Vol.2011-SLDM-149 No.36
Vol.2011-EMB-20 No.36
2011/3/19

Responsive Multithreaded Processorにおけるリアルタイム実行支援機構の設計と実装

松本康平[†1]　梅尾寛之[†2]　山崎信行[†2]

本論文では優先度付き SMT アーキテクチャを採用している Responsive Multithreaded Processor (RMTP) のリアルタイム実行を支援する，スレッドスケジューリング機構と目標 IPC 制御機構の設計と実装を行う．スレッドスケジューリング機構は，従来ソフトウェアスケジューラが行ってきた，起動するスレッドやスイッチするコンテキストの選択とデッドラインミスの検出をハードウェアで行うことでオーバヘッドを削減する．また，目標 IPC 制御機構は，RMTP の Instruction per Cycle (IPC) 制御機構の制御に用いられる目標値を制御することで予測される実行命令数と実際の実行命令数の誤差を縮小させ，タスクの実行時間予測性を高める．目標 IPC 制御機構を用いることで，4 スレッド同時実行において，誤差を最小で 2.60×10^{-5} %まで抑制することが出来た．

Design and Implementation of Support Scheme for Realtime Execution on Responsive Multithreaded Processor

Kohei Matsumoto,[†1] Hiroyuki Umeo[†2]
and Nobuyuki Yamasaki[†2]

This paper describes a design and implementation of a thread-scheduling scheme and a target-IPC control scheme for Responsive Multithreaded Processor (RMTP) which adopts an architecture based on a prioritized SMT architecture. Instead of a software scheduler, the former scheme selects a runnable thread, detects deadline misses, and chooses contexts that should be switched. The latter scheme controls a target-IPC (Intstruction per Cycle) that is used in the IPC control scheme of RMTP and minimize the error between the predicted and actual execution rates, improving a predictability of execution time. The result of the implementation of the target-IPC control scheme shows that the error is reduced to 2.60×10^{-5} % in case of 4 threads concurrency.

1. はじめに

リアルタイムシステムではタスクが時間制約を満たす必要があり，更にタスクの実行時間を正確に予測できることが望ましい．リアルタイムシステムに SMT アーキテクチャ[1)2)] を採用する場合，優先度によりスレッドを制御することで時間制約に対応し，かつタスクの IPC(Instruction per Cycle) を制御し安定化させることで実行時間予測性を高めることが考えられる．本研究の実装対象である Responsive Multithreaded Processor (RMTP)[3)] は 8 スレッド並列実行ならびに優先度付き SMT アーキテクチャを採用したリアルタイム処理用プロセッサである．スレッドに優先度を設けることで，高優先度スレッドが予想された実行時間内にタスクを完了することを目的としている．更に，現在最新版である Dependable Multithreaded Processor (DRMTP) には IPC 制御機構[5)] が実装されており，IPC を可能な限り安定化させることで実行時間の予測可能性を高めている．

一方で，現在の RMTP は起動させるスレッドの選択・デッドラインミスの検出・コンテキストスイッチを行うスレッドの選択をソフトウェアで行っており，スケジューリングの際のオーバヘッドが大きい．文献 4) では起動させるスレッドの選択とコンテキストスイッチを行うスレッドの選択をハードウェアで行う機構の研究が行われたが，デッドラインミスの検出は行っていない．

また，現在の IPC 制御機構は命令のフェッチやキャッシュミス，複数のタスク間の競合などにより，実際にコミットした命令数が目標値に達しなかった場合，後でその誤差を修正することが出来ない．したがって，実際の実行時間が予測される実行時間を必ず上回ってしまう．

上記の問題の解決のため，本論文では上記の各種スレッドの選択とデッドラインミスの検出をハードウェアで行うことでスケジューリングを行う機構と，IPC 制御機構において用いられる目標 IPC を制御する機構を設計及び実装する．これにより，RMTP のリアルタイム実行を支援することを目的とする．

†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 Schience, Graduate School of Science and Technology, Keio University

Vol.2011-SLDM-149 No.36
Vol.2011-EMB-20 No.36
2011/3/19

2. Responsive Multithreaded Processor

本章では，本研究の実装対象である Responsive Multithreaded Processor (RMTP) について述べる．

2.1 Responsive Multithreaded Processing Unit

RMT Processing Unit (RMTPU，**図 1**) は RMTP のプロセッシングユニットである．Simultaneous Multithreading (SMT) アーキテクチャをベースとし，それに優先度を導入することで特定のスレッドを優先的に処理できる優先度付き SMT アーキテクチャを採用している．RMTPU は最大 8 スレッド並列実行が可能である．

SMT とは，1 クロックサイクル毎にコンテキストスイッチを行うことで個々のスレッドのレイテンシを隠蔽する細粒度マルチスレッディングと，複数の命令発行スロットを持つことで IPC を向上させるスーパースカラを組み合わせたアーキテクチャである．これにより，1 クロックサイクル内に複数のスレッドから複数の命令を発行することが可能になり，システム全体の IPC が向上する．一方で，SMT アーキテクチャにおいてシングルスレッドの性能は低下し，実行時間の変動が大きくなるという欠点がある．

2.1.1 コンテキストキャッシュ

システムにおけるタスクの時間制約を守るため，スケジューラはリアルタイムスケジューリングを行い適切に CPU 時間をタスクに割り当てる必要がある．スケジューラはシステムで実行する全ての処理に対し，スケジューリングアルゴリズムにしたがって優先度を決定する．優先度によって処理の実行順序が決定すると，スケジューラは優先度の高い順に処理が実行されるようにコンテキストスイッチを行う．ソフトウェアでコンテキストスイッチを行う場合は Load 命令と Store 命令を繰り返すことによる大量のメモリアクセスが発生し，オーバーヘッドが大きくなる．

コンテキストスイッチの際のオーバヘッドを削減するために，RMTPU はオンチップで 32 スレッドのコンテキスト情報を保持出来る専用キャッシュを搭載している．実行スレッドが 8 を超えた場合にはソフトウェアスケジューラによってスイッチすべきスレッドを選択し，ハードウェアによってコンテキストスイッチが行われる．コンテキストスイッチそのものはハードウェアで実行されるため 4 クロックサイクルで完了するが，ソフトウェアスケジューラによるスレッドの選択には数千クロックサイクルを要する[4]．文献 4) ではスレッドの選択をハードウェアで行うスレッドスケジューリング機構が設計及び実装されている．この機構によって，3178 クロックを要していたコンテキストキャッシュを 10 クロックで行うことが出来る．

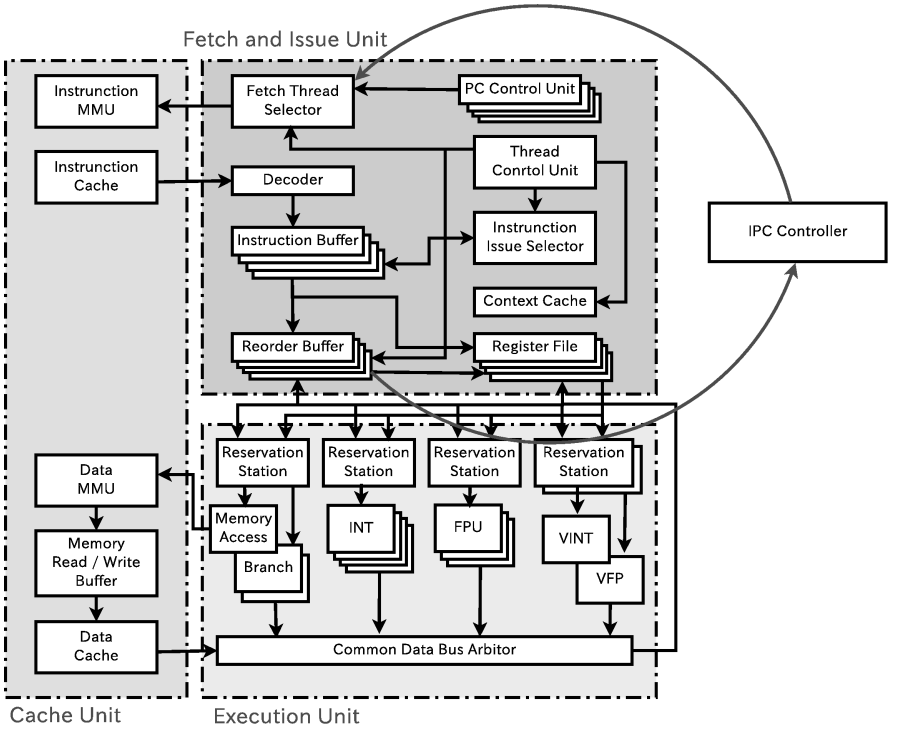

図 1 RMTPU のブロック図
Fig. 1 Block diagram of RMTPU

スレッドスケジューリング機構は一定時間（Sampling Period）毎に起動し，プロセッサが保持するすべてのスレッドのリリース時間を調べる．リリース時間を迎えたスレッドが 1 つないし複数ある場合は，その中で最も優先度の高いものを選択する．更に，実行スレッドの中から最も優先度が低いものを選択し，リリース時間を迎えたスレッドと優先度を比較してコンテキストキャッシュを行うかどうかを判断する．

文献 4) のスレッドスケジューリング機構は以上のようにスレッドのリリースのみを対象としており，デッドラインミスの検出は行っていない．

Vol.2011-SLDM-149 No.36
Vol.2011-EMB-20 No.36
2011/3/19

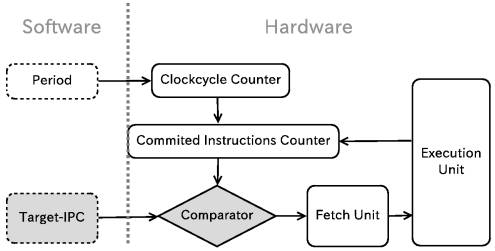

図 2 RMTP における IPC 制御機構（フィードフォワード制御）
Fig. 2 IPC control scheme (using feed-forward control) in RMTP

2.1.2 IPC 制御機構

SMT を採用した他のプロセッサと同様に，RMTP は実行時間の変動が大きいという欠点があった．最新版の RMTP である Dependable RMTP (DRMTP) では，RMTPU に IPC 制御機構を設けることでこの欠点を克服している[5]（図 1）．

図 3 は RMTPU に設けられた，フィードフォワード制御を用いた IPC 制御機構のブロック図である．ソフトウェアで制御周期 (Period) と目標 IPC (Target-IPC) を設定し，Commited Instruction Counter が 1 制御周期内にコミットされた命令数をカウントする．Comparator は目標 IPC とコミットされた命令数を比較し，コミット数が目標値を超えた場合は Fetch Unit にフェッチ停止信号を送る．制御周期内にコミット数が目標値を超えなかった場合は何もしない．

現在の IPC 制御機構は上記の通り，コミット数が目標値に達しなかった場合には何も行わない．したがって，タスクの実行開始直後のフェッチやデコードによるコミット不足や，常に起き得るキャッシュミス等によるコミット不足を補うことが出来ない．その結果，実際の実行時間が予測した実行時間を必ず上回るという問題が生じる．

3. 設計及び実装

本章では，RMTPU の目標 IPC 制御機構とデッドラインミス検出機能を組み込んだスレッドスケジューリング機構の設計と実装方法について述べる．

3.1 スレッドスケジューリング機構の設計

スレッドスケジューリング機構の設計は文献 4) で設計されたものにデッドラインミス検出機構を組み込む形で行う．

従来のスレッドスケジューリング機構はハードウェアによってあらかじめ周期の最小単位 (Sampling Period) を定義しておき，定義した Sampling Period ごとにカウントアップするカウンタと，周期スレッド管理用のテーブルを持つ．テーブルのエントリ数は，ハードウェアコンテキストのエントリ数とコンテキストキャッシュのエントリ数の合計である 40 エントリである．このテーブルでは以下の情報を管理する．

Thread ID: スレッド生成時にテーブルにスレッドの ID が登録される
Periodic Mode: スレッドが周期タスクかどうかを示す
Timer: 周期実行を行うかどうかを示す
Period: 指定した周期が格納される
Release Time: リリース時間を示す
Release Bit: リリース時間に到達したことを示す

従来のスレッドスケジューリング機構は Sampling Period ごとに起動し，リリースすべきスレッドがないかどうかチェックする．リリースすべきスレッドが 1 つまたは複数あった場合は，最も優先度の高いスレッドを選択し，スイッチ候補とする．これと同時にハードウェアコンテキスト上にある最も優先度の低いスレッドを選択し，コンテキストキャッシュ上のスイッチ候補であるスレッドの優先度と比較する．比較の結果，コンテキストキャッシュ上のスレッドの優先度の方が高ければ，コンテキストキャッシュを行う．

今回の機構ではデッドラインミス検出のため，上記の周期スレッド管理用テーブルに新たに以下の情報を管理させる．

Deadline: デッドラインである時間を示す
Deadline Miss Bit: デッドラインミスが発生したことを示す

スレッドスケジューリング機構は以上の情報を元に，リリースすべきスレッドのチェックと同時にデッドラインミスを起こしたスレッドがないかチェックする．デッドラインミスを起こしたスレッドがあった場合は，該当するスレッドの ID をレジスタに保存し，同時に割り込みをかける．**図??**は今回のスレッドスケジューリング機構の概要図である．

3.2 目標 IPC 制御機構の設計

図 4 に目標 IPC 制御を組み込んだ IPC 制御機構のブロック図を示す．図中の Threshold と Target-IPC Controller が IPC 制御機構に追加したユニットである．Target-IPC Controller は，ソフトウェアから目標 IPC(Target-IPC) を，Commited Instruction Counter からはコミット数 (commit count) を受け取る．更に，Target-IPC Controller 自らが出力した，修正済みの目標 IPC (Controlled Target-IPC) をフィードバックして，フェッチの上限値 (fetch bound) とする．Target-IPC Controller は fetch bound と commit count の差

Vol.2011-SLDM-149 No.36
Vol.2011-EMB-20 No.36
2011/3/19

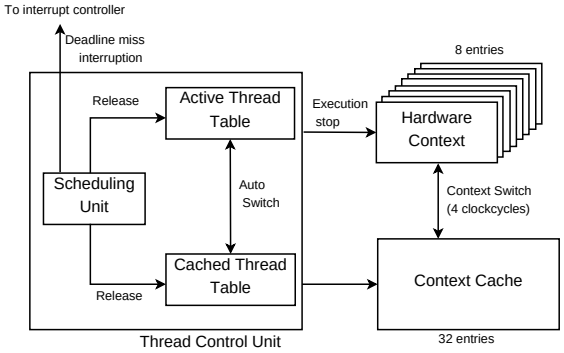

図 3 提案するスレッドスケジューリング機構の概要図
Fig. 3 schematic of the proposed thread-scheduling mechanism

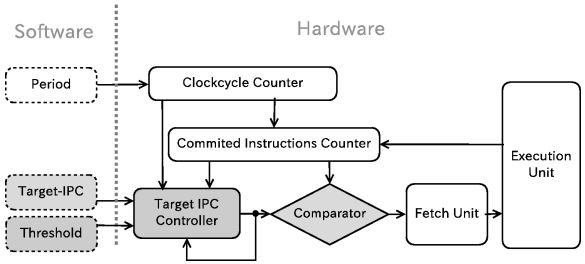

図 4 目標 IPC 制御機構を組み込んだ IPC 制御機構
Fig. 4 IPC controle scheme with target-IPC controller

を取り，この差を Target-IPC に加算して Controlled Target-IPC とし，Comparator に出力する．

3.2.1 スレッショルドの設定と誤差レジスタ

マルチスレッディングにおいて，スレッドは互いに資源を共有する．よって，あるスレッドの IPC が急激に変化した場合，他のスレッドの IPC に影響を与え，結果的にタスク全体が不安定になってしまう．

そこでコミット数と目標値との誤差が大きい場合には，誤差をそのまま次の制御時の目標値に加算せず，誤差を一定の範囲内に収めることが出来るように誤差の上限値・下限値（スレッショルド）を設定できるようにする．スレッショルドの設定はソフトウェアで行う（図 4 の Threshold）．

表 1 RMTPU のパラメータ表
Table 1 parameters for RMTPU

命令フェッチ数	8
同時命令発行数	4
同時命令完了数	4
キャッシュ方式	LRU
キャッシュサイズ（命令，データ）	32KB
ページサイズ／スレッド	16MB
動作周波数	50MHz

スレッショルドは式 (1) の通り，目標 IPC に対する比を指定することで決定する．

$$Threshold = Target\text{-}IPC \pm Target\text{-}IPC \times parameter \quad (1)$$

例えば，目標 IPC(Target-IPC) が 0.6，スレッショルド設定値 (parameter) が 1/4 であれば，スレッショルド上限値は 0.725，下限値は 0.475 となる．

設定値は 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128 の 7 種類の値を任意に組み合わせることで指定する．すなわち，1/128 から 127/128 までの値を指定することが可能である．

スレッショルドを超えた分は破棄されずにレジスタに保存され，次以降の制御を待つ．

4. 評　　価

設計した目標 IPC 制御機構を Verilog-HDL を用いて実装し，NC-Verilog を用いた RTL シミュレーションによって評価を行った．スレッドスケジューリング機構の評価は行っていない．

4.1 評 価 環 境

プロセッサは現在最新版である Dependable Responsive Multithreaded Processor (DRMTP) を利用した．評価に用いた RMTPU のパラメータを**表 1** に示す．

評価に用いたベンチマークを以下に示す．ベンチマークを処理するタスクの優先度は matrix が最も高く，以下，sort，gzip，md5 と続く．

- matrix
  32x32 要素の int 型の行列の演算を行う．
- sort
  1024 要素の unsigned int 型のデータをクイックソートにより整列する．
- gzip

Vol.2011-SLDM-149 No.36
Vol.2011-EMB-20 No.36
2011/3/19

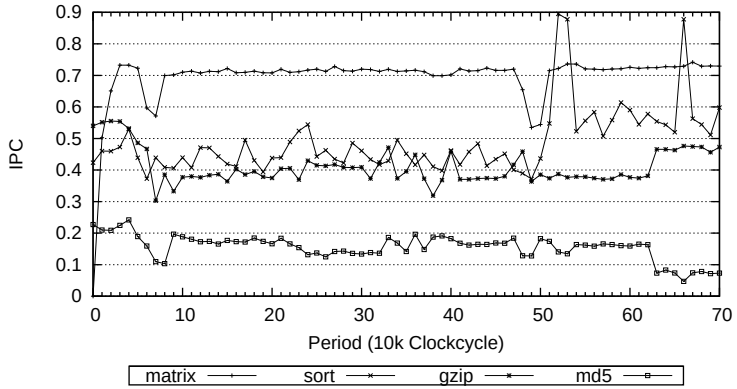

図 5 マルチスレッド実行時におけるベンチマークの IPC 推移
Fig. 5 Transition of IPC in case of multi-thread benchmark execution

512 要素の char 型のデータを gzip 形式に圧縮する．

- md5

  9216 要素の unsigned char 型のデータから 128 ビットのハッシュ値を生成する．

これらのベンチマークをマルチスレッド実行したときの IPC の時間推移を**図 5** に示す．いずれのスレッドにおいても IPC が大きく不規則に変動しており，実行時間の正確な予測が困難であることが分かる．

IPC 制御機構に用いる制御周期は 10000 クロックサイクルである．また，各ベンチマークの目標 IPC を**表 2** に示す．目標 IPC は各ベンチマークのシングルスレッド実行時の IPC の 50〜80 %とした．

目標 IPC 制御機構で用いるスレッショルドの設定パラメータは 1/4 とする．したがって，各ベンチマークにおける目標 IPC の上限値および下限値は**表 3** の通りになる．

実行時間予測性の検証は平均誤差率を求めることで行う．平均誤差率は式 (2) で定義される．

$$AverageErrorRate = \frac{|\sum(targetIPC - commitedInstruction)|}{targetIPC} \times 100 \qquad (2)$$

表 2 各ベンチマークの目標 IPC
Table 2 target-IPCs used for each benchmark

benchmark	target-IPC
matrix	0.6
sort	0.4
gzip	0.4
md5	0.2

表 3 各ベンチマークにおける目標 IPC の上限値・下限値
Table 3 target-IPCs used for each benchmark

benchmark	upper threshold	lower threshold
matrix	0.725	0.475
sort	0.500	0.300
gzip	0.500	0.300
md5	0.250	0.150

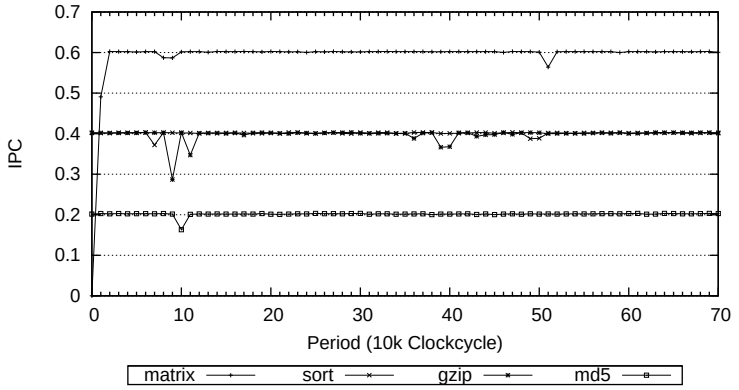

図 6 目標 IPC 制御を行わなかった場合の IPC 推移
Fig. 6 Transition of IPC in case without target-IPC control

4.2 評価結果

図 6 は目標 IPC 制御を行わない従来の IPC 制御機構による IPC 制御の様子，**図 7** は目標 IPC 制御機構を組み込んだ新たな IPC 制御機構による IPC 制御の様子である．図 6 では時折コミット数が目標値に達していないが，図 7 ではコミット数の不足分を補うように目標値が制御されている様子が確認できる．

表 4 に，1000 制御周期後の平均誤差率を示す．without new scheme は目標 IPC 制御機構がない IPC 制御機構の場合，with new scheme は目標 IPC 制御機構がある場合である．従来の機構ではおおよそ 10^{-1} %のオーダーで誤差が発生しているが，新たな機構では最小で 1.25×10^{-5} %となった．

Vol.2011-SLDM-149 No.36
Vol.2011-EMB-20 No.36
2011/3/19

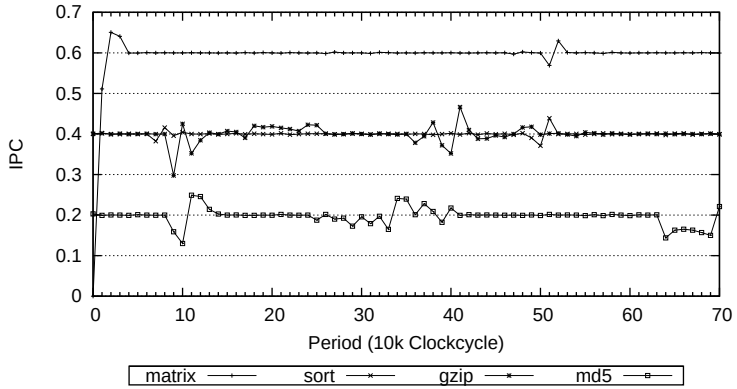

図 7 目標 IPC 制御を行った場合の IPC 推移
Fig. 7 Transition of IPC in case with target-IPC control

表 4 目標 IPC 制御機構の有無による平均誤差率の比較
Table 4 Comparison of average error rates between with and without target-IPC control scheme

benchmark	without new scheme (%)	with new scheme (%)
matrix	3.32×10^{-1}	4.12×10^{-4}
sort	4.07×10^{-1}	2.60×10^{-5}
gzip	5.13×10^{-1}	4.17×10^{-4}
md5	1.06	1.25×10^{-3}

4.3 論理合成

論理合成は TSMC 製 0.13μm プロセスのライブラリを用いて Synopsys Design Compiler 2009.06 により行った．**表 5** は目標 IPC 制御機構のハードウェアの面積である．Target-IPC Controller / thread は 1 スレッド当たりの面積であり，Total は目標 IPC 制御機構の追加により増加した面積の合計である．Total には 8 スレッド分の目標 IPC 制御機構の他にスレッショルド設定用の制御レジスタなどの面積が含まれる．従来の RMTPU の面積は 54.49mm^2 であるから，全体の面積に対して約 0.55 %の増加となる．

5. まとめ

本論文では，Responsive Multithreaded Processor (RMTP) のリアルタイム実行を支援することを目的として，スレッドスケジューリング機構と目標 IPC 制御機構を提案した．

表 5 目標 IPC 制御機構のハードウェアの面積
Table 5 Hardware size of target-IPC control scheme

Target-IPC Controller / thread	$3.746 \times 10^{-2} mm^2$
Total	$2.986 \times 10^{1} mm^2$

目標 IPC 制御機構により，マルチスレッド実行時のフィードフォワード制御において従来の IPC 制御機構よりも実行時間予測性を高めることが出来た．これにより，理論通りの時間でタスクを実行することが可能になる．

今回はスレッドスケジューリング機構は設計のみ行い評価を行っていない．今後はスレッドスケジューリング機構単体及び IPC 制御機構との連携を評価し，ハードウェアのみで RMTP のリアルタイム実行を支援できることを示す予定である．

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．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) Tullsen, D. M., Eggers S. J., Erner, J. S., Levy, H. M., Lo, J. L. and Stamm, R. L.: Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor, *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, pp.91–202 (1996).
2) Tullsen, D., Eggers, S. and Levy, H.: SImultaneous multithreading: Maximizing on-chip parallelism, *Computer Architecture, 1995. Proceedings. 22nd Annual International Symposium on*, pp.392–403 (1995).
3) Yamasaki, N.: Responsive multithreaded processor for distributed real-time system, *Journal of Robotics and Mechatronics*, Vol.17, No.2, pp.130–141 (2005).
4) 梅尾寛之，水頭一壽，武田　瑛，加藤真平，山崎信行：Responsive Multithreaded Processor におけるスレッドスケジューリング機構の設計と実装，情報処理学会研究報告，Vol.22, pp.55–60 (2009).
5) 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, No.12 pp.2206–2215 (2010).