戦略的創造研究推進事業 CREST

研究領域「情報社会を支える新しい高性能情報処理技術」

研究課題「ヒューマノイドのための
実時間分散情報処理」

研究終了報告書

研究期間　平成 15 年 10 月～平成 21 年 3 月

研究代表者：松　井　俊　浩

（独）産業技術総合研究所
デジタルヒューマン研究センター副センター長

§ 1　研究実施の概要

本プロジェクトは、ヒューマノイドロボットの性能および信頼性の向上のための新しい実時間分散型の情報処理を研究開発する。従来のヒューマノイドロボットは、1-2 個のプロセッサによる集中制御が行われていたが、約 30 の関節プロセッサによる分散制御を実現することを目的とする。この並列処理によって、大きな処理性能が低消費電力で得られる。プロセッサをセンサやアクチュエータの近くに配置することで、信号線の延長を大幅に削減し、高い信頼性を実現する。一方、分散制御のため、通信およびマルチプロセス(スレッド)を実時間化し、優先度や資源割り当て等を行う基盤ソフトウェアが必要になる。

本研究は、次の三つから構成される。**①RMTP の設計・開発**:実時間性にすぐれたマルチスレッド向きアーキテクチャ、低クロックでも性能を発揮するための並列性、実時間通信チャネルを備えたプロセッサを作成する、**②実時間基盤ソフトウェア**:Linux-2.6 を拡張して優先度に基づく実時間スケジューリングを可能にし、プログラムの実行時間を静的に予測する技術の研究。**③ロボット実証**:認識、対話、歩行、経路計画などの機能をモジュラーに構成することで分散化し、並列処理を活用して性能向上を図る。

- **Responsive Multithread Processor (RMTP)の研究開発**

本研究では、ヒューマノイドに組み込んで分散実時間制御を実現するために、実時間性をハードウェアレベルでサポートするマルチスレッドプロセッサアーキテクチャ、低クロックでも画像処理や音声処理等のマルチメディア演算に対して高性能を発揮する並列性の高いベクトル演算アーキテクチャ、実時間通信リンク、SoC (System-on-Chip)化及び SiP (System-in-Package)化に関する研究開発を行う。それら個々の研究成果を融合し、さらに I/O 制御や各種省電力機能等を搭載した分散実時間制御用プロセッサである Motion Responsive Multithreaded Processor (M-RMTP)をシステム LSI 技術を用いて実現する。M-RMTP は、ヒューマノイドの分散リアルタイム制御を容易に行うために、実時間演算処理機能(RMT PU)、実時間通信機能(Responsive Link)、コンピュータ用周辺機能(DDR SDRAM I/Fs, DMAC, PCI-X, IEEE1394, RS-232C, I/O, etc.)、制御用周辺機能(PWM generators, PWM input, Pulse counters, etc.)を 1 チップに集積している(図 1-1 参照)。

RMT PU は、リアルタイム処理を行う際に、明示的なコンテキストスイッチを行うことなく、RMT 実行(優先度付同時マルチスレッド実行)に変換してハードウェアでオーバヘッドなしにリアルタイム処理を実現する。256 レベルの優先度付スレッドを 8 スレッド同時に実行し、さらに 32 コンテキスト分のコンテキストキャッシュを内蔵しているので、オンチップ上で仮想的に 40 スレッドを同時実行可能である。それ以外にも、割り込みによるスレッドを起床する機能やマルチメディア演算ユニット(2 次元ベクトル演算ユニット(Int, FP))等の特徴的な機能を有

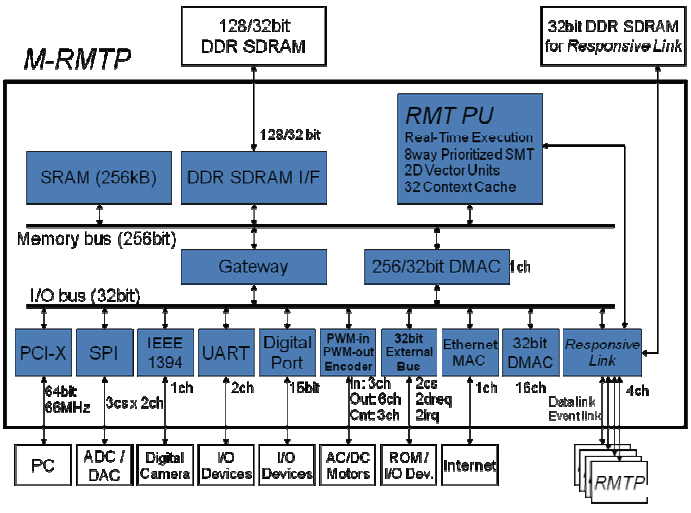


図 1-1 M-RMTP のブロック図

している。

また、Responsive Link は、通信におけるプリエンプションを、優先度によるパケットの追い越しという技術により世界で初めて実現し、リアルタイム通信をハードウェアで実現した。さらに、ISO/IEC において国際標準化を行い、ISO 24740 という国際標準規格にした。

M-RMTP の設計は、論理設計からレイアウトまで慶應義塾大学山﨑研究室で実施した。製作は世界最大のファウンダリメーカである TSMC に委託し、0.13μm、CMOS8 層の銅配線で、9.98 ㎜角のダイを作成した。M-RMTP では、非常に多数の IO ピンを入出力し、かつ SiP (Sysem-in-Package)実装を容易にするため、チップ表面に極小のハンダボールを直接実装するフリップチップという最先端技術を用いて IO ピンを設計し、1040 ピンもの多数のピンをチップ上に実装している(図 1-2 参照)。M-RMTP をロボットの体内に組み込むために、超小型のシステムボードである M-RMTP SiP を開発した(図 1-3 参照)。M-RMTP SiP は、M-RMTP ベアダイ、DDR SDRAM x 4、Flash memory x 2、Ethernet Phy、レギュレータ、各種受動素子を、わずか 33mm 角に集積している。

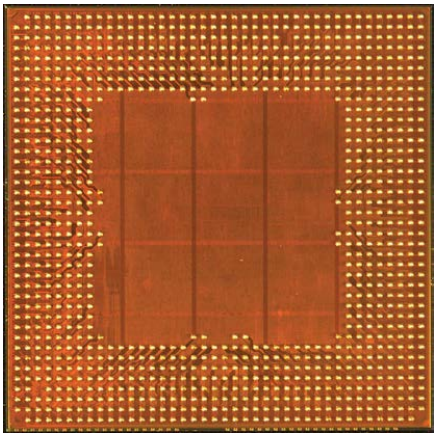


図 1-2 M-RMTP のダイ写真とフリップチップ

図 1-3 M-RMTP SiP

- **実時間分散基盤ソフトウェア**

RMTP のアーキテクチャを活用しつつ、他のプロセッサにも適用可能な実時間オペレーティングシステムと実時間性解析ソフトウェアツールを開発する。

ロボットには、低レベルのサーボから、実環境の認識、行動の計画など広範なソフトウェアを必要とする。そのため、ロボット用のオペレーティングシステムは、専用とすべきではなく、多数のロボット研究者が開発プラットフォームにできる、汎用の OS を採用すべきである。そのため、汎用のオープンソース・オペレーティングシステムである Linux をベースにして、実時間制御向きの拡張を加える。実時間性の追求は、さまざまな方法がある。クリティカルセクションのロックや割り込みの取り扱いに新規なアイデアを採用する実験的な色彩の強い Shi-Linux と、従来ロボット制御の一定の実績のある ART-Linux の改良の二つの研究開発を行う。

実時間オペレーティングシステムは、ロボットプログラムの実行時の実時間保証を目的としているが、実時間性は、プログラムの性質とプロセッサの負荷との関係に大きな影響を受ける。たとえば、プロセッサにとって過負荷となるようなプログラムセットを与えたのでは、実行時に実時間処理の安定性が損なわれ、ロボットの転倒事故につながる。実行しようとしているプログラムが、デッドライン(周期)を守れるかどうかは、ソフトウェアの開発時点で検査できるのが望ましい。そのようなプログラムの実時間性を解析するために、WCET(最悪実行時間)解析ツールを開発する。プログラムのソースコードから、所用実行時間を予測し、与えられた周期で実行できることを保証する。

- **ロボット実証**

上記で開発するRMTPおよび実時間基盤ソフトウェアをヒューマノイドロボットに搭載し、より安定して高機能のロボットの実時間・分散制御を行う。従来の集中型制御に変わって、分散制御を行うための、関節に搭載する小型の制御ノードの設計・開発、これらを用いた分散制御型のヒューマノイドを設計・試作する。ヒューマノイドロボットには、産業技術総合研究所と川田工業のHRP-3をベースに改造を加える。

RMTPは、クロックは遅いが、マルチスレッドとベクトル化による並列化によってスケーラブルなパフォーマンスを発揮する。そのために、サーボ、画像処理、動作計画などのロボットソフトウェアの並列化を行う。最終的に、従来の1/5～1/10の高速のサーボループによって疑似力制御が可能なヒューマノイドが安定な歩行を行えることを目標とする。

- **開発の経緯**

最初に、ロボットに必要な計算能力の予測と実現可能なRMTPの性能から、分散型のヒューマノイド制御を構想した。実時間性能が与える影響、現在のヒューマノイドロボットの問題と限界、ハードウェアアシストの可能性などを検討した。その後、RMTPの開発、基盤ソフトウェアの開発、ロボットの設計のそれぞれを、三つのグループに分かれて開始した。

RMTPの基本アーキテクチャは、2003年までにほぼ完成していたが、詳細な設計は、バックエンド(レイアウト)設計まで含めると1年くらいを要する。LSIの試作に2ヶ月、テストと選別に1ヶ月、パッケージやSIP、評価ボードへの組み込みに2-5ヶ月程度を要するので、1回の試作を完了するまでに1.5年程度を要する。最初のRMTPであるμRMTP-1の試作が完了したのは、2006年(3年次)の1月であった。その間、並行してRMTPの命令レベルシミュレータの開発、RMTP用コンパイラの整備と、このシミュレータを用いてLinuxの移植(RMTPコードの生成)を行った。2006年夏までに、μRMTP-1の実機を用いてLinuxを動作させる努力をしたが、投機実行とキャッシュに大きな問題が判明し、割り込みで誤動作することがわかった。並行して開発していたSiPにも実装上の問題が判明した。

バグのないRMTPとするために、大量のテストベクタを作成して特にキャッシュ、割り込み、ページ例外の検証を行い、問題を修正した。RTLシミュレーションでLinuxが起動することまで確認して、M-RMTP-3(NEDOプロジェクトで試作を継承したので番号が変わっている)の試作を行った。テープアウトが2008年1月、チップ納品が3月末に行われ、その後は、パッケージングと評価ボードでのテストを行っている。この試作が最終年度に入ってしまったので、2008年にはヒューマノイドロボットをRMTPで動作させることは断念し、実時間ソフトウェアの研究とロボット開発は、RMTPとは切り離して実施した。

- **成果**

以下に主な成果を列挙する。

① 実時間・分散向きの新規なプロセッサアーキテクチャ RMTP を提案し、Linux がほぼ動作するレベルまでの実装を行った。
② 実時間分散制御に必要な通信機能を Responsive Link として開発し、ISO/IEC-24740 として国際規格化を達成した。
③ Linux に実時間スケジューリング機能を実現した。各種のプロセッサとメモリモデルに基づいて C と C++プログラムに適用可能な WCET 解析ツールを開発した。
④ ヒューマノイドロボットの実時間分散制御のために、関節に分散する計算ノードによって高信頼で負荷分散の可能なアーキテクチャを提案し、ノードモジュールを開発した。

§2 研究構想及び実施体制

(1) 研究構想

我が国では、ホンダが 1996 年に等身大のヒューマノイド(人間型)ロボットの流麗な 2 足歩行に成功し、それ以来ヒューマノイドロボットの研究開発が急速に進展した。それまでの代表的なロボットであるマニピュレータや車輪移動型のロボットと異なり、ヒューマノイドロボットの制御には、次のような特徴がある。

① 自由度数: 従来ロボットの数倍から 10 倍もの多数の関節を制御する
② 実時間制御の安定性: 全関節が1ms 以下の周期で安定にサーボされる必要がある。サーボが不安定になるとロボットは転倒する。情報処理系全体の高信頼化が必要である
③ 多機能性: 視覚や聴覚の認識機能、手足の多様な動きを使って、多様な動作と機能を実現する。そのための認識機能と動作計画機能には、大きな計算パワーを必要とする
④ 自律性: バッテリまで含めて self-cotained である。ネットワークから大きな計算パワーが得ることはできない。低消費電力化と超小型実装が求められる

これらの要請にもかかわらず、ヒューマノイドロボットの情報処理系には、一般的な PC に実時間 OS を組み合わせた集中制御系がとられてきた。この方法では、近い将来、計算能力、信頼性、実時間性、省電力性などで限界が来るのが明白である。そこで、本プロジェクトでは、これらの問題を抜本的に解決する、ヒューマノイドロボットの実時間・分散制御系を構想した(図 2-1)。ヒューマノイドロボットの高性能化を目指して、サーボや計算を行うプロセッサ、実時間制御のための基盤ソフトウェア、並列・分散型のヒューマノイドロボット制御の三つの研究を行う。以下の情報処理機能を目標とする。

1. RMT プロセッサによるスケーラブルな処理能力
   - Pentium と μRMTPx10PE による並列・分散処理
2. 実時間制御
   - 軌道生成 5ms → 1ms
   - サーボ 1ms → 0.2ms
   - 通信 全身が 0.1ms で同期
3. 高信頼・耐故障性
   - 信号配線長を 1/20、コネクタ数 1/10
   - 冗長ネットワーク
4. 低消費電力
   - 情報処理系で 80W 以下
5. ロボット実証
   - ならい動作、力制御、高度な認識、計画機能

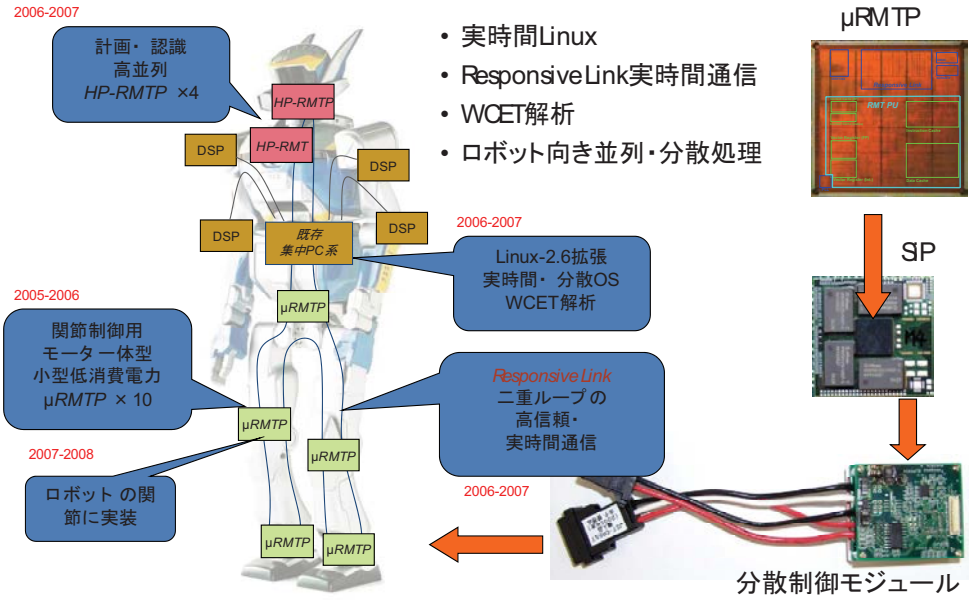


図 2-1 研究開発の計画

情報系全体は、ロボット特有の実時間性の実現を第一の目標とする。Pentium 等の汎用プロセッサが、実行時間の予測性やタスクスイッチのペナルティの犠牲を払って、マルチステージのパイプライン、多段キャッシュメモリ、投機実行などを導入することでスループットの向上を図っているのに対し、実世界の外部イベントに即応できるようなプロセッサ、RMTP (Responsove Multi-Threaded Processor) を開発する。RMTP は、ロボットに搭載するという物理制約を満たすために、ロボット制御に必要な IO 機能をも集積し、システム・オン・チップとして省電力化と小型実装を図る。省電力化のためには、クロックを落とし、それによって電源電圧を下げる。そして性能の低下に対しては並列性の増大で対抗する。並列性は、ベクトル化とマルチスレッドの 2 種類を用意する。

実時間性を保証する基盤ソフトウェアとしては、プログラムの実行時間を静的に予測する WCET (最悪実行時間) 解析と、実行時にプロセスの優先度に応じてプロセッサを割り当てる実時間スケジューラを研究する。

ロボット実証として、RMTP による関節制御ノードを開発し、関節ごとにプロセッサが配置されるアーキテクチャでの分散的なロボット制御を研究する。通信によって全身の数十のノードが同期するために、Responsive Link の実時間通信を活用する。

プロジェクトは、上記のプロセッサ開発、基盤ソフトウェア開発、ロボット実証の三つのグループに分かれて実施した。3 つの要素は、前者に後者が積層され、プロセッサ開発に他のすべての研究が依存する危険性がある。3つの研究を逐次的に接続するのでは、研究期間が不足するので、極力並行して開発が進められるように配慮した。具体的には、最初にプロセッサのソフトウェアシミュレータを開発し、プロセッサのハードウェアを開発する期間中に、シミュレータを用いて Linux の移植・開発作業が進められるようにした。同様に、プロセッサの完成を見越して、関節ノードやインタフェースカードの開発を並行して進めた。

当初は、μ と HP の 2 種類のプロセッサの開発を計画した。μ は、関節プロセッサ用で、パフォーマンスよりは、実時間性、省電力性や小型化に特徴がある。HP は、バス幅を広くし、クロックを上げ

たハイパフォーマンス版で、ロボットの中央に配置して、画像処理などの重い負荷の計算を担わせる予定であった。5 年間のプロジェクトの 2 年目には最初の μ-RMTP チップができ、3 年目にはその改良版ができて HP 版の設計に取りかかる計画であった。

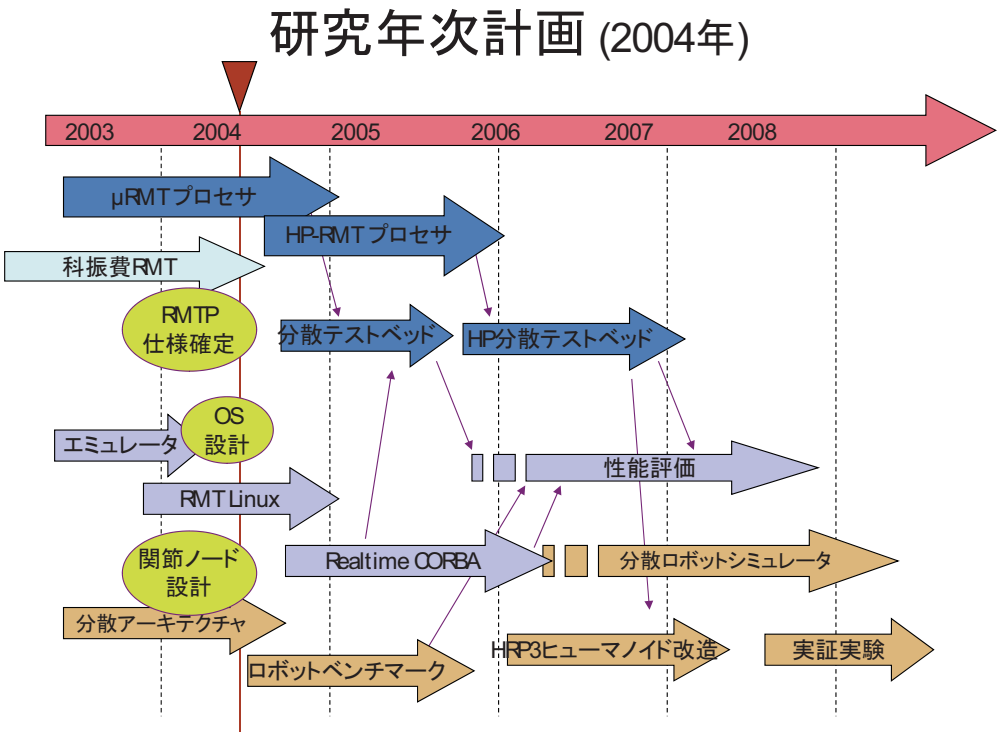


図 2-2 2004 年の研究スケジュール

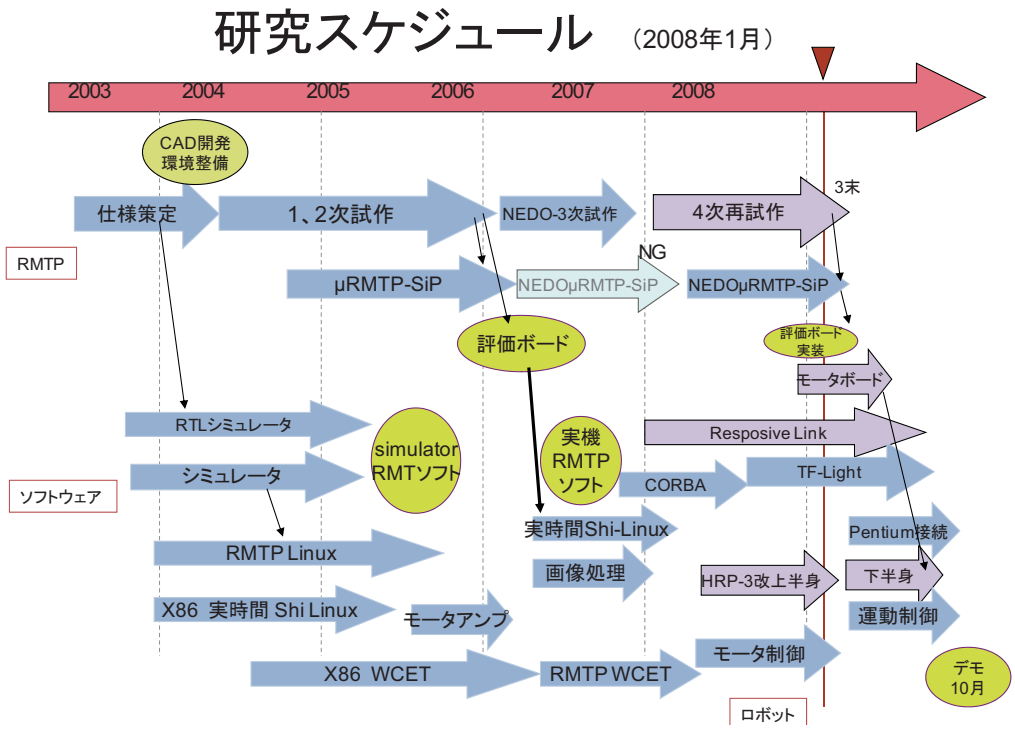


図 2-3 2008 年の研究スケジュール

図に 2004 年と 2008 年の研究スケジュールを示す（図 2-2, 2-3）。RMTP の第 1 版ができたのが 2006 年 1 月であるが、投機実行、割り込み、キャッシュなどに致命的問題が残存しており、コンパイラによる回避を試みたが、その範囲を超えていた。その後、関節制御ボードや RMTP 用 CORBA の開発は着々と進めたが、2 回目の試作が完了するのが 2008 年の夏になることが判明した時点で、ロボットへの RMTP システムの組込は断念した。RMTP は、Linux を動作させることを目標に再設定し、基盤ソフトウェアの実装とロボット実証は、Pentium を用いて行うこととした。

なお、HP-RMTP の作成にとりかかることはなかったが、プロセッサ中の資源を優先度の異なるタスク(スレッド)に割り当てる方法として、スレッドが実行できる命令の密度が与えられた値となるようにさまざまな資源の割り当てをフィードバックするアイデアが生まれ、特許を出願した。

RMTP の試作が困難を極めたのは、SoC として IO ピンが多いため、パッケージが高密度になり、一挙に SiP (System-in-Package)として高密度化をねらったために、各種の問題の解決が困難になったからである。2，3回の試作でたどりつくレベルとしては、比較的バグの少ない状態に達していると考えるが、パッケージングに多大な時間を要したため、5 年間で2，3回の試作にとどまり、実用レベルに達することができなかった。より確実な方法として、FPGA などによるシミュレーションを綿密に行う方法もあるが、現時点ではこれだけ高密度のプロセッサをシミュレーションできる FPGA の作成には、やはり多大な時間とコストを要するので、問題の本質的解にはならない。

実時間オペレーティングシステム、WCET 解析などは、RMTP アーキテクチャに深く依存しないように研究を進めたため、良い成果を得ることができた。ヒューマノイドロボットについても、DSP による分散型の制御を実装することができた。

(2) 実施体制

グループ名	研究代表者 又は 主たる 共同研究者 氏名	所属機関・部署・役職名	研究題目
システム LSI グループ	山﨑 信行	慶應義塾大学大学院	実時間通信・処理・制御用システムオンチップの研究開発
基盤ソフトウェアグループ	石川 裕	東京大学大学院情報理工学系研究科・教授	実時間オペレーティングシステムと開発環境の開発
ロボット実証グループ	松井 俊浩	産業技術総合研究所	ヒューマノイドロボットの分散制御系の研究
ロボット実証グループ	比留川 博久	産業技術総合研究所	ヒューマノイドロボットの分散制御系の研究
ロボット実証グループ	加賀美 聡	産業技術総合研究所	ヒューマノイドロボットの分散制御系の研究
ロボット実証グループ	金子 健二	産業技術総合研究所	ヒューマノイドロボットの分散制御系の研究
ロボット実証グループ	金広 文男	産業技術総合研究所	ヒューマノイドロボットの分散制御系の研究

§ 3　研究実施内容及び成果

3.1.　実時間通信・処理・制御用システムオンチップの研究開発（慶應義塾大学理工学部システム LSI グループ）

(1) 研究実施内容および成果

① Responsive Multithreaded Processor (RMTP)の概要

ヒューマノイドロボットのような大規模な組み込み系で、分散型の実時間情報処理を行うためのプロセッサを SoC (System-on-Chip)として研究開発する。ヒューマノイドロボットの各関節を制御するための小型プロセッサと、画像処理やプランニング等の大きな処理能力を必要とするプロセッサを、同一 ISA (Instruction Set Architecture)の 2 種類のプロセッサとして開発する。前者を μ RMTP、後者を HP-RMTP と呼ぶ。Responsive とは、実時間性を維持しながら、可能な限り外界のイベントに即時に応答する性質を指す。ここで、当初の研究計画では、4 年次までに HP-RMTP を作成する予定であったが、μ RMTP の開発が種々の理由（μ RMTP のバグ及び SiP 開発の遅れ等）により大幅に遅れてしまったので、マルチコアで高性能な HP-RMTP を作成する代わりに、μ RMTP の発展系である M-RMTP (Motion Responsive Multithreaded Processor)の設計・実装を行った。M-RMTP には、実時間演算処理機能(RMT PU)、実時間通信機能(Responsive Link)、コンピュータ用周辺機能(DDR SDRAM I/Fs, DMAC, PCI-X, IEEE1394, RS-232C, I/O, etc.)、制御用周辺機能(PWM generators, PWM input, Pulse counters, etc.)を 1 チップに集積している（図 3-1-1 参照）。

コンパイラ、オペレーティングシステム、ライブラリ等の開発を容易にするために、RMT PU の命令セットアーキテクチャは、代表的な RISC プロセッサである MIPS の上位互換とした。

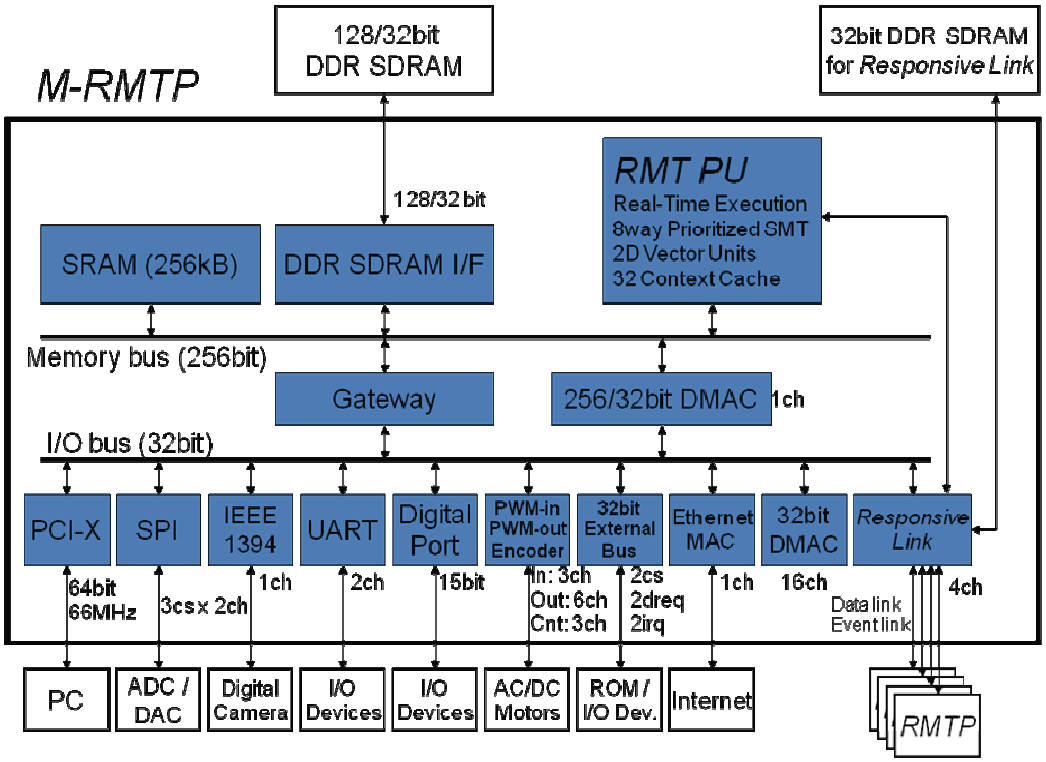


図 3-1-1 M-RMTP のブロック図

② RMT 機構（優先度付同時マルチスレッディング）

RMT PU の最も特徴的な機能は、ハードウェアによる実時間実行機構（RMT 機構）にある。通常のプロセッサで実時間処理を行う場合、従来はリアルタイムスケジューラが時間制約に基づき決定した優先度に従い、OS がプリエンプション（ソフトウェアによるコンテキストのセーブとリストア）を行っていたが、RMT PU ではソフトウェアによる明示的なコンテキストスイッチを行うことなく、RMT 実行（優先度付同時マルチスレッド実行）に変換して実時間実行を実現する。図 3-1-2(a)に通常のプロセッサにおけるリアルタイム処理（スケジューリング）の様子を示す。カーネルがコンテキストスイッチを処理するには、レジスタの退避と復帰、メモリ空間の切り替えなどが必要であり、その間は割り込みを受け付けられなくなる。また、命令ストリームが切り替わることにより、命令パイプラインやキャッシュのフラッシュが必要となる。このために、Pentium 系のプロセッサでは数千クロックの処理時間が必要となる（図 3-1-2(a)の System の部分）。ロボットのモータサーボでは、ユーザモードタスクの一回の処理は数百クロック程度の短時間で繰り返されることが多く、カーネルのコンテキストスイッチ時間は非常に大きなオーバヘッドとなる。

これに対して、RMT PU では、図 3-1-2(b)に示すように、リアルタイム処理を行う際に明示的なコンテキストスイッチを行うことなく、RMT 実行（優先度付同時マルチスレッド実行）に変換してハードウェアでオーバヘッドなしにリアルタイム実行を実現している。プロセッサは最大 8 スレッドのコンテキストをアクティブにすることができ、各々のスレッドには時間制約に基づいた優先度が付けられる。優先度の高いスレッドが優先的に演算器等の資源を使用し、余った資源を用いてより低い優先度のスレッドが同時実行される。これらの処理は、すべてハードウェアで行われるので、カーネル(ソフトウェア)の介入は不要であり、コンテキストスイッチのオーバヘッドはゼロとなる。つまり、RMT PU の

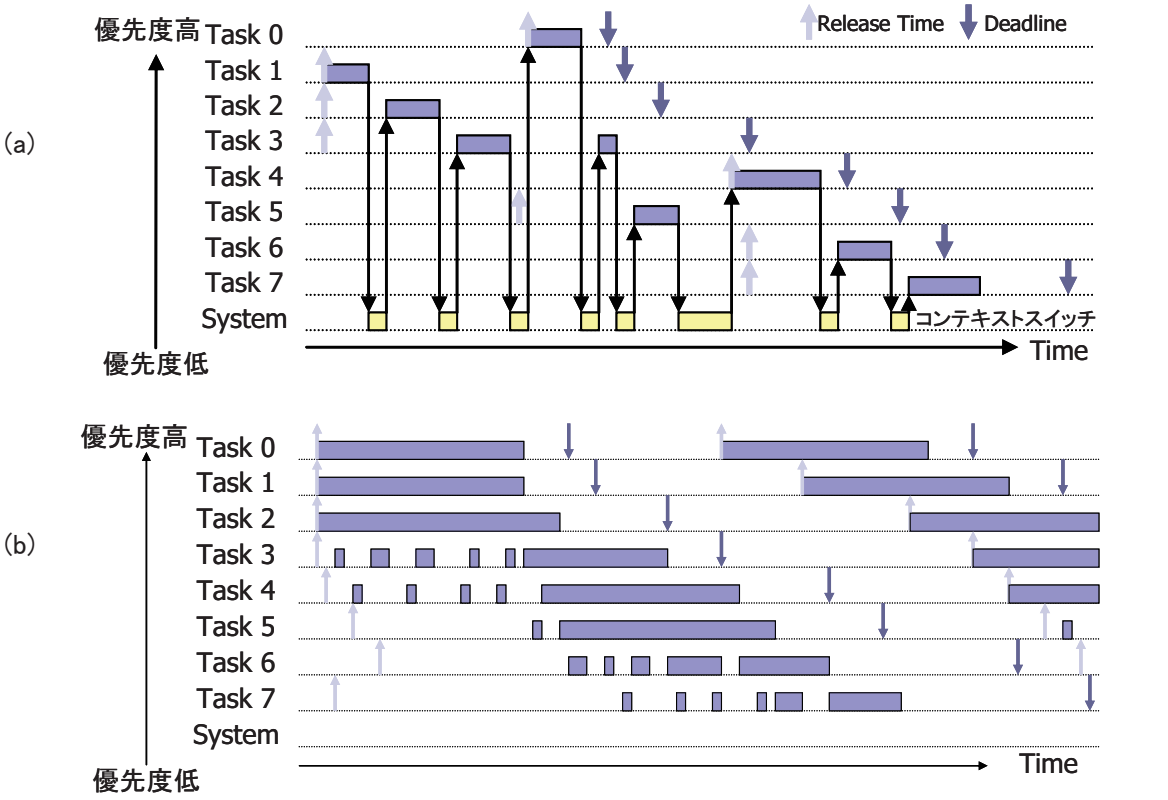


図 3-1-2 通常のシングルスレッドプロセッサでの実時間実行(a)と RMTP の実時間実行(RMT 実行) (b)

8 組のハードウェアコンテキスト（レジスタセット等）は、あたかもマルチプロセッサ用 RT-OS のタスクキューのように振る舞う。

図 3-1-4 に RMT PU のブロック図を示すが、この図中の黄色い機能ユニットにおいて、優先度制御を行っている。資源の割り当てアルゴリズムには、実時間性を優先した RMT モード、トータルなスループットを優先して極力実行ユニットの空きを減らす SMT モードがある。さらに、特定のスレッドに対してパイプラインを割り当てる Fixed Issue Slot モード（いわゆる資源予約）がある（図 3-1-3 参照）。

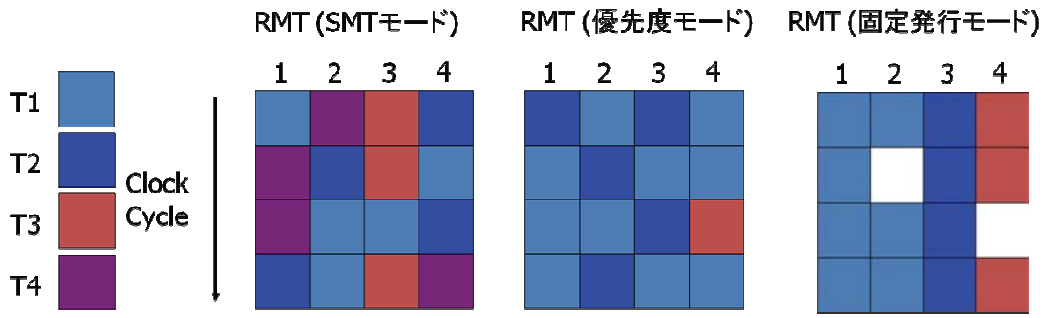


図 3-1-3　RMTP のリアルタイム実行モード （優先度： T1 > T2 > T3 > T4）

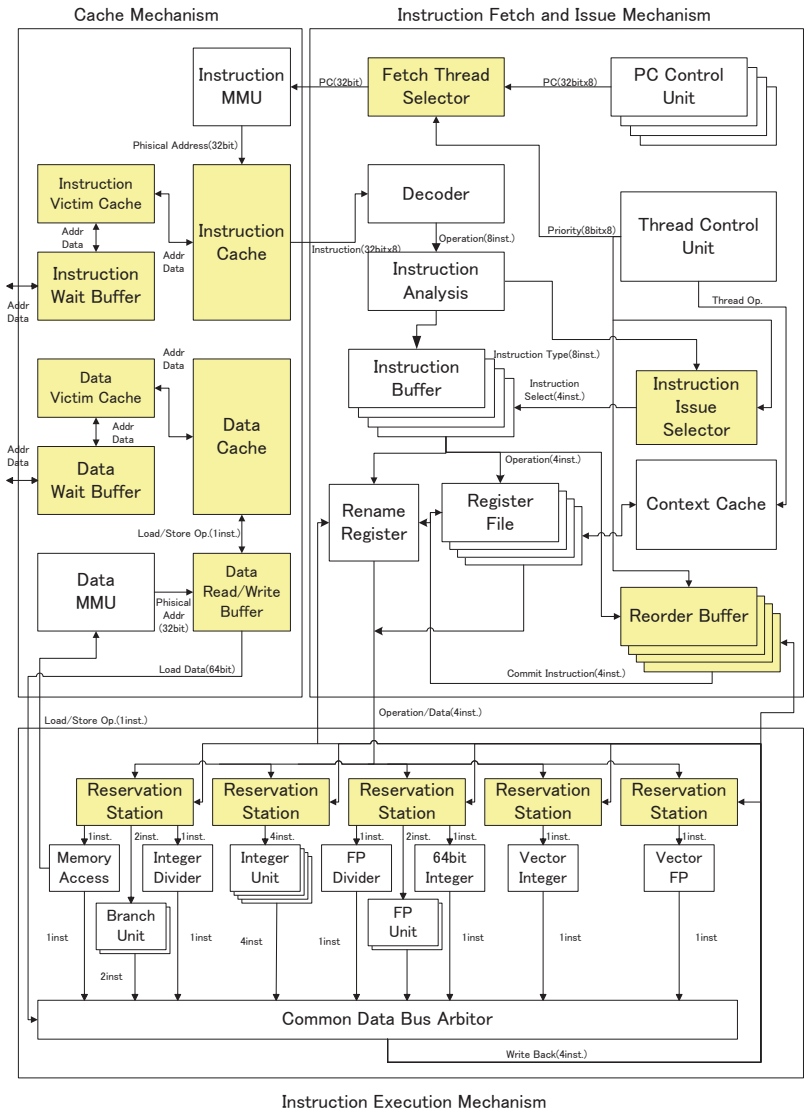


図 3-1-4 RMTP のプロセッシングコア(RMT PU)の構成

通常、リアルタイム処理の研究はソフトウェアで行っており、時間粒度は 1msec オーダである。それに対して、本研究では、リアルタイム処理をハードウェアで行うため、時間粒度 1 μ sec オーダのリアルタイム性を実現可能であり、従来研究とは比較にならな非常に時間粒度の細かいリアルタイム性を実現可能となる。リアルタイム処理のパラダイムシフトを可能とする。

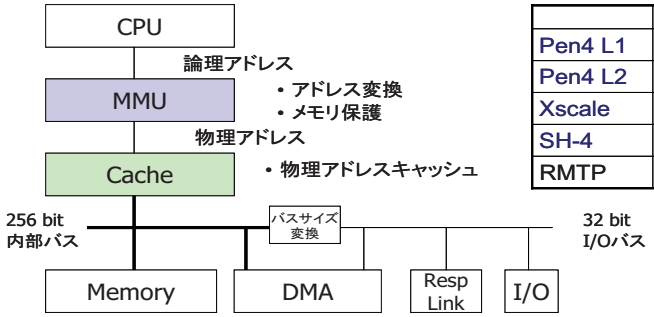
RMT PU は、8 本までのスレッドを同時に実行することができる。この同時実行スレッド数は、ほとんどのロボット応用にとって十分であるが、割り込みの応答性を良くするために割り込みハンドラを別スレッドに割り当てたり、カーネルを複数のスレッドで構成するようなアプローチをとるとスレッド数に不足が生じる。スレッド数不足を補うために RMT PU には、32 スレッド分のコンテキストを保持できるコンテキストキャッシュを備えている。これらは、ハードウェアスケジューリングの対象とはならないが、ソフトウェアのスレッド切り替え命令で 4 クロックでアクティブスレッドとスワップしてアクティベートすることができる。その結果、合計 40 本のリアルタイムスレッドを並行動作させることができる。

③ メモリインタフェース

8 スレッド同時実行のためには、大量の命令ストリームをメモリから絶え間なくフェッチする必要がある。そのためにメモリバンド幅の確保、キャシュの設計が重要となる。大きなメモリバンド幅を確保するために、内部メモリバスは 256 ビット幅として設計を行った。システム小型化の要求の強い M-RMTP では、外部メモリバス(DDR SDRAM IF)は 32 ビット幅と 128 ビット幅のどちらでも動作するように設計を行った。小型化が重要な場合は 32 ビット幅で接続し、性能が重要な場合は 128 ビット幅で接続できる。

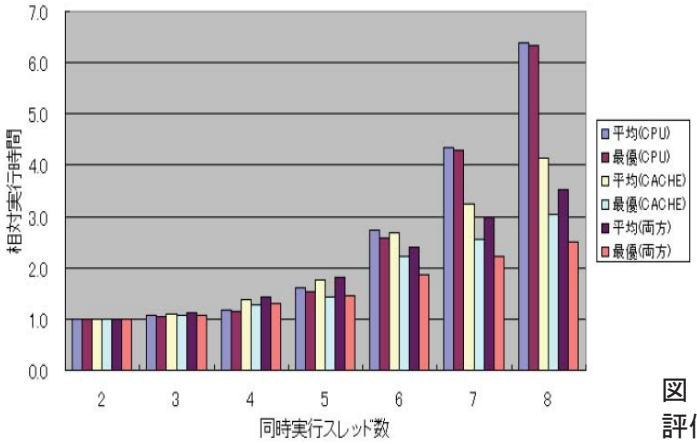
RMTP のキャッシュは、コンテキストの異なるマルチスレッドで散逸しがちなメモリアクセスの性能向上をねらって、組み込みプロセッサとしては大きめの I32KB+D32KB、8-way の 1 次キャッシュを設計・実装している(図 3-1-5 参照)。特徴的なのは、実時間性を実現するために、キャッシュの更新アルゴリズムとして優先度を選択できる点にある(従来方式の LRU も選択可能)。図 3-1-6 に 32KB の配列をソートするプログラムを実行した際のキャッシュミス率を示す。

スレッド数が 8 のとき、4-way ではミス率が 8%であったのが、8way にすることで 1.4%にまで下げることができた。ラインサイズは 32 バイトを基本とするが、ベクトル計算のような連続アクセスが期待されるアプリケーションでは、32 バイトの倍数でラインサイズを変更可能である。また、16 ライン分の victim cache を備え、キャッシュ競合が発生した場合のペナルティの低減を図っている。キャシュメモリは、トータルのスループットの向上に大きく寄与するが、メモリアクセスの順番や散逸性に左右されやすく、特にマルチスレッド環境では他のコンテキストのメモリアクセスによって性能が大きく変わることがある。これは、実時間処理にとって重要な実行時間の予測性を阻害する。そのため、



	I-cache	D-cache	Line size	N-way
Pen4 L1	12Ku	8KB	64B	4-8
Pen4 L2	256KB-1MB		128B	8
Xscale	32KB	32KB		32
SH-4	8KB	16KB	8B?	1-2
RMTP	32KB	32KB	32B	8

図 3-1-5 RMTP のキャッシュメモリの構成



同時実行スレッド数	初期参照ミス(%)	競合ミス(%)	合計ミス(%)
1	0.17	0.01	0.18
2	0.17	0.15	0.32
3	0.16	0.31	0.47
4	0.16	0.35	0.51
5	0.16	0.50	0.66
6	0.16	0.59	0.75
7	0.16	0.70	0.86
8	0.17	1.24	1.41

図 3-1-6 優先度付きキャッシュの性能評価

RMT PU では、追い出しアルゴリズムに優先度を用いるモードを備えると共に、キャッシュの一定量を特定のスレッドに固定的に割り当てるモードも設けている。

④ 2 次元ベクトルユニット

動画処理や音声処理などのいわゆるマルチメディア演算は、実時間性が要求されると同時に、高い演算性能を要求される。また、ロボット応用では、3, 4次元の座標変換、固有値計算、一般逆行列、画像処理などで行列演算が多用される。それらの演算は、基本的にデータ並列性を有しているので、低消費電力で高性能を発揮させるためには、命令アクセスを減らし、かつデータ並列性を利用したベクトル演算が有効である。RMT PU は、単精度および倍精度浮動小数点演算と、バイト／ハーフワード／フルワード長の整数計算のための 2 次元ベクトル演算ユニット、及び各々512 本のベクトルレジスタとを備えている。例えば、8bit 整数演算であれば、1 クロックあたり 32 要素の MAC 演算を行うことが可能な 2 次元ベクトル演算器を 2 系統有している（図 3-1-7 参照）。同様に単精度浮動小数点演算であれば、1 クロックあたり 16 要素の演算を行うことが可能な 2 次元ベクトル演算器を 2 系統有している。

ベクトルレジスタは、全スレッドで共有し、図 3-1-8 のようにさまざまなコンフィギュレーションの中からアプリケーションに適したレジスタ数を各スレッドに確保することができる。

従来のプロセッサアーキテクチャは、スカラーかベクトルかに別れて進化・発展してきたが、本プロセッサでは、スカラーとベクトルをバランスよく融合しており、世界的に見ても非常にユニークなプロセッサアーキテクチャであると言える。

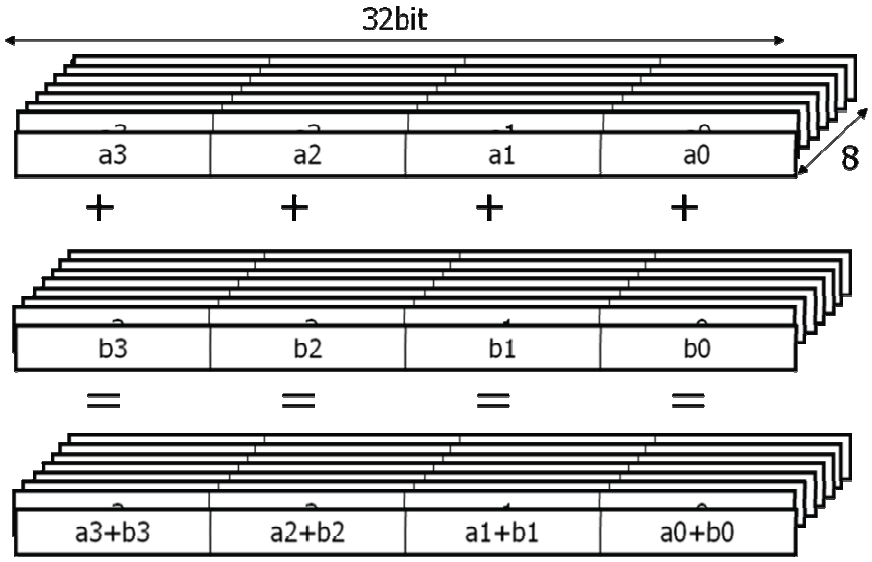


図 3-1-7 2 次元ベクトル演算器

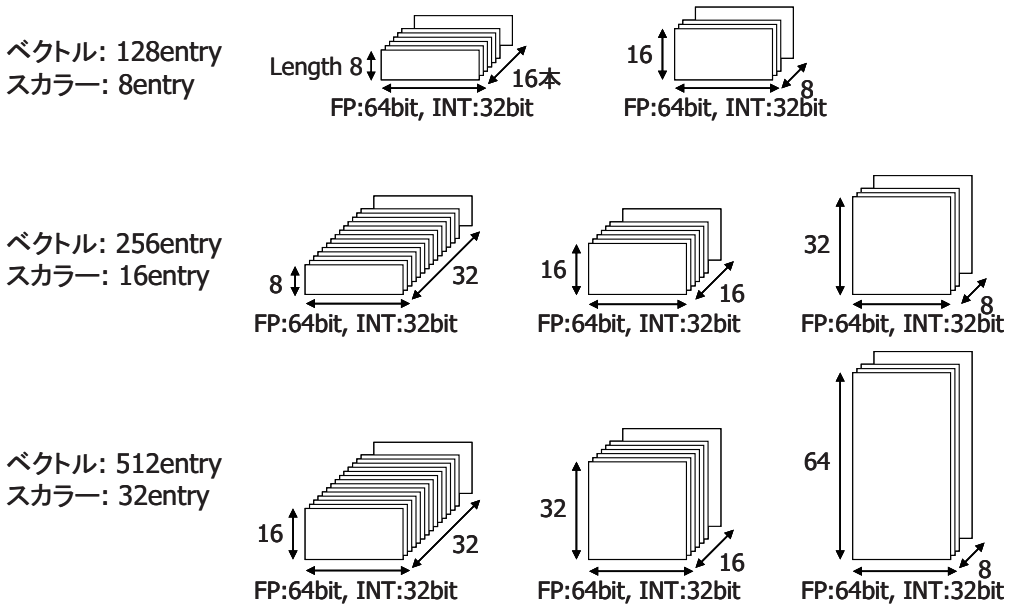


図 3-1-8 2 次元ベクトルレジスタの構成

⑤ Responsive Link (RL)

ロボットを分散方式で制御するためには、ノード間の実時間通信機構が必要である。先に述べたように、ロボットの全身で、高精度に同期した実時間制御を行うためには、Ethernet のようなパケットの衝突や再送が起こるような通信方式は不適当であり、実時間に向いたネットワークが必要となる。RMTP は、実時間通信ネットワークとして、次の特徴を有する Responsive Link(RL と略)を備えている(図 3-1-9 参照)。

- パケットに 256 レベルの優先度を付加し、優先度による追い越し機能つきネットワークスイッチを用いて、ノード通過時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。通信におけるプリエンプション機構を世界で初めて実現し、通信をリアルタイムスケジューリング可能に
- 通信路は高スループット通信向きのデータリンクと、低遅延通信向きのイベントリンク(各々は双方向)から構成される。データリンクは 64 バイト固定長パケット、イベントリンクは 16 バイト固定長パケットとする
- リンクの種類(データリンクとイベントリンク)およびパケットの優先度に応じて異なる経路を設定可能で、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、分散

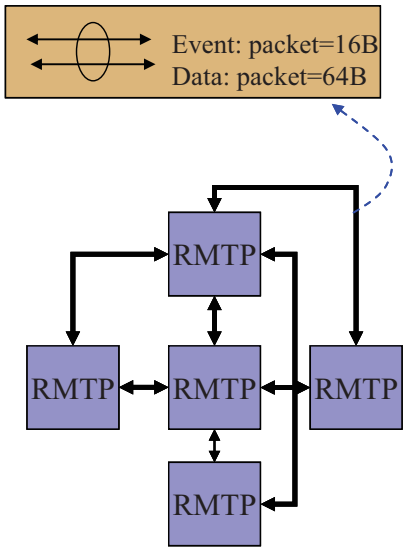


図 3-1-9 Responsive Link

管理型でパケットの加減速を制御可能

- パケットごとにハードウェアによる通信エラー訂正を 1 ホップごとに行い、再送を不要とする前方エラー訂正(FEC)を実装
- 通信速度を 12.5 ～ 800 [Mbps/link]で動的に変更可能

M-RMTP の RL の各ノードには 4 つのポートが実装され、ルーティングは各ノードがハードウェアで行う。32 ビットのヘッダを読み込んだだけでルーティング先が決まるカットスルー方式なので、ノード 1 段の遅延は 100Mbps 時に 320ns、ロボット全体では 3 マイクロ秒程度で通信可能である。低遅延かつ高バンド幅でロボットの体内ネットワークを構築することが可能である。

従来のリアルタイム通信は、USB や IEEE1394 に代表されるアイソクロナス転送をベースとした周期的なソフトリアルタイム通信しかなかった。一方で、Responsive Link はノード毎に高優先度パケットが低優先度パケットを追い越すことを実現したので、通信におけるプリエンプションを実現して、ハードリアルタイム通信を実現可能とした。そのため、トップダウンに通信をリアルタイムスケジューリング可能にした。

RL は、1998 年に提案し、1999 年に SPARC コアや各種 IO と共に Responsive Processor として SoC 実装を行った。その後、RMTP の製作に当たって、プロトコルの改良及び通信速度を引き上げた実装を行っている[山﨑 2004]。国内では、情報処理学会試行標準 ITSJ-TS 0006:2003 として標準化が完了し、国際的には、ISO/IEC JTC1 SC25 WG4 において国際標準化作業を進め、ISO 24740 として標準化を完了した[RL 標準-URL]。

⑥ 省電力制御

プロセッサの消費電力は、ほぼクロック速度に比例し、電源電圧の 2 乗に比例して増大する。また、クロック速度を上げるためには、電源電圧を上げる必要がある。一方、I/O やメモリをドライブするためには、標準的な 2.5 - 3.3V などの駆動回路が必要となる。消費電力は、プロセスルールにも大きな影響を受ける。従来、スケール縮小によって比例的に省電力化が可能であったが、ゲートの大きさに比してソース・ドレイン間の距離が極端に短くなってきたため、リーク電流が増大し、90nm 以下の縮小では省電力効果が薄れてきている。

M-RMTP は、コアの最大動作周波数を 100-133MHz 程度に抑え、同時にコアの電源電圧を 1V 以下にすること、及び、High Vt プロセスを用いた最新の低消費電力スタンダードセルを通常のスタンダードセルと組み合わせて使用することで電力消費を削減している。クロックを遅くすることによる性能低下は、並列度を大幅に上げることで補う。

M-RMTP には、大規模なベクトルユニットと多数の I/O を集積しているが、すべてのアプリケーションでそれらを使用するわけではない。DVFS (Dynamic Voltage and Frequency Scaling)を強力にサポートするために、M-RMTP のコア電圧は 0.8V(Max33MHz) ～1.2V (Max133MHz)の範囲で可変である。さらに、機能ブロック毎にクロックラインを独立して制御することが可能であり、I/O 毎にクロックを停止したり、周波数を滑らかに変化させることを可能にしている。例えば、ベクトル計算を行わない場合は、ベクトルユニットのクロックを停止させる。このように、コア電圧とクロックをきめ細かく制御することで省電力化を図る。

一層の省電力が必要となるのは、関節角度のバックアップである。ヒューマノイドロボットは、関節角の制御のため、モータ軸に結合されたロータリエンコーダからの毎秒数千のパルスをカウントしている。情報処理系が停止すると、このパルスカウントが停止し、関節角度が不定となる。すると、システムを再立ち上げする際には、マニュアルでの関節角度のリセットが必要となる。ヒューマノイドのように多数の関節がある系では、この作業は大きな手間となるので、バッテリによる関節角度のバックアップを行う。その際の消費電力を極小にするため、SoC 内蔵のエンコーダ部を独立して電源供給可能なように設計し、その部分をバッテリバックアップできるようにした。ヒューマノイドを使用していない際には、エンコーダ部の値はバッテリでバックアップし、プロセッサ部はクロック及び電源の供給を停止可能にした。

⑦ ロボットデバイス制御

M-RMTP には、モータの回転角をセンスするためのエンコーダカウンタ、モータへの電流指令を出すための PWM 出力、電流値をセンスするための PWM 入力、カメラから画像を入力するための IEEE1394 ポート等を集積している(図 3-1-1 参照)。デバイス制御では、AD/DA 変換器を用いてアナログ信号を入出力するケースが多いが、M-RMTP のデバイスインタフェースは、ノイズの影響を除くためにすべてデジタル I/O としている。これには高密度のデジタル LSI にアナログ回路を混載することの諸問題を回避する目的もある。AD/DA 変換器は、SPI に接続することも可能である。これらのデバイス制御の他に、ホストとなる PC と接続するための PCI や Ethernet、システム機能を拡張するための外部バス、コンソールのためのシリアル(RS-232C)等を集積している。

⑧ パッケージングと SiP 実装

μRMTP 及び M-RMTP の設計は、論理設計からレイアウトまでのすべてを慶應義塾大学山﨑研究室で実施した。製作は世界最大のファウンダリメーカである TSMC に委託し、0.13μm、CMOS8 層の銅配線で、9.98 ㎜角のダイを作成した(図 3-1-10 参照)。M-RMTP では、非常に多数の IO ピンを入出力し、かつ SiP (Sysem-in-Package)実装を容易にするため、チップ表面に極小のハンダボールを直接実装するフリップチップという最先端技術を用いて IO ピンを設計し、1040 ピンもの多数のピンをチップ上に実装している。

M-RMTP をロボットの体内に組み込むために、超小型のシステムボードである M-RMTP SiP を開発した(図 3-1-11 参照)。M-RMTP SiP は、M-RMTP ベアダイ、DDR SDRAM x 4、Flash memory x 2、Ethernet Phy、レギュレータ、各種受動素子を、わずか 33mm 角に集積している。従来の SoC は、プロセッサコアに mpeg 処理機能が集積されているものがほとんどであった。それに対して、M-RMTP は、まさにチップ上にシステムを構築しており、真の意味での SoC を実現していると考えられる。

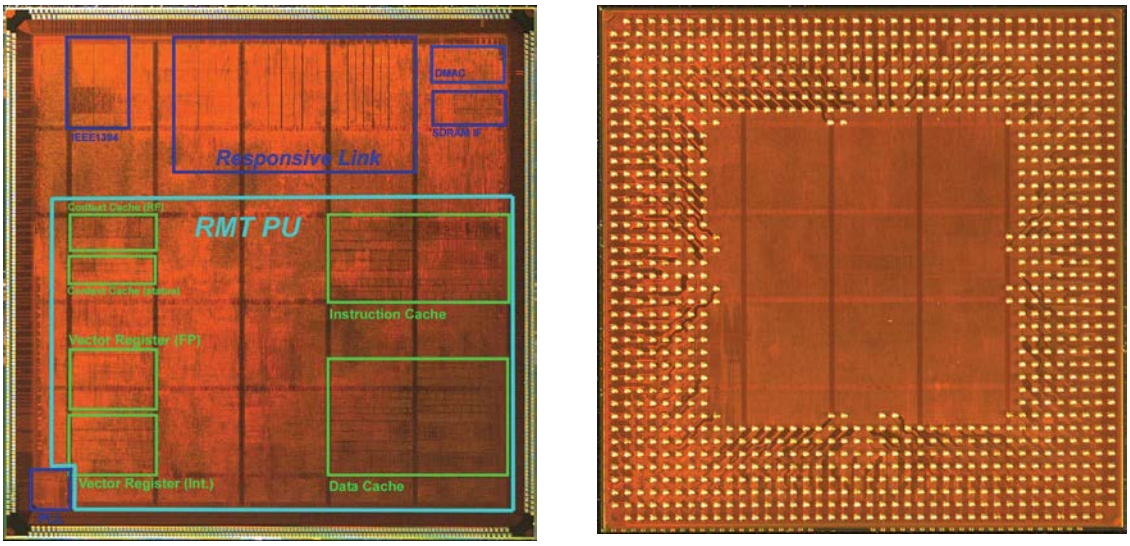


図 3-1-10　μRMTP のダイ写真とレイアウト（左）及び M-RMTP のダイ写真とフリップチップ（右）

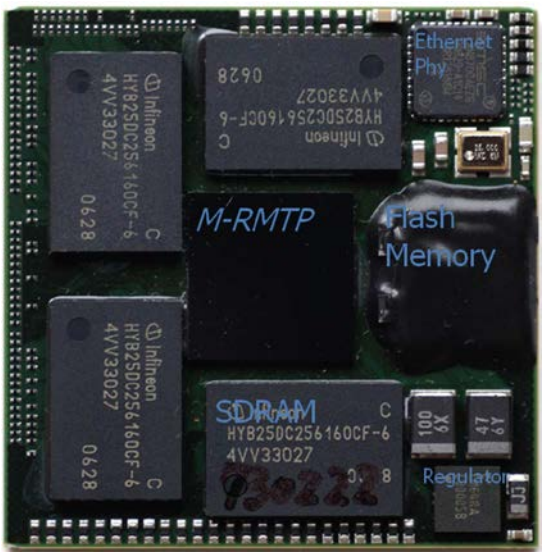


図 3-1-11 M-RMTP SiP

⑨ 性能評価

RMT PU の 100MHz でのピーク演算性能は、スカラ整数 400 MIPS、スカラ FP 200MFLOPS、ベクタ整数(32bit:3.2GIPS, 16bit:6.4GIPS, 8bit:12.8GIPS)、ベクタ FP(64bit:1.6GFLOPS、32bit:3.2GFLOPS)である。

階段状に異なる優先度を持つスレッド(th0 > th1 > … > th7)で離散コサイン変換を同時実行した際の IPC (Instruction Per Clock cycle)の変化の様子を図 3-1-12 のグラフに示す。まず、最高優先度スレッドの th0 が最も優先的に演算リソースを使用して、時刻 0 から IPC 0.6 程度で実行されている。次に優先度が高い th1 が余ったリソースを使用して、時刻 0 から IPC 0.58 程度で実行されている。同様に、th2 は IPC 0.4 程度で実行され、th3 は 0.07 程度で実行されている。優先度の高い th0, th1 が順に演算を終了すると、th2, th3 の優先度が高くなり、その結果そのスレッドの IPC が高くなり、

さらには優先度の低かった th4 や th5 の優先度が相対的に高くなり IPC も高くなって実行されている様子が分かる。以下、同様に、相対的に高い優先度のスレッドが終了するごとに、低い優先度のスレッドの優先度が相対的に高くなって IPC が高くなって実行されていることが分かる。このグラフより、まさに図 3-1-2 の RMT 実行(ハードウェアによる実時間実行)を実現できていることが分かる。

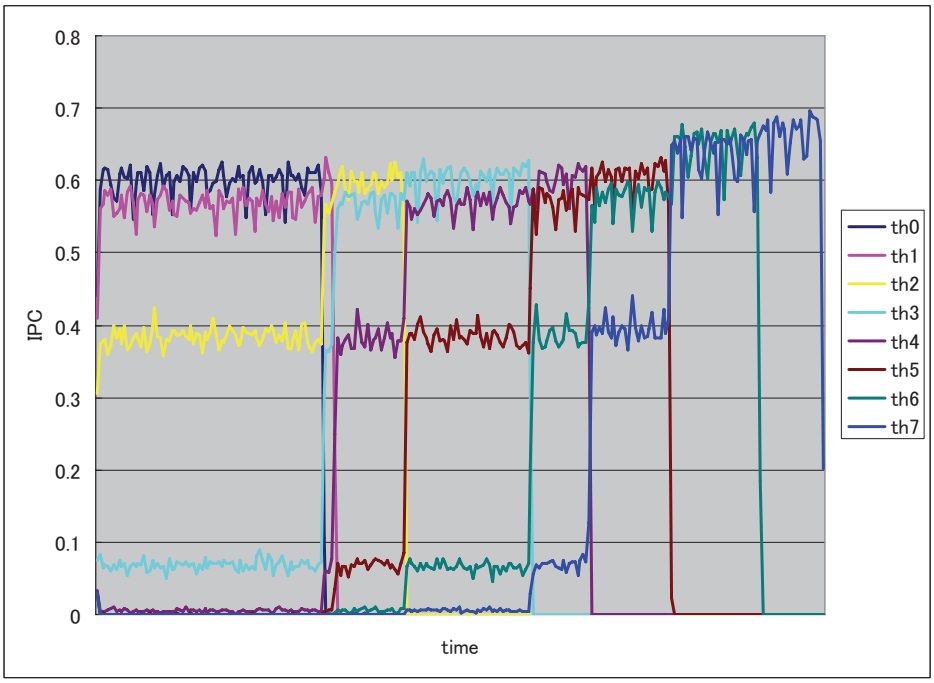


図 3-1-12 RMT 実行の様子

⑩ 参考文献

[山﨑 2003] 山﨑信行, 堀俊夫, “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, Vol.44, No.1, pp.6-13, 2003.

[山﨑 2004] 山﨑信行、「分散制御用リアルタイム通信 Responsive Link の設計及び実装」、情報処理学会論文誌コンピューティングシステム(ACS), Vol. 41, No. 5 (to appear), 2004.

[Yamasaki2004] Nobuyuki Yamasaki, “Design Concept of Responsive Multithreaded Processor for Distributed Real-Time Control, Journal of Robotics and Mechatronics, Vol. 16, No. 2, pp. 194-199, April, 2004.

[RL 標準-URL] http://www.itscj.ipsj.or.jp/ipsj-ts/02-06/toc.htm

(2) 今後期待される効果

本研究において設計・実装した RMTP は、IT 分野におけるパーソナルコンピュータのように、このデバイスさえあれば、ほぼすべてのロボットを低コストで制御可能になる可能性があるパーツとなりうる。近年の複雑なタスクを実行するロボットは、ひとつのプロセッサだけでは制御しきれないので、分散実時間制御を容易に実現可能になるデバイスに対する要求は大きい。したがって、RMTP は、分散実時間システムを容易かつ効率的に実現するための、実時間通信及び実時間処理を行なう

基本機能を有した共通プラットホームとなりうると考えられる。ユーザにとっては、ブロックを組み立てるように組み合わせて実時間システムを構築可能となれれば望ましい。

システム設計者は本 SoC（もしくは本 SiP）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ように可能となる。また、それら I/O を接続し固有の機能を有したRMTP 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、所望のロボット（分散実時間システム）を構築可能になると考えられる。これによって、様々なロボットの開発を低コストで促進できると考えられる。

また、RMTP は、ロボット以外の様々な分散実時間処理用途にも応用可能である。特にリアルタイム通信リンクの Responsive Link は、ISO/IEC において国際標準化を達成したので、様々な種類の分散実時間システムの構築に使用されると期待している。

3.2. 実時間オペレーティングシステムと開発環境の開発（東京大学　基盤ソフトウェアグループ）

(1) 研究実施内容及び成果

ロボット制御に限らず、実時間処理の目的は、決められた時刻（デッドライン）までに与えられた処理が終了することを保証することである。ロボットでは、その処理に必要なデータにはなるべく新しい情報を使う必要がある。終了時刻が保証されれば、新しいデータの取得をデッドライン寸前まで遅らせることができ、より実世界の変化に即応した、精度の高い制御を実現することができる。ロボットのソフトウェアは、大きく認識系、計画系、実行系に分けられる。認識系は、ビジョンや聴覚の処理を指し、数十ミリ秒以上の時間を要する。計画系は、目的地までの経路や足跡軌道を作り出すが、経路の長さや環境の複雑さに依存した処理時間を要し、実時間性は低い。これに対して、実行系は歩行中の各関節の軌道を制御するために、モータサーボを1ミリ秒以下の正確な周期で実行しなければならない。このように、ロボットのソフトウェアは、実時間制約の強いプロセスと、弱いプロセス、非実時間プロセスで階層的に構成される。これはまた、実時間プロセスが、非実時間プロセスの擾乱を受けやすいことも意味する。また、たとえ認識系が高速に異常を検知しても、行動に反映するためにはモータサーボを駆動する必要がある。このため、モータサーボ以上の即応性をシステムに課することは意味がない。また、モータサーボは、停止することなく継続的、周期的に実行されるので、全系のクロックとしての意味を持つ。

以上の考察から、本プロジェクトでは、モータサーボクラスの実時間プログラムには、WCET（Worst Case Execution Time:最悪実行時間）を予測して厳密なデッドラインを静的に保証するためのツール RETAS を研究開発した。また厳密な時間制約を保証する実時間 Linux である SHI-Linux の研究開発をおこなった。さらに、バッテリー寿命を長くするために、ソフトウェアによる電力制御機構の研究開発を行った。これらソフトウェア群は、RMTP に特化することなく Intel 社 x86 等の一般のプロセッサにも適用できる汎用性のある手法を提案した。平成 20 年 9 月時点において、RMTP 上で Linux を稼働させる環境を得ることができなかったため、本報告書において開発されたソフトウェアの評価は主に x86 上で行われた。

① WCET 予測ツール RETAS

一般に、最悪実行時間は、プログラムの実行時間を複数回計測して求められている。このような手法では、求められた最悪実行時間が実際に最悪値であるのか理論的に保証できないという問題がある。本プロジェクトでは、プログラムのソースコードを解析し最悪実行時間を予測するツールである RETAS を開発した。RETAS はソースコードを元に、全ての実行可能パスを考慮した上で最長となる実行時間を計算する。

RETAS は図 3-2-1 に示すとおり大きく 3 つの部分(実行フローの解析、メモリアクセス遅延の解析、命令実行時間の解析)から構成される。実行フローの解析では、改造した GCC(GNU Compiler Collection)でソースコードをコンパイルし、GCC の内部表現である RTL(Register Transfer Language)を解析し制御フローグラフを生成する。プログラムの実行時間は、メモリアクセスにかかる時間と、命令自体の実行時間に分けて計算する。メモリアクセス遅延の解析では、CPU アーキテクチャのキャッシュ機能をシミュレーションするツールを開発した。

シミュレーター上で RTL を実行してメモリアクセスパターンを求め、キャッシュやメインメモリの参照回数からメモリアクセスの時間を求める。命令実行時間の解析では、アセンブリコードからメモリアクセス命令を取り除き、メモリアクセスが生じないコードに変換する。さらに、基本ブロックの開始点と終了点に実行時間計測のための命令を挿入し、コード片を実機上で動かし基本ブロックの実行時間を計測する。メモリアクセス遅延の解析結果と命令実行時間の計測結果から、基本ブロックの実行時間が予測される。全ての基本ブロックの予想実行時間と制御フローから最悪実行パスを求めることにより最悪実行時間を予測する。

コンパイル時に静的に決定できない変数の値による不確定性については、プログラマによる pragma 文の挿入が必要である。例えば、関数引数がループ回数の上限値を決めているような場合、変数の取り得る値を pragma 文で挿入する。pragma 文は、通常のコンパイラではコメント行として扱われるため、プログラムのポータビリティは確保される。

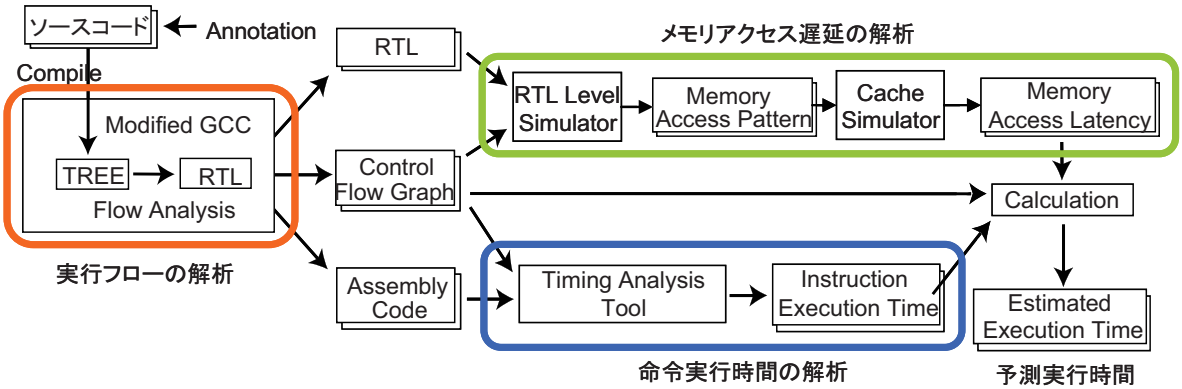


図 3-2-1 WCET 解析フロー

表 3-2-1 WCET 予測

プロセッサ	ベンチマーク	実測値（クロック）	予測値（クロック）	誤差（%）
Pentium-M	行列乗算	24741	33057	133
	挿入ソート	14442	23658	164
	フィボナッチ	3334	3911	117
XScale	行列乗算	48184	51277	106
	挿入ソート	19873	19967	100
	フィボナッチ	5110	5124	100

表 3-2-2 ロボットタスクの WCET 予測

プログラム	実測値	予測値	誤差(%)
サーボ	23373	53473	228
ステレオ	131.51M	165.71M	126
A*	16.09M	297.83M	1861

表 3-2-1 は、Pentium-M と XScale に本手法を適用した結果である。Pentium での予測値に大幅な誤差があるのは、仕様が明らかにされていない Pentium のもつ投機実行、パイプライン、キャッシュの振る舞いなどの挙動によるものと考えられる。XScale は Pentium に比べると単純なアーキテクチャであるため本手法で十分な精度が得られる。

表 3-2-2 は、実際にロボットで使用されているプログラムの Pentium-M 上での WCET 解析結果である。サーボタスクは、センサの入力によってサーボの動作を設定するタスクである。予測値と実測値に大きな差があるのは、タスク中の浮動小数点命令の実行時間が実測値と予測値とで大きく異なるためである。現在の RETAS では浮動小数点命令の実行時間はプロセッサのデータシートの値を参照して予測している。しかしデータシートの値と実際の実行時間には誤差があり、これが予測値と実測値との差として現れている。現在、基本ブロックの命令実行時間計測時に浮動小数点命令も計測できるように RETAS を改良している。

A*は経路の探索をするタスクである。A*では地図データの内容によってメモリアクセスパターンが変化するため、ソースコードを解析してもメモリアクセスパターンが求められない。タスク中の大半のメモリアクセスをキャッシュミスとして扱うため誤差が非常に大きくなっている。メモリアクセスパターンが予測できないタスクに対し pragma 文の挿入により予測精度を上げる改良を行っている。

② WCET 関連研究

WCET は様々な要素技術の組み合わせで求められる。これら要素技術はフロー解析および実行時間解析の 2 つに分けられる。

フロー解析では，プログラムのコードを基本ブロックを最小とするグラフ構造で表す．この構造は Control Flow Graph(CFG)と呼ばれる．既存研究はアセンブリコードを解析して CFG を生成するものが多い．最低限の CFG はアーキテクチャごとにアセンブリコードの解析器を用意するだけで求めることができる．しかし，WCET の予測には各基本ブロックが最大で何回実行されるのかといった追加の情報が必要である。このような情報を自動的に求めるには，データフローの解析を行なわなければならない。

データフロー解析については C 言語において自動的にループ回数を計算する手法 [healy, 1998]や C 言語を拡張しソースコード中に解析器へヒントを与える手法¥cite{kirner:01}などがある。

一般に、アセンブリ言語をフロー解析する場合は、言語自体がアーキテクチャ依存のため解析器の移植性は乏しい。また、変数などの意味情報が失われているため、データフロー解析が難しいという問題がある。C 言語のような高級言語を解析する場合は、コンパイラの最適化が考慮されないため実際の実行フローとは異なるフロー情報を求めることになる。

実行時間解析では、コードの一部分の実行時間を求める。つまり、あるプロセッサ上でどのように命令が実行されるのかを解析することになる。このため、プロセッサの様々な機構のモデル化について研究が行なわれている。例えば、命令キャッシュ[lim95, ottosson97]やデータキャッシュ[white97, lundqvist99]、分岐予測機構[colin00]、パイプライン[schneider99, stappert00]などのモデルである。しかし、プロセッサの内部動作に関する詳細な情報が公開されていないため、正確なモデルを構築することはできない。また、個々のモデルはアーキテクチャに強く依存しているため移植性に乏しい。

モデルを用いずに実行時間を解析する手法として、実際にコードを実機で実行、実行時間を計測する手法が挙げられる。モデルを用いて考えていたプロセッサの様々な機構は、実機での実行によりすべて計測結果に含まれる。ただし、単純に計測するだけではブラックボックステストになるため最悪値を保障することができない。

Pettersらはコンパイラの生成するCFGを使い実行不可能な経路を削減して実行回数を減らし、実行時間を測定する手法を提案している[petters99]。同様に、Lindgrenらは、様々な実行経路で実行時間を測定して連立方程式を立て、これを解くことで基本ブロックの実行時間を求めている[lindgren00]。Wenzelらはモデル検査手法を利用してテストデータを自動生成し、様々な経路の実行時間計測を可能としている[wenzel05]。ただし、これら手法は動的なメモリアクセスを含むプログラムの場合にメモリアクセス違反が問題となり、適用は静的なプログラムに限定される。

これら既存研究と異なり、本プロジェクトでは、内部仕様が明らかにされないアーキテクチャにも対応し、かつ、様々なアーキテクチャに適応できるように以下の手法を提案した。コンパイラ内部で利用される中間表現を利用することにより、アーキテクチャへの依存性をなくしコンパイラの最適化を考慮した実行可能経路を求める。また、中間表現でデータフローを解析し、基本ブロックの実行回数を計算する。実行時間解析は、実行時間の測定により求める。動的メモリアクセスを含むプログラムへも対応するため，実行時間をメモリアクセス遅延と命令実行時間に分け、シミュレータを用いてメモリアクセスの遅延を計算する。

③ 実時間オペレーティングシステム Shi-Linux

現状のHRP-2は、Linuxの実時間版であるART-LinuxにCORBAを搭載し、ロボット制御のためのプロセスをCORBAオブジェクトとして実装している。ソフトウェアの継承の観点から、RMTPのオペレーティングシステムには、Linux2.6をベースに実時間スケジューリングの拡張を加えたShi-Linuxを開発している。ロボット全体で100マイクロ秒以下の同期・応答性を目標とする。実時間スケジューリング法としては、EDF (Earliest Deadline First)、RM (Rate Monotonic)法などがよく知られている。実時間制約に応じて実行の優先度を与えるが、デバイスからの割り込みハンドラは、プロセスの優先度を無視して実行されるため、擾乱が生じる。Shi-Linuxは、この問題を解決する方法として、デバイスドライバを周期タスクとして実現する手法を採用したShi-Linux Version 1と、デバイスドライバの実行時期を実時間タスクの実行開始時刻との関係で動的に決定する手法を採用したShi-Linux Version 2の2つの版を開発した。

Linux Version 1では、常時デバイスからの割り込みを禁止し、全てのデバイスドライバは周期タスクとして実行される。タイマ割り込みによって起動されるスケジューラは、これら周期タスクのデッドラインおよびアプリケーションプログラムのデッドラインから動的に優先度を決めて、実行権を与える。Shi-Linux Version 1プロトタイプをX86プロセッサ向きに実現し、Pentium4　2.6Hzにおいて、2μsでのタスクスイッチ時間と12μsの横取り禁止期間（disk I/O処理がない場合)に抑制することに成功している。

Shi-Linux Version 1は、デバイスドライバを周期タスクとして実現するため、デバイスの応答性能はデバイスドライバタスクの周期に依存する。周期を短くすると応答性能が上がるが、無駄なデバイスポーリング時間が生じ、CPU利用率を低下させる。Shi-Linux Version 2では、デバイスからの割り込みが発生した場合に、実時間タスクとデバイスとの関係を動的に調べてデバイスドライバを起動

するかどうかを判断する。このために、デバイス割り込みを即時割り込み、実時間割り込み、遅延割り込みの 3 種類に分ける。

即時割り込みは、タイマ割り込みのような実時間処理に不可欠なデバイスからの割り込みである。即時割り込みが発生すると対応するデバイスドライバが直ちに起動される。

実時間タスクがデバイスからの応答を要求している場合に、そのデバイスからの割り込みは実時間割り込みとなる。実時間割り込みの実行優先度はそのデバイスの応答を待っている実時間タスクの実行優先度と等しい。実時間割り込みが発生すると、その時実時間タスクが動作していれば、その実時間タスクと実時間割り込みの優先度が比較される。実時間割り込みの実行優先度が高い場合には、実時間割り込みに対応するデバイスドライバが起動される。

即時割り込みでも実時間割り込みにも属さないデバイスからの割り込みは遅延を割り込みに分類される。全ての実行可能な実時間タスクの実行終了後に、本割り込みに対応するデバイスドライバが実行される。Linux Version 2 では、さらに、共有資源の排他利用による優先度逆転(priority inversion)問題を解決している。Linux2.6 では、共有資源の排他利用の 99.5 %は、20 μ s 以下の期間に収まっている。残りの 0.5%は、100 μ s を超えることもある。このため、非実時間タスクや優先度の低い実時間タスクが共有資源を長い期間排他利用すると、優先度の高い実時間タスクが同じ共有資源を利用しようとしても実行できないことによる優先度逆転が生じる。優先度逆転期間は共有資源の排他利用期間となるため、100 μ s を超えることになり、ロボットアプリケーションに要求される応答性能を満たせなくなる。

Linux の単一 CPU 実装では、あるタスクがカーネルの共有資源を排他利用している最中は、割り込みが禁止され、その排他利用が終了するまで、他のタスクに実行権が移らない。これにより、共有資源のクリティカルセクションが実現されている。ここで 2 つの優先度逆転問題が生じる。

1. 共有資源が排他制御されている間、他のより高い優先度のタスクが排他制御されている共有資源を使用しないのに、そのタスクの実行が延期されてしまう。

2. 共有資源が排他制御されている間、他のより高い優先度のタスクがその排他制御されている共有資源を使用したくても、その資源が解放されるまで実行が延期されてしまう。

上記1の優先度逆転現象を解決するために、Shi-Linux では以下の手法を導入している。

i) 共有資源を排他利用中でも割り込みを可能とし、即時割り込み処理だけは処理する。これによりタイマー管理が遅延なく処理され、スケジューラを呼び出すことが可能となる。

ii) 共有資源を排他利用中にタイマー割り込みによりスケジューラが呼び出され、現在利用されている共有資源を使用しないタスクで、かつ、そのタスクが現在実行中のタスクよりも優先度が高い場合、そのタスクに実行権を渡す。

上記 2 の優先度逆転問題を解決するために、Shi-Linux では、各タスクが共有資源の排他利用に使用されるロック変数の最大保持期間とそのロック変数を使用するタスクを記録する機構を導入し、以下の手法で優先度逆転問題を解決している。

i) ある非実時間タスクがロック変数を使って排他制御するとき、そのタスクが排他利用する期間中に、同じ資源を使用する実時間タスクが実行される可能性を動的に検査する。

ii) もし、そのような実時間タスクが実行される場合は、非実時間タスクの排他利用は待たされる。

iii) 共有資源の排他利用期間が短い場合には、本機構が適用されることによるオーバヘッドが問題となる。Shi-Linux では、本機構を適用する共有資源を最大排他利用期間に応じて選択できるようにしている。

3.0GHz Pentium 4 上で Linux 2.6.20 カーネルと Linux 2.6.20 を基にした Shi-Linux Version 2 を用いて、Shi-Linux の評価を行った。最大排他利用期間が 30 μs 以上となるロック変数の排他利用に本機構を適用した。指定された実行開始時刻と実際の開始時刻との遅延を計測した結果を表 3-2-3 に示す。Shi-Linux で以前最大遅延時間が残っているのは、Linux に割り込み禁止期間が存在するためである。

表 3-2-3 実行開始時刻遅延

	オリジナル Linux	Shi-Linux
最小遅延時間	0.85 μs	0.76 μs
最大遅延時間	268.85 μs	51.44 μs
平均	12.23 μs	3.25 μs

④ Shi-Linux 関連研究

Delayed Locking[Lee05]では、Shi-Linux 同様資源の排他利用を開始する前に、他のタスクが同じ資源を排他利用するかどうか調べ、他のタスクのほうが優先度が高い場合にはその資源の排他制御は待たされる。

Linux を元にした実時間システムにおいて、これまで割り込み問題を対処する手法はいくつか提案されている。一つは割込みスレッド方式である。TimeSys[TimeSys]と QNX[Hildebrand92]では、割り込み処理の主な部分はスレッドで実行される。割り込みが発生すると、対応した割り込み処理スレッドが起床され、割り込み禁止モードで割り込みスレッドが実行される。同手法では、割り込みハンドラは二つの部分に分かれる。一つは割り込み受付部分であり、もう一つは割り込みを処理する部分である。割り込み受付部分の処理は小さく、実時間タスクの実行への影響を最大限に抑えられる一方、割り込み処理スレッドを起床するため、コンテキストスイッチとタスクディスパッチャの回数が増大する。TimeSys や QNX では、スケジューラが割り込みスレッドに高い優先度を与えることがなく、割り込みスレッドが実質的にすべての実時間タスクの後に実行される。実時間システムでは、実時間タスクの優先度が必ず割り込み処理より高いわけではないため、割り込みスレッド方式はシステム優先度管理の問題が依然残っている。

割り込みスレッド方式のもう一つの手法は、割り込み処理スレッド内でデバイスの状態をポーリングする方式である[Kopetz89, Ishiwata99]。同手法ではタイマー割り込み以外、外部からの割り込みはすべて禁止され、割り込み処理スレッドは周期的にデバイスの状態をポーリングする。同方法で

は、実時間タスクが先行的に実行することができ、静的な優先度を指定した場合は、割り込みスレッドと実時間タスクの間に優先順序を指定することができる。しかし、割り込みスレッドによって周期的にデバイス状態をポーリングすると、CPU 使用率が落ちる。さらに、スケジューラが頻繁に割り込みスレッドを起床するため、タスクディスパッチャとコンテキストスイッチの回数が増大する。

Luis と Pedro[Luis06]は、予測可能な割り込み管理機構を提案している。本機構では、すべての周辺デバイスからの割り込みは一つの小さな割り込み処理プログラムで受理される。割り込み処理プログラムは現在処理中のタスクの優先度と受理された割り込みの優先度と比べ、受理された割り込みの優先度が高い場合は、シグナルによって、対応した割り込みスレッドを起床させる。本機構はシステム全体で優先度管理が可能となり、割り込み処理は実時間タスクの後に実行されることはなくなる。しかし、この割り込み管理機構では、実時間タスクと割り込みスレッドとの優先度指定は静的にしか指定できなく、多くのデバイス I/O 処理を待つ実時間タスクにとっては、すぐに処理して欲しい割り込みスレッドの実行が遅延されることになる。また、本方式でも割り込みスレッドを起床するためのタスクディスパッチとシステムコンテキストスイッチ回数が増大する。このように、割り込み問題を解決する割り込みスレッド方式では、タスクディスパッチャとコンテキストスイッチの回数が増大し、システム全体の負荷が増大する。このため、軽量的かつ動的な優先度つきの割り込み処理手法が必要になる。Shi-Linux Version 2 では、割り込みスレッドを用いずに、動的優先度に基づいて割込み処理を行うことが可能である。

⑤ プロセス単位スケジューラ

Pentium M プロセッサ等で実現されている DVFS 機能は、ソフトウェアから CPU の周波数と電圧を動的に変更する機能である。CPU 負荷に応じて、周波数と電圧を制御することにより省電力化が可能となる。従来実現されているソフトウェア技術では、デーモンプロセスにより大域的な CPU 負荷統計情報を用いて DVFS 機能により周波数と電圧を制御していた。

我々は、実行時に動的にプロセス単位で電力を制御するスケジューラのプロトタイプである

用途	Webサーバ	クライアント
機器名	Thinkpad X31	MY30V/C-F
CPU	PentiumM 1.6GHz	Pentium4 3.0GHz
メモリ	1GB	512MB
OS	Linux Kernel 2.6.16.20	
コンパイラ	gcc 4.0.4	
備考	バッテリーレス	

計算機の種類	タスクの種類	タスク
評価対象期	応答性能を必要とするタスク	WEBサーバ
	バッチタスク	prelink (Linuxで定期的に稼動しているデーモ
クライアント	WEB Client	Apache bench

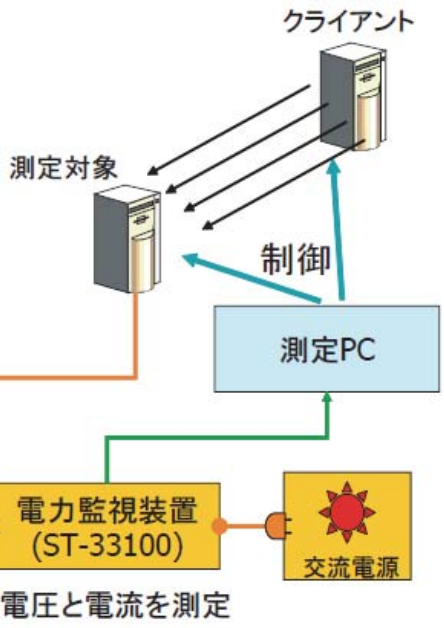


図 3-2-2 電力制御アルゴリズム計測環境

POPM (Process-Oriented Power Management)機構を Linux 上に実装した。Linux において、プロセスの優先度は以下の式で動的に決定される。

動的優先度 = 静的優先度(Nice 値) − (BONUS + 5)

Nice 値は、プロセス生成時に決められる値で、プログラマに開放されたプロセス優先度であるが、Nice 値は割り当てられるタイムスライスの長さも変えてしまうので実時間プロセスの操作には適当ではない。BONUS は、Linux のスケジューラによって決められる値で、0〜10 の値を取り、高い応答性を必要とするプロセスの優先度を上げる。本プロトタイプでは、プロセスのコンテキストスイッチを行なう時に、BONUS を用いて電力制御を行なう。すなわち、0 から 10 の値に対して、周波数 600 MHz から 1600 MHz の値にマッピングする。プロセスコンテキストスイッチ時に、この値で CPU 周波数を設定する。図 3-2-2 に示す計測環境とベンチマークプログラムを用いて本プロトタイプシステムを評価した。本ベンチマークでは、WEB サーバの平均応答性能を維持し、かつ、消費電力量がどのくらい減るかが評価軸となる。実現した方式に対する比較として、表 3-2-4 に示すような、既存 Linux で実現されている ondemand, powersave などの電力制御アルゴリズムならびに手動で WEB サーバには最大電力、prelink には最小電力を割り当てた時の性能として計測した。なお、既存 Linux の手法は、システム全体に適用され、プロセス単位での制御ではない。

図 3-2-3 は、平均応答性能を消費電力で除した値、すなわち消費電力当たりの応答性能を示しており、高いほど高性能である。提案手法 pccs は、消費電力量では、ideal、powersave に次いで、web 性能では ideal に次いで良い値が出ており、総合的な性能では、ideal(理想値)に次いで、既存の Linux の電力制御を凌ぐ性能が得られている。

表 3-2-4 Linux 電源制御アルゴリズム

performance	周波数最大に固定
powersave	周波数最小に固定
ondemand	定期的に CPU 負荷を調べて設定
conservative	Ondemand よりも長い周期で定期的に設定
ideal	WEB サーバに最大周波数、prelink に最小周波数を設定

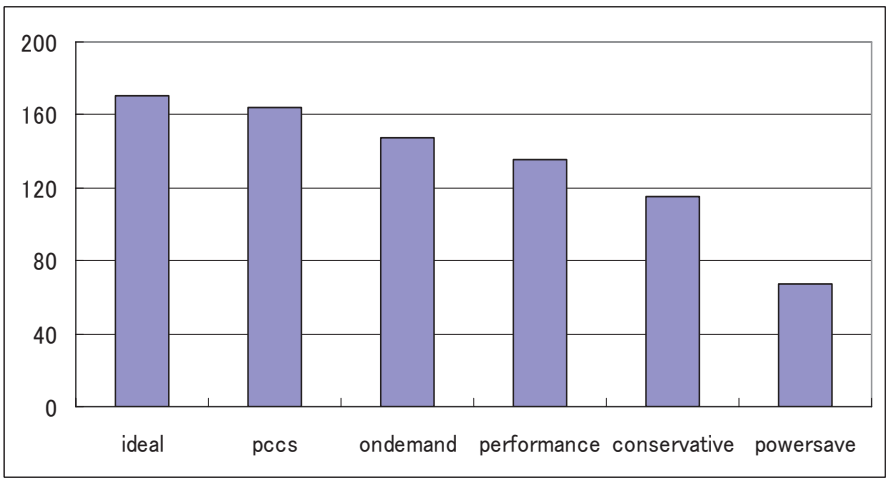


図 3-2-3 各種の電力制御モードでの消費電力当たりの web サーバ応答性能

⑥ プロセス単位スケジューラに関する関連研究

DVFS 技術は、高性能計算から組込み機器まで様々な領域で研究されている。高性能計算分野では、通信やメモリ性能がボトルネックになって CPU の演算性能が発揮できない場合に、DVFS 技術により性能バランスをとると共に省電力化する研究がおこなわれている[hsu05, nandini05, ge05]。組込み実時間処理分野では、実時間スケジューリングにおいて、デッドラインに余裕があるときに、DVFS 機能を用いて省電力化する研究がおこなわれている[pillai01, vishnu02, zhu05]。また、ソフト実時間システムにおいて統計的に CPU 利用率を求めて周波数を決めるアプローチもある[gruian01, yuan06]。

Yuan らは、EScheduler と呼ばれるソフト実時間システム向け CPU スケジューラを提案している[yuan06]。EScheduler と異なり、我々の提案機構 POPM は、I/O 利用率を考慮にいれ、プロセスの性質をシステム側で判断して自動的に電源管理する点が異なる。

⑦「実時間オペレーティングシステムと開発環境の開発」に関連する参考文献

[wenzel05] I. Wenzel, R. Kirner, B. Rieder, P. Puschner, “Measurement-Based Worst-Case Execution Time Analysis,” In Proc. of 3rd IEEE Workshop on Future Embedded and Ubiquitious Systems, pp. 7 – 10, 2005.

[petters99] S. Petters and G. Farber, “Making Worst Case Execution Time Analysis for Hard Real-Time Tasks on State of the Art Processors Feasible,” In Proc. of 6th International Conference on Real-Time Computing Systems and Applications, 1999.

[lindgren00] M. Lindgren, H. Hansson, and Thane, “Using Measurements to Derive the Worst-Case Execution Time,” In Proc. of RTCSA 2000, 2000.

[colin00] A. Colin and I. Puaut, “Worst Case Execution Time Analysis for a Processor with Branch Prediction,” Journal of Real-Time Systems, vol. 18, no. 2-3, pp. 249 – 274, Kluwer Academic Publishers, 2000.

[schneider99] J. Schneider and C. Ferdinand, “Pipeline Behavior Prediction for Superscalar Processors by Abstract Interpretation,” In Proc. of the ACM SIGPLAN 1999 workshop on Languages, Compilers, and Tools for Embedded Systems, pp. 35 – 55, 1999.

[stappert00] F. Stappert and P. Altenbernd, “Complete Worst-Case Execution Time Analysis of Straight-line Hard Real-time Programs,” Journal of Systems Architecture, vol. 46, no. 4, pp. 339 – 355, Elsevier North-Holland, Inc., 2000.

[ottosson97] G. Ottosson and M. Sjodin, “Worst-Case Execution Time Analysis for Modern Hardware Architectures,” ACM SIGPLAN 1997 Workshop on Languages, Compilers, and Tools for Real-Time Systems, 1997.

[lim95] S. Lim, et.al, “An Accurate Worst Case Timing Analysis for RISC Processors”, IEEE Trans. of Software Engineering, vol. 21, no. 7, pp. 593 – 604, 1995.

[kirner01] R. Kirner and P. Puschner, “Transformation of Path Information for WCET Analysis during Compilation,” Proc. of 13th Euromicro International Conference on real-Time Systems, pp. 29 – 36, 2001.

[healy98] C. Healy, M. S˝odin, V. Rustagi, and D. Whalley, ”Bounding Loop Iterations for Timing Analysis,” Proc. of 4th Real-Time Technology and Applications Symposium,” pp. 12 – 21, 1998.

[healy00] C. Healy, M. Sj˝odin, V. Rustagi, D. Whalley, and R. van Engelen, “Supporting Timing Analysis by Automatic Bounding of Loop Iterations,” Journal of Real-Time Systems, vol. 18, no. 2/3, pp, 129 – 156, 2000.

[lundqvist99] T. Lundqvist and P. Stenstr˝om, “An Integrated Path and Timing Analysis Method based on Cycle-Level Symbolic Execution,” Jounal of Real-Time Systems, vol. 17, no.2-3, pp. 183-207, Kluwer Academic Publishers, 1999.

[white97] R. T. White, et.al,“Timing Analysis for Data Caches and Set-Associative Caches,” Proc. of the 3rd IEEE Real-Time Technology and Applications Symposium (RTAS ’97), 1997.

[Luis06] E.L. Luis, M.Pedro, and D. de Niz, “Predictable Interrupt Management for Real-Time Kernels over conventional PC Hardware,” In Proc. of 12th IEEE RTAS, 2006.

[Timesys] TimeSys Corporation. TimeSys Linux. http://timesys.com/.
[Hildebrand92] D. Hildebrand, “An architectural overview of QNX,” In Proc. of the USENIX Workshop on Micro-kernels and Other Kernel Architectures, pp. 113-126, 1992.
[Kopetz89] H. Kopetz, et. al., “Distributed Fault-Tolerant Real-Time Systems: The MARS Approach,” IEEE Micro, 9(1), 1989.
[Lee05] Jupyung Lee and Kyu-Ho Park, “Delayed Locking Technique for Improving Real-Time Performance of Embedded Linux by Prediction of Timer Interrupt,” In Proc. of the 11th IEEE RTAS, 2005.
[Ishiwata99]石綿陽一, ``リアルタイム処理を実現する ART-Linux の設計と実装'', Interface, Vol.25, No.11, 1999.
[hsu05] C. Hsu and W. Feng, ″A Power-Aware Run-Time System for High-Performance Computing,″ ACM/IEEE conference on Supercomputing, 2005.
[nandini05] N. Kappiah, et.al, ″Just In Time Dynamic Voltage Scaling: Exploiting Inter-Node Slack to Save Energy in MPI Programs,″ ACM/IEEE conference on Supercomputing, 2005.
[ge05] R. Ge, et.al, ″Performance-constrained Distributed DVS Scheduling for Scientific Applications on Power-aware Clusters,″ SC05, 2005.
[pillai01] P. Pillai and K. G. Shin, ″Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems,″ SOSP 01, pp.89 - 102, 2001.
[vishnu02] V. Swaminathan, et.al, ″Experiences in Implementing an Energy-Driven Task Scheduler in {RT}-Linux,″ Real-Time and Embedded Technology and Applications Symposium, pp. 229-238, 2002.
[zhu05] Y. Zhu and F. Mueller, ″Feedback EDF Scheduling Exploiting Hardware-Assisted Asynchronous Dynamic Voltage Scaling,″ 2005.
[yuan06] W. Yuan and K. Nahrstedt, ″Energy-efficient CPU scheduling for multimedia applications,″ ACM Transactions on Computer Systems, vol.24, no.3, pp. 292-331, 2006.

(2) 研究成果の今後期待される効果

本研究成果である RETAS は、実用化に向けて、CREST の研究領域「実用化を目指した組込みシステム用ディペンダブル・オペレーティングシステム」(略称 DEOS)における研究課題「並列・分散型組込みシステムのためのディペンダブルシングルシステムイメージ OS」に引き継ぎ研究開発を進める。Shi-Linux 実現時に提案した割り込み手法や POPM 機構に関しても DEOS 研究領域において新たに開発されるディペンダブル Linux システムに反映していく。

3.3. ヒューマノイドロボット実証（産業技術総合研究所 ロボット実証グループ）

（1） 研究実施内容および成果

ヒューマノイドロボットの制御ソフトウェアは、Linux と CORBA の上に構築される。RMTP 上にこれらの環境を構築する作業と並行して、RMTP を中心にしたモータ制御モジュールの開発、画像処理や音響処理プログラムの評価、システムのアーキテクチャの検討を行った。

① モータ制御モジュール

各関節を分散的に制御するために開発した小型のモータ制御モジュールの主な仕様を表 3-3-1 に、その写真を図 3-3-1 に示す。モータ実装の制約から、45 ㎜角基盤に収める必要があり、1 枚の CPU ボードと 2 枚の高出力デジタルアンプを積層した形態とした。高出力デジタルアンプ部には1つの DC モータまたは AC モータ(ブラシレスモータ)を制御することができるように、6つのパワーMOSFET を搭載し、ソフトウェアによって使用する FET の数を変えられる構成とした。後部に見える従来のモータ制御モジュールに比して、体積比で 1/17 の縮小に成功している。アンプ部分の評価用に開発した DSP ボードと本アンプを用いて DC モータおよび AC モータを動作させる実験が完了し、RMTP とアンプを用いて DC および AC モータを回転させる実験にも成功している。AC モータは、耐久性にすぐれ高出力化が容易であるが、より高速で高精度の制御が必要であるため、ロボットへの実装は進んでいない。RMTP によって、AC モータの超小型サーボモジュールの実現が可能になる。

表 3-3-1 モータ制御モジュールの仕様

項 目	仕 様	備 考
基板寸法	40×45 [mm]	外出しｺﾈｸﾀ&ｹｰﾌﾞﾙ含まず
制御電源	DC 5 [V]	
モータ駆動電源	DC 48 [V]	
対象とするモータ	DCモータ & ACモータ	ﾌｧｰﾑｳｪｱによる切替
モータ駆動方式	PWM	ﾊﾟﾜｰMOSFET:6個搭載
連続定格電流	10 [A]	
対象とするｴﾝｺｰﾀﾞ	ｲﾝｸﾘﾒﾝﾀﾙｴﾝｺｰﾀﾞ	

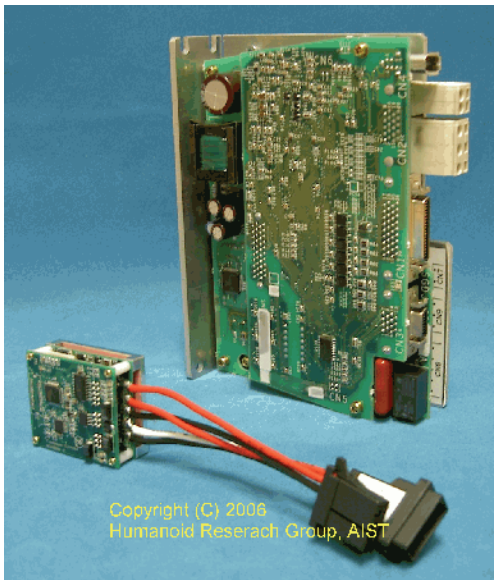


図 3-3-1 従来のモータ制御モジュール（後部)と、新規開発のモジュール

② 画像処理プログラムを用いた性能評価

ステレオ画像処理における相関演算を例題に、ベクタを使った並列化の性能を見るベンチマークを実施した。相関演算とは一方のカメラから得られた画像中の小領域に対応する画像を他方のカメラから得られた画像から検索する処理であり、相関度は SAD(小領域内の各画素値の差の絶対

値の和)によって評価する。RMTP のロジックバグを回避するためにコードの最適化が実行できないので正確な比較はできないが、C で記述された同一のプログラムを RMTP と Pentium で実行した場合、クロックに比例した実行時間がかかるという結果が得られている。またこのプログラムをベクタ化することによって 20 倍近い高速化が実現できている。したがって Pentium の20分の1のクロックの RMTP を用いて同程度のパフォーマンスを実現することが可能となっており、省電力化、高処理能力化が可能であることが確認できた。

③ システムアーキテクチャ

ヒューマノイドでの実証は、NEDO プロジェクトで開発中のヒューマノイドロボット HRP-3 をベースに体内情報系を置き換えて HRP-3 改とする形で行う。HRP-3 の上半身は Pentium を中核とした集中系として構成されており、開発したモータ制御モジュールのサイズをでも組み込みは困難である。そのため、HRP-3 改では下半身の全 14 自由度にモータ制御モジュールを用いる。HRP-3 改のハードウェアのシステム構成を図 3-3-2 に示す。HRP-3 において Pentium によって制御される各種 I/O は随時 RMTP による制御に切り替え、最終的には Pentium は PCI バスのコントロールと無線 LAN による外部との通信のために残す。

ソフトウェアシステム全体は、認識系、計画系を含めると非常な大規模になるが、μ RMTP を用いる運動制御系については、図 3-3-3 のような構成を準備している。全身運動制御の部分は複数の制御計算モジュールからなっており、それらのモジュールは HRP-3 では Pentium 上で1つのスレッドで順に実行される。HRP-3 改ではそれぞれのモジュールに実行用のスレッドを配置し、処理の依存関係に基づいて並列実行する計画である。

開発した HRP-3 改の外形図を図 3-3-4 に示す。RMTP-SIP が未完成なので、Pentium と専用 DSP による関節ノードを用いた分散制御系としている。オペレーティングシステムは、

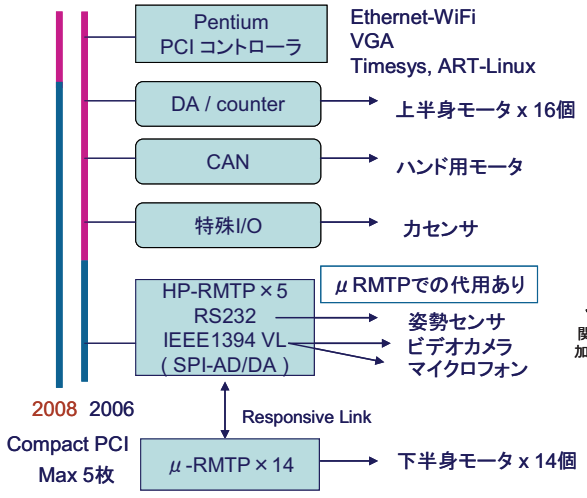


図 3-3-2 ヒューマノイドロボット制御用コンピュータの構成予測図

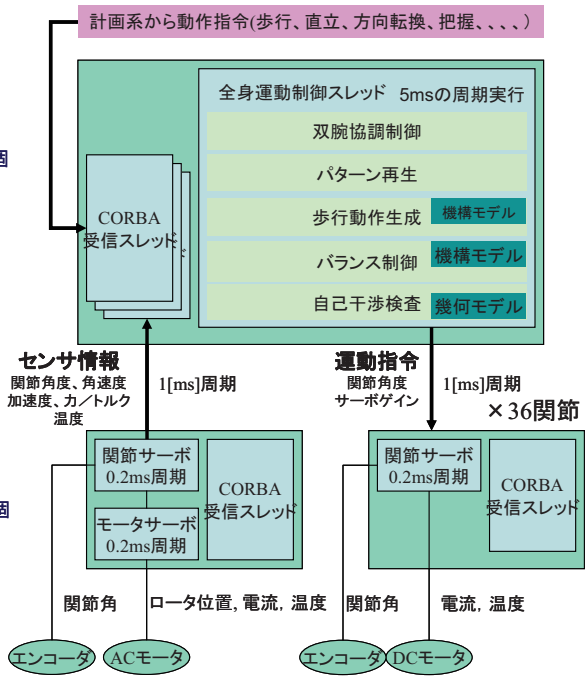


図 3-3-3 運動制御ソフトウェアの構成

ART-Linux-2.6 を用いている。

ART-Linux2.6 に関しては HRP2 の短周期軌道生成法を用いた歩行制御システムを利用して精度検証を行い、十分な精度を持っていることが確認できた。実験には、HRP-2 で使用しているのと同一機種の計算機(CPU は PentiumM-2.1[GHz])、および最新のシステムの例として Dell Precision490(CPU は Xeon5160-3.0[GHz]) の2つのシステムを用いた。これらのシステム上に、サーボI/O以外の計算処理を実装し、同一のソースコードから歩行制御を模擬した。具体的には、逆動力学演算を含む歩行軌道生成処理が 20[ms] 周期で、約 8.3[ms] (HRP2 での場合)の時間を消費して実行され、軌道の補間・再生、バランス制御及び、ログ出力の 3 つの 1[ms]周期のループが、それより高い優先度で実行されるという構成である。実行中に最高優先度に設定した 1[ms] 周期のループ(軌道の補間・再生ループ) の art_wait() 関数の直後でプロセッサのクロックカウンタの値を取得し、実行周期を計測した。計測された実行周期は、同一プロセス内のログ出力スレッドにより名前つきパイプを通して、非実時間プログラムに送られ、同プログラムが TCP ソケット通信により外部に送出することで、外部 PC を用いて記録した。その他、計測中には telnet 接続したターミナルに、ロボットの状態表示として、20[ms] につき 10 行程度のメッセージを表示している。表 1 に ART-Linux 2.6 で 30[s] および 2400[s] 間、前の周期の実行からの経過時間を計測し、周期の 1[ms] からの誤差を求めたもの(実験 1 と 2)と、比較のため HRP-2 12 号機上の ART-Linux2.4 で同様に計測したもの(実験 3 と 4)、最新のシステムとして Dell Precision490 で計測したもの(実験 5 と実験 6)を示す。

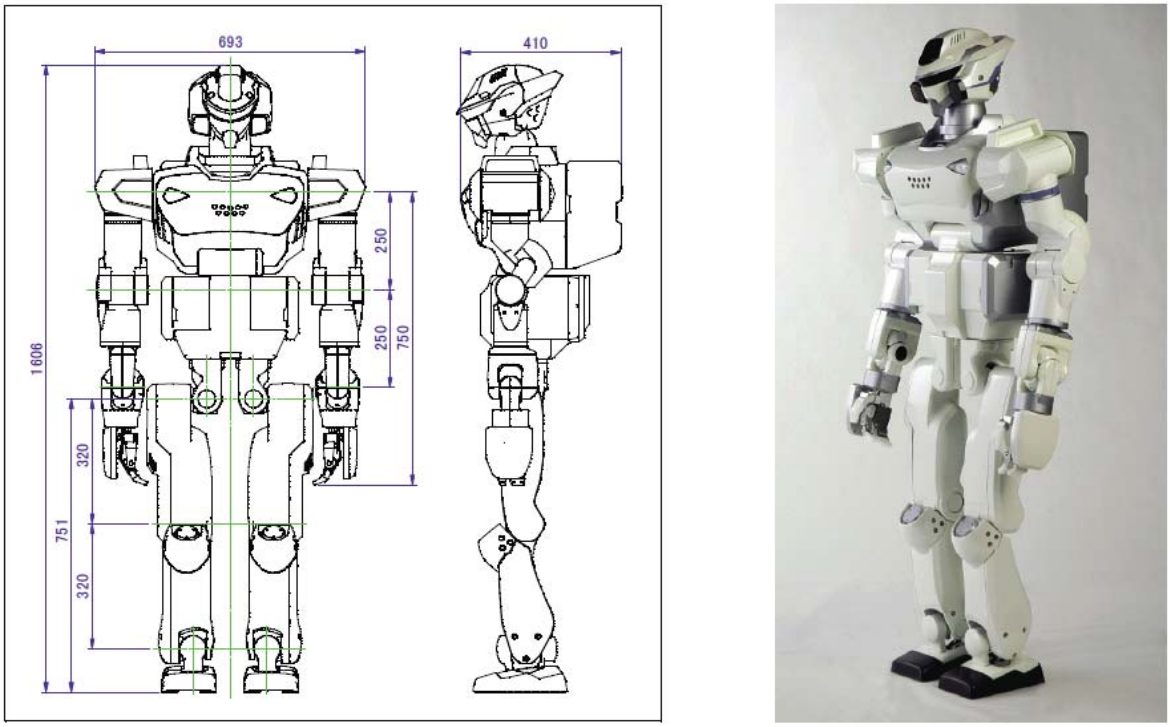


図 3-3-4 HRP3-改の寸法および外形

表 3-3-2　ART-Linux2.6 による HRP2 歩行制御系の 1ms 周期サーボループの性能評価

	Exp.1	Exp.2	Exp.3	Exp.4	Exp.5	Exp.6
ART Version	ART2.6		ART2.4		ART2.6	
CPU	PentiumM 2.1GHz				Xeon 5160 3.0GHz	
Chipset	855GME				5000X	
Duration	30[s]	2400[s]	30[s]	2400[s]	30[s]	2400[s]
Minimum Error [μs]	-2.2	-2.4	-2.1	-3.7	-2.4	-11
Maximum Error [μs]	1.9	4.3	2.1	3.5	2.1	11
Abs. Ave. of Error [μs]	0.19	0.25	0.11	0.21	0.29	0.29
Ave. of Error [μs]	-0.16	-0.16	-0.06	-0.16	-0.23	-0.23
St. Dev. of Error [μs]	0.23	0.28	0.19	0.20	0.24	0.23

(2) 研究成果の今後期待される効果

研究成果としては、ヒューマノイドロボットの分散系の設計、実時間 Linux の安定化とヒューマノイド制御への適用、小型関節制御モジュールの開発、ロボット制御用 RMTP 並列ライブラリなどがある。これらは、いずれも今後複雑化、大規模化するロボット制御にとって重要な技術であり、自由度数が増えていくに従って、適用範囲が広がっていくと考えられる。広く普及させていくには、プログラミング環境の整備、適用事例の追加、モジュールインタフェースの標準化などが必要となろう。標準化については、RT ミドルウェアの開発が経済産業省の支援の下、産業技術総合研究所を中心に推進されており、今回の成果も RT ミドルウェアの形で整備していくのが得策と考えられる。

§4 研究参加者 （○ 代表者 ＊ CREST 研究費による雇用者）

①RMT-LSI グループ（実時間通信・処理・制御用システムオンチップの研究開発）

	氏 名	所 属	役 職	研究項目	参加時期
○	山崎 信行	慶應義塾大学	教授	システム LSI の設計	H15.10 ～ H21.3
	小林 秀典	慶應義塾大学大学院	博士課程	開発環境の設計	H15.10 ～ H18.3
	伊藤 務	慶應義塾大学大学院	博士課程	システム LSI の設計	H15.10 ～ H18.3
＊	伊藤 務	慶應義塾大学	CREST 研究員	システム LSI の設計	H18.4 ～ H19.5
	加藤 真平	慶應義塾大学大学院	博士課程	開発環境の設計	H18.4 ～ H21.3
	船岡 健司	慶應義塾大学大学院	博士課程	開発環境の設計	H.18.4 ～ H21.3
	武田 瑛	慶應義塾大学大学院	修士課程	開発環境の設計	H19.4 ～ H21.3
	藤田 裕士	慶應義塾大学大学院	修士課程	開発環境の設計	H19.4 ～ H21.3

②基盤ソフトウェアグループ（実時間オペレーティングシステムと開発環境の開発）

	氏 名	所 属	役 職	研究項目	参加時期
○	石川 裕	東京大学大学院 情報理工学系研究科	教授	基盤ソフトウェア開発	H15.10 ～ H21.3
＊	載 毛兵	東京大学大学院 情報理工学系研究科	CREST 研究員	基盤ソフトウェア開発	H16.4 ～ H21.3
＊	山本 啓二	東京大学大学院 情報理工学系研究科	CREST 研究員	基盤ソフトウェア開発	H16.6 ～ H21.3
＊	野村 哲弘	東京大学大学院 情報理工学系研究科	CREST 研究員	基盤ソフトウェア開発	H18.4 ～ H19.3
＊	宮川 大輔	東京大学大学院 情報理工学系研究科	CREST 研究員	基盤ソフトウェア開発	H18.4 ～ H19.3

③ロボット実証グループ（ヒューマノイドロボットの分散制御系の研究）

	氏 名	所 属	役 職	研究項目	参加時期
○	比留川 博久	産業技術総合研究所 知能システム研究部門	グループリーダー	ヒューマノイド実証	H15.10～
	金広 文男	産業技術総合研究所 知能システム研究部門	主任研究員	ヒューマノイド実証	H15.10～

	斉藤　元	産業技術総合研究所 知能システム研究部門	産総研ポスドク	ヒューマノイド実証	H15.10～ H17.3
	金子　健二	産業技術総合研究所 知能システム研究部門	主任研究員	ヒューマノイド実証	H16.4～
	加賀美　聡	産業技術総合研究所 デジタルヒューマン研究センター	チームリーダー	ヒューマノイド実証	H15.10～
＊	吉村　志穂	産業技術総合研究所 デジタルヒューマン研究センター	CREST　事務員	CREST 事務	H15.12～ H16.6
＊	佐伯　奈保子	産業技術総合研究所 デジタルヒューマン研究センター	CREST　事務員	CREST 事務	H16.8　～ H17.3
	稲邑 哲也	大学共同利用機関法人 情報・システム研究機構国立情報学研究所	助教授	ヒューマノイド実証	H15.10～ H19.3

§5 招聘した研究者等

なし

§6 成果発表等

(1) 原著論文発表 (国内(和文)誌 11 件、国際(欧文)誌 3 件)

- 薄井弘之, 内山真郷, 伊藤務, 山﨑信行, “Responsive Multithreaded Processor の命令供給機構”, 情報処理学会論文誌, Vol. 45, No. SIG 11(HPS 7), pp. 105−118, 2004.
- 小林秀典, 山﨑信行, ″資源制約を持つ拡張インプリサイスタスクの実時間スケジューリング″, 情報処理学会論文誌コンピューティングシステム, Vol. 46, No.SIG 16, pp. 69 − 84, 2005.
- 加藤真平, 小林秀典, 山﨑信行, “SMT 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”, 情報処理学会論文誌コンピューティングシステム, Vol. 47, No. SIG 12, pp. 133−146, September, 2006.
- 船岡健司, 加藤真平, 山﨑信行, “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コンピューティングシステム, Vol. 48, No. SIG 3, pp. 1−12, Feburary, 2007.
- 金広文男, 石綿陽一, 齋藤元, 赤地一彦, 宮森剛, 五十棲隆勝, 金子健二, 比留川博久, “実時間 Ethernet を用いたヒューマノイドの分散 I/O システム”, 日本ロボット学会誌, Vol. 25, No. 3, April, 2007.
- 加藤 真平, 山﨑信行, “RMd2−SIP:マルチプロセッサにおける実時間スケジューリングアルゴリズム”, 情報処理学会論文誌: コンピューティングシステム, Vol. 48, SIG 13, pp. 270−286, September, 2007.
- 加藤 真平, 山﨑信行, ポーショニング方式に基づく動的優先度スケジューリング. 情報処理学会論文誌: コンピューティングシステム, Vol. 49, SIG 2, pp.113−126, March, 2008.
- 石綿 陽一, 加賀美 聡, 西脇 光一, 松井 俊浩, “シングル CPU 用 ART−Linux 2.6 の設計と開発”, 日本ロボット学会誌, Vol. 26, no. 6, pp.546−−552, September, 2008.
- 船岡 健司, 加藤 真平, 山﨑 信行, “マルチプロセッサ用の実時間電圧周波数制御”, 情報処理学会論文誌コンピューティングシステム, Vol. 1, No. 2, pp. 96−110, 2008 年.
- 加藤 真平, 山﨑 信行, “マルチコアプロセッサにおける非周期タスクの応答性向上”, 情報処理学会論文誌コンピューティングシステム, Vol. 1, No. SIG 2, pp. 83−95, 2008 年.
- 船岡 健司, 加藤 真平, 山﨑 信行, “マルチプロセッサ用の最適な実時間スケジューリング”, 情報処理学会論文誌コンピューティングシステム, 2008 年 (採録決定).
- N. Yamasaki, “Responsive Multithreaded Processor for Distributed Real−Time Systems″, Journal of Robotics and Mechatronics, Vol. 17, No. 2, pp. 130−141, 2005.
- T. Itou and N. Yamasaki, “Design and Implementation of the Multimedia Operation Mechanism for Responsive Multithreaded Processor”, Journal of Robotics and Mechatronics, Vol. 17, No. 4, pp. 456−462, August, 2005.
- K. Yamamoto, Y. Ishikawa, T. Matsui, “Portable Execution Time Analysis Method″, the 12th IEEE International Conference on Embedded and Real−Time Computing Systems and Applications, pp. 267−270, 2006.

(2) 学会発表(国際学会発表及び主要な国内学会発表)

① **招待講演** (国内会議 10 件、国際会議 4 件)

- 松井俊浩, “ヒューマノイドロボットのための実時間分散情報処理”, 電子情報通信学会・情報処理学会共催, 実時間処理ワークショップ(RTP2004), 産業技術総合研究所, 2004 年 3 月.
- 松井俊浩,“ヒューマノイドロボットが 2010 年代の Si デバイスをけん引”, 日経マイクロデバイス 20 周年記念講演会−デバイスイノベーション−, 2005 年 7 月.
- T. Matsui, N. Yamasaki, Y. Ishikawa, H. Hirukawa, and S. Kagami, “Distributed Real−Time Processing for Humanoid Robots”, RTCSA2005, 2005. 8. 18.
- 松井俊浩, “ロボットのエレクトロニクスと情報処理 −実時間分散制御向きプロセッサ”, Industry Strategy and Technology Forum (ISTF−2006) , 2006 年 10 月.
- 松井俊浩, “ロボット技術の動向と実時間分散制御向きプロセッサ”, 日本半導体ベンチャー協会 (JASVA) IP 設計委員会 講演会, November, 2006.
- 松井俊浩, “人間並みの情報処理は可能か? −ヒューマノイドロボット情報系の再構築”, 思考展開フォーラム, 大日本スクリーン(株) 主催, 京都, January, 2007.
- 加賀美 聡, “ヒューマノイドロボットの低レベル自律性向上”, 日本機械学会ロボティクス・メカトロニクス部門東北地区特別講演会 No.07 ヒト/生命の理解とロボティクス, 2007.
- 加賀美 聡, “ヒューマノイドロボットの視聴覚・行動システムの研究”, 東京大学 IRT 第一回 IRT フォーラム, 2007.

- 加賀美 聡, “ロボットの情報科学技術～ロボットがかしこく動くには？～”, 日本科学未来館イベント:展示の前で研究者に会おう！, 2007.
- T. Matsui, “Real-Time Processor for Robot Control”, IEEE Computer Elements Workshop, June, 2007.
- T. Matsui, “Real-Time Processing for Robot Control”, International WS on Home Robots and Manufacturing Technology, Korea, October, 23, 2007.
- T. Matsui, “Real-Time Robot Control”, Int. Meeting for Future of Electron Devices, Kansai, IMFEDK 2008, May, 2008.
- 山﨑信行, “分散リアルタイム制御用 SoC: Responsive Multithreaded Processor”, 第 21 回 回路とシステム軽井沢ワークショップ, pp. 593-598, April 21-22, 2008.
- 加賀美 聡, “三次元地図作成と自己位置同定および自律移動ロボットへの応用”, ロボット工学セミナー第 42 回シンポジウム ロボットの環境把握と自己位置推定, 千葉工業大学, June, 2008.

② **口頭発表** （国内会議 56 件、国際会議 37 件）

- 松井 俊浩, “ヒューマノイドのための実時間分散情報処理”, デジタルヒューマン・ワークショップ 2004, 日本科学未来館, 2004 年 3 月.
- 松井 俊浩, 比留川 博久, 石川 裕, 山﨑 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”, 電子情報通信学会技術研究報告:実時間処理に関するワークショップ, pp. 1-7, March, 2004.
- 小林 秀典, 山﨑 信行, “実時間オペレーティングシステム RT-Frontier におけるインプリサイス計算支援機構”, 電子情報通信学会技術研究報告:実時間処理に関するワークショップ, Vol. 103, No. 736, pp. 7-12, March, 2004.
- 薄井 弘之, 内山 真郷, 伊藤 務, 山﨑 信行, “Responsive Multithreaded Processorにおける実時間処理用命令供給機構”, 電子情報通信学会技術研究報告:実時間処理に関するワークショップ, pp. 15-20, March, 2004.
- 山﨑 信行, “Responsive Multithreaded Processorの全体設計”, 電子情報通信学会技術研究報告:実時間処理に関するワークショップ, pp. 9-14, March, 2004.
- 橘内 和也, 山﨑 信行, “レスポンシブリンクを用いた実時間マルチキャスト機構”, 電子情報通信学会技術研究報告:実時間処理に関するワークショップ, pp. 49-54, March, 2004.
- 佐々木 貴宏, 山﨑 信行, “リアルタイムチャネルを用いた End-to-End の実時間通信機構”, 電子情報通信学会技術研究報告:実時間処理に関するワークショップ, pp. 87-92, March, 2004.
- 山本啓二, 石川裕, 松井俊浩, “静的実行時間予測ツール構築のためのフレームワーク”, ソフトウェア科学会, 第 8 回 プログラミングおよび応用のシステムに関するワークショップ, 2005 年 3 月.
- 加藤真平, 小林秀典, 山﨑信行, “SMT プロセッサを対象としたリアルタイムスケジューリング″, コンピュータシステム・シンポジウム, Vol. 2005, No. 18, pp.109 - 118, 2005.
- 山浦康史, 鎌田展秀, 平哲也, 山﨑信行, “可動型赤外線センサを用いた動的スキャンによる地図生成システムの設計と実装″, 第 23 回日本ロボット学会学術講演会, pp.1-4, 2005.
- 新井誠一, 真垣郁男, 伊藤務 山﨑信行, “Responsive Multithreaded Processor のキャッシュシステムにおける低消費電力機構”, DA シンポジウム 2005, No. 9, pp.273-278, 2005.
- 中村哲朗, 小林秀典, 山﨑信行, “Responsive Multithreaded Processor における排他制御機構の設計と実装”, 並列／分散／協調処理に関する『武雄』サマーワークショップ, pp. 57-64, 2005.
- 村中延之, 伊藤務, 新井誠一, 山﨑信行, “Responsive Multithreaded Processor のスレッド間同期機構の設計と実装”, 電子情報通信学会技術研究報告:組込技術とネットワークに関するワークショップ, pp. 31-36, 2005.
- 加藤 真平, 小林 秀典, 山﨑 信行, “SMT プロセッサにおける統合的なリアルタイムスケジューリング機構″, 第 7 回 組込みシステム技術に関するサマーワークショップ,pp. 47-53, 2005.
- 加藤 真平, 橘内 和也, 小林 秀典, 山﨑 信行, “SMT 機構における実行時間変動を考慮した実時間スケジューリング手法”, 電子情報通信学会技術研究報告:組込技術とネットワークに関するワークショップ, pp. 7-12, 2005.
- 伊藤 務, 山﨑 信行, “Reponsive Multithreaded Processor のスレッド制御機構″,第 7 回 組込みシステム技術に関するサマーワークショップ, pp. 79-83, 2005.
- 新井誠一, 真垣郁男, 伊藤務 山﨑信行, “Responsive Multithreaded Processor のキャッシュシステムにおける低消費電力機構″, DA シンポジウム 2005, No. 9, pp.273-278, 2005.
- 中村 拓, 伊藤 務, 新井 誠一, 山﨑 信行, “Responsive Multithreaded Processor の分岐予測器の設

計と実装”，情報処理学会システムLSI設計技術研究会，Vol. 104，No. 738，pp. 37-42，2005.
- 中村哲朗，小林秀典，山﨑信行，″Responsive Multithreaded Processorにおける排他制御機構の設計と実装″，並列／分散／協調処理に関する『武雄』サマーワークショップ，pp. 57-64，2005.
- 小林 秀典，山﨑 信行，″実時間OS RT-Frontierにおけるインプリサイス計算機構″,第7回 組み込みシステム技術に関するサマーワークショップ，pp. 84-89，2005.
- 真垣郁男，伊藤務，山﨑信行，″Responsive Multithreaded Processorの命令供給機構およびIPC制御機構の設計と実装″,第7回 組み込みシステム技術に関するサマーワークショップ，pp. 71-78，2005.
- 村中延之，伊藤務，新井誠一，山﨑信行，″Responsive Multithreaded Processorのスレッド間同期機構の設計と実装″，電子情報通信学会技術研究報告:組込技術とネットワークに関するワークショップ，pp. 31-36，2005.
- 佐々木 貴宏，小林 秀典，山﨑 信行，″Responsive Linkを用いたリアルタイム通信ライブラリの設計および実装″，電子情報通信学会技術研究報告:組込技術とネットワークに関するワークショップ，pp. 61-66，2005.
- 山﨑信行，“分散リアルタイム制御用μRMTProcessorの設計”，第23回日本ロボット学会学術講演会，2005.9.15
- 得津 覚，杉山 悠，古城 直樹，岡田 慧，稲邑 哲也，稲葉 雅幸，“物を叩いたときに出る音を覚えて学習するヒューマノイドの研究”,第23回日本ロボット学会学術講演会，pp.3C32 (CD-ROM)，2005.9.17
- 杉山 悠，吉海 智晃，岡田 慧，稲邑 哲也，稲葉 雅幸,“生活音の認識と音源定位によるヒューマノイドの動作計画のための日常生活環境推定”，第 23 回日本ロボット学会学術講演会,pp.3C33 (CD-ROM),2005.9.1
- 松井俊浩，比留川博久，金子健二，加賀美聡，山﨑信行，石川裕，齋藤 元，“0 ヒューマノイドロボットの実時間分散制御”， 第23回日本ロボット学会学術講演会， 2005.9.15
- 戴 毛兵，石川裕，“Shi-Linux 実時間カーネル”，組込みシステム技術に関するサマーワークショップ(SWEST-9)，2005.
- 山本啓二，石川裕，松井俊浩，“実時間予測ツールの設計と実装”、情報処理学会 研究報告 2005-ARC-164，pp. 79-84，2005.
- 野村哲弘,石川裕，“リアルタイム通信のスケジューリング”，情報処理学会 研究報告 2005-OS-100，pp. 79-84，2005.
- 宮川大輔，石川裕，“プロセス単位電力制御機構の予備評価”，情報処理学会 研究報告 2005-OS-100，pp. 65-72，2005.
- 戴 毛兵，石川裕，“予測可能リアルタイムカーネルの設計と実装”，情報処理学会 研究報告 2005-OS-99，pp. 121-126，2005.
- 金子健二，金広文男，比留川博久，松井俊浩，山﨑信行，“ヒューマノイドのための実時間分散制御 -RMT Processor 搭載分散小型モータアンプの設計-”，日本ロボット学会第24回学術講演会予稿集，pp. 3H16，2006.
- 宮川大輔，石川 裕，“低消費電力のためのスケジューリングアルゴリズム”，情報処理学会 研究報告 2006-OS-102，pp. 17-22，2006.
- 宮川大輔，石川 裕，“電力制御スケジューラのプロトタイプ実装”，情報処理学会 研究報告 2006-OS-103，pp. 109-115，2006.
- 戴 毛兵，石川 裕，“厳密な周期タスク実行を支援する実時間Linux カーネルの実装”，情報処理学会研究報告 2006-OS-103，pp. 41-46，2006.
- 山本啓二，石川 裕，松井俊浩，“移植性の高い実行時間予測手法の設計と実装”，情報処理学会 研究報告 2006-ARC-169，pp. 127-132，2006.
- 中村哲朗，加藤真平，小林秀典，山﨑信行，“リアルタイム性を考慮したフィードバック制御による動的周波数制御手法”，電子情報通信学会技術研究報告： 組込技術とネットワークに関するワークショップ，Vol.105，No.670，pp. 7-12，筑波大学，March 16-17,2006.
- 加藤真平，小林秀典，山﨑信行，“拡張性及びオーバーヘッドを考慮したRMT Processor用リアルタイムスケジューラの設計と実装”，電子情報通信学会技術研究報告： 組込技術とネットワークに関するワークショップ，Vol. 105，No. 669，pp.31-36，March 16-17,2006.
- 船岡健司，加藤真平,小林秀典,山﨑信行,”RMT Processor 上のスキップ可能なタスクを扱うリアルタイムスケジューリング機構”，電子情報通信学会技術研究報告:組込技術とネットワークに関するワークショップ，Vol.105，No. 669，pp. 25-30，March 16-17,2006.
- 松井俊浩，“ヒューマノイドのための実時間分散情報処理”，CREST シンポジウム，November，2006.

- 船岡健司, 加藤真平, 山﨑信行, “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”, コンピュータシステム・シンポジウム, Vol. 2006, No. 14, pp. 3-12, 名古屋国際会議場, November 29-30, 2006.
- 加藤真平, 山﨑信行, “マルチプロセッサにおけるスケジュール可能性を向上するために動的優先度スケジューリングアルゴリズム”, 電子情報通信学会技術研究報告:組込技術とネットワークに関するワークショップ, Vol. 106, No. 602, pp. 43-48, March 15-16, 2007.
- 船岡健司, 加藤真平, 山﨑信行, “マルチプロセッサ用実時間スケジューリングにおける周波数制御手法”, 電子情報通信学会技術研究報告:組込技術とネットワークに関するワークショップ, Vol. 106, No. 602, pp. 49-54, 広島市立大学, March 15-16, 2007.
- 藤田裕士, 加藤真平, 山﨑信行, “Responsive Linkを用いた実時間通信におけるアドミッションコントロール”, 電子情報通信学会技術研究報告:組込技術とネットワークに関するワークショップ, pp. 31-36, March, 2007.
- 片山陽平, 加藤真平, 山﨑信行, “Responsive Link の追い越し用バッファによる待ち時間を有効利用した実時間通信”, 電子情報通信学会技術研究報告:組込技術とネットワークに関するワークショップ, pp. 37-42, March, 2007.
- 石綿 陽一, 加賀美 聡, 西脇 光一, 松井 俊浩, “シングル CPU 用 ART-Linux 2.6 の設計と開発”, 第 25 回日本ロボット学会学術講演会, pp.1037, 千葉工業大学, 千葉県習志野市, September, 2007.
- 武田瑛, 船岡健司, 加藤真平, 山﨑信行, “優先度付き SMT におけるデッドラインからの遅延時間を保証可能なソフトリアルタイムスケジューリング”, 電子情報通信学会技術研究報告:組込技術とネットワークに関するワークショップ, pp. 37-42, March, 2007.
- 加藤真平, 山﨑信行. “ポーショニング方式に基づく動的優先度スケジューリング”. コンピュータシステム・シンポジウム, pp. 163-172, November 27-28, 2007.
- 武田瑛, 船岡健司, 加藤真平, 山﨑信行, “マルチプロセッサにおけるグローバル RM に基づくリアルタイムスケジューリングアルゴリズム”, 電子情報通信学会技術研究報告:組込技術とネットワークに関するワークショップ, Vol. 107, No.559, pp.191-196, March, 2008.
- 藤田裕士, 加藤真平, 山﨑信行, “Responsive Link を用いた実時間通信における非周期パケットスケジューリング”, 電子情報通信学会技術研究報告:組込技術とネットワークに関するワークショップ, March, 2008.
- 武田瑛, 船岡健司, 加藤真平, 山﨑信行, “マルチプロセッサにおけるグローバル RM に基づくリアルタイムスケジューリングアルゴリズム”, 電子情報通信学会技術研究報告:組込技術とネットワークに関するワークショップ, Vol. 107, No. 559, pp. 191-196, March, 2008.
- 西脇 光一, 加賀美 聡, “短周期軌道生成法を用いたヒューマノイドの不整地歩行制御”, 第 13 回ロボティクスシンポジア予稿集, pp.463--469, March, 2008.
- 戴 毛兵, 石川 裕, “リアルタイム Linux のための軽量割り込み処理」”, 先進的計算基盤システムシンポジウム (SACSIS 2008), 2008 (発表予定).
- 加藤真平, 山﨑信行, “マルチコアプロセッサにおけるリアルタイム処理の応答性向上”, 先進的計算基盤システムシンポジウム, pp. 273-282, June 11-13, 2008.
- 加藤 真平, 山﨑 信行 “Linux カーネル用リアルタイムスケジューリングモジュール”, コンピュータシステムシンポジウム, November 12-13, 2008.
- S. Kato, H. Kobayashi and N. Yamasaki, ″U-Link Scheduling: Bounding Execution Time of Real-Time Tasks with Multi-Case Execution Time on SMTProcessors″, In Proceedings of The 11th IEEE International Conference onEmbedded and Real-Time Computing Systems and Applications , pp. 193-197,2005.
- S. Kato, K. Kitsunai, H. Kobayashi, N. Yamasaki, “Towards Real-Time Scheduling on Responsive Multithreaded Processor”, The 2005 International Conference on Parallel and Distributed Processing Techniques and Applications , pp. 1240-1246, Las Vegas, USA, June 27-30, 2005.
- S. Kato, H. Kobayashi, N. Yamasaki, ″U-Link Scheduling: Bounding Execution Time of Real-Time Tasks with Multi-Case Execution Time on SMT Processors″, 11th IEEE RTCSA , pp. 193-197,2005.
- N. Yamasaki, “Responsive Multithreaded Processor for Distributed Real-Time Processing”, 9th International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems, January 23-25, 2006.
- S. Kato, N. Yamasaki, “Extended U-Link Scheduling to Increase the Execution Efficiency for SMT Real-Time Systems”, The 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, pp. 373-377, Sydney, Australia, August 16-18, 2006.

- S. Kato, N. Yamasaki, “Extended U-Link Scheduling to Increase the Execution Efficiency for SMT Real-Time Systems”, The 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, pp. 373-377, Sydney, Australia, August 16-18, 2006.
- F. Kanehiro, Y. Ishiawata, H. Saito, K. Akachi, G. Miyamori, T. Isozumi, K. Kaneko, H.Hirukawa, “Distributed Control System of Humanoid Robots Based on Real-time Ethernet,” Proc. of the IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS-06), pp. 2471-2477, 2006.
- F. Kanehiro, Y. Ishiwata, H. Saito, K. Akachi, G. Miyamori, T. Isozumi, K. Kaneko, H. Hirukawa, “Distributed Control System of Humanoid Robots Based on Real-time Ethernet”, Proceedings of 2006 IEEE Int. Conf. on Intelligent Robotics and Systems (IROS2006), pp.2471-2477, 2006.
- K. Yamamoto, Y. Ishikawa, T. Matsui, “Portable Execution Time Analysis Method,” the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, pp. 267 - 270, 2006.
- N. Yamasaki, “Responsive Link for Distributed Real-Time Processing”, International Workshop on Innovative Architecture for Future Generation High Performance Processors and Systems, January, 2007.
- K. Funaoka, S. Kato, N. Yamasaki, “A Context Cache Replacement Algorithm for Pfair Scheduling”, The 15th Conference on Real-Time and Network Systems, pp. 57-64, LORIA, Nancy, France, March 29-30, 2007.
- N. Yamasaki, I. Magaki and T. Itou, “Prioritized SMT Architecture with IPC Control Method for Real-Time Processing”, The 13th IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 12-21, April, 2007.
- S. Kato and N. Yamasaki, “Feedback Controlled Server for Scheduling Aperiodic Tasks”, The 4th International Conference on Embedded Real-Time Computing Systems, pp. 402-407, July, 2007.
- S. Kato and N. Yamasaki, “Real-Time Scheduling with Task Splitting on Multiprocessors” , The 13th IEEE International Conference on Embedded and Real-TIme Computing Systems and Applications, pp. 441-450, August, 2007.
- S. Thompson, T. Horiuchi, S. Kagami, “An Environment Driven Model of Human Navigation Intention for Mobile Robots”, Proceedings of The 13th IASTED International Conference on Robotics and Applications, Wurzburg, Germany, August, 2007.
- S. Kagami, S. Thompson, Y. Sasaki, Y. Nishida, H. Mizoguchi, T. Enomoto, “Calling from the Other Room″ by Ceiling Ultrasonic Locator and Microphone Array”, Proceedings of IEEE/RSJ 2007 International Conference on Intelligent Robots and Systems(IROS2007), pp.2028--2034, San Diego, USA, October, 2007.
- S. Kato and N. Yamasaki, “Fixed-Priority Scheduling on Prioritized SMT Processor”, The 19th IASTED International Conference on Parallel and Distributed Computing and Systems, pp. 116-123, November, 2007.
- K. Funaoka, S. Kato and N. Yamasaki, “Real-Time Static Voltage Scaling on Multiprocessors”, The 19th IASTED International Conference on Parallel and Distributed Computing and Systems, pp. 142-149, November, 2007.
- K. Nishiwaki, S. Kagami, “Sensor Feedback Modification Methods that are Suitable for the Short Cycle Pattern Generation of Humanoid Walking”, Proceedings of IEEE/RSJ 2007 International Conference on Intelligent, Robots and Systems(IROS2007), pp.4214--4220, San Diego, USA, November, 2007.
- D. Miyakawa, Y. Ishikawa, ″Process Oriented Power Management,″ the IEEE Second Internal Symposium on Industrial Embedded Systems -- SIES’2007, 2007.
- D. Miyakawa, Y. Ishikawa, ″Process Oriented Power Management,″ the IEEE Second Internal Symposium on Industrial Embedded Systems -- SIES’2007, 2007. (CD-ROM)
- M. Dai, T. Matsui, and Y. Ishikawa, ″A Light Lock Management Mechanism for Optimizing Real-Time and Non-Real-Time Performance in Embedded Linux,″ To appear at IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, 2008.
- A. Takeda, K. Funaoka, S. Kato and N. Yamasaki, “Soft Real-Time Scheduling with Tardiness Bounds on Prioritized SMT Processors”, The IASTED International Conference on Parallel and Distributed Computing and Networks, pp. 252-257, Febrary, 2008.
- Y. Fujita, S. Kato and N. Yamasaki, “Real-Time Communication and Admission Control over Responsive Link”, The IASTED International Conference on Parallel and Distributed Computing and Networks, pp. 131-138, Febrary, 2008.

- K. Funaoka, S. Kato and N. Yamasaki, “Optimal Real-Time Scheduling for Efficient Aperiodic Services on Multiprocessors”, The IASTED International Conference on Parallel and Distributed Computing and Networks, pp. 245-251, Febrary, 2008.
- S. Kato and N. Yamasaki, “Portioned Static-Priority Scheduling on Multiprocessors”, The 22nd IEEE International Parallel and Distributed Processing Symposium, pp. CD-ROM, April, 2008.
- K. Funaoka, S. Kato and N. Yamasaki “Energy-Efficient Optimal Real-Time Scheduling on Multiprocessors”, The 11th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, pp. 23-30, May, 2008.
- M. Stilman, K. Nishiwaki, S. Kagami, “Humanoid Teleoperation for Whole Body Manipulation”, Proceedings of 2008 IEEE International Conference on Robotics and Automation, pp.3175--3180, May, 2008.
- K. Funaoka, A. Takeda, S. Kato and N. Yamasaki, “Dynamic Voltage and Frequency Scaling for Optimal Real-Time Scheduling on Multiprocessors”, The 3rd IEEE International Symposium on Industrial Embedded Systems, pp. 27-33, June, 2008.
- K. Funaoka, S. Kato and N. Yamasaki, “Dynamic Voltage and Frequency Scaling for Optimal Real-Time Scheduling on Multiprocessors”, In Proceedings of the 3rd IEEE International Symposium on Industrial Embedded Systems (SIES2008), CD-ROM, 2008.
- K. Funaoka, S. Kato and N. Yamasaki, “Work-Conserving Optimal Real-Time Scheduling on Multiprocessors”, The 20th Euromicro Conference on Real-Time Systems, pp. 13-22, July, 2008.
- K. Funaoka, S. Kato and N. Yamasaki, “New Abstraction for Optimal Real-Time Scheduling on Multiprocessors”, The 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, pp. 357-364, August, 2008.
- S. Kato and N. Yamasaki, “Global EDF-based Scheduling with Efficient Priority Promotion”, The 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, pp. 197-206, August, 2008.
- M. Dai, T. Matsui, and Y. Ishikawa, ″A Light Lock Management Mechanism for Optimizing Real-Time and Non-Real-Time Performance in Embedded Linux,″ To appear at IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, December, 2008.
- S. Kato and N. Yamasaki. Linux T-ReX: A Modular Real-Time Kernel. In Proceedings of the 10th Real-Time Linux Workshop (RTLWS10), 2008.
- S. Kato and N. Yamasaki, “Portioned EDF-based Scheduling on Multiprocessors”, In Proceedings of the 8th ACM/IEEE International Conference on Embedded Software (EMSOFT2008), 2008.
- S. Kato and N. Yamasaki, “Scheduling Aperiodic Tasks using Total Bandwidth Server on Multiprocessors”, In Proceedings of the 6th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC2008), 2008 (To appear).

③ **ポスター発表**　　（国内会議 7 件、国際会議 3 件）

- 山﨑信行,”分散リアルタイム処理用 Responsive Multithreaded Processor”, TX テクノロジー・ショーケース ツクバ・イン・アキバ 2005, 2005.9.21～22
- 村中延之, 伊藤務, 山﨑信行, “RMT Processor 用の柔軟なハードウェア同期機構”, 第 8 回 組み込みシステム技術に関するサマーワークショップ, pp. 127-133, 遠鉄ホテル エンパイヤ (静岡県浜松市舘山寺町), July 13-14, 2006.
- 村中 延之, 伊藤 務, 山﨑 信行, “RMT Processor 用の柔軟なハードウェア同期機構,” 第 8 回 組み込みシステム技術に関するサマーワークショップ, pp. 127-133, 遠鉄ホテル エンパイヤ (静岡県浜松市舘山寺町), July 13-14, 2006.
- 加藤真平, 山﨑信行, “マルチコアを対象とした応答性の高い実時間カーネル”, 第 9 回 組み込み技術に関するサマーワークショップ, 2007.
- 船岡健司, 加藤真平, 山﨑信行, “実時間マルチプロセッサスケジューリング機構”, 第 9 回 組み込みシステム技術に関するサマーワークショップ, pp.89-91, 2007.
- 加藤真平, 山﨑信行, “モジュール型リアルタイム Linux”, 第 10 回 組み込みシステム技術に関するサマーワークショップ, September 4-5, 2008.
- 藤田 裕士, 加藤 真平, 山﨑 信行, “Responsive Link 用の非周期パケットスケジューリング”, 第 10 回 組み込み技術に関するサマーワークショップ, September 4-5, 2008.
- M. Takahashi, K. Nishiwaki, S. Kagami, H. Mizoguchi, “Development and Evaluation of An Attitude

Measuring System that uses, Acceleration Information of Walking for Biped Robots", Proceedings of The 6th Annual IEEE Conference on Sensors(Sensors2007), pp.292--295, Atlanta, USA, October, 2007.

- K. Nishiwaki, S. Kagami, "Walking Control on Uneven Terrain with Short Cycle Pattern Generation", Proceedings of IEEE-RAS International Conference on Humanoid Robots (Humanoids2007), Pittsburgh, PA, USA, November, 2007.
- S. Kagami, Y. Sasaki, S. Thompson, T. Hujihara, T. Enomoto, "Loudness Measurement of Human Utterance to a Robot in Noisy Environment", Proceedings of Third International Conference on Human-Robot Interaction (HRI2008), pp.217-224, March, 2008.

(3) 特許出願

① 国内出願 (4 件)

1. 発明の名称:マルチスレッド中央演算装置および同時マルチスレッディング制御方法
   発明者:山﨑信行
   出願人:慶應義塾大学
   出願日:2005 年 6 月
   特願 2005-167427

2. 発明の名称:マルチタスク処理装置、及びマルチタスク処理方法、並びに、プログラム
   発明者:石綿 陽一, 加賀美 聡, 西脇 光一, 松井 俊浩
   出願人:独立行政法人産業技術総合研究所
   出願日:2007 年 9 月
   特願 2007-236087

3. 発明の名称:マルチタスク処理装置、及びマルチタスク処理方法、並びに、プログラム
   発明者:石綿 陽一, 加賀美 聡, 西脇 光一, 松井 俊浩
   出願人:独立行政法人産業技術総合研究所
   出願日:2007 年 9 月
   特願 2007-236087

4. 発明の名称:実時間タスクにおけるブロック回避方法
   発明者:石綿 陽一, 加賀美 聡, 西脇 光一, 松井 俊浩
   出願人:独立行政法人産業技術総合研究所
   出願日:2007 年 9 月
   特願 2007-236390

② 海外出願 (2 件)

1. 発明の名称:Context switching method, context switching unit, context switching program, storage medium, and central processing unit
   発明者:山﨑信行
   出願人:慶應義塾大学と科学技術振興機構の共同出願
   出願日:2005 年 6 月
   PCT 出願番号:PCT/JP03/15838

2. 発明の名称:Multithreaded central processing unit and simultaneous multithreading control method
   発明者:山﨑信行
   出願人:慶應義塾大学
   出願日:2006 年 6 月
   PCT 出願番号:PCT/JP2006/311022

(4) 受賞等

① 受賞

- 山﨑信行, 情報処理学会システム LSI 設計技術研究会, 2004 年 7 月. (優秀論文賞)
- 加藤 真平, 小林 秀典, 山﨑 信行, “SMT プロセッサにおける統合的なリアルタイムスケジューリング機構”, 第 7 回 組込みシステム技術に関するサマーワークショップ (SWEST7), ポスター発表, 2005 年 8 月. (プレゼンテーション賞)
- N. Yamasaki, 9th International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems, January 2006. (Presentation Award)
- S. Thompson, “Realtime Autonomous Navigation with a 3D Laser Range Sensor”, International Conference on Autonomous Robots and Agents 2006, 2006. (Best Paper Presentation)
- 宮川大輔, 情報処理学会 OS 研究会, 2006 年 5 月. (最優秀学生発表賞)
- 加藤 真平, 山﨑 信行, “RMT Processor を対象としたフィードバック制御サーバ”, 第 8 回 組み込みシステム技術に関するサマーワークショップ (SWEST8), ポスター発表, 2006 年 7 月. (オリジナリティ賞)
- 宮川大輔, 情報処理学会 OS 研究会, 2006 年 8 月. (最優秀学生発表賞)
- T. Matsui, “Real-Time Processor for Robot Control”, IEEE Computer Elements Workshop, June, 2007. (ベストプレゼンテーション賞)
- 山﨑信行, 情報処理学会 情報規格調査会, 2007 年 7 月.(標準化貢献賞)
- 山﨑信行, 経済産業省 国際標準化奨励者表彰, 2007 年 10 月.(産業技術環境局長表彰)
- 船岡 健司, 加藤 真平, 山﨑 信行, “マルチプロセッサ用実時間スケジューリングにおける周波数制御手法”, 電子情報通信学会技術研究報告:組込技術とネットワークに関するワークショップ (ETNET2007), Vol. 106, No. 602, pp. 49-54, 2007 年 3 月. (優秀学生発表賞).
- 加藤 真平, 山﨑 信行, “マルチプロセッサにおけるスケジュール可能性を向上するために動的優先度スケジューリングアルゴリズム”, 電子情報通信学会技術研究報告:組込技術とネットワークに関するワークショップ (ETNET2007), Vol. 106, No. 602, pp. 43-48, 2007 年 3 月. (優秀学生発表賞).
- K. Funaoka, S. Kato and N. Yamasaki, “Energy-Efficient Optimal Real-Time Scheduling on Multiprocessors”, In Proceedings of the 11th IEEE International Symposium on Object/Component/Service-oriented Real-Time Distributed Computing (ISORC2008), pp. 23-30, 2008. (Best Student Paper Award)
- S. Kato and N. Yamasaki, “Global EDF-based Scheduling with Efficient Priority Promotion”, In Proceedings of the 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA2008), pp. 197-206, 2008. (Best Paper Award).
- 山﨑信行, 2008 年 6 月, 情報処理学会 情報規格調査会.(国際規格開発賞)
- 加藤 真平, 山﨑 信行, “モジュール型リアルタイム Linux”, 第 10 回 組み込み技術に関するサマーワークショップ (SWEST10), ポスター発表, 2008 年 9 月. (ベストポスター賞)

② 新聞報道

なし

③ その他

- 松井俊浩, “ヒューマノイド・ロボットが 2010 年代の Si デバイスをけん引”, NIKKEI MICRODEVICES, pp. 75-82, 2004 年 8 月
- 松井俊浩, “組込システム開発事例, ロボット”, IT Text 組込システム, 阪田史郎, 高田広章 編著, オーム社, October, 2006.
- 山﨑信行, “分散制御とセンサ技術”, 電気学会誌, Vol. 127, No. 3, pp. 156-160, March, 2007.

(5) その他特記事項

- Responsive Link 標準化
  - 国内:情報処理学会試行標準 ITSJ-TS 0006:2003
  - 国外:ISO/IEC 24740

§7 研究期間中の主な活動

年月日	名称	場所	参加人数	概要
H15.9.29	キックオフミーティング	デジタルヒューマン研究センター	10	
2003年10月 ～ 2008年10月	チーム会議 TV会議	デジタルヒューマン研究センター、つくば知能システム研究部門、東京大学、慶應義塾大学	3-10	プロジェクト推進のためのチーム会議を3日から 2 週間おきに開催
2003年～ 2008年	領域ヒアリング	JST事務所	５－９	領域総括、アドバイザへの進捗報告 年2回
2003年～ 2008年	CREST シンポジウム	駒場エミナース、科学未来館など	50	2005年には中間報告、他は展示発表
H17.4.28	講習会	産総研臨海副都心センター	10	ソフトウェア開発環境講習会
H17.5.24	講習会	産総研臨海副都心センター	10	HRP-2ソフトウェア講習会
H18.10.20	サイトビジット	産総研つくば	14	進捗報告とデモ

§8 結び

全体を3つの要素研究として 3 グループが実施したが、RMTP の開発に手間取ったため、3つのステージを完全に接続することはできなかった。ヒューマノイドロボット向けに、プロセッサと OS のような情報系の基盤から開発するというのは、チャレンジングで、ヒューマノイドロボットに経験のある日本の強みを生かしたプロジェクトであった。RMTP 特有の、ロボットのための実時間アーキテクチャはの技術者の関心を呼び、多くの招待講演を行い、学会賞も獲得した。レスポンシブリンク実時間ネットワークが、ISO の国際規格に承認されたことも大きな成果である。最近は、Linux に実時間機能を入れようとする試みが多くなされており、Shi-Linux や ART-Linux は、そこに大きな影響を与えられると考えられる。ロボットを分散的に制御するのは、まだ一般的ではないが、これからロボットが複雑さを増していけば、集中制御から分散制御に移行するのは必然と考えられる。

RMTP 開発を今後継続するには、まだ多くの費用がかかる。LSI 開発とは、実験ではなく大量に生産して始めて成り立つビジネスであるので、企業への技術移転とセットで考える必要がある。大量生産を可能にするには、RMTP を使ったアプリケーションを明確にする必要がある。ロボットにこだわらず、デジタルシネマのような、リアルタイム性と計算能力の両方を必要とするアプリケーションが突破口となるかもしれない。いずれにせよ、大学や研究所の考えだけで開発を継続すべきではない。

ソフトウェアは、LSI に比べてはるかに改版が容易であり、漸近的な開発を続けられる。実時間スケジューラや WCET 解析は、ディペンダブル情報システムの要素であり、ディペンダブルオペレーティングシステムの CREST プロジェクトとして発展させる。

ロボットは、DSP を用いて分散アーキテクチャでの制御はできたが、関節ノードに計算を分担させるまでには至っていない。分散させると、関節が大きくなること、システム全体のメンテナンスが困難になること、などの問題も認識されている。また、Pentium を用いて、2 足歩行軌道の生成を 1 秒周期から 20ms 周期に短縮したところ、不整地歩行能力や安定性維持に顕著な効果があることがわかった。短周期での実時間性を追求する我々の方針には間違いがなかったことになる。今後、ヒューマノイドにさらに多様な動作をさせること、人と共存するために安定性を増すことなどの研究が展開すると思われるが、ヒューマノイドならではアプリケーションが不明確なことと、コストがかかることから、しばらくは、基礎研究として継続するべきと考える。

研究のマネジメントについては、JST の本部と領域事務所に多大なご支援をいただいた。各種の競争的プロジェクト予算の中では、年度ごとに必要な予算を割り当てられること、年度の途中であっても、加速的な予算の配布を受けられること、備品管理を JST が行ってくれること、案件によっては JST 管理予算と受託機関管理予算を選択できることなど、研究を進めやすくする観点からマネージされていると感じた。年に 2 回以上行われる領域のヒアリングとシンポジウムでは、領域総括および評価委員の方々からさまざまなアドバイスを頂いた。感謝の意を表します。