＣ符号のグラフ表現におけるサイクルの長さを、可能な限り長くすることが重要である。図２の例示的表現では、長さ４の例示的な短いサイクルが示されている。図２に示された長さ４のサイクルのような短いサイクルは、和積アルゴリズムの性能を低下させる。

【００２４】

和積アルゴリズム

和積アルゴリズムは、ＬＤＰＣ符号を復号する反復アルゴリズムである。和積アルゴリズムは、メッセージ・パッシング・アルゴリズムまたは確率伝搬としても知られる。和積アルゴリズムの詳細については、たとえば、それぞれ参照によって本明細書に組み込まれている、Ａ．Ｊ．Ｂｌａｎｋｓｂｙ、Ｃ．Ｊ．Ｈｏｗｌａｎｄ、「Ａ　６９０－ｍＷ　１－Ｇｂ／ｓ　１０２４－ｂ，Ｒａｔｅ－１／２　Ｌｏｗ－Ｄｅｎｓｉｔｙ　Ｐａｒｉｔｙ

－Ｃｈｅｃｋ Ｄｅｃｏｄｅｒ」、ＩＥＥＥ Ｊ．Ｓｏｌｉｄ－Ｓｔａｔｅ Ｃｉｒｃｕｉｔｓ、Ｖｏｌ．３７、４０４～４１２頁（２００２年３月）、Ｄ．Ｅ．Ｈｏｃｅｖａｒ、「ＬＤＰＣ Ｃｏｄｅ Ｃｏｎｓｔｒｕｃｔｉｏｎ Ｗｉｔｈ Ｆｌｅｘｉｂｌｅ Ｈａｒｄｗａｒｅ Ｉｍｐｌｅｍｅｎｔａｔｉｏｎ」、ＩＥＥＥ Ｉｎｔ’ｌ Ｃｏｎｆ．ｏｎ Ｃｏｍｍ．（ＩＣＣ），Ａｎｃｈｏｒａｇｅ，ＡＫ、２７０８～２７１２頁（２００３年５月）を参照されたい。

【００２５】

ビット・ノードｉから検査ノードｊへのメッセージは、次式で与えられる。

【数１】

$$Q_{i,j} = \sum_{l \in B_i, l \neq j} R_{l,i} + \lambda_i \qquad (3)$$

【００２６】

本明細書で用いられる表記は、本明細書の末尾にある表において定義されている。検査ノードｊからビット・ノードｉへのメッセージは、次式で与えられる。

【数２】

$$R_{j,i} = s_{j,i} \cdot \phi \left( \sum_{l \in C_j, l \neq i} \phi \left( \left| Q_{l,j} \right| \right) \right) \qquad (4)$$

ここで、

【数３】

$$s_{j,i} = \prod_{l \in C_j, l \neq i} sign\left(Q_{l,j}\right); \text{および}$$

$$\phi(x) = -\log \tanh(x/2) = \log \frac{e^x + 1}{e^x - 1}.$$

【００２７】

事後対数尤度比（ｌｏｇ－ｌｉｋｅｌｉｈｏｏｄ ｒａｔｉｏ）（ＬＬＲ）とも呼ばれる、ビットｉについての事後情報値$\Lambda_i$は、次式で与えられる。

【数４】

$$\Lambda_i = \sum_{l \in B_i} R_{l,i} + \lambda_i .$$

【００２８】

ＬＤＰＣ復号器

ＬＤＰＣ符号を復号する和積アルゴリズムを実装する際の重大な課題は、メッセージのパッシングを管理することである。検査ノードおよびビット・ノードの機能性はいずれも比較的単純なので、それらを個々に実現することは、少数のゲートだけで可能である。主たる課題は、機能ノード間のメッセージ・パッシングに必要な帯域幅を実現することである。

【００２９】

ハードウェア共用復号器アーキテクチャ

図３は、例示的なハードウェア共用ＬＤＰＣ復号器アーキテクチャ３００のブロック図である。図３に示すように、一般化されたＬＤＰＣ復号器アーキテクチャ３００は、検査ノードまたはビット・ノードの機能性をそれぞれ実現する多数の機能単位３１０、３２０と、メッセージを格納してグラフ接続性を実現するメモリ構造３５０とを備える。制御ロジック３３０が、メモリ構造３５０の構成を制御する。ハードウェア共用ＬＤＰＣ復号器

アーキテクチャ３００の実装の詳細については、たとえば、Ｅ．Ｙｅｏら、「ＶＬＳＩ Ａｒｃｈｉｔｅｃｔｕｒｅｓ ｆｏｒ Ｉｔｅｒａｔｉｖｅ Ｄｅｃｏｄｅｒｓ ｉｎ Ｍａｇｎｅｔｉｃ Ｒｅｃｏｒｄｉｎｇ Ｃｈａｎｎｅｌｓ」、ＩＥＥＥ Ｔｒａｎｓ．Ｏｎ Ｍａｇｎｅｔｉｃｓ、Ｖｏｌ．３７、Ｎｏ．２、７４８～７５５頁（２００１年３月）を参照されたい。

そのようなハードウェア共用アーキテクチャは、復号器の面積を減らすことが確認されている。

【００３０】

修正されたＬＤＰＣパリティ検査式

本発明は、前述のＬＤＰＣパリティ検査式の各成分を再編成することにより、メモリおよびクロック・サイクルの要件の改善が可能であることを認識している。式（４）で示された検査ノードの計算は、次のように、いくつかの成分に分けることが可能である。

【数５】

$$\rho_{i,j} = \phi\left(\left|Q_{i,j}\right|\right) \qquad (5)$$

ここで、$\rho_{i,j}$は、したがって、ビット・ノードｉと検査ノードｊとの間の、ビット・ノードから検査ノードへのメッセージ$Q_{i,j}$の変換された絶対値であり、「｜｜」は絶対値の表記である。

【００３１】

$\sigma_{i,j} = sign(Q_{i,j})$ （６）

したがって、$\sigma_{i,j}$は、ビット・ノードｉと検査ノードｊとの間の、ビット・ノードから検査ノードへのメッセージ$Q_{i,j}$の正負符号である。

【００３２】

【数６】

$$P_j = \sum_{l \in C_j} \rho_{l,j} \qquad (7)$$

$P_j$は、検査ノードｊに接続されたすべてのビット・ノードについての、ビット・ノードから検査ノードへのメッセージの変換された絶対値の和として、検査ノードｊについて計算される。

【００３３】

【数７】

$$S_j = \prod_{l \in C_j} \sigma_{l,j} \qquad (8)$$

【００３４】

所与の検査ノードｊに対する$S_j$は、したがって、検査ノードｊに接続されたすべてのビット・ノードについての、ビット・ノードから検査ノードへのメッセージの正負符号の積である。

【００３５】

次に、検査ノードｊからビット・ノードｉへのメッセージの絶対値および正負符号は、次式で与えられる。

【数８】

$$\left|R_{j,i}\right| = \phi(P_j - \rho_{i,j}) \qquad (9)$$

$sign(R_{j,i}) = S_j \cdot \sigma_{i,j}$ （１０）

したがって、式（４）による、検査ノードからビット・ノードへのメッセージの従来の計算では、現在のビット・ノードｉが計算から除外されるが（$l \in C_j$、$l \neq i$）、本発

明では、検査ノードｊに接続されたすべてのビット・ノードｌについて、ビット・ノードから検査ノードへのメッセージの変換された絶対値$\rho_{l,j}$の和として、中間値$P_j$が計算され、その後、$P_j$から$\rho_{i,j}$を差し引くことにより、検査ノードｊからビット・ノードｉへのメッセージ$R_{j,i}$の絶対値が計算される。

【００３６】

この例示的実施形態では、ビット・ノードの計算は、２の補数の形で実行され、検査ノードの計算は、符号・絶対値（ｓｉｇｎ　ｍａｇｎｉｔｕｄｅ）（ＳＭ）の形で実行される。

【００３７】

直列和積アーキテクチャを有するＬＤＰＣ復号器

図４は、本発明の機能を組み込んだＬＤＰＣ復号器４００のブロック図を示す。図４に示されたＬＤＰＣ復号器４００の例示的実施形態は、$d_c=3$および$B_i=\{j1, j2, j3\}$の場合の例を示す。要素４３０、４３５、４４０、４５０、４７０、４７５、４７８、および４８０は、並列の検査ノード更新ユニットを備える検査ノード更新ブロックを形成する。各時間サイクルにおいて、１つのビット・ノードに関係する計算が実行される。したがって、ビット・ノード１からビット・ノードｎまでについての計算をすべて実行するには、ｎサイクルかかる。ビット・ノード１は、１番目の時間サイクルの間に計算されるものとする。そして、（ｎ・（ｋ－１）＋ｉ）番目の時間サイクルにおいて、ビット・ノード更新ユニット４１０は、ｋ番目の反復におけるｉ番目のビット・ノードに対応する$d_c$個のメッセージ

【数９】

$$Q_{i,j}^{k}$$

, ｊ∈$B_i$を生成する。これら$d_c$個のメッセージは、ステージ４３０において、２の補数による数表現から、符号・絶対値による数表現に変換される。絶対値部分は、関数φを実行する、並列の変換ユニット４３５－１～４３５－３に送られ、これらのユニット４３５からの出力は、式（５）で定義されるように、

【数１０】

$$\rho_{i,j}^{k}$$

, ｊ∈$B_i$である。

【００３８】

これらの

【数１１】

$$\rho_{i,j}^{k}$$

, ｊ∈$B_i$は、加算器および読み出し／書き込み回路からなる、$d_c$個の変換済み絶対値更新ユニット４４０に供給され、そこで、検査ノードｊに接続されたすべてのビット・ノードにわたる和

【数１２】

$$P_j^k$$

が計算され、後で、次の反復のために

【数１３】

$$P_j^k$$

が取り出される。並列の変換済み絶対値更新ユニット４４０は、メモリ４６０内のｍ個のメモリ要素のうちの任意の$d_c$個の要素にアクセスすることが可能であり、各要素は、ｑビットを格納し、ｑは、この例示的実施形態において

【数１４】

$P_j^k$

を表すために用いられるビットの数である。これらの並列の変換済み絶対値更新ユニット４４０は、反復の最後（すなわち、（ｎ・ｋ）番目の時間サイクル）に、式（７）において定義される

【数１５】

$P_j^k$

，ｊ∈１．．．ｍがｍ個のメモリ要素に格納されるように、関係するメモリ要素を更新する。言い換えると、これらのメモリ要素は、関係するρの値の累積和を保持する。

【００３９】

ｋ番目の反復におけるメモリ要素ｊ内の中間値が、ｉ＝１．．．ｎに対して

【数１６】

$P_j^k(i)$

で与えられる場合、（ｎ・（ｋ－１）＋ｉ）番目の時間インスタンスにおける累積和は、次式で与えられる。

【数１７】

$$P_j^k(i)=\begin{cases}P_j^k(i-1)+\rho_{i,j}^k & j\in B_i \text{ の場合}\\ P_j^k(i-1) & \text{それ以外の場合}\end{cases} \qquad (11)$$

【００４０】

これは、反復の最後（たとえば、（ｎ・ｋ）番目の時間インスタンス）には、次式のようになる。

【数１８】

$$P_j^k=P_j^k(n) \qquad (12)$$

【００４１】

ビット・ノードから検査ノードへのメッセージ

【数１９】

$Q_{i,j}^k$

，ｊ∈Ｂ$_i$の正負符号は、式（６）で定義される

【数２０】

$\sigma_{i,j}^k$

，ｊ∈Ｂ$_i$であって、以下のように処理される。前述の手順と同様に、

【数２１】

$\sigma_{i,j}^k$

，ｊ∈Ｂ$_i$は、別の、ＸＯＲゲートおよび読み出し／書き込み回路からなる、ｄ$_c$個の正負符号更新ユニット４５０の集合に供給される。これらの並列の正負符号更新ユニットは、反復の最後に、式（８）で定義される

【数２２】

$S_j^k$

，ｊ∈１．．．ｍがｍ個のメモリ要素に格納されるように、メモリ４６０内の関係するメモリ要素を更新し、各メモリ要素は、１ビットを格納する。言い換えると、これらのメモ

リ要素は、正負符号ビットσの値の累積和を保持する。積は、ＸＯＲゲートによって得られる。前のパラグラフで説明したように、ｋ番目の反復におけるメモリ要素ｊの中間値が、ｉ＝１．．．ｎに対して

【数２３】

$$S_j^k(i)$$

で与えられる場合、累積積は、次式で与えられる。

【数２４】

$$S_j^k(i)=\begin{cases} S_j^k(i-1)\cdot\sigma_{i,j}^k & j\in B_i \text{の場合} \\ S_j^k(i-1) & \text{それ以外の場合} \end{cases} \qquad (13)$$

【００４２】

これは、反復の最後（たとえば、（ｎ・ｋ）番目の時間インスタンス）には、次式のようになる。

【数２５】

$$S_j^k=S_j^k(n) \qquad (14)$$

【００４３】

各時間サイクルにおいて計算された

【数２６】

$$\rho_{i,j}^k$$

，ｊ∈Ｂ$_i$は、先入れ先出し（ＦＩＦＯ）バッファ４２０－１（図４のバッファ－１）にも送られる。ＦＩＦＯ　４２０－１は、ｄ$_c$・ｑ個の列（この例示的実施形態では、サイクルごとにｄ$_c$個のρの値が生成され、各ρはｑビットで表される）と、ｎ個の行とからなる。時間サイクルｎ・（ｋ－１）＋ｉにおいて、

【数２７】

$$\rho_{i,j}^k$$

，ｊ∈Ｂ$_i$の集合は、ＦＩＦＯ　４２０－１の後部に供給され、前の反復からの

【数２８】

$$\rho_{i,j}^{k-1}$$

，ｊ∈Ｂ$_i$の集合は、バッファ４２０－１の前部から読み出される。（ｎ・ｋ）番目の時間サイクルには、すべての

【数２９】

$$P_j^k$$

，ｊ＝１．．．ｍがメモリ４６０において使用可能であり、すべての

【数３０】

$$\rho_{i,j}^k$$

，ｉ＝１．．．ｎ，ｊ∈Ｂ$_i$が第１のＦＩＦＯバッファ４２０－１において使用可能である。そして、ＦＩＦＯ　４２０－１の最上行が、

【数３１】

$$\rho_{1,j}^k$$

，ｊ∈Ｂ$_1$を保持し、その次の行が、

【数３２】

$\rho_{2,j}^{k}$

，ｊ∈Ｂ$_{2}$を保持し、バッファ４２０－１の最後の行が、

【数３３】

$\rho_{n,j}^{k}$

，ｊ∈Ｂ$_{n}$からなる。

【００４４】

各時間サイクルにおいて計算された

【数３４】

$\sigma_{i,j}^{k}$

，ｊ∈Ｂ$_{i}$は、別のＦＩＦＯバッファバッファ４２０－２（図４のバッファ－２）に供給される。第２のＦＩＦＯバッファ４２０－２は、ｄ$_{c}$個の１ビット列（各σ$_{i,j}$は１ビットで表されるので）と、ｎ個の行とからなる。

【００４５】

時間サイクルｎ・（ｋ－１）＋ｉにおいて、

【数３５】

$\sigma_{i,j}^{k}$

，ｊ∈Ｂ$_{i}$の集合は、ＦＩＦＯ　４２０－２の後部に供給され、前の反復からの

【数３６】

$\sigma_{i,j}^{k-1}$

，ｊ∈Ｂ$_{i}$の集合は、バッファ４２０－２の前部から読み出される。

【００４６】

次に、前の反復ｋ－１からの、メモリ４６０に保存された

【数３７】

$P_{j}^{k-1}$

，ｊ＝１．．．ｍと、第１のＦＩＦＯバッファ４２０－１に保存された

【数３８】

$\rho_{i,j}^{k-1}$

，ｊ∈Ｂ$_{i}$とから、検査ノードからビット・ノードへのメッセージ

【数３９】

$R_{j,i}^{k-1}$

の絶対値の計算の手順を説明する。必要な

【数４０】

$P_{j}^{k-1}$

，ｊ∈Ｂ$_{i}$は、メモリ４６０から読み出され、

【数４１】

$\rho_{i,j}^{k-1}$

，ｊ∈Ｂ$_{i}$は、第１のＦＩＦＯバッファ４２０－１から読み出される。次に、並列の、変換された絶対値の減算ユニット４７０－１～４７０－３によって、各ｊ∈Ｂ$_{i}$についての差

【数４２】

$$\left(P_j^{k-1}-\rho_{i,j}^{k-1}\right)$$

が計算され、対応する結果が、関数φを実行する、並列の変換ユニット４７５－１～４７５－３に渡される。これらの変換ユニット４７５の出力は、（ｋ－１）番目の反復における、ｄ$_c$個の検査ノードからｉ番目のビット・ノードへの対応するメッセージの絶対値、すなわち、式（９）による

【数４３】

$$\left|R_{j,i}^{k-1}\right|$$

，ｊ∈Ｂ$_i$である。

【００４７】

（ｋ－１）番目の反復における、検査ノードからビット・ノードへのメッセージの正負符号（ｓｉｇｎ

【数４４】

$$(R_{j,i}^{k-1})$$

）は、同様に計算される。必要な

【数４５】

$$S_j^{k-1}$$

，ｊ∈Ｂ$_i$は、メモリ４６０から読み出され、

【数４６】

$$\sigma_{i,j}^{k-1}$$

，ｊ∈Ｂ$_i$は、第２のＦＩＦＯバッファ４２０－２から読み出される。次に、（この例示的実施形態では、それぞれがＸＯＲゲートを用いる）並列の正負符号処理ユニット４７８－１～４７８－３により、各ｊ∈Ｂｉについて、積

【数４７】

$$S_j^{k-1}\cdot\sigma_{i,j}^{k-1}$$

が計算される。これらの積は、（ｋ－１）番目の反復における、ｄ$_c$個の検査ノードからｉ番目のビット・ノードへの対応するメッセージの正負符号ビット、すなわち、式（１０）によるｓｉｇｎ

【数４８】

$$(R_{j,i}^{k-1})$$

，ｊ∈Ｂ$_i$である。

【００４８】

検査ノードからビット・ノードへのメッセージの正負符号および絶対値は、ｄ$_c$個の、符号・絶対値から２の補数への変換ユニット４８０に渡される。これらのユニット４８０からの結果は、

【数４９】

$$R_{j,i}^{k-1}$$

，ｊ∈Ｂ$_i$であり、これが、ビット・ノード更新ユニット４１０の入力になる。ビット・ノード更新ユニット４１０は、次式（式（３）も参照）に従って、ｋ番目の反復についての、ビット・ノードから検査ノードへのメッセージ

【数５０】

$$Q_{i,j}^{k}$$

を計算する。

【００４９】

【数５１】

$$Q_{i,j}^{k} = \sum_{l \in B_i, l \neq j} R_{l,i}^{k-1} + \lambda_i \qquad (15)$$

【００５０】

メモリの必要量およびスループット

例示的な第１のＦＩＦＯバッファ４２０－１のサイズは、ｎ・ｄ$_{c}$・ｑビットであり、例示的な第２のＦＩＦＯバッファ４２０－２のサイズは、ｎ・ｄ$_{c}$ビットである。

【００５１】

Ｐ$_{j}$（ｊ＝１．．．ｍ）に必要なメモリの量は、２・ｑ・ｍビットであり、２の倍数は、繰り返しｋおよびｋ－１に関係する値に用いられる。

【００５２】

Ｓ$_{j}$（ｊ＝１．．．ｍ）に必要なメモリの量は、２・ｍビットであり、２の倍数は、繰り返しｋおよびｋ－１に関係する値に用いられる。

【００５３】

必要なメモリの総量は、（２・ｍ＋ｎ・ｄ$_{c}$）・（ｑ＋１）ビットである。

【００５４】

本開示の方法は、各時間サイクルにおいて、ｄ$_{c}$個の、検査ノードからビット・ノードへのメッセージと、ｄ$_{c}$個の、ビット・ノードから検査ノードへのメッセージとを計算する。したがって、１つのビット・ノード更新ユニット４１０で、１回の反復につき、ｎブロックの長さのデータを処理するのに、ｎサイクルかかる。

【００５５】

標準的なＬＤＰＣ復号アーキテクチャと比較すると、本開示のアーキテクチャは、（２・ｍ＋ｎ・ｄ$_{c}$）・（ｑ＋１）ビットだけのメモリ空間を必要とし、１回の反復につきｎサイクルしかかからない。これに対し、一般的な直列和積アーキテクチャは、２・ｎ・ｄ$_{c}$・（ｑ＋１）ビットのメモリ空間を必要とし、１回の反復につきｍ＋ｎサイクルを必要とする。

【００５６】

直列和積アーキテクチャを有する、連結ＬＤＰＣ復号器

ＬＤＰＣ符号は、符号間干渉（ＩＳＩ）およびノイズによって損傷したチャネルのビット・エラー・レートを改善するために用いられることが可能である。図５は、典型的な通信システム５００を示し、システム５００は、ＬＤＰＣ符号化器５１０、（符号間干渉およびノイズを持ち込む）ＩＳＩチャネル５２０、軟入力軟出力（ＳＩＳＯ）検出器６１５、およびＬＤＰＣ復号器６００を備える。図５では、ＬＤＰＣ復号器６００は、後で図６と併せて詳述されるように、ＳＩＳＯ検出器６１５と連結される。ＳＩＳＯ検出器６１５は、チャネル出力値および付帯情報値をＬＤＰＣ復号器６００から取り込み、これらは、アプリオリ情報値として使用される。ＳＩＳＯ検出器６１５は、付帯情報値を出力し、これは、次の反復のためのアプリオリ情報値λ$_{i}$として、ＬＤＰＣ復号器６００によって使用される。ＬＤＰＣ復号器からの付帯情報値は、ＳＩＳＯ検出器用のアプリオリ情報値として使用され、ＳＩＳＯ検出器からの付帯情報値は、ＬＤＰＣ復号器用のアプリオリ情報値として使用される。ＳＩＳＯ検出器は、ＩＳＩチャネルを考慮に入れて、たとえば、当該技術分野では周知であるＭＡＰアルゴリズムまたは軟出力ビタビ・アルゴリズム（ＳＯＶＡ）を用いて、付帯情報値を計算し、ＬＤＰＣ復号器は、ＬＤＰＣ符号を考慮に入れて、付帯情報値を計算する。このシステムの１回の反復は、ＳＩＳＯ検出器６１５およびＬＤＰＣ復号器６００による、１回のデータの処理からなる。

【００５７】
図６は、本開示の、連結されたＳＩＳＯ検出器およびＬＤＰＣ復号器のアーキテクチャを詳細に示す。ＳＩＳＯ検出器６１５は、ＬＤＰＣ復号器６００からのチャネル出力値および付帯情報値を取り込み、付帯情報値を出力する。この出力された付帯情報値は、次の反復のためのアプリオリ情報値$\lambda_i$として、ＬＤＰＣ復号器６００によって使用される。本発明では、ＬＤＰＣ復号器６００を通るすべてのパスにおいて、ビット・ノードｉからその検査ノードへの$d_c$個のメッセージ、すなわち、
【数５２】
$\hat{Q}_{i,j}$
, $\forall j \in B_i$が、ビット・ノードｉのアプリオリ情報値またはアプリオリＬＬＲ $\lambda_i$と等しいという事実、すなわち、次式が成り立つことを利用する。
【数５３】

$$\hat{Q}_{i,j} = \lambda_i, \quad \forall j \in B_i. \tag{16}$$

【００５８】
検査ノードの計算は、前述と同様に、部分に分割される。
【数５４】

$$\hat{\rho}_i = \phi\left(|\lambda_i|\right) \tag{17}$$

$$\hat{\sigma}_i = \operatorname{sign}\left(\lambda_i\right) \tag{18}$$

$$\hat{P}_j = \sum_{l \in C_j} \hat{\rho}_l \tag{19}$$

$$\hat{S}_j = \prod_{l \in C_j} \hat{\sigma}_l \tag{20}$$

【００５９】
検査ノードｊからビット・ノードｉへのメッセージの正負符号および絶対値は、次式で与えられる。
【数５５】

$$\left|\hat{R}_{j,i}\right| = \phi\left(\hat{P}_j - \hat{\rho}_i\right) \tag{21}$$

$$\operatorname{sign}\left(\hat{R}_{j,i}\right) = \hat{S}_j \cdot \sigma_i \tag{22}$$

【００６０】
これらの
【数５６】
$\hat{R}_{j,i}$
, $j \in B_i$を用いて、ＬＤＰＣ復号器からＳＩＳＯ検出器６１５に渡される付帯情報値は、付帯情報計算ユニット６１０によって、次式のように計算されることが可能である。
【００６１】
【数５７】

$$\Lambda_{ext,i} = \sum_{l \in B_i} R_{l,i}\,. \tag{23}$$

【００６２】

この付帯情報値は、ＳＩＳＯ検出器６１５によって、アプリオリ情報値として使用される。

【００６３】

図６は、連結システムとして提案されたアーキテクチャ６００のブロック図を示す（$d_c$＝３および$B_i$＝｛ｊ１，ｊ２，ｊ３｝）。ビット・ノードｉに関係する、各アプリオリＬＬＲ $\lambda_i$は、ＳＩＳＯ検出器６１５から出力される付帯情報値に等しく、２の補数から符号・絶対値への変換ユニット６３０に渡される。絶対値は、関数Ｆφを実行する変換ユニット６３５に送られ、ユニット６３５の出力は、式（１７）で定義される

【数５８】

$\hat{\rho}_i^k$

である。ｋ番目の反復およびｉ番目のビット・ノードについての変換された絶対値

【数５９】

$\hat{\rho}_i^k$

は、メモリ内の

【数６０】

$\hat{P}_j^k$

, $\forall j \in B_i$を更新する、$d_c$個の、変換済み絶対値更新ユニット６４０に供給される。同様に、ｓｉｇｎ

【数６１】

$\hat{\sigma}_i^k$

は、別の、メモリ内の

【数６２】

$\hat{S}_j^k$

, $\forall j \in B_i$を更新する、$d_c$個の、正負符号更新ユニット６５０の集合に渡される。さらに、

【数６３】

$\hat{\rho}_i^k$

および

【数６４】

$\hat{\sigma}_i^k$

が、第１のＦＩＦＯバッファ６２０－１および第２のバッファ６２０－２に、それぞれ供給される。

【００６４】

（ｋ－１）番目の反復についての、検査ノードからビット・ノードへのメッセージ、すなわち、

【数６５】

$\hat{R}_{j,i}^{k-1}$

, $j \in B_i$は、図４と併せて前述された手順と同様に計算される。しかしながら、第１のＦＩＦＯバッファ６２０－１から１つの値だけが使用され（絶対値を得るために

【数６６】

$\hat{\rho}_i^{k-1}$

が使用される）、第２のＦＩＦＯバッファ６２０－２から１つの値が使用される（正負符号を得るために

【数６７】

$$\hat{\sigma}_i^{k-1}$$

が使用される）。したがって、ＦＩＦＯバッファ６２０における必要メモリ量は、図４と併せて前述されたスタンドアロン・アーキテクチャの場合の$d_c$分の１である。

【００６５】

メモリの必要量およびスループット

例示的な第１のＦＩＦＯバッファ６２０－１のサイズは、ｎ・ｑビットであり、例示的な第２のＦＩＦＯバッファ６２０－２のサイズは、ｎビットである。

$P_j$（ｊ＝１．．．ｍ）に必要なメモリの量は、２・ｑ・ｍビットである。

$S_j$（ｊ＝１．．．ｍ）に必要なメモリの量は、２・ｍビットである。

必要なメモリの総量は、（２・ｍ＋ｎ）（ｑ＋１）である。

【００６６】

本開示の方法は、各時間サイクルにおいて、$d_c$個の、検査ノードからビット・ノードへのメッセージと、付帯情報値とを計算する。したがって、１回の反復につき、ｎブロックの長さのデータを処理するのに、ｎサイクルかかる。

【００６７】

本開示のアーキテクチャは、式（１７）～（２３）で定義される和積アルゴリズムを実行する。本開示の方法は、和積アルゴリズムにおいて最小ＬＬＲ値だけが考慮される、ＷｕとＢｕｒｄによって提案された、あまり最適でない方法より、良好なエラー・レート性能を有する。さらに、ＷｕとＢｕｒｄによって提案された方法は、連結ＬＤＰＣ復号器システムにのみ当てはまる。本発明は、（２・ｍ＋ｎ）・（ｑ＋１）だけのメモリ空間を必要とし、１回の反復につきｎサイクルしかかからない。

【００６８】

表記

本明細書では、以下の表記を用いている。

ｉは、ビット・ノードのインデックスである。

ｊは、検査ノードのインデックスである。

ｋは、反復のインデックスである。

$Q_{i,j}$は、ビット・ノードｉから検査ノードｊへのメッセージである。

$R_{i,j}$は、検査ノードｊからビット・ノードｉへのメッセージである。

$\lambda_i$は、ビットｉに関するアプリオリ情報値またはアプリオリ対数尤度比（ＬＬＲ）である。

$\Lambda_i$は、ビットｉに関する事後情報値または事後ＬＬＲである。

$\Lambda_{ext,i}$は、ビットｉに関する付帯情報値または付帯ＬＬＲである。

$B_i$は、ビット・ノードｉに接続された検査ノードの集合である。

$C_j$は、検査ノードｊに接続されたビット・ノードの集合である。

ｎは、ビット・ノードの数である。

ｍは、パリティ検査ノードの数である。

$d_r$は、パリティ検査行列の行重みである。

$d_c$は、パリティ検査行列の列重みである。

ビット・ノードは、１．．．ｎである。

検査ノードは、１．．．ｍである。

【００６９】

本発明の例示的実施形態を、デジタル論理ユニットに関して説明してきたが、当業者であれば自明であるように、様々な機能を、デジタル・ドメインにおいてソフトウェア・プログラムの処理ステップとして、回路素子または状態機械によるハードウェアの形で、またはソフトウェアとハードウェアの組み合わせの形で、実装することが可能である。その

ようなソフトウェアは、たとえば、デジタル信号プロセッサ、マイクロコントローラ、または汎用コンピュータの形で用いられることが可能である。そのようなハードウェアおよびソフトウェアは、集積回路内に実装される回路の中で実施されることが可能である。

【００７０】

したがって、本発明の機能は、方法、およびそれらの方法を実行する装置の形で実施される。本発明の１つまたは複数の態様は、たとえば、記憶媒体に格納されるコードであれ、機械にロードされ、かつ／またはその機械によって実行されるコードであれ、あるいは何らかの伝送媒体上を伝送されるコードであれ、プログラム・コードの形で実施されることが可能であり、そのプログラム・コードが、コンピュータのような機械にロードされ、その機械によって実行される場合は、その機械が、本発明を実行する装置になる。プログラム・コード・セグメントは、汎用プロセッサに実装される場合には、プロセッサとの組み合わせで、特定の論理回路と類似の動作を行う装置を提供する。

【００７１】

当該技術分野において知られているように、本明細書で説明された方法および装置は、それ自体が、コンピュータ可読媒体上で実施されるコンピュータ可読コード手段を有するコンピュータ可読媒体を含む、製品として、販売されることが可能である。コンピュータ可読プログラム・コード手段は、コンピュータ・システムと組み合わされて、本明細書で説明された方法を実施するか、本明細書で説明された装置を作成するステップのすべてまたは一部を実行するよう動作可能である。コンピュータ可読媒体は、記録可能媒体（たとえば、フロッピー（登録商標）・ディスク、ハード・ドライブ、コンパクト・ディスク、メモリ・カード、半導体デバイス、チップ、特定用途向け集積回路（ＡＳＩＣ））であってもよく、伝送媒体（たとえば、光ファイバを含むネットワーク、ワールドワイド・ウェブ、ケーブル、または時分割多重アクセス、符号分割多重アクセスなどを用いる無線チャネル、あるいは無線周波数チャネル）であってもよい。コンピュータ・システムでの使用に好適な、情報を記憶することが可能である、周知の、または開発された、任意の媒体を使用することが可能である。コンピュータ可読コード手段は、コンピュータが、磁気媒体上の磁気変動や、コンパクト・ディスクの表面の高さ変動のような、命令およびデータを読み取ることを可能にする、任意のメカニズムである。

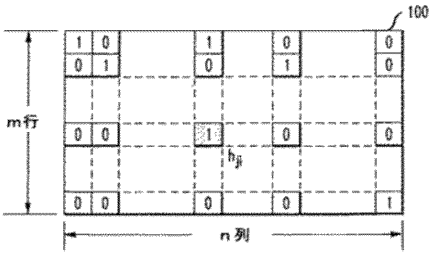
【００７２】

メモリおよびバッファは、分散型であっても局所的であってもよく、プロセッサは、分散型であっても単独であってもよい。メモリおよびバッファは、電気的、磁気的、または光学的なメモリとして実装されてもよく、あるいはそれらまたは他のタイプの記憶装置の任意の組み合わせであってもよい。さらに、用語「メモリ」、「バッファ」、および「ＦＩＦＯバッファ」は、媒体からの読み出し、または媒体への書き込みが可能な任意の情報を包含するのに十分であるよう広く解釈されなければならない。この定義によれば、ネットワーク上の情報も、関連付けられたプロセッサがネットワークから取得できることから、メモリ内にある。

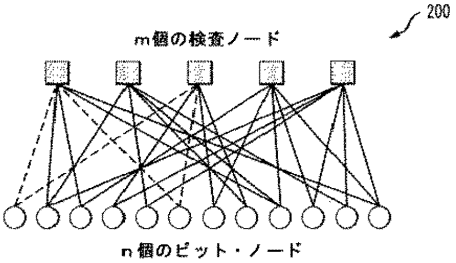
【００７３】

本明細書で図示および説明された実施形態および変形形態は、本発明の原理の例示に過ぎないこと、ならびに、本発明の範囲および趣旨から逸脱することなく、様々な修正が、当業者によって実施可能であることを理解されたい。

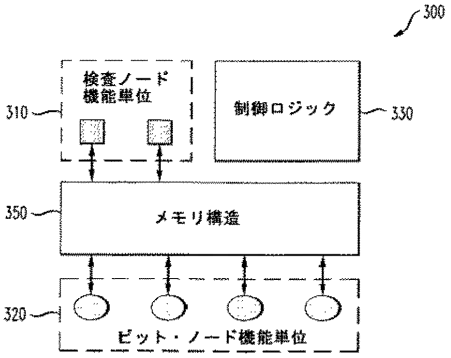
【図１】


100
m行
n列
$h_{ji}$

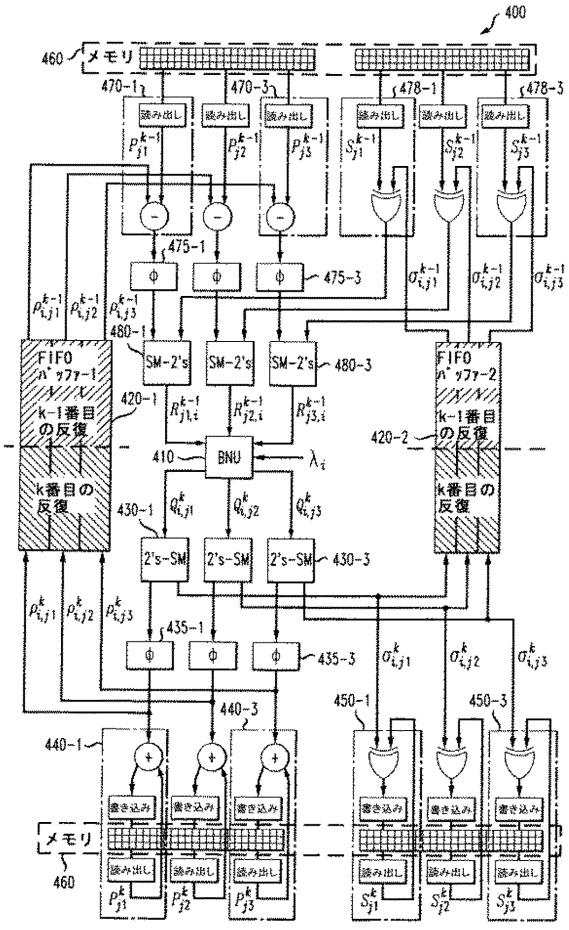
【図２】


200
m個の検査ノード
n個のビット・ノード

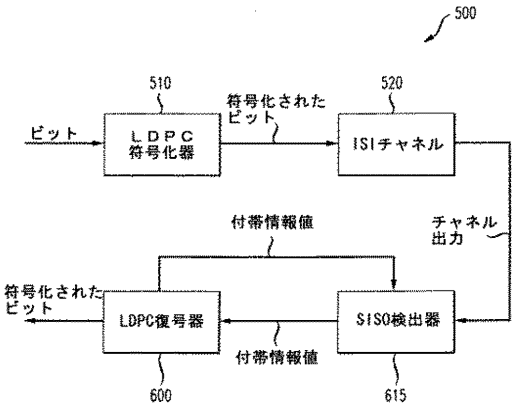
【図３】


300
検査ノード機能単位
310
制御ロジック
330
メモリ構造
350
ビット・ノード機能単位
320

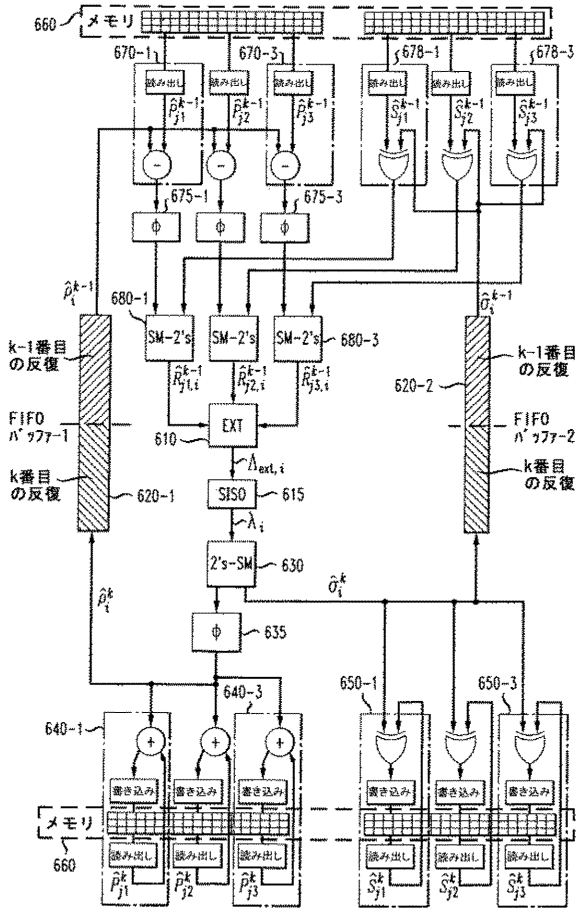
【図４】


400
460
メモリ
470-1
470-3
478-1
478-3
読み出し
475-1
475-3
480-1
480-3
SM-2's
FIFO
n°ラファー1
FIFO
n°ラファー2
k-1番目の反復
k番目の反復
420-1
420-2
410
BNU
430-1
430-3
2's-SM
435-1
435-3
440-1
440-3
450-1
450-3
書き込み

【図５】


500
510
ビット
LDPC符号化器
符号化されたビット
520
ISIチャネル
チャネル出力
付帯情報値
符号化されたビット
LDPC復号器
SISO検出器
付帯情報値
600
615

【図６】


660
メモリ
670-1
670-3
678-1
678-3
読み出し
675-1
675-3
680-1
SM-2's
680-3
k-1番目の反復
FIFO バッファ-1
620-2
FIFO バッファ-2
k番目の反復
610
EXT
620-1
SISO
615
2's-SM
630
635
640-3
640-1
650-1
650-3
書き込み
メモリ
660

フロントページの続き

(72)発明者　ルワン　ラトナヤケ
アメリカ合衆国　０２１３８　マサチューセッツ　ケンブリッジ，　オックスフォード　ストリート　３３　エムデー３１３

Ｆターム(参考) 5J065 AD01 AD07 AG05 AH01