

インプリサイス計算モデルにおける 温度を考慮した動的電圧周波数制御の実機評価

溝谷 圭悟^{1,a)} 上田 陸平¹ 高須 雅義¹ 千代 浩之¹ 松谷 宏紀¹ 山崎 信行¹

受付日 2013年11月13日, 採録日 2014年5月17日

概要: マイクロプロセッサの高密度化にともない, 上昇するプロセッサ温度の制御がより重要な課題となっている. 消費電力を削減することでプロセッサの過度な温度上昇を防ぐ手法として Temperature-Aware Dynamic Voltage and Frequency Scaling (TA-DVFS) が提案されている. 本研究では, タスクを必須部分と付加部分に分割するインプリサイス計算モデルを TA-DVFS に適用することで, プロセッサ温度の上昇を防ぎつつ残ったプロセッサの余裕時間を活用してタスクの実行結果の品質を向上させる. TA-DVFS によりタスクの実行時間が増加しても, 付加部分の処理を中断することでリアルタイム性を維持する. 実機評価により, 実際に TA-DVFS にインプリサイス計算モデルを適用した際の温度変化および実行結果の品質向上の傾向を調査した. また, 得られた評価結果から消費電力とタスクの実行結果の品質の間にあるトレードオフを調査した.

キーワード: 組込みリアルタイムシステム, 動的電圧周波数制御, プロセッサ温度, インプリサイス計算モデル

Experimental Evaluation of Temperature-Aware DVFS on Imprecise Computation Model

KEIGO MIZOTANI^{1,a)} RIKUHEI UEDA¹ MASAYOSHI TAKASU¹ HIROYUKI CHISHIRO¹
HIROKI MATSUTANI¹ NOBUYUKI YAMASAKI¹

Received: November 13, 2013, Accepted: May 17, 2014

Abstract: As microprocessors grow in density, management of processor temperature is becoming a more important issue. Temperature-Aware Dynamic Voltage and Frequency Scaling (TA-DVFS) has been proposed to prevent significant processor temperature increases by reducing power consumption. In this paper, we apply an imprecise computation model which splits each task into mandatory and optional parts to TA-DVFS. It prevents processor temperature increases and utilizes remaining slack to improve quality of tasks while preserving the time constraints by terminating the execution of the optional part. We investigate the trend of temperature distribution and quality of tasks as TA-DVFS is practically applied to an imprecise computation model through experimental evaluation. Moreover, we investigate the trade-off between power consumption and quality of tasks by analyzing experimental evaluation results.

Keywords: embedded real-time systems, dynamic voltage and frequency scaling, processor temperature, imprecise computation model

1. 序論

近年, 組込みリアルタイムシステムの高機能化およびマ

イクロプロセッサの高密度化にともない, 消費電力の削減と上昇するプロセッサ温度の制御がより重要な課題となっている. プロセッサの発熱により生じる不具合を防ぐため, 冷却ファンやヒートシンクなど様々なハードウェアによるプロセッサの冷却方法が用いられてきた. しかし, ハードウェアによる冷却は容量の圧迫につながるため, 特に搭載

¹ 慶應義塾大学大学院理工学研究科
Graduate School of Science and Technology, Keio University,
Yokohama, Kanagawa 223–8522, Japan

^{a)} mizotani@ny.ics.keio.ac.jp

可能な容量に限りがある組込みリアルタイムシステムでは、ソフトウェアによるプロセッサの温度制御技術が非常に重要である。ソフトウェアによるプロセッサ温度制御の手法の1つとして、温度制約違反を防ぐために動的にプロセッサの供給電圧と動作周波数を調整する Temperature-Aware Dynamic Voltage and Frequency Scaling (TA-DVFS) があげられる。動作周波数および供給電圧の低減により消費電力を削減することで、効果的にプロセッサ温度の低下を図ることができる。しかし、温度が閾値を超過した際に動作周波数と供給電圧を下げる手法でリアルタイム性を維持するためには、プロセッサ利用率を低くする必要があり、動作周波数と供給電圧を下げない場合は大幅な余裕時間が発生してしまう。

余裕時間を利用してタスクの結果の品質を向上させる手法として、インプリサイズ計算モデル [8], [10] が提案されている。インプリサイズ計算モデルでは、タスクは必須部分 (mandatory part) と付加部分 (optional part) に分けられる。必須部分は許容可能な最低限の品質を持つ結果を生成し、付加部分は必須部分が生成した結果の品質を向上させる。付加部分は中断可能であり、付加部分に割り当てる余裕時間がなくなったとしても、付加部分を中断することでデッドラインミスを回避することが可能である。

本研究では、TA-DVFS により動的に変化する余裕時間を活用するため、TA-DVFS へのインプリサイズ計算モデルの適用を行う。中断可能な付加部分を持つインプリサイズ計算モデルを適用することで、消費電力削減により過度な温度上昇を防いだうえで、リアルタイム性を維持しつつタスクの実行結果の品質を向上させる。また、実機評価を行うことで、実際に TA-DVFS にインプリサイズ計算モデルを適用した際の温度制御および実行結果の品質向上の傾向を調査した。さらに、余裕時間を利用する2つの手法の間にあるトレードオフの関係について、得られた実機評価の結果から消費電力と実行結果の品質の観点で調査を行った。

本論文の構成は次のとおりである。まず、2章で背景について述べる。次に、3章で関連研究について述べる。4章では本研究で扱う TA-DVFS へのインプリサイズ計算モデルの適用について述べ、5章でスケジューラの実装方法について述べる。6章で実装の実機評価およびその考察を行う。最後に、7章で結論を述べる。

2. 背景

2.1 インプリサイズ計算モデル

インプリサイズ計算モデル [8], [10] は、一般計算モデル [6], [9] に計算の精度を向上させる部分である付加部分を追加したモデルである。インプリサイズ計算モデルでは、画像処理やロボットの経路選択のような生成された結果に対して精度が要求される処理を考慮している。付加部分は

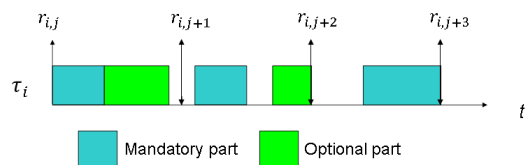


図1 インプリサイズ計算モデルの例

Fig. 1 Imprecise computation model example.

必須部分完了後に実行可能になる非リアルタイム処理であるため、付加部分で精度を向上させるために要求された実行時間を完了する必要はない。付加部分に割当て可能な時間が要求実行時間より短い場合、付加部分の実行を中断して精度の低い結果を生成する。図1にインプリサイズ計算モデルの例を示す。タスク τ_i は必須、付加部分の順で実行する。時刻 $r_{i,j+2}$ にリリースしたジョブは必須部分の実行を完了すると同時にデッドラインを迎えるため、付加部分を破棄する。

2.2 温度制御技術

半導体の温度と半導体内での化学反応速度の関係式はアレニウスの式により以下のように表すことができる。

$$k = A \times \exp\left(-\frac{E_a}{RT}\right) \quad (1)$$

k は反応速度、 A は定数、 E_a は活性化エネルギー、 R はボルツマン定数、 T は絶対温度である。また、この式を変形して以下のように表すことができる。

$$\log k = -\frac{E_a}{R} \frac{1}{T} + \log A \quad (2)$$

この式が示すように、半導体の温度が上昇すると反応速度が上昇し、半導体の寿命が短くなる。その結果、高熱状態での稼働が予期せぬエラーやプロセッサの損傷の原因となる。

本研究ではソフトウェアでの代表的な温度制御の手法である TA-DVFS に焦点を当てる。TA-DVFS では、動的に動作周波数および供給電圧を下げることで消費電力を削減し、それによりプロセッサの温度上昇を抑える。

ここで、プロセッサの温度と消費電力の関係について述べる。半導体素子のジャンクション温度と消費電力の関係は以下の式で表すことができる。

$$T_j = \theta_{ja} P_d + T_a \quad (3)$$

T_j はジャンクション温度、 T_a は周辺温度、 θ_{ja} は熱抵抗、 P_d は消費電力である。この式が示すように、消費電力を削減することで、プロセッサ内部の温度上昇を抑えることが可能である。

さらに、消費電力と動作周波数および供給電圧の関係について以下で述べる。近年のプロセッサなどの集積回路はCMOS回路で構成されており、その消費電力 P_{TOTAL} は以下の式で表すことができる。

$$P_{TOTAL} = P_{SW} + P_{LEAK} \quad (4)$$

P_{SW} はスイッチング電力, P_{LEAK} はリーク電力で, プロセッサの消費電力はこの2つの和で表すことができる. スイッチング電力は回路が動作して CMOS 回路で ON/OFF の切替えが発生することによって生じる電力であり, 以下の式で表される.

$$P_{SW} = \alpha \times C \times V_{DD}^2 \times f \quad (5)$$

α は動作率, C は負荷容量, V_{DD} は供給電圧, f は動作周波数である. スイッチング電力は供給電圧の2乗に比例し, 動作周波数に比例する. また, リーク電力は以下の式で表すことができる.

$$P_{LEAK} = I_{LEAK} \times V_{DD} \quad (6)$$

I_{LEAK} はリーク電流で V_{DD} は供給電圧である. リーク電流は電源が投入されていれば, プロセッサの動作, 停止にかかわらず発生する.

消費電力のうちスイッチング電力は, 式 (5) に示すように供給電圧の2乗と動作周波数にそれぞれ比例する関係にあり, 供給電圧と動作周波数を下げることで削減することが可能である. また, 消費電力を削減することで, 式 (3) に示すようにプロセッサ内部のジャンクション温度の上昇を抑えることが可能である. 以上のようにして, TA-DVFS では動的な動作周波数と供給電圧の調整により消費電力を削減することでプロセッサの温度上昇を防ぐ.

2.3 インプリサイス計算モデルと温度制御

2.1 節で述べたように, インプリサイス計算モデルは画像処理やロボットの経路選択などの生成された結果に精度が要求される処理を考慮している. 特に制御に大出力のモータが必要なヒューマノイドロボット [12] の場合, プロセッサやモータの排熱を考慮した省電力化とロボット制御用タスクの処理結果の精度向上の両立が求められる. このような要求を受けて本研究では, 温度制御技術である TA-DVFS にタスクの実行結果の品質を向上させるインプリサイス計算モデルを適用する. TA-DVFS により過度な温度上昇を防いであうで, インプリサイス計算モデルによりタスクの実行結果の品質を向上させる.

3. 関連研究

3.1 インプリサイス計算モデル

インプリサイス計算モデルのスケジューリングアルゴリズムとして Mandatory-First with Earliest Deadline (M-FED) [3] が提案されている. M-FED は, Earliest Deadline First (EDF) [9] に基づく動的優先度スケジューリングアルゴリズムである. M-FED は必須, 付加部分をそれぞれ EDF でスケジューリングする. ただし, 必須部分は付加部分よりも必ず優先度が高い. そのため, EDF 同様にシング

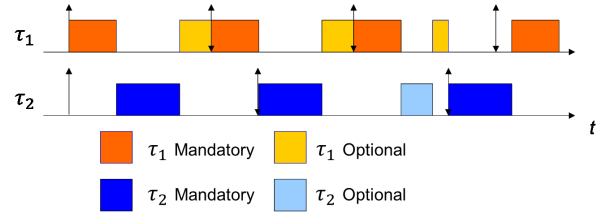


図 2 M-FED スケジューリング例

Fig. 2 M-FED scheduling example.

ルプロセッサにおけるスケジューリング可能なタスクセットのプロセッサ利用率の上限は 100% である. M-FED では一般計算モデルのタスクに関しても, 付加部分の実行時間を 0, すなわち必須部分のみのタスクとすることで EDF と同様のスケジューリングを行うことが可能である. M-FED でスケジューリングした例を図 2 に示す. τ_1 と τ_2 の必須部分をスケジューリングした後, 残ったプロセッサ時間で可能な限り τ_1 と τ_2 の付加部分のスケジューリングを行う.

Optional with Least-Utilization (OPT-LU) [1], [2] は付加部分の最悪実行時間が既知の場合, 付加部分の精度関数を最大化するスケジューリングアルゴリズムである. OPT-LU では, EDF のような最適なスケジューリングに基づき, タスクの必須部分のデッドラインミスを発生させない範囲で付加部分を優先して実行する. タスク τ_i の精度関数 f_i の集合を $F = \{f_1, f_2, \dots, f_n\}$, 付加部分の最悪実行時間 o_i の集合を $O = \{o_1, o_2, \dots, o_n\}$, 正の有理数を b_i , 付加部分に割当て可能な時間を t_i , 割当て可能な余裕時間を S と定義した場合, OPT-LU は以下の式を満たすようにスケジューリングを行う.

$$\begin{aligned} & \text{maximize} && \sum_{i=1}^n f_i(t_i) \\ & \text{subject to} && \sum_{i=1}^n b_i t_i = S \\ & && t_i \leq o_i \quad i = 1, 2, \dots, n \\ & && 0 \leq t_i \quad i = 1, 2, \dots, n \end{aligned}$$

3.2 TA-DVFS

TA-DVFS では, アルゴリズムに従って動的に動作周波数と供給電圧を調整することで温度制約違反を防ぐ. 最も単純で確実なのは, 温度センサから読み取った温度が閾値を超過した場合に動作周波数および供給電圧を下げる手法である [7]. また, 面積やコストのかかる温度センサの代用として, プロセッサ内のパフォーマンスカウンタを利用してプロセッサの温度を解析する手法もある [5]. しかし, 上記の研究は非リアルタイムシステムを対象としており, このような TA-DVFS の手法をリアルタイムシステムで用いた場合, 動作周波数と供給電圧を下げるタイミングが温度に依存するために, プロセッサ利用率が高い状態で動作周波数と供給電圧を下げた際にスケジューリング不可能な状態に

陥る危険性がある．この手法をリアルタイムシステムに適用した場合，リアルタイム性を保証するためにはプロセッサ利用率が限定され，供給電圧と動作周波数を下げない場合は大幅な余裕時間が生じてしまう．

一方で，プロセッサ温度に対して厳密に閾値を定めて動作周波数および供給電圧の調整を行うのではなく，プロセッサ利用率を制限せずに利用可能な余裕時間を割り当てることで消費電力を減らしつつベストエフォートに温度制約違反を減らす手法もある [4]．この手法では，PID 制御により実行時間を見積もってタスクを 2 つに分割し，分割したそれぞれのタスクに対して温度制約を満たしつつ消費電力を最小化する余裕時間を割り当てる分配を求める．ただし，タスクの実行時間の見積りおよび余裕時間を割り当てる分配の計算により計算量が増大するという問題点が存在する．

本研究では，リアルタイムシステムでの TA-DVFS の下で，リアルタイム性の維持のためにプロセッサ利用率が制限されながらも残った余裕時間を活用する手法としてインプリサイス計算モデルを用いる．

4. インプリサイス計算モデルを適用した TA-DVFS

4.1 システムモデル

本研究で扱うシステムモデルについて述べる．本研究では電圧と周波数の制御が可能なシングルプロセッサ上でのリアルタイムスケジューリングを対象としている．また，プロセッサの動作周波数に依存しない外部タイマとプロセッサ温度を読み取る温度センサが必要である．システムには n 個の独立した周期タスクで構成されるタスクセット $\Gamma = \{\tau_1, \tau_2, \dots, \tau_n\}$ が与えられる．各タスク τ_i は (m_i, o_i, T_i, D_i) のパラメータを持つ． m_i は必須部分の最悪実行時間， o_i は付加部分の最悪実行時間， T_i は周期， D_i は相対デッドラインである．今回扱うモデルでは相対デッドラインは周期に等しいものとする．プロセッサ利用率は $U = \sum_i m_i / T_i$ で定義される．タスクの付加部分の完了はスケジューリングの成否に無関係なため，タスクのプロセッサ利用率に付加部分の最悪実行時間 o_i は含まれない．

4.2 ソフトウェア構成

3.2 節で述べたように，プロセッサ温度が閾値を超過した場合に動作周波数および供給電圧を下げる手法ではタスクの実行時間の予測が困難でありリアルタイム性を維持できない．

そこで本研究では，リアルタイム性を維持しつつ TA-DVFS で動的に変動する余裕時間を活用するため，インプリサイス計算モデルの適用を行う．余裕時間が利用できる場合は付加部分を実行して生成結果の品質を向上させ，動作周波数が下がり余裕時間が不足した場合は付加部分を破

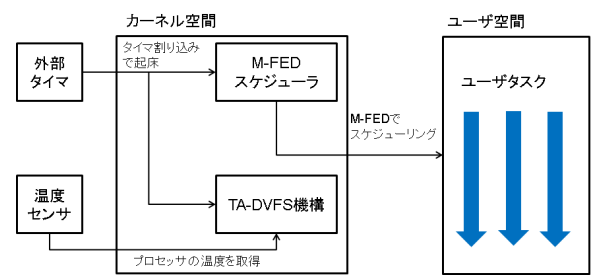


図 3 ソフトウェア設計

Fig. 3 Software design.

棄してリアルタイム性を維持する．

図 3 に本研究で扱うソフトウェア構成の関係図を示す．M-FED スケジューラおよび TA-DVFS の機構を実装し，それぞれを実行するカーネルタスクをプロセッサの動作周波数に影響されない外部クロックを用いたタイマが割り込みにより起床させる．TA-DVFS では温度センサでプロセッサ温度を読み取り，温度に応じてプロセッサの動作周波数および供給電圧を調整する．M-FED スケジューラは必須部分を実行中のユーザスレッドを持つタスクから優先的にスケジューリングし，残った余裕時間に対して付加部分を実行中のスレッドを持つタスクをスケジューリングする．以上のシステム構成により，TA-DVFS によってタスクの実行時間が変動するが，M-FED のスケジューラによりリアルタイム性を維持しつつ付加部分を実行することで余裕時間を有効に活用できる．

4.3 スケジューリング機構

周波数の変更によりタスクの実行時間が変化するため，TA-DVFS に適用するインプリサイス計算モデル向けのスケジューリングアルゴリズムは，可能な限りスケジューリング可能性が高いアルゴリズムが望ましい．そこで本研究では，動的スケジューリングアルゴリズムでありプロセッサ利用率 100% までスケジューリング可能な M-FED を用いる．TA-DVFS により動作周波数を下げるとタスクの実行時間が増加するが，M-FED は必須部分を優先的に EDF でスケジューリングし，付加部分の実行は中断することが可能なため，必須部分が最低動作周波数で EDF によってスケジューリング可能ならばリアルタイム性を満たすことができる．

図 4 に TA-DVFS へのインプリサイス計算モデル適用時のスケジューリングの例を示す．図中において τ_1 の 2 つ目のジョブのリリース時にプロセッサ温度が閾値を超過し，周波数および電圧を下げたことで以降のタスクの実行時間が増加している．プロセッサ温度の低い起動時には余った余裕時間を用いて付加部分を実行し，タスクの処理結果の品質を向上させている．一方プロセッサ温度が閾値を超えると周波数および電圧を下げるためタスクの実行時間が増加するが，付加部分を破棄することでリアルタイム性を維持しつつ温度を下げていく．

4.4 TA-DVFS の機構

本研究で用いる TA-DVFS の機構について述べる．図 5 に TA-DVFS で行う処理のフローチャートを示す．TA-DVFS の処理は専用のスレッドを生成して行う．一定周期ごとに TA-DVFS を行うスレッドを起床させ，温度センサからプロセッサ温度 θ を取得する．ここでプロセッサ温度が閾値 θ_{th} 以上である場合には，動作周波数と供給電圧を下げることでプロセッサの温度上昇の防止を図る．この閾値 θ_{th} はプロセッサが規定の温度制約を違反しないようにその値を定める．一方で θ が $\theta_{th} - \beta$ 以下であり，かつ動作周波数および供給電圧が下がっている場合には，動作周波数および供給電圧を元の値に戻す．ここで，プロセッサ温度が閾値 θ_{th} 付近で振動することを防ぐため，供給電圧を下げる温度の閾値と供給電圧を元に戻す温度の閾値との間で β の値だけ間隔をとっている．そして一連の TA-DVFS の処理が終わった後，スレッドは次の周期までコンテキストを退避する．以上のようにして，TA-DVFS の処理を行う．

この TA-DVFS の手法ではプロセッサ利用率ではなく温度を閾値として動作周波数および供給電圧を制御するため，プロセッサ利用率が高い場合はデッドラインミスが発生する可能性がある．3.2 節で述べたようにプロセッサ利用率

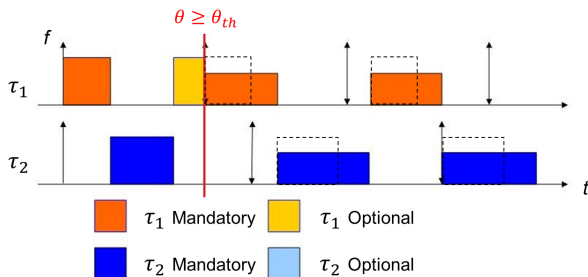


図 4 TA-DVFS における M-FED スケジューリング例
Fig. 4 M-FED scheduling example on TA-DVFS.

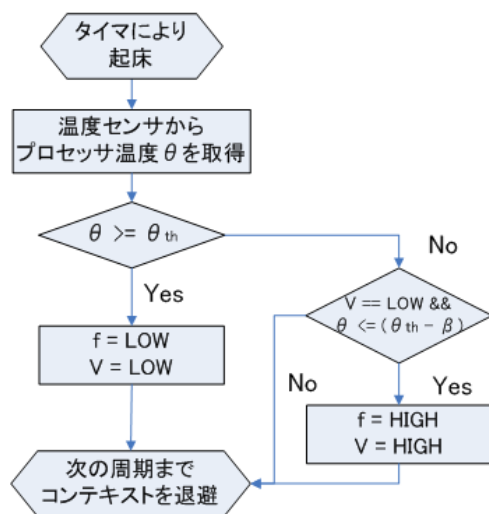


図 5 TA-DVFS の処理
Fig. 5 Routine of TA-DVFS.

を元に動作周波数と供給電圧の制御を行う TA-DVFS の手法もあるが，その場合は温度上昇の抑制はベストエフォートとなるのに加え，TA-DVFS へのインプリサイス計算モデルの適用にあたって消費電力削減と実行結果の品質向上のトレードオフを考慮し，それぞれへ余裕時間を割り当てる比重を決定するモデル定義が必要となる．よって，本論文の調査結果がこのモデル定義に貢献するものであると考え，消費電力と品質を考慮したモデル定義は論文の範囲外とし，温度を閾値とした TA-DVFS の手法を採用する．

5. 実装

本章では，実装対象のプロセッサおよびパッケージと，インプリサイス計算モデルを TA-DVFS に適用するための M-FED スケジューラの実装方法について述べる．

5.1 D-RMTP

本研究では，実機評価を行うための実装対象として Dependable Responsive Multithreaded Processor (D-RMTP) [11], [13], [14] を用いた．D-RMTP は 1 つのコアで最大 8 スレッドを同時実行する優先度付き SMT アーキテクチャであり，ハードウェア資源の利用効率が高いため低い動作周波数で高いスループットを実現できる．

5.1.1 D-RMTP SiP

本項では D-RMTP および TA-DVFS に必要な機構を搭載した System in Package (SiP) について述べる．図 6 は，D-RMTP と各種デバイスを集約した SiP の全体図である．SiP 上には 3.3 V から 1.0 V に減圧する DC/DC コンバータとポテンショメータが実装されており，SPI インタフェースを用いてポテンショメータを制御して抵抗値を変えることで，D-RMTP の供給電圧を 0.8~1.1 V の間で制御することができる．同様に，SPI を通じて熱電対を用いた温度センサの値を読み取ることで，プロセッサ温度を検出することができる．また，D-RMTP では各ハードウェアモジュールに供給するクロックの分周比を指定することができる分周器によってソフトウェアでの動的な周波数制御を可能としており，前述した供給電圧の制御および熱電対を用いた温度センサを組み合わせることで TA-DVFS を

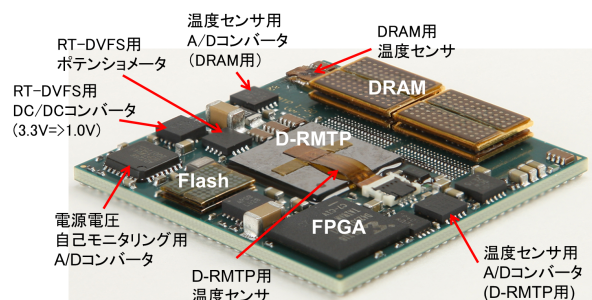


図 6 D-RMTP SiP
Fig. 6 D-RMTP SiP.

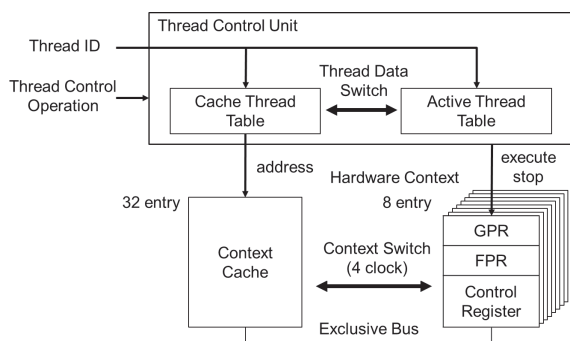


図 7 コンテキストキャッシュを使用したコンテキストスイッチ

Fig. 7 Context switch using context cache.

実現する.

5.1.2 コンテキストキャッシュ

D-RMTP ではスレッドのコンテキストを格納することのできる専用オンチップメモリとしてコンテキストキャッシュ [14] が実装されている. 通常コンテキストスイッチが発生すると, ソフトウェアによってメモリにコンテキストを退避させる必要があり, 大きなオーバーヘッドが発生する. コンテキストキャッシュはハードウェアコンテキストとのデータ転送を専用バスを介して行うことで, 4 クロックサイクルでコンテキストの転送を完了することを可能とする.

図 7 にコンテキストキャッシュを利用したコンテキストスイッチの概略を示す. 8 つのハードウェアコンテキストを割り当てるスレッドをアクティブスレッド, コンテキストキャッシュに格納されているスレッドをキャッシュスレッドと呼び, これらのスレッドは Thread Control Unit 内の 2 種類のスレッドテーブルによって管理されている. 各スレッドには識別用のスレッド ID が付与されており, アクティブスレッドとコンテキストキャッシュ間のコンテキストスイッチ時には D-RMTP 専用のスレッド制御命令を用いて, 入れ替えるスレッドのスレッド ID を Thread Control Unit に伝える. スレッド制御命令をうけとった Thread Control Unit はスレッド ID で内部に保持しているキャッシュスレッドのテーブルを検索し, コンテキストキャッシュにアクセスするためのアドレスを取得してアクティブスレッドとのコンテキストの入れ替えを行う. コンテキストスイッチで入れ替えるコンテキストは汎用レジスタ, 浮動小数点レジスタおよび制御レジスタの値であり, コンテキストキャッシュは 32 スレッド分のコンテキストを保持できる. これにアクティブスレッド数を加えた 40 スレッド以内であれば, メモリへのコンテキスト退避は不要である. 全スレッド数が 40 スレッドを超えた場合はコンテキストキャッシュに収まらないためメモリへの退避が必要となるが, 本研究では組込みリアルタイムシステムを対象としているため, 40 スレッド以内と仮定しても十分なスレッド数であるといえる.

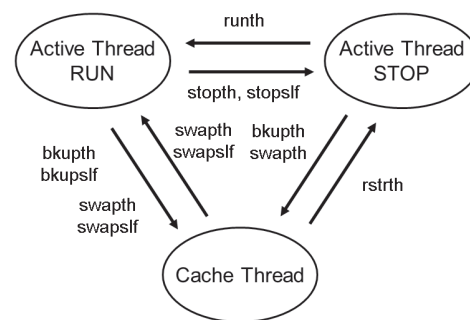


図 8 D-RMTP におけるスレッドの状態遷移図

Fig. 8 State transitions of threads on D-RMTP.

表 1 スレッド制御命令

Table 1 Thread-control instructions.

命令	内容
mkth	新たにスレッドを生成する
delth	指定したスレッドを削除する
runth	指定スレッドを Active Thread RUN 状態に遷移させる
stopth	指定スレッドを Active Thread STOP 状態に遷移させる
stopslf	自身を Active Thread STOP 状態に遷移させる
bkupth	指定スレッドをコンテキストキャッシュに退避する
bkupslf	自身をコンテキストキャッシュへ退避する
rsttrh	指定したキャッシュスレッドを復帰させる
swapth	アクティブスレッドとキャッシュスレッドを交換する
swapslf	自身と指定したキャッシュスレッドを交換する

図 8 に D-RMTP におけるスレッドの状態遷移図を, 表 1 にスレッド制御命令の内容を示す. プロセッサが実行するスレッドは, レジスタファイルやプログラムカウンタなどの資源が確保されており, かつ Active Thread RUN の状態にあるスレッドのみである. スレッドの状態は図 7 の Active Thread Table および Context Thread Table 内に保持されており, スレッド制御命令によって図 8 に示すように状態遷移が行われる. また, 以上のコンテキストキャッシュを用いたスレッド制御命令によるコンテキストスイッチの実装方法は, 文献 [15] ですでに述べられている.

5.2 M-FED スケジューラ

図 9 にスケジューリングを行う際にタスクを挿入するキューの構造を示す. Wait Queue (WQ) と優先度別の Run Queue (RQ) を管理しており, 必須部分を実行可能なタスクには高い優先度が, 付加部分を実行可能なタスクには低い優先度が必ず与えられる. スケジューラは RQ にあるタスクを EDF でスケジューリングする. ただし, 必ず優先度の高いタスクから実行するため, RQ 内に必須部分を実行可能なタスクが存在しない場合にのみタスクの付加部分は実行される. そして, 実行を完了したタスクはスケジューラによって次のリリース時刻まで WQ に格納される. また, 低優先度の RQ 内でデッドラインを迎えたタスクは, 付加部分の実行を中断して高優先度の RQ に移動した後, 必須部分の処理から実行を再開する. このようにし

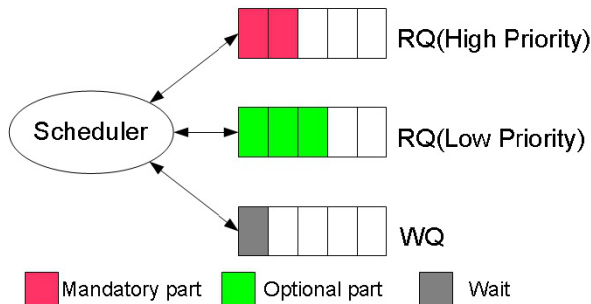


図 9 タスクキュー

Fig. 9 Task queue.

て、M-FED スケジューラを実現する。

以上で述べた実装は既存のインプリサイス計算モデル用タスクキューの実装方法 [16] を基にした、M-FED アルゴリズムのスケジューラへの実装方法であり、実装対象を選ばずに適用可能である。しかしながら、スケジューラにおけるスレッドのコンテキストスイッチの実装方法は実装対象のプロセッサに依存する。5.1.2 項で述べたように、本研究では実機評価における実装対象として D-RMTP を用いるため、スケジューラによりタスクを実行するスレッドのコンテキストスイッチの際にコンテキストキャッシュを用いてオーバヘッドを無視可能なサイクル数にまで削減することが可能である。これにより、コンテキストスイッチのオーバヘッドを無視した理論上のスケジューリングとほぼ同様のリアルタイムスケジューリングが実際のシステム上で実現可能である。特にインプリサイス計算モデルでは実行部分によってタスクの優先度が変わるため、一般計算モデルよりもコンテキストスイッチが多く発生する。よってインプリサイス計算モデルのタスクをスケジューリングする M-FED スケジューラにおいて、コンテキストキャッシュによるコンテキストスイッチのオーバヘッド削減は有効である。

このような M-FED スケジューラの実装により、TA-DVFS で周波数および電圧が変化して付加部分を実行する余裕時間がなくなったとしても、付加部分の実行を中断することで必須部分の実行のリアルタイム性を維持することが可能である。

5.3 インプリサイス計算モデルのタスク

図 10 にインプリサイス計算モデルの擬似コードを示す。タスクの実行状態を示す構造体メンバ `part` の初期値は MANDATORY である。まず、`execute_mandatory` 関数で必須部分を実行する。必須部分の実行完了後、`end_mandatory` 関数を呼び出す。`end_mandatory` 関数では、付加部分の実行時間を割当て可能かどうか判定を行う。付加部分の割当て時間がない場合は `part` の値は MANDATORY のままであり、付加部分は実行されず `end_task` 関数を実行する。付加部分の割当て時間がある場

```
execute_mandatory();
end_mandatory();
if (current_task->part == OPTIONAL)
    execute_optional();
end_task();
```

図 10 インプリサイス計算モデルの擬似コード

Fig. 10 Pseudocode of imprecise computation model.

```
void end_mandatory(void)
{
    task_t *task = current_task();
    if (task->deadline > current_time()) {
        deactivate_task(task, task->rq);
        task->part = OPTIONAL;
        task->priority += PRIORITY_OPTIONAL;
        activate_task(task, task->rq);
        schedule(task->rq);
    }
}
```

図 11 end_mandatory 関数

Fig. 11 end_mandatory function.

合は `part` の値は OPTIONAL となり `execute_optional` 関数で付加部分を実行し、付加部分の実行が完了した場合は `end_task` 関数を実行する。`end_task` 関数では `part` の値を MANDATORY に戻した後、タスクを次のリリース時刻まで WQ に格納する。

付加部分実行の判定および付加部分への遷移を行う `end_mandatory` 関数の擬似コードを図 11 に示す。`end_mandatory` 関数では、呼び出したタスクがデッドラインを過ぎていないか判定を行い、デッドラインを過ぎていなければ、タスクを一度 RQ から取り除き、タスクの実行状態を表す構造体メンバ `part` を OPTIONAL に、優先度を表す構造体メンバ `priority` を低優先度に変更して RQ に再び格納し、スケジューリングを行う。このとき、`priority` の値が高いタスクは低優先度、`priority` の値が低いタスクは高優先度である。

図 12 にデッドラインミスした付加部分の中断処理を行う関数の擬似コードを示す。`optional_deadline_check` 関数は RQ 内の付加部分を実行中の各タスクがデッドラインミスしていないか判定を行う。デッドラインミスしたタスクが必須部分を実行中だった場合、中断処理は行われず実行を継続する。付加部分を実行中のタスクがデッドラインミスしていた場合は、RQ から取り除いてそのタスクの実行状態 `part` を MANDATORY に、優先度 `priority` を高優先度に戻した後、タスクの実行結果を保存して実行スレッドを破棄する。タスクが周期タスクの場合は、スレッドを再

```

void optional_deadline_check(task_t *task)
{
    if (task->deadline <= current_time()) {
        deactivate_task(task, task->rq);
        task->part = MANDATORY;
        task->priority -= PRIORITY_OPTIONAL;
        thread_reset(task);
    }
}

```

図 12 optional_deadline_check 関数

Fig. 12 optional_deadline_check function.

び生成し，リリース時刻まで WQ に格納する。

6. 評価

本章では，まず D-RMTP のコンテキストキャッシュの予備評価をシミュレーションで行う．その後，TA-DVFS におけるインプリサイス計算モデル適用について，実機を用いて温度分布，平均温度，消費電力量，実行結果の品質，スケジューラビリティの評価を行う。

6.1 コンテキストキャッシュの予備評価

実機評価を行う前に，5.1.2 項で述べた，コンテキストキャッシュの有無によるコンテキストスイッチのオーバーヘッドへの影響についてシミュレーションによる予備評価を行う．Cadence 社の NC-Verilog を用いた Register Transfer Level (RTL) シミュレーションにより，D-RMTP 上でコンテキストキャッシュを用いずにレジスタ情報のスイッチを行った場合とコンテキストキャッシュを用いた場合のクロックサイクル数をそれぞれ測定する。

D-RMTP の構成を表 2 に示す．キャッシュは命令キャッシュおよびデータキャッシュを 32 KByte ずつ備え，SRAM は 64 KByte，SDRAM は 64 MByte を備えている．コンテキストキャッシュを用いない場合は，コンテキストキャッシュによって転送される対象である，整数レジスタ 32 本と浮動小数点レジスタ 8 本，そしてスレッドごとに固有の制御レジスタ 8 本の値をメモリ上に退避した後，コンテキストスイッチ先のスレッドの各レジスタの値を読み出すことでコンテキストスイッチを行う．本研究は組込みリアルタイムシステムを対象としているため，コンテキストスイッチに最低限必要なレジスタ情報の切替えにかかるクロックサイクル数のみ測定を行う。

表 3 にコンテキストキャッシュを使用しない場合とコンテキストキャッシュを使用した場合でコンテキストスイッチにかかるクロックサイクル数の測定結果を示す．表 3 が示すように，コンテキストキャッシュを使用しない場合はコンテキストスイッチに少なくとも 590 クロックサイクルかかり，キャッシュにヒットしない場合は最大 1,380 ク

表 2 D-RMTP の構成

Table 2 Outline of the D-RMTP.

Clock Frequency	31.25 MHz
Active Threads	8
Fetch Width	8
Integer Register	32-bit × 32-entry × 8-set
Integer Renaming Register	32-bit × 64-entry
FP Register	64-bit × 8-entry × 8-set
FP Renaming Register	64-bit × 64-entry
ALU	4 + 1 (Divider)
FPU	2 + 1 (Divider)
SIMD Units	1
FP Vector Units	1 (4-FPU × 2-line)
Branch Unit	2
Memory Access Unit	1

表 3 コンテキストスイッチのクロックサイクル数

Table 3 Clock cycles of context switch.

コンテキストキャッシュの使用	無	有
クロックサイクル数	(Cache Hit) 590	12
	(Cache Miss) 1,380	

ロックサイクルかかる．一方で，コンテキストキャッシュを使用した場合は 5.1.2 項で述べたスレッド制御命令の実行と 4 クロックサイクルでのレジスタ情報のスイッチにより合計 12 クロックサイクルでコンテキストスイッチを完了することができている．また，コンテキストキャッシュは専用バスを用いたレジスタ情報の転送を行うため，クロックサイクル数がキャッシュの影響を受けず予測性も高い．以上から，D-RMTP に実装されているコンテキストキャッシュが，コンテキストスイッチのオーバーヘッドを十分無視可能なクロックサイクル数まで削減し，理論上のスケジューリングとほぼ同様のリアルタイムスケジューリングを可能にするという結果が得られた．コンテキストキャッシュを持たない他プロセッサでは，コンテキストスイッチの度にメモリアクセスが必要であり少なくとも数百クロックサイクルのオーバーヘッドが発生してしまう．動作周波数の高い汎用のプロセッサでは数百から数千程度のクロックサイクル数は問題とならないが，動作周波数が低く細粒度の処理が求められる組込み用のプロセッサではコンテキストスイッチにおけるオーバーヘッドはタスクのスケジューリングに大きな影響を与える．特に，5.2 節で述べたタスクの付加部分の実行によるコンテキストスイッチの増加のほかにも，TA-DVFS による低周波数での命令の実行が起りうる本研究において，このコンテキストスイッチのオーバーヘッド削減は有効であるといえる。

6.2 測定環境

実機評価に用いる実機と測定環境について述べる．本研究では，D-RMTP を搭載した評価キットを用いて評価を

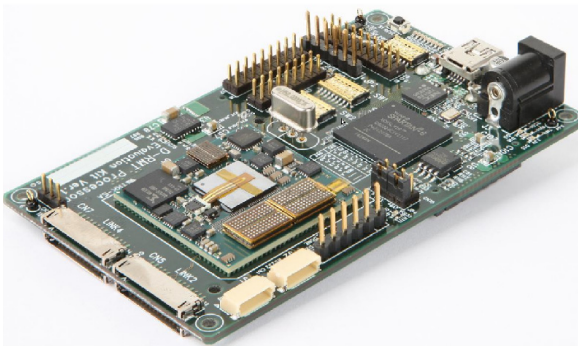


図 13 D-RMTP 用評価キット
Fig. 13 Evaluation kit for D-RMTP.

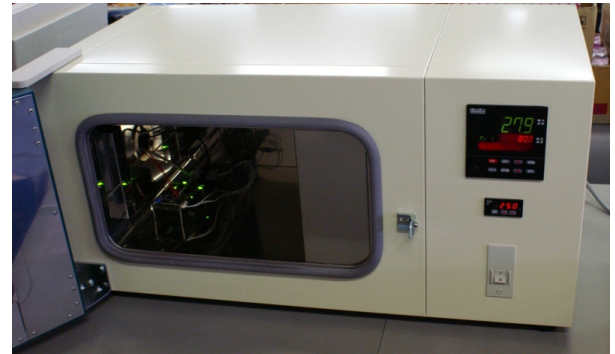


図 15 評価環境外観
Fig. 15 Evaluation environment.

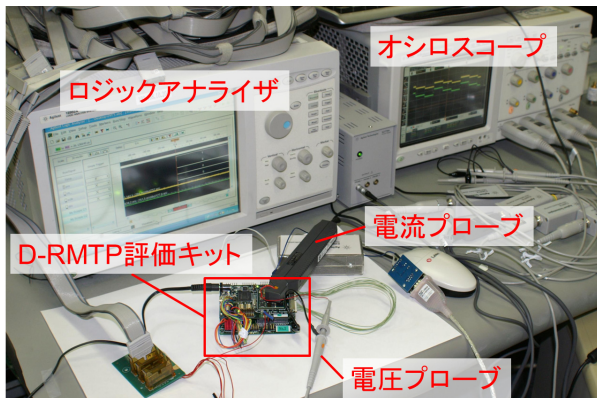


図 14 測定用機材
Fig. 14 Measuring machines.

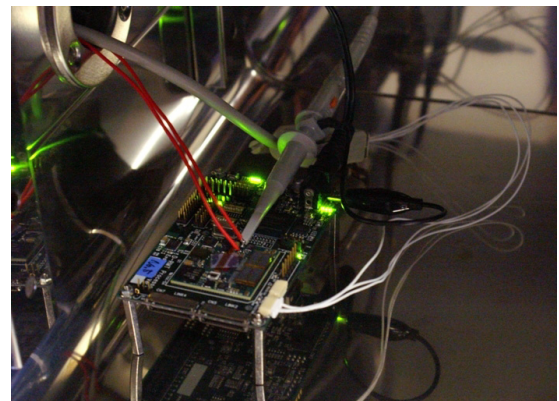


図 16 恒温槽内部
Fig. 16 Inside thermostatic oven.

行う。D-RMTP は優先度付き SMT アーキテクチャであるが、本研究ではシングルプロセッサでのスケジューリングを想定しているため、同時に実行するユーザスレッドは 1 スレッドのみとする。実機評価に用いる評価キットの外観を図 13 に示す。5.1.1 項で述べた、TA-DVFS を行う環境を実現するための各種デバイスを含めた SiP が搭載されている。評価キット上には SiP のほかに、各ハードウェアモジュールに対応するコネクタや I/O ピン、クロックを供給する水晶、FPGA などが搭載されている。また、図 14 に測定に用いた機器を、図 15 および図 16 に恒温槽を用いた評価の様子をそれぞれ示す。恒温槽の内部に評価キットを設置し、一定の環境温度で測定を行った。測定ではロジックアナライザとオシロスコープを同期しており、I/O ピンからの出力を共通のトリガーとして用いている。D-RMTP 上に用意した短絡ループに電流プローブおよび電圧プローブを取り付けて D-RMTP に供給される電流値と電圧値を測定する。測定した電流値と電圧値のサンプリング値に対して、測定時間で積分を行うことにより正確な消費電力量を求める。評価では恒温槽内に評価キットを設置して測定し、指定した温度環境で評価を行う。電圧プローブや電源などの必要なケーブルはケーブル孔を通して恒温槽内部の評価キットに接続する。電流プローブは高温による影響を考慮し、短絡ケーブルをケーブル孔を通して恒温槽外部で

取り付ける。

評価に用いた各機材は以下のとおりである。

- オシロスコープ：Agilent infiniium DSO 80404B
 - 電圧プローブ：10073C
 - 電流プローブ：N2783A
- ロジックアナライザ：Agilent 16902A
- 恒温槽：ISUZU VTEC-18

6.3 評価パラメータ

評価に用いた各パラメータについて本節で述べる。評価用のタスクセットとして、周期がそれぞれ 20 ms, 40 ms, 80 ms のタスク 3 つをシステムへ与えてシングルスレッド実行を行う。2.1 節で述べたように、インプリサイス計算モデルは主に画像処理やロボットの経路選択などのアプリケーションを想定している。このようなアプリケーションでは、座標情報の導出や変換など行列の乗算を用いる処理が多く行われる。よって想定するアプリケーションを考慮し、今回の評価に用いるタスクでは必須部分付加部分ともに正方行列の乗算を行う。タスクの実行中は行列の乗算を繰り返し行い、反復回数はそのタスク最悪の実行時間により定める。

次に、評価用タスクの最悪実行時間およびプロセッサ利用率について述べる。付加部分の処理は必須部分を実行中

のタスクが存在しない場合のみ実行可能なため、タスクの必須部分と付加部分の実行割合は特に実行結果の品質向上の評価に影響する。今回は各タスクの必須部分と付加部分の実行割合は1:1として評価を行った。そして、各評価にてシステム全体のプロセッサ利用率を与えることでタスクの最悪実行時間を決定する。ただし、2.1節で述べたようにインプリサイス計算モデルにおけるプロセッサ利用率は必須部分の最悪実行時間と周期により決定するため、このプロセッサ利用率に付加部分のプロセッサ利用率は含まれない。システム全体のプロセッサ利用率からタスクの必須部分の実行時間が定まり、必須部分と付加部分の実行割合は1:1としているため、同様に付加部分の実行時間も定まる。評価におけるプロセッサ利用率は10%から100%の範囲とする。前述したようにこのプロセッサ利用率には中断可能な付加部分は考慮されない。よって、付加部分を含めた場合のプロセッサ利用率は、20%から200%の範囲となる。

評価でのTA-DVFSにおけるパラメータについて述べる。動作周波数および供給電圧の設定の組合せは、HIGHの場合はそれぞれ31.25 MHzおよび1.1 Vであり、LOWの場合はそれぞれ15.62 MHzおよび0.88 Vである。前述したタスクセットに関するパラメータはすべて状態HIGHの値としている。TA-DVFSの処理の周期はタスクセットの周期の最小公倍数（ハイパーピリオド）である80 msとした。TA-DVFSの閾値 θ_{th} は環境温度から環境温度+10°Cの範囲として、1°Cごとに評価を行った。TA-DVFSの処理において元の電圧と周波数に戻す閾値の決定に用いられる β の値は、予備評価を行って以降の評価で用いる値を決定する。測定時間は1分間、測定のサンプリングレートは1,000 Sample/s、環境温度は60°Cおよび70°Cとして測定を行った。

最後に、評価における指標を定義する。通常タスクの実行結果における品質向上の度合いはタスクの処理内容により異なるが、今回の評価では品質向上の度合いは付加部分の演算が実行された割合に比例するものと仮定した。ここで付加部分の実行割合は、付加部分の全演算を実行して得られる演算結果のうち、実際に実行されて得られた演算結果の割合から求めた。また、トレードオフ解析のための評価関数を式(7)に示す。

$$F = \frac{Q}{P} \quad (7)$$

ここで Q は付加部分の実行割合、 P は消費電力量である。すなわち評価関数 F は消費電力量あたりの品質向上の度合いを表す。

6.4 β の予備評価

TA-DVFSにおいて供給電圧と動作周波数を下げる閾値 θ_{th} と元に戻す閾値との間隔である β の値を決定する。 β

表 4 β による評価への影響

Table 4 Influence on evaluation by β .

β [°C]	1	2	3	4	5
電圧・周波数変更回数	125	34	5	1	1
平均温度 [°C]	65.85	65.54	65.23	64.78	64.78
付加部分の実行割合	0.34	0.34	0.28	0.07	0.07
総消費電力量 [J]	25.19	20.82	16.38	15.62	15.62
評価関数 F [1/kJ]	13.49	16.33	17.09	4.48	4.48

の値が大きいほど供給電圧と動作周波数を元に戻す機会が少なくなり、 β の値が小さいほど供給電圧と動作周波数の変更回数が増加する。この β の値を変化させた場合の評価指標への影響を調べ、それを元に以降の評価で用いる β の値を決定する。ただし、付加部分をつねにすべて実行できるほどプロセッサ利用率が低い場合は、 β の値が大きいほど消費電力が削減され評価関数 F の値が大きくなることが明らかであるため、プロセッサ利用率は低周波数での動作時もデッドラインミスが発生しない最大の利用率である50%で測定を行った。

表 4 に β の値を閉区間 [1°C, 5°C] の範囲で変化させたときの電圧と周波数の変更回数と平均温度、付加部分の実行割合、総消費電力量、そして評価関数 F への影響の調査結果を示す。環境温度は60°C、 θ_{th} は65°Cとして測定を行った。

β の値が4以上の場合は電圧と周波数を元に戻すことができず、消費電力を削減できる代わりに付加部分はほとんど実行されない。一方で、 β の値が小さいほど供給電圧と動作周波数の変更回数は増加し、なおかつプロセッサ温度が高い値で振動することとなるため、付加部分の実行割合が増加する代わりに消費電力量も増大する。

よって、供給電圧の切替えが頻発してプロセッサ温度が振動することがなく、評価関数 F が最大となった $\beta = 3$ の設定を以降の評価で用いることとした。

6.5 温度変化の評価

TA-DVFSによりプロセッサの温度上昇を抑えられることを示す。プロセッサ利用率50%における環境温度60°Cおよび70°Cでのプロセッサ温度のヒストグラムをそれぞれ図 17 および図 18 に示す。ヒストグラムの横軸はプロセッサ温度を、縦軸は各温度が測定中に温度センサにより検出された頻度をそれぞれ表している。また、各 θ_{th} における環境温度60°Cおよび70°Cでのプロセッサの平均温度の評価をそれぞれ図 19 および図 20 に示す。平均温度の評価のグラフの横軸は θ_{th} を、縦軸は測定中のプロセッサ温度の平均値をそれぞれ表している。必須部分のプロセッサ利用率が50%以上の場合は、付加部分を含めるとプロセッサ利用率は100%以上であり、プロセッサはつねにいずれかのタスクを実行している状態となる。よってプロセッサ利用率が50%を超えても実際のプロセッサの負荷は

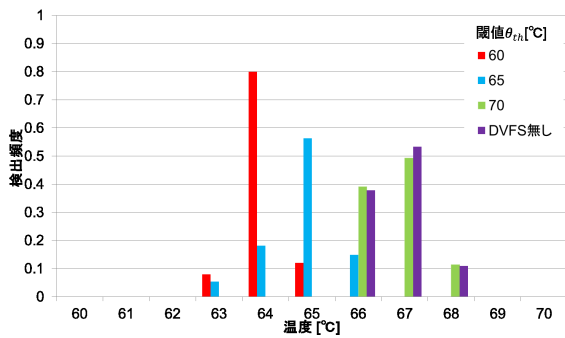


図 17 温度のヒストグラム (環境温度 60°C)

Fig. 17 Histogram of temperature (environmental temperature 60°C).

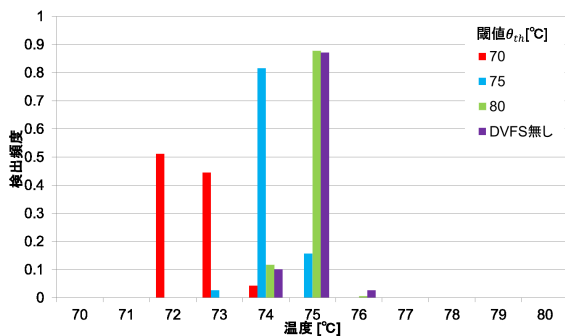


図 18 温度のヒストグラム (環境温度 70°C)

Fig. 18 Histogram of temperature (environmental temperature 70°C).

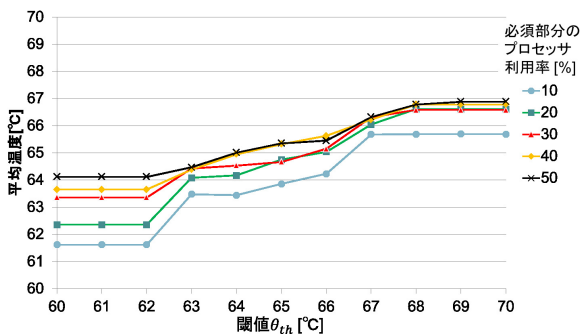


図 19 平均温度 (環境温度 60°C)

Fig. 19 Average of temperature (environmental temperature 60°C).

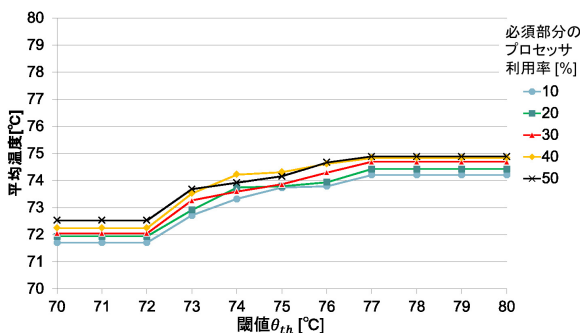


図 20 平均温度 (環境温度 70°C)

Fig. 20 Average of temperature (environmental temperature 70°C).

変化せず、消費電力量および温度変化がプロセッサ利用率 50%の場合と変わらないため、グラフからプロセッサ利用率 60%以上の場合の評価結果を省略した。

図 17 および図 18 より、TA-DVFS を行わない場合に対して、TA-DVFS を行った場合はプロセッサ温度がそれぞれ設定した θ_{th} 周辺に分布しており、 θ_{th} を超過した場合にそれ以上の過度な温度上昇を防げることが見て取れる。ただし、2.2 節の式 (3) で示したように、電圧と周波数の制御ではジャンクション温度が周辺温度を下回ることではないため、プロセッサ温度が環境温度未満となることはない。よって、 $\theta_{th} - \beta$ が環境温度未満となる場合は動作周波数と電圧を HIGH の状態に戻す条件が満たされることがないため、動作周波数と電圧はつねに LOW の状態となる。 $\beta = 3$ であるとき、環境温度 60°C では θ_{th} が 63°C 未満の場合、環境温度 70°C では θ_{th} が 73°C 未満の場合がこれに該当する。一方で、 θ_{th} が検出されるプロセッサ温度を超える場合は電圧と周波数を下げることがなくなるため、TA-DVFS を行わない場合と温度分布にほとんど差がない。また、図 19 および図 20 より、それぞれ θ_{th} が高くなると電圧と周波数を下げる機会が減ることでプロセッサの平均温度が上昇する傾向にあることが見て取れる。各プロセッサ利用率で比較した場合、プロセッサ利用率が高く、より多くの演算を行うほどにプロセッサの平均温度は高くなる傾向にあった。

6.6 消費電力量の評価

環境温度 60°C および 70°C において TA-DVFS に M-FED を適用した場合の各 θ_{th} における総消費電力量を図 21 および図 22 にそれぞれ示す。グラフの横軸は θ_{th} を、縦軸は測定中の総消費電力量をそれぞれ表している。6.5 節で述べたようにプロセッサ利用率が 50%を超えると消費電力量は変化しないため、グラフの煩雑化を防ぐためにプロセッサ利用率 60%以上の場合の評価結果は省いた。

図 21 および図 22 より、 θ_{th} が高いほど消費電力量は増加することが分かる。これは、 θ_{th} が高いほど TA-DVFS によって動作周波数と供給電圧を低下させる条件が厳しくなり、消費電力削減の効果が低下するためである。また、TA-DVFS により温度をトリガーとして消費電力を削減するため、 θ_{th} がある一定以上の値となると大きく総消費電力量が増加している。各プロセッサ利用率で比較すると、環境温度および上限付近の θ_{th} における総消費電力量の差は小さいが、プロセッサ利用率が低い方が演算量は少なく総消費電力量が小さい傾向にあった。一方で、プロセッサ利用率が低い場合の総消費電力量がプロセッサ利用率が高い場合の総消費電力量を大きく上回っている場合がある。これは、プロセッサ利用率が高い場合では θ_{th} 以上のプロセッサ温度となって電圧と周波数を下げて消費電力を削減するが、プロセッサ利用率が低い場合では θ_{th} 以上のプロ

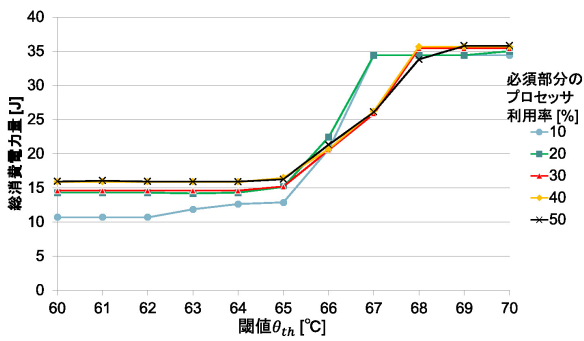


図 21 消費電力量 (環境温度 60°C)

Fig. 21 Energy consumption (environmental temperature 60°C).

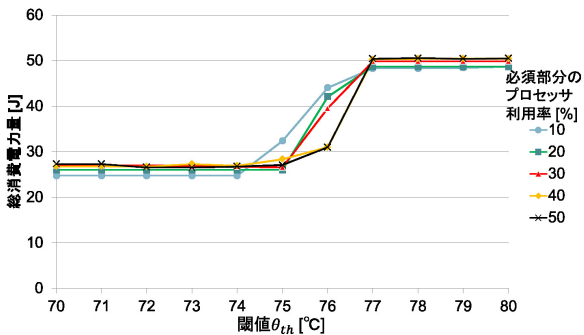


図 22 消費電力量 (環境温度 70°C)

Fig. 22 Energy consumption (environmental temperature 70°C).

セッサ温度とならずに消費電力削減が行われない場合があるためである．図 21 では $\theta_{th} = 67^\circ\text{C}$ での結果が，図 22 では $\theta_{th} = 76^\circ\text{C}$ での結果がこれに該当する．

6.7 実行結果の品質の評価

次に，環境温度 60°C および 70°C において TA-DVFS に M-FED を適用した場合の各 θ_{th} におけるタスクの実行結果の品質向上の度合いを図 23 および図 24 にそれぞれ示す．6.3 節で定義したように，付加部分の実行割合を品質向上の度合いと見なして評価を行う．グラフの横軸は θ_{th} を，縦軸は付加部分全体の演算のうち，実際に実行された割合をそれぞれ表している．

図 23 および図 24 より，TA-DVFS によりプロセッサ温度の過度な上昇を防ぎつつ，残った余裕時間で付加部分によるタスクの品質向上が得られている．30%以上のプロセッサ利用率では， θ_{th} が高いほど周波数と電圧を下げる機会が減り，余裕時間がより多く残るために付加部分の実行割合が増加する結果となった．また，プロセッサ利用率が低いほど利用できる余裕時間は増えるため，付加部分の実行割合が増加する結果となった．プロセッサ利用率が 60%以上の場合では，低動作周波数の状態ではプロセッサ利用率が 100%を超えるため， θ_{th} が一定以下の値となると余裕時間がなくなり付加部分の実行割合が 0 となった．

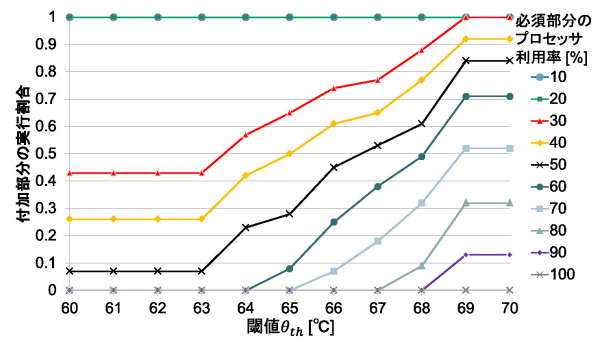


図 23 実行結果の品質 (環境温度 60°C)

Fig. 23 Quality of execution result (environmental temperature 60°C).

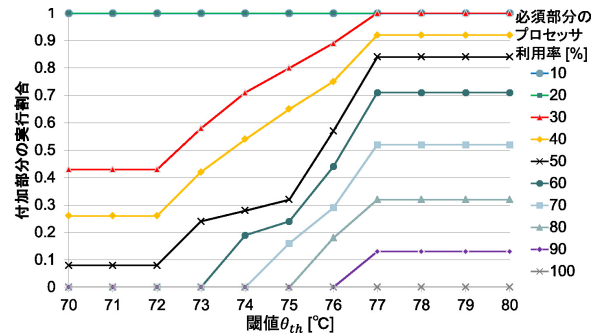


図 24 実行結果の品質 (環境温度 70°C)

Fig. 24 Quality of execution result (environmental temperature 70°C).

一方で，プロセッサ利用率が 20%以下の場合には，つねに TA-DVFS により周波数と電圧を下げていたとしても付加部分をすべて実行するのに十分な余裕時間が存在する．そのため，プロセッサ利用率が 20%以下では， θ_{th} によらずつねに付加部分の実行割合が 1 となった．

6.8 スケジューラビリティ評価

今回の評価では状態 LOW の動作周波数は状態 HIGH の約半分であるため，プロセッサ利用率が 50%を超えると動作周波数を下げた際にタスクのデッドラインミスが発生する可能性がある．そこで，プロセッサ利用率 60%以上の場合でスケジューラビリティの評価を行い，環境温度 60°C および 70°C における評価結果を図 25 および図 26 にそれぞれ示す．グラフの横軸は θ_{th} を，縦軸はスケジューラビリティをそれぞれ表している．スケジューラビリティはリリースしたうちデッドラインまでにタスクの必須部分の実行を完了できた割合を表す．4.1 節で述べたように，中断可能なタスクの付加部分の実行はスケジュールの成否に含まれない．

図 25 および図 26 より， θ_{th} の値が低く状態 LOW になる機会が多いほどスケジューラビリティが低下していることが見て取れる．3.1 節で述べたように M-FED はプロセッサ利用率 100%までスケジュール可能なため， θ_{th} の値

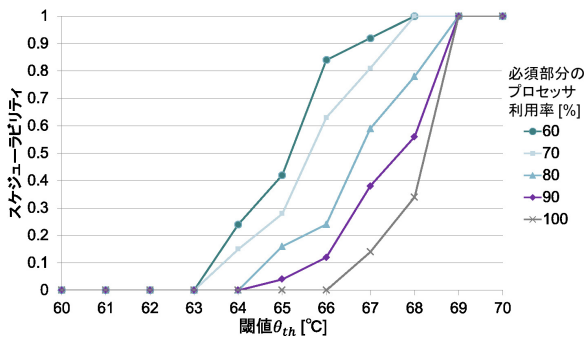


図 25 スケジューラビリティ (環境温度 60°C)

Fig. 25 Schedulability (environmental temperature 60°C).

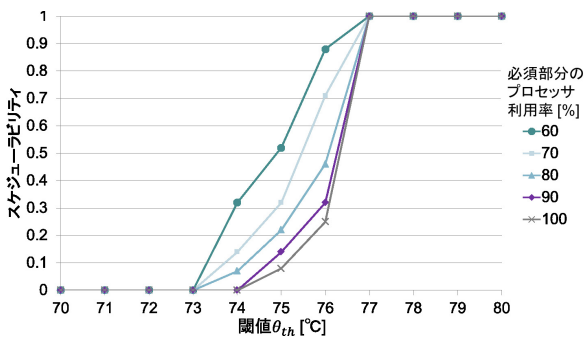


図 26 スケジューラビリティ (環境温度 70°C)

Fig. 26 Schedulability (environmental temperature 70°C).

が高くつねに状態 HIGH のままの場合はスケジューラビリティは 1 となる。一方で、 θ_{th} の値が低くつねに状態 LOW となる場合は、プロセッサがつねに過負荷となるため、スケジューラビリティは限りなく 0 に近づく。図 17 および図 18 から、TA-DVFS を行わなかった場合に検出された最大温度は環境温度 60°C では 68°C、環境温度 70°C では 76°C であったため、 θ_{th} の値がこれを超える場合ではつねに状態 HIGH である。また、6.5 節で述べたように、 θ_{th} の値が環境温度 60°C では 63°C 未満、環境温度 70°C では 73°C 未満の場合はつねに状態 LOW となる。そして、実行中に TA-DVFS による状態 HIGH と LOW の切替えが発生する場合は、状態 LOW となった際の一時的なプロセッサの過負荷を状態 LOW から HIGH に戻した後の余裕時間で処理することになる。よって、 θ_{th} の値が低く状態 LOW となる時間が長いほど、プロセッサ利用率が高いほど過負荷を処理しきれなくなりスケジューラビリティは低くなる。

6.9 トレードオフ解析

6.6 節と 6.7 節で示した評価結果より、 θ_{th} が高いほど TA-DVFS が余裕時間を利用する機会が減り消費電力削減の効果は低下するが、付加部分の処理に利用可能な余裕時間は増加する。すなわち、消費電力削減と実行結果の品質はトレードオフの関係にある。そこで、今回の評価環境において消費電力あたりの品質向上の度合いが最も高い値

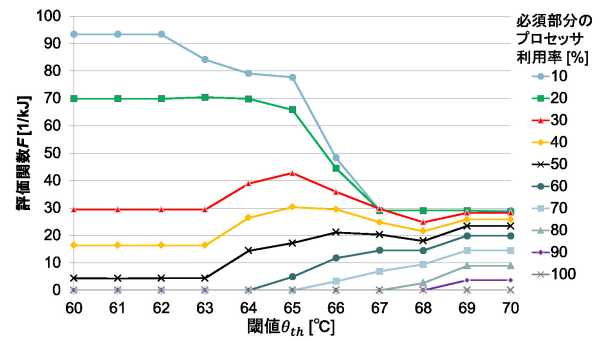


図 27 消費電力量と実行結果の品質の関係 (環境温度 60°C)

Fig. 27 Relation between energy consumption and quality of execution result (environmental temperature 60°C).

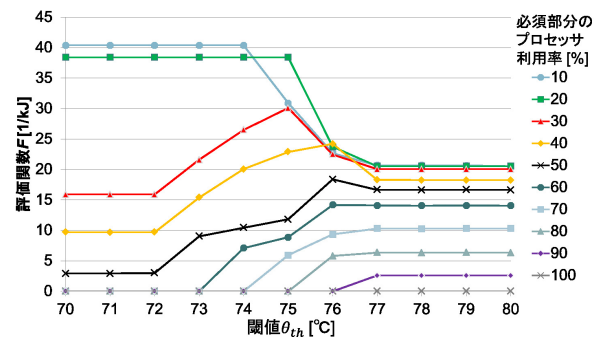


図 28 消費電力量と実行結果の品質の関係 (環境温度 70°C)

Fig. 28 Relation between energy consumption and quality of execution result (environmental temperature 70°C).

を得られる θ_{th} の値を求めるために、6.2 節で定義した評価関数 F を用いる。

環境温度 60°C および 70°C において各 θ_{th} における評価関数 F の推移を図 27 および図 28 に示す。グラフの横軸は θ_{th} を、縦軸は評価関数 F をそれぞれ表している。

プロセッサ利用率 20% 以下の場合のように、動作周波数と供給電圧が下がってもなお付加部分をすべて実行できるほどプロセッサ利用率が低い場合においては、品質が変化しないため θ_{th} を低くしてつねに動作周波数と供給電圧を下げたほうが消費電力あたりの品質は高い値を得られた。一方で、プロセッサ利用率 30% 以上においては、消費電力量が大幅に上昇する直前の θ_{th} 付近で評価関数 F が最大となる傾向にあるという結果が得られた。ただし、プロセッサ利用率が高くなる程に消費電力量の変化は緩やかに、品質向上の度合いの変化は急となっているために、プロセッサ利用率が高くなるにつれて消費電力の削減が行われないほど高い θ_{th} での評価関数 F の値が他の θ_{th} での値と比べて高くなる傾向にある。そのため、図 27 におけるプロセッサ利用率 50% 以上の場合では、消費電力が大幅にあがる直前の θ_{th} である 65°C 付近ではなく、 $\theta_{th} = 70^\circ\text{C}$ で評価関数 F が最大となっている。

以上のようにして、消費電力あたりで最大の品質が得られる θ_{th} の設定を導出可能であることを示した。ただし、

実際には付加部分の実行によるタスクの処理結果の品質および精度向上の度合いはタスクの処理内容に依存するため、その度合いを考慮して θ_{th} を定める必要がある。2.3 節で述べたように、ヒューマノイドロボットの制御などでは温度制御と処理結果の品質向上の両立が求められる。そのようなシステムにおいては、今回の評価のようにトレードオフを解析したうえで、消費電力量を削減しつつ可能な限り品質向上を図ることが非常に重要である。

7. 結論

本研究では、TA-DVFS においてリアルタイム性を維持しつつ動的に変化する余裕時間を活用する方法として、インプリサイス計算モデルの適用を行った。M-FED スケジューラを実装し、TA-DVFS を実現可能な環境で実機評価を行った。

評価では、実機上で TA-DVFS への M-FED の適用を行い、過度な温度上昇を防ぎつつタスクの実行結果の品質を向上できることを示した。また、評価結果からタスクの実行結果の品質と消費電力量のトレードオフについて調査した。消費電力量あたりの実行結果の品質を算出し、実行結果の品質向上の度合いと消費電力量の観点から適切な TA-DVFS の閾値を決定可能であることを示した。

今後の課題としては、今回得られた調査結果を元に消費電力とタスクの実行結果の品質、そして実行時間との間にあるトレードオフの関係をふまえた統合的なスケジューリング手法の提案を行う予定である。このような場合タスクの品質の定義や消費電力と品質の比重など、実タスクにおける要求を考慮することが必要であるため、それらを反映するためのシステムモデルの提案も行う予定である。また、インプリサイス計算モデルを適用可能な一般的なベンチマークの一例として H.264 のデコードがあげられる。H.264 の I フレームと P フレームのデコードを必須部分とし、B フレームのデコードを付加部分とすることでインプリサイス計算モデルを適用することが可能である。今後は、このような一般的なベンチマークを評価において利用する予定である。

謝辞 本研究は科学技術振興機構 CREST の支援によるものであることを記し、謝意を表す。

参考文献

- [1] Aydin, H., Melhem, R., Mosse, D. and Mejfa-Alvarewz, P.: Optimal Reward-Based Scheduling of Periodic Real-Time Tasks, *Proc. 20th IEEE Real-Time Systems Symposium*, pp.79–89 (1999).
- [2] Aydin, H., Melhem, R., Mosse, D. and Mejfa-Alvarewz, P.: Optimal Reward-Based Scheduling for Periodic Real-Time Tasks, *IEEE Trans. Computers*, Vol.50, No.2, pp.111–130 (2001).
- [3] Baruah, S.K. and Hickey, M.E.: Competitive On-line Scheduling of Imprecise Computations, *IEEE Trans.*

- Computers*, Vol.47, pp.1027–1033 (1996).
- [4] Gupta, N. and Mahapatra, R.: Temperature-Aware Energy Management for Real-Time Scheduling, *International Symposium on Quality Electronic Design*, pp.91–96 (2011).
- [5] Lee, J.S., Skadro, K. and Chu, S.W.: Predictive Temperature-Aware DVFS, *IEEE Trans. Computers*, Vol.59, No.1, pp.127–133 (2010).
- [6] Lehoczky, J., Sha, L. and Ding, Y.: The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior, *Proc. 20th IEEE Real-Time Systems Symposium*, pp.166–171 (1989).
- [7] Li, D., Chang, H.-C., Pyla, H.K. and Cameron, K.W.: System-level, Thermal-aware, Fully-loaded Process Scheduling, *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp.1–7 (2008).
- [8] Lin, K.-J., Natarajan, S. and Liu, J.W.: Imprecise Results: Utilizing Partial Computations in Real-Time Systems, *Proc. 8th IEEE Real-time Systems Symposium*, pp.210–217 (1987).
- [9] Liu, C. and Layland, J.: Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment, *Journal of the ACM*, Vol.20, pp.46–61 (1973).
- [10] Liu, J.W.S., Shih, W.-K., Lin, K.-J. and Bettati, R.: Imprecise Computations, *Proc. IEEE*, Vol.82, No.1, pp.83–94 (1994).
- [11] Suito, K., Ueda, R., Fujii, K., Kogo, T., Matsutani, H. and Yamasaki, N.: Dependable Responsive Multithreaded Processor for Distributed Real-Time Systems, *IEEE Micro*, Vol.32, No.6, pp.52–61 (2012).
- [12] Urata, J., Nakanishi, Y., Okada, K. and Inaba, M.: Design of High Torque and High Speed Leg Module for High Power Humanoid, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.4497–4502 (2010).
- [13] Yamasaki, N.: Design Concept of Responsive Multithreaded Processor for Distributed Real-Time Control, *Journal of Robotics and Mechatronics*, Vol.16, pp.194–199 (2004).
- [14] Yamasaki, N.: Responsive Multithreaded Processor for Distributed Real-Time Systems, *Journal of Robotics and Mechatronics*, Vol.17, pp.130–141 (2005).
- [15] 船岡健司, 加藤真平, 山崎信行: Pfair スケジューリングにおけるコンテキストキャッシュの有効利用, 情報処理学会論文誌, Vol.48, No.SIG 3, pp.1–12 (2007).
- [16] 千代浩之, 武田 瑛, 船岡健司, 山崎信行: Rate Monotonic に基づく拡張インプリサイスタスク用リアルタイムスケジューリング, 情報処理学会論文誌, Vol.52, No.8, pp.2365–2377 (2011).



溝谷 圭悟

2013 年度慶義塾大学理工学部情報工学科卒業。現在、同大学大学院理工学研究科開放環境科学専攻修士課程に在籍。リアルタイムシステムにおける温度制御、省電力化技術の研究に従事。



上田 陸平 (学生会員)

2012 年慶應義塾大学理工学部情報工学科卒業。2014 年同大学大学院理工学研究科開放環境科学専攻修士課程修了。リアルタイムシステムにおける省電力化技術の研究に従事。



高須 雅義

2013 年慶應義塾大学理工学部情報工学科卒業。現在、同大学大学院理工学研究科開放環境科学専攻修士課程に在籍。リアルタイムシステムにおける省電力化技術の研究に従事。



千代 浩之 (正会員)

2008 年慶應義塾大学理工学部情報工学科卒業。2012 年同大学大学院理工学研究科開放環境科学専攻博士課程修了。博士(工学)。2012 年度より 2013 年度まで慶應義塾大学理工学部情報工学科訪問研究員、日本学術振興会特別

研究員 (PD)。現在、慶應義塾大学理工学部情報工学科助教。リアルタイムシステム、オペレーティングシステム、分散ミドルウェア等の研究に従事。IEEE 会員。



松谷 宏紀 (正会員)

2004 年慶應義塾大学環境情報学部卒業。2008 年同大学大学院理工学研究科後期博士過程修了。博士(工学)。2009 年度より 2010 年度まで、東京大学大学院情報理工学系研究科特別研究員、日本学術振興会特別研究員

(SPD)。現在、慶應義塾大学理工学部情報工学科専任講師。コンピュータアーキテクチャとインターコネクトに関する研究に従事。IEEE、電子情報通信学会各会員。



山崎 信行 (正会員)

1991 年慶應義塾大学理工学部物理学科卒業。1996 年同大学大学院理工学研究科計算機科学専攻博士課程修了。博士(工学)。同年電子技術総合研究所入所。1998 年 10 月慶應義塾大学理工学部情報工学科助手、同専任講師、

同助教授を経て、2013 年 4 月同教授。リアルタイムシステム、プロセッサアーキテクチャ、並列分散処理、システム LSI、ロボティクス等の研究に従事。日本ロボット学会、IEEE 各会員。