制御系マルチコア ハードウェアの特徴とユースケース

ルネサス エレクトロニクス株式会社
IoT・インフラ事業本部/コアIP開発統括部
技術ソリューション企画部

鈴木 均（hitoshi.suzuki.ue@renesas.com）

制御ソフトウェア大規模化とマイコンへの要求の増大

大規模ソフトウェアの動作のためにマイコンへの要求性能が増大
⇒ マルチコア・マイコンが必要

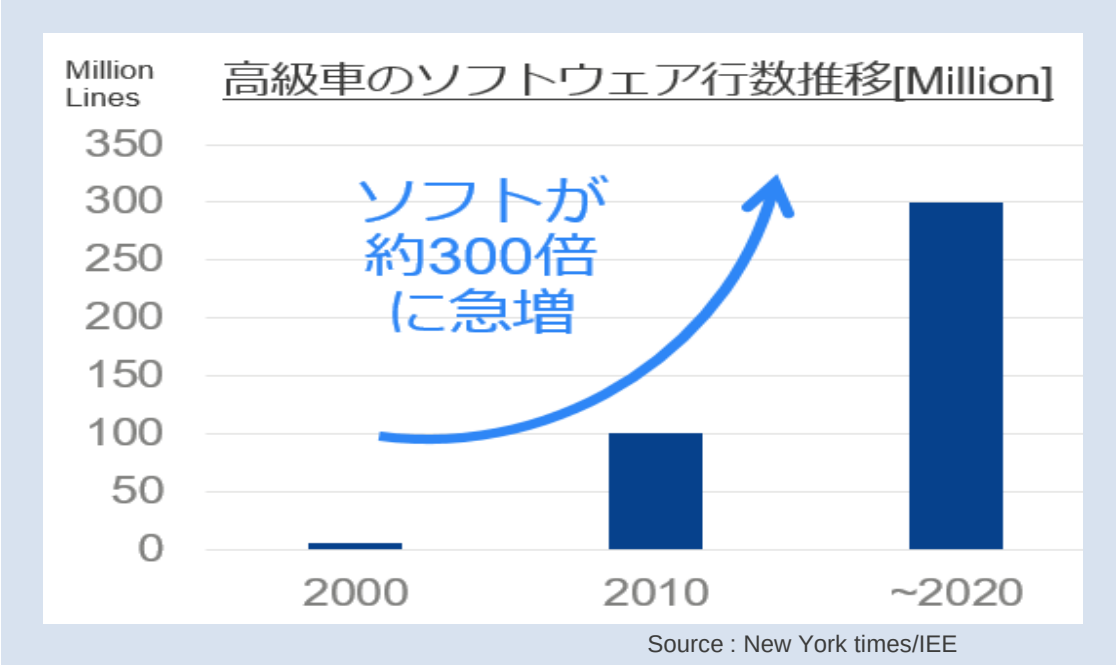
スマートフォン
1200万ライン

ソフト量の
比較
(2010年時点)

高級車
1億ライン

高級車の全ソフト量はスマートフォンの約8倍
自動運転時代に向け、さらなる増大が見込まれる

Source: 各調査会社のデータから当社が推定

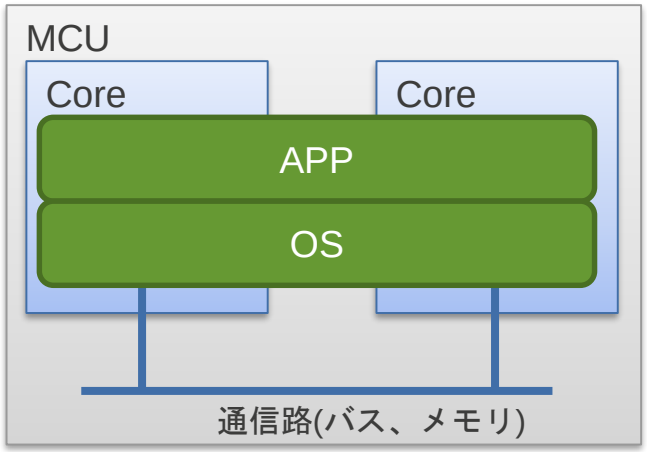


Source : New York times/IEE

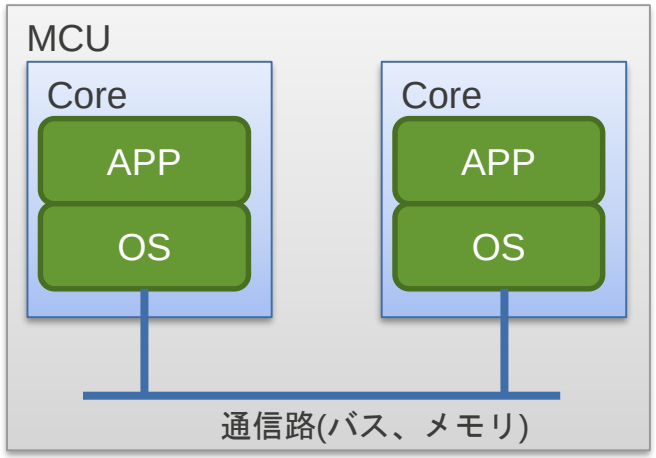
制御システムにおける
マルチコアユースケース

制御システムにおけるマルチコアの代表的ユースケース

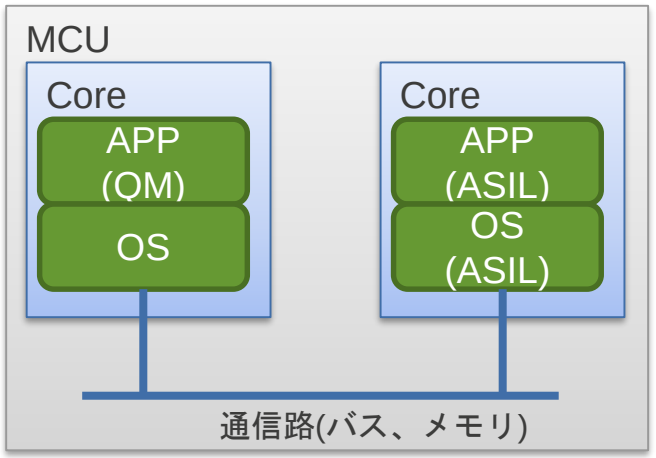
- 制御アプリケーションの高性能化 ⇒ 負荷分散型
- 複数のアプリケーションの組合せ・統合 ⇒ 機能統合型
- 機能安全への対応 ⇒ 機能分離・時間分離



負荷分散型マルチコア



機能統合型マルチコア

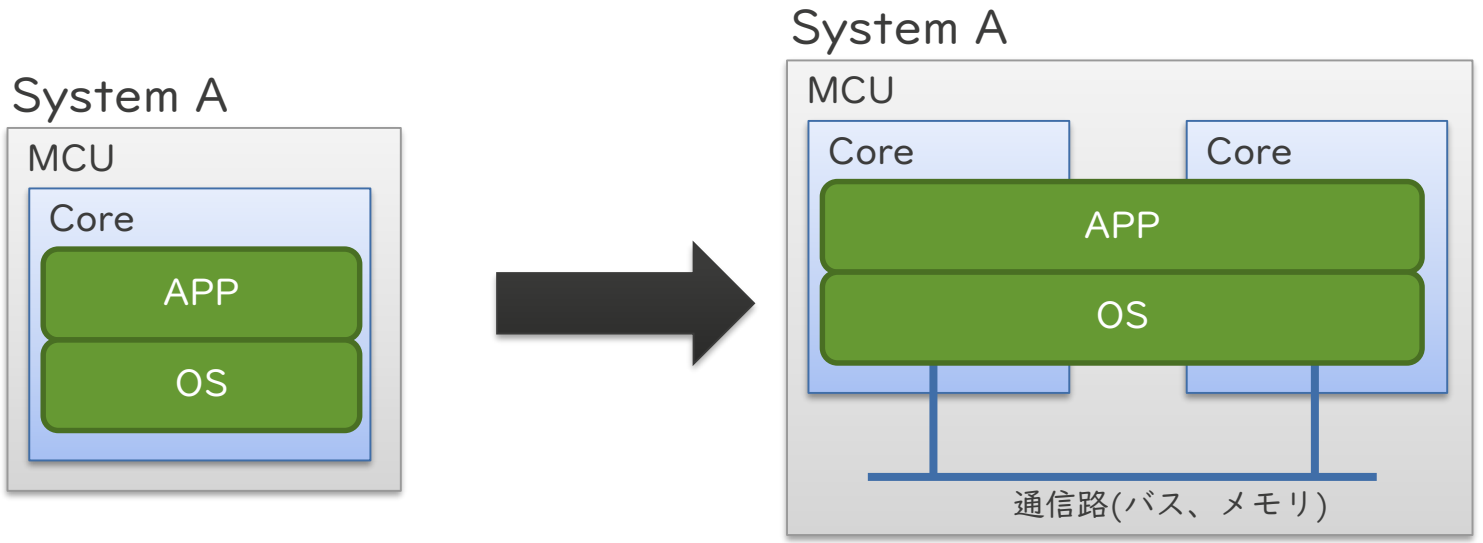


機能分離・時間分離

ASIL: Automotive Safety Integrity Level
QM: Quality Management Level (非安全要件)

アーキテクチャ・パタンI: 処理の高度化(負荷分散)

目的:
　単一のシステムを分割し並列実行することで、実行性能を向上させる



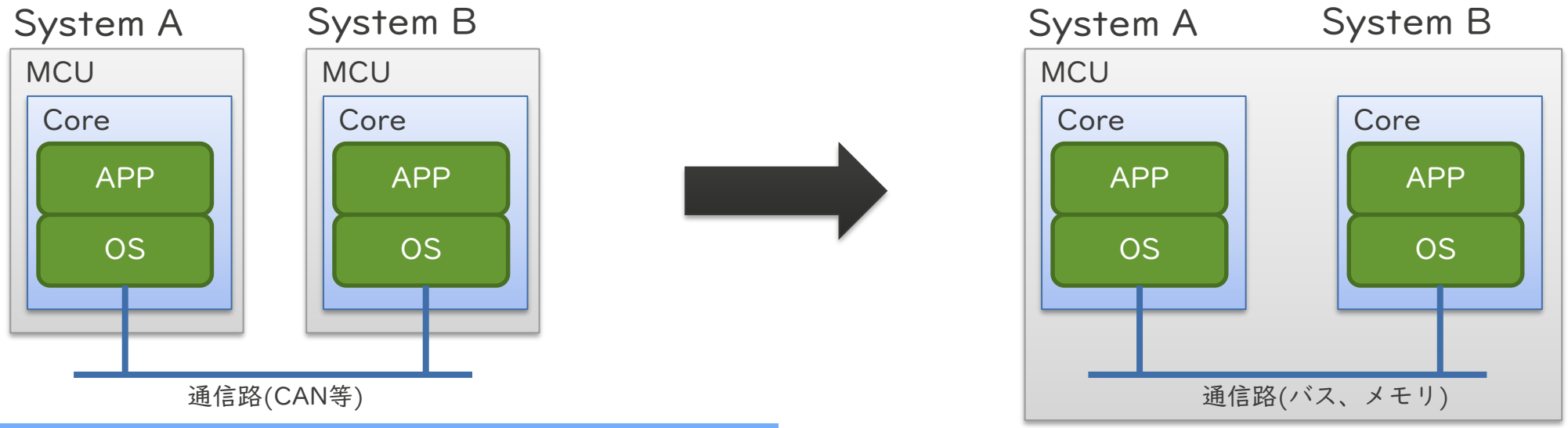
利点
- ✓マルチコア化による実行性能向上

欠点
- ✓並列化に適したアプリ構成でない場合、性能が期待通りに向上しない
- ✓バスや共有メモリを介したデータ交換のオーバーヘッドが生じる

必要とされる技術
- ✓並列化の粒度: アプリケーション以下
- ✓ソフトウェアの並列化支援ツール
- ✓マルチコアOS
- ✓高速な共有メモリ
- ✓高速なCPU間イベント通信

アーキテクチャ・パタン2: システムの統合

目的
複数のシステムを統合し、BOMコストを削減する



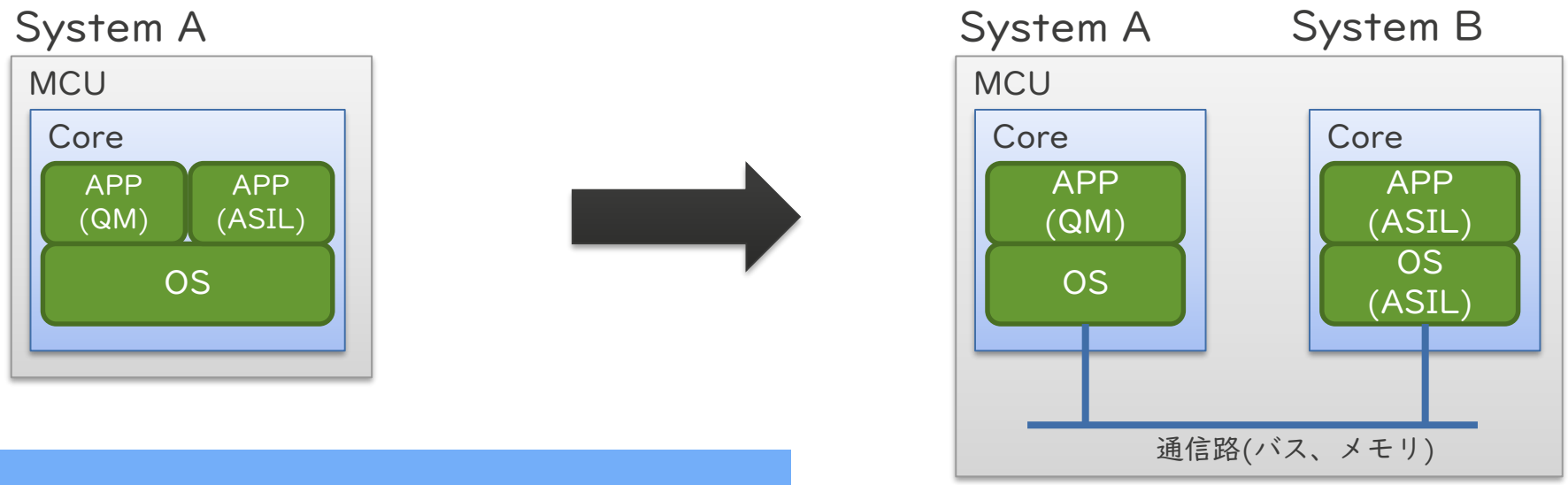
利点
- ✓部品点数削減によるBOMコスト低減
- ✓通信路がチップ内部で閉じるため、通信レイテンシが削減され、効率的な協調動作が行える

欠点
- ✓バスやメモリの競合による性能劣化(干渉)が発生
- ✓統合により共通化された外部部品などの故障により、共通故障の影響範囲が大きくなる問題が生じる

必要とされる技術
- ✓並列化の粒度: アプリケーション単位
- ✓仮想化技術／ハイパーバイザ
- ✓OS間通信フレームワーク
- ✓メモリ・周辺装置のパーティショニング技術
- ✓干渉の少ないバス・システム

アーキテクチャ・パタン3: 機能分離・時間分離

目的
　システムを要求特性(ASIL等)で分割し、複雑さを減少させ、検証性・説明性等を確保する



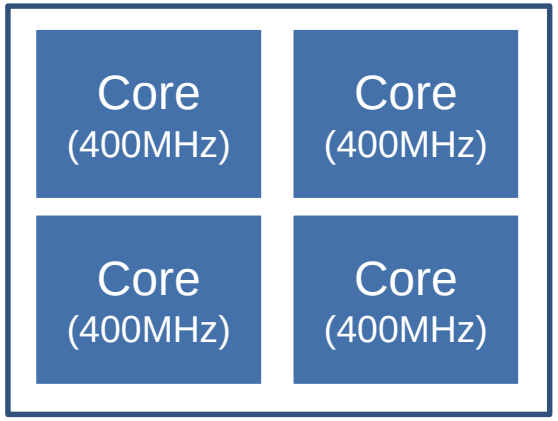
利点
- ✓異なる性質のソフトウェア間の干渉が低減できる
- ✓OSごと分離することで、アプリの要求特性に応じたコンパクトなOSを利用できる

欠点
- ✓バスや共有メモリを介したデータ交換のオーバーヘッドが生じる

必要とされる技術
- ✓並列化の粒度:アプリ単位 or アプリ以下
- ✓仮想化技術／ハイパーバイザ
- ✓共有メモリ
- ✓メモリ・周辺装置の分離技術
- ✓帯域分離が可能なバス・システム

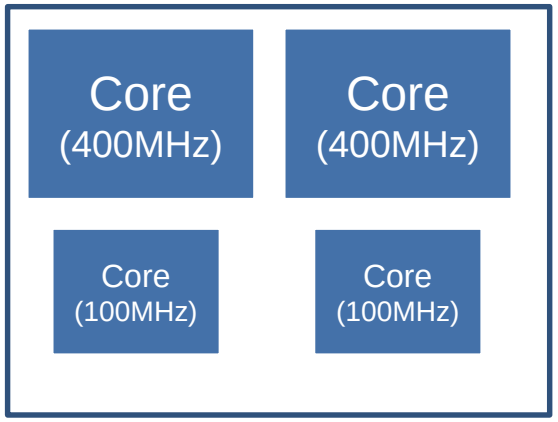
制御系マルチコア・ハードウェア

マルチコアの種類



■ 対称型マルチコア
(ホモジニアス)

✓ソフトウェアを分割配置する際、結果が予想しやすい



■ 非対称型マルチコア
(ヘテロジニアス)

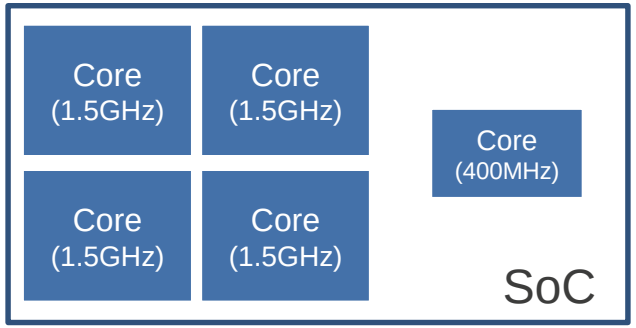
✓チップの使用目的に応じた役割分担が明確な場合、最適

■ 制御系 (MCU:マイクロコントローラ)

✓比較的小規模な処理の集合のため、分割配置しやすいホモジニアス構成が主流

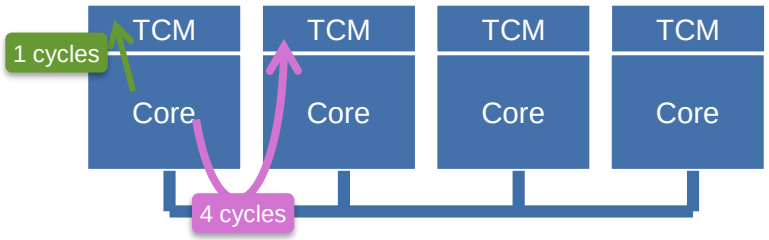
■ 情報系 (SoC:システム・オン・チップ)

✓高スループットを実現する高性能コアを複数と、システム制御を行うリアルタイム処理コアのヘテロジニアス構成が主流

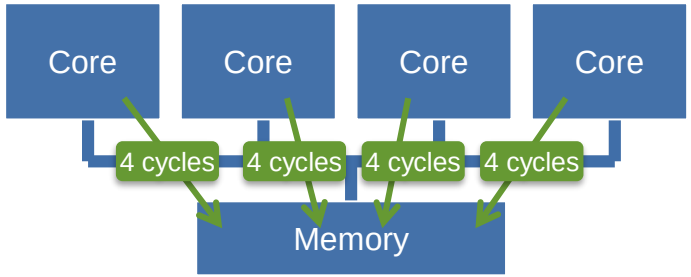


メモリ構成

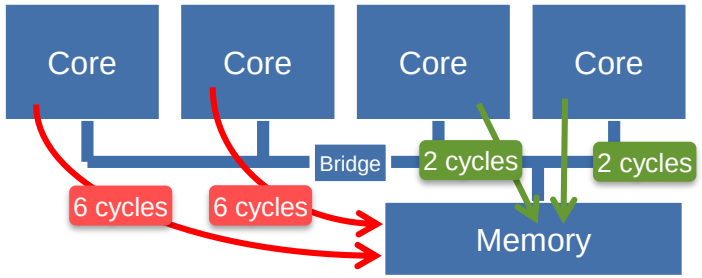
メモリ構成にはそれぞれ特徴があり、それらを生かした運用が必要となる



- TCM: Tightly coupled memory (Local memory)
  - 各コアが高速に利用できるメモリとして、コア内部に持つ小容量のデータ・メモリ
  - 主にスタックや、重要なデータを置く
  - メモリ操作を行う際に、待ち時間(ペナルティ)が発生しない または 非常に少ない
  - 通常、TCMにはそのコアの処理でのみ利用するプライベートなデータを配置するが、共有メモリとしても利用できるように他コアからのアクセスを許容する仕様を採る場合がある

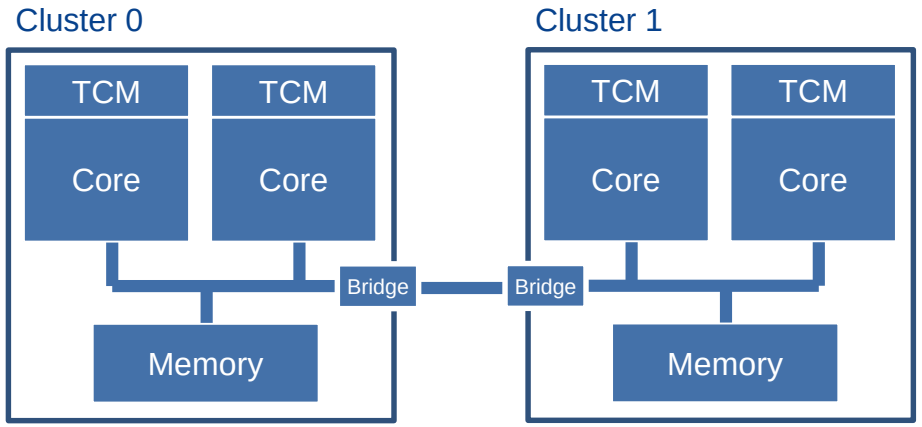


- 等距離共有メモリ
  - すべてのコアからのアクセス性能が同一である共有メモリ
  - ソフトウェアをどのコアに配置しても性能が大きく変化せず使いやすい
  - コア数の増加に伴い、アクセス性能が低下しやすい



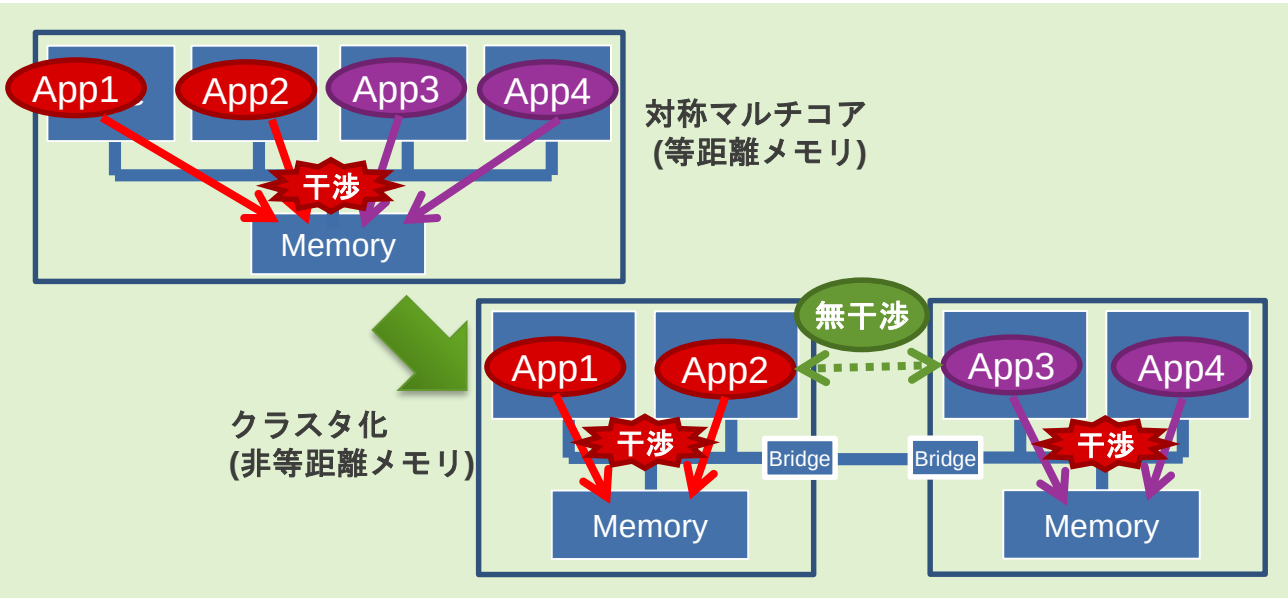
- 非等距離共有メモリ
  - コア毎にアクセス性能が異なる共有メモリ
  - 近傍のコアに対しての性能が良く、コア数が増えた場合も近傍コアでは性能劣化が少ない
  - ソフトウェアを移動した場合に大きく性能が低下するため、配置計画の際には注意が必要

クラスタ構造

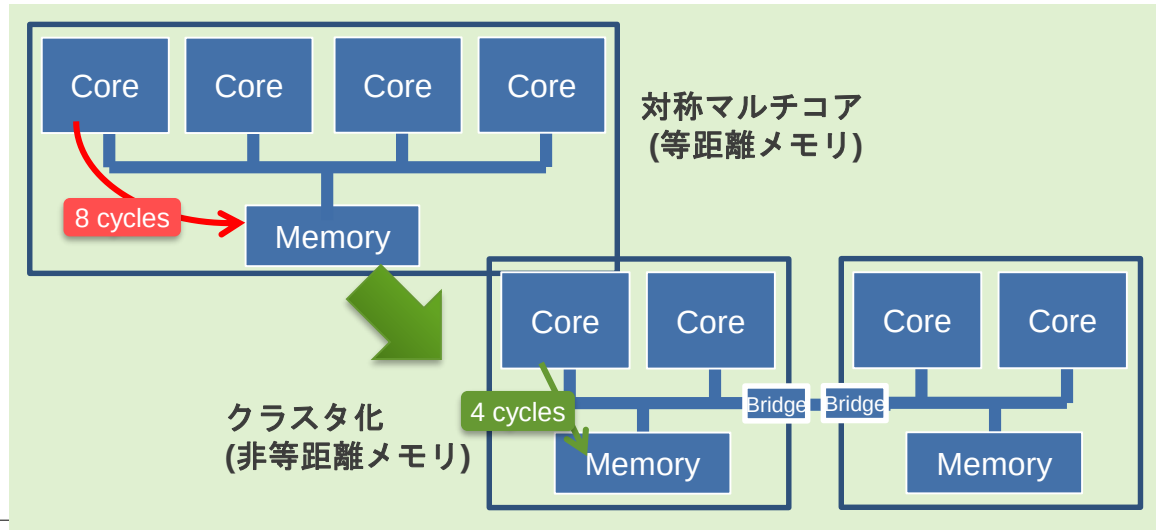


- 複数のコアを結合し、階層的にマルチコア・システムを構成する場合に、結合した個々の塊をクラスタと呼ぶ
  - サブシステム、アイランドなどと呼ばれることもある
  - クラスタ構造を採ることでハードウェア性能の最適化が実現できる
    - ✓クラスタ毎に回路レイアウトを行うことで、メモリアクセス性能や周波数性能等の引き上げが可能
    - ✓電力管理など、ハードウェア資源の階層的な管理単位としても利用される

相互干渉の低減



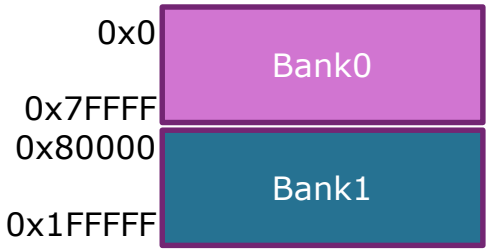
性能引き上げ

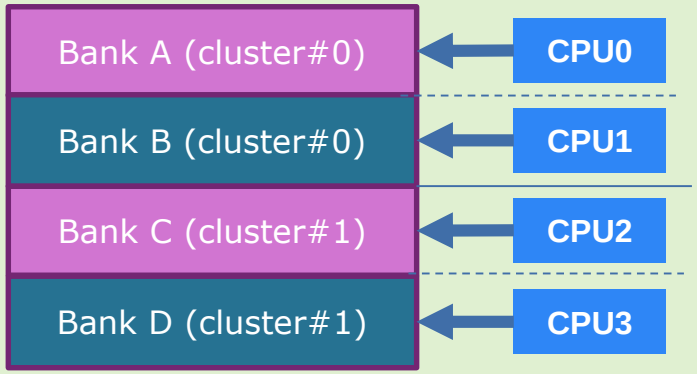


メモリ・インターフェース

■メモリ・バンク

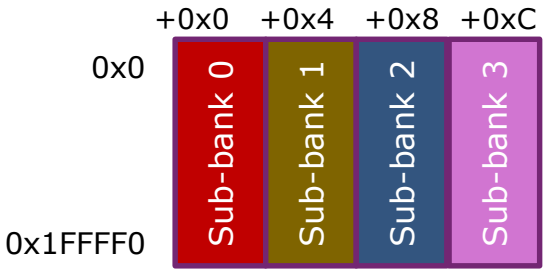
- ✓ アドレスの上位側の範囲で分割
- ✓ ソフトウェアで意識的に競合の回避をコントロール可能
- ✓ コンパイラのメモリ・セクションを用いて、個別に利用する





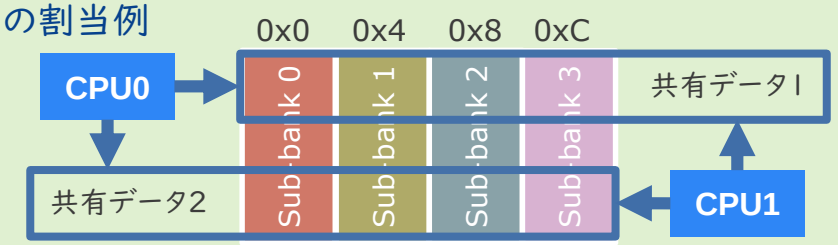
■メモリ・サブバンク

- ✓ アドレスの下位側の範囲で分割する方法
- ✓ ソフトウェアで意識しなくても、確率的にアクセス効率が向上する
- ✓ シングルCPUによる連続したアドレスへのアクセスでも効果あり

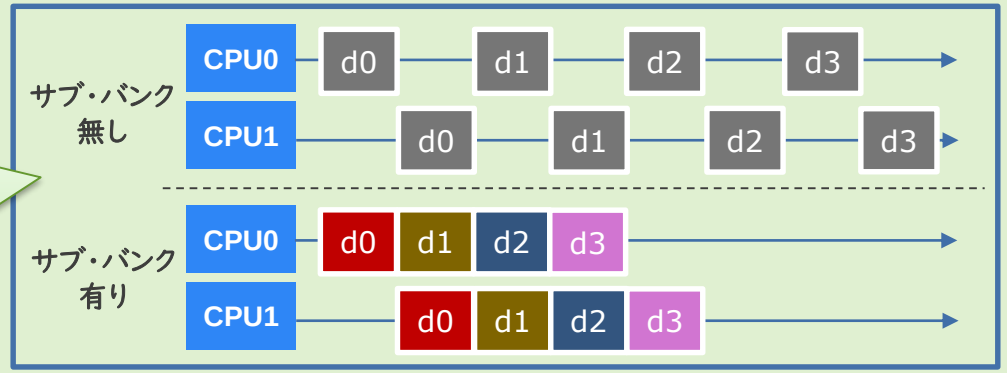


- ➢ 同じバンクに置いた共有データを異なるCPUから同時にアクセスする場合、競合が発生しにくい
- ➢ また、連続アドレスへのアクセス時には、ずれながらアクセス出来て、ペナルティは最小限となる

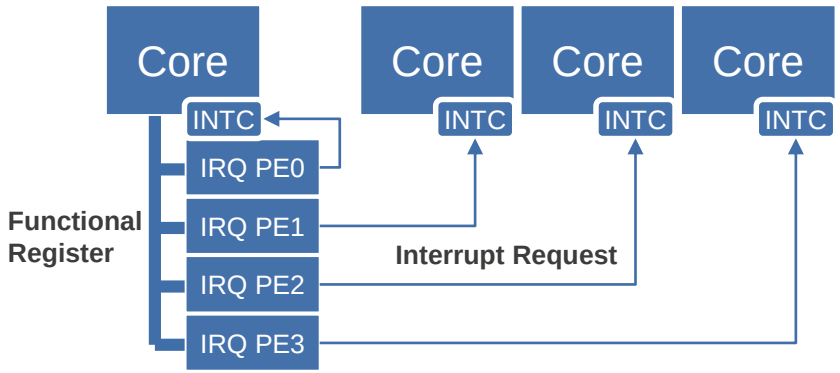
共有データの割当例



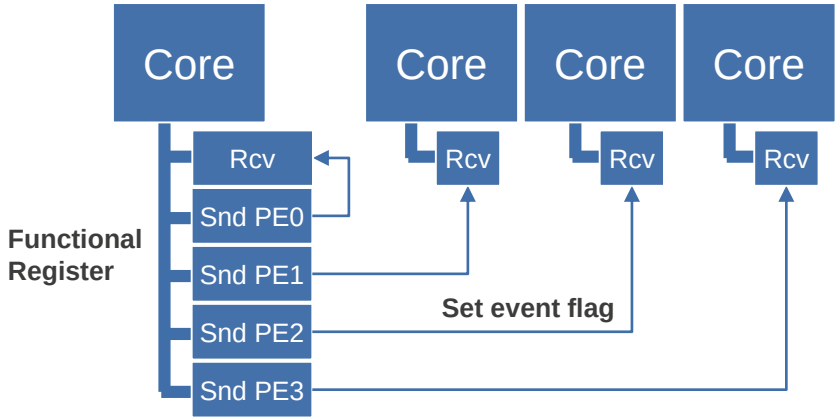
動作タイミング例



プロセッサ間割込み／イベント(メッセージ)通知



IRQ: Interrupt request register



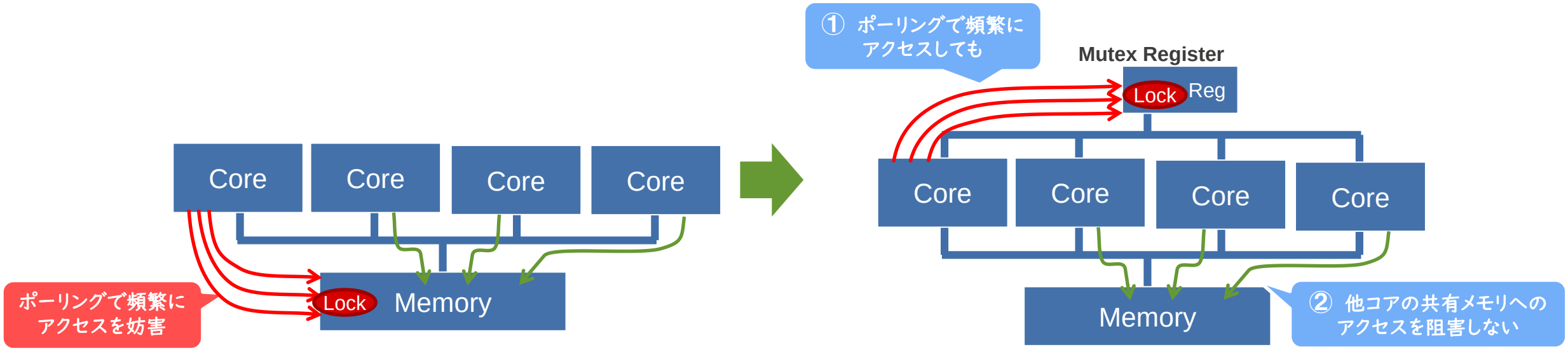
Snd: Event Send register
Rcv: Event Receive register

- プロセッサ間割り込み
  - プロセッサ間の処理起動に用いる
    - ✓ RPC: Remote Procedure Call
  - ソフトウェアから任意のコアへ割込みイベントを発生させる
  - 割込みのマスクや、同時に複数コアに通知する機能など、付加的な機能もあり

- プロセッサ間イベント(メッセージ)通知
  - 他のプロセッサに対して、あるイベントが発生したことを通知する
    - ✓ イベント内容はソフトウェアからセットする
    - ✓ フラグ(1ビット)の情報だけでなく、一定のサイズのメッセージを乗せられる場合もあり
    - ✓ 共有メモリと違い、送信者と受信者が明確であることが優位点
  - プロセッサ間割込みと同様の仕組みだが、受信側での処理起動が伴わず、任意のタイミングで受け取ることが可能であることが特徴
    - ✓ クリティカルなシステムでは、非確定性を排除するためにプロセッサ間割込みより重宝する
  - 受信側のプロセッサの状態が制御できる場合もある
    - ✓ スリープ状態を解除する／イベント待ち状態を解除する など

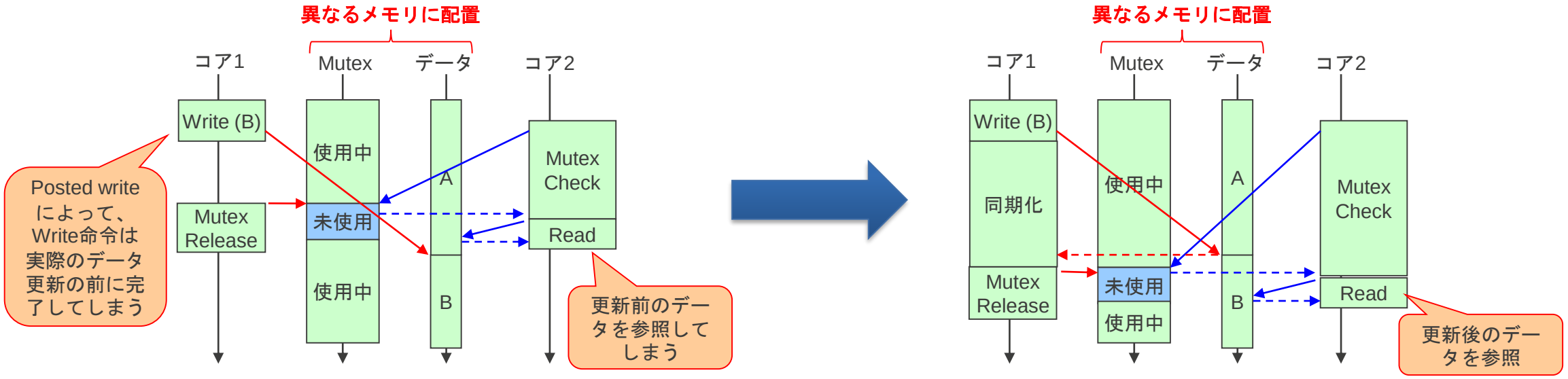
排他制御用レジスタ (Mutex register)

- プロセッサ間の排他制御(Mutex)の際の共有メモリアクセス負荷を低減するための専用のレジスタ機能
  - 通常、Mutexのためのロック変数は、共有メモリ上に配置するため、処理速度が低下する場合がある
    - ✓このため、あるコアが頻繁にロック変数にポーリングを行うと、共有メモリへのアクセスが集中し、他のコアの動作が阻害される
      - 他コアの処理速度が低下するため、結局、ロック変数を取得したいコアも、ロックが解放されずに処理能力が低下する
  - この問題を、共有メモリとはアクセス経路が異なる専用レジスタによって回避する
    - ✓専用レジスタへのアクセスは、共有メモリのアクセス経路には影響を与えない
  - 複数ビットを用いることで、バリア同期などにも利用可能



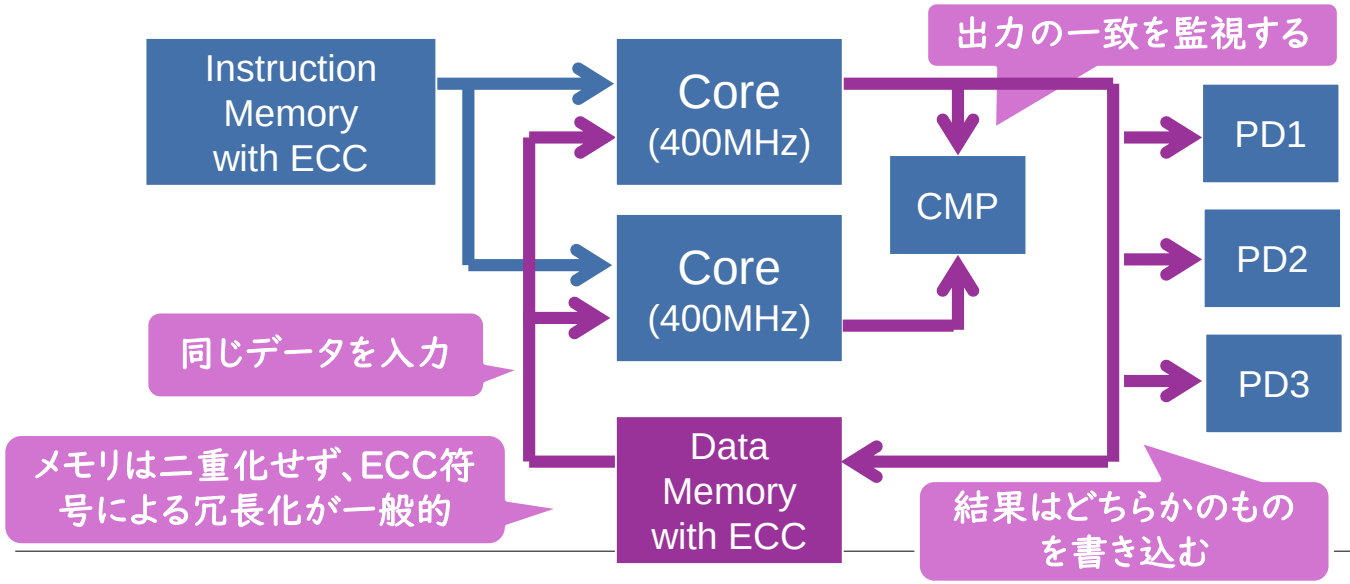
同期化処理

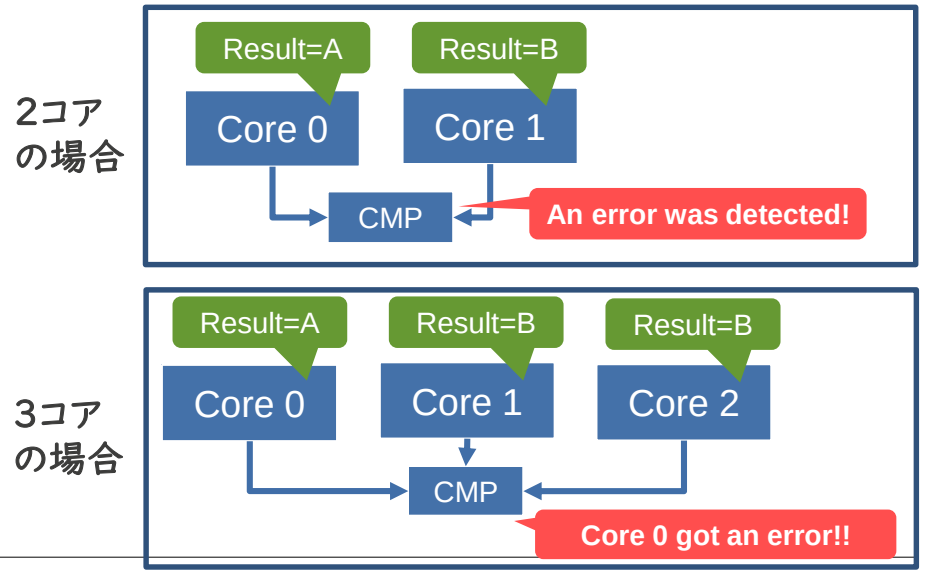
- 共有メモリの書き込み後に、書き込んだデータが他のコアから読み出せることを保証するために、同期化処理が必要となるケースが存在する
  - 同期化処理は、CPU内のバッファやバス・システムのバッファなどが滞留する書込要求が、真に書き込まれて、他のコアから読んでも、その内容が読めることを保証するための手続きである
- 同期化処理が必要なケース
  - MutexとそのMutexで管理する共有データを異なるメモリに割り当てている場合
    - ✓例: 排他制御レジスタ上のMutexで、共有メモリ上のデータへのアクセスを排他制御する場合
  - 同じメモリに割り当てている場合であっても、アーキテクチャがStoreの追越しを許可している場合(Weak Ordering)



(参考) ロックステップ・モード

- 複数のコアに同じ計算をさせ、結果を比較し、片方が誤った場合を検出する動作を、Lock-stepモードと呼び、特に高信頼が必要な分野で用いられる。
  - 2つのコアは同時、あるいは数クロックずらした状態で同一の入力を用いて動作する。
    - ✓したがって、出力は完全に一致するはずであり、相違がある場合はどちらかが動作を誤った状態であり、システムは速やかに安全な状態へと制御する必要が生じる
  - 2コアによるロックステップは、誤りの検出のみが行える。3つ以上の場合、多数決による誤ったコアを発見できる
  - ロックステップ・モードと、通常モードを切替可能なシステムも存在する
- ロックステップは、ソフトウェアからは1つのコアとしてしか利用できないため、アプリケーション開発観点では、マルチコアから除外して考える





各ハードウェア機能のユースケースごとの有用性

	負荷分散型	機能統合型	機能分離・時間分離	SHIMでの対応状況
対称型マルチコア	◎	○	○	○
非対称型マルチコア	X	△	△	○
TCM	○	○	○	○
等距離共有メモリ	◎	△	△	○
非等距離共有メモリ	X	○	○	○
クラスタ構造	X	◎	◎	○
メモリ・バンク	X	○	○	○
メモリ・サブバンク	○	○	○	△ (間接的にはアクセス時間に反映される)
プロセッサ間割り込み	△	○	○	○
プロセッサ間イベント(メッセージ)通知	○	○	○	○
排他制御用レジスタ	◎	△	△	○
同期化処理	◎	◎	◎	X

SHIMを活用することで効率的にマルチコア・ソフトウェア開発に関わるハードウェア機能を理解できる

同期化処理はハードウェアにより様々な違いがあり、定型化が困難
→ UM等をユーザが調査する必要あり

特に低レイテンシな通信が必要
処理を分割配置する際に対称性が重要

処理間の影響を低減する仕組みが有用
通信頻度が少なめで、分割した際の処理量にもばらつきがあるため、非対称であっても良い

◎:必須　○:有用　△:あれば嬉しい　X:問題あり

車載制御向けマルチコアの実例

製品: Renesas RH850シリーズのマルチコア機能

- Renesas
  - RH850シリーズ：RH850/E2x-FCC2
    - ✓ https://www.renesas.com/jp/ja/about/press-center/news/2018/news20180327.html
    - ✓ https://monoist.atmarkit.co.jp/mn/articles/1803/28/news039.html

マルチコア機能	RH850シリーズ
対称型マルチコア	○ (6コア:同一性能コア)
非対称型マルチコア	–
TCM	○
等距離共有メモリ	○ (クラスタ内)
非等距離共有メモリ	○ (クラスタ間)
クラスタ構造	○ (3クラスタ)
メモリ・バンク	○
メモリ・サブバンク	○
プロセッサ間割り込み	○
プロセッサ間イベント(メッセージ)通知	△ (プロセッサ間割込みの要求フラグで代用可)
排他制御用レジスタ	○

RH850の利用想定

■ 車載制御特性に合わせた高性能・機能統合を両立する クラスタ型マルチコア・アーキテクチャ

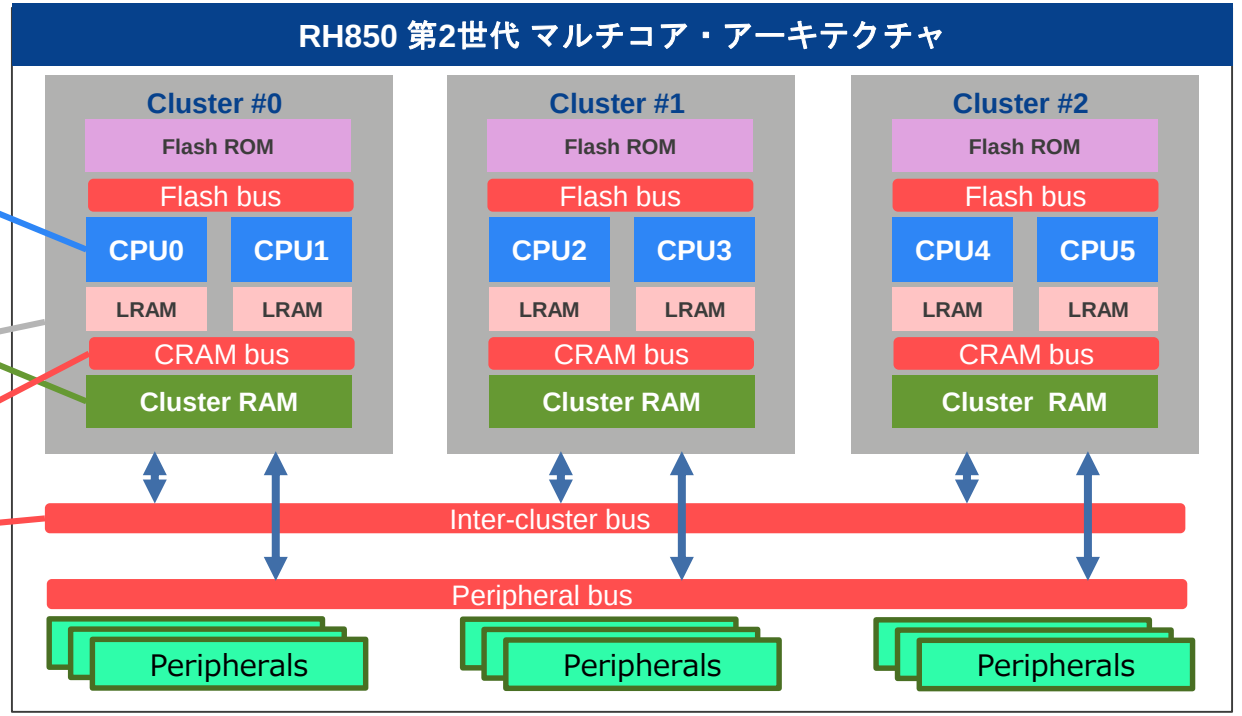
- ➢複数のアプリケーションを搭載する可能性あり
- ➢機能安全的観点から、資源／時間の分離性の重要性が高い
- ➢単体アプリの実行性能向上のニーズあり
  - ✓ただし、制御処理の並列性はあまり高くない (2~3並列が限界)
  - ✓行列演算などは、専用アクセラレータ(RH850の場合は、SIMD演算器)で対応

高スループットCPUコア

高速共有メモリで結合された
密結合マルチコア

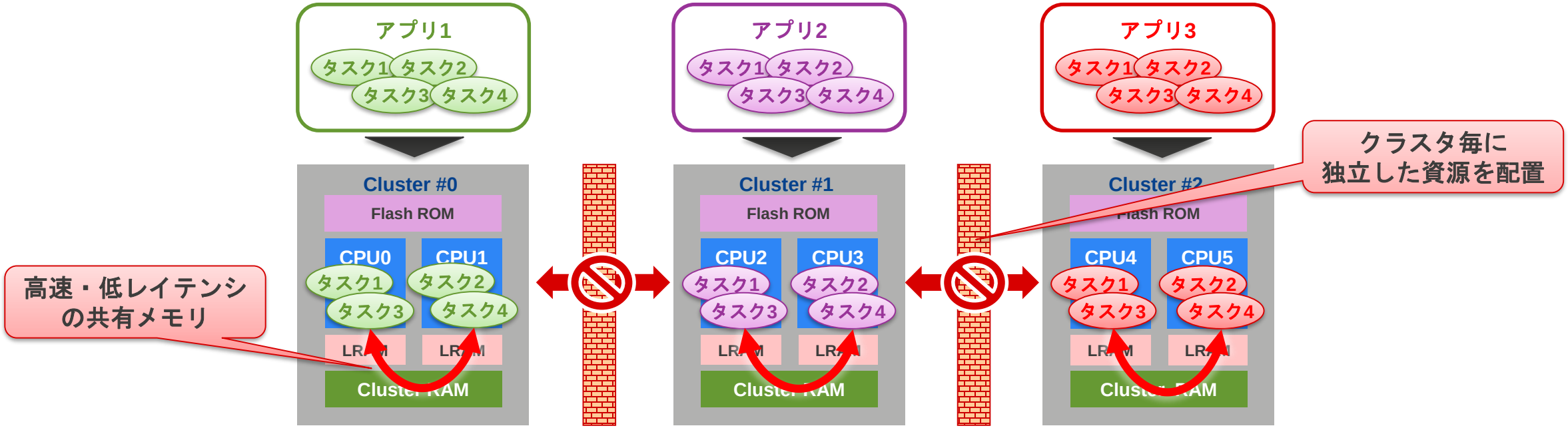
資源を分離した独立性の高い
クラスタ構成

予測性・非干渉性の高い
バス & メモリ機構



RH850 マルチコア・アーキテクチャの狙い

- 高速な共有メモリ(Cluster RAM)のデュアル・コアでアプリ処理を高速化
- 資源分離されたクラスタで Freedom from Interference を達成



機能統合ケース と 負荷分散ケース の組合せを想定

⇒ マルチコア・ソフトウェア開発を支援する仕組み・ツールが必要

S/Wレベルと並列化手法

機能統合

機能分離/
時間分離

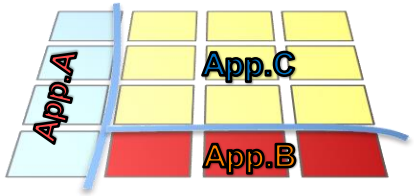
負荷分散

■ Application/Function level :

- ✓ 意味的な並列性。設計者の意図が、正しい解であることが通常
- ✓ コア毎に別々のアプリを動作させる
- ✓ SW 分割は容易だが、負荷の不均衡により、コアの利用率を上げるのが難しい

⇒ 設計者による指定で並行動作、**仮想化(ハイパーバイザ)、OS支援**

アプリ並列

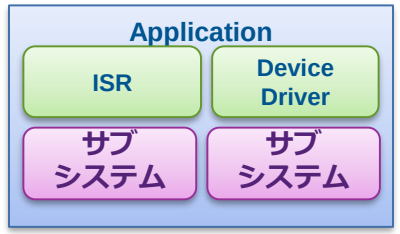


■ Inter TASK(Thread) level :

- ✓ 動作単位の性質により、設計者による分割は可能。
- ✓ ただし、動作タイミングを含めた関係性が既定しにくい
- ✓ アプリを構成するタスク群を別々のコアに割り当てて動作させる
- ✓ タスク配置の最適化や、排他制御などに高いスキルと多くの手間が必要

⇒ **OS支援、タスク配置支援ツール(計画／可視化)**

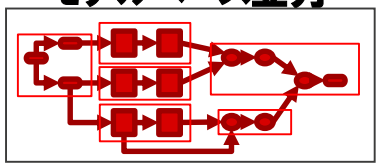
タスク（スレッド）並列



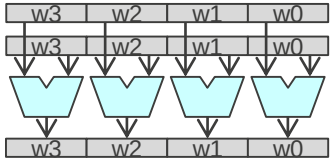
■ Inner TASK level :

- ✓ コード内の独立ブロックやループを別々のコアに割り当てて動作させる
- ✓ 手動で実現するのは手間がかかりすぎる。自動化ツールが必要
  - ⇒ **並列化コンパイラ、モデルベース並列化**

モデルベース並列



データ並列



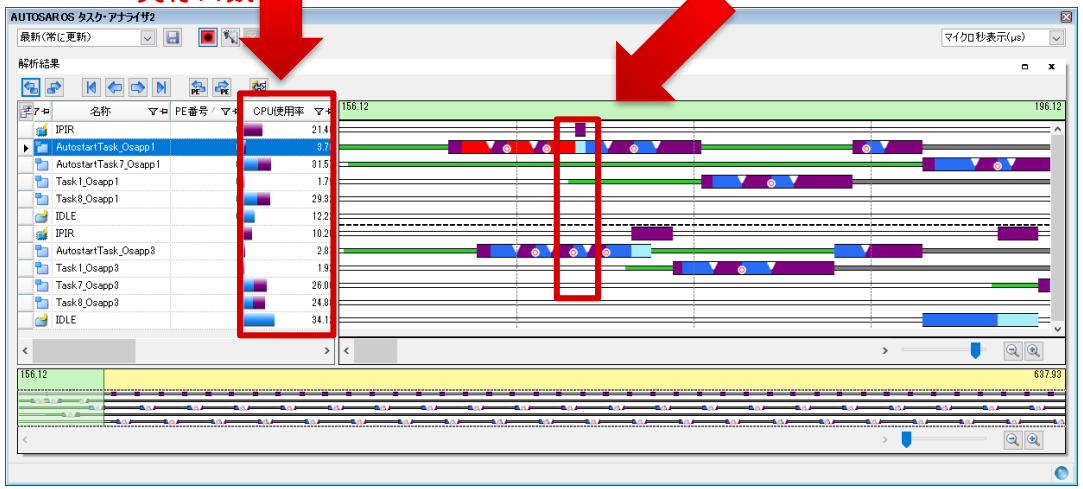
マルチコア支援ツール (1)

■ タスク・レベル設計支援:

TA Tool Suite (Timing Architect)
T1 (GLIWA)
SymTA/S (Symta vision)
Task Analyzer (Renesas Electronics)

性能プロファイル情報
- CPU使用率
- 実行時間(累計、平均、最大、最小)
- 実行回数

コア間相互作用

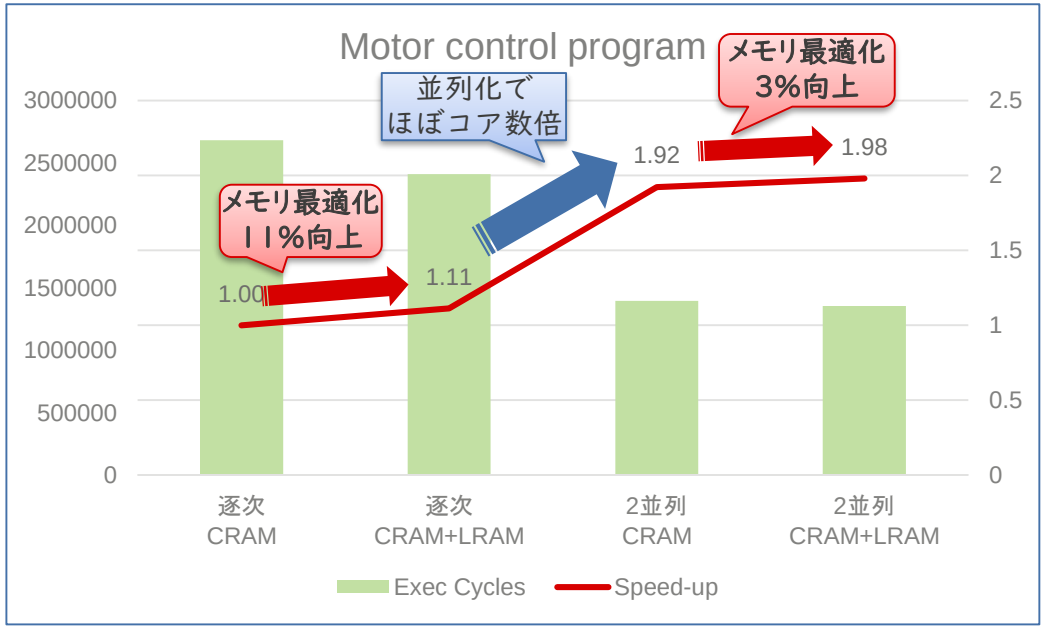


タスクアナライザ (ルネサスエレクトロニクス)

■ 並列化コンパイラ:

OSCARTech®Compiler (Oscar Technology)
SLX Tool Suite (Silexica)

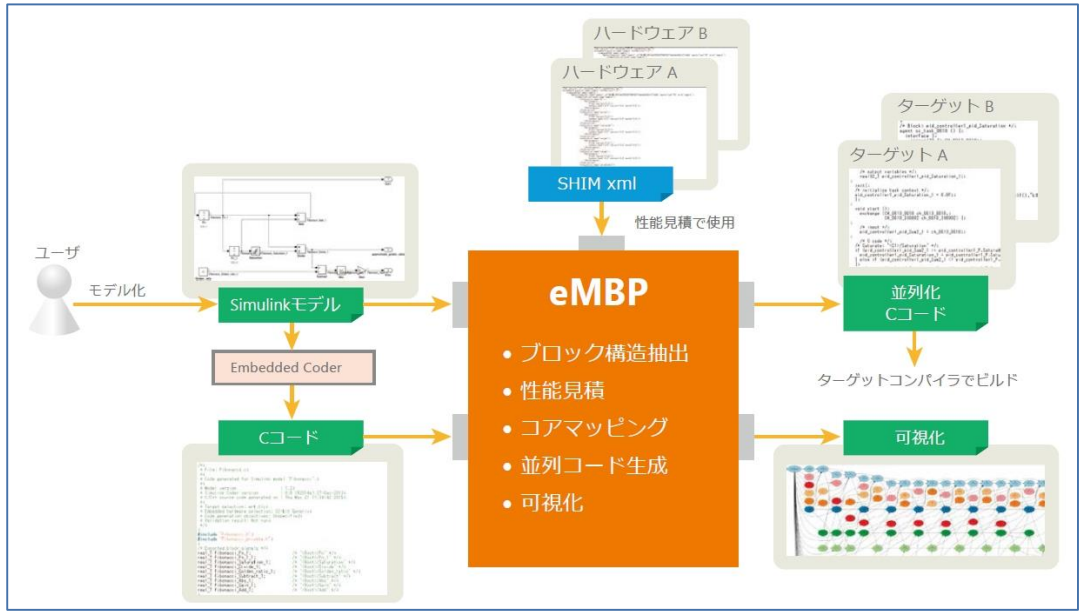
並列化が難しい制御系ソフトで高い並列性能を発揮

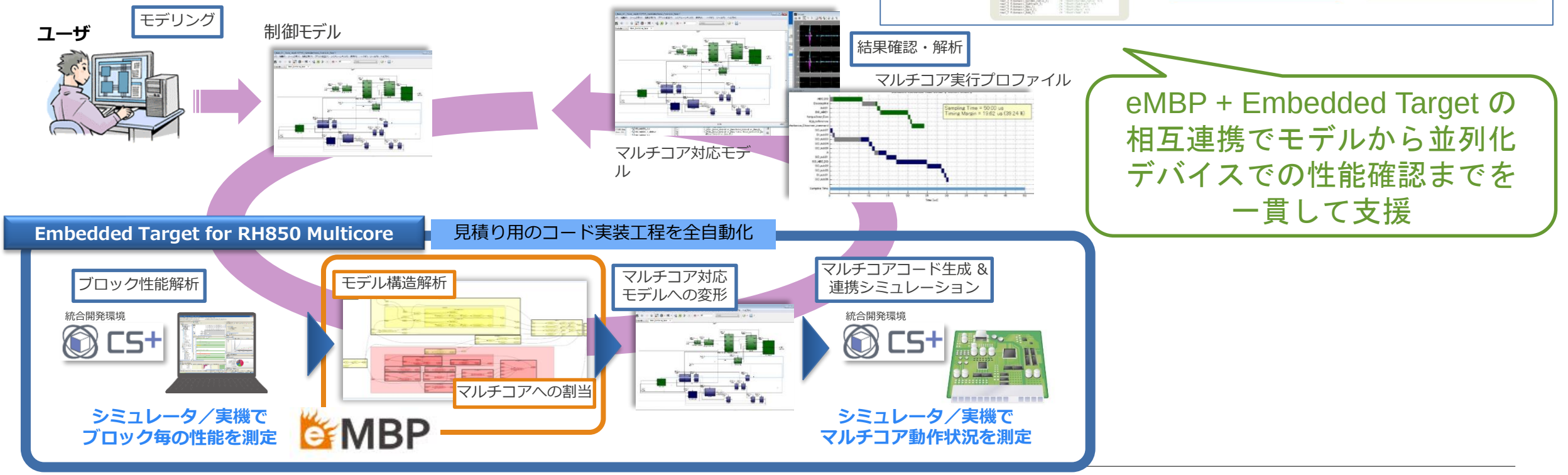


OSCARTech®コンパイラ性能評価
LRAM:ローカルメモリ(高速), CRAM:共有メモリ(低速)

マルチコア支援ツール (2)

■ モデルベース並列化:
eMBP (eSOL)
Embedded Target for RH850 Multicore(Renesas)





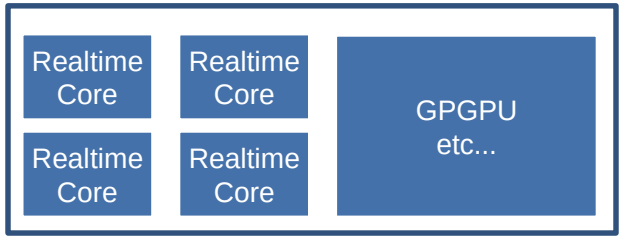
eMBP + Embedded Target の
相互連携でモデルから並列化
デバイスでの性能確認までを
一貫して支援

制御系マルチコアの展望

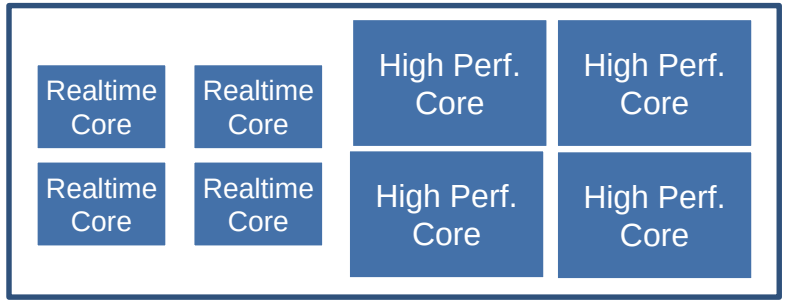
今後の制御マルチコアとEMCの活動

■ 制御システムの高度化、複雑化が更に進行
- ➢Deep Learning適用 (非線形システム，MIMO)
- ➢並列探索／最適化問題
- ➢機能統合、集中化 (セントラル・コンピューティング)

➢ 制御コア・クラスタ+アクセラレータによるヘテロ構成
- ✓ GPGPU、DFP、DRP、CNN専用アクセラレータ etc...



➢ 制御コア・クラスタ+高性能コア・クラスタによるヘテロ構成



■ ソフトウェア設計は、より難易度が高くなっていく
- ➢ハードウェア仕様を正確に把握する必要あり
- ➢マルチコア設計手法の一般化、ツールによる支援

組込みマルチコアコンソーシアム (EMC)

SHIM委員会	モデルベース並列化委員会	マルチコア適用委員会
✓ ハードウェア抽象記述SHIMのIEEE標準化 ✓ 支援ツール間のIF整備、流通改善	✓ マルチコア設計支援ツールの開発 ✓ SHIM活用の支援ツールのリファレンス整備	✓ マルチコア・ソフトウェア設計のガイドライン整備 ✓ 事例／設計手法の共有