社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－ARC－169 (11)
2006／8／1

新しいクリティカルパス判定基準を用いたクリティカルパス予測器の評価

千代延 昭宏† 佐藤 寿倫††

近年のマイクロプロセッサには処理性能を維持しつつ消費電力を削減することが求められている．我々はプログラム実行中のクリティカルパス情報で低速・高速な演算器を使い分ける省電力アーキテクチャを提案している．本稿では，クリティカルパス予測器を更新する際に分岐予測ミス情報，キャッシュミス中に実行される命令の情報などを利用することを提案する．また，プログラム実行中のトレース情報を用いて測定したクリティカルパス予測器の予測精度について報告する．

Evaluating the Critical Path Predictors Using Critical Path Detection Criteria

AKIHIRO CHIYONOBU† and TOSHINORI SATO††

Recently, microprocessors are required to reduce energy consumption maintaining its computation performance. Microprocessors we are proposing have two types of functional units distinguished in terms of their execution latency and power consumption. Only critical instructions are executed on power-hungry functional units, and thus the total energy consumption can be reduced without severe performance loss. In this paper, we propose new critical path detection criteria that utilize information of miss branch predicted instruction instruction executed during cache miss, and so on. Those criteria make the information updating critical path predictor. We evaluate critical path prediction accuracy comparing with the trace information executing program.

1. はじめに

近年携帯情報端末や組み込みシステムにおいても高い処理性能が求められるようになっており，高性能なプロセッサが搭載されている．しかしこれらのシステムには高い処理性能が求められる一方で，その消費する電力も少量であることが求められている．プロセッサの消費電力と性能はトレードオフの関係にあるため，特に携帯情報端末用プロセッサでは電力消費量が設計における大きな制約となる．マイクロプロセッサの消費電力 P_{active} と遅延時間 t_{pd} はそれぞれ，

$$P_{active} \propto fC_{load}V_{dd}^2 \quad (1)$$

$$t_{pd} \propto \frac{V_{dd}}{(V_{dd}-V_{th})^{\alpha}} \quad (2)$$

で求めることができる．ここで，f はクロック周波数，C_{load} は負荷容量，V_{dd} は電源電圧，そして V_{th} はデバイスの閾値電圧である．また，α はキャリア移動度の飽和を表すパラメタで通常 1.3〜1.5 の値をとる[1]．式 (1) からわかるように電源電圧を下げることが電力削減においては最も効果が高い．しかし，式 (2) に従えば，電源電圧の低下はゲート遅延を増大してしまい，クロック周波数が低下してしまう．このことは，マイクロプロセッサの性能低下につながる．この問題に対して本研究では，プログラム実行中のクリティカルパスに着目した．具体的にはレイテンシと電源電圧の異なる演算器を用意し，実行時間に影響を与えるクリティカルパス上の命令は高速かつ電力消費の大きな演算器に実行させ，クリティカルパス上にない命令は低速かつ消費電力の小さな演算器で実行させる．このことにより，実行時間を増やすことなくプロセッサの消費電力を抑えることが可能となる．本稿では上記のアーキテクチャ実現に不可欠なクリティカルパス予測器の予測精度向上について述べる．

2. クリティカルパス情報を用いた省電力命令スケジューリング

現在プロセッサにおける低消費電力化の研究はキャッ

† 九州工業大学大学院 情報工学研究科
Graduate School of Computer Science and System Engineering, Kyushu Institute of Technology
†† 九州大学 システム LSI 研究センター
System LSI Research Center, Kyushu University

シュメモリや命令ウインドウをターゲットにしているものが多い．これは，プロセッサの消費電力に占めるキャッシュメモリや命令ウインドウの消費電力の割合が高いためである．このためキャッシュメモリや命令ウインドウのプロセッサに占める消費電力は減少する傾向にあるが，今後は演算器の消費する電力が相対的に増加していくと予測される．このため，本研究では演算器の低消費電力化をねらうこととする．

現在多くのプロセッサは実行しようとする命令を機能ユニットに対してアウト・オブ・オーダに発行することでプログラムの実行時間を少なくしている．プログラムの実行時間は，プロセッサの処理性能と実行している命令間の依存関係によって決まる．

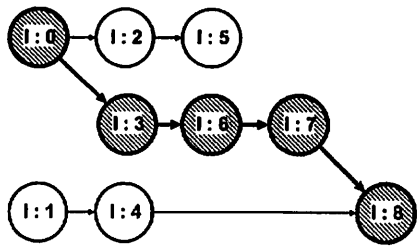

図 1　クリティカルパス

我々は実行サイクル数を増やさずにプロセッサの低消費電力化を実現するため，プログラムの実行時間を決めるというクリティカルパスの特性に着目した[2]．クリティカルパス上の命令はプログラムの実行時間を決定するが，クリティカルパス上にない命令はクリティカルパス上の命令の開始を遅らせない限りプログラムの実行速度に影響を与えない．この点に着目してプロセッサの持つ ALU の構成を変更する．現在多くのプロセッサは ALU を複数持っているが，これらの ALU として消費電力の異なるものを用意する．例えば，一部の ALU の電源電圧を下げる．電源電圧を下げることによってレイテンシは増加するが電力消費を下げることが可能になる．このレイテンシと電源電圧の異なる演算器を持つプロセッサに対し，命令をクリティカルパス上の命令かどうかという情報に基づいて，実行時間に影響を与えるクリティカルパス上の命令は高速かつ電力消費の大きな演算器で実行し，クリティカルパス上にない命令は低速かつ電力消費の小さな演算器で実行するように命令をスケジューリングする．この様に命令をスケジューリングすることで，プログラム全体の実行時間を変化させることなくプロセッサの消費電力を下げることができ，電源電圧を下げることによる動作速度低下という弊害を隠蔽することが可能となる．

クリティカルパス情報を利用して電力消費を削減しようとする試みは他にもあるが[3),4)]，いずれもエネルギー消費量ではなくピークの電力値を削減することが目的であり，我々の目的とは異なる．

3. クリティカルパスモデル

これまで我々は毎サイクル命令ウインドウ内にある命令間のデータ・フローグラフ (Data Flow Graph: DFG) を作成し，グラフの最長パスをクリティカルパスと見なすパス情報テーブル (Path Information Table: PIT)[5)] が特定する情報を正確なクリティカルパスとして研究を行ってきた．しかし PIT は DFG から最長パスを特定できても，命令のレイテンシやハードウェア制限といった情報を DFG に反映できない．このため PIT から得られるクリティカルパス情報は，実際のクリティカルパスとは若干異なっていることが予想される．

そこで，今回 Fields らの提案するクリティカルパスモデル[6)] を用いて正確なクリティカルパス情報を得ることにする．Fields らのモデルは PIT が用いない情報を利用してクリティカルパスを特定する．このため，正確なクリティカルパスを特定可能である．しかし彼らのモデルはトレース情報を用いることから，ハードウェアとして実装することは困難である．次節で Fields らの提案するクリティカルパスモデルについて説明する．

3.1 Fields らのクリティカルパスモデル

Fields らのクリティカルパスモデルは命令間のデータ依存・データの作成される時間だけでなく，制御依存，ハードウェア依存などを考慮して DFG を作成する．そのため命令の各実行フェーズを 3 つのノードとして捉える．それらは，命令ディスパッチの D ノード，命令実行の E ノード，命令コミットの C ノードである．ノード間に依存がある場合は，各ノードをエッジで繋げる．各エッジの重みは，エッジの終点にあるノードが実行可能になるまでのレイテンシである．これらのノードを用いて，命令実行中のイベントの依存とデータの依存を表す．いくつかのエッジが表す依存を以下に示す．紙面の都合上，詳細は文献 6) を参照されたい．

- D → E エッジ：命令はディスパッチされてからでないと実行できない
- E → C エッジ：実行が完了してからでないとコミットできない
- D → D エッジ：イン・オーダディスパッチ
- C → C エッジ：イン・オーダコミット
- E → E エッジ：命令間のデータ依存
- E → D エッジ：分岐予測ミス (予測に失敗した分岐命令の E ノードから正しい分岐先命令の D ノードへのエッジ)
- C → D エッジ：リオーダバッファ容量制限 (リオーダバッファの最古の命令から次にディスパッ

チされる命令のDノードへのエッジ)
以上のように表現される各命令のノード間の依存関係から，最終命令のCノードから最初にディスパッチされた命令のDノードへと最も重いエッジを辿ることでクリティカルパスを特定する．

4. クリティカルパスを特定する機構

4.1 クリティカルパス予測器

命令の実行時に当該命令がクリティカルか否かを判断する機構として，クリティカルパス予測器[2),6),7)]とPIT[5)]がある．Tuneらと筆者らが提案しているクリティカルパス予測器[2),7)]を用いたこれまでの検討では，クリティカルパス予測器の予測精度が不十分であるため，必ずしも性能と省電力が両立できるとは限らないことが分かっている．一方，PITは上述のクリティカルパス予測器よりも正確にクリティカルパスを特定できることが分かっている．しかし，Fieldsらのクリティカルパス予測器[6)]とPITはハードウェアが複雑で，機構自身が多くの電力を消費する．したがって，複雑度の小さなクリティカルパス予測器を改良する必要がある．

我々の提案するクリティカルパス予測器を図2に示す．クリティカルパス予測器はタグを持たないダイレクトマップキャッシュと同様の構成をしている．各エントリは飽和型のアップ・ダウンカウンタである．各エントリは命令のディスパッチ時に命令アドレス(PC)，クリティカルパス履歴，分岐履歴，ロード命令のヒット・ミス履歴などを用いてアクセスされる．命令に対応するカウンタの値が閾値よりも大きければその命令はクリティカルであると予測され，閾値よりも小さければその命令はノンクリティカルと予測される．またクリティカルパス予測器は，対応する命令がコミットする際に更新される．コミットする命令が実行中にクリティカルパス上にあったと判断されると対応するカウンタはインクリメントされ，そうでないと判断された場合はデクリメントされる．カウンタの飽和値や予測時の閾値，カウンタのアップ幅，ダウン幅は予測器の実装に依存する．命令が実行中にクリティカルパス上にあったかどうかの判断にはヒューリスティックスを用いる．

4.2 クリティカルパス予測器の更新情報

前節で述べたようにクリティカルパス予測器は命令実行中の振る舞い情報を用いて更新される．このため，実行された命令がクリティカルか否かを特定するクリティカルパス判定基準の正確さがクリティカルパス予測精度を左右する．これまでの研究で，Tuneらによって提案されているクリティカルパス判定基準(QOLD, ALOLD)が特定するクリティカルパス情報は十分に正しくないことが分かっている[8)]．このため，我々はTuneらのクリティカルパス判定基準と併用すること

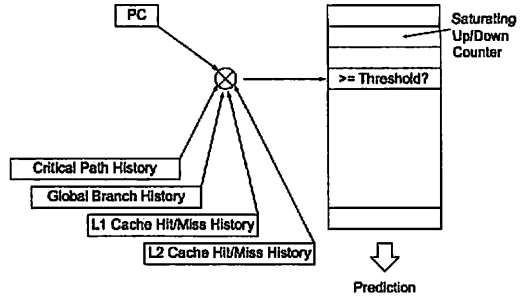

図2 提案するクリティカルパス予測器

でクリティカルパス特定精度を向上させる新たなクリティカルパス判定基準を提案している[9)]．表1にTuneらのクリティカルパス判定基準と我々の提案するクリティカルパス判定基準を示す．我々が提案するクリティカルパス判定基準はQA，BM，L1，L2，E-DFUG[10)]の5つである．以下でそれぞれについて説明する．

QAはTuneらによって提案されているQOLD, ALOLDを組み合わせたクリティカルパス判定基準である．命令ウインドウやアクティブリストで最も古くなった命令は，新たな命令がプロセッサに供給されることを妨げているため，クリティカルパスに影響を与える．BMは分岐命令に着目したクリティカルパス判定基準である．分岐予測に失敗した命令は投機実行中の命令を破棄し，正しいパスを命令フェッチからやり直す必要がある．このことはクリティカルパスに影響を与える．L1，L2はロード命令のキャッシュメモリへのヒット・ミスに着目したクリティカルパス判定基準である．キャッシュミスが解決されるまで必要とするデータがプロセッサに供給されないため，クリティカルパスに影響を与える．E-DFUGはキャッシュミス中に実行される命令に着目したクリティカルパス判定基準である．一般にメモリ階層の上位へのアクセスはプロセッサから見ると非常に遅い．これによりメモリアクセス中に実行される命令は，メモリアクセスを引き起こした命令の実行終了後にはノンクリティカルになる．逆に当該メモリアクセス命令に依存する命令はクリティカルとなる．各クリティカルパス判定基準は，それぞれ組み合わせて使うことができる．

5. 評価環境と評価結果

SimpleScalarツールセット[11)]を利用して，Fieldsらのモデルからクリティカルパスのトレースを採取した．そのトレースを利用して，省電力アーキテクチャの評価，クリティカルパス予測器の更新，クリティカルパス判定基準との比較を行う．命令セットはPISA命令セットを用いた．命令ウインドウは128エントリとした．ベンチマークはSPEC 2000 CINTである．どのプログラムも，先頭の20億命令でウォームアッ

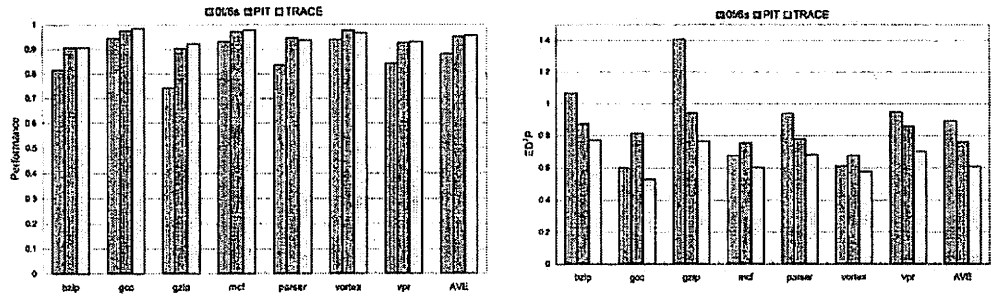

図 3 トレース情報でスケジューリングした場合の結果

表 1 クリティカルパス判定基準

QOLD[7]	命令ウインドウ中で最古の命令を クリティカルとする
ALOLD[7]	アクティブリスト中で最古の命令を クリティカルとする
QA	命令ウインドウ中で最古の命令，もしくは アクティブリスト中で最古の命令をクリティカル とする
BM	分岐予測に失敗した分岐命令を クリティカルとする
L1，L2	キャッシュミスしたロード命令を クリティカルとする
E-DFUG	キャッシュミス中に実行された命令を ノンクリティカルとする

プし，続く 5 億命令をシミュレーションした．

5.1 トレース情報を利用した省電力アーキテクチャの評価

トレース情報を用いて我々の提案する省電力アーキテクチャを評価した．高速な演算器と低速な演算器のレイテンシはそれぞれ 1 : 2 となるようにする．低速な演算器は，高速な演算器をパイプライン動作させるものに相当し，レイテンシは 2 に増加するが，スループットは 1 のままである．これらの演算器の電源電圧は文献 12) で紹介されているものから動作周波数の比が 1 : 2 となる組み合わせを用いた．またトレースを用いた際の理想的な結果を得るため，トレース情報がクリティカルな場合は必ず高速な演算器を，ノンクリティカルな場合は必ず低速な演算器を使用できるようにした．命令発行幅は固定である．図 3 に結果を示す．図 3 において，左図は処理性能を右図はエネルギー遅延二乗積 (Energy Delay Square Product: ED^2P) をそれぞれ示している．ED^2P はパワー削減による遅延の増加と省電力効果のトレードオフを定量的に評価するために用いられる指標である．電圧制御を用いて省電力化を試みる場合の評価に適している[13]．ED^2P は，高速・低速な演算器がそれぞれ何回ずつ使用されたかをカウントし，その回数とそれぞれの演算器を使用した際のエネルギー遅延を乗して求めることとした．処理性能につしてはグラフが高い方が好ましく，ED^2P についてはグラフが低い方が好ましい．それぞれの結果は全ての演算器が高速な場合で正規化されている．図中の TRACE はトレース情報を用いてスケジューリングした結果を，PIT は PIT を用いてスケジューリングした結果を，0f/6s は全ての演算器が低速な場合をそれぞれ表している．処理性能低下の度合いは，トレース情報，PIT のどちらを用いても大きな差は見られない．一方 ED^2P はトレース情報を用いてスケジューリングをした方が省電力化を達成できている．このことから，Fileds のモデルから作成したトレースは PIT よりも正確にクリティカルパスを特定できていることがわかる．

5.2 クリティカルパス判定基準の評価

トレース情報とクリティカルパス判定基準を比較した結果を図 4 に示す．紙面の都合上それぞれの結果の平均のみを掲載する．左図は各クリティカルパス判定基準のみの結果を，右図は横軸に示された各クリティカルパス判定基準に加え，E-DFUG を適用した場合の結果である．横軸は PIT と Tune らによって提案されたクリティカルパス判定基準，我々の提案するクリティカルパス判定基準を表している．縦軸はトレース情報との一致を示している．下から，トレースとクリティカルパス判定基準の両方がクリティカルだった場合 (f_C_heuri_C)，トレースはクリティカルだったが，クリティカルパス判定基準ではノンクリティカルだった場合 (f_C_heuri_NC)，トレースはノンクリティカルだったがクリティカルパス判定基準はクリティカルだった場合 (f_NC_heuri_C)，トレースとクリティカルパス判定基準の両方がノンクリティカルだった場合 (f_NC_heuri_NC) をそれぞれ示している．

各クリティカルパス判定基準ごとの結果を見てみる．クリティカルパス判定基準を組み合わせるにつれ，正しくクリティカルを特定する頻度が増すが，ノンクリティカルの場合にクリティカルとする頻度も

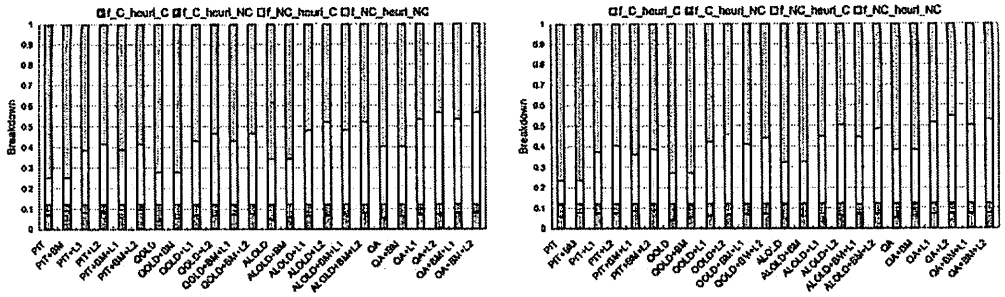

図 4　トレース情報とクリティカルパス判定基準の比較

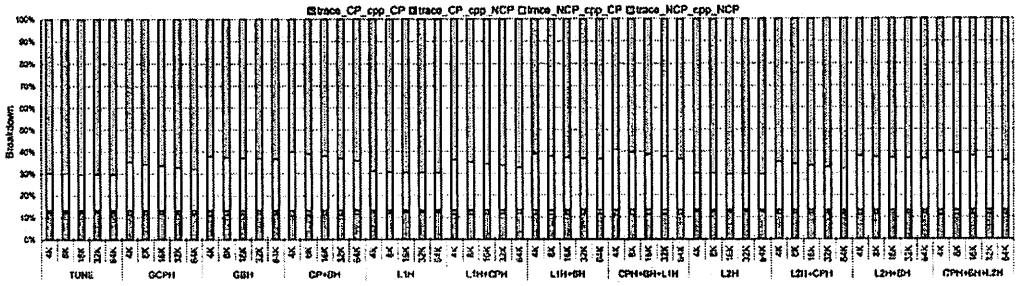

図 5　トレース情報でクリティカルパス予測器を更新した場合の結果

増えることが分かる．しかし BM を適用した場合は，間違った特定を増やすことなくクリティカルと特定する頻度が増加している．E-DFUG を適用した場合，f_NC_heuri_C が減り，f_NC_heuri_NC が増加している．具体的には f_NC_heuri_C は最大 3.5%，平均で 2% 減り，f_NC_heuri_NC は最大で 3.8%，平均で 2% 増加する．このことから，E-DFUG を用いた場合に正しくノンクリティカルを特定する頻度が増すことがわかる．正しい CP モデルがクリティカルとする部分については，若干 f_C_heuri_C が減り，f_C_heuri_NC が増える．しかしその差は最大で 1%，大部分は 0.05% 以下である．このことから E-DFUG を適用することで，より正確にノンクリティカルな命令を特定できることがわかる．

5.3　クリティカルパス予測器の評価

最も予測精度の高いクリティカルパス予測器の構成を調べるため，提案する各予測器をトレースを用いて更新した．クリティカルパス予測器が持つカウンタは 6 ビットとし，予測時の閾値は 4，カウンタのアップ幅は 4，ダウン幅は 1 とした．結果を図 5 に示す．図において，横軸はクリティカルパス予測器とエントリ数を表す．**TUNE** は Tune らの提案する構成を，**GCPH**，**CPH** はクリティカルパス履歴を用いる構成，**GBH**，**BH** は分岐履歴を用いる構成，**L1H**，**L2H**，は各キャッシュメモリへのヒット・ミス情報を用いる構成をそれぞれ示している．プラス記号 (+) でつながっている構成は，クリティカルパス予測器のインデックス作成に複数の情報を用いてることを示している．グラフの縦軸はトレース情報と予測結果の一致の度合いである．

クリティカルパス予測器のエントリ数に比例して，どの構成でも正しく予測する割合が増加している．正しくクリティカルと予測する割合よりも，正しくノンクリティカルと予測する割合が大きく改善している．これはエントリの増加に伴い，エントリの競合が減ったためと考えられる．

TUNE と **L1H**，**L2H** が最もよい予測精度を示しているが，予測器の構成を変化させても予測精度に顕著な差は見られない．クリティカルパス履歴，分岐履歴のどちらか，またそれらとキャッシュメモリへのヒット・ミス情報を組み合わせた場合，**TUNE** と比較して間違った予測の割合が増加する．特に間違ってクリティカルと予測する割合が増加する．このため，クリティカルパス履歴，分岐履歴を用いるとエントリの競合が起きやすくなるものと予想される．また，間違ってクリティカルと予測する割合は分岐履歴を用いた場合の方がクリティカルパス履歴を用いた場合よりも増加する．これは分岐先が同一方向へ偏っていることが多いためである．

次にトレース情報で更新した際に高い予測精度を示した **TUNE** をクリティカルパス判定基準が特定する情報で更新した際の予測精度を示す．クリティカ

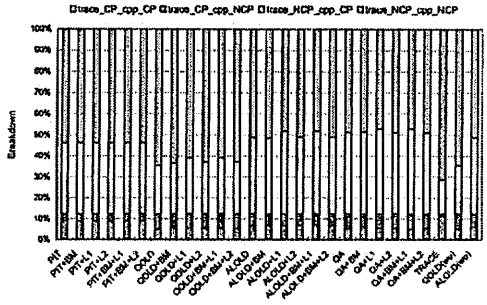

図 6 クリティカルパス判定基準で更新した結果 (TUNE)

ルパス判定基準には各クリティカルパス判定基準と E-DFUG を併用したものを用いる．図 6 に結果を示す．図において横軸は E-DFUG と併用したクリティカルパス判定基準である．TRACE はトレース情報をを用いて更新した場合，QOLD(wo)，ALOLD(wo) は E-DFUG を併用しない場合の結果である．どのクリティカルパス判定基準の組み合わせもトレース情報を用いて予測器を更新した場合よりも予測精度が低いことがわかる．クリティカルパス判定基準で良い結果が見られた BM や E-DFUG を適用した場合について結果を見てみる．残念ながら E-DFUG を適用する前と後の QOLD，ALOLD と比較しても予測精度の違いは見られない．BM と QOLD を併用すると正しくクリティカルと予測する割合が増加するが，間違ってクリティカルと予測する割合も増加する．BM と ALOLD，QA を併用した場合は，僅かではあるが予測精度の改善が見られる．しかし，間違った予測をする割合は QOLD が最も小さい．このため，新たに提案したクリティカルパス判定基準からは期待された効果が得られないことがわかる．

6. まとめと今後の課題

トレース情報を用いて正確なクリティカルパス情報とクリティカルパス判定基準，クリティカルパス予測器の特定・予測精度を明らかにした．その結果，新たに提案したクリティカルパス判定基準は特定精度を改善するが，予測器の予測精度には大きな影響を与えないことがわかった．トレース情報でクリティカルパス予測器を更新した場合に最も良い予測精度が得られていることから，今後はさらに正確なクリティカルパス判定基準について研究を行う予定である．

謝辞 本研究の一部は，文部科学省科学研究費補助金 (No. 16300019, No.176549) の援助によるものです．

参 考 文 献

1) Hiramoto, T. and Takamiya, M.: Low Power and Low Voltage MOSFETs with Variable Threshold Voltage Controlled by Back-Bias, *IEICE Transactions on Electronics*, Vol. E83-C, No.2 (2000).
2) 千代延昭宏, 佐藤寿倫, 有田五次郎：低消費電力プロセッサアーキテクチャ向けクリティカルパス予測器の提案, 並列／分散／協調処理に関する『湯布院』サマー・ワークショップ (SWoPP) (2002).
3) Pyreddy,R. and Tyson,G.: Evaluating Design Tradeoffs in Dual Pipelines, in *Workshop on Complexity-Effective Design* (2001).
4) Seng,J.S., Tune,E.S. and Tullsen,D.M.: Reducing power with dynamic critical path information, in *Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture* (2001).
5) 小林良太郎, 安藤秀樹, 島田俊夫：データフロー・グラフの最長パスに着目したクラスタ化スーパースカラ・プロセッサにおける命令発行機構, 並列処理シンポジウム (2001).
6) Fields,B.A., Rubin,S. and Bodik,R.: Focusing Processor Policies via Critical-Path Prediction, in *Proceedings of the 28th Annual International Symposium on Computer Architecture*, pp. 74–85 (2001).
7) Tune, E., Liang, D., Tullsen, D. M. and Calder,B.: Dynamic Prediction of Critical Path Instructions, in *Proceedings of the 7th International Symposium on High-Performance Computer Architecture*(2001).
8) 千代延昭宏, 佐藤寿倫：プログラムの実行時における命令の重要度決定に関する検討, 並列／分散／協調処理に関する『松江』サマー・ワークショップ (SWoPP) (2003).
9) 千代延昭宏, 佐藤寿倫：クリティカルパスを特定するためのヒューリスティックスの改善, 先進的計算基盤システムシンポジウム (2006).
10) Chiyonobu, A. and Sato, T.: Energy-Efficient Instruction Scheduling Exploiting Memory Access Slack, in *Memory performance: Dealing with Applications , systems and architecture Workshop held in conjunction with PACT 2005* (2005).
11) Burger, D. and Austin, T.: The simplescalar toolset version 2.0, Technical Report CS-TR-97-1342, University of Wisconsin (1997).
12) Intel Corp.: Intel Pentium M Processor Datasheet (2004).
13) Martonosi,M., Brooks,D. and Bose,P.: Modeling and Analyzing CPU Power and Performance: Metrics, Methods, and Abstractions, in *SIGMETRICS 2001 / Performance 2001 - Tutorials* (2001).