ホモジニアスマルチコアにおけるコンパイラ制御低消費電力化手法

白子　準[†]　押山直人[†]　和田康孝[†]
鹿野裕明[††]　木村啓二[†,††]　笠原博徳[†,††]

近年，半導体集積密度に応じたスケーラブルな性能向上，それに伴い増加する消費電力の抑制といった観点から，1 つのチップ上に複数のプロセッサを集積するチップマルチプロセッサ (CMP) アーキテクチャが幅広い分野で活用されている．しかしながら CMP アーキテクチャにおいて高実効性能・低消費電力といった要求を満たすためには，実行するプログラムの適切な並列化とチップ上のリソースのきめ細かな電圧・動作周波数制御を実現するコンパイラサポートが必要不可欠である．本論文では，各プロセッサコアが等価である OSCAR タイプのホモジニアスマルチコアプロセッサにおいて，各プロセッサの電源の ON/OFF・周波数電圧制御 (FV 制御) をマルチグレイン並列化環境下でコンパイラが適切に判断し低消費電力化を行なうコンパイル手法を提案する．また本手法を OSCAR コンパイラに組み込み，SPEC95 ベンチマークのうち 101.tomcatv, 110.applu に対して提案手法が自動決定した電源及び電圧・周波数の制御を行った際の性能について述べる．提案手法により applu において 4 プロセッサ使用時に処理性能を維持したまま 60.7% の消費エネルギー削減 tomcatv において 4 プロセッサ使用時にデッドライン制約を保証したまま 45.4% の消費エネルギー削減が達成された．

Compiler Control Power Saving Scheme for Homogeneous Multiprocessor

Jun shirako,[†] Naoto Oshiyama,[†] Yasutaka Wada,[†]
Hiroaki Shikano,[††] Keiji Kimura[†,††] and Hironori Kasahara [†,††]

A chip multiprocessor architecture has attracted much attention to achieve high effective performance and to save the power consumption, with the increase of transistors integrated onto a chip. To this end, the compiler is required not only to parallelize program effectively, but also to control the volatage and clock frequency of computing resources carefully. This paper proposes a power saving compiling scheme with the multigrain parallel processing environment that controls Voltage/Frequency and power supply of each core on the multiprocessor. In the evaluation, OSCAR compiler with the proposed scheme achieves 60.7 percent energy savings for SPEC CFP95 applu using 4 processors without performance degradation, and 45.4 percent energy savings for SPEC CFP95 tomcatv using 4 processors added deadline constraint.

1. はじめに

半導体集積度向上に伴う，1 チップ上に搭載可能なトランジスタ数の増加と共に，スケーラブルな性能向上を達成できるプロセッサの開発が望まれている．このようなプロセッサアーキテクチャとして現在チップマルチプロセッサ (マルチコアアーキテクチャ) が注目されている．またそのようなプロセッサでは処理性能のみでなく，増大する消費電力をいかに抑えるかが大きな課題となっており，この問題を克服する手段としてもチップマルチプロセッサは有望視されている．

マルチプロセッサの実効性能向上のためには，プログラムからの適切なグレイン (粒度) での並列性抽出，キャッシュやローカルメモリの最適利用及び DMAC 利用を含めたプロセッサ間データ転送オーバーヘッドの最小化，そしてそれらの効果的なスケジューリングが必須である．これを実現するために従来より自動並列化コンパイラの研究が行われており[1)~3)]，これらの研究・開発によりループ並列化技術は大きな進歩を遂げた．しかしながら現在ではループ並列化手法は成熟期に至っており，今後マルチプロセッサシステム上での大幅な性能向上を達成するためにループ並列性以外の並列性を利用する並列化手法が必要とされている．マルチレベルの並列性を利用するコンパイラとしては NANOS コンパイラ[4)]，PROMIS コンパイラ[5)]，そして OSCAR コンパイラ[6)~8)] が挙げられる．特に OSCAR マルチグレイン自動並列化コンパイラでは，プログラム中の粗粒度タスク並列処理，ループレベル並列処理，近細粒度並列処理を組み合わせたマルチグレイン並列処理を世界で唯一実現している．また OSCAR コンパイラは抽出したマルチグレイン並列性に応じ，プログラムの各部分の並列性に見合った適切なプロセッサの割当てや複数のループ (すなわち粗粒度タスク) 間にまたがる広域的なキャッシュメモリ最適化も実現している．

この並列性に応じたプロセッサの割り当て機能では，プログラム中で並列性が小さい部分においては処理の低オーバーヘッド化及び低消費電力化のために使用不要と判断されたプロセッサへの電源供給の遮断や，処理終了のデッドライン制約を満たす範囲での電圧・動作周波数の低減といった制御が重要となる．このような低消費電力化手法には様々なものが提案されている．例えばキャッシュミス回数測定用カウンタや命令キューなどのハードウェアサポートにより実行時にプログラム中の各フェーズにおける負荷を判断し，不必要なリソースを停止する Adaptive Processing[9)] や，計算資源の各部分に対して実行時の負荷に応じた周波数・電圧制御 (FV

† 早稲田大学理工学部 コンピュータ・ネットワーク工学科
Department of Computer Science, Waseda University
†† アドバンストチップマルチプロセッサ研究所
Advanced Chip Multiprocessor Research Institute

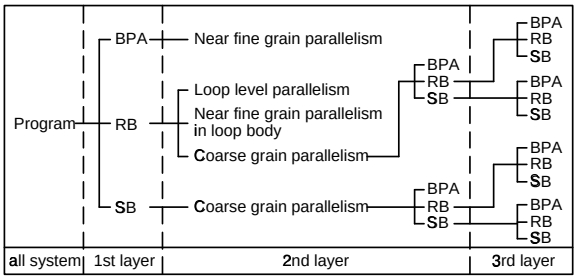

図 1 階層的マクロタスク定義

制御) を行う Online Methods for Voltage and Frequency Control[10] などがある．

本論文では OSCAR 型ホモジニアスマルチプロセッサにおいて，

- プログラム各部における不要プロセッサの停止
- プロセッサへの処理不均衡時の FV 制御
- プログラムのデッドライン制約に応じた FV 制御

をコンパイラが自動判定しプログラムに適用することでプロセッサにおける消費電力の削減を計る，コンパイラ制御低消費電力化手法を提案する．

2. マルチグレイン並列処理

本章では，提案する低消費電力制御で前提としているマルチグレイン並列処理における粗粒度タスク並列処理について述べる．粗粒度タスク並列処理とは逐次プログラムを階層的に粗粒度タスク分割し，生成された粗粒度タスクすなわちマクロタスクをプロセッサエレメント (PE)，もしくはプロセッサグループ (PG) に割り当てて実行することによりマクロタスク間の並列性を利用する並列処理手法である[11]．

2.1 粗粒度タスク生成

粗粒度タスク並列処理では，プログラムは基本ブロックまたはその融合ブロックで構成される疑似代入文ブロックである BPA，DO ループや後方分岐により生じるナチュラルループで構成される繰り返しブロック RB，サブルーチンブロック SB の 3 種類のマクロタスク MT[8] すなわち粗粒度タスクに分割される．繰り返しブロック RB やサブルーチンブロック SB に対しては，その内部をさらにマクロタスク分割し階層的なマクロタスク構造を生成する (図 1)．

2.2 粗粒度タスク並列性抽出

マクロタスク生成後，各階層においてマクロタスク間のデータ依存と制御フローを解析し，マクロタスク間のデータと制御のフローを表すマクロフローグラフ[6),8)] を生成する．

次に，階層的に生成されたマクロフローグラフに対し最早実行可能条件解析[6),8)] を適用し，階層的なマクロタスクグラフ MTG[6),8)] を生成する．ここで最早実行可能条件とは，制御依存とデータ依存を考慮したマクロタスクの最も早く実行を開始してよい条件でありマクロタスクグラフが粗粒度タスク並列性を表す．

2.3 プロセッサグループとプロセッサエレメント

階層的なマクロタスクグラフを効果的に処理するため，プロセッサのグルーピングを行う．複数のプロセッサエレメント PE をソフトウェア的にグループ化し，プロセッサグループ PG を定義する．この PG がマクロタスクを処理する単位であり，SB や RB など内部マクロタスクグラフが定義されている場合はプロセッサグループ内の PE を更にグルーピングする．

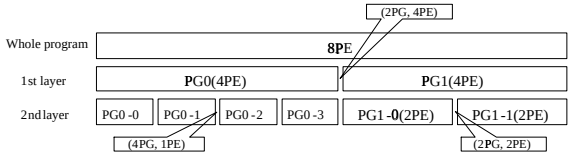

図 2 プロセッサグループ・プロセッサエレメントの階層的定義

図 2 にプロセッサグループ・プロセッサエレメントの階層的な定義の例を示す．図中，第 1 階層 (1st layer) では (2PG, 4PE) の構成をとり 2 並列で粗粒度タスク並列処理を行ない，4 PE を割り当てられた第 2 階層 (2nd layer) ではそれぞれ (4PG,1PE) で 4 並列，(2PG, 2PE) で 2 並列という粗粒度タスク並列処理を行っている．このように階層的なプロセッサ構成を定義することでそれぞれのマクロタスクグラフの並列性に適したグルーピングを行なうことが可能となる．

2.4 並列処理階層自動決定手法

抽出された各階層の粗粒度タスク並列性をプロセッサに効率よく割当てることが，マルチグレイン並列処理の性能向上において重要である．OSCAR コンパイラでは並列処理階層自動決定手法[11),12)] により，推定した各階層のマクロタスクグラフの並列度を用いて適切な PG 数・PE 数を決めることで，抽出されたマルチグレイン並列性の効果的な利用を実現している．さらに各マクロタスクの並列性の最大値を推定し，そのマクロタスクを処理するのに必要十分なプロセッサ数を推定する．これにより過剰と判断されたプロセッサにはオーバーヘッド最小化のため処理を割り当てず，不要な並列処理オーバーヘッドを削減している．

2.5 プロセッサグループへのマクロタスク割り当て

上述のように各マクロタスクグラフに合わせて生成したプロセッサグループがそのマクロタスクグラフを処理する単位となり，当該マクロタスクグラフ上のマクロタスクを各プロセッサグループに割り当てる．マクロタスクグラフ上に条件分岐が無い場合はコンパイル時に静的にスケジューリングが行われ，各プロセッサグループの処理するマクロタスクがコンパイル時に決定される．マクロタスクグラフが条件分岐等の実行時不確定性を含む場合は，実行時にタスクをプロセッサグループに割り当てる動的スケジューリングルーチンをコンパイラが自動生成し，並列プログラム中に埋め込むダイナミックスケジューリング方式がとられる．

本論文ではスタティックスケジューリングを利用した場合の低消費電力制御手法について述べる．下記説明においては，MT はマクロタスク，MTG はマクロタスクグラフ，PG はプロセッサグループ，PE はプロセッサエレメントを表わすものとする．また疑似代入文ブロックを BPA，繰り返しブロックを RB，サブルーチンブロックを SB と表記する．

3. コンパイラによる低消費電力化手法

2 章で述べたマルチグレイン並列処理により，プログラム中に存在するマルチレベル並列性を最大限引き出すことが可能となる．しかし利用可能な計算資源に対し常に十分な並列性が抽出されるとは限らず，このような場合には必要以上のプロセッサを動作させるための無駄な電力消費が発生してしまう．また，定められた時刻までに処理を終了すればよいようなリアルタイム処理では，ゆっくりと低動作周波数・低電圧で処理を行った方が消費電力を低く抑えられる場合がある．こ

表 1　周波数と電圧，エネルギーの比率

state	FULL	MID	LOW	OFF
frequency	1	1/2	1/4	0
voltage	1	0.87	0.71	0
dynamic energy	1	3/4	1/2	0

表 2　リーク電流による消費電力の比率

state	FULL	MID	LOW	OFF
static power	1	1	1	0

のような最小時間で処理を行う場合と，リアルタイム制約を満たす場合の両方に対する低消費電力化制御手法を提案する．

まず実行時間最小スケジューリングモードでは，プログラムのクリティカルパスにあたる処理に対しては低消費電力制御を行わず，本手法適用前の最小実行時間を保証する．デッドライン制約モードではプログラム終了のデッドラインを保証する範囲で，電力消費を最小になるように制御を行う．

3.1　低消費電力制御の対象モデル

ここでは本手法が制御対象とするマルチプロセッサシステムは以下の機能を持つとして議論を進める．

- プロセッサコア毎に周波数が可変
- 周波数に応じて電圧も低減可能
- プロセッサコア毎に電源の ON/OFF が可能

周波数・電圧 (FV) 制御には様々なアプローチがあるが，ここでは取り得る周波数の状態は離散的とし，各周波数状態に対して適切な電圧がハードウェア制約により定められているものとする．表 1 に各周波数における電圧，動作電流による消費エネルギーの比率の 1 例 (FULL を 400MHz，MID を 200MHz，LOW を 100MHz とし 90nm テクノロジを仮定した例) を示す．ここで消費エネルギーとは，同じ clock の処理を行った場合に消費されるエネルギーを表す．電源制御に関しては，完全に電源を遮断するため表 2 のように OFF の状態ではリーク電流による電力消費もないものとするが，プロセッサコアへのクロック供給を遮断できる場合は動作電力が 0，リークによる消費電力が通常と同じという状態を追加することができる．これらパラメータ類および状態の数は任意に変えることが可能である．また周波数を切り替えた際に，安定状態となるまでの状態遷移分のオーバーヘッドが存在するが，これもパラメータとして与えられるものとする．

3.2　低消費電力制御の対象 MTG

条件分岐等の実行時不確定性の有無により各 MTG に対してダイナミックスケジューリングかスタティックスケジューリングかが選択されるが，前述のように本論文ではスタティックスケジューリング適用 MTG のみを制御対象とする．ただし MTG の一部のみが条件分岐を含む場合，条件分岐のない部分的 MTG を擬似的に別階層として扱うことで本手法が適用可能となる．また各演算命令の処理コスト (クロック数) と消費エネルギーのリストを用いることで，周波数が FULL である場合の各 MT の処理コストと消費エネルギーを算出する．MID，LOW の周波数および消費エネルギーについては表 1 の比率を用いて計算する．

3.3　FV 制御適用 MTG の実行終了制約

以下ではスタティックスケジューリングの適用結果をもとに，各 MT に対して適切な動作周波数・電圧を決定する手順を述べる．すなわち図 3 のように，スタティックスケジューリングにより処理時間が最小となるよう MT が各 PG へ割り当てられているものとする．図中の Time が時間経過を表

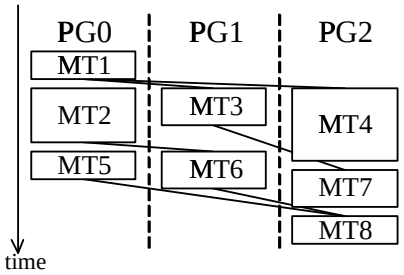

図 3　スタティックスケジューリング結果

しており，周波数が FULL の場合の 1 clock を単位時間とする．また MT 間の実線はデータ依存を表してる．まず FV 制御適用時の当該 MTG の実行終了時間を表すため，以下の定義を行う．MT_i に対して，

T_i：FV 制御適用後の MT_i の処理時間

T_{start_i}：MT_i の実行開始時刻

T_{end_i}：MT_i の実行終了時刻

を定義する．この時点では T_i は未定であり，当該 MTG の開始時刻を 0 とする．entry node である MT_1 ように，当該 PG が最初に実行する MT でありかつデータ依存する MT がないような MT_i の実行開始時刻 T_{start_i} は

$$T_{start_i} = 0$$

となり，実行終了時刻は

$$T_{end_i} = T_{start_i} + T_i = T_i$$

となる．一方それ以外の MT_i に関しては，当該 PG が先行的に実行するマクロタスク MT_j と MT_i がデータ依存するマクロタスク集合 $\{MT_k, MT_l, ...\}$ が終了した時点で MT_i の実行が開始されるため，実行開始時刻は

$$T_{start_i} = max(T_{end_j}, T_{end_k}, T_{end_l}, ...)$$

となり，終了時刻は

$$T_{end_i} = T_{start_i} + T_i$$

となる．図 3 を例として考えると MT_2，MT_3 は MT_1 が終了した時点で実行されるため開始時刻は $T_{start_2} = T_{start_3} = T_{end_1} = T_1$ となり，終了時刻は $T_{end_2} = T_{start_2} + T_2 = T_1 + T_2$，$T_{end_3} = T_{start_3} + T_3 = T_1 + T_3$ である．また MT_6 は MT_2 と MT_3 が終了した時点で実行が開始され，$T_{start_6} = max(T_{end_2}, T_{end_3}) = max(T_2 + T_1, T_3 + T_1)$ である．ここで max の各引数において共通する項は，max の外に出すことが可能であるため MT_6 では

$$T_{start_6} = max(T_2 + T_1, T_3 + T_1) = max(T_2, T_3) + T_1$$

となる．同様に計算し，exit node である MT_8 の終了時刻は $T_{end_8} = T_1 + T_8 + max(T_2 + T_5, T_6 + max(T_2, T_3), T_7 + max(T_3, T_4))$ と表される．

ここで一般形を考えると，exit node の終了時刻は

$$T_{end_e} = T_m + T_n + ... + max_1(...) + max_2(...) + ...$$

と表記される．entry node の実行開始時刻を 0 としたため，この T_{end_e} が FV 制御を適用した際の当該 MTG の実行終了時間となり，T_{MTG} と表記する．また当該 MTG に対して定められた処理終了時間を $T_{MTG\_deadline}$ とし，

$$T_{MTG} \leq T_{MTG\_deadline}$$

を満たすように各 MT_i の動作周波数を決定することが本手法の基本方針である．

3.4　FV 制御による低消費電力化

本節では，FV 制御適用時の MT_i の処理時間を T_i，FV 制御適用 MTG に定められた処理終了の時間を $T_{MTG\_deadline}$

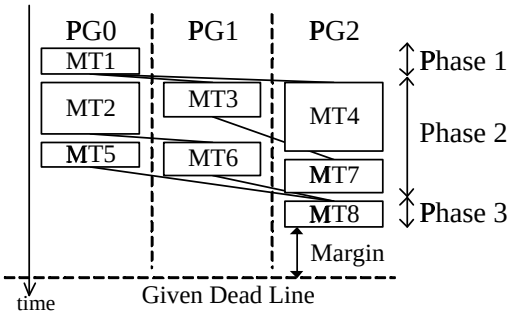

図 4 MTG の実行終了時間と Phase

として

$T_{MTG} = T_m + T_n + ... + max_1 + max_2 + ...$ ⋯ (a)

$T_{MTG} \leq T_{MTG\_deadline}$ ⋯ (b)

を満たす範囲で各 MT に適切な周波数を決める手法について述べる．便宜上 (a) 式の各項 T_m，T_n，... および max_1，max_2，... に対応する MT の集合それぞれを Phase と呼ぶこととする．これらの項は当該 MTG の実行時間の一部分であり，各項が表す時間帯が重なることはない．このため，異なる Phase が並列に実行されることもない (図 4)．ここで動作周波数を F_n と表し，$Phase_i$ に対して以下のパラメータを定義する．

$T_{sched_i}(F_n)$：周波数 F_n におけるスケジューリング長

$Energy_i(F_n)$：周波数 F_n における総消費エネルギー

$T_{sched_i}(F_n)$ は $Phase_i$ 全体を周波数 F_i で処理した際にかかる処理時間であり，$T_{sched_i}(FULL)$ が (a) 式における各項の取りうる最小値である．また $Energy_i(F_n)$ は $Phase_i$ 内に存在する MT を周波数 F_n で処理した場合の消費エネルギーの総和である．ここで F_n から F_m へ $Phase_i$ の周波数を一段階落とした場合を考える．スケジューリング長は $T_{sched_i}(F_n)$ から $T_{sched_i}(F_m)$ へ増加し，消費エネルギーは $Energy_i(F_n)$ から $Energy_i(F_m)$ へ減少する．これらを用いて，次のような利得 $Gain_i(F_m)$ を定義する．

$Gain_i(F_m) = -\frac{Energy_i(F_m) - Energy_i(F_n)}{T_{sched_i}(F_m) - T_{sched_i}(F_n)}$

$Gain_i(F_m)$ は動作周波数を変化させた際の，スケジューリング長の単位時間当りの増加に対する消費エネルギーの減少量を表す．すなわちスケジューリング長の増加分が等しい場合，$Gain_i(F_m)$ が大きい $Phase_i$ に優先して FV 制御を適用することで，より大きな消費電力削減が望める．

次に T_{MTG} が取り得る最小値を求めるが，これは $T_{sched_i}(FULL)$ の総和である．この最小値を $T_{MTG\_deadline}$ から引いた値を当該 MTG の余裕度 T_{MTG_margin} と定義する．すなわち

$T_{MTG\_margin} = T_{MTG\_deadline} - \sum T_{sched_i}(FULL)$

である．当該 MTG が最小実行時間で終了しなければならない場合，$T_{MTG\_margin} = 0$ であり各 Phase は動作周波数 FULL で実行と判定される．$T_{MTG\_margin} > 0$ である場合は T_{MTG_margin} の範囲内で，$Gain_i(F_m)$ に応じて各 Phase の電圧・動作周波数を低減させる．つまり消費エネルギーがより低減しやすい箇所の周波数を優先して落とすことで，効果的な低消費電力化が可能となる．ここで Phase が単一の MT ならば MT の動作周波数は Phase と同じとなる．Phase が複数 MT の集合であり max の項で表される場合は max の各引数に対して同様に Phase を定義し，動作周波数を決定する．以下に提案する FV 制御手法の手順を述べる．各 Phase の初期動作周波数は全て FULL とする．

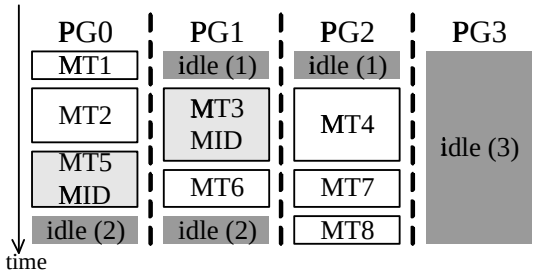

図 5 FV 制御適用結果

3.4.1 各 Phase に対する動作周波数決定アルゴリズム

step.1 対象 Phase の選択

各 Phase において，現在の動作周波数を F_n，それより一段階低い周波数を F_m とする．周波数の決定していない中で (初期状態では全 Phase)，利得 $Gain_i(F_m)$ が最も大きい $Phase_i$ を対象 Phase として選択する．**step.2** へ

step.2 対象 Phase の FV 制御適用判定

対象 Phase に対して，F_n から F_m への変更条件を以下の通りとする．

(1) 周波数切り替え時の状態遷移のオーバーヘッドも含め，余裕度 T_{MTG_margin} の範囲内で周波数 F_m で処理が終了する

(2) F_m 時の電力消費が FV 制御非適用時や電源遮断時よりも小さい

両方の条件を満たすか？

Yes：対象 Phase を F_m に変更．**step.3** へ

No：対象 Phase を F_n と決定．**step.4** へ

step.3 余裕度 T_{MTG_margin} の更新

対象 Phase をオーバーヘッドも含め動作周波数 F_m で処理するのに必要な時間を計算し，T_{MTG_margin} から引く．また対象 Phase の最低の動作周波数が F_m でありこれ以上落とせない場合，動作周波数を F_m と決定する．**step.4** へ

step.4 終了判定

終了条件は以下の通りである．

(1) 全ての Phase の周波数が決定する

(2) T_{MTG_margin} が 0 となる

のいずれかを満たすか？

Yes：終了．3.4.2 節の Phase に対する FV 制御に進む．

No：継続．**step.1** へ

終了時に T_{MTG_margin} が 0 でない場合，残りの T_{MTG_margin} は以下の条件を満たす $Phase_i$ の余裕度として与えられる．

- 周波数が最低でない
- 利得 $Gain_i(F_m)$ が最も大きい

3.4.2 Phase 内に対する FV 制御アルゴリズム

step.1 Phase 内 MT による分類

Phase が単一の MT であるか？

Yes：Phase の周波数をその MT の周波数とする．**exit**

No：**step.2** へ

step.2 max 項に対する FV 制御

Phase が複数の MT の集合であり max_i と対応する場合は，定められた動作周波数で Phase 全体を実行した際の実行時間を求め，これを $T_{max_i\_deadline}$ とする．max_i と，その引数 arg_{i_j} は以下の通り．

$max_i = max(arg_{i\_1}, arg_{i\_2}, ...)$

表 3　消費電力および状態遷移時間

dynamic power	220[mW]
static power	2.2[mW]
delay(FULL - MID - LOW)	0.1[ms]
delay({FULL, MID, LOW} - OFF)	0.2[ms]

$arg_{i\_j} = T_{i\_j\_m} + T_{i\_j\_n} + ... + max_{i\_j\_1} + max_{i\_j\_2} + ...$ すなわち arg_{i_j} は次の条件を満たすものとする．

$T_{i\_j\_m} + T_{i\_j\_n} ... + max_{i\_j\_1} + max_{i\_j\_2} ... \leq T_{max_i\_deadline}$

上式の各項を Phase と定義し，Phase に対する周波数決定 3.4.1 節に進む．

この際，各 arg の動作周波数 FULL 時の実行時間を計算し，実行時間の大きい順 (余裕度の小さい順) に FV 制御を適用する．複数の arg 中に同一の MT を表す項が含まれる場合があるが，一度 FV 制御が適用された MT に対しては周波数が既に決定されているため変更はされない．

3.4.1 節，**??** 節を再帰的に適用することで全 MT に対して適切な周波数が決定される．

3.5　電源制御による低消費電力化

3.4 節のように当該 MTG に対して FV 制御が適用され，次に電源の ON/OFF を制御することでプロセッサが処理を行っていない idle 部分の省電力化を計る．MTG 内に idle 部分が発生するのは次の 3 種類である．

(1)　データ依存がある MT の開始前
(2)　全割り当て MT の終了時
(3)　並列処理階層決定手法による idle

図 5 にそれぞれの場合の idle を示す．ここで PG3 は階層決定手法により不要と判断されたプロセッサであり，処理は割り当てられていない．以下の条件を満たす idle 部分に対し，電源遮断を適用する．

- 状態遷移オーバーヘッドも含め，idle の時間で電源 ON/OFF の切り替えが可能
- 電源遮断により電力消費が削減可能

3.6　MT の内部 MTG に対する低消費電力化

以上により当該 MTG の各 MT_i に対し，適切な動作周波数が決定される．しかし MT_i 内部に更に MTG_i が定義される場合，MTG_i 上に存在する MT を更に細かく制御した方がより消費電力を抑えられる場合がある．このため MT_i の実行時間 T_i より MTG_i が満たさなければならない実行終了時間 $T_{MTG_i\_deadline}$ を求め，同様に FV 制御と電源制御を適用する．このように MTG_i の内部を更に細かく制御した場合と，MTG_i 内部は制御せず MT_i 全体を同一の周波数で処理をした場合の消費エネルギーを比較し，より省電力効果の高い方を選択する．

4. 性 能 評 価

本章では提案する低消費電力化制御手法を OSCAR コンパイラに実装し，コンパイラ内で見積もられた電力消費推定結果について述べる．評価に用いた周波数，電力などのパラメータは表 1，2 の通りである．今回はプロセッサに対する電力消費の評価であるため，メモリやバスの消費電力は含まれていない．動作時の電力消費については演算命令毎の消費エネルギーから算出するのが望ましいが，今回は消費電力は常に一定であるとした．消費電力および FV 切り替え時の状態遷移時間を表 3 に示す．90nm のテクノロジ，400MHz を想定し，電力については Wattch[13] を用いた電力シミュレータより測定した．状態遷移時間や電圧・動作周波数の関係については Cooperative Voltage Scaling 手法[14] を参考にした．評価には SPEC95FP より 110.applu，101.tomcatv を用いた．今回は組み込み系を想定しているため，プロファイルを適用して回転数の精度等を高めている．

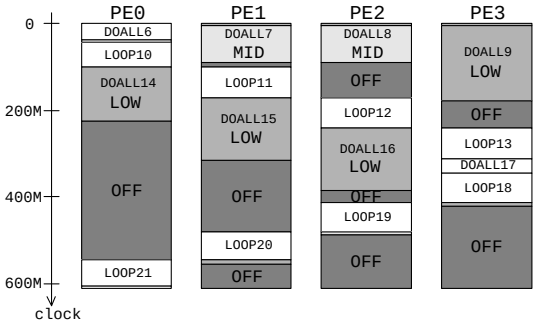

図 6　applu の低消費電力制御適用結果 (4PE)

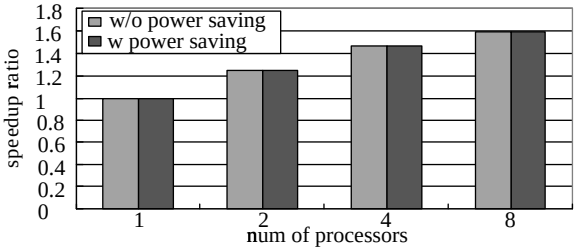

図 7　applu の速度向上率

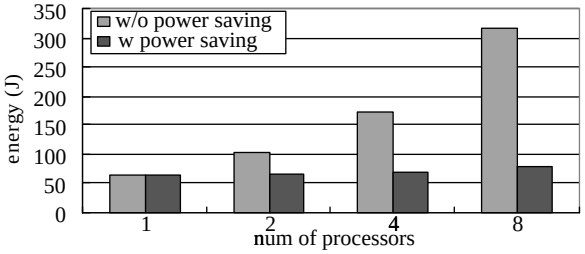

図 8　applu の総消費エネルギー

4.1　applu における最小スケジューリングモード

applu では，サブルーチン SSOR のメインループが実行時間のほぼ全体を占めている．このループのループボディは一部に条件分岐を含むため，手動で条件分岐のない部分を別階層とし，この部分に本手法を適用した．この階層は実行時間のうち約 7 割を占める．更にインライン展開とループ分割を行い，データローカリティを高めるようリストラクチャリングを行った．実行時間最小スケジューリングモードでの本手法を適用した場合，この階層は図 6 のように制御された (4 プロセッサ時)．クリティカルパスに相当する DOALL6，LOOP10〜13，DOALL17，LOOP18〜21，DOALL22 は余裕度が 0 であるため動作周波数 FULL と決定され，それ以外の MT はそれぞれの余裕度に応じて MID および LOW と決定された．更に idle となる部分に対しては電源遮断が適用され，低消費電力化が計られている．

この時の速度向上率を図 7 に，総消費エネルギーを図 8 に示す．それぞれ横軸がプロセッサ数であり，左のバーが本手法を用いない場合，右が本手法を適用した結果である．図 7 のように本手法の最小スケジューリングモードでは性能低下は見られず，適用前の処理性能を維持していることが分かる．

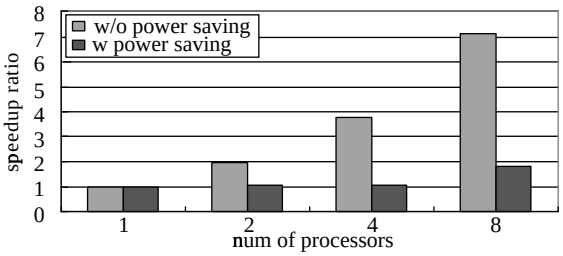

図 9 tomcatv の速度向上率

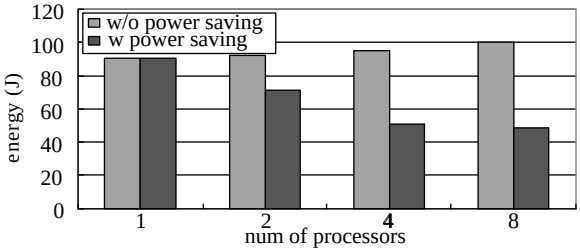

図 10 tomcatv の総消費エネルギー

図 8 はこの状況下でプロセッサが idle となる部分を利用して省電力化を行った場合の消費エネルギーであり，本手法を用いることで 2 PE 時で 36.3 %(102[J] から 65.0[J] へ低減)，4 PE 時で 60.7 %(174[J] から 68.4[J] へ低減)，8 PE 時で 75.5 %(318[J] から 77.9[J] へ低減) の消費エネルギーが削減されたことが分かる．

4.2 tomcatv におけるデッドライン制約モード

tomcatv ではデッドライン制約モードでの本手法の評価を行った．1 例として 1 プロセッサでの処理時間をデッドラインとした場合の速度向上率を図 9 に，総消費エネルギーを図 10 に示す．図 9 のように提案手法を用いない場合はプロセッサ数の増大とともに速度向上率は上がるが，本手法適用時には単一プロセッサ処理時間をデッドラインとし，この範囲で電圧・動作周波数を落として低消費電力化を計るため，並列処理時の速度向上率は逐次処理とほぼ同等であった．ただし 4 PE 時でほぼ全 MT の周波数が LOW となるため，8 PE 時で速度向上が得られた．図 10 が示すように消費エネルギーについてはプロセッサ数が増える毎に本手法を用いた場合の低消費電力化の効果は高まり，2 PE 時で 21.6 %(92.1[J] から 72.2[J] へ低減)，4 PE 時で 45.4 %(95.0[J] から 51.9[J] へ低減)，8 PE 時で 51.6 %(100[J] から 48.4[J] へ低減) の消費エネルギー削減という結果となった．

5. ま と め

本論文では，OSCAR 型ホモジニアスマルチコア上でのコンパイラの制御による低消費電力化手法を提案した．本手法では実行時間最小スケジューリングモードとデッドライン制約モードが存在し，本手法適用前の実効性能の保証，および可能な範囲での電力最小化といった様々な要求にフレキシブルに対応可能である．

提案手法を OSCAR コンパイラに組み込み，コンパイラ内で推定された消費エネルギーを検証したところ SPEC95FP の applu で最小処理時間での実行を保証したまま最大 75.5 % の消費エネルギー削減，tomcatv ではデッドライン制約を単一プロセッサでの処理時間とした場合において，デッドライン制約を満たしつつ最大 51.6 % の消費エネルギー削減を達成することができた．

今後の課題としてはシミュレータ上で電力など様々なパラメータを変化させた場合の詳細な評価，ダイナミックスケジューリング適用時での低消費電力制御などが挙げられる．

6. 謝　　辞

本研究の一部は NEDO “先進ヘテロジニアスマルチプロセッサ研究開発”，及び STARC (半導体理工学研究センター) “並列化コンパイラ協調型チップマルチプロセッサ技術” の支援により行われた．

参 考 文 献

1) M.Wolfe: High Performance Compilers for Parallel Computing, *Addison-Wesley Publishing Company* (1996).
2) Eigenmann, R., Hoeflinger, J. and Padua, D.: On the Automatic Parallelization of the Perfect Benchmarks, *IEEE Trans. on parallel and distributed systems*, Vol. 9, No. 1 (1998).
3) Hall, M. W., Anderson, J. M., Amarasinghe, S. P., Murphy, B. R., Liao, S., Bugnion, E. and Lam, M. S.: Maximizing Multiprocessor Performance with the SUIF Compiler, *IEEE Computer* (1996).
4) Gonzalez, M., Martorell, X., Oliver, J., Ayguade, E. and Labarta, J.: Code Generation and Run-time Support for Multi-level Parallelism Exploitation, *Proc. of the 8st International Workshop on Compilers for Parallel Computing* (2000).
5) Saito, H., Stavakos, N. and Polychronopoulos, C.: Multithreading Runtime Support for Loop and Functional Parallelism, *Proc. of the International Symposium on High Performdance Computing* (1999).
6) 本多弘樹, 岩田雅彦, 笠原博徳: Fortran プログラム粗粒度タスク間の並列性検出手法, 電子情報通信学会論文誌, Vol. J73-D-1, No. 12, pp. 951–960 (1990).
7) H.Kasahara and et al: A Multi-grain Parallizing Compilation Scheme on OSCAR, *Proc. 4th Workshop on Language and Compilers for Parallel Computing* (1991).
8) 笠原博徳: 最先端の自動並列化コンパイラ技術, 情報処理, Vol.44 No. 4(通巻 458 号), pp.384-392 (2003).
9) Albonesi, D. H. and et al: Dynamically tuning processor resources with adaptive processing, *IEEE Computer* (2003).
10) Wu, Q., Juang, P., Martonosi, M. and Clark, D. W.: Formal Online Methods for Voltage/Frequency Control in Multiple Clock Domain Microprocessors, *Eleventh International Conference on Architectural Support for Programming Languages and Operating Systems* (2004).
11) 小幡元樹, 白子準, 神長浩気, 石坂一久, 笠原博徳: マルチグレイン並列処理のための階層的並列処理制御手法, 情報処理学会論文誌, Vol. 44, No. 4 (2003).
12) 白子準, 長澤耕平, 石坂一久, 小幡元樹, 笠原博徳: マルチグレイン並列性向上のための選択的インライン展開手法, 情報処理学会論文誌, Vol. 45, No. 5 (2004).
13) Brooks, D., Tiwari, V. and Martonosi, M.: Wattch: A Framework for Architectural-Level Power Analysis and Optimizations, *Proc. of the 27th ISCA* (2000).
14) Kawaguchi, H., Shin, Y. and Sakurai, T.: μ ITRON-LP: Power-Conscious Real-Time OS Based on Cooperative Voltage Scaling for Multimedia Applications, *IEEE Transactions on multimedia* (2005).