Vol.2012-ARC-201 No.11
2012/8/1

レジスタ・ファイルと実行ユニットにおけるアクティビティ・マイグレーション

井上 聖等[1,a)]　三輪 忍[1]　中田 尚[1]　中村 宏[1]

概要：近年のチップは，ホット・スポットから発せられる熱によって，安全に動作できる周波数が制限されてしまっている．よって，ホット・スポットの温度を下げることができれば，チップの動作周波数を向上させることができ，プロセッサ性能は向上する．アクティビティ・マイグレーションは，あるモジュールで行われる処理をそれと同等の機能を有する別のモジュールへと移すことで，性能を維持しつつモジュールの温度上昇を抑える技術である．本稿では，ホットなモジュールの 1 つであるレジスタ・ファイルと実行ユニットに関して，アクティビティ・マイグレーションの適用を検討する．

キーワード：熱，アクティビティ・マイグレーション，レジスタ・ファイル，実行ユニット

1. はじめに

近年のプロセッサでは，チップから発生する熱が性能を制限する大きな要因となっている．冷却装置に求められる冷却能力を表す **TDP（Thermal Design Power）**は，この 20 年間で急激に上昇した．その結果，TDP は実用的な冷却能力の限界に達し，頭打ちとなっている．市場に出回っているチップの動作周波数は，過剰に高温な状態となることがないよう，クリティカル・パスの遅延によって定められる理論的な動作限界よりも低めに設定されているのが普通である．実際，特殊な冷却装置を用いれば，近年のプロセッサは 8.4GHz で動作させることも可能である [10]．このように，現在の冷却装置の性能は，LSI の微細化によってもたらされるプロセッサの性能向上に追いついていないのが現状である．

チップには**ホット・スポット**と**コールド・スポット**が存在することが知られている [1], [7], [11], [12]．前者の例は演算器，後者の例はキャッシュである．両者の温度差は，場合によっては 30 度にも達する [1]．回路は，一部分でも温度が正常に動作する限界値を上回ると，全体として正常に動作することを保証できなくなる．そのため，チップが正常に動作する限界の温度は，ホット・スポットのそれによって決まる．なお，ホット・スポットはチップのごく一部の領域に限られることもわかっている．

ホット・スポットの温度を抑えることができれば，チップのピーク温度と動作限界温度との間に余裕が生まれる．その分，チップの動作周波数を向上させることができ，その結果，プロセッサ性能は向上すると考えられる．

アクティビティ・マイグレーションは，モジュールで行われる処理をそれと同等の機能を有する別のモジュールへと移すことで，性能を維持しつつモジュールの温度上昇を抑える技術である [2], [3], [6]．モジュールの温度は，アクティビティが時間的に集中する，すなわち，それを使用し続けることによって上昇し，使用しなければ低下する．そこで，モジュールがある程度の温度に達したら使用を中止することにより，それ以上の温度上昇を抑制する．中止している間は，別に用意しておいたモジュール（複製）を利用して処理を継続する．これを繰り返すことで，性能を落とすことなくモジュールの温度上昇を抑制できる．

後述するように，アクティビティ・マイグレーションにはいくつかのデメリットが存在するが，それらのデメリット，および，温度上昇の抑制効果は，マイグレーションを行う時間粒度と複製の数とに依存する．例えば，使用するモジュールをサイクルごとに切り替えれば温度上昇は抑えられるが，切り替え回数が増えるほど，切り替えの際のオーバヘッドは無視できなくなる．また，マイグレーション先の候補が多いほど，個々のモジュールを長期間休ませることができる反面，面積オーバヘッドは増大する．モジュールごとに最適な時間粒度と複製数が存在すると考えられるが，それがどの程度であるかはわかっていない．

本稿では，ホットなモジュールの 1 つである [1], [7]，レジスタ・ファイルと実行ユニットに着目する．チップのピー

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

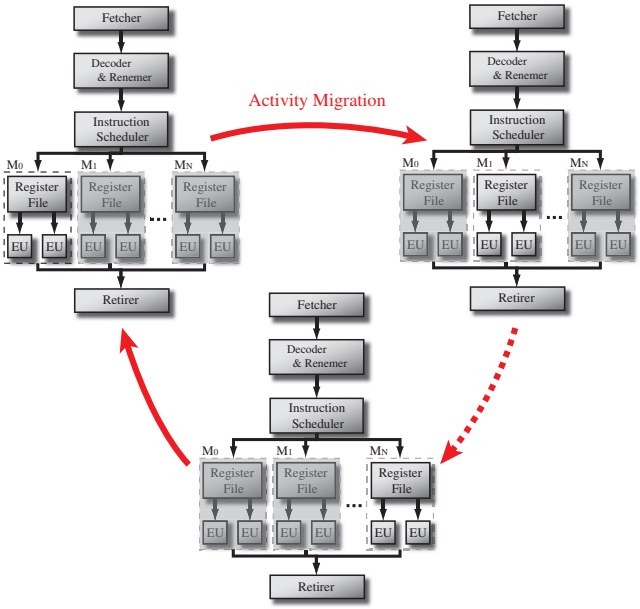

図 1 アクティビティ・マイグレーション

ク温度を下げるということは，これらのモジュールのピーク温度を下げることに相当する．今回，これらのモジュールを対象に，アクティビティ・マイグレーションの時間粒度と複製数に関する検討を行った．

本稿の構成は以下のようになっている．まず次章でアクティビティ・マイグレーションについて詳しく述べる．続く 3 章では，レジスタ・ファイルと実行ユニットのアクティビティ・マイグレーションの実現方法を検討する．4 章で評価を行い，5 章で発熱の問題を緩和するその他の研究に触れ，最後 6 章でまとめと今後の課題について述べる．

2. アクティビティ・マイグレーション

本章ではまずアクティビティ・マイグレーションについて説明し，次いで，マイグレーションの時間粒度と複製の数が温度に与える影響について述べる．続く 2.3 節においてアクティビティ・マイグレーションの課題を述べた後，最後 2.4 節にて既存研究をまとめる．

2.1 概要

アクティビティ・マイグレーションは，モジュールのアクティビティをそれと同等の機能を有する別のモジュールへと移すことで，性能を維持しつつモジュールの温度上昇を抑える技術である [2], [3], [6]．モジュールの温度は，それを使用し続けることによって上昇し，使用しなければ周辺温度に達するまで徐々に低下する．そこで，アクティビティ・マイグレーションでは，モジュールを一定期間使用したら使用を中止し，使われていなかった別のモジュールを使用して処理を継続する．これにより，使用を中止したモジュールの温度を下げつつ，実行を継続できる．

図 1 にアクティビティ・マイグレーションを行うプロセッサのブロック図を示す．図ではレジスタ・ファイルと実行ユニットをマイグレーションの対象としている．なお，アクティビティ・マイグレーションは他のモジュール，例えば，デコーダあるいはコア全体に対して適用することも可能である．

アクティビティ・マイグレーションを行うプロセッサでは，図に示すように，マイグレーション対象のモジュールが複数個設けられる．ただし，それらは常に 1 つしか使用されず，複数が同時に使用されることはない．図では網掛けによって使用していない状態を表している．消費電力を抑えるため，未使用モジュールの電源は遮断されることもある．図では，元々のレジスタ・ファイルと実行ユニット（M_0）に加え，マイグレーション先として N 個の複製（$M_1 \sim M_N$）を用意してある．

プロセッサは，モジュールのうちの 1 つ（ここでは M_0 とする）を用いてプログラムの実行を進める（図の左上の状態）．この処理は，複製を持たない，普通のプロセッサと同様である．モジュールを使用し続けることによって，その温度は徐々に上昇する．

モジュールの使用開始から一定期間が経過すると，マイグレーションが行われる．マイグレーションは，具体的には以下のようにして行う．まず，マイグレーション対象のモジュールの上流に位置するパイプラインを止める．下流のパイプラインについては，そのモジュールで実行中のすべての命令がモジュールでの処理を終えてから停止する．そして，使用していたモジュールの使用を中止し，未使用のモジュールの中から 1 つ（図では M_1）を何らかの方法（例えばラウンド・ロビン）で選択する．

モジュールを変更する際はいくつかのプロセッサ状態のコピーが行われることもある．例えば，レジスタ・ファイルのマイグレーションの場合は，プログラムの実行を正しく継続するためには，それまで使用していたレジスタの値すべてを新しいレジスタ・ファイルへコピーしなければならない．TLB のマイグレーションの場合は，マイグレーションにより TLB 内の情報がすべて失われてしまうと，マイグレーション後に TLB ミスが多発し，性能が著しく低下してしまう．この性能低下を抑えるためには，TLB の内容をすべてコピーする必要がある．このように，マイグレーションの際には，状態をコピーするためのオーバヘッドが存在する．

状態のコピーが完了すると，選択したモジュールを用いて中断していた処理を再開する（図の右上の状態）．それまで使用していたモジュールは，使用を止めたことによって，温度が徐々に低下することになる．

マイグレーションを何度か繰り返し，モジュール M_0 がマイグレーション先として選択されたとしよう．M_0 の温度が定常温度にまで下がっていれば，M_0 の最高温度は，それまでの M_0 のアクティビティの量によって抑えられ

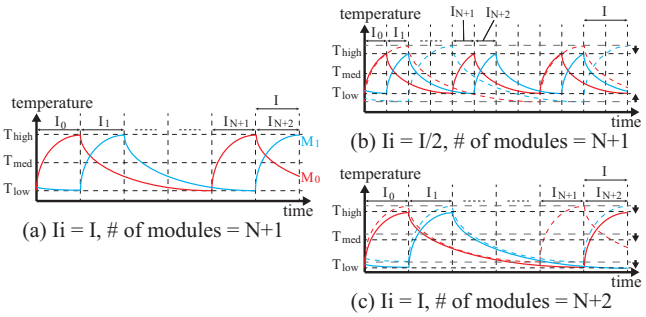

図 2 間隔と複製数がモジュールの温度に与える影響

る．すなわち，モジュールの温度をある一定の範囲に抑えることができる．これについては次節で詳しく述べる．

このようにしてアクティビティ・マイグレーションは，プログラムの実行を大きく妨げることなく，モジュールの温度上昇を抑制する．

2.2 マイグレーション間隔と複製数が温度に与える影響

アクティビティ・マイグレーションは，マイグレーションの間隔を短くするほど，また，複製の数を増やすほど，モジュールの温度上昇を抑えることができる．**図 2** を用いてこれを説明する．

図 2 (a) は，$N+1$ 個のモジュールを I サイクルごとに使用する場合の温度変化を表している．横軸は時刻，縦軸は温度であり，図には 2 つのモジュール M_0, M_1 の温度をそれぞれ赤色，青色の線で表してある．モジュール M_0 は区間 $I_0, I_{N+1}, \cdots$ で使用され， M_1 は区間 $I_1, I_{N+2}, \cdots$ で使用されるものとする．

アクティビティ・マイグレーションによって各モジュールの温度は一定の範囲内に抑えられる．最初は M_0 の温度は定常状態（T_{low}）とする．区間 I_0 で，モジュール M_0 が使用されることによって，M_0 の温度は徐々に上昇し，やがてピーク温度（T_{high}）に達する．そして，次の区間（I_1）に移ると，M_0 の使用は中止され，M_1 の使用が開始される．それにより，M_1 の温度は徐々に上昇し，M_0 の温度は徐々に下降する．マイグレーションを繰り返し，再び M_0 を使用する順番となる頃には，十分休んだことにより，M_0 の温度は最低温度（T_{low}）にまで低下している．M_0 を再び使用すると，M_0 の温度は再び上昇し始め，前回とほぼ同じ温度（T_{high}）に達する（区間 I_{N+1}）．

マイグレーション間隔を 1/2 にした場合の温度変化を図 2 (b) に示す．区間 I_0 に示すように，モジュール M_0 の使用期間が半分になったことで，温度の上昇幅は低減し，ピーク温度（T_{high}）は低下する．一方，未使用の期間（$I_1 \sim I_N$）も半分になるため，温度の下降幅も減少し，最低温度（T_{low}）は上昇する．間隔をさらに短くしていくと，全モジュールの温度の均一化が進み，ピーク温度と最低温度はその中間温度（T_{med}）に近づいていく．そのため，振動の中心となる温度（T_{med}）は (a) の場合と変わらない．

このように，間隔を短くすることは，温度変化の振幅を小さくする効果がある．

図 2 (c) は (a) からモジュールを 1 つ増やした場合の温度変化を表したものである．各区間の長さは変わらないため，各モジュールの温度上昇幅は変わらない（区間 I_0 を参照）．一方，未使用の期間は (a) の場合よりも長くなるため，より周囲の温度に近づく．その結果，モジュールの温度は (a) よりも低い位置で振動することになる．

このように，マイグレーションの間隔，複製の数ともにピーク温度を抑える効果があるが，最適な間隔，複製の数はわかっていない．間隔が長い時は，複製が少なくても十分な冷却期間を設けることができ，複製を増やす効果は低い．間隔が短くなるほど，十分な冷却効果を得るにはより多くの複製が必要と考えられるが，どの程度の数が必要かは不明である．4 章ではこれを明らかにする．

2.3 課題

2.3.1 マイグレーションの距離

実際のチップ上では，各モジュールの温度は，隣接するモジュールの影響を受ける．高温なモジュールに隣接するモジュールは，自身のアクティビティが低くても，境界の温度が上昇してしまう．例として，図 1 の M_0 と M_1 が隣接してレイアウトされており，M_0 から M_1 へマイグレーションする場合を考える．その場合，M_1 の使用を開始すると，M_1 の温度が徐々に上昇し，その結果，隣接する，使用を中止した M_0 の温度が下がりにくくなってしまう．

したがって，マイグレーション元のモジュールとマイグレーション先のモジュールは，なるべく離れた位置にレイアウトした方がよい．しかし，両者の距離が離れると，一般に，それらと接続する他のユニット（図では Instruction Scheduler や Retirer）との距離も離れることになる．その結果，両者を結ぶ配線が長くなり，アクセス・レイテンシが悪化してしまう．

このように，複製をどのように配置し，それらをどのような順序で使用するかは，考慮すべき点がある．これについては次章以降で詳しく述べる．

2.3.2 性能ペナルティ

アクティビティ・マイグレーションは，以下の 2 つの要因によって，性能が低下する可能性がある．

(1) マイグレーション時のパイプライン・ストール
(2) 複製にアクセスする場合の遅延

2.1 節で述べたように，マイグレーションを行う際はパイプラインを一旦停止させる必要がある．モジュールにもよるが，単純にパイプラインを止めて再開するだけでも，最低 1 サイクルはかかる．FPU などレイテンシの大きなユニットの場合は，より大きなペナルティを被ることになる．

また，前述のように，ストール時には状態のコピーも行われる．例として，レジスタ・ファイルのすべてのエント

リの内容をコピーする場合を考える．レジスタ・ファイルのエントリ数を 128，ライト・ポートの数を 4，リード/ライト・レイテンシをそれぞれ 1 サイクルとすると，コピーには 33 サイクルを要する計算となる．TLB や BTB など，多くのエントリからなる表のコピーにはそれ以上の時間が費やされる．

上述の性能ペナルティはマイグレーションの度に発生する．したがって，マイグレーションの間隔が短くなるほど，より多くのペナルティを被ることになる．前述のように，ピーク温度を抑制するためには間隔は短い方がよいが，この性能ペナルティとはトレードオフの関係にある．

なお，モジュールの電源遮断と復帰を行う場合であっても，それらの処理に要する時間がストール時間に悪影響を与えることはない．前者の処理の遅延はパイプラインの再稼働と並列に行うことで隠蔽できる．また，後者の処理はパイプライン停止に間に合うように前もって行えばよい．

その他の性能ペナルティとして，複製にアクセスする場合の遅延が挙げられる．前節で述べたように，複製の配置によっては，複製とそのアクセス元のモジュールとの距離が離れ，配線長が延びてしまう．このようなレイアウトで複製を使用すると，モジュールにアクセスするたびに上述の遅延の影響を受けることになる．最悪，パイプライン段数が増加する可能性がある．

本稿の冒頭で述べたように，アクティビティ・マイグレーションは，チップのピーク温度を抑制し，それによる動作周波数の向上を目指したものである．したがって，アクティビティ・マイグレーションによってプロセッサ性能が向上するか否かは，これらの性能ペナルティと，動作周波数の向上による性能向上との大小関係による．

2.3.3 トランジスタ数の増加

アクティビティ・マイグレーションにはモジュールの複製を必要とする．したがって，複製の数を N 個とすると，モジュールの面積は N 倍に増加してしまう．

このように，アクティビティ・マイグレーションはチップの面積を増大させるが，この問題は LSI の微細化が進むにつれて緩和されると考えられる．なぜなら，微細化によって，チップ上には，電力バジェットのために使用できないトランジスタ（Dark Silicon）が多数生まれると言われているからである [4]．

アクティビティ・マイグレーションにおいて使用しないモジュールの電源を遮断してしまえば，それらの消費電力はゼロである．よって，これらの余剰トランジスタを用いて複製を作ればよい．

2.3.4 リーク電力の増加

モジュール数が N 倍になるということは，それらのリーク電力も N 倍になることを意味する．演算器などのタイミング・クリティカルなモジュールでは，一般に，閾値電圧が低く，リーク電力が大きいトランジスタが利用される．我々の評価によると，45nm で製造された 4 個の ALU のリーク電力は，100°C の時には 1.7W にも達する．複製を 10 個用意したとすると，ALU のリーク電力だけで 17 W も消費してしまう．

この問題は，前述のように，未使用のモジュールの電源を遮断することによって解決できる．複製がいくつあろうが，使用するモジュールは常に 1 つである．使用順が回ってきた時に始めて，そのモジュールの電源を投入すればよい．電源の ON/OFF には余分な電力を消費することになるが，その影響は今後評価する．

2.3.5 ダイナミック電力の増加

状態をコピーするためには，コピー元，および，コピー先のモジュールに対して追加のアクセスを必要とする．128 エントリのレジスタ・ファイルをコピーする場合は，コピー元に対して 128 回のリード・アクセスが，コピー先に対して 128 回のライト・アクセスが追加で必要となる．

上記の消費電力の増加は，コピーの発生頻度が少ない，すなわち，マイグレーション間隔が十分長ければ問題ない．しかし，前述のように，モジュールの温度上昇を抑えるためには，マイグレーション間隔はなるべく短い方がよく，その結果，この消費電力の増加が無視できなくなる可能性がある．これについても今後詳しく評価する．

2.4 従来研究

Heo らは，Out-of-Order スーパスカラ・プロセッサのさまざまなモジュールに関して，アクティビティ・マイグレーションの効果を評価している [6]．彼らは，マイグレーションの対象を 1) コアそのもの， 2) データ・キャッシュと D-TLB 以外のすべてのモジュール， 3) 命令キュー，マップ表，実行ユニット，レジスタ・ファイルの 4 つ，4) 実行ユニットとレジスタ・ファイルのみ，としている．そして，面積，温度，性能等を評価した結果，4) の構成が最もよいと結論づけている．また，Heo らはマイグレーションの時間粒度についても評価しており，100K サイクルごとにマイグレーションするとよい，と述べている．ただし，彼らの研究では複製の数を 2 つに限定しており，複製の数を増やした場合の効果については評価していない．また，レイアウトについてもほとんど考慮していない．

Chaparro らは，クラスタ化スーパスカラ・プロセッサのクラスタをローテーションで使用することによってクラスタの温度上昇を抑制する，クラスタ・ホッピングという手法を提案している [2], [3]．クラスタには，レジスタ・ファイルや実行ユニットだけでなく，命令キューやデータ・キャッシュも含まれる．彼らの方法では，マイグレーション時にレジスタおよび命令キューのエントリはコピーするが，データ・キャッシュはコピーしない．そのため，マイグレーション後にはデータ・キャッシュ・ミスが頻発し，性能が著しく低下してしまうという問題がある．

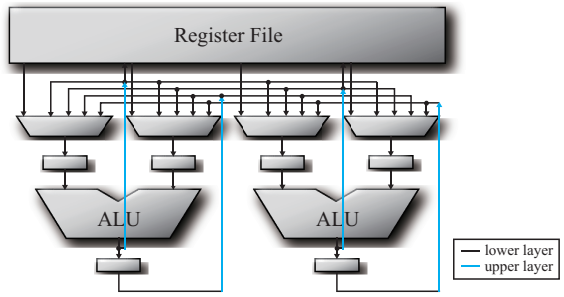

図 3 レジスタ・ファイルと実行ユニット周辺の回路構成

3. レジスタ・ファイルと実行ユニットにおけるアクティビティ・マイグレーション

本章では，まず，実行ユニット周辺の回路構成を示し，レジスタ・ファイル，実行ユニット，バイパス・ネットワークは分離するのが難しく，これらを 1 つの単位としてマイグレーションした方がよいことを述べる．次いで，実際のフロアプランを例に，上述のユニットの複製をどのように配置したらよいかを検討する．最後に，マイグレーションの効果を左右する，ストール・サイクル数の見積りを行う．

3.1 マイグレーションの空間粒度

ALU 周辺の回路構成を**図 3** に示す．図は，整数系の命令を 2 命令同時に発行可能なプロセッサを想定しており，レジスタ・ファイルのレイテンシは 1 サイクルを仮定している．各モジュールの配置は物理構成を意識してある．

図に示すように，複数個の ALU は隣り合って並んでおり，レジスタ・ファイルはその上部に配置される．レジスタ・ファイルの各リード・ポートは，セレクタを介して各 ALU の各ソース・ラッチへと接続されている．ALU の出力は，上層の配線層を通り，レジスタ・ファイルのライト・ポートへと接続される．ALU の出力の他に，それをラッチした結果がレジスタ・ファイル側へと戻される．これらの信号線はセレクタの入力へと接続されており，これによりバイパス・ネットワークを形成する．

我々の評価によれば ALU はレジスタ・ファイルより 6°C も温度が高い．そのため，ホット・スポットの温度を下げるという観点からは，ALU のみマイグレーションを行えば十分ではある．しかし，以下で述べるように，ALU のみをマイグレーションの対象とすることはほとんど不可能と言ってよい．

複製された ALU を用いて依存関係にある命令を back-to-back に実行するためには，図 3 の ALU と同様，ALU の出力を入力へと戻すパス，すなわち，バイパス・ネットワークが必要である．仮に，複製の ALU にバイパス・ネットワークを設けなかったとすると，複製を使用してプログラムを実行している間は，すべてのソース・オペランドは，レジスタ・ファイルへの書き込みが完了するのを待って読み出さなければならない．back-to-back な実行の放棄は，プロセッサ性能を著しく悪化させる．

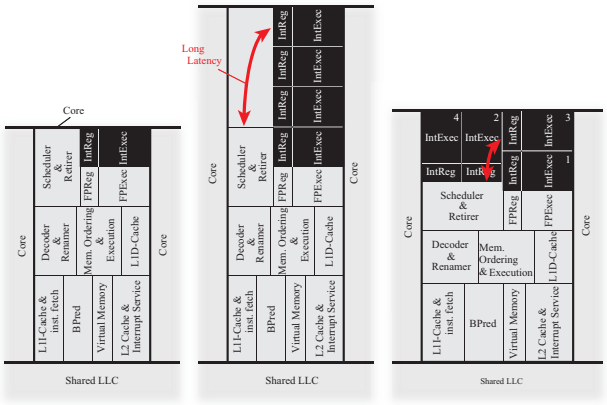

図 4 複製を配置する際のフロアプラン

また，ALU とバイパスのみを複製するという選択肢も現実的でない．ALU とバイパスのみを複製し，レイアウトを行った結果，複製した ALU とレジスタ・ファイルとの距離が離れ，その間の配線長が長くなったとしよう． レジスタ・ファイルと ALU 間の配線遅延の増加は，レジスタ・ファイルのリード・レイテンシを悪化させる．それにより，パイプラインがさらに深化する可能性がある．レジスタ・ファイルのパイプライン化は，分岐予測ミス・ペナルティが増加するだけでなく，バイパスが複雑化するという問題も引き起こす [13].

したがって，レジスタ・ファイルも含め，レジスタ・ファイル，実行ユニット，バイパス・ネットワークをまとめてマイグレーションの対象とするのが現実的である．

3.2 複製の配置とその使用順序

2.3 節で述べたように，複製の配置の仕方によっては，パイプラインが深化する可能性がある．Scheduler とレジスタ・ファイルの距離が離れてしまい，それらを結ぶ配線が長くなると，実効レイテンシが増加する．また，レジスタ・ファイルと Retirer の距離が離れてしまうと，コミット・レイテンシが増加する．増加したレイテンシが 1 サイクル以内に収まっていればよいが，そうでない場合は，ラッチを新たに挿入し，ステージに分割しなければならない．

図 4 (a) は，Nehalem アーキテクチャのフロアプランの一部である [5]．チップの下部に共有のラスト・レベル・キャッシュが位置し，各コアはその上部に配置される．コアの内部では，モジュール間の配線長を考慮して，依存関係にあるモジュール同士が近い位置に配置される．例えば，Fetcher と Decoder は隣り合って配置される．本稿でマイグレーションの対象とするレジスタ・ファイルと実行ユニットは右上に位置しており，Scheduler や Retirer と

隣り合っている．

マイグレーションのためにレジスタ・ファイルと実行ユニットを 3 つ追加する場合を考える．図 4 (b) は，図 4 (a) のレイアウトは変えずに，単純に複製を追加した場合を表している．3 つの複製は，元々レジスタ・ファイルと実行ユニットがあった場所の上部に一直線に配置される．このようなレイアウトを行なってしまうと，Scheduler と最上部に位置する複製との配線長が長くなってしまい，所定のサイクル内にアクセスできなくなる可能性がある．

図 4 (c) は，Scheduler と Retierer を取り囲むように複製を配置した場合のレイアウトである．複製の配置にともない，他のモジュールの配置にも変更が及んでいる．明らかに，こちらのレイアウトの方が各レジスタ・ファイルと Scheduler との距離が近い．このように，パイプライン段数の増加を防ぐためには，複製も含めて，モジュール間の配線長を考慮してレイアウトし直した方がよい．

なお，2.3.1 項で述べたように，マイグレーションによるモジュールの冷却効果を十分に得るためには，マイグレーション元のモジュールとマイグレーション先のモジュールとの距離は遠いほうがよい．そのようなモジュールの使用順序は，図 4 (c) のフロアプランで言えば，例えば，各 ALU の右肩の数字の順となる．

3.3 ストール・サイクル数の見積り

2.3.2 項で述べたように，マイグレーション時に発生するストールはプロセッサ性能を低下させる．本節では，レジスタ・ファイルと実行ユニットのマイグレーションにおいて何サイクル程度のストールが発生するかを見積もる．

ストール・サイクル数は 2 つにわけて考えることができる．1 つはパイプラインの停止によって消費されるもの，もう 1 つは状態のコピーによって消費されるものである．

前者のサイクル数は以下のようにして計算できる．レジスタ・ファイルと実行ユニットのマイグレーションは，まずは命令発行を停止することから始まる．発行を停止した時点では，発行済みだがまだレジスタ・リードが行われていない命令がパイプライン中に存在する．これらの命令が実行され，レジスタ書き込みが完了するまでは，使用するモジュールを変更できない．すなわち，レジスタ読み出し，実行，書き込みのサイクル数分のストールが必要となる．それぞれ 1 サイクルとすると，計 3 サイクルとなる．

後者のサイクル数はいくつの状態をどのような手段でコピーするかに依存する．レジスタ・ファイルと実行ユニットのマイグレーションにおいては，最低限，使用中のレジスタの値を新しいレジスタ・ファイルにコピーしておかなければならない．レジスタ・ファイルに元々備わっているポートを用いてコピーを行うとすると，4 命令発行のプロセッサにおいては，1 サイクルにつき最大 4 つのレジスタ値を読み出す/書き込むことができる．未使用のものも含め，単純にすべての値をコピーしたとすると，128 エントリのレジスタ・ファイルのコピーには 33 サイクルかかる．

以上まとめると，マイグレーションごとに数十サイクル（上記の例では 36 サイクル）のストールが発生すると考えられるが，このペナルティが許容できるか否かは，マイグレーションの発生頻度に依存する．10K サイクルに 1 回マイグレーションが行われる程度ならば，ストール率は 0.36% に過ぎず，十分許容可能といえる．一方，1K サイクルに 1 回マイグレーションする場合は，ストール率は 3.6% にも達し，許容しがたいものがある．動作周波数が向上できたとしても，ペイできない可能性が高い．このような場合は，専用線を設けるなど，コピーに要する時間を減らす何らかの工夫が必要だろう．

このように，マイグレーションの間隔によって取り得る実装は変わってくる．次章の評価では，どの程度の間隔でマイグレーションを行えばよいかを明らかにする．

4. 評価

シミュレータを用いてレジスタ・ファイルと実行ユニットのアクティビティ・マイグレーションの初期評価を行った．本章ではその結果を述べる．

4.1 評価方法

複製数とマイグレーション間隔によって，アクティビティ・マイグレーションの効果がどのように変化するかを測定した．複製の数は 0 ～ 5 まで変化させた．マイグレーション間隔は $1K \sim 10M$ サイクルまで変化させて測定した．

評価に用いたプロセッサの基本構成を**表 1** に示す．プロセッサは 4 つのコアを持つが，そのうちの 1 つのみが稼働しており，他の 3 つのコアはスリープしている（消費電力がゼロの）状況を想定する．このプロセッサの各コアにレジスタ・ファイルと実行ユニットを追加し，マイグレーションによってピーク温度がどのぐらい低下するか，また，それによって動作周波数がどこまで向上するかを評価する．

表 1 ベース・プロセッサの構成

Parameters	Remarks
# of Cores	4
Frequency	3.5GHz
Way	4
Depth	7 stages
Exec Units	ALU:4, Mult:2, FPU:1, Mem:2
Instruction Queue	INT:32, FP:16
Load/Store Queue	32 (in each)
ROB	80 entries
Register File	INT:128, FP:128
L1I-Cache	64KB, 16B/line, 2 way, 2 cycles
L1D-Cache	64KB, 16B/line, 2 way, 2 cycles
LLC (per core)	2MB, 16B/line, 8 way, 32 cycles

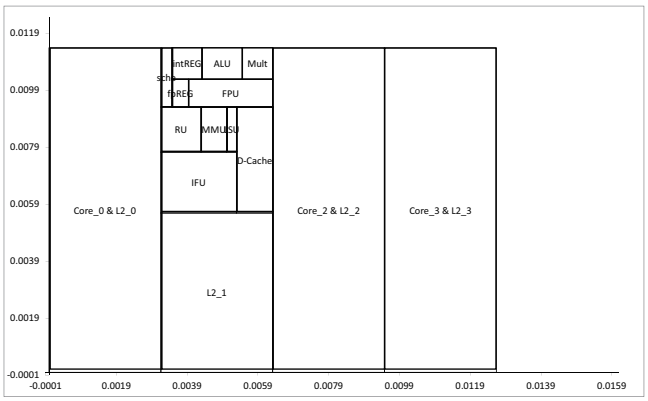

図 5 ベース・プロセッサのフロアプラン

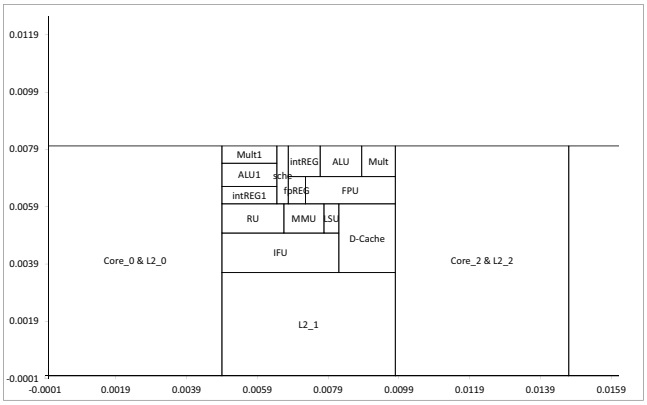

図 6 複製 1 個の時のフロアプラン

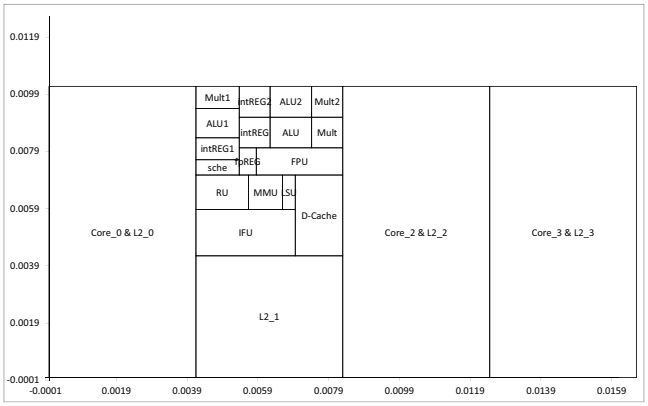

図 7 複製 2 個の時のフロアプラン

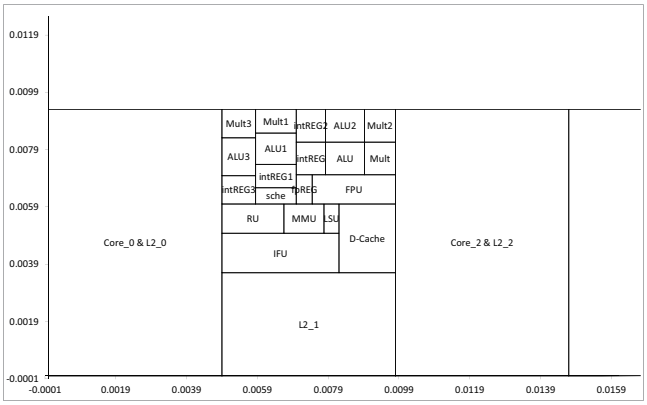

図 8 複製 3 個の時のフロアプラン

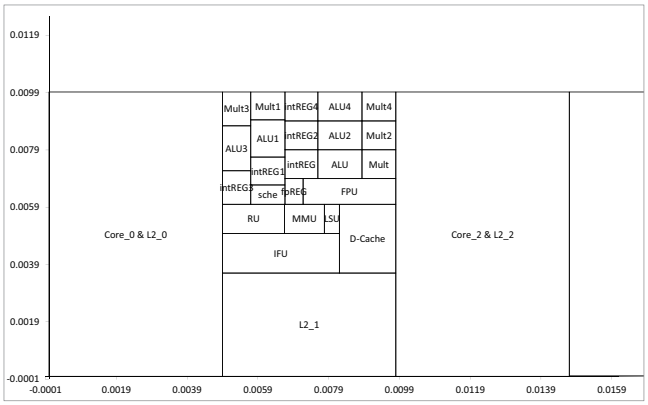

図 9 複製 4 個の時のフロアプラン

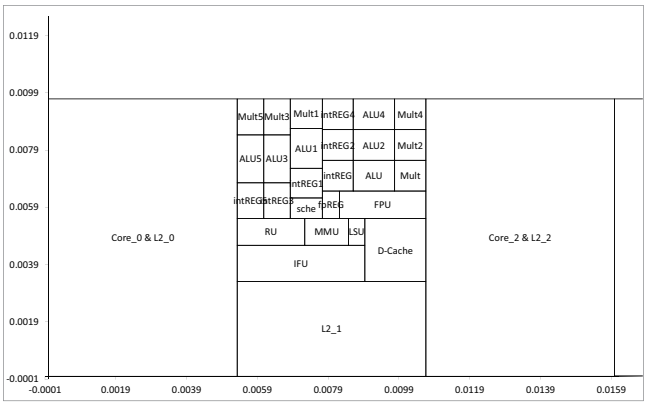

図 10 複製 5 個の時のフロアプラン

複製数が 0 ～ 5 個の時のフロアプランを**図 5**～**図 10** に示す．これらのフロアプランは，表 1 のパラメタを用いて McPAT[9] によるシミュレーションを行い，それによって得られた各モジュールの面積を用いて，Nehalem のフロアプランを参考にして独自に作成した．複製を含むフロアプランについては，3.2 節で述べたように，Scheduler を囲むように複製を配置することを意識し，各モジュールの縦横比と位置を調整した．

これらのフロアプラン，および，McPAT による電力シミュレーションの結果を用いて，HotSpot[7] による温度シミュレーションを行う．

簡単のため，IntREG，ALU 以外のモジュールは時間によって使用状況が変わらないものとする．モジュールの平均稼働率を $x\%$ とする時，モジュール m の消費電力 P_m は以下の式によって計算できる．

$$P_m = P_m^{Leak} + P_m^{PeakDynamic} \times x/100 \qquad (1)$$

ただし，P_m^{Leak}，$P_m^{PeakDynamic}$ は，それぞれ，モジュール m のリーク電力，および，ピーク時のダイナミック電力である．このようにして計算した各モジュールの電力の時系列を HotSpot の入力とした．

その他，シミュレーションに用いたパラメタを**表 2** に

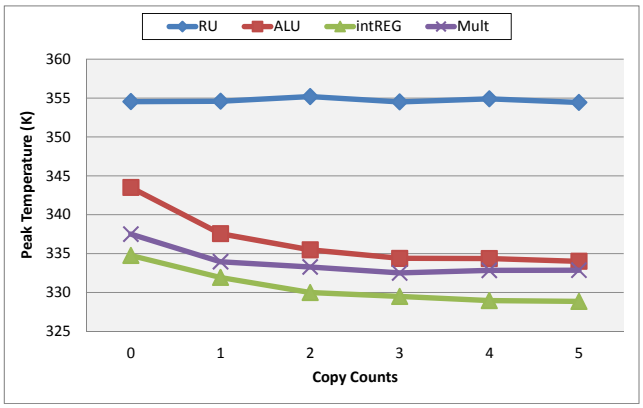

図 11 複製数を変えた時の温度変化

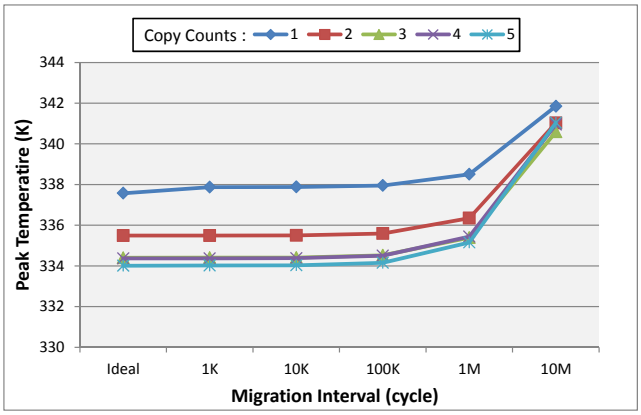

図 12 マイグレーション間隔を変えた時の ALU の温度変化

まとめる．プロセスは 45nm を想定する．チップの厚みは，HotSpot の論文に記載されていた値（130nm の時で $0.5mm$）をもとに，単純にスケーリングして求めた．周辺温度は 40°C とした．なお，次節で述べる評価はすべて，簡単のため，マイグレーションによるパイプライン・ストールは発生しないものと仮定している．

4.2 評価結果

4.2.1 複製数が及ぼす影響

図 11 に，複製の数を変えた時の各モジュールのピーク温度を示す．横軸は複製の数，縦軸は温度である．4 つの折れ線は，それぞれ，Renaming Unit （RU），ALU，IntREG，Mult の温度を表す．各モジュールの稼働率は 70%，マイグレーション間隔は理想的に短いものと仮定する．すなわち，ここでは複製を持つモジュールの消費電力は以下のようにする．

$$P_m = P_m^{Leak} + P_m^{PeakDynamic} \times x/100 \times 1/N \qquad (2)$$

ただし N は複製数とする．

図より，マイグレーション対象のモジュール（ALU，IntReg，Mult）は，複製数を増やすにしたがい，ピーク温度が徐々に低下する．特に ALU は，マイグレーションをまったく行わない場合（複製数 0）と比べて，複製 1 つを使ってマイグレーションすることで，ピーク温度が 5.7°C も低下する．複製数を増やしていくとピーク温度はさらに低下し，温度低下は，大体どのモジュールも複製 3 個で頭打ちとなる．その時の ALU の温度は 61.3°C と，マイグレーションを行わない場合の温度（70.4°C）と比べて，9.1°C も低下する．

なお，図に示すように，今回の評価では，ALU ではなく RU が最もホットなモジュールであった．RU 内では，その電力のほとんどがマップ表によって消費されていた．マップ表は，テーブルのエントリ数こそ少ないものの，毎サイクルアクセスされる，高負荷なユニットである．そのため，ホット・スポットの 1 つであっても不思議ではない．動作周波数を向上させるためには，マップ表においてもアクティビティ・マイグレーションを行う必要があろう．ALU よりも高温となった原因については今後調査する．

表 2 シミュレーションに用いたその他のパラメタ

Parameters	Remarks
Process	45nm
Ambient Temperature	313.15K
Chip Thickness	0.2mm
Silicon Thermal Conductivity	100W/(m·K)
Heat Sink Side	60mm
Heat Sink Thickness	6.9mm
Heat Sink Thermal Conductivity	400W/(m·K)
Heat Spreader Side	36.5mm
Heat Spreader Thickness	1.87mm
Heat Spreader Thermal Conductivity	400W/(m·K)

4.2.2 マイグレーション間隔が及ぼす影響

マイグレーションを行う間隔が温度低下にどの程度影響をおよぼすのかを**図 12** に示す．図中の縦軸は温度 (K)，横軸はマイグレーション間隔 (サイクル)，凡例は複製数を示している．また，横軸の Ideal とは前節の図 11 における ALU の温度である．今回は電源遮断を行わない，すなわち利用していない複製モジュールがリーク電力を消費しているものとする．マイグレーション間隔 $1K \sim 10M$ サイクルの間で 10 倍刻みで，ALU のピーク温度を観測した．

図より，マイグレーション間隔は $100K$ サイクルで飽和しており，いずれの複製数についてもほぼ理想値に近い値となっている．また，複製数によるマイグレーション間隔の影響の違いは見られない．このことより，レジスタと実行ユニットのマイグレーションに伴うストールは十分に無視できるといえる．

4.2.3 周波数ゲイン

マイグレーションによって動作周波数をどの程度向上できるのかを**図 13** に示す．図中の縦軸は周波数 (GHz) を，横軸はモジュールの複製数を，各折れ線は想定する閾値温度を表している．3 ～ 5.5GHz の範囲で周波数を変化させて，その時のピーク温度から閾値温度ごとの動作限界を求めた．マイグレーション間隔は理想とした．ただし，ここ

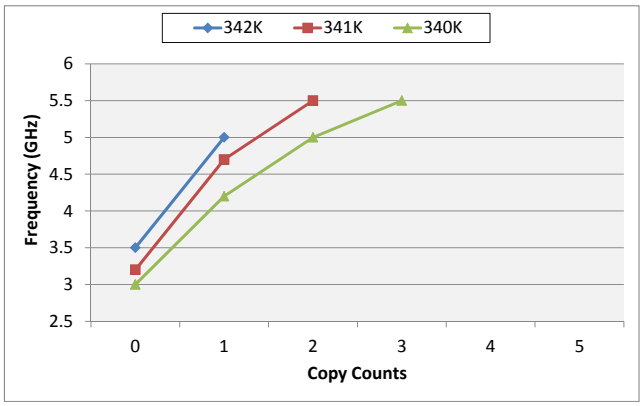

図 13 周波数ゲイン

ではレジスタ・ファイルと実行ユニットのマイグレーションの効果を調べるため，最も高温であった RU については何らかの方法（例えば RU のマイグレーション）でコールド・スポットにできた（稼働率 0% になった）と仮定して評価を行っている．

図より，測定した範囲内では，複製数を増やすにつれて周波数が大きく向上している．特に，閾値温度を $340K$ と想定した場合，マイグレーションを行わない場合は 3.0GHz が動作限界だったのに対し，複製 3 つを用いてマイグレーションすることによって動作限界を 5.5GHz にまで引き上げることができる．向上率に直すと 183% にもなる．

5. 発熱に関するその他の研究

商用プロセッサには，不意の温度上昇によるビット反転，あるいは，回路そのものの物理的な破壊を防ぐ目的で，温度を動的に管理する機構が組み込まれている．チップ内のいくつかの場所に埋め込まれた温度センサによって，チップ内の温度を監視する．そして，温度がある一定の閾値（70 ～ 100 °C）を上回ると，温度を下げるためのメカニズムが動作する．

Intel 社の Core i7 の場合は，DVFS，クロック・モジュレーション，強制シャットダウンの 3 段階の方法によって，温度を調整する [8]．

チップ内の温度が閾値を超えると，まず DVFS による温度調整が行われる．動作電圧，ならびに動作周波数を最低にする．ダイナミック電力は動作周波数に比例し，動作電圧の 2 乗に比例する．そのため，電圧と周波数を低下させることによって，消費電力を大幅に削減できる．

DVFS を行なってもチップの温度が閾値を下回らない場合は，さらにクロック・モジュレーションが行われる．クロック・モジュレーションはクロックの duty cycle を減らす，すなわち，生成されたクロックのうち，実際に回路に供給するクロックの割合を減らす技術である．生成されたクロックの x% を供給する場合，残りの $(100 - x)$% の間は回路の状態が変化しない．よって，ダイナミックな消費電力を x% に抑えることができる．

クロック・モジュレーションを行った後でも，チップの温度が閾値を超えた状況が 20 msec 続いた場合は，強制的にシャットダウンされる．

このように，現在のプロセッサで用いられている温度管理技術は，故障を防ぐことを目的としたものである．アクティビティ・マイグレーションのように，プロセッサ性能の向上を目的としたものではない．

6. まとめと今後の課題

本稿では，ホットなモジュールの 1 つであるレジスタ・ファイルと実行ユニットを対象に，アクティビティ・マイグレーションを適用する場合の初期検討を行った．一次近似的な評価の結果からは，マイグレーションによって，チップの動作周波数を大幅に向上できる可能性があることがわかった．今後は，詳細なシミュレーションを行い，マイグレーションの効果を厳密に検証したいと考えている．

謝辞 本研究の一部は科研費（課題番号 24700044）による．

参考文献

[1] Black, B. et al.: Die Stacking (3D) Microarchitecture, *MICRO-39*, pp. 469–479 (2006).
[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] Hennessy, J. L. and Patterson, D. A.: *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, 5th edition (2011).
[6] Heo, S. et al.: Reducing power density through activity migration, *ISLPED*, pp. 217–222 (2003).
[7] Huang, W. et al.: Hotspot: A compact thermal modeling method for CMOS VLSI systems, *IEEE Transactions on Component Packaging and Manufacturing Technology*, Vol. 14, pp. 501–513 (2006).
[8] Intel: Intel Core i7-900 Desktop Processor Extreme Edition Series and Intel Core i7-900 Desktop Processor Series on 32-nm Process (Datasheet, Volume 1).
[9] Li, S. et al.: McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures, *MICRO-42*, pp. 469–480 (2009).
[10] PCWorld: http://www.pcworld.com/article/239858/amd_breaks_overclocking_record_leaves_the_competition_in_the_dust.html.
[11] Poirier, C. et al.: Power and Temperature Control on a 90nm Itanium-Family Processor, *ISSCC*, pp. 304–305 (2005).
[12] Skadron, K. et al.: Temperature-aware microarchitecture, *ISCA-30*, pp. 2–13 (2003).
[13] 三輪 忍ほか：小容量 RAM を用いたオペランド・バイパスの複雑さの低減手法，*ACS19*, Vol. 48, No. SIG.13, pp. 58–69 (2007).