SHIM準拠
マルチ・メニーコア システムと
モデルベース開発環境

組込みマルチコアサミット2014

ルネサス エレクトロニクス株式会社
CPUシステムソリューション部　近藤弘郁

目次

- 組込みマルチコアマイコンへの取り組み
- デバイス開発における標準モデルSHIMの活用
- メニーコアによるHEV制御
- マルチ・メニーコアとモデルベース開発

情報系では、様々な分野でマルチ・メニーコア化が進んできたが、制御系でのマルチ・メニーコア化は今後の大きな課題となっている。
本講演では、マルチ・メニーコアの制御系アプリケーションへの浸透を目ざし、NEDO省エネプロジェクトで標準化を進めているSHIMに準拠したメニーコアチップとそのモデルベース開発環境について紹介する。

組込みマルチコアマイコンへの取り組み

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

- 直噴リーンバーンや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

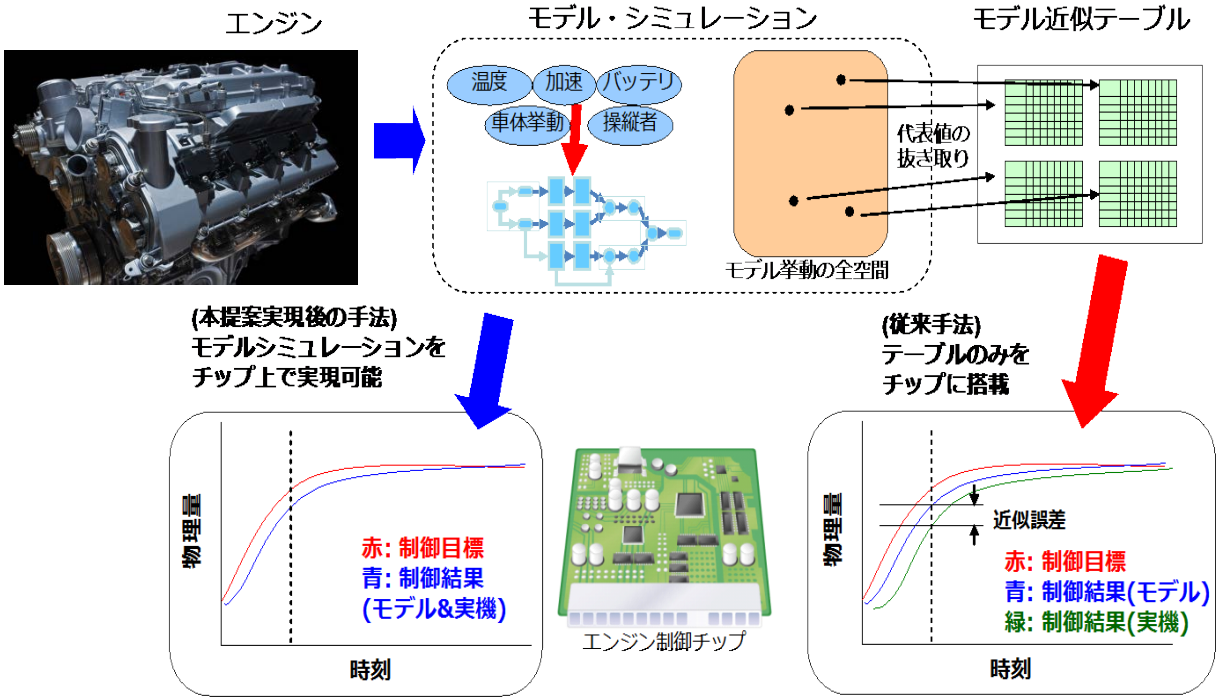
マイコンへの要求

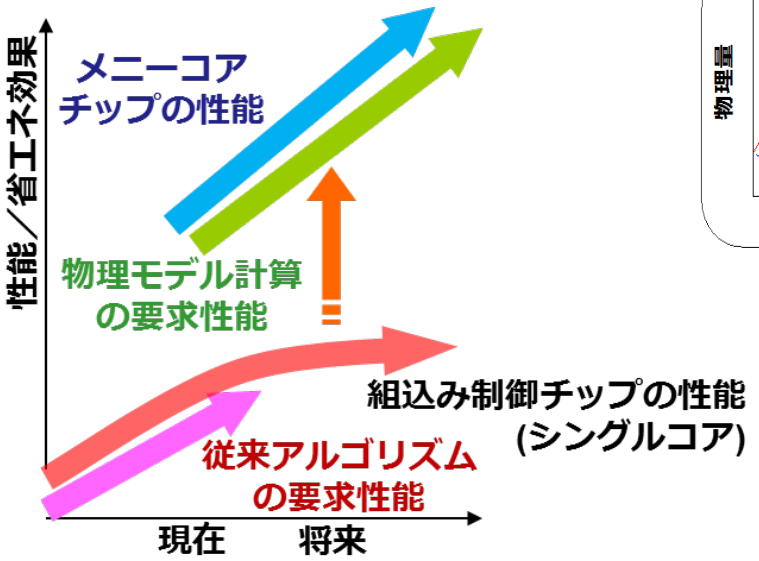
高性能なマイコン要求
(従来の3～5倍の性能)
限られた電力範囲での
高性能化が必須

先進的な制御アルゴリズムの実用化

■ 例：モデル予測制御

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（消費電力500mW以下）で実現

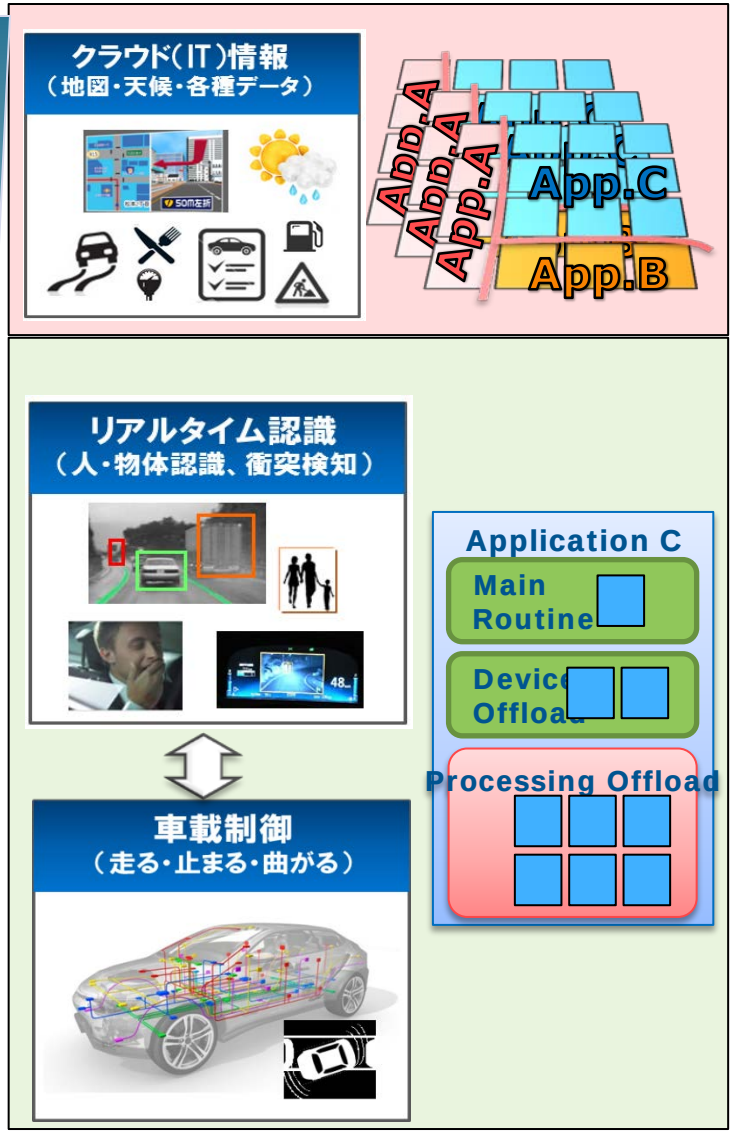




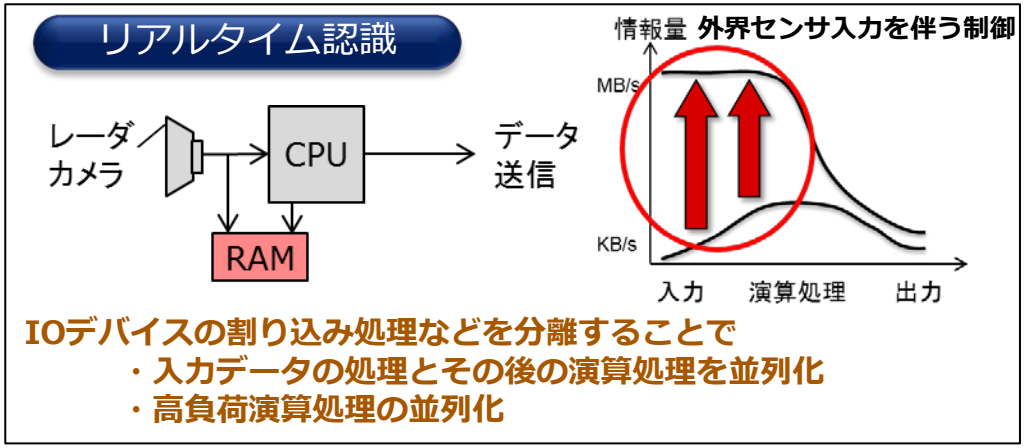
■高精度な制御により燃費向上
- 先進的な制御の導入
  （物理モデルによる予測制御）
- マルチ・メニーコアにより実現

制御マルチコア実現の課題　並列化の抽出

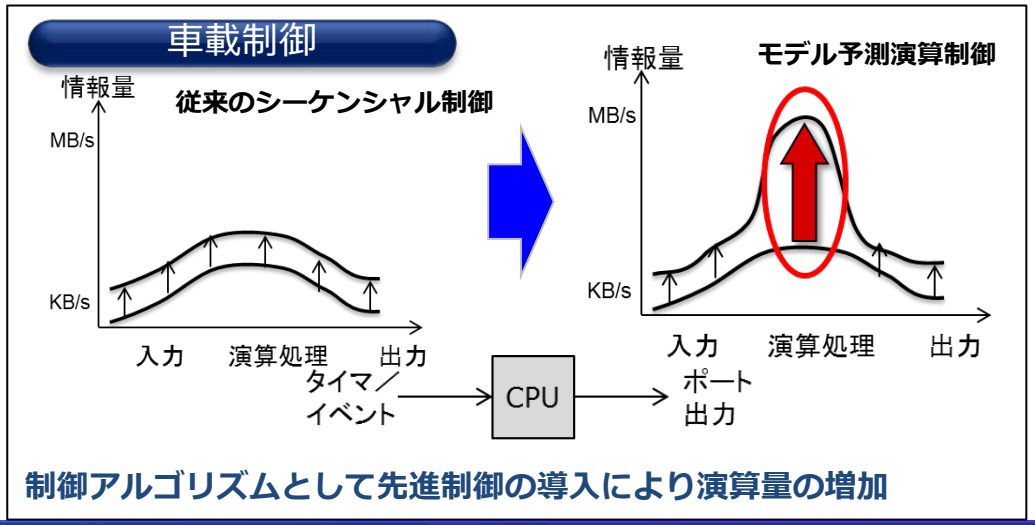
並列度



■ ITシステムにおけるマルチコア
　アプリの並列配置は比較的容易
　実時間制御性に対する要求が低い



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



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

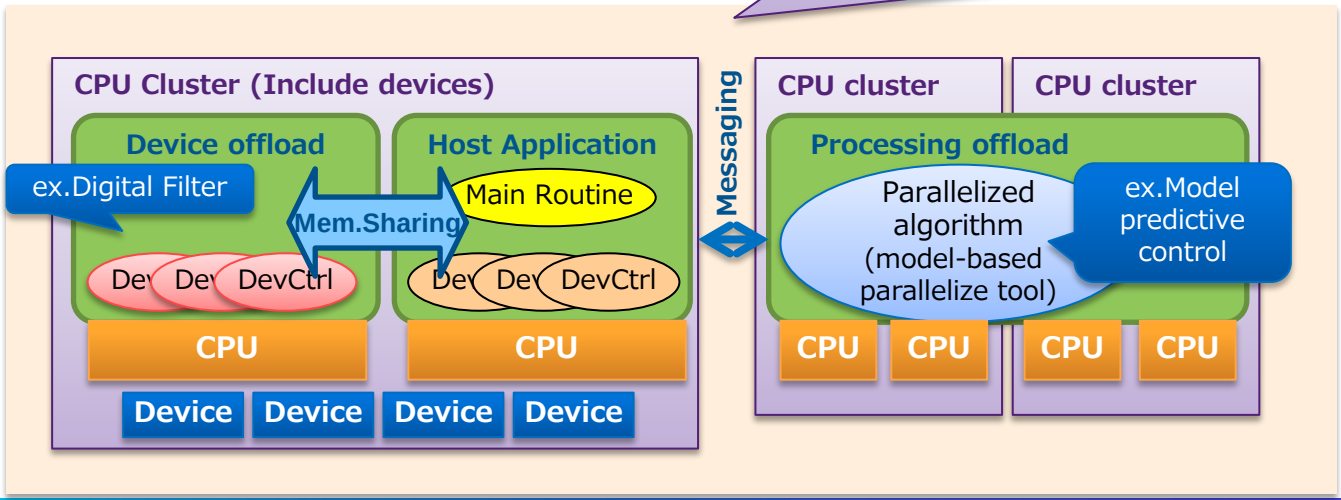
制御システムの並列化アプローチ

制御システムに潜在する並列性を利用

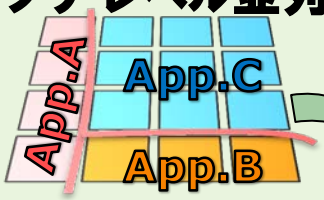
- アプリの並列配置
- 機能ブロック毎の並列化
- モデルベース並列化

アプリケーション内を周期特性毎に並列化

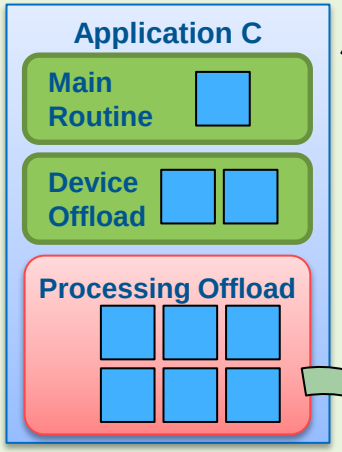
- デバイス処理の分離（非同期であることが主）
- 高負荷演算の分離



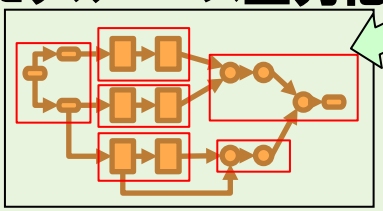
アプリ・レベル並列



機能ブロックの並列



モデルベース並列化

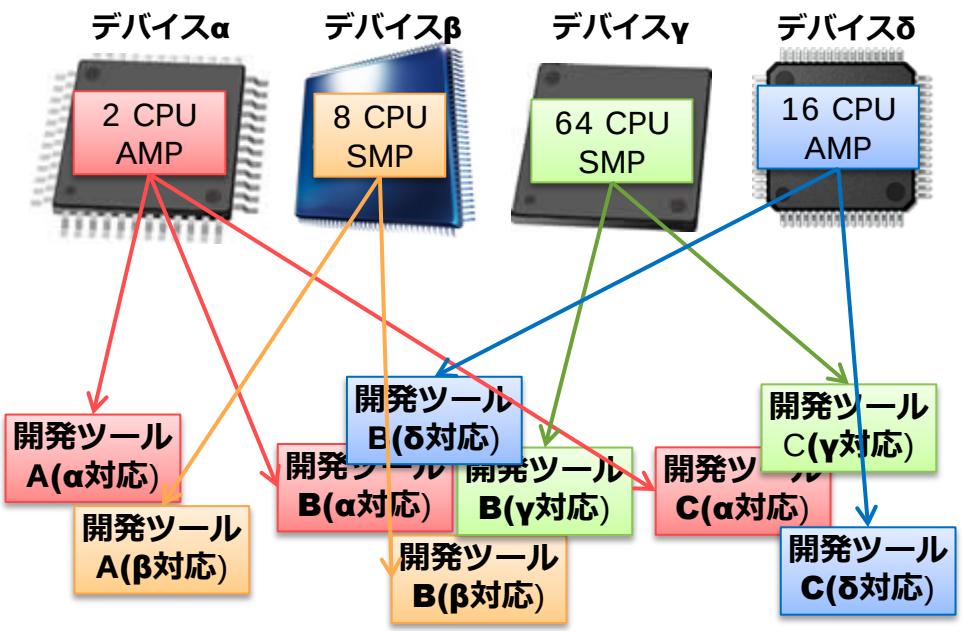


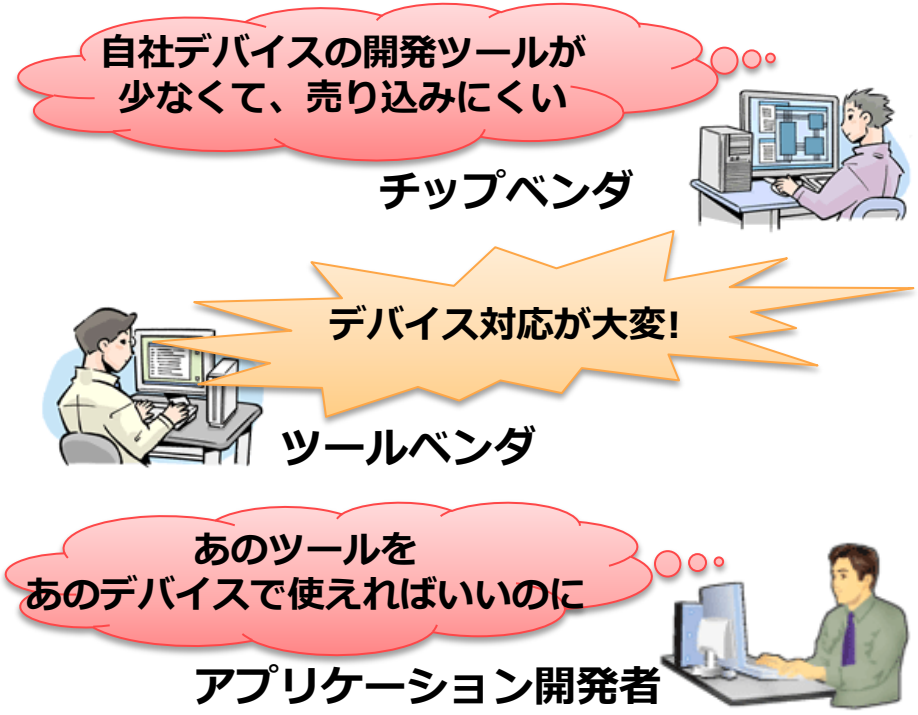
デバイス開発における標準モデルSHIMの活用

マルチ・メニーコア普及への課題

- ソフトウェア・システムの抱えるサブシステムの数／処理規模により、
  最適なCPU構成、メモリ・モデルが異なる
  ⇛ **多種多様なマルチコア・メニーコア製品が登場**
- 結果として

開発ツールの個別対応が増加し、ツールチェーン構築が困難

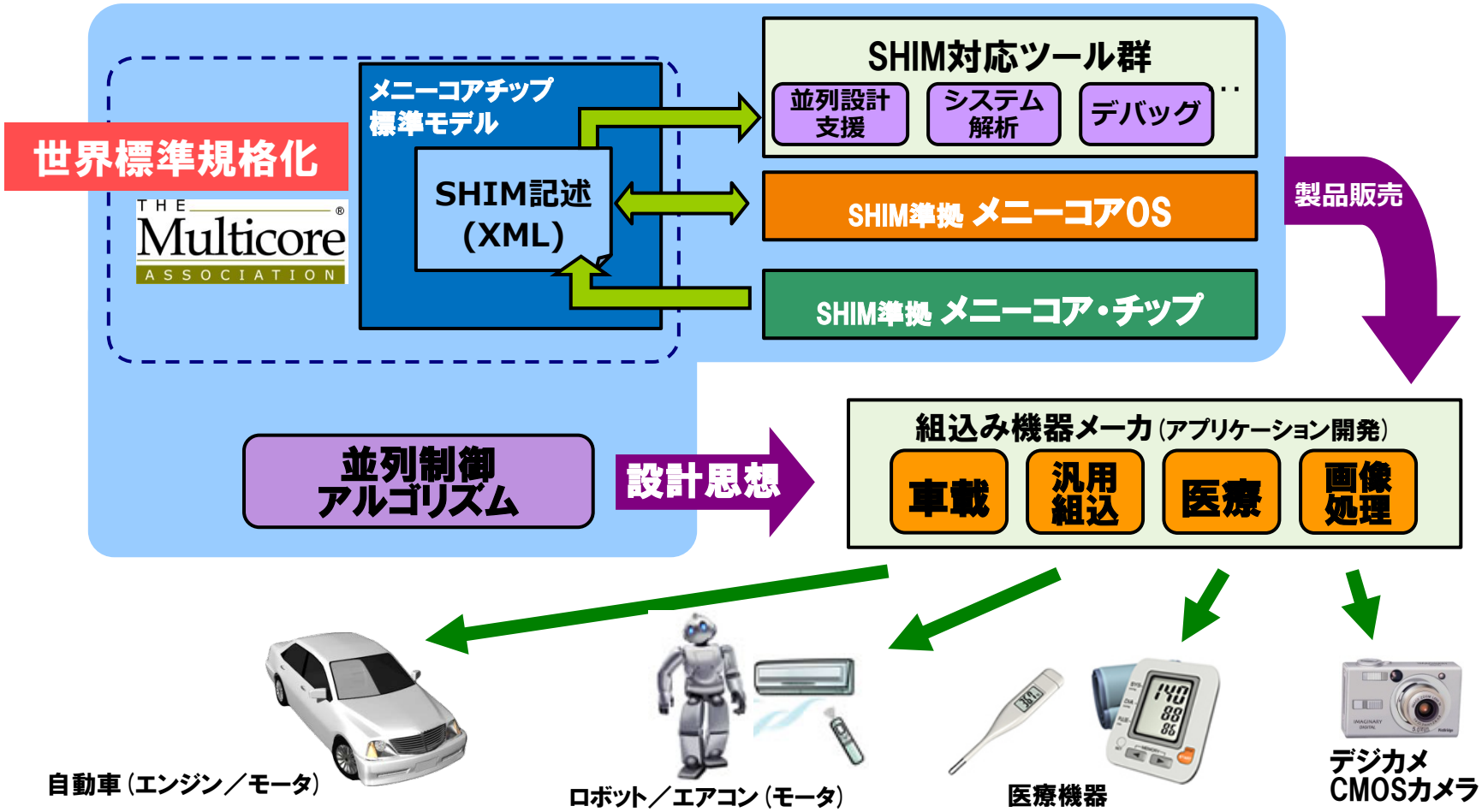




マルチ・メニーコア標準プラットフォーム

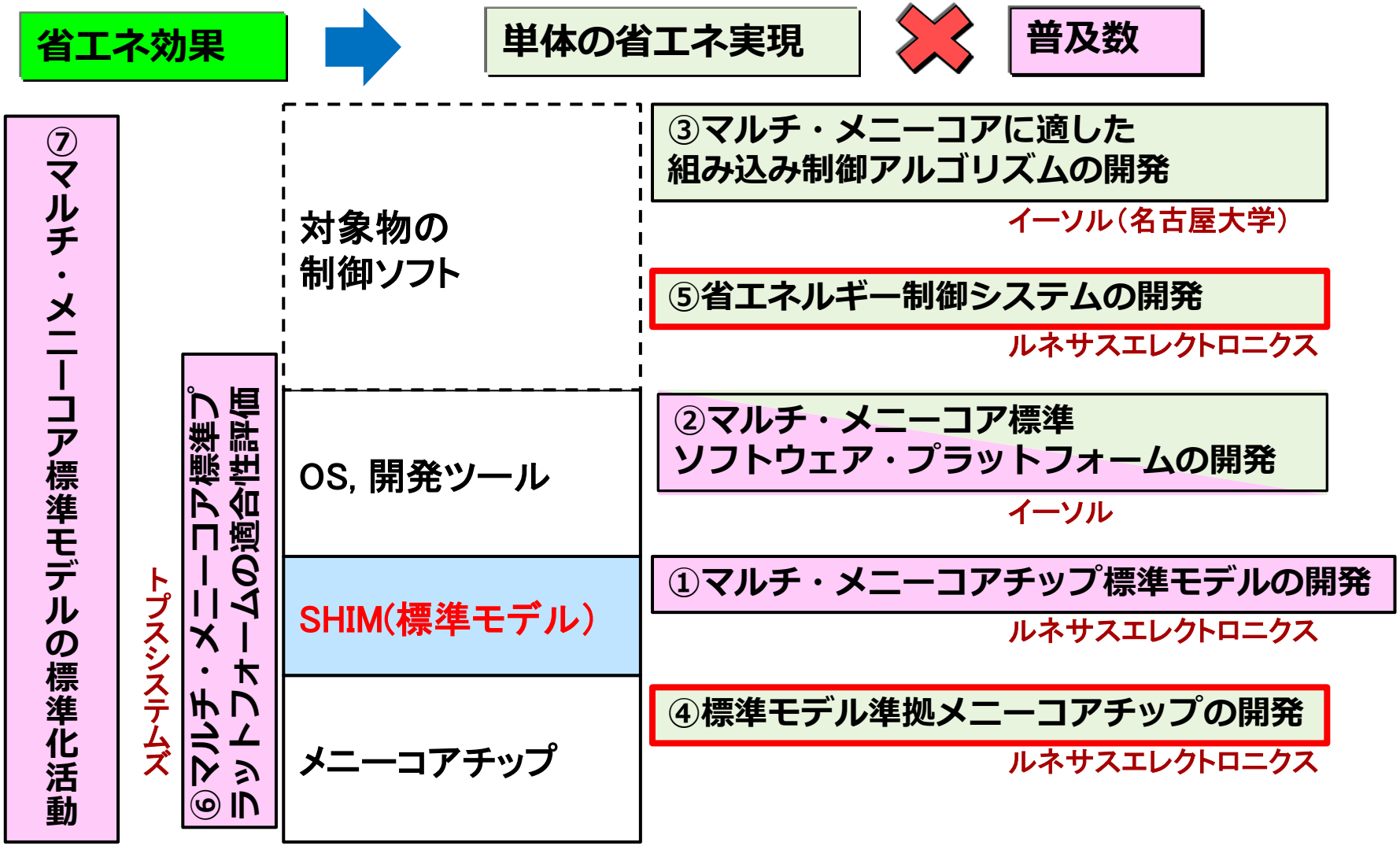
組込み機器制御全般に適用可能なマルチ・メニーコア標準プラットフォームにより、マルチ・メニーコアを "誰もが使えるモノ"に

SHIM: Software-Hardware Interface for Multi-many-core



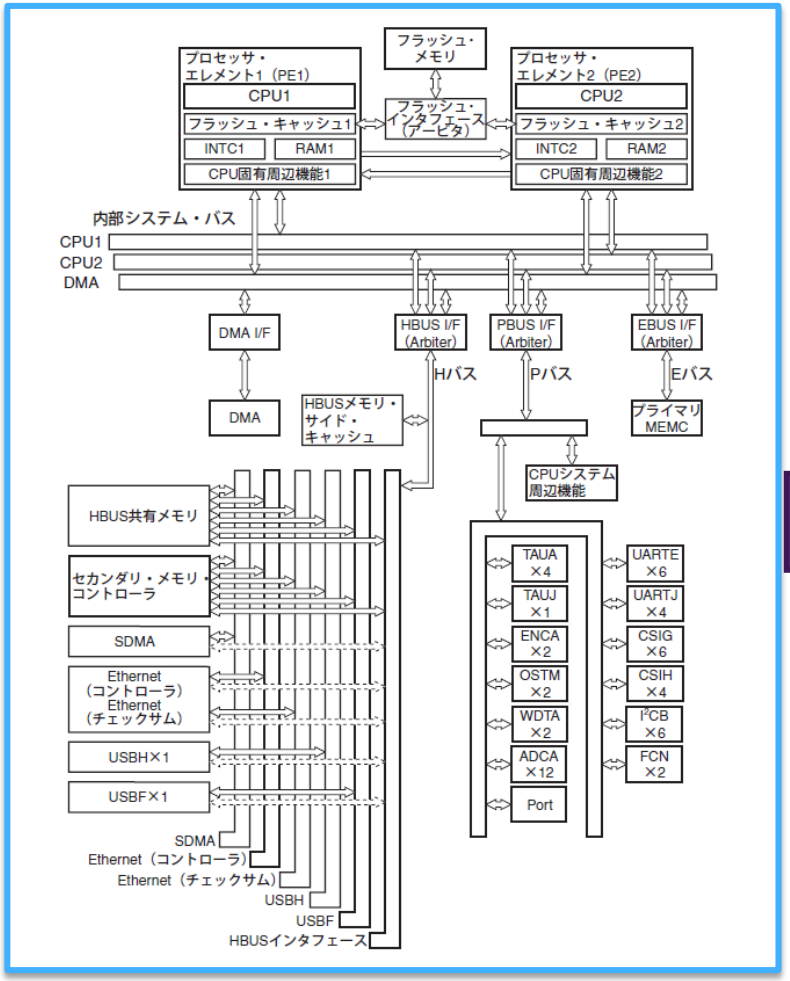
マルチ・メニーコア標準プラットフォーム

NEDO戦略的省エネルギー技術革新プログラム

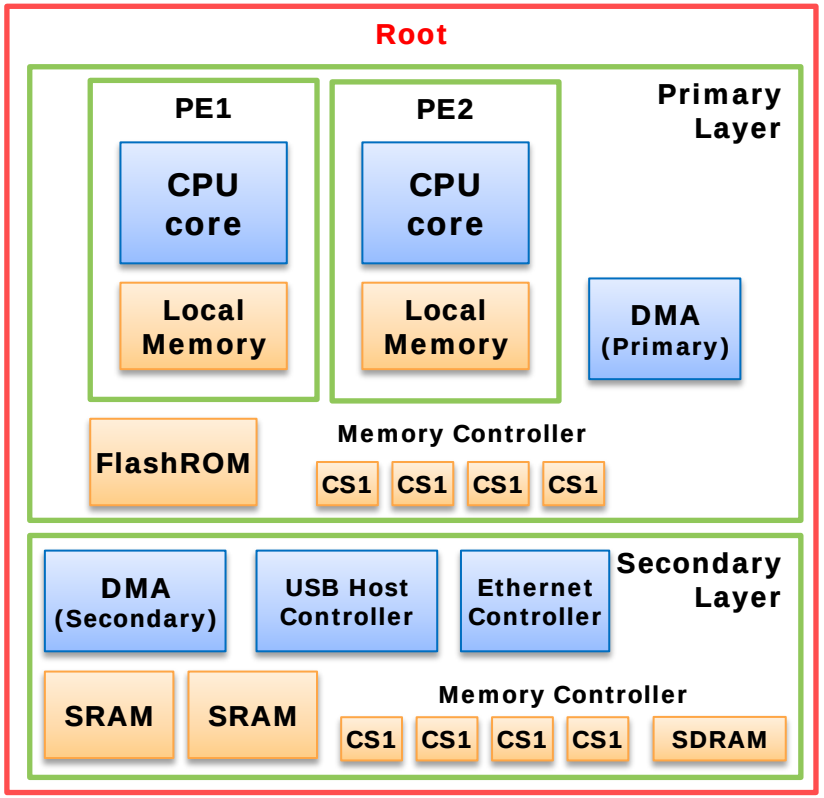


デュアルコアマイコンの記述例

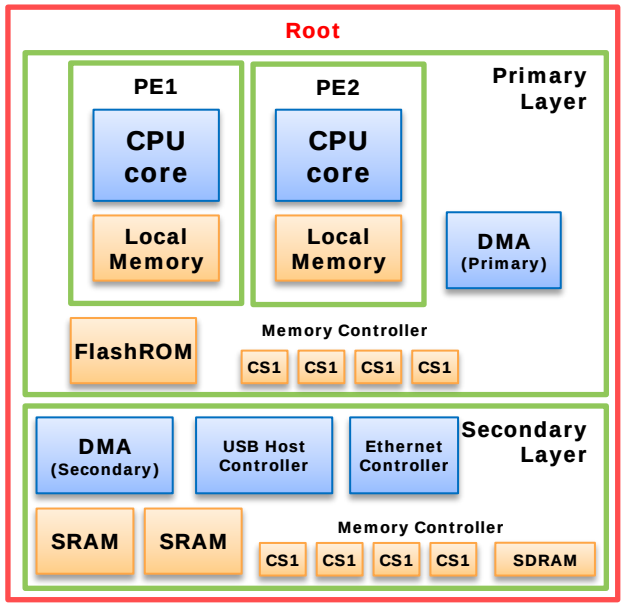
マイコン仕様



SHIMモデル (Hardware Components)



デュアルコアマイコンの記述例（SHIMモデル→SHIM記述）



SHIMモデル

SHIM記述（XML）

XML Explorer - [v850_E2_MN4.shim.xml]

File Edit View Format Tools Help

XPath: Launch...

v850_E2_MN4.shim.xml

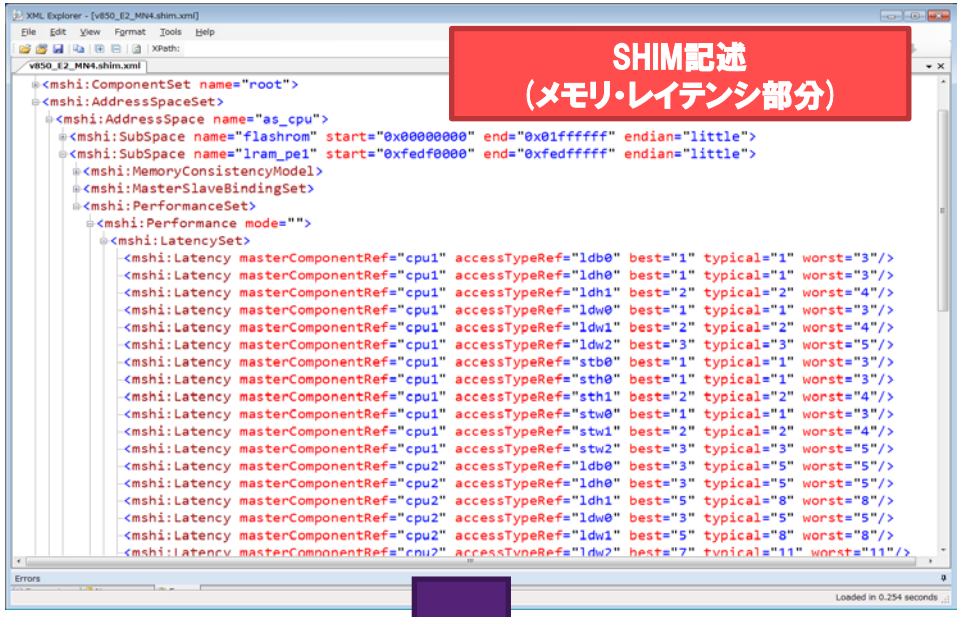
```
<mshi:SystemConfiguration name="V850E2/MN4(upd70F3515)">
  <mshi:ClockFrequencySet>
  <mshi:SystemClock unit="MHz">
  <mshi:ComponentSet name="root">
    <mshi:ComponentSet name="PrimaryLayer">
      <mshi:ComponentSet name="pe1">
        <mshi:MasterComponent name="cpu1" masterType="cpu" arch="v850e2v3" pid="0x000
        <mshi:SlaveComponent name="lram1" rwType="rw" size="64" sizeUnit="kb"/>
      <mshi:ComponentSet name="pe2">
        <mshi:MasterComponent name="cpu2" masterType="cpu" arch="v850e2v3" pid="0x000
        <mshi:SlaveComponent name="lram2" rwType="rw" size="64" sizeUnit="kb"/>
      <mshi:MasterComponent name="pdma" masterType="dma" arch="v850e2dma" pid="" tran
      <mshi:SlaveComponent name="flashrom" rwType="r" size="2" sizeUnit="mb"/>
      <mshi:SlaveComponent name="pmemc_cs1" rwType="rw" size="32" sizeUnit="mb"/>
      <mshi:SlaveComponent name="pmemc_cs2" rwType="rw" size="64" sizeUnit="mb"/>
      <mshi:SlaveComponent name="pmemc_cs3" rwType="rw" size="64" sizeUnit="mb"/>
      <mshi:SlaveComponent name="pmemc_cs4" rwType="rw" size="64" sizeUnit="mb"/>
    <mshi:ComponentSet name="SecondLayer">
      <mshi:MasterComponent name="sdmac" masterType="dma" arch="???" pid="" translati
      <mshi:MasterComponent name="usbhost" masterType="other" arch="???" pid="" trans
      <mshi:MasterComponent name="ethernet" masterType="other" arch="???" pid="" tran
      <mshi:SlaveComponent name="shram1" rwType="rw" size="32" sizeUnit="kb"/>
      <mshi:SlaveComponent name="shram2" rwType="rw" size="32" sizeUnit="kb"/>
      <mshi:SlaveComponent name="smemc_cs0" rwType="rw" size="16" sizeUnit="mb"/>
```

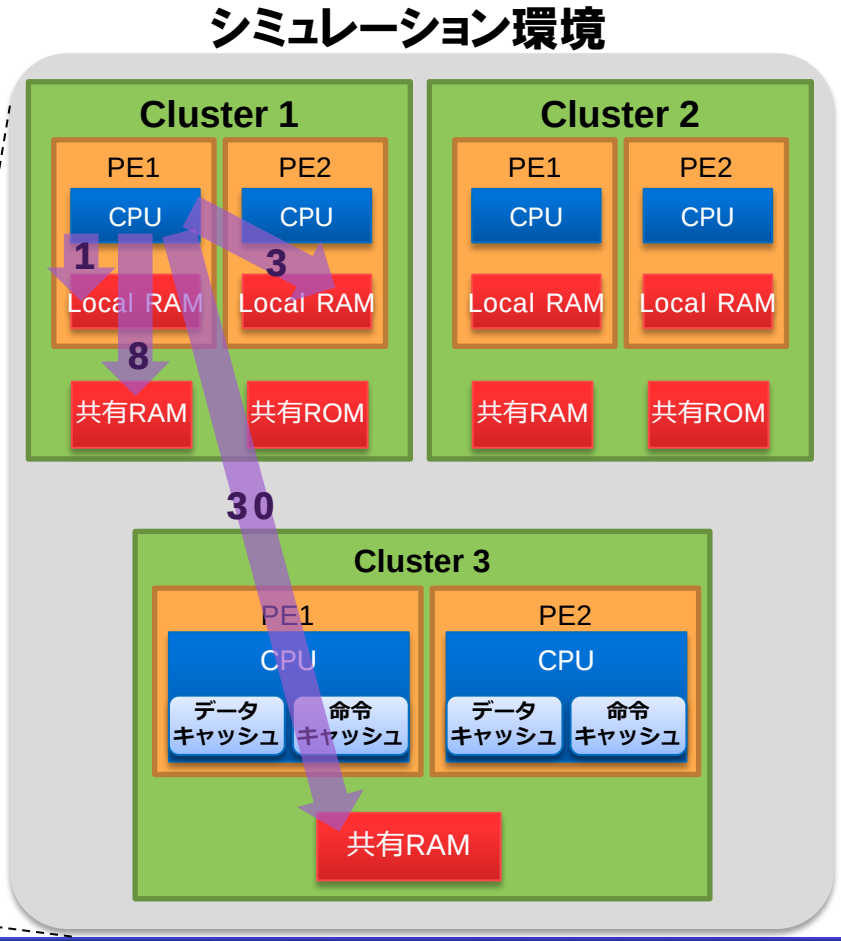
Errors

/mshi:SystemConfiguration 5 child nodes Loaded in 0.281 seconds

シミュレータでのSHIM活用案： メモリレイテンシモデル

- SHIM記述に応じたシミュレーション環境を生成
  - 様々なマルチコア製品を、共通のシミュレータでシミュレーション可能





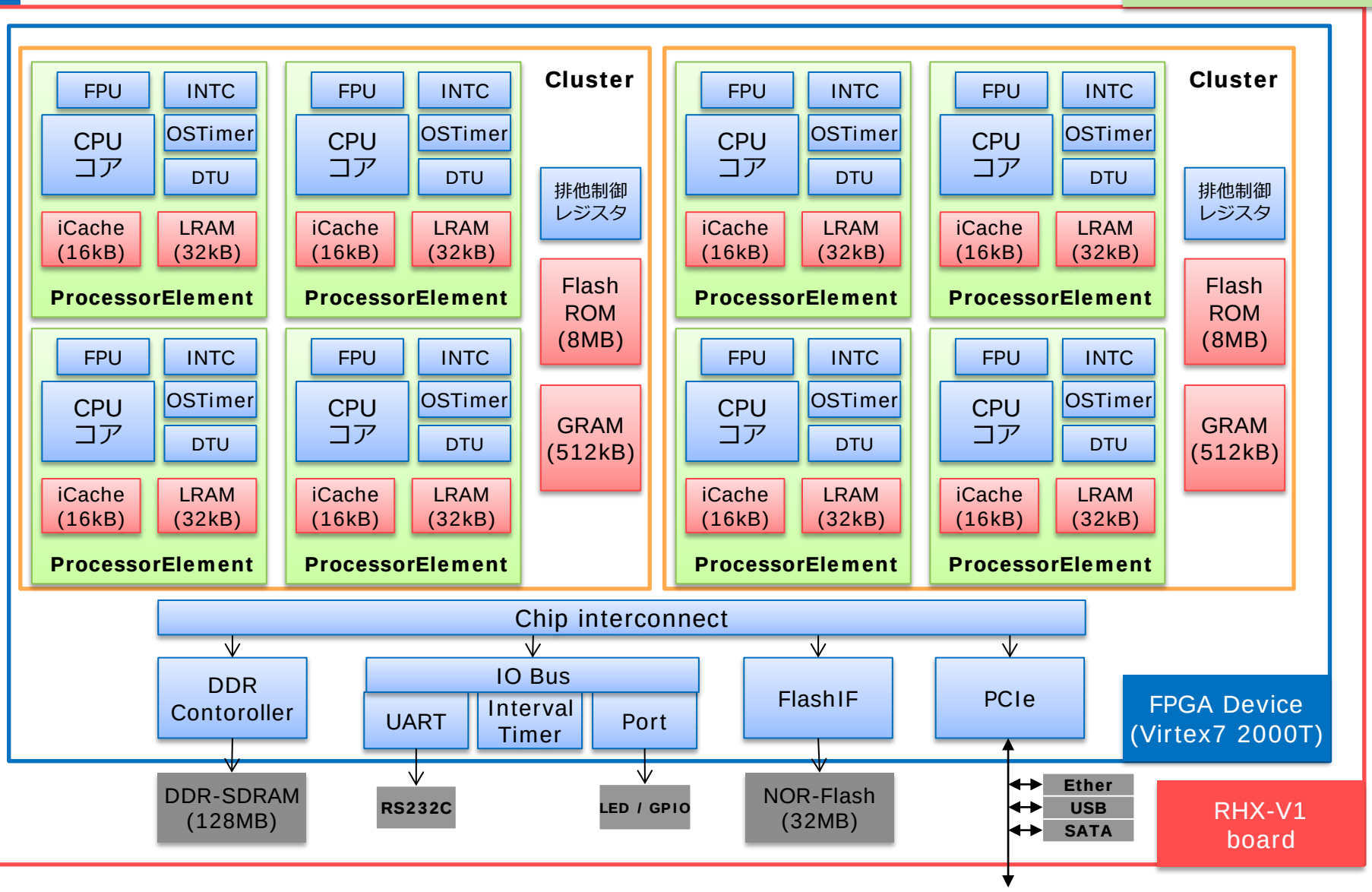
マルチ・メニーコアによるHEV制御

16コア搭載MCUエミュレーションボード概要

- ハードリアルタイム制御向けCPUコアを搭載
  - 16kB 2way 命令キャッシュ
  - 高速アクセス可能なCPU直結RAM（32kB, 1cycle Latency）
- 4コア搭載のCPUクラスタを4つ搭載 = Total 16コア
  - クラスタ毎に共有RAMを搭載（512kB, 8cycle Latency）
- モデルベース並列化指向の階層型バス・システム
  - 低レイテンシ・高スループットなパケット型プロトコルを採用
  - NUMA（Non-Uniformed Memory Access）
  - CPU直結RAM、クラスタ共有RAM、システム共有RAMの3段階層
- コア間排他／同期の支援機構を搭載
  - CPU間割り込み
  - 排他／同期処理用高速レジスタ

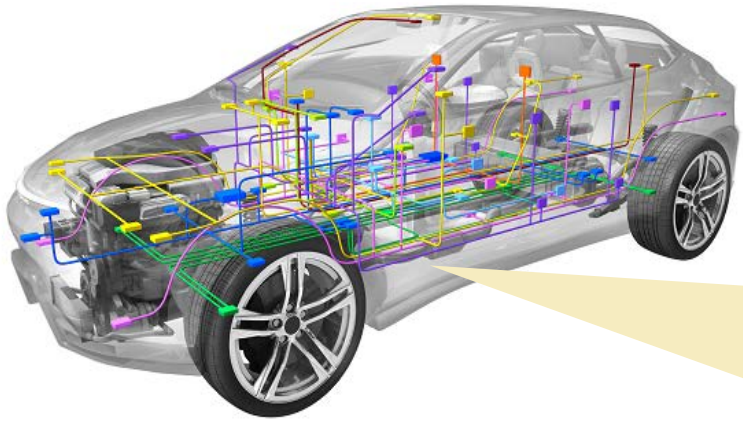
16コア搭載MCUエミュレーションボード

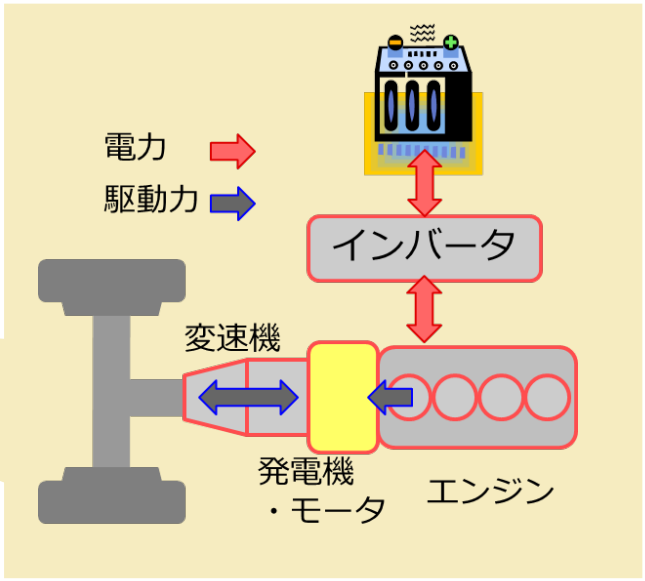
DTU: Data Transfer Unit
FPU: Single Precision FPU



ハイブリッド・エンジン（HEV） 制御

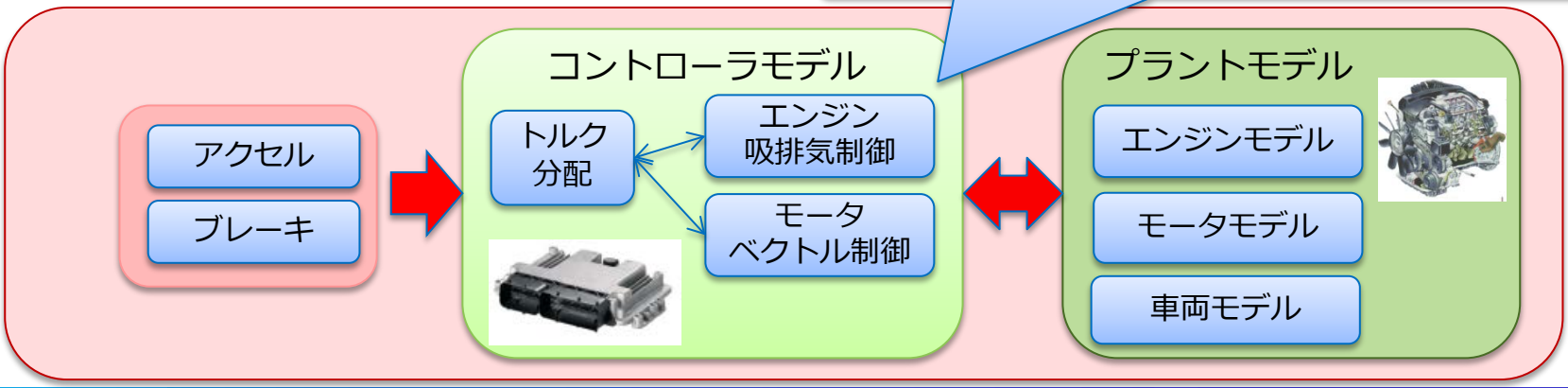
■ HEVの制御並列化を対象に実現可能性を評価





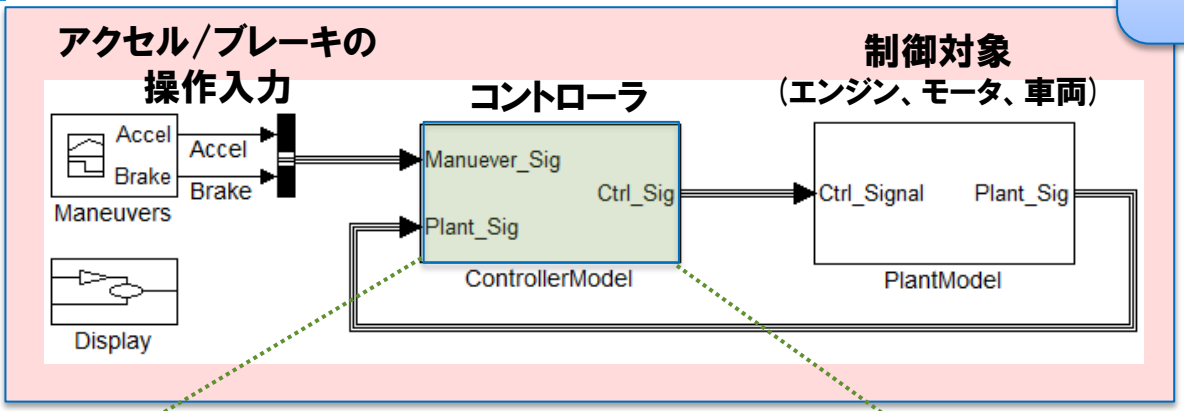
省エネ（燃費向上）のため、自動車駆動系の制御システムはさらに複雑になり、制御にかかる計算量が増加

制御には、モデル予測制御を採用

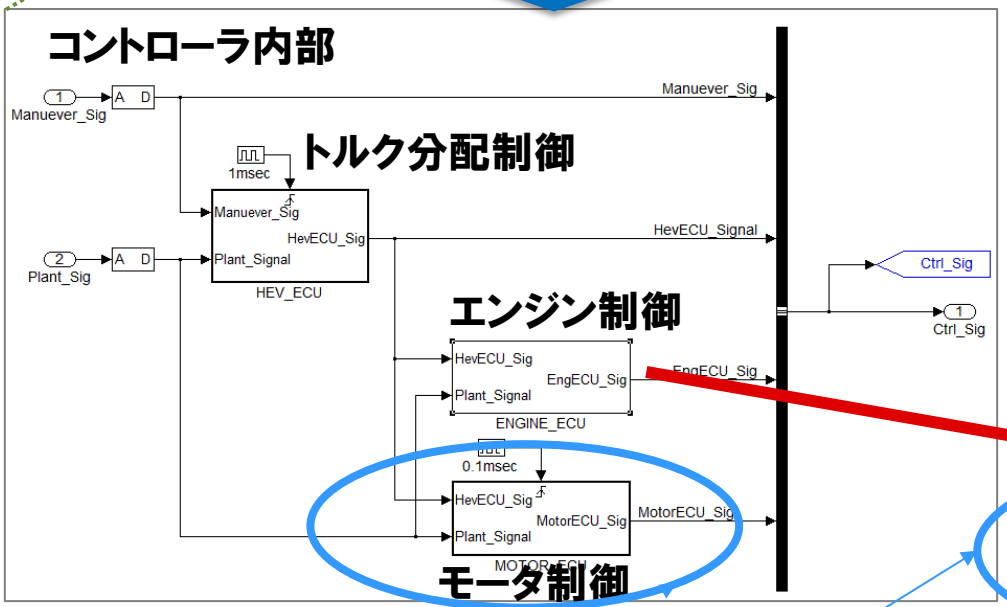


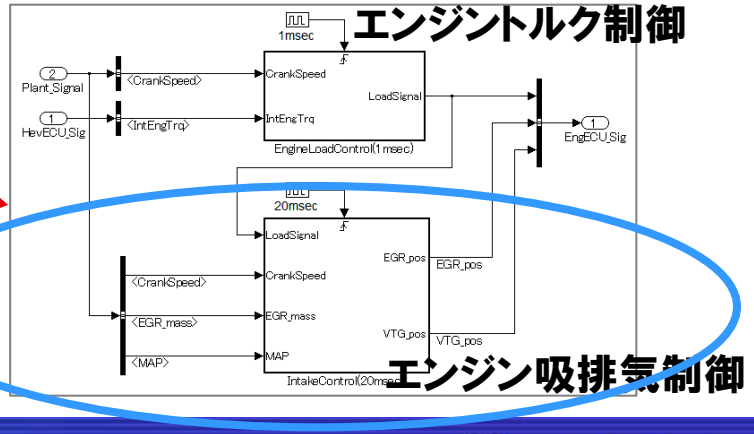
HEV制御モデル

より精度の高い制御を実現し、
燃費向上・快適性・加速性能を
得る



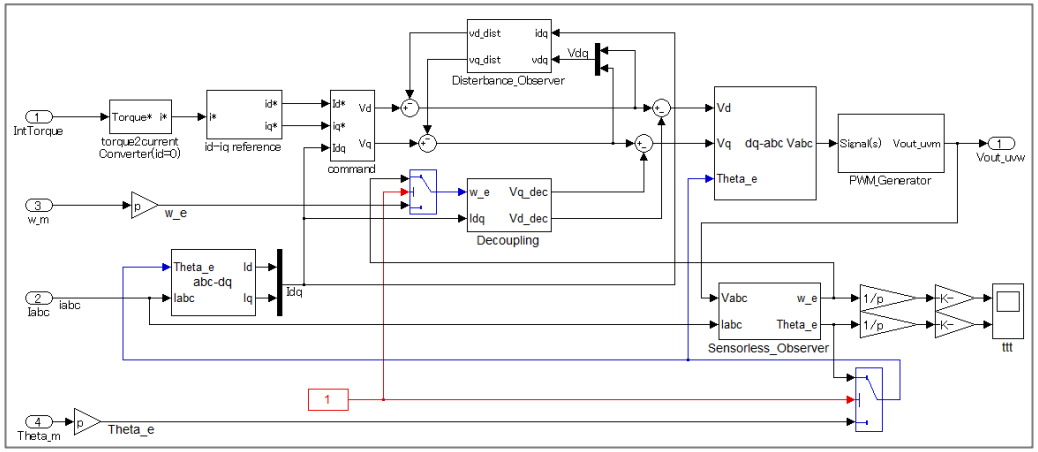
エンジン 吸排気制御 4ms周期	ディーゼルエンジンの吸排気と吸気圧力を制御。モデル予測を使い、制御量（バルブ開度、排気弁位置）を計算。
モータ 制御 100us周期	ACモータ回転を制御。ベクトル制御、オブザーバによる外乱除去、非干渉化、センサレス化など。
トルク 分配制御 1ms周期	エンジンとモータに分配するトルク量を制御。簡単な計算式とテーブル参照により分配トルク量を計算。
エンジン トルク制御 1ms周期	エンジントルクを制御。テーブル参照による単純制御。



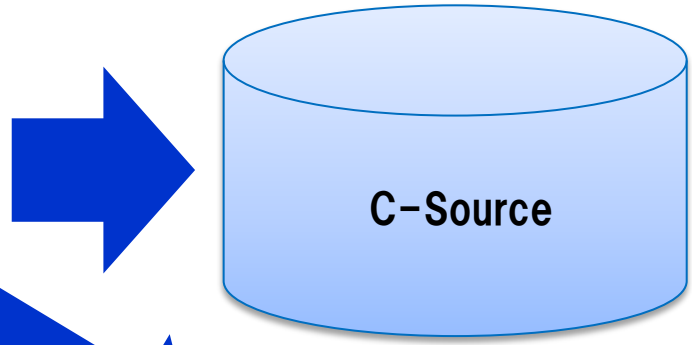


Auto Code Generation + 並列化ツール

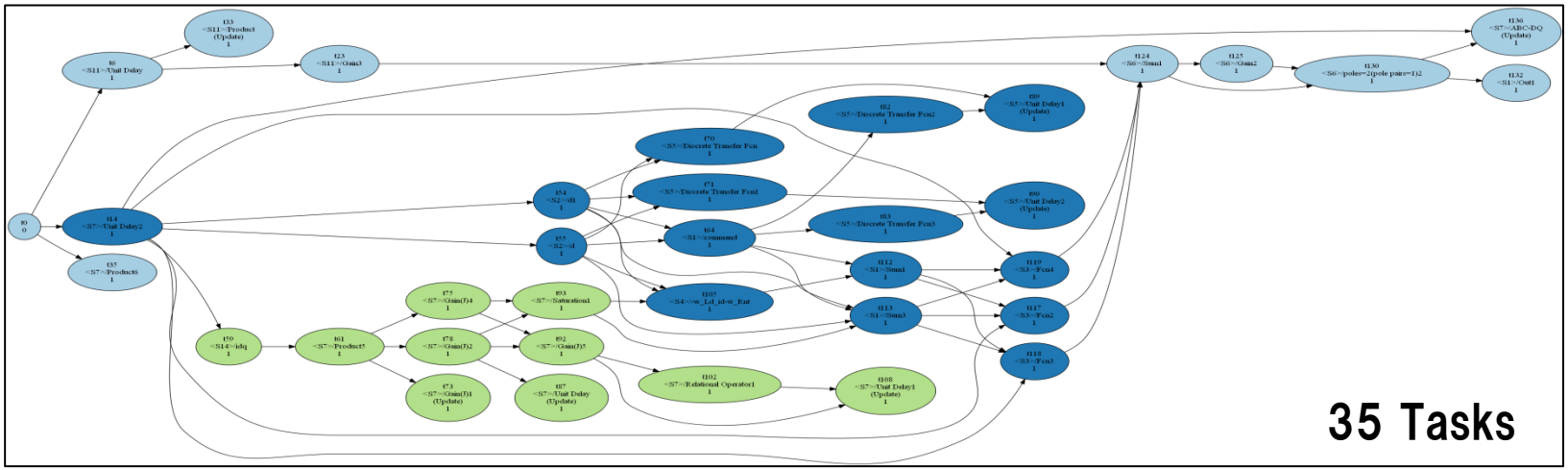
Controller Model in Simulink



ACG + 並列化ツール



Task Flow Graph



HEV制御並列化手法

2段階の並列化を組み合わせて利用

サブシステムを周期特性毎に並列配置

- デバイス処理の分離
- 異なる制御周期のサブシステム

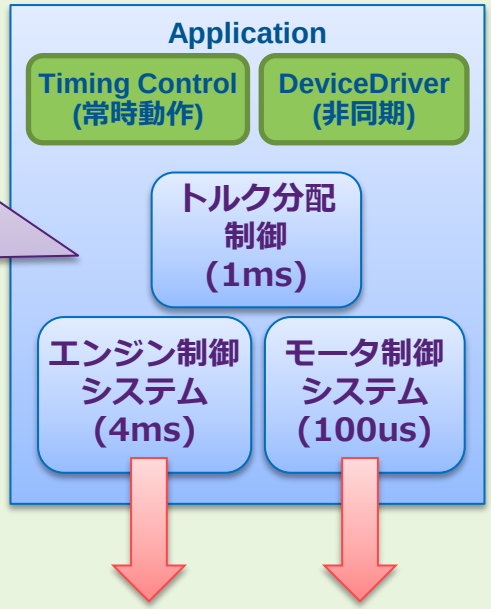
➡ 従来からあるマルチタスク手法を用い、
そのまま並列タスクとして実行

ブロック間依存関係を考慮し並列化

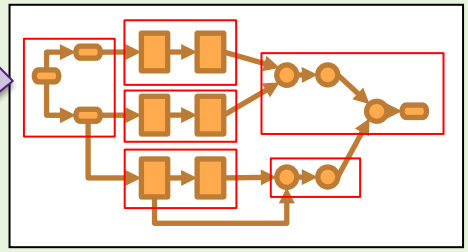
- サブシステム内の処理並列性をモデルを元に抽出
- ブロック毎の実行プロファイルを元に、並列実行順序を計画
- 並列化コードは、逐次コードから自動生成

➡ モデルから半自動で並列化コードを生成

機能ブロックの並列化



モデルベース並列化



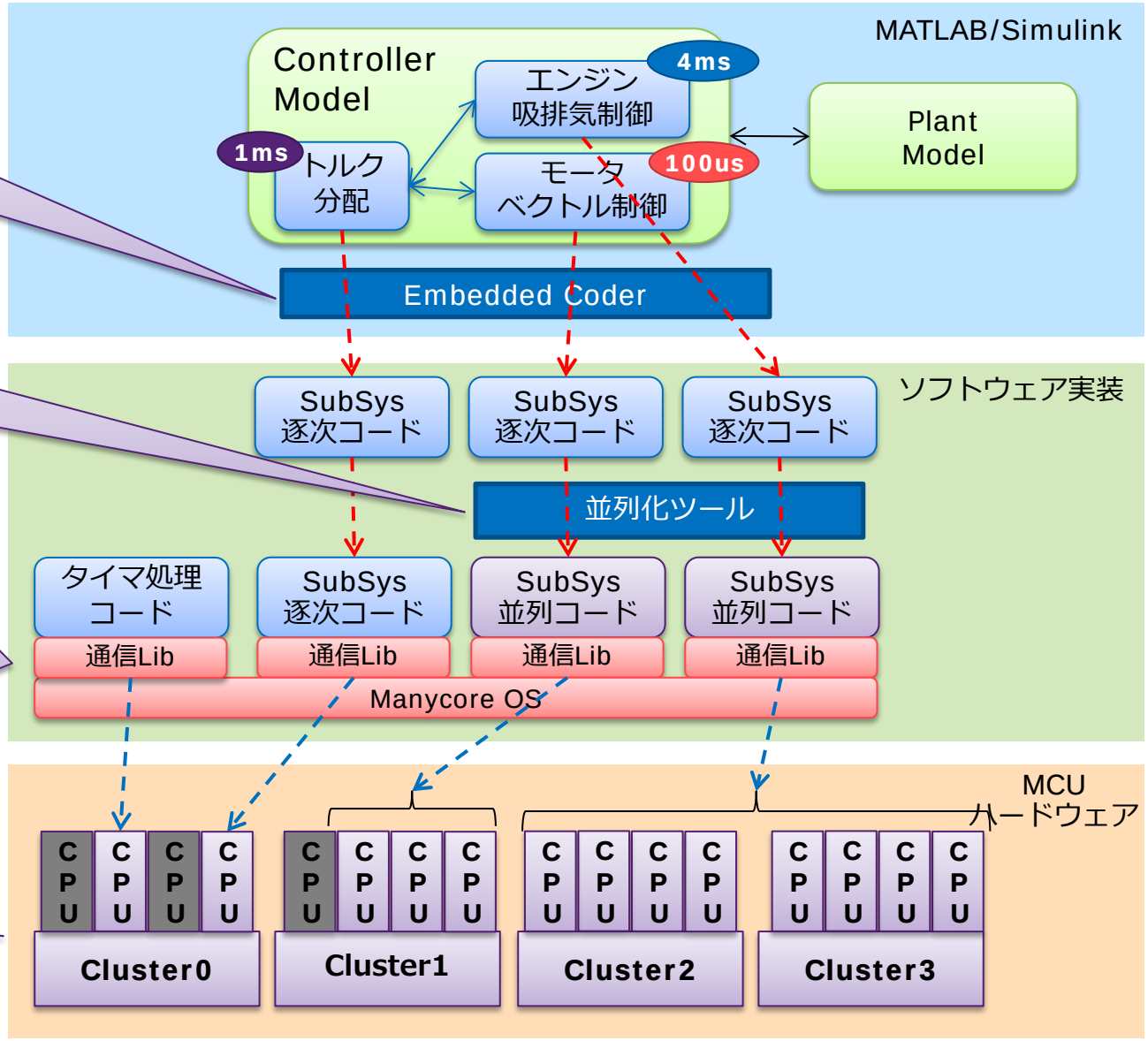
HEV制御並列化ソフトウェア実装フロー

Step1.
レートの異なるブロック毎に逐次コードを生成

Step2.
並列化ツールにより、逐次コードをそれぞれ並列化

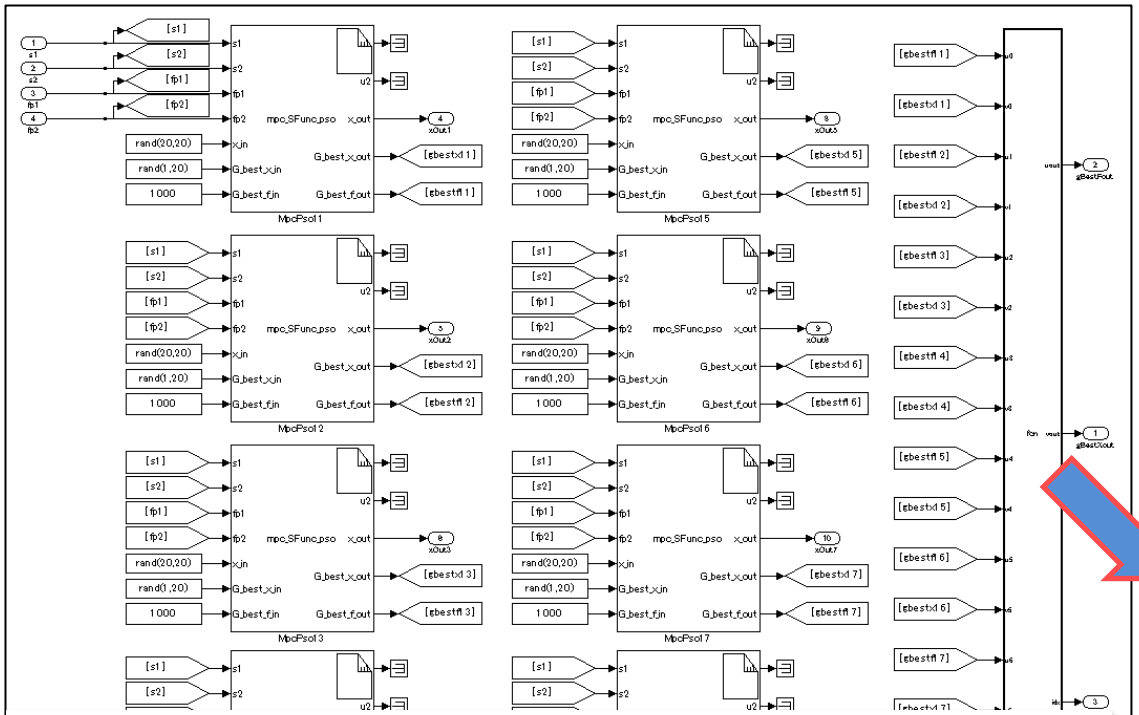
Step3.
SubSystem間の通信ライブラリを追加し、並列化コードをManycore OS上に実装。

Step4.
CPU上のハードウェア・スレッドにマッピングして実行



ディーゼルエンジン制御コントローラ概要

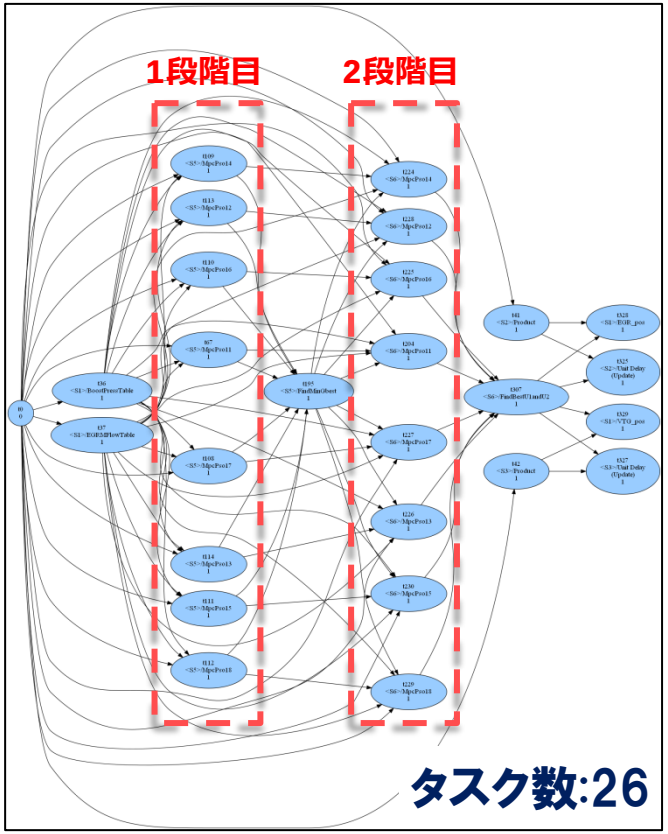
コントローラモデル



制御対象	**ディーゼルエンジンの吸排気システム** **互いに干渉する吸排気量と吸気圧力を制御**
制御周期	4 ms
制御アルゴリズム	PSO(particle swarm optimization)**を使ったモデル予測制御**
入出力	**入力**: **吸気圧力**, **排気量** **出力**: **バルブ開度**、**排気弁位置**

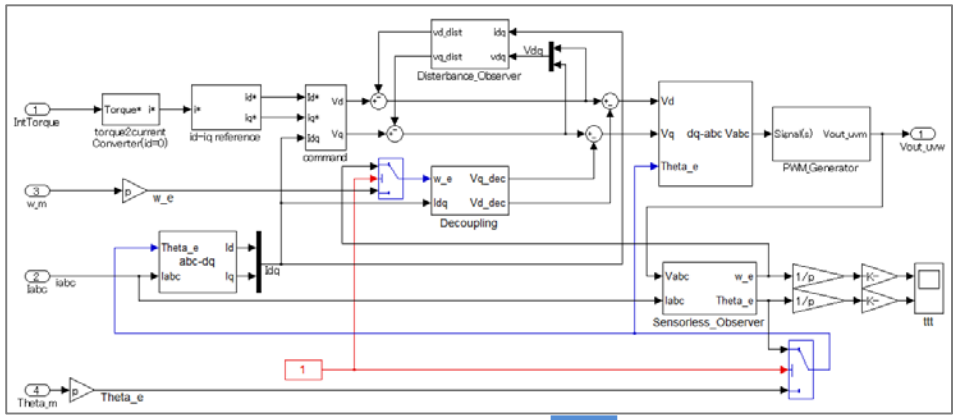
モデル予測制御による**最適制御値探索**を、8並列、2段階、で実施。
合計で、モデル予測によるローカル最適値探索を16回実施。

タスクフローグラフ



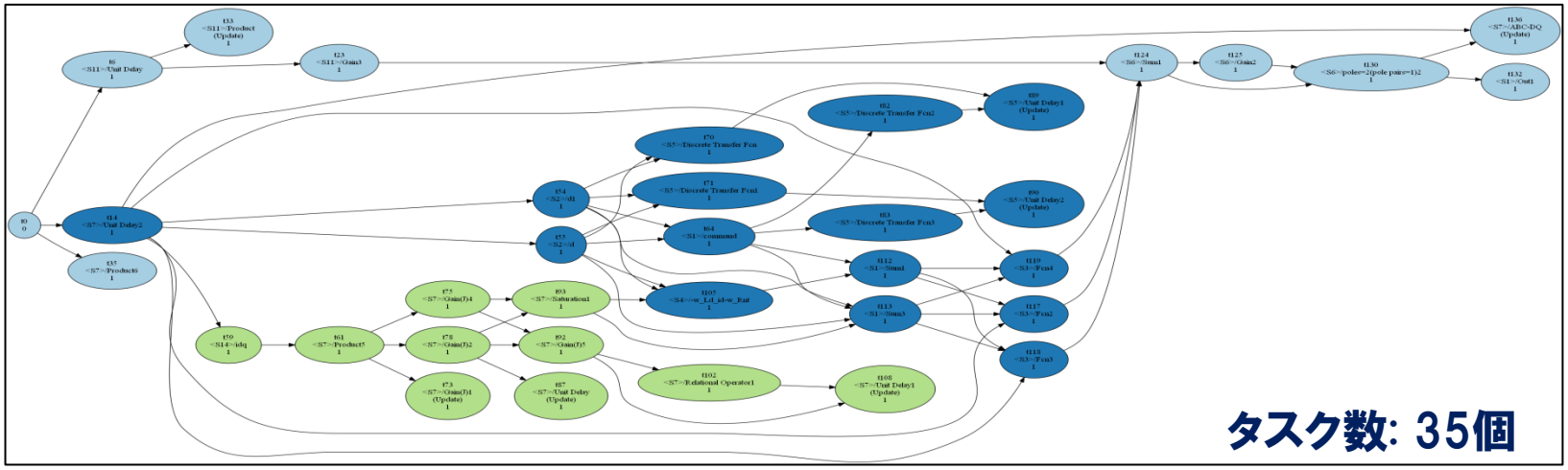
モータ制御コントローラ概要

コントローラモデル



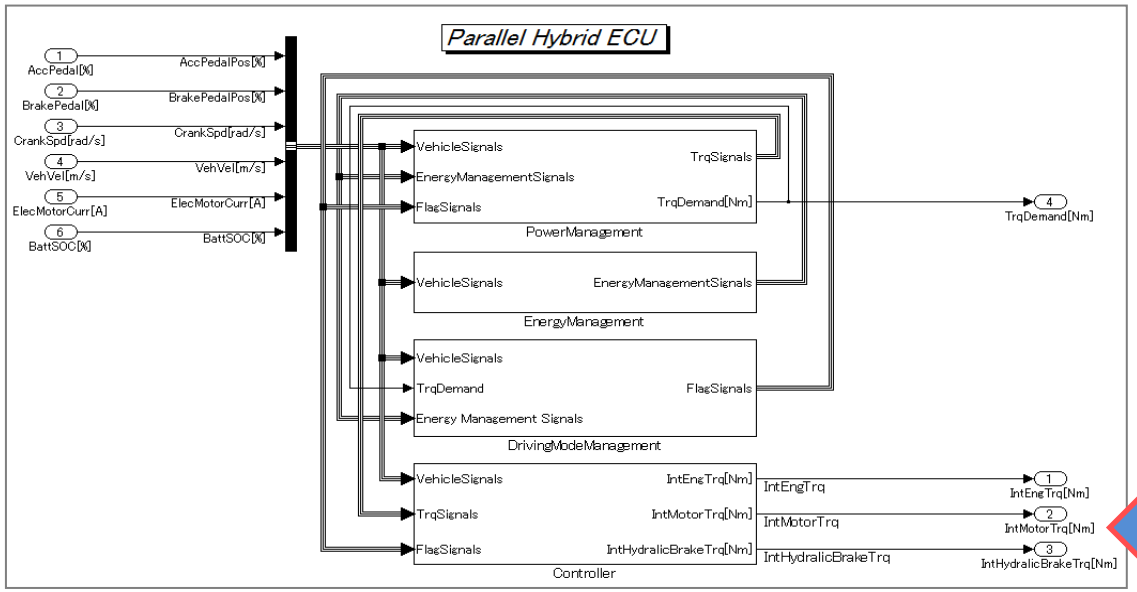
制御対象	ACモータ （モータとエンジンを搭載するハイブリッド自動車モータが制御対象）
制御周期	100 us
制御アルゴリズム	ベクトル制御、干渉除去制御、 オブザーバによる外乱除去、センサレス制御
入出力	入力: 目標トルク, 三相電流値 出力: PWM三相電圧値

タスクフローグラフ

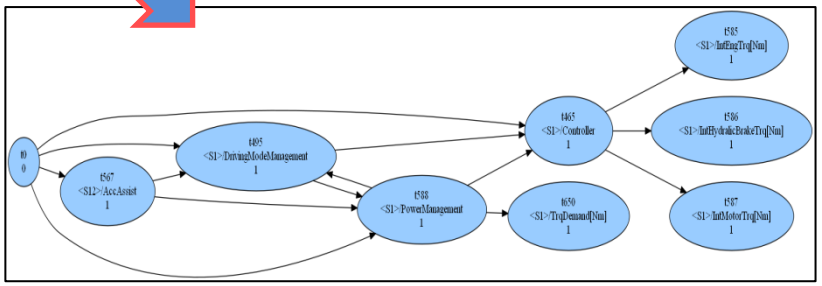


トルク分配制御コントローラ概要

コントローラモデル



タスクフローグラフ



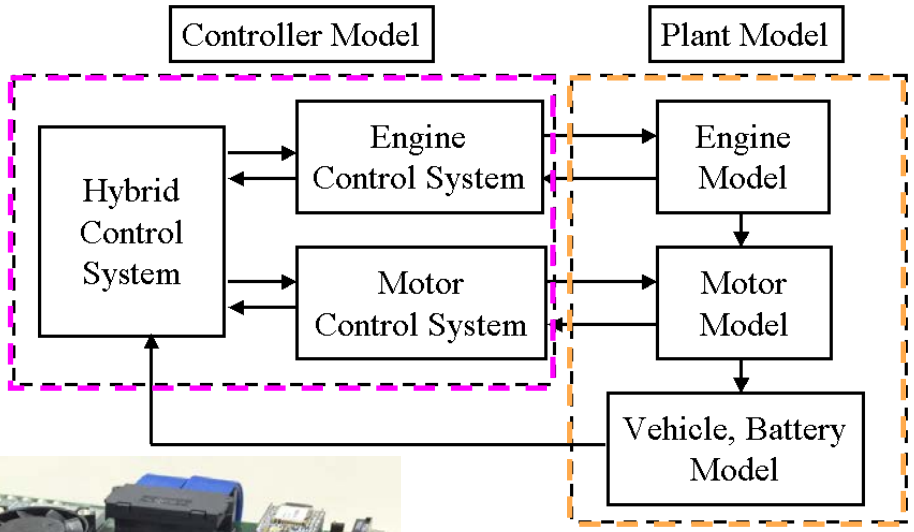
タスク数: 8個

制御対象	モータとエンジンに分配するトルク量を制御
制御周期	1 ms
制御アルゴリズム	簡単な計算式とテーブル参照でトルク分配量を計算
入出力	入力: アクセル、ブレーキ、車速、電池量、等 出力: モータトルク、エンジントルク、ブレーキトルク

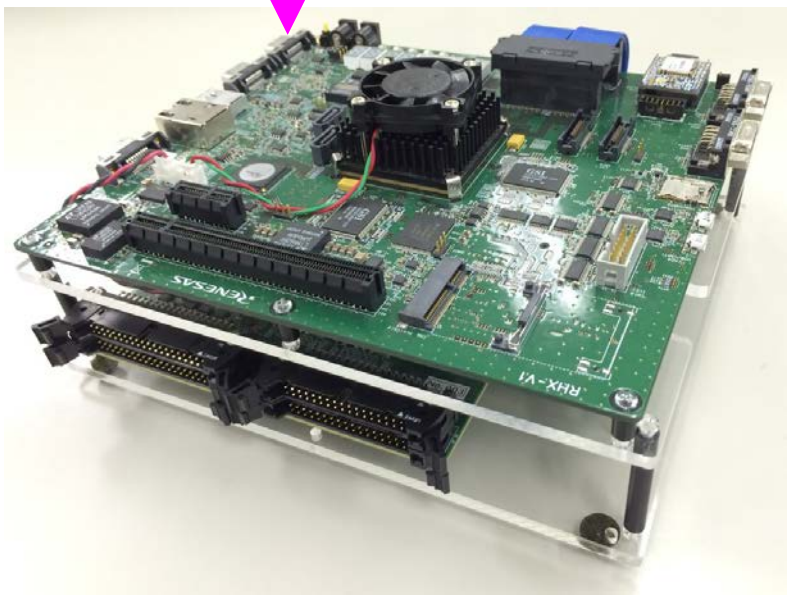
※制御周期1msに対し、シミュレータ実測による計算量は4,000サイクル程度で、並列化不要と判断

HEV制御HILS システム構成

コントローラモデル
・HEV制御システム



プラントモデル
・エンジンモデル
・モータモデル
・車両モデル

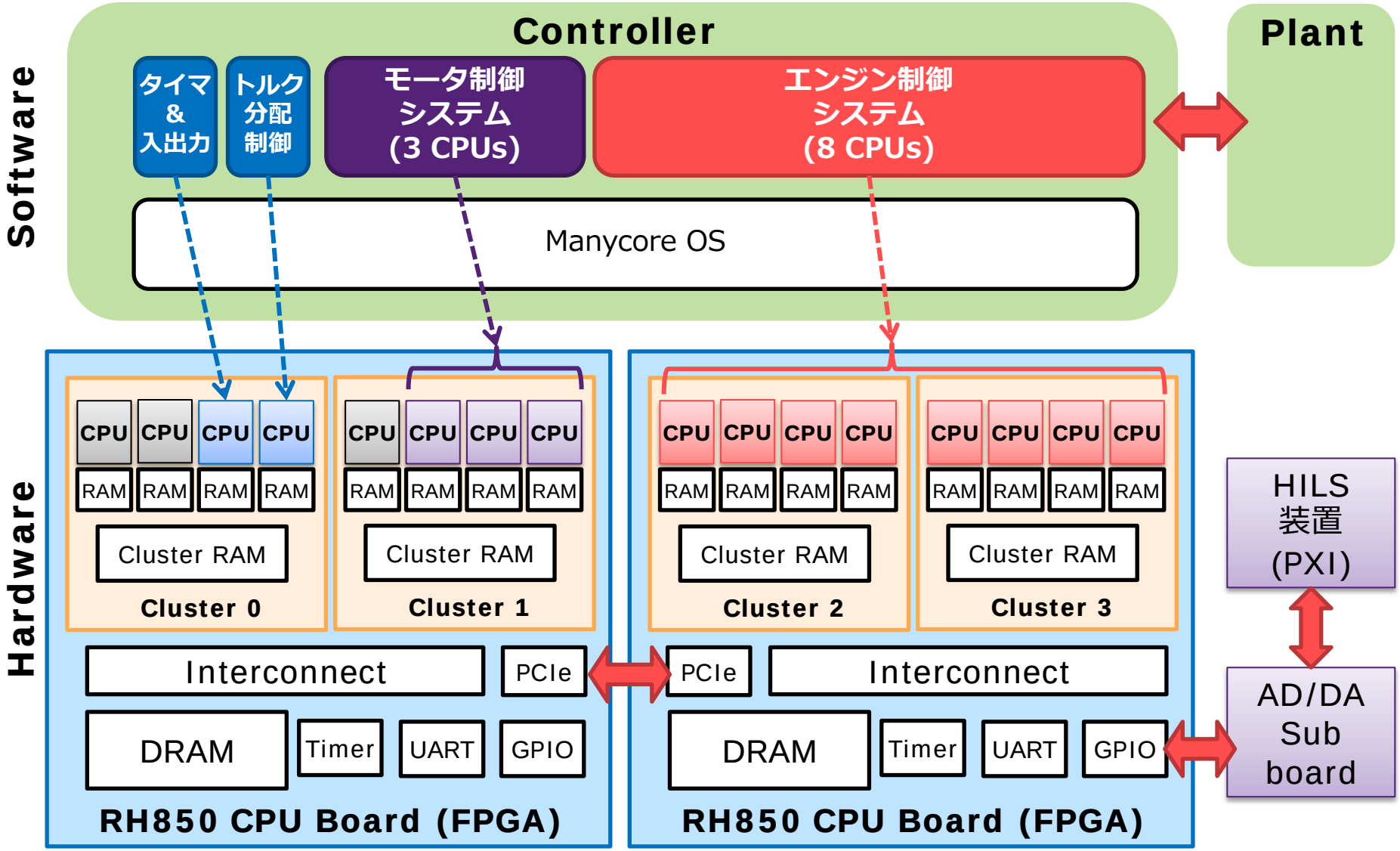


16コア搭載MCUエミュレーションボード

HEVシミュレータ
（HILS）

HILS: Hardware in the Loop

HEV制御HILS システム構成



HEV制御アプリ例による並列化評価結果

- モデル予測制御を適用したエンジン吸排気制御を並列化
  ⇒ 逐次実行に比べて、8並列で5倍以上の性能比

- 要求制御周期 4msに対しては. . .
  - 逐次実行では1GHz超CPUが必要
    ⇒ 車載CPUでの実現は、消費電力的に非常に困難
  - 並列実行では200MHz 8コアで要求を達成

メニーコアならば、より低消費電力で、
モデル予測制御を用いた最適化問題を解き
燃費・快適性・加速性能を向上させることが可能

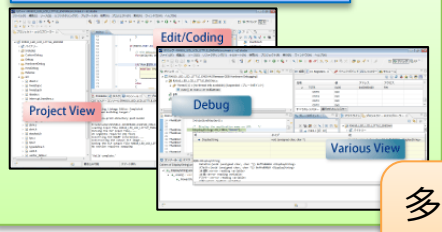
要求制御周期　　エンジン吸排気制御 = 4ms

アプリ	並列数	ボード実測値 24MHz	200MHz換算	性能比
		Worst（ms）	Worst（ms）	
エンジン制御	1 コア	168.1	20.2	1.00
	8 コア	29.0	3.48	5.79

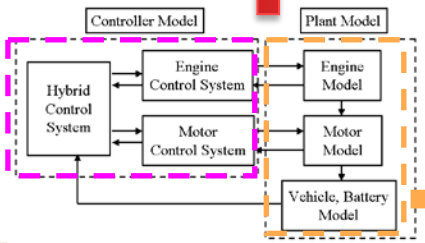
ET展デモ環境

モデルからコード生成

統合開発環境



コントローラモデル
HEV制御システム

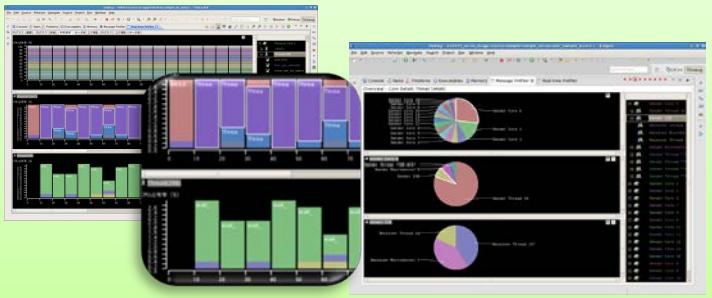


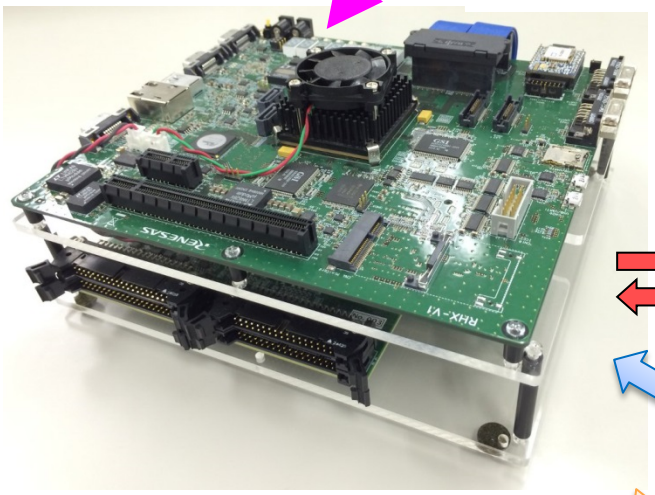
プラントモデル
・エンジンモデル
・モータモデル
・車両モデル

多数のコアの情報を
整理して分析・改善

イーソル メニーコアOS & 解析ツール

Realtime Profiler　　Message Profiler





メニーコア評価ボード

HILS装置
(PXI)

ホストPC

E1エミュレータ

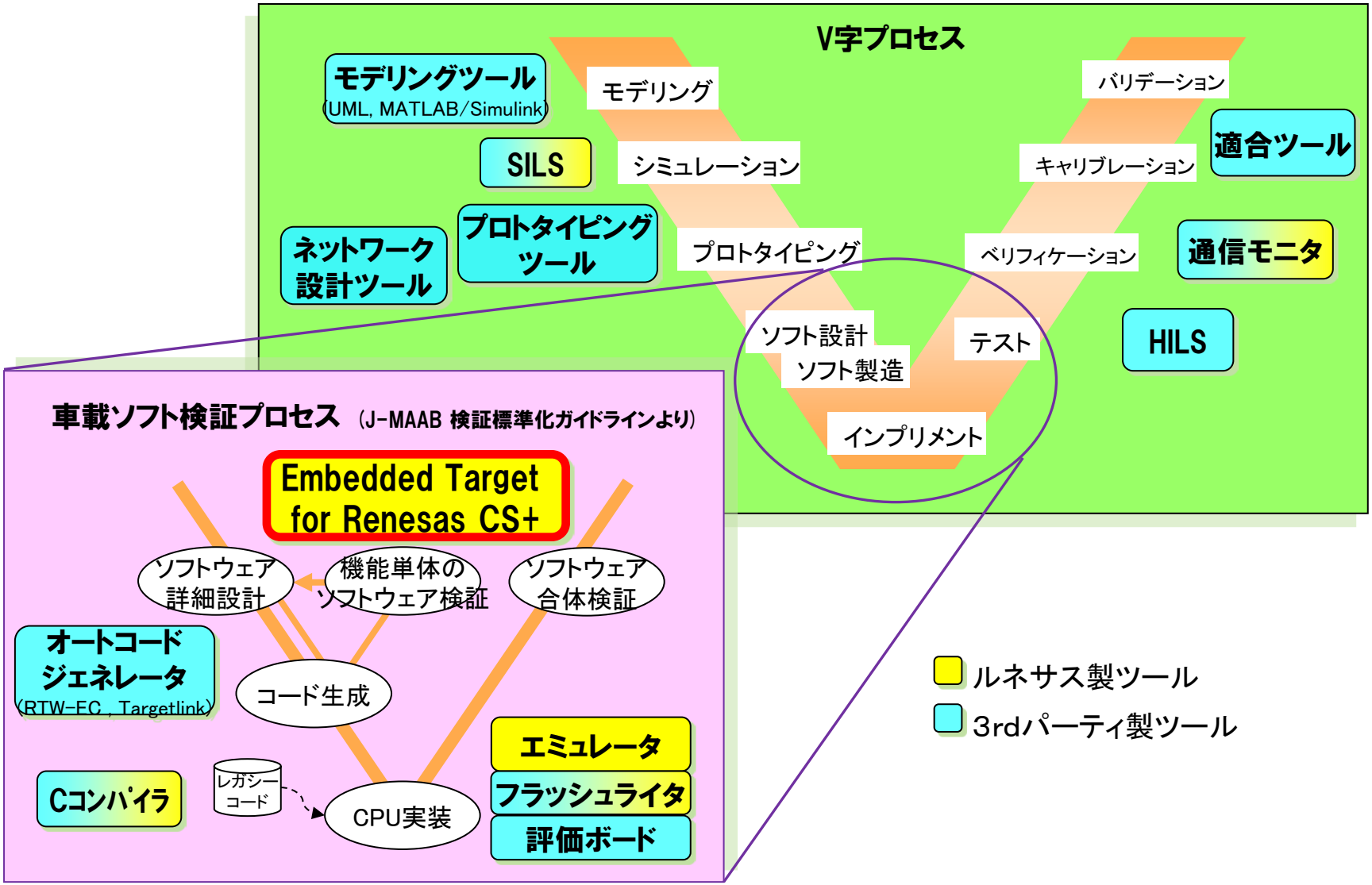
HEV制御の動作情報
(LabView)

先進的なモデル予測制御を
メニーコアMCUで実現

”一気通貫のマルチ／メニーコア開発環境” を実現

マルチ・メニーコアとモデルベース開発

ルネサスが提供する車載ソフト検証プロセスの環境



Embedded Target for Renesas CS+ 概要

- MATLAB/Simulinkと、ルネサス社製統合開発環境CS+を連携させて、アルゴリズム検証を行い、お客様のモデルベース開発を支援します

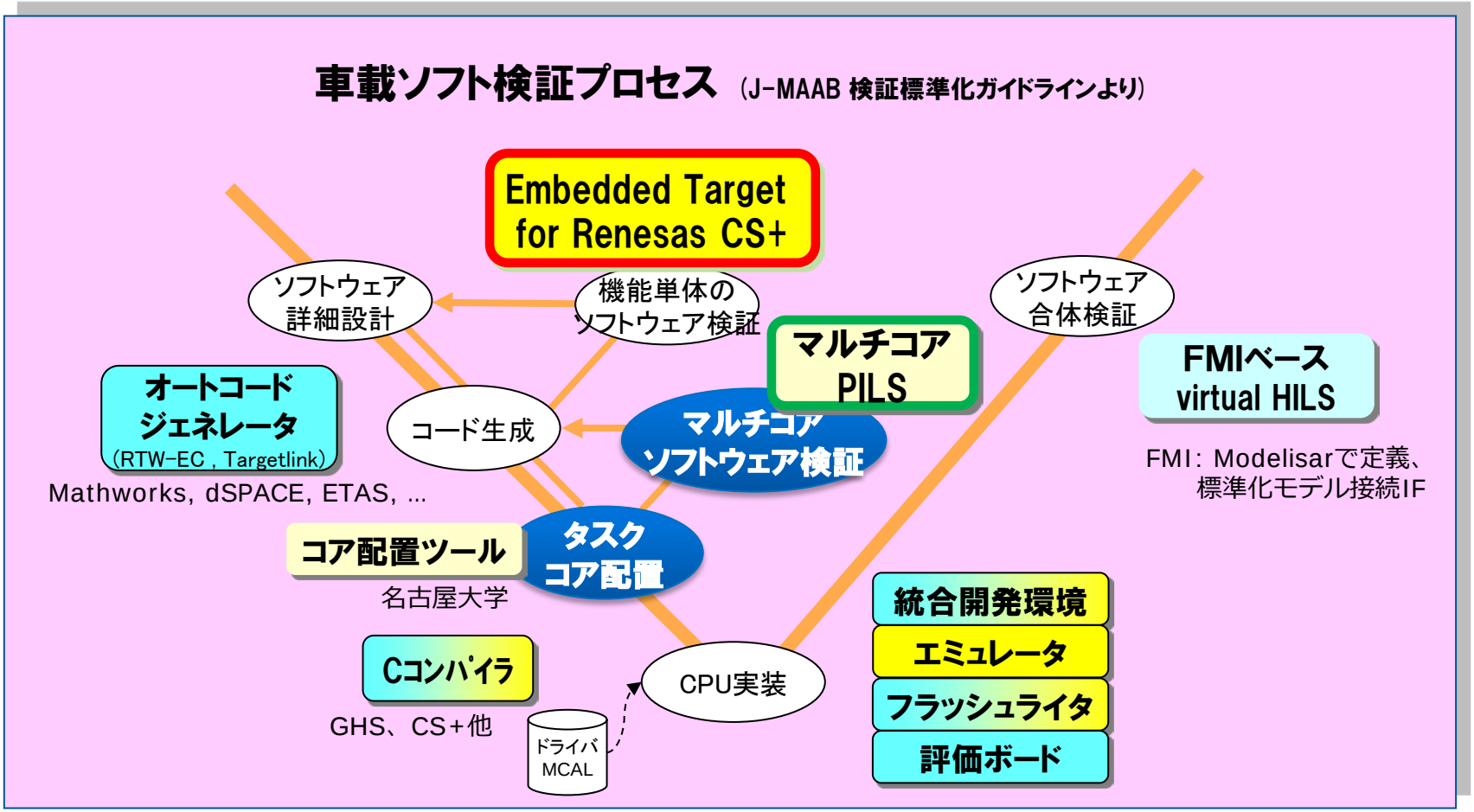
- 特長・機能

・ Simulinkモデルにおけるアルゴリズムをターゲット上で実行するPILS環境を構築
  - Simulink Coder, Embedded CoderによるCコード生成
  -統合開発環境CS+によるCコードの自動ビルド
・ モデルと実コードによる特性の違いを評価
・ ターゲットでの実行時間を測定
・ 統合開発環境CS+が対応する各種マイコンをサポート

（PILS: Processor In the Loop Simulation）

（マイコンの対応時期は個別にお問合せください）

マルチコアに必要な車載ソフト検証プロセスの環境



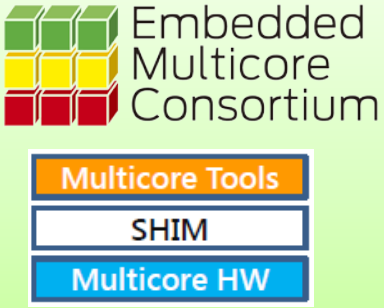
Embedded Target for Renesas CS+ :　アプリSW(制御) + プラントで検証
　　デバイス・シミュレータを制御し、Simulinkとの協調シミュレーションを行う。
Virtual HILS :　アプリSW+ドライバSW(制御) + プラントで検証

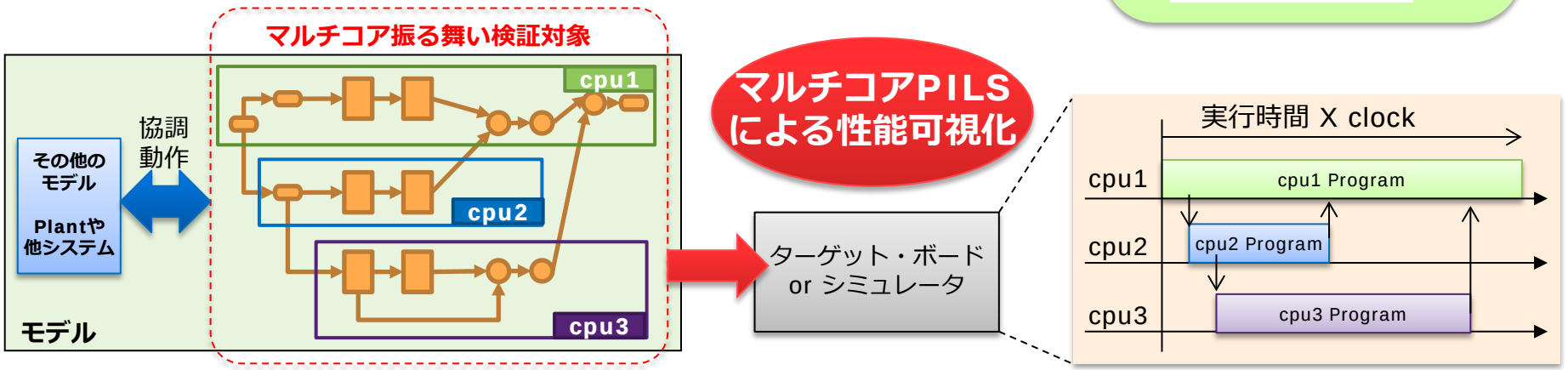
ルネサス製ツール
研究中
3rdパーティ製ツール
3rdパーティ企画中

検証支援：マルチコアPILS環境

- 制御システムをマルチコアで動作させた時の振る舞いの早期検証
  - システム動作の妥当性検証
  - 並列実行時の実行時間の把握
    - 相互関係（同期・順序など）を含めた動作を再現
  - ボトルネック解析
    - 効率的にマルチコアを利用できているか？

→ツールとの連携が鍵

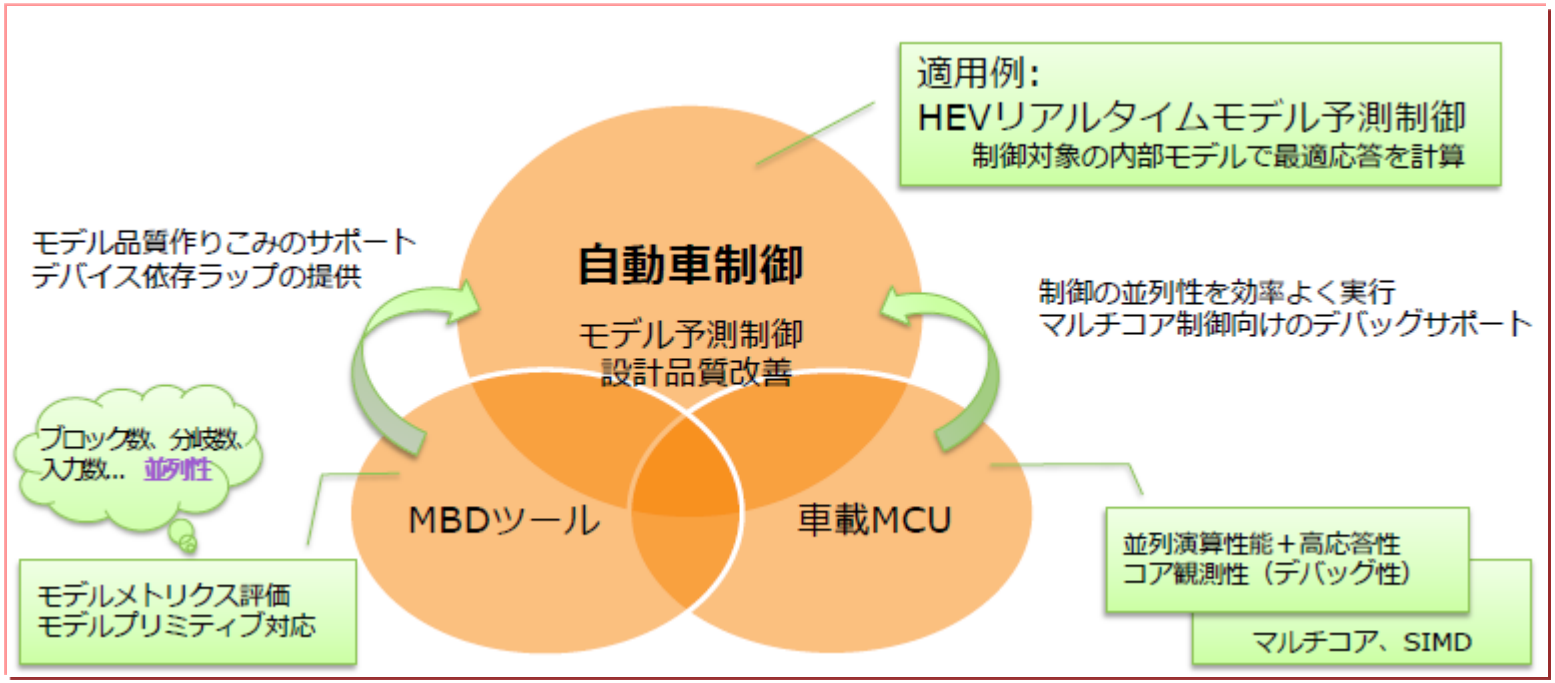




マルチ・メニーコアで実現する次世代自動車制御

- ◆ 制御性追求＝モデル予測制御　⇒　演算量限界のブレークスルー
- ◆ 設計品質追求　⇒　ACGのマルチコア対応
  マルチコア検証環境の充実

MBDマルチコア対応を制する者が次世代制御を制する



ルネサス エレクトロニクス株式会社