修士論文

題目

可変パイプライン段数プロセッサのための細粒度な段数切換コントローラ

指導教員

佐々木敬泰 助教

平成21年度年度

三重大学大学院 工学研究科 情報工学専攻
計算機アーキテクチャ研究室

野村和正 (408M517)

内容梗概

近年，モバイルプロセッサでは，性能向上に伴う消費エネルギーの増加が問題となってきている．消費エネルギーが多いと，発熱，故障のトラブルや，ノートパソコンなどのバッテリー持続時間の減少の原因となってしまうためである．そのため，現在様々な高性能かつ低消費エネルギーを実現する手法が提案されている．

その手法の一つとして可変パイプライン段数プロセッサ (VSP; Variable Stages Pipeline) や Pipeline Stage Unification(PSU) が提案されている．可変パイプライン段数プロセッサはパイプライン段数を動的に変更するアーキテクチャである．可変パイプライン段数プロセッサは負荷が高い場合にはパイプライン段数を増やし高周波数で動作させ，負荷が低い場合にはパイプライン段数を減らし低周波数で動作させることにより，高性能かつ低消費エネルギーを実現している．また，可変パイプライン段数プロセッサは現在主流である消費電力削減手法 Dynamic Voltage Scaling (DVS) よりも消費エネルギーを削減可能であることが示されている．

VSP や PSU を実現するためには，実行時に負荷の変動を監視することにより最適なパイプライン段数を予測し，制御を行うコントローラが必要となる．しかし，従来提案されているコントローラは切換えの粒度が粗く，負荷の変動に対して常に最適なパイプライン段数で動作しているとはいえない．そこで，本論文ではメモリアクセス数を用いた細粒度に予測を行うパイプライン段数切換コントローラを提案し，評価を行った．結果，従来コントローラと比較し，電力遅延積において約 1 割の改善が得られ，ハードウェア量も少なく抑えることができた．

Abstract

Recently, specially in mobile processors, the increase of energy consumption by enhancing performance becomes serial problem. it causes decrease of the battery continuance, fever trouble or trouble of note PC when energy consumption is increased. So achievement of both low energy and high-performance is demanded. Therefore various low energy and high-performance techniques have been proposed.

Variable Stages Pipeline (VSP) and Pipeline Stage Unification (PSU) are proposed as one of low energy and high-performance techniques. VSP changes the number of the pipeline stages dynamically and frequency. When load of the processor is high, VSP are increased the number of the pipeline stages and operated at high frequency. When load of the processor is low, VSP are reduced the number of the pipeline stages and operated at low frequency, VSP realizes high efficiency and low energy consumed in this way. And VSP can reduce the energy consumption more than Dynamic Voltage Scaling (DVS) of the one of the famous energy consumption reduction technique.

VSP and PSU need controller which predict the suitable number of pipeline stage by watching a change of the load of the processor. To predict the suitable number of pipeline stage, pipeline controller is proposed by Yao. However, the grain of the pipeline stage change in this conventional controller is coarse. So it cannot be said that the conventional controller always operates with the number of the most suitable pipeline stages. Here, in order to achieve better performance, I proposed fine grain pipeline controller based on Dynamic Memory AccessAnalyzing. my evaluation results, proposed controller can achieve about 10% better performance than conventional processor in Energy-Delay Product. Proposed controller can hold down the quantity of hardware.

目 次

図目次

表目次

1 背景

近年，プロセッサの性能向上に伴う消費エネルギーの増加により，低消費エネルギーと高性能の両立が要求されている．消費エネルギーの増加は発熱の増加やバッテリー持続時間の減少につながるためである．

そのため，プロセッサの負荷に応じてプロセッサの動作を切換えることで，高性能かつ低消費エネルギーを実現する手法が検討されている．その中の手法の一つに DVS (Dynamic Voltage Scaling) [1] [2] と呼ばれる手法が提案されている．この手法は，電源電圧と周波数が可変なアーキテクチャであり，プロセッサの負荷が高い時には電圧と周波数を高くし，負荷が低いときには性能があまり必要でないため電圧と周波数を低くすることにより，高性能かつ低消費エネルギーを実現する手法である．DVS では負荷の監視に OS を利用し，待ちプロセスの数などをもとに負荷を算出している．この手法は，現在の Intel 社や AMD 社などのプロセッサに導入されている．しかし，この手法は将来的に消費エネルギー削減効率の低下が予想されている．なぜなら近年 CMOS の電源電圧は低下の一途をたどっており電源電圧の下げ幅は小さくなっているためである．

そこで，電源電圧に依存しない低消費電力手法として，可変パイプライン段数プロセッサが提案されている．可変パイプライン段数プロセッサとは負荷に応じてパイプライン段数と周波数を動的に切換えることで低消費電力化を実現する手法である．パイプラインレジスタへのクロック供給のエネルギーは膨大であるため，可変パイプライン段数プロセッサでは，性能があまり必要でない場合にはパイプライン段数を少なくし，クロック供給を行うパイプラインレジスタの個数を減らすことで低消費エネルギーを実現している．可変パイプライン段数プロセッサは電源電圧に依存しないため，将来的に効果が期待される．この可変パイプライン段数プロセッサの一手法として VSP (Variable StagePipeline) [3] [4] [5] が提案されている．

可変パイプライン段数プロセッサである VSP を実用化するためには最適なパイプライン段数を予測し，パイプライン段数を動的に切換えるコントローラが必要となる．既に最適なパイプライン段数を予測するコントローラが Yao らによりが提案されている [9]．しかし，このコントローラはパイプライン段数の切換間隔が長いため細かい負荷の変動には対応できず，VSP の性能を十分に発揮できない．また，ハードウェアコストも大規模なものとなる．

そこで，本論文では細粒度に最適なパイプライン段数を予測するパイ

プライン段数切換コントローラを提案する．今回行った評価の結果，メインメモリへのアクセス数とプロセッサの負荷は密接に関係していることがわかった．そのため，提案コントローラでは負荷を測定するためにメインメモリアクセス頻度に着目し，動的にパイプライン段数を変更するよう設計した．しかし，細粒度な段数切換えを行った場合，段数切換えにかかるオーバーヘッドを無視できない．そこで，パイプライン段数変更のオーバヘッド削減手法を同時に提案し，評価によりその有効性を明らかにする．

提案コントローラをVSPに組み込み，評価を行った結果，従来コントローラと比較して電力遅延積において約1割の改善が得られた．また，ハードウェア設計を行い評価した結果，一般的なプロセッサと比較すると提案コントローラのハードウェアコストや消費電力は無視できるほど小さいという結果が得られた．

2　可変パイプライン段数プロセッサ

2.1　パイプラインプロセッサに関する概括

プロセッサ内では，命令の読み込み (Fetch)，解釈 (Decode)，レジスタ読み込み (Register Read)，実行 (Execution)，結果の書き込み (Write Back) などの複数の段階的な処理を行うことにより 1 つの命令が処理される．パイプラインプロセッサの動作を図 2.1 に示す．図 2.1 では Fetch を F，Decode を D，Register Read を R，Execution を E，Write Back を WB としている．図 2.1 に示すように，それぞれの段階においての処理は異なった回路で処理されるため，1 クロックで段階ごとに並列に動作させることができる．このように段階的に分けて並列に命令を動作させるプロセッサをパイプラインプロセッサと呼ぶ．パイプラインプロセッサではパイプライン段数を分けることにより，1 クロックで動作させるべき回路は小さくなるため高周波数にすることができ，単位時間当たりに実行できる命令が増えることから高性能を実現している．しかし，パイプライン段数が増えれば，ステージ間で命令の処理内容を記憶しておくためのレジスタが必要となるため，消費電力は増えるという欠点がある．そのため，一般的な商用プロセッサを高性能なプロセッサにするため，パイプライン段数をできるだけ増やし，高周波数で動作させたいが，単純にパイプライン段数を増やすと消費電力が大きくなるために，現在では約 10 段程度のパイプラインプロセッサが採用されている．

2.2　可変パイプライン段数プロセッサの概要

可変パイプライン段数プロセッサとは，パイプライン段数と周波数を負荷に応じて切換えることにより，負荷が高い場合には高性能を発揮させ，負荷が低い場合には性能を低下させる代わりに消費電力を抑えるという手法である．可変パイプライン段数プロセッサには，本研究室で提案している VSP と，パイプラインステージ統合 (PSU:Pipeline Stage Unification) [6] [7] [9]，DPS (Dynamic Pipeline Scaling) [11] [12] などの方式が提案されている．また，これらの可変パイプライン段数プロセッサはパイプライン段数に対して以下のような特徴を備えている．

- 多段パイプライン段数時，性能や動作はパイプライン段数が可変ではないプロセッサと同等である．

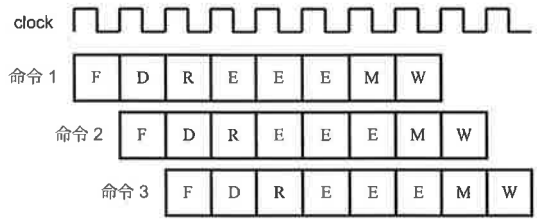

図 2.1: パイプラインプロセッサの動作

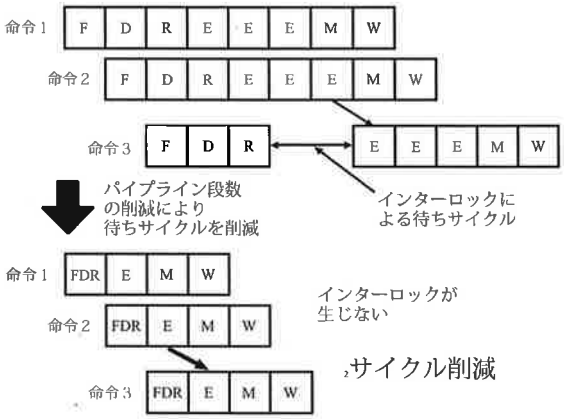

図 2.2: データ依存待ちサイクルの低減

- 少段パイプライン段数時，パイプラインレジスタへのクロック供給の必要がなくなるため，クロックドライバの消費エネルギー低減に繋がる
- 少段パイプライン段数時，データ依存による待ちサイクルの低減，分岐予測ミスペナルティ低減に繋がる（図 2.2，図 2.3）．図 2.2，図 2.3 ともに水平方向が時間軸である．

図 2.2 において，多段パイプライン段数時の状態で命令を実行中に命令 2 と命令 3 に依存関係がある場合には，命令 3 の実行を命令 2 の処理結果が得られるまで待つ必要がある．しかし，少段パイプライン段数時の状態では命令に依存関係がある場合において，実行に必要となるクロック数が 1 クロックであるため待ち時間は発生しない．図 2.3 においても，少

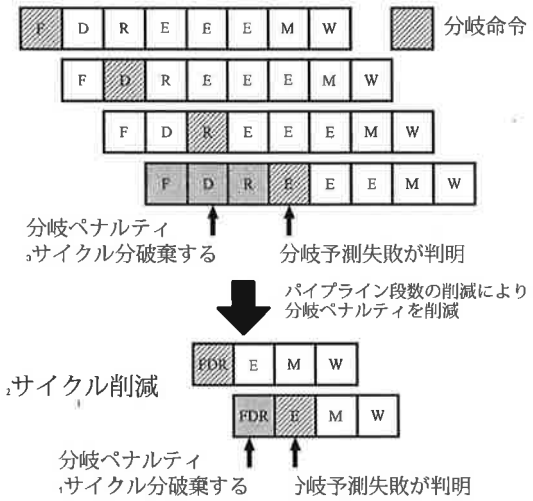

図 2.3: 分岐予測ミスペナルティ低減

段パイプライン段数時ではパイプライン段数が少ないために分岐予測ペナルティ削減ができる．

VSP などの可変パイプライン段数プロセッサでは，パイプライン段数を 2 種類以上へ切換可能であるが，今後，説明を簡潔にするために，現在 VSP で用意されている高速モードと低消費電力モードの 2 通りのパイプライン段数だけを用いて説明を行う．高速モードと低消費電力モードは図 2.4 の構成となっている．高速モードはプロセッサの性能を最大限に活かすために，パイプライン段数が多く，高周波数のモードであり，本論文ではパイプライン段数は 9 段とする．低消費電力モードは消費エネルギーを抑えるために，パイプライン段数が少なく，低周波数のモードであり，本論文ではパイプライン段数は 3 段とする．

モード別のアーキテクチャの様子を図 2.5，図 2.6 を用いて述べる．ここでは，高速モードにおけるパイプラインの中段である四段分だけを取り出して説明を行う．図 2.5，図 2.6 を比較すると，高速モードで存在していたパイプラインレジスタ B，D を，低消費電力モードでは使用しないため，その分レジスタへの電力供給を減らすことができ，低消費電力となる．また，高速モードで存在していた論理回路 A と B が，パイプラインレジスタを低消費電力モードでは統合されている．ここで周波数を下げてやると，この統合された論理回路が 1 クロックで動作するようになり，データ依存待ちサイクル時間の低減と分岐予測ミスペナルティの

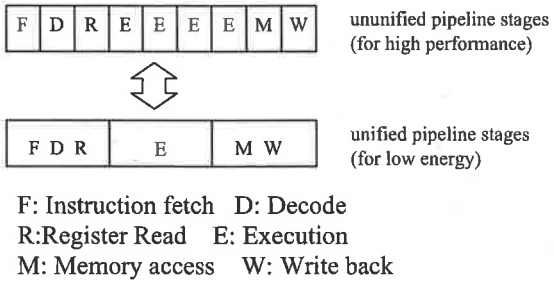

図 2.4: モード別パイプライン段数

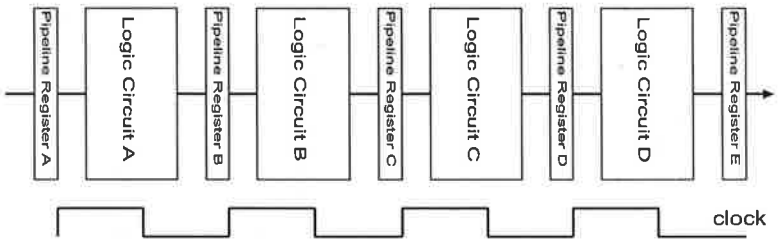

図 2.5: 高速モード

低減が実現される.

2.3 パイプライン統合手法

可変パイプライン段数プロセッサでは，パイプライン段数を変更できるようにするため，本来は D-FF だけで構成されているパイプラインレジスタにマルチプレクサを追加し，データを次の回路へ流すのか，D-FF で保存するのかを選択するようにしている．図 2.7 はパイプラインレジスタを使用するかどうかを選択する回路である．マルチプレクサ MUX に制御信号を与えることで選択を行うことができる．高速モードでは D-FF を使用し，パイプライン段数を増やし，周波数を上げることで高性能を実現する．低消費電力モードでは D-FF は使用せず，周波数を下げることにより低消費電力を実現できる．また，図 2.7 の回路は，図 2.5 に示したパイプラインレジスタ B，D の部分に挿入され，高速モードでは D-FF が選択され，低消費電力モードでは D-FF 使用しないよう選択される．ま

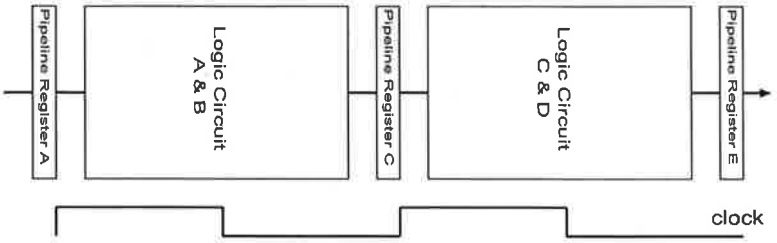

図 2.6: 低消費電力モード

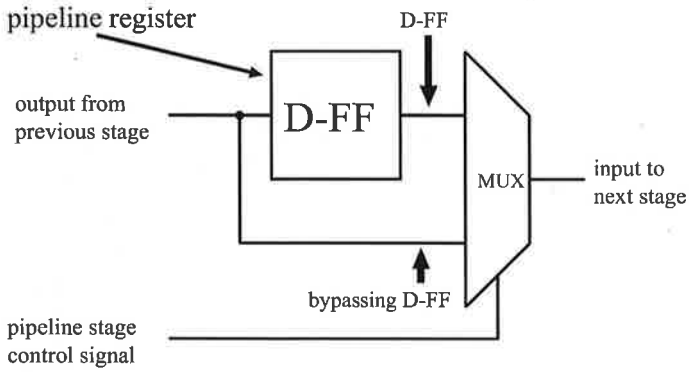

図 2.7: D-FF+MUX

た，他の部分のパイプラインレジスタでは必ず D-FF を選択することとなるため，図 2.7 の回路は挿入されない．

2.4 VSP について

可変パイプライン段数プロセッサの実現方式の一つとして VSP (Variable Stage Pipeline) [3] が存在する．これまでに述べた可変パイプライン段数プロセッサの特徴を VSP はすべて含んでいるが，VSP ではさらに低消費電力化を行うために，LDS-Cell という回路を加えている．この LDS-Cell により，グリッチという無駄な電力の発生を抑えることができる．LDS-Cell については第 2.4.1 節で説明を行う．

VSP プロセッサの構成図を図 2.8 に示す．また，現在 VSP プロセッサ

はMIPS R3000と命令互換性を持ち，高速(High Speed)モードと低消費電力(Low Energy)モードの2つの状態を持つ．これら2つのモードの特徴を以下に示す．

- 高速モード
  - 多段パイプラインであり，LDS-cellはパイプラインレジスタとして動作する．
  - デコードステージにgshare分岐予測ユニットを搭載する．
  - レジスタ間接をのぞく無条件分岐は分岐予測ユニットにおいて100%の分岐予測が可能である．
  - インターロックと演算結果のフォワーディング機構を搭載している．
- 低消費電力モード
  - 少段パイプラインであり，LDS-cellはグリッチの緩和を行うD-ラッチとして動作する．
  - 遅延分岐，遅延ロード，フォワーディングによって分岐ペナルティやデータ依存によるインターロックが発生しない．
  - 分岐予測ユニットは使用しないので停止する．
  - 分岐予測ユニットやバイパスされて使用しなくなったパイプラインレジスタのクロックを止めることでパイプラインレジスタで消費されるエネルギーを削減することが出来る．

2.4.1 要素技術

VSPの重要な技術として，LDS-Cell (図2.9)と呼ばれるものが存在する．VSPではLDS-Cellを使用することにより，無駄な消費電力となるグリッチを緩和することができ，さらなる低消費電力化を図っている．ここでいうグリッチとは，電子回路にあらわれる無駄な電気信号の変動のことであり，パルス周期の突然の変化や，ゲート遅延・配線遅延のばらつきなどで生じてしまう．グリッチは一度発生すると次の回路へ伝播され，後段の電子回路ではさらにグリッチが発生してしまう．そこで，LDS-Cellを挟むことにより，グリッチの緩和を行う．

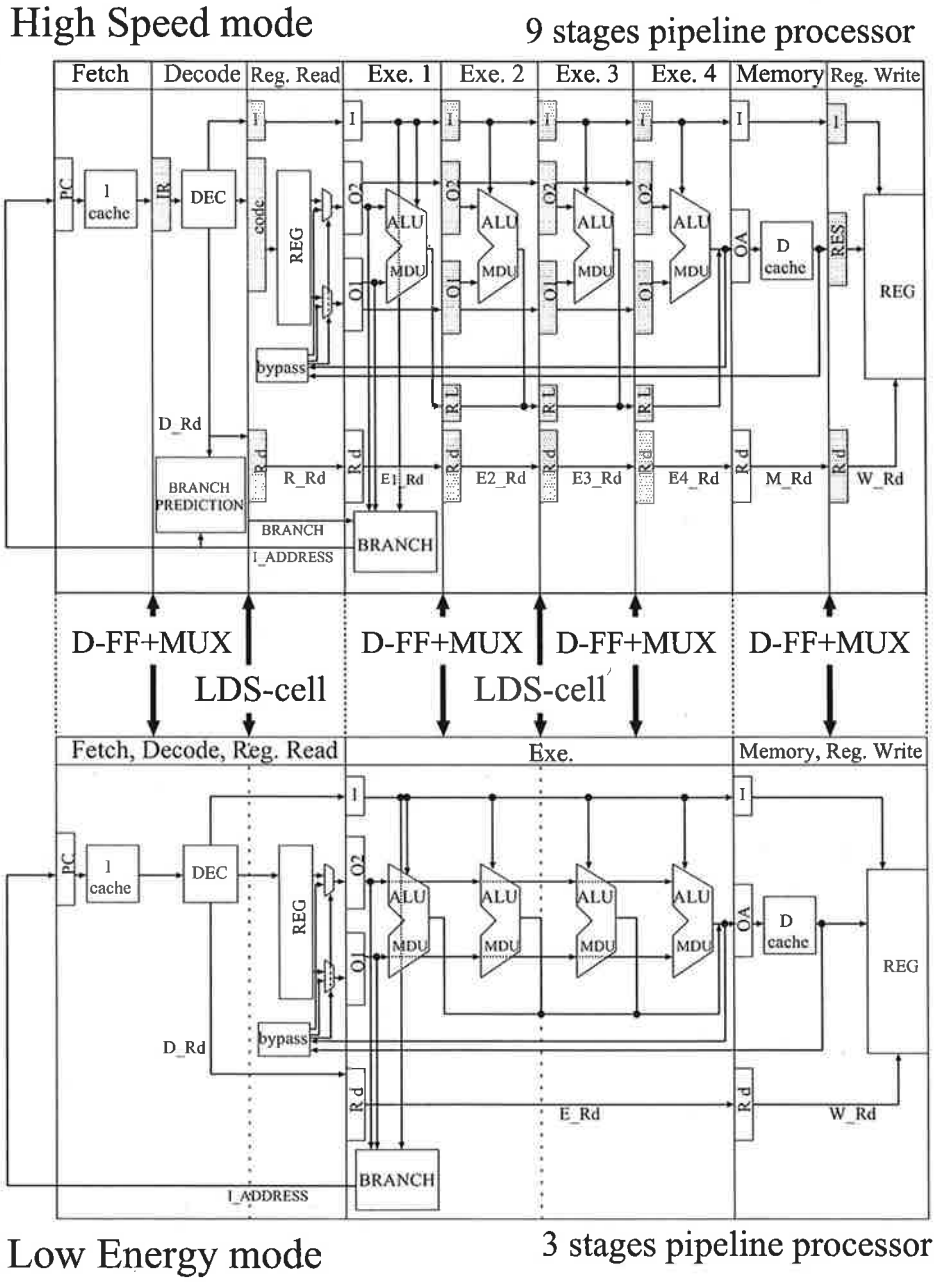

図 2.8: VSP プロセッサの構成図

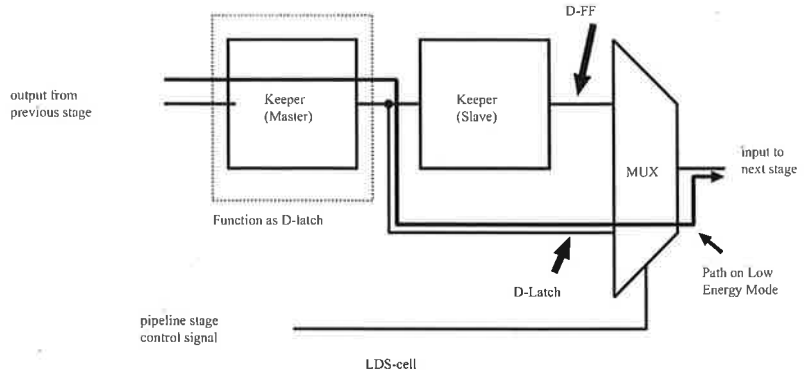

図 2.9: LDS-Cell

VSP において，高速モード時ではパイプラインレジスタとして使用しているが，低消費電力モード時ではラッチとして機能させている部分がある．組み合わせ回路の中段付近にラッチを挟むことで，図 2.10 に示すようにグリッチを緩和することができる．これは，ラッチはクロックが下がるまでは後段の組み合わせ回路に電気信号を伝達させない性質があるためである．具体的には，図 2.9 のようなマスター・スレーブ型 D-FF において，高速モード時には 2 つのキーパを直列につなぐことで通常の D-FF の機能を実現する．一方，低消費電力モード時には，スレーブ側のキーパをバイパスさせることで D-latch として機能させることで，グリッチの伝播を緩和する．

2.4.2 VSP の性能

VSP の関連研究である DVS と PSU との比較が文献 [3] により検証された．高速モード時の実行時間，消費電力の検証結果を図 2.11，図 2.12 に示し，低消費電力モード時の実行時間，消費電力の検証結果を図 2.13，図 2.14 に示す．

文献 [3] で示されたこれらの結果より，高速モード時において VSP は電力は DVS より少し大きくなっているが，実行時間は同じである．低消費電力モード時において VSP は DVS よりも約半分ほど実行時間と電力を削減できることがわかっている．また，今後 DVS の効力は減少していくと思われるため，VSP の有効性はさらに高まると予想される．

また，可変パイプライン段数プロセッサに関する研究として VSP 以外

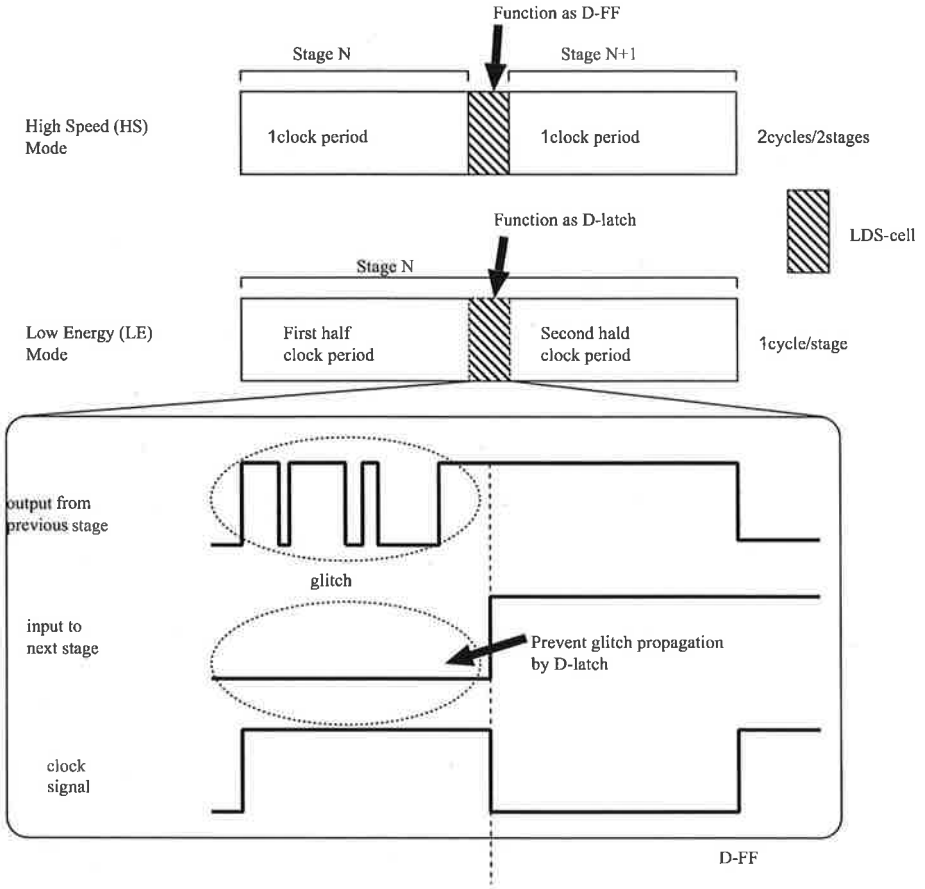

図 2.10: グリッチ緩和

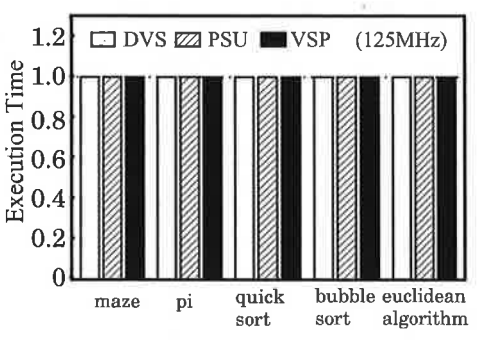

図 2.11: 高速モード：実行時間

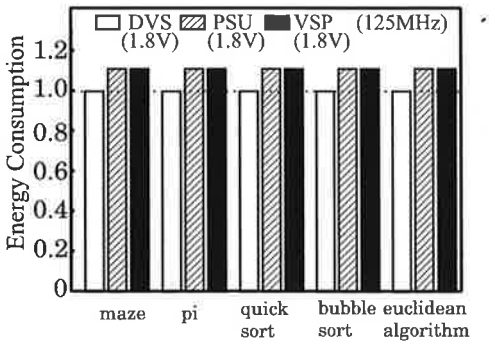

図 2.12: 高速モード：電力

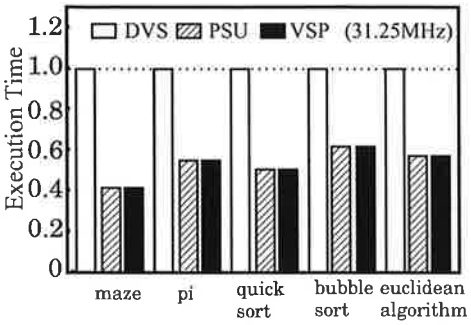

図 2.13: 低消費電力モード：実行時間

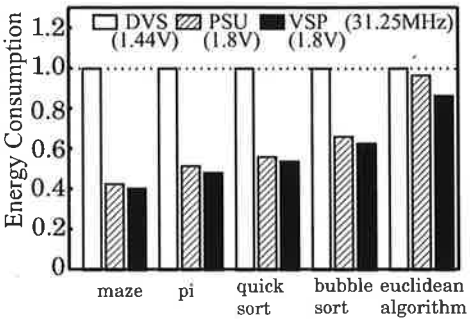

図 2.14: 低消費電力モード：電力

にも PSU が存在する．VSP は現在，２つのモード内でしかパイプライン段数が可変でしかないが，PSU ではパイプライン段数をある程度自由に変更可能である．しかし，PSU ではグリッチの緩和を行っていないため，VSP の方が PSU より 3%程度消費電力が低い．

2.5 パイプライン段数切換コントローラ

これまで VSP ではモードを固定した上で性能評価を行い，その有効性が示されてきた．しかし，実際には特定のモードで固定して利用する可能性は低いため，動的に負荷を検出し，適宜最適なモードに切換えるためのコントローラが必要となる．

DVS において電源電圧と周波数の切換制御を行うコントローラが実用化されている．DVS コントローラでは，OS が負荷を監視し，負荷に応じて電源電圧と周波数を変更している．しかし，DVS コントローラと同じような手法で可変パイプライン段数プロセッサのコントローラを設計してしまうのでは適切ではない．なぜなら，電源電圧変更には多くの時間を消費してしまうため，DVS コントローラでは切換頻度は細かく設計されてなく，切換頻度を細かくできないからである．しかし，可変パイプライン段数プロセッサは周波数を変更するための数サイクルでモードを切換えることができるため，DVS よりも細粒度なコントローラが適している可能性がある．

そこで，可変パイプライン段数プロセッサのためのコントローラとして，Yao らによりコントローラが既に提案されている [9]（以下,「従来コントローラ」と呼ぶ）．従来コントローラは DVS コントローラと比較し，

細粒度にパイプライン段数を切換えることができる．そこで，次節では Yao らによるコントローラについて述べた上でその問題点を指摘する．

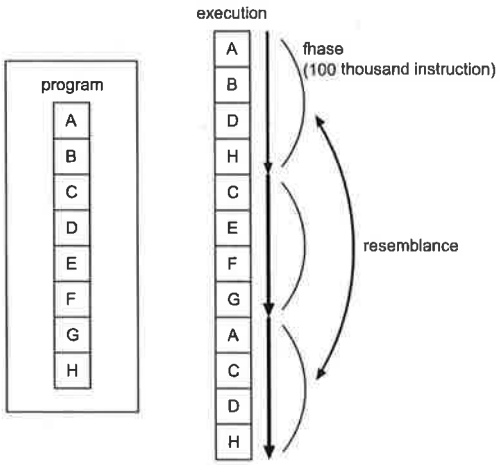

図 3.15: signature

3 従来コントローラと問題点

本節では従来コントローラ [9] について概括した上で，その問題点を指摘する．このコントローラは 10 万命令単位（以下,「フェーズ」と呼ぶ）でパイプライン段数の切換えを行っている．また，このコントローラは 1 フェーズで実行されたプログラムの位置を示す signature と，その signature の値に対して最適なパイプライン段数を記憶しておく履歴表からなる．

3.1 フェーズ

一般にプログラムは数万命令程度で挙動を取得すると，似たような挙動を行うフェーズが現れるという特徴があることが文献 [10] で述べられている．これは，プログラムの同じ部分をアクセスする場合が多々存在し，アクセスされた部分が同じであれば挙動が似ているためである．ここで，図 3.15 を用いて詳しく説明を行う．例えば，プログラムを区間 A ～ H に分割ができているものとする．この時に，図 3.15 に示すような実行が行われた場合，1 フェーズ目の実行と 3 フェーズ目の実行はほぼ同じプログラムの部分にアクセスを行っている．このようにアクセス部分が類似しているフェーズは挙動も類似することになる．

従来コントローラでは，異なるフェーズ間でも実行された位置が類似している場合には，同じような挙動をとるという特徴を活かしている．アクセスされた位置が類似しているフェーズごとに，最適なパイプライン段数を検出し，最適なパイプライン段数が検出された後のフェーズでは最適なパイプライン段数で動作させるという仕組みとなっている．

3.2 signature

従来コントローラでは，同じような挙動を行うフェーズを検出するためにsignatureを用いる．signatureの生成される過程を図3.16を用いて説明を行う．まず，プログラムをプログラムカウンタに応じて128分割のものとみなし，signatureもプログラムの分割に合わせた128bitのレジスタ構成にしておく．図3.16では説明の簡略化のため，8bitの構成にしている．フェーズを実行した場合，プログラムのさまざまな位置が実行され，アクセスのあったプログラムの分割部分に対応するsignatureのビットが立つ仕組みになっている．図3.16のフェーズ(1)とフェーズ(3)はアクセスされた順番は違うが，プログラムのアクセスされた部分がほぼ同じであるため．挙動や作成されるsignatureはほぼ同じになる．アクセスされる部分が類似している場合のフェーズは同じ負荷となることが多いため，類似したフェーズごとに最適なパイプライン段数を予測することは有効である．

フェーズを実行した後，生成されたsignatureをhistory table(履歴表)で記憶するが，すべてのsignatureに対して最適なパイプライン段数を記録するとハードウェア量が膨大になるため，16個のsignatureのみ記憶できるようにしている．また，16個のsignatureのみの記憶では，全てのsignatureのパターンに対応できないため，それぞれのsignatureのビット毎に一致比較を行い，類似しているsignatureは同じsignatureとして履歴表へ記憶する．このような動作をフェーズ毎に行うことで，挙動が似ているフェーズを検出し，履歴表への記録を行う．

3.3 signatureと履歴表を用いた予測

次に正確なパイプライン段数を予測する方法について述べる．従来コントローラでは履歴表を用いて予測を行っている．履歴表には，signature,

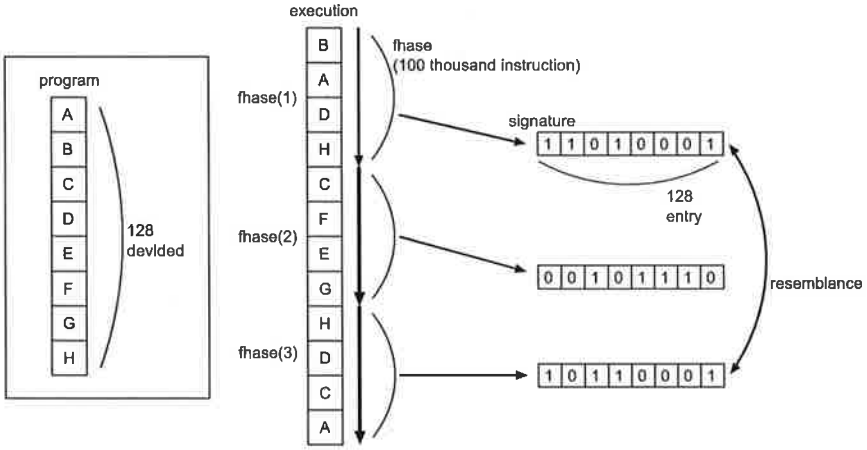

図 3.16: signature 作成の例

それぞれのモードでの電力遅延積（消費エネルギー × 実行時間），最適なパイプライン段数などを記録しておくことができる．

10 万命令実行毎に signature が生成され，履歴表へ signature が送信される（図 3.17 の A）．この signature に一致，あるいは類似したエントリを履歴表の中から検出できなかった場合，新たに signature を登録し，次の 10 万命令間の実行はモード A で実行するようプロセッサへ返答する (図 3.17 の B)．モード A での実行終了後，この signature においてモード A で実行した場合の電力遅延積を履歴表へ登録する (図 3.17 の C)．

その後，同じ signature が履歴表へ送信された場合，モード B で実行するようプロセッサへ返答する (図 3.17 の D)．このように，それぞれのモードで実行させ，電力遅延積をモード毎に比較することで，次回以降のフェーズでは最適なパイプライン段数で実行することができる (図 3.17 の E)．

3.4 問題点

従来コントローラでは，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制御を行っている．しかし，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

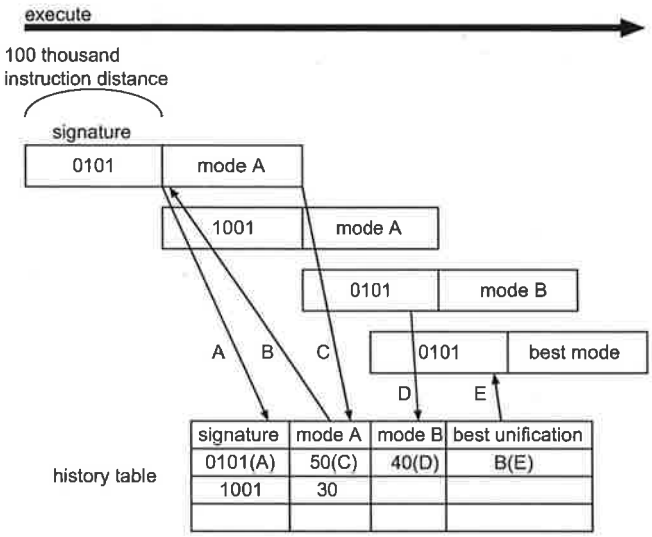

図 3.17: プログラム実行と履歴表

結果，切替粒度が粗く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．さらに，それぞれの signature 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．また，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．

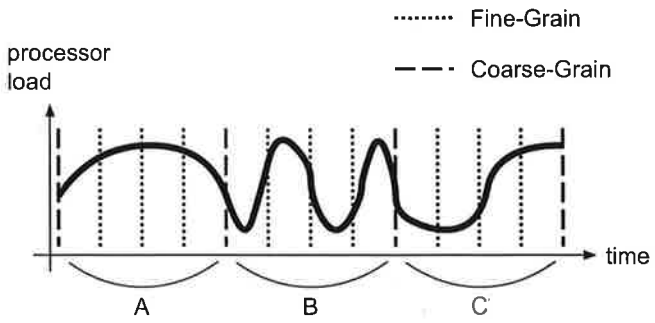

図 4.18: 切換周期における違い

4 細粒度切換手法の提案

本論文では従来コントローラよりも可変パイプライン段数プロセッサの性能を十分に発揮できるコントローラ作成のために，1) 従来コントローラでは細粒度な切換えに対応できないため，細粒度なコントローラを提案する，2) 細粒度に切換えを行う場合，切換オーバヘッドが無視できなくなるためオーバーヘッド低減手法を提案する，についての2手法の提案を行う．また，切換オーバヘッドについての詳細は第4.3節で説明を行う．

4.1 細粒度切換えについて

本節では細粒度に切換えを行った場合の利点について述べる．従来コントローラでは10万命令に1度の割合で切換を行っている．しかし，10万命令に1度の切換えでは細かい負荷の変動がある場合に対応できない．この様子を図4.18に示す．図4.18では縦軸がプロセッサ負荷，横軸が実行時間である．図4.18のAでは負荷の変動が細かくないため，切換周期が粗粒度（Coarse-Grain）の場合にも負荷に応じたモード切換えが可能である．しかし，図4.18のB,Cの場合には切換周期よりも負荷の変動が細かいため負荷に対応した切換えを行うことができない．そのため，今回は細かい負荷の変動にも対応できるよう細粒度（Fine-Grain）なコントローラを提案する．細粒度にすることにより，粗粒度では対応できていなかった図4.18のB,Cのような細かい負荷の変動にも対応できることになる．

4.2 細粒度コントローラの提案

本節では提案VSPコントローラの仕組みについて述べる．本論文で提案するコントローラはプロセッサの負荷に応じてモードを切換える手法である．負荷が高く高速な処理が必要だと考えられる場合には高速モードに，負荷が低く高速な処理が必要ない場合には低消費電力モードに切換えることで，高性能かつ低消費エネルギーが実現できる．負荷の取得方法が誤っていた場合，最適でないモードへ切換えを行うことがあるため，コントローラを設計する上で，負荷の動的検出方法は重要である．単位時間当たりにどのくらい命令が実行されているかということが負荷の指標の一つであるが，命令の依存関係やメインメモリアクセス速度などの問題から，命令はスムーズに流れず，プログラムの特性によって負荷にばらつきが生じる．負荷をOS側で検出することもできるが，細粒度なコントローラを設計することから考えると，ハードウェアでの実現が必要である．

プロセッサの負荷を判断する指標を決定するため，プロセッサシミュレータsimplescalar [13] [14]を用い，SPECベンチマーク [18]の10本を実行した結果，負荷の変動と注目したパラメータの変動が似ているものを検出しようと試みた．実験を行った結果，メインメモリアクセスが特に命令の流れを遮っていることがわかった．現在プロセッサの処理速度に比べメインメモリアクセス速度の遅さが問題となっているが，今後もメインメモリアクセス速度の遅さは問題となると考えられる．そこで，今回load命令とstore命令の頻度からモード切換える制御を与えることとした．load命令とstore命令の回数をカウントする場合，キャッシュヒットとキャッシュミスを区別なくカウントするが，キャッシュミス回数をカウントする手法も実装し比較したところ，評価結果の変化は見られなかった．そのため，実装が容易だと考えられるload命令とstore命令の回数をカウントする手法を選択した．また，モード切換えを行うしきい値は，いくつか考えられるパターンにおいてシミュレーションの評価を行い，よりよい結果が得られた値を用いた．

図4.19にVSPコントローラの概念図を示す．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32 entry queueに渡される．32 entry queueは32個の3bit D-FFで構成されるキュー構造になっており，3bit D-FFに1サイクル中のロード，ストア命令の命令数が保存できる．32エントリであるため，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

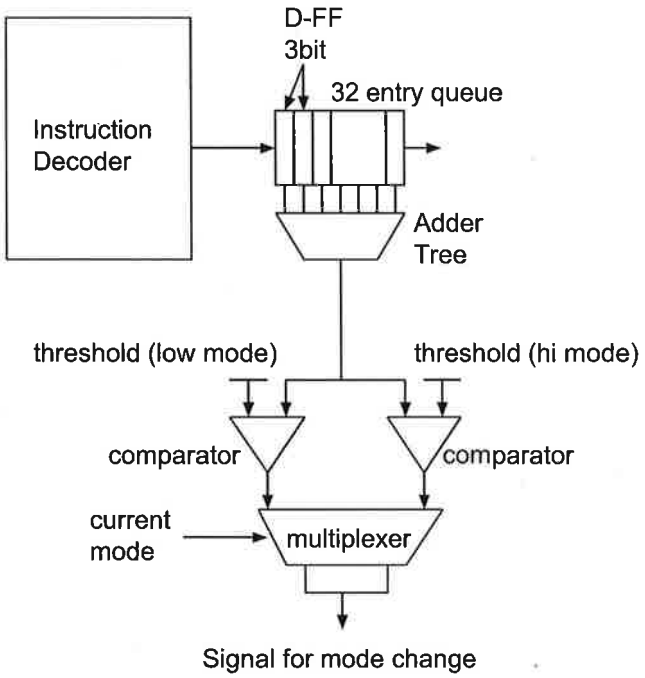

図 4.19: 提案コントローラに必要なハードウェア

る．また，デコーダから命令数が送られてくる度に一番古い命令数は破棄される．Adder Tree は加算機の集合であり，最新 32 サイクル分のロード，ストア命令の合計命令数を計算する回路である．この回路は，32 entry queue ではロード，ストア命令数が 1 サイクルごとに記憶されているため，32 サイクル全体の命令数を導き出すために必要となる．comparator は Adder Tree から得られたロード，ストアの合計命令数と，あらかじめ設定しておいたしきい値とを比較し，モード切換えを行うかを制御する部分である．このように，提案 VSP コントローラは図 4.19 における回路を追加するだけで構成可能ある．

4.3 切換オーバーヘッド

本論文ではモード切換えを細粒度に行う VSP コントローラを作成するが，細粒度な設計の場合，モード切換えを行うためのオーバヘッドの影響を多く受けてしまう．VSP においてモード切換えを行うためには以下

の２つのオーバヘッド，すなわち 1) 周波数変更を行うためのオーバヘッド，2) パイプラインレジスタ数が減少する場合に，パイプラインフラッシュをしなければならないため発生するオーバヘッド，が発生する．

モード切換えを行う場合，前者のオーバヘッドは必ず発生してしまうため，回避することができない．しかし，周波数変更にかかるオーバヘッドはたかだか 2, 3 サイクルであるため無視できる．後者は，高速モードから低消費電力モードへ切換える場合に発生する．VSP のパイプラインレジスタの中には高速モードでは使用されるが，低消費電力モードでは使用されないものがある．このようなパイプラインレジスタは，高速モード動作時ではデータが保存されるが，高速モードから低消費電力モードへモード切換えを行った場合，低消費電力モード時ではこのパイプラインレジスタを使用できないため，高速モード時に保存したはずのデータを呼び出すことができなくなり処理が正常に行えない．この問題を解決するために，高速モードから低消費電力モードへ切換えを行う場合，パイプラインレジスタを一度フラッシュさせ，パイプラインレジスタ内に残っていた命令を初めからやり直すという手法が考えられる．しかし，この方法では処理が進んでいたはずの命令を初めからやり直さなければならないので性能が悪化してしまう．これが後者のオーバヘッドである．このオーバヘッドは数十サイクル以上のオーバヘッドがかかってしまうため，細粒度にモード切換えを行う場合には，このオーバヘッドが積み重なり，性能を悪化させる原因となってしまう．

4.4 切換オーバーヘッド低減手法の提案

そこで，高速モードから低消費電力モードへの切換え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，モード切換えのオーバヘッドを隠蔽する手法を提案する．分岐予測とはプログラムカウンタの飛び先が 2 パターンある命令においてどちらの飛び先を次から実行するのかを予測するものである．例えば，C 言語の if 文では if 文の中へ飛ぶのか，外へ飛ぶのかを予測を行う処理が分岐予測である．分岐予測はパイプラインプロセッサでプログラムを実行している場合に効果的である．なぜなら，分岐命令以降の命令を分岐予測命令の完了を待たず投機的に実行を行うことができるためである．よって，分岐予測の結果がプログラムの実行と同じ飛び先を示していれば，分岐命令以降の命令をオーバヘッドなしで実行できる．しかし，分岐予測が外れていた場合，分岐命令以

降の命令は破棄する必要がある．近年では分岐予測の研究が盛んに行われており，分岐予測の成功率は90%以上であるため，分岐予測を行うことによりプロセッサはより高性能を発揮できる [15] [16].

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% であり，分岐予測ミスは高度な分岐予測を行ったとしても4%程度は発生してしまう．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，モード切換えを分岐予測ミス時に限定しても細粒度なモード切換えを実現できるため，本論文では分岐予測ミスをモード切換オーバヘッド削減のために活用する．

図4.20を用いてオーバーヘッド削減手法を説明する．図4.20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．ここで，分岐予測ミスが発生した場合，データが残るのはプロセッサの命令処理ステージのうち実行ステージ以降のパイプライン内だけである（図4.20のB）．この理由は，現在のプロセッサでは基本的に分岐予測は当たるものだとし，分岐予測で得られた分岐先から命令を読み込む．しかし，分岐予測が外れた場合は，分岐命令以降に読み込んだ命令とは異なった分岐先から命令を読み込むべきであったため，既に読み込まれた命令は実行すべきでない命令である．そのため，これらの命令は破棄されることになり，実行ステージ以前の命令はなくなる．

次に，実行ステージ以前のデータはフラッシュされ，パイプラインレジスタ内は空になる（図4.20のA）．モード切換時において実行ステージ以降の処理は高周波数で高速モードとして動かし，実行ステージ以前の処理はパイプライン段数を変更し低周波数で低消費電力モードとして動かすという期間（以下,「マイグレーションモード」）を設ける．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，新しくフェッチされた命令がリザベーションステーションに登録され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（図4.20のC）．その時，低消費電力モードに初めて完全に切換えることができる．このようにすることで，モード切換え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.

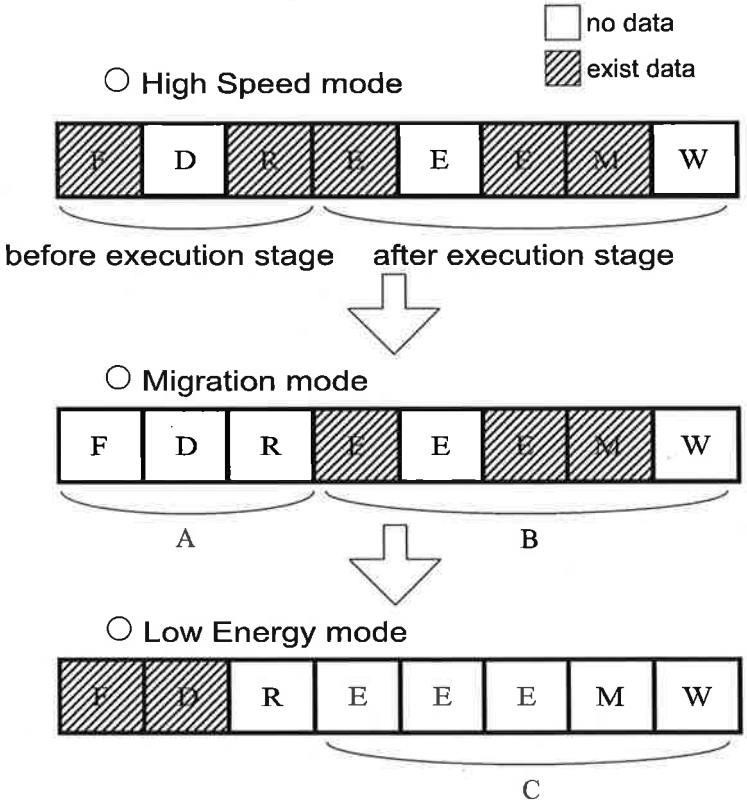

図 4.20: 提案手法の説明図

4.5 実装

本節では細粒度コントローラの具体的な実装方法について述べる．図4.19に細粒度コントローラの概念図を示したが，Adder Treeは加算機の集合であり，電力をかなり消費する．そこで，新たに図4.21のようにハードウェア構成の変更を行った．このような構成にすることで図4.21のCのTotal Numに32 entry queueに保存されているロード，ストア命令数の総和が保存され，Adder Treeは必要がなくなる．Total Numにロード，ストア命令数の総和が保存される手順を以下に示す．図4.21のAの減算器はDecoderから得られたロード，ストア命令数から32 entry queueに保存されている一番古いロード，ストア命令数の減算を行う．これにより，Total Numに保存すべきロード，ストア命令数の差分が得られる．その後，図4.21のBの加算器で図4.21のCのTotal Numと図4.21のAから得られたロード，ストア命令数の加算を行うことで，32 entry queueに保存されているロード，ストア命令数の総和がTotal Numに保存される．このようにすることで，消費電力の高い加算機の集合を用いない方法で提案コントローラを実現することができる．さらなる低消費電力化のため，一般的なDFFを用いていたレジスタ部分をTSPC-DFFに変更した[20].

図4.22にハードウェア低減手法も導入したコントローラの概要図を示す．図4.22において，_(アンダーバー)がついているものは使用するビットを指定しているものであり，アンダーバーの後が0ならばその信号の0 bit目，1ならばその信号の1 bit目を示す．それぞれの信号の意味を表4.1に示す．低消費電力モードから高速モードへ切換えを行う場合は，設定したしきい値との比較が正であれば切換信号をプロセッサへと渡すことができる．高速モードからマイグレーションモードへの切換えを行う場合は，設定したしきい値との比較結果が正であり，プロセッサから分岐予測ミスが起きている場合に切換信号をプロセッサへと渡す．マイグレーションモードから低消費電力モードの切換えを行う場合は，実行段階以降に命令が存在していないことを確認でき次第，切換信号をプロセッサへと渡している．この場合以外は現在のモードを維持するよう設計を行った．

今回従来コントローラの電力，ハードウェア量についての評価は行っていない．しかし，DFFは比較的電力と面積の大きい部分であるが，提案コントローラではDFFを100個程度の用いているのに比べ，従来コントローラではDFFを数万個用いているために，従来コントローラは提案コントローラより電力，ハードウェア量ともかなり多いと考えられる．

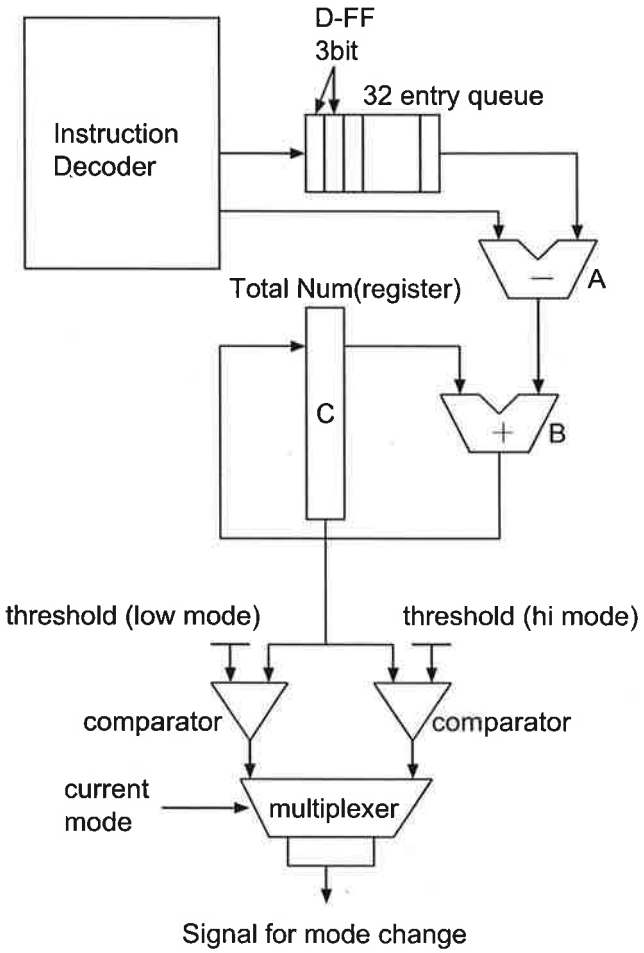

図 4.21: 加算機を削減した場合のハードウェア構成

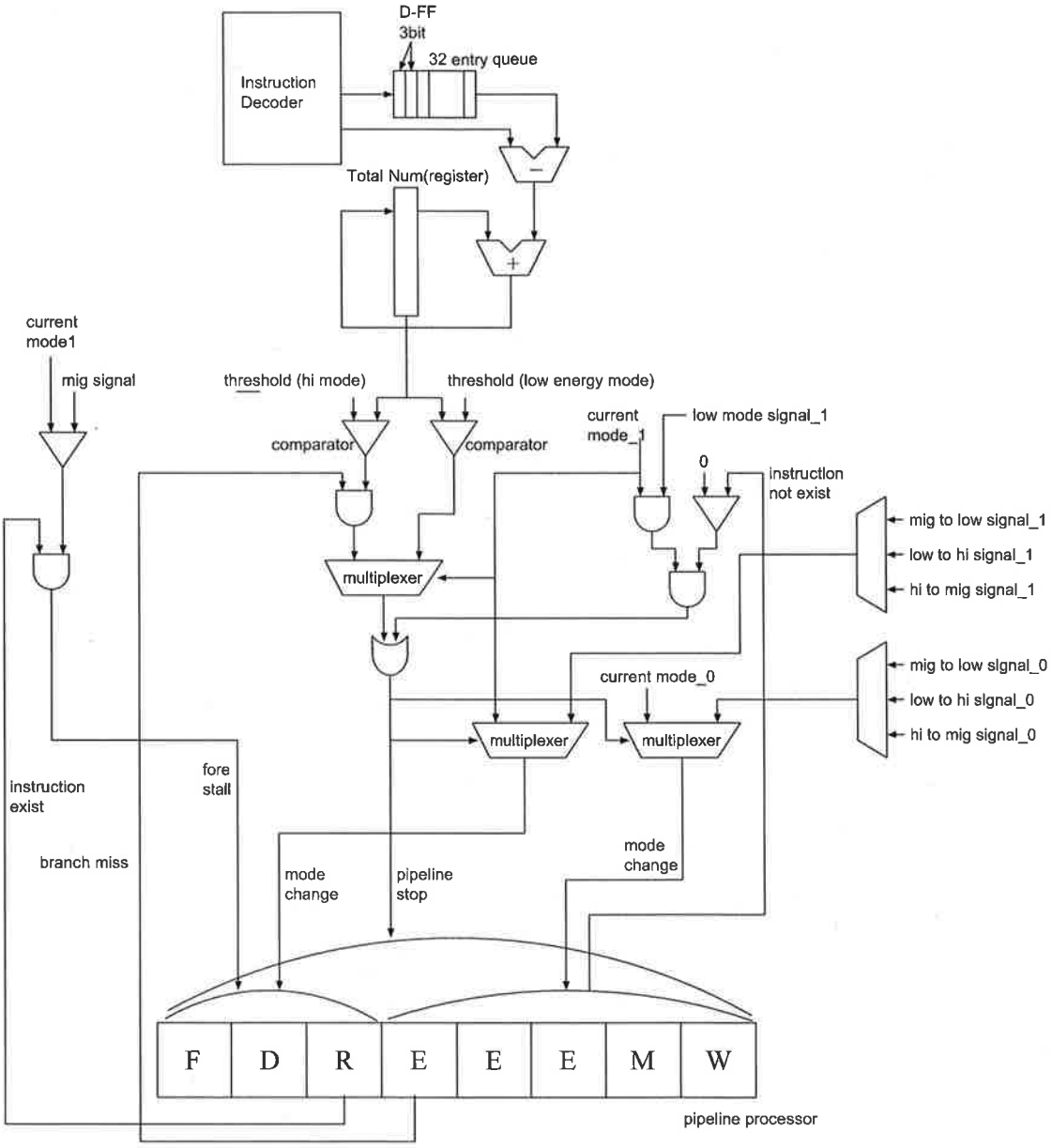

図 4.22: オーバヘッド低減手法を含めたハードウェア構成

表 4.1: 主な信号線の意味

信号名	意味
current mode	現在のモード信号
low mode signal	低消費電力モードの信号
hi mode signal	高速モードの信号
mig mode signal	マイグレーションモードの信号
threshold(hi mode)	高速モードのしきい値
threshold(hi mode)	低消費電力モードのしきい値
branch miss	プロセッサ側で分岐予測ミスがおきたかどうか
instruction exist	指定したステージで命令が処理されているかどうか
mode change	次サイクルのモード信号
fore stall	実行段階以前をストールさせる信号
pipeline stop	モード切換を行う際周波数を変更するために必要となる信号

5 消費エネルギーとエネルギー遅延積

本章では回路の消費エネルギーについての評価指標の定義を行う．高性能なプロセッサを評価する場合には実行時間を見比べればよく，低消費電力プロセッサにおける評価では主に消費エネルギーを見比べればよい．しかし，今回は高性能かつ低消費エネルギーのプロセッサを作成することを目標としているため，実行時間と消費エネルギー両方について見比べる必要がある．そのため，次節で消費エネルギーについての定義を述べ，第5.2節で高性能かつ低消費エネルギーのプロセッサの評価を行う場合によく用いられる電力遅延積について述べる．

5.1 消費エネルギーの定義

CMOSで構成されたプロセッサの消費エネルギーEは消費電力P，実行時間Tを用いて以下の式によって求められる．

$$E = P * T = St * C * V^2 * G * cycle$$

Stはゲートのスイッチング確率，Cは容量（ゲート容量，配線容量を含む），Vは電源電圧，Gはアクティブなゲート数，cycleはアプリケー

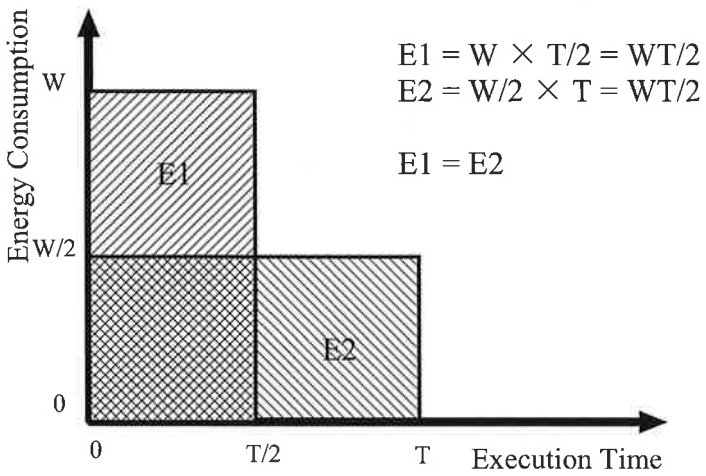

図 5.23: 実行時間と消費電力

ションを実行するサイクル数である．図 5.23 は式を用いて動作周波数を変更した時の消費エネルギーを計算したものである．図 5.23 からも分かるように消費エネルギーは動作周波数に依存しない．周波数を高くして実行時間を小さくすると消費電力が増大し，逆に周波数を低くして消費電力を小さくすると実行時間が増大してしまうからである．これが消費エネルギーと消費電力の違いである．

5.2 電力遅延積の定義

本研究では，どちらがより低消費エネルギーと高性能の両立を達成出来ているかを評価するために電力遅延積を用いる．電力遅延積は消費エネルギー E と実行時間 T を用いて以下の式によって求められる．

$$\text{電力遅延積} = E * T$$

この電力遅延積が小さいほど低消費電力と高性能の両立が達成できている．電力遅延積では周波数に依存しない消費エネルギーに周波数に依存する実行時間をかけているため，消費エネルギーの指標とは異なり，周波数に依存する値となる．また，電力遅延積は消費電力に実行時間の 2 乗をかけても求めることができる．そのため，消費電力と性能について評価する場合は性能の方が重要視された指標となるが，今回は消費エネルギーと性能についての評価であるので電力遅延積を用いた．

6 性能評価

本論文ではSimpleScalar Tool Set 内の out-of-order 実行シミュレータをベースに提案コントローラと従来コントローラを組み込んだ．電力の評価は，消費電力を見積もるツールである wattch [17] を SimpleScalar 上に組み込み評価を行った．命令セットは SimpleScalar PISA であり，表 6.2 に示すように，SPECint2000 の 10 本をベンチマーク・プログラムとして用いた．実行は最初の 20 億命令をスキップした後，20 億命令を測定に用いた．

表 6.2: ベンチマーク

ベンチマーク名	処理内容
ammp	計算化学
art	画像認識，ニューラルネットワーク
bzip2	圧縮
equake	地震波伝播シミュレーション
gcc	C 言語コンパイラ
gzip	圧縮
mcf	組み合わせ最適化
parser	文字列処理
vpr	FPGA の配置配線
vortex	オブジェクト指向データベース

6.1 評価環境

表 6.3 にシミュレーションにおいて仮定したプロセッサの構成を示す．同時に，低消費電力モードでは周波数の関係上，表 6.4 に示すようにキャッシュアクセスやメインメモリアクセスのために必要なサイクル数を変更した．低消費電力モード時の場合，メモリに与える周波数を変更しメモリに対する消費電力も低減する方法もあるが，今回はメモリの周波数を変更せずシミュレーションを行った．また，本論文ではパイプライン段数を高速モードは 20 段と仮定し，低消費電力モードは 5 段とした．近年のプロセッサと比較するとかなり段数の多い設定となっているが，これは 20 段以外の段数では SimpleScalar の設計が困難であるためである．

表 6.3: プロセッサ構成

processor	issue width 8, RUU 64 entry, LSQ 32 entry, int ALU 8, int mult/div 4, fp ALU 8, fp mult/div 4, memory port 8
branch prediction	PHT 8K entry history width 6 の gshare BTB 2K entry PAS 16 entry
L1 instruction cache L1 data cache L2 cache	64KB/32B line/1-way 64KB/32B line/1-way 2MB/64B line/4-way
memory	first reference 64 cycle
TLB	forward interval 2 cycle

表 6.4: アクセスレイテンシ

モード	高速モード	低消費電力モード
L1	1 cycle	1 cycle
L2	6 cycle	2 cycle
memory	18 cycle	12 cycle
TLB miss	30 cycle	10 cycle

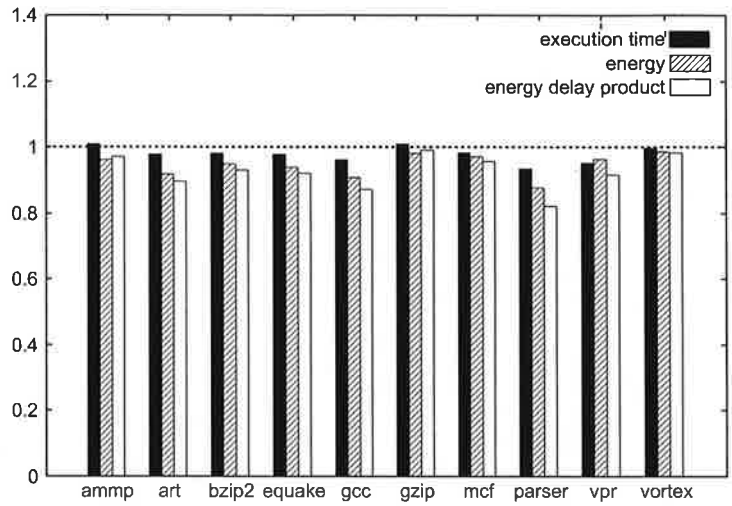

図 6.24: 評価結果

6.2 評価結果

このような評価環境で，従来コントローラと提案手法を Simple Scalar 上に実現し評価を行った．図 6.24 に，上記 2 手法の実行時間，消費エネルギー，電力遅延積の評価結果を示す．また，図 6.24 はすべて従来コントローラを 1 と正規化した．

今回，多くのベンチマークにおいて実行時間，消費エネルギー，電力遅延積において提案手法の方が良い結果を示した．これはパイプライン段数変更の細粒度化と切換え時のオーバーヘッド低減の効果であると考えられる．ammp などの負荷の変動が少ないプログラムではあまり効果は得られなかったが，負荷の変動があるプログラムに対しては提案手法の有効性を示せた．いくつかのベンチマークで実行時間が増えているが，これは提案手法が電力遅延積が改善されるように設計しているためだと考えられる．そのため，実行時間は増えているが，それ以上に消費エネルギーが低減でき，電力遅延積も改善できている．また，全体的に実行時間より消費エネルギーの方が良い結果が得られているが，これは提案手法が高速モードより低消費電力モードで実行される場合の方が多い設計となっているためだと考えられる．

6.3 ハードウェア規模の評価

第6.2節の評価はプロセッサ部分のみの評価であり，コントローラを含めた電力評価を行っていない．そのため，本節ではハードウェアシミュレータ nanosim を用いてコントローラにかかる面積，電力の評価を行った．本研究室で試作されたVSPプロセッサと，商用プロセッサの1つとして主にノートパソコンで用いられている Intel Atom プロセッサ [19] との比較を行った結果を表6.5に示す．また，提案コントローラを他プロセッサに組み込んだ場合の面積，電力の割合を表6.6に示す．

表 6.5: 面積，電力結果

	提案コントローラ	VSP プロセッサ	Intel Atom プロセッサ
面積	2821 トランジスタ	50 万トランジスタ	4700 万トランジスタ
電力	8.28μW	高速モード時：72mW 低消費電力モード時：14.4mW	10W

表 6.6: 提案コントローラの面積，電力割合

	VSP プロセッサ	Intel Atom プロセッサ
面積割合	0.005642%	0.00006%
電力	高速モード時：0.0001% 低消費電力モード時：0.0005694%	0.000000828%

評価結果を見ると，提案コントローラが他プロセッサと比較して無視できるほどの面積と電力であることがわかる．そのため，提案コントローラを他プロセッサに組み込んでもプロセッサに影響を与えることはないと考えられる．

7 まとめ

本論文では，VSP を最大限に活かすための細粒度なパイプライン段数切換コントローラについて示した．従来コントローラでは切換頻度が粗いために切換えを行えなかった細かな負荷の変動に対しても切換えを行うことができる細粒度なコントローラを設計した．細粒度なコントローラを設計するため，ハードウェアで構成を行うと同時に，負荷の変動を正確に予測するため load，store 命令数に着目をした．同時に，細粒度にパイプライン段数を切換えた場合に問題となるオーバヘッドを削減する手法を提案した．オーバーヘッド削減手法が細粒度に切換えを行う場合に対しての妨げにならないように，分岐予測ミスをきっかけにモード切換えをさせることでオーバーヘッドを低減するよう設計を行った．

これらの手法をプロセッサシミュレータ simplescalar に実装を行い，従来コントローラと比較したところ，電力遅延積において 1 割程度の改善が得られた．また，ハードウェア設計での評価結果から面積や電力も一般的なプロセッサと比較し無視できるほど小さいことがわかった．

謝辞

本研究を行うにあたり，多くの助言をいただきました近藤利夫教授，大野和彦講師，並びにご指導，ご助言いただきました下さいました佐々木敬泰助教に深く感謝いたします．また，様々な局面にてお世話になりました計算機アーキテクチャ研究室の皆様にも心より感謝いたします．

参考文献

[1] Min Yeol Lim, Vincent W.Freeh, "Determining the Minimum Energy Consumption using Dynamic Voltage and Frequency Scaling", IEEE, 2007.

[2] Johan Pouwelse, Koen Langendoen, Henk Sips, "Dynamic Voltage Scaling on a Low-Power Microprocessor", 7th ACM Int, 2001.

[3] 市川 裕三，佐々木 敬泰，弘中 哲夫，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 ", ACS, 2006.

[4] 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）.

[5] Yuji Ichikawa, Takahiro Sasaki, Tetsuo Hironaka, Kazuya Tanigawa, Toshiaki Kitamura, and Toshio Kondo: A Design of Prototype Low Energy Processor by Variable Stages Pipeline Technique, Proc. of International Technical Conference on Circuits/Systems Computers and Communications（ITC-CSCC2005），Vol.2, pp.561-562（2005 年 7 月）.

[6] 嶋田 創，安藤 秀樹，島田 俊夫，"低消費電力化のためのパイプライン"，情報処理学会研究報告，ARC, 2001.

[7] 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"，先進的計算基盤システムシンポジウム SACSI2003, 2003.

[8] Jun YAO, Hajime SHIMADA, Yasuhiko NAKASHIMA, Shin-ichro MORI, Shinji TOMITA " An EDP Study on the Optimal Pipeline Depth for Pipeline Stage Unification Adoption,", Information Processing Society of Japan (IPSJ), 2006.

[9] Jun YAO, Shinobu MIWA, Hajime SHIMADA, Members, and Shinji TOMITA, "A Dynamic Control Mechanism for Pipeline Stage Unification by Identifying Program Phases", IEICE TRANS. INF. & SYST., Vol. E91-D, pp.1010-1022, APRIL 2008.

[10] A.S. Dhodapkar and J.E. Smith, "Managing multi-configuration hardware via dynamic working set analysis," Proc. 29th Annual International Symposium on Computer Architecture, pp.233-244,IEEE Computer Society, 2002.

[11] Jinson J Koppanalil, Prakash Ramrakhyani, Sameer Desai, Eric Rotenbergm, Anu Vaidyanathan : A Case for Dynamic Pipeline Scaling, ACM 2002

[12] 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) .

[13] The SimpleScalar Tool Set. Version 3.0, "http://www.simplescalar.com/".

[14] Todd Austin, Eric Larson, Dan Ernst, "SimpleScalar: An Infrastructure for Computer System Modeling," Computer, vol. 35, no. 2, pp. 59-67, Feb. 2002.

[15] James E. Smith, A study of branch prediction strategies, Proceedings of the 8th annual symposium on Computer Architecture, p.135-148, May 12-14, 1981, Minneapolis, Minnesota, United States.

[16] Scott McFarling, "Combining Branch Predictors," WRL Technical Note TN-36, June 1993.

[17] David Brooks, Vivek Tiwari, Margaret Martonosi, Wattch: a framework for architectural-level power analysis and optimizations, p.83 - 94, ISCA '00, 2000.

[18] Joshua J. Yi, David J. Lilja, "Simulation of Computer Architectures: Simulators, Benchmarks, Methodologies, and Recommendations," IEEE Transactions on Computers, vol. 55, no. 3, pp. 268-280, Mar. 2006, doi:10.1109/TC.2006.44

[19] Intel Atom Processor Z540, Z530, Z520, Z510, and Z500 on 45 nm Process Technology Datasheet, March 2009.

[20] 草場 律, 近藤 利夫：高性能セミスタティック TSPC DFF の検討, 電子情報通信学会論文誌. C-II, エレクトロニクス, II-電子素子・応用 J81-C-2(5) pp.469-476 1998.