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

2005－ARC－165（11）
2005／12／1

マルチプロセッサ向けμITRON OS の開発

鈴木 貴久† 　上方 輝彦†

†富士通研究所 〒211-8588 神奈川県川崎市中原区上小田中 4-1-4

E-mail: †suzuki.takahisa@jp.fujitsu.com, †kamigata.teruhi@jp.fujitsu.com

あらまし 我々は，組み込み分野でのマルチプロセッサ向けのソフトウェア実行環境を実現することを目的として，マルチプロセッサ向けμITRON OS を開発した．

この OS は，μITRON 仕様に準拠した OS と，マルチプロセッサのソフトウェア実行環境に必要な機能を提供するマルチプロセッサ向け拡張モジュールから構成され，分散メモリ型マルチプロセッサ，共有メモリ型マルチプロセッサの両方に対応することが可能である．

キーワード 組み込み，マルチプロセッサ，実行環境，OS，μITRON，分散メモリ

μITRON based Operating system for Multiprocessor

Takahisa SUZUKI, Teruhiko KAMIGATA

†Fujitsu Laboratories 4-1-1 Kamikodanaka, Nakahara-ku, Kawasaki, Kanagawa 211-8588 Japan

E-mail: †suzuki.takahisa@jp.fujitsu.com, †kamigata.teruhi@jp.fujitsu.com

abstract To provide a software execution environment for embedded multiprocessor systems, we have developed a μITRON based operating system for multiprocessor. This operating system consists of a operating system based on μITRON specification and a multiprocessor extension module that provide functions to execute multiprocessor software. This operating system provides software execution environment for both shared memory multiprocessor system and distributed memory multiprocessor system.

keyword embedded, multiprocessor, operating system, μITRON, distributed memory

1. はじめに

近年，組み込みプロセッサの分野においても高性能で低消費電力なプロセッサを実現するための技術としてマルチプロセッサ技術が注目されている．

マルチプロセッサシステムでは複数のプロセッサで分散して処理を行うことで，システム全体の処理能力を向上することができる．このためには，ハードウェアだけでなく，分散して処理を実行するためのソフトウェア実行環境が必要になる．

マルチプロセッサ向けの実行環境としては対称型マルチプロセッサ(SMP:Symmetric MultiProcessor)向けの OS が知られている．

しかしながら，SMP 向けの OS は，高度な演算や大量の処理を行う大規模なサーバー向けに開発された OS であり，高いリアルタイム応答性と少ないメモリ使用量を要求される組み込みシステムには適していない．

一方で，組み込みシステム向けのマルチプロセッサ OS としては μITRON4.0[1] 仕様に準拠した“TOPPERS/FDMP”カーネル[2]が知られているが，μITRON はタスク間でメモリ空間を共有しており，タスク間で異なるメモリ空間を使用することは前提として

いないため，全てのプロセッサがメモリ空間を共有する共有メモリ型のマルチプロセッサシステムを前提とすることになる．

しかしながら，共有メモリ型のマルチプロセッサシステムではプロセッサの数が増えるとメモリに対するアクセスが集中してメモリアクセスレイテンシが増大するという問題がある[3]．

我々は，μITRON 仕様のシングルプロセッサ向けOSに，マルチプロセッサ向けOSとして必要な機能と，分散メモリ型のマルチプロセッサシステムに対応するために必要な機能を提供する“マルチプロセッサ向け拡張モジュール”を追加することで，分散メモリ型のマルチプロセッサシステムにも対応したマルチプロセッサ向けのμITRON OS を開発した．

2. 組み込みソフトウェアのアーキテクチャ

組み込みソフトウェアは，図 1 のように複数のタスクから構成されている．それぞれのタスクは，大抵の場合，動画エンコーダや音声エンコーダのように，一つの大きな機能を提供し，独立して動作することが可能である．

このため，複数のプロセッサで異なるタスクを並列に処理することが出来る．

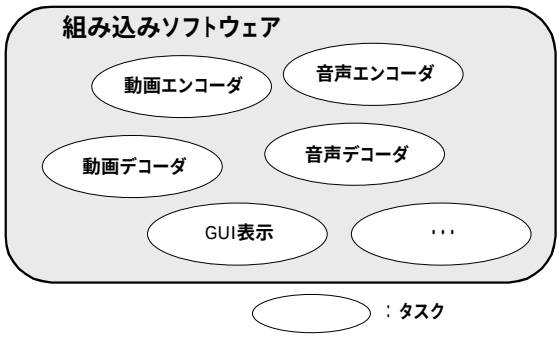

図 1：組み込みソフトウェアの構造

3. マルチプロセッサ向け実行環境

開発したマルチプロセッサ向けμITRON OS を図 2 に示す．各プロセッサはそれぞれシングルプロセッサ向けのμITRON OS，マルチプロセッサ向け拡張モジュール，独立したメモリ空間を持っており，その上で各プロセッサに割り当てられたタスクが動作している．

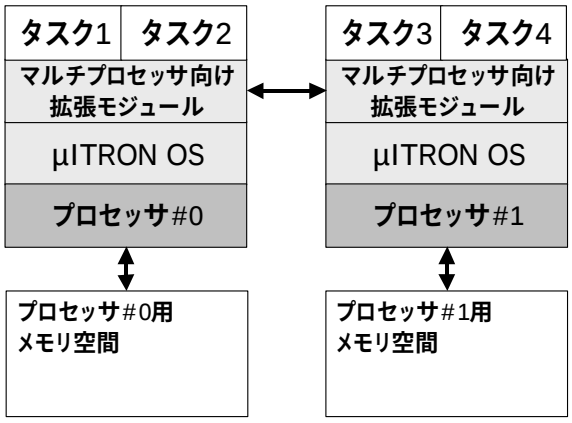

図 2：実行環境

物理的にメモリを共有している場合でも，このようにプロセッサ毎にメモリ空間を独立させることで，不用意に他プロセッサによりデータが書き換えられることが無くなり，バグの発生を抑えることが出来る．

プロセッサ間でメモリ空間を共有させたい場合は，図 3 に示すように共有メモリ空間を持つことも出来る．

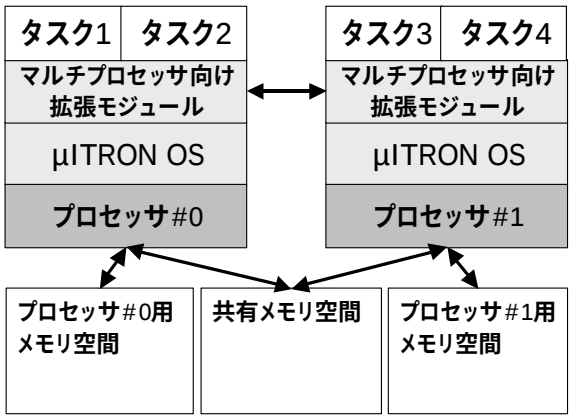

図 3：メモリを共有した実行環境

マルチプロセッサ向け拡張モジュールが無い場合は，各プロセッサはシングルプロセッサの時とまったく同じ環境になる．

マルチプロセッサ向け拡張モジュールは，シングルプロセッサの実行環境に対して以下の機能を提供することで，マルチプロセッサ向けの実行環境を実現する．

- プロセッサを跨いだサービスコール
- プロセッサを跨いだデータ転送

3.1. プロセッサを跨いだサービスコール

プロセッサを跨いだサービスコールでは，異なるプ

ロセッサに対してμITRON のサービスコールを発行する機能を提供する．

このときの動作シーケンスを図 4 に示す．プロセッサを跨いだサービスコールを行う場合，タスクはマルチプロセッサ向け拡張モジュールに対してプロセッサを跨いだサービスコールを要求する．マルチプロセッサ向け拡張モジュールは，タスクより指定された要求先のプロセッサのマルチプロセッサ向け拡張モジュールを介して，要求先プロセッサのμITRON OS に対してサービスコールの実行を要求する．サービスコールの実行が完了したら，マルチプロセッサ向け拡張モジュールを介してサービスコールの実行結果を通知する．

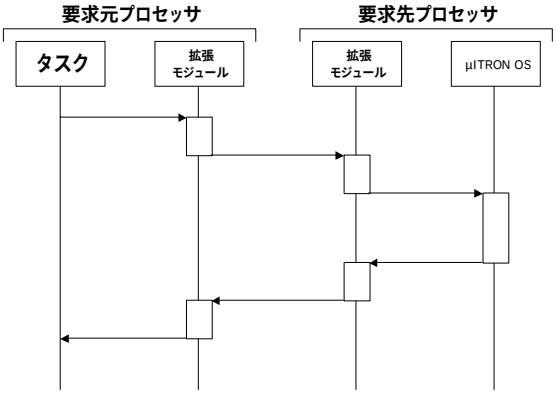

図 4：プロセッサを跨いだサービスコール

マルチプロセッサ向け拡張モジュールが提供するプロセッサを跨いだサービスコールを表 1 に示す．

プロセッサを跨いだサービスコールの API は要求するサービスコール名の先頭に“ RMP_ ”をつけた名前になる．また，引数としては要求するサービルコールの引数と要求先のプロセッサ ID を指定する．

表 1：プロセッサを跨いだサービスコール

プロセッサを跨いだサービスコール	**タスク管理**	RMP_act_tsk RMP_sta_tsk RMP_ter_tsk RMP_ref_tsk RMP_ref_tst
	タスク間同期･通信	RMP_sig_sem RMP_set_flg RMP_clr_flg RMP_ref_flg RMP_snd_mbx RMP_ref_mbx RMP_snd_dtq RMP_psnd_dtq RMP_tsnd_dtq RMP_ref_dtq
	メモリプール	RMP_get_mpf RMP_ref_mpf RMP_rel_mpf

3.2. プロセッサを跨いだデータ転送

プロセッサを跨いだデータ転送では，異なるメモリ空間の間でデータを転送する機能を提供する．

マルチプロセッサ向け拡張モジュールは，プロセッサを跨いだデータ転送が要求されると，指定された送信元アドレス，送信先プロセッサ，送信先アドレス，データサイズに基づいて，送信先から送信元へデータを転送する．

また，通常データ転送を行う場合は，データ転送が終了したら送信先のプロセッサに対してデータ転送が終了したことを通知する必要がある．マルチプロセッサ向け拡張モジュールは，データ転送が終了したら自動的に送信先プロセッサに対して，プロセッサを跨ったサービスコールを要求する機能も提供する．

マルチプロセッサ向け拡張モジュールが提供するプロセッサを跨いだデータ転送を表 2 に示す．

表 2：プロセッサを跨いだデータ転送

プロセッサを跨いだデータ転送	**転送のみ**	RMP_Move
	サービスコール付	RMP_Move_sig_sem RMP_Move_set_flg RMP_Move_snd_mbx RMP_Move_snd_dtq RMP_Move_psnd_dtq RMP_Move_tsnd_dtq

3.3. 実装

プロセッサを跨いだサービスコールでは，サービスコールの要求先プロセッサと要求元プロセッサの間で

サービスコールの引数・戻り値として数ワード程度のデータを送受信する必要がある．また，プロセッサを跨いだデータ転送では送信元プロセッサのメモリ空間から送信先プロセッサのメモリ空間へデータを転送する必要がある．

これらのデータ転送の実現方法は対象とするプロセッサにより異なるため，マルチプロセッサ向け拡張モジュールの内部は，図 5 のように対象のプロセッサにより異なる部分(ハードウェア依存層)と，対象のプロセッサに依存しない共通の部分(ハードウェア非依存層)に分かれる．ハードウェア依存層はプロセッサの間での小量のデータ転送，異なるメモリ空間の間でのデータ転送などのハードウェアに依存する処理を行う．ハードウェア依存層を利用することで，ハードウェア非依存層はハードウェアに依存しない処理のみを行うことが出来る．

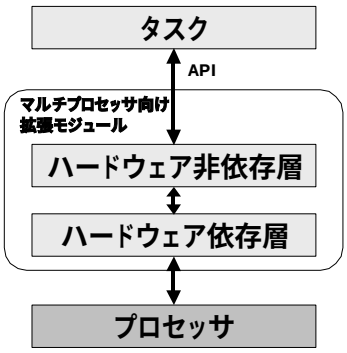

図 5　拡張モジュール内部構造

4. マルチプロセッサ向けプログラム開発

本章では MPEG2 複合化ソフトウェアを例として，マルチプロセッサ向けμITRON OS を利用した，マルチプロセッサ向けプログラム開発について説明する．

4.1. タスク分割

マルチプロセッサ向けμITRON OS を利用して MPEG2 複合化プログラムを並列処理するためには，MPEG2 複合化プログラムをタスクに分割する必要がある．

MPEG2 複合化プログラムでは，MPEG2 が規定するスライス層の間には依存が無く並列に処理できることが知られている[4]．そこで，図 6 のように複数のスライス処理をまとめて一つのタスクとする．このように，一つのタスクを大きくすることで，タスク間の通信を少なくし，処理の効率を上げることが出来る．また，スライス処理の前後に，圧縮されたデータの入力を行う前処理部と，複合化した動画の表示を行う後処理部があるが，これらもまとめて一つのタスクとする．

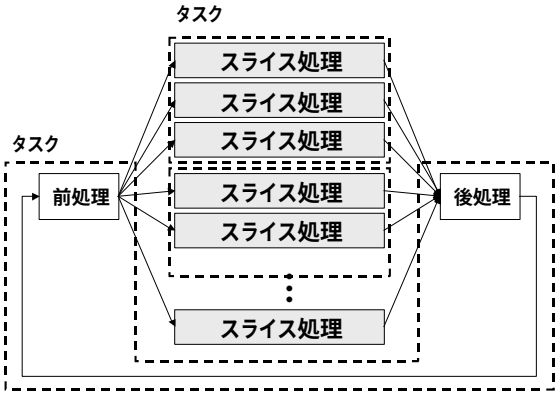

図 6　タスク分割

4.2. タスク間通信の挿入

分割したタスクの間で同期を取る必要がある場合は，μITRON のタスク間通信のサービスコールを挿入する．

ここでは，前/後処理を行うタスクで前処理が終了すると，スライス処理を行う全てのタスクに対して処理の開始を通知し，スライスの処理が終了すると前/後処理を行うタスクに対して処理の終了を通知する必要がある．ここで，サービスコールを利用して終了の通知を行う．

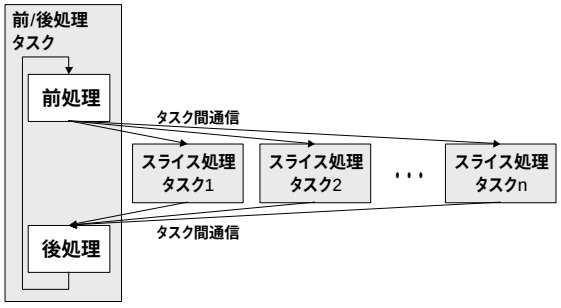

図 7　タスク間通信の挿入

4.3. プロセッサへの割り当て

分割したタスクをプロセッサへ割り当てる．このとき，効率よく並列処理を行うためには以下の点が重要である．

- 各プロセッサの処理量を均一にすること
- プロセッサ間の通信を少なくすること

そこで, 図 8 に示すようにスライス処理タスクをプロセッサの台数分作成し，各プロセッサに一つずつ割り当てる. 前/後処理のタスクはスライス処理とは同時に実行出来ないため, 適当なプロセッサに割り当てる.

また，プロセッサ間の通信を少なくするため，入出力のデータは共有メモリ領域に置くことにする.

プロセッサへのタスクの割り当てが完了したら，異なるプロセッサ間でのタスク間通信のサービスコールを, プロセッサを跨いだサービスコールに置き換える.

これにより，マルチプロセッサでの並列処理が可能になる.

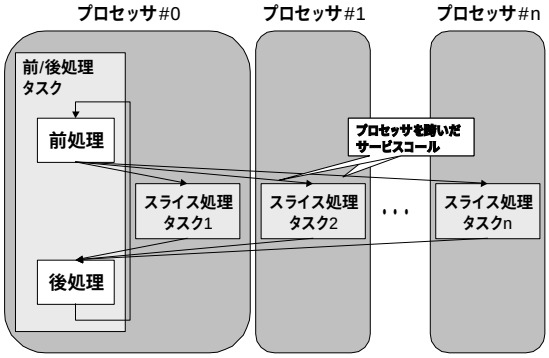

図 8　マルチプロセッサ化

5. まとめ

本稿では，マルチプロセッサ向けのソフトウェア実行環境として，μITRON 仕様をベースにしたマルチプロセッサ向けμITRON OS について報告した.

この実行環境は，マルチプロセッサ向け OS としての機能を提供し，分散メモリ型マルチプロセッサ，共有メモリ型マルチプロセッサどちらにも対応することが出来る.

また, MPEG2 を例としてこの OS を利用したマルチプロセッサ向けプログラムの開発方法についても報告を行った.

文献

[1] http://www.sakamura-lab.org/TRON/ITRON/home-e.html

[2] http://www.toppers.jp/en/index.html

[3] Ehab S. Al-Shaer ,“Distributed Memory Management:Design Issues and Future Trends”

[4]T. Kodaka et al., “Parallel Processing using Data Localization for MPEG-2 Encoding on OSCAR Chip Multiprocessor,” *Proc. Of IWIA’04, Jan. 2004.*