社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2004－ＳＬＤＭ－114　（２）
2004／3／18

Responsive Multithreaded Processorの全体設計

山崎　信行†

† 慶應義塾大学理工学部　〒223-8522 横浜市港北区日吉 3-14-1
E-mail: †yamasaki@ics.keio.ac.jp

あらまし　本論文は，並列分散リアルタイム処理用 *Responsive Multithreaded (RMT) Processor* の全体設計について述べる．*RMT Processor* は，並列分散リアルタイム処理に必要な機能のほとんどを 1 チップに集積したシステム LSI である．具体的には，1 チップにリアルタイム処理機能 (*RMT PU*, Cache, etc.)，リアルタイム通信機能 (*Responsive Link II*)，コンピュータ用周辺機能 (DDR SDRAM I/Fs, DMAC, PCI64, USB2.0, IEEE1394, etc.)，制御用周辺機能 (PWM Generators, Pulse Counters, etc.) を集積している．*RMT Processor* は，リアルタイムスケジューラにより優先度付けされた処理や通信を，優先度に従い，通信や処理の追い越しまたは調停を行うことによって，リアルタイム通信／処理を実現した．時間粒度を非常に小さく高精度に行うことを可能にしたと同時に，高機能・高性能と低消費電力の両立を実現して，大規模な並列分散リアルタイムシステムを構築可能にした．
キーワード　リアルタイム，マルチスレッド，レスポンシブリンク，システム LSI, RMT

Total Design of Responsive Multithreaded Processor

Nobuyuki YAMASAKI†

† Faculty of Science and Technology, Keio University
3-14-1, Hiyoshi, Kouhoku-ku, Yokohama, 223-8522 Japan
E-mail: †yamasaki@ics.keio.ac.jp

Abstract　This paper describes the total design of *Responsive MultiThreaded (RMT) Processor* for parallel/distributed real-time processing. *RMT Processor* is a system LSI that integrates almost all functions for parallel/distributed real-time processing. Concretely, *RMT Processor* integrates real-time processing functions (*RMT PU*, Cache, etc.), a real-time communication function (*Responsive Link II*), computer peripherals (DDR SDRAM I/Fs, DMAC, PCI64, USB2.0, IEEE1394, etc.), and control peripherals (PWM Generators, Pulse Counters, etc.). *RMT Processor* can execute real-time tasks and communications prioritized by the real-time scheduler, overtaking and/or arbitrating them by hardware, so that real-time processing and communication can be realized. *RMT Processor* also realizes both high performance and low power. Large scale parallel/distributed real-time systems can be realized by using *RMT Processors*.
Key words　Real-Time, Multithraed, Responsive Link, System LSI, RMT

1. はじめに

リアルタイムシステムの研究はオペレーティングシステム，ネットワーク，プロセッサ等の様々な分野で行われており，その応用分野は，ロボット，ユビキタスシステム，インテリジェントビル，アミューズメントシステム等，多種多様のシステムに広がりつつある．ネットワークで接続されたリアルタイムシステムは特に分散リアルタイムシステムと呼ばれ，産業的にもこれからの発展が期待されている．

本論文では，分散リアルタイムシステムを容易に実現することを目指したシステム LSI である *Responsive MultiThreaded Processor*（以下，*RMT Processor*）の設計について述べる．*RMT Processor* は，分散リアルタイムシステムの実現に必要なほとんどの機能を 1 チップに集積したシステムオンチップである．

RMT Processor は非常に大規模なシステム LSI(約 14[Mgate]) であり，各機能ブロックごとにそれぞれ新規な機構を設計・実装している．本論文だけで全ての設計を述べることはできないので，本論文では *RMT Processor* の全体の設計思想及びチップ全体の設計について述べる．

2. リアルタイム性

我々は，あらゆる処理／通信にリアルタイム性を有することを目標に *RMT Processor* の設計を行い，実際にシリコンエリアのかなりの部分をリアルタイム性維持のために使用する．

ここで，リアルタイム性とは，通信や処理等の真偽が時間にも依存するという性質である．狭義には，与えられた時間制約（デッドライン）を守るということを意味する [1].

2.1 ハードリアルタイムとソフトリアルタイム

リアルタイム性は，ハードリアルタイムとソフトリアルタイムの二つに大別することができる．

ハードリアルタイム性とは，必ず時間制約を守らなければならない性質であり，時間制約を少しでも破ると価値が 0 になる性質である．

ソフトリアルタイム性とは，時間制約を多少破ることを許容する性質であり，時間制約を破っても価値はただちに 0 にはならない．多くの場合，時間制約を破ると，時間経過と共に価値が急激に減少していく性質を指す．

ハードリアルタイム性の一例としては，分散制御がある．ソフトリアルタイム性の一例としては，VOD 等のネットワークを介した MPEG のデコードがある．

また，リアルタイム性を要求するアプリケーション（タスク）の種類によって，以下のように特徴が分かれる．

ハードリアルタイム性　主に制御系の通信や演算を行うタスクが要求するリアルタイム性であり，以下のような特徴がある．

通信　データ量は小さいが，レイテンシ（遅延）に対する要求が厳しい．スループットよりレイテンシを重視する．

演算　演算量は小さい場合が多いが，時間制約を厳守する必要性がある．

時間粒度とデッドライン　時間粒度が小さく，デッドラインが短い場合が多い．(100[μ sec]〜10[msec] 程度)

ソフトリアルタイム性　主にマルチメディア系の通信や演算を行うタスクが要求するリアルタイム性であり，以下のような特徴がある．

通信　データ量が非常に大きく（ストリーミング等），レイテンシよりもスループットを重視する．

演算　演算量は比較的大きい（MPEG のデコード等）が，時間制約は制御系に比較すれば厳しくない．

時間粒度とデッドライン　時間粒度が比較的大きく，デッドラインが長い場合が多い．(10[msec]〜1[sec] 程度)

このように，同じリアルタイム性といっても，ソフトリアルタイムとハードリアルタイムでは，リアルタイム性を要求するアプリケーションも異なるし，特徴も異なることが分かる．我々は，これら性質の異なるリアルタイム性を同時に扱うことができることを目指して *RMT Processor* の設計・実装を行った．

2.2 リアルタイムスケジューリング

分散リアルタイムシステムのような複雑なシステムでは，全ての起こりえる場合を想定することは不可能であるので，リアルタイムスケジューリングが必須となる．

古典的なリアルタイムスケジューラには EDF(Earliest Deadline First) や RM(Rate Monotonic) 等があるが，大抵のリアルタイムスケジューラは，ある一定時間間隔（ティック）毎に優先度に従ってプリエンプションを行いながら実行を行う [2]. 図 1 に EDF によるスケジューリングの例を示す．EDF では，デッドラインが近いタスクほど高い優先度を与えてスケジューリングを行う．

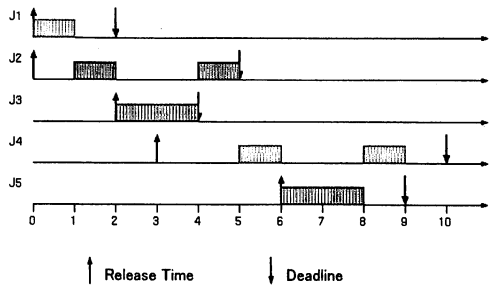

図 1　EDF スケジューリング
Fig. 1　EDF Scheduling

プリエンプションは，演算処理の場合，コンテキストスイッチに相当し，通信の場合，パケットの追い越しに相当する．従って，演算処理に関してはコンテキストスイッチを，通信に関してはパケットの追越を最適に行うリアルタイム通信・処理アーキテクチャを実現することを目標とする．

我々が開発したプロセッサ（ハードウェア）は，基本的にこれらのリアルタイムスケジューラ（ソフトウェア）でスケジューリングされ，適切な優先度が付与された演算や通信を最適に (つまりリアルタイムに) 実行することを目的とする．

3. *RMT Processor* の設計思想

前章で述べたとおり，リアルタイム性はリアルタイムスケジューラによって優先度に変換されるが，*RMT Processor* では，非常に単純な I/O(RS-232C 等) 以外の全ての機能ユニットにおいて，優先度を用いた追い越しや調停を行うという設計思想を貫く．また，この実現ためには，シリコンエリアをある程度広く使用しても構わないという設計思想をとる．場合によっては，本来の機能のみを単独で設計・実装するのに必要なシリコンエリアと同程度のシリコンエリアをリアルタイム性実現のために用いている機能ユニットも存在する．

ここでは詳細は省略するが，例えば *RMT PU* においては，キャッシュ，命令フェッチ，命令発行，命令実行等の機能ユニットにおいて優先度による追い越しまたは調停機構を設計・実装している．*Responsive Link II* 等の他の機能ユニットにおいても同様な設計を行っている．

また，実際に LSI 化を行うので，*RMT Processor* は設計当初からシリコン（バックエンド設計）を意識してアーキテクチャ設計を行っている．例えば，HDL では多ポートのマルチポートメモリを容易に記述可能であり，多ポートのマルチポートメモリを使用すれば設計は楽にできる場合が多い．しかしながら，実際にシリコン上に多ポートメモリを設計しようとすると非常

に大きなシリコンエリアを必要としレイテンシも長くなってしまう．さらにはマルチポートメモリの設計自身に大きな労力を必要とするので，現実的ではない．従って，*RMT Processor* の設計においては，マルチポートメモリはデュアルポートメモリまでという制限を設けて設計を行っている．この例のように，我々は現実的なチップを作成するために，アーキテクチャ設計時にシリコンにおとすことを前提として様々な制約を設け，LSI 設計を行った．現在の最先端プロセスを用いた大規模 LSI 設計においては，バックエンド設計しだいでそのチップが動作するかどうか，性能がでるかどうかが決まってくるが，それはフロントエンド設計をバックエンド設計を意識して設計しているかどうかにかかっているので，非常に重要である．

リアルタイム処理を必要とする組み込み用途においては，バッテリ等で駆動される場合も多く，基本的に低消費電力を要求される．また，最近の分散リアルタイム制御においては，動画処理や音声処理等の比較的大きな演算処理を要求されるソフトリアルタイム処理と，制御コマンド生成等のためのハードリアルタイム処理を同時に要求される．高性能演算処理と低消費電力は相反する要求であるが，この相反する要求の実現を目指して設計を行う．

以上のように，*RMT Processor* では，

- リアルタイム性をハードウェアで実現するためのシリコンエリアの積極的な使用
- 全ての機能ユニットにおいて優先度を用いた追い越し／調停機構
- バックエンド設計を意識した現実的なアーキテクチャ設計
- 高性能・高機能処理と低消費電力の両立

を設計思想として，チップ全体を設計した．

4. *RMT Processor* の設計

4.1 *RMT Processor* の概要

RMT Processor は，分散リアルタイムシステムを実現するために，リアルタイム通信・処理・制御を同時にハードウェアレベルで行うことを目的にして設計を行ったシステム LSI である．先に開発した *Responsive Processor* [3] ではハードウェアによるリアルタイム通信機構 (*Responsive Link*) のみ有し，リアルタイム処理機構は有していなかったが，*RMT Processor* では，ハードリアルタイム通信・処理及びソフトリアルタイム通信・処理の両方を行うことができるように設計した．

分散リアルタイムシステムを容易かつ効率的に実現するには，リアルタイム通信及びリアルタイム処理を行なうための基本機能を有した共通プラットホームを用意し，それらをブロックを組み立てるように組み合わせてシステムを構築できるようにすれば良いと考えられる．プラットホームに必要な機能としては，リアルタイム処理機能，リアルタイム通信機能，コンピュータ用周辺機能，各種周辺制御機能が考えられる．プラットホームとして様々なシステムの中に容易に組み込んで使用できるようにするために，*RMT Processor* は以下の機能を全て 1 チップに集積 (System-on-a-chip) している．図 2 に大まかなブロック図を，図 3 にチップのレイアウトを示す．

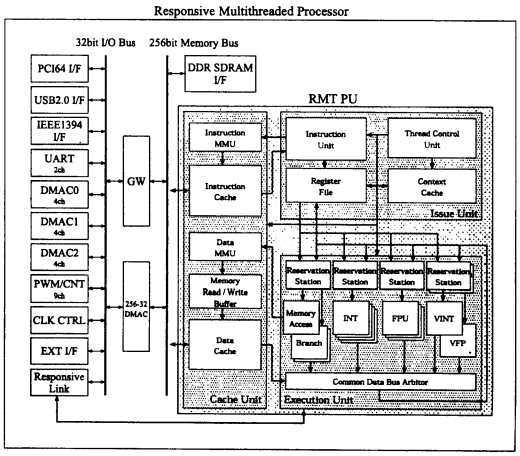

図 2　RMT Processor のブロック図
Fig. 2　Block Diagram of RMT Processor

- リアルタイム処理機能 (*RMT PU*, Cache, etc.)
- リアルタイム通信機能 (*Responsive Link II*)
- コンピュータ用周辺機能 (PCI64, USB2.0, IEEE1394, DDR SDRAM I/Fs, DMAC, etc.)
- 各種周辺制御機能 (PWM Generators, Pulse Counters, etc.)

システム設計者は本チップに必要な I/O（センサ，アクチュエータ，ディジタルカメラ等）を接続するだけで必要な機能を実現できる．それら I/O を接続し固有の機能を有した *RMT Processor* をそのシステムにふさわしいトポロジで *Responsive Link* を用いて複数個接続することによって，分散リアルタイムシステムを構築する．

チップ単体で使用した場合でも，例えば，USB2.0 や IEEE1394 に接続されたディジタルカメラでキャプチャした動画像を内蔵ベクトルユニットでリアルタイム処理しながら，同時にその処理結果を基にパルスカウンタと PWM 発生器を制御してアクチュエータの制御を行い，システム全体のリアルタイム制御を行う（ビジュアルフィードバック制御）というようなことが容易に可能である．

4.2 *Responsive Link*

RMT Processor は，リアルタイム通信として *Responsive Link II* を設計・実装している．*Responsive Link II* は，リアルタイム通信規格レスポンシブリンクに準拠している [4]．*Responsive Link II* のリアルタイム通信アーキテクチャは，分散リアルタイムシステム用であることを十分に考慮して設計を行っており，以下のようなユニークな特徴を有している．

- ハードリアルタイム通信（イベントリンク）とソフトリアルタイム通信（データリンク）を分離して独立に通信
- パケットに優先度を付加し，ノード毎に高優先度パケットが低優先度パケットの追い越しを行う
- リンクの種類毎に別経路を設定することが可能

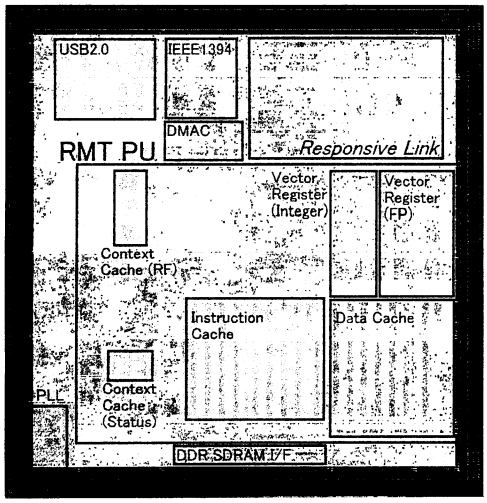

図 3 RMT Processor のレイアウト
Fig. 3 Layout of RMT Processor

- パケットの優先度が異なると，優先度毎に別経路を設定することが可能で，専用回線や迂回路を容易に実現可能
- ノード毎に優先度を付け替えることが可能であり，分散管理型でパケットの加減速を制御可能
- 256 レベルの優先度
- ハードウェアによる前方エラー訂正 (FEC)
- 通信速度を動的に変更可能

これらの特徴的な機能によって，柔軟なリアルタイム通信を実現している [3]．現在のレスポンシブリンクの一方向の最大通信速度は 800[Mbaud] である．通信速度は消費電力を抑えるために動作中に動的に変更可能である．*RMT Processor* には，5 組（5×5 のネットワークスイッチ）の *Responsive Link II* が設計・実装されている．そのうち 1 組は *RMT PU* に接続し，4 組がチップ外部に出ている．全 2 重でデータとイベントの 2 系統を 1 組 (1 本) として，1 チップ上にはそれが 5 組あるので，バックボーンとしては約 20[Gbps/chip] のスイッチング速度を有している．

RMT PU と *Responsive Link II* は，2 系統の接続方法で非常に密に接続を行っている（図 2 参照）．

まず一つ目の接続方法としては，通常の I/O と同様に 32bit 内部バスに接続している．*RMT PU* からは単なる 32bit I/O として扱われるが，DMAC と *Responsive Link II* 内の Dual Port Memory (DPM) を用いてハードウェアによるパケットの自動送受信ができるように設計を行っている．

二つ目の接続方法としては，低レイテンシが望まれる event link に関してのみ *RMT PU* の内部プロセッサバスに直接接続している．この接続の場合，*Responsive Link II* の event link は *RMT PU* からはプロセッサ内部レジスタとして取り扱われ，*RMT PU* は内部レジスタに読み書きすると，event link に対して送受信することができる．この機能によって，非常にレイテンシの小さな通信を可能にすると同時に，プロセッサ間共有レジスタを実現できる．

表 1 *RMT PU* のピークパフォーマンス
Table 1 Peak Performance of *RMT PU*

32bit Scalar Integer	1.2 GIPS
64bit Scalar Floating Point	600 MFLOPS
32bit Vector Integer	9.6 GIPS
64bit Vector Floating Point	4.8 GFLOPS
Power	Max. 8 W

Responsive Link は，現在，国内においては情報処理学会試行標準 ITSJ 0006:2003 として標準化されている [4]．国際的には，ISO/IEC JTC1 SC25 にお い標準化作業を行っている．これらの標準化により，異なるシステム間でのリアルタイム通信も可能になり，より大規模な分散制御システムが実現可能になると考えられる．

4.3 *RMT PU*

RMT PU のリアルタイム処理アーキテクチャ(*RMT Architecture*) は，ハードウェアで様々なレベルのリアルタイム処理をサポートしている．*RMT Architecture* は，SMT [5] [6] に近い細粒度マルチスレッディング機構と優先度を融合したリアルタイム処理向けマルチスレッドアーキテクチャであり，以下のようなな特徴を有している．また，*RMT PU* のインストラクションセットは MIPS 上位互換である [7]．

- RMT アーキテクチャ
  - 8 スレッド同時実行（優先度付 SMT）
  - 256 レベルの優先度
  - 32 スレッド格納可能なコンテキストキャッシュ
  - ハードウェアによるコンテキストスイッチ
  - 優先度を用いた全てのファンクショナルユニットの制御
  - 割り込み発生時のハードウェアによるスレッドの制御
  - 共有レジスタを用いた複数スレッド間のデータ共有
  - 同期ユニットを用いた複数スレッド間の同期
- 高性能ベクトルプロセッサ
  - 複数スレッドによるベクトルユニットの共有
  - 柔軟な複合演算のサポート

これらの特徴的な機能によって，柔軟なリアルタイム処理を実現している．

我々はリアルタイム性の実現が目的であり，絶対性能にはあまりこだわらないが，*RMT Processor* は単なるマイクロプロセッサとしてもある程度の性能を有している．チップが 300[MHz] で動作時のピークパフォーマンスを表 1 に示す．マルチスレッディング機構により，表 1 の性能を同時に引き出すことが可能である．

RMT PU は上記の機能を実現するために非常に複雑な機構を有しているので，命令供給機構の詳細設計及び評価に関しては文献 [8] を，命令実行機構の詳細設計及び評価に関しては文献 [9] を参照されたい．

4.4 バス構成

RMT Processor は非常に多くの I/O ペリフェラルをチップ

上に内蔵し，それらを内部バスで接続したシステムLSIである(図2参照).

*RMT PU*とメモリの間は広いバス幅で接続して高いバンド幅を確保した方が性能を向上させる事ができるが，I/Oの多くは32bit以下のデータ幅しか必要としない．そこで内部バスを各モジュールが必要とするデータバス幅によって256bitメモリバスと32bit I/Oバスに分け，それらをデータサイジング機構を用いて結合する．この機構によって各モジュールはバス幅の違いを気にすることなくデータ転送を行うことができる．またI/Oからメモリへのデータ転送をより効率的に行うため，このデータサイジング機構の機能を有するDMAの設計を行っている．

4.5 キャッシュ

従来のリアルタイムシステムにおいては，キャッシュは不必要なものであった．キャッシュのヒット時とミス時の性能差が非常に大きいので，リアルタイム処理の時間予測性を妨げるものとして悪者扱いされ，WCET(Worst Case Execution Time)を用いるリアルタイムシステムにおいては，キャッシュをオフにして使用されたり，キャッシュオンの場合でも，全てキャッシュに外れる場合の実行時間を元にスケジューリングが行われていた．しかしながら，現在の高性能プロセッサにおいてキャッシュは必要不可欠なので，我々はキャッシュにも優先度機構を儲け，性能と時間予測性の両立を目指す．

ここで，*RMT Processor*では同時に実行される複数のスレッドがメモリアクセスを行うため，キャッシュメモリへ大きな負荷がかかる．しかしそのような状況においても優先度の高いスレッドの実行速度を維持するために，キャッシュエントリの入れ換えにおいては優先度の低いエントリから入れ換え対象としていく機能を設計した[10]．また*RMT PU*から内部バスへアクセスを行うためのバスインタフェースユニットでは，最大16個までの要求をキューに保持することができる．この時，優先度の高いスレッドの要求は，先にキューに存在している他の要求よりも優先してバス権を獲得することができるようにした．

一方，多数のI/Oからデータが転送されるため，キャッシュされているデータの無効化は効率的に行われなければならない．また，シノニムを防ぎ，複数スレッドのキャッシュデータの識別とデータの共有を効率的に行うことも考慮すると，物理キャッシュを用いた方が良いと考えられる．そこで，キャッシュをアクセスする前にアドレス変換を行うために，プロセッシングコアとキャッシュの間にMMUを備えた物理キャッシュ構成を採用している．

4.6 低消費電力機構

*RMT Processor*には高性能・リアルタイムな演算・通信と低消費電力の相反する要求がある．ここで，消費電力*Power*は，以下の式で求めることができる．

$$Power = f \times P \times C_{load} \times V_{dd}^2$$

電源電圧V_{dd}の2乗に比例して消費電力は大きくなるので，V_{dd}を低くするとチップ全体の消費電力が大きく下がる．我々は，V_{dd}が1.0[V]という超低電圧プロセスを用いることによって，チップ全体における消費電力を大きく削減する．

次に，周波数fを用いた電源管理ユニットを設計した．電源管理ユニットから階層的にクロックツリーを張り，全ての機能ユニットの周波数を256段階に動的に独立して変更できるよう設計を行った．さらに，電源管理ユニットで機能ユニット毎にクロックを停止させること及び局所リセットを行うことも可能にした．これらの機構によって，速度と周波数のトレードオフを考慮した動作をソフトウェア的に実現することが可能である．この際，クロック分配の機能ユニットの単位としては，I/Oは種類毎に，*RMT PU*内は大きな機能ユニット単位に行っている．

例えば，*RMT Processor*はオンチップに様々なI/Oを集積しているが，アプリケーションとして使用しないI/Oのクロックは静的に停止することによって，消費電力を削減できる．あるいは，スカラ演算しか行わないスレッドしか動いていない場合，大きな電力を消費するベクトルユニットのクロックを動的に停止する．ベクトル演算を行う場合は，再度，動的にクロックを供給する．

また，リアルタイム処理においては，最高速度で動作する必要はなく，そのタスクに必要な速度で動作すればよい．例えば，MPEGのデコードを行う際には，そのリアルタイム処理に間に合うようにクロックを落とすように電源管理ユニットを制御することによって消費電力を削減することができる．

このように，*RMT Processor*では，電源管理ユニットをソフトウェア(RT-OS等)が細かく管理することによってリアルタイム処理と低消費電力を両立する．

4.7 ファブリケーション

*RMT Processor*の設計においては，我々はレイアウト等のバックエンド設計まで全て行っている．我々は，GDS IIの設計及び検証(DRC等)まで行い，チップ製造に関しては，以下のようなTSMCの最先端プロセスで製造を行った（図3参照）．

- 製造メーカ：TSMC
- プロセスルール：0.13[μm] CMOS 8層Cu配線
- ゲート数：約14Mゲート
- 動作電圧：
  - Core: 1.0[V]
  - I/O: 2.5[V]
- ダイサイズ：10.0[mm] x 10.0[mm]
- チップサイズ：4.0[cm] x 4.0[cm] BGA

5. 評　価

*RMT Processor*の各種機能の評価を全て述べることはできないので，ここでは*RMT Processor*の重要な機能の一つである*RMT PU*のリアルタイム性能の評価を示す．

5.1 コンテキストスイッチ

まず，コンテキストスイッチのオーバーヘッドを評価する．

*RMT PU*のハードウェアコンテキストをメモリに書き込み，また読み出すというプログラムを実行し，その実行結果をソフトウェアで行うコンテキストスイッチのオーバーヘッドとする．ソフトウェアコンテキストスイッチと*RMT PU*のスレッドス

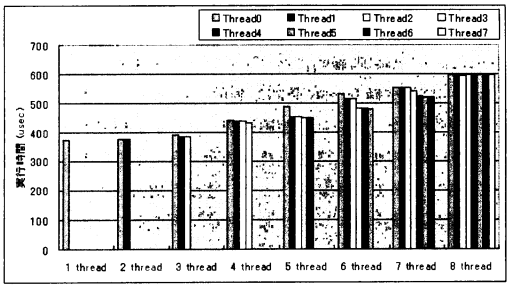

図 4 優先度を用いない場合のソートの実行時間
Fig. 4 Execution Time without Priority

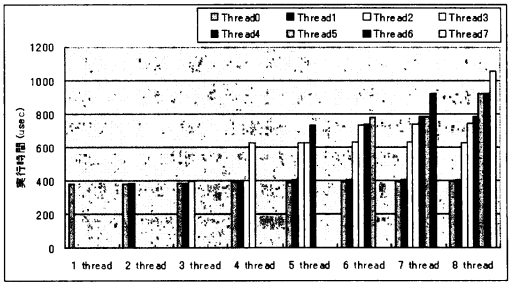

図 5 優先度を用いた場合のソートの実行時間
Fig. 5 Execution Time with Priority

ワップ命令（SWAPTH 命令）によるコンテキストキャッシュとハードウェアコンテキスト間のコンテキストスイッチのコストを以下に示す．

ソフトウェアコンテキストスイッチ 590 クロック (セーブ 277 +リストア 313)

ハードウェアコンテキストスイッチ 4 クロック

このように，*RMT Processor* はソフトウェアで行うよりも非常に高速なコンテキストスイッチ機能を提供している．現在開発されているような RT-OS と異なり，コンテキストスイッチの時間粒度や回数の制約にとらわれない RT-OS の開発などが可能になると考えられる．

5.2 リアルタイム処理性能

次に優先度を用いたリアルタイム処理の評価を行う．評価プログラムにはクイックソートを用いている．図 4 に優先度を用いない場合の実行時間，図 5 に優先度を用いた場合の実行時間を示す．各スレッドは全く同じクイックソートを同時に実行を開始している．優先度を用いる場合は Thread 0 の優先度が最も高く，段階的に Thread 7 の優先度が最も低くなるように設定した．図 4 の優先度を用いない場合，スレッド数が増加するに従って各スレッドの実行時間が徐々に増加した．これはスレッド数が増えると演算ユニット等の資源の競合多くなるためと考えられる．しかしながら，トータルの演算性能は向上している．

一方，図 5 の優先度を用いる場合，優先度の最も高いスレッド (Thread0) の実行時間は同時実行スレッド数が増えても約 400[μsec] でほぼ一定であり，1 スレッドで実行したときとほとんど変化がないことが分かる．同様に，他の優先度のスレッドについても，その優先度に従ったある一定時間以下での処理時間が実現されていることが分かる．例えば，Thread2 の処理時間は，約 610[μsec] 以下であることが分かる．

このように，*RMT Processor* では優先度に従った処理時間の保証を実現している．

6. ま と め

本論文では，並列分散リアルタイム処理用 *RMT Processor* の全体設計について述べた．*RMT Processor* は，並列分散リアルタイム処理に必要な機能のほとんどを 1 チップに集積した大規模システム LSI であり，具体的には，1 チップにリアルタイム処理機能 (*RMT PU*, Cache, etc.)，リアルタイム通信機能 (*Responsive Link II*)，コンピュータ用周辺機能 (DDR SDRAM I/Fs, DMAC, PCI64, USB2.0, IEEE1394, etc.)，制御用周辺機能 (PWM Generators, Pulse Counters, etc.) を集積している．*RMT Processor* は，リアルタイムスケジューラにより優先度付けされた処理や通信を，付与された優先度に従い，処理や通信の追い越しまたは調停をハードウェアで行うことによって，処理時間および通信時間の保証を可能にした．この機能によって，リアルタイム処理の時間粒度を非常に小さく高精度に行うことを可能にし，大規模な並列分散リアルタイムシステムを構築可能にした．

謝 辞

本研究は文部科学省の科学技術振興調整費の支援により行われた．また *RMT Processor* の一部は科学技術振興事業機構 CREST の研究成果でもある．

文 献

[1] J. A. Stankovic: "Misconceptions about real-time computing", IEEE Computer, pp. 2–10 (1988).
[2] J. W. S. Liu: "Real-time systems", Prentice Hall, pp. 159–179 (2000).
[3] 山崎, 松井 : "並列分散リアルタイム制御用レスポンシブプロセッサ", 日本ロボット学会誌, **19**, 3, pp. 68–77 (2001).
[4] http://www.itscj.ipsj.or.jp/ipsj-ts/02 06/toc.htm.
[5] S. J. Eggers, J. S. Emer, L. Henry M, J. K. Lo, R. L. Stamm and T. Dean M: "Simultaneous multithreading : A platform for next-generation processors.", IEEE Micro, **17**, 5, pp. 12–19 (1997).
[6] D. M. Tullsen, S. J. Eggers, J. S. Emer, L. Henry M, J. K. Lo and R. L. Stamm: "Exploiting choice: Instruction fetch and issue on an implementable simultaneous multithreading processor", Proceedings of the 23rd Annual International Symposium on Computer Architecture (1996).
[7] http://www.ny.ics.keio.ac.jp/rmt/.
[8] 薄井, 内山, 伊藤, 山崎 : "Responsive Multithreaded Processor における実時間処理用命令供給機構", RTP2004.
[9] 伊藤, 山崎 : "分散リアルタイムネットワーク用プロセッサとその応用", 情報処理学会論文誌 : コンピューティングシステム, **44**, (2003).
[10] 佐藤, 内山, 伊藤, 山崎, 安西 : "リアルタイム処理用マルチスレッディングプロセッサの優先度に基づくキャッシュサブシステム", 情報処理学会研究報告 2001-ARC-143, pp. 37–42 (2001).