Vol.2012-ARC-199 No.11
2012/3/28

演算器アレイ割り当て型スーパスカラ実行の効率向上検討

嶋　田　　創[†1]

近年，電力効率向上を目的とした演算器アレイ構成が広く注目されている．演算器アレイ構成は制御構造が単純なループ内命令列の実行に高い電力効率を示すが，プログラム中の制御構造が複雑な部分の命令列の実行は苦手とする．本論文では，制御構造が複雑な部分の命令列の実行を効率化するため，演算器アレイのリソースを効果的に利用したアウトオブオーダ実行スーパスカラの実現に関して検討結果について記す．

A Study of Improvement Efficiency on ALU Array Allocatable Super Scalar Processor

Hajime Shimada[†1]

Recently, to improve power performance efficiency, an ALU array architecture is widely spotlighted. The ALU array arechitecture gives high power performance efficiency for instructions in simple control flow loops but it does not gives good power performance efficiency for complicated control flow part instructions. In this paper, I notate a study of achieving out-of-order superscalar execution with effectively utilizing ALU array architecture resources, which is aimed to improve efficiency in complicated control flow part instructions.

1. はじめに

プログラムの実行時に消費されるエネルギーを削減し，電力効率を効率させることは近年のプロセッサに求められる事の 1 つである．この電力効率を大幅に向上させるアーキテクチャとして，近年では粗粒度再構成アーキテクチャ(CGRA: Cource Grain Reconfigurable Architecture)[1),2)] を始めとする，大量の演算器をアレイ状に配置するアーキテクチャが着目されている．

これらの演算器アレイを用いるアーキテクチャでは，プログラムのホット・ループ中の各演算命令をアレイ状に配置されている演算器に割り当て，各イタレーションのデータを個々の演算器で並列に実行することで処理性能を向上させると同時に，演算結果のレジスタ/ファイルへのライトバックを削減することで演算処理あたりの消費電力を抑える．このようなアーキテクチャは，メディア処理などのプログラムの制御構造が複雑でなく，ホット・ループの開始や終了を容易に特定できるプログラムでは十分な有用性が示されている．しかしながら，これらのアーキテクチャは制御構造が複雑なプログラムの実行は苦手であり，このようなプログラムを単純にこれらのアーキテクチャ上で実行した場合，大幅に電力効率が悪化することが考えられる．そのため，既存研究には，既存のスーパスカラ・プロセッサや VLIW プロセッサと組み合わせ，制御構造が複雑なプログラムの実行をそちらで行うことによって，種々のプログラムを実行した場合においてもトータルの電力効率を向上させる試みがなされている[3)]．しかしながら，このような構成は回路面積を増大させることが多く，回路面積コスト的に好ましくない．

そこで，本論文では，制御構造が複雑なプログラムの実行を演算器アレイ上で行う形を基本とし，既存のスーパスカラ・プロセッサで利用されている命令レベルの並列性の利用方法を演算器アレイ上で実現する形で利用を行う演算器アレイ割り当て型スーパスカラ実行構成を考える．その上で，スーパスカラ実行に必要となる回路資源の追加を少なくして電力効率を少しでも改善するために，アレイ割り当て型命令ウィンドウ構成とアレイ間パイプライン用レジスタのリネーム用物理レジスタ利用について検討する．

以下，2 節では演算器アレイ割り当て型スーパスカラ実行の考え方を示し，3 節で，その効率向上のアイデアを予備評価を交えて簡単な検討を行う．4 節では関連研究について延べ，最後に，5 節でまとめと将来の展望を述べる．

2. 演算器アレイ割り当て型スーパスカラ実行

2.1 前提とする演算器アレイ

図 1 に演算器アレイ割り当て型スーパスカラ実行のベースとする，一般的な演算器アレイの構成を示す．図中の演算器アレイは 4x4 の構成を例として示しており，各演算器は 2 つのソース・レジスタ値に対して演算を行い，1 つのデスティネーション・レジスタ値を出力

†1 奈良先端科学技術大学院大学情報科学研究科
Graduate school of Information Science, NAIST

してパイプライン・レジスタに格納する形を前提としている．

命令やソース・レジスタ値や図の上方にあるレジスタ・ファイルより供給され，演算器や，演算器をバイパスするデータ・パスを介して，図の下方にある演算器にも供給される．このバイパスするデータ・パスの増加は配線の複雑さの増加につながるため，図の下方にある演算器は可能な限り図の上方の演算器の出力からソース・レジスタ値を受け取るように演算を割り当てなければならない．演算結果は，演算器からレジスタ・ファイルにつながるライトバック用のデータ・パスを介してライトバックされたり，最上段の演算器にソース・レジスタ値として供給されたりする．

メモリ・アクセスについては，図の右側に示すように，演算器のレイテンシと同レベルのレイテンシでアクセス可能なキャッシュが配置されており，メモリ・アクセス命令は演算器で実効アドレスを生成した後，キャッシュアクセスを行う[⋆1]．

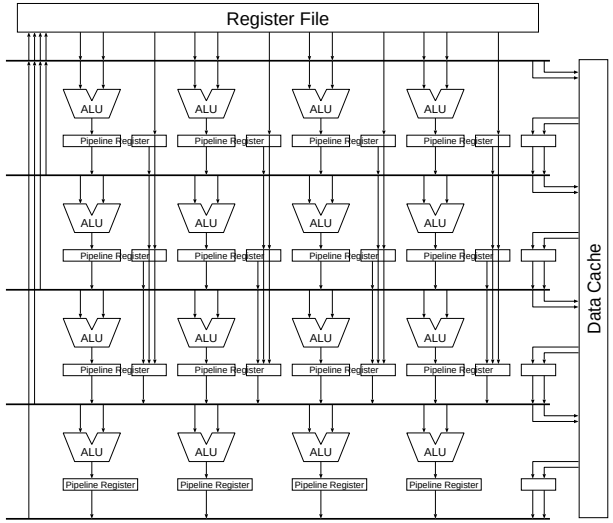

図 1 ベースとする演算器アレイの構成

⋆1 図のような多ポートのキャッシュを 1 メモリ・アレイとして実現するとレイテンシが過大となるため，マルチバンク構成にするなどの工夫が必要であるが，本論文ではその詳細は検討しない．

2.2 演算器への命令割り当てによるスーパスカラ実行

図 2 に演算器アレイ割り当て型スーパスカラ実行のベースとする演算器アレイの実行の様子を示す．例として，9 命令からなる配列の加算を行うループの擬似命令列を用いている．黒色の命令列は 1 イテレーション目であり，灰色の命令列は 2 イテレーション目を示している．

図に例示されるように，演算器アレイ割り当て型スーパスカラでは，同時にフェッチされた最大 4 命令 (フェッチ・グループ) の命令に対してフェッチ・グループ内，および，先行するフェッチ・グループの命令に対してデータ依存関係の解析を行い，アレイのデータ・パスを利用してソース・レジスタ値を受け取れる演算器に配置される．投機実行からの回復処理を容易にするため，各フェッチ・グループの命令は演算器アレイ内の任意の配置するのではなく，各フェッチ・グループごとに決められた範囲内に配置される．図の例では，各フェッチ・グループ内の命令は 2x2 の範囲に配置されるものとしてある．

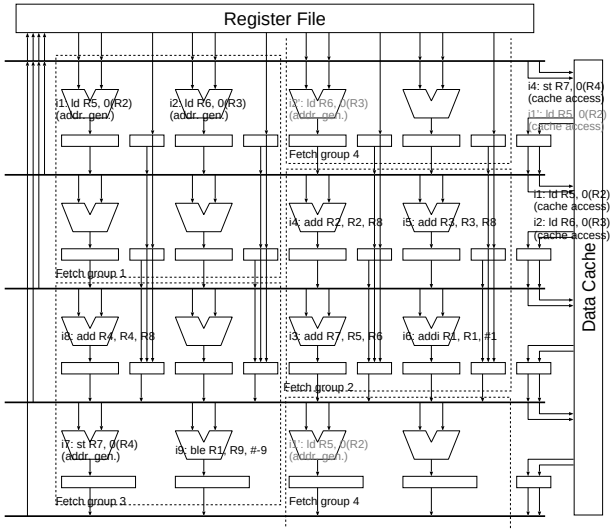

図 2 演算器アレイ割り当て型スーパスカラ実行の概要

3. 演算器アレイ割り当て型スーパスカラ実行の効率向上

一般的なアウトオブオーダ実行のスーパスカラ・プロセッサ (以下 OoO-SS) では，命令レベル並列性を引き出すために命令ウィンドウやリネーム用の物理レジスタを必要とする．本論文では，演算器アレイ内の資源を用いて上記を実現することにより，回路資源という点から，演算器アレイ割り当て型スーパスカラ実行の効率向上を試みる．

3.1 アレイ割り当て型命令ウィンドウ構成

一般的な OoO-SS では，フェッチした命令のソース・レジスタ値が揃って実行可能となるまで，命令は命令ウィンドウというバッファで待ち合わせることになる．これは，一般的な OoO-SS では演算器の数は少なく，命令が実行となるまで演算器で命令を待ち合わせるよりは，命令ウィンドウで待ち合わせても，他に実行可能となっている命令を演算器で実行させた方が性能が高くなるからである．一方，一般的な演算器アレイを用いるアーキテクチャでは演算器数が多いため，演算器において命令を待ち合わせても，他に利用可能な演算器で実行可能となっている演算を行うことは可能だと考える．

そこで，演算器アレイ割り当て型スーパスカラ実行においては，命令ウィンドウを廃止し，演算器アレイ自体を命令ウィンドウとすることを提案する．アクセラレータ用の演算器アレイにおいて，各演算器には，その演算器に実行させる演算の種類や演算にソース・レジスタ値の供給先を指定するためのレジスタが付属しており，この部分を演算や命令スケジューリングに必要な命令情報の保存先として利用する．この演算器アレイを実行待ちの命令のバッファとして利用する様子は，すでに図 2 に示した形になる．

命令ウィンドウのもう 1 つの機能である，ソース・レジスタ値が利用可能の可否を判別して命令自体が実行可能かどうかを判別する機能は，演算器アレイ・アクセラレータの実現においても必要な，演算器間のデータ依存関係を示す機構を利用する形で実現する．**図 3** に，提案するデータ依存表現方法を示す．図中の演算器のソース側に示されている”RF(R2)“はソース・レジスタ値はレジスタ・ファイル中の R2 から得られることを示し，“Array(4,2)“はソース・レジスタ値は x=4, y=2 の演算器の出力として得られることを示す．データ依存関係上で後方の命令は，演算器に割り当てられた後，データ依存関係上で先方の命令の実行完了フラグを確認し，

なお，次節で述べるアレイ間パイプライン・レジスタのリネーム用の物理レジスタ利用とも関連するが，先方の命令の実行結果は，新たにその演算器に割り当てられた命令の実行が完了し，なおかつ，後方の命令の実行が完了するまでレジスタ・ファイルにライトバックされない．そのため，後方の命令が演算器アレイ内で実行待ちをしている間に，ソース・レジスタ値が演算器アレイ中から失われるということは起こりえない．

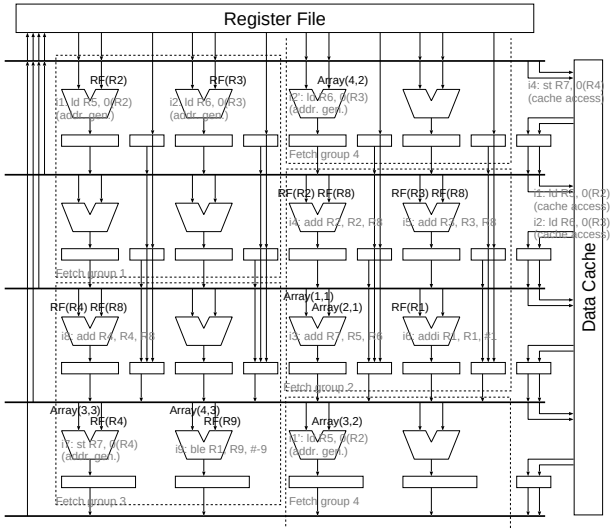

図 3 演算器アレイ割り当て型命令ウィンドウのデータ依存表現

3.2 アレイ間パイプライン・レジスタのリネーム用物理レジスタ利用

一般的な OoO-SS では，出力依存や逆依存という偽の依存を解消するため，レジスタ・リネーミングという手段を用いる．このレジスタ・リネーミングを行うためには追加の物理レジスタが必要となり，一般的な OoO-SS では論理レジスタ用の物理レジスタ・ファイルとは別にリネーム用の物理レジスタ・ファイルを準備したり，論理レジスタとリネーム用のレジスタを包含可能な大容量物理レジスタ・ファイルを準備する形を取る．一般的に，レジスタ・ファイルは完全な多ポート構成を取るため，回路面積コストは大きい．

そこで，演算器アレイ割り当て型スーパスカラ実行においては，演算器の出力側のパイプライン・レジスタをリネーム用物理レジスタとすることで，リネーム用物理レジスタにかかる回路面積コストを削減する．具体的には，図 3 に示すように，データ依存関係上で後方の命令が，演算器アレイ中に存在するデータ依存関係上で先方の命令の結果を利用する場合，ソース・レジスタ値の読み出し先としてリネーム用物理レジスタを記すのではなく，

演算器の出力側パイプライン・レジスタを示す形を取らせる．これにより，偽の依存を気にすることなく，演算器アレイに後続の命令を割り当てることが可能となる．このレジスタ・リネーミングによる偽の依存の解消の例として，図 3 中の命令 i7, 命令 i8 を挙げる．図中で R4 を更新する命令 i8 は，R4 の値を使う命令 i7 よりも上方の演算器に割り当てられているが，演算器上において，ソース・レジスタ値の読み出し先をレジスタ・ファイルとすることで，逆依存を解消している．

この構成では，命令の実行完了後にその結果をできるだけ長期間に渡ってパイプライン・レジスタに割り当てておくことにより，後方の命令がソース・レジスタ値をレジスタ・ファイルを介さずに読み出す可能性が高くなる．そのため，ある命令の実行結果は，その演算器に次の命令が割り当てられるまでレジスタ・ファイルへのライトバックを行わず，パイプライン・レジスタに残しておくものとする．これにより，レジスタ・ファイルの読み出しポート数削減による回路面積削減が可能となる[⋆1]．同時に，この構成は，新たに当該論理レジスタに結果を出力する命令が割り当てられ，レジスタ値の生存期間が終了したレジスタ値のレジスタ・ファイルへのライトバックを削減することも可能となる．これにより，ライトバックにかかる消費電力やレジスタ・ファイルの書き込みポート数削減による回路面積削減が可能となる．**図 4** に予備評価としてレジスタ値の生存期間を示す．図の横軸は「何命令後までレジスタ値が生きているか」を命令数で示したものであり，縦軸はその命令数までにレジスタ値の生存期間が終了する命令の累積値である．図より，平均で 4x4 の演算器アレイを用いた場合は約 23%，6x6 の演算器アレイを用いた場合は 29%，8x8 の演算器アレイを用いた場合は 31%のレジスタ・ファイルへのライトバックを削減できることが見込めることが予想される．

4. ま と め

本論文では，演算器アレイを用いるアクセラレータとその上で実行されるプログラムに対し，プログラム中の制御構造が複雑な部分の処理を高速化するために演算器アレイ割り当て型スーパスカラ構成を利用することと，その効率向上について簡単な検討を行った．

検討の結果，一般的な演算器アレイの構成を利用する場合において，アウトオブオーダ型スーパスカラを構成する要素のうち，命令ウィンドウとレジスタ・リネーミング用物理レジスタについては，演算器アレイの構成要素を利用して実現できるという見通しを得た．今後の課題として，アウトオブオーダ型スーパスカラ構成要素のうち特に回路面積コストが大きな命令スケジューリング部 (アレイへの命令割り当て部)，レジスタ・マップ表，命令のリタイア処理等をアクセラレータとしての演算器アレイの構成要素を利用して低い追加コストで実現する方法の検討が挙げられる．また，演算器アレイ間のデータ・パスの複雑さと命令割り当てはトレード・オフの関係にあると考えられるため，コスト対性能比に優れた設計点を模索する必要もある．

⋆1 後方の命令によるレジスタ値の読み出しが完了するまで，その命令を演算器アレイから削除できないため，新たな命令の演算器への割り当てが遅れて性能が低下する恐れがあるが，レジスタ・ファイルへのライトバックを遅らせるのが妥当かという評価は今後の課題とする．

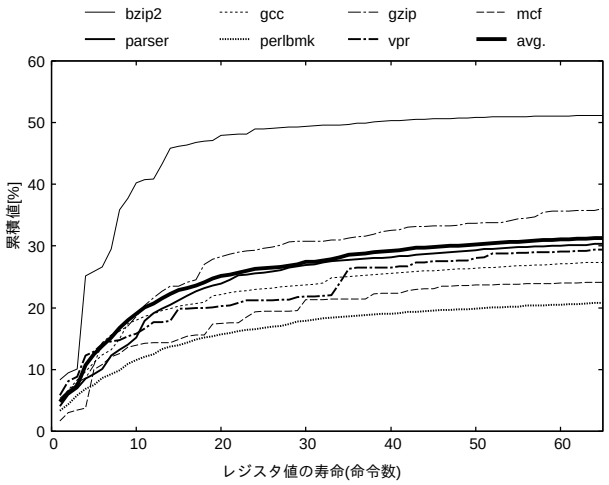

図 4 レジスタ値の生存期間

参 考 文 献

1) B. Mei, S. Vernalde, D. Verkest, H. D. Man, and R. Lauwereins, “Exploiting Loop-Level Parallelism on Coarse-Grained Reconfigurable Architectures Using Modulo Scheduling,” DATE-2003, pp. 297-301, 2003.
2) B. Bougard, B. B. D. Sutter, D. Verkest, L. V. Perre, and R. Lauwereins, “Coarse-Grained Array Accelerator for Software-Defined Radio Baseband Processing,” IEEE Micro, Vol. 28, No. 4, pp. 41-50, 2008.
3) 中田尚, 上利宗久, 中島康彦, “画像処理向け線形アレイ VLIW プロセッサ,” 先進的計算基盤システムシンポジウム SACSIS2009, pp. 293-300, 2009.