| 報告番号 | ※甲　　第　　号 |
|---|---|

# 主　論　文　の　要　旨

論文題目　リアルタイム OS のハードウェア化による
マルチコア組込みシステムの高速化

氏　　名　丸山　修孝

## 論 文 内 容 の 要 旨

本論文の目的は，組込みシステムにおけるアプリケーションの高速化，特に高速化要求が高くまた従来技術で高速化が困難なネットワークプロトコル処理や機械制御処理の高速化である．本論文では，従来技術による高速化における障害や問題点を明確にすると共に，問題を解決し高速化を実現する新たな手段としてリアルタイム OS（RTOS）のハードウェア化を提案する．以下本論文の要旨を述べる．

### 第一章　概要

本論文の概要説明である．

### 第二章　組込みシステムの高速化技術

組込みシステムの共通した要求として低コスト，低消費電力を挙げることができる．コンシューマ機器では低コスト・低消費電力が重要であるが自動車や工場のように多量に組込みシステムを使用する産業分野においても，全体として低コスト・低消費電力が求められており，これを実現するためには個々の組込み装置の低コスト・低消費電力化を実現する必要である．さらに分野を問わずモバイルシステムが増加しており，バッテリー動作のため低消費電力化が強く望まれる．

一方、従来技術によるアプリケーションの高速化は，コアの高速化により実現する手法が主流である．コアのクロックレートの高速化は，周辺のメモリやバスシステムの高速化をも要求するため，システム全体のコスト・消費電力高を招く．したがって従来手法により上記低消費電力化・低コスト化という条件を満たしつつアプリケーションの高速化を実現することは困難である．

組込みシステムでは複数のソフトウェアモジュール（タスク）を管理実行させるた

め，RTOS を使用することが多い． RTOS は複数のタスクのスケジューリングを行い，またタスク間での同期機能や通信機能など共通で使用できる機能を提供する．RTOS を利用することにより次のようなメリットを得ることができる．

(1) 定義された API (Programming Interface) 上にアプリケーションソフトウェアを開発するため，ソフトウェアの部品化，再利用化が容易であり，ソフトウェア開発生産性が向上する．

(2) スケジューリング機能，同期機能や通信機能など複雑な機能が既に RTOS に実装されており，システムの信頼性が向上する．

このため組込みシステムでは RTOS を利用することが望ましい．

ネットワークプロトコル処理は複数の処理が同時に発生する．例えば，送信と受信の要求は独立して発生するため処理も独立にすべきである．またパケット毎に処理すべきものとコネクション単位で処理すべきものは別タスクとすべきである．こうしたことからネットワークプロトコルはマルチタスクとして実現されることが多く，またこのため頻繁に RTOS 機能を使用してタスク間の同期通信を行っている．したがって，ネットワーク処理における RTOS 処理の CPU 占有率は極めて高く，実験結果では純然たるプロトコル処理の 3 倍もの CPU 時間を RTOS 処理が占有しており，RTOS のオーバヘッドがネットワーク性能向上の妨げになる．

機械制御においては高度なリアルタイム性能を実現することが重要であり，RTOS のオーバヘッドがリアルタイム性を実現する上での障害になる．リアルタイム性とは周期的，かつ確実な実施を求められる制御である．周期を短くすることにより高精度なモーター制御が可能になる．周期を短くするためには高速な割り込み応答が必須である．要求性能が厳しいアプリケーションの例では，割り込み発生から 1$\mu$ 秒以内に周期処理ソフトウェアが立ち上がらなければならない．RTOS を利用するとそのオーバヘッドのためこのような厳しい割り込み応答性能を実現することが難しい．

本論文では，RTOS 性能を著しく向上させため，RTOS のハードウェア化を提案する．これにより，コアの性能を上げることなくアプリケーションの性能を大幅に向上させることができ，低コスト，低消費電力という条件も満たすことができる．

## 第三章　シングルコア対応 RTOS のハードウェア化による TCP/IP 処理の高速化と低消費電力化

本章では，シングルコア対応 RTOS のハードウェア化（HWRTOS）と，これによる TCP/IP スループットの向上について提案する．本システムのためにオリジナルのコアを開発し， HWRTOS とコア間に専用インターフェースを定義し，密結合 HWRTOS を実現した．本 HWRTOS は 40 種類の ITRON 仕様の API を実装し，既存の ITRON ベースで開発されたソフトウェアの容易なポーティングを実現する．

一般に RTOS は数千個のキューが必要である．従来技術でキューをハードウェア

で実現しようとした場合，ハードウェア FIFO を使用するのが一般的であるが，数千個のキューを LSI に実装することはコスト増につながる．本章では仮想キューという新しいキューイングシステムを提案した．仮想キューはキュー情報の可逆性のある圧縮技術であり，数千個のキューを少ない回路量で実現し，また極めて短時間でのキュー操作を実現する．

本 HWRTOS の性能を測定したところ，従来のソフトウェア RTOS（SWRTOS）の API 実行時間が約 300 サイクル程度から 600 サイクル程度であるのに対し，ARTESSO HWRTOS の性能は 10 サイクル時間前後であった．従って従来比数十倍の性能を達成した．また 90nm プロセスを使用した ASIC を試作開発した．この ASIC 上に TCP/IP ファームウェアを実装しスループットを測定した，結果 50MHz の動作クロックにおいて，従来の SWRTOS を使用した汎用 CPU では 11Mbps であったのに対し，本システムでは 125Mbps であった．また 100Mbps のスループット時の消費電力は従来技術の 1/7 であり，本システムが低消費電力であることを実証した．

第四章 超高速応答を実現するハードウェア割り込み処理機構

本章では第三章で開発した HWRTOS にさらに 2 つの機能拡張をし，割り込み応答性能を向上させた．一つは tick オフローディング機構，もう一つは IIA（Interrupt Invoked API）オフローディング機構である．

まず tick オフローディングについて説明する．割り込み応答性能を劣化させる要因として割り込み禁止状態がある．長い割り込み禁止状態を作り出す原因は tick 処理である．tick 処理は，RTOS で使用しているウエイトタイマの更新，タイムアウトの検出，タイムアウト時の処理等を行う．従って tick 処理は RTOS の重要な管理情報を更新する処理であり，他の処理に比較し優先度を高くしかつ割り込み禁止で実行される．このため割り込み応答時間に大きな影響を与える．本章では tick 処理を完全にハードウェア化し，ソフトウェアによる tick 処理を不要とする tick オフローディング機構を提案した．これによりコアの動作クロックを高速にすることなく，割り込み応答性能を大幅に向上させまた割り込み応答時間の変動もほとんどなくなった．

次に IIA オフローディングについて説明する．割り込み発生時における従来の SWRTOS の処理の流れは以下の通りである．まず現在実行中のタスクを中断し，ISR（Interrupt Service Routine）が起動され，ISR が割り込み処理を実行し，完了すると再びタスク処理に戻る．「割り込み発生から再びタスク処理に戻るまでにかかる時間」は，SWRTOS では数百～数千サイクル時間を要していた．本章では ISR 機能を定型化し，ハードウェア化することによりこの時間を大幅に短縮する手法を提案した．これを IIA オフローディングと呼ぶ．IIA オフローディングにより，コアの動作クロックを高速にすることなく，「割り込み発生から再びタスク処理に戻るまでの時間」は数百～数千サイクルから 14～17 サイクル時間に短縮した，またこの時間の変動幅を大幅に圧縮

した.

以上, RTOS を使用した環境においても, 高速な動作クロックを使用することなく, すなわちコスト・消費電力の増加を伴うことなく, 割り込み応答性能の大幅な向上を実現し, 組込みシステムにおける機械制御処理の高速化実現を可能にした.

第五章　シングルコア対応疎結合ハードウェア RTOS 搭載 産業ネットワーク用 SOC

本章では第三章で開発した HWRTOS に改造, 機能追加を行い, 汎用プロセッサである ARM コアからこの HWRTOS を利用できるようした疎結合 HWRTOS を提案する. ルネサスエレクトロニクス(株)ではこのプラットフォームを利用し, FA 用ネットワーク処理用 ASSP を開発, 現在量産中である.

第三章、第四章ではオリジナル・コアを使用する. しかし組込みシステム用の汎用 LSI においては, ユーザは ARM 社のコアの実装を求めている. これは既に開発したソフトウェア資産を利用しやすいこと, 今まで使用していた開発環境を利用できること, コアとしてのスケーラビリティがありまた信頼性が高いことが理由である. オリジナル・コアと HWRTOS は, 専用インターフィースで接続され密結合 HWRTOS を構成するが, ARM コアは専用インターフェースがない. このため本章では, ARM バスを利用して ARM コアと HWRTOS を接続する疎結合 HWRTOS を提案した。さらに第四章で提案した tick オフローディングおよび IIA オフローディング機構がコアと並列実行できるように改造を行い, コアのアベイラビリティを向上させた.

開発・量産した ASSP (R-IN32M3) は ARM コアとして Cortex-M3 100MHz を採用, 産業用 Ethernet を実現するため Ethernet を 2 ポート実装した. また機械制御用に CAN を 2 ポート, I2C を 2 ポート, CC-Link を 1 ポート実装した.

第六章　マルチコア対応 RTOS のハードウェア化による性能向上

本章ではマルチコア対応の密結合 HWRTOS を提案する. まずマルチコア RTOS として必要であるロックシステムに関し検討し, ジャイアントロック方式を採用した. そしてジャイアントロック方式に基づくアーキテクチャ設計を行った. アーキテクチャ設計にあたっては第四章のシステムをベースとし, またマルチコア特有の課題であるスタベーション, 排他制御の課題を防止する回路を実装した.

アーキテクチャ設計に基づき, FPGA を使用してオリジナル・コアを 8 個実装した試作システムを開発し, 機能・性能評価を行った. この結果コアを跨いだ API 実行時間が, SWRTOS では 850～900 サイクル時間であったのに対し, 開発したマルチコア対応密結合 HWRTOS は 25～27 サイクルであった.

以上, マルチコア組込みシステムにおいても, HWRTOS 上での動作環境を実現

した．すなわち低動作クロックレート，つまり低コスト・低消費電力下において，マルチコア組込みシステムでネットワーク処理，機械制御処理の高速化の実現を可能にした．

学位関係