Vol.2010-MPS-77 No.13
2010/3/4

GPGPUによるLDPC符号復号の高速化に関する予備評価

野里裕高[†1]　高橋栄一[†2]　村川正宏[†2]
古谷立美[†1]　樋口哲也[†2]

我々が先に提案したLDPC符号高速生成法の部分検証として，本稿ではLDPC符号復号の高速化に関する予備評価を行った．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誤り訂正符号として近年注目を集めているが，性能の良い誤り訂正符号の性能評価には膨大な時間が必要になってしまうことが，LDPC符号の実用化に際する障害の一つになっていると考えられる．そこで，LDPC符号性能評価にハードウェアを使用して大幅な処理時間の短縮を実現することで，提案するLDPC符号高速生成法で生成したLDPC符号の実用化を可能にする．本稿では，GPGPUによるLDPC符号性能評価の高速化の有効性を検証した．GPGPUによる高速化を適用したLDPC符号性能評価では，実用的なサイズのLDPC符号の性能評価に関して最大4.927倍の高速化が実現できた．

Evaluation about acceleration of a Decoding method for LDPC Codes using GPGPU

Hirotaka Nosato,[†1] Eiichi Takahashi,[†2]
Masahiro Murakawa,[†2] Tatsumi Furuya[†1]
and Tetsuya Higuchi [†2]

This paper reports on the results of the evaluation for a new approach to obtain LDPC(Low-Density Parity-Check) codes with GPGPU(General-Purpose Graphics Processing Unit) deployed on the machine. LDPC codes are well known to show high performance in communications reaching the Shannon limit. On the other hand, it is also well known that it requires huge amount of time to find good codes, especially in evaluating a code that has been calculated. We deployed GPGPU on our machine for reducing the calculation time. Our approach with GPGPU results in the fact that it runs more than 4(appx. 4.927) times faster than the one without GPGPU.

1. はじめに

近年，情報通信技術の目覚ましい発展により，通信の高速化や記憶媒体の大容量化がなされ，情報通信が一般に広く普及している．最近では，モバイル端末を使用したインターネット利用者が年々増加しており，無線通信の性能向上が期待されている．無線通信は，有線の場合と比べて通信品質が劣化しやすいため，誤り訂正技術[1)]による通信品質の向上が必要不可欠である．最近ではLDPC（Low-Density Parity-Check，低密度パリティ検査）符号[2)]と呼ばれる誤り訂正符号が注目を集めている[3)]．

LDPC符号は，疎な検査行列により定義される符号[4)]であり，Shannon限界[5)]と呼ばれる通信性能の限界値に近い性能を引き出せる[6)]ことで知られている．しかしながらLDPC符号にとって，以下の2点が発展の障害になっていると考えられる．

- LDPC符号の構成方法に関する問題
- LDPC符号性能評価にかかる処理時間の問題

これらの問題点を解決する方法として，我々はLDPC符号高速生成法を提案した[7)]．提案する方法は，確率的探索[8)]により性能の良いLDPC符号を発見し，高速化手法を適用して処理時間を短縮することで，2つの問題点を解決し，LDPC符号の実用化拡大を可能にする．

本稿では，提案するLDPC符号高速生成法の部分検証として，GPGPU（General Purpose Graphic Processing Unit）[9)]によるLDPC符号性能評価の高速化に関する予備評価を行った．LDPC符号性能評価において最も処理に時間がかかる復号処理に関して，一部の処理をGPGPUによる高速化を適用した．高速化の評価として処理時間の比較を行った結果，最大4.927倍の高速化が実現できた．

以下，**2.**では，LDPC符号について説明し，**3.**では，LDPC符号復号法であるSum-product復号法について述べる．**4.**では，GPGPUによるLDPC符号復号の方法と行った検証に関する考察を述べ，**5.**では結論を述べる．

†1 東邦大学大学院 理学研究科
Graduate School of Science, Toho University
†2 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)

Vol.2010-MPS-77 No.13
2010/3/4

2. LDPC 符号

LDPC 符号は 1963 年に Robert G.Gallager によって発明された誤り訂正符号で，1990 年代に高い誤り訂正能力を有することで注目され始め，今後，更なる実用化の拡大が期待されている符号である．以下，**2.1** で LDPC 符号の定義について，**2.2** で LDPC 符号を使用した通信の方法について，**2.3** で LDPC 符号の性能評価についてそれぞれ述べる．

2.1 LDPC 符号の定義

LDPC 符号は検査行列[1] と呼ばれる二値の行列により定義され，検査行列の非零要素の数が非常に少ない行列となる特徴を持っている[4]．検査行列の構成により，LDPC 符号は 2 種類に分けられる．LDPC 符号検査行列の各行と各列の非零要素数がそれぞれ等しいものをレギュラー LDPC 符号と呼び，レギュラー LDPC 符号でないものをイレギュラー LDPC 符号と呼ぶ[3]．イレギュラー LDPC 符号の中にはレギュラー LDPC 符号より良い性能を示す符号があることが確認されている[10]．

2.2 LDPC 符号を使用した通信

LDPC 符号を使用した通信の流れを図 1 に示す． 送信するデータを送信語，符号化により生成されるデータを符号語，受信するデータを受信語と呼ぶ．送信側では，あらかじめ送信語を k ビット毎の固定長に分割しておく．

まず，送信語は符号化処理により n ビットの符号語に変換される．符号語は送信語と生成行列と呼ばれる行列を乗算することにより生成される．生成行列とは，検査行列から生成することができる $k \times n$ のサイズの行列である．符号化により付加される冗長ビットは $n-k$ ビットである．

次に，符号語を送信する．通信路では，雑音の影響によりデータ誤りが起こる可能性がある．データ誤りは符号語に対してデータビットをランダムに反転させることにより表現される．

受信語に誤りが存在するかどうかを検証するためには，シンドロームと呼ばれる $n-k$ ビットのベクトルを計算する．シンドロームの計算は，受信語と検査行列を乗算することで計算され，受信語に誤りが存在しない場合には，0 ベクトルとなる．シンドロームが 0 ベクトルでない場合は，受信語に対して LDPC 符号復号法による復号処理を行い，誤り訂正を行う．

2.3 LDPC 符号性能評価

LDPC 符号の性能は，LDPC 符号を使用した通信における復号処理後の BER（Bit Error Ratio，ビット誤り率）で表現される．LDPC 符号性能評価は，LDPC 符号検査行列に基づいて符号化，通信路シミュレーション，復号の 3 つの処理をソフトウェアで実行することにより BER を算出する．

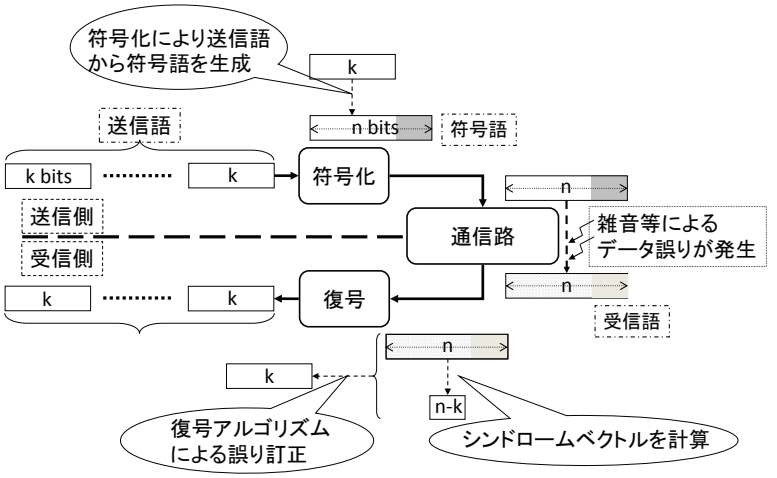

図 1 LDPC 符号を用いた通信の流れ

LDPC 符号を使用した通信における符号化，及び通信路シミュレーションは単純な行列演算により表現できるが，復号に関しては，計算量の多い処理が必要になるため，比較的処理が煩雑になる．そのため，LDPC 符号性能評価では復号処理に最も時間がかかる．

3. LDPC 符号復号法

LDPC 符号復号法には Sum-product 復号法と呼ばれる反復復号を特徴とする方法が一般的に用いられる[3]．Sum-product 復号法は事後確率を計算する MAP（Maximum A Posteriori，最大事後確率）復号法[4] の近似アルゴリズムであり，確率領域 Sum-product 復号法と対数領域 Sum-product 復号法と呼ばれる本質的に等価な 2 種類の復号法が存在する．

Vol.2010-MPS-77 No.13
2010/3/4

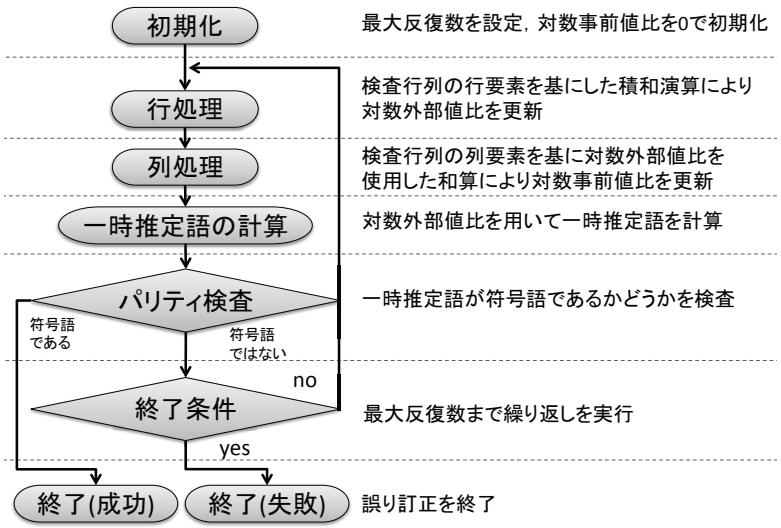

図 2 対数領域 Sum-product 復号法の処理

Sum-product 復号法は LDPC 符号検査行列の非零要素の数が多くなるほど計算量が多くなる特徴を持っている．しかしながら，各行や各列に対する処理は並列化が可能であるため，ハードウェアを使用した高速化が可能である．**3.1** にて Sum-product 復号法の処理手順について説明する．

3.1 Sum-product 復号法の処理手順

ここでは，数値計算がしやすく実装向きである対数尤度比を使用した対数領域 Sum-product 復号法を使って Sum-product 復号法の処理手順を図 2 に示す．

初期化処理として，対数事前値比と呼ばれる値を全て 0 とし，反復復号における反復の最大回数を決定しておく．

まず，LDPC 符号検査行列の行要素に基づいて対数外部値比を更新する処理である行処理を行う．対数外部値比の更新では，対数尤度比及び対数事前値比を使用した積和演算を行う．

行処理を終えた後，列処理と呼ばれる演算処理にて対数事前値比を更新する．列処理では，LDPC 符号検査行列の列要素に基づいて先ほど算出した対数外部値比を使用した和算を行う．

対数尤度比，及び対数外部値比を使用した和算から推定される復号語を一時推定語とする．

そして一時推定語に対してシンドローム計算によるパリティ検査を行い，一時推定語にデータ誤りが存在しないかどうかを判定する．一時推定語に誤りが存在しない場合は，誤り訂正成功として処理を終了する．そうでない場合は，最大反復数に達するまで繰り返し行処理から行う．最大反復数分だけ処理を繰り返しても誤りが訂正できない場合は，訂正失敗として処理を終了する．

4. GPGPU による LDPC 符号復号

本稿では，LDPC 符号復号法である対数領域 Sum-product 復号法に対して GPGPU による高速化を適用し，その効果について検証を行った．**4.1** で LDPC 符号性能評価における処理時間の割合について行った検証結果について説明し，**4.2** で GPGPU による高速化を適用する範囲とその方法について述べる．**4.3** では，GPGPU による高速化を適用した LDPC 符号性能評価処理に関して行った検証結果と考察について述べる．

4.1 LDPC 符号性能評価における処理時間の割合

表 1 の条件にて実用最小サイズの LDPC 符号を使用して性能評価を行った結果，性能評価における各処理時間の割合は図 3 に示すような結果が得られた．SNR（Signal-to-Noise Ratio）は，信号対雑音比のことで，通信路での雑音の量をデシベル値で表現する．

bash の内部コマンドである time を使用して処理時間を計測した結果，100 試行の平均処理時間は 3.846 秒であった．LDPC 符号性能評価処理の中でも，大きく時間の割合を占めている処理が，対数領域 Sum-product 復号法の行処理，列処理，一時推定語の計算処理である．これら 3 つの処理をまとめると 98.11% となり，LDPC 符号性能評価の大部分が復号処理であることが確認できる．

4.2 高速化適用方法

本稿では，**4.1** で行った検証の条件にて LDPC 符号性能評価に関して GPGPU による対数領域 Sum-product 復号法の高速化を行った．実験では，LDPC 符号性能評価において最も処理時間の割合が多い行処理について GPGPU による高速化を適用した．

GPGPU による対数領域 Sum-product 復号法の処理手順を図 4 に示す． GPGPU による行処理を行うために，まず GPU 側のメモリ上にデータをコピーする．コピーするデータは，行処理の積和演算に使用する LDPC 符号検査行列と対数尤度比である．次に GPU 上

OS	WindowsXP Pro 32bit
CPU	Intel Core2Quad Q6600 2.40GHz
メモリ	3GB
コンパイラ	GCC4.3.2
プロファイラ	GNU gprof 2.18.50
Cygwin	version 1.5.25
行列サイズ	500 × 1000
各行，各列の非零要素数	6，3
LDPC 符号復号法	対数領域 Sum-product 復号法
復号おける最大反復回数	20 回
通信路シミュレーション時の SNR	2.0dB
通信路シミュレーション時の雑音モデル	AWGN モデル[4)]

表 1 LDPC 符号性能評価における実験条件

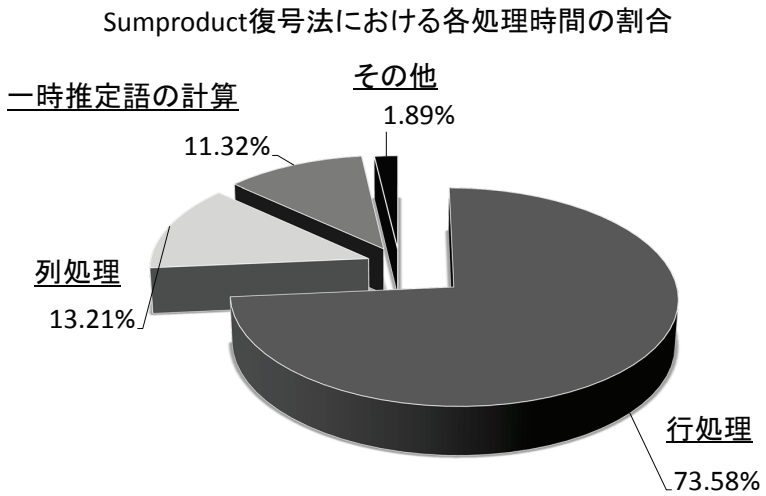

図 3 LDPC 符号性能評価における各処理時間の割合

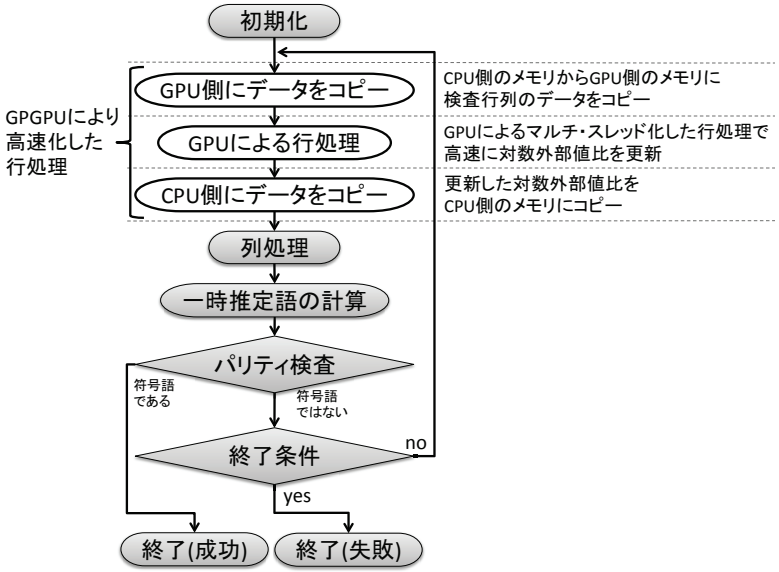

図 4 GPGPU による対数領域 Sum-product 復号の処理手順

で，マルチスレッドにより行処理を高速に行う．そして更新した対数外部値比を CPU 側のメモリにコピーを行い，行処理を終了する．

GPU 上でのマルチスレッドによる行処理では，各行における計算処理を並列に行うことで，高速化を実現する．これにより，LDPC 符号検査行列の行に m 個の非零要素がある場合，m 倍の高速化が可能になる．更に，GPU が持っている除算や対数関数といった演算に特化した計算ユニットを使用して行処理における除算や対数計算を行った．これにより，高速に計算を実行できる[9)]．

4.3 検証結果と考察

GPGPU を適用することによる検証条件の詳細は表 2 の通りである．高速化を適用した LDPC 符号性能評価を使用して比較を行った結果を表 3 に示す．

100 試行の平均を算出した結果，GPGPU を使用した場合の処理時間は 0.649 秒であり，GPGPU を使用することで 3.986 倍の高速化が可能であった．最も高速化率が高い試行に

OS	WindowsXP Pro 32bit
CPU	Intel Core2Quad Q6600 2.40GHz
メモリ	3GB
コンパイラ	GCC4.3.2
プロファイラ	GNU gprof 2.18.50
Cygwin	version 1.5.25
行列サイズ	500×1000
各行，各列の非零要素数	6，3
LDPC 符号復号法	対数領域 Sum-product 復号法
復号おける最大反復回数	20 回
通信路シミュレーション時の SNR	2.0dB
通信路シミュレーション時の雑音モデル	AWGN モデル[4]
GPU	NVIDIA GeForce8800GTX
CUDA コンパイラ	NVCC for CUDA2.3
GPU プロファイラ	cudaprof

表 2 GPGPU を使用した LDPC 符号性能評価における実験条件

	算出した BER	GPGPU なし	GPGPU あり	高速化率
100 試行の平均	2.985×10^{-3}	2.588 秒	0.649 秒	3.986 倍
最も高速化率が高い試行	6.828×10^{-4}	8.312 秒	1.687 秒	4.927 倍

表 3 高速化に関する比較結果

おいては，4.927 倍の高速化が実現できた．

検証結果より，GPGPU による LDPC 符号性能評価の高速化は有効であるとわかり，行処理に対して GPGPU を適用することにより，4.927 倍の高速化が実現できた．更に高速化率を高めるためには，対数領域 Sum-product 復号法における列処理及び，一時推定語の計算に関しても GPGPU を適用する必要がある．今回検証で使用したグラフィックボードは比較的低スペックの GPU を搭載した製品であるため，現行のモデルの GPU を搭載したグラフィックボードを使用して処理を実行することで処理時間を短縮できると考えられる．

5. 結　　論

本稿では，我々が提案した LDPC 符号高速生成法の部分検証として，GPGPU を使用した LDPC 符号性能評価の高速化を行い，その有効性を検証した．LDPC 符号性能評価で最も処理時間がかかる Sum-product 復号法の行処理に関して，GPGPU による高速化を適用した結果，500×1000 のレギュラー LDPC 符号の性能評価処理について最大 4.927 倍の高速化を実現した．検証結果より，GPGPU による LDPC 符号性能評価の高速化が有効であることを確認できた．更に高速化を行う方法として，今後は最新の GPU を搭載した環境にて，Sum-product 復号法における列処理及び一時推定語の計算にも GPGPU による高速化を適用することが考えられる．今後は，GPGPU による高速化の性能を向上させ，プログラムを拡張してイレギュラー LDPC 符号の性能評価を行い，結果についての報告を行う予定である．

参 考 文 献

1) 江藤良純, 金子敏信. 誤り訂正符号とその応用. オーム社, 1997.
2) R.Gallager. Low-density parity-check codes. *M.I.T.Press,Cambridge,MA*, 1963.
3) 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.
4) Shu Lin. and Daniel J.Costello. *Error control coding second edition.* PEARSON Prentice Hall, 2004.
5) C.E. Shannon and W.Weaver. The mathematical theory of communication. *University of Illinois Press*, 1963.
6) D.Mackay and R.M. Neal. Near shannon limit performance of low density parity check codes. *Electron Lett32(18)*, August 1996.
7) 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. LDPC 符号高速生成システムの提案と評価. 情処学 MPS 研報, MPS-73, 2009.
8) 相吉英太郎, 安田恵一郎. メタヒューリスティクスと応用. 電気学会, 2007.
9) NVIDIA Corporation. CUDA Zone. http://www.nvidia.co.jp/, 2010.
10) ThomasJ. Richardson, M.Amin Shokrollahi, and RudigerL. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 47, NO. 2*, FEBRUARY 2001.