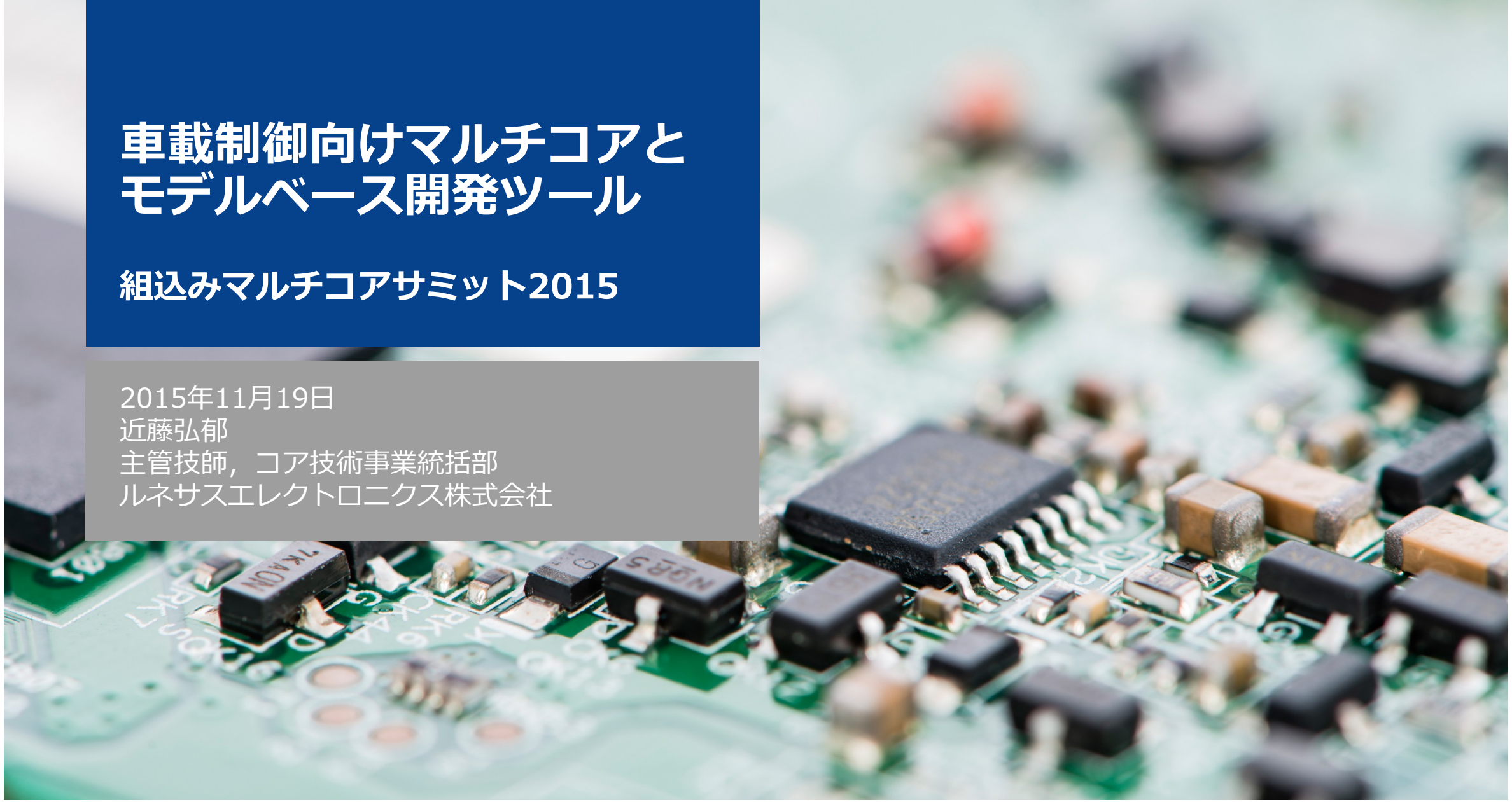
車載制御向けマルチコアとモデルベース開発ツール

組込みマルチコアサミット2015

2015年11月19日
近藤弘郁
主管技師，コア技術事業統括部
ルネサスエレクトロニクス株式会社



アジェンダ

車載制御向けマルチコアとモデルベース開発におけるマルチコア活用の開発環境について紹介する。この環境では、MATLAB/Simulinkモデルでブロック単位にコア割り当てを指定し、マルチコア向けコードを自動生成、MILS/PILSのBack-to-backテストや、ボトルネック箇所を判断が可能となる。

車載制御向けマルチコアへの取組み

クルマの低燃費化

エンジンが燃費向上の主役に

- これまで、HEV（電動化）、アイドリングストップ、CVT採用

今後の低燃費エンジン開発には、

- 高度化する制御演算を
- 高性能かつ低電力で処理

低燃費エンジンの実現に向けた課題

- 直噴リーンバーン、HCCI等の新燃焼方式導入
- 新方式を実現するアルゴリズムとＣＰＵ性能
- 各種センサからのフィードバック制御の高速化

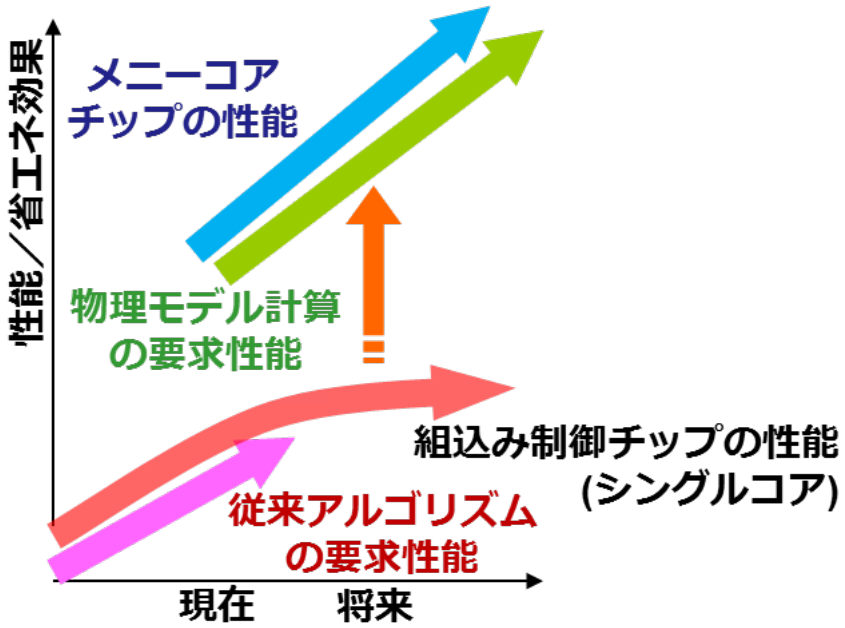
マイコンへの要求

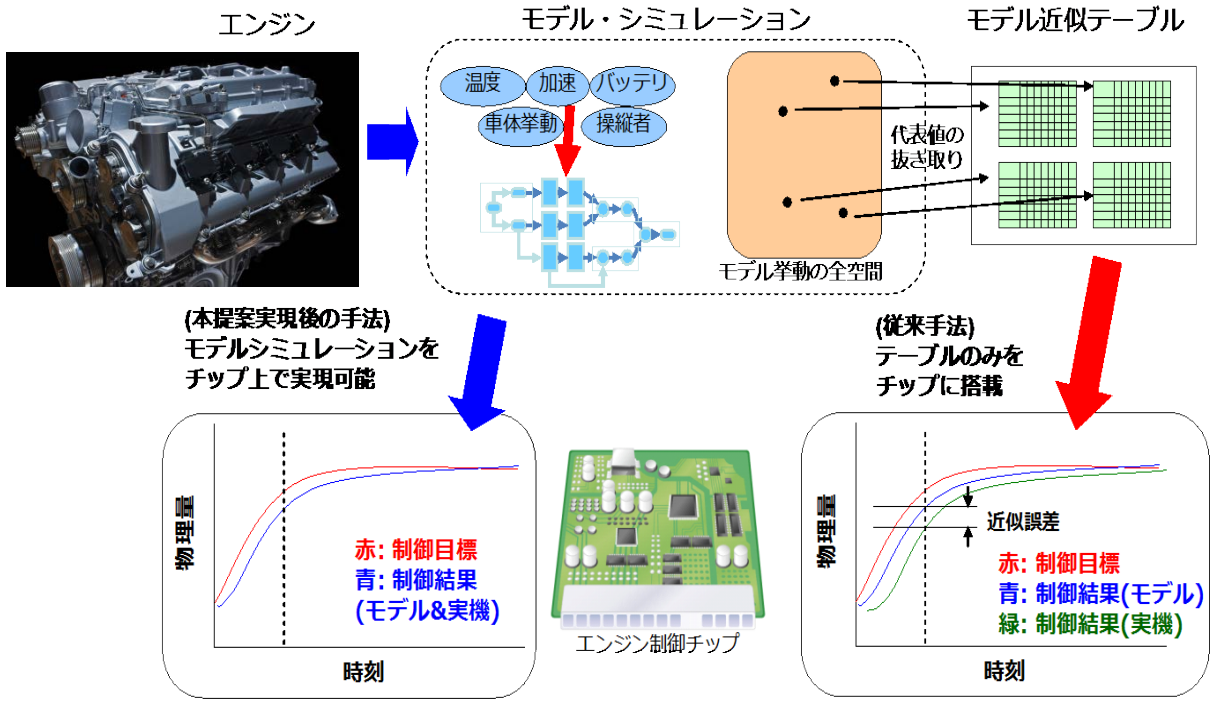
高性能なマイコン要求
（従来の３～５倍の性能）

限られた電力範囲での高性能化が必須

先進的な制御アルゴリズムの実用化
モデル予測制御など

先進的制御アルゴリズム（燃焼モデル計算等）を、組込み制御機器向けチップで実現





高精度な制御により燃費向上
- 先進的な制御の導入
  （物理モデルによる予測制御）
- マルチ・メニーコアにより実現

制御マルチコア実現の課題
並列性の抽出

並列性の抽出しやすさ

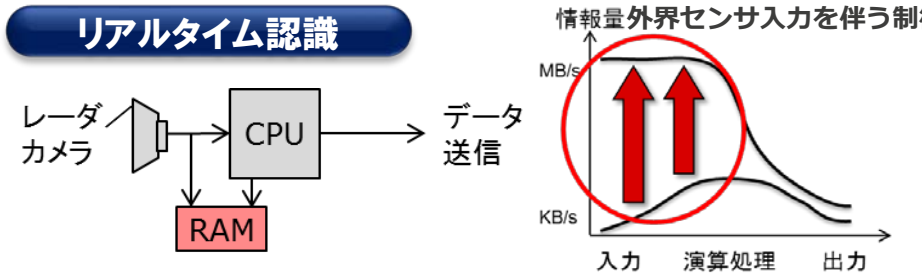
かんたん ⟶ 難しい

- ✓ データ・セットが小さく、再利用が多い
- ✓ 処理時間が短く、複数の処理が複雑な起動関係で動作
- ✓ リアルタイム性の保証が必要

ITシステム

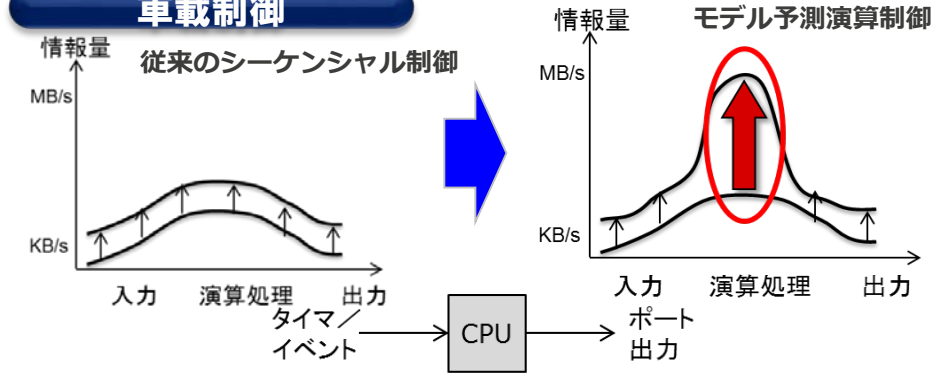
アプリの並列配置は比較的容易・実時間制御性要求が低い

リアルタイム認識



IOデバイスの割り込み処理などを分離することで
- 入力データの処理とその後の演算処理を並列化
- 高負荷演算処理の並列化

車載制御



制御アルゴリズムとして先進制御の導入により演算量の増加

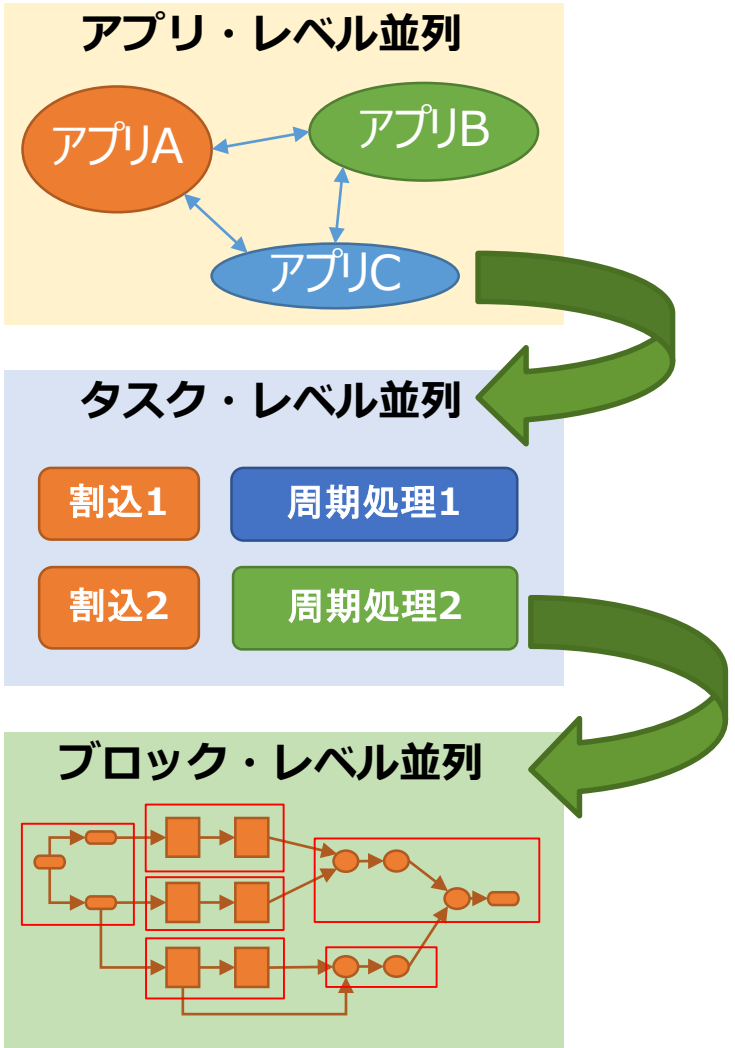
設計階層ごとの並列化手法
制御システムの抽象度毎に階層的に並列化

- **アプリケーション・レベル**　異なるアプリケーションを分散実行
  - 並列性の抽出: システム設計段階で分離済み
  - スケジュール: 処理負荷は多様だが、基本的には並行実行
- **タスク・レベル／粗粒度並列**　アプリに含まれる複数のタスクを分散実行
  - 並列性の抽出: アプリ設計段階で分離済み
  - スケジュール: 周期起動、イベント起動。OS等による管理
- **ブロック・レベル／細粒度並列**
  - 依存関係のないコード・ブロック，ループを並列化
  - クリティカルな処理のレイテンシを改善

  並列性の抽出: 処理内容依存、手動での抽出は困難

  スケジュール: 依存関係が複雑、手動での最適配置は困難

ツールによるS/W設計支援が必要不可欠



ブロック・レベル並列化の特徴

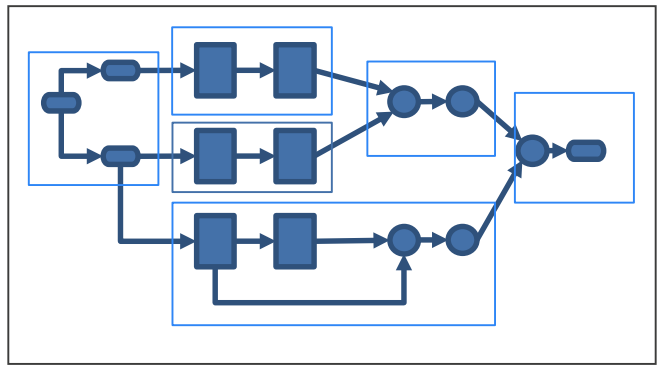
コード・ブロックのデータ・フローに応じて並列化

- 処理の先頭から、最後までは一気に実行可能
- 個々のブロック粒度はかなり小さいため、同期処理／データ交換のオーバーヘッドは最小化する必要あり
- 制御タスクで**抽出可能な並列度は低め (2~4以下)**

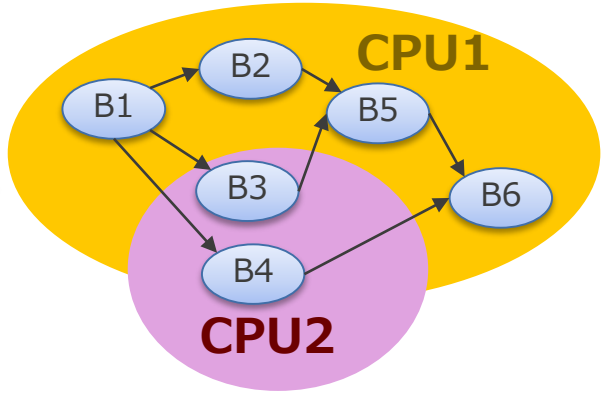
⇒ **高速共有(密結合)メモリが必須**

(同期処理、データ交換の双方に有効)

タスク内処理構造



並列化プラン



並列コード

```
…
func_B1();
set_flag(A);
func_B2();
wait_flag(B);
func_B5();
wait_flag(C);
func_B6();
…
```

```
…
wait_flag(A);
func_B3();
set_flag(B);
func_B4();
set_flag(C);
…
```

組込み制御向けマルチコアの課題

メモリ共有構造にハードウェア的な制限あり

- リアルタイム保証のため、キャッシュ等の平均性能改善技術は利用不可
- 全コアから等距離の配置は、性能面で問題あり

面積・消費電力の増大

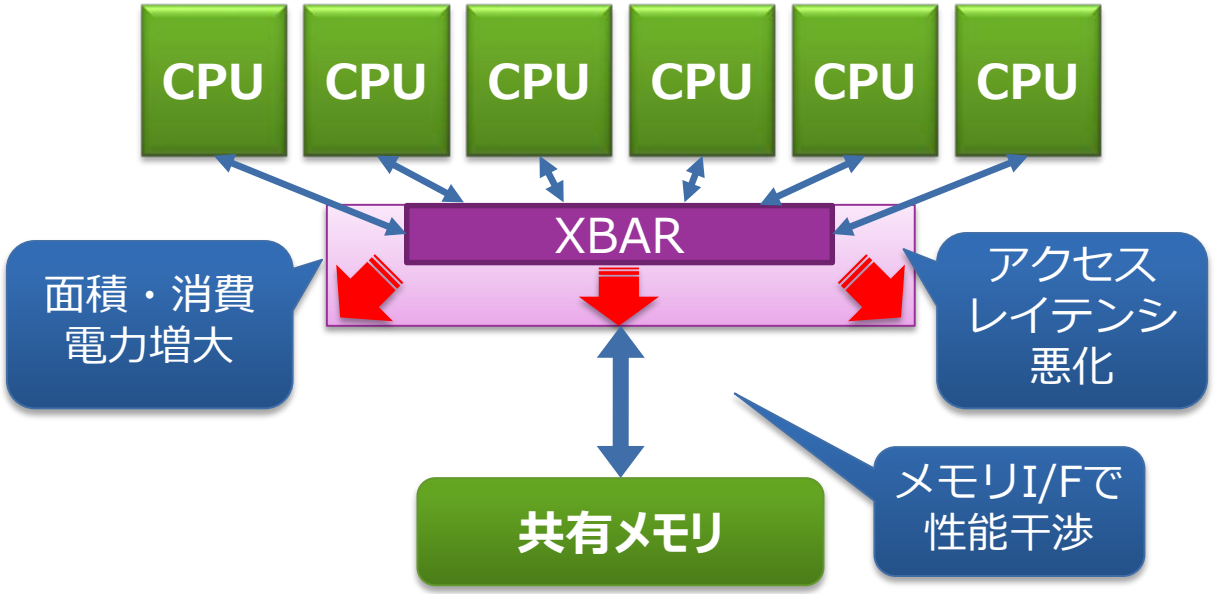
コスト／パッケージ／基板実装に影響

アクセス・レイテンシの悪化

密結合な並列動作時に、大きく性能劣化

CPU同士がメモリ・アクセスで干渉

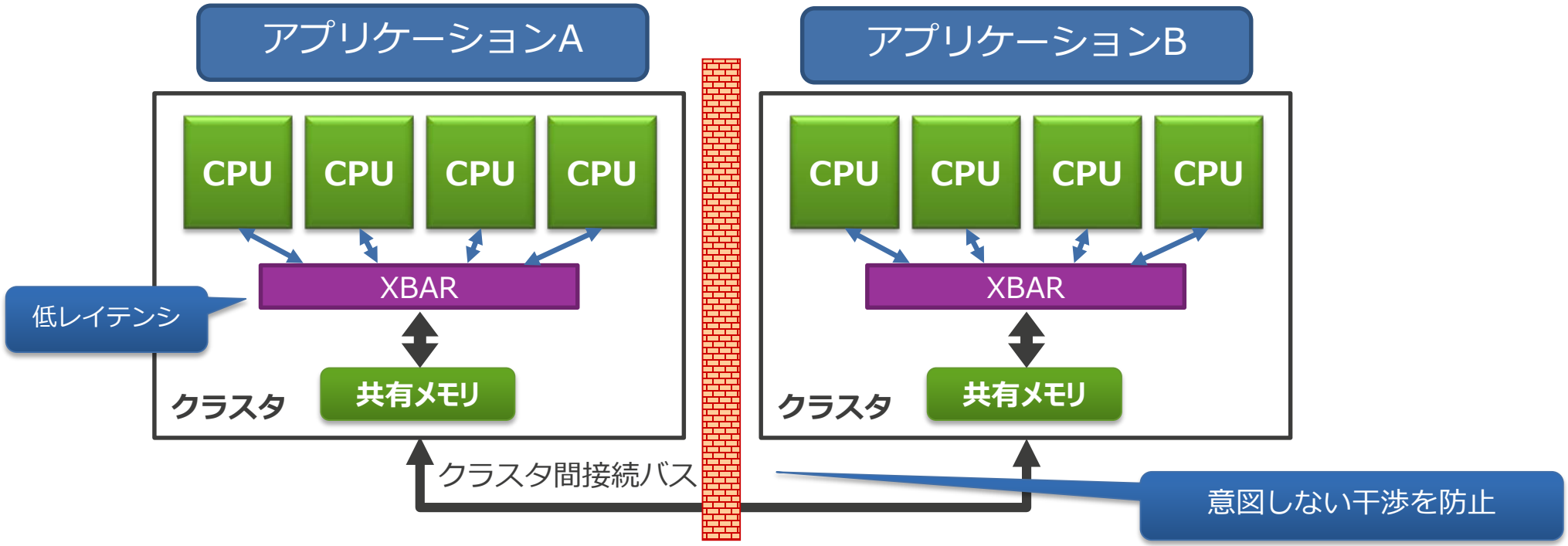
アプリで作りこんだタイミングが、
システム結合時に容易に崩れてしまう



クラスタ構造による高速化と低干渉化

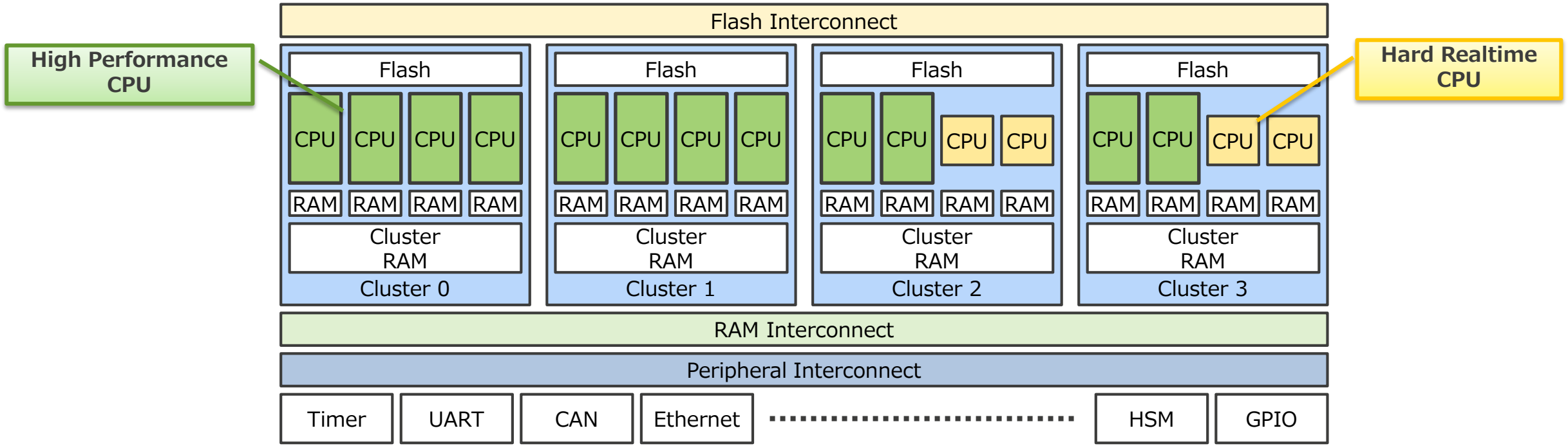
クラスタ共有メモリによる高性能化　　⇒ タスク／ブロック・レベルの並列化を高効率実行

クラスタ構成により低干渉化　　⇒ アプリケーション毎のタイミング検証結果を継承可能



制御マイコンにおけるメニーコア構造
クラスタ型ヘテロジニアス・アーキテクチャ

- アプリ間の非干渉化に適したクラスタ構成
- 高速なクラスタ内共有メモリ
- 制御特性に合わせたCPU選択
  - ◆高性能CPU & ハード・リアルタイムCPU

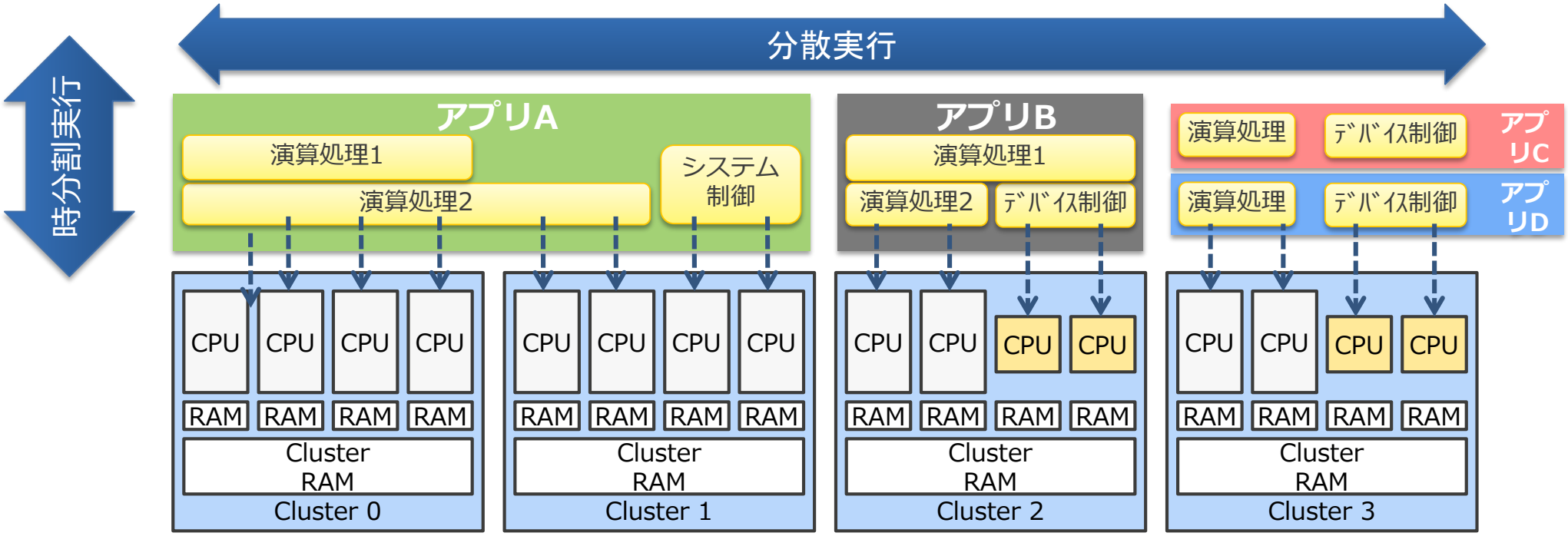


並列コード配置の一例

アプリケーションは、クラスタ構成を利用した分散実行と、OSによる時分割実行の使い分け

アプリケーション内は、タスク／ブロック・レベル並列を利用した密結合並列実行

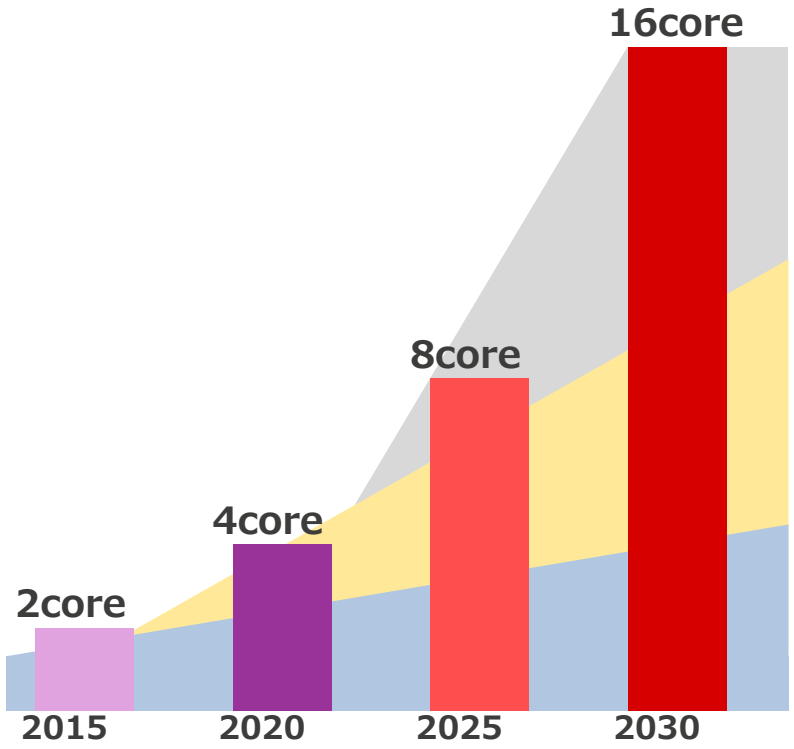
低ジッタ動作が必要なデバイス制御は、ハードリアルタイムCPUで実行



制御アプリケーションの並列性活用

2030年までに、3レベルの並列化を順次適用拡大

各レベルの並列性を総合して、高い並行動作を実現する

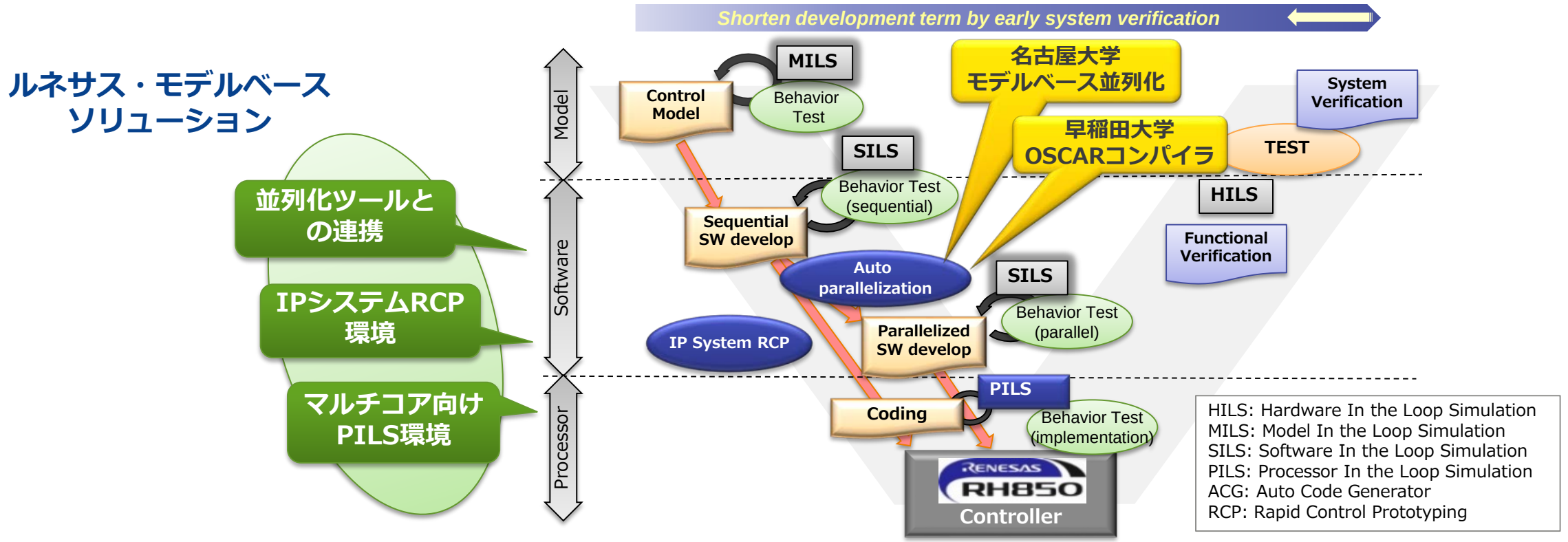


レベル	並列性抽出	スケジューリング
Block-level Parallelism	自動並列化 (並列化ツール)	自動スケジュール (OS or 並列化ツール)
Task-level Parallelism	設計者による分割 (モデリングツール)	設計者の意図を反映し、OSが行う
Application-level Parallelism	設計者による分割 (モデリングツール)	設計者が明示的に指定

組込み制御と
モデルベース設計ソリューション

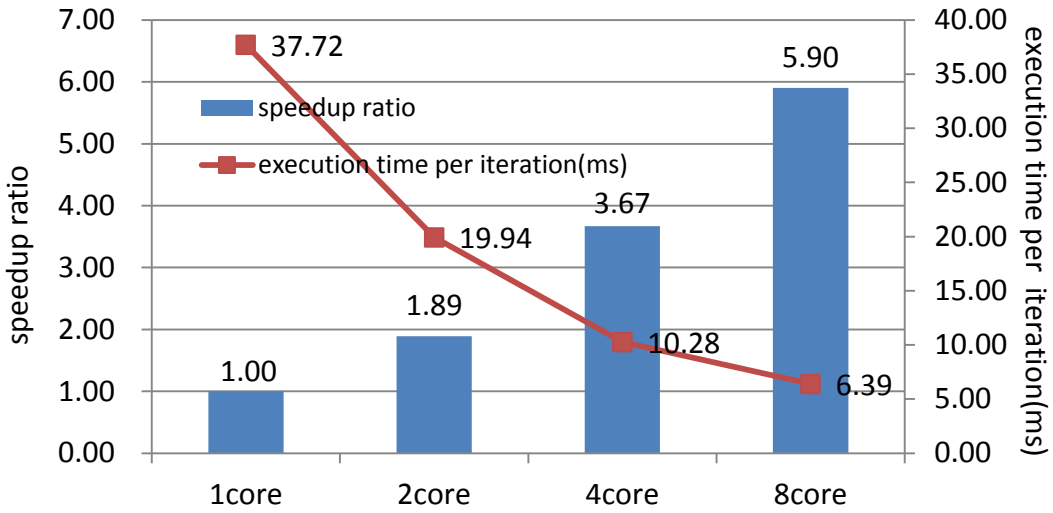
制御モデルベース開発へのルネサスの取組み

組込みソフト開発に対するお客様のニーズに応え、
パートナーツールと連携してモデルベース開発を支援

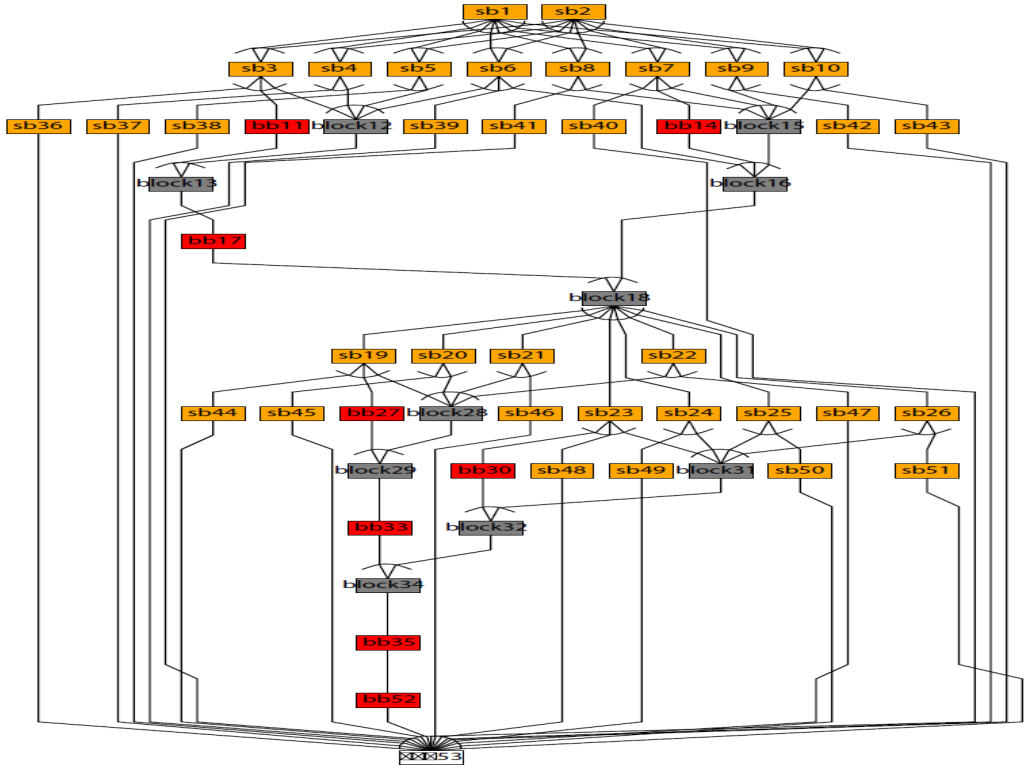


自動並列化研究の例
早稲田大学OSCARコンパイラ

- 早稲田大学笠原・木村研究室にて研究開発
  - 産学連携推進パビリオン ブース番号:UI-12 にて展示
- OSCARコンパイラ製品版開発中
- モデルベース設計ツールが生成したCソースを自動並列化



8コアで 5.9 倍の速度向上



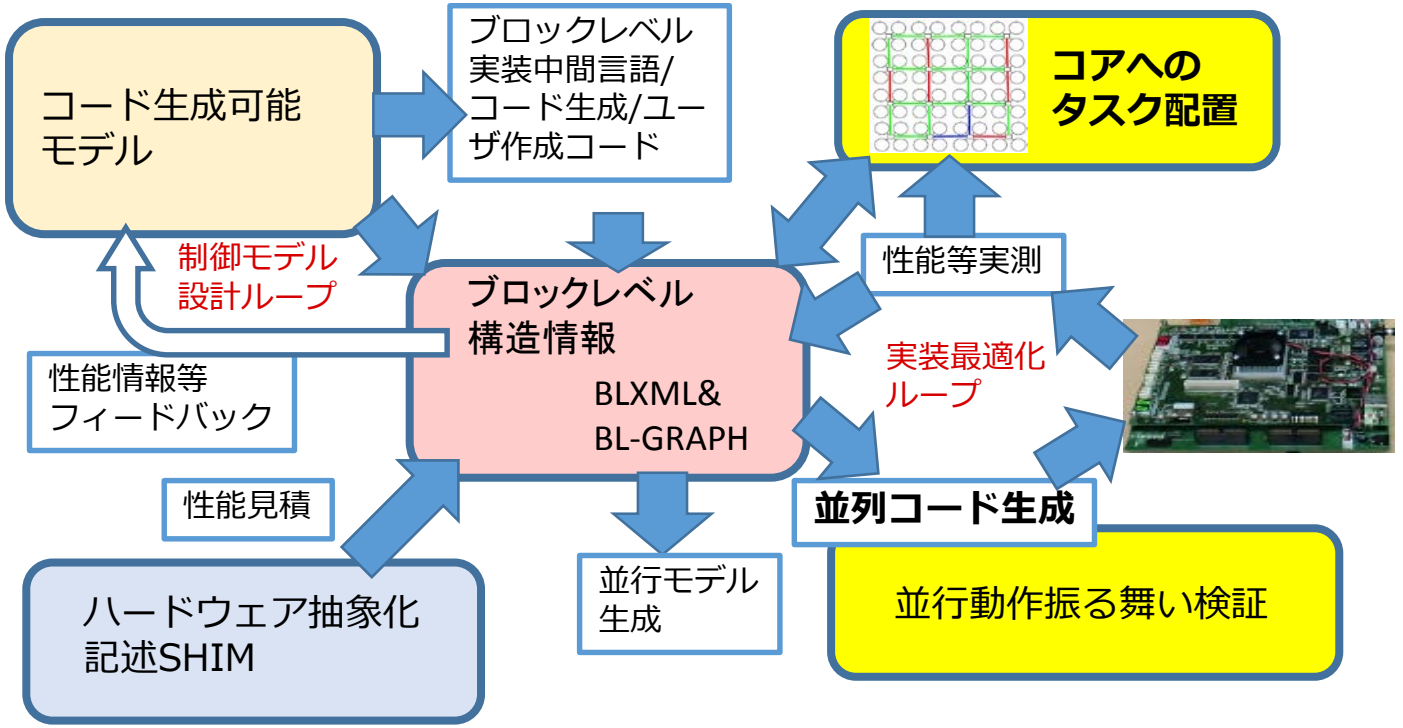
OSCARコンパイラ解析結果
自動生成されたマクロタスクグラフ

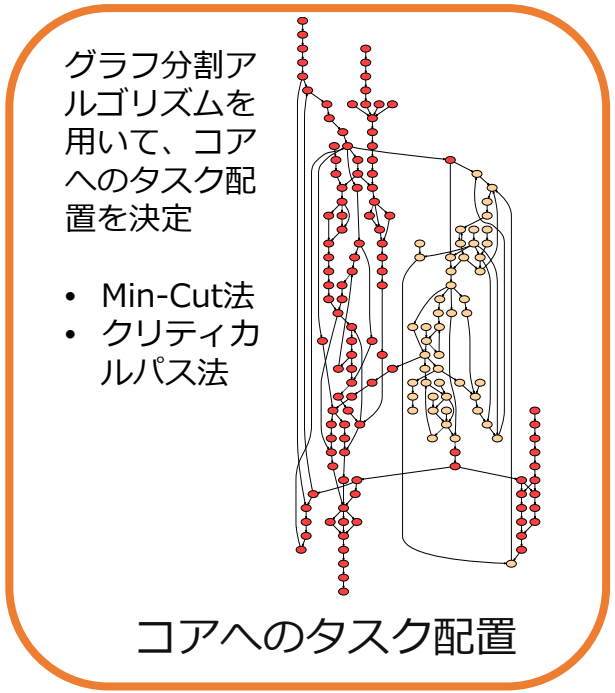
エンジン吸気制御モデルによる
RH850ベース・マルチコア並列化評価結果

資料提供：早稲田大学理工学術院 笠原研究室・木村研究室

自動並列化研究の例
名古屋大学モデルベース並列化

- 名古屋大学 枝廣研究室にて研究開発
  - ◆ 産学連携推進パビリオン ブース番号:UI-03 にて展示
- モデル記述上の情報から並列化
- ソースコードは分解統合せず、モデル構造との対応が維持された並列コードを生成

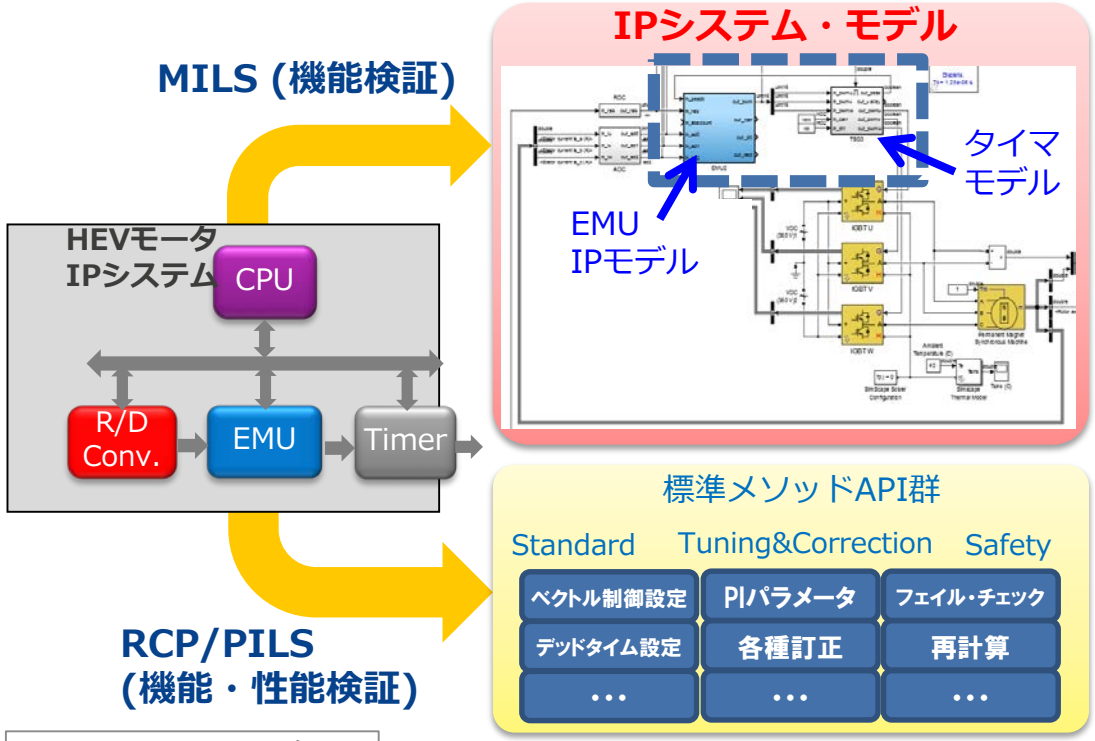




資料提供：名古屋大学大学院情報科学研究科 枝廣研究室

ルネサスの取組み例
RH850 IPシステムRCP環境

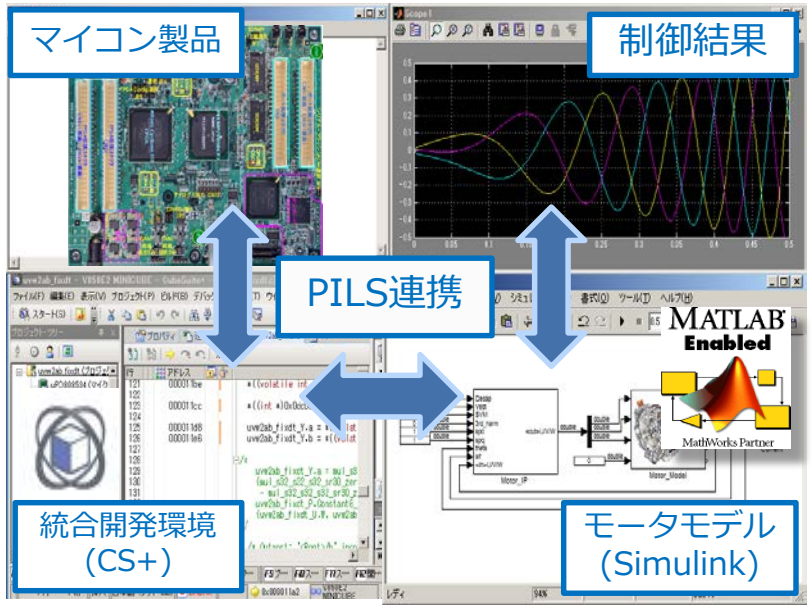
モデル化した制御用IPブロックを用い、Simulink上で機能検証(MILS)



EMU: モータ制御IP

標準メソッドAPIによるラピッド・プロトタイピング

- IP制御コードを自動生成
- PILSによる実IPの機能検証／性能検証が可能



IPシステムRCP(PILS)環境

ルネサスの取組み例
RH850マルチコア向けモデルベース連携環境

マルチコアマイコンは使いにくい？

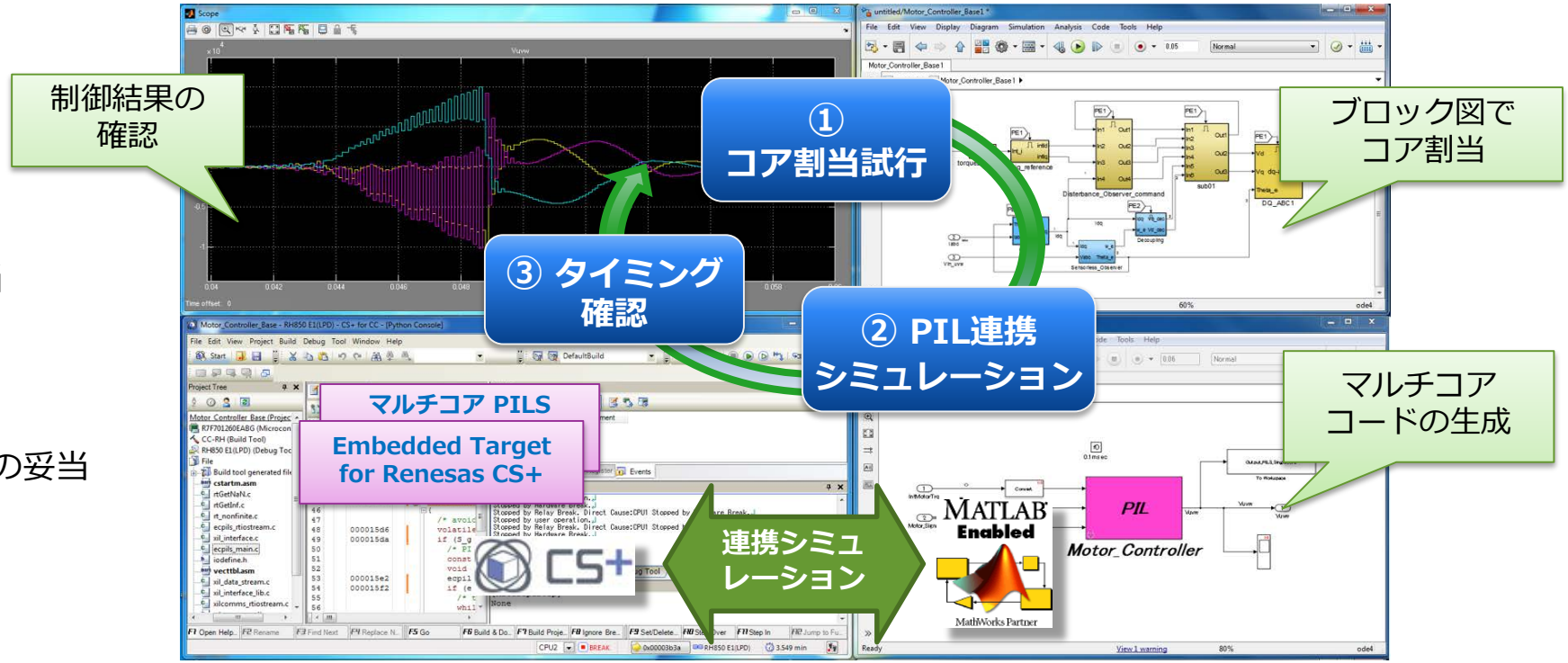
- マルチコア制御ソフト(同期・通信)開発が困難
- コントローラをコア分割した際の性能影響が予測しづらい

RH850マルチコアPILS環境

- Simulinkのブロック図でコア割当

  → マルチコアコードを自動生成
- 実機・シミュレータとの連携

  → モデル分割・マルチコア割付の妥当性を確認



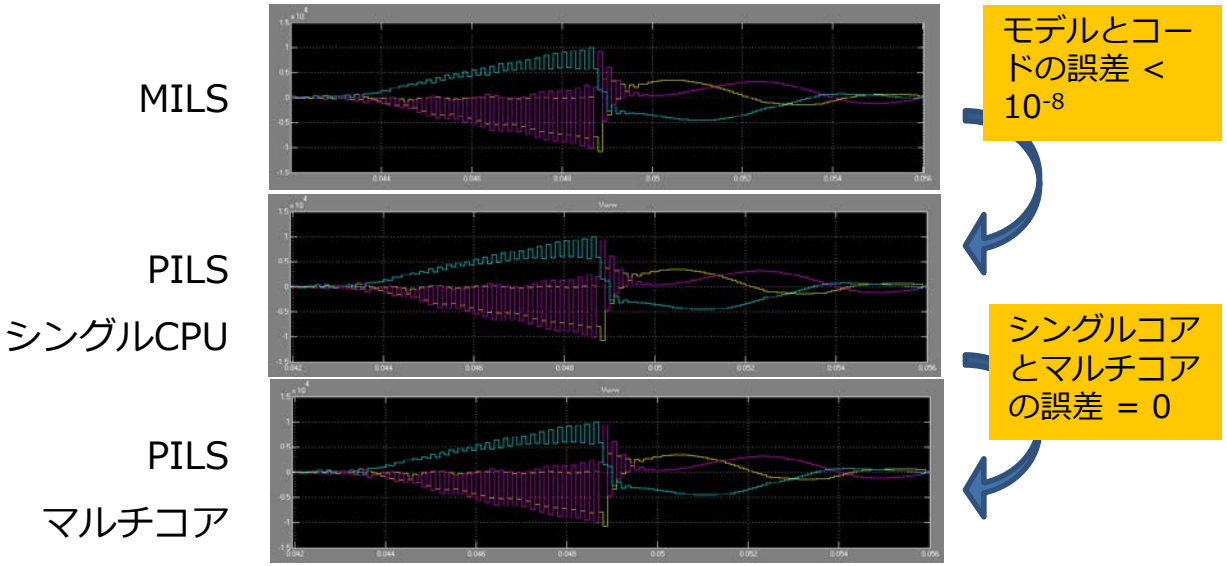
ルネサスの取組み例
マルチコアPILSによる機能検証と性能検証

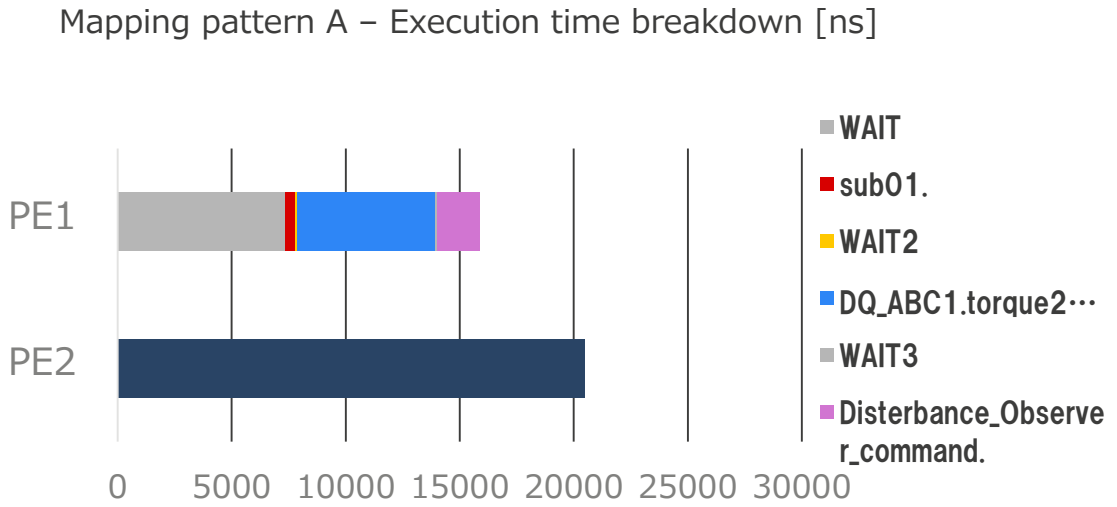
機能検証

- MILSとPILSのコントローラ出力の比較確認

性能検証

- サブシステム毎の実行時間や、PE間の待ち合わせ時間を確認



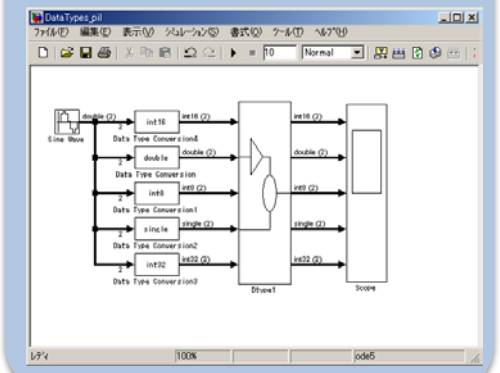


ルネサスの取組み例
モデルベース並列化ツールとの連携

モデルベース並列化ツールとの連携例

- コア割当プランを入力、PILS可能なモデルに自動変形
- PILSで取得した実測時間情報を並列化ツールへフィードバック

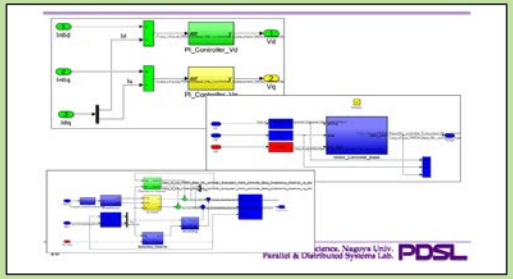
MathWorks Simulink



構造解析

名古屋大学 モデルベース並列化ツール

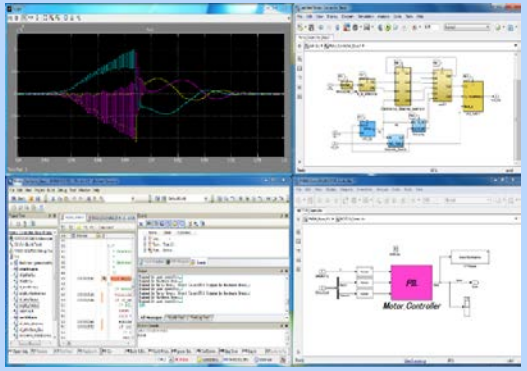
並列性抽出 コア割当



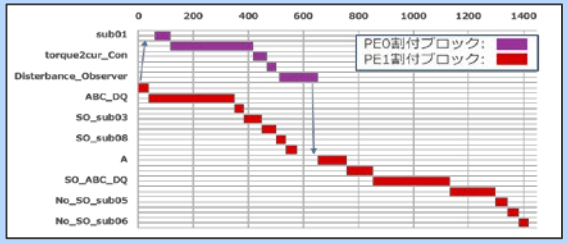
実測値

コア割当情報

ルネサス マルチコアPILS



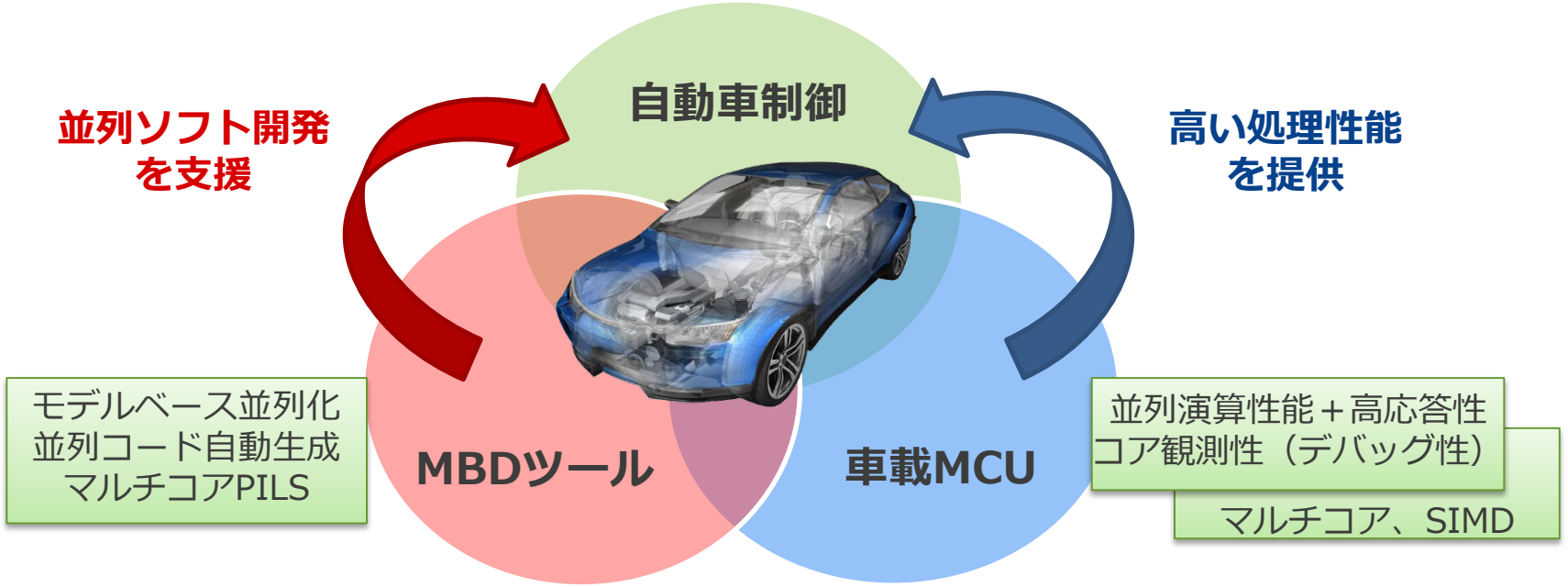
実行プロファイル



マルチ・メニーコアで実現する次世代自動車制御

- 制御性追求 ⇒ メニーコアによる演算量限界のブレークスルー
- 設計品質追求 ⇒ モデルベース連携マルチコア検証環境の充実

安全・快適・エコロジーな次世代自動車の実現に貢献



www.renesas.com