6J-5

# 2パス限定投機システムにおける消費電力の検討

菅野　智之 † 大津　金光 † 横田　隆史 † 馬場　敬信 † 大川　猛 †
† 宇都宮大学工学部情報工学科

## 1　はじめに

近年では, マルチコアプロセッサを搭載したコンピュータが普及している. また, プログラムの速度向上手法としてそれらマルチコアプロセッサを使用した並列実行手法が提案されている. しかし, 提案された並列実行手法は, 並列実行による速度向上率よりも, 並列実行によって発生する消費電力の増加率が高ければ電力効率の悪い手法となる. そこで, 電力効率も良い並列実行手法を考えるには, 速度を評価するだけでなく, 電力も評価しなければならない.

並列実行手法として我々は, プログラム中のループの実行頻度の高い 2 つの実行経路 (以下, パス) について最適化したコードを生成し, プログラム実行時にどちらのパスが実行されるかを予測し, 投機的にマルチスレッド実行を行うアーキテクチャとして 2 パス限定投機システム PALS を開発している [1].

本稿では, 2 パス限定投機システム PALS のシミュレータへの電力評価機能の実装を行う.

## 2　2パス限定投機システム PALS

PALS ではプログラム内のループの実行頻度が高い 2 つのパスについて最適化された投機コードをあらかじめ用意する. そして, プログラム実行時にどちらのパスが実行されるのかを予測し, 予測したパスの投機コードを実行する. 図 1 では, PALS の構成を示している. TU (Thread Unit) は予測されたパスを実行する機構であり, TU を制御し, パスを予測する機構が TMU (Thread Management Unit) である. また, MB (Memory Buffer) は投機的なメモリアクセスを適切に処理するために各 TU が持つ記憶装置であり, LS(Load Shelter) は MB の記憶領域が不足した場合の補助的な記憶装置である.

### 2.1　PALS シミュレータ pals

pals は PALS の動作をクロックレベルで模擬するシミュレータである. pals の開発ベースとして ISIS-SimpleScalar[2] を用いている. ISIS-SimpleScalar は C++により記述されており TU は ISIS-SimpleScalar における既存のプロセッサユニットを構成するクラスを継承することによって, TU における独自の機能を実装した. また, PALS 独自の機構である TMU, MB および LS は, 新規にクラスを作り, 実装を行った.

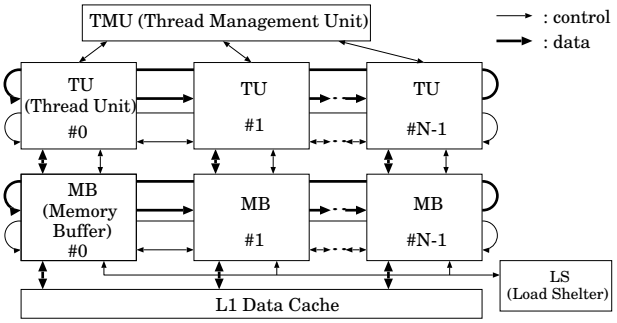


図 1: PALS のハードウェア構成

## 3　電力評価ツール Wattch

Wattch[3] は, SimpleScalar[4] をベースにして開発された電力機能搭載のシングルプロセッサシミュレータである. Wattch は電力評価を行う各ユニットに対してアクセス回数を求め, そのアクセス回数をもとに電力消費量を算出するプログラムとなっている. Wattch では一般的なプロセッサ内部のユニットに対して配列構造, CAM (Content Addressable Memory), 組み合わせ回路, クロック回路の 4 つにわけて静電容量を求め, 式 1 で消費電力を算出する. 配列構造とはレジスタファイルの構造のようなデコーダやワードライン, ビットラインを用いた構造である. 稼働率 $\alpha$ は (アクセスサイクル数) / (サイクル数) で求めることができる.

$$P_{active} = \alpha * C_{load} * f * V_{dd}^2 \qquad (1)$$

$\alpha$ : 稼働率　　$C_{load}$ : 静電容量
$f$ : クロック周波数　　$V_{dd}$ : 電源電圧

我々は先に ISIS-SimpleScalar に Wattch の電力計測機能をクラス化して実装することによって拡張可能な CMP (Chip Multi Processor) シミュレータを開発した [5]. 本研究ではこれを発展させ, pals 全体での電力評価を行えるようにする.

## 4　PALS への電力評価機能の実装

PALS における TMU や MB といった独自の機構の電力は, 配列構造, CAM, 組み合わせ回路, クロック回路のうちの当てはまる構造で静電容量を決めて算出する. また, pals の TU, TMU, MB の各クラスに対してアクセスカウンタと電力計算処理コードを追加し, 各ユニットの独立した電力統計の出力を可能にした.

### 4.1　TU の電力計測機能の実装

TU は汎用プロセッサに, 投機的なスレッドを制御するための機能を追加したものである. このため汎用プロセッサ部分の電力計測機能は文献 [5] と同様な手法

Consideration on Power Consumption of Two-Path Limited Speculation System
†Tomoyuki Kanno, Kanemitsu Ootsu, Takashi Yokota, Takanobu Baba and Takeshi Ohkawa
Department of Information Science, Faculty of Engineering, Utsunomiya University (†)

で実現する．

投機的なスレッドを制御するための機構として ISB (Initial State Buffer) がある．PALS では投機的にスレッドを実行するために投機失敗等により各 TU は実行中のスレッドを破棄する場合がある．スレッドを破棄した後は新たなスレッドを開始するため, TU のレジスタは初期化しなければならない．この初期化のために参照されるのが ISB であり, その中には各レジスタデータの投機スレッド実行開始点での初期値が格納されている．ISB では通常のレジスタファイルと同様に配列構造として静電容量を算出し, 電力計測処理を TU のクラスに実装した．

### 4.2 TMU の電力計測機能の実装

TMU は TU の制御とパスの予測を行う機構である．TMU の主な電力を消費するユニットはパス予測器であり, パス予測器は 2 レベル分岐予測器と同様の構造をしているため配列構造として静電容量を算出し, 電力計測処理を TMU のクラスに実装した．

### 4.3 MB の電力計測機能の実装

MB は投機的なスレッドのメモリアクセスを処理するために各 TU が持つ記憶装置である．MB は連想メモリ方式でデータを記憶している．したがって MB では CAM として静電容量を算出し, 電力計測処理を MB のクラスに実装した．また, L1 データキャッシュについても, 文献 [5] と同様に配列構造として静電容量を算出し, 電力計測処理を MB のクラスに実装した．

## 5 実行結果

図 2 でエラトステネスの篩を用いた素数判定プログラムを SimpleScalar で実行した場合の各ユニットの消費電力を 1 として pals で実行した場合と比較した結果を示す．比較を行うユニットは, rename (リネームユニット), bpred (分岐予測器), window (命令ウィンドウ), lsq (ロード ストアキュー), regfile (レジスタファイル), icache (L1 命令キャッシュ), dcache (L1 データキャッシュ), alu (ALU), resultbus (リザルトバス), clock (クロック), total (プログラム全体) である．図 3 は SimpleScalar で実行した場合のサイクル数と pals で並列実行した場合のサイクル数をもとに算出した速度向上率を示す．

図 2 より, SimpleScalar と pals のプログラム全体の消費電力を比較すると, pals の消費電力が TU 1 台の場合, SimpleScalar で実行した場合の 50% の電力しか消費しなかった．これは PALS が最適化された投機コードを実行したことにより, 各ユニットのアクセスする回数が減少したためであると考えられる．clock の電力に関しては他のユニットの消費電力の合計値が電力を決定する要素となるので, total と同様な結果となったと考えられる．また, 図 2 より, TU 2 台では TU 1 台に比べて, TU 台数の増加により 1.8 倍程度電力を消費しているが, TU 4 台では TU 2 台に比べて約 1.1 倍しか電力を消費しなかった．これは図 3 より, 速度

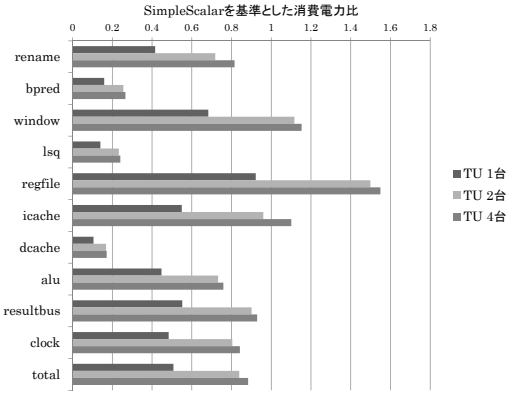


図 2: プログラム実行時の消費電力比

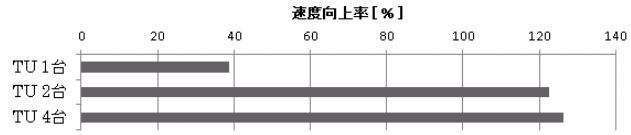


図 3: プログラム実行時の速度向上率

向上率が TU 4 台では TU 2 台に比べて 1.1 倍しか増加しなかったので, 消費電力を算出する式 1 の稼働率 $\alpha$ も 1.1 倍程度しか増加しなかったためであると考えられる．

## 6 おわりに

本稿では 2 パス限定投機システム PALS シミュレータ pals に Wattch による電力計測機能を実装し, pals の独自機構である ISB, パス予測器, MB について電力計測機能を追加した．また, 実際にプログラム実行を行い消費電力を求め, SimpleScalar と pals で各ユニットの電力消費量を比較した．

今後は, 他のプログラムを実行した場合の各ユニットの電力計測結果を比較し, 実行結果の検証を行う．

**謝辞**

本研究は，一部日本学術振興会科学研究費補助金 (基盤研究 (C)21500050，同 (C)21500049) の援助による．

**参考文献**

[1] 十鳥弘泰ほか: “2 パス限定投機方式を実現するマルチコアプロセッサ PALS の提案”, 信学技報, Vol.109, No.319(CPSY2009-46), pp.19-24, 2009.

[2] 藥袋俊也ほか: “ISIS-SimpleScalr の実装”, 情報処理学会研究報告, 2004-Arc-160, pp.29-34, 2004.

[3] David Brooks et al: “Wattch: A Framework for Architectural-Level Power Analysis and Optimizations”, in Proc of ISCA, pp.83-94, 2000.

[4] SimpleScalar LLC, http://www.simplescalar.com/.

[5] 佐藤裕輔ほか: “拡張可能な CMP シミュレータの電力評価環境構築”, 情報処理学会 第 71 回全国大会, pp.1-93 〜1-94, 2009.