計算機アーキテクチャ　103－24
設　計　自　動　化　69－24
(1993. 12. 17)

# リアルタイムOSのハードウェア化とその評価

板橋光義† Andy Utama† 仲野　巧†† 塩見彰睦† 今井正治†

†豊橋技術科学大学　情報工学系　††豊田工業高等専門学校　情報工学科

†〒440　愛知県豊橋市天白町字雲雀ヶ丘1-1
††〒471　愛知県豊田市栄生町2-1

あらまし　多くの組み込みシステムでは、応用プログラムの生産性、信頼性、保守性の向上及び各種の処理要求に対して高速性を実現するためにリアルタイムOSを使用している。OSは従来ソフトウェアで実現されることが普通であるが、この方法では高速化に限界がある。我々は、リアルタイムOSの1つである$\mu$ITRONに着目し、そのシステムコールの処理をハードウェア化することでOSの高速化を目指す。本稿では、実現したOSチップ(STRON)のアーキテクチャ、評価及びその有効性について述べる。STRONチップを用いることでシステムコール処理時間、割り込み禁止時間及びタスク切り替え時間が1/10～1/3に短縮できることが知られた。

和文キーワード　組み込みシステム、リアルタイムOS、ITRON、$\mu$ITRON、STRON、FPGA

# Implement Real-Time OS by Hardware and Evaluation

Mitsuyoshi ITABASHI† Andy Utama† Takumi NAKANO††
Akichika SHIOMI† Masaharu IMAI†

†Toyohashi University of Technology　††Toyota College of Technology

†1-1 Hibariga-oka, Tempaku-cho, Toyohashi-shi, Aichi, 441 Japan
††2-1 Eisei-cho, Toyota-shi, Aichi, 471 Japan

Abstract　Most of the embedded systems utilize real-time OS's because of the productivity, reliability, and maintainanceability of application programs. While OS's are usually implemented in software, there is a limitation in the performance. This paper proposes a method to implement a real-time OS ($\mu$ITRON) in hardware to achieve the highest performance as possible. The VLSI architecture and the performance of the designed OS chip(STRON) are shown. According to the performance evaluation, it was known that the system-call processing time, the interrupt prohibition time, and context switch time of the STRON can be reduced to 1/10 ~ 1/3 compared to the pure software implementation of $\mu$ITRON.

英文 key words　embedded system, real-time OS, ITRON, $\mu$ITRON, STRON, FPGA

## 1 はじめに

現在、マイクロプロセッサは、家庭用電化製品、産業用ロボット、飛行機の自動操縦装置などの幅広い分野で使用されている。このような組み込みシステムは、年々高機能化しておりソフトウェアの生産性、信頼性、保守性の向上が重要な課題となっている。

また、産業機器を制御する組み込みシステムでは、家庭電化製品などの制御と比較してより厳しい処理能力が要求される。特に、工場のライン制御や産業用ロボットなどの組み込みシステムでは、複数のセンサからの信号を処理要求としてリアルタイムに受け付け、応答する能力が必要である。

これらの組み込みシステムの多くではリアルタイムOSが使用されている。特に東京大学の坂村らによって提案されたITRON(Industrial The Realtime Operating system Nucleus)[1]は、このようなリアルタイムOSの1つであり業界標準のOSとなりつつある。

しかし、リアルタイム性の厳しい組み込みシステムでは、ソフトウェアで実現されたOSの処理時間を短縮させることには限界がある。そこで、ITRONのシリーズの1つである$\mu$ITRONに準拠したOSのハードウェア化を行い、処理時間の短縮を試みた。

本稿では、リアルタイムOSに要求される性能、それを実現するITRONの概要について述べる。次に$\mu$ITRONのサブセットをハードウェアで実現したOSチップ(以下、STRON-1: Silicon TRON Version-1)のアーキテクチャ、実現方法及びその性能の評価結果について述べる。

## 2 リアルタイムOSに対する要求

マイクロコンピュータを利用した組み込みシステムでは、外部で発生したいろいろな信号や環境の変化をセンサなどで検出して適切な処理を行い、外部に応答信号を出力する。また、外部に対して周期的に処理結果を出力する能力が必要となる。

組み込みシステムでは、ある1つの目的を持った作業をプログラムの生産性、信頼性、保守性を向上させるために複数のタスクに分割し、OSが各タスクの実行や資源の管理を行う。そのため複数のタスクが並行に処理される。また、タスクを処理すること自体が目的ではなく、処理結果を使って他の機器を制御することが目的であり、処理時間に対する要求が厳しく、処理が遅れるとシステムの制御が不能になる場合がある。

このように組み込みシステムのOSには、リアルタイム性が要求される。リアルタイム性とは、「応答時間が十分速いこと、あるいは応答速度の最悪の値が予測できること」と定義される[1]。

一般に、リアルタイムOSに求められる性能は、(1)割り込み、タスク切り替え及びシステムコール中におけるクリティカルセクションが短いこと、(2)OSの処理による最悪値の時間が保証されていることである。

## 3 ITRONの概要

ITRONは、産業用の組み込みシステムに使用することを目的として作られたリアルタイムOSであり、汎用16ビットマイクロプロセッサを対象としている。

ITRONの特徴は、(1)いろいろな種類のプロセッサに実装できるOSであるにもかかわらず、その処理性能を高くするために、必要以上の標準化や仮想化を行っていないこと、(2)豊富な機能を持つOSであるにもかかわらず、適応化を考慮し実際のオブジェクトコードをコンパクトにできることである。

$\mu$ITRON仕様は、シングルチップでもリアルタイムOSが利用できるように、ITRON仕様を改良したものであり、機能がわかりやすく、適応化しやすいように変更されている。

$\mu$ITRONの応用分野は、ITRONと基本的には同じであるが、ITRONが汎用16ビットマイクロプロセッサを対象としているのに対し、$\mu$ITRONでは、シングルチップ・プロセッサや8ビットマイクロプロセッサを対象としている。したがって、ITRONが、大規模、高機能で高価格なものに向いているのに対して、$\mu$ITRONは小規模で高機能、低価格が要求される装置分野に向いている。

$\mu$ITRON仕様の主な特徴は次のとおりである[2]。

- プロセッサの仮想化をしない
- 豊富な機能を持つ
- 高速な応答が可能である
- 適応化を考慮している

従来、OSはソフトウェアにより実現されているが、組み込みシステムのようなリアルタイム性が要求されるような場合、ソフトウェアだけではOSの

性能を向上させるには限界がある。そこで、次の設計方針によりOSのハードウェア化を実現する。

1. μITRONの仕様に準拠したOSをハードウェア化する
2. ハードウェアよる処理の高速性を生かしOSの処理時間を短縮する
3. リソースの処理及びタスクのスケジューリングを並列に実行することにより処理時間を短縮する

これによりリアルタイムOSのシステムコール処理時間、タスクの選択時間及び割り込み禁止時間の短縮を達成されると考えられる。

また、以上の機能は、大規模なFPGAまたは、ASIC技術を用いればを容易にハードウェアで実現できる。

次節では、STRON-1のアーキテクチャについて述べる。

# 4 STRON-1のアーキテクチャ

今回、ターゲットとしているFPGAの規模は、ゲート数万程度である。そのため実現できるタスク数及びOSの機能に制限があるので、STRON-1の機能は、単純なアプリケーションを想定した評価実験が行えるように必要最低限なものにとどめた。

以下、STRON-1で実現したμITRONの機能及びSTRON-1のアーキクチャについて述べる。

## 4.1 STRON-1用のOS

STRON-1は、汎用CPUとの間でインターフェースを行うソフトウェアが必要である。STRON-1チップとインターフェースを行うソフトウェア部分を総称してSOS（以下、SOS:Silicon tron OS）と呼ぶことにする。

SOSは、タスクと割り込みハンドラを同じに扱い、割り込みハンドラの起動時にはOSが介入する。すなわち、割り込みによるハンドラの起動はSOSの管理のもとでコンテキストの切り替えが行われる。表1にSOSの概略仕様を示す。

リアルタイムOSとして使用するμITRONの機能には、次のようなものがある。

- タスク管理機能
- タスク付属同期機能

表1: STRON-1の仕様

| 項目 | 条件 |
|---|---|
| 最大タスク数 | 5 |
| タスクの優先度 | 1〜7 |
| 最大イベントフラグ数 | 5 |
| 最大セマフォ数 | 5 |
| セマフォの形式 | 計数型 |

- 同期通信機能
- 割り込み管理機能
- メモリプール管理機能
- 時間管理・タイマハンドラ機能
- システム管理機能

STRON-1で実現するμITRONの機能及びシステムコールは次のとおりである。

1. タスク管理機能
   sta_tsk, rot_rdq, chg_pri, get_tid, tsk_sts
2. タスク付属同期機能
   slp_tsk, wai_tsk, wup_tsk, can_tsk
3. 同期、通信機能（イベントフラグ、セマフォ）
   set_flg, clr_flg, wai_flg, flg_sts, sig_sem
   wai_sem, sem_sts
4. 割り込み管理機能
   def_int, ret_int

これら以外の機能の実現は、今後の課題である。

## 4.2 汎用CPUとのインターフェース

STRON-1は汎用CPUに外部接続することによりリアルタイムOSの機能を提供する。評価用ボードのプロセッサは、モトローラ社のMC68000を使用した。図1にMC68000とSTRON-1のインタフェース信号についてブロックレベルで示す[3]。

STRON-1は、入出力コントローラ、イベントフラグ管理、セマフォ管理、タイマ管理、タスク管理及びスケジューラの5つのサブモジュールから構成されている。次節では、各サブモジュールの詳細について述べる。

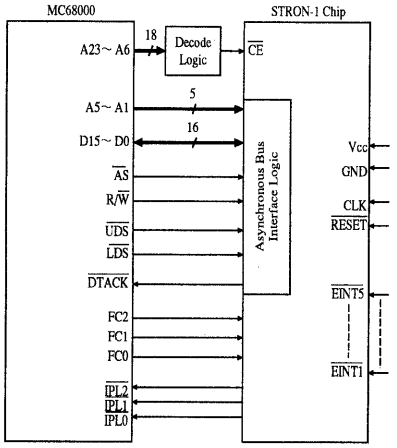


図 1: MC68000 と STRON-1 のインタフェース

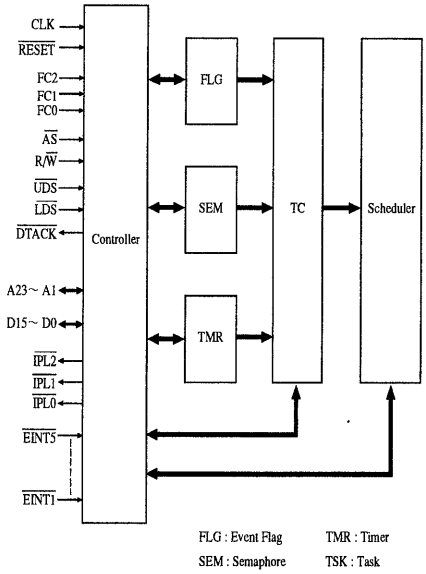


図 2: STRON-1 のブロック図

## 4.3 入出力コントローラ

入出力コントローラは、MC68000 とのインターフェース、外部割り込み信号の管理、タスクの切り替え管理及び各サブモジュールが動作するために必要なデータの管理を行う。

サブモジュールが動作するために必要なデータは、主に機能コード及びレジスタ ID などである。図 2に STRON-1 のブロック図を示す。

タスクの切り替えが起こる場合、STRON-1 はスタックポインタだけを格納し、タスクのコンテキストはメモリ上に格納し、コンテキストの退避及び復帰処理はソフトウェアで行われる。

タスク切り替えの流れは以下のとおりである。

1. STRON-1 は MC68000 に対して割り込み信号を入力しタスク切り替え用のディスパチャを起動する
2. MC68000 は、現在処理しているタスクのコンテキストをメモリ上に退避し、スタックポインタを STRON-1 に転送する
3. STRON-1 は、MC68000 から転送されたスタックポインタを格納し、次に処理するタスクのスタックポインタに切り替える
4. MC68000 は、STRON-1 から転送されたスタックポインタを用いてコンテキストを復帰する

## 4.4 イベントフラグ管理モジュール

イベントフラグ管理モジュールは、5 つの 1 ビットのフラグレジスタとその制御回路から構成される。図 3にイベントフラグ管理モジュールの構成を示す。

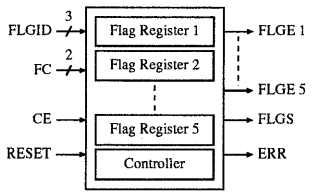


図 3: イベントフラグ管理モジュールの構成

イベントフラグ管理モジュールへの入力はフラグレジスタ ID(FLGID)、機能コード (FC) 及び制御信号 (RESET, CE)、出力は各フラグレジスタの状態 (FLGE1～FLGE5)、入力に対するステータス情報 (FLGS) 及び制御信号 (ERR) である。

機能コードは、set、reset 及び reference の 3 種類がある。与えられた機能コードが set または、reset の場合、FLGID により選択されたフラグレジスタにそれぞれ、“1” または、“0” がセットされる。

また、与えられた機能コードがreferenceの場合、FLGIDにより選択されたフラグレジスタの現在の状態がFLGSに出力される。

各フラグレジスタの信号はタスク管理モジュールに入力され、フラグ待ちタスクの状態を制御するために用いられる。

### 4.5 セマフォ管理モジュール

セマフォ管理モジュールは、5つの3ビットのセマフォカウンタ、FIFO及びその制御回路で構成される。FIFOは5bit3段のシフトレジスタで実現される。図4にセマフォ管理モジュールの構成を示す。

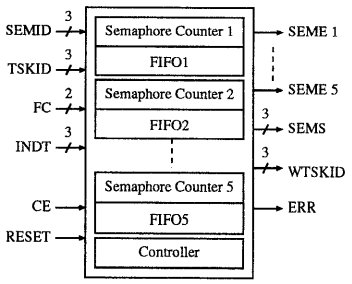


図 4: セマフォ管理モジュールの構成

セマフォ管理モジュールへの入力はセマフォカウンタ ID(SEMID)、機能コード (FC)、タスクID(TSKID)、初期化データ (INDT) 及び制御信号 (RESET, CE)、出力は各セマフォカウンタの状態 (SEME1～SEME5)、入力に対するステータス情報 (SEMS)、待ちタスクID(WTSKID) 及び制御信号 (ERR) である。

機能コードは、up、down、initial及びreferenceの3種類がある。機能コードがinitialの場合、SEMIDにより選択されたセマフォカウンタに対して資源の初期データがセットされる。

また、機能コードがupまたは、downの場合、SEMIDにより選択されたセマフォカウンタは、“+1”または、“-1”される。

資源がない状態で機能コードがdownの場合、タスクIDが各セマフォのFIFOに格納され、タスクはセマフォ待ちになる。セマフォ待ちタスクがある状態で機能コードがupの場合、セマフォカウンタは、“+1”されず待ちタスクIDがWTSKIDに出力される。

さらに、機能コードがreferenceの場合、SEMIDにより選択されたセマフォカウンタの現在の状態がSEMSに出力される。

セマフォカウンタの信号は、タスク管理モジュールに入力され、セマフォ待ちタスクの状態を制御するために用いられる。

### 4.6 タイマ管理モジュール

タイマ管理モジュールは、5つの16ビットのタイマとその制御回路から構成される。図5にタイマ管理モジュールの構成を示す。

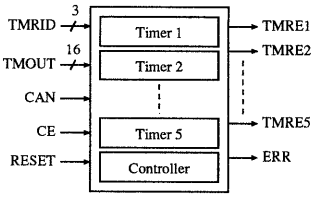


図 5: タイマ管理モジュールの構成

タイマ管理モジュールへの入力はタイマID (TMRID)、タイマにセットする値 (TMOUT)、タイマ動作のキャンセル (CAN) 及び制御信号 (RESET, CE)、出力は各タイマの状態 (TMRE1～TMR5) 及び制御信号 (ERR) である。

コントローラでこのモジュールが選択された場合、TMRIDにより選択されたタイマに対してTMOUTがタイマにセットされタイマが動作を開始し、タスクは時間待ちになる。

タイマの信号はタスク管理モジュールに入力され、時間待ちタスクの状態を制御するために用いられる。

### 4.7 タスク管理モジュール

タスク管理モジュールは、5つのタスク状態レジスタとその制御回路から構成される。

タスク管理モジュールは、イベントフラグ (FLGE)、セマフォ (SEME)、タイマ (TMR) の各管理モジュール、外部割り込装置 (INT) 及びコントローラ (RD/$\overline{\text{WA}}$) から信号を受け取り、主にタスクの状態を管理する。図6にタスク管理モジュールの構成を示す。

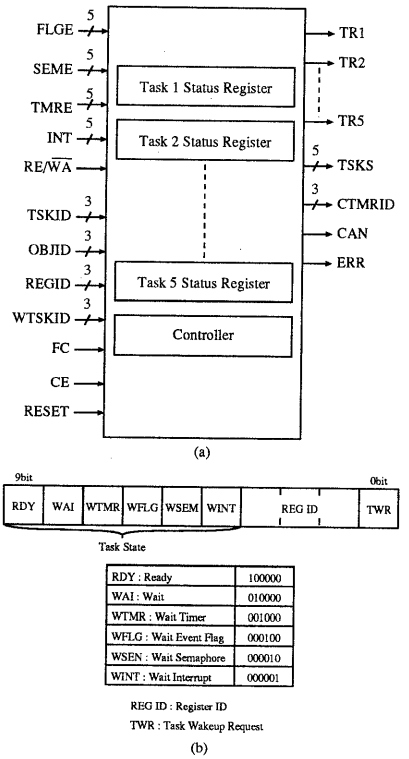


図 6: タスク管理モジュールの構成

他の入力はタスク状態レジスタ ID(TSKID)、オブジェクト ID(OBJID)、レジスタ ID(REGID)、待ちタスク ID(WTSKID)、機能コード (FC) 及び制御信号 (RESET, CE)、出力は各タスク状態レジスタの状態 (TR1～TR5)、入力に対するステータス情報 (TSKS)、キャンセルとなるタイマ ID(CTMRID)、キャンセル (CAN) 及び制御信号 (ERR) である。

機能コードには、status、wakeup、cancel 及び reference の 4 種類がある。

以下では、タスク管理モジュールの主な処理である status について述べる。

機能コードが status の場合、OBJID によって待ちの対象となるオブジェクト[1]を選択し、REGID により選択されたレジスタの状態を用いタスクの状態を変化させる。

対象となるオブジェクトのレジスタの信号が “0” の場合にはタスクは資源待ち状態、“1” の場合には READY 状態になる。

タスク管理モジュールは、10 ビットのタスク状態レジスタから構成され、上位 6 ビットでタスク状態を識別し、次の REG ID で各待ち資源のレジスタ ID を識別する。また、TWR はタスクの起床要求の有無のために用いられる。

タスク管理モジュールの信号は、スケジューラに入力され、タスク選択を行うために用いられる。

## 4.8 スケジューラ

スケジューラは、5 つのタスク状態レジスタ、比較器、エンコーダとその制御回路から構成される。

スケジューラは、タスク優先度の変更及びタスク選択を行う。

図 7にスケジューラの構成を示す。

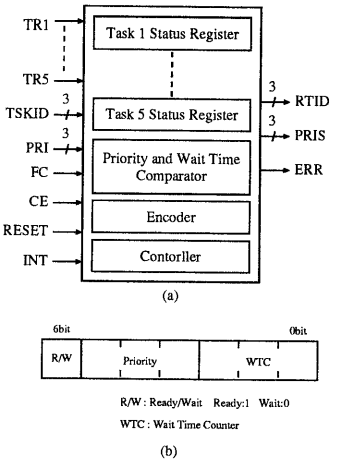


図 7: スケジューラの構成

スケジューラへの入力は、タスク管理ブロックからの信号、タスク ID(TSKID)、優先度 (PRI)、機能コード (FC) 及び制御信号 (RESET, CE)、出力は次に処理されるタスク ID(RTID)、入力に対するステータス情報 (PRIS)、制御信号 (ERR) である。

機能コードには、set 及び reference の 2 種類がある。

set または、reference の場合、TSKID により選択されたタスク状態レジスタの優先度を PRI に変更または、PRIS に出力する。

[1]ITRON 仕様の OS ではタスク、イベントフラグセマフォなどのシステムコールの操作対象となるものの総称をオブジェクトと呼ぶ。

スケジューラは、7ビットのタスク状態レジスタから構成され、上位1ビットはタスクの状態を識別し、このビットが"1"のときREADY状態、"0"のときWAIT状態とする。また、次のPriorityはタスクの優先度を管理するために、WTCはタスクの待ち時間を管理するために用いる。

タスクの選択方法は、トーナメント方式でタスク状態レジスタの値を比較し、タスク状態レジスタの値が最も大きいタスクを選択する。

## 5 STRON-1の設計

STRON-1は以下に示す動作条件を定め設計を行った。

- MC68000と同じクロック周波数(10MHz)で動作する
- MC68000の1命令の処理時間内でSTRON-1の動作が完了する

STRON-1の機能記述は、NTTで開発された高位合成系PARTHENON[4]のハードウェア記述言語SFL(Structured Function description Language)を使用した。

PARTHENONは動作シミュレータ、論理回路シンセサイザ、オプティマイザから構成されるVLSI設計システムである。本研究では、PARTHENONを用いて機能検証及びタイミング検証を行った。

表2にPARTHENONで論理合成時に用いた設計条件[5]を示す。

表2: 設計条件

| 項目 | 条件 |
|---|---|
| 動作周波数 | 10MHz |
| 接続負荷容量 | 0.05pF |
| セルライブライリ | VSC470:0.8$\mu$CMOS |

表3にSTRON-1の各サブモジュールゲート数及び設計結果を示す。

## 6 STRON-1の性能評価

STRON-1の性能評価は、リアルタイムOSの基本性能を算出することで行った。また、STRON-1との性能比較を行うためにSTRON-1と同じ機能をソフトウェアで実現した評価用のOS(以下、EOS:Evaluation OS)を作成した。

表3: 各モジュールのゲート数及び設計結果

| モジュール名 | ゲート数 |
|---|---|
| イベントフラグ | 229 |
| セマフォ | 2,913 |
| タイマ | 2,392 |
| タスク | 1,910 |
| スケジューラ | 2,479 |
| コントローラ | 6,140 |
| 合計 | 16,063 |
| 面積 | 3334.88k($\mu$m)$^2$ |
| 最大遅延時間 | 89.6ns |

表4: SOSのの各処理に必要なクロック数

| 項目 | クロック数 |
|---|---|
| システムコール処理 | 150~217 |
| タスク切り替え処理 | 371 |

EOSのカーネル記述言語は、C言語及び一部にアセンブリ言語、一方、SOSのカーネル記述言語は、アセンブリ言語を用いた。

今回、リアルタイムOSの基本性能を示す指標として次の3つを用いた[6]。

1. 割り込み禁止時間
   「割り込み禁止時間」は、プロセッサが外部割り込みに対するサービスを禁止している時間
2. タスク切り替え時間
   「タスク切り替え時間」は、あるタスクから別のタスクへ実行を切り替えるために必要な時間
3. システムコール処理時間
   「システムコール処理時間」は、カーネルがシステムコールを実行するのに必要な時間

表4にSOSの各処理に必要なクロック数を示す。

表5にEOSとSOSの各項目の最大時間を示す。また、MC68000上に実現された市販の$\mu$ITRON(以下、UDEOS/m00)の性能を共に示す。ただし、このUSEOS/m00の動作周波数は16MHzである。表5のSOSの値は、動作周波数と各処理に必要なクロック数をもとに算出し、動作周波数10MHz、クロッ