Vol.2019-ARC-234 No.16
Vol.2019-SLDM-186 No.16
2019/1/31

DMAのカスケード接続による間接ロードの高速化

柏俣 智哉[1,a)]　北村 俊明[2]　木村 啓二[1]　笠原 博徳[1]

概要：アクセラレータで計算を高速に行うためには高いデータ供給能力が重要となる．しかしながら，流体計算や構造計算などでよく利用される疎行列計算では，配列間接参照によるメモリのランダムアクセスが発生し，データ供給能力が低下してしまう．さらに，アクセラレータに対して配列間接参照を行うための非効率的なプログラム処理，あるいは複雑なメモリアクセス機構が要求される．本稿では，ローカルメモリからアクセラレータへデータを供給する構成を前提とし，主記憶から配列間接参照によりローカルメモリ上の連続領域にデータを転送する，カスケード接続されたDMA構成を提案する．本DMA構成により，アクセラレータのメモリアクセス機構をシンプルに保ったまま効率の良い配列間接参照が可能となる．本DMA構成をFPGAにより実装し，疎行列–密ベクトル積を用いて評価を行った．評価の結果，同じくFPGA上に実装されたスカラプロセッサと比較し，提案手法とベクトルプロセッサを組み合わせたシステムで最大12.1倍の性能を得られた．

1. はじめに

配列間接参照は様々なアプリケーションで用いられている一方，メモリを過度にアクセスすることからプログラム高速化の障壁となる．配列間接参照では，まずインデックスをロードした後，アドレスを計算しデータをロードしなければならない．ここで，インデックスのロードが終わらなければアドレスが計算できないためパイプラインストールが発生する．また，データのロードに関してはランダムアクセスになる場合があり，これによりメモリアクセスレイテンシがさらに増大しストールが長引き，結果として計算効率の低下につながる．

配列間接参照が主に問題となるアプリケーションとしては流体計算や構造計算でよく用いられる有限要素法が挙げられる．これらの計算で規模の大きい連立一次方程式の解を求めるためには間接法であるCG法(共役勾配法)を用いると速く計算できる場合が多い．CG法では疎行列–密ベクトル積が主要カーネルであり，これを高速化することが全体の計算を高速化する上で重要である．疎行列–密ベクトル積の高速化手法としては主にデータのオーダリング，収束しやすい前処理手法，疎行列の格納形式の工夫が挙げられる．いずれの工夫をしても入力ベクトルから必要なデータだけを抽出するギャザーの操作が必要であることから，性能はメモリのランダムアクセス性能に左右される．

計算機の計算能力向上のために，GPUをはじめとしたアクセラレータが広く利用されている．これらアクセラレータの能力を十分に引き出すためには，アクセラレータに絶えずデータを供給し続ける必要がある．しかしながら，配列間接参照を伴うプログラムの場合は前述の通り，(1) インデックスのロードとデータのロードといった2段階のメモリアクセスが必要となる，(2) データのロードがランダムメモリアクセスとなる，といった特徴がある．そのため，アクセラレータに非効率的なメモリ操作を要求される，あるいは上記操作を行う複雑なメモリアクセス機構が要求される．すなわち，アクセラレータに十分な計算資源を用意しても，これらを活用する効率的なデータの供給は困難である．

上記の問題を解決するため，本稿ではカスケード接続されたDMAコントローラ構成を提案する．本提案手法はまず，アクセラレータがローカルメモリを持つ構成を前提とする．また本DMA構成では，1段目のDMAコントローラによりインデックスをロードし，2段目のDMAコントローラによりデータをローカルメモリに転送する．これにより，主記憶に対して配列間接参照を行い，ローカルメモリ上の連続領域にデータをロードする．このような構成を取ることにより，アクセラレータのメモリアクセス機構がシンプルに構成可能となり，かつアクセラレータの計算とデータ転送のオーバーラップによりプログラム実行時間の短縮を可能とする．

[1] 早稲田大学大学院基幹理工学研究科
School of Fundamental Science and Engineering, Waseda Univ.
[2] 早稲田大学 アドバンスドマルチコアプロセッサ研究所
Advanced Multicore Processor Research Institute, Waseda Univ.
[a)] kashi@kasahara.cs.waseda.ac.jp

ⓒ 2019 Information Processing Society of Japan

Vol.2019-ARC-234 No.16
Vol.2019-SLDM-186 No.16
2019/1/31

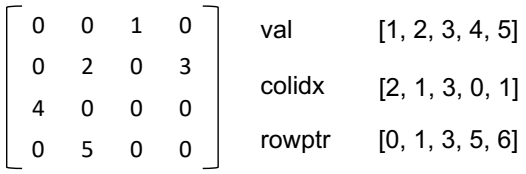


図 1: CSR 形式

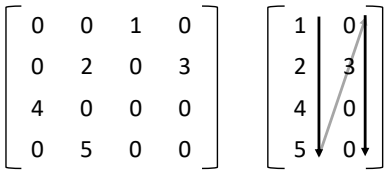


図 2: ELL 形式

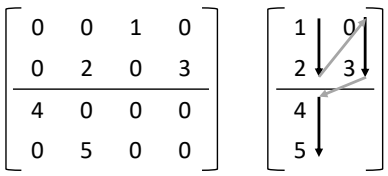


図 3: SELL 形式

```
for(i = 0; i < N;i++)
  out[i] = in[idx[i]];
```

図 4: ギャザー転送のコード例

本稿の構成は以下の通りとなる: 第 2 節で対象となる疎行列の代表的な格納形式を紹介する．第 3 節で提案する DMA 構成手法を述べる．第 4 節で評価について報告する．第 5 節で関連研究を紹介し，第 6 節でまとめる．

2. 疎行列–密ベクトル積

疎行列–密ベクトル積は配列間接参照の性能が全体の性能を左右する代表的なアプリケーションであり，本稿の評価で利用する．疎行列密ベクトル積が配列間接参照の性能に左右されるのは，演算強度が低いことから非零要素が存在する入力ベクトルの要素を集約する時間が相対的に長くなるためである．また，疎行列の特徴や使用する計算機の特徴により最適な格納形式は変化し，適切な格納形式を利用が性能向上のために重要である．これまでに COO 形式，CSR 形式，ELL 形式，JDS 形式など多くの格納形式が提案されている．本節では評価に用いた CSR 形式，SELL 形式と SELL 形式の元になっている ELL 形式について説明する．

2.1 CSR 形式

CSR 形式は最も一般的な疎行列の格納形式の一つである．図 1 に疎行列とそれを CSR 形式に変換したものを示す．行列を行方向に圧縮したもので，value, colidx, rowptr の 3 つの配列で表現される．value は行列の値をつめたものであり，colidx は列番号を記録したもの，rowptr は各行が格納されている先頭の添字を表現したものである [1].

2.2 ELL 形式

ELL 形式は図 2 に示すように行列を左側に寄せた後，最も長い行に合わせる形で 0 を埋め込むことで長方形の行列に変形し，更にその行列を左上から列方向に走査し値と列の番号を保存していく．このようにすることで隣り合うデータの間に依存関係がなく，水平加算が不要になるため SIMD 化およびその後の高速化が容易となる．一方で行ごとの要素数にばらつきがある場合，埋め込む 0 の数が多くなり，実行効率が悪化する．

2.3 SELL 形式

SELL 形式は図 3 に示すように疎行列をパラメータ S ごとに分割した後にそれぞれを ELL 形式でエンコードしたものである．S 行ごとに分割することによって 0 のパディングはセグメント内の最も要素数の多い行に合わせれば良く，ELL 形式と比較してパディングの量を減らすことができる [1]．また，帯行列に近い疎行列であった場合に ELL 形式では行列が持つローカリティを活かしにくかったが，S を調整することによりローカリティを高めることが可能になる．さらにセグメントの幅を SIMD 幅の整数倍にすることでアクセラレータの SIMD 演算器を効率よく利用することができる．セグメントごとに分割して実行することが容易であることから，ローカルメモリの利用やマルチコア化を容易にすることができる．これらの利点から今回の評価に用いることにした．

3. DMA のカスケード接続

本節で提案する DMA コントローラの構成手法とその利用例を説明する．

本節では，ギャザーと呼ばれる読み込み側が一段の間接参照となるケースを中心に扱う．ギャザー転送の例を図 4 に示す．図中，idx の部分をインデックス，飛び飛びにアクセスされる in をデータとしている．この転送の場合は図 5 のように前段の DMA がインデックスを連続的にロードし，それをアドレスに変換，データ転送用ディスクリプタを生成し後段の DMA の入力とすることにより間接参照を行う DMA を構築する．

前段の DMA は基本的に通常の DMA のリード部分のみを取り出したもので，連続領域のデータをストリーミング転送する．その後，後段の DMA のディスクリプタの形式に合わせてアドレスの計算及びデータのフォーマットを揃える．後段の DMA はディスクリプタに従ってロードしたインデックスから得られたアドレスからデータをロードし格納先アドレスに転送する．DMA のインターフェイスが

ⓒ 2019 Information Processing Society of Japan

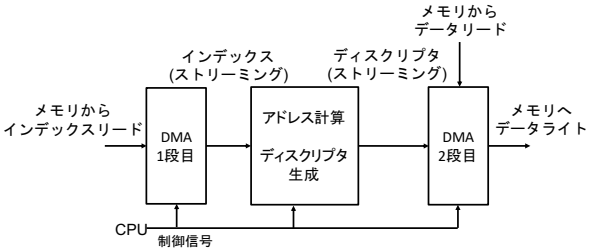


図 5: ギャザー転送の場合の DMA の構成例

```
for(i = 0; i < N;i++)
  out[idxout[i]] = in[idxin1[idxin2[i]]];
```

図 6: 多段間接参照の例

```
for(i = 0; i < N;i++)
  tmparr1[i] = idxin1[idxin2[i]]
for(i = 0; i < N;i++)
  tmparr2[i] = in[tmparr1[i]]
for(i = 0; i < N;i++)
  out[idxout[i]] = tmparr2[i]
```

図 7: 多段間接参照を一段の間接参照に分解した例

メモリマップドなレジスタしか持たない場合には適切な変換を行うことが必要となる．

3.1 複雑な転送

3.1.1 スキャター転送

この DMA の構成はギャザー転送だけではなくスキャター転送にも利用可能である．スキャター転送を行うためには後段の DMA に対して送出されたアドレスを書き込み先として利用すれば良い．そのためにはフォーマット変換部分でディスクリプタの書き込み先アドレスをインデックスから計算したアドレスにすることで実現できる．

3.1.2 多段間接参照

今までは一段の間接参照を例として見てきた．間接参照が二段以上連なる場合の対処方法としては二通りあり一つ目は DMA を増やし，チェインの回数を増やすものである．

例えば，図 6 に示すような多段の複雑な間接参照が発生したときは，idxin1，idxin2，in，idxout の 4 つに対する read の DMA と out に対する write を DMA をカスケード接続することで DMA による転送が可能になる．二つ目の手法としては一段の間接参照を繰り返し行う方法で，例の場合では図 7 のように一段の間接参照の繰り返しに分解することで一段の間接参照を行う DMA しか持たないシステムでも実行可能となる．

3.2 想定するメモリアーキテクチャ

本 DMA 構成の特性を活かすためのメモリアーキテクチャを以下に述べる．十分な性能を得るためにはインデックスのロードとデータのロード及びデータのライトが同じメモリではなく異なるメモリに対してなされることが望ましい．また，間接参照を行う場合，インデックスが時間的局所性を持つ場合がある．そのため，インデックスを格納するローカルメモリを用意することによって主記憶へのアクセスを低減することができる可能性がある．具体的には 1 段目の DMA がローカルメモリに同時にインデックスを配置し，次からインデックスをローカルメモリからのロードに切り替えることで主記憶へのアクセスを低減することができる．また，この切り替えの管理をコンパイラを用いて行わせることによってソフトウェアの開発工数を掛けることなく上記の構成が利用可能となる．

4. 評価

FPGA 上に提案手法を実装し，その上で疎行列密ベクトル積を実行することで評価を行った．実装は Intel 社製 FPGA が搭載された C5P development Kit(以下 C5P) を用いた．評価環境の詳細は 4.1.3 節で述べる．

4.1 ハードウェアの実装

実装したハードウェアの構成図を図 8 に示す．システムは CPU とベクトルアクセラレータ，連続転送用の DMA，提案手法の DMA からなり，オンチップメモリであるローカルメモリとメモリコントローラを介して DDR3 に接続されている．

4.1.1 カスケード DMA の実装

カスケード DMA は 2 つの連続転送用 DMA とインデックス–アドレス変換によって構成されている．今回の評価ではインデックスの一要素のサイズとデータの一要素のサイズが同じであるため，DMA の構成を図 9 のように一段目の DMA がインデックスの読み込みとデータの書き込みを行うようにしても差し支えない．このようにすることによって，CPU からは一段目の DMA の制御だけで DMA 全体を管理することが可能になる．また，データの書き込みを 1 段目が行うことによってディスクリプタ生成において書き込み先の管理を行う必要性がなくなるメリットも存在する．一段目と二段目の 2 つの DMA は Intel が提供する Modular Scatter-Gather DMA を用いて構成することができる．Modular Scatter-Gather DMA は詳細な設定が可能であり，取得したデータをストリーミングする機能やストリーミングされているデータをメモリに書き出す機能を標準で持つ．また，プリフェッチという，ディスクリプタをメモリから DMA 自身が取得する機能を持っており，この機能は Prefetcher がメモリから取得したディスクリプタを DMA のディスパッチャにストリーミングインターフェイスを介して転送をすることで実現されている．この Prefetcher の出力に合わせてディスクリプタを生成するようにし，Prefetcher と置き換えることで 2 段目の DMA のディスクリプタをストリーミングすることが可能になる．

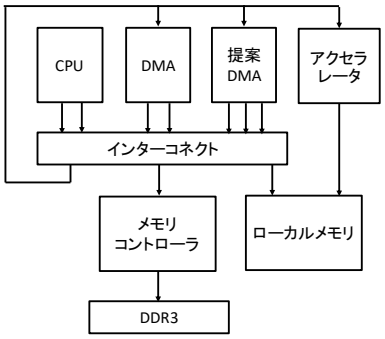


図 8: 実装したシステム

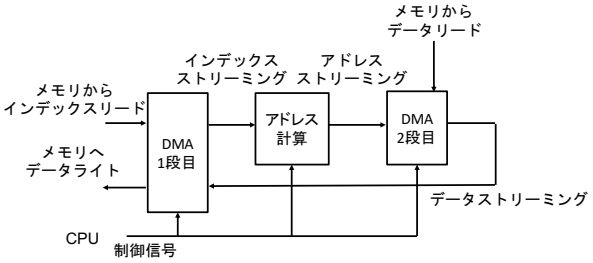


図 9: 実装したカスケード DMA

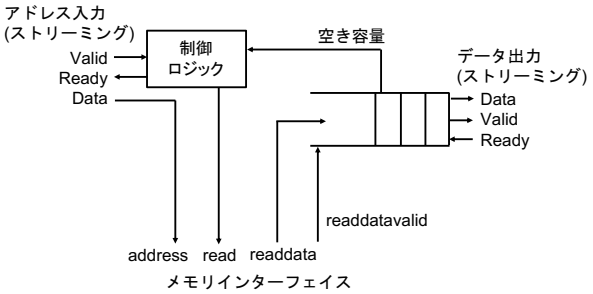


図 10: 2 段目以降の間接参照を行う DMA の構成例

しかし，この方法によって制作された DMA は Modular Scatter-Gather DMA がディスクリプタ間でのパイプライン化に対応していないため，ディスクリプタ間のパイプライン化に対応した新規の DMA と置き換えた．新規の DMA の構成図を図 10 に示す．この DMA はアドレスをストリーミングで受け取りメモリに対してリードコマンドを出すものである．メモリからのレスポンスがあったときにストリーミング先が受け入れ可能であるとは限らない．そのため FIFO を挿入し予め受け入れ可能な数だけコマンドを出力するような管理を行い FIFO が溢れないようにしつつスループットを向上させている．

4.1.2 ベクトルアクセラレータ

今回評価に用いたアクセラレータは筆者等が開発している OSCAR ベクトルマルチコアアーキテクチャ [5] で用いているものである．現在はベクトル命令は単精度浮動小数点のみ実装されており，8 個の加算と 8 個の乗算を同時に行うことが可能である．ベクトルプロセッサこのアクセラレータ自体は主記憶にアクセスしないことを前提に制作されており，すべてのメモリアクセスはローカルメモリを経由することになる．主記憶–ローカルメモリ間のデータ転送には自立動作型のデータ転送ユニット (DTU) を利用することが前提となっている．このベクトルアクセラレータは In-order であるが，ベクトル命令の動作中にスカラ演算をオーバーラップして動作させることが可能であり，アドレス計算や条件分岐などのオーバーヘッドを低減している．

4.1.3 評価環境

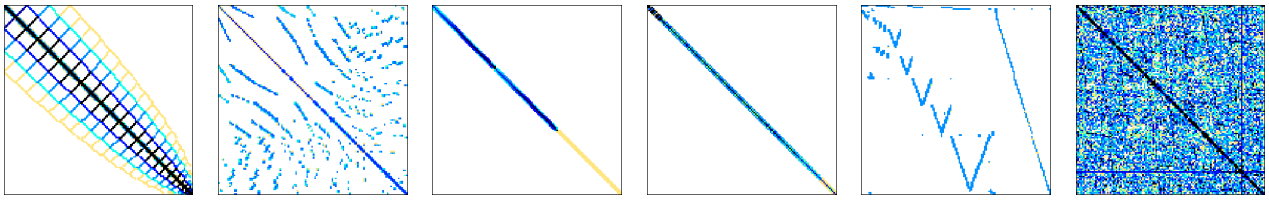
評価には C5P development Kit(以下 C5P) を用いた．C5P には intel 社の Cyclone V FPGA (330K LE)，1GB DDR3，64MB SDRAM が搭載されている．今回の評価では SDRAM は使用せず，DDR3 のみを利用した．CPU には NIOS II /fast[2] に NIOS II Floating Point Hardware2 Component[3] を付加したものを利用している．I$および D$は 32KB ずつであり，ローカルメモリも 32KB 用意した．コンパイラには nios2-elf-gcc を用いた．最適化オプションには O3 を用いた．

この DMA のみをメモリコントローラに接続して FPGA のコンパイルを行ったときこの DMA の FPGA のリソース使用量は 637.3ALM であった．また，動作周波数は Slow 1100mV 85C Model で 94.5MHz であった．システム全体を実装したとき，全体のロジック使用量は 52,592ALMs(46%) で DSP は 89 個 (26%)，M10K は 864 個 (71%) であった．このうちアクセラレータは 36136.5ALMs，80 個の DSP，418 個の M10K を消費しており，通常の DMA は 431.5ALMs，20 個の M10K を消費している．提案手法の DMA は 704.9ALMs，2 個の DSP，10 個の M10K，CPU は 1164.3ALMs，3 個の DSP，74 個の M10K を消費している．ローカルメモリとアクセラレータは 45MHz，それ以外は 50MHz で駆動させ評価を行った．

4.2 ソフトウェアの実装

今回の評価には疎行列–密ベクトル積を用いた．カーネルは初期出力ベクトルに疎行列と密ベクトルをかけ合わせたものとし，係数は掛けないものとした．入力ベクトルは 0,1,2,... とした．疎行列の値及び入出力のベクトルは単精度浮動小数点数を用い，疎行列中の整数はすべて 32bit の整数型を用いた．評価は以下の 6 ケース行った

- CPU
  - CSR 形式
  - CSR 形式 (提案手法適用)
    ギャザー転送が終わるまでビジーウェイト
  - SELL 形式
- CPU + アクセラレータ
  - SELL 形式 (通常の DMA 転送)
    連続転送の部分のみ通常の DMA で転送
    ギャザー部分は CPU 転送
  - SELL 形式 (提案手法適用)



(a) benzene (b) msc01440 (c) shyy41 (d) bcsstk25 (e) lhr11c (f) coupled

図 11: 評価に用いた疎行列の非ゼロ要素の分布 [4]

連続転送部分は通常の DMA で転送
ギャザーは提案手法で転送

– SELL 形式 (提案手法適用，オーバーラップあり)
アクセラレータと提案手法の DMA，通常の DMA をオーバーラップ実行

SELL 形式はパラメータとしてセグメントのサイズ S があるが，今回用いたベクトルアクセラレータの最大ベクトル長に合わせて 256 とした．

評価に用いた行列は University of Florida Sparse Matrix Collection[4] から選んだ．ツールチェインによる制約により過度に大きい行列を避けて選択した．一方で様々な特徴の行列を選ぶようにした．評価に用いた行列を表 1 に示す．また，それぞれの非零要素の分布を図 11 に示す．University of Florida Sparse Matrix Collection から Matrix Market 形式でダウンロードした行列を python のライブラリである scipy を用いて CSR 形式にし，さらにそれを python を用いて SELL 形式に変更した．評価において行列は予め CSR 形式および SELL 形式で C 言語の配列に変換し，メモリ上にデータが載った状態で実行するように実装した．また，定数伝播を防ぐために疎行列のデータと計算カーネルは異なるオブジェクトファイルになるようにした．

4.3 評価結果

評価結果を表 1 および図 12 に示す．評価値には 10 回測定した平均値を用いた．

SELL 形式でアクセラレータを用いて評価した場合に通常の DMA を用いた場合は CPU での実行と比べて性能向上は限定的であった．しかし，提案手法を用いた場合には CSR 形式を CPU で実行したものと比べて最大で 6.1 倍の性能向上が見られた．更に DMA による転送とアクセラレータでの実行をオーバーラップさせた場合には最大で 12.1 倍の性能向上が見られた．CSR 形式の実行においては提案手法を用いない方が良い結果を示した．これは CPU で実行した際にはキャッシュが利用されることでレイテンシがある程度小さくなりローカルメモリにプレロードするコストに見合わなかったと考えられる．coupled は SELL 形式に直した時点で CSR 形式の 6%にまで性能低下をしていた．その他の行列でも性能低下をしていたが，これは CPU で実行する場合に SELL 形式は出力ベクトルの局所性が少なくなるためだと考えられる．また，性能低下にばらつきが見られたのは SELL 形式に変換するにあたって行ごとの非零要素のばらつきによって 0 をパディングする量が変わるためであると考えられる．

5. 関連研究

汎用的に間接参照に用いることができる本研究の先行研究として，ベクトル計算機におけるリストベクタが挙げられる．また間接参照を汎用的に高速化したものではないが疎行列–密ベクトル積を FPGA を用いて高速化した研究が存在する．

5.1 ベクトルプロセッサにおけるリストベクター

リストベクターは間接参照を並列的に行うためのハードウェアである．リストベクタはインターリーブなど同時に多数のメモリアクセスに耐えられるようなメモリアーキテクチャを持つシステムを対象とし，十分な数のメモリインターフェイスから同時にコマンドを出すことで並列に間接参照を行うことを可能にしている．しかしながら，ロードストアユニットが直接メモリアクセスを行うためレイテンシによってストールが発生しうることが問題点として挙げられる．提案手法ではランダムアクセスを行う DMA が独立していることによりオーバーラップ実行が出来れば演算器のストールを削減することが可能である．

5.2 FPGA による疎行列密ベクトル積の実装

FPGA に専用ハードを実装することで疎行列–密ベクトル積を高速化を行う研究が存在する．文献 [6] では PE ごとにローカルな入力ベクトルのキャッシュとシステムで共有している行列のキャッシュを持ち乗算と加算をパイプライン的に行うことで高速化を行っている．文献 [7] では実装したアーキテクチャに適した疎行列の格納方式を提案し FPGA 中の多バンクなデータメモリにベクトルを持たせ，並列にギャザー操作を行わせており，GPU や CPU と比較して電力効率に優れている．一方で間接参照に対する汎用的な解決方法ではなく，アプリケーションごとに実装を切り替えなければならない．提案手法は汎用的に利用することが可能であり，アプリケーションやカーネルが頻繁に切

ⓒ 2019 Information Processing Society of Japan

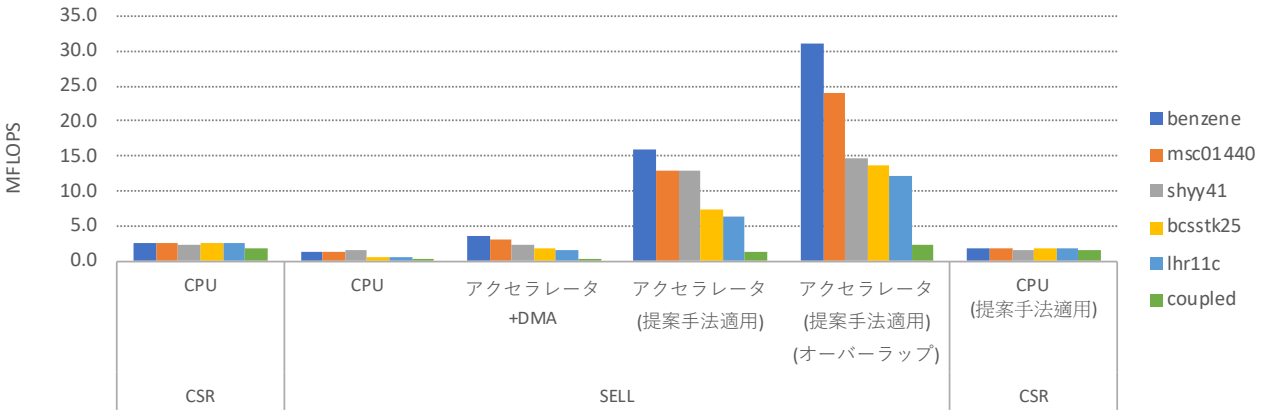


図 12: 性能評価結果

表 1: 行列の特徴と性能評価結果 (単位: MFLOPS)

				CSR	SELL				CSR
疎行列	NNZ	一辺のサイズ	一行あたりの NNZ	CPU	CPU	ACC+DMA	ACC (提案手法)	ACC (提案手法) (オーバーラップ)	CPU (提案手法)
benzene	242669	8219	29.5	2.59	1.48	3.75	15.98	31.19	1.90
msc01440	44998	1440	31.2	2.57	1.29	3.23	12.90	23.99	1.85
shyy41	20042	4720	4.2	2.34	1.64	2.34	12.88	14.82	1.62
bcsstk25	252241	15439	16.3	2.63	0.73	1.86	7.35	13.76	1.85
lhr11c	231806	10964	21.1	2.66	0.65	1.66	6.51	12.31	1.86
coupled	97193	11341	8.6	2.00	0.13	0.32	1.31	2.50	1.74

り替わる場合でも時間的オーバーヘッドが掛からない．

6. まとめ

本稿では DMA のカスケーディング接続を行うことで簡便に間接参照を行うことができる DMA 構成手法を提案した．さらに提案手法を FPGA 上に実装し評価を行った．評価を通しアクセラレータと本 DMA を組み合わせて用いた場合に最大 12.1 倍高速化することが確かめられ，有効性を確認した．

今後の課題として，マルチコアによる性能評価やコンパイラによる提案 DMA の制御などが挙げられる．

謝辞

本研究の一部は科研費挑戦的研究（萌芽）18K19786 の助成により行われた．

参考文献

[1] 佐藤駿一, 高橋大輔 : GPU における SELL 形式疎行列ベクトル積の実装と性能評価, 情報処理学会研究報告, Vol. 2018-HPC-164, No. 3, pp. 1-6 (2018)

[2] Intel Corporation: Nios II Processor Reference Guide, available from 〈 `https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/nios2/n2cpu-nii5v1gen2.pdf`〉 (参照 2018-12-20)

[3] Intel Corporation: Nios II Custom Instruction User Guide, available from 〈 `https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug\_nios2\_custom\_instruction.pdf`〉 (参照 2018-12-20)

[4] Davis, A.T. and Yifan HU: The University of Florida Sparse Matrix Collection, *ACM Transactions on Mathematical Software*, Vol 38, No. 1, pp.1:1-1:25 (2011)

[5] 宮本一輝, 牧田哲也, 高橋 健ほか: OSCAR ベクトルマルチコアプロセッサのための自動並列ベクトル化コンパイラフレームワーク, 情報処理学会研究報告, Vol. 2018-ARC-230, No.13, pp.1-6 (2018)

[6] Nagar, K.K. and Bakos, D.J: A Sparse Matrix Personality for the Convey HC-1, Proc. *IEEE International Symposium on Field-Programmable Custom Computing Machines* (FCCM), pp. 1-8, IEEE (2011)

[7] Fowers, J., Ovtcharov, K., Strauss, K., Chung, S.K and Stitt, G.: A High Memory Bandwidth FPGA Accelerator for Sparse Matrix-Vector Multiplication, Proc. *IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines* (FCCM), pp.36-43, IEEE (2014).

ⓒ 2019 Information Processing Society of Japan