MBP（モデルベース並列化）を用いたクロスレイヤ設計

2017年11月16日

名古屋大学情報学研究科
組込みマルチコアコンソーシアム
枝廣　正人

アウトライン

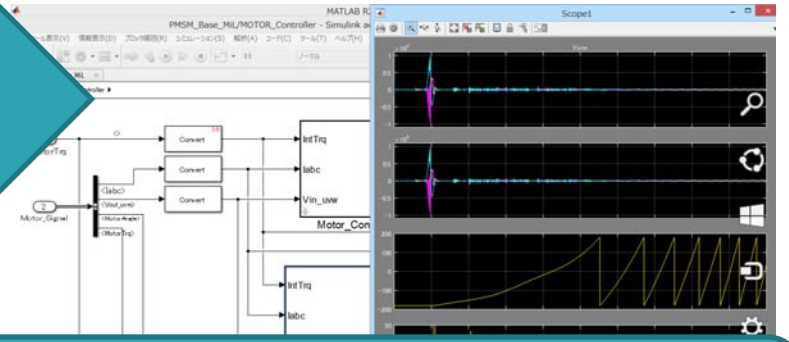
- モデルベース開発（MBD）とクロスレイヤ設計
- クロスレイヤ設計に必要な技術

モデルベース開発(MBD)

制御対象
（プラント）

1．数理モデル化

MATLAB/Simulink

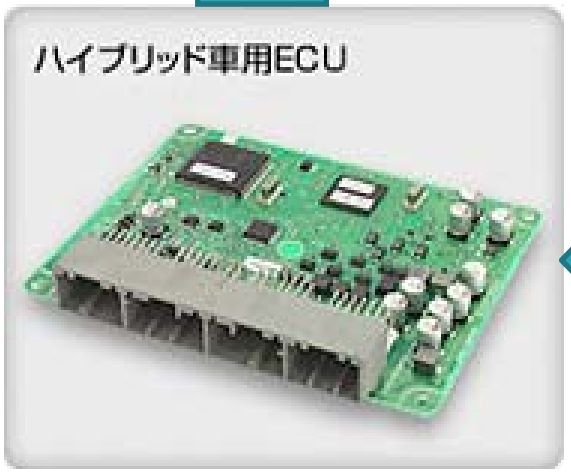


2．制御器の数理モデル設計
（シミュレーションベース）

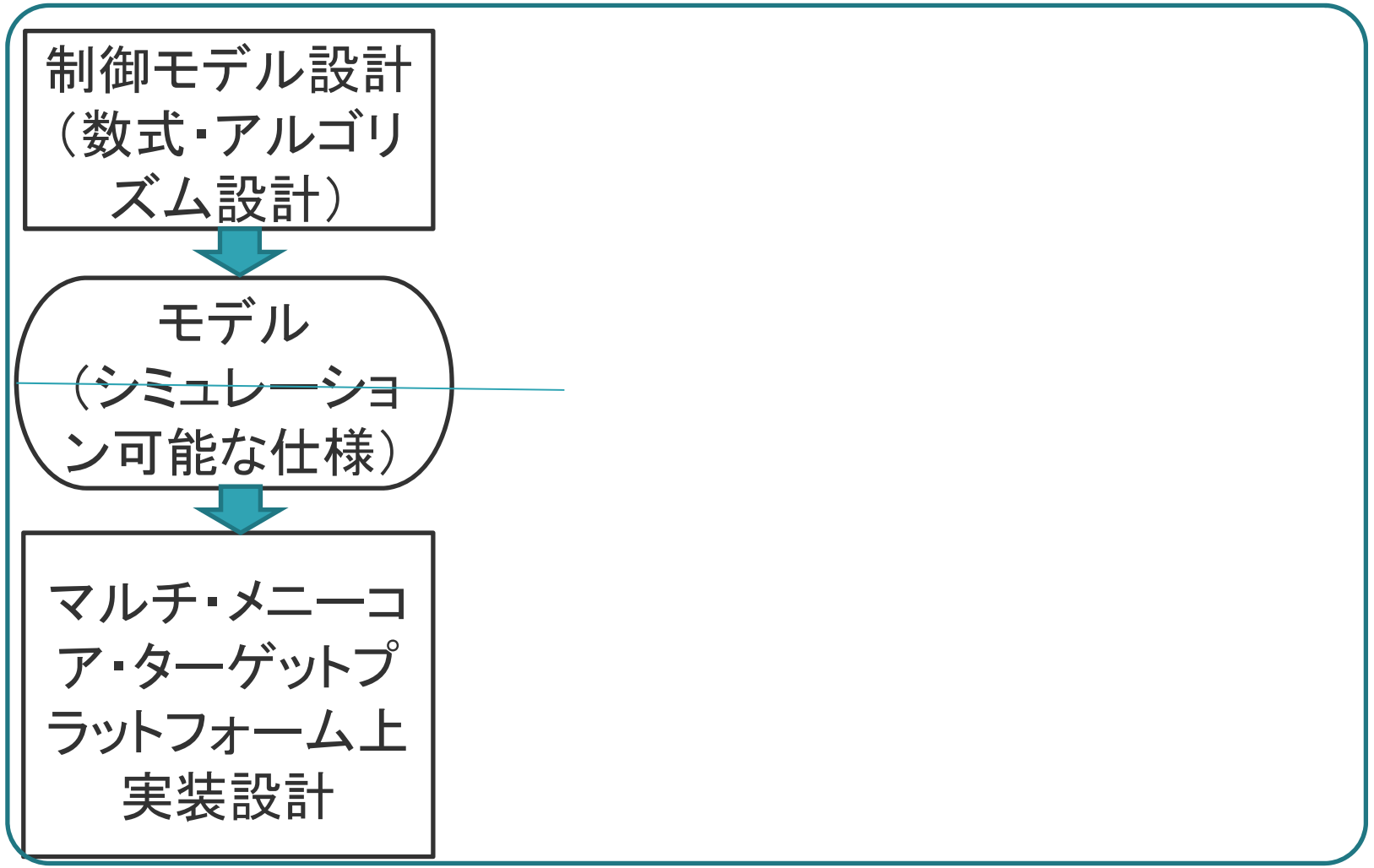
制御器（コントローラ）

3．実装コード化

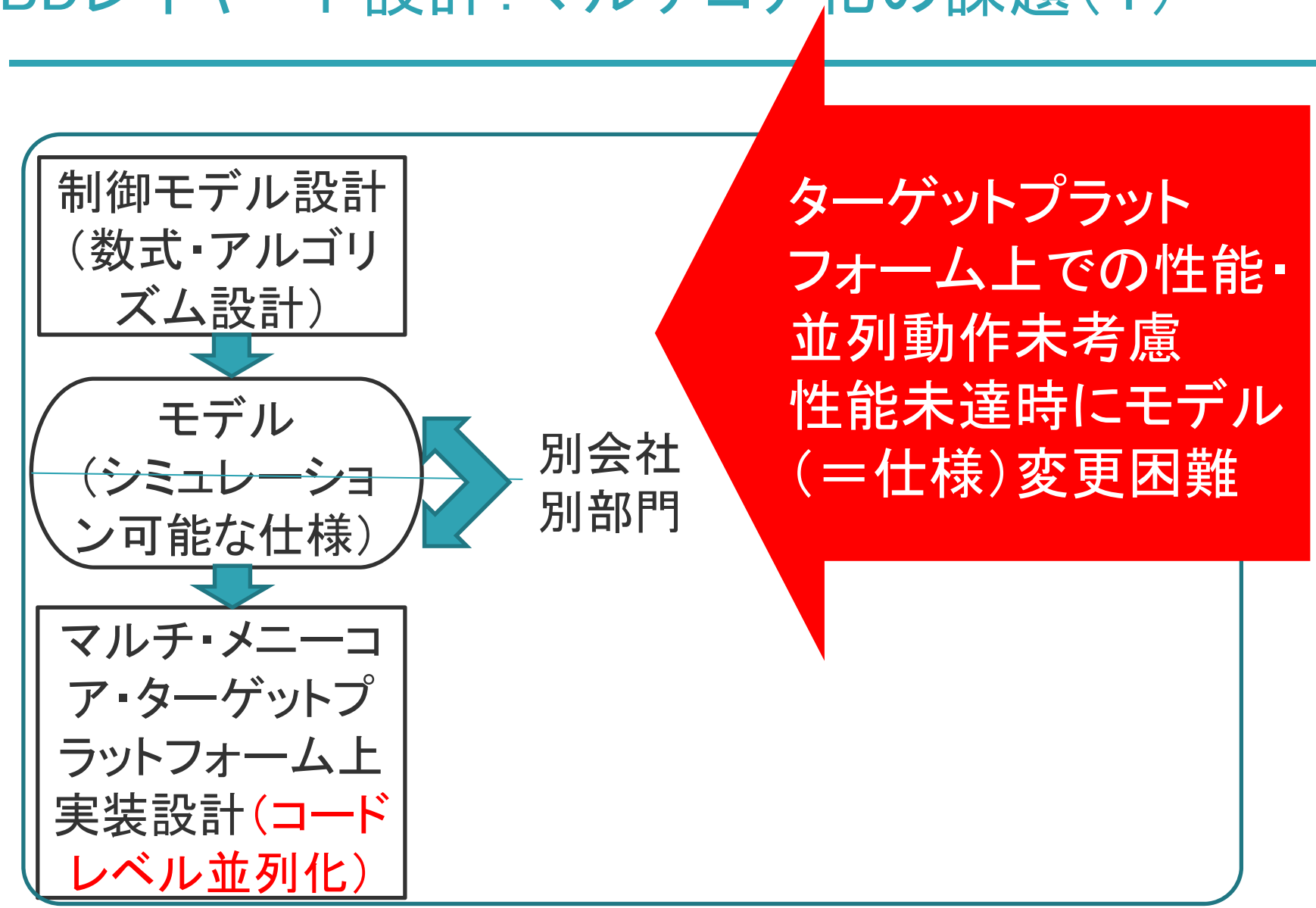
制御



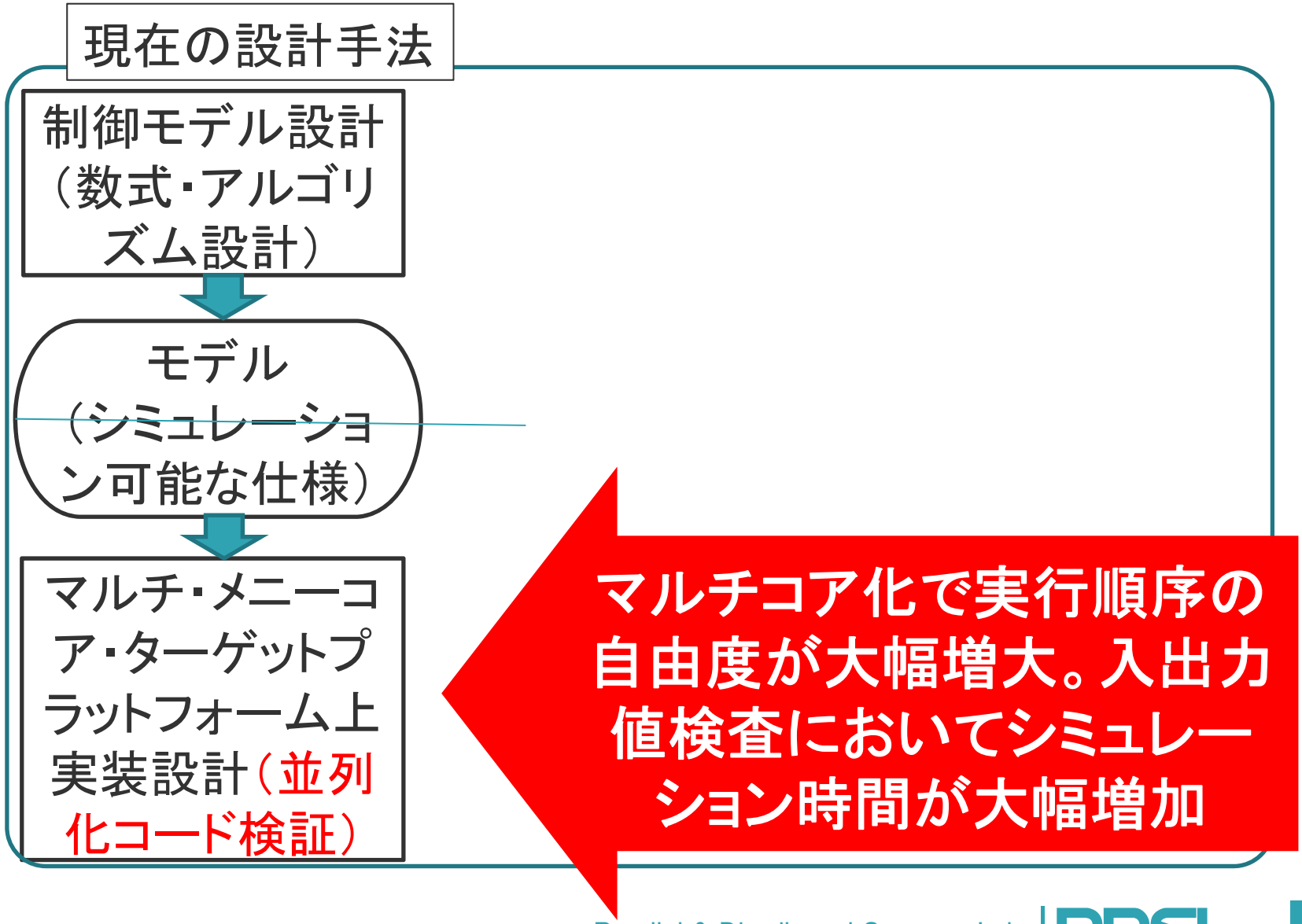
MBDの現状（レイヤード設計）



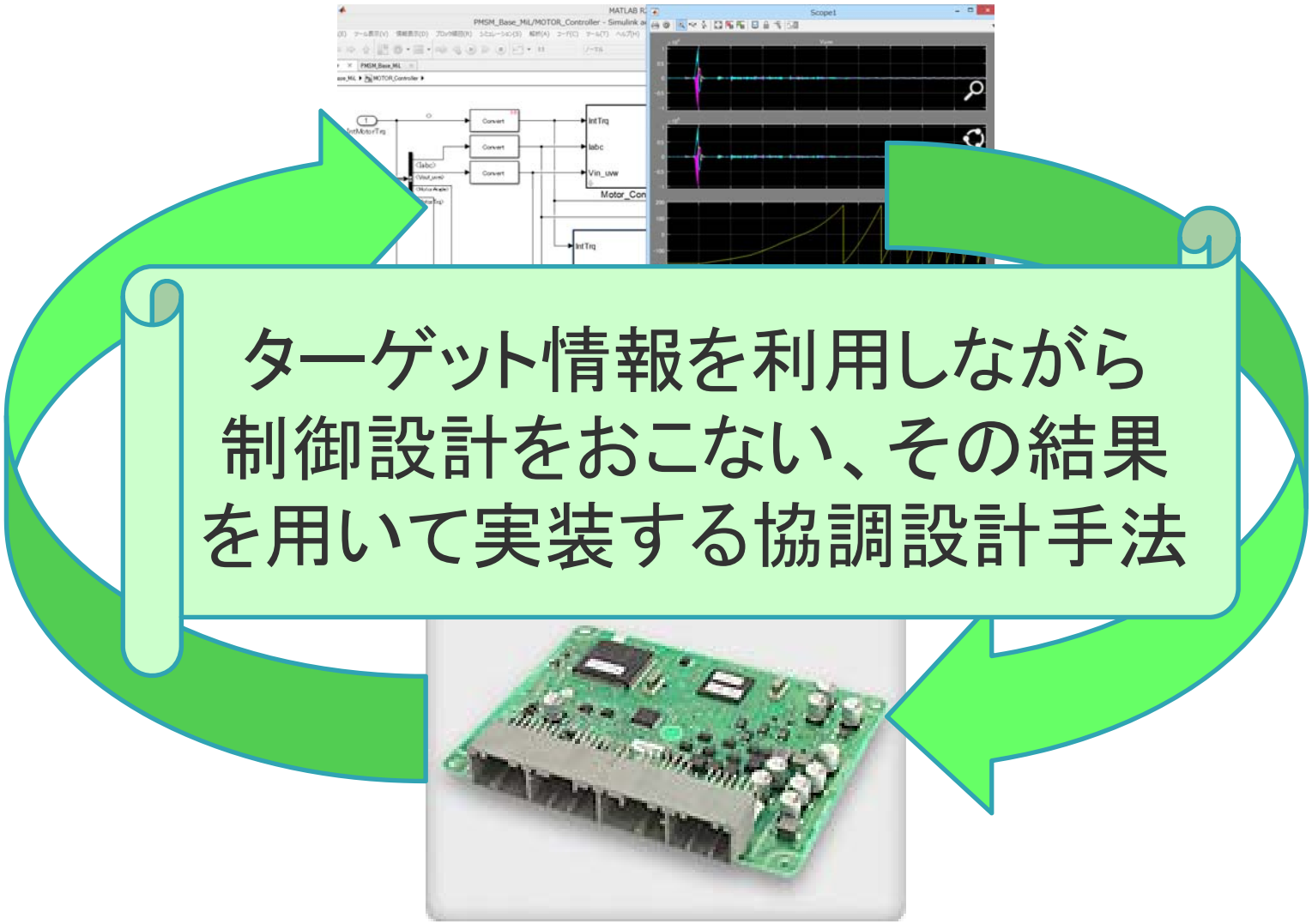
MBDレイヤード設計：マルチコア化の課題（1）



MBDレイヤード設計：マルチコア化の課題（2）



MBDにおけるクロスレイヤ設計とは



クロスレイヤ設計　手順とメリット

1. ハードウェア抽象化記述SHIMを用い、ターゲット情報を抽象化し、モデルレベルで利用可能にする
   - SHIMについては次の講演で
2. モデルベース並列化(MBP)により、ターゲット情報を用いてモデルレベルで自動並列化設計・検証、ターゲット向け並列化コード生成
3. 並列性能・動作をモデルにフィードバックし、並列動作を意識した制御設計を支援

- メリット
  - 協調設計による設計収束時間短縮
  - 上流での並列動作設計による検証容易化
  - 次世代ターゲット向け制御アルゴリズム開発加速

事例

モデルベース開発からTOPPERS搭載システムへのクロスレイヤ自動設計を利用したマルチコアモータ制御実装

～第7回TOPPERS活用アイデア・アプリケーション開発コンテスト 銅賞

～ 組込みシステムシンポジウム2017 優秀ポスター賞

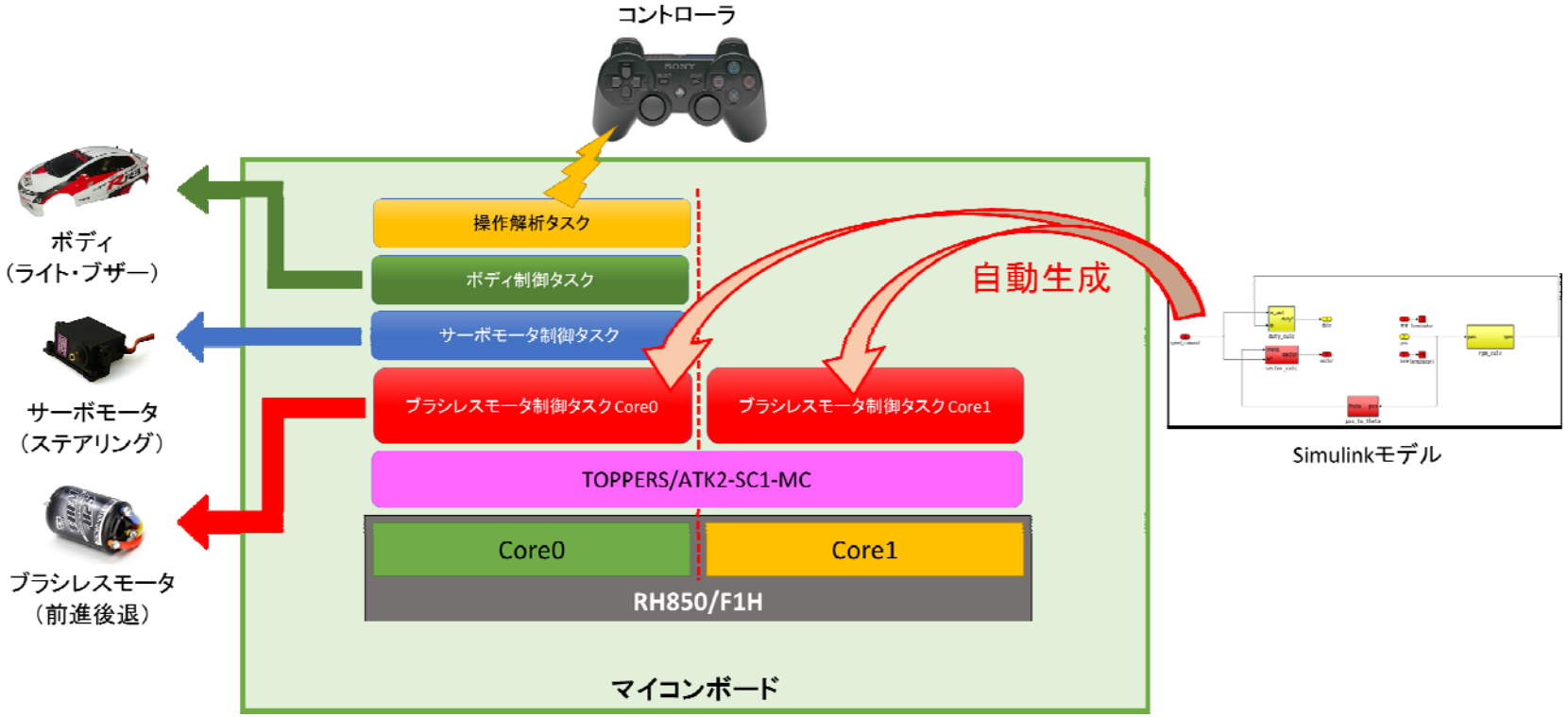
名古屋大ブース（D-52－⑭）にてデモ中！

デモ動画

https://youtu.be/eszVE9wc_c

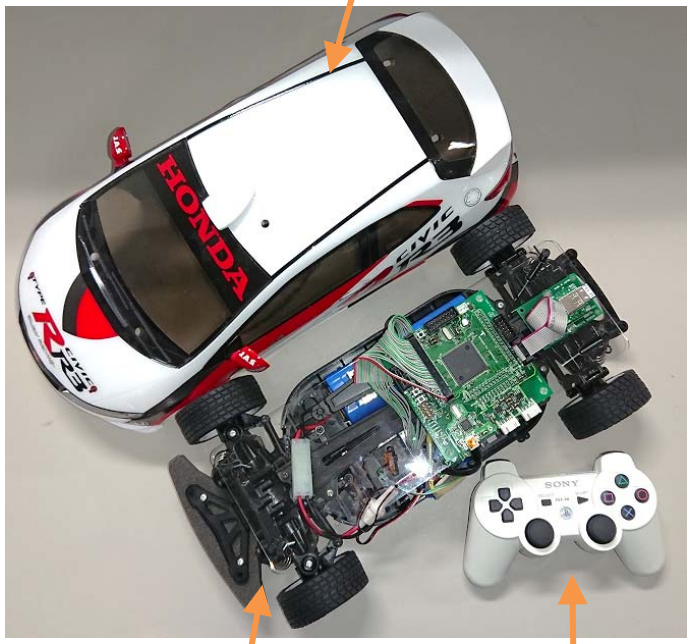
システム全体イメージ

- PS3コントローラで操作
- RH850/F1H(デュアルコア)を利用
- TOPPERS/ATK2-SC1-MC上で動作
- ブラシレスモータ制御はクロスレイヤ設計手法によってモデルから自動生成された並列タスクで実行



ハードウェア

ボディ
（ヘッドライト，ブレーキランプ，ブザー等が搭載）

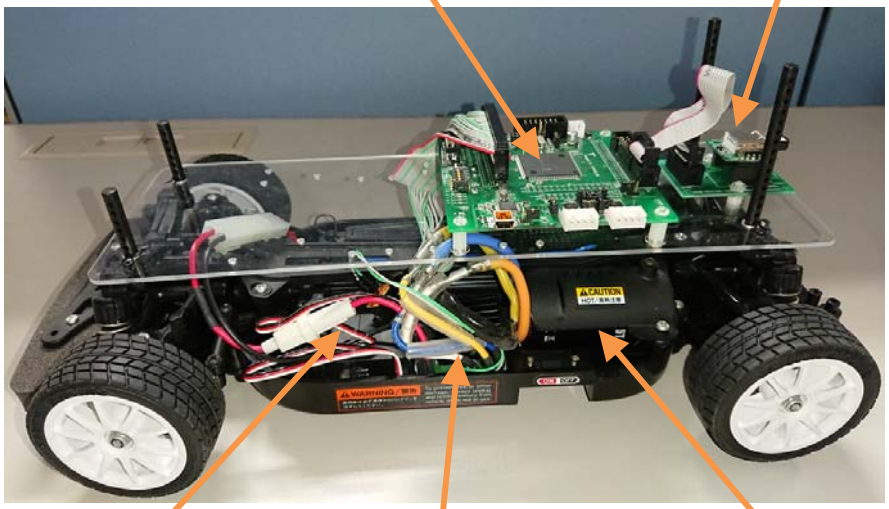


本体

PS3コントローラ

F1Hボード
（HSBRH850F1H176）

Bluetoothモジュール
（SBDBT5V）



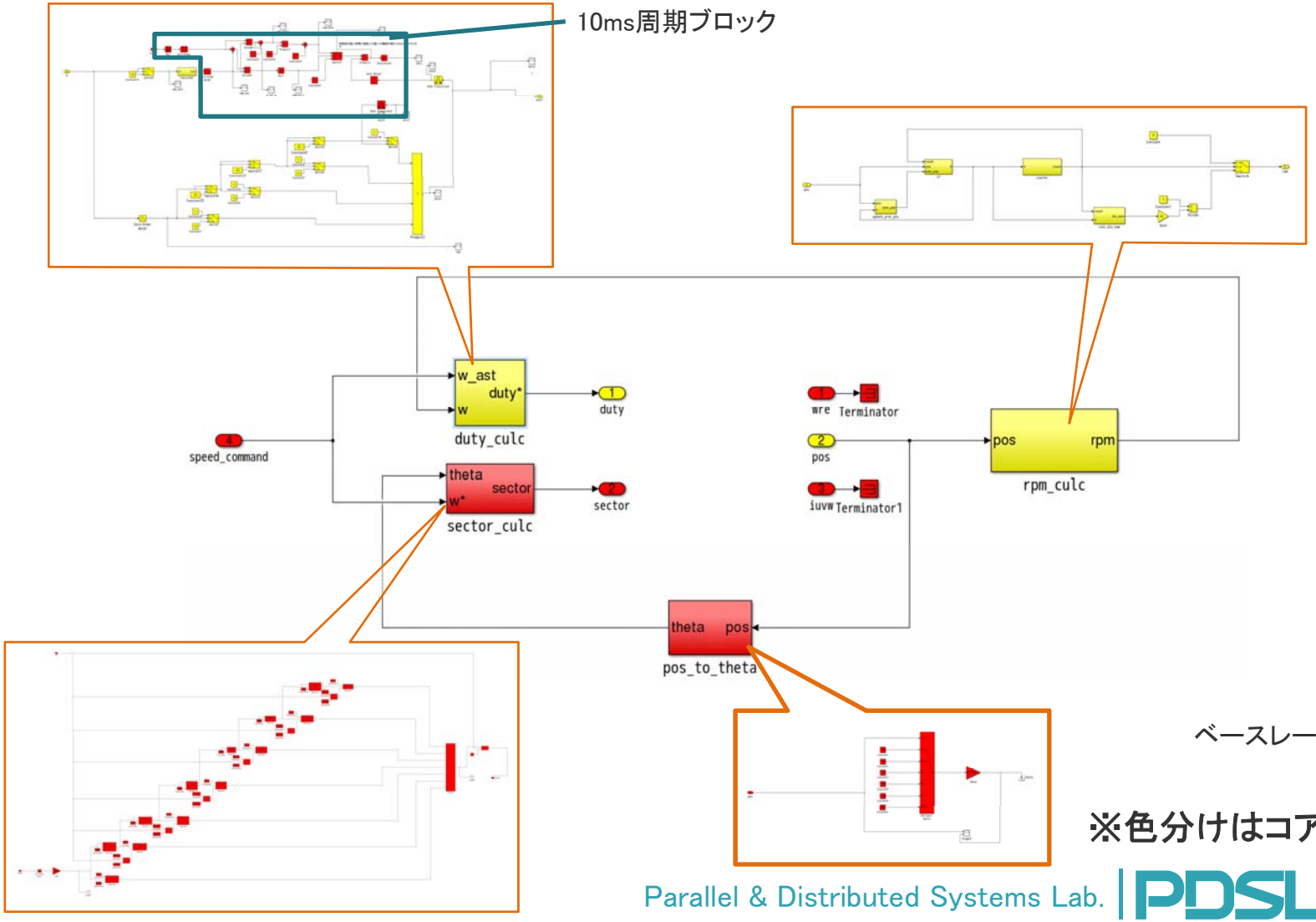
サーボモータ

組込用小型
モータドライバボード

ブラシレスモータ

ブラシレスモータ制御モデル

- クロスレイヤ設計手法を用いて，制御設計および自動並列化したモデル

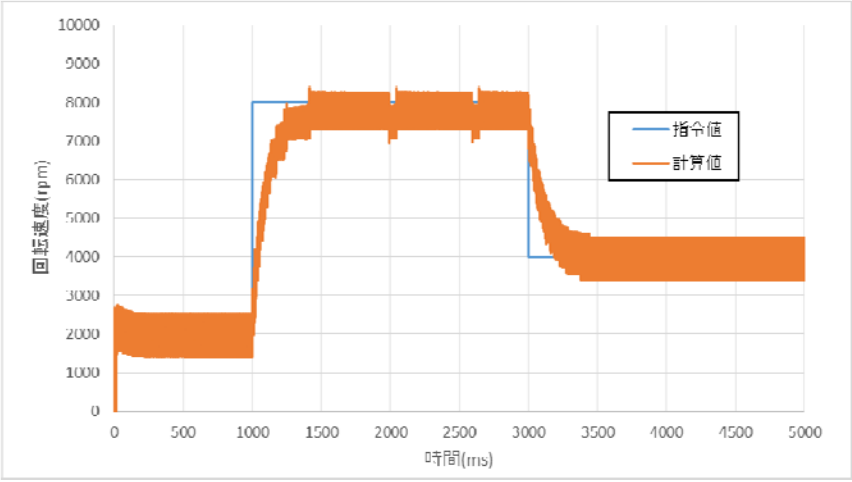


ベースレート：100us

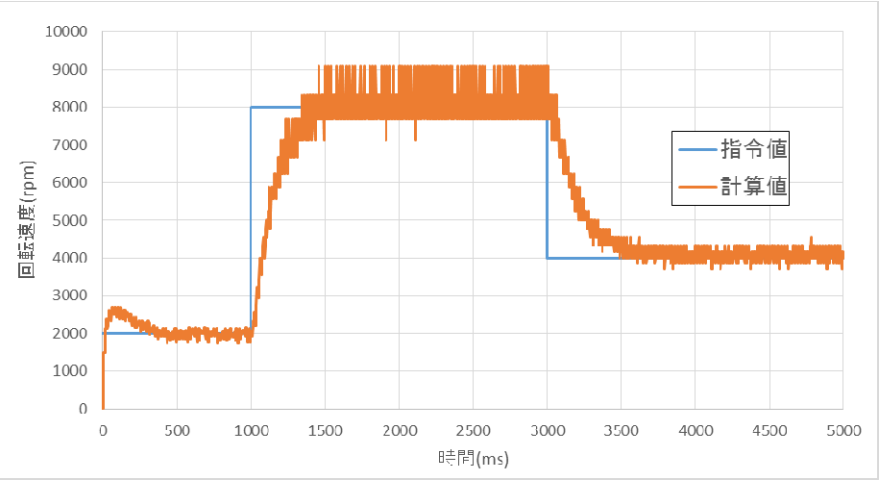
※色分けはコア割当を表す

並列コードの性能評価

- 出力比較



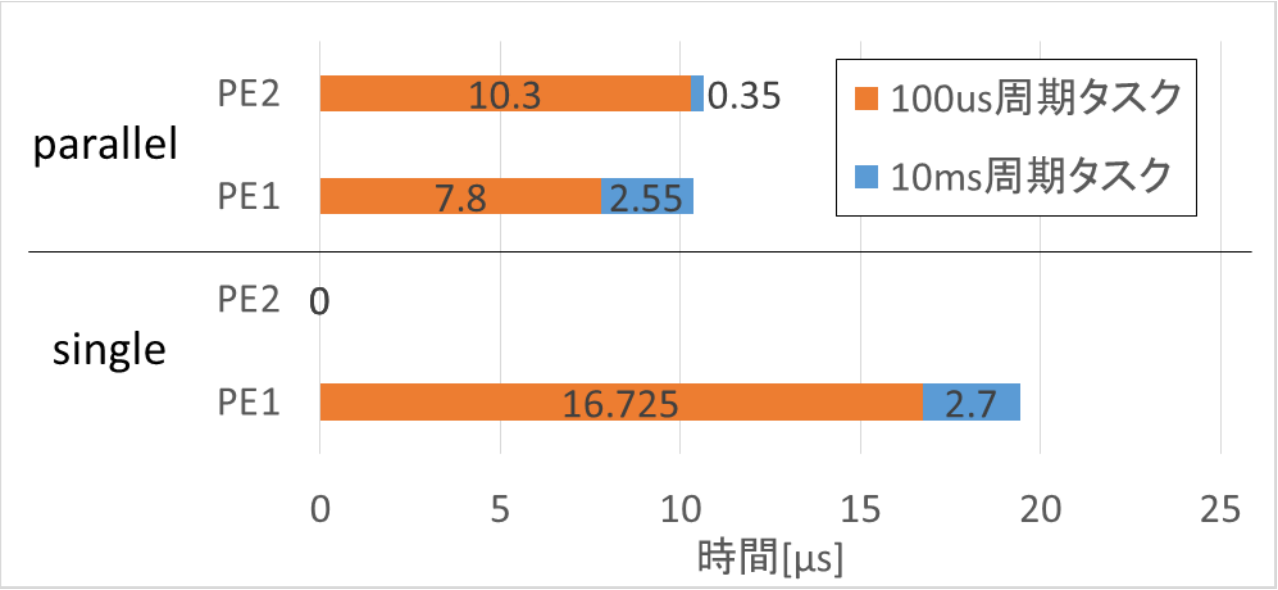
シミュレーション結果



実際の結果

並列性能比較

- 実行時間



- 100us周期タスク　1.62倍
- 100us周期タスク + 10ms周期タスク　1.82倍

アウトライン

- モデルベース開発（MBD）とクロスレイヤ設計

- クロスレイヤ設計に必要な技術

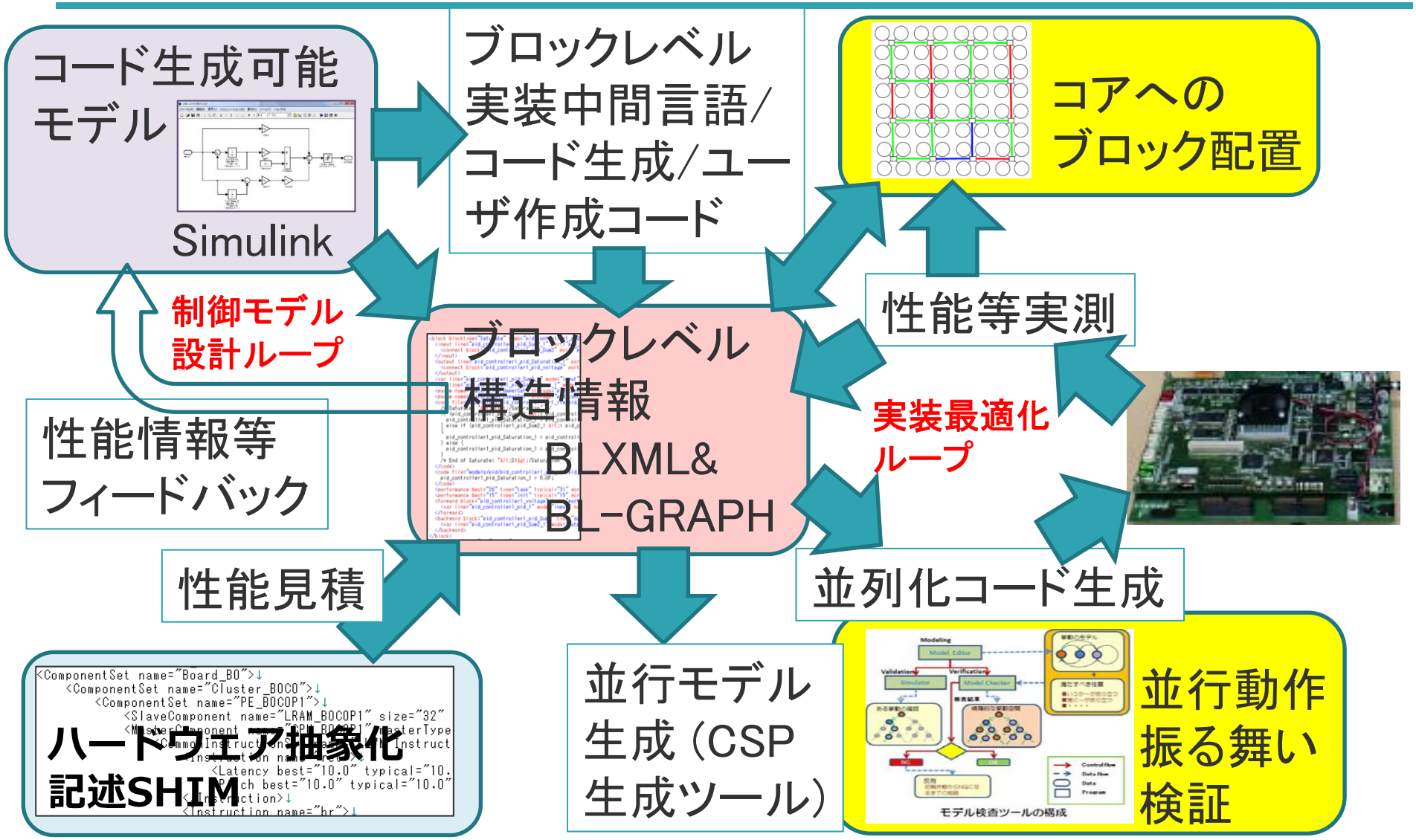
クロスレイヤ設計に必要な技術

- 性能見積(次の講演で)
  - ハードウェア抽象化記述SHIMにより処理時間見積
- Simulinkブロックのコアマッピング
  - 負荷分散，通信最小化を意識したコア割当を自動算出
- 並列コード生成
  - コア割当をもとに並列実行可能なコードを自動生成
- 並列動作検証
  - デッドロック，読み書き順序逆転等の並列動作による問題発生を検出
- 可視化
  - コア割当をモデルに反映させ，更なる高並列制御設計につなげる

クロスレイヤ設計に必要な技術

- 性能見積もり(次の講演で)
  - ハードウェア抽象化記述SHIMにより処理時間見積
- Simulinkブロックのコアマッピング
  - 負荷分散，通信最小化を意識したコア割当を自動算出
- 並列コード生成
  - コア割当をもとに並列実行可能なコードを自動生成
- 並列動作検証
  - デッドロック，読み書き順序逆転等の並列動作による問題発生を検出
- 可視化
  - コア割当をモデルに反映させ，更なる高並列制御設計につなげる

モデルベース並列化(MBP)設計検証フロー

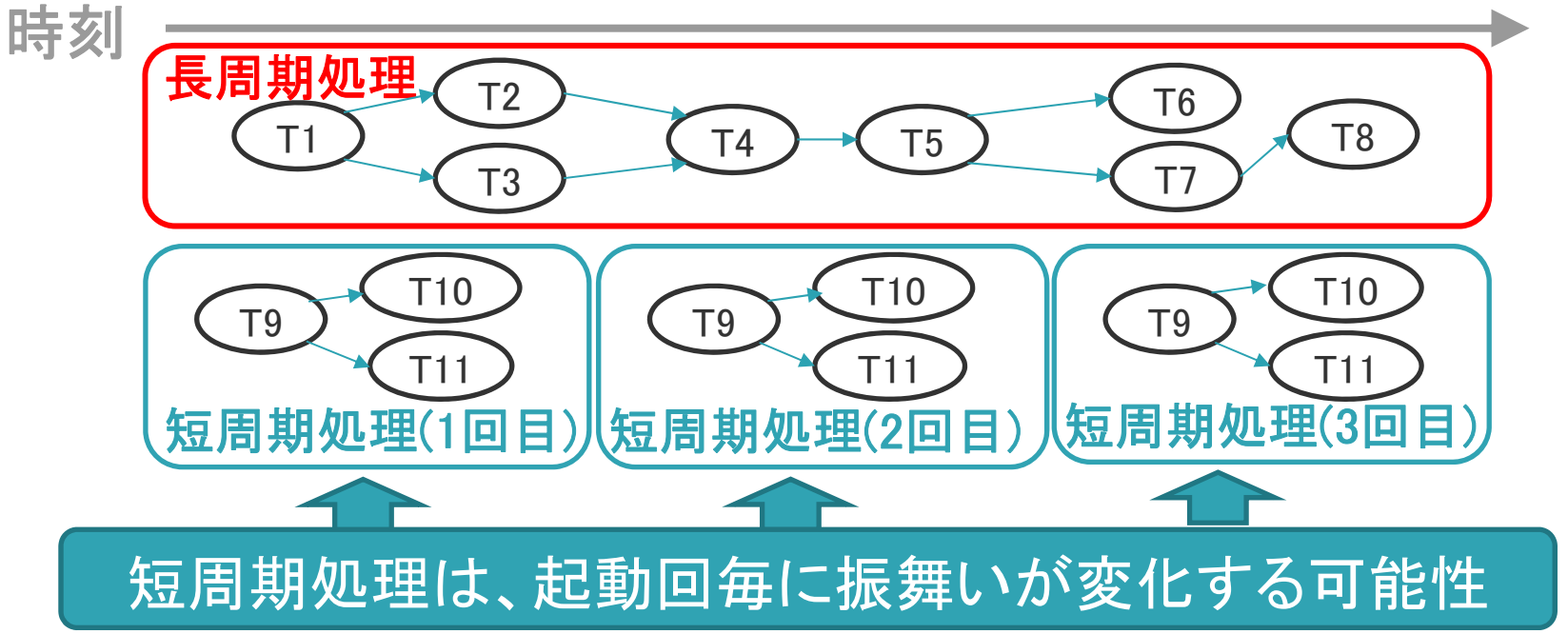


最近の成果概要

- 2017年4月 eSOL社よりeMBP製品発表
- 並列化アルゴリズムが2017年度情報処理学会コンピュータサイエンス領域奨励賞受賞
- 機能拡張
  - 性能評価強化（マルチレート）
    - コア・制御周期ごとにデッドラインミス、マージン評価
  - 性能ヘテロジニアス・アーキテクチャ対応
    - 性能ヘテロ：同一ISA、コアごとに性能が異なるアーキテクチャ
  - ユーザ指定強化
    - ブロックのグループ化（グループ単位でコア割当）
    - ブロックの割当コア指定
    - Atomic Subsystem単位並列化
  - OS連携強化

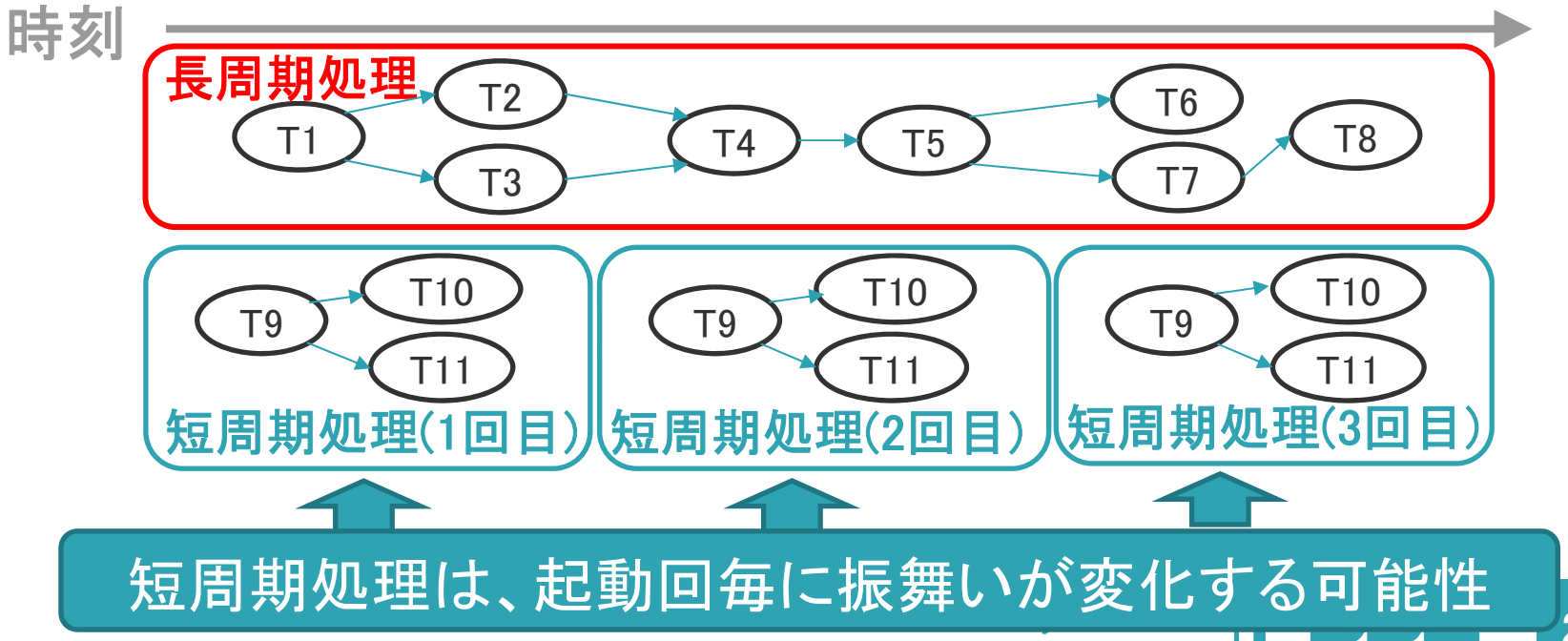
マルチレート制御における性能指標

- マルチレート制御における組み合わせの問題
  - 長い周期の処理が1回動作する間に、短い周期の処理が複数回動作する
  - 短い周期のプログラムは、起動タイミングによって、異なるタスク(長周期)と同時に実行し、振舞いが変化

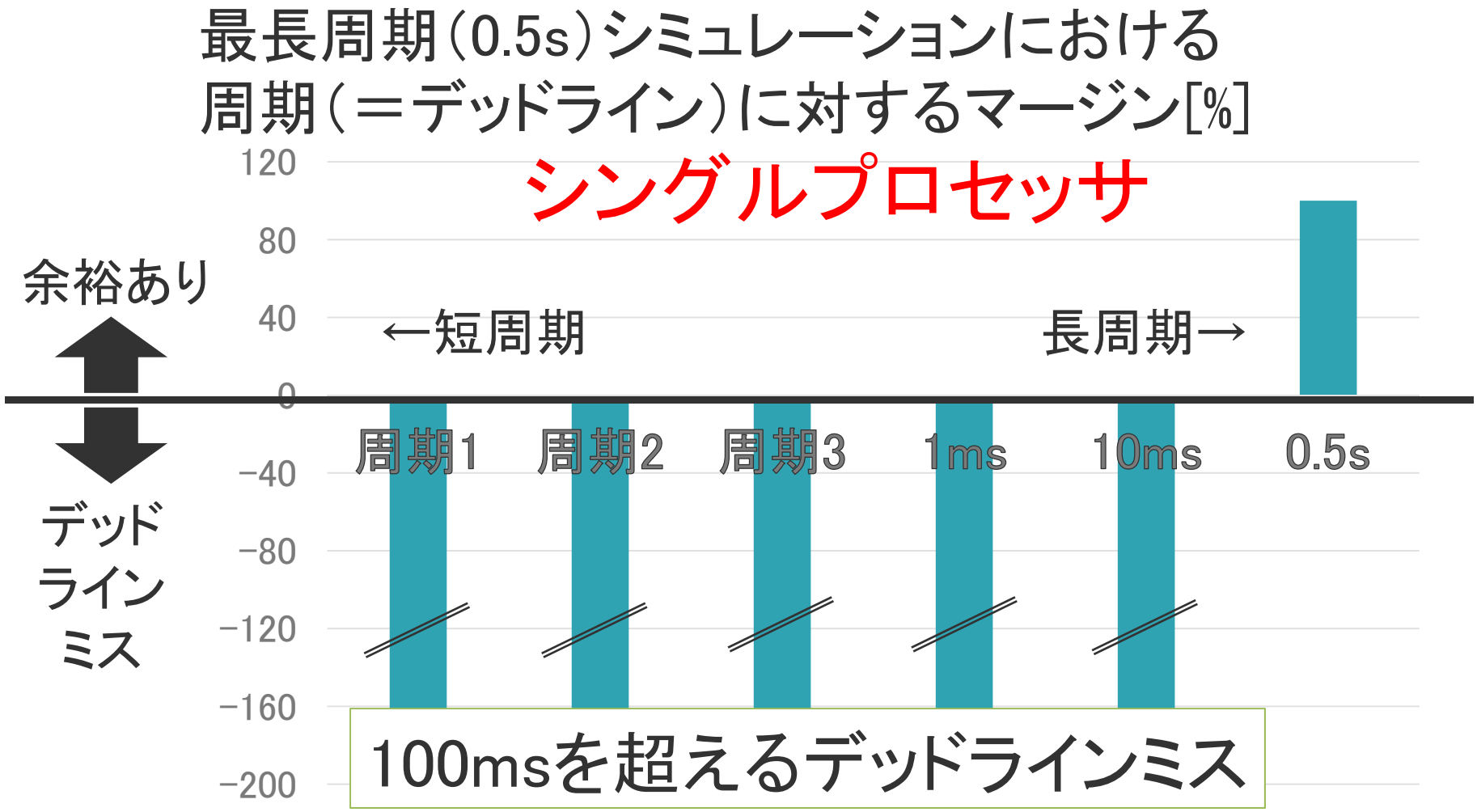


性能評価強化(マルチレート)

- 長周期時間だけモデルレベル高速シミュレーション
  - 各周期ごとにデッドラインに対するマージンのうち、最も悪いものを選択
  - マージンがプラス → システムが要求を満たし、成立
  - どれだけマージンがあるか? → マージンの変化が性能向上率と同義

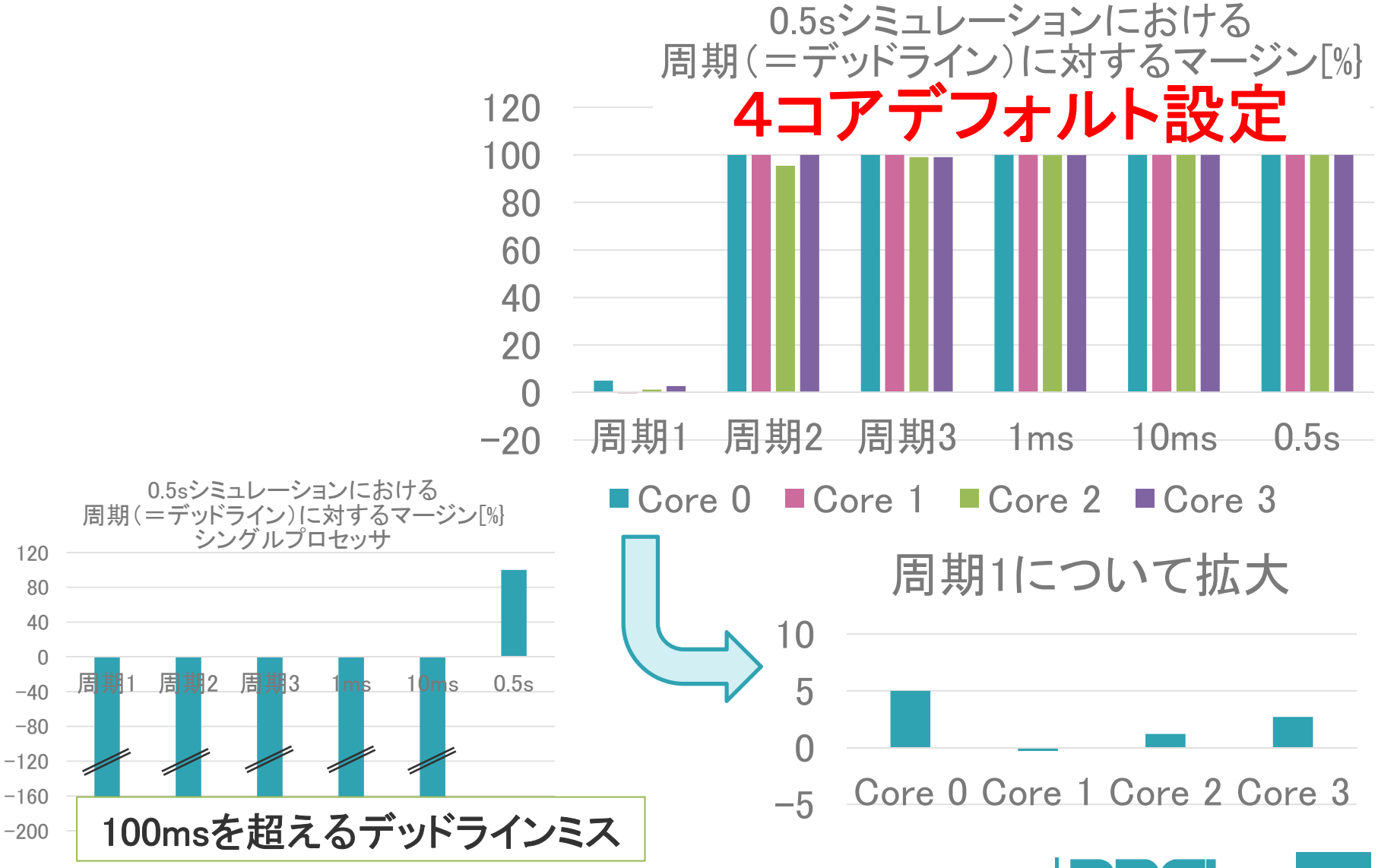


ハイブリッドモデルでのシミュレーション結果 (1)



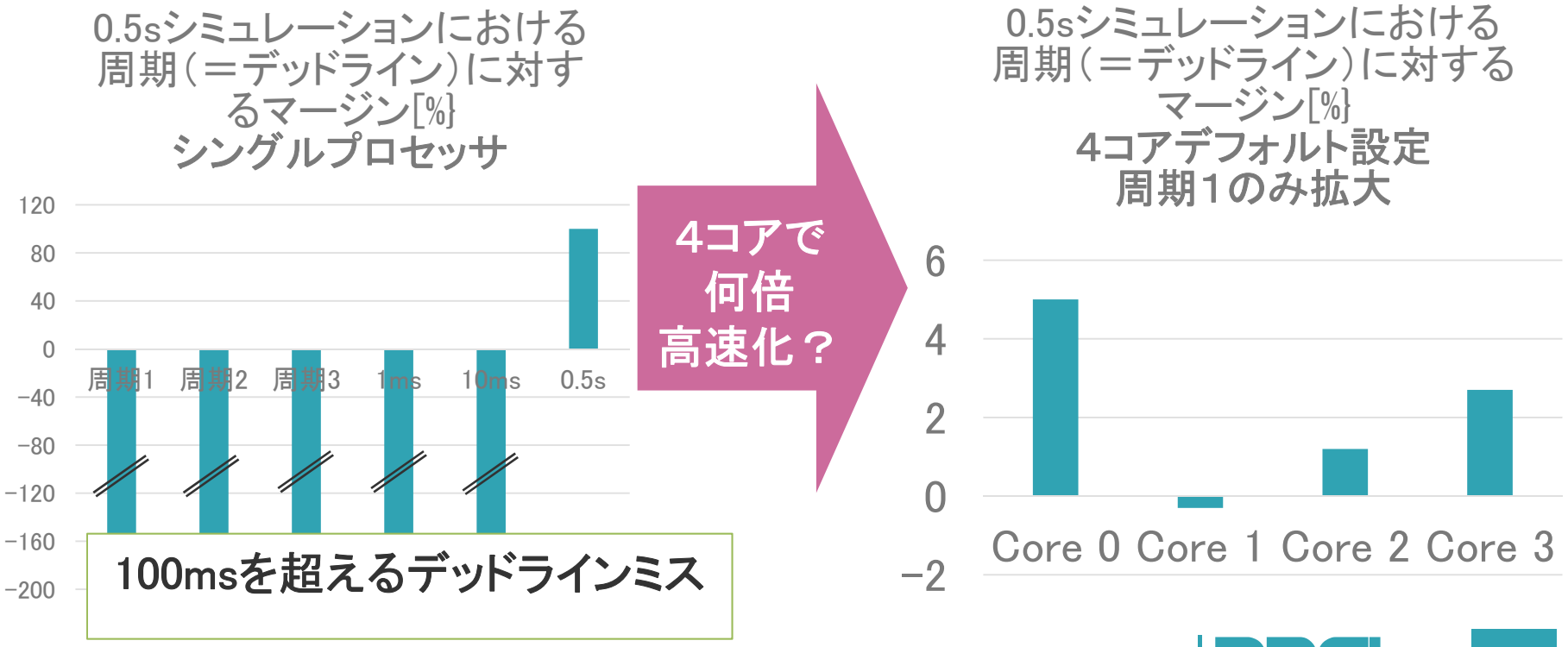
(ブロック数:506，0.5秒シミュレーションで約1億ブロック評価
シミュレーション実行時間30秒)

ハイブリッドモデルでのシミュレーション結果 (2)



マルチレート制御における並列化性能向上

- 性能向上率よりデッドラインに対するマージンが重要
  - デッドラインを割っていない ≒ システムが成立
  - 単に「NコアでX倍」という話ではない

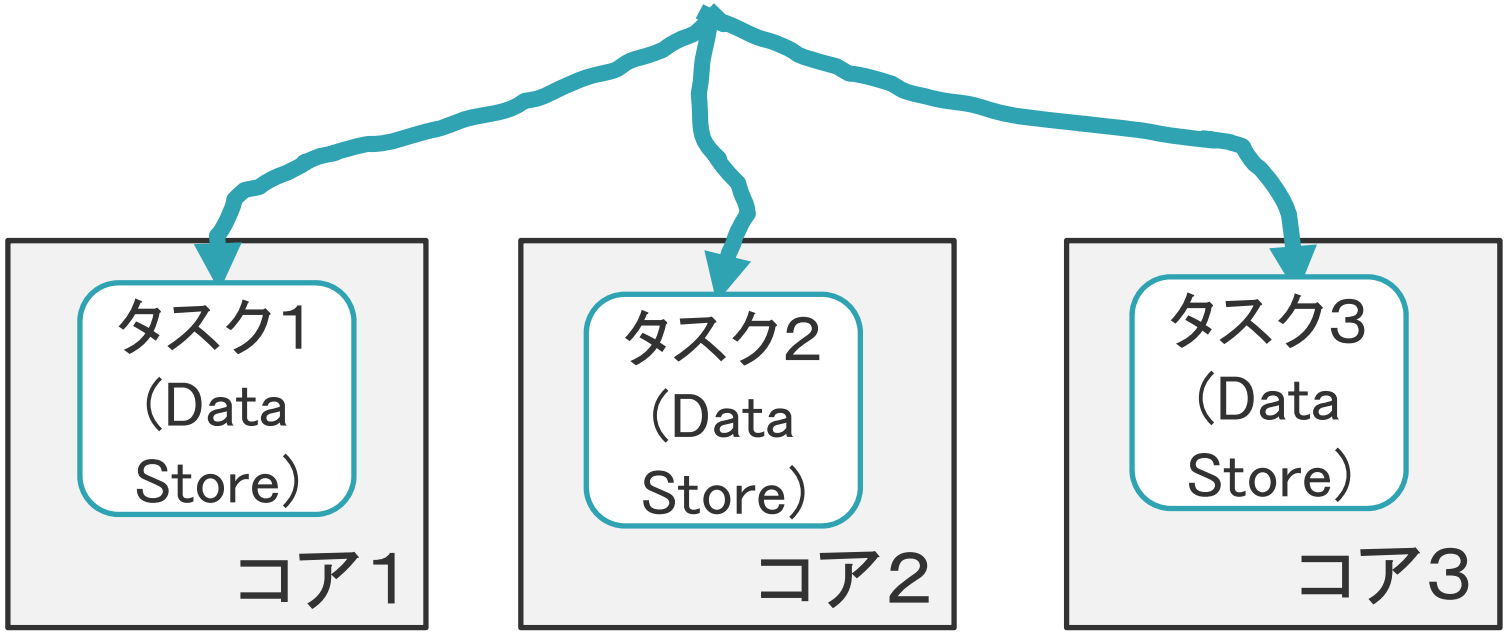


クロスレイヤ設計に必要な技術

- 性能見積もり（次の講演で）
  - ハードウェア抽象化記述SHIMにより処理時間見積
- コアマッピング
  - 負荷分散，通信最小化を意識したコア割当を自動算出
- 並列コード生成
  - コア割当をもとに並列実行可能なコードを自動生成
- **並列動作検証**
  - **デッドロック，読み書き順序逆転等の並列動作による問題発生を検出（CSPによるモデル検査、産総研と共同研究）**
- 可視化
  - コア割当をモデルに反映させ，更なる高並列制御設計につなげる

並列化コードに対する課題

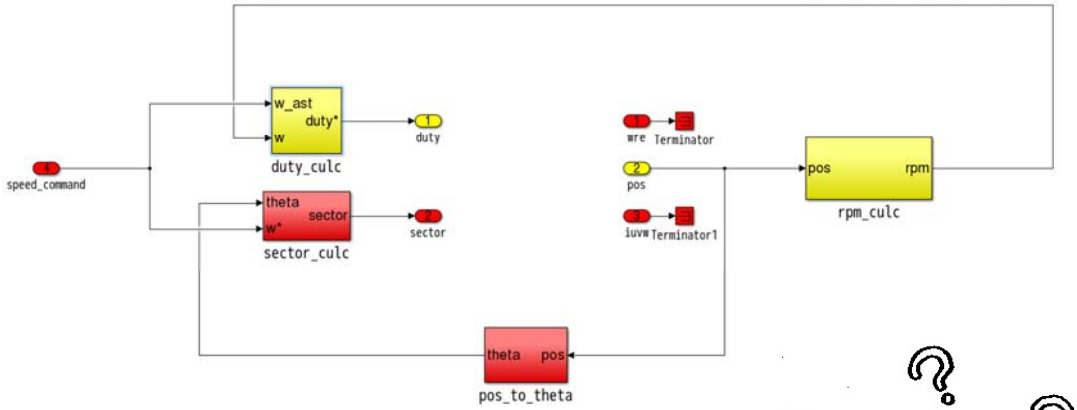
- 並列・並行実行により、タイミングに関する自由度が指数関数的に増大



デッドロック？　読み書き順序逆転？

自動コード生成ツールに対する課題

- もとのSimulinkモデルと並列化コードは等価なのか？



```
/*
 *  メインタスク
 *
 *  ユーザコマンドの受信と，コマンドごとの処理実行
 */
TASK(MainTask)
{
        uint8           command;
        uint8           task_no;
        uint32          i;
        CoreIdType      coreid = GetCoreID();

        TickType        val = 0U;
        TickType        eval = 0U;

        syslog(LOG_EMERG, "activate MainTask! @ core%d", coreid);
        /*
         *  タスク番号・コマンドバッファ初期化
         */
        task_no = (uint8) (0);
        for (i = 0U; i < (sizeof(command_tbl) / sizeof(command_tbl[0])); i++) {
                command_tbl[i] = 0U;
        }

        /*
         *  MainCycArm0, MainCycArm1を周期アラームとして設定
         */
        SetRelAlarm(MainCycArm0, TICK_FOR_10MS, TICK_FOR_10MS);
        SetRelAlarm(MainCycArm1, TICK_FOR_10MS, TICK_FOR_10MS);
```

検証項目

a. 元のSimulinkモデルから抽出された並列性によって実機上で問題が起こらないこと

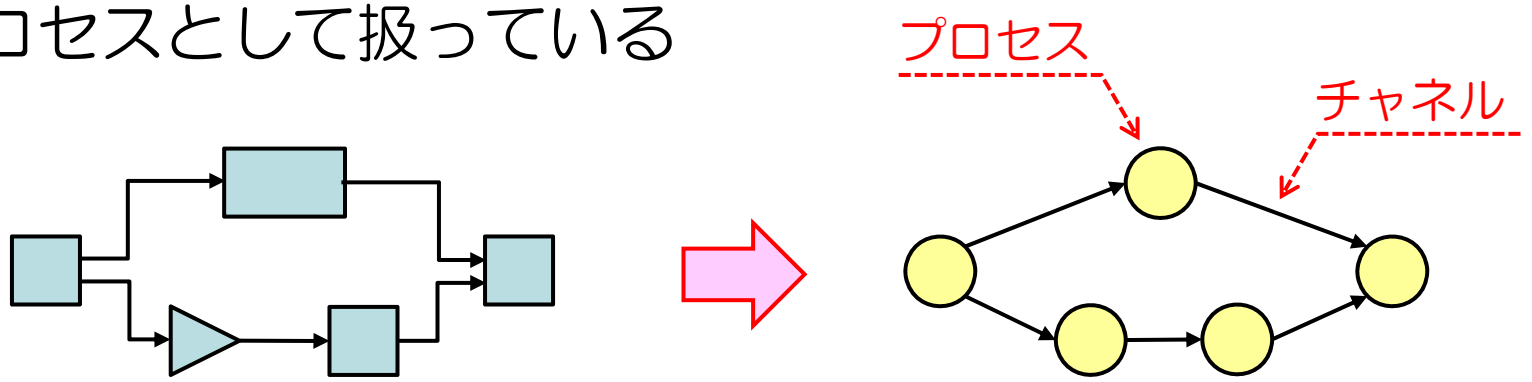
- 並列化コードを用い、従来のback-to-backテストを実施
  （論理的正当性の検証、ここでは省略）
- 並列化コードをCSPでモデル化し、並列化が原因で発生する問題を網羅的に検証

b. 元のSimulinkモデルと（抽出された並列性による自由度を除き）等価な並列化コードが生成されること

- 並列化アルゴリズムの明確化（形式化）
  （CSPによるアルゴリズムの記述を作成）
- 並列化アルゴリズムの正当性を証明
  （並列化によって制御ブロック間の依存関係が崩れないことを証明）

CSPとは

- CSP *1：並行処理を記述する仕様記述言語とその解析方法一式
  - CSPではプロセスをチャネルで接続して並行システムを表現する
  - CSPは機能安全の国際規格IEC61508で推奨されている形式手法のひとつ
    - IEC61508ではHighly Recommended,
    - ISO26262(2011)ではFormal VerificationがRecommended
  - CSPに基づく検証ツール（PAT, FDR等）やプログラミング言語（Go等）がある
- 現在、CSP生成ツールではSimulinkモデルのブロックをそれぞれプロセスとして扱っている

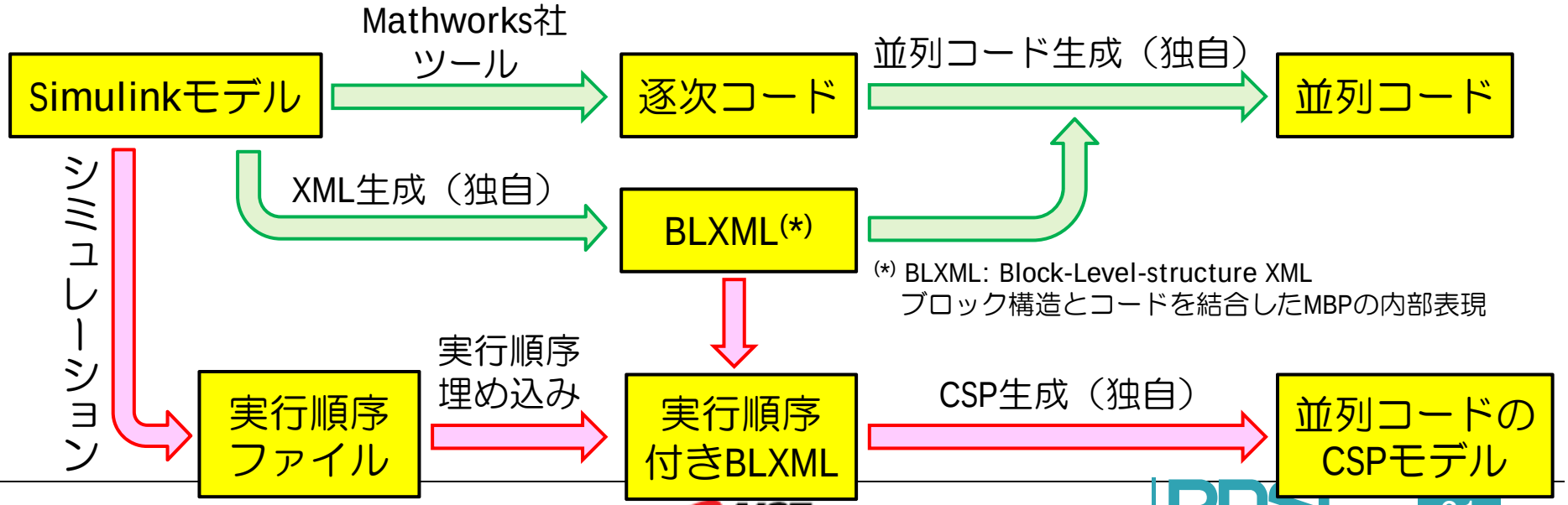


*1 CSP: Communicating Sequential Processes, C. A. R. Hoare, Oxford大学, 1978

a. 並列化が原因で発生する問題の検証

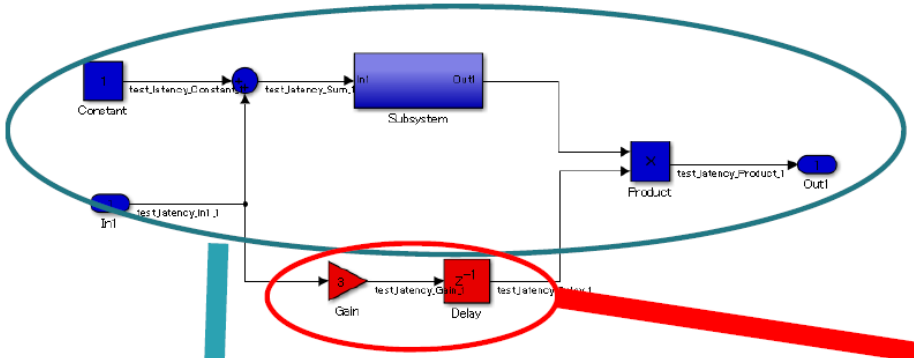
■ 生成されたCSPモデルを用いてモデル検査

- 機能要件
  - デッドロック：deadlock-freeを利用して検証
  - 読み書き順序逆転：LTL（線形時相論理）を利用した順序保証検証
- 非機能要件
  - デッドラインミス：時間情報付CSPモデルとdeadline制約で検証
    （deadline制約：deadline以内に終了しないとデッドロックする）



a. CSPモデルの生成例

※色ごとにコア割当て



ブロックプロセス

```
Test_latency_Delay() =
ch_test_latency_Delay_test_latency_Product!1 ->>
 test_latency_Delay ->>
 Wait[1];
 Skip;
```

コアプロセス

```
Core_0()=
Test_latency_Delay;
Test_latency_Gain;
Test_latency_Delay_recv;
Skip;
```

コアプロセス

```
Core_1()=
Test_latency_In1;
Test_latency_Constant;
Test_latency_Sum;
Test_latency_Subsystem_In1;
Test_latency_Subsystem_Saturation;
Test_latency_Subsystem_Out1;
Test_latency_Product;
Test_latency_Out1;
Skip;
```

モデル全体のプロセス+検証項目

```
System() =
Core_1 ||
Core_0;

System_deadline() = System deadline[4];

#assert System deadlockfree;
#assert System_deadline deadlockfree;
```

a. 階層分割によるCSPモデルの検証例

約10000ブロックのSimulinkモデルを分割することでの検証適用確認

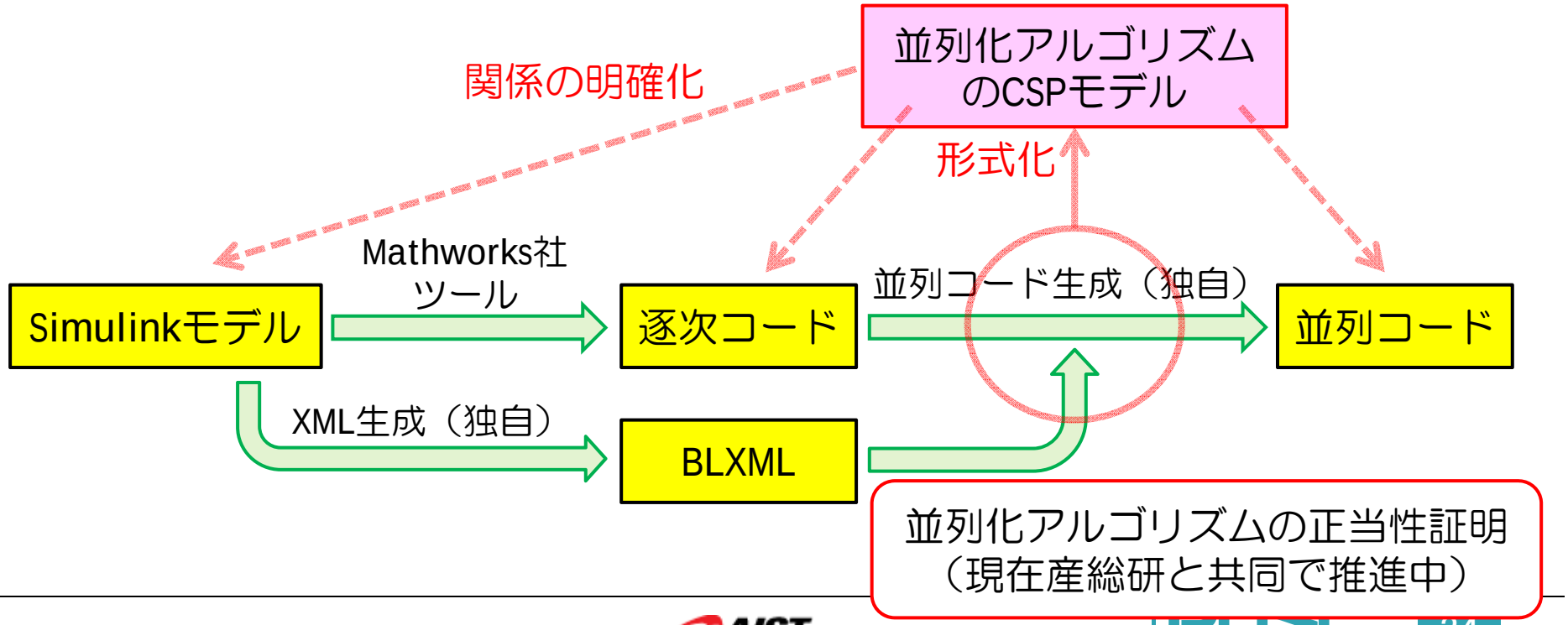
– 34個のCSPモデルに分割

CSP	The number of Block Process	Time(s)
Original	3370	Not Complete
1	28	0.01
2	27	0.01
3	66	0.07
4	27	0.01
5	272	55
⋮	⋮	⋮
34	27	0.01
total	3440	528.79

※一部抜粋

b. 並列化アルゴリズムの正当性の証明

- 並列化アルゴリズム自身をCSPで厳密に記述し、その正当性を証明する
  - 機能要件
    - 並列コードはSimulinkモデルのブロック間依存関係を満たすこと
    - 逐次コードと並列コードの出力が等しくなること

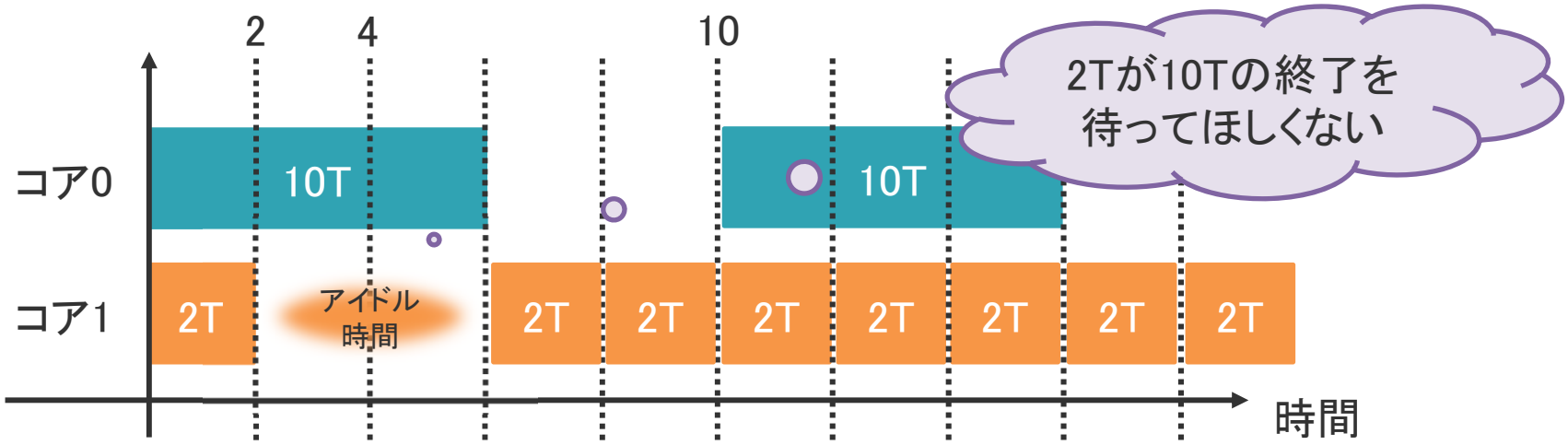


クロスレイヤ設計に必要な技術

- 性能見積もり(次の講演で)
  - ハードウェア抽象化記述SHIMにより処理時間見積
- コアマッピング
  - 負荷分散，通信最小化を意識したコア割当を自動算出
- 並列コード生成
  - コア割当をもとに並列実行可能なコードを自動生成
- 並列動作検証
  - デッドロック，読み書き順序逆転等の並列動作による問題発生を検出
- 可視化
  - コア割当をモデルに反映させ，更なる高並列制御設計につなげる

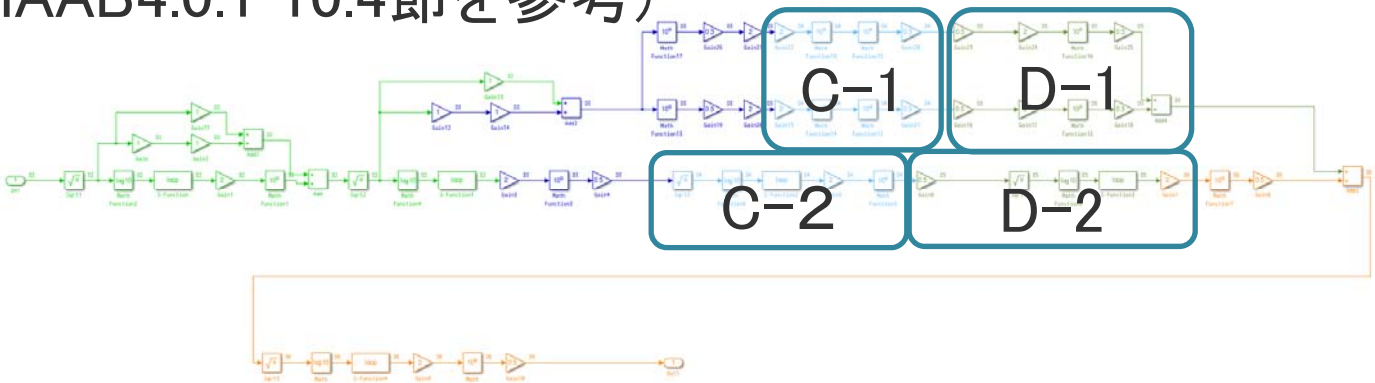
例：マルチレートコード生成の課題

- アプリケーションをOSの一つのタスクとして、
  アプリケーション内でスケジューリングしたい場合、
  シングルタスクモードでコード生成する
- シングルタスクモードのスケジューリング図
  （JMAAB4.0.1 10.4節を参考）
  – 2Tタスク実行の5回に1回、10Tタスクを実行

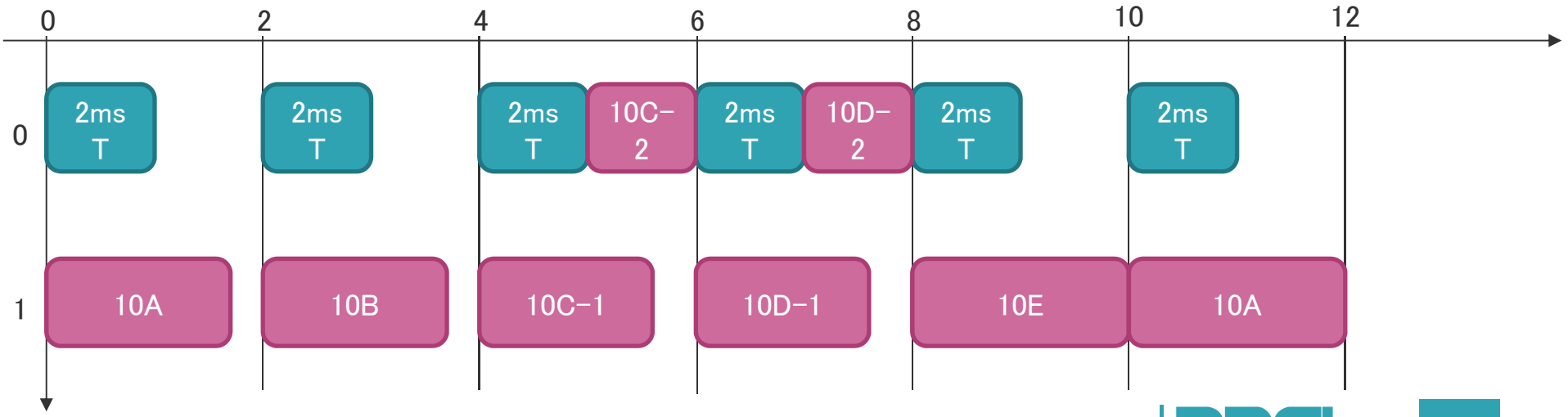


モデルレベルで解析、最適化

- 長周期を分割、短周期を混在してコア割当最適化
  (JMAAB4.0.1 10.4節を参考)



- スケジューリング結果



最適化後の実行アニメーション

まとめ

- モデルベース並列化(MBP)を用いたクロスレイヤ設計手法について述べた
  - RCカーのモータ設計を例題としてクロスレイヤ設計について紹介した
- 最近の研究成果について紹介した
  - コアマッピング
  - 検証
  - 可視化
- 成果は組込みマルチコアコンソーシアム内での試用を経て実用化を目指したい