社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2009－MPS－73 (12)
2009／3／5

LDPC符号高速生成システムの提案と評価

野里裕高[†] 石田由香里[†] 高橋栄一[††]
村川正宏[††] 梶谷勇[††]
古谷立美[†] 樋口哲也[††]

我々は先にLDPC符号高速生成システムの提案を行ったが，本論文では，高速化手法の有効性を評価した．LDPC符号は通信性能の限界値に極めて近い性能を引き出すことができる符号として注目を集める一方で，性能の良いLDPC符号の構成方法が確立されていないことと，性能の良いLDPC符号の評価に膨大な処理時間がかかることがLDPC符号の発展における課題となっている．提案システムでは，多目的遺伝的アルゴリズムと高速化手法を用いることで，LDPC符号の持つ問題点をそれぞれ解決する．今回の検証ではハードウェアアクセラレータの有効性を確認することで，先に確認した汎用クラスタの高速化性能と併せて課題を解決できるほどの高速化が可能になる見通しを得た．

Proposal and estimation of High-speed Generated System for LDPC Codes

Hirotaka Nosato,[†] Yukari Ishida,[†] Eiichi Takahashi,[††]
Masahiro Murakawa,[††] Isamu Kajitani,[††] Tatsumi Furuya[†]
and Tetsuya Higuchi [††]

This paper reports on the estimation of the system we have proposed for speed-up in generating LDPC codes. LDPC code is attracting attention due to the considerable potential for error correction that approaches the theoretical limits. However, the following things have impeded the implementation of suitable LDPC codes, uncertainty and time-consuming for generating the suitable LDPC codes. We have run simulations for the system with the hardware accelerator we proposed and have confirmed that our approach with hardware accelerator in conjunction with parallel programming is one of a solution to the speed-up.

1. 序　論

ここ数年における情報技術の目覚ましい発展により，現在の情報機器の性能は以前と比べ格段に向上され，記憶媒体の大容量化，通信の高速化を可能にしている．今後期待される更なる大容量化，高速化の実現にとって，誤り訂正技術の性能向上は必須である．現在に至るまで様々な誤り訂正符号が考案され，誤り訂正符号を用いることで通信時に起こる情報誤りを訂正して，通信の信頼性を高めてきた．近年では，LDPC（Low Density Parity Check）符号[1]と呼ばれる誤り訂正符号が非常に注目を集めている[2]．

LDPC符号はGallagerによって1963年に考案された．LDPC符号はShannon限界と呼ばれる，理論的に計算された通信性能の限界値[3]に極めて近づく性能を有していることが広く認識されるまでの約30年間，一度も脚光を浴びることがなかった符号である．現在では，無線通信の規格であるDVB-S2(衛星デジタル放送)，IEEE802.3an(10Gイーサネット)に採用される[4]など，今後更なる発展が期待されている．

しかし，LDPC符号にとって2つの問題点が発展の障害になっている．1つ目は性能の良いLDPC符号を構成するためのアルゴリズムが確立されていないことである．現状では，数学的手法やランダムに生成した多数の符号の中から，性能の良いものを適宜選択している．2つ目は，性能の良い符号に対する通信路シミュレーション・ソフトウェアを用いた性能評価に膨大な時間がかかることである．

これらの問題点を解決するために，我々はLDPC符号高速生成システムを提案した[5]．提案システムは以下の2つの特徴を持ち，それらによりLDPC符号の問題点を解決する．

(1) 多目的遺伝的アルゴリズム（Multi-Objective Genetic Algorithm: MOGA）[6]を用いた最適もしくは準最適な符号の探索
(2) 汎用クラスタとハードウェアアクセラレータによるMOGAおよび符号評価の高速化

本稿では，ハードウェアシミュレータを用いた検証により高速化手法の有効性を示し，実用化に向けた今後の展望について検討する．

以下，2節で本題であるLDPC符号について説明し，3節では提案手法がいかにしてLDPC符号の問題点を解決するかを述べる．4節では，ハードウェアシミュレータを用いた検証について詳述し，検証結果について考察する．最後に5節では，今後の展望を含めた本稿の結論を述べる．

† 東邦大学大学院 理学研究科
Graduate School of Science, Toho University
†† 産業技術総合研究所 情報技術研究部門
ITRI, National Institute of Advanced Industrial Science and Technology (AIST)

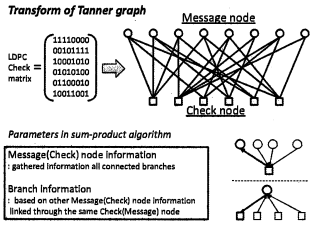

図 1 タナーグラフを用いた Sumproduct アルゴリズムの概要
Fig. 1 The elements of Sumproduct Algorithm with tanner graph

2. LDPC 符号

本節では，本研究で対象とした誤り訂正符号である LDPC 符号についての概要，LDPC 符号を用いて符号化したデータを復号するためのアルゴリズム，そして LDPC 符号が持つ問題点についてそれぞれ説明する．

2.1 誤り訂正符号と LDPC 符号

誤り訂正技術はディジタル通信やストレージにおける信頼性確保に必須の技術である．誤り訂正符号を用いることで，ノイズの影響で誤りが生じたデータを元のデータに正しく復元することが可能である．そのため，高速データ通信や大容量ストレージにおいてのデータ品質が保証される．

LDPC 符号は，検査行列と呼ばれる二値の行列により定義される．特徴として，検査行列は大部分の要素が 0 で構成される疎行列であり，自身の名前の由来にもなっている．符号化には，検査行列から自動的に生成可能な生成行列と呼ばれる行列を用い，検査行列は受信データに誤りが含まれるかどうかを判定するのに用いる．判定の結果誤りが生じていた場合には，検査行列に基づく復号アルゴリズムを用いて誤りを訂正する．LDPC 符号の誤り訂正性能は，BER（Bit Error Rate:ビット誤り率）を用いて表される．

2.2 復号アルゴリズム

誤り訂正符号を適用した転送データは，復号アルゴリズムを用いることでデータ上の誤りを訂正され，元のデータに復元される．LDPC 符号においては，一般的に Sumproduct アルゴリズム [2] と呼ばれる反復復号を特徴とする手法を用いる．反復復号とは，復号した結果をフィードバックして初期値として復号を行うことを繰り返す手法である．

Sumproduct アルゴリズムの概要について，**図 1** に示すタナーグラフと呼ばれる検査行列を変換した 2 部グラフを用いて説明する．タナーグラフでは，検査行列における行に対応する図中のノードをチェックノード，列をメッセージノードと呼ぶ．チェックノードとメッセージノードを結ぶ枝は検査行列内の 1 が立つ場所を表している．各メッセージノードの持つ情報は，チェックノードを介して接続されている他のメッセージノードを基に求めることができる．また，各チェックノードの持つ情報も同様に，メッセージノードを介して接続される他のチェックノードを基にして求められる．Sumproduct アルゴリズムではメッセージノードとチェックノードの情報を互いに更新し，ノードの持つ情報から元のデータを推定することを繰り返すことで復号を行う．推定したデータに対してパリティチェックを行い，エラーが検出されなければ復号終了として，処理を終了する．タナーグラフ中の枝の数（検査行列内の 1 の個数）が多いほど，1 回の繰り返しにかかる計算時間が多い．つまり，同等の誤り訂正性能を示す検査行列においては，検査行列に冗長な情報が少ないほど，性能評価のかかる時間が短くなる．

Sumproduct アルゴリズムを実装する際，自身の持つ繰り返し構造はハードウェア化向きである．そのため，ハードウェアを用いた実装を行うことで性能評価にかかる時間の大幅な短縮が期待できる．また，処理時間を短縮する方法として，Sumproduct アルゴリズム内の計算式の一部に近似式を用いた min-sum 復号法 [2] という手法が存在する．min-sum 復号法では，近似により計算精度は若干劣るが，通常の Sumproduct アルゴリズムより高速な復号が可能である．

2.3 LDPC 符号の問題点

LDPC 符号は Shannon 限界に極めて近づく性能を有する符号ではあるが，LDPC 符号の実用化には問題点が 2 つある．

1 つ目は性能の良い LDPC 符号を構成するためのアルゴリズムが確立されていないことである．現状では，数学的手法やランダムで生成した多数の符号の中から，性能の良いものを適宜選択している．しかし，符号長が長いほど解空間は増大するため，現状の方法を用いてもトライアルアンドエラーを繰り返し行うことで多大な手間がかかる．また符号長が長い LDPC 符号の性能を多数の視点から考慮する場合，さらに余計に手間がかかる場合があると考えられる．BER とハードウェア実装時の回路規模の 2 つを LDPC 符号の性能とすると，ハードウェア実装時の回路規模は検査行列内の 1 の個数と密接な関係にあると言える [7]．なぜなら，Sumproduct 復号法において 1 の個数が Sumproduct 復号法の計算量を握っているため，1 の個数が増えると処理量が多くなり必然的に回路規模が大きくなるからである．そのため，2 つの性能を満たすような LDPC 符号を構成する際，2 つの性能間に存在するトレードオフの関係を考慮する必要が生じるので，その分だけ余計に手間がかかってしまうことになる．

2 つ目は，性能の良い符号に対する通信路シミュレーション・ソフトウェアを用いた性能評価に膨大な時間がかかることである．性能評価において，BER を 1 桁下げるためには 10 倍の計算量が必要になる．そのため，LDPC 符号の更なる発展には高速化が必須である．

3. LDPC 符号高速生成システム

本章では，提案する LDPC 符号高速生成システムについて説明する．まず LDPC 符号の問題点を解決するために提案した LDPC 符号高速生成システムについて述べる．そして，本提案手法が持つ 2 つの特徴である MOGA[6]，高速化手法についてそれぞれ説明する．また，図を用いた詳

しい説明については，文献[5),7)]を参照されたい．

3.1 LDPC 符号高速生成システムの概要

提案システムでは，MOGA を用いることにより複数の視点から最適な LDPC 符号を効率よく探索する．また，MOGA の処理を高速化するために，汎用クラスタを用いて並列処理[8)]を行う．加えて，時間の性能評価を高速化するためにハードウェアアクセラレータを用いて高速計算を行う．以上の特徴を生かして提案システムを用いることで，前節で述べた LDPC 符号の問題点を解決し，LDPC 符号の普及拡大に寄与できる．

3.2 特徴 1：MOGA

本提案システムにおける MOGA では，LDPC 符号の持つ複数の性能指標間に存在するトレードオフを考慮した LDPC 符号を探索することを目的としている．この手法における最適な LDPC 符号は，パレート最適解と呼ばれる複数の最適解の集合として得られる．使用者は用途に合わせて各性能における重要度を決定し，それに対応した最適な LDPC 符号をパレート最適解の中から選択することが可能である．

3.3 特徴 2：高速化手法

本提案システムでは，汎用クラスタにおける MOGA 内部の処理を並列化[8)]，ハードウェアアクセラレータを用いた LDPC 符号性能評価における高速計算により，高速化を実現した．

汎用クラスタを用いた並列化では，MOGA における個体の性能評価のフェーズを並列化の対象とした．これにより個体の性能評価おいて，ノード数 X 個の汎用クラスタを用いることで最大 1/X 倍の時間短縮が可能である．

ハードウェアアクセラレータでは，反復復号を特徴とする復号アルゴリズムを用いたデータの復号処理を高速化した．なぜなら性能の良い検査行列ほど復号処理を用いた BER 算出に時間がかかるからである．同じ処理を繰り返す反復復号の処理は，ハードウェアによるデータ処理の並列化を行うことで大幅な時間短縮ができる．

4. 提案システムの検証

本節では，LDPC 符号高速生成システムおよびハードウェアシミュレータを用いた検証について述べる．今回の検証では，ハードウェアを用いた復号処理の高速化が可能であることを示す．以下では，検証についての詳細を述べ，検証の結果について説明する．

4.1 検証詳細

本稿では，シミュレータを用いた検証によりハードウェアの有効性を示す．まず，ハードウェア記述言語である VerilogHDL を用いて min-sum 復号アルゴリズムを実現し，次に，実現した VerilogHDL のコードをシミュレータを用いて論理合成を行った．そして，論理合成の結果得られる情報を波形で表現した図を用いて，ハードウェアにおける復号処理の動作検証を行った．さらに，高速化率の算出を行うために，ソフトウェアで実現した min-sum 復号法を用いて反復処理 1 回あたりの処理時間の計測を行った．

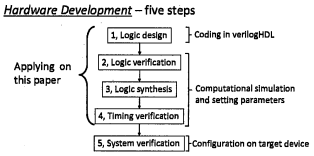

図 2 ハードウェア開発手順
Fig. 2 Development of hardware

今回の検証では，ハードウェアの動作検証にシミュレーションの結果を用いた．なぜなら，ハードウェア開発手順(**図 2** に示す) では，手順 4 までの情報をそのままハードウェアに書き込むため，手順 5 にて完成したハードウェアは，手順 4 までのシミュレーション結果通りに動作するからである．

ハードウェアシミュレーションでは，以下の検査行列を用いて動作検証を行った．

$$H = \begin{pmatrix} 1&0&1&0&0&1&0&0&1&0&0&0&0&0&0&0 \\ 1&0&0&1&0&0&1&0&0&1&0&0&0&0&0&0 \\ 1&0&0&0&1&0&0&1&0&0&1&0&0&0&0&0 \\ 0&1&0&1&0&0&0&1&0&0&0&1&0&0&0&0 \\ 0&1&0&0&1&0&1&0&0&0&0&0&1&0&0&0 \\ 0&1&0&0&0&1&0&1&0&0&0&0&0&1&0&0 \\ 0&0&1&1&0&1&0&0&0&0&0&0&0&0&1&0 \\ 0&0&1&0&1&0&1&0&0&0&0&0&0&0&0&1 \end{pmatrix} \quad (1)$$

通信路は AWGN 通信路 (信号対雑音比 (SNR)：2.0dB) を仮定し，送信データには全てのビットが 1 のデータを用いた．ハードウェアシミュレータには Altera 社の QuatusII7.2 を使用し，シミュレーションにおけるパラメータである動作周波数は 100MHz に設定した．演算精度は 16 ビットの固定小数点で表現した．そして，ソフトウェアによる復号処理の時間計測では，以下の 2 つの検査行列を使用した．

(i) (1) の検査行列
(ii) 行列内に 3000 個の '1' がランダムに存在する 500 × 1000 の検査行列

コードは C 言語で記述し，検証に用いた計算機のスペックは CPU: Intel Core2Quad 2.GHz, Memory: 3GB, OS: WindowsXP である．処理時間は乱数の種を適当に替えて，10 試行平均にて算出した．

4.2 結　　果

ハードウェアシミュレータによる検証結果を**図 3**，**図 4** に示す．図 4 は 1 回目の反復処理の部分を拡大した図である．縦軸には，min-sum 復号法で使用している各変数が示してあり，横軸は時間の流れを表している．図 3 において，初期化処理 (図中の"Initialize") にて，検査行列のデータ及び min-sum 復号法のパラメータをハードウェア上に読み込んでいる．初期化処理後，4 回目の反復処理においてデータの復号が成功し，復号処理を終了している．1 回の反復処理 (図中の"One of iteration") には，およそ 1.4μs の時間を必要とした．図 4 において，反復処理ではまず，メッセージ及びチェックノードの更新を行う (a)．次に，更新したデータを用いてでデータの推定及びパリティ

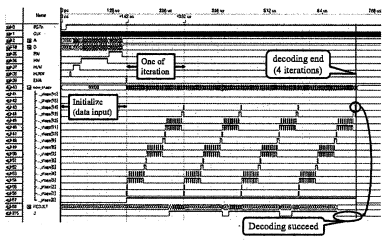

図 3 シミュレータによる検証結果
Fig. 3 Simulation result

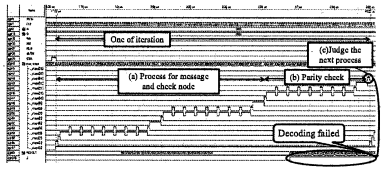

図 4 1 回目の反復処理の拡大図
Fig. 4 Detail of first iteration process

検査行列	1 回の復号処理にかかる時間 [秒]
(i)	1.6×10^{-4}
(ii)	8.4×10^{-3}

表 1 ソフトウェア min-sum 復号法による処理時間

チェックを同時に行う (b) . そして最後にパリティチェックの結果より次の処理を決定する (c) が，ここでは復号が失敗しているため次も反復処理が行われる．

ソフトウェアで実装した min-sum 復号法を用いた場合の各検査行列における処理時間は**表 1** の通りである．(i) 及び (ii) は，4.2 節に対応している．また，(1) の検査行列を用いた復号処理において，繰り返し回数 4 回で正しくデータを復号できた．

4.3 考　　察

実験の結果より，100MHz の動作周波数において処理中の波形の乱れもなく，4 回の反復処理により復号ができた．また，ソフトウェアでも同じ検査行列を用いて評価を行った結果，同等の性能を得られることを確認できた．このことより，min-sum 復号法のハードウェア実装は可能であると言える．そして，結果より得られたハードウェア及びソフトウェアの処理時間から，今回用いた検査行列の条件では，ハードウェアを用いることで約 100 倍の高速化が可能になる．以上のことより，(1) の検査行列においてハードウェアを用いた高速化が可能であると示せた．

実用化に向けた考察と高速化手法の必要性の再確認のために，500 × 1000 の行列にかかる処理時間と BER の関係を表す図を**図 5** に示す．図中の縦軸と横軸はそれぞれ復号処理にかかる時間と BER を示しており，破線と直線はそれぞれ実験結果を用いて算出したソフトウェアによる性能とハードウェアによる性能を示している．図 5 によると，ソフトウェアを用いた BER が 10^{-12} である LDPC 符号の性能評価には，252 年かかることになり，実用化は不可能である．しかしながら，検証で用いたハードウェアを用

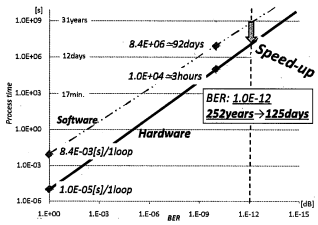

図 5 500 × 1000 行列における性能と処理時間
Fig. 5 Evaluation of 500 × 1000 matrix

いることで約 82 倍の高速化が可能であるため，252 年かかる処理をわずか 125 日で行うことが可能であると見込める．このことから，提案システムの実用化には，汎用クラスタ及びハードウェアアクセラレータを用いた高速化は必要不可欠である．

5. 結　　論

本論文では，提案システムを用いて LDPC 符号の抱える問題点解決を可能にするために，システムの特徴であるハードウェアアクセラレータを用いた高速化の有効性について検証した．検証の結果，提案システムにおいてハードウェアアクセラレータを用いた飛躍的な高速化が可能であることを示した．今後の課題としては，計算機とハードウェアの間のデータ通信に用いる規格を決定し，実装を行った上で実機を用いた提案システムの動作検証を行っていきたい．

参 考 文 献

1) R. Gallager. Low-density parity-check codes. *M.I.T.Press,Cambridge,MA*, 1963.
2) 和田山正. 低密度パリティ検査符号とその復号法. トリケップス, 2002.
3) C.E. Shannon and W Weaver. The mathematical theory of communication. *University of Illinois Press*, 1963.
4) D.J. Costello and G.D. Forney. Channl coding:the road to channel capacity. *Proceedings of the IEEE*, Vol. 95, No. 6, pp. 1150–1177, 2007.
5) 石田由香里, 野里裕高, 飯島洋祐, 村川正宏, 梶谷勇, 高橋栄一, 古谷立美, 樋口哲也. 可変長染色体 GA を用いた LDPC 符号の最適化設計システムの提案. 情処学 MPS 研報, MPS-68, 2008.
6) Carlos A. Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, Vol. 1, No. 3, pp. 269–308, 1999.
7) 野里裕高, 石田由香里, 飯島洋祐, 村川正宏, 梶谷勇, 高橋栄一, 古谷立美, 樋口哲也. 可変長染色体 GA を用いた LDPC 符号の最適化設計システムの高速化. 情処学 MPS 研報, MPS-68, 2008.
8) P. パチェコ. MPI 並列化プログラミング. 培風館, 2001.