ET2011 カンファレンス
TOPPERSセッション

マルチプロセッサ向けRTOSのテスト

本田　晋也
名古屋大学 大学院情報科学研究科
附属組込みシステム研究センター
honda@ertl.jp

最終更新日：2011年11月7日

はじめに

マルチプロセッサ向けRTOSであるTOPPERS/FMPカーネルの検証スイート開発の活動と検出した不具合について紹介

- アジェンダ
  - TOPPERS/FMPカーネル
  - コンソーシアム型研究によるテストスイート開発
  - 検出した不具合
  - 実行タイミング制御シミュレータ
  - まとめと今後の活動

TOPPERS/FMPカーネル

組込みシステムにおけるマルチプロセッサの利用

汎用システムと同様に組込みシステムでも
マルチプロセッサの利用が進んでいる.

性能と低消費電力の両立

- 1個の高速なプロセッサを用いるより，複数の低速なプロセッサを用いた方が電力効率がよい.

マルチプロセッサ対応のRTOS必要性

- プロセッサ間同期・通信機能の利用
  - RTOSの提供する機能によりシステム構築を容易に
- タスクマイグレーションの利用
  - プロセッサを効率良く使用するシステムを実現

TOPPERS/FMPカーネル

リアルタイム性と動的なタスク移動の両立を目指したRTOS

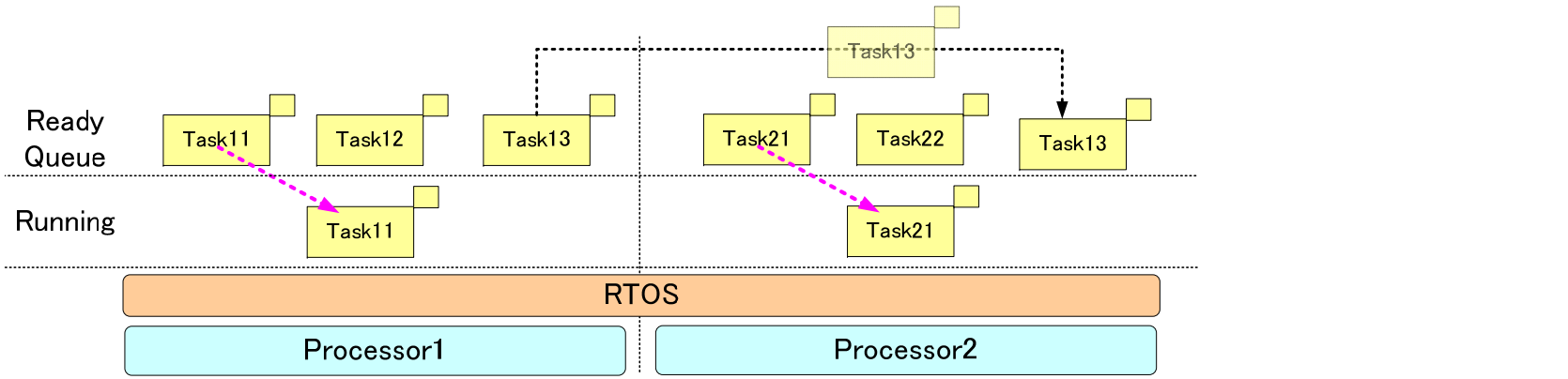
コア毎のタスクスケジューリング

- リアルタイム性の確保が（比較的）容易

タスクを移動させるAPIを追加

- 負荷変動への対応が可能
  - マイグレート可能なタイミングは要検討

ポリシとメカニズムの分離



TOPPERS/FMPカーネル

開発状況

- 2009/5： Release 1.0.0 オープンソースとして一般公開
- 2010/2： Release 1.1.0 バグフィックス
- 2011/7： Release 1.2.0 検証済みコードの公開

仕様

- 新世代カーネル統合仕様書として一般公開中

サポートプロセッサ

- ARM社 MPCore (ARM11/Cortex-A9)
- ALTERA社 Nios II
- ルネサス社 SH4A-MULTI, SH2A-DUAL
- ARMプロセッサの命令セット シミュレータ SkyEye

TOPPERS/FMPカーネル：利用事例

- シャープ株式会社 携帯電話945SH/002SH

出典: SoftBank webページ

コンソーシアム型研究によるテストスイート開発

FMPカーネルの検証（2009年時点）

- 包括的なテストは未実施
  - RTOSのテストは規模が膨大で工数が大きい
  - 大学では仕様検討/性能評価などの研究を中心としている
  - RTOSを実製品へ組み込む際，利用者側でテストを行っている

- FMPカーネルに対する検証
  - マルチプロセッサ対応RTOSのテスト開発に対する経験が少ない
  - テスト開発/実施の工数がシングルプロセッサより肥大化する
  - プロセッサ間の実行タイミングに依存したテストを考慮する必要がある

2009年度よりコンソーシアム型の研究組織を立ち上げ、
上記の問題を解決するためのテストスイートの開発を行っている

コンソーシアム型研究組織とは

名古屋大学 組込みシステム研究センターが
設定した研究テーマに対して，複数の企業・団体が
参加し共同で研究・開発を行う
(研究成果は一定期間後にオープンとする)

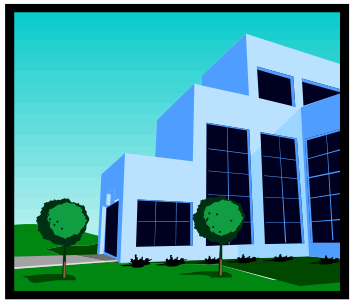
教員／研究者

コンソーシアム型
研究組織

研究者／技術者
(受託研究員)

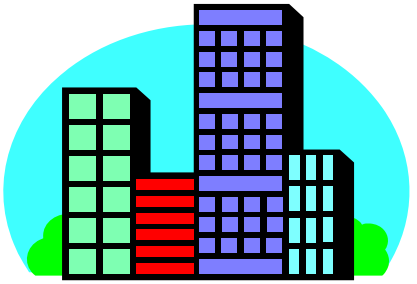
企業のRTOSに関する
経験／知見

・人材育成
・ビジネス発展



名古屋大学
組込みシステム研究センター(NCES)

産学官連携による
技術課題の解決



参加企業

マルチプロセッサ対応RTOSの研究・開発

- マルチプロセッサの重要性が高まり，利用したい企業が増えているが課題が多い
- マルチプロセッサ対応RTOSの自社開発は困難
  - 重複投資となり，製品の差別化に結びつかない
  - 各企業で個別に開発するよりも共同研究して共有した方が品質向上にも繋がる
  - マルチプロセッサの検証に関する経験が少ない
- 自社開発しなくてもRTOSを利用する上で，RTOS に詳しい技術者は必要

2009年度より，メインの研究テーマを「マルチプロセッサ対応RTOSの検証手法の開発」として，5社1機関の参加を得てコンソーシアムを形成

参加企業と体制(2009-2010年度)

参加企業(五十音順)

- 三洋電機株式会社(2009年度のみ)
- 株式会社デジタルクラフト
- 株式会社東芝セミコンダクター社
- 日本電気通信システム株式会社
- 富士ソフト株式会社
- 有限会社松浦商事(2009年度のみ)
- 宮城県産業技術総合センター
- ルネサスエレクトロニクス株式会社

体制

- 2009年度
  - 名古屋大学常駐：4名
  - 自社作業　　：4名
  - オブザーバ　　：3社
- 2010年度
  - 名古屋大学常駐：5名
  - 自社作業　　：1名
  - オブザーバ　　：3社

オブザーバ：月1回のミーティングに参加

FMPカーネル向けテストスイート

マルチプロセッサRTOSのテストのカテゴリ

仕様ベース

APIテスト (ブラックボックステスト)
処理単位テスト
SILテスト
クラス関連テスト
カーネル管理外割込みのテスト
カーネル起動/終了時の同期テスト

設計・ソースコードベース

APIテスト (ホワイトボックステスト)
ターゲット依存テスト
共通部ロック関数テスト
スタートアップモジュールテスト
コンフィギュレータテスト
機能拡張・チューニングテスト

エラー推測テスト

ロック区間テスト	割込み出口処理テスト
割込み禁止区間テスト	CPU例外出入り口処理テスト
タイマ割込みテスト	アイドル処理テスト
スピンロック中の割込みテスト	デッドロック回避テスト
マイグレーションテスト	バリア同期テスト

実施したテスト

- APIテスト
  - APIが仕様を満たしているかを確認
  - ソースコードカバレッジ確認
  - ブラックボックステスト,
    ホワイトボックステスト共に実施
  - 静的APIもテスト対象

- SILテスト
  - SIL(システムインタフェースレイヤ)APIが
    仕様を満たしているかを確認

- 処理単位テスト
  - テスト設計のみ実施
  - 設計段階でも不具合を検出

APIテスト概要

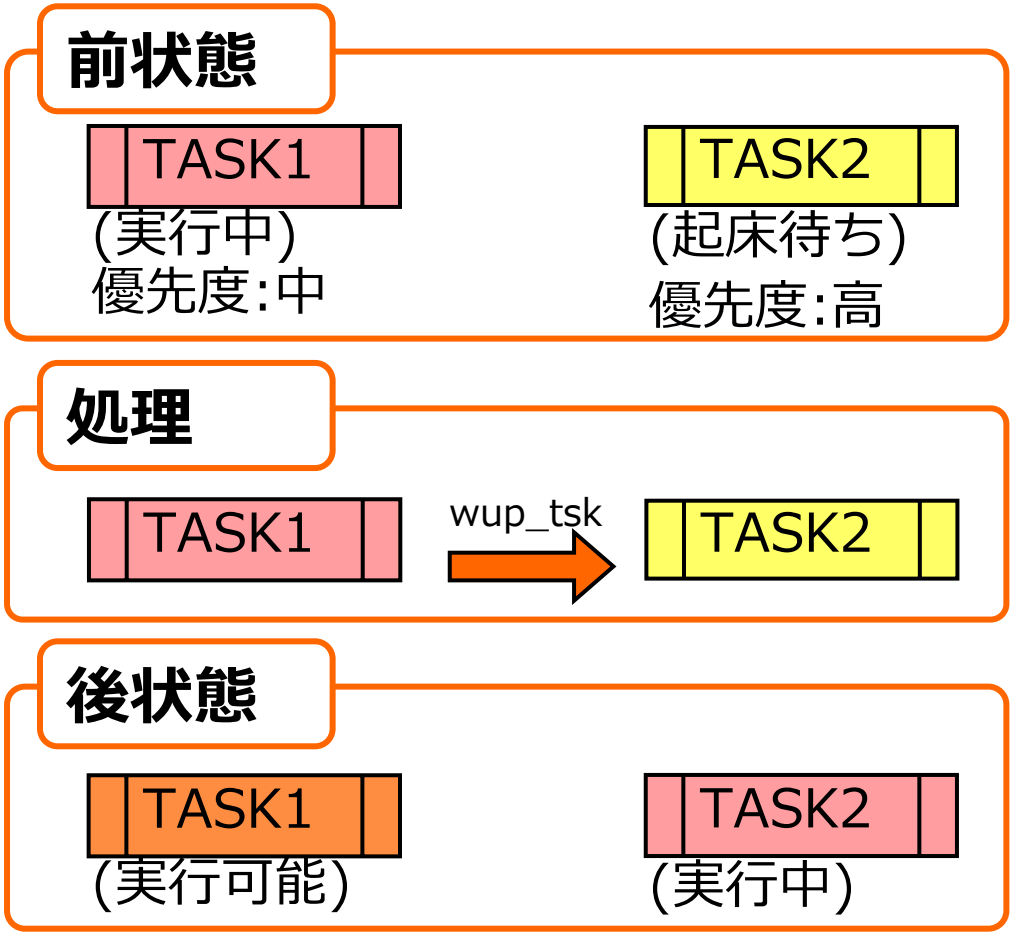
仕様書に基づいてAPI発行前後のシステム状態の変化を確認する

wup_tskの仕様抜粋

wup_tsk　タスクの起床
:
【機能】
tskidで指定したタスク(対象タスク)を起床する.
対象タスクが起床待ち状態である場合には,
対象タスクが待ち解除される.

仕様の振舞いを確認するテストを実施する

・自タスク以外のタスクを指定して呼び出す.
・起床待ち状態のタスクを指定する.
・対象タスクの優先度が，実行状態のタスクより高い場合，対象タスクが実行状態になること.



APIテスト設計プロセス

統合仕様書
API仕様

仕様カバレッジを
100%とする

事前に作成した条件
網羅表と結果が一致
することを確認

実行カバレッジ
ファイル
(gcovファイル)

シミュレータで実行し
カバレッジ取得

テストケース
(ブラックボックス)

**仕様/実装の
両面からテスト**

テストプログラム
・test.c
・test.h
・test.cfg

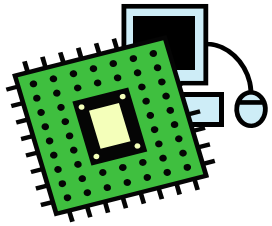
抽出したテストケースを
実施することでソース
コードのC1カバレッジが
100%となることを確認

**テストプログラムの自動生成
(TTG)**

C1カバレッジ確認
(条件網羅表)

条件網羅のための
テストケースを追加

テストケース
(ホワイトボックス)



テストケース数

機能名	ASP	FMP
タスク管理機能(task_manage)	181	604
タスク付属同期機能(task_sync)	205	280
タスク例外処理機能(task_except)	85	123
データキュー機能(dataqueue)	280	370
イベントフラグ機能(eventflag)	145	116
メールボックス機能(mailbox)	98	66
固定長メモリプール機能(mempfix)	97	66
優先度データキュー機能(pridataq)	264	322
セマフォ機能(semaphore)	106	91
周期ハンドラ機能(cyclic)	20	61
アラームハンドラ機能(alarm)	39	146
システム状態管理機能(sys_manage)	76	135
割込み管理機能(interrupt)	25	17
CPU例外管理機能(exception)	9	10
タスクの状態参照機能(task_refer)	39	88
システム時刻管理機能(time_manage)	6	3
スピンロック(spin_lock)	0	48
静的API	110	47
合計	**1,785**	**2,593**

- テストケースの抽出は，コンソーシアム参加企業におけるRTOS開発の経験を活かし，組み合わせ爆発を抑止（ポリシーを決定）
- FMPカーネルは，ASPカーネルのマルチプロセッサ拡張であるため，ASPカーネル用のテストケースは，そのままFMPカーネルで使用可能
- FMPカーネルで追加されたAPIやプロセッサを跨ぐテストケースをFMPカーネル用のテストケースとして追加
- FMPカーネルに対するテストケースは合計**4,378件**となった

テストプログラム自動生成の必要性

スクラッチプログラムで開発した場合の問題点

- 1つのテストプログラムとして開発？
  - 関数名や変数名の衝突
  - ターゲットによってはメモリ不足が発生

- テストケース毎にテストプログラムを開発？
  - テストケースの数だけロード→実行を繰り返す必要がある

- ターゲットシステムによってテストケースの実施可否(要否)が異なる
  - プロセッサ数/タイマアーキテクチャなどの違い
  - 未サポートAPIの扱い

4,000件以上のテストをスクラッチプログラムで
開発するのは工数・保守性の観点から非現実的である

テストシナリオからのテストプログラム生成

テストケース
(ブラックボックス)

テストケース
(ホワイトボックス)

テストケースを
前状態・処理・後状態
へ具体化する

形式化したテストシナリオ

前状態
　<API発行前のシステム状態>

処理
　<API発行処理>

後状態
　<API発行後のシステム状態>

TESRY記法

(TEst Scenario for Rtos by Yaml)

入力

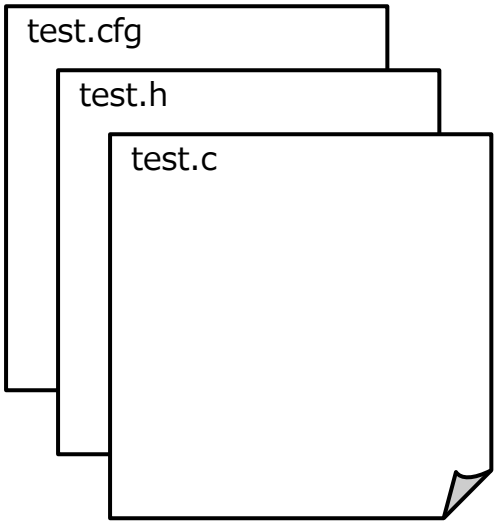
TTG

出力

(TOPPERS Test Generator)

・Rubyで開発
・約22,000行

テストシナリオを
実現するテストプログラム



TESRY記法

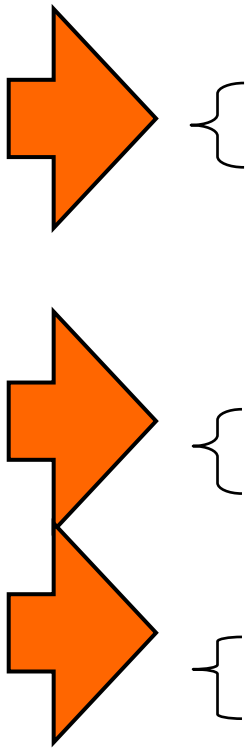
階層型データ形式言語である**YAML形式**を用いて
全カーネルオブジェクトの属性/状態の記述方法を定めた
(TESRY記法で記述したデータファイルを**TESRYデータ**と呼ぶ)

(例)wup_tskのテストケース

前状態
優先度中のTASK1が実行状態
優先度高のTASK2が起床待ち状態

処理
TASK1がwup_tsk(TASK2)を
発行してE_OKが返る

後状態
TASK1が実行可能状態となる
TASK2が実行状態となる



```
pre_condition:
  TASK1:
    type   : TASK
    tskpri : TSK_PRI_MID
    tskstat: running
  TASK2:
    type   : TASK
    tskpri : TSK_PRI_HIGH
    tskstat: waiting
    wobjid : SLEEP

do:
  id     : TASK1
  syscall: wup_tsk(TASK2)
  ercd   : E_OK

post_condition:
  TASK1:
    tskstat: ready
  TASK2:
    tskstat: running
```

前状態から変化のないパラメータは省略可能→

TTGのマルチプロセッサ拡張

- TESRY記法の拡張
  - 処理単位が実行されるプロセッサを指定可能に

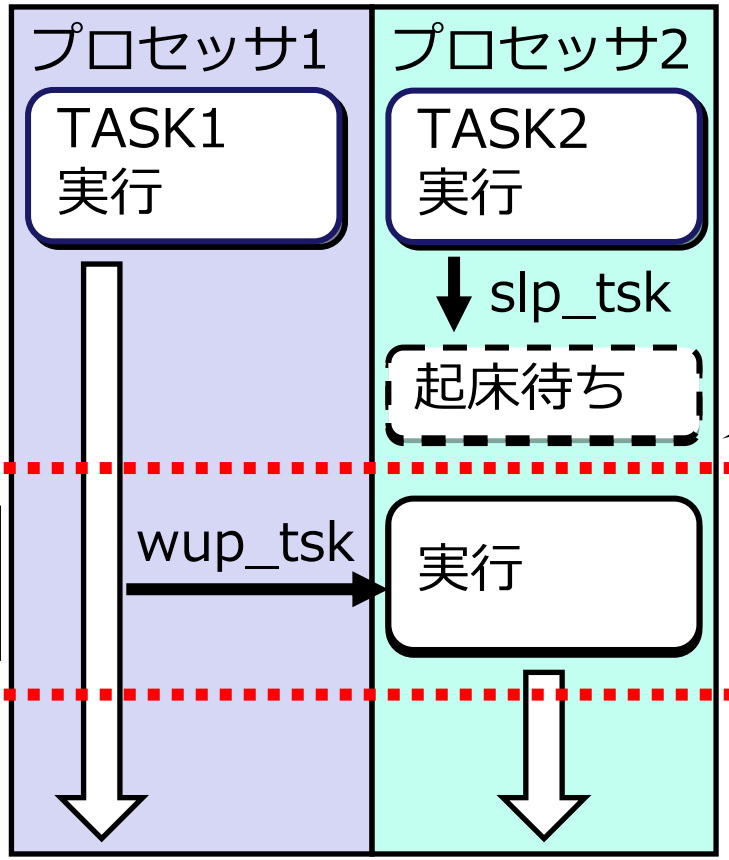
- テストプログラム生成の拡張
  - テストプログラム内でのプロセッサ間同期の実現
  - プロセッサ間同期が必要な箇所の整理
  - プロセッサ間同期の実現方法
    - RTOSの同期機能は使用できない
    - テストプログラムで使用する同期用のライブラリを整備

```
pre_condition:
  TASK1:
    type   : TASK
    tskstat: running
    prcid  : 1
  TASK2:
    type   : TASK
    tskstat: waiting
    wobjid : SLEEP
    prcid  : 2

do:
  id     : TASK1
  syscall: wup_tsk(TASK2)
  ercd   : E_OK

post_condition:
  TASK2:
    tskstat: running
```

同期処理の必要性

TESRYデータ

テストプログラムのフロー



前状態
- TASK1がプロセッサ1で実行状態
- TASK2がプロセッサ2で起床待ち状態

処理
TASK1がwup_tsk(TASK2)を発行してE_OKが返る

後状態
TASK2が実行状態となる

TASK2が起床待ちとなる前にTASK1がwup_tskを発行すると，テストの内容が異なってしまう

※テスト内容によっては正しく同期を行わないと無限ループや想定外のエラーが発生する

TESRYデータに定義した内容に従って動作するように，
適切な**同期処理**を行うテストプログラムを生成する必要がある

プロセッサ間同期ライブラリ

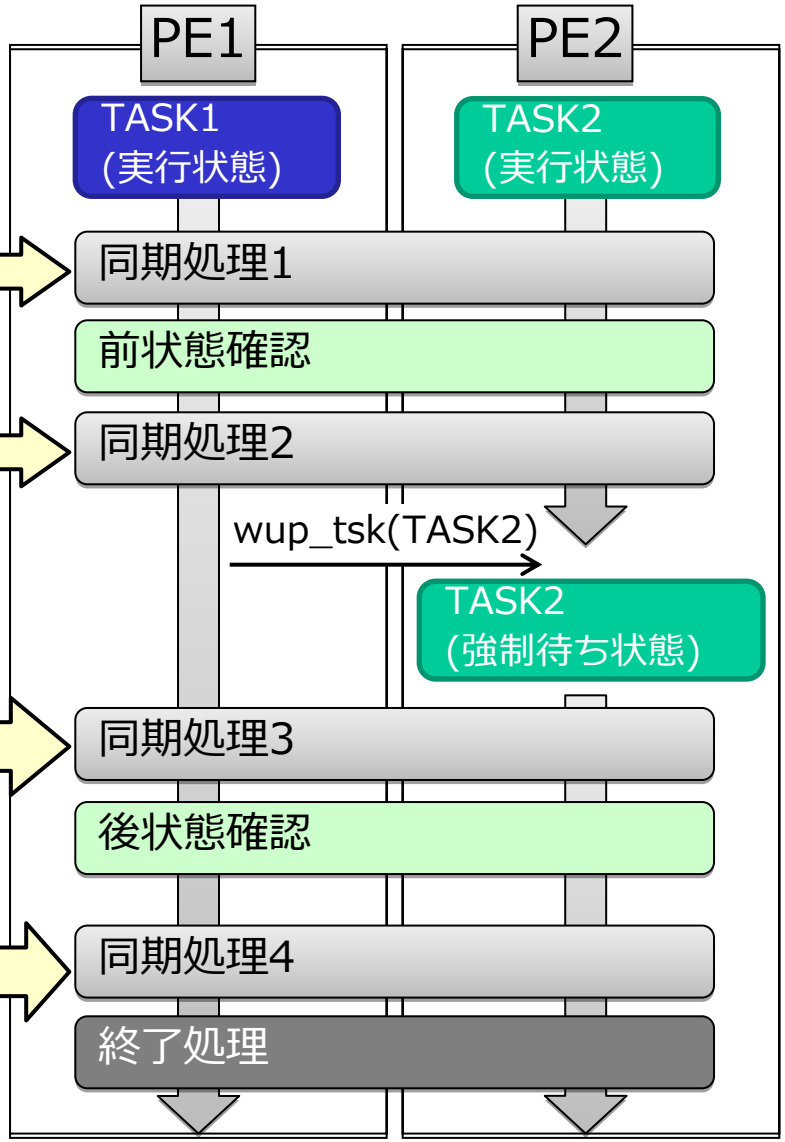
同期を行う状況と方法に分類して整理
- 同期パターン（状況）　…10種類
- 同期メカニズム（方法）　…　4種類

全プロセッサの**前状態作成**完了を待つ同期

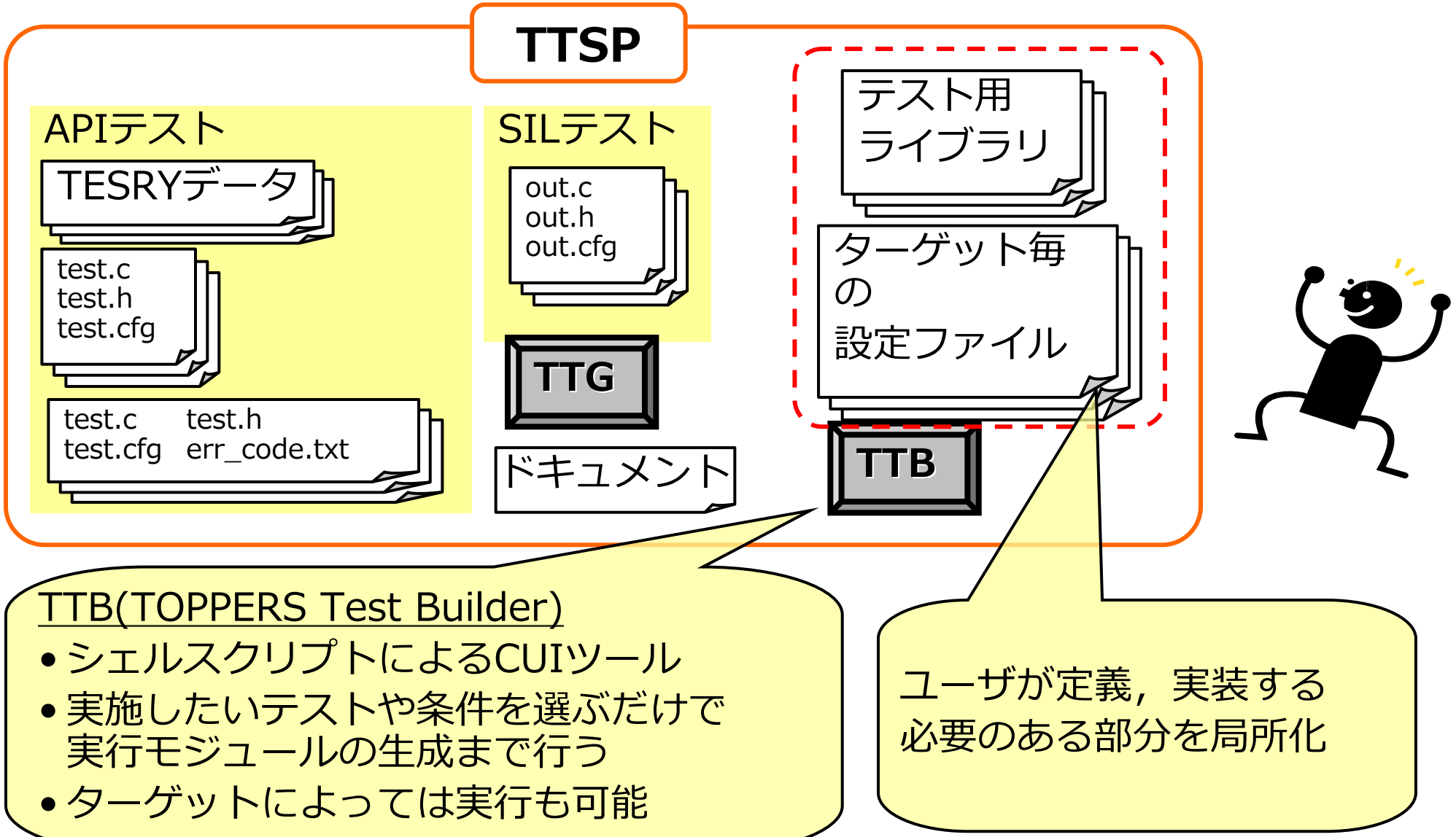
全プロセッサの**前状態確認**完了を待つ同期

テスト対象のAPIの発行処理完了まで，他のプロセッサでの処理を，後状態へと**先に進めさせない**ための同期

後状態確認の**処理完了**まで，実行状態の処理単位を終了させないための同期



TTSP(TOPPERS Test Suite Package)



TTB(TOPPERS Test Builder)

- シェルスクリプトによるCUIツール
- 実施したいテストや条件を選ぶだけで実行モジュールの生成まで行う
- ターゲットによっては実行も可能

ユーザが定義，実装する必要のある部分を局所化

TTSPの入手方法

TOPPERSサイトから
ダウンロード可能

http://www.toppers.jp/ttsp.html

- ASPカーネル用を公開
- FMPカーネル用は,
  2012年3月末に公開予定

検出した不具合

検出した不具合件数

年度	2009	2010		合計
テストカテゴリ	API	API	SIL	
統合仕様書	15	18	1	34
コンフィギュレータ	2	1	0	3
ASPカーネル	12	1	0	13
FMPカーネル	11	37	0	48
合計	**28**	**55**	**1**	**98**

(2011年9月現在)

網羅的にテストされていなかったため，不具合が多数検出された

不具合を解決し，以下をリリース

- 統合仕様書 Release 1.3.0
- ASPカーネル Release 1.7.0
- FMPカーネル Release 1.2.0

主要な不具合の分類

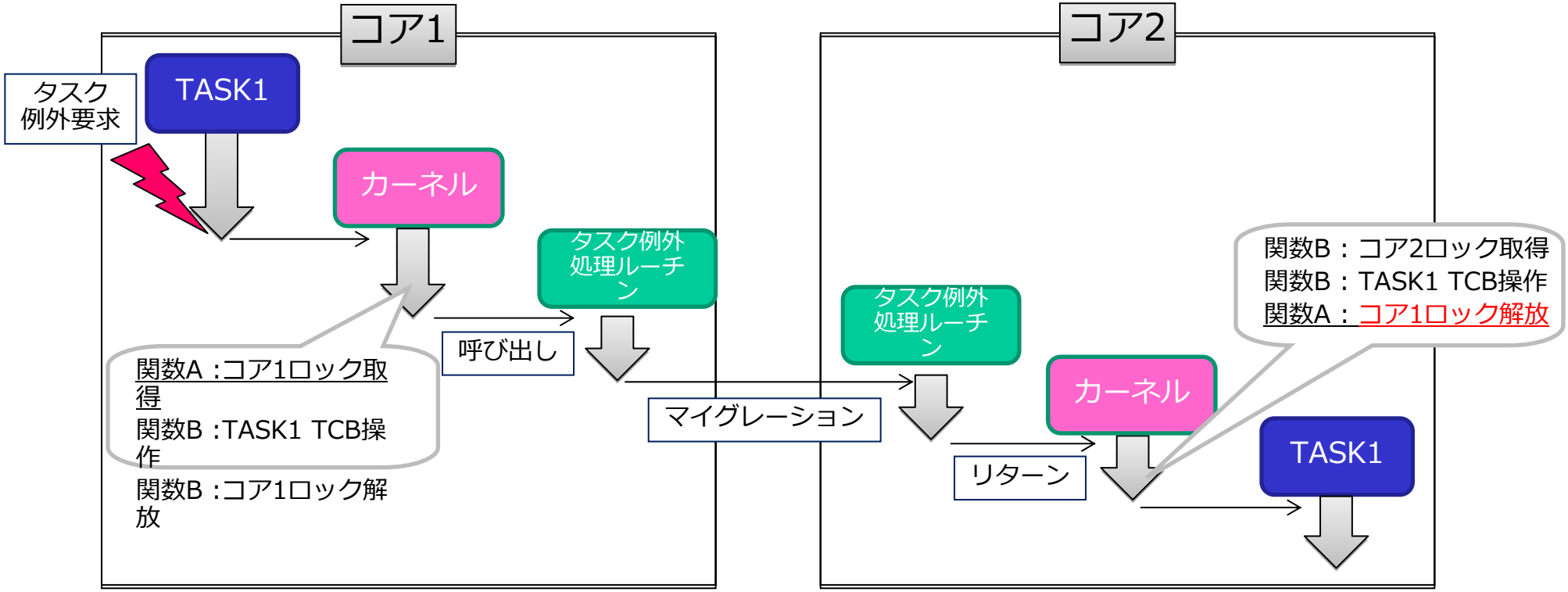
- ロックとマイグレーションに関連する不具合が多い

名前	主な原因	種別
過渡状態で一部システムコールを呼び出すと無限ループとなる	ロック取得ルーチンの考慮ミス	ロック
iset_flg()でディスパッチが起きる時カーネルが落ちる	ロック解放ルーチンの記述ミス	ロック
rel_mpf()でディスパッチが必要なのに発生しない	ロック解放ルーチンの記述ミス	ロック
ter_tsk()(ジャイアントロック時コード)でディスパッチが発生しないことがある	PCBの指定ミス（マイグレーション前のPCBを指定）	マイグレーション
強制待ち/他PEへのキューイングありのタスクをter_tsk()(プロセッサロック時コード)すると無限ループする	デッドロック回避時のリトライ条件の指定ミス.	ロック
TCL_ADMIN_PRCで囲ったアラームがPE2に割り付く	初期化時に登録するPCBのミス.	その他
スピンロック取得中にも関わらずTSPN_LOCが返らない	ロック解放とOS内部状態変更順序ミス.	ロック
【ARM/MPCore依存部】ARM11MPcoreによるSpurious Interrupt	ARMv7のコードのARMv6展開忘れ	その他
タスク例外処理ルーチン内でマイグレートした場合のロック解除漏れ	マイグレーションの考慮忘れ.	マイグレーション

タスク例外処理ルーチン内でマイグレートした場合のロック解除漏れ

- TCBの操作時は実行されているコアのロックを取得する必要がある
- 関数Aは取得したロックをローカル変数に記憶してロック解放時に使用
  - 取得するロックの判定処理を省略する最適化

  ➡ タスク例外処理ルーチン内マイグレーションした場合は取得するロックが変わるため正しくないコードである



不具合の分析

ロック関連の不具合

•不具合検出から原因の特定に手間がかかる

•テストケース自体は正常に終了する場合が多い

- 例)不具合のあるパスを実行した次のテストケースでロックが取得できずにテストが進まずエラーとなる

マイグレーション関連の不具合

•マイグレーションするタイミングの考慮漏れにより不具合が発生

実行タイミングに依存する不具合

•APIテストを各プロセッサの実行タイミングをずらして実行することよって発見された不具合もある.

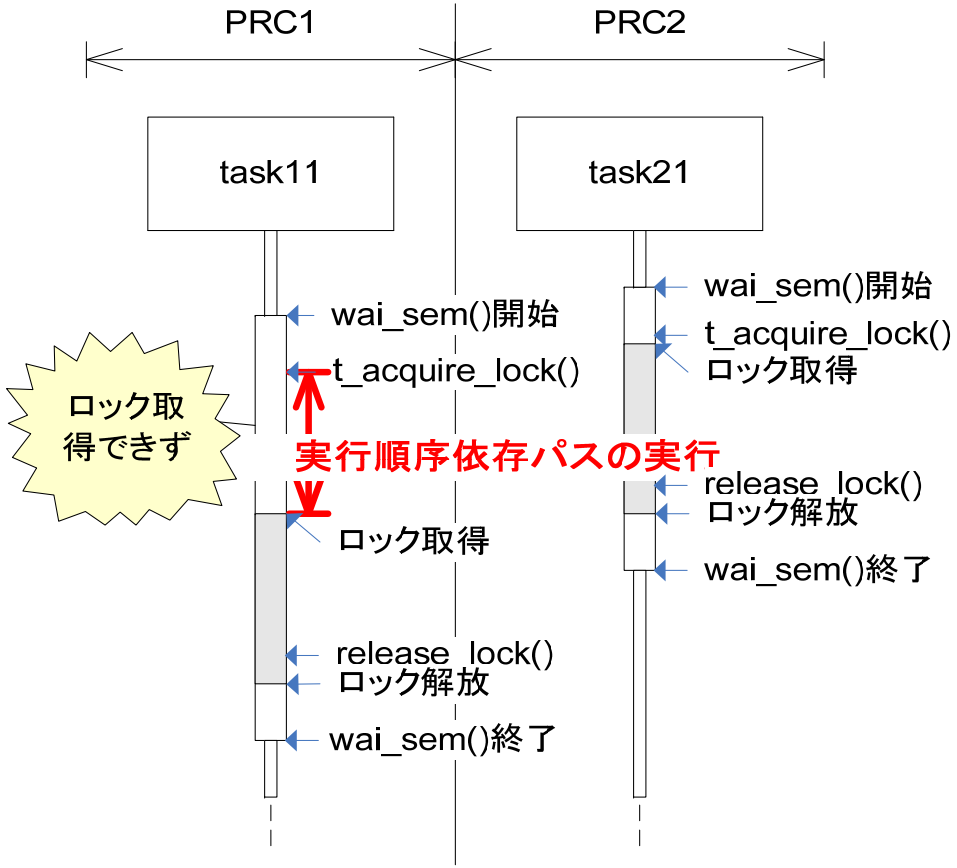
•これらの不具合は発生確立が低いため現象の把握も多くの時間が必要であった.

実行タイミング制御

実行順序に依存したパス（実行順序依存パス）

- FMPカーネル内には，プロセッサが特定の実行順序で実行した場合のみ実行されるパスが存在する

  例)APIの入り口でロックの取得を試み，他のプロセッサがロックを取得していたため，ロックの取得に失敗するパス



実行順序依存パスの実行方法

- 単純な方法：連続試行
  - 分岐を繰り返し実行することで実行順序依存パスを実行することを期待した手法

    ✗実行まで長時間要する可能性がある

例）実機(Altera Nios Ⅱ, 50Mhz)で計測

- 実行順序依存パスを実行するのに要した時間
- 100万回関数を呼び出したときの実行順序依存パスを実行した確率

関数名	分類	時間[秒]	確率[%]
t_acquire_lock	ロック取得前	0.1	24.1
t_acquire_tsk_lock_self	ロック取得後	0.4	14.1
ter_tsk	デッドロック回避	2.2	4.5
wait_tmout	デッドロック回避	ー	0.0

1m秒毎のタイマ割込みで実行される関数

6時間以上試行を繰り返しても実行されず．実行頻度が他の関数と比べて低いのが原因

連続試行の問題点：決定的な実行の必要性

- 期待した状況で実行したか不明
  - トレースログなどを解析する必要がある
  - 繰り返し実行したログの解析は大変

- 実行順序依存パスを実行しない場合の原因が複数存在
  1. 試行回数の不足
  2. 連続試行プログラムのミス
  3. テスト対象のプログラムのミス

実行順序依存パスを決定的に実行する方法が必要

実行制御シミュレータ TimingSim

FMPカーネルの実行順序に依存する分岐を
分岐網羅するためのシミュレータ

基本機構

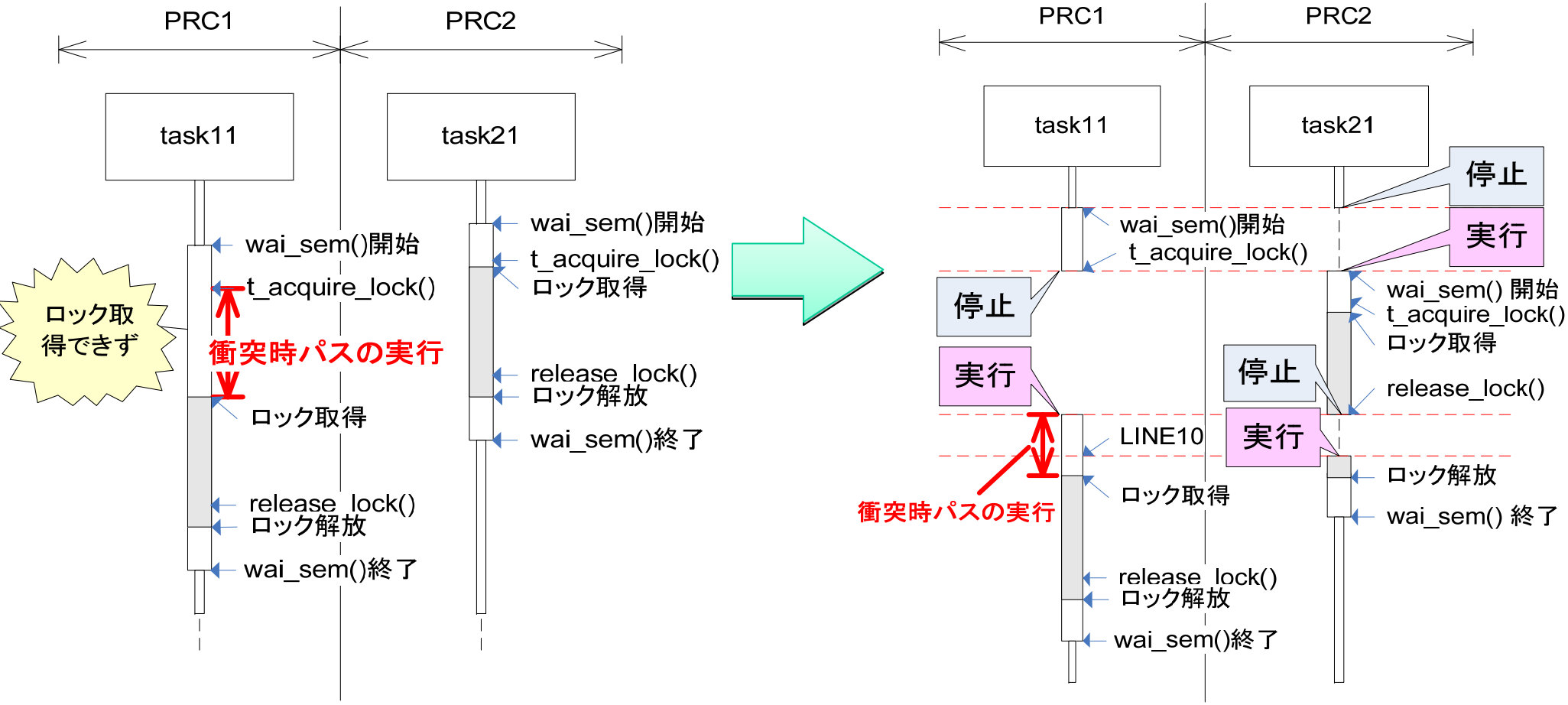
•ソースコード中の指定した箇所（フック）を実行した場合に実行する処理（アクション）を登録可能

- フック
  1. PCフック：登録したアドレスを実行した際にアクションを実行
  2. カレントフック：直ちにアクションを実行
- アクション
  1. プロセッサ動作制御アクション：プロセッサの実行と停止
  2. 割込み動作制御アクション：プロセッサに対して割込みを発生

この機構を用いて各プロセッサの実行順序をシミュレータに指示

実行制御シミュレータ TimingSim：実行例

- APIの入り口でロックの取得を試み，他のプロセッサがロックを取得していたため，ロックの取得に失敗するパスの実行例



まとめと今後の活動

まとめ

- FMPカーネル
  - マイグレーションをサポートしたマルチプロセッサ向けRTOS

- コンソーシアム型研究によるテストスイート開発
  - 4000件を超えるテストケースの開発
  - TTGによるテストプログラムの自動生成
  - 98件の不具合を検出し，カーネルの品質を高めることができた
  - マルチプロセッサRTOSの検証に対する多くの知見が得られた

- 実行制御シミュレータ TimingSim
  - 実行順序に依存したパスを確実に実行するためのシミュレータ

2011年度のコンソーシアム型研究：AUTOSAR OSの開発

- AUTOSAR OS仕様の明確化と問題点の修正
- 自由に使える日本語版の仕様書の作成
- オープンソース実装の開発
- テストスイート開発
- 性能評価

OSの仕様は異なるが，TTSPの設計思想は適用可能

AUTOSAR

参加企業(五十音順)

株式会社ヴィッツ
株式会社OTSL
株式会社サニー技研
株式会社デンソー
株式会社東芝
トヨタ自動車株式会社
日本電気通信システム株式会社
パナソニック アドバンストテクノロジー株式会社
富士ソフト株式会社
富士通VLSI 株式会社
ルネサス エレクトロニクス株式会社

2012年度のコンソーシアム型研究の活動予定

- AUTOSAR関連の研究・開発を継続して実施

- 実施予定内容
  - マルチプロセッサ対応 AUTOSAR OS の開発
    - 実装の開発
    - テストスイートの開発
  - AUTOSAR準拠の通信ミドルウェアの開発

- 2012年度からの参加も可能
  - 参加条件は調整中

問い合わせ先：nces-office@nces.is.nagoya-u.ac.jp