一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
CPSY2018-106,DC2018-88（2019-03）

IPC制御を用いたSMTプロセッサ向けRT-VFS手法

鈴木　宏海†　井出　陽介†　塚原　悠太††　山﨑信行†

† 慶應義塾大学 理工学部
†† 慶應義塾大学 大学院理工学研究科
E-mail: †{suzuki,ide,yuta,yamasaki}@ny.ics.keio.ac.jp

あらまし　組込みリアルタイムシステムはリアルタイム性の維持のみならず，消費電力を削減することが要求されている．本研究では Simultanus Multithreaded (SMT) プロセッサにおいて個々のスレッドの実行速度を制御できる Instructions Per Clock cycle (IPC) 制御機構を用いて論理プロセッサに割当てられたタスクの IPC を考慮して論理プロセッサの実行速度を制御することで利用率を均一化し，スケジューラビリティの維持と効率的な Real-Time Static Voltage and Frequency Scaling (RT-SVFS) を行う手法を提案する．また，IPC 制御により論理プロセッサの実行速度を動的に変化させることでタスクの実行時間を保ったまま動作周波数を低下させて消費電力を削減する Real-Time Dynamic Voltage and Frequency (RT-DVFS) 手法を提案する．シミュレーションによる評価の結果，提案手法は既存手法と比べてより高いスケジューラビリティと消費電力の削減が可能であることを示した．
キーワード　組込みリアルタイムシステム，SMT Processor，IPC 制御, RT-DVFS，RT-SVFS

Real-Time Voltage and Frequency Scaling Scheme with IPC Controlling for SMT Processor

Hiromi SUZUKI†, Yosuke IDE†, Yuta TSUKAHARA††, and Nobuyuki YAMASAKI†

† Faculty of Science, Keio University
†† Faculty of Science, Keio University
E-mail: †{suzuki,ide,yuta,yamasaki}@ny.ics.keio.ac.jp

Abstract　In the field of Real-Time embedded systems, both of high-performance and low-power consumption are required. In this paper, we propose an algorithm to make utilization of logical processors even in order to improve schedulability and achieve efficient Real-Time Static Volatage and Frequency Scaling (RT-SVFS) with Instructions Per Clock cycle (IPC) Control mechanism which controls the thread execution speed. In addition, we also propose a method of Real-Time Dynamic Voltage and Frequency Scaling (RT-DVFS) that can maintain executed time of tasks with dealing with the efficiency of task executing by IPC control. According to the simulation results, the proposed techques improve schedulability and power consumptions compared by the existing techniques.
Key words　Embedded Real-Time System, SMT Processor, IPC Control, RT-DVFS, RT-SVFS

1. まえがき

組込みリアルタイムシステムの多くはバッテリーで駆動するため，システムの消費電力はシステムの稼働時間に大きく影響を与える．従って消費電力を削減することは稼働時間の増大やバッテリーのコストダウンに繋がる重要な要素である．一般に組込みシステムで最も電力を消費するのはプロセッサであり，プロセッサの電力の削減はシステム全体の消費電力の削減に有効と言える．

低消費電力をソフトウェアによって実現する手法の一つに Real-Time Voltage and Frequency Scaling (RT-VFS) がある．RT-VFS はプロセッサの動作周波数と動作電圧をリアルタイム性を維持できる範囲で制御できる手法であり，動的に制御するものを Real-Time Dynamic VFS (RT-DVFS)，静的に制御するものを Real-Time Static VFS (RT-SVFS) と呼ぶ．

Simultaneous Multithreaded (SMT) プロセッサ [1] は，1 つのコア内で複数のスレッドを同時に実行することができるプロセッサである．同時実行されるタスクは 1 つのコアのハードウェア資源を共有するため，実行時間の予測性が低下する欠

点を持つ．これを解消するのが各スレッドの実行速度をソフトウェアによって制御する Instruction Per Clock cycle (IPC) 制御機構である．しかし，SMT プロセッサの各スレッドを論理プロセッサとみなした場合，全ての論理コアが同じ動作周波数と動作電圧によって実行される．そのため論理プロセッサの負荷に偏りが生じた場合，負荷の高い論理プロセッサがボトルネックとなりプロセッサ全体の消費電力が増大する．また，IPC 制御機構によって実行速度を制限することで論理プロセッサの負荷が変動してしまい，リアルタイム性を損なう恐れがある．

本研究では，IPC 制御機構を持つ SMT プロセッサである Responsive Multithreaded Processor (RMTP) を対象に，システムモデルにタスクの IPC を追加し，IPC 制御機構による論理プロセッサの速度と論理プロセッサの負荷の関係から各スレッドに対して効率的な実行速度の割当てを行うことで負荷を分散し，RT-SVFS による消費電力の削減を可能とする手法を提案する．同時に，アイドル中の論理プロセッサの実行速度を他の実行中の論理プロセッサへ移譲することでタスクの実行時間を保ったまま動作周波数を低下させる RT-DVFS 手法を提案する．

本論文の構成は次の通りである．2. では，RT-VFS について述べ，3. では IPC 制御機構について述べる．4. では関連研究について述べる．5. では提案手法を説明する．6. では提案手法の評価及びその考察について述べる．最後に 7. で本論文をまとめる．

2. Real-Time Voltage and Frequency Scaling

組込みシステムはバッテリーを電源とした携帯機器も多く，消費電力は駆動時間の長さに直結するため，低消費電力の要求は大きい．また，組込みシステムではプロセッサが占める消費電力が大きく，プロセッサの省電力化はシステム全体の消費電力に大きく寄与する．

ソフトウェアによって動作電圧と周波数を制御し，消費電力を削減する手法に Volatage and Frequency Scaling (VFS) がある．動作周波数の低下はタスクの実行時間に影響を与えるため，組込みリアルタイムシステムでは全てのタスクのリアルタイムシステム性を満たしつつその範囲で動作電圧を制御する Real-Time Voltage and Frequency Scaling (RT-VFS) が用いられる．特に動的に動作周波数と動作電圧を制御する手法を RT-Dynamic VFS (RT-DVFS)，静的に動作周波数と動作電圧を制御する手法を RT-Static VFS (RT-SVFS) と呼ぶ [2]．

3. IPC 制御機構

Instructions Per Clock cycle (IPC) とは，プロセッサが 1clock あたりいくつの命令を処理できるかを示す尺度である．IPC はハードウェア，ソフトウェアの両方に依存する．IPC の上限値は完全にハードウェアによって規定される．また，IPC は浮動小数点演算のレイテンシやメモリアクセスのレイテンシ，命令間の依存関係によるストールや，分岐予測のミスなどによって低下する．ソフトウェアはメモリアクセスの順番や変数のリネームなどを活用して，これらのハードウェアによる IPC の低下を抑える命令流をハードウェアに与えることで IPC の低下を防ぐことができる．また，マルチスレッド機能を持つプロセッサの場合，複数のタスクが同時にメモリアクセスを行うときなどにリソースの奪い合いが発生し，それぞれのタスクをシングルプロセッサで実行した場合よりも IPC が低下することがある．タスクにはそれぞれメモリアクセスの多寡や浮動小数点演算の多寡などタスクごとの特色があり，タスクの平均 IPC はタスク毎に異なる．一般に平均的なタスクを単独で実行した場合の IPC は 0.5 程度と言われている．

IPC 制御機構 [7] は，スレッドの実行速度を制御するための機構である．スレッドに対し指定したクロック周期の間のコミット数を測定し，単位時間あたりのフェッチを制御することでスレッドの実行速度を制御する．IPC 制御機構によってスレッドの実行速度を制御することで,，単一のスレッドがハードウェア資源を独占することを抑制し全てのスレッドがタスクを実行することができるため，各スレッドの実行時間予測性を高めることができる．

4. 関 連 研 究

RT-DVFS 手法として，Pillai らはシングルスレッドプロセッサ向けの Cycle-Conseving (CC) DVFS 手法及びリアルタイム性を満たす動作周波数を求める Look-Ahead (LA) アルゴリズムを提案した [3]．CC はタスクの実際の実行時間が最悪実行時間より短くなることを想定し，実行が早期に終了した分次に実行されるタスクの実行周波数を低下させる．タスクの実行時間予測を導入することで周波数を低下させる RT-DVFS 手法として，Zhu らは EDF スケジュールに対し実行時間予測にフィードバック制御を導入し予測した実行時間に基づき動作周波数を落とす手法を提案した [4]．また，Muhammad らは実行時間予測に強化学習を用いる手法を提案した [5]．しかし，SMT プロセッサの場合，全てのスレッドが動作周波数と動作電圧を共有するため，論理プロセッサに対してこれらの手法を適用することは困難である．また，スレッド間の資源競合によってタスクの実行時間の予測性が低下することも SMT プロセッサでの DVFS を困難にしている．SMT を対象とした RT-DVFS 手法として Hetero Efficiency to Logical Processor (HeLP) が提案されている [8]．SMT に対する RT-SVFS 手法としては，Yamada らが IPC 制御機構を用いて Proportionate IPC (PIPC) を提案した [12]．PIPC は論理プロセッサの実行速度を割り当てられたタスクセットの負荷に応じて比例配分することで各論理プロセッサの負荷を均一化する．

5. 提 案 手 法

5.1 システムモデル

本論文では，IPC 制御機構を持つ SMT Processor である Dependable Responsive Multithreaded Processor (D-RMTP) [9] を用いたパーティションスケジュールを想定する．

プロセッサには M 個の論理プロセッサ$\{LP_1, LP_2, \ldots, LP_M\}$が存在し，プロセッサは 1 クロックに IPC_{max} 個の命令を発行可能とする．また，各 LP_j の目標 IPC を ipc_j^{lp} と表す．システムの開始時にタスクセットが与えられ，タスクセットは n 個の周期タスク$\{\tau_1, \tau_2, \ldots, \tau_n\}$から構成されている．全てのタスクは互いに独立に処理を行い，任意の時点でプリエンプション可能であるとする．各タスク τ_i は (C_i, T_i, ipc_i) のタプルで定義され，C_i はタスクを単独で実行した場合の最悪実行時間，T_i は周期，ipc_i は単独実行時のタスクの平均 IPC を表す．各タスクは周期ごとに実行され，その実行単位をジョブと呼ぶ．

全てのタスクの相対デッドラインは周期に等しく，3. 節より，全 LP の IPC は一定と仮定する．プロセッサの動作周波数は l 段階の値$\{f_1, f_2, \ldots, f_l \mid f_1 < f_2 < \cdots < f_l\}$ が設定可能であるとする．動作電圧 V_k は周波数 f_k でプロセッサを動作させるのに必要な最低値であるとする．

IPC 制御が行われている LP でタスクを実行した場合，タスクの実行効率 e は

$$e = \min\left(1, \frac{target\ ipc\ of\ LP}{nonblocking\ ipc\ of\ task}\right) \quad (1)$$

と表される．分子は LP の目標 IPC であり，分母はタスクの IPC を意味する．LP に十分な目標 IPC が割り当てられていても，タスクの並列性が十分でなかった場合には目標 IPC に達することはない．そのため，実行効率の最大値は 1 となる．IPC を考慮しないタスクのプロセッサ利用率を $u_i = C_i \ / \ T_i$ と定義し，IPC を考慮したタスクのプロセッサ利用率を $u_i^{actual} = u_i \ / \ e_i$ と定義する．

m 番目の LP に割り当てられたタスクセットを Π_m とし，その LP の利用率は $U(\Pi_m) = \sum_{\tau_i \in \Pi_m} u_i$ 及び $U^{actual}(\Pi_m) = \sum_{\tau_i \in \Pi_m} u_i^{actual}$ と表される．

5.2 プロセッサ利用率を均一化する IPC 割当て手法

u_k^{actual} $(\tau_k \in \Pi_m)$ は ipc_m^{lp} が $(0,\ ipc_k]$ の区間においては $(\infty, u_k]$ の単調減少で連続な関数であるため逆関数を求めることができる．この関数を用いて最大利用率の論理プロセッサと同じ利用率になるように IPC を決定することで，全 LP のプロセッサ利用率を合わせることができる．

Algorithm 1 に論理プロセッサの IPC を求める擬似コードを示した．LP には 2 行目から 11 行目にかけて Worst Fit でタスクが割り当てられており，$U(\Pi_m)$ によって降順にソートされている．$U(\Pi_m)$ が最大の LP_1 には Π_1 で最大の IPC を持つタスクと等しい ipc_1^{lp} が与えられる．これにより $U_1^{actual} = U_1$ となる．LP_2 から LP_M は U_1^{actual} と等しくなるよう目標 IPC を決定する．この時，$U_1 \geq U_2 \geq \ldots \geq U_M$ であり，$U_j = \sum u_k\ (\tau_k \in \Pi_j)$ であるため U_j^{actual} は $(\infty, U_j\ (\leq U_1)]$ の連続関数である．よって中間値定理から必ず U_1^{actual} と等しくなる目標 IPC が存在することが保証される．33 行目で全 LP の IPC の合計が最大値に収まっているかを判定し，超過していた場合には U_1^{actual} を再設定して ipc^{lp} を再計算する．

Algorithm 1 Worst-Fit and IPC Balancing Control

```
 1: set {U_1, U_2, ..., U_M} to 0
 2: for i = 1 to n do
 3:    set min_id to 1
 4:    for t = 1 to M do
 5:       if U_t < U_min_id then
 6:          set min_id to t
 7:       end if
 8:    end for
 9:    assign τ_i to LP_min_id
10:    update U_min_id
11: end for
12: sort LP with decreasing utilization (U_1 > U_2 > ··· > U_M)
13: for i = 1 to M do
14:    sort LP_i.τ with increasing ipc
15: end for
16: set ipc_1^lp = LP_1.τ.last_element
17: for i = 2 to M do
18:    for j = 1 to num of LP_i.τ do
19:       set a and b to 0
20:       for k = 1 to num of LP_i.τ do
21:          if k < j then
22:             add b to LP_i.u_k
23:          else
24:             add a to LP_i.u_k × LP_i.ipc_k
25:          end if
26:       end for
27:       if a/LP_i.ipc_j + b < U_1^actual then
28:          break for
29:       end if
30:    end for
31:    set ipc_j^lp to a/U_1 − b
32: end for
33: if Σ_j ipc_j^lp > IPC_max then
34:    set ipc_1^lp to ipc_1^lp × IPC_max Σ_j ipc_j^lp
35:    goto line 17
36: end if
```

5.3 IPC 制御を併用した RT-DVFS

提案手法の実行例を図 5.2 に示す．簡単のため，全てのタスクの IPC は十分に大きく，LP の目標 IPC を大きくすることで実行効率を自由に制御できると仮定している．全ての論理プロセッサの目標 IPC は 1 とする．

時刻 1 で LP_2 がアイドル期間に入ると，LP_2 に割り当てられた目標 IPC が他の論理プロセッサに 0.5 ずつ移動する．タスクは十分に大きい IPC を持つため，残っている 2 つのタスクの実行効率はそれぞれ 1.5 倍になる．この時，動作周波数を $\frac{2}{3}$ 倍に落としても，タスクの見かけ上の実行量が変化しない．続いて時刻 2 で LP_3 もアイドル期間に入ることで動作している LP は LP_1 のみになった．そのため，LP_1 の目標 IPC を 3 にすることで，実行しているタスクの実行効率は 3 倍になる．よって実行周波数を $\frac{1}{3}$ 倍にすることができる．時刻 3，4 では実行を行っている論理プロセッサはそれぞれ 1 つのみであるため，目標 IPC をそれぞれの論理プロセッサに移動させることで，動作

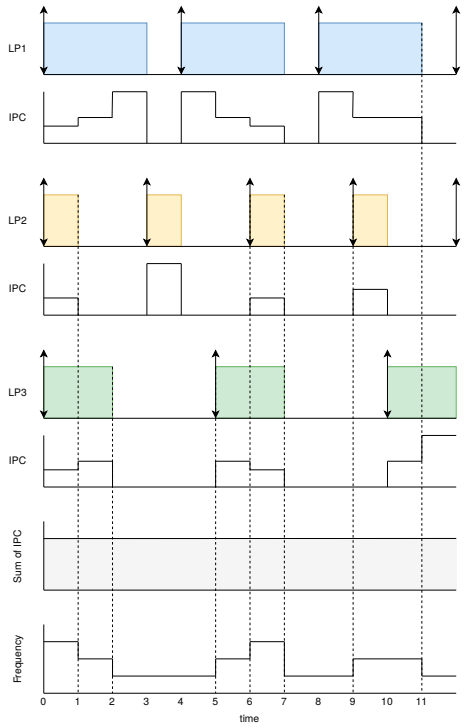

図 1　LP の目標 IPC を動的に変化させた場合の実行例
Fig. 1　Example of IPC Migration

Algorithm 2 IPC migration for DVFS

```
set α to ∞
for i = j to M do
    if LP_j has jobs then
        set α to min(α, ipc_active,j / ipc_j^lp)
    end if
end for
if α > 1.0 then
    for j = 1 to M do
        if LP_j has jobs then
            set ipc_j^lp to ipc_j^lp × α
        else
            set ipc_j^lp to 0
        end if
    end for
    if Σ_j ipc_j^lp > IPC_max then
        set ipc_j^lp to IPC_max × ipc_j^lp / Σ_j ipc_j^lp
        set α to α × IPC_max / Σ_j ipc_j^lp
    end if
else
    for j = 1 to M do
        set ipc_j^lp to ipc_j,default^lp
    end for
end if
```

周波数を落としたまま実行を行っている．

実際にはタスクの実行効率は 1 を超えられないため，論理プロセッサの目標 IPC とジョブの IPC の両方を考慮する必要がある．また，プロセッサ自身の IPC を超えて IPC を引き上げることもできない．以上 2 点を考慮した目標 IPC の割当てについて以下に示す．

ある時刻 t において論理プロセッサ LP_j で実行されているジョブのタスクを $\tau_{active,j}$ とおく．LP_j でその時刻ジョブが実行されていない時，LP_j は $\tau_\phi = (0, 0, \infty)$ を実行していると置き換えると，目標 IPC を引き上げることのできる最大倍率 α は全ての論理プロセッサ lp_j に対して

$$\alpha = \min\left(ipc_{active,j}/ipc_j^{lp}\right) \tag{2}$$

と定義される．α が 1 以上の時，ジョブを実行している全ての論理プロセッサの目標 IPC を α 倍することで実行中のジョブの実行効率を α 倍にすることができる．この時周波数を $1/\alpha$ 倍まで落としても，ジョブの実行効率は結果として 1 倍であり，リアルタイム性を損なうことはない．全ての論理プロセッサがアイドル状態の時，即ち $\alpha = \infty$ の場合，周波数を 0 にすることはできないため，取りうる周波数の中で最も低い周波数に設定するものとする．また，引き上げた論理プロセッサの目標 IPC の合計がプロセッサ自身の IPC を超える場合，各論理プロセッサの目標 IPC に応じて比例配分する．

DVFS を行うための目標 IPC 割当てアルゴリズムの擬似コードを Algorithm 2 に示す．$ipc_{j,default}^{lp}$ はシステム開始時に決定された各論理プロセッサの目標 IPC で，システム開始時に予め適当な値に設定されており，ipc_j^{lp} はシステム開始時に $ipc_{j,default}^{lp}$ で初期化されているものとする．

6. 評　　価

本章では 5. で提案した手法の有効性をシミュレーションにより評価する．最初に Algorithm 1 で示したアルゴリズムのスケジューラビリティを評価する．次に Algorithm 2 で示したアルゴリズムの消費電力について評価を行う．消費電力はパラメータとして RMTP の動作電圧，動作周波数及び動作コア数について予備実験を行い，消費電力を算出するためのパラメータとしてその結果を利用した．

評価シミュレーションに用いるタスクセットの生成条件について以下に述べる．タスクは，プロセッサ利用率が [0.01, 0.5]，周期が [1, 20]，IPC が [0.3, 1.3] の中から一様分布からランダムで選ばれたパラメータを持つ．最悪実行時間は周期と利用率から計算して求める．タスクは与えられた最悪実行時間を過不足なく全て使った時，実行終了とする．プロセッサの IPC は 4，論理プロセッサの数を 8 とした．この時，プロセッサは最大で 1 クロックあたり 4 命令を処理できるため，プロセッサ利用率の最大値は 400% となる．タスクのプロセッサ利用率を C_i / $T_i \times ipc_i$ として， タスクセットを生成した．タスクセット全体のプロセッサ利用率を 10% から 400% まで 10% 刻みにそれぞれ 500 個生成し，その平均値を評価値とした．

5. 2 節で提案した手法及び既存手法の各アルゴリズムでタスクセットを LP に割り当てた際のスケジュール成功率を図 2 に示す．パーティションスケジュールとして EDF を想定し，IPC を考慮したプロセッサ利用率が 100% を超えていない場合

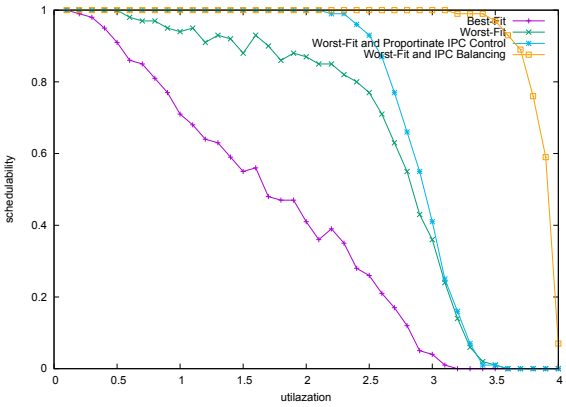

図 2　タスク割当ての成功率

Fig. 2　Evaluation of schedulablity

表 1　D-RMTP 評価キットの動作周波数倍率と動作電圧

Table 1　Frequency and voltage on D-RMTP Evaluation Kit

	動作周波数倍率			
	1.00	0.50	0.33	0.25
動作電圧	1.07 V	0.86 V	0.82 V	0.78 V

スケジュール成功とした．タスク IPC を考慮せず一律で同じ目標 IPC を論理プロセッサに与えている Worst-Fit，Best-Fit では，プロセッサ利用率が低いタスクセットでもスケジュールに失敗していることがわかる．また，目標 IPC を固定しているため，高い IPC を持ったタスクが生成されるとスケジュールに失敗してしまい，スケジュール成功率が不規則に上下している．また，PIPC は Worst-Fit，Best-Fit に比べスケジュール成功率の予測性が向上し，高いスケジュール成功率を示しているが，プロセッサ利用率が 200% を超えたあたりから急激にスケジュール成功率が低下している．これは PIPC がタスク自身の IPC を考慮しない目標 IPC の割当てを行っているために各 *LP* に割り当てられた目標 IPC が 0.5 付近から大きく動くことがなく，IPC の高いタスクによってプロセッサ利用率が上昇してしまっているためと考えられる．提案手法は図に示されているようにどのプロセッサ利用率においても既存手法に比べて高いスケジュール成功率を持ち，タスクセットのプロセッサ利用率が 200% 程度までは殆どのタスクセットをスケジューリングできていることがわかる．各論理プロセッサに目標 IPC を設定することで，割り当てられたタスクの実行に必要な処理能力を獲得できたためであると考えられる．

次に消費電力の評価について述べる．消費電力はシミュレーションにより評価した．シミュレーションのパラメータとして D-RMTP 評価キットの消費電力を測定した．表 1 に D-RMTP 評価キットの設定可能な動作周波数倍率と動作電圧を，図 3 に各動作電圧において動作可能な周波数で実行した場合の動作スレッド数と消費電力を示した．動作電圧を下げる毎に消費電力が大きく低下し，動作電圧を維持したまま周波数を下げることでも消費電力が低下している．また，動作スレッド数によっても消費電力が僅かながら変化している．図 4 に 1.07V で動作させた場合の動作周波数と消費電力の相関を示した．この結果

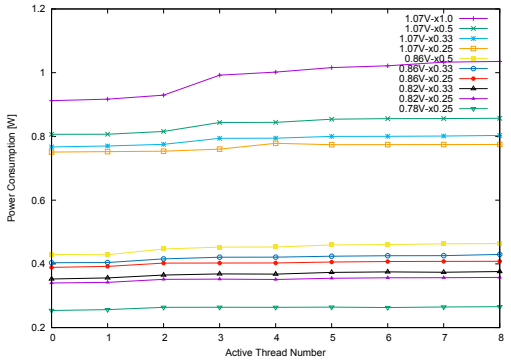

図 3　D-RMTP 評価キットの消費電力

Fig. 3　Power consumption on D-RMTP Evaluation Kit

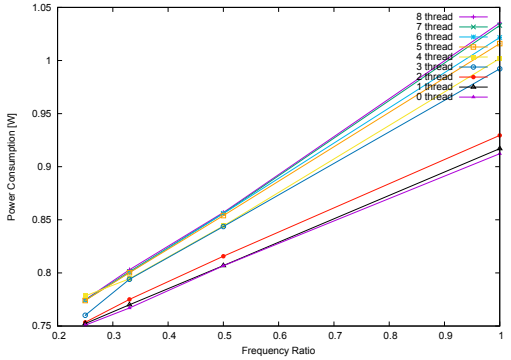

図 4　FS 時の D-RMTP 評価キットの消費電力

Fig. 4　Power consumption with frequency scaling on D-RMTP Evaluation Kit

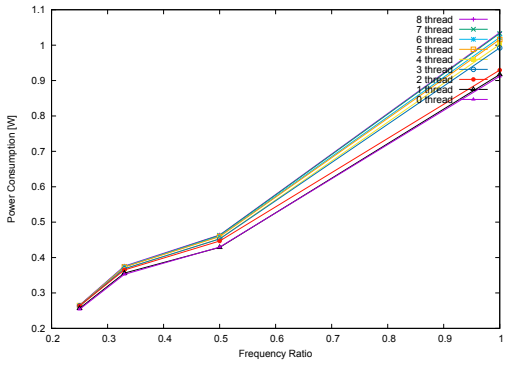

図 5　VFS 時の D-RMTP 評価キットの消費電力

Fig. 5　Power consumption with voltage and frequency scaling on D-RMTP Evaluation Kit

より，Frequency Scaling (FS) での消費電力を一次回帰による補間直線を用いて求めることとした．図 5 に動作周波数とともに動作電圧を落とした場合の動作周波数と消費電力の相関を示した．結果より，Voltage and Frequency Scaling (VFS) での中間周波数での消費電力を，実測値の線形補間として求めることとした．また，全論理プロセッサがアイドル状態の時，シングルコアプロセッサの周波数を 0 にすることはできないため，VFS を行う場合の最低動作周波数倍率を 0.25 倍とした．

予備評価によって得られた消費電力を用いてシミュレーションによる評価を行った．結果を図 6. に示した．各論理プロセッ

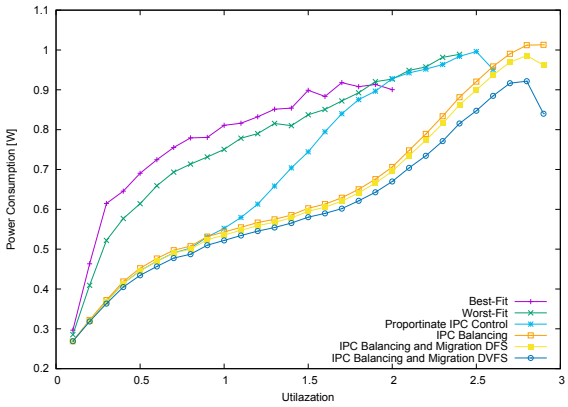

図 6　シュミレーションによる電力評価
Fig. 6　Evaluation of power consumption

サは Partitioned Rate- Monotonic によってスケジュールされており，スケジュール可能性解析によるスケジュールに成功したタスクセットを対象にして平均値をとっている．

IPC 制御を行わない Worst-Fit，Best-Fit によるパーテイショニングでは，論理プロセッサのプロセッサ利用率に偏りが生じ，要求周波数が増大してしまったために RT-SVFS を効果的に行うことができず消費電力が増大している．また，生成されたタスクの IPC によってプロセッサ利用率が変化するためタスクセットの利用率に対し消費電力が不規則に上下している．

提案手法である IPC balancing はこれらの既存手法や，Proportionate IPC より低消費電力で実行できていることがわかる．これらは RT-SVFS によって消費電力の削減を行っているため，この結果は提案手法が論理プロセッサの負荷分散をより効率的に行えていることを示している．さらに，IPC balancing に加えて IPC migration を行うことで，RT-DVFS を行った場合最大で 1 割程消費電力を削減できている．これは RM のプロセッサ利用率上限が 100% でないために，RT-SVFS を行った後もアイドル状態になる論理プロセッサが存在していることを利用している．動作電圧を変更せず，RT-DFS のみを IPC migration を用いて行った場合，消費電力は RT-DVFS 時より増大しているものの，さらなる低消費電力化には成功している．なお，Utilazation が 2.9 の時，IPC migration によって消費電力が大きく低下している．これはタスクセットをランダムに生成したことで，IPC の大きいタスクを多く含むタスクセットが生成され，IPC migration による周波数の低下が発生する頻度が増大したことに加えて，Utilazation の増大によってスケジュールに成功したタスクセットが減少したために平均値に誤差が生じたと考えられる．

7. ま と め

本論文では，IPC 制御機構を持つ SMT プロセッサ向けの RT-VFS 手法として，SMT プロセッサのスレッドを論理プロセッサとみなし，論理プロセッサに割り当てられたタスクの IPC を考慮しながら各スレッドの目標 IPC を定めることで，論理プロセッサ間の負荷を均一にする手法を提案した．また，論理プロセッサがアイドル状態の時，そのスレッドに与えられていた目標 IPC を他の実行中のスレッドの目標 IPC に振り分けることで，実行されているタスクの見かけ上の実行速度を維持したまま動作周波数を低下させる RT-DVFS 手法を提案した．評価では，ランダムに生成されたタスクセットに対し，スケジュール成功率を測定し，また IPC 制御機構を備えた SMT プロセッサである RMTP の消費電力を測定してその結果を用いて消費電力のシミュレーションを行い，提案手法が高いスケジューラビリティを持つこと，より低い消費電力でタスクセットを実行可能であることを示した．今後は実際に IPC を測定したタスクを用いて，実機による消費電力の評価を行う予定である．

謝辞　本研究は国立研究開発法人 科学技術振興機構 (JST) の助成を受けたものであることを記し，謝意を表す．

文　献

[1] Tullsen, Dean M.and Eggers, Susan J.and Levy, Henry M., "Simultaneous Multithreading: Maximizing On-chip Parallelism," SIGARCH Comput.Archit.News, vol.23, pp.392-403, 1995.
[2] Liu, C.L.and Layland, James W., "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," J.ACM, vol.20, pp.46-61, 1973.
[3] P.Pillai and K.G.Shin, "Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems," Proceedings of the 18th ACM Symposium on Operating Systems Principles, pp.89-102, 2001.
[4] Zhu Y.and Muller F., "Feedback EDF scheduling exploiting dynamic voltage scaling," Pro-ceedings of the Real-Time and Embedded Technology and Applications Symposium , pp.84–93, 2004.
[5] Fakhruddin Muhammad Mahbub ul Islam and Man Lin, "Hybrid DVFS Scheduling for Real-Time Systems Based on Reinforcement Learning," IEEE Systems Journal, vol.11, pp 931-940, 2017.
[6] Kei Fujii and Hiroyuki Chishiro and Hiroki Matsutani and Nobuyuki Yamasaki, "Dynamic Voltage and Frequency Scaling for Real-Time Scheduling on a Prioritized SMT Processor," The 1st International Workshop on Cyber-Physical Systems, Networks, and Applications, pp.9-15, 2011.
[7] 稲垣文二，山崎信行，"リアルタイム実行のための優先度付き SMT プロセッサ用 IPC 制御機構," 情報処理学会論文誌, vol.51, pp.2206-2215, 2010.
[8] Kei Fujii and Hiroyuki Chishiro and Hiroki Matsutani and Nobuyuki Yamasaki, "Dynamic Voltage and Frequency Scaling for Real-Time Scheduling on a Prioritized SMT Processor" The 1st International Workshop on Cyber-Physical Systems, Networks, and Applications, pp.9-15, 2011.
[9] K.Suito and R.Ueda and K.Fujii and T.Kogo and H.Matsutani and N.Yamasaki, "The Dependable Responsive Multithreaded Processor for Distributed Real-Time Systems," IEEE Micro, vol.32, pp.52-61, 2012.
[10] H.Aydin and Q.Yang, "Energy-Aware Partitioning for Multiprocessor Real-Time Systems," Proceedings of the 17th IEEE International Parallel and Distributed Processing Symposium, pp.113-121, 2003.
[11] M.R.Garey and D.S.Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," W.H.Freeman, 1979.
[12] 山田賢治，羽鳥雄介，萩原秀麿，溝谷圭悟，高須雅義，山崎信行, RMT Processor における IPC 制御を用いたリアルタイム静的電圧周波数制御, 組み込み技術とネットワークに関するワークショップ ETNET2015, vol.114, pp.101-106, 2015.