Vol.2011-ARC-196 No.15
2011/7/28

イン・オーダ・パイプラインに適した 可変パイプライン段数プロセッサ制御機構の実装と評価

中林 智之[†1]　佐々木 敬泰[†1]
大野 和彦[†1]　近藤 利夫[†1]

近年，モバイル端末等の高性能化に伴う消費電力の増大が問題となっており，低消費電力と高性能の両立が要求されている．そこで著者らは，その要求を両立する手法として可変パイプライン段数アーキテクチャ (VSP;Variable Stages Pipeline) を提案している．VSP はプロセッサにかかる負荷に応じてパイプライン段数を動的に変更することで低消費電力と高性能の両立を目指す手法である．VSP の最適なパイプライン段数を予測するコントローラとして，著者らはメモリアクセスに着目したコントローラを提案している．しかし，このコントローラは Out-of-Order 実行型プロセッサへの適用を想定した手法であり，In-Order 実行型プロセッサに実装したときの有効性の評価が行われていない．また，プロセッサシミュレータ SimpleScalar を用いて評価を行っていたため，詳細な電力評価が行われていなかった．そこで本稿では，In-Order 実行型プロセッサに適した VSP コントローラを検討した上で，ハードウェア実装を行い，その有効性を明らかにする．評価の結果，提案するコントローラを実装するために必要なハードウェア規模は，プロセッサの 1%程度と非常に小規模で，電力遅延積を最大を 7%改善できることが明らかとなった．

Implementation and Evaluation of Controller for Variable Stages Pipeline Processor based on In-order Pipeline

Tomoyuki Nakabayashi,[†1] Takahiro Sasaki,[†1]
Kazuhiko Ohno[†1] and Toshio Kondo[†1]

Recently, the increase of the energy consumption of mobile computers caused by performance enhancement becomes one serious problem. Therefore, a lot of research for low energy and high performance computing is carried out. In order to reduce energy consumption, variable stages pipeline architecture (VSP), which dynamically varies the pipeline depth according to workload, is proposed. It is essential to predict accurate workload and apply suitable mode to bring out the best performance of the VSP. This paper proposes a mode controller that is suitable for the VSP that adopts in-order single pipelilne architecture. This paper implements the controller using Verilog HDL, and evaluates it using Synopsys PrimeTime PX. According to evaluation result, the mode controller can be implemented with a few additional hardware, which is only 1% of overall processor, and improves 7% energy-delay-products.

1. はじめに

近年，モバイルコンピューティングからハイパフォーマンスコンピューティングに至るまで広い分野において消費電力の増大が問題となっており，低消費電力と高性能を両立するための研究が盛んに行われている．現在の代表的な低消費電力化手法の 1 つである DVFS (Dynamic Valtage and Frequency Scaling)[1] は，動的に電源電圧と動作周波数を変化させることで消費電力を低減する．消費電力は電源電圧の 2 乗に比例するため，DVFS は消費電力を低減する手法としては有効である．しかし，プロセス技術の進歩により電源電圧が年々低下しており，将来的に電源電圧変化幅が減少するため消費電力削減効果の低下が予想される．また，動作周波数の低下に比例して性能が低下するという問題点もある．さらに，DVFS では数百命令単位で消費電力を削減することはできない．なぜなら，DVFS では電源電圧を変更する際，回路の充放電による電力オーバーヘッドが発生し，このオーバーヘッドが大きいためである．そのため，DVFS では切換の粒度は 1 万〜10 万サイクルのオーダーに設定する必要がある[2]．

そこで電源電圧に依存しないアーキテクチャレベルの低消費電力化手法として，パイプライン段数を動的に変更できる構造をもつプロセッサが提案されている[3)–11)]．これらの手法では，高性能が必要な場合には多段のパイプラインとして動作し，性能が必要でない場合には少段のパイプラインとして動作させることで，低消費電力と高性能を両立することを目指している．その手法の 1 つとして，我々は可変パイプライン段数プロセッサ（VSP;Variable Stages Piprline）を提案している[8)–11)]．

これらの動的にパイプライン段数を変更する手法では最適なパイプライン段数を予測し，適切なタイミングでパイプライン段数の切換を行うコントローラが重要となる．そこで，著者らは最適なパイプライン段数を予測するコントローラとして，メモリアクセス頻度に着目

†1 三重大学大学院工学研究科
Graduate School of Engineering, Mie University

したコントローラを提案している[9]．著者らは現在までに，プロセッサシミュレータを用いて Out-of-Order 実行型プロセッサにおいて，このコントローラが有効であることを明らかとしている．

しかしながら，より低電力が要求される組み込み分野において広く用いられている In-Order 実行型プロセッサにこのコントローラを適用した場合の効果は明らかになっていない．また，これまでプロセッサ全体での回路レベルの評価が行われておらず，詳細な電力およびハードウェアコストが明らかになっていなかった..

そこで，本稿ではまずこのコントローラを In-Order 実行型プロセッサに実装し，その有効性を評価する．さらに，In-Order 実行型プロセッサに対してより適したコントローラを検討した上でハードウェアを実装し，そのハードウェア規模および電力について評価を行う．

提案したコントローラを Verilog HDL を使用して実装し評価を行った結果，提案コントローラはプロセッサ全体に対してわずか 1%のハードウェア規模で実現することが可能で，電力遅延積を最大 7%向上できることが明らかとなった．

2. 可変パイプライン段数プロセッサ

可変パイプライン段数プロセッサ (VSP) は動作周波数とパイプライン段数をプロセッサにかかる負荷に応じて動的に切換えることで低消費電力と高性能の両立を実現する手法の 1 つである．具体的には，多段パイプライン構成で動作周波数を高くし高速に動作させる HS (High Speed) モードと，少段パイプライン構成で動作周波数を低くすることで消費電力を低減する LE (Low Energy) モードを用意し，2 つのモードを負荷に応じて切換えることで低消費電力と高性能の両立を目指している．

VSP は PSU (Pipeline Stage Unification) [5)–7)] と同様に図 1 のようなパイプライン構造を持つ．PSU では，パイプライン段数を動的に変更するためにパイプラインレジスタを図 2 のような構成 (以下,「D-FF+MUX」と呼ぶ) に変更している．多段パイプラインの構成で動作する場合には D-FF+MUX の上のパスを有効にする，一方で，パイプラインステージを統合し少段パイプラインの構成で動作する場合には下のパスを有効にすることでパイプライン段数を動的に変更している．

しかし，パイプラインステージ統合時には，統合によって組合せ回路が巨大化し，発生するグリッチが増加するという問題点がある．ここで，グリッチとは回路にあらわれる無駄な電気信号の変動のことであり，ゲート遅延，配線遅延のばらつきなどで生じる無駄な信号変化のことである．また，グリッチには一度発生すると次の回路に伝播され，後段の回路ではさらにグリッチが発生するという特徴がある．著者らの行った予備評価により，グリッチによる電力消費は回路の種類にも依存するが，おおよそ回路規模の 2 乗に比例することがわかっている．

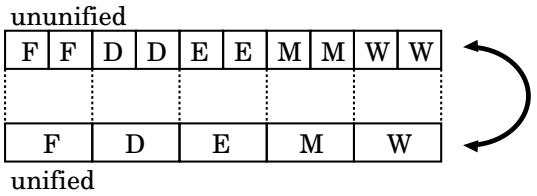

図 1 可変段数パイプライン

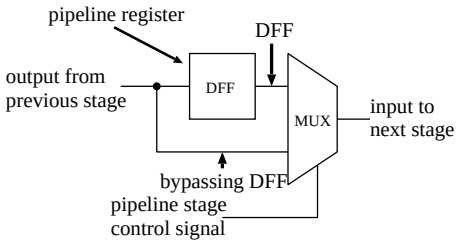

図 2 DFF+MUX

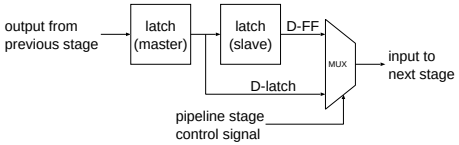

図 3 LDS-cell のブロック図

そこで，VSP では統合するパイプラインステージのパイプラインレジスタを全て DFF+MUX に置換するのではなく，一部を LDS-cell という特殊なセルに置換することでグリッチの伝播を緩和する．LDS-cell は図 3 のような構成になっており，HS モードでは DFF として動作させ，LE モードではラッチとして動作させることによりグリッチの発生を緩和している．このように VSP では，PSU で問題となっていたパイプラインステージ統合時のグリッチによる消費電力を LDS-cell を導入することにより 13%の電力削減に成功し

ている[11]．

3. モード切換コントローラ

PSU や VSP を効率的に使用するためには，実行中のプログラムに最適なパイプライン段数を予測し，動的にパイプライン段数を変更するパイプライン段数切換コントローラが必要となる．本節では，現在までに提案されているコントローラについて概括する．

3.1 Yao らの手法

パイプライン段数切換コントローラの 1 つとして，Yao らによる手法が提案されている[7]．このコントローラでは，フェーズと呼ばれる 10 万命令の切換区間でパイプライン段数の切換を行っている．一般にプログラム実行開始時点では，どのフェーズを何段のパイプライン構成で実行するのが最適であるのかは未知である．そこで Yao らは，あるフェーズを実行する場合，初回実行時には HS モード，2 回目実行時には LE モードで動作し，そのフェーズをそれぞれのモードで実行した時の電力遅延積を履歴表に登録する[⋆1]．そして，3 回目以降に該当フェーズを実行する場合，より電力遅延積が良いモードを履歴表を参照することで選択し，そのモードで動作する．

Yao らの手法では，このような予測を行うことで予測精度の高いパイプライン段数の切換を行っている．しかしながら，以下のような 4 つの問題点がある．

- 高い精度でフェーズを検出するためには 10 万命令程度プログラムを実行する必要がある．その結果，切換間隔が 10 万命令単位となるため細かな負荷の変動には追従できず，可変パイプライン段数プロセッサの性能は十分に発揮できない．
- それぞれのフェーズに対し，それぞれのモードで実行を行い，モードごとに比較を行った結果を次回からのモードに用いるため，性能の悪いモードで 1 度は実行しなければならない．
- 多数のレジスタからなる履歴表を持つためハードウェア規模は大きなものとなる．
- 履歴表に電力遅延積を記録するため，消費エネルギーを計測するための回路が必要となる．

そこで，Yao らの手法の問題点を解決するために，著者らは数百命令程度の間隔でパイプライン段数を切換える細粒度コントローラを提案している[9]．

⋆1 文献 7) では段数の違いによるモードを 3 状態持っているが，ここでは説明の簡略化のため HS モードと LE モードの 2 状態で説明する

3.2 細粒度切換コントローラ

3.2.1 細粒度切換に関する考察

細粒度切換コントローラの詳細説明に先立ち，まず細粒度に切換を行った場合の利点について述べる．パイプライン段数切換によって発生するオーバーヘッドは，以下の 2 つである．

(1) 周波数変更を行うためのオーバーヘッド．
(2) パイプラインレジスタが減少する場合 (HS モードから LE モードへの移行) に，パイプラインフラッシュに起因するオーバーヘッド．

パイプライン段数切換を行う場合，前者のオーバーヘッドは必ず発生するが，周波数変更にかかるオーバーヘッドはたかだか 2，3 サイクルであるため無視できる．その一方で，後者のオーバーヘッドは，通常数十サイクルから数百サイクル程度のオーバーヘッドとなるために細粒度でパイプライン段数を切換える場合には無視できない．しかしながら，詳細は第 3.2.3 項で述べるが，著者らはこのオーバーヘッドを隠蔽する手法を提案しており，パイプライン段数切換に起因するオーバーヘッドは非常に少ない．そのため，VSP では数百命令程度で細粒度にパイプライン段数切換を行ったとしても電力削減効果を得ることが可能である．

Yao らの手法のように 10 万命令単位で切換制御を行った場合，負荷の変動が激しいプログラムではパイプライン段数の切換が追従できず，不適切なパイプライン段数でプログラムを実行する区間が発生する．この様子を図 4 を使って示す．図 4 は縦軸がプロセッサ負荷，横軸が実行時間である．図 4 の A の区間では負荷の変動が粗いため，切換周期が粗粒度 (図 4 の Coarse-Grain) の場合にも負荷に応じたパイプライン段数の切換が可能である．しかしながら，図 4 の B，C の区間では切換周期よりも負荷の変動が細かいために，負荷の変動に対応した切換を行うことができない．そこで，著者らは細かい負荷変動にも対応できるよう細粒度なコントローラの提案を行っている．図 4 の Fine-Grain のようにパイプライン段数の切換を細粒度にすることにより，Yao らのコントローラでは対応できていなかった図 4 の B，C のような細かい負荷変動にも対応でき，より電力削減効果を得ることができる．

3.2.2 Out-of-Order 実行型プロセッサに適した細粒度切換コントローラ

著者らはこれまでに，履歴表を用いることなくプロセッサの負荷を予測し，予測結果に応じて適切なパイプライン段数に切換える手法を提案してきた．

文献 9) では，プロセッサシミュレータ SimpleScalar[12] を用い，Spec2000 ベンチマークの中から 10 本のプログラムを実行した結果，Out-of-Order 実行型プロセッサにおいて，メモリアクセスがプロセッサの負荷と相関性が高いことを明らかとした．一般に，ロード命令

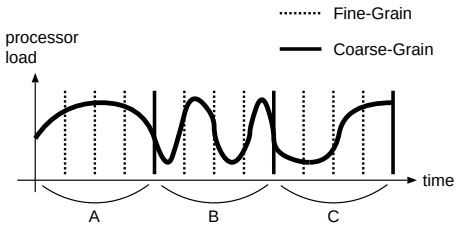

図 4 切換周期における違い

とストア命令の供給が過多の場合，キャッシュミスにより後続のデータ依存のある命令の発行ができなくなり，パイプラインにバブルが発生し実行効率の低下および電力の浪費に繋がる．そこで，著者らの提案しているコントローラでは，ロード/ストア命令の回数をカウントし，ロードストア命令の頻度が高い場合には，LE モードへと移行する．LE モードでは相対的にメモリアクセスレイテンシが短くなり，データ依存による待ちサイクルが短くなるために，メモリアクセスが多い場合にも実行効率があまり低下せず，消費電力を削減することが可能である．このように，メモリアクセスが頻発する場合に LE モードへと移行することにより，性能をあまり低下させることなく消費電力を削減している．

このモード切換コントローラのブロック図を図 5 に示す．Instruction Decoder は一般的なプロセッサのデコーダである．デコーダから現在のサイクルのロード/ストア命令の命令数が 32entry FIFO に渡される．32entryFIFO は 32 個の D-FF で構成される FIFO 構造になっており，最新 32 サイクル分のロード/ストア命令の命令数を保存できる．過去 32 サイクルのロード/ストア命令の合計数としきい値とを比較器 (cmp) で比較し，パイプライン段数の切換を行う．このように，著者らの提案しているコントローラは僅かな回路を追加するだけで実現可能ある．さらに，過去 32 サイクル分のデータから負荷を予測するため，細かい負荷変動にも追従できる．

3.2.3 モード切換によるオーバーヘッド削減手法

第 3.2.1 で記述した通り，パイプライン段数を切換える時には 2 つのオーバーヘッドが存在する．そのうち後者のパイプラインフラッシュに起因するオーバーヘッドは HS モードから LE モードへ切換える場合に発生する．HS モードから LE モードへ切換える場合，パイプラインステージの統合が行われるため，統合前のステージ，例えば図 6(A) の F，D，R に有効な命令が入っていた場合，このまま LE モードへ移行すると D，R ステージの命令は後続の F ステージの命令により破壊され，プロセッサは回復不能な状態になる．この問

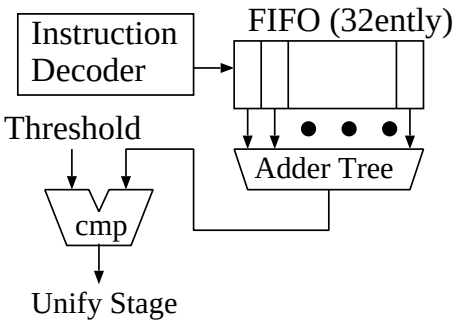

図 5 細粒度切換コントローラのブロック図

題を解決するために，HS モードから LE モードへ移行する場合にパイプラインレジスタを全てフラッシュし，命令をやり直すと数十から数百サイクルのオーバーヘッドが発生してしまい，パイプライン段数切換によるオーバーヘッドが無視できなくなる．特に，著者らが提案しているような細粒度切換手法ではより致命的な問題となる．

そこで，著者らは HS モードから LE モードへの切換を行うタイミングを分岐予測ミス発生時に限定することで，パイプライン段数切換のオーバーヘッドを隠蔽する手法を提案している．一般的なプログラムにおいて，分岐命令の出現頻度は 11%〜17% と言われており，分岐予測ミスは高度な分岐予測を行ったとしても 4%程度は発生する．このことから，数百命令に一度程度は分岐予測ミスが発生することになる．従って，パイプライン段数切換を分岐予測ミス時に限定しても細粒度なパイプライン段数切換を実現できるため，分岐予測ミスをパイプライン段数切換オーバヘッド削減のために活用している．

図 6 を用いてオーバーヘッド削減手法を説明する．図 6(A) に示すように，あるサイクルにおいて，高速モードの状態で命令がいくつか流れているとする（図 6 の網かけ部分は有効な命令を表す）．ここで，分岐予測ミスが発生した場合，図 6(B) に示すように投機的実行に失敗した命令（E2 内の命令），および当該分岐命令の後続命令はすべて破棄されるため，実行ステージ以前の F, D, R ステージの命令はすべて無効化される．従って，実行ステージ以前のステージは空になる．

ここで，図 6(C) に示すように,「マイグレーションモード」と呼ぶモードに移行する．マイグレーションモードとは，実行ステージ以降は HS モードのままで，実行ステージ以前はパイプラインステージを統合し，低周波数で動かすモードである．

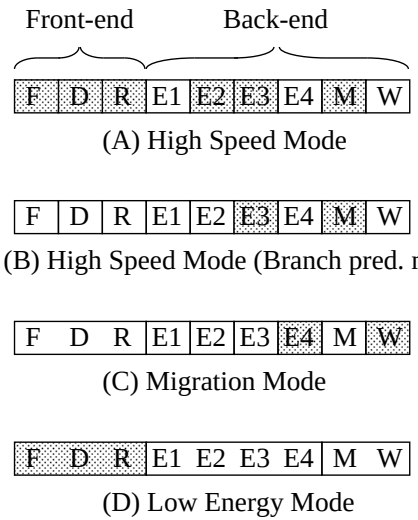

図 6 パイプライン段数切換によるオーバーヘッド削減手法

マイグレーションモード中は実行ステージ以降の命令の方が速く処理されるため，図 6(D) に示すように新しくフェッチされた命令が実行ステージに到着する頃には実行ステージ以降の命令はすべて実行され，実行ステージ以降のパイプラインレジスタ内が空になる．その時，LE モードに完全に移行する．このようにすることで，パイプライン段数切換のためにかかっていたパイプラインフラッシュによるオーバヘッドの削減を目指す．

実行ステージ以降の命令完了に時間が掛かり，新たにフェッチした命令が実行ステージの直前に到達した場合は新規の命令フェッチを停止し，LE モードに完全に移行した後，命令フェッチを再開する．しかし，シミュレーション実験を行った結果，マイグレーションモードで命令フェッチが停止する確率は 1%未満で，性能への影響はほとんどないと考えられる．

著者らはこれまでに，パイプライン段数切換オーバーヘッドを削減する手法と細粒度なパイプライン段数切換コントローラ (以降,「従来細粒度コントローラ」と呼ぶ) を SimpleScalar に実装し評価を行った結果，Yao らの手法と比較して電力遅延積を約 10%改善できることを明らかとしている．

これまでの著者らの研究では，高性能プロセッサへの適用を想定し，Out-of-Order 実行型プロセッサに適した切換手法の提案を行い，その有効性を示してきた．しかしながら，より低電力であることが要求される組み込み分野で広く使用されている In-Order 実行型プロセッサにおける従来細粒度コントローラの有効性は評価されていない．また，プロセッサシミュレータである SimpleScalar 上に実装し評価を行っているため，詳細な電力評価やマイグレーションモードの追加を含めた追加ハードウェア量に関する評価が行えていない．そこで本稿では，細粒度切換コントローラを In-Order 実行型プロセッサに実際に Verilog-HDL を用いてハードウェア実装し，その有効性を明らかとする．

4. In-Order 実行型プロセッサに適したコントローラの提案

4.1 従来細粒度コントローラの In-Order 実行型プロセッサにおける評価

In-Order 実行型プロセッサに適したコントローラの提案に先立って，本項では従来細粒度コントローラを In-Order 実行型プロセッサにそのまま適用した場合の評価結果とその問題点を示す．

MIPS R3000 命令互換の In-Order 実行型プロセッサにパイプライン段数切換オーバーヘッド削減手法と従来細粒度コントローラを Verilog HDL を用いて実装した．また，ROHM0.18μm CMOS プロセスで Synopsys Design Compiler と Jupiter XT を使用して論理合成およびクロックツリーの挿入を行ったネットリストを用いて，PrimeTime PX にて電力を測定し，電力遅延積で評価を行った．電力遅延積は，消費電力と実行時間の積をとったもので，低電力と高性能との両立を評価するために用いられる指標であり，数値が低いほど良い結果となる．プロセッサの詳細な構成を表 1 に示す．パイプライン段数切換のためのしきい値として，文献 9) で最適なしきい値を求めているが，今回の評価ではプロセッサ構成が大きく異なるため，いくつかのしきい値についてそれぞれ評価を行った．その結果，実行した 4 つのベンチマーク全てにおいて，常に LE モードで実行されるか，常に HS モードで実行されるようなしきい値の時に最も良い電力遅延積となり，細粒度な切換を行っても電力遅延積を改善することができなかった．

この原因を解明するために，HS モードと LE モードでそれぞれ 100 命令を 1 フェーズとして，1 フェーズごとの電力遅延積とロード/ストア命令の合計数との関係を調査した．図 7 はベンチマークで使用した quick sort プログラムの 45,000 命令から 55,000 命令の実行結果を解析したグラフである．グラフの横軸は 1 フェーズを 100 命令の実行とした時のフェーズ，縦軸は EDP of HS mode と EDP of LE mode に対しては 1 フェーズ中の電力遅延積，Sum of Load / Store に対しては 1 フェーズ中のロード/ストア命令の合計数となっている．電力遅延積は数値が低いほど良い指標であるため，図 7 中では，フェーズ 460 から 480 な

表 1 プロセッサ構成

issue width	1
pipeline	9 stage (HS mode) 3 stage (LE mode)
clock frequency	100 MHz (HS mode) 25 MHz (LE mode)
branch prediction	PHT 1K entry history width 4 の gshare
L1 data cache	2KB/32B line/1-way
L1 cache access latency	1 cycle (HS mode) 1 cycle (LE mode)
memory access latency	9 cycle (HS mode) 3 cycle (LE mode)

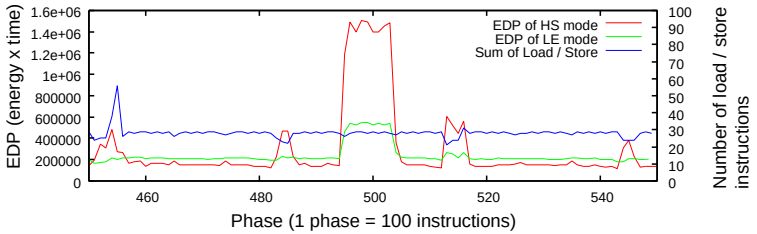

図 7 電力遅延積とロード/ストア命令の合計との関係

どは HS モードで，フェーズ 495 から 505 などは LE モードでそれぞれ実行すると良い結果が得られる．図 7 では数フェーズ単位で最適なモードが変化しており，細粒度な切換を行うことで電力遅延積を改善することが可能であることがわかる．しかしながら，グラフは省略するが，quick sort の図 7 以外のフェーズや他のベンチマークプログラムにおいても，ロード/ストア命令の合計数と電力遅延積との相関は低く，従来細粒度コントローラでは最適なモード切換が行えていないことが明らかとなった．

4.2 In-Order 実行型プロセッサに適した細粒度コントローラの提案

In-Order 実行型プロセッサに適した細粒度コントローラを設計するために，In-Order 実行プロセッサにおいて電力遅延積と相関の高いパラメータについて調査を行った．その結果，In-Order 実行型プロセッサでは，キャッシュミスと分岐予測ミスが電力遅延積に大きく影響を与えることがわかった．図 8 に電力遅延積とキャッシュミスおよび分岐予測ミスとの関係を示す．LE モードにおいては分岐予測ミスが発生しないため，図 8 中の分岐予測ミス

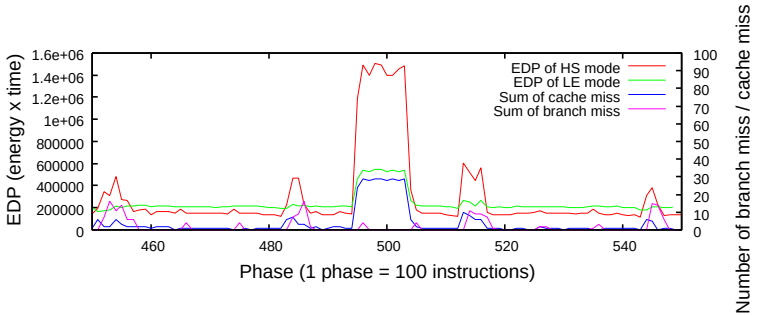

図 8 電力遅延積とキャッシュミスおよび分岐予測ミスとの関係

の合計は HS モードで実行した場合の数値となっている．図 8 より，キャッシュミスと分岐予測ミスが多く発生した場合，該当フェーズの HS モードの電力遅延積が大きく悪化するということがわかる．それに対して，LE モードではキャッシュミスが多く発生した場合にのみ電力遅延積が悪化するが，悪化の割合は HS モードより低い．これは，LE モードでは相対的にメモリアクセスレイテンシが小さくなるためである．また，今回の解析では，HS モードで分岐予測ミスが多発する場合に LE モードでは電力遅延積が減少する傾向があることがわかった．分岐予測ミスが多発するフェーズでは分岐命令の数が比較的多く，分岐命令の実行は他の命令の実行と比較して動作するロジックが少ない．そのため，分岐予測ミスが発生しない LE モードでは実行サイクルが増加せず消費電力が下がるので電力遅延積が減少する．従って，キャッシュミスおよび分岐予測ミスが多発するフェーズでは LE モードがより最適なモードであるといえる．

以上のことから，キャッシュミスおよび分岐予測ミス回数を監視し，その回数が一定数以上であれば LE モードへと移行すれば最適なモードへの移行が可能である．In-Order 実行型プロセッサにおいては，キャッシュミスおよび分岐予測ミスがそのまま IPC (Instruction Per Cycle) に影響するため，IPC を監視することで 2 つの指標を同時に監視することが可能である．そこで，過去 32 サイクルに完了した命令の合計数をしきい値と比較してパイプライン段数を変更する手法を提案する．

しかしながら，この手法の問題点として，現在の VSP では LE モードでは分岐予測ミスが発生しないため IPC がキャッシュミスにのみ影響され，その結果，LE モードから HS モードへの移行が適切に行われない場合がある，という点が挙げられる．また，現在の VSP

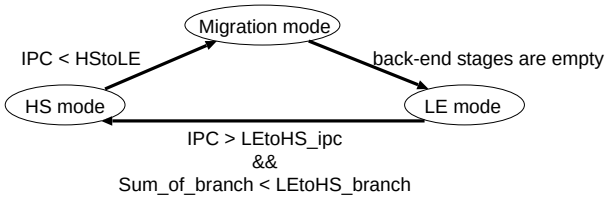

図 9 提案するモード切換手法

では消費電力を削減するために LE モードでは分岐予測器を停止しているため，LE モード時に分岐予測ミス回数を監視することはできない．そこで，分岐予測ミスが多く発生するフェーズでは比較的分岐命令の数が多いという傾向を利用し，LE モードから HS モードへの移行には過去 32 サイクルに完了した命令の合計数と分岐命令の合計数をしきい値と比較してパイプライン段数の変更を行う．

図 9 は提案手法のステートマシンを示している．図 9 のように，HS mode で動作しているときに IPC がしきい値 HStoLE より小さくなれば Migration mode へ移行する．逆に，LE mode で動作しているときには，IPC が LEtoHS_ipc より大きく，かつ，Sum of branch が LEtoHS_branch となった場合に HS モードへと移行を行う．従来細粒度コントローラと比較して，提案手法では 32 エントリの FIFO とその総和を求める回路が余分に必要となるが，これはプロセッサ全体と比較すると十分に小規模であり，大きなオーバーヘッドにはならない．

5. 評　　価

提案するコントローラを 4.1 項と同様の方法で設計し，評価を行った．モード切換のためのしきい値は，これまでの評価から，IPC が 0.6 以下の場合には LE モードの方が電力遅延積が良くなる確率が高いため，HS モードから LE モードへの切換は，過去 32 サイクルに完了した命令の合計が 20 より少ない場合とした．LE モードから HS モードへの切換は，過去 32 サイクルに完了した命令の合計が 32 で，分岐命令の合計が 6 より小さい時とした．これは，いくつかのしきい値で実験を行い，最も結果がよかった組合せである．

ベンチマークプログラムは MiBench[13] にて配布されているものの中から，整数の 2 乗根を求める int sqrt，long 型の変数中で 1 のビット数を数える bit count，文字列の検索をする string search，文字列をクイックソートする quick sort を用いた．quick sort のアルゴリズムには Newlib[14] のものを利用した．また，ベンチマークで使用するデータは PrimeTime PX の計算時間を考慮し，10 万命令程度で終了するように調整した．

図 10 にその評価結果を示す．図 10 (1) の縦軸は HS モードでの結果を 1 として正規化した電力遅延積，横軸はベンチマークプログラムである．図 10 (1) から，提案手法は全てのベンチマークプログラムで HS モードより良い結果となっていることがわかる．特に quick sort ではより電力遅延積が低い HS モードよりも 7%電力遅延積を改善できている．図 10(2) は，quick sort の実行時間と消費電力をそれぞれ HS モードの値で正規化して表したものである．提案コントローラでは，HS モードと比較して，実行時間が 6%増加するが，12%消費電力を削減できるため，電力遅延積として 7%の改善が見られる．図 11 に quick sort プログラムを解析した結果の一部を示す．図 11 より，提案手法がより最適なモードに移行してプログラムを実行していることがわかる．しかしながら，著者らが行った試算では，100 命令毎に最適なモード切換を行った場合，quick sort では 20%近く電力遅延積が改善可能である，という結果が出ているため，まだ改善の余地が残っていると考えられる．

ベンチマークの bit count と string search については，常に LE モードで実行した場合に ED 積が最良になるプログラムであるため，細粒度に切換えても LE モード時より電力遅延積が改善できない．int sqrt も bit count や string search と同様に常に LE モードで実行した場合に電力遅延積が最良になるプログラムであるが，提案手法では HS モードに近い電力遅延積となっている．この原因を調査したところ，int sqrt は他のベンチマークと異なり，データ依存により IPC が低下していることが判明した．本稿で提案したコントローラでは，LE モードから HS モードへ切換える際，データ依存を全く考慮していないため，モード変更が過剰に発生してしまい，電力遅延積が悪化したと考えられる．LE モード時に HS モードでのデータ依存を考慮することは困難であるので，この問題点を解決するために，1) 動的に切換しきい値を学習させる． 2) 一度 LE モードに移行したら一定時間は切換を行わない．などの工夫が必要であると考えられる．

最後に，提案コントローラのハードウェア規模について述べる．本稿で提案したコントローラを実装するのに必要なハードウェアは，プロセッサ全体の約 0.9%と小規模で，電力はプロセッサ全体の 0.5%程度であった．

6. おわりに

本研究では，In-Order 実行を行う可変パイプライン段数プロセッサに適した細粒度なパイプライン段数切換コントローラを提案した．提案したコントローラはモードを固定した場合と比較して最大で電力遅延積を 7%改善できることが明らかとなった．

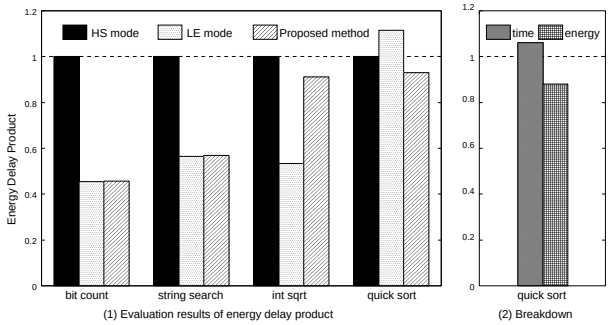

図 10 電力遅延積評価

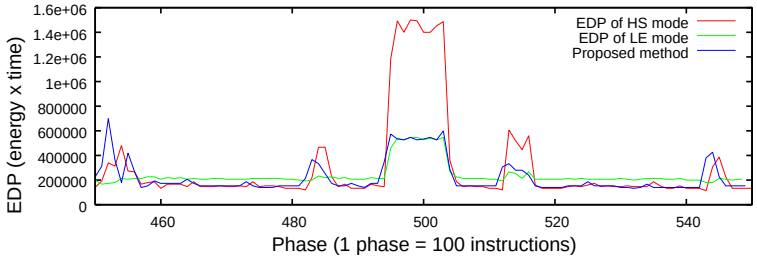

図 11 quick sort の解析結果

今後の研究としては，今回提案したコントローラではデータ依存により IPC が低下している場合に対応できないため，データ依存による IPC の低下にも対応可能な手法を提案する必要がある．また，今回用いたベンチマークは 10 万命令程度と小規模なものなので，より大規模なベンチマークを用いた評価を行う必要がある．

謝辞 本研究の一部は科研費補助金（19700042）の援助を受けている．また，LSI 設計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，ローム株式会社の協力で行われたものである．

参 考 文 献

1) J. Pouwelse, K. Langendoen, H. Sips, “Dynamic Voltage Scaling on a Low-Power Microprocessor“, Proc. of The 7th ACM Int. Conf. on Mobile Computing and Networking (Mobicom), pp .251-259, July 2001.
2) Atsuki Inoue, “Design Constraint of Fine Grain Supply Voltage Control LSI”, Proc. of the 16 th Asia and South Pacific Design Automation Conference (ASP-DAC 2011), pp. 760-765, January, 2011.
3) Koppanalil, J., Ramrakhyani, P., Desai, S.,Vaidyanathan, A. and Rotenberg, E., “A Case for Dynamic Pipeline Scaling”, Proc. of Int. Conf. on Compilers, Architecture, and Synthesis for Embedded Systems 2002, pp. 1-8, 2002.
4) Efthymiou, A. and Garside, J. D., “Adaptive Pipeline Depth Control for Processor Power-Management”, Proc. of Int. Conf. on Computer Design 2002, pp. 454-457, 2002.
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 SACSI2003, pp. 283-290，2003 年 5 月
6) 嶋田 創，安藤 秀樹，島田 俊夫，“パイプラインステージ統合と DVS の併用による消費電力の削減（省電力方式）”，情報処理学会論文誌（コンピューティングシステム），Vol. 48，No. SIG3(ACS17)，pp. 75-87，2007.
7) Jun Yao, Shinobu Miwa, Hajime Shimada and Shinji Tomita, “A Dynamic Control Mechanism for Pipeline Stage Unification by Identifying Program Phases”, IEICE Transactions on Information and Systems, Vol. E91-D, pp. 1010-1022, 2009.
8) 市川 裕二，佐々木 敬泰，弘中 哲夫，谷川 一哉，北村 俊明，近藤 利夫，“可変パイプラインを用いた低消費エネルギープロセッサの設計と評価 “, 情報処理学会論文誌. (コンピューティングシステム) Vol.47, pp. 231-242，2006 年 5 月
9) Takahiro Sasaki, Kazumasa Nomura, Tomoyuki Nakabayashi, Kazuhiko Ohno and Toshio Kondo, “Fine Grain Controller for Variable Stages Pipeline Processor”, Proc. of The 25th Int. Technical Conf. on Circuits/Systems, Computers and Communications, pp. 748-751, July 2008.
10) 中林 智之，佐々木 敬泰，大野 和彦，近藤 利夫，“クロック系消費電力に着目した可変段数パイプラインプロセッサの低電力化 “，電子情報通信学会論文誌，Vol. J94-D, No. 4，pp. 646-656，2011 年 4 月.
11) Tomoyuki Nakabayashi, Takahiro Sasaki, Kazuhiko Ohno and Toshio Kondo, “Design and Evaluation of Variable Stages Pipeline Processor Chip”, Proc. of Int. Symposium on Information and Automation, November 2010, (In press).
12) Todd Austin and Eric Larson and Dan Ernst, “SimpleScalar: An Infrastructure for Computer System Modeling”, Computer, Vol. 35, No. 2, pp. 59-67, 2002.
13) MiBench. http://www.eecs.umich.edu/mibench/
14) Newlib. http://sourceware.org/newlib/