Vol.2014-ARC-208 No.12
2014/1/23

ターボ・モード強化のための面積効率に優れたマイクロプロセッサとその設計手法

三輪 忍[1,a)]　井上 聖等[1]　中村 宏[1]

概要：多くのプログラムを加速できるターボ・モードは商用プロセッサにおいて広く採用されている．ターボ・モード中に利用できる最大動作周波数は，CPU 負荷の大きいプログラムを用いた温度試験を通じて，温度制約違反を起こさないようにある程度の温度マージンを設けた上で定められている．したがって，プログラム実行中のプロセッサの温度上昇を抑えることができれば，同一温度マージンのもとでより高い動作周波数を利用することができ，その結果，ターボ・モードによる性能向上効果を高めることができる．本論では，少量のハードウェアを追加することでプロセッサの温度上昇を抑制し，ターボ・モードの能力を増強する方法について述べる．また，そのようなプロセッサの設計手法も併せて提案する．評価の結果，提案手法により 2.8% の面積増加で最大 14.5% の性能向上を達成した．

1. はじめに

チップ温度は近年のコンピュータ・システムの性能を左右する重要な要素である．深刻なダメージを受けるのを防ぐため，プロセッサには動作可能な上限温度が定められており，常にこの温度以下で動作することが求められる．商用プロセッサにおいて広く採用されているターボ・モードはチップ温度を著しく上昇させる．ターボ・モードは動作電圧と周波数を上昇することでプロセッサを加速する方法であるが，用いる電圧と周波数が高すぎる場合は上記の温度制約を違反してしまう．チップ温度が上限温度を上回った場合は電圧と周波数の低下あるいはクロック・ゲーティング等の手段により，チップが冷却される [11]．温度制約違反を起こした場合は大幅に性能が低下するため，現在流通しているチップは，典型的な使用状況では例えターボ・モード中であっても制約違反を起こさぬよう，製品出荷前に行われる温度試験の結果にある程度の温度マージンを加えた上で利用可能な最大周波数が定められている．

本論では，少量のハードウェアを追加することでチップの温度上昇を抑制し，ターボ・モードの能力を増強する方法について述べる．プロセッサ内の各ユニットが発生する熱はそれぞれの消費電力に依存し，各ユニットの消費電力はそのアクティビティに依存する．したがって，熱源となるユニットに関してその複製を用意しておき，それらを公平に利用すればユニットの温度上昇を抑えることができる．その結果，同一温度マージンのもとでより高い動作周波数を利用できるようになり，ターボ・モードによる性能向上効果を高めることができる．

実は，上記のようなチップの温度管理手法とその利用方法を提案するのは，本論が初めてではない．我々と同様のコンセプトは 2003 年に既に Heo らによって提案されており，上記の温度管理手法はアクティビティ・マイグレーション (以下 AM とする) と呼ばれている [8]．その後，別タイプの AM が Chaparro らによって提案されたが [2], [3]，どちらも AM を適用する場合の空間粒度については十分な検討を行っていない．彼らはプロセッサ内の粗粒度なモジュール（例えばレジスタ・ファイルと実行ユニット群のセット）に対して AM を適用しており，細粒度なモジュール（例えば個々の ALU）に対する適用は検討していない．また，従来研究では AM の対象となるモジュールやその複製数は経験的に定められており，あるプロセッサに対してどのモジュールを何個複製すればよいか，といった方法論はまだない．

本論では細粒度 AM とそれを用いるプロセッサの設計手法を提案する．後述するように，プロセッサ内の限られたモジュールがチップ温度を上昇させるため，そのようなモジュールのみを複製すれば，AM による性能向上を維持しつつ面積増加を抑えることができる．本論で提案する設計手法は，上記のようなモジュールをアーキテクチャ設計の段階で検出し，適切な複製数を算出する．熱源の検出と複

[1] 東京大学
The University of Tokyo
[a)] miwa@hal.ipc.i.u-tokyo.ac.jp

製数の算出は，我々が提案するモジュールの温度モデルを用いて行う．

本論の構成は以下の通りである．まず，2 章においてターボ・モードとチップ温度の関係を説明し，続く 3 章では AM について解説する．4 章で研究のモチベーションを述べた後，5 章で細粒度 AM を用いるプロセッサの設計手法を提案する．評価方法は 6 章で，結果は 7 章で述べる．8 章で関連研究を概観し，最後 9 章でまとめる．

2. ターボ・モードとチップ温度

近年の CMP には，さまざまなプログラムをアクセラレーション可能なターボ・モードが搭載されている．いくつかのコアがアイドル状態にある時，プロセッサはターボ・モードと呼ばれる残りのコアの周波数がクロック・アップされた状態となる．ターボ・モードは動作周波数と動作電圧の上昇という単純な仕組みで大きな性能向上を得られることから，多くの商用プロセッサで採用されている [5], [11].

Turbo Boost のホワイト・ペーパー [11] によれば，ターボ・モードは以下の要因により制約を受ける．

- システムの電力供給能力
- プロセッサの消費電流
- プロセッサの消費電力
- プロセッサ温度

より高い周波数と電圧を使用することを考えた場合，これらの要因のうち最初の 3 つに関しては，現在のシステムでも問題ないと言える．周波数と電圧を上昇させてもこれらによる制約を受けることがほとんどないことは，さまざまなオーバークロッキングの結果が示唆している．その一方で，温度制約は致命的である．CPU の周波数と電圧を上昇させていくとチップ温度は急激に上昇するのに対し，一般的な冷却装置はそれを許容できる程の能力を有していないためである．

ターボ・モード中のプロセッサは次のように動作する．プロセッサがアクティブ状態（ACPI の P0 状態 [9]）にある時，CPU の周波数と電圧は，各 CPU メーカが定めた利用可能な最大値まで上昇する．このオーバークロック状態は，チップ温度が制約を満たす限り継続される．チップ温度が制約に近づくと，プロセッサは周波数と電圧を低下させる（場合によってはクロックも停止する）ことにより，消費電力を抑えることでチップを冷却する．チップ温度がある程度のレベルに戻ると，プロセッサは再びオーバークロック状態となる [11].

参考までにチップ温度によってターボ・モードの効果がどのように変化するかを評価したので，その結果を**図 1** に示す．図 1 はターボ・モード中のアクティブ・コアの平均動作周波数を表している．2 本ずつ並んだ棒グラフの左は標準的な環境でプログラムを実行した場合（「/w fan」），右は CPU ファンを止めて実行した場合（「/wo fan」）である．CPU は Core i7-3930K，CPU ファンは CNPS9900 MAX を使用した．この CPU の最大周波数は，通常時が 3.2 GHz なのに対し，ターボ・モードによって最大 3.8 GHz にまで上昇する．なお，CPU ファンを用いた時のアクティブ・コアの温度は 40 度前後であるのに対し，ファンを停止した時のそれは 90 度前後であった．

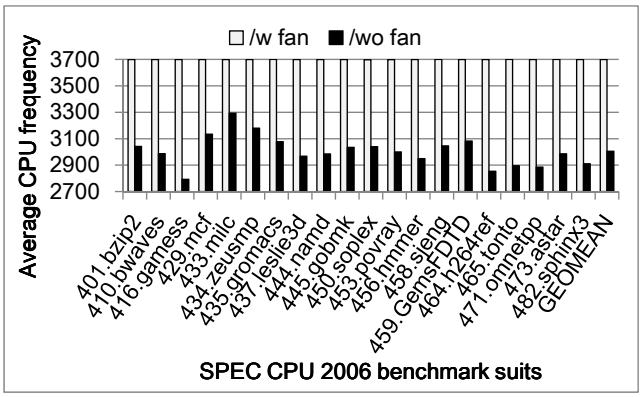

図 1 ターボ・モード中の平均動作周波数

図 1 より，まず，ファンを使用した場合（「/w fan」）は，ターボ・モード中であっても温度制約違反がほとんど起きないことがわかる．どのプログラムも，ターボ・モード中の平均動作周波数はほぼ 3.7 GHz を示している．これは，上記 CPU において 3 コアがアクティブ状態にある時の，ターボ・モード時に利用できる最大周波数と等しい．したがって，上記の実験中は 3 コアが常時アクティブ状態にあったと推測される．

一方，ファンを停止した場合（「/wo fan」）は，チップ温度が上昇した結果，ターボ・モード中の周波数が大幅に抑制されていることがわかる．周波数の抑制幅はプログラムによって異なり，429.mcf のようなメモリ・バウンドなアプリではあまり抑制されない一方，464.h264ref のような CPU バウンドなアプリでは抑制幅が大きい．これは，CPU バウンドなアプリの方がコアの消費電力が大きく，チップ温度が上昇しやすいためである．

詳細は紙面の都合により省略するが，上記の周波数抑制の結果，「/w fan」と「/wo fan」におけるプログラムの実行時間の差は最大で 32.6% にも達する．このように，温度制約違反を起こすと性能が大幅に低下することから，現在流通している CPU は，典型的な使用状況では例えターボ・モード中であっても制約違反を起こさぬように，出荷前に行われる入念なテストを通してターボ・モード時に利用可能な最大動作周波数を設定していると考えられる．具体的には，さまざまな周波数と電圧の組み合わせで CPU 負荷の高いプログラムを実行し，その時のチップ温度と上限温度とを比較し，一定の温度マージンを持つものの中から最適な組み合わせを選択しているものと考えられる．したがって，AM によってチップ温度の上昇を抑えることができれば，出荷時により高い周波数と電圧の組み合せわを

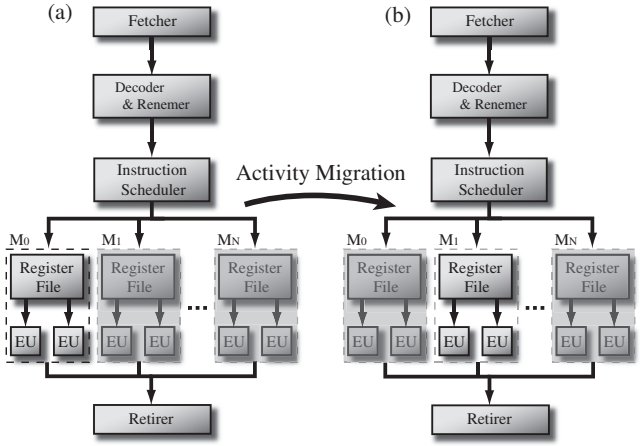

図 2 AM のコンセプト

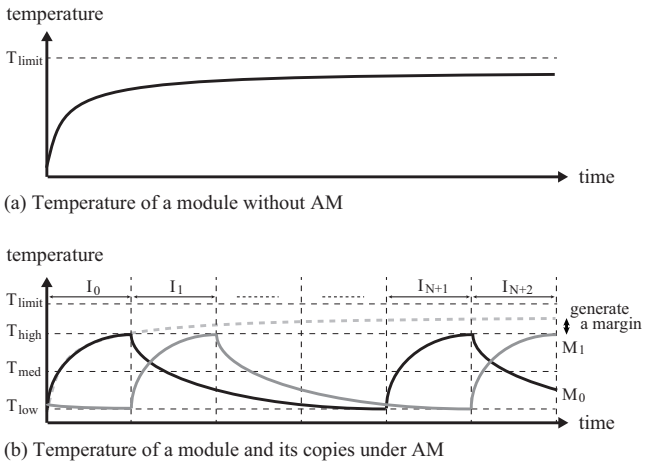

図 3 ユニットの温度変化

選択できるようになり，その結果，ターボ・モードの効果を今以上に高めることができる．

3. アクティビティ・マイグレーション

本章では，チップ温度の上昇を抑制する技術である AM について詳しく述べる．

3.1 概要

図 2 に AM を行うプロセッサのブロック図を示す．図では実行ユニット群とレジスタファイルがマイグレーション対象となっている場合を想定している．図に示すように，AM を行うプロセッサでは，対象となるユニットの N 個のコピーが設けられ，それらが排他的に利用される．図では非アクティブなユニットを影付きで表現してある．

AM を行うプロセッサは，チップ温度が閾値に達するまでは，プログラムを普通に実行する．図 2(a) はこの状況を表したものである．図ではユニット M_0 が計算に使用され，残りのユニットは非アクティブ状態にある．プログラムが実行されることで M_0 の温度は徐々に上昇する．

ユニットの温度が閾値に達すると，プロセッサはパイプラインを停止し，計算に使用するユニットを別のものへと切り替える．ユニットの切り替えは温度に基づいて行うのではなく，一定サイクルおきに定期的に行われることもある [8]．図 2(b) ではユニット M_1 が次に使用するユニットとして選択されている．次に使用するユニットが選択されると，レジスタ値など計算の継続に必要なプロセッサ状態のコピーが行われる．

上記の処理が完了すると，プロセッサは実行を再開する．使用を中止したことにより，それまで使用していたユニットの温度は徐々に低下する．上記を繰り返すことで，AM はチップ温度の上昇を抑えることができる．

図 3 にユニットの温度変化を表す．AM を行わない場合は，図 3(a) に示すように，プロセッサがユニットを使い続けることでユニットの温度は時間とともに上昇し，最終的には飽和する．一方，AM を行った場合は，上述の飽和温度に達する前に使用するユニットの切り替えが行われる．その結果，図 3(b) に示すように，ユニットのピーク温度は上記の温度よりも低いある値（T_{high}）で抑えられる．

上述のように AM はピーク温度と LSI が動作可能な上限温度との間に大きなマージンを生み出すことができるため，その分を動作周波数の向上に利用することができる．例えば，元論文ではリング発振器に AM を適用した場合に動作周波数が 15.9% 向上することが報告されている [8]．しかし，実際にプロセッサに AM を適用した場合にどの程度の周波数向上が得られるかはまだわかっていない．

AM はチップ温度の上昇を抑える有効な手段であるが，以下に述べる 2 つのデメリットがある．

性能ペナルティ　AM は以下の 2 つの理由により IPC を低下させる．

1 つ目は，使用ユニットを切り替える際，命令をフラッシュし，プロセッサ状態をコピーしなければならないことによる．これらの処理には，AM を行うユニットにもよるが，数十サイクルを要する．そのため，頻繁な切り替えは IPC を大きく損なうことになる．

もう 1 つは，複製の追加によってチップのレイアウトが変化し，ユニット間の距離が拡大してしまうことによる．つまり，AM を行うプロセッサでは，ユニット間に通常よりも長い配線が必要となる．それでも動作周波数を維持しようとするのであれば，長くなった配線に対して更なるパイプライン化を施さねばならない．その結果，IPC は低下することになる．

面積増加　明らかに AM はチップ面積を増加させる．LSI の微細化が続くと使い切れないトランジスタが生まれると予想されているが [4], [6], [13], [19]，プロセッサの実装面積を抑えることは依然として重要なテーマと言える．なぜなら，AM による面積増加を抑制できれば，ダイ上の空いた部分を他の目的（例えば専用ハードウェアの搭載 [6], [19]）に使用できるからである．

3.2 従来の AM 研究とその問題点

これまでの AM 研究においては，経験的に AM の対象と複製数が決められていた．すなわち，ホット・スポットとして知られているユニットを AM の対象とし，それらの複製を 2 ～ 4 個程度追加するということが行われてきた．しかも，AM の空間粒度はフェッチャやリネーマというかなり大きな粒度であった．細粒度 AM は省面積かつ高性能なプロセッサを実現できると考えられるが，その効果を評価したものはまだないのが現状である．

Heo らは文献 [8] において 4 つのタイプの AM を提案している．コア，コア内の L1D キャッシュと TLB を除いた全ユニット，上記ユニットからさらにフェッチャを除いた全ユニット，レジスタ・ファイルと実行ユニットからなるユニット群，それぞれを AM 対象とした 4 つである．どの AM も複製数は 2 個である．彼らは 4 つのタイプそれぞれの IPC 低下率のみを評価し，4 番目の AM が最も面積効率がよいと結論づけている．しかし，彼らは AM によって上記のプロセッサのピーク温度がどの程度低下し，それにより周波数がどの程度向上するかまでは評価していない．

Chaparro らは AM をクラスタ化プロセッサに適用することを提案した [2], [3]．彼らは 2 つのクラスタ間で AM を行った場合に，IPC が 14% 低下するものの，チップ温度を最大 33% 低下できることを示した．しかし，上記の結果によって周波数がどの程度向上し，その結果プロセッサ性能がどの程度改善するかまでは評価していない．また，彼らは Heo らの方法との比較を行っておらず，彼らの方法が面積効率の点で本当に優れているのかは不明である．

我々は，こうした経験的な方法では面積効率に優れたプロセッサの実現は難しいと考える．なぜなら，経験的な方法では熱源ではないユニットを AM の対象として選んでしまう可能性を排除できないからである．また，経験的な方法では，複製を何個用意すればよいかを決定するのも難しい．よりよい構成を探す場合は総当たり的な方法を採らざるを得ないが，そうした方法は AM の粒度が細かくなるにつれ困難となる．これについては次章で詳しく述べる．

3.3 AM の空間粒度

実は，細粒度 AM は，粗粒度 AM と比べてほとんどデメリットがない．AM の対象を適切に選択できれば，細粒度 AM はチップ内の温度ばらつきをより抑えることができる．しかも，細粒度 AM ではピーク温度の抑制に不要なユニットが複製されてしまうのを防ぐことができるため，少ない面積増加で高い温度抑制効果が得られる．

唯一とも言える欠点は，細粒度 AM はクリティカルな配線を長くする可能性があることである．例えば，ある ALU を AM の対象とした場合，その複製を追加することで ALU 群全体の面積は増加してしまう．このことは ALU 間の距離が増大することを意味する．その結果，オペランド・バイパスを 1 サイクル以内に行うのが困難となるかもしれない．さらに，いくつかの ALU はレジスタ・ファイルとの間に長い配線が必要となり，その結果，レジスタ・アクセス・レイテンシが増加する可能性がある．

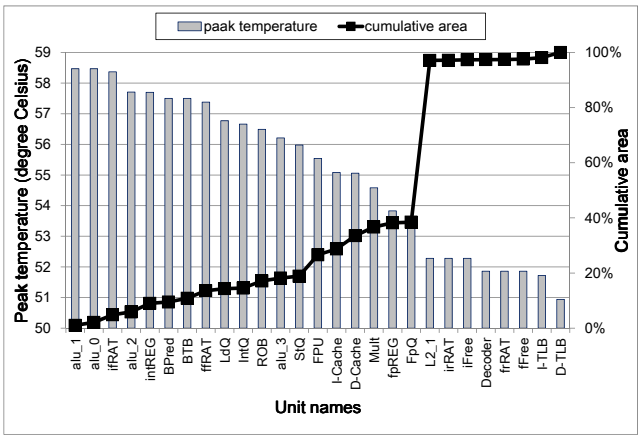

図 4 各ユニットのピーク温度とコア内に占める面積割合

実は，上記のバイパスの問題は，同時に使用する ALU 群を適切に選択することで緩和できる．物理的に隣り合った ALU をペアとして使用すればよい．ALU の AM では同時には使用されない ALU 間のバイパスは必要ないため，この方法により長いバイパスを排除できる．また，後述するように，実際のプロセッサでは，ユニット・レベルの AM ではクリティカルなユニットのパイプライン化は必要ない．そのため本論ではユニット・レベルの AM に着目する．

4. 本研究のモチベーション

全てのホット・ユニットが熱源となるわけではない．例えば，ALU とレジスタ・ファイルはホット・スポットの 1 つとして知られているが，これらのユニット群すべてが等しく発熱するわけではない．大量の熱を発生するユニットがある一方，そうではないユニットもある．面積効率を考えた場合，前者のみを AM の対象とした方がよい．

図 4 に各ユニットのピーク温度とコア内に占める面積を示す．プロセッサ構成等の実験方法の詳細は 6 章で述べる．グラフの横軸はユニット，縦軸は温度または面積割合を表す．棒グラフは温度を，折れ線は面積を表している．

グラフより，多くのユニットがよく似た高い温度を示していることがわかる．alu_1 から BTB までの上位 7 つのユニットの温度差は 1 度以内である．これらのユニットの面積はコア面積の 10.8% を占めている．したがって，仮にこれらのユニット全ての複製を 1 つずつ用意したとすると，チップ面積は 10.8% も増加することになる．

全ての高温ユニットが熱源ではないことを**図 5** に示す．図は図 4 のプロセッサが 456.hmmer の実行を終えた時の温度マップである．図より，intREG（整数レジスタ・ファイル）と BTB は高温であるが，それは周囲のユニットによって温められた結果であることが伺える．BTB は ifRAT

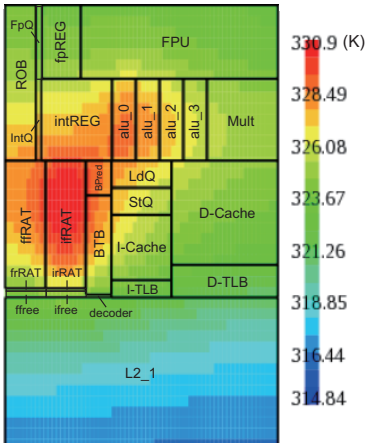

図 5 4 のプロセッサが 456.hmmer の実行を終えた時の温度マップ

(整数フロントエンド・レジスタ・エイリアス・テーブル) によって，intREG は ifRAT と alu_0 によって温められたように見える．

前述のように，AM は対象となるユニットの使用を停止することによりその消費電力を抑え，ユニットの温度上昇を抑える技術である．したがって，熱源ではない，周囲のユニットに温められることで高温となったユニットを複製することは，チップ温度の低下にあまり役に立たない．面積を浪費するだけである．そのため，ifRAT のような熱源となるユニットを特定し，それのみを AM の対象とすることが求められる．

3.2 節で述べた経験的な方法では，熱源となるユニットを正確に検出するのは難しいと考える．経験的な方法では，intREG のような無駄なユニットを複製の対象に選んでしまう可能性がある．さらに，経験的な方法では，新しく開発するプロセッサのように，そもそもどのユニットが高温となるかがわからない場合には，AM の対象を選ぶことができない．加えて，複製を追加することによってユニットの温度がどのように変化するかがわからないため，何個の複製を追加すればよいかがわからない．粗粒度 AM であれば総当たりによって AM の対象と複製数を求めることもできるかもしれないが，細粒度 AM のように AM の適用対象となるユニットが増えてくると総当たりは現実的ではない．そのため，熱源となるユニットを発見し，適切な複製数を決定する方法が必要とされている．

5. 細粒度 AM を用いるプロセッサの設計手法

本章では細粒度 AM を行うプロセッサの設計法を述べる．提案手法では，AM を行わない通常のプロセッサに対し，熱源となるユニットを特定し，それらの適切な複製数を決定する．そして，それを元に現在の設計を変更する．これらはすべてアーキテクチャ設計の段階で行う．

熱源の特定と複製数の決定は，いずれも我々が提案する温度モデルを用いて行う．以下，詳しく述べる．

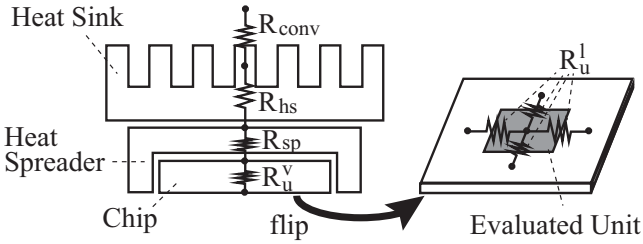

図 6 AM を行うプロセッサ設計のための温度モデル

5.1 AM を行うプロセッサ設計のための温度モデル

熱源を特定するためには，各ユニットがどの程度温度を上昇させる能力を有しているかを見積もることが重要である．このために，**図 6** に示すように，チップ内に当該ユニットのみが存在する状況を想定し，その時の温度上昇を見積もることにした．周囲のユニットからの熱伝導を排除することにより，ユニット自身が持つチップを温める能力を見積もることができる．

定常状態における物質の温度 T は以下の式で定義される．

$$T = R \times P + T_{amb} \tag{1}$$

ただし，R は物質の熱抵抗，P は物質のピーク消費電力，T_{amb} は周辺温度である．したがって，上記の式は物質の温度に関する保守的な見積もりを提供する．

ここで評価対象のユニットと周囲のユニットは独立した物質と見なすことができる．その場合，上記の式において T_{amb} は周囲ユニットの温度に相当し，$R \times P$ はユニット自身の発生する熱による温度上昇を意味する．そのため，この $R \times P$ が大きいユニットを熱源と見なすことにした．

ユニットの熱抵抗 R_u は以下のように表せる [10], [18].

$$R_u = \frac{1}{1/(R_{conv} + R_{hs} + R_{sp} + R_u^v) + 4/(R_u^l)} \tag{2}$$

ただし，R_{conv} は対流熱抵抗，R_{hs} はヒート・シンクの熱抵抗，R_{sp} はヒート・スプレッダの熱抵抗，R_u^v はユニットの垂直方向の熱抵抗，R_u^l はユニットの水平方向の熱抵抗である（図 6）．

紙面の都合により説明の詳細は省略するが，上記の熱抵抗はそれぞれ以下の形で記述できる．

$$R_{conv} = \frac{1}{h \times A_u} \tag{3}$$

$$R_{hs,sp} = \frac{t}{k_{hs,sp} \times A_u} \tag{4}$$

$$R_u^v = \frac{t}{k_u \times A_u} \tag{5}$$

$$R_u^l = \frac{c}{t \times sqrt(A_u)} \tag{6}$$

ただし，A_u はユニットの面積，$h, t, k_{hs,sp,u}, c$ はそれぞれの物質に固有の正定数である．

以上をまとめると，ユニット自身の熱による温度上昇 $T_{self}(= R \times P)$ は以下のように記述できる．

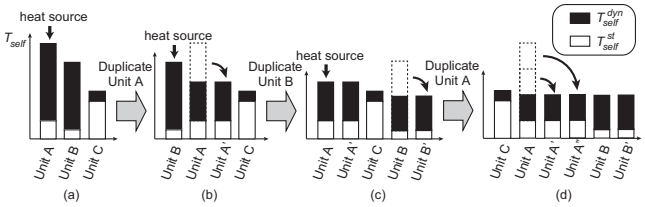

図 7 温度モデルを用いた熱源の特定と複製数の決定方法

$$T_{self} = (\alpha + \frac{1}{\beta \times sqrt(A_u) + \gamma}) \times \frac{P_u}{A_u} \quad (7)$$

ただし，P_u はユニットのピーク消費電力，α, β, γ はいずれも正定数である．これらの正定数の値はカーブ・フィッティングにより求める．

複製追加の効果を見積もる際は，ユニットのダイナミック電力による温度上昇を考えることが重要である．上記の式より，T_{self} はユニットの消費電力に比例することから，ダイナミック電力による温度上昇（T^{dyn}_{self}）とスタティック電力によるそれ（T^{st}_{self}）との単純な和として表せる．

計算に必要な P_u と A_u は McPAT[14] のようなプロセッサの電力/面積シミュレータを用いて求めることを想定している．そうして求めた値と上記のモデルを用いて，熱源の特定と複製数の決定を行う．次節ではその方法を述べる．

5.2 熱源の特定と複製数の決定方法

図 7 に示すように，我々は熱源の特定と複製数の決定を段階的に行う．図は 3 つのユニット（A, B, C）からなるプロセッサに対して提案手法を適用した場合を表している．図の黒い棒グラフは各ユニットの T^{dyn}_{self} を，白い棒グラフは T^{st}_{self} を表す．

提案アルゴリズムは以下のようなものである．まず，前節で述べた方法により，すべてのユニットの T_{self} を計算し，それらを昇順にソートする（図 7(a)）．その結果，現在の構成において最も支配的な熱源が特定できる．図 7(a) ではユニット A がこれにあたる．そこでこのユニットを AM の対象と見なし，当該ユニットの複製を 1 つ追加する．

続いて，得られた新しいプロセッサ構成に対し，再度上記の手続きを行う．理想的な AM の元ではすべての複製は公平に使用されることから，$N-1$ 個の複製が追加された場合の T^{dyn}_{self} は追加前の $1/N$ となる．T^{st}_{self} は元のままであることに注意されたい．したがって，ユニット A の複製を 1 つ追加したプロセッサでは，図 7(b) に示すように，ユニット B が一番の熱源となる．そこで，ユニット B を次なる AM の対象とし，その複製を 1 つ追加する．

上記の手続きを数回繰り返すことで，さまざまな複製を持ついくつかのプロセッサ構成が得られる．これらの設計の候補に対して詳細なシミュレーションを行い，最もよいと思われるものを最終的な設計として選択する．

提案手法ではいくつかの候補に対する詳細シミュレーションが必要とされるが，その候補数は現実的にはそれ程多くないと考えられる．なぜなら，後述するように，実際のプロセッサでは熱源となるユニットはそう多くはないからである．また，評価の章で示すように，複製の追加は面積効率を急激に悪化させる．現実的には，上記の手続きは 1 ～ 2 回程度繰り返せば十分である．

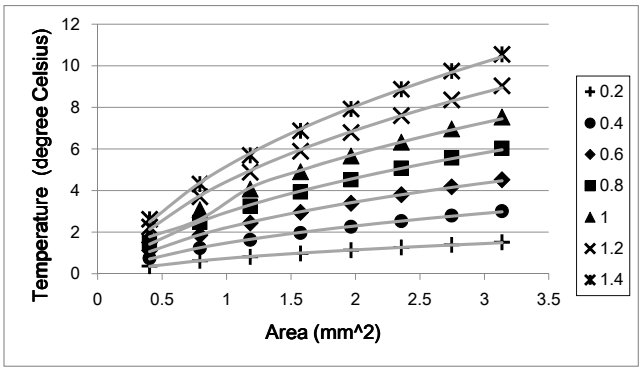

図 8 提案モデルに対するカーブ・フィッティングの結果

6. 評価方法

まず，提案モデルの 3 つの正定数（α, β, γ）をカーブ・フィッティングによって求めた．チップの中央に上方から見て正方形となるユニットを 1 つ置き，その面積と電力密度を変化させることによって学習に用いるサンプリング点を得た．面積は 0.403 から 3.14 mm^2 の範囲で，電力密度は 0.2 から 1.4 W/mm^2 の範囲で変化させた．各サンプリング点に対し，定常状態におけるユニットの温度を HotSpot 5.0[10] を用いて計算した．シミュレーション条件を**表 1** に示す．このようにして得られた 58 個の学習データに対し，カーブ・フィッティングを行った．その結果を**図 8** に示す．図より，提案モデルはユニットの温度上昇を正しくモデリングできていることがわかる．なお，図は $\alpha = 86.24, \beta = 0.00075, \gamma = 0.011364$ の時の結果である．

次に，このモデルを用いて，**表 2** に示すベース・プロセッサ（BASE）に対して細粒度 AM を行うプロセッサを

表 1 チップと冷却機構の評価条件

Parameters	Remarks
Chip conditions	
Process technology	45nm
Thickness	0.625mm
Ambient temperature	40 °C
Silicon thermal conductivity	100W/(m·K)
Heat sink conditions	
Side	60mm
Thickness	6.9mm
Thermal conductivity	400W/(m·K)
Heat spreader conditions	
Side	32mm
Thickness	1.87mm
Thermal conductivity	400W/(m·K)

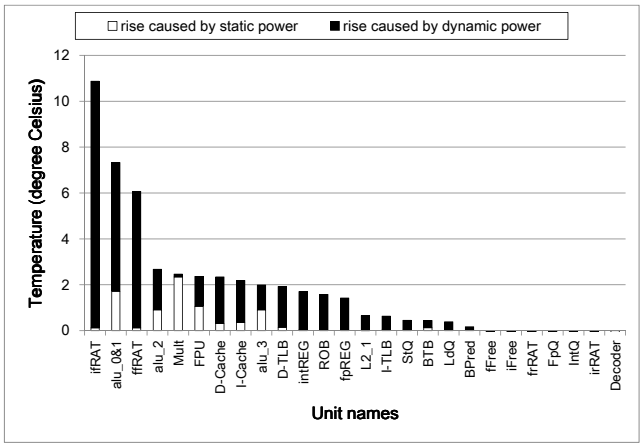

図 9 BASE における T_{self} の見積もり結果

設計する．本論ではシングル・スレッド・プログラムが 1 コアで稼働し，それがターボ・モードによって加速される状況を想定することから，BASE はシングル・コア・プロセッサとした．基本 CPU 周波数は 3 GHz とした．

BASE の各ユニットの消費電力と面積の見積もりは，改良した McPAT 0.8[14] と Onikiri2[16] を用いて行った．McPAT の元々のコードにはいくつかのバグ（例えばメモリ・アレイの評価関数やデコーダの負荷容量の計算）があるため，我々のシミュレータではそれらの修正を行ってある．また，McPAT が算出する消費電力は，ユニット内のトランジスタがすべてスイッチングした場合のピーク値であり，過剰な値である．実際の使用状況に照らした消費電力を算出するためには，各ユニットのアクティビティを掛け合わせる必要がある．このために，改造した Onikiri2 を用いた．SPEC CPU 2006 を実行し，1M サイクル毎の全ユニットのアクティビティを計算した．そうして，ユニット毎に最も高かったアクティビティを求め，それと McPAT が計算したピーク消費電力を掛け合わせたものをそのユニットの実効的なピーク消費電力とする．

図 9 に BASE における各ユニットの T_{self} を示す．横軸はユニットを，縦軸は温度を表す．黒い棒グラフは T_{self}^{dyn} を，白い棒グラフは T_{self}^{st} を表している．いくつかの CPU バウンドなプログラムでは IPC が 2 を超えるため，alu_0 と alu_1 はまったく同じアクティビティを示す．両者は普通は隣り合って実装されるため，これら 2 つを 1 つの AM 対象と考えてよく，図ではそのように見なしている．図に示すように，BASE において最も支配的な熱源は ifRAT，次いで統合した ALU である．それぞれの T_{self} は 10.9, 7.33 度であった．そこで，このプロセッサに対して 5.2 節で述べた手続きを 2 回適用し，2 つの細粒度 AM を行うプロセッサ（PRO1, PRO2）を得た．その構成を**表 3** に示す．

上記のプロセッサを粗粒度 AM を行うプロセッサと比較する．文献 [8] より，すべての実行ユニットとレジスタ・ファイルを複製する方式（AM-D）を評価する．これは，上記文献の著者らがこの方式が最も面積効率がよいと主張しているためである．文献 [2], [3] からは，クラスタ間の AM を行う方式を比較対象とする．IPC を維持するため，バックエンドをクラスタ化する代わりに，バックエンドの複製を追加する（この方式を CLST とする）ことでクラスタ間の AM を実現する．これらの方式で複製されるユニットは，具体的には**表 4** のようになる．

上述したすべてのプロセッサのフロアプランを設計した．BASE のフロアプランは図 5 で既に示していることから，残りのプロセッサのフロアプランを**図 10** から**図 13** に示す．なお，図において複製は灰色で色づけしてある．これらのフロアプランは，基本的には 4 コアの Nehalem プロセッサのフロアプラン [7] を元に設計した．CLST に関しては，基本プロセッサのレイアウトの差を考慮しつつ，文献 [3] のフロアプランを参考に設計した．なお，各ユニットの面積は McPAT により算出した．

配線長の増加の影響は次のようにして見積もった．フロアプランニングにおいては，一般に，ユニット間を結ぶ配線の長さはユニット間のマンハッタン距離で近似される．そこで，まずは BASE の中で最も長いマンハッタン距離を有する接続関係にある 2 つのユニットを求める．その結果，ffRAT と FpQ（浮動小数点命令キュー）間の接続に最

表 2 ベース・プロセッサの構成

Parameters	Remarks
Fetch/decode/commit	4 instructions
Rename Latency	2 cycles
Dispatch Latency	2 cycles
Issue Latency	4 cycles
Instruction Queue	INT:32, FP:16
Load/Store Queue	LD:32, ST:32
Exec units	Int:4, FP:1, Mem:2
Register files	INT:128, FP:128, 1 cycles
L1I/D caches	64KB, 16B/line, 2 way, 2 cycles
L2	2MB, 16B/line, 8 way, 8 cycles
Memory	66.7 ns (200 cycles at 3.0 GHz)
Frequency	3.0 GHz (normal)

表 3 提案手法により設計された細粒度 AM を行うプロセッサ

Name	Remarks
PRO1	with one copy of ifRAT (the first configuration in our algorithm)
PRO2	with one copy of ifRAT and the unified ALUs (the second configuration in our algorithm)

表 4 比較対象の粗粒度 AM

Name	Remarks
AM-D	Register files (INT and FP), four ALUs, multiplier and FPU are doubled
CLST	Schedulers (INT/FP and load/store queues) and D-Cache are doubled in addition to AM-D

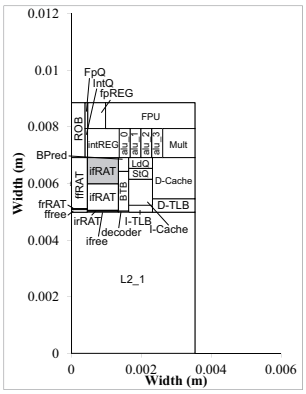

図 10 PRO1 のフロアプラン

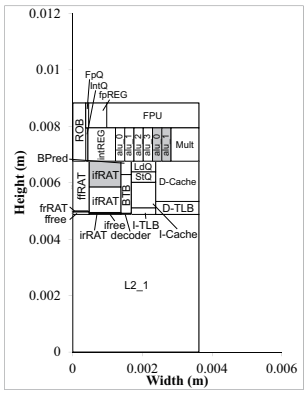

図 11 PRO2 のフロアプラン

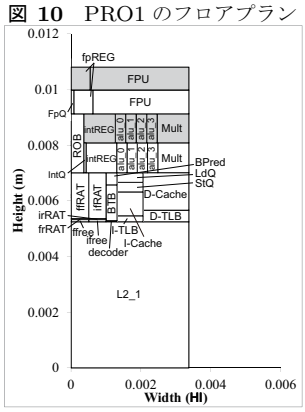

図 12 AM-D のフロアプラン

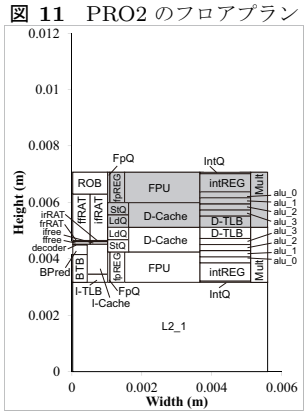

図 13 CLST のフロアプラン

も長い配線が必要であることがわかった．我々は，ディスパッチ・レイテンシ（2 サイクル）の半分がこの配線の遅延であり，これが BASE における 1 サイクルで信号伝搬可能な最長の配線長と見なすことにした．次いで，図 10 から図 13 の各フロアプランに対し，接続関係にあるすべてのユニットのマンハッタン距離を計算し，BASE のそれと比較した．そして，マンハッタン距離が BASE よりも大きくなった配線に対し，その長さが上述の 1 サイクルでの信号伝搬を許容できる配線長を上回っていた場合は，その配線に追加のパイプライン化が必要とみなすことにした．こうして見積もった各ユニットのパイプライン段数の増分を**表 5** の第 2 列に示す．

表の第 3 列に各プロセッサのマイグレーション・オーバヘッドをまとめる．RAT のマイグレーション（PRO1, PRO2）には 9 サイクルのストールが発生するものとする．

表 5 AM を行うプロセッサのパイプライン段数とマイグレーション・オーバヘッド

Name	Additional stage	Migration overhead
PRO1	rename+1	9 cycles stall
PRO2	rename+1, IntExec+1	9 cycles stall, reschedule
AM-D	dispatch+1, issue+1	33 cycles stall, reschedule
CLST	dispatch+4, IntExec+1	33 cycles stall, reschedule, D-Cache flush

これは，BASE プロセッサの RAT が 12 リード/4 ライト可能な 32 エントリの CAM であり，そのレイテンシが 1 サイクルだからである．バックエンド・パイプラインの停止が必要な方式（PRO2 以下）に関しては，マイグレーション時にすべてのインフライト命令がフラッシュされ，実行再開後に再発行されるものとする．レジスタ値のコピーには 33 サイクル要するものとした．これは，レジスタ・ファイルが 12 リード/4 ライト可能な 128 エントリの RAM で構成されており，そのレイテンシが 1 サイクルだからである．また，文献 [3] にしたがい，CLST のマイグレーション時には L1D キャッシュをフラッシュする..

前述のように，ターボ・モード時に利用可能な最大周波数は，通常は，出荷前に行われる温度試験を通して決定される．2 章でみたように，現在流通している CPU の多くは，この最高周波数で動作した時のチップ温度と上限温度との間に十分なマージンを設けているようである．今回の実験では，最も CPU バウンドなプログラムである 456.hmmer を温度試験用のプログラムと見なし，これが実行された時の最高温度と上限温度（90 度 [12]）との間のマージンが 15 度となるように最大動作周波数が設定されるものと想定した．そして，AM により，この最大動作周波数がどの程度向上し，その結果，プロセッサ性能がどの程度改善するかを評価した．IPC の評価には Onikiri2 を，面積と電力評価には McPAT を，温度評価には HotSpot を使用した．

その他のシミュレーション条件を以下にまとめる．クロック周波数と電源電圧は比例関係にあると仮定した．具体的には，$V = 0.19F + 0.62$（V は電圧，F は周波数）の式にしたがうものとする．我々はこの式が十分な精度を有することを実機を用いて確認した．評価には SPEC CPU 2006 の中から 24 本のプログラムを使用する．使用しない残りの 5 本については，我々の環境でコンパイルできなかったか，もしくは，Onikiri によるシミュレーション中にアボートしたため，評価の対象外とした．各プログラムにつき，最初の 1G サイクルをスキップし，残りの 6G サイクルを評価の対象とした．SimPoint[17] を使用しなかったのは，温度評価には連続する十分な長さの電力トレースが必要だったためである．BASE のメモリ・アクセス・レイテンシは 200 サイクルであるが，この値は CPU 周波数に応じて変化させる．これは，ターボ・モード中は CPU 周波数のみがクロック・アップされ，メモリ周波数は変化しないことによる．マイグレーション間隔は，元論文にしたがい [8]，100K, 1M, 10M サイクルとした．したがって，マイグレーションによるパイプラインのストール率は無視できる程小さい（最悪 0.033%）．

7. 評価結果

表 6 に各プロセッサが利用可能な最大動作周波数を示す．なお，表の結果は，マイグレーション間隔に関して面

積効率が最も優れていたもののそれである．表に示すように，細粒度 AM は粗粒度 AM とほぼ同等の周波数ゲインを達成している．これは，粗粒度 AM では，BASE の最も支配的な熱源である ifRAT が複製されていないことによる．一方で提案手法では ifRAT を AM の対象としたことにより，少ない面積増加で高いピーク温度の抑制効果が得られていることがわかる．

図 14 に各プロセッサがターボ・モードによって最大周波数で動作した時のスループット（Instruction Per Second）を表す．グラフの横軸はプログラム，縦軸はスループットである．5 つの棒グラフは，左から順に，BASE，AM-D，CLST，PRO1，PRO2 を表している．ただし，図のスループットはすべて BASE のそれで正規化してある．

図より，細粒度 AM を行うプロセッサは他よりも高い性能を示すことがわかる．AM-D と CLST の BASE に対する相対スループットは平均で $-0.17\%, 4.17\%$ だったのに対し，PRO1 と PRO2 のそれは $9.98\%, 6.02\%$ と改善している．スループットの改善率は特に CPU バウンドなプログラムで大きく，456.hmmer においては，PRO1 は BASE よりも 14.5% も高いスループットを示している．これは，CPU バウンドなプログラムの方がターボ・モードによる CPU 周波数向上の恩恵を受けやすいためである．

図 15 に各プロセッサの IPC を示す．グラフより，CLST は平均で 9.28%，最悪 18.1%（433.milc）と大きな IPC 低下を引き起こす．このように IPC 低下が大きいため，CLST は周波数ゲインは大きい（表 6）ものの，スループットはあまり向上していない．これとは対照的に，PRO1 と PRO2 の IPC 低下率は平均で $4.21\%, 4.59\%$ と低いため，図 14 に示したような大きなスループット改善を達成できている．

表 7 にマイグレーション間隔を変えた時の相対スループット（整数系プログラムの結果の平均値）を示す．マイグレーション・オーバヘッドが無視できる場合は，マイグレーション間隔が短くなる程，チップの温度上昇を抑えることができ，スループットは改善する．PRO1 の結果はこの状況を表したものである．一方，CLST はこのような傾向を示していない．これは，CLST はキャッシュがフラッシュされるため，マイグレーション間隔が短いとそれによる性能ペナルティが無視できなくなるためである．また，AM-D は性能がほとんど改善していない．これは，AM-D は熱源となる ifRAT をマイグレーション対象としておらず，実行ユニットやレジスタ・ファイルの複製を追加してもチップ温度はほとんど変わらなかったことによる．

表 8 に各プロセッサの面積をまとめる．表に示すように，粗粒度 AM は大きなチップ面積の増加を引き起こす．これは，intREG や乗算器といった，熱源ではないユニットもマイグレーション対象として複製してしまったことによる．一方，当初の期待通り，細粒度 AM は少ない面積増加に抑えることができている．PRO1, PRO2 それぞれの面積増加は $2.8\%, 4.8\%$ であった．

以上の結果から，細粒度 AM は高性能かつ省面積なプロセッサを実現する有望な方法であると言える．

表 6 各プロセッサが利用可能な最大動作周波数

Processor	CPU frequency (relative frequency)
BASE	5.4 GHz (×1)
AM-D	5.6 GHz (×1.04)
CLST	6.2 GHz (×1.15)
PRO1	6.2 GHz (×1.15)
PRO2	6.0 GHz (×1.11)

表 7 マイグレーション間隔を変えた時の BASE に対する相対スループット（表は整数系プログラムの結果の平均値）

Name	15 C margin		
	100K	1M	10M
AM-D	0.997	0.998	0.998
CLST	1.022	1.043	1.023
PRO1	1.100	1.074	1.019
PRO2	1.024	1.059	0.978

表 8 各プロセッサの面積

Name	Area (mm^2)	Relative area
BASE1&2	30.481	1
AM-D	36.400	1.194
CLST	39.413	1.293
PRO1	31.327	1.028
PRO2	31.956	1.048

8. 関連研究

Karpuzcu らは，我々と同様，プロセッサの温度制約に着目した高速化手法を提案している [13]．彼らは大量のコアをプロセッサに搭載し，それらを使い捨てにする手法を提案した．アクセラレーション時には，コア・プールの中から 1 つを選択し，それが壊れる程の大きな電圧と周波数を用いてプログラムを実行する．壊れた場合は，使用するコアをまだ壊れていないものへと切り替える．この方法はコア単位の AM の一種と見なすことができる．この方法は多くのプログラムの性能を改善するが，明らかな欠点はハードウェア寿命が低下することである．

従来の DTM においても，ホット・ユニットを複製することでチップを冷却する方法は提案されてきた．しかし，複製の選択はいずれもケース・スタディ的に行われていた．Skadron らは，温度シミュレーションの結果から，整数レジスタファイルを複製することを提案した [18]．Chaparro らは，クラスタの温度に着目し，クラスタ・ホッピングと呼ばれる使用するクラスタを定期的に切り替える手法を提案した [2], [3]．これらの方法とは異なり，我々の方法では温度モデルに基づいて複製対象と複製数を適切に決定する．

AM 以外にもチップの温度上昇を抑制する手法がいくつか提案されている．Bailis らはプログラムの実行中にアイ

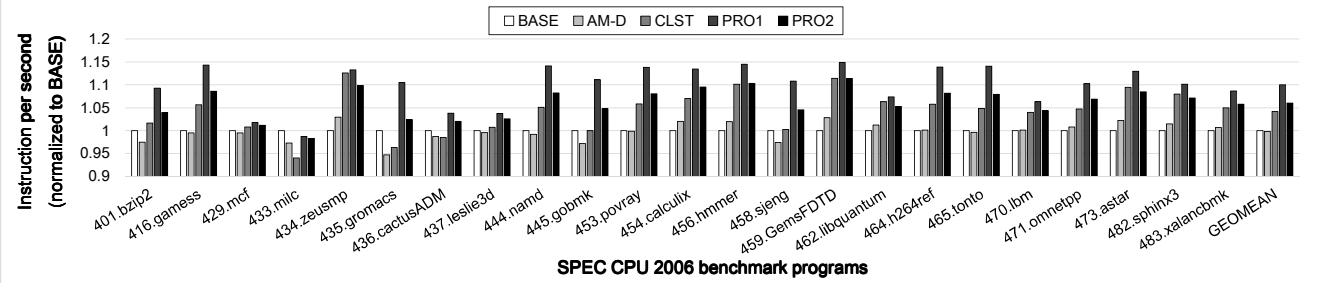

図 14 各プロセッサのスループット

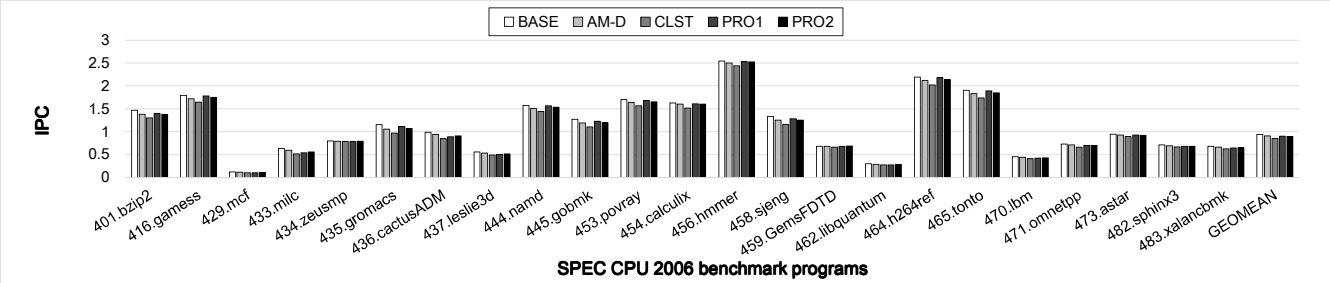

図 15 各プロセッサの IPC

ドル・サイクルを強制的に挿入することで，DVFS ベースの DTM を行うプロセッサにおいて高い性能を達成できることを示した [1]．Powell らはいくつかのユニットの内部回路を公平に使用する方法を提案している [15]．これらの方法は，AM とは異なりチップ面積の増加を引き起こさないものの，その温度抑制効果は限定的である．

9. まとめ

本論ではユニット・レベルの AM とそれを適用したプロセッサの設計手法を提案した．粗粒度 AM とは異なり，ユニット・レベル AM はチップ温度の抑制に不要なユニットの複製を防ぐことができる．チップ内では限られた少数のユニットのみが熱源となることから，ユニット・レベル AM は面積増加を抑えつつ高い性能を達成できる．上記の事実を詳細な評価により確認した．

LSI の微細化が進むにつれユニットの電力密度は増加する．そのため，ユニット・レベル AM に代表されるチップの温度上昇抑制手法は，今後ますます重要な役割を担うようになるだろう．

謝辞 本研究の一部は科学研究費補助金（課題番号 24700044）による．

参考文献

[1] Bailis, P. et al.: Dimetrodon: Processor-level Preventive Thermal Management via Idle Cycle Injection, *DAC-48*, pp. 89–94 (2011).
[2] Chaparro, P. et al.: Thermal-Aware Clustered Microarchitectures, *ICCD*, pp. 48–53 (2004).
[3] Chaparro, P. et al.: Thermal-Effective Clustered Microarchitectures, *TACS-1* (2004).
[4] Esmaeilzadeh, H. et al.: Dark silicon and the end of multicore scaling, *ISCA-38*, pp. 365–376 (2011).
[5] Funck, M. and Peterson, R.: Performance Implications of POWER7 Model 780's TurboCore Mode (white paper).
[6] Goulding-Hotta, N. et al.: GreenDroid: An Architecture for the Dark Silicon Age, *ASP-DAC*, pp. 100–105 (2012).
[7] Hennessy, J. L. and Patterson, D. A.: *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, 5 edition (2011).
[8] Heo, S. et al.: Reducing power density through activity migration, *ISLPED*, pp. 217–222 (2003).
[9] Hewlett-Packard Corp. et al.: Advanced Configuration and Power Interface Specification rev. 4.0 (2010).
[10] Huang, W. et al.: HotSpot: A compact thermal modeling method for CMOS VLSI systems, *IEEE Trans. on CPMT*, Vol. 14, pp. 501–513 (2006).
[11] Intel Corp.: http://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html?wapkw=turbo+boost.
[12] Intel Corp.: CPU Monitoring with DTS/PECI (white paper) (2010).
[13] Karpuzcu, U. R. et al.: The BubbleWrap Many-Core: Popping Cores for Sequential Acceleration, *MICRO-42*, pp. 447–458 (2009).
[14] Li, S. et al.: McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures, *MICRO-42*, pp. 469–480 (2009).
[15] Powell, M. D. et al.: Balancing Resource Utilization to Mitigate Power Density in Processor Pipelines, *MICRO-38*, pp. 294–304 (2005).
[16] Processor Simulator Onikiri 2: http://www.mtl.t.u-tokyo.ac.jp/∼onikiri2/.
[17] SimPoint: http://cseweb.ucsd.edu/ calder/simpoint/.
[18] Skadron, K. et al.: Temperature-aware microarchitecture, *ISCA-30*, pp. 2–13 (2003).
[19] Taylor, M. B.: Is Dark Silicon Useful? Harnessing the Four Horesemen of the Coming Dark Silicon Apocalypse, *DAC-49*, pp. 1131–1136 (2012).