Vol.2011-SLDM-149 No.24
Vol.2011-EMB-20 No.24
2011/3/18

RMT プロセッサにおける実行効率を向上するリアルタイムスケジューリング

吉住　修[†1]　石川洋輔[†2]　山崎信行[†2]

SMT プロセッサでは，同時に実行するタスク間でハードウェア資源の競合が発生し，実行効率が変動するため，リアルタイムスケジューリングが容易ではない．また，同時に実行するタスクによってスループットが低下してしまう可能性がある．Kato らが提案した U-link スケジューリングは，SMT プロセッサにおいて同時に実行するタスクを実行効率が最大となる組合せに固定することで実行効率の変動を抑制する．しかしながら，U-link スケジューリングではスレッド上でタスクが実行されない時間（アイドル時間）が発生するためスループットの向上が制限されてしまう．本研究では，RMT プロセッサが持つ優先度付き SMT 機構を利用しアイドル時間を削減することで，U-link スケジューリングと比較して，スループットを向上すると同時に，安定した実行効率を実現する．評価の結果，提案手法が U-link スケジューリングと比較して，実行時間の変動を抑制しつつ，タスク拒否率を最大 10%削減できた．

Bounding Execution Time of Real-Time Tasks with Multi-Case Execution Time on RMT Processor

Osamu Yoshizumi,[†1] Yousuke Ishikawa[†2] and Nobuyuki Yamasaki[†2]

It is not straightforward to schedule tasks on SMT processors, since the execution efficiency fluctuate due to resource competition. Also, some conbinations of co-scheduled tasks cannot increase throughput efficiently. U-link Scheduling proposed by Kato bounds the fluctuation of the execution efficiency by limiting the combination of the co-scheduled tasks. However, throughput is limited by time does not run any task on a thread(Idle Time) in U-link Scheduling. In this paper, We realize stable execution time and high throughput by reducing Idle Time compared to U-link Scheduling by using Prioritized SMT on RMT processor.The evaluation result shows that proposed method bounded the fluctuation of execution time, then reduce task rejection ratio by up to 10 % compared to U-link Scheduling.

1. はじめに

多くの組込みシステムでは，リアルタイム性だけでなく，スループットも重要になる．スループットを向上させる手段として動作周波数を上げることが考えられるが，組込みシステムでは消費電力も考慮しなければならない．そのため，近年では Simultaneous Multithreading（SMT）[1] や Chip Multiprocessor（CMP）[2] などを用いたプロセッサが主流となっている．特に SMT は 1 つのコアにおいて複数スレッドでハードウェア資源を共有することで CMP より資源の利用効率を上げることができるため，低い動作周波数で高いスループットを実現できるが，ハードウェア資源の競合によって実行効率が変動するので，リアルタイムタスクのスケジューリングが容易ではない．リアルタイムスケジューリングにおいて，時間予測性は非常に重要であり，最悪実行時間に基づいたスケジューリングは，実際の実行時間と最悪実行時間の差がより大きく広がり問題となるため，SMT の特性を考慮したスケジューリング手法が必要となる．Kato らが提案した U-link スケジューリング[3] は，SMT プロセッサにおいて同時に実行するタスクを実行効率が最大となる組合せに固定することで，実行時間の変動を抑制すると同時にスループットを向上させる．しかしながら，U-link スケジューリングではスレッド上でタスクが実行されない時間（アイドル時間）が発生するためスループットの向上が制限されてしまう．本研究では，RMT プロセッサが持つ優先度付き SMT 機構を利用し，高スループットを実現すると同時に，安定した実行時間を実現する．提案手法では，優先度付き SMT 機構においてタスクが実行されるスレッドに適切な優先度を付与することでアイドル時間を削減する．

本研究の構成は，以下の通りである．第 2 章では，関連研究について述べる．第 3 章では，本研究が対象とするシステムモデルについて述べる．第 4 章では，本研究で提案する RMT プロセッサ用 U-link スケジューリングについて述べる．第 5 章では，RMT プロセッサ用 U-link スケジューリングの有用性を評価する．最後に，第 6 章では結論と今後の課題を述べる．

†1 慶應義塾大学理工学部情報工学科
Department of Information and Computer Science, Faculty of Science and Technology, Keio University
†2 慶應義塾大学大学院理工学研究科開放環境科学専攻
Department of Computer Science, Graduate School of Science and Technology, Keio University

Vol.2011-SLDM-149 No.24
Vol.2011-EMB-20 No.24
2011/3/18

2. 関連研究

SMT プロセッサにおけるハードリアルタイムシステム向けのスケジュール方式として，Kato らは U-link スケジューリング[3] を提案した．U-link スケジューリングは，SMT プロセッサにおいて，スループットを向上するために，同時に実行するタスクを実行効率が高くなる組合せに固定し，固定された組合せを同時実行セットと定義する．同時実行セットは周期タスクのシステム使用率（予測実行時間 ÷ 周期），及び実行効率に基づいて構築される．同時実行セット内の軸となるタスクの集合を軸セット（Axis set）と定義し，軸スレッドと呼ばれる特定のスレッドで実行される．スケジューラは軸スレッドで実行するタスクを軸セットより選択することで実行するべき同時実行セットを判別し，軸タスク以外のタスク（一般タスク）を自動選択する．これにより，競合する可能性があるハードウェア資源を制限し，実行効率の変動を緩和することができると同時に，実行効率が高くなるタスクの組合せで実行することができ，安定して高いスループットを保つことができる．**図 1** はタスク $\tau_1 \sim \tau_4$ を 2 スレッドで実行した様子を示す．軸スレッドは $Thread1$ であり，実行効率はシングルスレッドでの実行における IPC に対する相対 IPC と定義する．IPC は，単位時間に実行する命令数である．図 1 において同時実行セットは (τ_1, τ_2)，(τ_3, τ_4) であり，同時実行セット内のタスクが常に同時に実行していることがことを示す．しかしながら，$Thread2$ でどのタスクも実行しない時間（アイドル時間）が発生する．これは，同時実行セット内のタスクごとのシステム使用率が異なるためである．アイドル時間の発生により，スループットが低下する．

我々はリアルタイムシステムにおける優先度の概念を SMT に応用した優先度付き SMT 機構を持つ RMT プロセッサ[4] を開発している．RMT プロセッサはスレッド毎に優先度を設定でき，スレッド間で資源の競合が発生した際には，優先度にしたがってスレッドに資源を割当てる．提案手法である RMT プロセッサ用 U-link スケジューリングは，Kato らが提案した U-link スケジューリングに優先度付き SMT 機構による優先度制御を付加することでアイドル時間を削減し，U-link スケジューリングより高いスループットを実現する．

3. システムモデル

本研究で用いるプロセッサには，M 個の論理プロセッサ $LP_1 \sim LP_M$ が存在する．演算器等の機能ユニットに関しては，汎用性を考慮して複数の種類があり，それぞれ $\lambda_1 \sim \lambda_L$ と表現する．機能ユニット λ_l の数を $N(\lambda_l)$，機能ユニット λ_l のレイテンシを $D(\lambda_l)$ とする．また，各ユニットの個数は，各ユニットの並列実行可能な命令数を指している．

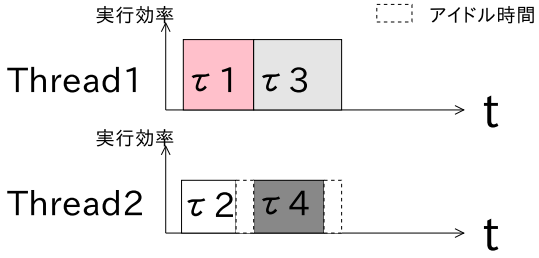

図 1 U-link スケジューリングの実行例
Fig. 1 Example of U-link Scheduling

システム実行開始時までにタスクセット τ が与えられている．τ は，1∼N までの周期タスク（$\tau_1 \sim \tau_N$）の集合である．すべてのタスクの周期は互いに整数倍であるとし，システム実行中に新たなタスクが挿入されることはない．各同時実行セットは $\tau_1^{'} \sim \tau_K^{'}$ とし，同時実行セットの集合を $\tau^{'}$ と定義する．同時実行セット $\tau_k^{'}$ に属する i 番目のタスクを $\tau_{k,i}^{'}$ と表す．それぞれの周期タスクは，独立に処理を行い，どの時点でもプリエンプション可能である．各タスク τ_i は，(C_i, T_i, I_i) というタプルで表現される．C_i は，シングルスレッドにおける実行時間を示す．T_i はタスクの周期であり，相対デッドラインでもある．I_i は各実行ユニット λ_l を利用する命令数を返す関数であり，事前にシステム設計者が指定するものとする．τ_i のシングルスレッド実行時のシステム使用率は $U_i = C_i/T_i$ で表し，システム負荷は $U(\tau) = \sum_{\tau_i \in \tau} U_i$ と定義する．RMT 実行時のシステム使用率は $U_i^{'}$ と表す．同時実行セットセット $\tau_k^{'}$ の軸タスク $\tau_{k,A}$ のシングルスレッドにおけるシステム使用率を $U_{k,A}$ と表す．すべてのタスク $\tau_i (\tau_i \in \tau)$ には，$P(\tau_i)$ が与えられる．$P(\tau_i)$ はタスク τ_i が実行されるスレッドに付与される優先度を示す．優先度は値が大きいほど高優先度である．

4. RMT プロセッサ用 U-link スケジューリング

本章では，RMT プロセッサにおいてリアルタイム性を保証しつつ高スループットを実現するため，RMT プロセッサにおける実行効率を考慮したリアルタイムスケジューリングを提案する．提案手法である RMT プロセッサ用 U-link スケジューリングは，Kato らが提案した U-link スケジューリングに優先度付き SMT 機構における優先度制御を付加することでアイドル時間を削減し，U-link スケジューリングより高いスループットを実現する．**図 2**

Vol.2011-SLDM-149 No.24
Vol.2011-EMB-20 No.24
2011/3/18

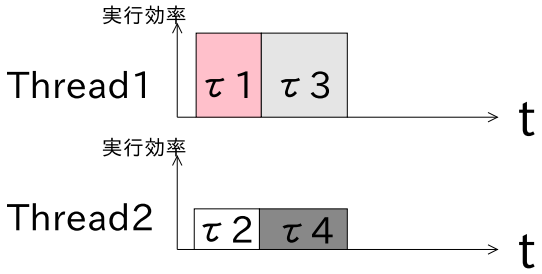

図 2 RMT プロセッサ用 U-link スケジューリングの実行例
Fig. 2 Example of U-link Scheduling for RMT

は RMT プロセッサ用 U-link スケジューリングの実行例を示す．図 1 の U-link スケジューリングの実行例ではアイドル時間が発生しているが，図 2 では優先度制御により適切な実行効率で実行することで，アイドル時間を削減できていることを示す．

4.1 RMT 実行におけるシステム使用率の計算

RMT 実行におけるシステム使用率は，相対命令使用率，ユニット競合率，実行効率を基に求められる．相対命令使用率，実行効率を求める式は U-link スケジューリングと同様である．また，ここで求めるシステム使用率は最悪値である．

- 相対命令使用率（$R_l(\tau_i)$）
  タスク τ_i の全命令中で，機能ユニット λ_l を使用する命令の割合を，使用率に関しての軸タスク $\tau^{'}_{k,A}$ より正規化したものである．これにより，一定時間内で，τ_i が実行する命令のうち，機能ユニット λ_l を利用する命令の割合を求めることができる．

$$R_l(\tau_i) = \Big(I_i(\lambda_l) / \sum_{m=1}^{L} I_i(\lambda_m)\Big) \times (U_i / U_{k,A}) \tag{1}$$

- ユニット競合率（$F_l(\tau^{'}_{k,i})$）
  ユニット競合率は，タスク $\tau^{'}_{k,i}$ に対する機能ユニット λ_l の利用低下率を示したものであり，式（2） より求められる．ユニット競合率は 1 以上の値であり，1 の時は競合が発生しないことを示す．

$$F_l(\tau^{'}_{k,i}) = max\{1, \Big(\int_0^{R_l(\tau^{'}_{k,i})} A(r)dr + High(\tau^{'}_{k,i})\Big) / R_l(\tau^{'}_{k,i})\} \tag{2}$$

$$A(r) = max\{1, Co(r)/N(\lambda_l)\} + Low(r) \tag{3}$$

$$Co(r) = \sum_{\tau_j \in \tau^{'}_k} c(\tau_j) \begin{cases} c(\tau_j) = 1 & : ((P(\tau_j) = P(\tau^{'}_{k,i})) \wedge (R_l(\tau_j) \geq r)) \\ c(\tau_j) = 0 & : ((P(\tau_j) \neq P(\tau^{'}_{k,i})) \vee (R_l(\tau_j) < r)) \end{cases} \tag{4}$$

$$Low(r) = \begin{cases} 0 & \Big(r > \Big(\sum_{(\tau_j \in \tau^{'}_k) \wedge (P(\tau_j) < P(\tau^{'}_{k,i}))} R_l(\tau_j)\Big) / N(\lambda_l)\Big) \\ 1 & \Big(r \leq \Big(\sum_{(\tau_j \in \tau^{'}_k) \wedge (P(\tau_j) < P(\tau^{'}_{k,i}))} R_l(\tau_j)\Big) / N(\lambda_l)\Big) \end{cases} \tag{5}$$

$$High(\tau^{'}_{k,i}) = (\sum_{(\tau_j \in \tau^{'}_k) \wedge (P(\tau_j) > P(\tau^{'}_{k,i}))} R_l(\tau_j)) / N(\lambda_l) \tag{6}$$

$A(r)$ は式（3）で表すものとし，相対命令使用率 r においての，優先度が $\tau^{'}_{k,i}$ 以下のタスクからの競合の程度を示す．$A(r)$ は関数 $Co(r)$，$Low(r)$ 及び $N(\lambda_l)$ から表される．$Co(r)$ は相対命令使用率 r において $\tau^{'}_{k,i}$ と優先度が等しいタスクの数を表す．$\tau^{'}_{k,i}$ と優先度が等しいタスクの数が多いほど競合によるコストが増加し，$N(\lambda_l)$ 以下になると競合は発生しない．$Low(r)$ は相対命令使用率 r における，$\tau^{'}_{k,i}$ より優先度が低いタスクからの競合の程度を表す．低優先度タスクからの影響は機能ユニット λ_l を低優先度のタスクがすでに使用している場合にのみ発生する．よって，$\tau^{'}_{k,i}$ より優先度が低いタスクからの競合による影響は，$\tau^{'}_{k,i}$ と優先度が等しいタスクからの競合による影響に比べて制限される．$High(\tau^{'}_{k,i})$ は，優先度が $\tau^{'}_{k,i}$ より高いタスクから実行を阻止される割合を示す．$High(\tau^{'}_{k,i})$ は，$\tau^{'}_{k,i}$ より優先度が高いタスクが常に $\tau^{'}_{k,i}$ の実行を阻止することを仮定している．よって $R_l(\tau^{'}_{k,i})$ の値に関係なく，$High(\tau^{'}_{k,i})$ の割合で実行が阻止される．

- 実行効率（$E(\tau_i)$）
  タスク τ_i の実行効率は，各種命令のレイテンシと式（2）で求めたユニット競合率より式（7）で表される．

Vol.2011-SLDM-149 No.24
Vol.2011-EMB-20 No.24
2011/3/18

$$E(\tau_i) = \sum_{l=1}^{L} D(\lambda_l) / \sum_{l=1}^{L} (F_l(\tau_i) \times D(\lambda_l)) \quad (7)$$

実行効率より，タスク τ_i の RMT におけるシステム使用率が求められる．式 (8) に実行時間，及びシステム使用率を求める式を示す．

$$U_i^{'} = U_i / E_i \quad (8)$$

4.2 優先度の付与

RMT プロセッサ用 U-link スケジューリングでは同時実行セット生成後，同時実行セット内のタスクが実行するスレッドに優先度付き SMT における優先度を付与することで，同時実行セットの生成で削減できなかったアイドル時間を削減する．優先度付き SMT では，リソースの競合が発生した場合，高優先度のタスクにリソースを割当てることができる．優先度付与のアルゴリズムは次のようになる．

(1) RMT 実行において，優先度を付与しない場合のシステム使用率が大きいタスクから高い優先度を付与

(2) 優先度が高いタスクより順番に，$P(\tau_{k,i}^{'}) > P(\tau_{k,j}^{'})$ かつ $U^{'}(\tau_{k,i}^{'}) < U^{'}(\tau_{k,j}^{'})$ となるタスクを探索し，条件が成り立つタスクに対して $P(\tau_{k,j}^{'}) = P(\tau_{k,i}^{'})$ とした場合のシステム使用率を計算

(3) $(max(U_{k,u}^{'} | \tau_{k,u}^{'} \in \tau_k^{'}) - min(U_{k,p}^{'} | \tau_{k,p}^{'} \in \tau_k^{'}))$ の値が小さくなる場合に，優先度を $P(\tau_{k,j}^{'}) = P(\tau_{k,i}^{'})$ と設定

(4) $P(\tau_{k,i}^{'}) > P(\tau_{k,j}^{'})$ かつ $U^{'}(\tau_{k,i}^{'}) < U^{'}(\tau_{k,j}^{'})$ が成り立つすべてのタスクに対して (2), (3) を繰返し実行

アルゴリズム (1) において $P(\tau_{k,i}^{'}) \leq P(\tau_{k,j}^{'})$ となるタスクに対して，アルゴリズム (2) 以降で $P(\tau_{k,i}^{'}) > P(\tau_{k,j}^{'})$ となることはない．アルゴリズム (2), (3) を実行することで，(1) で設定された優先度での実行に比べてアイドル時間を削減できる．各同時実行セットに対してのシステム使用率の計算は最大で $((M^2 - M)/2) + 1$ 回必要となる．しかしながら，RMT プロセッサ用 U-link スケジューリングでは新たなタスクが到着しないことを仮定しているので，優先度の付与は開始時に一度だけである．また，システム起動前に静的に計算することも可能であるため，オーバーヘッドがリアルタイムシステムに影響を与えるとは考えにくい．

4.3 スケジュール可能性

ハードリアルタイムにおけるスケジューリングアルゴリズムでは，リアルタイム性を保証するためにスケジューリング可能性を定式化する必要がある．式 (9) を満たすタスクセットはスケジュール可能である．また，ページ数に制限があるため，証明は省略する．

$$\sum_{\tau_k^{'} \in \tau^{'}} max\{U_{k,u}^{'} \ | \tau_{k,u}^{'} \in \tau_k^{'}\} \leq 1 \quad (9)$$

実行する同時実行セットは EDF によって選択される．EDF におけるスケジュール可能なシステム使用率は 1 であるため，RMT 実行における使用率からスケジュール可能性を定義できる．

5. 評　　価

本章では U-link スケジューリングを比較対象とし，RMT プロセッサ用 U-link スケジューリングの実機評価を行う．評価では，提案手法がスループットを向上し，実行効率の変動を抑制できることを示す．評価指標は，**表 3** に示すタスクに対しての，タスク拒否率，実行時間の変動，タスクごとの実行時間とする．表 3 に示した各タスクの命令数はプログラムを逆アセンブルすることにより求めた．また，シングルスレッドにおける実行時間 C_i は事前に RMT プロセッサで評価した結果．8.8μsec，7.7μsec，95μsec であったが計算を簡単にするため表 3 の値とした，タスクはそれぞれ，整数演算の多い PID 制御，浮動小数点演算の多い PID 制御，配列処理をである．文献 5) によると，実際のヒューマノイドロボット等のアプリケーションは，タスク数及び周期の組合せが多くないので評価においてタスク数を限定した．RMT プロセッサの構成，及び各種機能ユニットのレイテンシはそれぞれ **表 1**，**表 2** である．ただし，RMT プロセッサの優先度付き SMT にはスレッドが 8 つ存在するが，同時実行できるのは 4 スレッド程度であるため，論理プロセッサ数を 4 個とする．

Vol.2011-SLDM-149 No.24
Vol.2011-EMB-20 No.24
2011/3/18

表 1 RMT プロセッサの構成
Table 1 RMT Processor Configuration

Clock Frequency	100MHz
Fetch Width	8
ALU	4 + 1(Divider)
FPU	2 + 1(Divider)
Branch Unit	2
Memory Access Unit	1

表 2 各種命令のレイテンシ
Table 2 The delay of each instruction type

ALU	ALU D	FPU	FPU D	MAU	BU
1	9	3	10	15	1

表 3 評価に使用するタスク
Table 3 Execution task

タスク	ALU	ALU D	FPU	FPU D	MAU	BU	C_i	T_i
τ_1	26	1	0	0	12	1	$10\mu sec$	$100\mu sec \sim 400\mu sec$
τ_2	15	0	11	1	11	1	$10\mu sec$	$100\mu sec \sim 400\mu sec$
τ_3	27	0	0	0	31	5	$100\mu sec$	$1msec \sim 4msec$

5.1 タスク拒否率

RMT プロセッサ用 U-link スケジューリングのタスク拒否率を評価する．スケジューラにスケジュール可能性を判定する機構を実装し，U-link スケジューリングと比較して多くのタスクを処理できたことを示す．また，ハードリアルタイムシステムではデッドラインミスは許容されないため，スケジュール可能性判定によって許可されなかったタスクセットはスケジューリングするタスクから除外される．**図 3** に，システム負荷を変化させていった場合のタスク拒否率を示している．

タスク拒否率 =
（受け入れを拒否されたタスクセット数）/（システムに投入されたタスクセット数）

図 3 より U-link スケジューリングと比較して，どのシステム負荷においても RMT プロセッサ用 U-link スケジューリングのタスク拒否率が低く，最大で 10%削減している．これは，タスクが実行するスレッドに最適な優先度を付与することで，RMT 実行におけるシステム使用率の最大が削減できたためである．タスク拒否率が低下すると，より多くのタスクを処理できるため，スループットが向上したといえる．

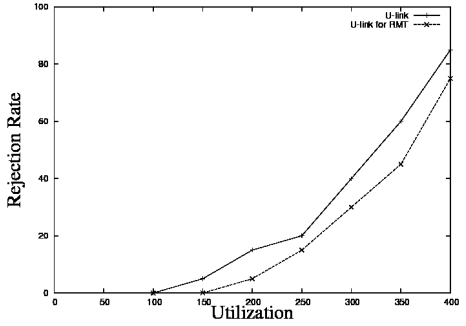

図 3 タスク拒否率（スケジュール可能性判定有り）
Fig. 3 Task rejection ratio with a feasibility test

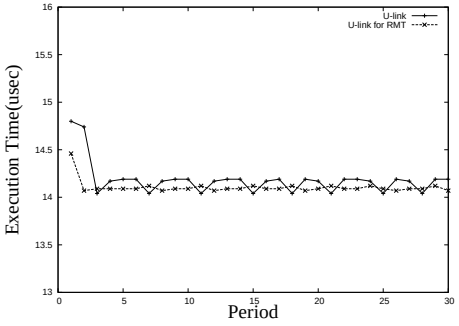

図 4 実行時間の変動
Fig. 4 Fluctuation of execution time

5.2 実行時間の変動

RMT プロセッサ用 U-link スケジューリングの実行時間の変動について評価する．**図 4** は，システム負荷 100%において，ある同時実行セットの各周期における実行時間の変移を示している．RMT プロセッサ用 U-link スケジューリングは U-link スケジューリングに比べて実行時間の変動を抑制できたことがわかる．タスクが実行するスレッドに優先度を付与することで，機能ユニットを使用するタスクが常に高優先度タスクとなるためである．

5.3 タスクごとの実行時間

RMT プロセッサ用 U-link スケジューリング及び U-link スケジューリングにおける各タスクの平均実行時間及び予測実行時間を評価する．**図 5**，**図 6**，**図 7** に，各タスクの平均実行時間及び予測実行時間を示している．今回の評価では，RMT プロセッサ用 U-link スケジューリング，及び U-link スケジューリングに関して，実際の実行時間よりも過大に評価を行っていることがわかる．これは，ユニット競合率の見積りが過大であるためである．ユニット競合率の計算では，同じユニットを使用する命令が常に競合することを仮定しているため，実際のユニット競合率よりも大きな値を見積もっている．しかしながら，ユニット競合率を過小評価すると予測実行時間よりも過小評価されるため，スケジュール可能性判定

Vol.2011-SLDM-149 No.24
Vol.2011-EMB-20 No.24
2011/3/18

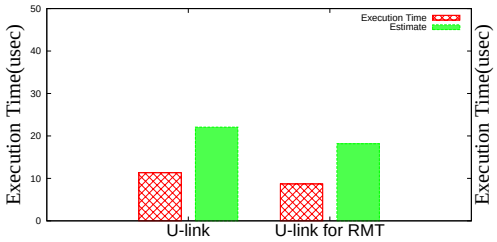

図 5 τ_1 の実行時間
Fig. 5 Execution time of τ_1

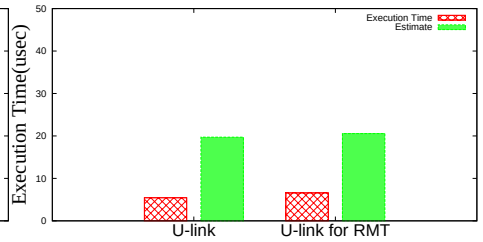

図 6 τ_2 の実行時間
Fig. 6 Execution time of τ_2

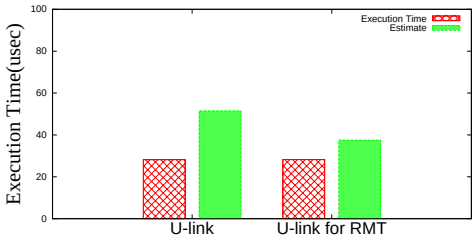

図 7 τ_3 の実行時間
Fig. 7 Execution time of τ_3

が適切に動作せず，デッドラインミスを引き起こす可能性がある．この点を考慮すると，競合率を大きく見積もることでデッドラインミスを防止できたと考えられる．また，τ_1 及び τ_3 では，U-link スケジューリングと比較して予測実行時間が削減できている．これは，適切な優先度を付与したことにより，効率的にタスクへリソースを与えることができるためである．

6. 結　　論

本研究では，RMT プロセッサにおいてリアルタイム性を保証しつつ，高スループットを実現するため，RMT プロセッサ用 U-link スケジューリングを提案し，その有効性を評価した．表 3 に示したタスクを用いた評価では，U-link スケジューリングと比較してタスク拒否率を最大約 10%改善できた．

RMT プロセッサには優先度付き SMT の他に IPC 制御機構が存在する．今後の課題として，RMT プロセッサ用 U-link スケジューリングにおいて IPC 制御機構を使用することで，スループットを向上することが挙げられる．また，本研究ではプログラムフローの解析を行っていない．今後の課題として，プログラムフローの解析を行うことによって，さらに実行効率の変動を予測することも挙げられる．

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．また，本研究の一部は文部科学省グローバル COE プログラム「環境共生・安全システムデザインの先導拠点」に依るものであることを記し，謝意を表す．

参　考　文　献

1) D. M. Tullsen, S. J. Eggers and H. M.Levy: Simultaneos Multithreading: Maximizing On-Chip Parallelism.,International Syposium on Conputer Architecture,22, pp.392-403 (1995).
2) K. Olukotun and B. Nayfe and L. Hammond and K. Wilson and K.Chang: The Case for a Single-Chip Multiprocessor,Architectural Support for Programming Languages and Operating Systems,15, pp.51-59 (2003).
3) Shinpei Kato, Hidenori Kobayashi and Nobuyuki Yamasaki: U-link Schedulink:Bounding Execution Time of Real-Time Tasks with Multi-Cose Execution Time on SMT Processor,International Conference on Embedded and Real-Time Computing Systems and Application,11, pp.193-197 (2005).
4) N. Yamasaki: Responsive Multithreaded Processor for Distributed Real-Time Systems,Robotics and Mechatronics,17, pp.130-141 (2005).
S. Kato and N. Yamasaki: Study of Real-Time Scheduling under Prioritized Simultaneous Multithreading,IEEE International Real-Time System Symposium,27, pp. 97-100 (2006).
5) 松井俊浩, 比留川博久, 石川裕, 山崎信行, 加賀美聡, 堀俊夫, 金広文男, 斉藤元, 稲邑哲也：ヒューマノイド・ロボットのための実時間分散情報処理, 実時間処理に関するワークショップ,pp.1-7 (2004).