可変電源電圧プロセッサに対するリアルタイムタスクスケジューリング手法

大隈, 孝憲
九州大学大学院システム情報科学研究科情報工学専攻

石原, 亨
九州大学大学院システム情報科学研究科情報工学専攻

安浦, 寛人
九州大学大学院システム情報科学研究科情報工学専攻

出版情報：電子情報通信学会技術研究報告. CPSY, コンピュータシステム. 98 (449), pp.87-94, 1998-12-11. 電子情報通信学会
バージョン：
権利関係：

論　文

可変電源電圧プロセッサに対するリアルタイムタスクスケジューリング手法

大隈　孝憲[†]　　石原　亨[†]　　安浦　寛人[†]

Real-Time Task Scheduling for a Variable Voltage Processor

Takanori OKUMA[†], Tohru ISHIHARA[†], and Hiroto YASUURA[†]

あらまし
動的に電源電圧を変化させ，その性能と消費エネルギーを制御できる可変電源電圧プロセッサをリアルタイムシステムに用いる場合のタスクスケジューリング手法を提案する．タスクの処理に対して電源電圧を下げることで，システムの消費エネルギーを削減することができるが，電源電圧を下げ過ぎると，タスクを処理する時間が長くなり，リアルタイム性が満たされなくなる．本稿では，タスクスケジューリングの際に CPU 時間と電源電圧を同時に割当てることにより，時間制約を満たす範囲でのシステムの消費エネルギーを最小化する手法を提案し，その効果をシミュレーション実験によって評価する．

キーワード　低消費電力設計，可変電源電圧プロセッサ，実時間処理，スケジューリング

1. はじめに

システムの全機能を 1 チップで実現するシステム LSI は，微細加工技術の進歩によって製造が可能となり，携帯型情報機器をはじめオーディオ装置，自動車などの様々な製品に組込まれるようになってきた．これらのシステムの多くは実時間処理を行うため，リアルタイム性への要求が厳しく，高速化が望まれる．一方，LSI の大規模化に伴い，チップの発熱が高集積化と高速化を制限する最大の要因となっている．また，軽量のエネルギー源で多様な処理を長時間実行するシステムの実現に対する要求も高まっている．

システムにリアルタイム性を持たせるためには，高い速度性能を持つ汎用のプロセッサを利用することが考えられるが，多くの場合，低消費エネルギー化に対する要求が満足されない．なぜなら，一般的にプロセッサのエネルギー消費とプロセッサの動作周波数との間にはトレードオフの関係があるためである．つまり，高速化と低エネルギー化を同時に満たすことは困難である．この問題を解消するために，文献 [7] において，電源電圧を動的に変更できる可変電源電圧プロセッサが提案されている．可変電源電圧プロセッサは以下の性質を持つ．

- プロセッサは電圧制御命令を持ち，アプリケーションやオペレーティングシステムがこの命令を使うことにより，プロセッサの電源電圧とクロック周波数を任意の実行ステップで変更できる．
- プロセッサは電源電圧の変更による回路遅延の変化に合わせたクロック周波数を発生する．

可変電源電圧プロセッサにより，制約時間が厳しく，短い時間で処理しなければならないタスクには，高い電圧で高速に処理を行ない，制約時間が緩く，高い処理能力を必要としないタスクには，低い電圧で低電力に処理を行なうことができる．一般に消費電力は電源電圧の 2 乗に比例するので，低い電圧で処理を行なうことは電力削減に大きく貢献できる．図 1 に例を示す． この例において電圧，周波数，エネルギーの関係を図中の表のように仮定する．この仮定の下で実行サイクル数が 1000M サイクルのタスクを制約時間 25 秒以内に完了しなければならないとき，5.0V で実行した場合と 4.0V で実行した場合とでは，両方とも制約時間を満たしているが，4.0V で実行した方がエネルギー消費が少なくなる．

プロセッサに要求される性能はアプリケーションによって様々であり，この性能要求の違いを利用した電力削減を可能にするのが可変電源電圧プロセッサである．プロセッサの電源電圧の制御はアプリケーションやオペレーティングシステムによって行なわれる．

[†] 九州大学 大学院システム情報科学研究科 情報工学専攻
Department of Computer Science and Communication Engineering, Kyushu University

Supply Voltage	4.0V	5.0V
Clock Frequency	40MHz	50MHz
Energy/Cycle	25nJ	40nJ

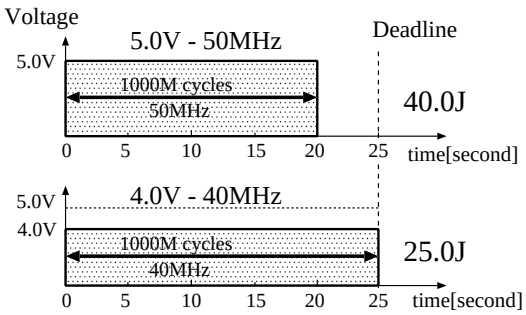

図 1　電力削減の例
Fig. 1　An example of energy reduction

よって，アプリケーションがリアルタイム制約を満たすように電源電圧の制御を行なうためのコンパイラ技術や，オペレーティングシステム技術の確立が重要となる．文献 [3] において，コンパイル時にエネルギー消費を最小にする時間的な電圧割り当てを決定し，オブジェクトコード中に電圧制御命令を挿入する手法が提案されている．

可変電源電圧プロセッサを利用した場合，シングルタスク環境では制約時間を一杯に使うことにより電圧を下げることができる．しかし，マルチタスク環境の場合，リアルタイム性を損なうことなくシステムの消費エネルギーを最小化するように各タスクの電圧を決定することが重要である． そこで本稿では，各タスクに CPU 時間だけでなく電圧も割り当てるタスクスケジューリング手法を提案する． タスクスケジューリングとは，オペレーティングシステムの処理のうち重要な処理の一つであり，各タスクを決められた時間内に処理するために，タスクの実行順序を決定する処理である．本手法を使うことにより，制約時間を満たす範囲でシステムの消費エネルギーを最小化することができる．

文献 [1] において Hong らは，可変電源電圧プロセッサをベースとしたシステム LSI の設計手法を提案している．更に，プリエンプションを許さないハードリアルタイムシステムに対して，消費エネルギーを最小化する経験的なオフライン方式のスケジューリングアルゴリズムも提案している．また，文献 [2] において，Hong らは，可変電源電圧プロセッサ上での非周期タスクで構成されたプリエンプションを許すシステムに対して，電源電圧を動的に割当てるスケジューリングアルゴリズムを提案している．しかし，全てのタスクが最大電圧で実行されると仮定したときに，全てのタスクが時間制約を満たすようなスケジュールが存在するタスクセットを，文献 [2] で提案されているアルゴリズムによってスケジュールしたときに，リアルタイム性が保証できない場合がある．本稿では，全てのタスクが最大電圧で実行されると仮定したときに，実現可能なスケジュールが存在するならば，各タスクに割当てる電圧を低減させても，必ずリアルタイム性を保証できるような電圧割当てアルゴリズムを提案する．

本稿の構成は以下の通りである．第 2 章で電源電圧と遅延時間の関係について述べる．第 3 章では，システムの消費エネルギーの最小化を目的とするスケジューリング手法を提案する．第 4 章では，本手法を使うことによる消費エネルギーの削減効果をシミュレーション実験により評価し，最後に第 5 章でまとめと今後の課題を述べる．

2. 電源電圧と遅延時間の関係

CMOS（Complementary Metal Oxide Semiconductor）トランジスタの飽和領域において，出力電圧が 0 から V_{dd} まで遷移する時間 τ は式 1 によって近似できる [12].

$$\tau = \frac{K \cdot V_{dd}}{(V_{dd} - V_{th})^{\alpha}} \tag{1}$$

V_{th}：しきい値電圧

α は電子の移動度の低下を表すパラメータで，短チャネル化により α の値は 2 から 1 に近づきつつある．

式 1 は，しきい値電圧より大きい範囲で電源電圧を変化させるとき，遅延時間は電源電圧にほぼ反比例することを示している．さらに，CMOS 論理ゲートがスイッチする時に消費される電力は，式 2 で表される [13].

$$P = C \cdot f \cdot {V_{dd}}^2 \tag{2}$$

f：単位時間あたりのゲートのスイッチング回数
C：素子の寄生容量

式 2 は電源電圧の低減による消費電力の削減効果が大きいことを示している．一方，単純に電源電圧を下げると，CMOS 論理回路の遅延時間が増加して動作速度は遅くなるが，電圧に合わせてトランジスタのしきい値電圧 V_{th} も下げると動作速度は低下しないことも式 1 から読み取れる．さらに，電源電圧を変更するとき，CMOS 論理回路の消費電力と動作速度との間にはトレードオフの関係が成り立つ．

3. 低消費電力化を目的とするスケジューリング手法

3.1 対象とするシステム

リアルタイムシステムは，通常アプリケーションとそれをコントロールするリアルタイムオペレーティングシステムから構成される． リアルタイムシステムにおいて，リアルタイム処理を達成させるためには，プログラムを複数のタスクに分割して，それら 1 つ 1 つの処理があらかじめ予測できたり，既知でなければならない．スケジューラの仕事は，外部イベントが検出されると，デッドライン時刻に間に合うようにこれらのタスクをスケジューリングすることである．

本手法では，シングルプロセッサシステムを対象とし，コアプロセッサとして電源電圧を動的に変更することのできる可変電源電圧プロセッサを使用する．可変電源電圧プロセッサは有限の動作モードを持つものとする．つまり，電源電圧は連続的に変化するのではなく，離散的に変化する．また，プロセッサの動作モードを変更する電圧制御命令は OS（スケジューラ）のみ使用可能であり，ユーザアプリケーションは利用できないものとする．つまり，プロセッサの動作モードはタスクの実行が切り替わるとき，すなわち，タスクスイッチのときのみ変更されることを意味する．

本手法においてスケジューラの行なう主な仕事は次の通りである．

- 各タスクに対しプロセッサで処理される時間を割り当てる．この操作を順序割当てと呼ぶ．
- 各タスクに対し実行されるときのプロセッサの電源電圧を割り当てる．この操作を電圧割り当てと呼ぶ．

電圧を変化させるには, 通常 $100\mu s$ の時間を要する．本手法では電圧切り替えの間はプログラムが停止していると仮定する．また，本手法では数千 M サイクルの比較的大きいタスクを対象とするため, 電圧切り替えに要する時間は十分無視できるものとする．

3.2 定　　義

一般にリアルタイムタスク J_i は以下の要素で特徴づけられる [10]．

- a_i：到着時刻．タスク J_i を実行するための条件がそろった時刻．
- O_i: 最悪実行時間．割り込みなしでタスク J_i が実行されるときのプロセッサの必要とする時間の最大値．
- d_i: デッドライン時刻．システムの損害を避けるためにタスク J_i が完了していなければならない時刻．
- s_i: 実行開始時刻．タスク J_i が実行を開始した時刻．
- e_i：実行終了時刻．タスク J_i が実行を完了した時刻．
- L_i: 余裕時間．終了時刻からデッドライン時刻までの余りの時間．

$$L_i = d_i - e_i$$

本稿では，各タスクの消費するエネルギーも考慮するため，以上の要素に以下の 5 つの要素を加えてタスクのモデルとする．

- X_i：最悪実行サイクル数．
- F_i：動作周波数．O_i, X_i, および F_i は以下の関係がある．

$$O_i = \frac{X_i}{F_i}$$

- V_i：電源電圧．
- C_i：平均負荷容量．

$$C_i = \sum_{k=1}^{G} CL_k \cdot Swit_k$$

ここで，G は プロセッサのゲート数，CL_k は各ゲートの負荷容量，$Swit_k$ は各ゲートの 1 サイクル当りの平均スイッチング回数である．

- E_i：最悪消費エネルギー．E_i, C_i, X_i, および V_i は以下の関係がある．

$$E_i = C_i \cdot X_i \cdot {V_i}^2$$

上記の要素のうちの幾つかを図 2 に示す．

各タスクが実行されたときの動作周波数 F_i と，電源電圧 V_i は，そのタスクの実行が終了するまで不変であると仮定する． しかし，プリエンプションによっ

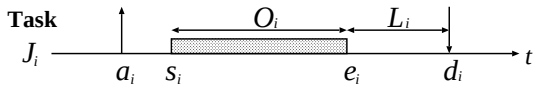

図 2　リアルタイムタスクのモデル
Fig. 2　Typical parameters of a real-time task

て実行が中断され実行が再開されるときに，プリエンプション前とは異なる動作周波数や電源電圧が割り当てられる可能性はある．

また，可変電源電圧プロセッサのモデルを以下に示す．

- (v_j, f_j)：プロセッサの動作モード．プロセッサの電源電圧を v_j とすると，プロセッサは クロック周波数 f_j で動作する．
- $m = |\{(v_j, f_j)\}|$：動作モードの数．
- $V_{max} = \max(v_j)$：最大電圧

例えば タスク J_3 が プロセッサの動作モード (v_2, f_2) で実行されたとき

$$V_3 = v_2, \qquad F_3 = f_2$$

と表される．

タスクスケジューリングは, 動的もしくは静的に行なうことができる．前者は実行時にスケジューリングを決定し, 後者は起動前に決定する．動的にスケジューリングを行なう場合, スケジューリングに必要とする時間は直接的な処理からすれば余計な時間である．このオーバーヘッド時間を出来るだけ短くすることが, 動的なスケジューリングに求められる．

3.3　静的な電圧割り当て

静的な電圧割り当てを行なうためには，静的に順序割り当てが行なわれていなければならない．リアルタイム制約を満たす順序割り当てに関するアルゴリズムは既にいくつか提案されている．Earliest Deadline First(EDF) [5], [9]，Bratley's algorithm [6]，および Latest Deadline First(LDF) [4] 等である．いずれのアルゴリズムも実現可能な順序割り当てが存在するならば，スケジュール可能なアルゴリズムである．以下, 各アルゴリズムに関して簡単に説明する．EDF は非周期的に起動されるタスクについてデッドラインの早い順に実行するアルゴリズムである．また，Bratley のアルゴリズムはプリエンプト禁止条件下で実行可能なスケジュールを生成するアルゴリズムである．最後に LDF はタスク間に実行順序の指定がある場合に実行可能なスケジュールを生成するアルゴリズムである．

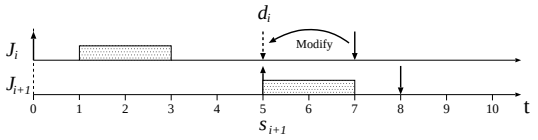

図 3　変換規則 1 の例
Fig. 3　An example of modification rule 1

これらのアルゴリズムにより，各タスクに CPU 時間を割り当てる．このとき，全てのタスクが V_{max} で実行されると仮定して CPU 時間を割り当てる．その後，システムの消費エネルギーを最小化するように各タスクに電圧を割り当てる．電圧割り当てに求められる要件は，順序割り当てのときに実現可能なスケジュールが存在するならば，リアルタイム性を損なうことなく電圧割り当てを行なわなければならないことである．

本節では静的な順序割り当てが施されたタスクセットに対して電圧を静的に割り当てる問題について定式化を行なう．

3.3.1　諸　準　備

問題を簡単にするために，静的に CPU 時間が割り当てられたタスクセットに対して以下の処理を行ない，タスクセットを分割する．

変換規則 1

図 3 のように，どのタスクも実行されない時間が存在する場合，この時間をアイドル時間と呼び，アイドル時間を区切りにタスクセットを複数のサブセットに分割する．つまり，タスク J_i とタスク J_{i+1} の間にアイドル時間が存在するとき，タスク J_i とタスク J_{i+1} は別のサブセットに分類される．このとき，タスク J_i は J_i を含むタスクのサブセット（$\mathcal{J}_i$ とする）の最後に実行されるタスクであり，タスク J_{i+1} は J_{i+1} を含むタスクのサブセットの最初に実行されるタスクである．更に，$J_k \in \mathcal{J}_i$ かつ $d_k > s_{i+1}$ が成り立つタスク J_k に対して，J_k のデッドライン時刻 d_k を次のように変換する．

$$d_k = s_{i+1} (= a_{i+1})$$

変換規則 2

あるタスクが別のタスクによってプリエンプトされ，図 4 の J_i ようにタスクがいくつかのサブタスクに分割された場合，これらのサブタスクを別のタスク

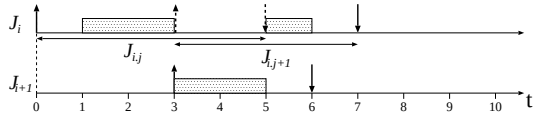

図 4　変換規則 2 の例
Fig. 4　An example of modification rule 2

として扱う．つまり，タスク J_i が $n-1$ 回プリエンプトされて，n 個のサブタスクに分割されたとき，これらのサブタスクを $J_{i.1}, \ldots, J_{i.n}$ と定義する．このとき，タスク $J_{i.j}$ の到着時刻，および，デッドライン時刻は次のように定義される．

$$a_{i.1} = a_i \qquad d_{i.n} = d_i$$
$$a_{i.j} = e_{i.j-1} \qquad d_{i.j} = s_{i.j+1}$$

さらに，最悪実行サイクル数 $X_{i.j}$ は次のように定義される．

$$X_{i.j} = \frac{e_{i.j} - s_{i.j}}{\sum_{k=1}^{n}(e_{i.k} - s_{i.k})} \cdot X_i$$

3.3.2　静的な電圧割り当て問題の定式化

静的な電圧割り当て問題は実行順序に関して既にスケジューリングされているタスクセット $\{J_1, \ldots, J_n\}$（添字の小さい順に実行される），及び，各タスクごとに最大実行サイクル数 X_i，デッドライン時刻 d_i，平均負荷容量 C_i が与えられ，また，プロセッサの動作モードの集合 $\{(v_j, f_j) \mid j = 1, \ldots, m\}$ が与えられたとき，制約条件（式 3）を満たし，消費エネルギー（式 4）を最小化するように関数 $\sigma : \mathbf{N} \to \mathbf{M}$ を定める問題として定式化できる．ここで σ は，$\sigma(i) = j$ とすると $(V_i, F_i) = (v_j, f_j)$ となる関数であり，$\mathbf{N}(\mathbf{M})$ は，1 から $n(m)$ までの整数の集合である．

$$\forall i = 1, \ldots, n \qquad \sum_{k=1}^{i} \frac{X_k}{F_k} \leqq d_i - a_1 \tag{3}$$

$$E = \sum_{i=1}^{n} C_i \cdot X_i \cdot {V_i}^2 \tag{4}$$

この問題を解くことにより消費エネルギーを最小化する最適な電圧を割り当てることができる．

3.3.3　静的な電圧割当てアルゴリズム

図 5 に, 静的な電圧割り当て問題を解くための SS アルゴリズムを示す. SS アルゴリズムでは, 探索木を用いて枝狩りの手法で網羅的にエネルギーの最小となる電圧割り当てを調べる．最悪の場合の計算時間は $O(m^n)$ になる．

SS Algorithm:
- 入力
  - 3.3.1 節の変換規則 1 と変換規則 2 が施されているタスクセット $\{J_1, \ldots, J_n\}$.
  - プロセッサの動作モード $\{(v_j, f_j) \mid j = 1, \ldots, m\}$
- 出力
  - 消費エネルギー E を最小化する最適な電圧割当て.
- 処理
  - 電圧に関する探索木グラフを使う.
  - 検索木 の i 段目のノードは タスク J_{i-1} に対して割当てられる電圧を表す.
  - 制約条件 (式 3) を満たさないパスが見つかればそのパスの検索をそれ以上行なわない.
  - タスク J_n まで到達したパスに対してそれぞれ消費エネルギーを計算し, 消費エネルギーが最小となるパスを見つける.

図 5　SS アルゴリズム
Fig. 5　Pseudo code of SS Algorithm

3.4　動的な電圧割り当て

動的に電圧を割り当てるとき，スケジュール時間は短いほうがよい．我々の提案する動的な電圧割り当ては次に実行されるタスクにだけ電圧を割り当てることを想定している．そのとき，将来実行されるタスクがリアルタイム制約を犯さないように電圧を割り当てなければならない．

ここで，将来実行されるタスクがリアルタイム違反を起こさないために, 次に実行されるタスクが実行に費やせる最大の時間を占有時間と定義する．この占有時間の値が求まれば, スケジューラは占有時間内に終了するような電圧を次のタスクに割り当てるだけで, リアルタイム性を必ず保証し, 低エネルギー化が図れる．また，占有時間が長ければ長いほど低い電圧を割り当てることができ，更なるエネルギーの削減が可能である．つまり, スケジューラの重要な仕事は次に実行されるタスクの占有時間の長さを決定することである．

ここで 2 つのアルゴリズムを提案する．SD アルゴリズムと DD アルゴリズムである．これらのアルゴリズムは次に実行されるタスクの占有時間の長さを決定するアルゴリズムである．それぞれ対象とするシステムが違い，SD アルゴリズムは，全タスクの到着時刻があらかじめ分かっているシステム，DD アルゴリズムは，全タスクの到着時刻があらかじめ分かってい

ないシステムを対象とする．全タスクの到着時刻があらかじめ分かっているシステムとして, 周期タスクのみで構成され, 外部からの割り込みによって起動されるタスクが存在しないシステムがあげられる．逆に外部からの割り込みよって起動されるタスクが存在するシステムでは, 外部からの割り込みがいつ来るのか分からないので, タスクの到着時刻が分からないシステムになる．

図 6，図 7 にそれぞれ SD アルゴリズム，および，DD アルゴリズムを示す．アルゴリズムは両者ともに以下の 3 つのステップで構成される．

（1） CPU Time Allocation

（2） Start Time Assignment

（3） End Time Prediction

以下, 各ステップを簡単に説明する．(1) では, 全てのタスクを V_{max} で実行すると仮定して CPU 時間を割り当てる．この時, 各タスクの実行サイクル数は最悪の場合で考える．さらに, 各タスクの s_i，O_i，L_i の値をそれぞれ $s_i^{V_{max}}$，$O_i^{V_{max}}$，$L_i^{V_{max}}$ とし，(2) と (3) で利用する．(2) では，次に実行されるタスクの占有時間の始点を決定する．この始点は前に実行されたタスクの終了時刻によって変動する．タスクの実際の実行サイクル数は入力データによってバラツキがあり, 最悪の場合より早く終了することの方が多い．つまり前に実行されたタスクが早く終了すればするほど, 次に実行するタスクの開始時刻を早めることが可能である．(3) では, 次に実行されるタスクの占有時間の終点を決定する．終点の決定はリアルタイム性を考慮して慎重に行わなければならない．終点は, (1) における次に実行されるタスクの終了時刻からどれだけ後ろに伸ばせるかが鍵である．

対象とするシステムの違いから, (1),(2),(3) はそれぞれ，SD では静的, 動的, 静的に行ない, DD では全て動的に行なう．

3.4.1 SD アルゴリズム

SD アルゴリズムでは，静的に (1), (3) を行い, 動的に (2) を行なう．(1) では, 既存の順序割り当てアルゴリズムを用いて CPU 時間の割り当てを行なう．図 6 における Input のタスクセットは既に (1) が静的に行なわれた状態である．SD アルゴリズムにおいて, 始点はスケジューラが呼び出された現在の時刻（t）であり，終点は, 対象タスク（J_k）の (1) における終了時刻（$s_k^{V_{max}} + O_k^{V_{max}}$）から, まだ実行されていないタスクの余裕時間の最小値（$\min_{i \geqq k}(L_i^{V_{max}})$）を加えた時刻になる．SD の対象とするシステムでは, 将来実行されるタスクの到着時刻があらかじめ分かっているので, まだ実行されていないタスクの余裕時間があらかじめ計算可能である．そのため，その最小値分だけ後ろにずらしてもリアルタイム性を保証することができる．

SD Algorithm:

- Input:
  - 3.3.1 節の変換規則 2 が施されたタスクセット $\{J_1, \ldots, J_n\}$.
  - プロセッサの動作モード $\{(v_j, f_j) \mid j = 1, \ldots, m\}$
  - 次に実行されるタスク J_k
  - 現時刻 t
- Output:
  - J_k の占有時間 T_{sd}
- Algorithm:

$$T_{sd} = s_k^{V_{max}} + O_k^{V_{max}} + \min_{i \geqq k}(L_i^{V_{max}}) - t$$

図 6　SD アルゴリズム
Fig. 6　Pseudo code of SD Algorithm

［定理 1］ SD アルゴリズムにおいて，あるタスク J_k の占有時間の終点時刻は J_k のデッドライン時刻よりも前である．つまり以下の式が成り立つ．

$$s_k^{V_{max}} + O_k^{V_{max}} + \min_{i \geqq k}(L_i^{V_{max}}) \leqq d_k \qquad (5)$$

［定理 2］ SD アルゴリズムにおいて，ある時刻 t で，タスクの実行が J_k に切り替るとき，タスク J_k が占有時間 T_{sd} 内に終了するようなプロセッサの動作モードが必ず存在する．

定理 1，定理 2 の証明は省略する．以上の定理より，タスクの到着時刻があらかじめ分かっているシステムに対して，SD アルゴリズムによって求まる占有時間内に終了するような最低の電圧を割り当てれば，リアルタイム性は必ず保証され，エネルギーをある程度削減することができる．

3.4.2 DD アルゴリズム

DD アルゴリズムでは (1),(2),(3) を全て動的に行なう．それは，将来実行されるタスクの到着時刻があらかじめ分かっていないからである．(1),(2) は動的に行なうか, 静的に行なうかの違いだけで, SD と DD は同じことを行なっている．違うのは (3) だけである．SD の対象とするシステムでは, まだ実行されていないタスクの余裕時間の最小値分だけ終点を伸ばすことができた．しかし, DD の対象とするシステムでは, 将来実行されるタスクの到着時刻があらかじめ分かっ

DD Algorithm:

- Initial values:
  - レディーキュー $\mathcal{R} := \emptyset$
  - $t_s := 0$
  - $J_{exe} := J_{idle}$（アイドルタスク）
- Input:
  - プロセッサの動作モード $\{(v_j, f_j) \mid j = 1, \ldots, m\}$
  - 今まで実行されていたタスク J_{exe}
  - J_{exe} に割当てられた電圧 V_{exe}
  - 現時刻 t
- Output:
  - 次に実行されるタスク J_{ne}
  - J_{ne} の占有時間 T_{ne}
- Algorithm:

```
if new task arrived then
    J_ar := arrival task
    if Priority(J_ar) > Priority(J_exe) then
        R := {J_exe} ∪ R
        if J_exe ≠ J_idle then
            X_exe := the rest of X_exe
        end if
        t_s := t + O_ar^Vmax
        J_ne := J_ar
        T_ne := t_s − t
    else
        R := {J_ar} ∪ R
        J_ne := J_exe
        Supply voltage is unchnaged
    end if
else if J_exe finished then
    J_hi := R の中の優先度の高いタスク
    R := R − {J_hi}
    t_s := t_s + O_hi^Vmax
    J_ne := J_hi
    T_ne := t_s − t
end if
```

図 7　DD アルゴリズム
Fig. 7　Pseudo code of DD Algorithm

ていないので, 終点を伸ばす行為は危険である．つまり, 終点は (1) における次に実行されるタスクの終了時刻になる．

［定理 3］ DD アルゴリズムにおいて，あるタスク J_k の占有時間の終点時刻は J_k のデッドライン時刻よりも前である．

［定理 4］ DD アルゴリズムにおいて，ある時刻 t で，タスクの実行が J_k に切り替るとき，タスク J_k が占有時間 T_{dd} 内に終了するようなプロセッサの動作モードが必ず存在する．

定理 3, 定理 4 の証明は省略する．以上より，タスクの到着時刻があらかじめ分かっていないシステムに対して，DD アルゴリズムによって求まる占有時間内に終了するような最低の電圧を割り当てれば，リアルタイム性は必ず保証され，エネルギーをある程度削減することができる．

表 1　プロセッサの動作モード
Table 1　Processor mode

Supply Voltage	Clock Frequency
5.0V	50MHz
4.0V	44MHz
2.5V	32MHz

表 2　タスクセット
Table 2　Task set

Task	X_i[cycle]	C_i[F]
J_1	10000000	10
J_2	8000000	15
J_3	15000000	20
J_4	5000000	5
J_5	4000000	30

4. シミュレーション実験

4.1 実験の概要

表 1 で表される 3 種類の動作モードを持つ可変電源電圧プロセッサ上で表 2 で表されるタスクセットを実行させるシミュレーション実験を行なう． 表 2 のタスクセットを，あるシナリオで実行させたときの消費エネルギーを以下の 4 つの場合において求める．

Normal:　全てのタスクを電圧 V_{max} で実行．

SS:　静的に順序割当てを行ない，そのタスクセットに対して静的に割り当てた電圧で実行．

SD:　静的に順序割当てを行ない，そのタスクセットに対して，SD アルゴリズムによって動的に割り当てられた電圧で実行．

DD:　DD アルゴリズムによって動的に割り当てられた電圧で実行．

今回の実験で用いたシナリオは，表 3，表 4 で表されるシナリオである．シナリオのうち，到着時刻（a_i）と，デッドライン時刻（d_i）は SS，SD の場合において前もって分かっているものとし，DD の場合においては動的にしか分からないものとする．また，実行サイクル数はそれぞれのタスクを実行したときの実際のサイクル数であり，全ての場合において動的にしか分からないものである．シナリオ 1 とシナリオ 2 の違いは，シナリオ 2 の方がデッドライン時刻に余裕があるだけで，後は全て同じである．表 2 のタスクセットをこれらのシナリオで実行した場合，以下のような順序で実行される．

$$J_1 \rightarrow J_2 \rightarrow J_3 \rightarrow J_4 \rightarrow J_3 \rightarrow J_5$$

表 3　シナリオ 1
Table 3　Scenario 1

Task	a_i[sec]	d_i[sec]	Execution Cycle
J_1	0	0.2	9300000
J_2	0	0.4	7000000
J_3	0	0.8	14000000
J_4	0.4	0.5	3000000
J_5	0.5	1.2	3000000

表 4　シナリオ 2
Table 4　Scenario 2

Task	a_i[sec]	d_i[sec]	Execution Cycle
J_1	0	0.3	9300000
J_2	0	0.5	7000000
J_3	0	0.9	14000000
J_4	0.4	0.7	3000000
J_5	0.5	1.4	3000000

表 5　シナリオ 1 の実験結果
Table 5　Results for scenario 1

Task	Normal	SS	SD	DD
J_1	5.0V	5.0V	5.0V	5.0V
J_2	5.0V	5.0V	5.0V	5.0V
J_3	5.0V	5.0V	2.5V	5.0V
J_4	5.0V	5.0V	5.0V	5.0V
J_3	5.0V	4.0V	4.0V	4.0V
J_5	5.0V	2.5V	2.5V	2.5V
Energy	1665J	1110J	1036J	1130J

表 6　シナリオ 2 の実験結果
Table 6　Results for scenario 2

Task	Normal	SS	SD	DD
J_1	5.0V	5.0V	4.0V	5.0V
J_2	5.0V	5.0V	4.0V	5.0V
J_3	5.0V	4.0V	2.5V	5.0V
J_4	5.0V	4.0V	2.5V	5.0V
J_3	5.0V	5.0V	2.5V	4.0V
J_5	5.0V	4.0V	2.5V	2.5V
Energy	1665J	778J	634J	1130J

J_3 は，優先度の高い J_4 が到着しプリエンプションにより実行が中断され，残りの処理が J_4 の終了後実行されている．

4.2 実験結果

シナリオ 1，シナリオ 2 の消費エネルギーの結果をそれぞれ表 5，表 6 に示す．表ではそれぞれの場合において，各タスクが実行されたときのプロセッサの電源電圧を表している．また，表中のタスクの順序はタスクが実行された順に並べてある．

シナリオ 1 において，消費エネルギーは Normal と比較して SS では 33%，SD では 38%，DD では 32% 削減された．また，シナリオ 2 において，SS では 53%，SD では 62%，DD では 32% 削減された．

4.3 実験の考察

実験結果より以下のことが確認できる．

- SS，SD，DD，は Normal より悪くなることはない．
- SS では，消費エネルギーが最小になる最適な電圧を静的に割り当てることができるが，タスクが最悪実行サイクルより早く終了した場合の時間的余裕を，静的には見積もれないために利用することが出来ないので，それが本当に最適である可能性は低い．
- SS および SD では，電圧を割り当てる際に余裕時間（L_i）も利用できるため，デッドライン時刻に余裕があるほど，エネルギー消費の削減効果は大きい．
- 逆に DD では，タスクの到着が動的にしか分からないので，余裕時間（L_i）を利用することができず，デッドライン時刻に余裕を持たせても，それ以上のエネルギー消費が削減効果は得られない．

シナリオ 1，シナリオ 2 ともに SD が最も良い結果が得られているが，SD がが最良であるという保証はない．全てのタスクが最悪実行サイクル数で終了した場合は，SS の方が最良になる．どちらが最良かは入力データによる．

5. おわりに

本稿では，可変電源電圧プロセッサに対するタスクスケジューリングの際に CPU 時間だけでなく，電源電圧も割り当てることによって，時間制約を満たす範囲でのシステムの消費エネルギーを最小化する手法を提案した．電源電圧を動的に割り当てることによって，入力データ次第でエネルギー消費を大幅に削減することができる． また，SD アルゴリズム，および，DD アルゴリズムによって求めた占有時間内に終了するような最低の電圧が，システム全体の消費エネルギーを最小化する最適な割り当てである保証はないが，占有時間内に終了するような電圧を割り当てれば必ずリアルタイム性を保証することができる．また，SD アルゴリズム，および，DD アルゴリズムは数サイクルの処理であるため，スケジューリングのオーバヘッドはほとんどない．今後は，更にエネルギーを削減するために，あるタスクが占有時間内に終了するための電圧としてどの電圧を割り当てれば良いかを決定するアルゴリズムについて研究を行なっていきたい．

謝辞

本研究は，一部 STARC プロジェクト番号：PJ-

No.987「コアプロセッサベースシステム LSI の最適化設計技術に関する研究」の支援による．

文　献

[1] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. Srivastava. “Power Optimization of Variable Voltage Core-based Systems”. In *Design Automation Conference*, 1998.

[2] I. Hong, M. Potkonjak, and M. B. Srivastava. “On-Line Scheduling of Hard Real-Time Tasks on Variable Voltage Processor”. In *Proc. of ICCAD-98*, pages 653–656, 1998.

[3] T. Ishihara and H. Yasuura. “Voltage Scheduling Problem for Dynamically Variable Voltage Processors”. In *Proc. of International Symposium on Low Power Electronics and Design(ISPLED’98)*, pages 197–202, August 1998.

[4] E. Lawler. “Optimal Sequencing of a Single Machine Subject to Precedence Constraints”. Managements Science, 19, 1973.

[5] C. L. Liu and J. Layland. “Scheduling Algorithms for Multiprogramming in a Hard Real-time Environment”. *Journal of the ACM*, 20(1):46–61, 1973.

[6] P. Bratley, M. Florian, and P. Robillard. “Scheduling with Earliest Start and Due Date Constraints”. Naval Research Logistics Quarterly, 18(4), 1971.

[7] T. Ishihara and H. Yasuura. “Optimization of Supply Voltage Assignment for Power Reduction on Processor-Based Systems”. In *Proc. of SASIMI ’97*, pages 51–58, 1997.

[8] T. Ishihara and H. Yasuura. “Programmable Power Management Architecture for Power Reduction”. *IEICE Transaction on Electronics*, E81-C(9), September 1998.

[9] W. Horn. “Some Simple Scheduling Algorithms”. *Naval Research Logistics Quarterly*, 21, 1974.

[10] Giorgio C. Buttazzo. *“HARD REAL-TIME COMPUTING SYSTEM”*. Kluwer Academic Publishers, 1985.

[11] Vadim Gutnik and Anantha P. Chandrakasan. “Embedded Power Supply for Low-Power DSP”. *IEEE Trans. VLSI Systems*, Vol 5, pages 425–435, Dec 1997.

[12] 菅野 卓雄 監修・飯塚 哲哉 編. *CMOS* **超** *LSI* **の設計**. 培風館, 1996.

[13] 榎本 忠儀. *CMOS* **集積回路**. 培風館, 1996.

[14] T. Okuma, T. Ishihara, and H. Yasuura . “Real-Time Task Scheduling for a Variable Voltage Processor”. *Proc. of ISSS-99*, pages 25–29 1999.

（平成 5 年 10 月 25 日受付，8 年 9 月 27 日再受付）

大隈　孝憲

昭和 50 年生．平成 9 年九州大学大学工学部情報工学科卒．平成 11 年九州大学大学院システム情報科学研究科修士課程修了．同年九州大学大学院システム情報科学研究科博士課程進学．低電力システム設計手法の研究に従事．システム LSI のためのソフトウェア技術の研究に従事．情報処理学会の会員．日本学術振興会特別研究員．

石原　　亨　（学生員）

昭和 48 年生．平成 7 年九州大学大学工学部情報工学科卒．平成 9 年九州大学大学院システム情報科学研究科修士課程修了．同年九州大学大学院システム情報科学研究科博士課程進学．低電力システム設計手法の研究に従事．平成 10 年情報処理学会九州支部奨励賞を受賞．電子情報通信学会，情報処理学会の会員．日本学術振興会特別研究員．

安浦　寛人　（正員）

昭和 51 年京都大学工学部情報工学科卒．昭和 53 年京都大学工学研究科修士課程（情報工学専攻）修了．昭和 55 年より京都大学工学部助手．京都大学工学部電子工学科助教授を経て，平成 3 年より九州大学大学院総合理工学研究科情報システム学専攻教授．平成 8 年より九州大学大学院システム情報科学研究科情報工学専攻教授．昭和 60 年 6 月 9 日より昭和 61 年 3 月 31 日米国カリフォルニア大学バークレイ校客員研究員．昭和 64 年 1 月より平成 4 年 10 月京都高度技術研究所非常勤研究員．VLSI システムの設計手法と CAD の研究およびハードウェアアルゴリズムの研究に従事．昭和 57 年電子通信学会学術奨励賞，昭和 63 年および平成 6 年電子情報通信学会論文賞，平成 4 年情報処理学会論文賞，平成 5 年情報処理学会坂井記念特別賞および Best Author 賞をそれぞれ受賞．1998 年 ICCAD-98(計算機援用設計に関する国際会議)の大会委員長．電子情報通信学会，情報処理学会，IEEE，ACM，EATCS などの会員．九州システム情報技術研究所非常勤研究室長を兼務．