機能分散マルチプロセッサ用リアルタイムOS TOPPERS/FDMPカーネル

本田　晋也

名古屋大学 情報科学研究科

honda@ertl.jp

アジェンダ

マルチプロセッサ用のリアルタイムOSに対する取り組み，
特にTOPPERS/FDMPカーネルについて紹介

- TOPPERSプロジェクトにおけるマルチプロセッサへの取り組み
  - 機能分散マルチプロセッサ用リアルタイムOS
  - 対称型マルチプロセッサ用リアルタイムOS
- TOPPERS/FDMPカーネル
  - 仕様
  - 実装
- シングルプロセッサ用のアプリケーションの移植（排他制御）

TOPPERSプロジェクトにおける
マルチプロセッサへの取り組み

組込みシステムにおけるマルチプロセッサの利用

従来のマルチプロセッサシステムの利用

- 従来から複数のプロセッサを使用したシステムは存在
  - 性能が必要な一部のシステム
  - 部品に含まれている場合 (例:音源チップ)
  - 信頼性要件が異なるサブシステム (例:機械制御とGUI)
  - ➡ コスト等の関係で広く用いられることはなかった

オンチップマルチプロセッサ (マルチコア型プロセッサ)

- 複数のプロセッサコアを1チップに集積
  - MPCore, MeP, UniPhier, FR1000, MP211, Cell
- 低消費電力化で高性能を実現するためには、高速なシングルプロセッサ構成より、低速なマルチプロセッサ構成が有利
- 消費電力，発熱の関係でこれ以上クロックを引き上げられないため,複数のプロセッサを用いる

マルチプロセッサ向けOSの必要性

これまでマルチプロセッサ向けソフトウェア開発

- 各プロセッサに独立にシングルプロセッサ用のリアルタイムOSを載せる（プロセッサによってはOSレスも）
- プロセッサ間の同期・通信は，（OSではなく）アプリケーションソフトウェアで実現

OSによるサポートの必要性

- アプリケーション毎にプロセッサ間の同期・通信を実現する必要があるため，開発工数が増える
- ある処理を別のプロセッサに移そうとすると，同期・通信部分のプログラムの作り直しが必要
- 処理を動的にプロセッサに割り当てる機能が必要

マルチプロセッサのタイプ

OSでサポートするという観点からは，
幾つかのタイプに分類できる

密結合マルチプロセッサ

- 各プロセッサからアクセスできる共有メモリを持つ
- 構成（使用方法）によりさらに2種類に分類できる．
  - 対称型マルチプロセッサ（SMP，SMT）
  - 機能分散/非対称型マルチプロセッサ（FDMP，AMP）

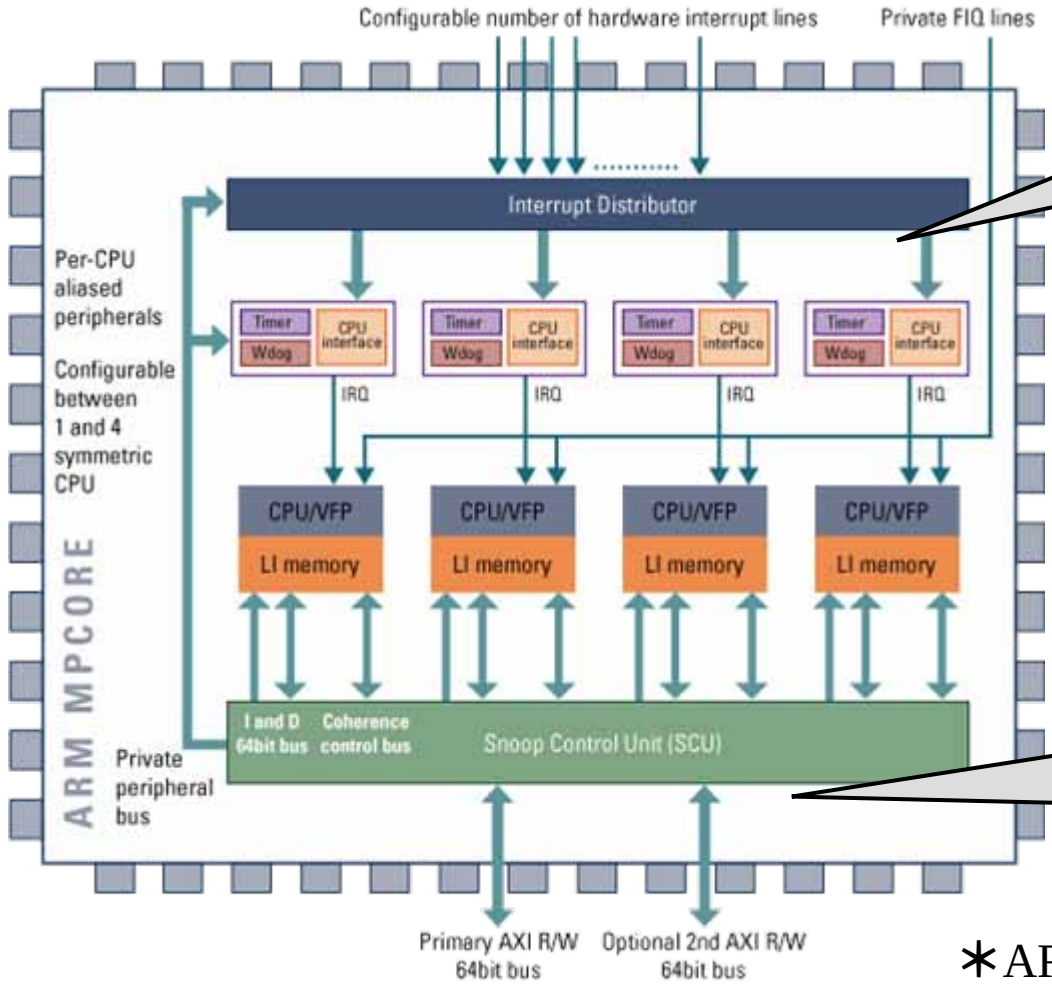
→ それぞれ，ハードウェアの構成とは独立．両方の形態を実現可能なものや，片方の形態しか実現できないものもある

疎結合マルチプロセッサ

- 共有メモリを持たない（分散システム）
- 組込みコンポーネント仕様の枠組みで扱う

対称型マルチプロセッサ向けハードウェアの例

- ARM MPCore ：1〜4個のマルチプロセッサ構成



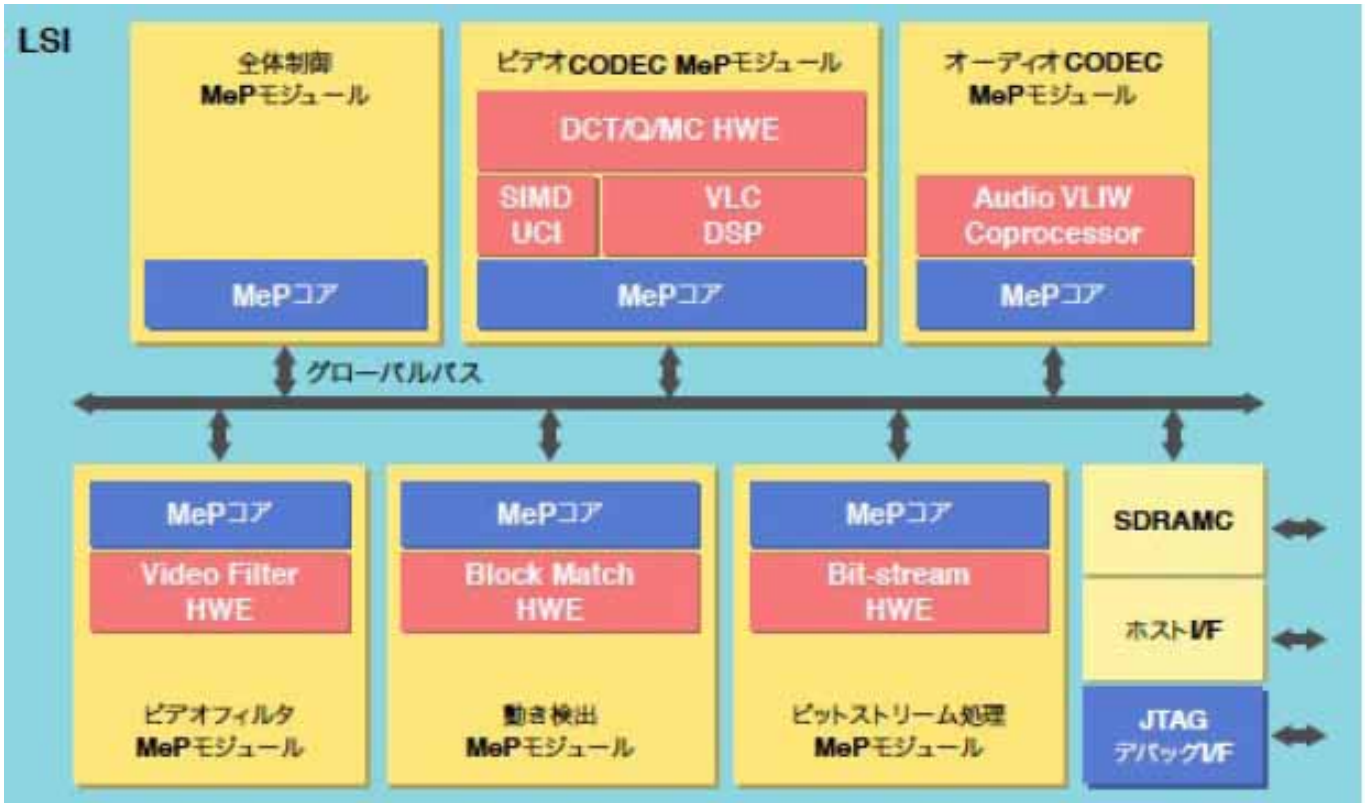
割込み分配回路

キャッシュ・コヒーレンシ制御

＊ARM ウェブサイトより

機能分散マルチプロセッサ向けハードウェアの例：ASIC

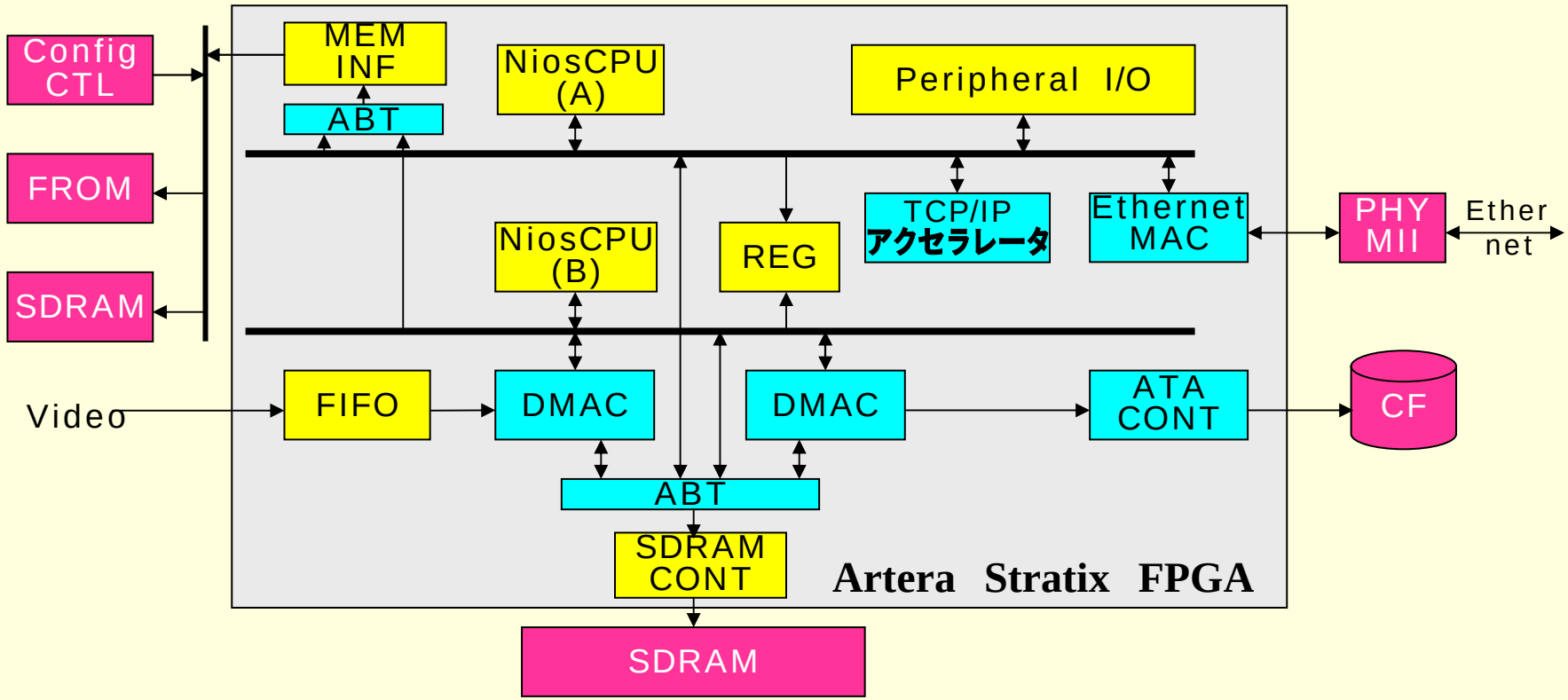
- 東芝MeP(Media embedded Processor)プロセッサコアを用いたMPEG2コーデックLSIの構成例



＊東芝半導体 製品カタログ「MeP(Media embedded Processor)概説」より

機能分散マルチプロセッサ向けハードウェアの例：FPGA

- ALTERA　Niosプロセッサコアを用いた
  画像ストレージ機能とネットワーク機能（Ethernet）の構成例



- 開発元：株式会社ワイ・デー・ケーYDKテクノロジーズ

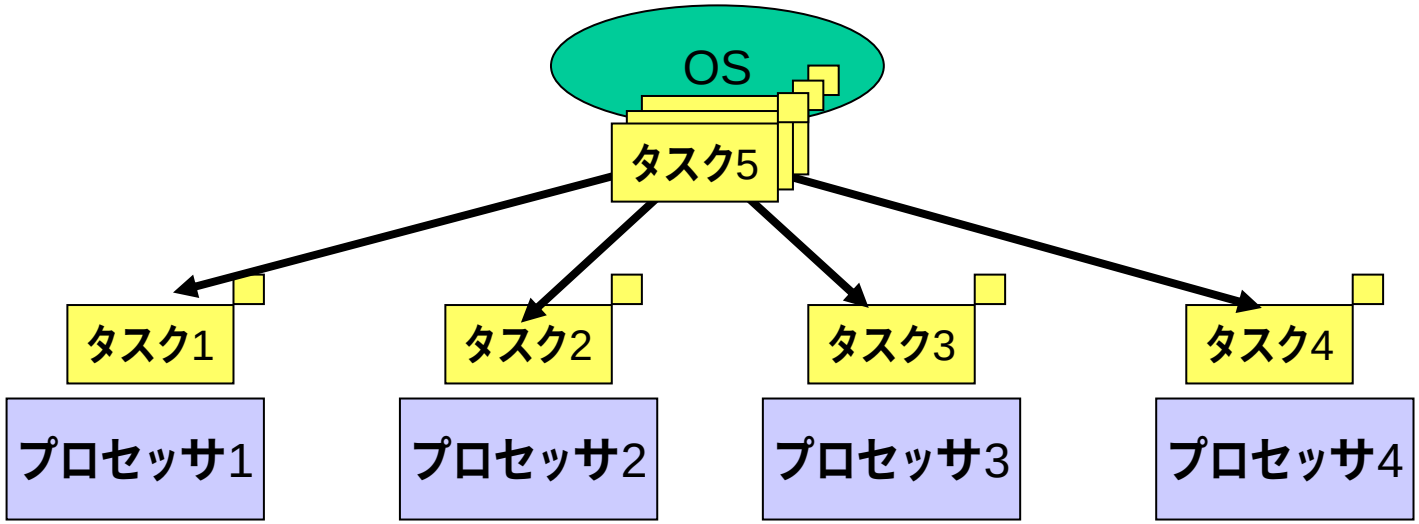
対称型マルチプロセッサ

各プロセッサから全てのリソースアクセスすることができ，
どの処理をどのプロセッサでも実行可能なモデル

- 動的な負荷分散（スループット）を目的としており，汎用システムで一般的なモデル
- ユーザーはどのアプリがどのプロセッサで実行されているか意識する必要がない
- システムの機能が固定されていない（汎用システムに近い）場合に有効

対称型マルチプロセッサ：OSの機能

- 各プロセッサに負荷（タスク）を動的に分散する
  - タスクがどのプロセッサで実行されるかは，実行されるまで分からない
- 割込みハンドラに関しては，プロセッサのアーキテクチャに依存して実装毎に扱いは異なる
  - 特定のプロセッサで実行する
  - 優先度の低いタスクを実行中のプロセッサで受け付ける



対称型マルチプロセッサ：メリットとデメリット

メリット

- 動的な負荷分散が可能で，システムのスループットを向上させることが可能
- 単一のチップを複数の製品に展開可能
- ソフトウェアの作りこみなしに性能を得られる可能性

デメリット

- 全てのリソースを各プロセッサからアクセス可能とするため，高速で複雑なバスやコヒーレントキャッシュが必要不可欠
- ハードウェアのコストや消費電力は大きくなる傾向になる
- 動的な負荷分散を行うため，各タスクのリアルタイム性の保証が困難になる
- 割り込みハンドラも動的にするとさらに困難に

機能分散/非対称型マルチプロセッサ

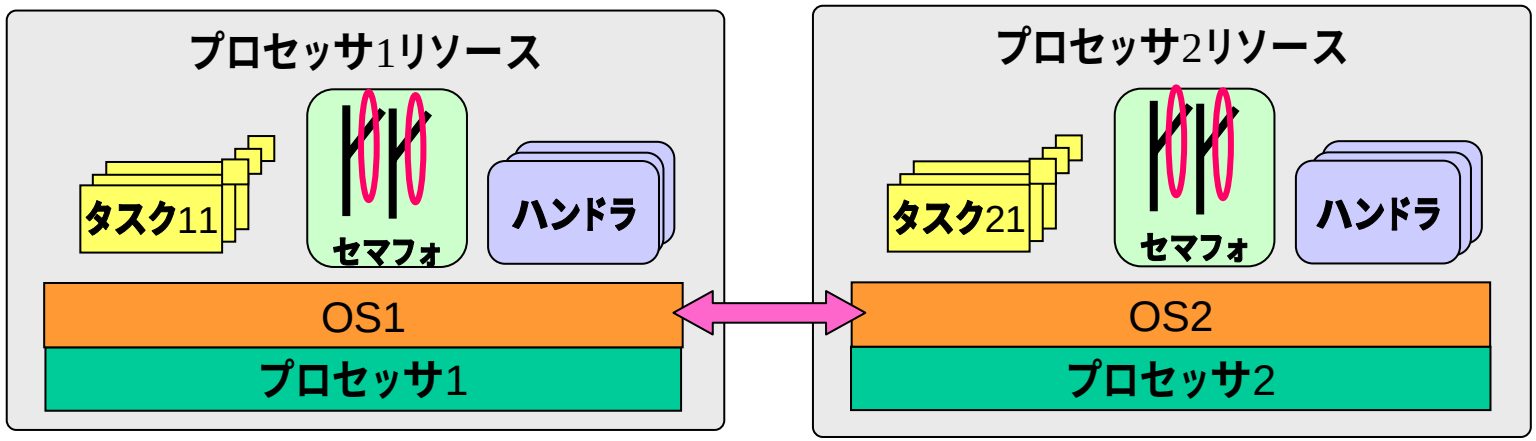
プロセッサ毎に機能を固定するモデル

- 組込みシステムは機能が固定されている場合が多いため，このモデルが用いられることが多い
- ハードウェア的にはプロセッサの機能が対称であっても，ソフトウェア的に機能を固定すると，機能分散マルチプロセッサに分類される
- 異なる種類のプロセッサを組み合わせる，ヘテロジニアスマルチプロセッサも機能分散マルチプロセッサの一種
  - ➡ OSの作りは異なる

機能分散/非対称型マルチプロセッサ：OSの機能

- タスクやセマフォといったオブジェクトはプロセッサ毎に固定され，動的にはプロセッサを移動することはなく，タスクはプロセッサ毎にスケジューリングする

- プロセッサ内とプロセッサ間と意識する必要のない同期･通信機能を提供する



機能分散/非対称型マルチプロセッサ：メリットとデメリット

メリット

- 機能を固定することで，ソフトウェア(OS)やハードウェア構成を用途に応じて最適化することが可能
  - プロセッサ間でコードを共有しないため，コヒーレンシキャッシュ等の複雑な回路を必要としない
  - バス構成やメモリ階層を多段にして性能向上を図れる
- 個別のプロセッサの性質はシングルプロセッサに近く，対称型と比較してリアルタイム性の保証が容易

デメリット

- OSでは動的な負荷分散をサポートしていないため，アプリケーションレベルで実現する必要がある
  - 各プロセッサに同一の処理を実行するタスクを用意して，切り替える

TOPPERSプロジェクトの狙い

現世代のリアルタイムOSの決定版の構築

!約20年間に渡るITRON仕様の技術開発成果をベースに

- ITRON仕様の標準的なオープンソース実装を用意することで，企業の開発投資をより先端的なソフトウェア部品や開発環境の開発に向ける
  - ➡ **TOPPERS/JSPカーネル**：μITRON4.0仕様RTOS

次世代リアルタイムOS技術の開発

- 組込みシステムの要求に合致し，ITRONの良さを継承した，次世代のリアルタイムOS技術を開発する
  - ➡ **TOPPERS/FDMP/SMPカーネル**：マルチプロセッサ用RTOS

組込みシステム技術者の育成への貢献

- オープンソースソフトウェアを用いた教育コースや教材の開発

マルチプロセッサ用リアルタイムOS開発の現状

機能分散型と対称型用の
リアルタイムOSの仕様とその実装を開発

機能分散型マルチプロセッサ用リアルタイムOS

- TOPPERS/FDMP (Function Distributed Multiprocessor) **カーネル**
- 2005**年にカーネル仕様を公開**
- 2006**年4月にカーネル実装をオープンソースとして公開** (1.1)
  - 9**月に**1.1.1**をリリース** (ARM MPCore**対応**)

対称型マルチプロセッサ用リアルタイムOS

- TOPPERS/SMP (Symmetric Multiprocessing) **カーネル**
- EPSON**と名古屋大学との共同研究**
- **カーネル仕様を会員向けに公開**

TOPPERS/FDMPカーネル（仕様）

TOPPERS/FDMPカーネル

機能分散マルチプロセッサ用のリアルタイムOS

- **μITRON4.0仕様を機能分散マルチプロセッサ向けに拡張**
- **マルチプロセッサ独自の機能は極力設けない**

アプリケーション開発を効率化

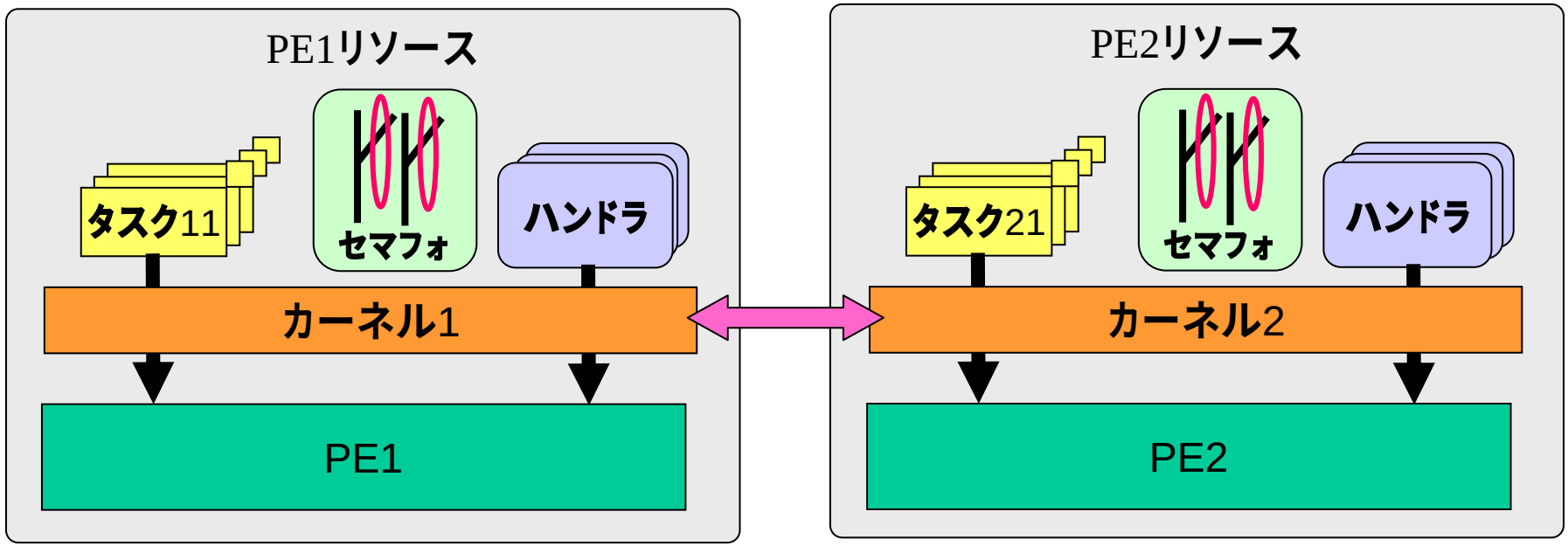
- **μITRON仕様のAPIでプロセッサ間の同期・通信が可能**
- **μITRON仕様OS向けのソフトウェア資産が活用可能**

実装も同時に開発

- **仕様の妥当性を確認するために，仕様の策定と同時に開発を進める**

第8回 LSI IPアワード IP賞受賞!

オブジェクト管理の概念図



- タスク，セマフォ，割込みハンドラ等のカーネルオブジェクトは特定のプロセッサに固定
- タスクやハンドラは，プロセッサを跨いで全てのオブジェクトにアクセスが可能

コンフィギュレーションファイル (静的APIの拡張)

- プロセッサ毎に囲みを作りその中にそのプロセッサに属するオブジェクトの静的APIを記述する
- コンフィギュレータはプロセッサ毎のディレクトリを作成して，kernel_cfg.cとkernel_id.h及び，クラスIDを付加したオブジェクトIDが記述されているclass_id.hを生成
- クラスIDの割付はコンフィギュレータにより行う

PE1に属するオブジェクトの生成情報

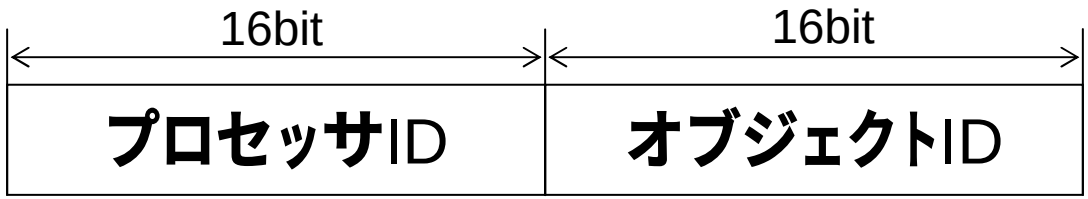
```
local_class PE1{
    INCLUDE("¥"sample1-dual1.h");
    CRE_TSK(TASK1, { TA_HLNG, (VP_INT) 1, ..);
    CRE_TSK(TASK2, { TA_HLNG, (VP_INT) 2, ..);
    ...
    CRE_CYC(CYCHDR1, { TA_HLNG, 0, ..);
    #include "../systask/timer.cfg"
    ....
}
local class PE2{
    INCLUDE("¥"sample1-dual2.h");
    CRE_TSK(TASK1, { TA_HLNG, (VP_INT) 1, ..);
    CRE_TSK(TASK2, { TA_HLNG, (VP_INT) 2, ..);
    ...
    CRE_CYC(CYCHDR1, { TA_HLNG, 0, ..);
    #include "../systask/timer.cfg"
    ....
}
```

PE2に属するオブジェクトの生成情報

システムコール

プロセッサを跨いでμITRONのシステムコールを実行可能

- **μITRON仕様ではオブジェクトをID番号で識別する**
  - **セマフォの取得** ： wai_sem(ID semid)
- **ID番号を拡張し，所属するプロセッサを示すようにする**
  - **上位16ビットをプロセッサID，下位16bitをオブジェクトID**
  - **ID番号はコンフィギュレーションツールで自動生成し，マクロを生成するため，シングルプロセッサ用のプログラムをそのまま利用可能**



TOPPERS

TOPPERS/FDMPカーネル (実装)

実装 : 対象 (動作) アーキテクチャー

現状のTOPPERS/FDMPカーネルは以下の
アーキテクチャーで動作する

- 各プロセッサのデータ (rodataも含む) を置くメモリが全てのプロセッサから同一のアドレスでアクセス可能であること
- 任意のプロセッサに割込み (プロセッサ間割込み) を発生可能であること
- プロセッサ間での排他制御のための機構を持つこと.
  - 例 : test & set 命令，Mutex回路
- プロセッサ間の排他制御機構を用いてロックをプロセッサ数x2個作成可能であること.
  - だたし2個でも動作は可能.

実装：特徴とサポートターゲット

性能を重視した実装

- プロセッサ数に対するスケーラビリティ（ロック単位）
- リアルタイム性（特に割込みに対する応答性）を損なわないための工夫（ロック取得ルーチンの工夫）

ポーティングが容易

- JSPカーネルをベースに実装しているため，JSPカーネルでサポートしているプロセッサへのポーティングは特に容易

サーポートターゲット

- ARM MPCore, Altera Nios2, Xilinx Microblaze, 東芝 MeP

専用コンフィギュレータ

- 拡張APIをサポート

シングルプロセッサ用のアプリケーションの移植 (排他制御)

シングルプロセッサ用のアプリケーションの移植 (排他制御)

シングルプロセッサ用アプリケーションを，FDMPカーネル上に
移植する際には，排他制御に注意する必要がある

シングルプロセッサでの排他制御の実現方法

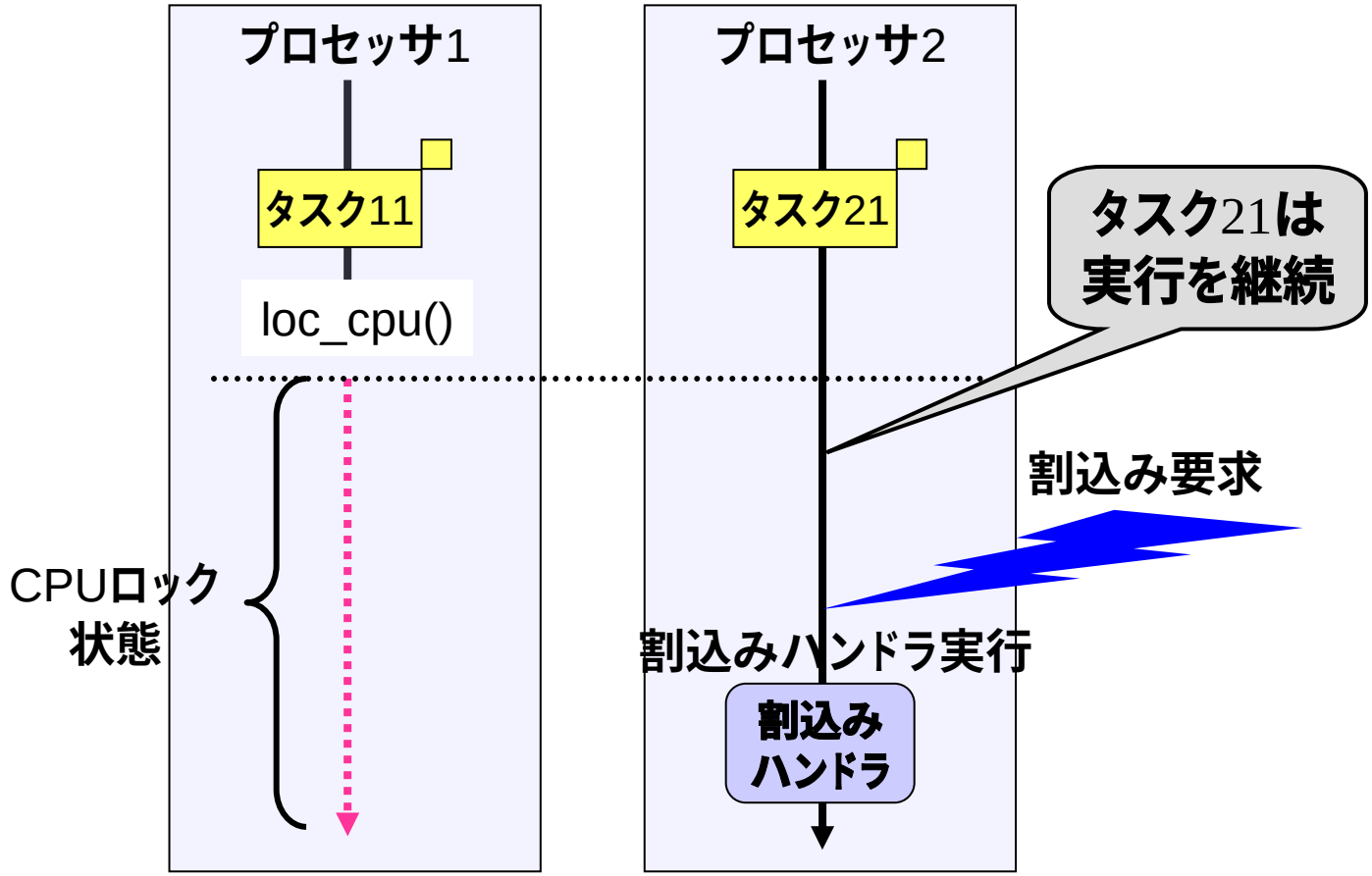
- CPUロック (割込み禁止) やディスパッチ禁止を使う場合が多い

➡マルチプロセッサ環境では排他を実現できない

μITRON仕様の排他関連機能

- CPUロック状態 (割込みの禁止とディスパッチの禁止)
  - タスク間，割り込みハンドラ間，タスク-割込みハンドラ間，
    の排他制御を実現
- ディスパッチ禁止状態
  - タスク間の排他制御を実現
- セマフォ・ミューテックス
  - タスク間の排他制御を実現

マルチプロセッサ環境下でのCPUロック状態



プロセッサ1がCPUロック状態になったとしても，プロセッサ2はその影響を受けず，
タスクの実行は継続され，割込みの受付と割込みハンドラの実行が可能である

排他制御：マルチプロセッサ環境下での排他制御

シングルプロセッサ用のアプリケーションの移植に関しては，
CPUロック状態により排他制御を行っている箇所を
その用途により変更する必要がある

- ディスパッチ禁止も同様

タスク間の排他制御

- セマフォやミューテックスを用いた排他制御に書き換える

タスク-割込みハンドラ間の排他制御

- 排他制御を行う必要があるタスクと割込みハンドラは，同じプロセッサで動作させるようにする
  - 現状では，異なるプロセッサ間のタスクと割込みハンドラ間の排他制御を実現する方法はない
    （SMPカーネルと同様の拡張も考えられる）

まとめ

まとめ

TOPPERSプロジェクトにおけるマルチプロセッサへの取り組み
- 機能分散マルチプロセッサ用リアルタイムOS
- 対称型マルチプロセッサ用リアルタイムOS

TOPPERS/FDMPカーネル
- 仕様と実装

シングルプロセッサ用のアプリケーションの移植（排他制御）
- 排他制御

今後の予定
- 実システムへの適用評価
- SMP向けハードウェアに対する最適
  - カーネルオブジェクトの共有
  - データ構造

TOPPERSプロジェクトの
ブースでMPCore上で動作する
FDMPカーネルを展示中