特別研究報告書

プログラムの実行特性に応じた
パイプラインステージ統合の動的適用

指導教員　　富田眞治 教授

京都大学工学部情報学科

中山英卓

平成19年2月9日

プログラムの実行特性に応じた パイプラインステージ統合の動的適用

中山英卓

内容梗概

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方が要求されている．この要求を満たすために, 現在 DVS(Dynamic Voltage Scaling) と呼ばれる方式が導入されている．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．

これに対して，PSU(Pipeline Stage Unification) と呼ばれる手法が提案されている．PSU は単純ではあるが，効果的に消費エネルギーの削減をすることができる．PSU では DVS 同様に，プロセッサの消費電力を削減するために，クロック周波数を低下させるが，DVS とは異なり，電源電圧を低下させるのではなく，パイプライン・レジスタをバイパスさせることによって複数のパイプライン・レジスタを統合する．PSU により消費電力削減できる理由は以下の 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・レジスタのステージ統合によりプロセッサのパイプラインが短くなる．これにより，プログラムの実行に必要なサイクルが削減され電力消費の時間を短くすることができる．

PSU を用いて効果的な EDP(エネルギー遅延積) を実現するために，PSU の制御方法の研究が行われている．過去の研究から、プログラムは，キャッシュ・ミス率，IPC，エネルギー消費などの点からいくつかの類似したフェーズに分割されることが示されている．これに基づき，シグニチャという類似したフェーズを検出する仕組みを用いて，過去に実行したフェーズと類似したフェーズでは，過去の学習結果より得られた最適な PSU の統合度を割当て，プログラム実行完了までの EDP を最小化する制御を行っている．

しかしこの方法では，一定長の命令インターバル中は，統合度を変えずに制御するため，インターバル中にプログラムの実行特性が変化してもすぐには対応できない．例えば100K 命令という命令インターバル単位で統合度を切替えている場合には，インターバル中の統合度を一定にしたままであり，分岐予測ミスなどでプロセッサのCPI が増加した時に，実行サイクル数が増加し，EDP の増加につながる．

この分岐予測ミス率によるEDP の増加は，パイプラインが短いほど小さくなる．なぜなら，パイプラインが短い方が分岐予測ミス・ペナルティが小さくなるからである．したがって，分岐予測ミス率が頻発する時にPSU でパイプラインを短くすれば，EDP の増加は最小限に抑えられると考えられる．

そこで，本論文では，プログラム実行中の分岐予測ミス率に対して，PSU の統合度の切替えを行うことを提案する．統合度の切り替えには、直近の128 の分岐予測のヒット／ミスの記録より求めた分岐予測ミス率を用いる。この分岐予測ミス率が、統合度1 と統合度2 の切り替えのための閾値を越えた時に統合度を2 に切り替え、下回った場合は統合度を1 に切り替えるものとする。同様に、統合度2 と統合度4 を切り替える閾値も存在し、同様のポリシーで統合度を切り替える。これにより、従来のシグニチャを用いる方法と比較して、可変長のインターバルで統合度を切り替えと、ハードウェア・コストの削減を実現できる。

評価を行うにあたり、まず、解析で統合度の切り替えを行う閾値の目安をつけた後、シミュレーションによる評価行った。評価の結果、SPECint95 ベンチマーク8 個の平均では、統合度1 と統合度2 の切り替えの閾値を20%以上、統合度2 と統合度4 の切り替えの閾値を20%以上にした場合にEDP を最小化できるという結果を得た。上記の閾値の範囲は非常に広いため、この広い範囲でEDP を最小化できる原因を調査した。その結果、プログラムの実行中における分岐予測ミス率は、実効時間の80%において20%以下であるため、20%以上の領域で閾値を変化させても、統合度の選択状況に大きな影響を与えないためであることが分かった。

Dynamic Pipeline Stage Unification Adoption Depending on Execution Characteristic of Program

Hidetaka Nakayama

Abstract

Rrecent 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 redudes supply voltage when a proceessor runs at a low clock frequency. This saves energy consumption for program exection. 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 a alternative, a method called pipeline stage unification or PSU is proposed. PSU reduces power consumption by inactivating and bypassing pipeline register and using a shallow pipeline during the program exection. PSU saves power consumption in the folowing ways. Firstly, PSU saves power consumption by stopping the clock signal to bypassed pipeline registers. Secondly, PSU reduces the program exection times by reducing penalthies and latencies with shallow pipeline.

There are some researches about PSU control for reducing Enegy Delay Product or EDP. According to previous reaserch, it is shown that programs can be divided into several phases in which program would have similar behaviors including cache miss, IPC and power consumption.Based on this assumption, there are two PSU control methods.One method uses phase detection hardware and the other method uses a history tabel.They detect similar phases by using a hardware called signature and use the same degree of PSU for the similar phases. The method with history table stores signature data and suitable unificationo degree into a entry.When the processor execute same phase ,the processor use this imformation to minimize EDP.

However, this method uses the same pipeline stage unifcation degree during the instruction interval. This method cannot change unification degree immediately when the branch misprediction rate or cache miss rate increases.The increase of branch misprediction rate causes the increase of exection cycles and

EDP.

The increase of EDP can be prevented by using a shallow pipeline, because a shallow pipeline will have lower CPI due to decreased branch misprediction penalties compared to the deep pipeline.Therefore,there is a possibility that using a shaollow pipeline for high branch misprediction reduce EDP efficiently.

Based on this assumption, this research described in this paper is foucusing on a method of controling PSU degree according to branch misprediction rate. Two thresholds are prepared for changing PSU degree. One is used for switching from PSU degree 1 called U1 to U2 when branch misprediction rate becomes larger than the threshold, or switching from U2 to U1 when branch misprediction becomes lower than the threshold.The other is used for switching from U2 to U4 in the same way.The advantaeges of this method over the method with phase detection hardware as mentioned above are as follow:

1. The method with phase detection hardware cannot change PSU degree during the instruction interval like 100K instruction.The method propsed in this paper can choose vourious length as the instruction interval length , according to branch misprediction rate.
2. The hardware of the method proposed in this paper is smaller than the hardware of the phase detection method.

Before evaluating EDP of this method, I estimated two best threshold values for reducing EDP.The results show that the average EDP for SPECint95 benchmarks is minimized when the threshold used for switching from U1 to U2 is more than 20% and the threshold used for U2 and U4 is more than 20%.This range is very large.Therefore, I searched for the reason why EDP is minimized in this large range.I found that branch misprediction rate is less than 20%, during 80% of the program execution time. Therefore,EDP change little when two thresholds are chnaged in the range.

プログラムの実行特性に応じた パイプラインステージ統合の動的適用

目次

第１章　はじめに

近年のモバイル・プロセッサでは，低消費エネルギーと高性能の両立が要求されている。この要求を満たすために、現在 DVS(Dynamic Voltage Scaling) と呼ばれる方式が導入されている。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，ソフト・エラー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(Pipeline Stage Unification) と呼ばれる手法が提案されている．[1, 2, 3]．PSU は単純ではあるが，効果的に消費エネルギーの削減をすることができる．PSU では DVS 同様に，プロセッサの消費電力を削減するために，クロック周波数を低下させるが，DVS とは異なり，電源電圧を低下させるのではなく，パイプライン・レジスタをバイパスさせることによって複数のパイプライン・レジスタを統合する．PSU により消費電力削減できる理由は以下の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・レジスタのステージ統合によりプロセッサのパイプラインが短くなる．これにより，プログラムの実行に必要なサイクルが削減され電力消費の時間を短くすることができる．たとえば，フロントエンド・パイプラインを短縮させることにより，分岐予測ミス・ペナルティは削減され，実行サイクル数は削減される．

PSU を用いて，効果的なエネルギー遅延積 (EDP: Energy Delay Product) の削減を行うために，PSU の統合度制御の研究が行われている．文献 [4, 5] では，プログラムの実行を 100K 命令の命令インターバル単位で見た場合，キャッシュ・ミス率，IPC，エネルギー消費などが類似したいくつかのフェーズに分かれると述べている．これに基づき，文献 [6] では，シグニチャという類似したフェーズ

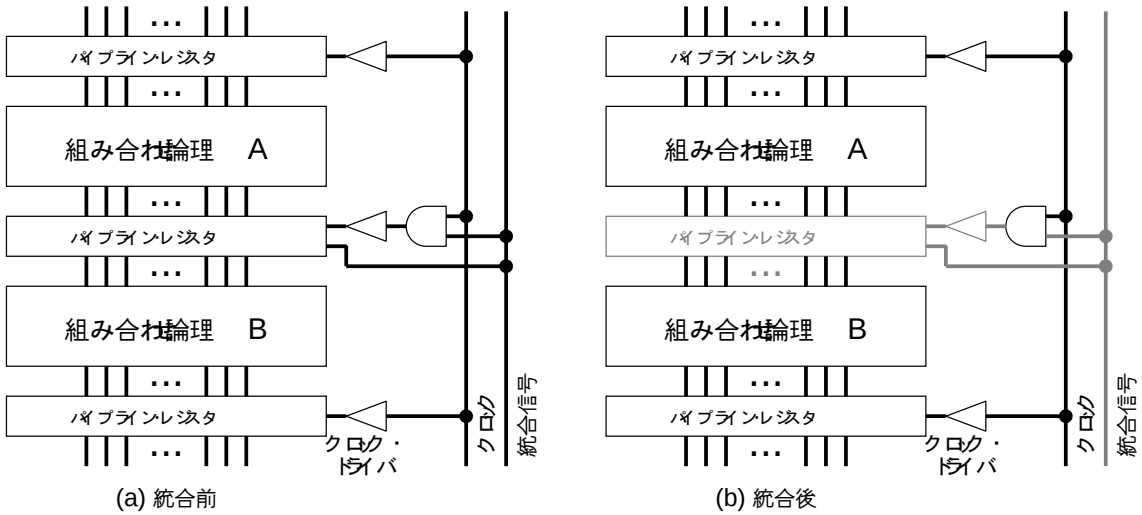

図 1: クロック分配線の変更と統合信号の追加

を検出する仕組みを用いて，類似フェーズを実行中にフェーズの EDP を最小化する統合度を見つけ出し，プログラムを実行する方法を 2 つ述べている．この方法では，一定長の命令インターバル中は，統合度を変えずに制御するため，インターバル中にプログラムの実行特性が変化してもすぐには対応できない．これに対して，本論文では，プログラム実行中の分岐予測ミス率に応じて，PSU の統合度の制御を行う方法を提案する．上の文献 [6] で述べた方法では，統合度を切替えるために，次の最適な統合度を決定するためにサンプリングする必要があるが，本論文で提案する方法では不要であり，さらに，ハードウェアも比較的単純でコストが小さいという特徴がある．

残りの論文の構成は，以下の通りである．まず 2 章で PSU の仕組みについて概観し，関連研究としてシグニチャを用いて PSU の動的適用を行う方法について説明する．次の 3 章では，本論文で提案する分岐予測ミス率をトリガとした PSU の動的適用について述べ、4 章で評価結果を示し、5 章でこの論文のまとめとする．

第 2 章　研究背景

2.1　PSU の仕組み

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．

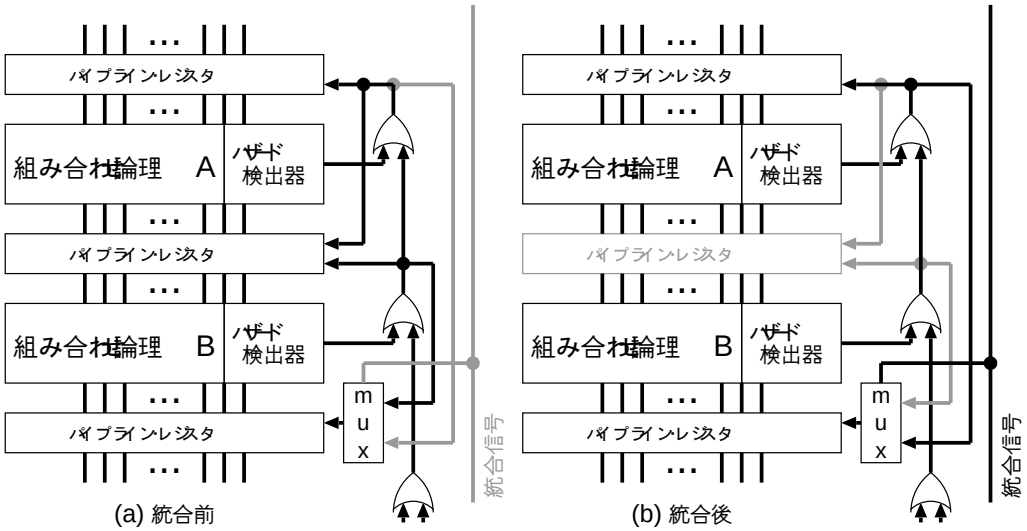

図 2: インターロック回路

2.2 PSU による消費エネルギー削減量の解析

本節では，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：

$$E = P \times T_{ex} \tag{1}$$

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：

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

$$T_{ex} = \frac{N}{IPC \times f} \tag{3}$$

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．

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：

$$P_{PSU}(f_{low}, U) = \left(P_{total} - P_{clock} + \frac{P_{clock}}{U}\right) \times \frac{f_{low}}{f_{max}} \tag{4}$$

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 (1) 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：

$$E_{PSU}, n(f_{low}, U) = \frac{P_{PSU}(f_{low}, U) \times T_{ex}(f_{low}, U)}{P_{total} \times T_{ex}(f_{max}, 1)} \tag{5}$$

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 (3) と (4) を式 (5) に代入することにより，以下の式を得ることができる：

$$E_{PSU}, n(f_{low}, U) = \frac{IPC_{max}}{IPC_{low}} \times \left\{1 - k \times \left(1 - \frac{1}{U}\right)\right\} \tag{6}$$

ただし，

$$k = \frac{P_{clock}}{P_{total}} \tag{7}$$

である．

式 (6) 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%[7]），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．

2.3 フェーズ検出器を用いた PSU の制御

2.1 節で述べた PSU を用いて効果的な EDP(エネルギー遅延積) を実現するために，PSU の制御方法の研究が行われている．

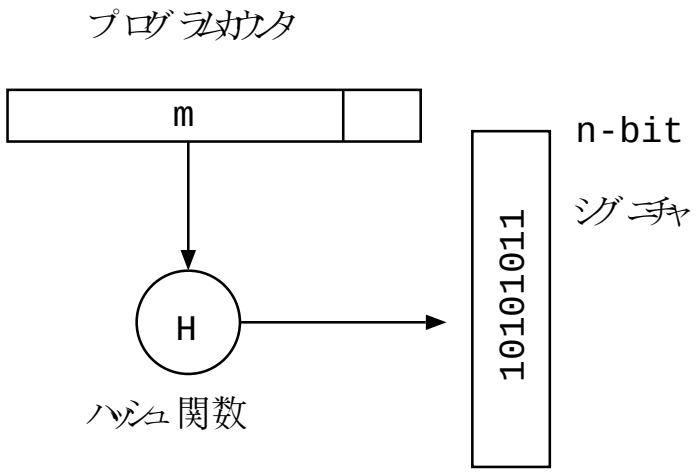

図 3: シグニチャ

文献 [4, 5] では，プログラムは，キャッシュ・ミス率，IPC，エネルギー消費などの点からいくつかの類似したフェーズに分割されると述べている．これに基づき文献 [6] では過去に実行したフェーズと類似したフェーズでは，過去の学習結果より得られた最適な PSU の統合度を割当て，プログラム実行完了までの EDP を最小化する制御を行っている．プログラムのフェーズの検出には，シグニチャと呼ばれるベクトルが用いられている．次の 2.3.1 節と 2.3.2 節では，フェーズ検出器を用いた PSU の統合度の制御方法 2 つを述べる．

2.3.1 フェーズ検出器のみを用いた統合度の切替え

文献 [4, 5] では，プログラムの実行を 100K 命令のインターバル単位で分割してみた場合，キャッシュ・ミス，IPC，エネルギー消費の点から類似の特徴を示すフェーズに分割されることが示されている．フェーズは類似性を持つ命令インターバルが集まったものである．各インターバル間の類似性は，図 3 に示すシグニチャという n ビット・ベクトルを用いて比較される．各インターバルの最初でシグニチャはクリアされ，インターバルの実行中は，命令アクセスが行われるときに，プログラムカウンタの中から m ビットを選び，ハッシュ関数を通して，n ビット・シグニチャ中の 1 ビットをセットする．なお，シグニチャ S_1 と S_2 の比較は，

$$\delta = \frac{number\_of\_"1"bit\_in(S_1 \oplus S_2)}{number\_of\_"1"bit\_in(S_1 + S_2)} \tag{8}$$

によって定められたシグニチャ間の距離 δ が，閾値より大きな場合に異なると

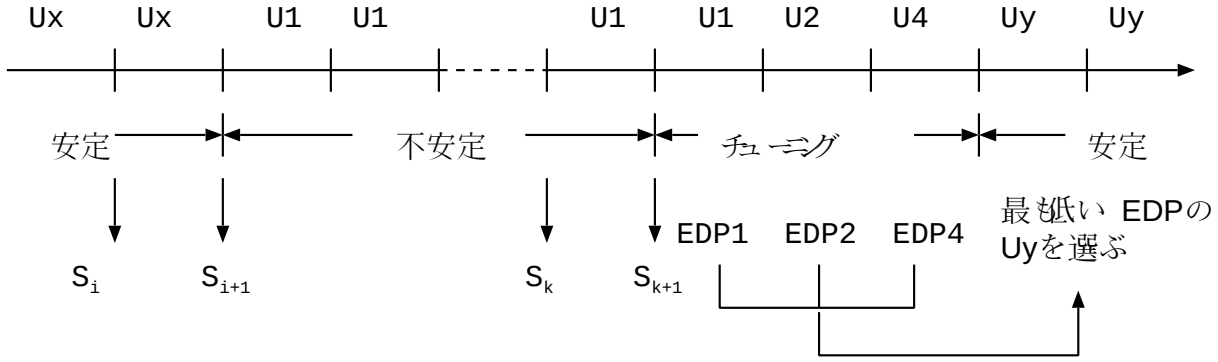

図 4: シグニチャを用いた PSU 統合度の制御

されている．文献 [4] では，閾値は 0.5，シグニチャの大きさ n は 1024，命令インターバルは 100K としている．なお，$number\_of\_"1"bit\_in()$ は，シグニチャ中の 1 の数を表したものである．

このシグニチャ比較を用いると，プログラムは類似のインターバルが集まった．フェーズに分割されることになる．フェーズが類似していれば，エネルギー消費の点からも類似する．そこで文献 [6] では，類似したフェーズでは同じ PSU の統合度で実行し，異なるフェーズを検出したときに，統合度を変更することを考える．以下に，PSU の統合度の制御を述べる．

プログラム実行中のフェーズの状態と PSU の統合度の制御を図 4 に示す．隣り合うインターバル間のシグニチャが等しい間は，統合度 Ux の安定状態で実行している．隣接のシグニチャが異なるインターバルに変わると，不安定状態に移り，その後の 3 つのインターバルで統合度 1，統合度 2，統合度 4 に切替える．そのうちの最適な EDP を持つ統合度 Uy を選択して，次の安定状態へ移る．

この方法では，インターバル間の比較に用いるシグニチャを求める回路，および，シグニチャを比較する回路をハードウェアとして必要とする．

2.3.2 フェーズ検出器と履歴テーブルを用いた統合度の切替え

2.3.1 節では，プログラム実行中に異なるフェーズが現れる度に，統合度 1，統合度 2，統合度 4 に切替えて，次の統合度に最適な統合度を選択していた．異

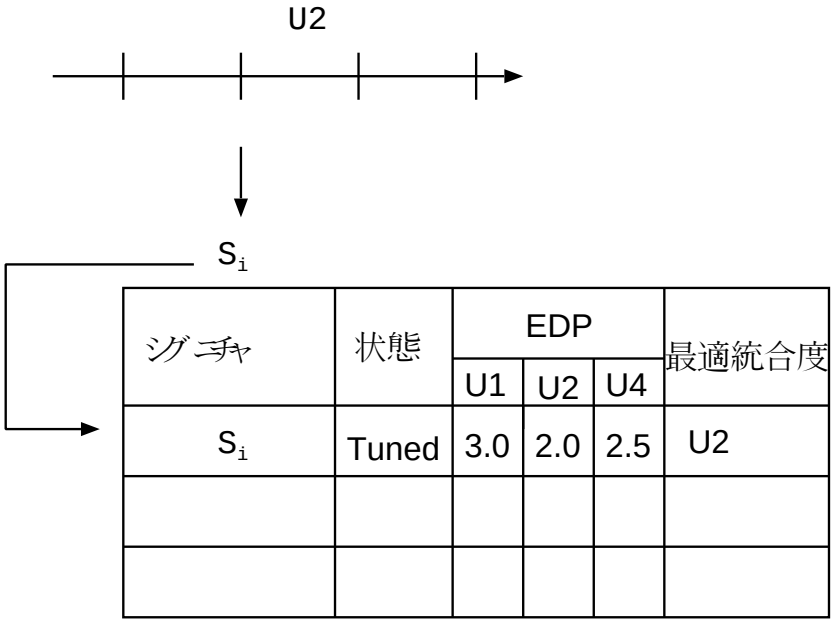

図 5: 履歴テーブルを用いた PSU 統合度の制御

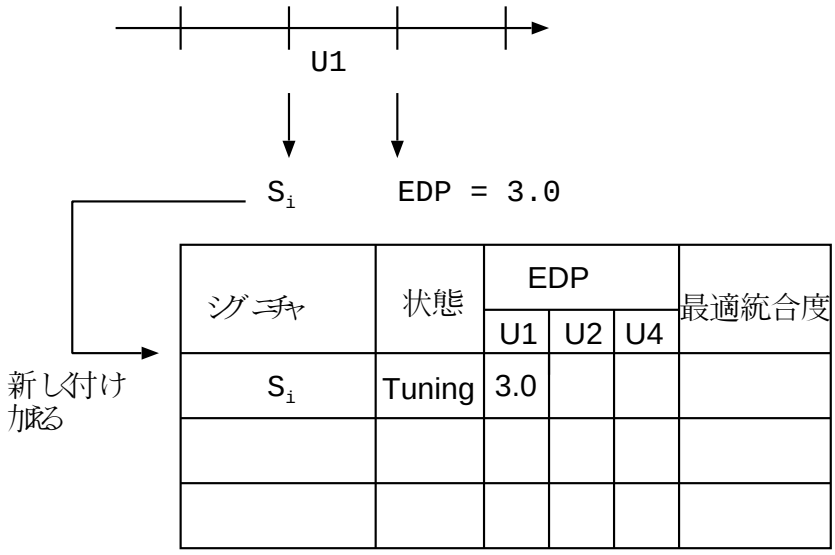

図 6: 履歴テーブルを用いた PSU 統合度の制御 2

なる統合度が現れる度に，統合度を最適化していたのでは，効率が悪い．そこで，プログラム実行中に類似のフェーズが繰り返し起こることを利用して，過去に現れたインターバルと類似のインターバルが現れた場合に，履歴テーブルにある最適な統合度の情報を用いて，次のインターバルの統合度を予測することにする．

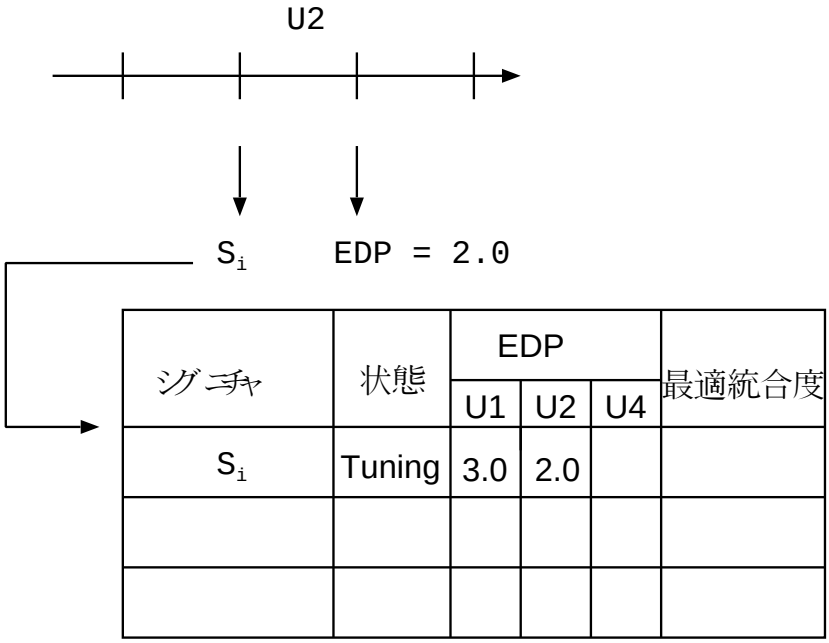

図 7: 履歴テーブルを用いた PSU 統合度の制御 3

プログラム実行中の PSU の統合度の制御を図 5，6，7 に示す．2.3.1 節同様に命令インターバル単位で統合度を割り当てて制御を行う．この方法では，インターバルのシグニチャS_iを得ると，過去の履歴テーブル中のシグニチャの中に類似したものがあるかどうか調べる．類似したシグニチャがある時をヒット，ないときをミスと呼ぶ．以下，3 つの場合がある．図 5 の場合，つまり，ヒットした場合，そのエントリの持つ最適統合度を次のインターバルの統合度として選択している．次に図 6 の場合，ヒットしない場合，現在のシグニチャを履歴テーブルに新たに付け加えて，次のインターバルは統合度 1 として実行し，付け加えたシグニチャはチューニング状態とする．最後に図 7 の場合，ヒットはしたが，エントリがチューニング状態であった場合は，次のインターバルは，まだ調べていない統合度を選択して実行する．

この方法では，2.3.1 節で必要となる回路以外に，履歴テーブルをハードウェアとして必要とする．

第 3 章　分岐予測ミス率を用いた PSU の動的適用

2.3 節で述べた切替え方法では．例えば 100K 命令という命令インターバル単位で統合度を切替えているため，インターバル中にプログラムの実行特性が変

化しても，統合度の変更を行わずに実行を進めている．短いインターバルに変更して，統合度の切替えを行うことによってこの問題を解決できるかもしれないが，異なるシグニチャが増えすぎたり，シグニチャを比較する回路が増加して，消費電力が増加する恐れが出てくる．インターバル中の統合度を一定にしたままでは，分岐予測ミスなどでプロセッサの CPI が上がった時に，実行時間が増加し，EDP の点から考えて望ましくない．文献 [2] より，IPC の変化に大きく影響する要素として，分岐予測ミス・ペナルティを減少させることが示されている．そこで，実行中の分岐予測ミス率の変化に対応して IPC を向上させるために，分岐予測ミス率をトリガとして，統合度上げて IPC を向上させることを考える．

本章では，分岐予測ミス率に応じて，統合度を切替えた場合の EDP を最小化する方法について述べる．3.1 節では，統合度を切替える方法について，3.2 節では，消費電力の評価値をとして用いる EDP を抑えるには，どの程度の分岐予測ミス率で統合度を切替えるとよいのかを解析的に見積もる．さらに，3.3 節では，分岐予測ミス率を計算するハードウェアについて，3.4 節では 2.3 節で述べたフェーズ検出器を用いた PSU 統合度の制御方法に対する，分岐予測ミス率を用いた統合度制御方法の利点を述べる．

3.1 分岐予測ミス率を用いた PSU 統合度の切替え方法

プログラム実行中に分岐予測ミスが起こると，パイプラインにある命令をすべてフラッシュして，再び命令でパイプラインを満たす時間が必要になる．この時にパイプラインが長ければ，命令でパイプラインを満たすための時間がかかり，CPI が増加し，実行時間が長くなり，EDP の増加につながる．この分岐予測ミス率による EDP の増加は，パイプラインが短いほど小さくなる．なぜなら，パイプラインが短い方が命令でパイプラインをを満たすために必要な時間が少なくなるからである．したがって，分岐予測ミス率が頻発する時に PSU でパイプラインを短くすれば，EDP の増加は最小限に抑えられると考えられる．

そこで，直近の分岐予測ミス率の値に応じて，新たな統合度に変更する方法を本研究では提案する．プログラム実行中の統合度の制御を図 8 に示す．実行中，毎サイクル，直近の分岐予測ミス率の値に応じた分岐予測ミス率を更新するカウンタを準備し，更新した分岐予測ミス率の値に応じた統合度に切替えて，次のサイクルへと移る．更新するために，統合度 U1 と U2，U2 と U4 を切替え

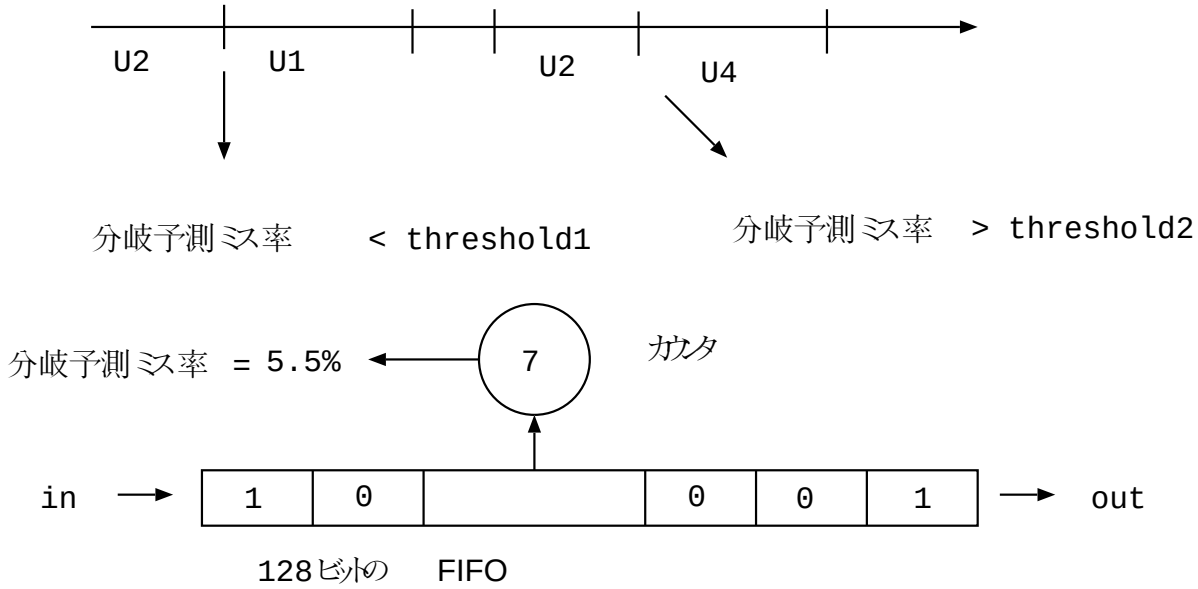

図 8: 分岐予測ミス率をトリガとした統合度の制御

るための閾値 threshold1，threshold2 を設ける．

分岐予測ミス率が

- threshold1 未満であれば統合度 1
- threshold1 以上，threshold2 以下であれば統合度 2
- threshold2 より大きければ統合度 4

に統合度を切替えることにする．なお，現在のプログラムの実行特性を速やかに統合度の選択に反映させるため，直近の 128 個の分岐予測の結果を用いて計算している．分岐予測の結果は，FIFO に蓄え，直近の分岐予測の結果のみを使用できるようになっている．この FIFO については 3.3 節で述べる．

3.2 閾値の解析

3.1 節で述べた統合度の切替えを用いて，分岐予測ミス率がどんな値の時に統合度を切替えると，消費電力を抑えることができるかを解析的に見積もる．

消費電力の評価値として，EDP(エネルギー遅延積) を用いる．EDP は以下の式で表される．

$$EDP = P \times T_{ex} \times T_{ex} \tag{9}$$

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

れる．

$$P = a \times f \times C \times V^2 \times \left(1 - \frac{U-1}{U} \times m\right) \tag{10}$$

$$T_{ex} = \frac{N}{IPC \times f} \tag{11}$$

ここで，f はクロック周波数，C はスイッチするノードの全容量，a はアクティビティ・ファクタ，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．

また，式 (10) に示された PSU の消費電力の計算方法は以下の通りである．パイプライン・ステージを統合すると，パイプライン・レジスタへのクロック分配が抑制され，クロック・ネットワークの消費電力が減少する．統合度 U において停止するパイプライン・レジスタはの割合は $(U-1)/U$ である．m をプロセッサの全消費電力に対するクロック・ネットワークの消費電力とすると，統合度 U で動作する PSU の消費電力は式 (10) のように表される．

式 (10)、(11) を (9) に代入し，IPC の逆数である CPI に置き換えると，

$$EDP = a \times C \times V^2 \times \left(1 - \frac{U-1}{U} \times m\right) \times N^2 \times CPI^2 \times \frac{1}{f} \tag{12}$$

と表される．

さらに，CPI については，分岐予測ミスによる CPI の増分を考えると，

$$CPI = CPI_{ideal} + P_{misprediction} \times penalty_{misprediction} \times F_{branch} \tag{13}$$

と表される．

ここで，分岐予測ミス，キャッシュ・ミスがない理想状態での CPI を CPI_{ideal} とし，$P_{misprediction}$ は分岐予測ミス率，$penalty_{misprediction}$ は分岐予測ミス・ペナルティ，F_{branch} は分岐の出現頻度とする．なお，本論文では，CPI に影響を与える要因として，分岐予測ミス率のみを考え，L1，L2 キャッシュなどの影響を除いた状態を仮定している．

分岐予測ミス率の変化に応じて，統合度 U1，U2，U4 のどれを選べば，EDP が小さく抑えられるかを述べる．分岐予測ミス率が小さい場合には，(13) 式を見ると，どの統合度でも CPI の増加が少ない．したがって，式 (12) よりクロック周波数の高い U1 の方が，U2 や U4 より遅延時間が小さく，EDP 値を抑える

ことができると考えられる．

一方，分岐予測ミス率が大きくなると，式 (13) より，分岐予測ミス・ペナルティが U2 や U4 より大きい U1 は CPI が増加する．つまり，式 (12) の CPI 値が増加して，U1 の EDP 値が U2 や U4 の EDP 値より大きくなる．したがって，分岐予測ミス率が大きくなるにしたがって，U1 を U2 や U4 に切替えた方が，EDP 値を小さく抑えることができると考えられる．

分岐予測ミス率 EDP の関係を統合度別に表したグラフを，図 9 に示す．横軸が分岐予測ミス率で，縦軸が EDP 比である．分岐予測ミス率が増えるにつれ，EDP を小さくする統合度は，U1 から U2，U2 から U4 へ変わることがわかる．分岐予測ミス率の変化に応じて，統合度 U1，U2，U4 から適切なものを選んで EDP 値を小さく抑えたい．そこで，U1 の EDP 値が U2 の EDP 値を越えるときの分岐予測ミス率と，U2 の EDP 値が U4 の EDP を越える時のミス率を求める．まず，統合度 U1 と U2 における EDP 値が等しくなる時は式 (12) より，

$$\begin{aligned} & a \times C \times V^2 \times \left(1 - \frac{1-1}{1} \times m\right) \times N^2 \times CPI_1^2 \times \frac{1}{f_1} \\ = & a \times C \times V^2 \times \left(1 - \frac{2-1}{2} \times m\right) \times N^2 \times CPI_2^2 \times \frac{1}{f_2} \end{aligned} \tag{14}$$

ここで，a，C，V，N は両辺で同じ値としている．m は文献と同様に 0.3 と仮定した．CPI_1，f_1 は統合度 1 の時，CPI_2，f_2 は統合度 2 の時の CPI とクロック周波数である．$f_1 = 1$ とすると，$f_2 = 0.5$ である．

式 (14) に値を代入して整理すると

$$CPI_1 = 1.3 \times CPI_2 \tag{15}$$

となり，式 (13) を代入すると，

$$\begin{aligned} & CPI_{ideal} + P_{misprediction} \times penalty_{U1} \times F_{branch} \\ = & 1.3 \times (CPI_{ideal} + P_{misprediction} \times penalty_{U2} \times F_{branch}) \end{aligned} \tag{16}$$

ここで，[8] より F_{branch} は 0.2 と仮定する．$penalty_{U1}$、$penalty_{U12}$，$penalty_{U4}$ は，統合度 U1，U2，U4 のときの分岐予測ミス・ペナルティであり，それぞれ 20，10，5 とする．

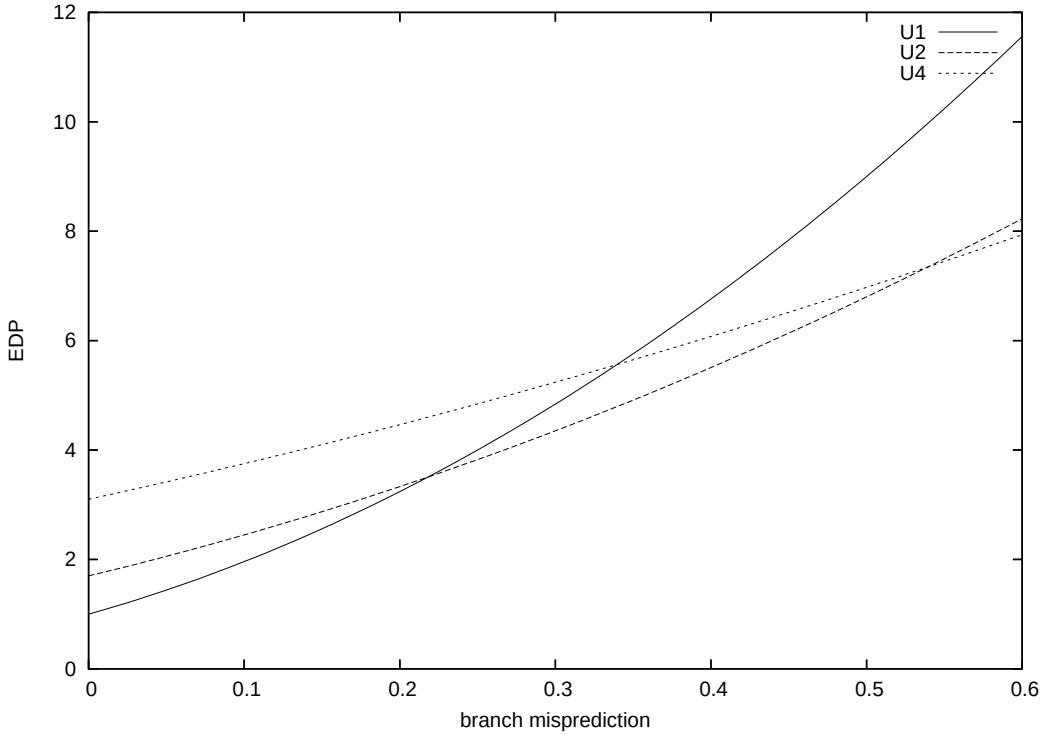

図 9: 分岐予測ミス率と EDP の関係

したがって式 (16) から U1 の EDP が U2 の EDP を越える時の分岐予測ミス率は

$$P_{misprediction} = \frac{3}{14} \times CPI_{ideal} \tag{17}$$

となる．統合度 U2 と U4 における EDP 値が等しくなる場合も同様の手順で求めると，

$$P_{misprediction} = \frac{7}{13} \times CPI_{ideal} \tag{18}$$

を得る．

以上のような考察から，分岐予測ミス率が (17)，(18) 式の値のときに，統合度を切替えると，EDP 値が小さく抑えることができることがわかる．

仮に理想 CPI の値が，1，0.5，0.33，0.25 の 4 つの場合の時の分岐予測ミス率の見積り値はおよそ表 1 のような値になる．ただし，式 (17) で求めた値が $P1_{misprediction}$ であり，式 (18) で求めた値が $P2_{misprediction}$ である．

表 1 のデータ値は，統合度を切替える閾値 threshold1，threshold2 の値が表 1

表 1: CPI_{ideal} に対する閾値 (%)

CPI_{ideal}	0.25	0.33	0.5	1
$P1_{misprediction}$	5	7	11	21
$P2_{misprediction}$	13	18	27	54

の周辺の値をとるときに，EDP 値がどのような値をとるか調べるために用いる．

3.3 分岐予測ミス率を用いた PSU 統合度切替えのハードウェア

3.1 節で述べた方法で，分岐予測ミス率は直近の 128 個の履歴を用いていた．この履歴を FIFO で保持する．ただし、ミス率を必要とする度に、FIFO の中身を集計することは，遅延時間の点から好ましくないと考え，ミスした数を集計するカウンタを用意する．カウンタは 7bit とする．FIFO は入力される値 (in) と，出力される値 (out) で更新される．カウンタの更新は次の通りになる．なお，分岐予測がヒットした場合を 1，分岐予測ミスした場合を 0 とする．

- in=0，out=0 あるいは in=1，out=1 の場合は，カウンタは変わらない．
- in=0，out=1 の場合は，カウンタはデクリメントされる．
- in=1，out=0 の場合は，カウンタはインクリメントされる．

FIFO とカウンタで構成されるため，シグニチャを用いる方法より，ハードウェア・コストは少ない．

3.4 分岐予測ミス率を用いた PSU 統合度切替えの利点

2.3 節で述べたフェーズ検出を用いた PSU の制御方法に対して，分岐予測ミス率を用いた PSU の制御方法は次の 2 つの利点がある．

- 3.3 節で述べたように，分岐予測ミス率を用いた PSU 統合度制御は，128bit の FIFO と，7bit のカウンタを用いており，ハードウェア・コストが小さい．一方，2.3 節のフェーズ検出器を用いた制御方法では，1024bit のシグニチャを求める回路やシグニチャを比較する回路，および，履歴テーブルを用いているため，ハードウェア・コストが増加する．
- フェーズ検出を用いた方法では，共に PSU の統合度の切替えを命令インターバル単位で行っている．このため，分岐予測ミスなどの要因で，CPI

が増加して，実行時間が長くなり，消費電力が増加した場合にも，同じ統合度で実行し続けることになる．一方，分岐予測ミス率を用いる方法では，固定長ではなく，分岐予測ミス率の変化に対応して，可変長のインターバルで統合度を切替えることができる．また，フェーズ検出を用いた方法のように次の最適統合度を決定するためのサンプリングを行う必要はない．

第4章　評価

4.1　評価環境

4.1.1　シミュレーション環境

SimpleScalar Tool Set[9] 中の out-of-order 実行シミュレータに 3.1 節で述べた PSU の制御方法を実装して，EDP を測定した．命令セットは SimpleScalarPISA である．表 2 に示すようにベンチマーク・プログラムとして，SPECint95 の 8 本を用いた．

表 3 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．さらに，3.2 節の式 (13) に示したように，CPI への影響は，キャッシュなどを除いて，単純化して分岐予測ミスによる影響のみを考慮している．シミュレーションでは，キャッシュ，メモリ，TLB を完全にして測定を行うことにする．

表 2: ベンチマーク

ベンチマーク	入力	実行命令数
compress95	bigtest.in	95M
gcc	genoutput.i	84M
go	2stone9.in	75M
ijpeg	specmun.ppm	450M
li	train.lsp	183M
m88ksim	ctl.in	100M
perl	scrabbl.in	80M
vortex	vortex.in	80M

表 3: プロセッサの構成

発行幅		8
RUU		64 エントリ
LSQ		32 エントリ
メモリポート		8
機能ユニット	整数 ALU	8
	整数乗除算	4
	浮動小数 ALU	8
	浮動小数乗除算	4
分岐予測	予測方式	gshare
	履歴	6 ビット
	インデックス	13 ビット
	BTB	2048 エントリ/4-way
	RAS	16 エントリ
キャッシュ		完全 (100%ヒット)

表 4: PSU において仮定する実行レイテンシ，分岐予測ミス・ペナルティ

統合度		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
分岐予測ミス・ペナルティ		20	10	5

4.1.2 パイプラインの仮定

図 10，11，12 に統合度 1，2，4 のパイプラインを示す．表 4 に，これらのパイプラインにおける命令の実行レイテンシ，分岐予測ペナルティを示す．統合度 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，整数/浮動小数点除算と平方根演算については，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．

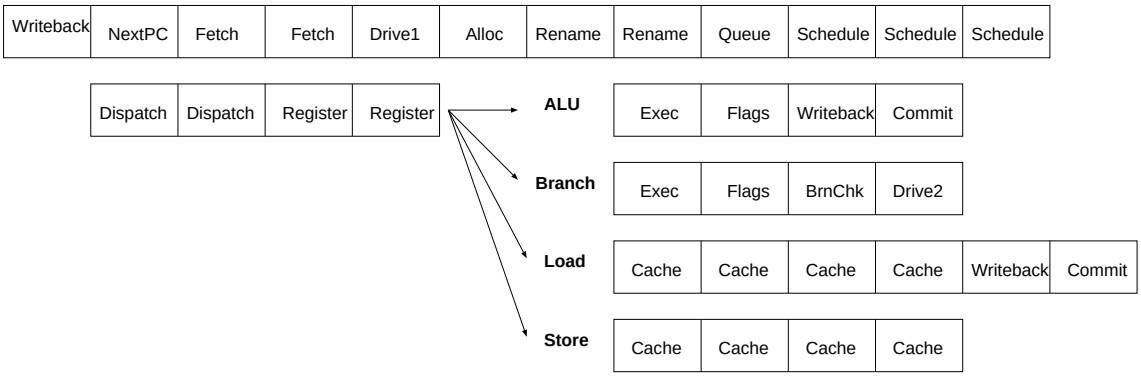

図 10: ベース

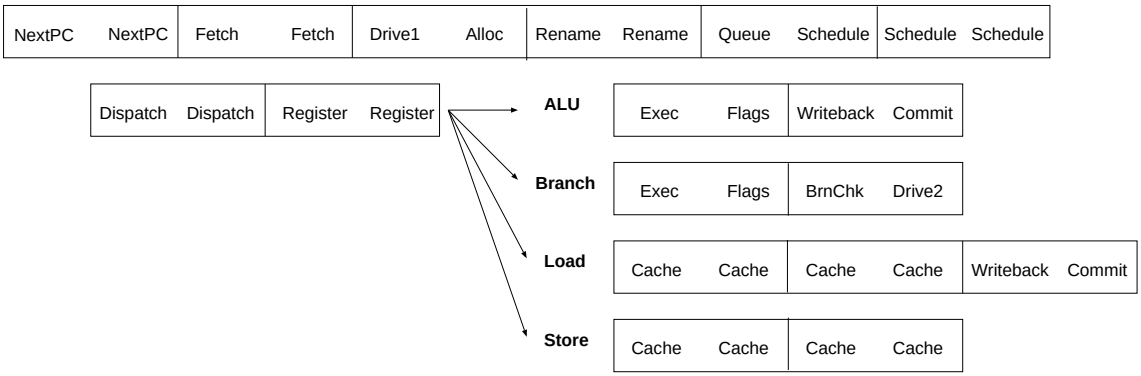

図 11: 統合度 2

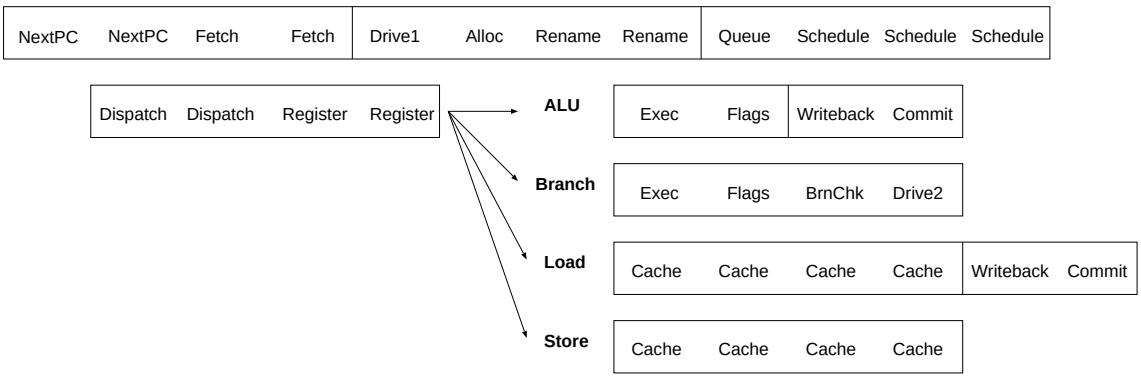

図 12: 統合度 4

4.1.3　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 [7, 10, 11, 12] によれば，その範囲は 18%〜40%である．そこで，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 [10]）．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．

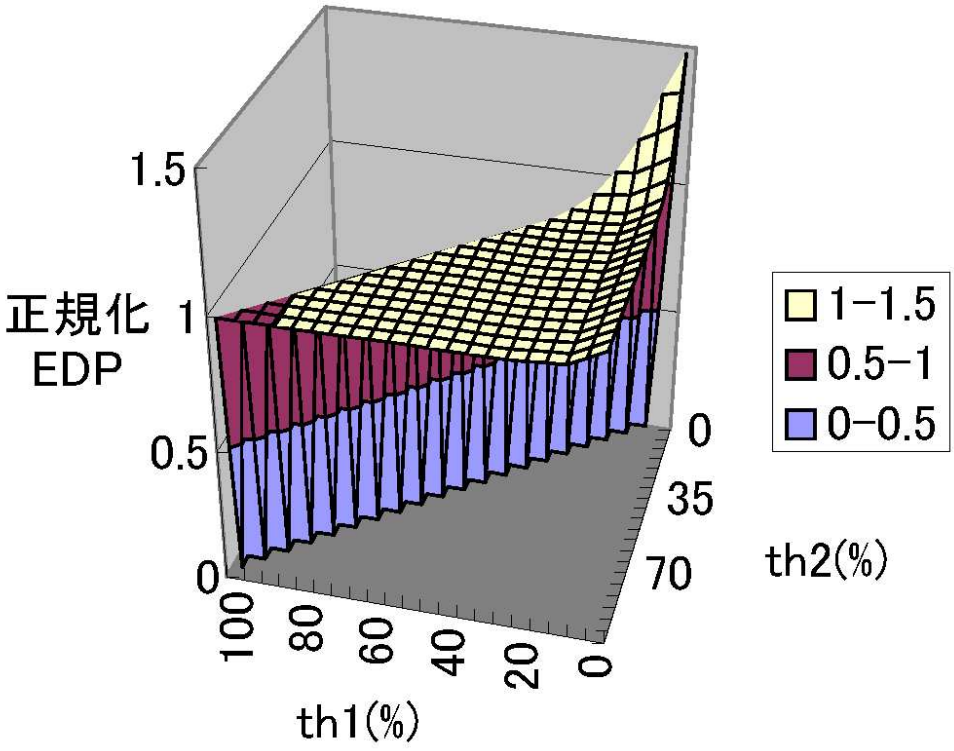

図 13: 閾値 t1,t2 を変化させたときの各ベンチマークの平均 EDP

4.2 評価結果

まず，SPECint95 の各ベンチマークについて，統合度 1，2，4 に固定して，EDP を測定した．次に，各ベンチマーク・プログラムについて，3 章で分岐予測ミス率に対して統合度を切替えるために定めた閾値 threshold1 を 0%から 100%まで，threshold2 を threshold1%から 100%まで 5%幅で変化させたときの，EDP の値を測定した．各ベンチマークのデータを平均してグラフ化したものが，図 13 となる．

グラフの z 軸は統合度 1 のときの EDP で正規化した EDP を表している．3.1 節で述べたように，分岐予測ミス率に対して，閾値 threshold1，threshold2 を設けている．threshold1 で，PSU の統合度 1 と 2 を切替え，threshold2 で，PSU の統合度 2 と 4 を切替えることになる．図 13 でグラフの軸で，x 軸方向の th1 は，統合度切替えのために変化させる閾値 threshold1 であり，y 軸方向の th2 は，閾値 thresholld2 を表している．z 軸の表すものは，閾値 th1，th2 がある値に設定された場合に，統合度を切替えて実行したときの，正規化を表す．y 軸の方向は通常と逆の方向からグラフを示している．

なお，平均化は，各ベンチマークを分岐予測ミス率により統合度を切替える方法で実行した場合の EDP を，同じベンチマークを統合度 1 で実行した場合の EDP で正規化することによって行う．

図 13 中で，(th1,th2)=(0,0)，(0,100)，(100,100) の付近は，ぞれぞれプログラムを統合度 4，2，1 で固定して実行した場合と同じことになる．例えば，(th1,th2)=(0,0) 付近では，閾値がどちらも 0 であるため，実行中に分岐予測ミスが 0 以上になる場合に統合度 4 で実行する．これは，常に統合度 4 で実行することと同じである．

次に測定結果の図 13 について特徴を述べる．3.2 節で，解析によって求めた EDP 値を低くするための概算値である，$P1_{misprediction}$，$P2_{misprediction}$ の周辺では，統合度が小さくなる傾向は見られた．th1 を凡その解析値である $P1_{misprediction}$ に近づけると，つまり，10%〜20%の付近に近づけると EDP が低くなる傾向は見られる．さらに th1 を大きくしたところ，EDP の値にはほとんど変化が見られない．また，th2 についても，20%周辺で EDP が低くなる傾向が見られるが，それ以上大きく th2 を変えてもほとんど変化が見られない．

また，前に述べたように，(th1,th2)=(0,0)，(0,100)，(100,100) 付近では，そ

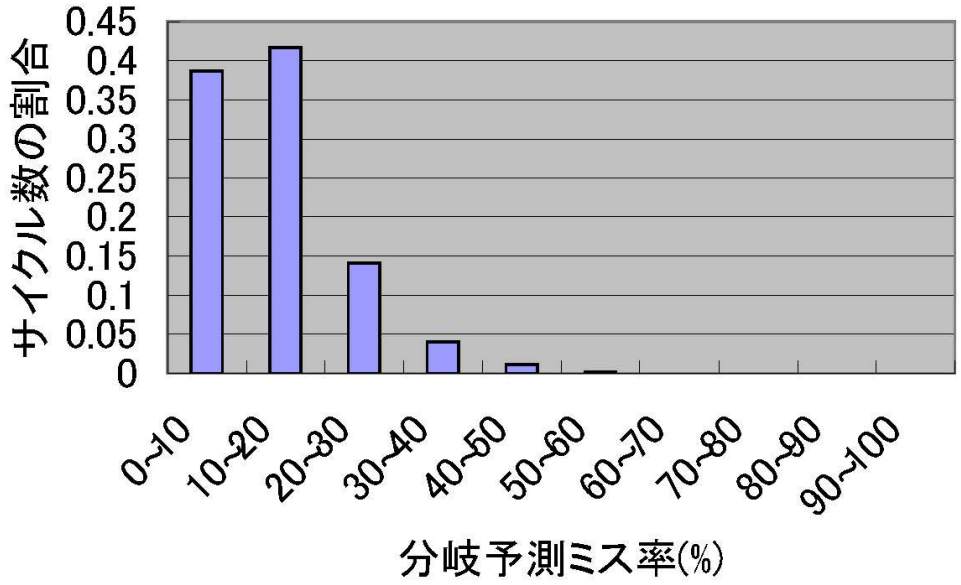

図 14: 各サイクルにおいてカウンタが示した分岐予測ミス率の分布

れぞれ統合度 4，2，1 固定で測定を行うことであるが，統合度 1 で固定した場合に低い EDP をとり，統合度 2，4 で固定した場合は比較的高い EDP を示した．統合度 1，2，4 の順に EDP 比の値は，1，1.08，1.36 であった．解析値の EDP 値が 1.01 周辺であったため，統合度 1 の場合よりも，解析値周辺の EDP が高くなっている．

図 13 のグラフの結果で EDP 値が変化しない部分が多い点をベンチマークの分岐予測ミス率の分布を平均した図 14 を用いて説明する．

SPECint95 の 8 本のベンチマーク平均の分岐予測ミス率の分布を図 14 に示す．分岐予測ミス率の範囲を，10%単位で 10 個の範囲に分割する．各ベンチマーク・プログラムを実行した場合の，各範囲の分岐予測ミス率のサイクル数の分布を測定した．図 14 は，各ベンチマークの分布の値を平均した値をグラフ化したものである．なお，簡単のため，統合度 1 に固定して実行した場合のプロセッサで測定を行った．

横軸は，分岐予測ミス率を 10%単位で分割した範囲を示している．単位は%である．縦軸が，該当範囲の分岐予測ミス率が，実行中にどの程度の割合あったかをベンチマークで平均したものである．図 14 を見ると，平均的に分岐予測ミス率が，低い値に固まって分布していることがわかる．例えば，0%～20%が約 80%を占めている．分岐予測ミス率が 20%以上は，15%と減少し，30%の分岐予

測ミス率はほとんどなくなる．

3.2 節の解析で，分岐予測ミス率が高くなるにつれ，統合度 1 から 2 へ，2 から 4 に切替える方が，EDP を低くするために適切であると述べた．なぜなら，分岐予測ミス率が大きくなるにつれて，分岐予測ミス・ペナルティの大きな統合度 1 では，実行サイクル数が増加して，EDP が増加する．そのため分岐予測ミス・ペナルティの小さな統合度 2 や 4 へ切替えることで，実行サイクル数を抑え，EDP を抑えることができるからである．

図 14 からわかるように，分岐予測ミス率は低い値に分布しているため，SPECint95 のベンチマークについて，統合度 1 で実行するのが適切だが，統合度 2 や 4 に切替えても EDP を抑えるには，効果的でないと考えられる．

以下，図 14 を用いて，図 13 を説明する．上で，SPECint95 の 8 本のプログラムは平均して，低い値に分岐予測ミス率が分布しているために，統合度 1 が適切で，統合度 2，4 が不適切だと述べた。このため図 13 中で，閾値 (th1,th2)=(0,0)，(0,100) のとき，つまり統合度 4，2 のときは，EDP 値は比較的大きくなる．また，閾値 (th1,th2)=(100,100) ，つまり統合度 1 のときは EDP が小さくなる．

また，図 13 で閾値 th1，th2 を変化させたときの値についても述べる．図 13 中で閾値 th1 を，15%周辺から解析値の 20%周辺に上げて，統合度 2 で実行するサイクル数を減らし，統合度 1 で実行するサイクル数を増加させると，EDP が削減されることから，解析値周辺に閾値を設定することで，EDP が小さく抑えられる傾向は若干見受けられる．これは，図 14 で分岐予測ミス率が 10%付近に多いために，閾値 t1 を 20%に上げると，10%付近の分岐予測ミス率のサイクルが適切な統合度 1 で実行されるために，EDP が低くなったと考えられる．

図 13 で閾値 th1 をさらに大きくしたところでは，EDP にほとんど変化はない．図 14 からわかるように，大きな分岐予測ミス率の割合はそもそも少ないため，どのような統合度で実行しても効果がないと考えられる．

また，th2 について，20%を越えた辺りから閾値を変化させてもほとんど EDP に影響がない．これについても 20%以上の分岐予測ミス率の命令はほとんどないために，統合度を変えても，ほとんど影響がないためと考えられる．

第5章　まとめ

プロセッサの消費電力を削減するために提案されているPSUは，クロック周波数を低下させ，パイプライン・レジスタをバイパスさせることによって複数のパイプライン・レジスタを統合する方法である．これにより，パイプラインが短くなり，プログラムの実行に必要なサイクルが削減され電力消費の時間を短くすることができる．

PSUを用いて効果的なEDPを実現するために，PSUの制御方法の研究が行われている．シグニチャという類似したフェーズを検出する仕組みを用いて，類似フェーズに対して，実行中にフェーズのEDPを最小化する統合度を見つけ出し，プログラムを実行する方法がある．しかし，この方法は一定長の命令インターバル中は，統合度を一定にしたままであり，分岐予測ミスなどでプロセッサのCPIが上がった時に，実行サイクル数が増加し，EDPの点から考えて望ましくない．

そこで，本論文では，プログラムの実行特性の変化の検出に分岐予測ミス率の変化を利用し，分岐予測ミス率の変化に応じて統合度を変更することを提案する．これにより，可変長のインターバルで統合度を変化させることやハードウェア・コストの削減が達成できる．この方法を用いて評価値のEDPを抑えるにはどの程度の分岐予測ミス率で統合度を切替えるとよいのかを解析的に見積もった上で，実際のEDPを評価した．

測定の結果，分岐予測ミス率を用いて統合度を切替える場合，解析によって想定した分岐予測ミス率の閾値，10%～20%周辺でEDPが低くなる傾向があることがわかった．しかし，平均値において，統合度を固定した場合と比較すると，統合度1の場合がEDPを低く抑えていることがわかった．理由として，ベンチマークのプログラムの分岐予測ミス率が小さな値に多く分布しているために，統合度を低く設定した方がEDPを削減できる傾向にあるからである．

過去の研究においては，平均すると統合度2以上の方がEDPを削減できるという結果になっていたが，今回の結果は異なっていた．これは，今回の統合度切り替えを分岐予測ミス率のみをトリガとし，なおかつ，不確定要素を削減するために完全なキャッシュを仮定したからだと考える．過去の研究では，PSUによるキャッシュ・ヒット・レイテンシの削減もPSUによる消費エネルギー削減に寄与していることが示されている．そのため、今後の研究においては，現

実的なキャッシュを仮定し，キャッシュ・ミス率等も考慮した統合度の選択を行う必要があると考えられる．

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，御指摘を賜わった福井大学の森眞一郎助教授，奈良先端科学技術大学院大学の中島康彦助教授，京都大学大学院の嶋田創助手，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] 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).

[4] A.S Dhodapkar and J.E Smith, “Managing Multi-Configuration Hardware via Dinamic Working Set Analysis”,29th Annual International Symposium on Computer Architecture,pp.233-244(2002).

[5] T.Sherwood,E.Perelman,G.Hamerly,S.Sair and B.Calder, “Discovering and Exploiting Program Phases”,IEEE Micro, Vol.23,No.6,pp.84-93(2003).

[6] J. Yao, H. Shimada, Y. Nakashima, S. Mori, and S. Tomita, “Program Phase Detection Based Dynamic Control Mechanisms for Pipeline Stage Unification Adoption,”1st International Workshop on Advanced Low Power Systems, pp. 39-46,(2006).

[7] M.K. Gowan, L.L. Biro and D.B. Jackson, “Power Considerations in the Design of the Alpha 21264 Microprocessor,” In proceedings the 35th De-

sign Automation Conference, pp.726–731 (1998).

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

[9] 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).

[10] 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).

[11] 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).

[12] 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).