Vol.2012-ARC-199 No.15
2012/3/28

多様なアクセスパターンに適応するアクセラレータ向けメモリアクセス機構

王　昊[†1]　姚　駿[†1]　中島康彦[†1]

我々は，画像処理や科学技術計算に向けの線形アレイ型アクセラレータ LAPP（Linear Array Pipeline Processor）を提案している．LAPP は最内ループの命令列をマップ機構により演算器アレイに写像する手法を用いて，プログラムを高速実行できる．しかし，LAPP は様々なプログラムにおける多様なメモリアクセスパターンに柔軟に対応できないという問題ある．また，演算器アレイの規模に比例して，メモリデータを演算器に供給するためのデータパスが増加してしまう．本稿では，多様なメモリアクセスを実現する EMAX（Energy-aware Multi-mode Accelerator eXtension）を提案する．EMAX は，プログラムのメモリアクセスパターンを考慮した上で，分散配置した中容量のキャッシュメモリを分割し，シフトレジスタと組み合わせることで，物理的にはシングルポートメモリでありながら，論理的には多ポートメモリを実現する．本提案メモリアクセス機構により，メモリデータを演算器に供給するためのデータパスを大幅に削減するだけでなく，同一プログラムであっても，LAPP と比較して，最内ループの命令列を削減し，演算器アレイの段数を削減できるため，更に高性能と低電力な新型演算器アレイアクセラレータを実現できると考えられる．

1. はじめに

近年，プロセッサにおける動作周波数と回路面積の増加に伴い，消費電力の問題がプロセッサ発展の壁になる．問題の解としては，我々は高性能かつ低電力を実現する線形アレイ型アクセラレータ LAPP を提案している．LAPP は高い命令レベル並列性を利用する VLIW プロセッサの演算器部分を線形に拡張し，複数段を連結する構造である．画像処理などの命令レベル並列性が高いプログラムにおける最内ループの命令列をマップ機構により，1 対 1 で演算器アレイに写像して並列実行することで，高性能を実現する[1)]．しかし，LAPP は様々なプログラムにおける多様なメモリアクセスパターンに，従来 VLIW プロセッサと同様のメモリアクセス機構を複数段用意することで処理するため，多様なメモリアクセスパターンに対して柔軟に対応できない．その結果，後続段へのデータ供給に多くのデータパスが必要となり，大規模演算器アレイの実現を困難としている[2)]．問題を解決するに，本稿では，多様なメモリアクセスを実現する EMAX を提案する．EMAX は，プログラムのメモリアクセスパターンを考慮した上で，分散配置した中容量のキャッシュメモリを分割し，シフトレジスタと組み合わせることで，メモリデータを演算器に供給するためのデータパスを大幅に削減する．また，同一プログラムであっても，LAPP と比較して，最内ループの命令列を削減し，演算器アレイの段数を削減できる．以降，2 章では LAPP におけるメモリアクセス機構及び問題点について詳述する．3 章では，2 章で述べる問題を解決する手法として新型メモリアクセス機構を提案する．4 章では評価結果を述べる．5 章を本稿のむすびとする．

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

2. LAPP

本章では，LAPP のメモリアクセス構造及び問題点について詳述する．**図 1** に LAPP の

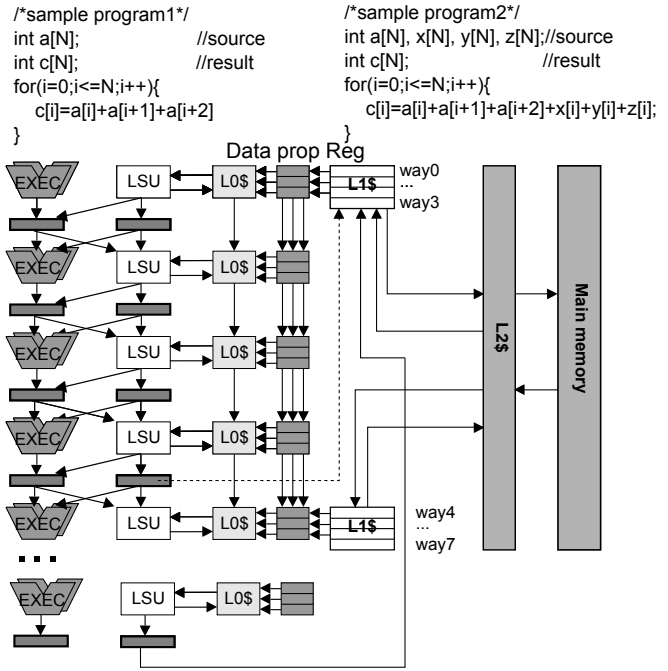

図 1 Execution and memory structure of LAPP

メモリアクセス機構を示す．LAPP のメモリアクセス機構は演算器（EXEC），1 次データキャッシュ（L1$），ローカルバッファ（L0$），ロード・ストアユニット（LSU）及び，1 次データキャッシュからローカルバッファへデータを伝搬するためのデータ伝搬レジスタ（Data Prop Reg）から構成される．そして，1 次データキャッシュを 4 ウェイに分割して使用する．ウェイ 1，2，3 の 3 つのウェイがデータの読み出し専用ウェイであり，ウェイ 0 は読み出しと書き込みの両方に使うウェイである．ここで，1 ウェイが 1 つの配列に対応する．各段においてロード命令を実行するため，1 次データキャッシュの代わりに，各段にローカルバッファを配置する．高速実行中は，毎サイクル各ウェイから 1 ワードのデータを読み出し，同時に 3 ワードのデータを次段のデータ伝搬レジスタへ転送しながら，ローカルバッファに取り込むことにより，あらかじめ決められたメモリアドレス範囲のデータをランダムに参照することができる．また，任意段においてストア命令を実行するために，各段にそれぞれ 1 ワードのストアバッファを設置する．ストアデータを順に次段へ伝搬させ，最終段のストアバッファから 1 次データキャッシュに演算結果を書き出す．演算器段数を超える長い命令列からなるループカーネルに対しては，マップできるように演算器段数を増設することになる．演算器段数の増加に伴い，ローカルバッファの容量を増加させる必要がある．先行研究では，ローカルバッファの容量の増加が，クリティカルパスに影響することを報告している[3]．その結果，多様なメモリアクセスパターンに対して，柔軟に対応できなくなり，後続段へ伝搬すべきデータが多く，必要な演算器段数が増加する，その結果，回路面積や消費電力に影響を与えることになってしまう．例として，図 1 にサンプルプログラムを示す．まず，プログラム 1 は，配列 a のエレメントの加算した結果を配列 c にストアする．ここで，ウェイ 1 が配列 a に対応し，ウェイ 0 が配列 c に対応する．演算した結果を後続段へ伝搬させ，4 段目でウェイ 0 に書き戻す．それに対して，サンプルプログラム 2 は，配列 a，x，y，z のエレメントを加算して，結果を配列 c にストアする．ウェイ 0 は配列 c に対応し，ウェイ 1，2，3，4 はそれぞれ配列 a，x，y，z に対応する．演算した結果を後続段へ伝搬して，8 段目でウェイ 0 に書き戻す．2 つのサンプルプログラムの実行を比較すると，ロード命令の増加に伴い，必要な演算器段数を増加し，後続段へのデータの伝搬のために消費電力も増える．

3. EMAX

本章では，多様なメモリアクセスパターンを考慮して，LAPP の段間配線数及び命令マップに必要な演算器段数を削減するために，新しいメモリアクセス機構である EMAX を提案する．

3.1 EMAX の仕組み

前述の通り，任意段でメモリを参照できるために，LAPP ではローカルバッファを各段に設置して毎サイクル次段へデータを転送する．そのデータの伝搬に使うデータパスを削減するために，EMAX では 4 段毎に 1 次データキャッシュを設け，1 次データキャッシュのある段のみでロード命令の実行を可能にする．**図 2** に EMAX のメモリシステムを示す．1 次データキャッシュに単一のウェイを搭載し，4 つのブロック（blk0，blk1，blk2，blk3）に分割して使用する．1 つのベースアドレス（BASE-AD）と 6 つのオフセットを組み合わせることにより，6 箇所のアドレスを同時に参照できる．1 次データキャッシュからロードしたデータをセレクタ（SEL）によって選択し，シフトレジスタの先頭に書き込む．次のサイクルから，データが順にシフトレジスタに流れる．そして，アドレス計算ユニット（EAG）と出力レジスタが 1 対 1 に対応し，EAG からのアドレス情報を比較する機構によって，シフトレジスタから出力レジスタへデータを転送する．以上により，使用可能なアドレス範囲に制約を設けた上で，物理的に読み出しに 1 ポート，書き込みに 1 ポートを備える一般的なメモリを用い，6 リード，2 ライトのメモリ機能を実現できる．各段にローカルバッファがなく，1 次データキャッシュから読み出したデータを後続段に伝搬させるデータパスは不要となるため，段間配線数を大幅に削減できる．そして，中容量バッファの配置により，様々なプログラムに対して，柔軟に対応できるという特性を持ち，必要な演算器段数が LAPP よりも少なくなる．

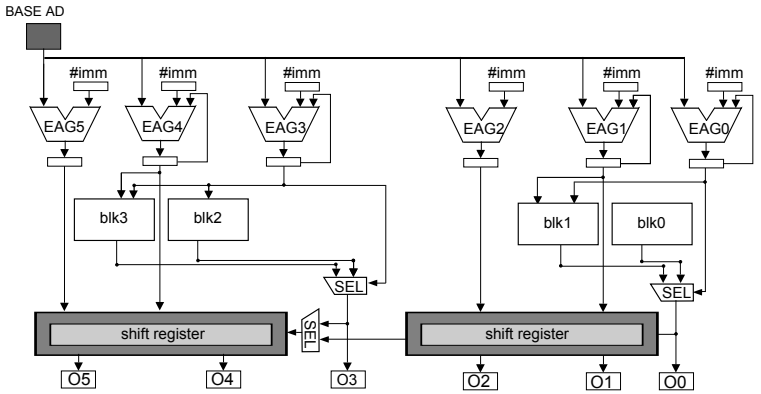

図 2 Memory structure of EMAX

3.2 メモリアクセス

本節では，サンプルプログラムを用いて 5 つのメモリ参照パターンについて EMAX での動作を説明する．

パターン 1：$A+x$．

固定のベースアドレスと広範囲のオフセットで示したメモリ領域に対してランダムに参照するパターンである．図 2 において EAG0 からのアドレスを way0.blk0 及び way0.blk1 に供給し，各々から読み出した値をセレクタでアドレスの上位ビットにより選択して出力する．

パターン 2：$A-3$，$A-2$，$A-1$，A．

図 3 のサンプルプログラムは，単調増加するベースアドレス a[i] に，前後 4 箇所の範囲制限をつけて，固定オフセット（+0，+1，+2，+3）で，同時に 4 箇所を参照するパターンである．EAG0 から提供したアドレスにより，way0.blk0 と way0.blk1 から読み出したデータをシフトレジスタに流しこむ．ベースアドレスの単調増加と共に，4 箇所のアドレスも単調増加するので，アドレス情報の比較が不要になり，シフトレジスタの内容を直接出力レジスタへ転送する．そして，次のサイクルから，データをシフトレジスタに流しながら，出力レジスタの内容が更新され．図 3 の例のように，クロックサイクル（CC）$n-1$ に，4 つのエレメント a[0]，a[1]，a[2]，a[3] がシフトレジスタから，出力レジスタへ転送される．次の CCn には，シフトレジスタの内容の更新につれて，a[1]，a[2]，a[3]，a[4] を出力レジスタに転送する．

パターン 3：$A-2$，$A-1$，A，$B-2$，$B-1$，B．

図 4 のサンプルプログラムは，1 つのウェイに 2 つの配列をプリフェッチして，パターン 2 と同様に，前後 3 箇所に制限をつけて，同時に 2 つの配列を参照するパターンである．blk0 と blk1 は配列 a，blk2 と blk3 は配列 b に対応する．EAG0 と EAG3 からアドレスを

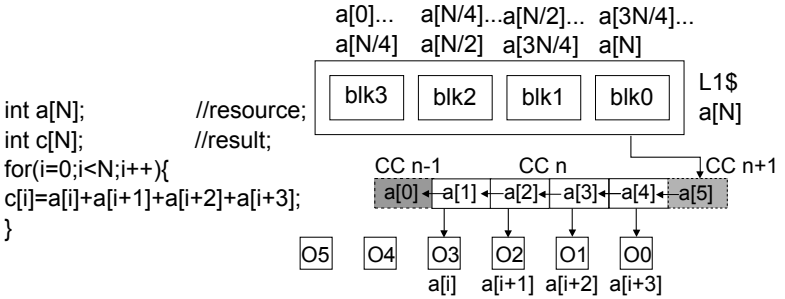

図 3 Example of Pattern 2 and its accessing.

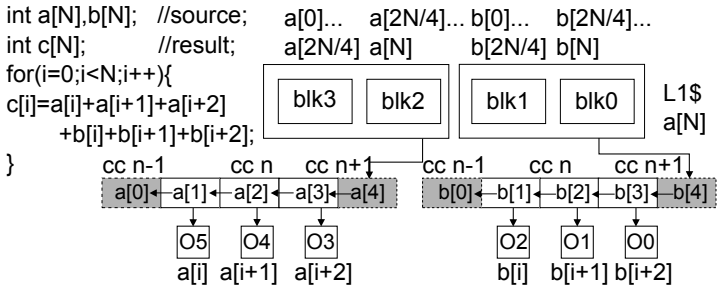

図 4 Example of Pattern 3 and its accessing.

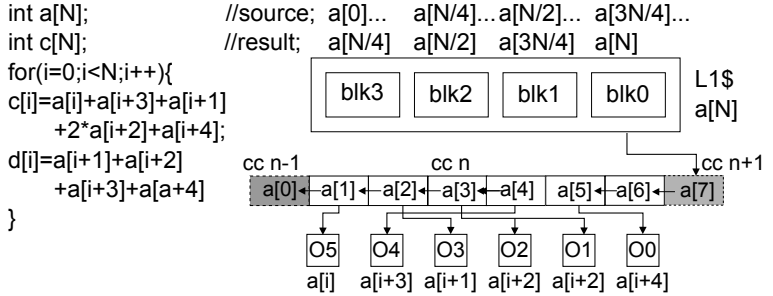

図 5 Example of Pattern 4 and its accessing.

供給し，各々way0 の blk0，blk1 と way0 の blk2，blk3 に参照する．パターン 2 のように，読み出したデータをシフトレジスタに流しこむと共に，出力レジスタの内容を更新する．

パターン 4：$A-a$，$A-b$，$A-c$，$A-d$，$A-e$，A．

図 5 のサンプルプログラムは，単調増加するベースアドレス A に，近傍 5 箇所の範囲制限をつけて，同時に 6 箇所を参照するパターンである．way0 が配列 a に対応する．CCn に，ベースアドレスを way0 の blk0 と blk1 に供給して，シフトレジスタの先頭に書き込む．各 EAG がアドレス計算した結果をアドレス情報比較機構によって，シフトレジスタの任意の位置と比較して，一致した部分のレジスタの内容を，対応する出力レジスタに書き込み．次の CC から，シフトレジスタにデータを流しながら，出力レジスタの内容も更新する．更に，d[i] の計算に，データを移動する代わりに，命令マップ位置を変化させることによって，データ移動に伴う電力や時間を削減することができる．

パターン 5：$A-2$，$A-1$，A，B，C，D．

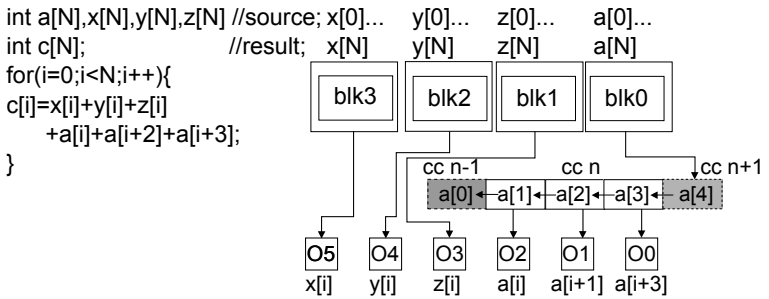

図 6 Example of Pattern 5 and its accessing.

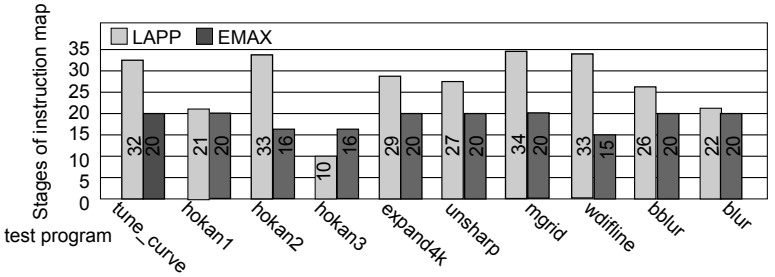

図 7 Instruction number of loop kernel in LAPP and EMAX

図 6 のサンプルプログラムでは，4 つの配列を処理する．まず，配列 a に対して，ベースアドレスが単調増加し，前後含む 3 箇所のアドレス範囲制限をつけ，3 箇所を同時にランダム参照するパターンである．単調増加するベースアドレスを way0.blk0 のみ供給し，シフトレジスタの先頭に流し込む，EAG2 と EAG5 から供給されたアドレス計算の結果がアドレス情報比較機構によって，シフトレジスタの任意の位置を比較する，一致した部分のレジスタの内容を対応する出力レジスタに書き込む．配列 x，y，z は，ベースアドレスが単調増加する配列であり，way0 の blk1，blk2，blk3 に 1 対 1 対応する，EAG1，EAG3，EAG4 からアドレスを供給し，blk1，blk2，blk3 を参照して，読み出したデータを直接出力レジスタに転送する．

4. 評価結果

本章では，LAPP と EMAX において，同一処理のベンチマークプログラムにおけるループカーネル内の命令数を比較する．LAPP と EMAX ともにループカーネル内の VLIW 命令数が必要な段数と等しくなる．結果を**図 7** に示す．今回使用したベンチマーク 10 個のうち 9 個において，EMAX が LAPP よりも少ない命令数となることがわかった．特に wdifline では，命令数を 50%に削減できており，段数を大幅に削減できることがわかった．今後は，他のベンチマークプログラムについても命令数の比較を行い，さらなる評価を行う予定である．

5. むすび

本研究は，複数中容量バッファを分散配置させ，中容量メモリと小容量シフトレジスタの組み合わせることにより，一定範囲のメモリ空間に対する多数のロード命令実行が可能となるメモリアクセス機構（EMAX）を提案した．先行研究である LAPP と比較して，後続段への伝搬データパスが不要となることに加え，同一処理においてループカーネル内の命令数を削減でき，演算器段数と段間配線数の削減できる．その結果，電力効率と演算性能を向上させることが期待できる．

6. 謝辞

なお，本研究の一部は先端的低炭素化技術開発（次世代低電力デバイス安定化計算機構成方式），科学研究費補助金（若手研究（B）課題番号 23700060）及び JST-ASTEP（FS 課題番号 AS232Z02313A）による．

参考文献

1) Kazuhiro Yoshimura, Takuya Iwakami, Takashi Nakada, Jun Yao, Hajime Shimada, Yasuhiko Nakashima: “An Instruction Mapping Scheme for FU Array Accelerator”, IEICE Transactions on Information and Systems, Vol. E94-D, No. 2, pp. 286–297, 2011.
2) 森高晃大，下岡俊介，吉村和浩，姚　駿，中田　尚，中島康彦: “大規模演算器アクセラレータのための複数 FPGA 連結手法”, デザインガイア 2011, pp. 9–14, 2011.
3) 下岡俊介, 吉村和浩, 中田　尚, 中島康彦: “演算器アレイ型アクセラレータにおけるローカルバッファの最適化”, 2011 年並列/分散/協調処理に関する『鹿児島』サマー・ワークショップ (SWoPP 鹿児島 2011), 情報処理学会研究報告 2011-196(18), pp. 1–6, 2011.