Vol.2014-ARC-208 No.17
2014/1/24

演算器アレイ型アクセラレータへのメモリインテンシブなアプリケーションの写像と性能評価

林大地[†1]　藤原知広[†1]　姚駿[†1]　中島康彦[†1]

概要：我々は，科学技術演算や画像処理，大規模シミュレーションといった並列度の高いアプリケーションの実行速度を向上させる演算器アレイ型アクセラレータとして EMAX (Energy-aware Multimode Accelerator eXtension) を提案してきた．これらのアプリケーションの多くはステンシル計算を含んでおり，演算データを再利用可能な EMAX での実行に適している．しかし，EMAX 上の演算器に対してアプリケーションを写像する際には，適切な位置に命令を写像しなければデータを再利用できず，必要なデータを再度ロードしなければならない．このため，メモリインテンシブなアプリケーションを EMAX で実行するには，レジスタの依存関係以外の要素も考慮する必要がある．本論文では，メモリインテンシブなアプリケーションを EMAX 上で実行する際に，演算対象となるデータの再利用率を高める命令写像手法について述べる．アプリケーションの命令写像においてデータの再利用率を考慮した場合としなかった場合について，性能測定シミュレータにより必要な EMAX の行数とデータの再利用率を測定し，EMAX ではデータを再利用可能な形で命令を写像するのが最も効率がよいことがわかった．

1. はじめに

近年，科学技術演算や画像処理，大規模シミュレーションに含まれるステンシル計算の高速化手法に関する研究が盛んに行われている．ステンシル計算は，演算対象となるデータ量に対して計算量が多いという特徴があり，データの再利用性を生かすアクセラレーションが求められる．

我々は，演算速度の高速化と省電力化を目的として，演算器アレイ型アクセラレータ（Energy-aware Multimode Accelerator eXtension：EMAX）[1][2]を提案している．シングルポートメモリと演算器を組み合わせたユニットを２次元アレイ状に並べた構造を持ち，小さな範囲のデータを各々のユニットが保持できる．これにより，アプリケーションのループ部において前ループで用いたデータを再利用可能になり，ホストとの通信量を削減することでメモリボトルネックが解消される．しかし，大気シミュレータ GRAPES や，有限差分 FD6 といった，メモリインテンシブなアプリケーションを実行する際には，データの再利用率を確保しつつ，多くのロード命令を演算器に対して効率よく写像する必要がある．そこで，これらのアプリケーションを EMAX に対して写像し，データの再利用率を性能測定シミュレータにより評価した．

2. EMAX の概要

本章では，EMAX 全体の構成と動作の説明を行ったあと，EMAX におけるデータの再利用手法について説明する．

2.1 EMAX の構造

図 1 の全体構成図[3]が示すように，EMAX は 2 次元アレイ状に配置したユニットから構成される．各行には 4 つ

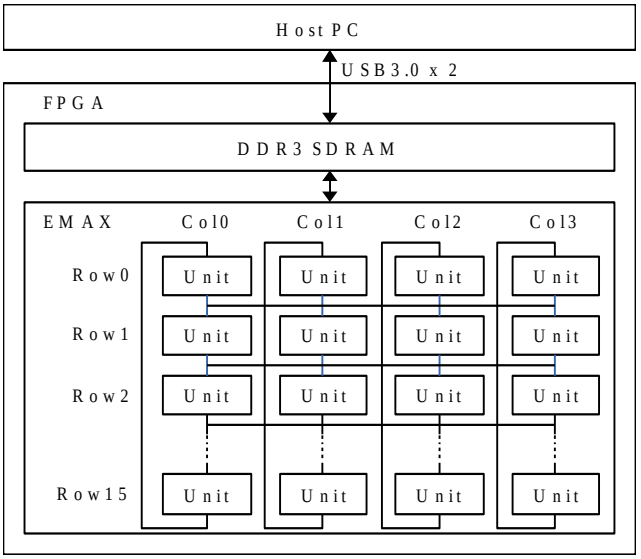


図 1　EMAX 全体構成

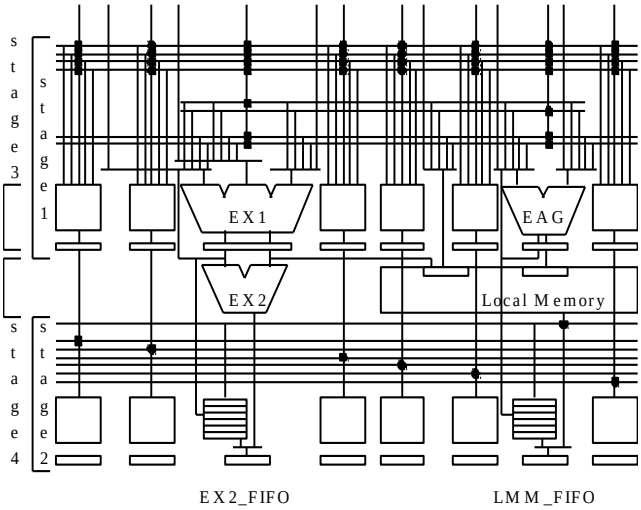


図 2　基本ユニットの構造

†1 奈良先端科学技術大学院大学
Nara Institute of Science and Technology

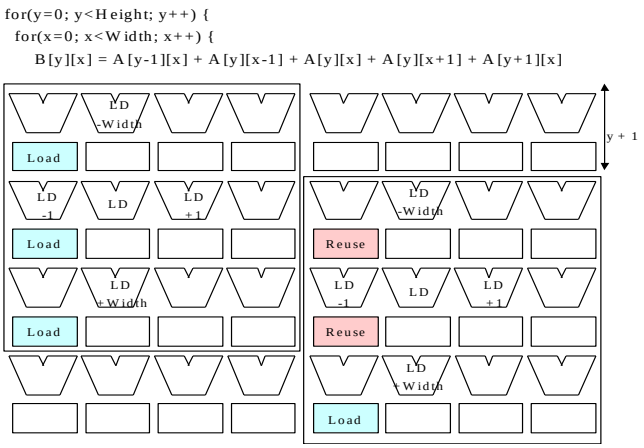


図 3　データの再利用

の基本ユニットが配置され，各々の基本ユニットは図 2 のような構造となっている．

Stage1 において基本ユニット内のレジスタ 8 本から 1 つを選択し，Stage2 において隣接ユニット間のレジスタ 4 つから 1 つを選択する．Stage3 では固定小数点算術演算または浮動小数点演算の前半（EX1）および，アドレス計算（EAG）を行い，Stage4 では固定小数点論理演算または浮動小数点演算の前半（EX2），およびローカルメモリ参照（Local Memory）を行う．EX1 はアドレス計算を行うことも可能である．なお，ステンシル計算特有の隣接要素参照のために，Local Memory から横方向にバスを配置して，EX1 および EAG のアドレス情報により参照可能な FIFO（EX2_FIFO および LMM_FIFO）へロードしたデータを共有することが可能である．

2.2　データの再利用手法

再利用可能なデータを増やし，ホストとの通信量を削減するために，EMAX では各々の基本ユニットがリング状の構造でつながっている．図 3 に，典型的なステンシル計算の例として 2 次元 5 点ステンシルのアプリケーションを EMAX に実装した場合の動作について示した．EMAX 命令は，y 回目における最内ループの実行が終わると，次の y+1 回目のループ実行における命令写像の開始位置を 1 行ずらす．これにより，前回のループ実行時にローカルメモリに読み込んだデータのうち，2 行分は共通のデータを使用するため，SDRAM からデータを改めてロードせずに再利用し，データロードにかかる時間や消費電力を削減することができる．

3.　メモリインテンシブなアプリケーション

評価に用いるメモリインテンシブなアプリケーションについて説明する．

3.1　jacobi

jacobi はヤコビ法を実装したカーネルで，線形方程式を解くための古典的なアルゴリズムである．本論文では，EMAX の性能を測定するために，図 4 の（a）で示すような 3 次元 7 点のステンシル計算の例としてこれを用いる．演算は，式 1 に従って行う．

$$B[x,y,z] = C_1A[x,y,z] + C_2(A[x \pm 1,y,z] + A[x,y \pm 1,z] + A[x,y,z \pm 1]) \quad (1)$$

A は入力配列，B は出力配列であり，C_1 と C_2 は係数である．式 1 では，7 つの格子点についてロードを行い，乗算 2 回，加算 6 回の 8 回の演算が必要である．

3.2　FD6

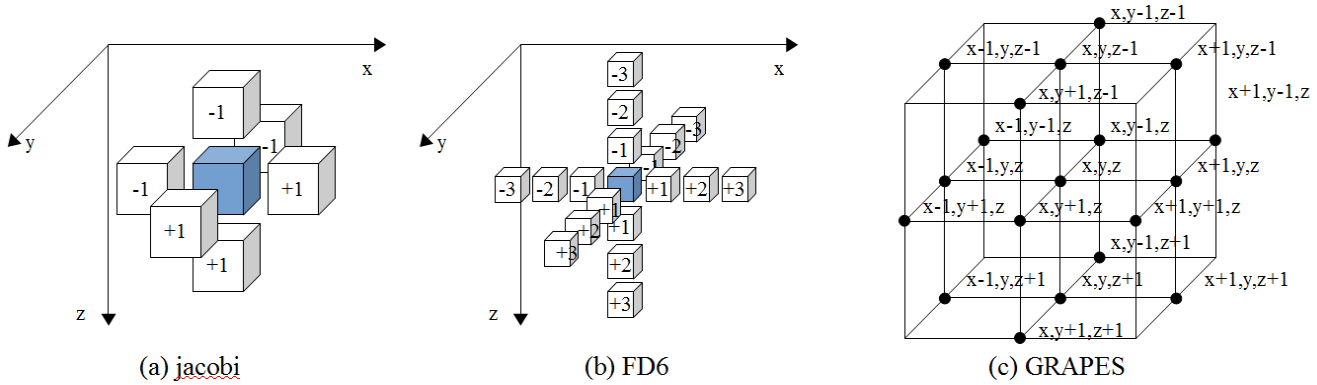
FD6 は，偏微分方程式を解く反復有限差分法で利用されるカーネルであり，次数 6 の有限差分計算を行う．これは式 2 で示される．

$$B[x,y,z] = C_1A[x,y,z] + C_2(A[x \pm 1,y,z] + A[x,y \pm 1,z] + A[x,y,z \pm 1]) + C_3(A[x \pm 2,y,z] + A[x,y \pm 2,z] + A[x,y,z \pm 2]) + C_4(A[x \pm 3,y,z] + A[x,y \pm 3,z] + A[x,y,z \pm 3]) \quad (2)$$

図 4 の（b）で示すように，3 次元 19 点のステンシル計算であり，19 の格子点についてロードを行い，乗算 4 回，加算 18 回の演算が必要である．

3.3　GRAPES

GRAPES（Global/Regional Assimulation and Prediction System）は，大気シミュレータの一種であり，式3に従ってステンシル計算を行うプログラムである．3重ループの最内ループに着目すると，図4の（c）のようなアクセスパター



(a) jacobi　　(b) FD6　　(c) GRAPES

図 4　メモリインテンシブなアプリケーション例

$$C[x,y,z] = \sum_{\delta_1=-1,\delta_2=-1,\delta_3=-1}^{\delta_1=1,\delta_2=1,\delta_3=1} A[i,x+\delta_1,y+\delta_2,z+\delta_3] \times B[x+\delta_1,y+\delta_2,z+\delta_3]$$

$$\text{,where } A[x+\delta_1,y+\delta_2,z+\delta_3] = \begin{cases} 1.0, & \text{if } \delta_1=0,\delta_2=0,\delta_3=0 \\ A[i,x+\delta_1,y+\delta_2,z+\delta_3], & \text{if } \delta_1\delta_2\delta_3 = 0 \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

ンとなる．3次元配列Bは演算対象となる空間を示しており，4次元配列Aにはこれに乗じる係数が格納されている．よって，配列Bから19回，配列Aから18回のロード（A[i,x,y,z]は常に1.0であるため不要）を行い，乗算18回，加算18回の演算が必要である．

4. EMAX への命令写像

メモリインテンシブなアプリケーションの命令列を EMAX 上に写像する手法について述べる．

4.1 FD6 の分析

ここでは FD6 を例にデータの再利用率を考慮した命令写像手法について述べる．

図 5 は，前章の式 2 を C 言語化したものである．EMAX では最内ループ x を実行するため，y 方向のループ実行毎に必要なデータをホストから EMAX へ送る必要がある．よって，図 6 のように 13 本のラインとしてデータを送信する．

また，y 回目のループ実行が終了した後，y+1 回目のループ実行に移る際には，命令写像の開始位置を 1 行ずらすため，メモリアクセスパターンは図 7 のようになる．

```
1 for (z = 0; z < Depth; z++) {
2     for (y = 0; y < Height; y++) {
3         for (x = 0; x < Width; x++) {
4             B[z][y][x] =
5               C1 * A[z][y][x] +
6               C2 * (A[z][y][x-1] + A[z][y-1][x] +A[z-1][y][x] +
7                     A[z][y][x+1] + A[z][y+1][x] +A[z+1][y][x]) +
8               C3 * (A[z][y][x-2] + A[z][y-2][x] +A[z-2][y][x] +
9                     A[z][y][x+2] + A[z][y+2][x] +A[z+2][y][x]) +
10              C4 * (A[z][y][x-3] + A[z][y-3][x] +A[z-3][y][x] +
11                    A[z][y][x+3] + A[z][y+3][x] +A[z+3][y][x])}}}
```

図 5　C 言語による FD6 の記述

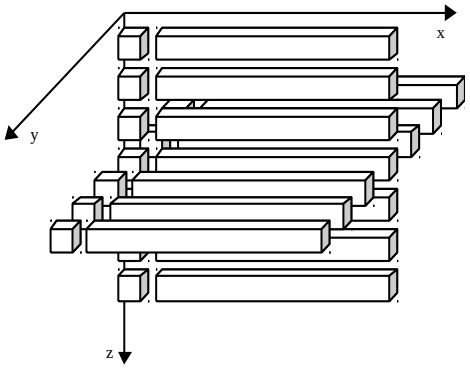


図 6　FD6 で x 方向のループ実行時に必要なデータ

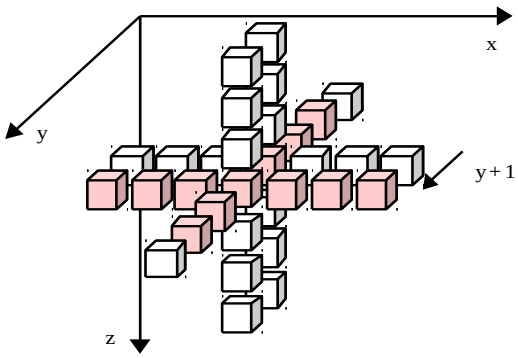


図 7　FD6 におけるメモリアクセスパターン

Row, Col	EX1/EX2 (FPU)	EAG	Local Memory
@0,0	add (ri+=,4),r10		
@0,1	add (ri+=,4),r11		
@0,2	add (ri+=,4),r12		
@1,0	add (ri+=,4),r10		
@1,1		ld (r10,0),r0	lmm_load
@1,2		ld (r11,0),r1	lmm_load
@1,3		ld (r12,0),r2	lmm_load
@2,0		ld (r10,-3840),r3	lmm_load
@2,1	fmul(ri,r0),r20		
@2,2	fmul(ri,r1),r21		
@2,3	fmul(ri,r2),r22		
@3,0		ld (r10,-2560),r4	lmm_load
@3,1	fma3(ri,r3,r20),r20		
@3,2	fadd(r21,r22),r21		
@4,0		ld (r10,-1280),r5	lmm_load
@4,1	fma3(ri,r4,r20),r20		
@5,0	add (ri+=,4),r10	ld (r10,12),r12	lmm_load
@5,1	ld (r10,4),r10	ld (r10,8),r11	
@5,2	ld (r10,-4),r8	ld (r10,0),r9	
@5,3	ld (r10,-12),r6	ld (r10,-8),r7	
@6,0	fmul(ri,r5),r22	ld (r10,1280),r13	lmm_load
@6,1	fma3(ri,r10,r20),r20		
@6,2	fma3(ri,r8,r21),r21		
@6,3	fmul(ri,r6),r23		
@7,0	fma3(ri,r12,r22),r22	ld (r10,2560),r14	lmm_load
@7,1	fma3(ri,r11,r20),r20		
@7,2	fma3(ri,r9,r21),r21		
@7,3	fma3(ri,r7,r23),r23		
@8,0	add (ri+=,4),r10	ld (r10,3840),r15	lmm_load
@8,1	fadd(r20,r22),r20		
@8,2	fma3(ri,r13,r21),r21		
@8,3	fma3(ri,r14,r23),r23		
@9,0	add (ri+=,4),r11		
@9,1	fadd(r20,r23),r20	ld (r10,0),r16	lmm_load
@9,2	fma3(ri,r15,r21),r21		
@9,3	add (ri+=,4),r12		
@10,1	fma3(ri,r16,r20),r20	ld (r11,0),r17	lmm_load
@10,3		ld (r12,0),r18	lmm_load
@11,1	fma3(ri,r17,r20),r20		
@11,2	fma3(ri,r18,r21),r21		
@12,1	fadd(r20,r21)	st -,(ri+=,4)	lmm_store

図 8　EMAX への再利用率を考慮した FD6 の記述

z 軸平面が共通している 7 本のラインのうち 6 本は，前回のループ実行時にロードしているので再利用可能であり，新規にホストからデータをロードする必要があるのは 7 本である．これにより，再利用可能なデータ量の理論値は約 46%となる．

4.2 データ再利用率を考慮した命令写像

FD6 の構造を分析した結果を基に，図 8 のように EMAX へ命令列を写像した．

図 7 より z 軸平面に再利用可能なデータがない z=±3, z=±2，z=±1 平面に存在する 6 本のラインについては，ループ実行毎に必ずデータを再ロードしなければならない．このため，データの再利用率を考慮する意味がないので，可能な限り必要な行数を削減できる書き方をするのが望ましい．よって，EMAX の 2 行目と，9-10 行目に再利用不可能なロード命令をまとめて写像する．

EMAX では y 回目のループ実行の終了毎に命令写像の開始位置を任意の行数分シフトすることができる. このため，

Row, Col	EX1/EX2 (FPU)	EAG	Local Memory
@0,0	add (ri+=,4),r10		
@0,1	add (ri+=,4),r11		
@0,2	add (ri+=,4),r12		
@1,0	add (ri+=,4),r10		
@1,1		ld (r10,0),r0	lmm_load
@1,2		ld (r11,0),r1	lmm_load
@1,3		ld (r12,0),r2	lmm_load
@2,0	add (ri+=,4),r10		
@2,1	fmul(ri,r0),r20	ld (r10,-3840),r3	lmm_load
@2,2	fmul(ri,r1),r21	ld (r10,-2560),r4	lmm_load
@2,3	fmul(ri,r2),r22	ld (r10,-1280),r5	lmm_load
@3,1	fma3(ri,r3,r20),r20	ld (r10,-4),r8	lmm_load
@3,2	fma3(ri,r4,r21),r21	ld (r10,-8),r7	
@3,3	fma3(ri,r5,r22),r22	ld (r10,-12),r6	
@4,1	fma3(ri,r8,r20),r20	ld (r10,8),r11	lmm_load
@4,2	fma3l(ri,r7,r21),r21	ld (r10,4),r10	
@4,3	fma3(ri,r6,r22),r22	ld (r10,0),r9	
@5,0	add (ri+=,4),r10		
@5,1	fma3(ri,r11,r20),r20		
@5,2	fma3(ri,r10,r21),r21		
@5,3	fma3(ri,r9,r22),r22	ld (r10,12),r12	lmm_load
@6,0	add (ri+=,4),r10		
@6,1	add (ri+=,4),r11	ld (r10,1280),r13	lmm_load
@6,2	add (ri+=,4),r12	ld (r10,2560),r14	lmm_load
@6,3	fma3(ri,r12,r22),r22	ld (r10,3840),r15	lmm_load
@7,1	fma3(ri,r13,r20),r20	ld (r10,0),r16	lmm_load
@7,2	fma3(ri,r14,r21),r21	ld (r11,0),r17	lmm_load
@7,3	fma3(ri,r15,r22),r22	ld (r12,0),r18	lmm_load
@8,1	fma3(ri,r16,r20),r20		
@8,2	fma3(ri,r17,r21),r21		
@8,3	fma3(ri,r18,r22),r22		
@9,1	fadd(r20,r21),r20		
@10,1	fadd(r20,r22)	st -,(ri+=,4)	lmm_store

図 9　EMAX への再利用率を考慮しない FD6 の記述

各ラインのデータを EMAX 上にある各行のローカルメモリにロードしておけば，y+1 回目のループ実行時には z=0 平面の 7 本のラインのうち，6 本のデータは再利用可能な状態となる．よって，EMAX における 3-9 行目の 1 列目にあるローカルメモリにデータをロードするための命令を写像する．

一方で，再利用不可能な部分のロード命令は同じ行にまとめて写像すると定義したにも関わらず，z+1，z+2，z+3 平面のロード命令が 9-10 行目に分割されているなど理想的な形で命令写像ができていない．これは，図 2 のように基本ユニット間を結ぶバスの本数に制約があるため，横方向へのデータ移動回数が限られてしまうためである．このバス制約を回避するために，ある列でロードしたデータははすぐ下の行の同じ列にあるユニットで演算するなど，可能な限り横方向へのデータ移動を防ぐように配置した結果が図 8 である．

このように命令を写像した結果，再利用可能なデータ量は理論値通り約 46%となり, 13 行に収まっていることが分かる．

4.3 データの再利用率を考慮しない命令写像

データの再利用率を考慮して EMAX に命令を写像すると，データのロードにかかる時間を削減することができ，性能向上が見込める．しかし，y 方向ループ実行毎に命令写像の開始位置をシフトされることを考慮してデータのロードに用いるローカルメモリの位置を行数毎に整列する必要がある．このため，データの再利用率を考慮してアプリケーションを写像した場合は，考慮しなかった場合と比較して行数が多くなり，必要なハードウェア量が多くなることが予測される．よって，データの再利用率を一切考慮せずに，FD6 の命令列を EMAX へ写像した場合を考える．

演算に必要なデータをどのローカルメモリにロードしなければならないかを考慮しなくて良いため，可能な限りデータ行数を削減するように命令を写像できる．しかし，前節で述べたようにバス制約が存在するため，すべての演算器とローカルメモリに命令とデータを写像することはできない．

バス制約を考慮しつつデータの再利用率を考慮しない FD6 を EMAX に写像した結果が図 9 であり, 必要な行数は 11 行となる．

4.4 Jacobi の分析と写像

FD6 と同様に jacobi のメモリアクセスパターンを分析した結果を図 10 に示す. 1 回のループ実行で 5 本のラインを用いて演算する必要があり，y 方向のループ実行終了毎に新たにロードが必要なのは 3 本である．よって，ループ実行毎に 2 本のラインを再利用することができるため，データの再利用率は 40%となる．また，データ再利用率を考慮した場合に必要な行数は 8 行であり，考慮しない場合に必要な行数は 6 行である．

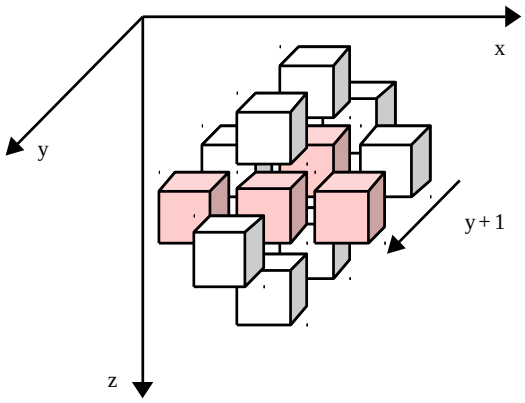


図 10　jacobi におけるメモリアクセスパターン

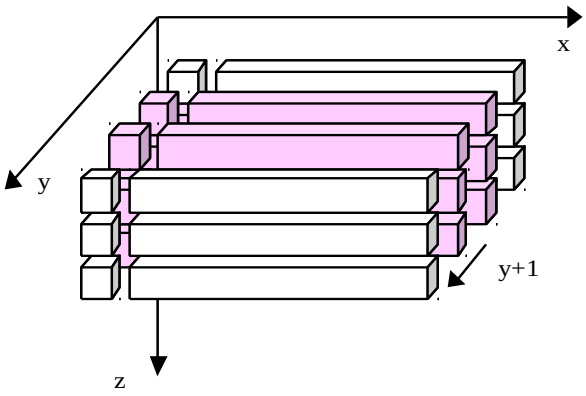


図 11　GRAPES　配列 B のメモリアクセスパターン

4.5　GRAPES の分析と写像

GRAPES における 4 次元配列 A は，ロードする 18 個の係数が最内ループの x に関して連続でなく，ランダムアクセスとなる．これにより，前回のループ実行時にロードしたデータを再利用できる余地がないため，3 次元配列 B の再利用についてのみ考慮する．

3 次元配列 B のメモリアクセスパターンを分析した結果を図 11 に示す. 1 回のループ実行で 4 次元配列 A は 18 本，3 次元配列 B は 9 本のラインを用いて演算する必要があり，y 方向のループ実行終了毎に新たにロードが必要なのは 18+9 本である．よって，ループ実行毎に 6 本のラインを再利用することができるため，データの再利用率は約 22%となる．また，データ再利用率を考慮した場合に必要な行数は 15 行であり，考慮しない場合に必要な行数は 13 行である．

5.　評価と考察

これまでの議論に基づいて，EMAX のクロックレベルシミュレータでメモリインテンシブなアプリケーションの評価を行った．使用したアプリケーションは jacobi，FD6，GRAPES の 3 つである．

前章で述べた要領で各アプリケーションを，データの再利用率を考慮した場合としなかった場合の件について

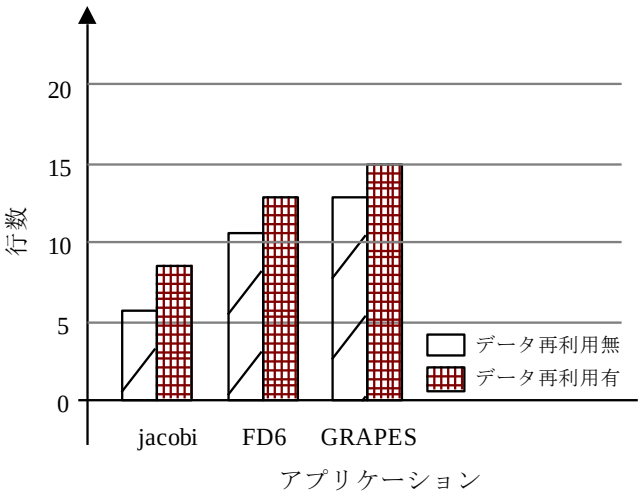


図 12　再利用率を考慮した場合に必要な行数比較

EMAX 命令列で記述した結果，必要な行数を比較したグラフを図 12 に示す. データの再利用率を無視して命令写像を行い行数の削減を試みても，それぞれ 2 段しか削減できていないことがわかる．これは，バス制約による命令写像位置の制限であり，データの再利用率を犠牲にして削減できる回路規模は最大でも 25%程度である．必要な行数やロード命令が多いものほど削減できるハードウェア量は減少することが予想される．

6.　おわりに

本稿では, ホストと EMAX 間のデータ転送量を削減するために, 演算データを再利用する手法について述べ, EMAX を用いて演算をする際にはデータの再利用率を考慮して命令列を写像することが最適であることを示した．今後は，EMAX の詳細設計を進め，基本ユニット間のバス本数の最適な構成の探索や，CPU との性能比較を行う予定である．

謝辞　本研究の一部は科学研究費補助金（基盤(A)24240005，萌芽 24650020，若手（B）23700060），および，半導体理工学研究センター（超低電圧で稼働できる耐エラープロセッサの製造性を向上させる手法）による．本研究は東京大学大規模集積システム設計教育研究センターを通し，シノプシス株式会社および日本ケイデンス社の協力で行われたものである．

参考文献

1) 王昊，姚駿，中島康彦: "GCC の vectorizer を利用した演算器 アレイ向け命令変換手法", 研究報告計算機アーキテクチャ (ARC), 2013-ARC-203 No.9, Feb. (2013)
2) 関賀，姚駿，中島康彦: "リング接続を利用しデータ移動を最小限にするアクセラレータの提案", 研究報告システム LSI 設計技術 (SLDM)SIG Technical Reports, 2013-SLDM-159, Vol.17, pp.1-6, Jan. (2013)
3) 藤原知広，姚駿，原祐子，中島康彦: "リング型アレイアクセラレータのマクロパイプライン化による性能見積もり", 研究報告計算機アーキテクチャ（ARC）, 2013-ARC-206, No.14, pp1-6, Jul. (2013)