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

2005－ARC－164 (11)
2005／8／3

GALS型SoCの低消費電力化のためのタスクスケジューリング手法

渡辺 亮[†] 近藤正章[†] 今井 雅[†]
中村 宏[†] 南谷 崇[†]

性能制約の下で動作するシステムにおいては，Dynamic Voltage Scaling(DVS) によって性能制約を満たしうる範囲で適切なクロック周波数・電源電圧を設定することにより消費エネルギーの削減が可能である．GALS(Globally Asynchronous Locally Synchronous) 型のシステム・オン・チップ (SoC) はプロセッサ毎に独立の周波数・電圧を設定できるため，空間的に負荷がばらつく際にも各プロセッサの設定を個別に最適化することで効率的な低消費エネルギー化が可能になる．本稿では，性能制約下での GALS 型 SoC のタスクスケジューリング・電圧スケジューリングにおいて，NP 困難であることが知られている消費エネルギー最適化問題に対し，現実的な時間で解を与える手法について述べる．

Task and Voltage Scheduling for Reducing Energy Consumption on GALS System-on-Chip

Ryo WATANABE,[†] Masaaki KONDO,[†] Masashi IMAI,[†]
Hiroshi NAKAMURA[†] and Takashi NANYA[†]

Energy consumption in systems working under a performance constraint, can be reduced by applying Dynamic Voltage Scaling (DVS). DVS lowers the clock frequency and supply voltage of the processor so that the execution time fits into a given performance constraint. In Globally Asynchronous Locally Synchronous (GALS) SoCs, individual frequency and supply voltage can be set for each processor. Thus, energy consumption is reduced effectively selecting frequency and supply voltage suitable for tasks on each processor. However, it is known to be NP-hard to find the energy-optimal task and voltage scheduling for GALS SoC with performance constraint. In this paper, we propose a method for finding the optimal scheduling with reasonable time costs.

1. はじめに

動画再生をはじめとする信号処理などのように，最低限満たすべき性能制約が存在する場合，Dynamic Voltage Scaling(DVS)[1)2)] を利用して性能制約を満たしうる最低限度までプロセッサの周波数・電源電圧を低下させ動作させることにより，実質的に性能を低下させることなくプロセッサの消費エネルギーを抑えることができる．例えば，石原らは，実行時間の制約下における消費エネルギーの最適化について考察し，適切な電圧設定を決定する手法を提案している[4)]．

近年では，複数のプロセッサを1チップに集積したシステムが多数現れている．このようなシステムでは，アプリケーションタスクを各プロセッサに割り当て，適切に通信を行いながら処理を進めていくことで高性能化が達成できる．このような複数のプロセッサによる並列処理においては，各プロセッサで実行するタスクの処理負荷にばらつきが生じることがある．ここで，Globally Asynchronous Locally Synchronous(GALS) 型の設計[3)] を導入し，チップ上の各プロセッサ内を同期領域 (Locally-Synchronous module) とし，プロセッサ間は非同期的に通信を行うようにすると，各プロセッサの負荷に応じた適切な周波数・電源電圧を個別に設定することができ，さらなる消費エネルギーの削減が可能になる．

1.1 関連研究

このようなマルチプロセッサシステム (より一般的には，汎用プロセッサに限らず DSP 等の様々な専用プロセッサや周辺回路を集積する SoC) の消費エネルギー最適化に関して，Chang らはタスクが既にプロセッサに割り付け済みであるという条件の下で，パイプライン化も考慮した消費エネルギー最適化手法を提案している[5)]．さらに，一般のタスクグラフについて分割と割り付けを行うアルゴリズムを提案している[6)]．Luo らは，プロセッサ間でタスクの移動を行い，消費電力の分布を平均化することでバッテリ駆動時間を延ばすスケジューリング手法を[7)]．また Zhang らは，タ

† 東京大学先端科学技術研究センター
Research Center for Advanced Science and Technology (RCAST), The University of Tokyo

スクのプロセッサへの割り付けや実行順序の設定が，電圧選択による消費エネルギー削減の可能性に与える影響について述べている[8]．しかし，これらの既存研究ではプロセッサの消費エネルギーを削減することが目的であり，プロセッサ間の通信に要するエネルギーは考慮されていない．

SoC マルチプロセッサシステムにおいては，バス等のインタフェースを介するプロセッサ間通信が性能面でも消費エネルギー面でも高コストであり，通信に非同期インタフェースを必要とする GALS 型システムではこの現象はさらに顕著である．したがって，プロセッサ内での演算に要する消費エネルギーを低減するために多数のプロセッサにアプリケーションタスクを分配する際には，通信のオーバーヘッドを考慮し，演算と通信に要する消費エネルギーの総和としてのシステムの消費エネルギーを低減するするようなタスクスケジューリング，および電圧スケジューリングが必要となる．

しかしながら，GALS 型 SoC システムにおける性能制約下でのタスク・電圧スケジューリングは複雑であり，消費エネルギー最適化問題は NP 困難であることが知られている[9][10]．そこで Varatkar らは，演算と通信の消費エネルギーのバランスを最適化するヒューリスティックアルゴリズムを提案している[9]．

1.2 本研究の意義

本稿では，性能制約を持ったアプリケーションを GALS 型 SoC 上で実行するときに，消費エネルギーを可能な限り抑えるように，アプリケーションタスクをプロセッサへのマッピングする手法，および各プロセッサの電圧を決定する手法について検討と評価を行う．NP 困難なスケジューリング問題に対して，問題の有用性を失わない範囲で適切な仮定を置くことにより，多項式時間で解くことができる問題となり，現実的な時間で優れたスケジューリングの解を求めることが可能になる．

Varatkar ら[9] の手法に対し，我々はアプリケーションの処理に異なる仮定である処理のパイプライン化を導入する．これによりタスク間の並列性を活用し，特にプロセッサ数の多い環境において，より柔軟なタスクの割り付け選択による最適化が可能であると考えられる．

2. 問題定義

本稿では，チップ上に M 個のプロセッサが搭載された GALS 型 SoC を対象に，システムを低消費エネルギー化するスケジューリングを求める問題を考える．各プロセッサはそれぞれが 1 つの同期領域として定義されており，それぞれ異なる周波数および電源電圧で動作することができるものとする．

また，対象のアプリケーションは N 個のタスクから構成されているとする．タスクは小単位のプログラムであり，その演算量および 2 つのタスク間の依存関係の有無，また依存関係のあるタスク間で受け渡しされるデータ量についてはソフトウェアプロファイリング等の手法により既知であるものとする．

以下の各節にて，スケジューリングにおける消費エネルギー最適化問題を表現するために用いるハードウェアおよびソフトウェアのモデルについて述べたのち，本稿における問題設定について述べる．

2.1 ハードウェアモデル

対象の GALS 型 SoC を記述するモデルとして有向グラフを導入し，これをハードウェアグラフと呼ぶ．図 1 にその例を示す．各頂点はプロセッサ P_i を表す．各プロセッサはいずれかのプロセッサクラス C_j に属しており，これは汎用プロセッサ，DSP，$\cdots$ といったプロセッサの種類を表す．各プロセッサクラス C_j は $\{TC_j, EC_j\}$ というパラメータの組を持っており，これらはクラス C_j に属するプロセッサが演算量 1 を処理完了するのに要する時間とエネルギーを表す．ここでいう演算量とは，演算の大きさを規格化した値である．これらの値はプロセッサの動作をプロファイリングすることによって得るものとする．プロセッサが複数のクロック周波数および電源電圧に対応する場合，クラス C_j は対応する周波数と電圧の組に対してそれぞれ $\{TC_j, EC_j\}$ の組を持つことになる．

P_i から出て P_j に入る有向枝は，P_i が送り側，P_j が受け側であるような片方向通信リンク L_{ij} を表す．各リンク L_{ij} はパラメータの組 $\{TL_{ij}, EL_{ij}\}$ を持ち，これはリンク L_{ij} が通信量 1 を転送完了するのに要する時間とエネルギーを表す．通信量は演算量と同様，通信データ量を規格化した値である．

2.2 ソフトウェアモデル

対象の SoC 上で実行するアプリケーションの処理の流れを表すため，非循環有向グラフを導入し，これをタスクグラフと呼ぶ．タスクグラフは，相互に依存関係を持ち，データのやりとりを行うタスクの集合を表現する．

図 2 にその例を示す．各頂点はタスク T_i を表す．各タスク T_i は，それがクラス C_j のプロセッサで処理される場合の演算量を表すパラメータ A_{ij} を持つ．クラス毎にタスクの演算量を定義する理由は，SoC 上に集積されたプロセッサの中に専用プロセッサが含まれうるためである．ある種類の専用プロセッサは，特定のタスクを汎用プロセッサより非常に高速に処理でき(すなわち，そのプロセッサから見た場合の演算量は小さい)，またその逆もありうる．したがってタスクの演算負荷の大きさは，タスク中の命令数だけでは正確に表現することはできない．

T_i から Tj に向かう有向枝は，2 つのタスク間に依存関係 D_{ij} が存在することを示す．各依存関係 D_{ij} は受け渡しされるべきデータ量を表すパラメータ d_{ij} を持つ．すなわちタスク T_i の実行が完了した後，タスク T_i からタスク T_j に d_{ij} に相当する量のデータが受け渡されなければタスク T_j は実行を開始できない．2 つのタスクが同じプロセッサ上で処理されるならば即座に実行が可能である．

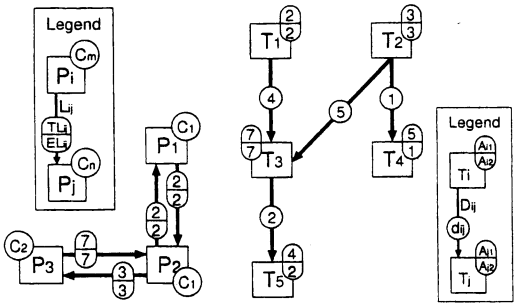

図 1　ハードウェアグラフの例.

図 2　タスクグラフの例.

2.3　モデルの簡単化

第 1 章で述べたように, GALS 型 SoC における性能制約下での消費エネルギー最適化問題は NP 困難であるため, 問題の有用性を失わない範囲で問題空間を狭めるためにいくつかの仮定をおく.

- **性能制約としてはスループット制約を与え, アプリケーションタスクは SoC 上でパイプライン的に実行される.**
  対象のアプリケーションには, ビデオエンコーダ・デコーダを始めとするストリーム処理系のアプリケーションを仮定する. これらのアプリケーションの主要な性能制約はスループットであり, メインループ内の一連の処理をタスクグラフで表すとして, これを与えられたスループットで処理しなければならない.
  この条件下で消費エネルギーを削減するため, アプリケーションはタスクレベルでパイプライン化して処理するものとする. パイプライン化によりタスク間の並列性を活用することができるため, スループットを維持したまま各プロセッサの負荷を削減することができる. スループット制約を表すパラメータには, パイプライン周期 $T_{pipeline}$ を用いる.
  スループットを維持するのが重要なアプリケーションという位置づけから, パイプライン化に伴うレイテンシ (パイプライン段数) の増加は無限に許すものとする.
- **SoC 上の全プロセッサ・通信リンクは均一.**
  SoC 上の全てのプロセッサおよび通信リンクは均一であるとする. すなわち, 対象とする SoC 上のプロセッサはすべてある種の汎用プロセッサであると考える. この仮定によってプロセッサクラスを考慮する必要がなくなり, パラメータ $\{TC_j, EC_j\}$ を $\{TC, EC\}$ で, また A_{ij} を A_i で, それぞれ置き換えることができる.
  また, 全てのリンクの速度と消費エネルギーは等しいとする. これにより, パラメータ $\{TL_{ij}, EL_{ij}\}$ を $\{TL, EL\}$ と置き換えることができる. さらに, レイテンシ制約が無いためパラメータ TL は不要である.
- **演算と通信は並列に処理可能.**
  プロセッサと通信リンクは並列に動作し, データを送受信している間にもプロセッサは演算を行うことができるとする. この仮定を用いると, プロセッサ間の通信に要する時間が演算処理のスケジューリングに影響を与えなくなり, 性能制約に反するのは, プロセッサのとりうる最大周波数を用いても $T_{pipeline}$ の時間内に実行を完了できない処理量のタスク群をプロセッサに割り付けた場合に限られる.
- **周波数・電圧は実行中には変更しない.**
  各プロセッサの周波数・電圧は自由に決めることができるが, それらは実行中には変更しないものとする.

これらの仮定により, 先に述べた最適化問題は, システム全体の消費エネルギーの最小値を与えるような, 各タスクのプロセッサへの割り付けパターンを求める問題と考えることができる.

3.　スケジューリングアルゴリズム

第 2 章で述べた定義と仮定により, タスク・電圧のスケジューリングの手順は, 各タスクをどのプロセッサで実行するかの割り付けを決定し, その後, 各プロセッサに最適な電源電圧を選択するという 2 つの手順に簡略化することができる.

3.1　電 圧 選 択

第 2 章でおいた仮定の下では, 各プロセッサの消費エネルギーを最小化する電源電圧選択アルゴリズムは単純である. タスク割り付けが完了すると, 各プロセッサに割り付けられたタスクの総演算量が求まるため, 総演算量をパイプライン周期 $T_{pipeline}$ 以内に実行完了しうるもののうちで最小の周波数・電圧に相当する $\{TC, EC\}$ の組を選んで設定する.

この電圧選択が他のプロセッサのスケジューリングの最適化に影響を与えることはない. すなわち, タスク割り付けが決まると各プロセッサの電圧設定はそれぞれ独立かつ一意に決定されるので, 消費エネルギーを最小化するタスク・電圧のスケジューリングを求めることは, 消費エネルギーを最小化するようなタスクのプロセッサへの割り付けパターンを求めることに等しい.

3.2　タスク割り付け

前節までで, スケジューリングの最適化のためには, 消費エネルギーを最小化するようなタスク割り付けのパターンを求めればよいことが分かった. しかしながら, タスク割り付けのパターンは M^N 通りあるため, プロセッサ数・タスク数が増加した場合には解の探索が現実的でなくなる.

本節では, 現実的な計算量で優れた消費エネルギー削減を達成するタスク割り付けパターンを探索する方法について述べる.

3.2.1　負荷平均化

プロセッサ間通信のエネルギーオーバーヘッドが存

在しない場合には，全てのプロセッサの演算負荷が均等になるようにタスクの割り付けを行い，全てのプロセッサを同程度の周波数・電圧で駆動するのが最もエネルギー的に効率が良くなると考えられる．

そこで，まずは全プロセッサの割り付け演算量を平均化する単純なアルゴリズムから始めることにする．図3にこのアルゴリズムの概略を示す．このアルゴリズムは演算量の大きなタスクから順番に，それまでに割り付けられた演算量の総量が一番小さいプロセッサを選んで割り付けを行っていく．得られる解は多くの場合でほぼ演算量を平均化したパターンになるが，割り付け演算量の分散を必ず最小化することは保証されない．

このアルゴリズムの計算量は $O(N)$ である．

```
procList = [P_1, P_2, ..., P_M]
taskList = [T_1, T_2, ..., T_N]
while (taskList is not empty)
    select a task T_i with largest A_i from taskList
    select a processor P_j from procList, with
        minimum sum of computation load
        of mapped tasks
    map T_i to P_j
    remove T_i from taskList
end while
```

図3　負荷平均化アルゴリズム

3.2.2　エネルギー先読み評価

通信リンクの消費エネルギーが0でない場合には，負荷平均化はあまり良い解を与えない．これは割り付け先プロセッサ選択時に通信エネルギーを無視しているためである．そこで，プロセッサ選択時の評価に通信リンクの消費エネルギー e_L を導入したアルゴリズムを考える．

負荷平均化は一種の分枝限定法である。演算量の大きなタスクから順に，M 個のプロセッサのどれに割り付けるかによる分枝操作を N 回行って，想定される全ての場合（M^N 通り）が作られる（図4）が，負荷平均化ではタスク T_i の割り付けに関する分枝操作を行う際，その直前の状態での割り付け演算量の総量が最小のプロセッサに割り付ける場合以外の分枝をすべて打ち切る．

一方，通信も考慮したエネルギー評価による分枝限定法では，分枝操作後に生成された各状態で，割り付けられた全タスクを処理するのに要するシステムの消費エネルギーを評価し，分枝の限定を行う．タスクは演算量の降順でソートされており，$T_1, T_2, \cdots, T_N$ の順番でタスクを割り付けるとする．図4は，2個のプロセッサ P_1, P_2 に対しタスク T_2 までの割り付けが完了し，次に T_3 の割り付けを決めようとする状況を表している．

T_i の割り付け先を決める際には，T_i および，その次に割り付けるタスクである T_{i+1} の割り付け後の全てのタスク割り付けパターンについて，システムの消費エネルギーを評価する．2個のタスクの割り付けパターンは M^2 通りあり，そのそれぞれについて，T_1〜T_{i+1} の処理に要する演算と通信の消費エネルギーの和を求める．これを図4に示した例のように，2個のタスクの割り付け先プロセッサ番号に対応して E_{jk} とする．

全てのパターンに対して E_{jk} を求めたら，最小の E_{jk} を与える割り付け以外の T_i の分枝，すなわち T_i を P_j に割り付ける以外の分枝を全て打ち切る．たとえば，図4の場合において $E_{11}, E_{12}, E_{21}, E_{22}$ のうち E_{12} が最小であったとする．このとき，T_3 の割り付け先は P_1 であるから，T_3 を P_2 に割り付ける分枝を打ち切る．

T_i の割り付けを決めるために，T_i の割り付け後のエネルギー E_j ではなく T_{i+1} の割り付け後のエネルギー E_{jk} を調べるのは，問題の性質上，部分問題における最適解からの分枝が全体での最適解に繋がっているとは限らないためである．望ましくない部分最適解を選択してしまうことを防ぐため，1個先のタスクまでの全割り付けパターンについてエネルギーの評価を行ってから（先読み評価），割り付け先のプロセッサを選択する．

このアルゴリズムの計算量は $O(M^2N)$ である．

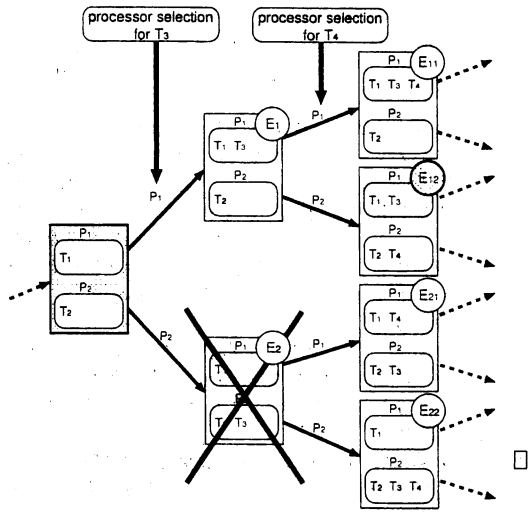

図4　分枝によるタスク割り付けパターンの生成と，分枝限定のための消費エネルギー評価の例．

3.2.3　先読み評価 + 解の繰り返し改良

エネルギー先読み評価のアルゴリズムはヒューリスティックであり，必ずしもその解がエネルギーを最適化するとは限らない．そこで，エネルギー先読み評価で求めた解からスタートして，さらに消費エネルギーを削減する解を求めるアルゴリズムを考える．

すべてのタスクの割り付けが完了した仮の解が与えられたとき，いずれかのタスクを選んで，現在割り付

けられているものと異なるプロセッサに移動させることにより新しい解を作ることができる．全てのタスクについてこのような移動を試行して新しい解を生成し，その中にもし現在の解より優れた解があればそれを選択し，新しい解を元にして再度タスクの移動を試行する．このようにして，より優れた解が見つからなくなるまで試行を繰り返して解の改良を行う．

タスクの移動だけでは演算量のばらつきが大きくなりやすく，良い解が見つかりにくいと考えられる．そこで，タスクの交換も試行パターンに含めるため，2回までのタスクの移動を全パターン試行してから，新しい解を選択する．a 新しい解の探索～解の選択の 1 回に要する計算量は $O(N^2M^2)$ である．

4. 評　　価

第 3 章で詳述したアルゴリズムの有効性を検証するため，ランダムに生成したタスクグラフによる評価を行った．

ハードウェアはプロセッサ数 4 個からなるものとする．周波数・電圧レベルは Pentium M の周波数・電圧モデルを線形近似したものを利用し，周波数・電圧は最大値以下であれば無段階で任意の値を設定可能であるとする．設定可能な最大周波数はスループット制約と合わせて設定しており，最大周波数は設定されたパイプライン周期内に演算量 250 を処理完了できる速さとした．すなわち，このシステムは演算量 1,000 までならばパイプライン周期内に処理を完了することができる．一方，消費エネルギーについては，最大電圧時に $EC = 1$ と定義し，それ以下の場合については周波数・電圧モデルに従って決定した．

通信リンクについては，プロセッサ間通信に要するエネルギーの大小にかかわらず提案手法が有効であることを検証するため，通信リンクの消費エネルギー EL の，プロセッサの消費エネルギー EC に対する相対値を様々に変化させながら評価を行った．

タスクグラフのタスク数は 12 個とし，各タスクの演算量およびタスク間の通信量はそれぞれ 10～250 の間でランダムに設定した．プロセッサに性能の余裕がどれだけ残っているかによりエネルギー削減効果は異なってくることから，提案手法の有効性がアプリケーション負荷の大きさに影響されないことを示すため，タスクの総演算量に 500, 700, 900 という制約条件を設けて各 20 個のタスクグラフを生成した．

評価にあたっては手法間の比較だけでなく，絶対的な解の質を評価するため，全探索によって求めた解も利用した．

図 5 の 3 つのグラフはそれぞれ，タスクの総演算量が 500, 700, 900 の場合について，各手法が与えたスケジューリングによるシステムの消費エネルギーを示している．消費エネルギーは，通信を全く考慮しない手法である負荷平均化の消費エネルギーを 1 とした相対値で表す．グラフの値は各 20 個のタスクグラフについて評価した値の平均値であり，横軸は通信リンクの消費エネルギー EL の相対値を表している．

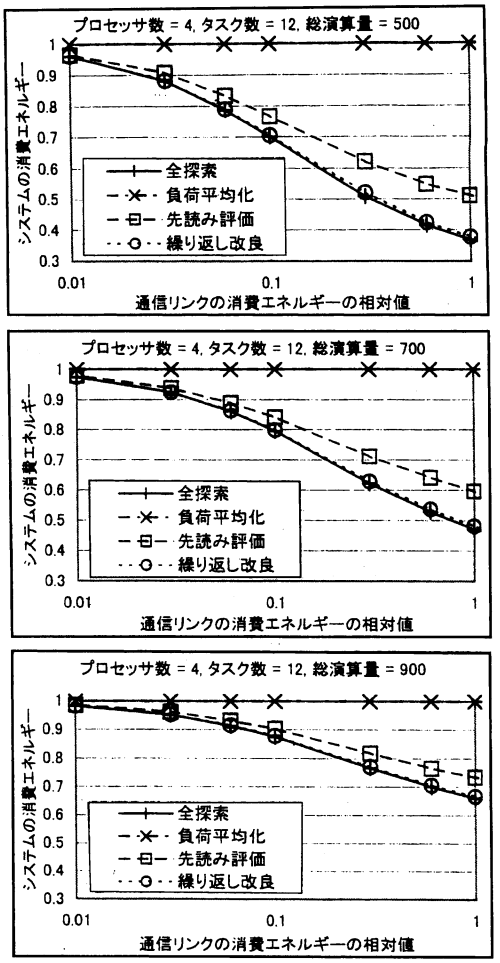

図 5　ランダムなタスクグラフによる評価結果 (上から，タスクの総演算量が 500, 700, 900 の場合)．横軸は通信リンクの消費エネルギー EL の，プロセッサの消費エネルギー EC に対する相対値，縦軸は各手法が与えたスケジューリングによるシステムの消費エネルギーを，負荷平均化の結果を 1 とした相対値で表したもの．

演算量平均化は EL が小さい場合には理論最適解である全探索との差が小さく，効果的に消費エネルギーを削減していると言えるが，削減対象が演算エネルギーにほぼ限られるため EL が大きくなるにつれて解が悪化する．これに対し，エネルギー先読み評価は EL の値にかかわらず優れた消費エネルギー削減を達成している．さらに，先読み評価で得られた解に対し繰り返し改良を実行することにより，ほぼ理論値と同じエネルギー削減を実現することができる．

また，タスクの総演算量が増加すると，性能制約を

満たしたまま組み合わせ可能なタスクのパターンが減少するため，消費エネルギー削減の余地は減少する傾向にあるが，そのような条件下でも可能な限りのエネルギーを削減できていることがわかる．

続いて，全探索による探索が不可能になるような大規模なタスクグラフを生成して評価を行い，問題のサイズが増加しても優れた消費エネルギー削減を実現できるかどうか検証した．プロセッサ数 8 個，タスク数 40 個の条件で，タスクの演算量・通信量は 10〜100 の間でランダム，タスクの総演算量は 2000，プロセッサが最大周波数において $T_{pipeline}$ 以内に処理可能な演算量は 400 とした．その他の条件は前の実験と同様である．

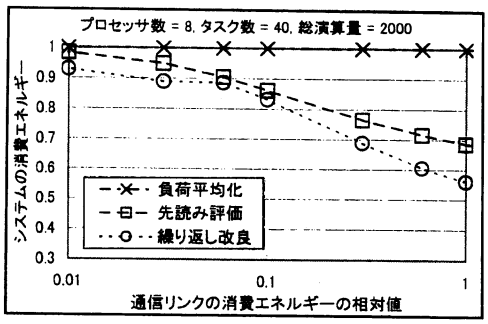

図 6　より大規模なタスクグラフによる評価結果．全探索の結果が含まれない点以外，グラフの構成は図 5 と同様である．

図 6 に評価結果を示す．全探索の結果が含まれない以外，グラフの構成は図 5 と同様である．問題のサイズを拡大しても，提案手法により効率的にシステムの消費エネルギーを削減できていることがわかる．

5. 結　　論

性能制約の下で動作するシステムにおいては，DVS によって性能制約に対し適切な周波数・電圧を設定することで消費エネルギーを削減できる．しかし，GALS 型 SoC における性能制約下でのの消費エネルギーを最適化するタスク・電圧スケジューリングを求める問題は NP 困難である．

我々はこの問題に対し，実用性を失わない範囲でいくつかの仮定を置くことにより，単純なアルゴリズムによって解を与えることを可能にした．消費エネルギーに演算と通信のそれぞれが占める比率は，システムの構成やインタフェースの種類，またテクノロジの進歩などで様々に変わりうるが，提案手法を用いると通信オーバーヘッドの比重に関わらず高い水準での消費エネルギー削減が可能であることが評価結果から示された．

今後は，実際のアプリケーションのプロファイリングにより得たタスクグラフを用いて評価を行い，提案手法が実用的に利用可能であることを検証する予定である．また，モデルの単純化のためにおいたいくつかの仮定について再検討を行い，より一般的な条件に対し適用可能なスケジューリング手法を構築していくことを考えている．

謝辞　本研究の一部は，科学研究費補助金 (基盤研究 (B)、課題番号 17300013)，及び (株) 半導体理工学研究センターとの共同研究によるものである．

参 考 文 献

1) T. Burd, T. Pering, A. Stratakos, R. Brodersen, "A Dynamic Voltage Scaled Microprocessor System", IEEE Journal of Solid-State Circuits, vol. 35, pp. 1571-1580, 2000.
2) Anantha P. Chandrakasan, Samuel Sheng, Robert W. Brodersen, "Low-Power CMOS Digital Design", IEEE Journal of Solid-State Circuits, Vol. 27, No. 4, pp. 473-484, Apr. 1992
3) J. Muttersbach, T. Villiger, H. Kaeslin, N. Felber, W. Fichtner, "Globally-Asynchronous Locally-Synchronous Architectures to Simplifu the Design of On-Chip Systems", Proceedings of 12th ASIC/SOC Conference, 1999.
4) T. Ishihara, H. Yasuura, "Voltage Scheduling Problem for Dynamically Variable Voltage Processors", Proceedings of the international symposium on Low Power Electronics and Design, pp. 197-202, 1998.
5) J.-M. Chang, M. Pedram, "Energy Minimization Using Multiple Supply Voltages", Proceedings of the international symposium on Low Power Electronics and Design, 1996.
6) J.-M. Chang, M. Pedram, "Codex-dp: Co-design of Communicating Systems Using Dynamic Programming", Proceedings of the conference on Design, Automation and Test in Europe, 1999.
7) J. Luo, N. K. Jha, "Battery-Aware Static Scheduling for Distributed Real-Time Embedded Systems", Proceedings of the 38th conference on Design Automation, pp. 444-449, 2001.
8) Y. Zhang, X. Hu, and D. Chen, "Task Scheduling and Voltage Selection for Energy Minimization", Proceedings of the 39th conference on Design Automation, pp. 183-188, 2002.
9) G. Varatkar, R. Marculescu, "Communication-Aware Task Scheduling and Voltage Selection for Total Systems Energy Minimization", Proceedings of the International Conference on Computer Aided Design, pp. 510-517, 2003.
10) H. El-Rewini, H. H. Ali, T. Lewis, "Task Scheduling in Multiprocessing Systems", IEEE Computer, pp. 27-37, Dec. 1995.