特別研究報告書

# ARM命令セットにおける
# パイプラインステージ統合の有効性の調査

指導教員　　富田 眞治 教授

京都大学工学部情報学科

岩田 浩明

平成18年2月10日

# ARM 命令セットにおける パイプラインステージ統合の有効性の調査

岩田 浩明

## 内容梗概

近年のモバイル・プロセッサにおいては，低消費エネルギーと高性能の両立が求められている．これらの要求に応えるため，現在では，Dynamic Voltage Scaling(DVS) と呼ばれる方式が導入されている．プロセッサの負荷が低い時にクロック周波数と電源電圧を低下させ，消費エネルギーを削減する DVS は現在では有効な方式であるが，将来の導体製造技術においては，電源電圧の可変範囲が縮小し，有効性が低下する．これに対して，半導体製造技術に依存しない方式として，パイプラインステージ統合（PSU: Pipeline Stage Unification）と呼ぶ方式が提案されている．PSU では，プロセッサの負荷が低い時にクロック周波数を低下させた後，電源電圧を最大に保ったまま複数のパイプライン・ステージを統合する．PSU で消費エネルギーを削減できる理由は以下の二つである．第 1 に，バイパスされるパイプライン・レジスタへのクロックの供給を止めることにより，クロック・ドライバの総負荷を減少させることができる．第 2 に，パイプライン・ステージの統合によりプロセッサのパイプラインが短くなることにより，プログラムの実行に必要なサイクル数が削減され，電力を消費する時間を短くすることができる．

今までの PSU は，SimpleScalar PISA という命令セットのみで評価されてきた．しかし，PSU ではステージ統合を行った時の IPC（Instruction Per Cycle）の向上にともなって消費エネルギーが削減されるため，命令セットやパイプラインの構成によって IPC が変化すれば，消費エネルギー削減量も変化する．そこで，本論文では，SimpleScalar PISA とは異なる命令セットにおける，PSU の有効性の検討と調査を行う．調査に用いる命令セットとしては，ARM 命令セットを用いた．これは，ARM 命令セットが多数の独特な拡張を持っており，SipmleScalar PISA のような一般的な RISC 命令セットとは IPC が大幅に異なる可能性が高い点と，現在，市場で大きな割合を占める点から選択した．

ARM 命令セットの大きな特徴として，条件実行，シフト・オペランド，ロード／ストア・マルチ命令，浮動小数点演算器の省略の 4 つがあげられる．以下，それぞれの PSU に与える影響を検討する．

- 条件実行：小規模な if-else 構文における分岐命令を削除でき，分岐による命令アドレスの変化によって命令フェッチが途切れたり，分岐予測ミスによって分岐予測ミス・ペナルティを被ることがなくなる．プログラム中の分岐が少ないということは，分岐予測ミス・ペナルティによって低下している IPC が少ないため，PSU を適用しても IPC がそれほど向上しないことになり，PSU の効果が少なくなると考えられる．
- シフト・オペランド：従来の 2 命令を 1 つにまとめるだけであり，PSU の効果に対する影響はほとんどないと考えられる．
- ロード／ストア・マルチ命令：PSU を適用すると，ロード／ストアのパイプラインが短くなり，連続するロード／ストア処理の完了が早くなる．よって，ロード／ストア・マルチ命令によるロード／ストアのパイプラインに負荷が集中する時間が短くなり，性能向上に結び付くと思われる．つまり，PSU の効果は大きくなると考えられる．
- 浮動小数点演算器の省略：soft-float ライブラリを用いることで実行命令数が増える．このため，実行時に分岐予測のための履歴表やキャッシュに保持すべき内容が増える．これによって，分岐予測成功率やキャッシュ・ヒット率が悪化するため，PSU による分岐予測ミス・ペナルティやキャッシュ・ヒット・レイテンシの削減の影響が大きくなると考えられる．つまり，PSU の効果が大きくなると考えられる．

以上より，平均すると全体では PSU の効果は大きくなると思われる．

評価では，まず，ARM 命令セットと SimpleScalar PISA の実行命令数比較を行った．実行命令数の比の幾何平均は 1.07 となり，ARM 命令セットでは，浮動小数点演算を soft-float ライブラリで整数演算に置き換えているにもかかわらず，平均ではそれほど実行命令数が変わらないことがわかった．PSU を適用した消費エネルギー削減の評価は，PSU を行う ARM 命令セットのパイプラインの仮定後にシミュレータを作成し，それを用いて行った．その結果を従来の SimpleScalar PISA の結果と比較した結果，ARM 命令セットは 2 ステージを統合した場合に SimpleScalar PISA よりも平均 14.5%，4 ステージを統合した場合に平均 17.2%消費エネルギーを削減できるという結果を得た．また，ARM 命令セットの方が消費エネルギーを削減できなかったベンチマーク・プログラムについては，当初の考察で PSU の効果が小さくなる拡張を多用していることが確認でき，考察は妥当であったと考えられる．

# Survy of Pipeline Stage Unification effective in ARM instruction set

Hiroaki IWATA

**Abstract**

Recent mobile processors are required to exhibit low-energy consumption as well as high performance. To satisfy these requirements, a method called dynamic voltage scaling or DVS is currently employed. DVS reduces energy consumption by decreasing the supply voltage when a processor runs at a low clock frequency. Although DVS is an effective method for reducing energy consumption, its effectiveness will be limited in future process generations because the variable supply voltage range will become small. As an alternative, we propose a method called pipeline stage unification or PSU, which unifies multiple pipeline stages when the processor runs at a low clock frequency, leaving the supply voltage at its maximum level. The reason why PSU can reduce consumption energy is as follows. Firstly, PSU saves power consumption by reducing the total load capacitance of the clock driver. This is accomplished by stopping the clock signal to bypassed pipeline registers. Secondly, PSU reduces the execution time of program by reducing the number of pipeline stages.

Previous PSU evaluations have only done with instruction set called SimpleScalar PISA. However, the energy consumption reduction with PSU is inverse proportion to the IPC improvement so that the energy consumption reduction will be changed if the instruction set or pipeline organization have changed. In this paper, I consider the effectiveness of PSU in the other instruction set. The instruction set which I use in this research is ARM instruction set. The reason why I chose that instruction set is as follows. Firstly, ARM instruction set have unique features compared to the typical RISC instruction sets such as SimpleScalar PISA. Secondly, ARM instruction set occupies large percent in current market.

There are four unique features in ARM instruction set such as conditional execution, shift operand, load/store multi instruction and abbreviation of floating point arithmetic unit. I have discussed influence to give each PSU.

- Conditional Execution : The compiler can eliminate branches from a small

if-else structure by using this feature. So, the processor can avoid aborting fetch by change of instruction address or suffering branch misprediction penalty. This feature reduces IPC loss from branch misprediction penalty so that it reduces the PSU effectiveness because IPC improvement from branch misprediction penalty reduction becomes small.

- shift operand : It summarizes two conventional orders in 1, and I think that there is little influence for an effect of PSU.
- Load/store multi instruction : Load/store pipeline has been shorten by applying PSU so that consecutive load/store operations will be finished quickly. According to this, load contribution of the load/store pipeline which is caused by load/store multi instruction will be alleviated. Thus, the PSU effectiveness becomes large with this feature.
- Abbreviation of floating point arithmetic unit : The executed instruction will increase by using a soft-float library. Then, the usage of pattern history table for branch prediction and caches will increase and the brach prediction accuracy and cache hit rate will become worse. Thus, the effectiveness of branch misprediction penalty reduction and cache hit latency reduction will increase and the effectiveness of PSU will increase.

On average I think that an effect of PSU grows big, than the above.

At first, I compared the number of the orders between ARM instruction set and the SimpleScalar PISA. So, the geometric mean of the ratio became 1.07. Though ARM instruction set did floating point arithmetic with a soft-float library, I understood that the number of the orders on average did not change between two instruction set. I evaluated the effectiveness of PSU in ARM instruction set by hand-made software simulator which is based on assumed pipeline organization. Then I compared that result with SimpleScalar PISA that the instruction set used in previous research. The evaluation result shows that the effectiveness of PSU for ARM instruction set is 14.5% larger than the SimpleScalar PISA in 2-stage unification and 17.2% larger in 4-stage unification.

# ARM 命令セットにおける
# パイプラインステージ統合の有効性の調査

## 目次

# 第1章　はじめに

近年のモバイルプロセッサでは，ノート型コンピュータの増加及び小型化や組み込み機器用のプロセッサの増加により，低消費エネルギーと高性能の両方が要求されている．低消費エネルギー化のメリットは明らかである．まず電池駆動の機器を考えてみる．消費エネルギーを低減すれば，駆動時間を延長できる．駆動時間を延ばす必要がない場合には，より容量の小さい電池を選択することで機器の外形寸法や重量，コストなどを低減可能となる．次に，電源コンセントから電力供給を受けるシステムを考えてみる．消費エネルギーを抑えれば，放熱用のファンやエアコンに求められる放熱性能を軽減できる．つまり，消費エネルギーが小さいとシステムの発生熱量が少なくなる．電源供給ユニットの外形寸法を小さくできるため，ファンの数を減らすことや，駆動音の小さいファンを採用することができる．結果としてシステムの駆動音が小さくなる．このほか消費エネルギーを低減することで，ホット・スポット（温度上昇が特に激しい部分）による部品実装密度の制限を解消できる可能性もある．部品実装密度が高まれば，一定の容積当たりのシステム処理性能を高められる．このように，消費エネルギーを低減することでシステムの外形寸法とコストの両方を抑えられることがわかる．

この低消費エネルギー化の要求を満たすために，**パイプラインステージ統合**（PSU: Pipeline Stage Unification）と呼ばれる手法が提案されている．[1, 2]. PSU は単純だが効果的に消費エネルギーを削減することができる．PSU ではプロセッサの負荷が低い時にクロック周波数を低下させ，余裕のできたクロック・サイクル時間を使って複数のパイプライン・ステージを統合する．この時，統合されたステージ間のパイプライン・レジスタはバイパスされる．PSU によって消費エネルギーを削減できる理由は次の 2 点である．まず第 1 に，バイパスされるパイプライン・レジスタへのクロックの供給を止めることにより，クロック・ドライバの総負荷を減少させることができる．これにより，消費エネルギーが削減される．第 2 に，パイプライン・ステージの統合によりプロセッサのパイプラインが短くなる．そうすることで，IPC（Instructions Per Cycle）が向上する．これにより，プログラムの実行に必要なサイクル数が削減され，エネルギーを消費する時間を短くすることができる．

今までの PSU は，MIPS 命令セットをベースとした命令セットである，Sim-

pleScalar PISA[3] のみで評価されてきた．しかし，後の章で説明するように，PSU はステージ統合を行った時の IPC の向上に反比例して消費エネルギーが削減される．よって，命令セットやパイプラインの構成が変化すればステージ統合時の IPC の向上量が変化するため，消費エネルギー削減量も変化する．そこで，本論文では，MIPS 命令セットとは異なる命令セットにおける，パイプラインステージ統合の有効性を調査する．

調査に用いる命令セットとしては，ARM 命令セットを用いた．以下，ARM 命令セットを選択した理由を 2 点述べる．第 1 の点としては，ARM 命令セットが実行命令数や条件分岐を削減することを意識して一般的な RISC（Reduced Instruction Set Computer）の命令セットに対して，様々な拡張を実装した命令セットという点である．SimpleScalar PISA は，一般的な RISC の命令セットに忠実であるため，同様に一般的な RISC の命令セットに忠実な命令セットを選択しても，SimpleScalar PISA とそれほど差がない結果となる．よって，全ての命令に対して条件付き実行を行うことが可能であったり，ALU 演算の実行時に，入力に対してシフトを行ってから演算が可能であるなど，特徴を持った ARM 命令セットは比較対象としてはうってつけであったためである．第 2 の点として，ARM 命令セットが近年において非常に多く用いられている命令セットであり，なおかつ，さらにその勢力をのばしている命令セットだからである．

本論文では，ARM 命令セットの各々の拡張が PSU の効果に対する考察を行い，平均すると，ARM 命令セットの拡張は PSU の効果をより高めるという結論を得た．この考察を検証するため，ARM 命令セットを実行するパイプラインを仮定してシミュレータを作成し，MiBench というベンチマーク・セットを用いて，従来の SimpleScalar PISA の結果と比較した．評価の結果，ARM 命令セットは 2 ステージを統合した場合に SimpleScalar PISA よりも平均 14.5%，4 ステージを統合した場合に平均 17.2%，PSU の効果が大きいという結果を得た．また，ARM 命令セットの方が消費エネルギーを削減できなかったベンチマーク・プログラムについては，当初の考察で PSU の効果が小さくなる拡張を多用していることが確認でき，考察は妥当性であったと考えられる．

残りの論文の構成は以下のとおりである．2 章では PSU について述べる．この章では，過去の PSU の研究，PSU を実装する際の回路，そして，PSU による消費エネルギーの削減について解析的に説明する．3 章では本研究の目的である ARM 命令セットで PSU の評価を行う意義について説明し，ARM 命令セッ

トの特徴の説明，及び，それが PSU に及ぼす影響について議論を行う．4 章で SimpleScalar PISA と ARM 命令セットのシミュレーション環境，及び各々のパイプラインの仮定について説明する．5 章で評価結果と考察を行う．最後に，6 章でこの論文についてまとめを行う．

# 第 2 章　パイプラインステージ統合（PSU）とは

パイプラインステージ統合（PSU: Pipeline Stage Unification）は嶋田らによって提案され [1]，Dynamic Voltage Scaling（DVS）という，現在の多くの商用プロセッサに採用されている消費エネルギー削減手法と比較されてきた．

まず，DVS について説明する．DVS はバッテリ持続時間要求やプロセッサ負荷に応じて，動的にクロック周波数と電源電圧を変更するものである．バッテリ持続時間要求が強いか，与えられた負荷が低ければ，クロック周波数を低下させる．さらに，延びたクロック・サイクル時間に信号の遅延を合わせ，電源電圧を低下させる．これにより，プログラム実行に要する消費エネルギーを削減する．

このように，DVS は消費電力を削減する有効な手法であるが，将来の半導体製造技術では，サブスレッショルドリーク電流の増加によって閾値電圧が下げにくくなる点や，過渡故障の増加という面から，将来のプロセス技術ではその有効性は減少する．

これに対して，PSU は半導体製造技術に依存しない消費エネルギー削減手法として提案された．PSU では負荷が低い時にクロック周波数を低下させ，余裕のできたクロック・サイクル時間を使って複数のパイプラインステージを統合する．PSU によって消費エネルギーを削減できる理由は次の 2 点である．まず第 1 に，バイパスされるパイプライン・レジスタへのクロックの供給を止めることにより，クロック・ドライバの総負荷を減少させることができる．これにより，消費電力が削減される．第 2 に，パイプライン・ステージの統合によりプロセッサのパイプラインが短くなる．そうすることで，IPC（Instructions Per Cycle）が向上する．これにより，プログラムの実行に必要なサイクル数が削減され，電力を消費する時間を短くすることができる．

過去の研究では，PSU は DVS よりも多く消費エネルギーを削減することができ，さらに，その削減量の差は，将来の半導体製造技術においてより大きくな

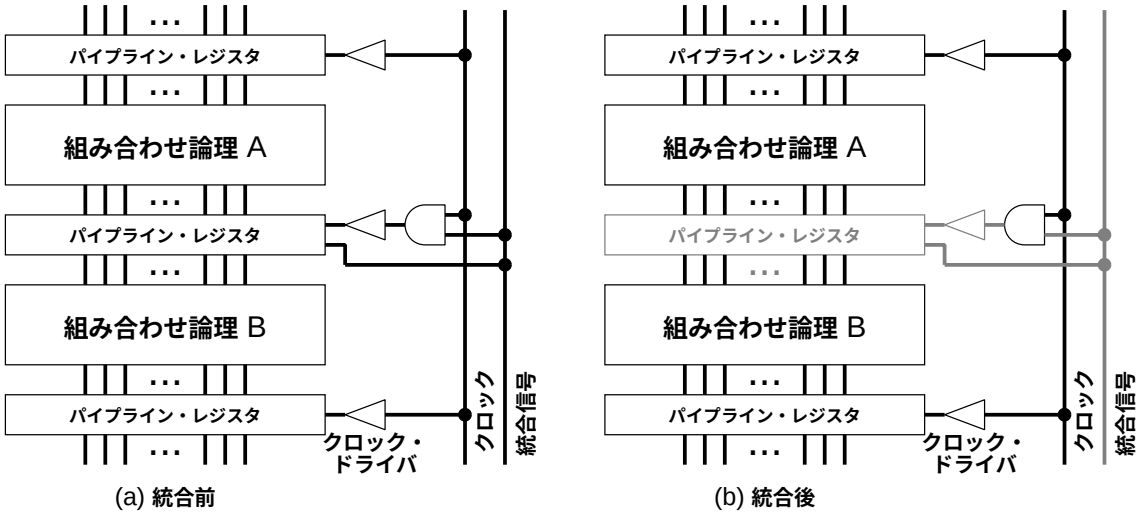



図 1: クロック分配線の変更と統合信号の追加

ることが示さている．[2, 4, 5, 6]．なお，PSU と同様に，動的にパイプラインの段数を変更する研究としては，Efthymiou らの Adaptive Pipe Depth Control[7] や Koppanalil らの Dynamic Pipeline Scaling[8] が存在する．

本章では，PSU の実装，及び消費エネルギーの解析について述べる．2.1 節では PSU を実現するための実装方法を述べる．2.2 節では PSU を適用した場合，消費エネルギー削減量の解析を PSU を行った場合について述べる．

## 2.1 PSU の実装

2.1.1 節で PSU を実現するための基本的な回路の変更を示す．PSU の大部分は，この節で説明する方法で実装できるが，例外的に取り扱わなければならない部分が存在する．2.1.2 節ではこれについて述べる．

### 2.1.1 PSU のための回路

図 1 に PSU に関連する信号線とパイプライン・レジスタとの結線関係を示す．説明を簡単にするために，2 ステージの統合を例としている．図 1 に示すように，パイプライン・レジスタには，クロックの階層ネットワークの最終段のクロック・ドライバの出力が入力されている．また，PSU のための信号線として，**統合信号**と呼ぶ，統合を指示する信号線が追加される．

図 1 (a) はステージを統合していない状態を，図 1 (b) は統合した状態を示す．図中の黒い部分は動作部分を示し，灰色の部分は動作していない部分を示

す．図 1 (a) は通常のパイプラインとして動作し，統合信号は 1 である．隣接する組合せ論理回路 A と B は，それらの回路の間のパイプライン・レジスタが動作しているため，異なったステージとして動作する．一方，図 1 (b) では，統合信号を 0 にすることによって組合せ論理回路 A と B の間のパイプライン・レジスタへのクロックが入力されなくなり，信号はバイパスされる．したがって，このパイプライン・レジスタは動作せず，2 つの組合せ論理回路は 1 つのステージとして動作する．

以上では 2 ステージ統合の場合についてのみ述べたが，統合信号を複数用意し，クロック・ドライバ停止のための信号を適切に制御することにより，さらに多くのステージの統合を行えるように拡張可能である．

ここで，パイプライン・レジスタには，フロントエンド，実行コア，バックエンド間にある命令ウィンドウなどデカップリング用の記憶素子を含まないことを注意しておく．これらは，パイプライン・レジスタと同じくステージをつなぐ記憶素子であるが，複数の命令の状態を記憶する機能が必要なので，バイパスさせることはできない．

パイプライン・レジスタをバイパスさせるには，2 つの方法が考えられる．1 つめの方法は，統合時にクロックとは無関係に信号が通過するようパイプライン・レジスタの論理を構成することである．透過型のラッチでパイプライン・レジスタを実現する場合，この論理の変更は容易である．2 つめの方法は，パイプライン・レジスタの後方にマルチプレクサを配置し，パイプライン・レジスタの出力と前方のステージの出力を統合信号により選択する方法である．この方法は，パイプライン・レジスタをどのような回路で構成しても適用可能である．

インターロック回路にもわずかな変更が必要である．図 2 に変更したインターロック回路を示す．図 2 (a) に統合していない状態での信号の流れを，図 2 (b) に統合した状態での信号の流れを示す．

一般に，あるステージのストール信号がアサートされるのは，そのステージでハザードを検出するか，または，後方のステージのストール信号が真のときである．図 2 に示した例では，$S(A)$ と $H(A)$ を，それぞれ，ステージ A のストール信号，ハザード信号とすると，$S(A)$ は以下の式により与えられる：

$$S(A) = H(A) + S(B) \tag{1}$$

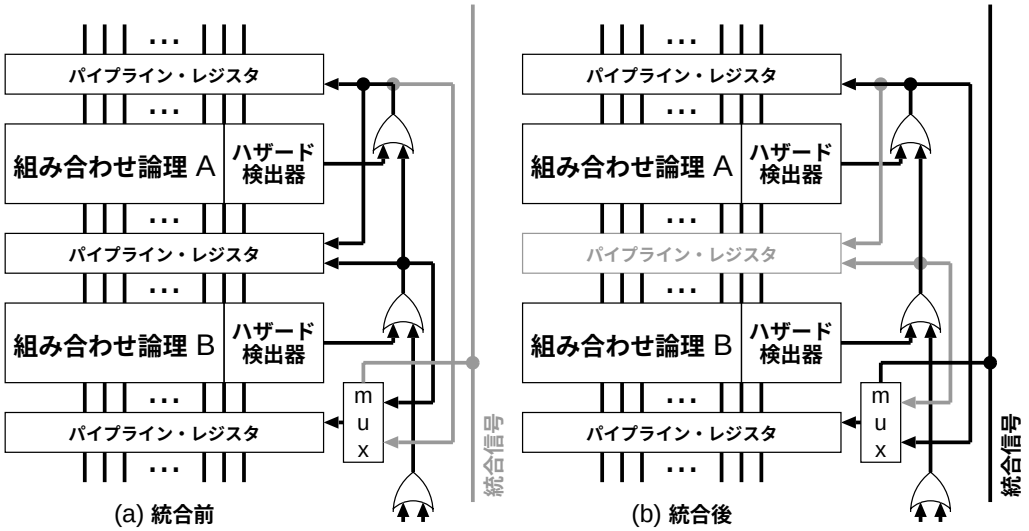



図 2: インターロック回路

ステージ A と B を統合する場合，統合後のステージのストール信号 $S(A+B)$ は，A または B がストールするときアサートされる必要があるので，以下のようになる：

$$S(A+B) = S(A) + S(B) \tag{2}$$

式 (1) を式 (2) に代入すると，

$$S(A+B) = H(A) + S(B) \tag{3}$$

となり，結局，$S(A)$ と等しいことが分かる．

ところで，ストール信号によりステージを実際にストールさせるためには，そのステージの前方にあるパイプライン・レジスタの更新を抑止し，後方のパイプライン・レジスタへ渡す信号を NOP に変更する必要がある．組合せ回路 A の前方のパイプライン・レジスタへ渡す信号は，統合前後で，それぞれ，$S(A)$, $S(A+B)$ なので，式 (1)，(3) より変更の必要がないことが分かる．一方，組合せ回路 B の後方のパイプライン・レジスタへ渡す信号については，統合するかどうかに応じて $S(B)$ と $S(A+B)$ を切り替える必要がある．このためにマルチプレクサが必要である．

以上では 2 ステージ統合の場合についてのみ述べたが，統合信号，および，関連する回路を複数用意して適切に動作させることにより，さらに多くのステー

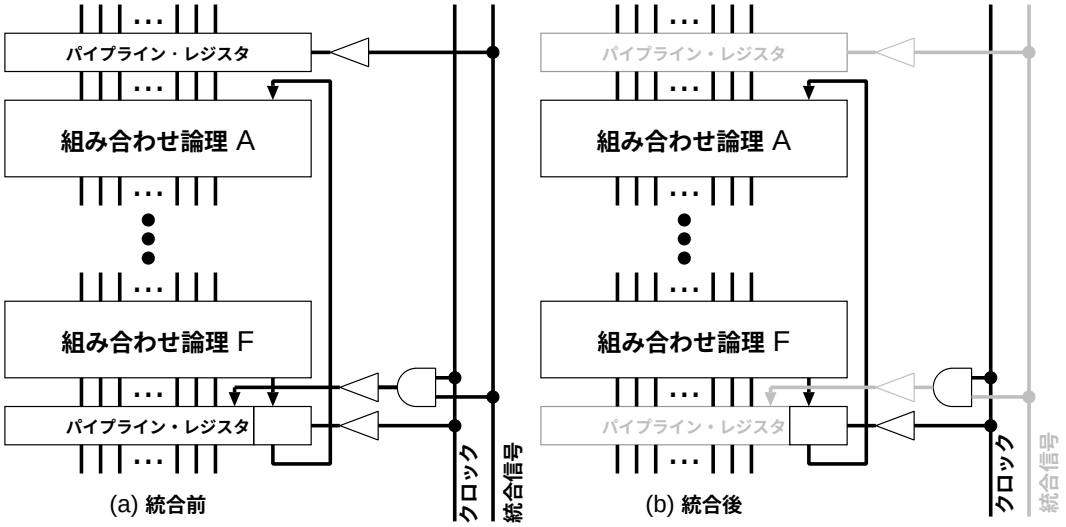


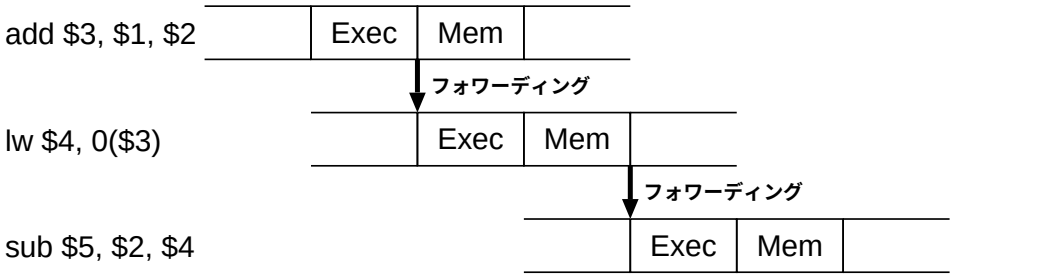

図 3: ループをなす信号パス中のパイプライン・レジスタ

ジの統合を行えるように，統合するステージ数に比例する程度の複雑さで拡張可能である．

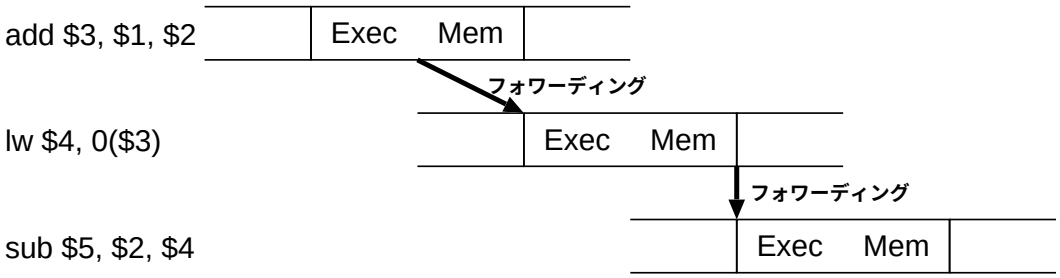
### 2.1.2 ループをなす信号パスにおけるパイプライン・レジスタ

前節で述べたように，統合するステージ間のパイプライン・レジスタは基本的にバイパスさせるが，例外がある．ループをなす信号パスにおいて，後方のステージへ信号を出力するパイプライン・レジスタは，統合するステージ間であってもバイパスさせない．図 3 に例を示す．図 3 (a) は，ステージ F の出力がパイプライン・レジスタを通り，後方のステージ A に入力され，再びステージ F に戻るという，ループをなす信号パスを持つパイプラインを示している．ステージ F とその次のステージを統合する際，図 3 (b) に示すように，統合されるステージ間のパイプライン・レジスタは基本的にバイパスさせるが，ループをなす信号パス上にある部分はバイパスさせない．パイプライン・レジスタをバイパスさせないことにより，ループをなす信号パスは正しいタイミングを維持できる．このようなループをなす信号パスの例として，プログラム・カウンタの読み出しと更新，命令ウィンドウやロード／ストア・キューの割当てにおけるヘッド／テール・ポインタの更新，命令発行とそれによる命令の wakeup，命令の実行結果のフォワーディングなどがあげられる．

図 4 に，命令の実行結果のフォワーディングのタイミングを例として示す．単純な DLX 型の 5 段のパイプライン [9] において，実行ステージ（Exec）とメ




(a) 統合前




(b) 統合後

図 4: 実行結果のフォワーディングのタイミング

モリ・アクセス・ステージ（Mem）を統合した場合を考える．Exec と Mem ステージの間のパイプライン・レジスタは基本的にバイパスされるが，実行結果のフォワーディングの部分ではバイパスされない．図 4 (a) および (b) は，それぞれ統合前および統合後のタイミングを示している．図 4 (b) に示すように，Exec ステージで得られた `add` 命令の結果は，バイパスさせない Exec と Mem ステージ間のパイプライン・レジスタで保持され，次のサイクルに，正しいタイミングで Exec ステージの `lw` 命令に渡される．一方，Mem ステージで得られる `lw` 命令の結果は，通常のタイミングで Exec ステージの `sub` 命令に渡される．統合後も，`add` 命令のレイテンシは 1 サイクルであるが，`lw` 命令のレイテンシは統合により，1 サイクルに短縮されることに注意されたい．

## 2.2 PSU による消費エネルギー削減量の解析

本章では，PSU による消費エネルギーの削減について解析的に説明する．一般に，プログラムの実行の際に消費するエネルギー $E$ は以下の式で表される：

$$E = P \times T_{ex} \tag{4}$$

ここで，$P$ は消費電力，$T_{ex}$ は実行時間である．$P$ と $T_{ex}$ は以下の式で与えられる：

$$P = a \times f \times C \times {V_{DD}}^2 \tag{5}$$

$$T_{ex} = \frac{N}{IPC \times f} \tag{6}$$

ここで，$a$ はアクティビティ・ファクタ，つまり，各ノードの平均スイッチング確率，$f$ はクロック周波数，$C$ はスイッチするノードの全容量，${V_{DD}}^2$ は電源電圧，$N$ は実行命令数，$IPC$ は 1 サイクルあたりの実行命令数の平均である．

2 章で述べたように，PSU は一時クロックのドライバを停止することによって消費電力を削減する．$U$ ステージ統合（以下，これを**統合度** $U$ と呼ぶ）でプロセッサが動作しているとき，理想的にはクロック・ドライバが消費する電力は $1/U$ となる．また，通常のプロセッサと同じく，総消費電力はクロック周波数の低下率に比例して削減する．したがって，クロック周波数 $f_{low}$ で動作する，統合度 $U$ の PSU プロセッサの消費電力は以下の式で表される：

$$P_{PSU}(f_{low}, U) = \left(P_{total} - P_{clock} + \frac{P_{clock}}{U}\right) \times \frac{f_{low}}{f_{max}} \tag{7}$$

ここで，$P_{total}$ と $P_{clock}$ はそれぞれ，通常モードにおけるプロセッサの総消費電力とクロック・ドライバの消費電力である．

式 (4) を用い，通常モードで正規化した消費エネルギーは以下の式で表される：

$$E_{PSU}, n(f_{low}, U) = \frac{P_{PSU}(f_{low}, U) \times T_{ex}(f_{low}, U)}{P_{total} \times T_{ex}(f_{max}, 1)} \tag{8}$$

ここで，$T_{ex}(f, U)$ はクロック周波数 $f$ と統合度 $U$ における実行時間である．$T_{ex}(f_{max}, 1)$ は通常モードにおける実行時間であることを注意されたい．式 (6) と (7) を式 (8) に代入することにより，以下の式を得ることができる：

$$E_{PSU}, n(f_{low}, U) = \frac{IPC_{max}}{IPC_{low}} \times \left\{1 - k \times \left(1 - \frac{1}{U}\right)\right\} \tag{9}$$

ただし，

$$k = \frac{P_{clock}}{P_{total}} \tag{10}$$

である．

式 (9) から分かるように，消費エネルギーは IPC の向上に反比例する（パイプラインが短縮されるため，$IPC_{max} < IPC_{low}$ であることに注意）．また，消費エネルギーの削減は，クロック・ドライバにより消費される電力が，プロセッサの総消費電力のどれだけの割合を占めているかということに依存する．この割合は，近年の高速なプロセッサでは非常に大きく（たとえば，Alpha 21264 では 32%[12]），この傾向は深いパイプラインや小さいクロック・スキューなどの達成のために将来も続き，PSU は消費エネルギーを大きく削減できると期待できる．

# 第 3 章　ARM 命令セットにおける PSU の効果

2 章の式より，PSU の消費エネルギー削減は，ステージ統合を行った時の IPC の向上に反比例する．よって，命令セットやパイプラインの構成が変化すればステージ統合時の IPC の向上量が変化するため，消費エネルギー削減量も変化する．そこで，本論文では，MIPS 命令セットとは異なる命令セットにおける，パイプラインステージ統合の有効性も調査する．

調査に用いる命令セットとしては，ARM 命令セットを用いた．以下，3.1 節では ARM 命令セットを選択した理由について述べ，3.2 節では命令セットの特徴について説明する，そして，3.3 節では ARM 命令セットにおいて PSU を実装した場合，ARM 命令セットの特徴より起こりうる PSU の効果の変化を考察する．

## 3.1　評価に用いる命令セットの選択

調査に用いる命令セットとしては，ARM 命令セットを選択した．以下，ARM 命令セットを選択した理由を 2 点述べる．第 1 の点としては，ARM 命令セットが実行命令数や条件分岐を削減することを意識して一般的な RISC の命令セットに対して，様々な拡張を実装した命令セットという点である．この拡張は，プロセッサの消費電力の削減を目的としており，全命令に対する条件実行フラグの付加，演算命令に対するシフト・オペランドの付加，ロード／ストア・マル

チ命令の採用等があげられる．これらの詳細については3.2節で述べる．これらの拡張によってプログラム・サイズが小さくなり，命令のメイン・メモリからの読み込みの削減や，条件分岐の削減による制御投機の削減等により，消費電力の削減を達成している．第2の点として，ARM命令セットが近年において非常に多く用いられている命令セットであるためである．特に，携帯電話やデジタル家電等に搭載されるプロセッサである組み込み用プロセッサの世界では，現時点で70%以上のシェアを獲得しておりさらに市場を伸ばしている傾向にある．このような，将来性のある命令セットにおいてPSUの有効性を示すことは，非常に意義があることだと考える．

## 3.2 ARM命令セットについて

本研究では，ARM命令セットの中の，ARM V4命令セット[10]をベースとする．この命令セットの特徴として，消費電力をさらに削減するために，浮動小数点演算命令を省略している点がある．これにより，浮動小数点演算器，浮動小数点レジスタ，ファイル，浮動小数点演算命令用デューダ等を省略できるため，大幅に消費電力やチップ面積を削減できる．しかし，プログラムの実行に浮動小数点演算を必要とする場合は，そのプログラムのコンパイル時にsoft-floatライブラリを用いて，浮動小数点演算を整数演算に変換しなくてはならない．これによるオーバーヘッドは実行命令数が増えるのみならず，変換された命令によって汎用レジスタが多数使われるため，非常に大きくなるものとなる．

ARM命令セットは一般的なRISC命令セットと同様に，ロード／ストア・アーキテクチャを採用している．これは，演算命令ではレジスタ中の値もしくは命令の中にエンコードされている即値のみを演算対象とする．よって，メモリ上の値に対する演算は，ロード命令にて値をレジスタに読み込んだ後に演算を行い，結果をストア命令にてメモリに書き戻すことになる．また，通常の整数演算は3オペランド形式であり，演算は2つのソース・レジスタの値に対して行われ，結果はデスティネーション・レジスタに格納される．

以下に，一般的なRISC命令セットとは異なる，ARM命令セットの独特な仕様について述べる．

- 条件実行

  ARM命令セットでは，全ての命令に対して条件実行が行える．これを実現するために，命令のビット列には4bitの条件フラグがある．これを用い

ることによって，プログラム中の小規模な if-else 構文を条件分岐命令なしで実現し，制御依存によるパイプライン・バブルを削減することができる．なお，条件分岐命令自体もこの条件実行を用いて実現されている．

- シフト・オペランド
  特定の値をシフトしてから演算するという処理は，通常の演算の中でよく見られるため，ARM 命令セットではこの 2 つの演算を 1 命令で実現可能な実装がされている．具体的には，全ての整数演算命令の片方のソース・レジスタ値に対して，即値もしくは任意のレジスタ値だけシフトを行った後，その演算を行うことができる．このため，シフタと演算器がシリアルに接続されている．
- ロード／ストア・マルチ命令
  ARM 命令セットでは，メモリ上の特定のアドレスを先頭とする 32 ビットの値の配列を，指定した複数の 0 から 15 番の汎用レジスタに読み出しする命令が存在する．これを，ロード・マルチ命令と呼ぶ．また，これとは逆の操作を行う，ストア・マルチ命令も存在する．これら 2 つの命令を使うことで，少ない命令数で配列に対する演算を実現することができる．
- 汎用レジスタの構成
  多くの RISC 命令セットでは，汎用レジスタは 32 本持っているが，ARM 命令セットでは 16 本となっている．これは，消費電力削減のためにレジスタ数を削減すること，および，命令のビット列中に条件フラグを付加するために，レジスタ指定に使うビット列を削減することを目的としたと考えられる．また，汎用レジスタ 15 番はプログラム・カウンタ（PC）となっており，この汎用レジスタ 15 番に対して演算をすることによって，PC 相対分岐を実現している．これにより，PC 相対分岐命令を削減している．

## 3.3 ARM 命令セット独特な仕様の PSU への影響の検討

3.2 節で示したように，ARM 命令セットには非常に独特な仕様があり，これが PSU の消費エネルギー削減に対して大きな影響を及ぼす可能性がある．以下，その影響について検討した結果を記す．

- 条件実行による影響
  ARM 命令セットでは条件実行を用いることにより，小規模な if-else 構文における条件分岐命令と無条件分岐命令を削除できる．これにより，分岐に

よる命令アドレスの変化によって命令フェッチがとぎれたり，条件分岐命令の分岐予測ミスによって分岐予測ミス・ペナルティを被ることがなくなるため，プログラムの実行の高速化という点で大きなアドバンテージとなる．参考文献 [2] に示されているように，SimpleScalar PISA での評価では，分岐予測ミス・ペナルティの削減が最も IPC の向上に影響している．条件分岐が少ないということは，PSU を適用する前の状態でも分岐予測ミス・ペナルティによって低下している IPC が少ないため，PSU を適用してもそれほど IPC が向上しないことになる．よって，ARM 命令セットにおいて，条件実行を多用して条件分岐を削減しているベンチマーク・プログラムでは，PSU の効果が少なくなると考えられる．

- シフト・オペランドによる影響
  シフト・オペランドを利用しても，従来の 2 命令を 1 つにまとめるだけであり，ロード／ストア・マルチ命令のように，特定のパイプラインに負荷が集中するようなことはない．よって，シフト・オペランドの使用による PSU の効果に対する変化は，ほとんどないと考えられる．
- ロード／ストア・マルチ命令の影響
  ロード／ストア・マルチ命令は命令数の削減に効果があるが，しかし，実際に実行する時には，複数回のロード／ストアとして実行することには変わらない．よって，ロード／ストア・マルチ命令の実行時にはロード／ストアのパイプライン以外が遊ぶ可能性が高く，処理の分散という点では嬉しくないと考えられる．

  以下，配列の各要素に対しての演算を例に挙げて考える．一般的な RISC 命令セットで実現する場合には，ロード／ストア命令で配列の 1 要素を読み出した直後に演算を行うような命令列になる．これをロード・マルチ命令を利用した場合，ロード・マルチ命令で配列の複数要素を読み出した後にそれらに対して演算を行うことになる．この場合，命令列はロード・マルチ命令の後に複数の演算命令が続くことになり，実際の実行では，複数のロードの実行の後に複数の演算という形になる．上記の場合，一般的な RISC 命令セットでは，ロード／ストアのパイプラインと演算器が交互に使われるのに対し，ロード／ストア・マルチ命令を使う場合は，ロード／ストアのパイプラインを集中的に使用された後，演算器が集中的に使用されるという結果になる．

PSU を適用すると，ロード／ストアのパイプラインが短くなり，連続するロード／ストア処理の完了が早くなる．よって，ロード／ストア・マルチ命令によるロード／ストアのパイプラインに負荷が集中する時間が短くなり，性能向上に結び付くと考えられる．よって，ロード／ストア・マルチ命令を多用する場合，PSU の効果は大きくなると考えられる．

- 浮動小数点演算器を省略したことによる影響
  一般的に，浮動小数点演算器を備えているプロセッサにおいても，浮動小数点演算は整数演算よりも実行に時間がかかるため，一般的なプログラムでは可能な限り浮動小数点演算は排除されてる．よって，浮動小数点演算器を省略した ARM 命令セットにおいても，多くのプログラムでは浮動小数点演算を soft-float ライブラリで整数演算に変換することによるオーバーヘッドはそれほど大きくないと予想できる．
  PSU に対する影響としては，PSU の効果を高める方向に働くと考えられる．以下，理由を説明する．soft-float を用いることによって実行する命令が増えるため，実行時に分岐予測のための履歴表やキャッシュに保持すべき内容が増える．これによって，分岐予測成功率やキャッシュ・ヒット率が悪化するため，PSU による分岐予測ミス・ペナルティやキャッシュ・ヒット・レイテンシの削減の影響が大きくなると考えられる．

上記より，ARM 命令セットでは PSU の効果が小さくなるよりも大きくなる要素の方が多いため，平均すると，一般的な RISC プロセッサに PSU を適用するより効果は大きくなると考えられる．

# 第 4 章　評価環境

2.2 章の式 (9) から分かるように，PSU の消費エネルギーには IPC と統合度 $U$ とクロック・ドライバの消費電力の割合 $k$ が関係する．本章では最初に IPC の評価環境を示す．次に，具体的にパイプラインを仮定し，どのように統合するか述べる．最後に，近年の商用プロセッサのクロック・ドライバの消費電力を示し，k の値を決定する．

## 4.1　シミュレーション環境

SimpleScalar PISA 命令セットの評価には SimpleScalar Tool Set [3] 中の out-of-order 実行シミュレータを用いた．また，ARM 命令セットの評価には研究の

表 1: MiBench の処理内容

| ベンチマーク名 | 処理内容 |
|---|---|
| basicmath | 機器の制御で使われる様々な演算の実行 |
| bitcount | 機器の制御で使われる様々なビット操作の実行 |
| qsort | 配列に対するクイック・ソート |
| susan | 画像に対する様々なフィルタ処理 |
| jpeg | 画像の圧縮／伸長 |
| dijkstra | Dijkstra 法によるネットワークの経路探索 |
| stringsearch | テキスト中の単語の検索 |
| blowfish | 暗号アルゴリズムによる暗号化／復号化 |
| rijndael | 暗号アルゴリズムによる暗号化／復号化 |
| sha | 一方向ハッシュ関数によるハッシュ値の生成 |
| CRC32 | 32bitCRC によるエラー訂正 |
| FFT | FFT による周波数変換 |
| adpcm | 音声の ADPCM 法による符号化／復号化 |

過程で作成したシミュレータを用いた．具体的には，ARM 命令セットを実行するシミュレータをパイプライン化およびスーパスカラ化し，深いパイプラインをもったシミュレータに拡張する．PSU を段階的に適用するため，ベースとなるパイプラインは深いパイプラインを持つとする．それぞれのシミュレータのパイプラインのステージ数を変化させ，IPC を測定した．ベンチマーク・プログラムとして，画像処理，音声処理，暗号処理の民生用機器で多用される処理を中心とした MiBench[11] より 13 本を用いた．各ベンチマーク・プログラムの内容については，表 1 に示す．ベンチマーク・プログラムのコンパイルには，SimpleScalar PISA は gcc ver.2.7.2.3，ARM 命令セットは gcc ver.4.0.0 を用いた．

表 2 に，シミュレーションにおいて用いたプロセッサの構成を示す．どちらのプロセッサも近年のプロセッサと同様に，深いパイプラインを持つと仮定した．メモリ・アクセス時間は，プロセッサのクロック周波数の低下に比例して遅くなると仮定した．そのため，プロセッサのクロック周波数によらず，メモリアクセスのサイクル数は一定である．

表 2: プロセッサの構成

| 命令セット | | SimpleScalar PISA | ARM 命令セット |
|---|---|---|---|
| フェッチ，デコード幅 | | 2 | |
| 命令ウィンドウ | | 64 エントリ | |
| メモリ・ポート | | 1 | |
| | | ロード／ストア・キュー/32 エントリ | ストア・バッファ/32 エントリ |
| 機能ユニット | | 整数 ALU × 2<br>整数乗除算 × 1<br>浮動小数点 ALU × 1<br>浮動小数点乗除算 × 1 | シフト× 1<br>ALU × 1<br>アドレス生成 × 1<br>ロード／ストア × 1 |
| 分岐予測 | 予測方式 | gshare | |
| | 履歴 | 6 ビット | |
| | インデクス | 13 ビット | |
| | RAS | 16 エントリ | |
| キャッシュ | L1 命令 | 64KB/32B ライン/2-way | |
| | L1 データ | 64KB/32B ライン/4-way | |
| | L2 統合 | 2MB/64B ライン/4-way | |
| メモリアクセス | レイテンシ | 128 サイクル | |
| TLB | | 完全 | |

## 4.2 パイプラインの仮定

本研究では統合度 1，統合度 2，統合度 4 の 3 種類の統合度を仮定する．図 5 と図 6 に統合度 1，統合度 2 および統合度 4 の SimpleScalar PISA，ARM 命令セットそれぞれのパイプラインを示す．各ステージの動作について説明する．まず，図 5 に示す，SimpleScalar PISA については以下のとおりである．

- NextPC：分岐予測による次の PC の決定
- Fetch：命令キャッシュからの命令フェッチ
- Drive1：フェッチした命令のデコーダへの転送
- Alloc：RUU （register update unit），LSQ （load/store queue）の割当て
- Rename：レジスタリネーミング
- Queue：RUU への書き込み
- Schedule：命令スケジューリング
- Dispatch：RUU からの発行
- Register：レジスタ読み出し
- Exec：命令の実行
- Flags：フラグの書き込み
- BrnChk：分岐命令の実行結果と分岐予測の比較
- Drive2：分岐結果のフロントエンドへの転送
- Cache：データキャッシュアクセス

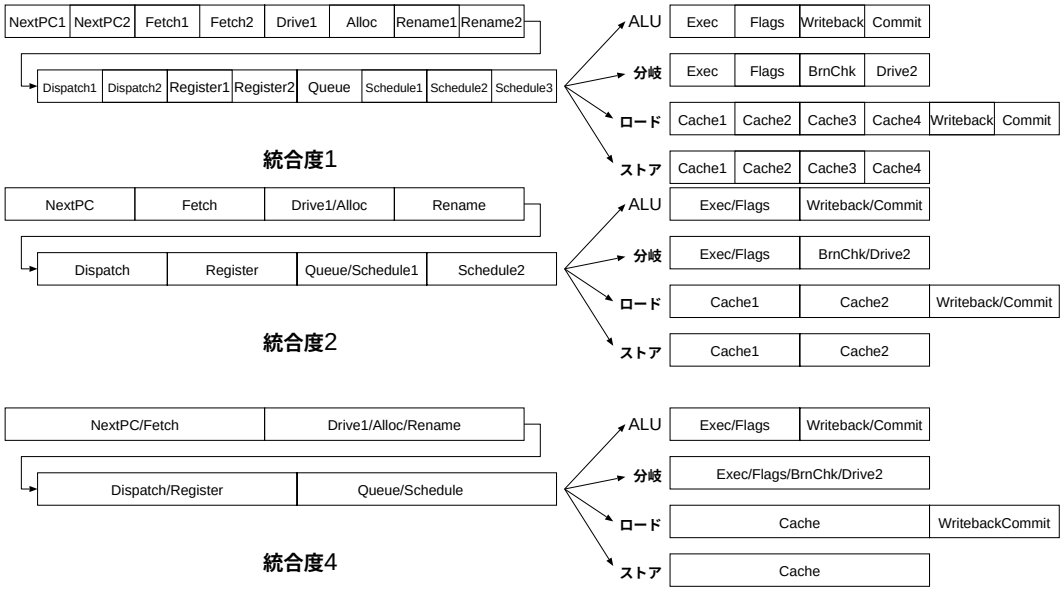



図 5: SimpleScalar PISA: 仮定した PSU のパイプライン

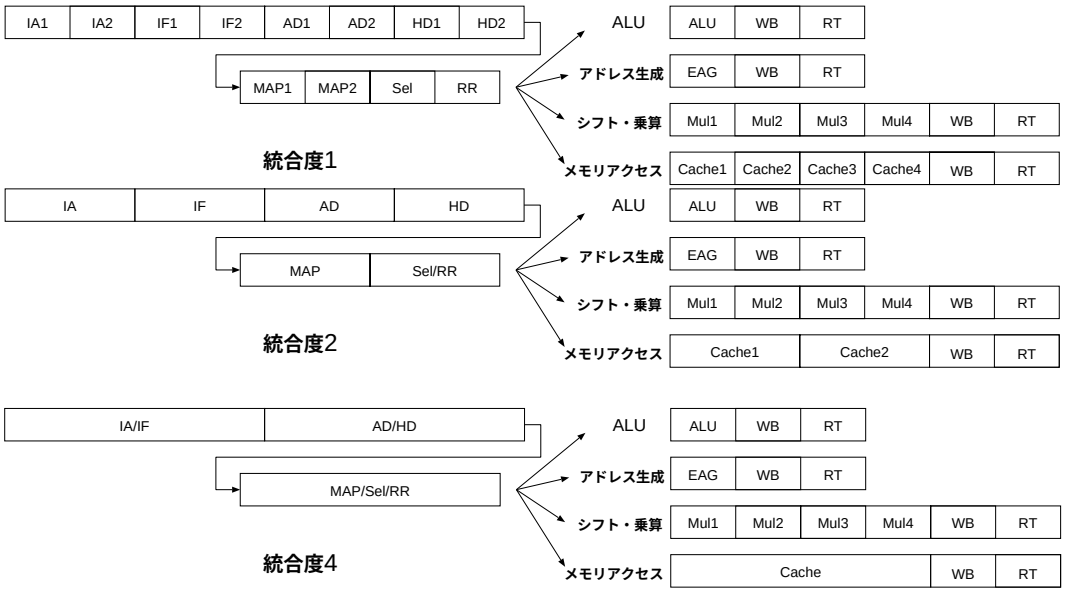



図 6: ARM 命令セット: 仮定した PSU のパイプライン

- Writeback：RUU へのライトバック
- Commit：実行結果のコミット

次に図 6 に示す，ARM 命令セットについては以下のとおりである．

- IA：アドレス生成

表3: SimpleScalar PISA の PSU において仮定する実行レイテンシ，キャッシュ・ヒット・レイテンシ，分岐予測ミス・ペナルティ

| 統合されるステージ数 | | 1 | 2 | 4 |
|---|---|---|---|---|
| クロック周波数 | | 100% | 50% | 25% |
| 実行レイテンシ | 整数乗算 | 3 | 2 | 1 |
| | 浮動小数点 ALU | 2 | 1 | 1 |
| | 浮動小数点乗算 | 4 | 2 | 1 |
| L1 キャッシュ・ヒット・レイテンシ | | 4 | 2 | 1 |
| L2 キャッシュ・ヒット・レイテンシ | | 16 | 8 | 4 |
| 分岐予測ミス・ペナルティ | | 20 | 10 | 5 |

- IF：命令フェッチ
- AD：ARM 命令のデコード
- HD：HOST 命令のデコード
- MAP：マッピング
- Sel：実行可能な命令の選択
- RR：命令に必要な値のレジスタの読みだし
- ALU：ALU 命令の実行
- EAG：アドレス生成の実行
- Mul：乗算命令の実行
- Cache：データキャッシュアクセス
- WB：レジスタの書き込み
- RT：命令のリタイア

表 3 に，SimpleScalar PISA パイプラインにおける命令の実行レイテンシ，分岐予測ミス・ペナルティ，キャッシュ・ヒット・レイテンシを示す．表の 2 行目に示すように，1，2，4 ステージを統合した場合，それぞれ最大クロック周波数の 100%，50%，25%で動作する．

なお，SimpleScalar PISA の整数/浮動小数点除算と平方根演算，同一資源を繰り返し使用し完全なパイプライン化はされておらず，ステージの統合はできないと仮定した．レイテンシはそれぞれ，20，12，24 サイクルとした．

表 4: ARM 命令セットの PSU において仮定するキャッシュ・ヒット・レイテンシ，分岐予測ミス・ペナルティ

| 統合されるステージ数 | 1 | 2 | 4 |
|---|---|---|---|
| クロック周波数 | 100% | 50% | 25% |
| L1 キャッシュ・ヒット・レイテンシ | 4 | 2 | 1 |
| L2 キャッシュ・ヒット・レイテンシ | 16 | 8 | 4 |
| 分岐予測ミス・ペナルティ | 13 | 7 | 4 |

表 4 に ARM 命令セットのパイプラインにおけるキャッシュ・ヒット・レイテンシと分岐予測ミス・ペナルティを示す．SimpleScalar PISA よりも短いパイプラインを仮定しているため，通常の演算の実行レイテンシの変化が無い点と，分岐予測ミス・ペナルティの減少が一定でない点が SimpleScalar PISA と大きく異なる点である．また，ARM 命令セットにおける乗算命令はマルチ・サイクルで実行されるが，その実行には同一資源を演算器繰り返し，パイプライン化されていないため，ステージの統合はできない．

## 4.3 クロックの消費電力の割合の仮定

プロセッサの総消費電力に対するクロックの消費電力の割合 $k$ は，プロセッサの設計によって異なる．文献 [12, 13, 14, 15] によれば，その範囲は 18%〜40%である．そこで，5 章の評価においては，これらの値のほぼ中央値である 30%と仮定した．また，クロック・ドライバが消費する電力は，駆動するパイプライン・レジスタ数に比例すると仮定し，単純に統合度 $U$ に反比例するとした．この仮定はおおまかではあるが，以下の理由でこの評価においては妥当であると考える．一般に，クロックは多段のネットワークにより分配するが，クロック・ドライバの消費電力のほとんどは，最終段のドライバで消費される（たとえば，Intel Itanium 2 の階層化クロック・ネットワークの場合，88%を最終段のドライバが消費している [13]）．また，最終段のドライバの負荷は，およそ，そのファンアウトであるパイプライン・レジスタの数に比例すると考えられる．

表 5: MiBench を用いた実行命令数の比較

| ベンチマーク名 | | ARM 命令セット | SimpleScalar PISA | 実行命令数の比 |
|---|---|---|---|---|
| basicmath | | 576091230 | 180209402 | 3.20 |
| bitcount | | 45494807 | 42067811 | 1.08 |
| qsort | | 17875297 | 46921050 | 0.38 |
| susan | smothing | 19987567 | 21480954 | 0.93 |
| | edges | 14252360 | 2060925 | 6.92 |
| | corners | 1978406 | 973244 | 2.03 |
| jpeg | encode | 28840714 | 25757691 | 1.12 |
| | decode | 6664125 | 7008181 | 0.95 |
| dijkstra | | 72039328 | 54881443 | 1.31 |
| stringsearch | | 129072 | 249396 | 0.52 |
| blowfish | encode | 26467242 | 36130970 | 0.73 |
| | decode | 26171050 | 36123242 | 0.72 |
| rijndael | encode | 23195748 | 36603391 | 0.63 |
| | decode | 23235391 | 36564574 | 0.64 |
| sha | | 12672995 | 11009425 | 1.15 |
| CRC32 | | 20533919 | 27406144 | 0.75 |
| FFT.inverse | | 102872865 | 94363468 | 1.09 |
| FFT | | 192431161 | 141276463 | 1.36 |
| adpcm | encode | 36307805 | 30548555 | 1.19 |
| | decode | 26374636 | 25244364 | 1.04 |
| 幾何平均 | | | | 1.07 |

# 第 5 章　評価結果，考察

## 5.1　ARM 命令セットと SimpleScalar PISA との命令数比較

3.1 節より，ARM 命令セットは浮動小数点演算が省略されている．このため，浮動小数点演算命令の処理を soft-float ライブラリを用いて，複数の整数演算命令に置き換えている．つまり，浮動小数点演算命令が多いプログラムは ARM 命令セットを用いると，命令数が大幅に増えることになる．一方で，ARM 命令

セットには条件実行による分岐命令の削減，シフト・オペランドによるシフト命令の削減，ロード／ストア・マルチによるロード／ストア命令の削減など，命令数が削減される要素もある．そこで，ARM 命令セットと SimpleScalar PISA の間にどれだけの実行命令数の差があるかを評価した．

表 5 に MiBench を実行した時の各ベンチマークの ARM 命令セット，及び SimpleScalar PISA における実行命令数，及びその比を表す．比は ARM 命令セット/SimpleScalar PISA とした．

表 5 の実行命令数の比に着目すると，basicmath が 3.20 倍，susan.edges が 6.92 倍，susan.corners が 2.03 倍と ARM 側の命令数が非常に多いベンチマークが存在する．basicmath は約 3%，susan.edges は約 2%，susan.corners は約 1%，浮動小数点演算命令が含まれていることが分かった．つまり，ARM 側では浮動小数点演算命令の処理を soft-float ライブラリで整数演算に置き換えているため，このような差が出たと考えられる．なお，今回の評価では，soft-float ライブラリによるオーバーヘッドの定量的な評価は行っていない．一方，ARM 命令セットの他の命令数削減の拡張により，qsort と stringsearch の実行命令数の比はそれぞれ 0.38, 0.52 と，ARM 側の実行命令数が非常に少ないという結果になっている．よって，実行命令数の比の幾何平均は，全プログラムの平均で 1.07，ARM 側に不利な 3 つのプログラムを除外すると 0.87 となった．

以上より，ARM 命令セットでは，浮動小数点演算を soft-float ライブラリで整数演算に置き換えているにもかかわらず，平均では SimpleScalar PISA とそれほど実行命令数が変わらないことが分かった．

## 5.2 各命令セットにおける PSU による消費エネルギーの変化

図 7 と図 8 にそれぞれ SimpleScalar PISA と ARM 命令セットによる PSU の消費エネルギーの評価結果を示す．グラフの縦軸は通常モード（結合度 1）における消費エネルギーで正規化した消費エネルギーであり，横軸はベンチマークの名前，及び平均を示す．各ベンチマークにおける 2 本の棒グラフはそれぞれ左より結合度 2，結合度 4 の正規化消費エネルギーを示している．なお，仮定を SimpleScalar PISA と ARM 命令セットのパイプラインを実際にゲートレベルで評価したわけではないため，各々のパイプラインの消費電力は分からない．よって，消費エネルギーの比較は個々の命令セット内でのみ行う．また，SimpleScalar PISA と ARM 命令セットの比較は，PSU を適用した時の消費エネルギー削減

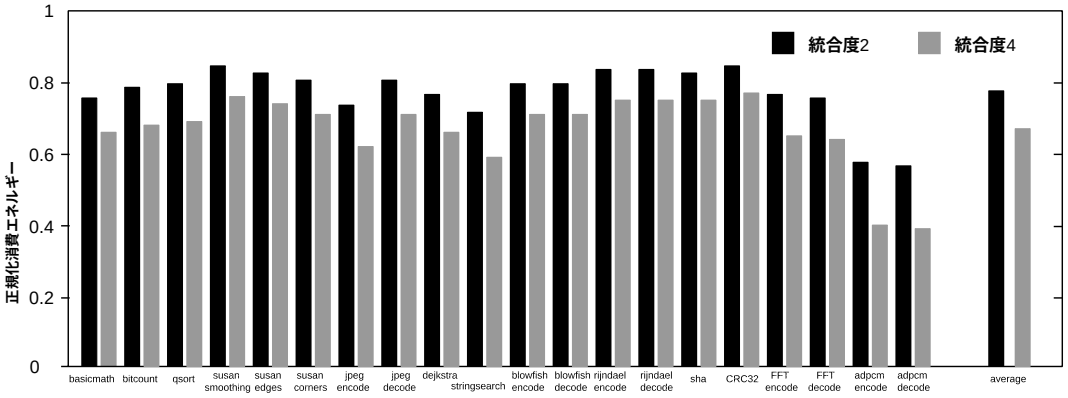



図 7: SimpleScalar PISA による統合度 2，4 の正規化消費エネルギー

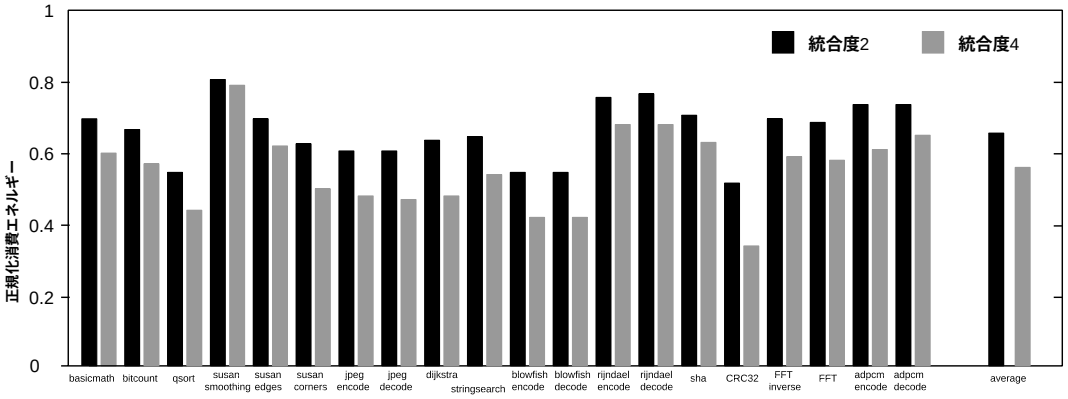



図 8: ARM 命令セットによる統合度 2，4 の正規化消費エネルギー

率で比較する．

SimpleScalar PISA の統合度 2 での PSU の正規化消費エネルギーは平均 0.770，統合度 4 での正規化消費エネルギーは平均 0.670 となった．一方，ARM 命令セットの PSU の統合度 2 の正規化消費エネルギーは平均 0.658，統合度 4 の正規化エネルギーは平均 0.555 となった．これより，全体的に ARM 命令セットの方が SimpleScalar PISA よりも PSU による消費エネルギーの削減率が大きいことがわかる．

また，ARM 命令セットと SimpleScalar PISA の結果と比較した結果，ARM 命令セットは 2 ステージを統合した場合に SimpleScalar PISA よりも平均 14.5%，4 ステージを統合した場合に平均 17.2%，PSU の効果が大きいという結果を得た．これは，3.2 節で考察したように，ARM 命令セットが一般的な RISC 命令セットより拡張した拡張した数々の要素が，PSU の効果を大きくする方向に働くことが多いことが証明されたと考えられる．

なお，adpcm.encode と adpcm.decode の 2 つのベンチマークでは，ARM 命令セットの方が PSU の効果が小さいとなった．この理由は，これら 2 つのベンチマークでは，3.2 節で PSU の効果を小さくする要素と考察した，条件実行を多用しているためである．adpcm.encode では，実に，ALU で演算された命令のうちの 55%が条件実行フラグが付加された命令であった．同様に，adpcm.decode では，ALU で演算された命令のうちの 38%が条件実行フラグが付加された命令であった．

# 第 6 章　まとめ

近年のモバイル・プロセッサは低消費エネルギーと高性能の両立が求められており，その要求に応えるために，PSU という方式が提案されている．過去の研究により，PSU は現在主流の消費エネルギー削減手法である DVS よりも消費エネルギー削減量が大きいことが示されており，また，その有効性は半導体の製造技術が進歩するにつれて大きくなることが示されている．しかし，今までの PSU は，SimpleScalar PISA のみでしか評価されていない．PSU では，命令セットの変化によってステージ統合を行った時の IPC の向上率が変化すると，消費エネルギー削減量も変化するため，異なる命令セットの評価も重要だと考えられる．そこで，本研究は，ARM 命令セットに対する PSU の有効性を調査した．

本論文では，ARM 命令セットの各々の拡張が PSU の効果に対する考察を行い，平均すると，ARM 命令セットの拡張は PSU の効果をより高めるという結論を得た．この考察を検証するため，ARM 命令セットを実行するパイプラインを仮定してシミュレータを作成し，MiBench というベンチマーク・セットを用いて，従来の SimpleScalar PISA の結果と比較した．評価の結果，ARM 命令セットは 2 ステージを統合した場合に SimpleScalar PISA よりも平均 14.5%，4 ステージを統合した場合に平均 17.2%，PSU の効果が大きいという結果を得た．また，ARM 命令セットの方が消費エネルギーを削減できなかったベンチマーク・プログラムについては，当初の考察で PSU の効果が小さくなる拡張を多用していることが確認でき，考察は妥当であったと考えられる．

# 謝辞

まず，この研究の機会を与えていただいた富田眞治教授に感謝の意を表します．また，本研究において，適切な御指導，御指摘を賜わった森眞一郎助教授，中島康彦助教授，嶋田創特任助手，三輪忍助手に心から感謝致します．

そして，日頃から助言，御指導をいただいた京都大学大学院情報学研究科通信情報システム専攻富田研究室の諸兄に感謝致します．

# 参考文献

[1] 嶋田創, 安藤秀樹, 島田俊夫,“低消費電力化のための可変パイプライン,” 情報処理学会研究報告, Vol. 2001-ARC-145, pp.57–62 (2001).

[2] 嶋田創, 安藤秀樹, 島田俊夫, “パイプラインステージ統合によるプロセッサの消費エネルギーの削減,” 情報処理学会論文誌, コンピューティングシステム, Vol. 45, No. SIG 1 (ACS 4), pp.18–30 (2004).

[3] D. Burger and T.M. Austin: “The SimpleScalar Tool Set”, Version 2.0, Technical Report CS-TR-97-1342, University of Wisconsin-Madison Computer Sciences Department (1997).

[4] H. Shimada, H. Ando, and T. Shimada, “Pipeline Stage Unification: A Low-Energy Consumption Technique for Future Mobile Processors,” The International Symposium on Low Power Electronics and Design 2003, pp.326-329, (2003).

[5] 嶋田創, 安藤秀樹, 島田俊夫, “パイプラインステージ統合： 将来のモバイルプロセッサのための消費エネルギー削減技術,” 2003 年先進的計算基盤システムシンポジウム SACSIS 2003, pp.283-290 (2003).

[6] H. Shimada, H. Ando, and T. Shimada, “Pipeline Stage Unification for Low-Power Consumption,” The 5th International Symposium on Low-Power and High-Speed Chips (COOL Chips V), pp.194-200 (2002).

[7] A. Efthymiou and J.D. Garside, “Adaptive Pipeline Depth Control for Processor Power-Management,” In Proceedings of International Comference on Computer Design 2002 pp.454-457 (2002).

[8] J. Koppanalil, P. Ramrakhyani, S. Desai, A. Vaidyanathan and E. Rotenberg, “A Case for Dynamic Pipeline Scaling,” In Proceesing of Compilers,

Architectures and Synthesis for Embedded Systems 2002, pp.1-8 (2002).

[9] J.L. Hennessy and D.A. Patterson, "Computer Architecture: A Quantitative Approach," 2nd Edition, Morgan Kaufmann Publishers Inc. (1996).

[10] ARM Inc. "ARM Architecture Reference Manual," (2000).

[11] M.R. Guthaus, J.S. Ringenberg, D. Ernst, T.M. Austin, T. Mudge, and R.B. Brown, "MiBench: A Free, Commercially Representative Embedded Benchmark Suite," In Proceedings of IEEE 4th Annual Workshop on Workload Characterization, pp.1-12, December 2001.

[12] M.K. Gowan, L.L. Biro and D.B. Jackson, "Power Considerations in the Design of the Alpha 21264 Microprocessor," In proceedings the 35th Design Automation Conference, pp.726–731 (1998).

[13] F.E. Anderson, J.S. Wells and E.Z. Berta, "The Core Clock System on the NextGeneration Itanium Microprocessor," *2002 IEEE International Solid-State Circuits Conference. Visual Supplement to the Digest of Technical Papers*, pp.110–111 (2002).

[14] L.T. Clark, E.J. Hoffman, J. Miller, M. Biyani, Y. Liao, S. Strazdus, M. Morrow, K.E. Velarde and M.A. Yarch, "An Embedded 32-b Microprocessor Core for Low-Power and High-Performance Applications," *IEEE Journal of Solid-State Circuits*, Vol.36, No.11, pp.1599–1608 (2001).

[15] P.E. Gronowski, W.J. Bowhill, R.P. Preston, M.K. Gowan, and R.L. Allmon, "High-Performance Microprocessor Design," *IEEE Journal of Solid-State Circuits*, Vol.33, No.5, pp.677–686 (1998).