デュアルOSと仮想化DVFSによるミックスドクリティカルシステムの消費エネルギー最小化

小森 工[†1]　増田 豊[†1]　石原 亨[†1]

概要：ミックスドクリティカルシステムのソフトウェアを効率的に開発するため，リアルタイム OS と汎用 OS の両方を利用するデュアル OS プラットフォームが提案されている．しかしながら，そのエネルギー最小化はほとんど検討されていない．本研究では各 OS による独立した省エネルギー制御を実現する仮想化 DVFS，および最悪実行時間と平均実行時間の差を活用する軽量な DVFS アルゴリズムを提案し，デュアル OS プラットフォームの消費エネルギーを効果的に削減する．市販のマイコンボードを用いた評価実験により，信頼系のリアルタイム応答を保証しながらシステム全体の消費エネルギーを 40 % 以上削減できることを示した．

1. はじめに

ミックスドクリティカルシステムとは異なる信頼性要求に基づく機能を同一のプラットフォームで実行する組み込みシステムである．クリティカルなシステムの例として，デッドライン違反や誤動作が人命に直結し得る組み込み機器が挙げられる．このような機器では，高い信頼性とリアルタイム性を両立するシステム制御が不可欠である．一方，情報処理のみを行う組み込みシステムでは，システムの信頼性要求は比較的低い．ミックスドクリティカルシステムの代表例である航空機や自動車においては，高信頼な機器制御と複雑な情報処理を同一のプロセッサで行う需要が高まっている．例えば，カーナビゲーションシステムは車載ネットワークにアクセスする際に高信頼なリアルタイム処理を必要とするうえ，クリティカルでないマルチメディア処理も行う．

プロセッサの消費エネルギーを削減する最も効果的な手法の一つに Dynamic Voltage and Frequency Scaling (DVFS) がある．DVFS ではプロセッサの電源電圧と動作周波数を動的に変更することで，計算性能と消費エネルギーのトレードオフを最適化する．近年では，DVFS をミックスドクリティカルシステムに適用する手法が盛んに研究されている [1–4]．これらの手法は [5] から派生したスケジューリング理論に基づいており，タスクに HI または LO のクリティカリティを与える．プロセッサも HI 及び LO の実行状態を持ち，タスクの最悪実行時間（worst-case execution time, WCET）はそれぞれの実行状態に対して与えられる．一つのスケジューラが全てのタスクを信頼性要求に関わらず扱うため，本研究ではこのアプローチをシングル OS プラットフォームと呼ぶ．

シングル OS プラットフォームは広く研究されている手法だが，OS の選択には困難が伴う．軽量なリアルタイム OS（RTOS）は小規模かつ検証可能であり，高信頼なリアルタイム性を提供する．しかしながら，機能性に乏しい RTOS は大規模かつ複雑なアプリケーションの実装には適していない．一方で，Linux のような汎用 OS（GPOS）は豊富なライブラリやユーティリティを備えているうえ，開発者はオープンソースソフトウェアを無改変で利用することができる．このような利点から，GPOS は優れた開発効率をもたらす．しかし，GPOS の大規模さと絶え間ないソフトウェアアップデートが検証を事実上不可能なものにしており，信頼性保証の観点で大きな課題を持つ．加えて無改変の GPOS は割り込みレイテンシが大きく，動作を予想することも難しいため，リアルタイム性を保証することができない [6–8]．まとめると，RTOS は小規模かつクリティカルなアプリケーションに適しているが大規模かつクリティカルでないものには不適である．また，GPOS は RTOS と対照的な適性を持つ．

もう一つのシングル OS プラットフォームの欠点は HI タスクのデッドラインを保証するため常に高速で動作する必要がある点である．加えてシングル OS プラットフォームはタスクの WCET を考慮して静的にスケジューリングを行うが，実際にはタスクはより短い平均実行時間（average execution time, AET）で終了する．そのためプロセッサの計算性能を低下させ消費エネルギーを削減する時間的余

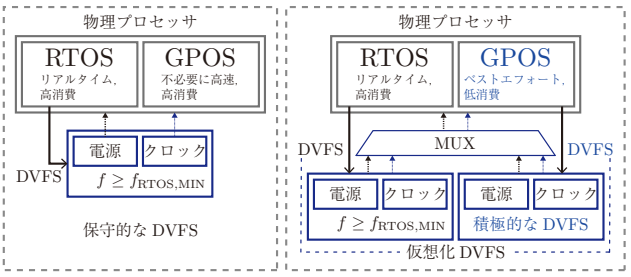
†1　現在，名古屋大学

裕が存在するが，既存手法はこの余裕を十分活用しているとはいえない．

ミックスドクリティカルシステムのソフトウェアを効率的に開発するため，RTOS と GPOS の両方を利用する手法が提案されている [9–14]．本研究ではこのアプローチをデュアル OS プラットフォームと呼ぶ．デュアル OS プラットフォームでは RTOS と GPOS は隔離され，RTOS がクリティカルなタスク，GPOS がクリティカルでないタスクを実行する．GPOS は典型的には RTOS のアイドル処理として動作するため，RTOS のリアルタイム性保証に影響を及ぼさない．簡単のため，以下では RTOS とその上で動作するクリティカルなアプリケーション，GPOS とその上で動作するクリティカルでないアプリケーションをそれぞれ同一視する．

デュアル OS プラットフォームはシングル OS プラットフォームと比較して開発効率の観点で非常に優れているが，そのエネルギー最小化はほとんど検討されていない．DVFS は計算性能の低下と引き換えに消費エネルギーを削減する．電源電圧の制御には一般的に 100 µs 程度かかるが [15]，RTOS がいつ割り込んで動作を開始するか予想することはできない．そのため単純には図 1(a) に示すように RTOS 基準の DVFS を行うことになる．しかし，RTOS はデッドラインを保証してリアルタイム性を担保するために，GPOS と比較して高い計算性能を必要とする．従って，GPOS 側では計算性能に余裕がある場合においても積極的に消費エネルギー削減を図れない問題がある．

本研究ではデュアル OS プラットフォームの消費エネルギーを効果的に削減する仮想化 DVFS を提案する．図 1(b) に提案手法の概要を示す．キーアイデアとして安定した電源とクロック系統をそれぞれの OS 専用に用意し，マルチプレクサで切り替える．提案ハードウェアは OS 切り替え時に計算性能を瞬時に変更できるため，GPOS が非常に低速で動作していた場合であっても RTOS を即座に高速動作させることができる．結果として各 OS の仮想化と同様に，計算性能を独立に制御することができる．本研究ではこの技術を仮想化 DVFS と呼称する．また，本研究では既存手法の限界を克服し，デュアル OS プラットフォームの消費エネルギーを最小化する軽量な DVFS アルゴリズムを合わせて提案する．



(a) 単純な実装　　(b) 提案手法

図 1: 提案する仮想化 DVFS の概要

本研究の貢献は以下の通りである．

- 提案する仮想化 DVFS はデュアル OS プラットフォームのエネルギー効率を向上する．ソフトウェアの開発者は DVFS のためにシングル OS プラットフォームを選択する必要がなくなり，次のようなデュアル OS プラットフォームの利点を享受できる．
  - GPOS の豊富な機能性を利用できる．
  - デュアル OS プラットフォームはタスクの実行モデルを必要としない．簡単のため RTOS の動作周波数を最大動作周波数に固定したとしても，GPOS 側は十分に消費エネルギーを削減することができる．さらに，GPOS 側の WCET 解析を実行する必要がない．
  - デュアル OS プラットフォームは多様なプログラミングパラダイムを利用できる．シングル OS プラットフォームは暗黙に割り込みを禁止し，周期タスクのみを考えている．対して，提案手法は割り込みや複雑なスケジューリングを必要とするソフトウェアにも適用できる．
- 既存のミックスドクリティカルシステムに対する DVFS 手法は電源電圧とクロック周波数が即座に切り替わることを前提にしているが，その実現方法を述べていない．本研究では 3.3 節において電源電圧を 1 µs 以内に切り替えるハードウェアを明示的に提案し，4 節において市販のプロセッサを用いて検証した．
- 本研究はデュアル OS プラットフォームの消費エネルギーを最小化する軽量な DVFS アルゴリズムを提案する．提案アルゴリズムは従来の WCET ベースの手法と異なり，GPOS の AET に則して計算性能を調整する．2.1 節で簡単な例を紹介し，3 節で詳細を述べる．

2. ミックスドクリティカルシステムに対するエネルギー最小化の困難性と解決法

2.1 シングル OS プラットフォーム

[5] から派生したシングル OS プラットフォームに対する DVFS アルゴリズムの欠点を示す．図 2 に示す簡単な

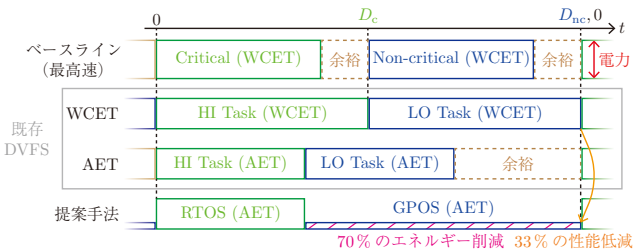


図 2: シングル OS プラットフォームに対するエネルギー最小化の困難性と提案手法．

例を考える．クリティカルタスクとノンクリティカルタスクが一つずつ存在し，共に時刻 $t = 0$ に起動する．D_c と D_{nc} はそれぞれクリティカルタスクとノンクリティカルタスクのデッドラインを示している．また，タスクはどちらも時刻 $t = D_{nc}$ において再度起動し，システムは以下同様の動作を繰り返すとする．プロセッサが最大クロック周波数で動作している際には図 2 上部に示すようにタスクはどちらもデッドライン以内に完了する．シングル OS プラットフォームではクリティカルタスクに HI，ノンクリティカルタスクに LO のクリティカリティを与えてスケジューリングを行う．その後 DVFS アルゴリズムがプロセッサの計算性能を調整し，図 2 中央上に示すようにタスクの WCET をデッドラインと一致させる．計算性能を落とすことで，最速動作時に存在した余裕を消費エネルギーの削減に還元出来る．

しかしながら，図 2 中央下に示す通り，タスクは平均して AET の時間で完了する．AET は WCET よりも短いため，実際には AET と WCET の差分だけ時間的余裕が存在する．一方，既存手法は静的にタスクごとのクロック周波数を決定するため，この余裕を利用して更なるエネルギー削減を行うことができない [16]．ここで，タスクは最速で WCET より 50 % 早く完了する [17] ことから AET が WCET の 80 % であったとする．図 2 下部に示すように，AET と WCET の差による余裕を活用できたとすると，ノンクリティカルタスクの計算性能を 33 % 低下させることができる．DVFS が計算性能の低下に対して 3 乗のスケールで消費エネルギーを削減すると仮定すれば [18]，ノンクリティカルタスクの消費エネルギーは 70 % 削減される．提案手法は RTOS 上で動作するクリティカルタスクのリアルタイム性を保証しつつ，GPOS 上で動作するノンクリティカルタスクの計算性能を AET に合わせて最適化するため，既存の DVFS アルゴリズムと比較して消費エネルギーをより多く削減する．

2.2 デュアル OS プラットフォーム

一般的なデュアル OS プラットフォームは図 3 に示すように RTOS と GPOS を実行する．プロセッサは三つの特権レベル，すなわち，ユーザ・スーパバイザ・ハイパバイザモードを持つ．仮想マシンモニタ（Virtual Machine Monitor, VMM）がハイパバイザモードで動作し，仮想 CPU コアを管理する．通常通り，GPOS はスーパバイザモードで動作することで，不具合を含む，あるいは悪意あるプログラムからカーネルをプログラムから保護し，アプリケーションはユーザモードで動作する．RTOS のアプリケーションはスーパバイザとユーザモードのどちらでも動作できるが，特権レベル切り替えのオーバヘッドを削減するため前者が好まれる．RTOS と GPOS はあたかも独立したハードウェアで動作しているかのごとく，基本的にはお互いを意識しない．

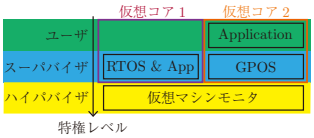


図 3: デュアル OS プラットフォームの例．

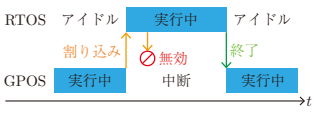


図 4: デュアル OS の実行形態．

RTOS と GPOS は VMM の下で図 4 に示すように動作する．GPOS は RTOS がアイドル状態の場合にのみ動作でき，GPOS 動作中のいかなる瞬間でも割り込みにより RTOS は実行権限を得られる．GPOS の割り込みは RTOS の動作中には全て無効化され，RTOS が全ての処理を完了してアイドル状態になると GPOS が動作を再開する．RTOS は GPOS に対して常に優先権を持つことになるため，リアルタイム性が保証される．

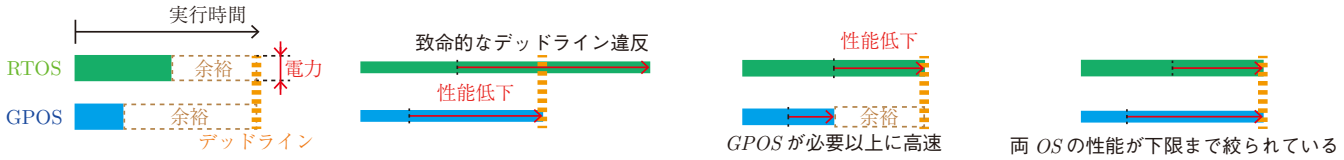
現状の仮想化手法で削減できるエネルギーには限界がある．デュアル OS プラットフォームの消費エネルギー削減を図 5 を用いて考える．エネルギー削減の前には図 5(a) に示すように RTOS と GPOS はデッドラインまでに処理を完了する．消費エネルギーを削減するためには，プロセッサの計算性能を下げなければならない．このとき，必然的に実行時間は伸びることになる．図 5(b) のように GPOS の実行時間がデッドラインに一致するよう計算性能を制御したとする．この場合 RTOS の実行時間も伸びてしまうためデッドライン違反が発生するが，この違反は一切許容できない．従って，省エネルギー制御では図 5(c) に示すように RTOS の実行時間をデッドラインに合わせる必要がある．しかしながら，この場合 GPOS が時間的余裕を持つため十分にエネルギーを削減できない．これが現状の仮想化手法におけるエネルギー最小化の限界である．提案手法は OS 毎に独立した計算性能の制御を可能にし，図 5(d) に示すように RTOS のリアルタイム性を確保しつつ GPOS の積極的なエネルギー削減を可能にする．

3. デュアル OS プラットフォームの消費エネルギー最小化

3.1 DVFS の諸原理

プロセッサはクロック毎に大量の MOSFET ゲートを駆動する．ここで，その総容量を C_{eff} とおく．容量 C を電圧 V で充放電する際には $CV^2/2$ のエネルギーが消費されるため，プロセッサは 1 サイクルで $E_d = C_{eff}{V_{DD}}^2/2$ のエネルギーを消費する．また，プロセッサはトランジスタの漏れ電流による静的エネルギーを同時に消費するが，本研究では E_d が支配的であるとみなして E_d のみを考える．

トランジスタのしきい値電圧 V_{TH} に比べて電源電圧 V_{DD} が十分大きいとき，プロセッサの最大動作周波数はパラメータ k と α を用いて $f_{MAX} = k(V_{DD} - V_{TH})^{\alpha}/V_{DD}$ と表



(a) エネルギー削減の前．高速，高消費かつリアルタイム．

(b) 計算性能が GPOS に合わせて調整された場合．低消費だが RTOS がデッドラインを違反する．

(c) 計算性能が RTOS に合わせて調整された場合．リアルタイムだがエネルギー効率は悪い．

(d) 提案する仮想化 DVFS は独立した性能調整を可能にする．リアルタイムかつ低消費．

図 5: デュアル OS プラットフォームに対するエネルギー最小化の困難性と提案手法．

せる．ここで，$E_{\mathrm{d}} \propto {V_{\mathrm{DD}}}^2$ の関係に着目する．これは V_{DD} を低下させることで E_{d} を二乗のスケールで削減できることを示している．一方，f_{MAX} は V_{DD} の低下に対してほぼ線形に悪化する [18]．DVFS はこうした関係に着目し，計算性能と引き換えに消費エネルギーの削減を図る．

最大周波数よりも低いクロック周波数 $f < f_{\mathrm{MAX}}$ でプロセッサを動作させることは，エネルギー削減の観点では効果がないことが知られている．従って，本研究では $f = f_{\mathrm{MAX}}$，すなわちプロセッサは与えられた V_{DD} に対して最大のクロック周波数で常に動作するとみなす．

3.2 時間パーティショニングによる DVFS アルゴリズム

3.3 節で詳述するが，提案ハードウェアは RTOS と GPOS の計算性能の独立な制御を可能にし，2.2 節で述べたデュアル OS プラットフォームの限界を克服する．本節では OS 間の協調動作によりシステム全体のエネルギー最小化を実現するため，軽量な DVFS アルゴリズムを提案する．提案アルゴリズムは RTOS が要求する必要最低限の計算性能のみに基づいて動作する．

以下では図 6 に示すような RTOS と GPOS の動作を仮定する．ここに，システム周期と呼ばれる時間間隔 T を定義する．時刻 $t = 0$ において RTOS は実行を開始し，時刻 $t = T_{\mathrm{RT}}$ において完了する．次に GPOS が実行を開始し，時刻 $t = T_{\mathrm{RT}} + T_{\mathrm{GP}}$ において完了する．その後クロックは時刻 $t = T$ まで停止し，システムは時刻を $t = 0$ として同様の動作を繰り返す．こうした実行方式は時分割スケジューリングもしくは時間パーティショニングと呼ばれ，ARINC 653 [19] のようないくつかの規格で定義されている．なお，割り込みは一切発生しないものとする．RTOS と GPOS の実行が ARINC 653 のように時間パーティショニングされている場合はスケジューリングの議論が簡単になるが，ソフトウェアの自由度が低くなる．そのため時間パーティショニングを仮定した議論の後，制限を緩和して一般のシステムへの拡張を行う．

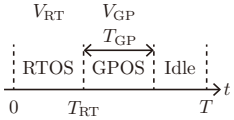


図 6: RTOS と GPOS の時間パーティショニング．

ここに，追加の変数を複数定義する．電源電圧 V_{RT} が RTOS 動作時に，V_{GP} が GPOS 動作時にプロセッサに供給されているとする．アイドル状態の電源電圧は静的エネルギーのみに影響するため，本研究では考慮しない．加えて，RTOS と GPOS の動作周波数をそれぞれ f_{RT}，f_{GP} とおく．先述したように，これらは電源電圧に対して自動的に決定される．さらに，RTOS に対する WCET 解析とデッドラインにより許容される最低のクロック周波数 $f_{\mathrm{RT,MIN}}$ が与えられる．

システム周期が完了するごとに，RTOS と GPOS の実行サイクル数はそれぞれ $C_{\mathrm{RT}} = T_{\mathrm{RT}}/f_{\mathrm{RT}}$, $C_{\mathrm{GP}} = T_{\mathrm{GP}}/f_{\mathrm{GP}}$ により算出される．ここで，$C_{\mathrm{RT,AVE}}$ と $C_{\mathrm{GP,AVE}}$ をこれまでの平均サイクル数から予想される次のシステム周期においての実行サイクルとする．

クロック周波数と電源電圧が連続的に変化させられる場合のエネルギー最小化を考える．例えば，クリティカルパスレプリカや可変出力の電源によりこうした機能が実現される．システムの消費エネルギーが最小になるための必要十分条件は $T_{\mathrm{RT}} + T_{\mathrm{GP}} = T$ かつ $f_{\mathrm{RT}} = f_{\mathrm{GP}}$ であるから，両 OS の最適な動作周波数は $f_{\mathrm{OPT}} = (C_{\mathrm{RT,AVE}} + C_{\mathrm{GP,AVE}})/T$ で与えられる．ただし，$f_{\mathrm{OPT}} < f_{\mathrm{RT,MIN}}$ である場合はデッドライン違反が発生してしまうため，リアルタイム性を担保できる範囲での最適解である $f_{\mathrm{RT}} = f_{\mathrm{RT,MIN}}$ と $f_{\mathrm{GP}} = C_{\mathrm{GP,AVE}}/(T - C_{\mathrm{RT,AVE}}/f_{\mathrm{RT,MIN}})$ を採用することになる．

次に，多くの商用プロセッサがそうであるようにクロック周波数と電源電圧を離散的にのみ変化させられる場合を考える．この場合エネルギー最小化は離散的なクロック周波数を変数，消費エネルギーを目的関数とする組み合わせ最適化問題となる．全ての組み合わせに対して消費エネルギーを計算する brute-force アルゴリズムにより，最適解を発見することができる．しかしながら，片方の OS に対するクロック周波数を決定できればもう一方は自動的に決定できることから計算量を削減することが可能である．

アルゴリズム 1 はそれぞれの OS に対して最適な電源電圧とクロック周波数を特定する提案 DVFS アルゴリズムの

アルゴリズム 1: 最適な電源電圧とクロック周波数の特定

```
1  Function CalcOptim(T, C_RT,AVG, C_GP,AVG, J, f_RT,MIN):
       Data: T: システム周期.
       Data: C_RT,AVG: RTOS の平均実行サイクル数.
       Data: C_GP,AVG: GPOS の平均実行サイクル数.
       Data: J = {(V_0, f_0), ···, (V_n, f_n)}: 動作条件の集合.
       Data: f_RT,MIN: RTOS の最小動作周波数.
       Description : 消費エネルギーを最小化する電源電圧と
                     クロック周波数の組み合わせを特定
                     する.
2      E_MIN ← ∞ ; // 最小値を初期化.
3      C' ← NULL ; // 最適解を初期化.
       /* 全ての動作条件に対して繰り返す.                    */
4      for (V_RT', f_RT') ∈ J do
5          if f_RT' < f_RT,MIN then
6              continue ; // RTOS にとって遅すぎる.
7          end
           /* RTOS の新しい実行時間を計算する.               */
8          T_RT' ← C_RT,AVE / f_RT';
           /* GPOS の最小動作周波数を計算する.               */
9          f_GP,MIN ← C_GP,AVE / (T − T_RT');
10         f_GP' ≥ f_GP,MIN を満たす最小の (V_GP', f_GP') ∈ C
             を見つける;
11         E ← 電源電圧 V_RT', V_GP' と実行サイクル数
             C_RT,AVE, C_GP,AVE による総エネルギー消費;
12         if E < E_MIN then
               /* これは最適解の候補.                        */
13             E_MIN ← E;
14             C' ← (V_RT', f_RT', V_GP', f_GP');
15         end
16     end
17     return C'
18 end
```

疑似コードである．10 行目の計算は例えば 1 クロックあたりの平均消費エネルギー $E_{\mathrm{AVG}}(V_{\mathrm{DD}})$ が与えられている場合は $E = C_{\mathrm{RT,AVE}} E_{\mathrm{AVG}}(V_{\mathrm{RT}}') + C_{\mathrm{GP,AVE}} E_{\mathrm{AVG}}(V_{\mathrm{GP}}')$ で求められる．

brute-force アルゴリズムの計算量は動作条件数 n に対して $O(n^2)$ で与えられるが，提案アルゴリズムは 10 行目において V_{GP}' と f_{GP}' を $O(1)$ の計算量で特定できれば全体の計算量が $O(n)$ で抑えられる．ここで，クロック周波数がある定数 s 刻みで与えられたとする．例えば，$f_0 = 20\,\mathrm{MHz}$, $f_1 = 40\,\mathrm{MHz}$, $\cdots$, $f_4 = 100\,\mathrm{MHz}$ のような場合である．このとき，インデックス $i = \lceil f_{\mathrm{GP,MIN}}/s \rceil - 1$ により $f_{\mathrm{GP}}' = f_i$ がわかる．例えば，$f_{\mathrm{GP,MIN}} = 70\,\mathrm{MHz}$ であれば $i = \lceil 70/20 \rceil - 1 = 3$ より $f_{\mathrm{GP}}' = f_3 = 80\,\mathrm{MHz}$ となる．

定数 s が与えられていない場合においても，ハッシュテーブルを利用することで f_{GP}' を高速に特定できる．クロック周波数は一般的に，f_i =75 MHz, 150 MHz, 220 MHz, 275 MHz, 300 MHz のように 1 MHz の解像度で与えられる．この場合，配列 a を $a[0..74] = 1$, $a[75..144] = 2$, $a[145..199] = 3$ のように設定することにより 1 MHz 単位の $f_{\mathrm{GP,MIN}}$ に対して $i = a[f_{\mathrm{GP,MIN}} - 76\,\mathrm{MHz}]$ が成立する．ただし，$f_{\mathrm{GP,MIN}} \leq 75\,\mathrm{MHz}$ である場合は $f_{\mathrm{GP}}' = 75\,\mathrm{MHz}$, また $f_{\mathrm{GP,MIN}} > 275\,\mathrm{MHz}$ である場合は $f_{\mathrm{GP}}' = 300\,\mathrm{MHz}$ とする．

3.3 仮想化 DVFS のハードウェア支援

アルゴリズム 1 によるエネルギー最小化を実現するためには，プロセッサの計算性能を OS と同時に切り替える必要がある．図 7 にプロセッサの計算性能を即座に切り替える提案ハードウェア構成を示す．通常，プロセッサはコアのクロック用に一つの PLL のみを搭載しているが，提案手法では RTOS と GPOS の各々に専用の PLL を用意する．また電源も別々に備えており，アナログマルチプレクサが二系統の電源を高速に切り替える．

ソフトウェアからは提案ハードウェアは図 8 のように見える．RTOS と GPOS に専用の二つの仮想プロセッサが同一の物理プロセッサ上で動作しており，また電源とクロックが個別に与えられる．既存の仮想化手法と同じく，割り込みにより RTOS は GPOS 動作中のいかなる場合でも実行を開始できる．このとき同時に提案ハードウェアが電源とクロックを切り替えるため，GPOS がどれだけ低速に動作していても RTOS は即座に高速動作ができる．こうしたハードウェア支援により，それぞれの OS の計算性能を独立に制御することができる．本研究ではこの技術を仮想化 DVFS と呼称する．

安定した電源電圧を複数用意して切り替える手法として，パワーゲーティング及び VDD ホッピングなどの手法が存在する．提案手法はこれらの電源切り替えとデュアル OS プラットフォームを組み合わせ，それぞれの電圧に役割（RTOS 専用と GPOS 専用）を与えたものともいえる．また複数の固定電圧源を切り替えて DVFS を行う voltage dithering という手法も存在する [20] が，提案手法はそれぞれの OS の電圧が可変である点が異なっている．

電源電圧の切り替え遅延においては，DC-DC コンバータの応答速度が支配的になる．切り替え時の DC-DC コンバータの挙動は負荷ステップ応答として解釈することがで

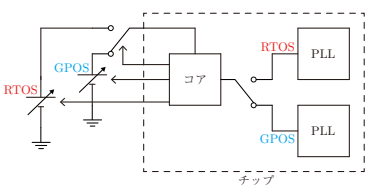


図 7: 提案するハードウェア構成．

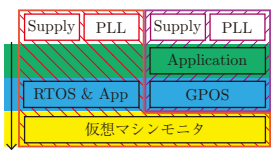


図 8: ソフトウェアから見た提案システム．

きる．切り替え前において，負荷電流は無視できる程度であるため平滑コンデンサの充電電圧は V_{RT} と一致する．ところが切り替え時の電流変化 ΔI により，コンデンサの直列抵抗成分 R_{ESR} において瞬時的な電圧降下 $R_{\mathrm{ESR}}\Delta I$ が発生する．許容されるアンダーシュート量と復帰時間が与えられれば，平滑コイルとコンデンサの値を決定することができる [21]．加えて，[22] によれば DC-DC コンバータの負荷ステップ応答は RLC 並列回路のステップ応答，すなわち二次系の減衰振動と類似する．

DC-DC コンバータの応答が要求を満たさない場合は，以下のような対策により解決できる．

- 平滑コイルとコンデンサを大きくし，応答時間を短くする [21]．
- タイミング違反を防ぐため，過渡状態にある間はクロック周波数を下げる．
- 切り替え前に電圧を高く設定しておき，収束後に下限値まで下げる．
- V_{RT}，V_{GP} とコアの電源を複数の並列なマルチプレクサで接続し，それらを一つずつ切り替えることで ΔI を減らす [23]．

まとめると，提案手法を実現する理想的なプロセッサは以下の機能を備えている．

- RTOS と GPOS を仮想化支援機能により同時に実行できる．
- コアのクロックを二つの PLL によって生成でき，同期機構のあるマルチプレクサで即座に切り替えられる．
- OS 切り替えに連動して電源電圧とクロック周波数が変更される．

3.4 アルゴリズムの軽量さと実用性

アルゴリズム 1 は GPOS に関してベストエフォートで計算性能を調整するため，RTOS により CPU 時間が使い果たされた場合は GPOS の CPU 時間が不足する．この場合は負荷に対して計算性能が足りておらず，クロック周波数を上げなければならない．しかしながら，どの程度性能が足りていないかを知る手段はないため，要求される最低のクロック周波数を得ることはできない．

GPOS の CPU 時間が足りない場合にクロック周波数をどう制御すべきか考える．現実的な方法として，CPU 時間が足りなくなった場合は両 OS のクロック周波数を最大にし，その後負荷が軽くなるにつれ徐々に落とす戦略が考えられる．

アルゴリズム 1 はシステムの時間パーティショニングを仮定しているが，これは議論を単純にするためであり必須ではない．アルゴリズム 1 の本質はシステム周期と実行時間の相対関係にあり，いつどちらの OS が動作しているか，どちらの OS が何回動作したか等には依存しない．瞬時のパフォーマンス変更を可能にするハードウェア構成もまた任意のタイミングにおける OS 切り替えを可能とする．この場合，両 OS の実行サイクル数は適当な周期で測定される必要がある．

ソフトウェアは時間パーティショニングされている必要はないが，実行サイクル数を測定する周期の選択は問題になり得る．短すぎる周期は測定による負荷を増大させ，長すぎる周期は負荷変動に対する応答を遅くする．そのため，一定周期に加えてプロセス起動などのイベント発生に合わせて測定と計算性能の制御を行う方法が考えられる．

3.5 マルチコアへの拡張

提案ハードウェアは同様の構成でマルチコアへ拡張を行うことができるが，コアごとに DC-DC コンバータを用意する構成は製造コストを増大させる．加えて，ロードバランスを考慮したアルゴリズム 1 の拡張が必要である．例えば，理想的な状況として全てのコアが全く同一の負荷状況であれば，シングルコアの場合と同様に電源は二つで済む．

4. 評価実験

電源電圧の切り替えレイテンシを測定する．3.3 節で述べたように，提案手法の実装に対して理想的なプロセッサは特殊な機能を備えている必要があるが，現在市販されているプロセッサに理想的なものは存在しない．そこで，本研究では NXP 社製 i.MX RT685 をターゲットとして提案手法の環境を模倣する．RT685 はアプリケーションプロセッサではなくマイクロコントローラであるが，コアや周辺回路の構成が単純であるためソフトウェアの実装が容易である．加えて評価ボードはコアの電源ラインにアクセスできるよう設計されており，実験を簡単に行うことができる．また，コアには ARM 社の TrustZone 技術が搭載されており，VMM による仮想化を実現できる．以上より，RT685 は提案手法の基本的な部分を検証するために適しているといえる．

表 1 に RT685 の動作条件を示す．電源電圧 V_{DD} と最大動作周波数 f_{MAX} の値はデータシートによる．これら動作条件に対して CoreMark ベンチマークを実行し，平均消費電流 I_{AVG} を測定した後，平均消費エネルギー $E_{\mathrm{AVG}} = V_{\mathrm{DD}} I_{\mathrm{AVG}} / f_{\mathrm{MAX}}$ に変換した．RT685 は 5 つの条件で動作し，典型的なプロセッサと同様に計算性能と消費エネルギーはトレードオフの関係にあるとわかる．

表 1: RT685 の動作条件．

V_{DD} [V]	f_{MAX} [MHz]	E_{AVG} [pJ]
0.7	75	86.0
0.8	150	103
0.9	220	129
1.0	275	156
1.13	300	199

図 10 に実験に用いた回路を示す．回路は主に三つの IC とその他要素で構成されている．

(1) LTC3542 は降圧 DC-DC コンバータであり，RTOS 用の電源電圧 V_{RT} を生成する．

(2) AD5160 はデジタルポテンショメータであり，DC-DC コンバータのフィードバック回路に挿入されている．ワイパーを調整することにより，V_{RT} を制御できる．

(3) ADG839 はアナログマルチプレクサであり，MCU からの信号 V_{SW} に応じて V_{RT} と V_{GP} をコアに分配する．

(4) PCA9250 は RT685 専用の電源管理 IC であり，通常はその出力をコアに供給する．本研究では PCA9250 の出力を GPOS 用の電源電圧 V_{GP} として扱う．

ソフトウェアは以下のように構成される．

- RTOS として TOPPERS ASP3 [24]，GPOS として FreeRTOS [25] を利用する．
- 割り込みハンドラの先頭において RTOS は V_{SW} を制御し，V_{core} を V_{RT} に切り替える．さらにタイマが起動され，1 µs 経過後に発火する．このマッチによりダイレクトメモリアクセスコントローラが起動し，CPU クロックの分周器レジスタに書き込みが行われることでクロック周波数が切り替わる．
- RT685 はコアのクロックを生成できる PLL を 1 つのみ搭載しているため，分周器を制御することでクロックの切り替えを模倣する．

4.1 達成可能なエネルギー削減量の概算

RT685 が図 9 に示すソフトウェアを実行している場合を考え，提案手法により達成可能なエネルギー削減量を概算する．RTOS と GPOS は 10 ms のシステム周期と 300 MHz のクロック周波数に対しそれぞれ 4.0 ms と 4.2 ms を費やす負荷を実行しているとする．本節では，この条件をベースラインと設定する．ここで，システムは時間パーティショニングされている必要はなく，それぞれの OS は 10 ms 内に合計でこれだけの時間実行されていることに注意する．また，便宜上，RTOS のデッドラインを 6.0 ms とする．これにより，RTOS の最低動作周波数は 220 MHz に定まる．これらの実行時間が WCET であると仮定すれば，両 OS の動作周波数は 275 MHz まで落とすことができる．この場合，ベースラインと比較して消費エネルギーを 28 % 削減できる．さらに AET が WCET の 80 % であると仮定すれば，GPOS の動作周波数をさらに 220 MHz まで落とすことができる．この場合，消費エネルギーは 40 % 削減さ

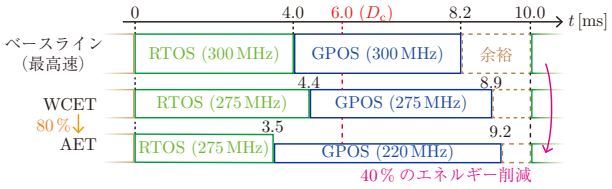


図 9: 提案手法による消費エネルギー削減．

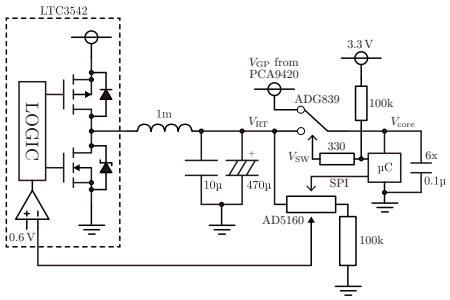


図 10: 実験に用いた回路（図 7 の具体化）．

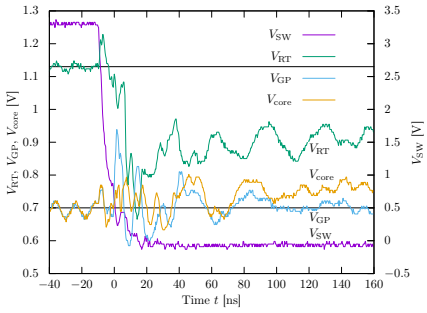


図 11: 0.7 V, 75 MHz から 1.13 V, 300 MHz への切り替え（ナノ秒精度）

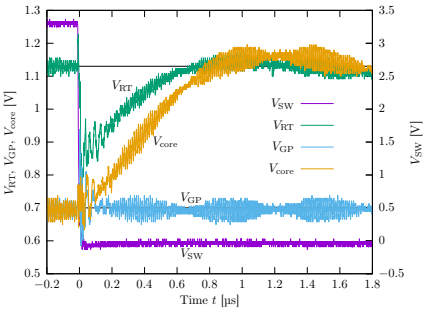


図 12: 0.7 V, 75 MHz から 1.13 V, 300 MHz への切り替え（マイクロ秒精度）．

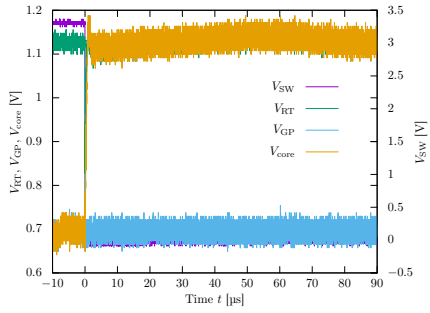


図 13: 0.7 V, 75 MHz から 1.13 V, 300 MHz への切り替え（10 マイクロ秒精度）．

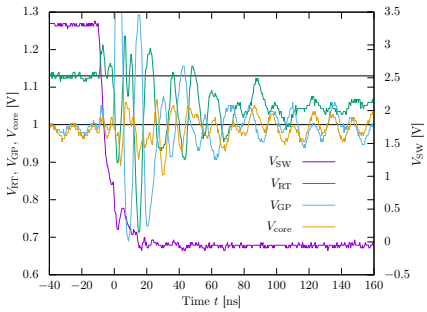


図 14: 1.0 V, 275 MHz から 1.13 V, 275 MHz への切り替え（ナノ秒精度）．

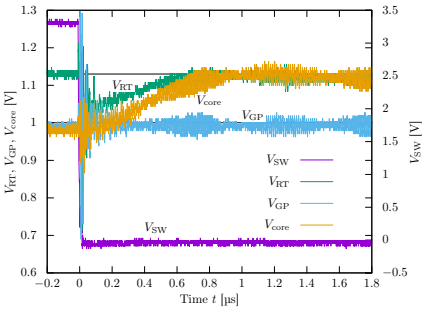


図 15: 1.0 V, 275 MHz から 1.13 V, 275 MHz への切り替え（マイクロ秒精度）．

れる．RTOS と GPOS はそれぞれ 220 MHz と 275 MHz で動作することもできるが，この組み合わせでは消費エネルギーがわずかに大きくなる．このように，提案アルゴリズムは複数の動作条件の候補から最適なものを選択する．

4.2 電圧切り替えのオーバヘッド測定

電圧変化が最大になる状況として，図 11 に 0.7 V, 75 MHz から 1.13 V, 300 MHz への切り替え時の電圧変化を示す．RTOS の割り込みハンドラが V_{SW} を制御することにより 3.3 V（ハイレベル）から 0 V（ローレベル）へと変化させる．ここで，V_{SW} が 0.8 V（アナログマルチプレクサのローレベル電圧の最大値）に達した瞬間を時刻 $t = 0$ とみなす．電源を切り替える瞬間に V_{RT} は急激に下降し，その後緩やかに上昇している．対して，V_{GP} は切り替えの瞬間に上昇している．最も重要な電圧である V_{core} は振動しながら緩やかに上昇している．

図 12 は図 11 をより広い時間範囲に展開したものである．V_{RT} の復帰に従い V_{core} も緩やかに上昇し，目標である 1.13 V への収束を 1 µs 以内に達成している．また，クロック周波数が 1 µs 経過時点で 75 MHz から 300 MHz に切り替わっているが，どの電圧波形にもその影響は現れていない．図 13 は同一の状況をさらに広い時間範囲で測定したものであり，V_{core} が切り替え時にオーバーシュートした後に定常値である 1.13 V を中心に振動している様子が確認できる．

図 14 は 1.0 V, 275 MHz から 1.13 V, 275 MHz への切り替え時，すなわち切り替え時間が最悪時の電圧変化である．出力電流の変化が DC-DC コンバータの電圧降下を引き起こすため，最悪の状況では電圧変化ではなく電流変化が最大になる．技術的な理由によりクロック周波数は変化させていないが，電源を切り替える瞬間の電流変化が重要であり，その後に周波数を切り替えるかどうかはアンダーシュート量に影響しない．ただし，この実験では 1 µs 経過後にプロセッサが 300 MHz で動作可能かは確認できない．

図 11 と比較すると，図 14 のアンダーシュート量は非常に大きく最大で 0.4 V を超えている．また，V_{GP} のオーバーシュート量も 0.4 V を超えている．V_{core} の振動は V_{RT}, V_{GP} と比較すれば小さいものの，最悪で 0.9 V まで低下している．RT685 の動作周波数は 0.1 V 毎に指定されているため，1.0 V 用のクロック周波数で動作している際に電源電圧が 0.9 V まで低下することは致命的なタイミング故障を引き起こしかねない．

図 15 は図 14 と同一の状況をより広い時間範囲で見たものである．図 12 と同様に，V_{core} はおおよそ 1 µs 以内に目標値である 1.13 V に到達している．また，図 15 の後 V_{core} は図 13 と同様に減衰振動する．

実験結果は図 10 の回路が 1 µs 以内に計算性能を切り替えられることを示している．この切り替え時間は一般的な商用の DVFS プロセッサと比較して 100 倍程度高速である [15]．しかしながら，プロセッサがタイミング故障を発生しなかったのは偶然に近いと思われるうえ，先行研究では ns オーダの切り替え時間も報告されている [23]．DC-DC コンバータの応答を改善するためには 3.3 節で述べたような工夫が求められる．

4.3 先行研究との比較

本節では提案手法と既存手法の比較について述べる．従来手法は [5] から派生したスケジューリング理論に基づいており静的にスケジューリングを行うため，クリティカルタスクの悲観的な WCET 見積もりにより計算性能が必要以上に高くなる．先行研究 [1–4] は準 WCET をクリティカルタスクに設定し，これを超過した場合はノンクリティカルタスクの実行を（[3] を除いて）放棄し本当のデッドラインを守る．準 WCET の超過はほとんど発生しないため，こうした手法は WCET に含まれる悲観性の影響を抑え，システムの平均消費エネルギーを大幅に削減する．しかしながら，ノンクリティカルタスクの安定した実行とエネルギー最小化は相反する要求であり，同時に達成することが難しい．加えて，大規模な汎用アプリケーションの WCET 分析は困難であるうえ，ユーザが WCET 分析の全く不可能な未知のソフトウェアをインストールする可能性もある．提案手法は動的に実行時間を測定するため，既存手法の言葉で表現すればノンクリティカルタスクの実行性能を保ちつつ消費エネルギーを最小化する最適な準 WCET を自動的に決定することができる．更に GPOS 側の WCET 解析を必要としないため，全く未知のアプリケーションにも対応することができる．

5. 結論

本研究では RTOS と GPOS の計算性能を独立に制御できる仮想化 DVFS を提案した．提案手法は OS の切り替えと同時に計算性能を即座に変更するため，GPOS の性能低下が RTOS の動作に影響しない．結果として OS 毎に独立した省エネルギー動作が可能となり，特に GPOS の消費エネルギーを積極的に削減できるようになる．加えて，提案 DVFS アルゴリズムが WCET と AET の差による既存手法の限界を克服する．

提案手法を実装できる理想的なプロセッサは現在のところ市場に存在しないため，仮想化支援機能 ARM TrustZone を実装した単純なマイクロコントローラである NXP i.MX RT685 を利用して提案手法を模倣した．評価実験により電圧切り替えが 1 µs 以内に完了すること，システム全体の消費エネルギーを 40 % 削減できることを確認した．

今後の課題は，提案手法に理想的なプロセッサを設計すること，電源電圧の切り替え時間を改善すること，マルチコアに対する拡張である．

謝辞

本研究は，JST，CREST，JPMJCR18K1 の支援を受けたものである．

参考文献

[1] Huang, P., Kumar, P., Giannopoulou, G. and Thiele, L.: Energy efficient DVFS scheduling for mixed-criticality systems, *International Conference on Embedded Software (EMSOFT)* (2014).

[2] Narayana, S., Huang, P., Giannopoulou, G., Thiele, L. and Prasad, R. V.: Exploring Energy Saving for Mixed-Criticality Systems on Multi-Cores, *Real-Time and Embedded Technology and Applications Symposium (RTAS)* (2016).

[3] Bhuiyan, A., Sruti, S., Guo, Z. and Yang, K.: Precise Scheduling of Mixed-Criticality Tasks by Varying Processor Speed, *International Conference on Real-Time Networks and Systems (RTNS)* (2019).

[4] Ali, I., Jo, Y.-I., Lee, S., Lee, W. Y. and Kim, K. H.: Reducing Dynamic Power Consumption in Mixed-Critical Real-Time Systems, *Applied Sciences* (2020).

[5] Vestal, S.: Preemptive Scheduling of Multi-criticality Systems with Varying Degrees of Execution Time Assurance, *International Real-Time Systems Symposium (RTSS)* (2007).

[6] Yang, J., Chen, Y., Wang, H. and Wang, B.: A Linux kernel with fixed interrupt latency for embedded real-time system, *International Conference on Embedded Software and Systems (ICESS)* (2005).

[7] Rybaniec, R. and Wieczorek, P. Z.: Measuring and minimizing interrupt latency in Linux-based embedded systems, *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments* (2012).

[8] Perneel, L., Fayyad-Kazan, H. and Timmerman, M.: Can Android be used for real-time purposes?, *International Conference on Computer Systems and Industrial Informatics (ICCSII)* (2012).

[9] Heiser, G.: The Role of Virtualization in Embedded Systems, *Workshop on Isolation and Integration in Embedded Systems* (2008).

[10] Kinebuchi, Y., Kanda, W., Yumura, Y., Makijima, K. and Nakajima, T.: A Hardware Abstraction Layer for Integrating Real-Time and General-Purpose with Minimal Kernel Modification, *Software Technologies for Future Dependable Distributed Systems* (2009).

[11] Zuo, B., Chen, K., Liang, A., Guan, H., Zhang, J., Ma, R. and Yang, H.: Performance Tuning Towards a KVM-Based Low Latency Virtualization System, *International Conference on Information Engineering and Computer Science (ICIECS)* (2010).

[12] Liu, M., Liu, D., Wang, Y., Wang, M. and Shao, Z.: On Improving Real-Time Interrupt Latencies of Hybrid Operating Systems with Two-Level Hardware Interrupts, *IEEE Transactions on Computers* (2011).

[13] Sangorrín, D., Honda, S. and Takada, H.: Reliable Device Sharing Mechanisms for Dual-OS Embedded Trusted Computing, *International Conference on Trust and Trustworthy Computing (TRUST)* (2012).

[14] Dong, P., Burns, A., Jiang, Z. and Liao, X.: TZDKS: A New TrustZone-Based Dual-Criticality System with Balanced Performance, *International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)* (2018).

[15] Min Allah, N., Wang, Y.-J., Xing, J.-S., Nisar, M. and Kazmi, A.-R.: Towards Dynamic Voltage Scaling in Real-Time Systems-A Survey, *IJCSES International Journal of Computer Sciences and Engineering Systems* (2007).

[16] Bhatti, M. K., Belleudy, C. and Auguin, M.: An inter-task real time DVFS scheme for multiprocessor embedded systems, *Conference on Design and Architectures for Signal and Image Processing (DASIP)* (2010).

[17] Bhatti, K., Belleudy, C. and Auguin, M.: Power Management in Real Time Embedded Systems through Online and Adaptive Interplay of DPM and DVFS Policies, *International Conference on Embedded and Ubiquitous Computing (EUC)* (2010).

[18] Burd, T., Pering, T., Stratakos, A. and Brodersen, R.: A dynamic voltage scaled microprocessor system, *IEEE Journal of Solid-State Circuits* (2000).

[19] ARINC: *653P1-5 Avionics Application Software Standard Interface, Part 1, Required Services* (2019).

[20] Gutnik, V. and Chandrakasan, A.: Embedded power supply for low-power DSP, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (1997).

[21] Redl, R., Erisman, B. and Zansky, Z.: Optimizing the load transient response of the buck converter, *Applied Power Electronics Conference and Exposition (APEC)* (1998).

[22] Gezgin, C.: Predicting load transient response of output voltage in DC-DC converters, *Applied Power Electronics Conference and Exposition (APEC)* (2004).

[23] Beigne, E., Clermidy, F., Lhermet, H., Miermont, S., Thonnart, Y., Tran, X.-T., Valentian, A., Varreau, D., Vivet, P., Popon, X. and Lebreton, H.: An Asynchronous Power Aware and Adaptive NoC Based Circuit, *IEEE Journal of Solid-State Circuits* (2009).

[24] TOPPERS: Toyohashi OPen Platform for Embedded Real-Time Systems: http://www.toppers.jp.

[25] FreeRTOS: https://www.freertos.org.