Vol.2015-HPC-151 No.10
2015/9/30

メモリ帯域圧縮ハードウェアを用いた数値計算の高性能化

上野 知洋[1,,a)]　佐野 健太郎[1,]　山本 悟[1]

概要：メモリ帯域は大規模な数値計算において，計算性能を決定する重要な要素である．帯域が不足した場合，データの供給が追い付かずに処理回路の性能を十分に発揮できないため，近年，帯域に影響されにくいハードウェアを用いたストリーム計算による数値シミュレーションが提案されている．しかしながら，この手法においても最大性能はメモリ帯域によって決定されるため，容易に帯域を向上させ得る技術が求められている．本研究では，FPGA を用いた数値ストリーム計算機のメモリ帯域を向上させるために，高スループットなデータ圧縮ハードウェアを用いたメモリ帯域圧縮の提案とその性能評価を行う．このハードウェアはメモリ-FPGA 間の通信データを圧縮することにより，利用可能なメモリ帯域を超える計算性能を実現する．帯域圧縮ハードウェアを FPGA 上に実装しスループットを評価した結果，最大でメモリ帯域の約 2 倍のスループットを FPGA 上で実現可能なことが分かった．

キーワード：データ圧縮, 数値シミュレーション, FPGA, ストリーム計算, 帯域圧縮

High-performance Numerical Stream Compuation with Hardware-based Memory Bandwidth Compression

Tomohiro Ueno[1,,a)]　Kentaro Sano[1,]　Satoru Yamamoto[1]

Abstract: Memory bandwidth is a critical factor for the performance of computation, especially in the large-scale numerical simulations. An insufficient bandwidth causes a lack of data supply to the processor, which prevents to exploit its available performance. On the other hand, a stream computing with dedicated hardware has many advantages such as high throughput. However, memory bandwidth still limit the maximum performance of the dedicated hardware. This study presents high throughput hardware-based data compression for memory bandwidth enhancement to imrove the performance of FPGA-based stream computing. This compressor reduces a required memory bandwidth and achieves higher throughput than the available memory bandwidth. We implement the hardware on FPGA and demonstrate the compression hardware can improve the throughput up to twice of memory bandwidth.

Keywords: Data compression, Numerical simulation, FPGA, Stream computing, Bandwidth compression

1. はじめに

専用設計ハードウェアを用いたストリーム計算は，省電力やメモリ帯域に影響されにくい等の点から特に大規模な数値シミュレーションへの適用が様々な形で提案，研究されている．専用回路において複数の処理をカスケードに実行することにより，通信あたりの計算量を増加させ効率的な処理が可能になる．この手法においては，専用設計の計算回路とメモリにより構成された計算機を用いてストリーム計算を行うため，計算性能は計算回路のデザインとメモリ帯域のみに依存する．計算回路の実装には回路再構成可能な FPGA 等が用いられており，自由な設計が可能である．そのため，実際の計算では利用可能なメモリ帯域によって最大性能が決定される．近年の FPGA に代表される回路再構成デバイスの性能向上に伴い，利用可能な計算資源は大きく増加し，高性能な計算回路の実装も可能になっている．一方，メモリ帯域の向上には，計算回路を実

[1] 東北大学
Tohoku University
[a)] ueno@caero.mech.tohoku.ac.jp

装したデバイスとメモリ自体の改良が考えられるが，ピン数や通信路等の機械的な向上は作業の困難さや汎用性の無さから現実的ではない．

この専用計算機において，データ圧縮を用いた計算性能向上手法が提案されている．これは，計算回路を実装したデバイス上でメモリへの送信前にデータ圧縮を適用し，再びデータを利用する際に回路への入力前に展開する手法である．このようなデータ圧縮の実現には，高いスループットと計算に殆ど影響しない低レイテンシであることが求められる．このような制約を満たすため，ハードウェアによるデータ圧縮の提案が行われている．本研究では，このデータ圧縮ハードウェアを用いた専用計算機におけるメモリ帯域の向上手法を提案する．

数値計算に用いられる浮動小数点データは，一般的なデータ圧縮手法であるハフマン符号等のエントロピー符号化による圧縮効果が小さい．この場合の圧縮手法として，データの連続性を用いた予測に基づく圧縮が提案されている．これは単純な多項式に直前の入力データを代入し，次の入力データを予測する手法である．予測値は実際の入力値と比較され，その差分を符号化して圧縮データとする．このようなデータ圧縮ハードウェアの数値計算への適用には，複数チャネルを圧縮する手法が必要となる．数値シミュレーションでは，一般に複数の変数に対して互いに参照しつつ計算する．ソフトウェアによるシミュレーションの場合，メモリやキャッシュに保持されたデータを必要な際に読み出して計算を行うが，ハードウェアにおけるストリーム計算では，計算パイプライン中に複数のチャネルを設けて各変数ごとに並列的な処理をしながら，必要な際に互いのチャネルの値を参照する．このため，専用計算機内の複数チャネルでは全てのチャネルが同期して計算を行う必要がある．数値データの圧縮は計算の精度維持のための可逆圧縮により圧縮後のデータ長が常に変動するため，固定されたメモリ帯域を出力頻度の異なる各チャネルに同期を維持できるように割り当てる符号化方式が求められる．

本研究では，複数チャネル処理が要求される数値計算ハードウェアに対して，高スループットなデータ圧縮ハードウェアによる性能向上手法を提案する．浮動小数点データに対して効果的な圧縮アルゴリズムと，複数チャネルの圧縮データを前述の条件を満たして符号化する多重化方式の提案を行う．提案するデータ圧縮ハードウェアは，圧縮性能と小面積を両立した設計であり，多数のチャネルに対しても面積オーバーヘッドを抑えることができる．また，複数チャネルを単一のメモリ送信路に符号化するモジュールと，逆に単一のチャネルにより送られてくる圧縮データを元のチャネルへと分配する復号化モジュールについても述べる．これらのハードウェアを最新の FPGA に実装して行った評価では，メモリ帯域を超えるスループットを FPGA 上において実現可能であることを示す．

本稿の構成は以下のとおりである．2 節では単一チャネルに対する浮動小数点データ圧縮，展開アルゴリズムとそのハードウェアデザインを示す．3 節では複数チャネルにデータ圧縮を適用した際の符号化方式と，それを実現するモジュールを示す．4 節ではデータ圧縮ハードウェアの性能評価と，圧縮による帯域向上を実機上における評価により示す．5 節では結論と今後の課題を述べる．

2. 単一チャネルに対するデータ圧縮ハードウェア

2.1 データ圧縮アルゴリズム

数値計算におけるメモリ帯域へのデータ圧縮適用は，圧縮が計算に影響を及ぼさないように完全な可逆圧縮が必要になる．浮動小数点データには一般的なエントロピー符号化による圧縮は効果的ではなく，データの一部分をシンボルと見なす不可逆圧縮も適用できない．さらに，ストリーム計算においてはデータ全体をあらかじめ参照することもできないため，シングルパスで単純かつ効果的な圧縮手法が必要になる．そのため，特に数値データの圧縮において予測を用いた効果的な方法がいくつか提案されている．この中には，2 次元計算空間において隣接する計算点データを使った Lorenzo 予測器によるものや，ハッシュテーブルを用いてデータの変動パターンを記録しておく手法などがある [1], [2], [3]．これらの研究に基づき，我々は予測を用いた浮動小数点データ圧縮アルゴリズムと，それを実現する高スループットデータ圧縮ハードウェアとを研究，提案してきた [4], [5]．

本研究では，ハードウェア化に際してできる限り遅延が小さく小面積な実装を考え，直前のデータを用いた一次元多項式による予測手法を採用した．これはラグランジュ多項式による補間を用いた手法であり，連続するデータが n 次の関数に従うと仮定した際の次の入力データを多項式に乗る値として予測するものである．本研究で用いる予測多項式を以下に示す．

$$p_i = f_{i-1}. \quad (n=1) \tag{1}$$
$$p_i = 2f_{i-1} - f_{i-2}. \quad (n=2) \tag{2}$$
$$p_i = 3f_{i-1} - 3f_{i-2} + f_{i-3}. \quad (n=3) \tag{3}$$
$$p_i = 4f_{i-1} - 6f_{i-2} + 4f_{i-3} - f_{i-4}. \quad (n=4) \tag{4}$$

ここで n は次数であり，これらは次の入力データを i 番目とした場合の式である．これまでの研究により，次数が高い程予測精度が向上することが示されており，ハードウェア化の際の面積や遅延も大きくないことから，本研究では $(n=4)$ の予測式を採用した．

本研究におけるデータ圧縮アルゴリズムを図 1，2 に示す．はじめに，計算コストを低くするために浮動小数点データのビット列を符号なし整数データとして扱えるように変換を行う．ここではデータの大小関係を維持するために，符号ビットに応じて異なる操作を行う．符号ビットが 1 の負の数の場合には全ビットを反転し，0 の場合には符

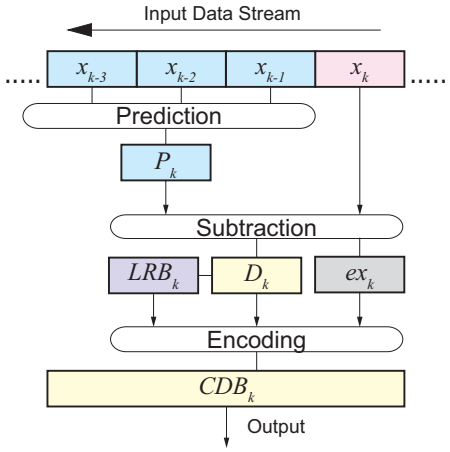

図 1　予測に基づくデータ圧縮アルゴリズム.

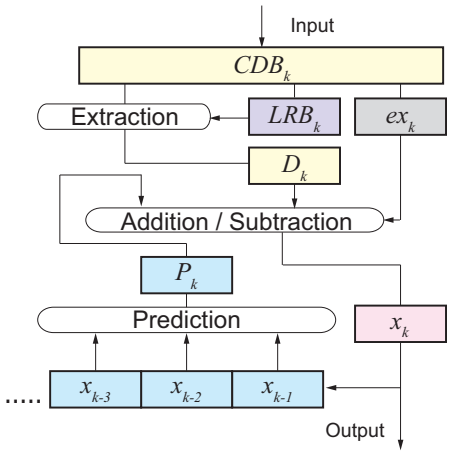

図 2　予測に基づくデータ展開アルゴリズム.

号ビットのみを反転して，符号なし整数データとみなす．次に，直前に入力された 4 つのデータを予測多項式に代入し予測値の計算を行う．予測のために直前の連続する 4 つのデータを保持しておき，予測式における各値への係数をかけ合わせた後に和算，減算を行う．得られた予測値と入力値はいずれも符号なし整数データとなっており，両値の差分を整数演算により求める．この際に予測値と入力値の大小関係を記録しておく．両値の差分は符号なし整数データとして得られるため，予測値が入力値に近い場合差分として得られるビット列には MSB から 0 が連続することになる．この不必要な 0 を除いた長さ (残差長) を Length of residual bits (LRB) として記録し，残差長が示す部分 (残差ビット) と LRB によって置き換える．最終的に，残差ビット，LRB，予測値と入力値との大小関係を示す Ex ビットの 3 つが圧縮後のデータとなる．

2.2　符号化

本研究では，圧縮後のデータを固定長のデータブロックとして出力している．これは圧縮したデータの送信時にメモリ帯域を十分に使い切るためである，データ圧縮による帯域向上は，可変長の圧縮データを固定幅のメモリ送信路に流すため，圧縮後のデータストリームの要求帯域とメモリ帯域との関係によりどのくらい計算性能が向上するかが決定する．圧縮後にデータ量が 2 分の 1 になる場合，ストールが発生せずにメモリ帯域を完全に使い切る理想的な状況では，計算回路内の帯域はメモリ帯域の 2 倍になる．このような理由から，効果的な帯域向上を行うためには計算回路自体の設計が重要になる．この点については 3 節において詳しく述べる．

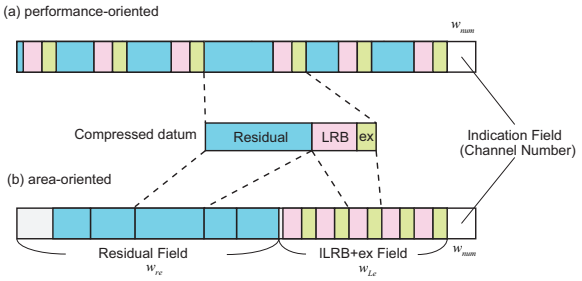

図 3　CDB の構造.

圧縮データの出力データブロックを本報告では Compressed Data Block(CDB) と呼ぶ．CDB は複数の圧縮データのビット列を連結したものであり，上記の理由から FPGA の出力ポートのビット幅と等しい長さとする．CDB の構成を図 3 に示す．圧縮性能を重視する場合，圧縮されたビット列をそのまま連結して一定の長さごとに切り出し他物を CDB として出力する．逆にハードウェアコストを重視する場合，圧縮したビット列のうち固定長部分と可変長部分を分割して CDB 内に収納する．前者の場合，CDB は連結されたビット列から個々のデータに関係なく切り出されるため，展開時に複数の CDB を同時に保持しておく必要がある．後者では，あらかじめ 1 つの CDB 内に収納される圧縮データ数が決まっているため，CDB を一つずつ展開することが出来る．このように選択肢を提供することにより，圧縮性能を重視する場合とハードウェアコストが問題になる場合とに分けて実装を変えて異なる要求に対応できる．

これに加えて，圧縮処理の簡略化によって回路面積を削減することが可能である．ハードウェアにおけるデータ圧縮において，可変長のデータから固定長のブロックを生成する処理は，データを連結するために複雑な操作が必要になる．CDB を生成する出力バッファにおいて，入力されてくる可変長圧縮データを積み上げるためには可変長のシフト処理が必要になり，レイテンシとハードウェアコストが増加する原因となる．我々のこれまでの研究においても，データブロックを生成する回路がハードウェアにおける性能を限定する部分となっていた．これに対して本報告では，可変長である残差ビットの長さを制限することによりこれらを軽減する手法を提案する．例として単精度浮動小数点データに対するデータ圧縮において，残差ビットは通常 0 から 32 ビットまでの 33 通り考えられる．その際，上記のシフト操作は 33 種類のシフトを実現する必要がある．ここで残差ビットの長さを 3 種類に限定すると，シフ

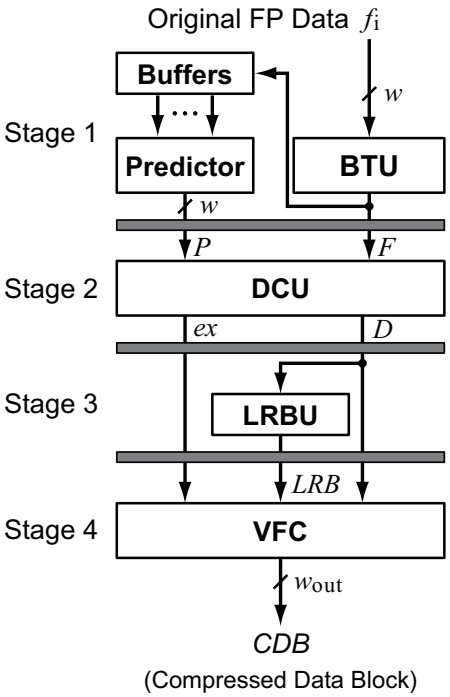

図 4　単一チャネルデータ圧縮ハードウェア.

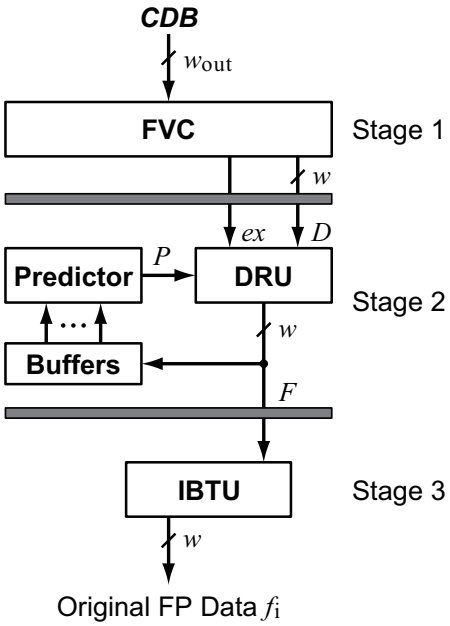

図 5　単一チャネルデータ展開ハードウェア.

トは 3 種類のみで済む．この場合には，残差ビットに不要な 0 が含まれることになり圧縮性能は低下するが，ハードウェアコストとレイテンシを大幅に削減出来る．CDB の構成及び，残差ビットの制限については 4 節において評価を行う．

2.3　ハードウェアデザイン

図 4，5 にデータ圧縮，展開ハードウェアの構造を示す．データ圧縮ハードウェアは 4 段のパイプラインとして設計されている．入力されたデータは整数変換器 (BTU) に入力され符号なし整数データへの変換される．ステージ 1 では，予測器と直前の値を保持するためのバッファがおかれ，次の入力値の予測計算が行われる．ステージ 2 では予測値と入力値との差分が差分計算ユニット (DCU) により求められ，同時に Ex が生成される．ステージ 3 において残差ビット長 LRB が生成され，ステージ 4 の variable-to-fixed length converter(VFC) 内の出力バッファで残差ビット，LRB，Ex の符号化と CDB の生成が行われる．

データ展開ハードウェアは 3 段のパイプラインであり，CDB から元のデータを抽出, 復元する．入力された CDB はステージ 1 にある fixed-to-variable length converter(FVC) 内の入力バッファにおいて，各圧縮データの要素に分解される．このバッファは圧縮データを抽出するごとにシフト操作を行い，常に先頭に次の圧縮データ要素が置かれるように動作する．固定長の LRB の位置は常に一定であるため，それを用いて残差ビット長を計算して抽出する．残差ビットは元の 0 を含むビット列に復元され，次のステージのデータ再構成ユニット (DRU) において予測値との演算により元のデータが生成される．復元されたデータは予測器と次のステージに送られる．最後のステージにおいて浮動小数点変換器 (IBTU) によって符号なし整数データから浮動小数点データに変換した後，数値データとして出力される．

3.　複数圧縮データチャネルの多重化

3.1　複数圧縮チャネル多重化の条件

数値シミュレーションにおいては数千から数万回の計算が必要となるため，圧縮データストリームが CDB として出力されメモリを経て再び入力されることが繰り返される．数値計算ではある変数を更新する際に別の変数を計算に用いるため，正しい計算を保証するためのデータのフォーマット，入力順序が守られていなければならない．一般にハードウェアによるストリーム計算では，データはある決められたフォーマットに従って計算空間を走査するように並んでいる．これが順に計算回路へと入力されるが，その際にそれぞれのチャネルに対応するように幅変換される．前節で述べたデータ圧縮を各チャネルに適用する場合，一度に 1 チャネル分の CDB しか出力されないため，計算回路の全チャネルに対して入力データが揃わない状況が発生する可能性がある．その場合場合，入力データが全てのチャネルに揃うまで計算が開始されず，データ待ちのためのストールが生じ，極端な場合にはデッドロックが発生することが考えられる．

一方，個々の圧縮データは可変長のため，固定長の CDB に含まれているデータの数は常に変動する．このため均等に帯域を分配する時分割多重化を適用した場合，圧縮率の高いチャネルの CDB には低いチャネルよりも多くのデータが含まれていることから，次の計算回路の入力においてチャネル間のデータ供給の偏りが生じる．例えばチャネル

A がチャネル B の 2 倍の圧縮率である場合，チャネル A から出力される CDB にはチャネル B の 2 倍のデータが含まれている．このような圧縮後のデータをチャネル間で均等に送信した場合，チャネル A の入力側にはデータが過剰に供給され，B の側には入力データが不足する状況になる．入力部分に FIFO などを用いることによって動作は可能だが，その場合にはハードウェアコストが膨大になる．

3.2 多重化方式

CDB に含まれる元データの数は，その時点における圧縮率に比例する．また，圧縮率が高いチャネルほど CDB の出力頻度は低下する．このことから，CDB の生成順に従ってデータの出力順序を決定すれば，計算回路の入力における同期をある程度保証できる．このような方式には，データ圧縮，展開ハードウェアにおいて，各パイプラインステージも計算回路と同様に同期して動作する必要がある．そこで，我々は計算回路とデータ圧縮，展開ハードウェアを全てのチャネルについて完全に同期して動作する仕様とした．

CDB の生成順に応じて出力順序を決定するために，データ圧縮ハードウェアからの出力要求を用いた出力制御を行う．圧縮後に CDB が生成された時点で出力要求信号が発せられる．それに対して制御側から出力許可信号が 1 チャネルに限って送られる．出力許可信号は出力要求が送られてきた順に各チャネルに送られ，両方の信号が立った時点で出力が確定する．このような手法では，1 チャネルのみが出力要求を出している場合にはそのチャネルに許可が与えられる．複数のチャネルから同時に出力要求が送られている場合には，あらかじめ決められているチャネル間の優先順位に従って出力許可を与える．

出力許可をできるだけ公平に与えるためには，各チャネルからの出力要求の順序を記録しておき，記録された順序に従って出力許可を与える方式が考えられる．ただし，完全に出力要求順序を記録する方式では，制御回路の大きさや遅延の増大が問題になるため，本研究では同時に出力要求を受け取った場合のみその信号を記録しておき，記録された信号がなくなるまであたらしい出力要求を受け付けない方式とする．複数の出力要求から 1 つのチャネルだけに許可を与える場合には，各チャネルからの出力要求をまとめたビット列 R_{out} に対して以下のような式により許可信号ビット列 E_{out} を生成する．

$$E_{out} = (R_{out} XOR (R_{out} - 1)) AND R_{out}. \quad (5)$$

この処理により，R_{out} における最も下位の 1 だけが取り出される．よってこの手法では，R_{out} の下位ビットに位置するチャネルが優先的に出力される．

入力側では，図 3 において示した CDB 内のチャネル番号を示す付加ビットにより正しいチャネルへと振り分けられる．入力側のデマルチプレクサは単純にこの付加ビットに応じて，データを分配する．データ展開ハードウェアへの入力は，全てのチャネルに CDB がそろった時点で開始される．

3.3 多重化ハードウェア

このような多重化を実現するハードウェアとして，我々は Multiple-Channel Encoder(MCE) と Multiple-Channel Decoder(MCD) を設計した．MCE は圧縮された複数チャネルに対して多重化と出力のスケジューリングを行い，MCD はメモリから送られてくる単一チャネルの圧縮データを正しいチャネルへと分配する．どちらのハードウェアも様々なチャネル数に対応しつつハードウェア性能を維持するためにツリー型の構造になっており，ツリーの結合点にあたる部分でそれぞれ入力チャネルに対して制御を行う．

図 6 に MCE のデザインを示す．MCE は 4 チャネルごとに出力を選択する複数のセレクタから構成され，階層的な構造となっている．例として 64 チャネルの場合，MCE は 3 ステージからなり，1 段目に 16 個，2 段目に 4 個，3 段目に 1 個の計 21 個の 4 チャネルセレクタが含まれている．動作周波数の向上のためパイプライン化を行い，チャネル数に応じてパイプライン段数も変化する．4 チャネルセレクタは前述した多重化方式に従って 4 つのチャネルのうち最大 1 つに出力許可を与える．そのため，下の段に行くほどセレクタへの入力頻度が高くなる．セレクタには出力要求信号を記録するレジスタが置かれ，複数の信号が同時に入力された場合にはこのレジスタに記録される．レジスタに記録が残っている場合，優先的にレジスタ内の記録チャネルが出力され，記録がない場合には入力に応じて出力が決定される．

次に，図 7 に MCD のデザインを示す．MCD は 8 チャネルごとに入力データを分配する複数のディストリビュータから構成される．64 チャネルの場合，MCD は 2 ステージからなり，1 段目に 1 個，2 段目に 8 個の計 9 個の 8 チャネルディストリビュータが含まれている．ディストリビュータは CDB 内に含まれるチャネル番号ビットを用いて元のチャネルへと CDB を分配する．64 チャネルの場合，チャネル番号ビットは 6 ビットとなるので，1 段目のディストリビュータは番号ビットの上位 3 ビットを参照し，2 段目では下位 3 ビットを参照する．MCD の出口には各チャネルに FIFO が置かれ，データ展開ハードウェアへの同期入力を保証している．

4. 評価

本研究の目的は，データ圧縮を用いた数値計算ハードウェアの帯域向上である．これまでの研究において提案した圧縮アルゴリズムは，数値計算データに対して高い圧縮性能を実現することが示されている．本稿では，まず前述したデータ圧縮における符号化手法について，圧縮性能と回路面積の関係について評価を行い，実装するハードウェアの選定を行う，また，実際に複数チャネルのデータ圧縮，

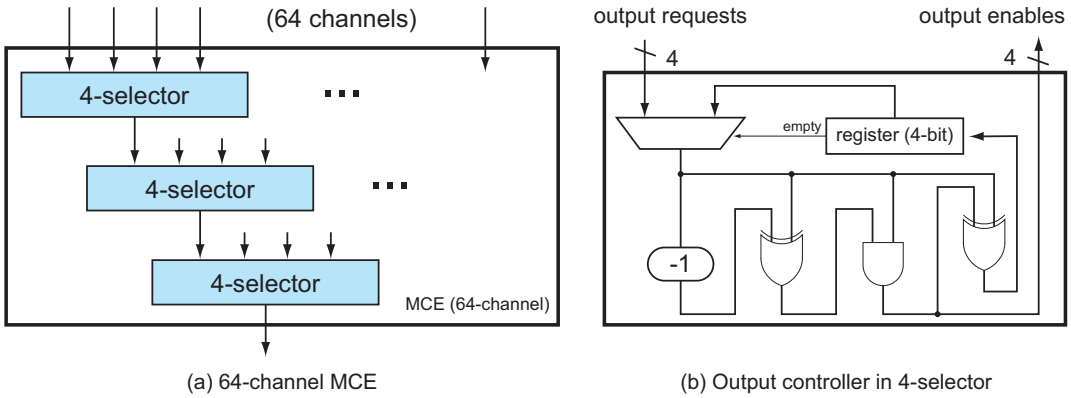

図 6　Multi-channel Encoder (MCE).

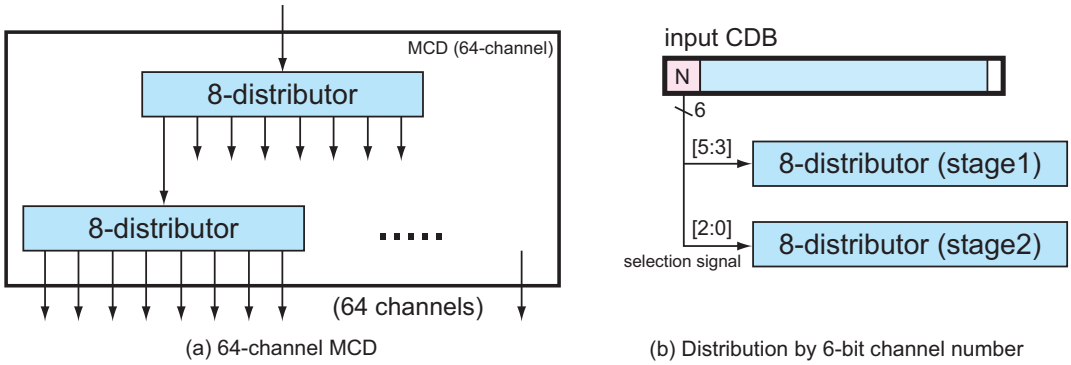

図 7　Multi-channel decoder (MCD).

展開ハードウェアを MCE，MCD とともに実装し，どの程度帯域を向上させることが出来るかを調査する．

4.1　回路面積と圧縮性能

2 節において提案した 2 種類の圧縮データの符号化について評価を行い，圧縮性能と回路面積の関係について評価するとともに，実際の実装に適したデザインを選定する．しかしながら，データ圧縮性能は対象のデータ自体の性質によって大きく左右される．本研究における予測に基づくデータ圧縮では，これまでの研究から計算対象の複雑さや計算格子における格子点間距離などによって，大きく圧縮性能が変わることが明らかになっている．データの種類による圧縮性能の偏りを排除するために，評価にはモデル化した残差ビット長の分布を用いる．ここでは，正規分布によって与えられる残差ビット長に対して，各符号化における圧縮性能をソフトウェアを用いたシミュレーションにより評価する．正規分布に代表される頂点とその周りになだらかに集まる残差ビット長の分布は，これまでに評価した全ての数値データにおいて確認されている．正規分布を用いることによって，頂点の位置 (期待値 σ) や分布の広がり (標準偏差 μ) を自由に変化させた評価用のデータを生成できる．この評価により得られた圧縮性能とハードウェア上に実装した際の回路面積により，提案した符号化手法についての評価を行う，

評価する符号化手法は，圧縮性能を重視した LRB をそのまま用いるものと，回路面積を重視した LRB を 3 つの値に制限するものである．圧縮対象は 32 ビットの単精度浮動小数点データとし，制限する LRB の組は (2，4，32)，(4，8，32)，(8，16，32) の 3 組とする．これらの符号化手法に対して，正規分布による LRB のモデル分布を圧縮した場合について，圧縮率を評価する．評価対象の LRB の分布として，σ の値を 1 から 31 まで，μ を 1 から 5 まで変化させた場合についてシミュレーションを行う．

図 8 に μ が 1，3，5 の場合の σ に応じた各符号化手法における圧縮率を示す．LRB が小さくなるにつれて圧縮性能が向上するのは共通しているが，LRB を制限する符号化では，とくに分布が一部分に集まっている場合に特定の領域において圧縮性能重視の符号化よりも高い性能を得ていることが分かる．一方で，特に分布が広い領域に跨っている場合には圧縮性能重視の符号化が全ての場合において高い圧縮率を示している．また，LRB を制限する場合も選択する値が大きい程，幅広い分布に対応した圧縮性能を発揮していることが分かる．全体として，圧縮性能については LRB をそのまま用いる符号化が最も優れており，制限する符号化では極端に狭い分布に対して高い圧縮性能を示す場合があることが示された．

次に，同じ符号化手法 4 種類に対して実装した場合の回路面積を評価した．用いた FPGA は ALTERA 社の Stratix V　GX5SGXEA7N2F45C2 であり，ハードウェアの記述は verilog HDL により行い，同社のハードウェア開発ツー

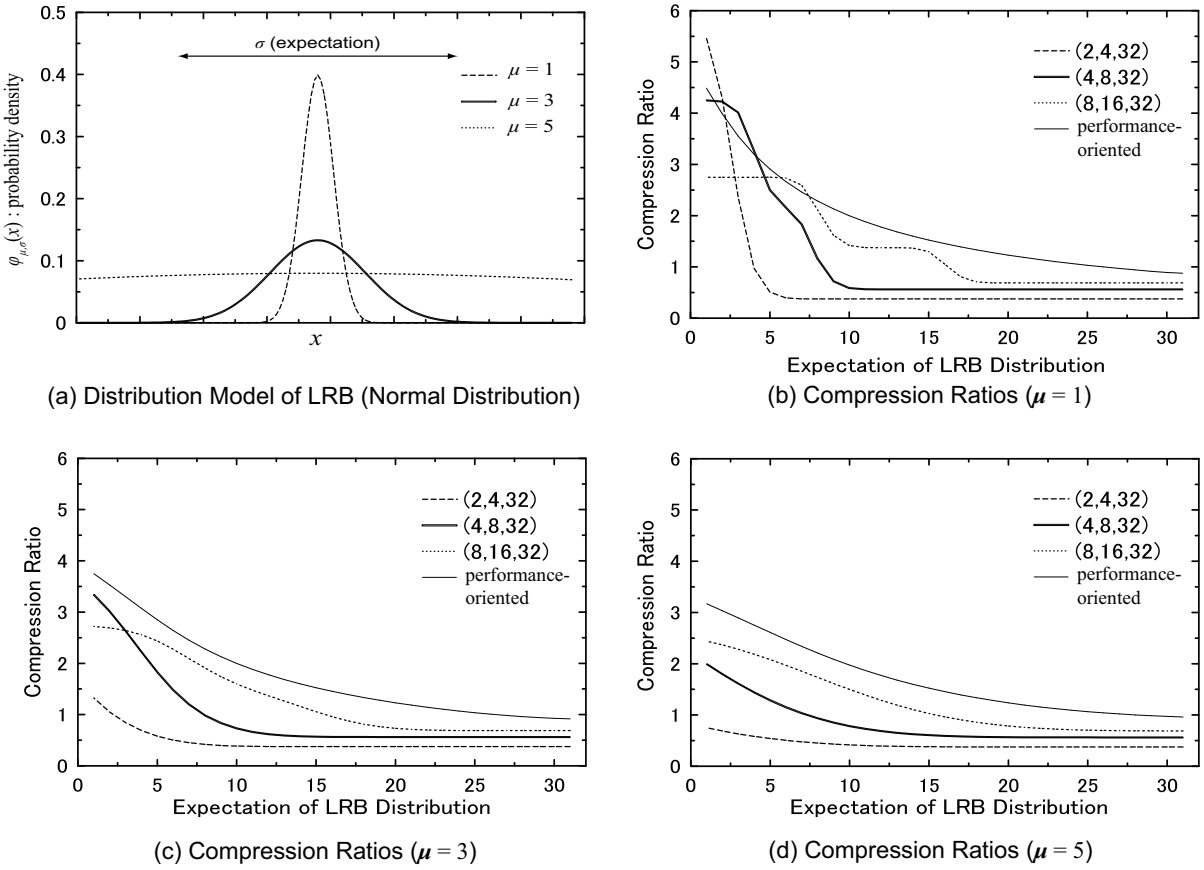

図 8　LRB 分布モデルに対する圧縮性能評価.

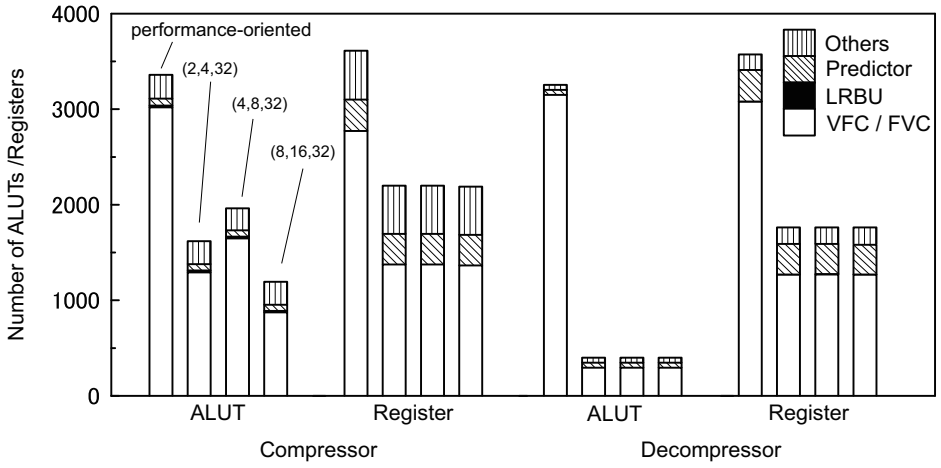

図 9　単一圧縮，展開ハードウェアの回路面積

ルである Quartus II により回路のコンパイルと面積評価を行った．評価したのは単一チャネルに対するデータ圧縮，展開ハードウェアである．結果を図 9 に示す．結果から，回路面積重視の符号化手法の回路面積が，性能重視の場合よりもかなり小さくなった．更に制限した場合についても，選択した LRB の組における最大公約数が大きい程，回路面積が小さくなっていることが分かる．これは，残差ビットを積み上げたビット列が取り得る長さの取り得る値が，最大公約数が大きい程限定されるためである．(2,4,32) の場合，蓄積残差ビット列は 2 ビット単位でビット列が伸びるのに対し，(8,16,32) の場合，蓄積されるビット列の長さは 8 ビット単位となるため，ハードウェア資源をより節約できる．

以上の結果から，実装に用いる符号化方式として (8，16，32) を選択した．圧縮性能は LRB に制限がない符号化の次に高い性能を発揮し，回路面積では最も優れた結果を残したためである．また，3 節において述べたとおり，圧縮による効果をできるだけ高めるにはチャネル数を増やすべきであるという点から，多数のチャネルへの適用も可能な小面積なハードウェアを選択した．

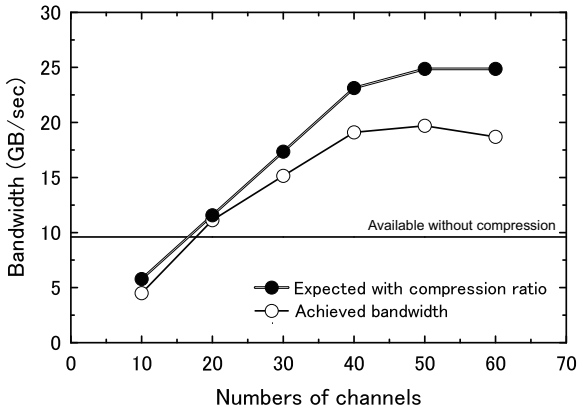

図 10 データ圧縮による計算帯域向上.

4.2 帯域向上の評価

データ圧縮による帯域向上について評価するために，FPGA 上にデータ圧縮，展開ハードウェアと MCE，MCD を接続して実装し，実際の数値計算データをストリームとして FPGA に入力し，データ展開とデータ圧縮の間の帯域を調査した．評価に用いたデータは，一様流れ中に平衡平板がある 2 次元計算空間を格子ボルツマン法 (2DLBM) により計算したもので，5000 から 60000 までのタイムステップにおいて 5000 イテレーション置きに計算途中のデータを取り出したものを用いて，その平均値を調べた．対象とするチャネル数は 2DLBM における変数にあわせて 10 から 60 まで 10 ずつであり，データは 32 ビットの単精度浮動小数点数，CDB は 512 ビット，動作周波数は 150MHz である．用いたメモリは 2 つの DDR3-1600 であり，DMA 転送により FPGA との間の通信を行った．

結果を図 10 に示す．縦軸は実現した計算帯域であり，横軸はチャネル数である．計算データは格子点数が 1440x480 であり，比較的高精度な計算である．評価において，計算帯域とともに実際の圧縮率を調査し，そこから得られる最大の帯域についてもグラフ中に示している．10 チャネルの場合，最大の計算帯域は $(4Byte \times 10 \times 150MHz = 6.0GB/s)$ であるため，150MHz における利用可能なメモリ帯域である 9.6GB/s よりも低く，圧縮による効果はない．20 チャネル以上になると，設計上の最大計算帯域が 9.6GB/s を超えるため，圧縮の効果が結果に表れている．全体の平均圧縮率は 2.59 であったため，理論上最大でメモリ帯域の 2.59 倍の計算帯域が得られる．実際の結果としては，50 チャネルの際にメモリ帯域の 2.05 倍の計算帯域が実現できている．

この圧縮率を基に考えると，チャネル数が 48 の時にチャネル数の増加による帯域向上は頭打ちになる．理想的には 48 以上にチャネル数が増加しても計算帯域は横ばいになるはずであるが，実際には 60 チャネルにおいて 50 チャネルよりも低下している．これは，チャネル数が増加したことによって展開ハードウェアの入力待ちによるストールが発生しているためだと考えられる．チャネル数が大きい場合，データが常に MCE 空出力され続けるため，スケジューリングの効果が低下して入力時の同期が困難になるためである．

5. おわりに

本報告では，データ圧縮によるハードウェア上における数値ストリーム計算のメモリ帯域向上手法について述べた．研究目的は，計算性能を制限するメモリ帯域に対して，データ圧縮，展開ハードウェアを適用して数値計算における計算帯域をメモリ帯域以上に向上させることであった．我々のこれまでの研究により実現していた，単一チャネルに対するハードウェアによる圧縮，展開を，複数チャネルへと拡張し実際の数値計算への適用が可能なシステムを構築した．その際に，多数のチャネルに適用するための面積削減手法と，それに伴う圧縮性能の低下について評価し，圧縮性能の低下を抑えつつ大きく回路面積を削減することに成功した．さらに，提案したハードウェアを用いた実機評価において，FPGA 上に実装した圧縮システムにより計算帯域を利用可能なメモリ帯域の約 2 倍に向上させることに成功した．

今後の研究として，実際の数値計算回路とともに評価を行い，演算性能がどの程度向上するかを実機上で評価する．また，データ圧縮によるメモリアクセス回数の削減に注目し，それによる電力消費の削減についても定量的な評価を行う．

謝辞 本研究の一部は、特別研究員奨励費 42000987 の支援により行われた．

参考文献

[1] Paruj Ratanaworabhan and Jian Ke and Martin Burtscher, Proceedings of Data Compression Conference, 2006, March.
[2] Bharat Sukhwani and Bulent Abali and Bernard Brezzo and Sameh Asaad, High-Throughput, Lossless Data Compression on FPGAs, 2011 IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines, May, 2011.
[3] Hisanobu Tomari and Mary Inaba and Kei Hiraki, Compressing Floating-Point Number Stream for Numerical Application, 2010 First International Conference on Networking and Computing, Nov, 2010.
[4] Tomohiro Ueno and Yoshiaki Kono and Kentaro Sano and Satoru Yamamoto, FPGA-based Implementation of Compact Compressor and Decompressor of Floating-Point Data-Stream for Bandwidth Reduction, Proceedings of the 2012 International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'12), 2012.
[5] Tomohiro Ueno and Yoshiaki Kono and Kentaro Sano and Satoru Yamamoto, Parameterized Design and Evaluation of Bandwidth Compressor for Floating-Point Data Streams in FPGA-based Custom Computing, Proceedings of the International Symposium on Applied Reconfigurable Computing, 2013.