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

九州大学 大学院 システム情報科学研究科 情報工学専攻
〒816-8580 福岡県春日市春日公園 6-1
E-mail:{okuma, ishihara, yasuura}@c.csce.kyushu-u.ac.jp

あらまし:
動的に電源電圧を変化させ，その性能と消費電力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，システムの消費電力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電力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

Real-Time Task Scheduling for Variable Voltage Processor

Takanori OKUMA　　Tohru ISHIHARA　　Hiroto YASUURA

Department of Computer Science and Communication Engineering, Kyushu University
6–1 Kasuga-koen, Kasuga, 816–8580, Japan
E-mail: {okuma, ishihara, yasuura}@c.csce.kyushu-u.ac.jp

Abstract:
This paper presents a real-time task scheduling technique for a variable voltage processor which can vary its supply voltage dynamically. On the variable voltage processor, task processing with lower supply voltage is able to reduce power consumption dramatically. But, lower voltage may violate time constraints of the real-time process. In this paper, we propose techniques by which CPU time and supply voltage are simultaneously assigned each task to minimize power consumption. Experimental results demonstrate effectiveness of the proposed technique.

Key Words: Low Power Design, Variable Voltage Processor, Real-Time Scheduling

1 はじめに

微細加工技術の進歩により, システムの全機能を 1 チップで実現するシステム LSI の製造が可能となり, 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が高まっている.

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, 低消費電力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電力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, 高速化と低電力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.

このプロセッサにより, 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 サイクル数のタスクを制約時間の 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っている.

可変電源電圧プロセッサを用いた場合, 制約時間を満たしつつ, どのような電圧を実行中のタスクに割り当てるかが問題となる. そこで本稿では, CPU 時間だけでなく電圧も割当てるタスクスケジューリング手法を提案する. 本手法により, リアルタイム制約を満たし, かつ, システムの消費電力, および, 消費エネルギーを最小化することができる.

電源電圧	4.0V	5.0V
Clock 周波数	40MHz	50MHz
Energy/Cycle	25nJ	40nJ

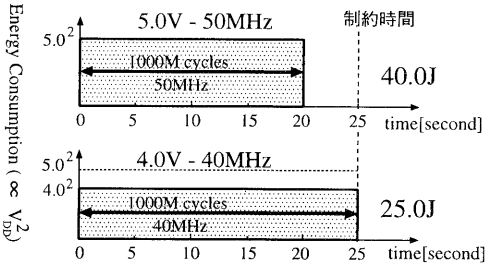

図 1: 電力削減の例

本稿の構成は以下の通りである. 第 2 章において, 提案するタスクスケジューリング手法の概要を述べ, 第 3 章において, システムの消費電力を最小化する電圧割当てアルゴリズムを提案する. そして, 第 4 章において, その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 タスクスケジューリング

2.1 システムの仮定

本稿で仮定するシステムについて以下に示す.

- コアプロセッサとして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.
- 電圧制御命令はスケジューラのみ使用でき, ユーザアプリケーションは使用できない.
- 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のみ変更される.

また, スケジューラが行なう仕事を以下に示す.

- 各タスクにプロセッサで処理される時間を割当てる. この操作を順序割当てと呼ぶことにする.
- 各タスクに実行されるときのプロセッサの電源電圧を割当てる. この操作を電圧割当てと呼ぶことにする.
- 順序割当てを行なった後に, 電圧割当てを行なう.
- 実行タスクの切替え時に電圧制御命令を実行し, 次に実行されるタスクに割り当てられた電圧にプロセッサの動作モードを切り替える.

2.2 スケジューリング手法

リアルタイムシステムでは, 通常, もっとも優先度の高いタスクを選択するという, 優先制御方式が採られる. 最初に ready 状態になったタスクを選択するというラウンドロビン方式とか, 一定時間が経過すると強制的に running 状態から ready 状態に戻し, 選択しなおす (リスケジューリングする) タイムシェアリング方式などはあまり採られない. また, スケジューリングアルゴリズムは, 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.

静的な順序割当てを行なうためには, 全てのタスクの到着時刻があらかじめ分かっていなければならない. 到着時刻があらかじめ分かっている場合, 以下の 3 つのシステムが考えられる.

(S-S1) 全てのタスクの到着時刻が同じシステム

(S-S2) 各タスクの到着時刻が異なり, プリエンプションを許さないシステム

(S-S3) 各タスクの到着時刻が異なり, プリエンプションを許すシステム

プリエンプションとは, 優先度の低いタスクが実行中であっても, 高い優先度のタスクが到着すれば, それを先に実行し優先度の高いタスクの実行が終わってから中断した低い優先度のタスクを実行する方式である.

また, 到着時刻があらかじめ分かっていない場合, 動的に順序割当てを行なうしかない. この場合, リアルタイムシステムに適しているシステムとして, 以下のものが挙げられる.

(D-S1) プリエンプションを許すシステム

リアルタイム制約を満たす順序割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. 静的に行なうアルゴリズムとして, (S-S1) のシステムに対しては EDD(Earliest Due Date) アルゴリズム [3], また, (S-S2), (S-S3) のシステムに対しては, それぞれ Tree Search アルゴリズム [5], EDF(Earliest Deadline First) アルゴリズム [4] である. 更に, EDF アルゴリズムは, 動的に行なうことで, (D-S1) のシステムに対しても適用できる. いずれのアルゴリズムも実現可能な順序割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.

上記のアルゴリズムによって静的または動的に順序づけられたタスクセットに対して, システムの消費エネルギーを最小化するように電圧割当てを行なう. 電圧割当てに求められる要件は, 順序づけられたタスクセットをプロセッサの動作モードとして最高電圧の状態で実行させたときにリアルタイム性が保証されているならば, リアルタイム性を損なうことなく電圧割当てを行なわなければならないことである.

3 電圧割当てアルゴリズム

3.1 モデル化

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.

到着時刻 a_i : タスク J_i を実行するための条件がそろった時刻.

最悪実行時間 O_i : 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.

デッドライン時刻 d_i : 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.

開始時刻 s_i : タスク J_i が実行を開始した時刻.

終了時刻 e_i : タスク J_i が実行を完了した時刻.

余裕時間 L_i : $L_i = d_i - e_i$, 終了時刻からデッドライン時刻までの余りの時間

本稿では, 各タスクの消費する電力も考慮するため, 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.

最悪実行サイクル数 X_i : タスク J_i の動的な実行サイクル数の最大値.

動作周波数 F_i : タスク J_i が実行されたときのプロセッサのクロック周波数. O_i と X_i と F_i には以下の関係がある.

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

電源電圧 V_i : タスク J_i が実行されたときのプロセッサに供給された電源電圧.

平均負荷容量 C_i : タスク J_i の平均負荷容量. ゲートの負荷容量, 1サイクルあたりのゲートのスイッチング回数, および, 実行サイクル数の積で表される.

消費エネルギー E_i : タスク J_i が実行されたときに消費されたエネルギー. E_i と C_i と V_i には以下の関係がある.

$$E_i = C_i \cdot V_i^2$$

上記の要素のうちの幾つかを図2に示す.

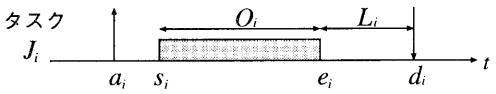

図 2: リアルタイムタスクのモデル

また, 可変電源電圧プロセッサのモデルを以下に示す.

動作モード (v_j, f_j) : 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.

動作モードの数 m : $m = |\{(v_j, f_j)\}|$

最大電圧 V_{max} : 動作モードの電圧のなかで最大な電圧. $V_{max} = \max(v_j)$

3.2 静的な電圧割当て

静的に電圧を割当てるためには, 静的に順序割当てが行なわれていなければならない. 本節では静的な順序割当てが施されたタスクセットに対して電圧を静的に割当てる問題について定式化を行なう.

3.2.1 定式化のための諸準備

問題を簡単にするために, 静的に順序割当てが施されたタスクセットに対して次のような処理を行なう.

(S-S2) や (S-S3) のシステムに対して静的に順序割当てを行なったとき, 図3の $3 \leq t \leq 5$ 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がある. この場合, この空白の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. このとき, 各サブセットの最後のタスクのデッドライン時刻を次のサブセットの最初のタスクの開始時刻に合わせる. 図3の例では J_2 のデッドライン時刻を $t = 7$ から $t = 5$ に変更する.

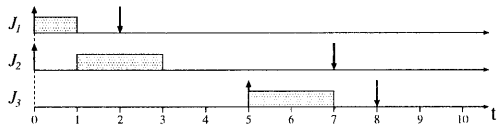

図 3: (S-S2) に対するスケジューリング例

また, (S-S3) のシステムに対して静的に順序割当てを行なったとき, 図4の J_2 のように, プリエンプションにより処理が中断されて, タスクが分割されたようになる場合がある. この場合, この分割されたものを別のタスクとして扱うことにする. このとき, それぞれのタスクの到着時刻とデッドライン時刻を次の例のようにする. 図4の例では, J_2 の前半を J_{2a}, 後半を J_{2b} とすると, J_{2a} のデッドライン時刻を J_{2b} の開始時刻 $(t = 5)$ にし, J_{2b} の到着時刻を J_{2a} の終了時刻 $(t = 3)$ にする.

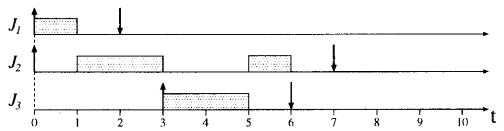

図 4: (S-S3) に対するスケジューリング例

3.2.2 静的な電圧割当て問題の定式化

静的な電圧割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 1) を満たし, 消費エネルギー (式 2) 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.

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

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

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 動的な電圧割当て

図 5 は, FFT プログラムを 100 種類の異なる入力データで実行させたときの, それぞれの実行サイクル数をグラフ化したものである. この図から分かるように, あるタスクの実際の実行サイクル数は, 入力データによりまちまちであり, 最悪実行サイクル数以下で終了することが多い. この場合, 実行されたタスクの終了時刻から最悪実行時刻までの時間的な余裕が生じる. 本稿で提案する動的な電圧割当てアルゴリズムでは, この時間的な余裕を利用して消費エネルギーの最小化を図っている.

動的に電圧を割当てる場合, 以下の 2 通りを考えることができる.

(a) 静的に順序割当てが施されたタスクセットに対して電圧を割り当てる.

(b) 動的に順序割当てを行ないながら同時に電圧も割当てる.

どちらの場合も, 動的なスケジューリングの際の時間的なオーバヘッドを考慮して, 次に実行されるタスクをどのような電圧で実行させたら良いかだけを考えることにより, スケジューリングの処理の負荷を軽くする.

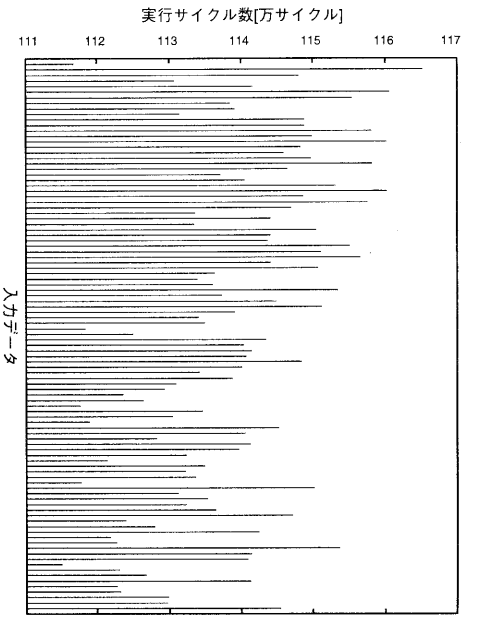

図 5: FFT の実行サイクル数

次に実行されるタスクに電圧を割当てるとき, (b) の場合では, 後に到着し実行されるタスクの予測がつかないので, 次に実行されるタスクのデッドライン時刻までの時間をいっぱいに使って, 出来るだけ低い電圧を割当てて良いかどうかの判断をつけるのは難しい. しかし, プロセッサの動作モードを最高電圧の状態に固定し, 全てのタスクが最悪実行サイクル数かけて終了した場合において, リアルタイム性が保証されている (電圧割当てアルゴリズムがリアルタイム性を保証するための前提条件) ので, その状態における次に実行されるタスクの終了時刻までは, 後で実行させるタスクのことを気にすることなく, プロセッサを占有できる. 一方, (a) の場合では, 後で実行されるタスクの予測がついてるため, その余裕時間も利用することによって, (b) の場合と比べて比較的柔軟に低い電圧を割当てることができる.

3.3.1 場合 (a) における電圧割当てアルゴリズム

図 6 に, (a) の場合における, 動的な電圧割当てアルゴリズムを示す. 図中の $s_k|_{V_{max}}$, $O_k|_{V_{max}}$, $L_i|_{V_{max}}$

- 前準備
  - 静的な順序割当てが施されたタスクセットに対して, 3.2.1 節と同様の処理を行なう.
  - 添字の小さい順に実行されるタスクセット $\{J_1, \ldots, J_n\}$
- 動的な電圧割当て処理
  - ある時刻 t において, タスクの実行が J_k に切り替わるとき, J_k に対して, J_k の処理が以下の時間内に終わるような最低の電圧を割り当てる.

$$s_k|_{V_{max}} - \max(t, a_k) + O_k|_{V_{max}} + \min_{i \geq k}(L_i|_{V_{max}})$$

図 6: (a) の場合での電圧割当てアルゴリズム

は, それぞれ, プロセッサの動作モードを最高電圧の状態に固定し, 全てのタスクが最悪実行サイクル数で終了した場合の各タスクの開始時刻, 実行時間 (最悪実行時間), 余裕時間である. いずれも, 静的な順序割当ての際に, 前もって見積もることができる.

$$s_k|_{V_{max}} - \max(t, a_k)$$

は, タスクが最悪実行時間よりも早く終了した場合の余った時間を表すと同時に, 次に実行するタスクがまだ到着していないなら, その到着時刻まで何もしないことを表している. また,

$$\min_{i \geq k}(L_i|_{V_{max}})$$

は, 後で実行されるタスクの余裕時間を表している.

3.3.2 場合 (b) における電圧割当てアルゴリズム

スケジューラが起動するのは, タスクが到着したときと, 実行タスクが終了したときである. タスクが到着したとき, スケジューラはそのタスクの優先度を現在実行中のタスクの優先度と比較し, 大きければ今到着したタスクに電圧を割り当てる. そして, そのタスクに実行を切り替え, 今割り当てた電圧へプロセッサのどー差モードを切り替えると同時に, 今まで実行していたタスクをレディキューへと挿入する. もし優先度が小さいなら, 今到着したタスクをレディキューへ

- 初期値
  $t_s := 0$, $\mathcal{R} := \emptyset$
  $\mathcal{R}$: レディキュー
- スケジューラの処理
  J_e : 今まで実行していたタスク
  t : 現在時刻

```
if タスクの到着時 then
    J_a: 到着タスク
    if Priority(J_a) > Priority(J_e) then
        R := J_e ∪ R
        X_e := 残りの X_e
        t_c := t_s − t
        t_s := t + O_a|_{V_max}
        t_c を退避
        実行/電圧を J_a/V_max に切り替える.
    else
        R := J_a ∪ R
        J_e の実行を再開する.
    end if
else if 実行タスクの終了時 then
    J_i: R の中で 1 番優先度の高いタスク
    R := R − {J_i}
    if J_i を始めて実行する then
        t_s := t_s + O_i|_{V_max}
    else
        t_s := t_s + 復帰した t_c
    end if
    V_i = minvol(t_s − t, X_i)
    実行/電圧を J_i/V_i に切り替える.
end if
```

図 7: 動的な順序/電圧割当てアルゴリズム

と挿入し, 今まで実行していたタスクの実行を再開させる. また, 実行タスクが終了したとき, スケジューラはレディキューの中から 1 番優先度の高いタスクを取り出す. そして, もしそのタスクを始めて実行するのであれば, 電圧を割り当て, そのタスクへ実行を切り替えると同時に, そのタスクへ割り当てられた電圧へプロセッサの動作モードを切り替える.

詳細な電圧割り当てのアルゴリズムを図 7 に示す. 図中にある $\mathrm{minvol}(t, X)$ は実行サイクル数 X の処理が時間 t 内に終わるような, プロセッサの動作モードにおける最低の電圧を返す関数である. また, t_s はプロセッサの動作モードを最高電圧の状態に固定し, 全

てのタスクが最悪実行サイクル数で終了した場合のタスクの終了時刻を動的に計算したものである.

4 実験

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. なお, 実験の対象とするシステムとしてプリエンプションを許すシステムとする.

Normal: 全てのタスクをプロセッサの動作モードの最高電圧で実行.

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, 3.3.1節の動的な電圧割当てアルゴリズムによって割り当てられた電圧で実行.

DD: 3.3.2節の動的な順序/電圧割当てアルゴリズムによって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分かるものとする. 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.

$$J_1 \rightarrow J_2 \rightarrow J_3 \rightarrow J_4 \rightarrow J_3 \rightarrow J5$$

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.

表 1: 動作モード

電圧	周波数
5.0V	50MHz
4.0V	40MHz
2.5V	25MHz

表 2: タスクセット

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

4.1 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. 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 27%, SD では 38%, DD では 16% 削減された. また, シナリオ2において, SS では 56%, SD では 58%, DD では 16% 削減された.

4.2 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, タスクが早く終了した場合の時間的余裕 (動的にしか分からない) を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薄い.
- 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があっても, エネルギー消費の削減効果はない.

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

表 3: シナリオ 1

Task	a_i[sec]	d_i[sec]	実行サイクル
J_1	0	0.2	9300000
J_2	0	0.4	7000000
J_3	0	0.8	14000000
J_4	0.4	0.7	3000000
J_5	0.5	0.9	3000000

表 4: シナリオ 2

Task	a_i[sec]	d_i[sec]	実行サイクル
J_1	0	0.5	9300000
J_2	0	0.7	7000000
J_3	0	1.4	14000000
J_4	0.4	1.0	3000000
J_5	0.5	1.5	3000000

表 5: シナリオ 1 の実験結果

Task	Normal	SS	SD	DD
J_1	5.0V	5.0V	5.0V	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	5.0V	5.0V	5.0V
J_3	5.0V	5.0V	5.0V	4.0V
J_5	5.0V	2.5V	2.5V	4.0V
Energy	1615J	1176J	999J	1357J

表 6: シナリオ 2 の実験結果

Task	Normal	SS	SD	DD
J_1	5.0V	4.0V	4.0V	5.0V
J_2	5.0V	4.0V	4.0V	5.0V
J_3	5.0V	5.0V	4.0V	5.0V
J_4	5.0V	2.5V	2.5V	5.0V
J_3	5.0V	2.5V	2.5V	4.0V
J_5	5.0V	2.5V	2.5V	4.0V
Energy	1615J	702J	685J	1357J

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電力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. 今後の課題は以下に示す通りである.

- 実用的なベンチマークプログラムを用いた評価
- スケジューリングのオーバーヘッドを考慮にいれた評価.

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.

参考文献

[1] Giorgio C. Buttazzo. *"HARD REAL-TIME COMPUTING SYSTEM"*. Kluwer Academic Publishers, 1985.

[2] 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.

[3] J.R. Jackson. "scheduling a production line to minimize maximum tardiness". Technical report, Management Science Research Project 43, University of California, Los Angels, 1955.

[4] 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.

[5] M. Florian P. Bratley and P. Robillard. "scheduling with earliest start and due date constraints". Naval Research Logistics Quarterly, 18(4), 1971.

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

[7] 石原 亨, 甲斐 康司, 安浦 寛人. "マイクロプロセッサにおけるアーキテクチャレベルの低消費電力化手法". 電子情報通信学会技術研究報告, *VLD96-72*, 1996 年 12 月.