

Embedded
Multicore
Consortium

www.embeddedmulticore.org

組込みマルチコアコンソーシアムが見る 組込み産業の課題

2024-11

名古屋大学 枝廣 正人

イーソル(株) 権藤 正樹

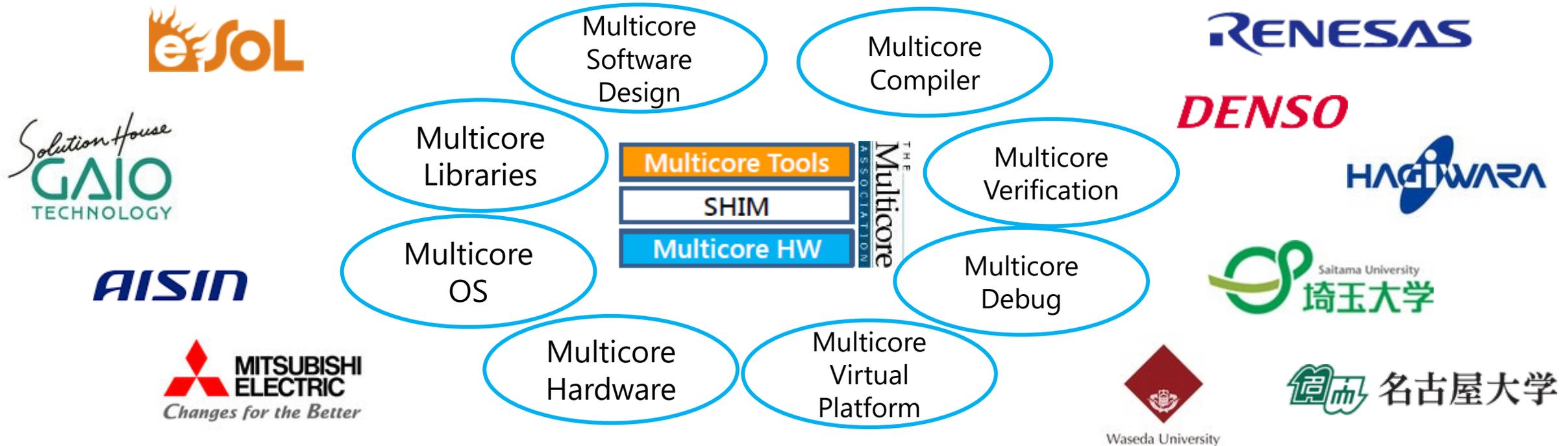
ガイオテクノロジー(株) 嶋田 卓尚

岩井 陽二

目次

1. 組込みマルチコアコンソーシアムについて
2. 組込みマルチコアコンソーシアムが見る組込み産業の課題
 - 「**Software-Defined**に向けてのソフトウェア課題」より

組込みマルチコアコンソーシアムとは



- システム、ソフトウェア、ツール、半導体の各レイヤが協力・連携し、前述の課題を解決する**エコシステムを構築するための産学合同**の場
- 組込みマルチコアに関する**技術開発加速と利用促進**
- **開発フローの確立とベンダ間ツール協調を支援**

コンソーシアム活動（3委員会を中心に）

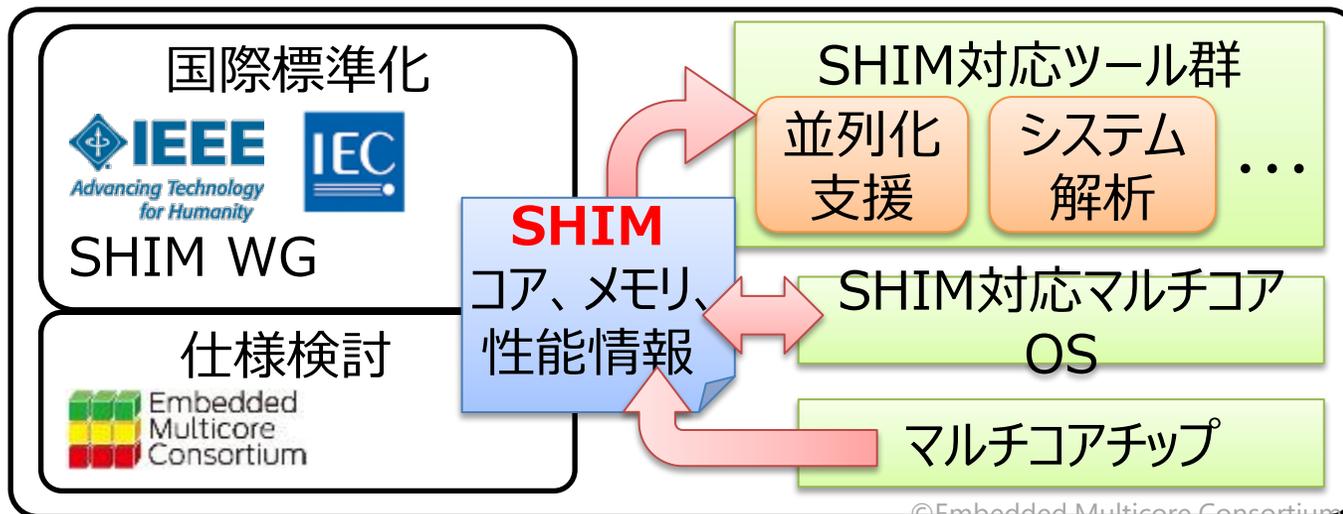
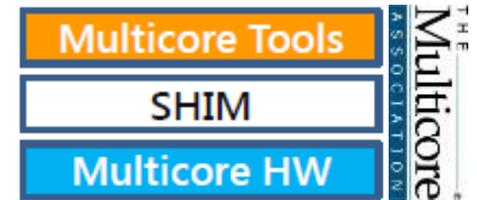
- マルチコア向け開発支援ツールのためのハードウェア抽象化記述**SHIM**標準化と導入支援（**SHIM委員会**）
 - **SHIM** (Software-Hardware Interface for Multi-Many-Core)
 - NEDO省エネPJから仕様提案、MCA標準として2015年2月V1.0、2019年1月V2.0、2020年1月**IEEE Std 2804-2019**、2023年10月**IEC 63504-2804**(IEEEとのDual Logo)
- リファレンスとして**SHIM**を利用したマルチコア向け設計支援ツール群を開発
 - MCAとしても公開するSHIM Editorと性能計測ツールに加え、設計支援ツール群を会員向けに無償公開。所定の期間経過後に一般にも公開する可能性有
 - **モデルベース並列化委員会**
- 様々な並列化手法の知見共有とガイドラインの検討
 - **マルチコア適用委員会**
- セミナー開催、技術情報提供、MCA標準化資料公開

MCA: Multicore Association
かつて存在したマルチコア
関係の国際標準化団体

SHIM

SHIM 2.0: IEEE 2804-2019
IEC 63504-2804

- SHIMの標準化に貢献 (Software-Hardware Interface for Multi-many-core)
 - 多様なマルチコアチップを抽象化したXML記述
 - コア種類・数、メモリ配置、アドレスマップ、通信、コア→メモリ性能情報等が、数百ページの説明書を読まずとも、機械的に読める
 - 性能情報の例：コアAからメモリ番地Xにアクセスしたときの(best, typ, worst)レイテンシ
 - ツール群、OS等がSHIM対応することにより、多様なマルチコアチップを共通的に扱えるようにすることが目的



```
<MasterSlaveBinding slaveComponentRef="LRAM_BI
  <Accessor masterComponentRef="CPU_BDCOP2">
    <PerformanceSet>
      <Performance>
        <accessTypeRef>Instruction_Fetch</acce
        <Pitch best="1.0" typical="1.0" worst:
        <Latency best="1.0" typical="1.0" wor
      </Performance>
    </PerformanceSet>
    <accessTypeRef>Load_Aligned_Byte</acce
    <Pitch best="1.0" typical="1.0" worst:
    <Latency best="1.0" typical="1.0" wor
  </Performance>
```

コア→メモリ性能情報
SHIM記述例

SHIM関連ツールの一般公開

Search or jump to... Pull requests Issues Marketplace Explore

openshim / shim2 Public Watch 2 Star 1

<> Code Issues 1 Pull requests Actions Projects Wiki Security Insights

master 1 branch 0 tags Go to file Add file Code

masakigondo Update README.md 152951e 12 days ago 14 commits

docs	first	10 months ago
plugins	plugin check	9 months ago
samples	first	10 months ago
schema		16 months ago
shim-measure		12 days ago
sources		10 months ago
tools		3 months ago
LICENSE.txt	add new file	2 years ago
README.md	Update README.md	12 days ago

shim-measure: 計測ツール
tools: SHIM Editor

About
No description, website, or t
Readme
MIT License

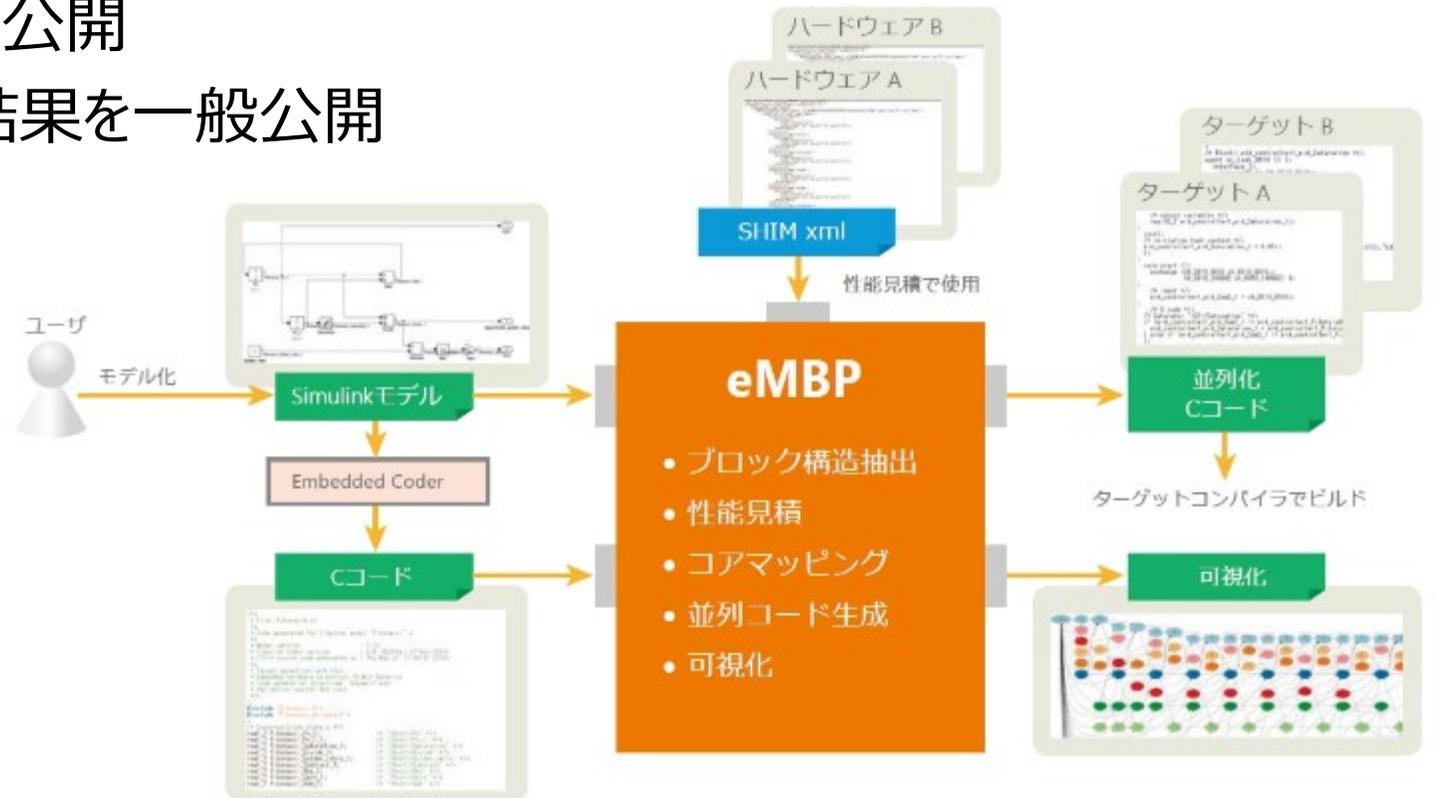
Releases
No releases published
Create a new release

Packages
No packages published
Publish your first package

©Embedded Multicore Consortium 2024

モデルベース並列化 (MBP)

- コード生成可能なSimulinkモデルから自動的に並列化コードを生成
 - 名古屋大学でプロトタイプ、eSOL社から実用化
 - プロトタイプを会員向けに公開
 - 簡単なサンプルモデルと結果を一般公開



マルチコア適用委員会

「マルチコア技術普及」に向けた活動

マルチコア適用ガイド

- マルチコア知見者が作成した4つのマテリアル
 - マルチコア適用ガイド
 - 制御系マルチコア・ハードウェアの特徴とユースケース
 - 自動車 機能安全へのマルチコア適用
 - 並列処理ソフトウェアの課題と対策技術
- 最新版では、4つパワーポイントデータを1つに集約し、EMCホームページ上で公開（HTML化/PDF化）

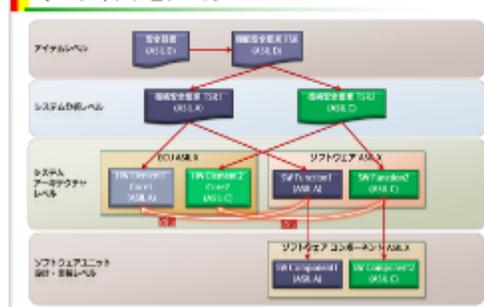
制御システムにおけるマルチコアの代表的ユースケース

- 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



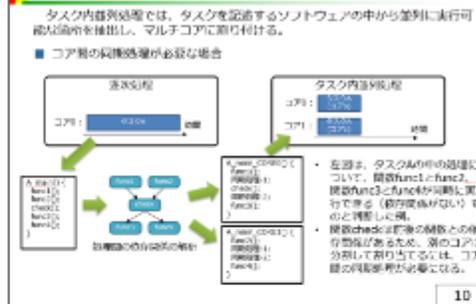
制御系マルチコア・ハードウェアの特徴とユースケースから抜粋

マルチコア適用の機能安全アーキテクチャデコンポジションの例



自動車 機能安全へのマルチコア適用から抜粋

1-2 タスク内で並列化する際の課題



マルチコア適用ガイドから抜粋

並列プログラムの課題

- 挙動理解の困難性
 - 非決定性
 - 並列プログラムの挙動には「非決定性」を含む場合がある。非決定性を持つプログラムでは、状態遷移の分岐が排他的でない。
 - テストプログラムや不具合事後の再現性が悪く、デバッグしにくい。
 - 複雑性
 - 並列に動作する複数のスレッドが別々の状態遷移を持つため、システムの状態はその組み合わせになり、複雑で、膨大な状態遷移空間をもつ。
 - システム挙動の網羅的な検証が困難
- 計算資源の有効利用
 - ソフトウェアの並列動作可能な計算資源へのマッピング

並列処理ソフトウェアの課題と対策技術から抜粋

ブログ・動画コンテンツ

ブログ：はじめての並列化

ブログ：並列処理の不具合と対策

ブログ：マルチコアの基本

はじめての並列化 · 2021/01/05

(1) OpenMPによるデータ並列化 [はじめての並列化]

マルチコアプログラミング未経験の人(私)が逐次処理プログラムをマルチコア化する挑戦記録を連載することになりました。

マルチコアソフト開発実証編
～初心者もマルチコアソフト開発～

マルチコア初心者からマルチコア開発者になるまで、マルチコア開発の基礎から応用まで、日々挑戦を繰り返して、日々学ぶ初心者の方にお役にたて、マルチコアの楽しさを伝えることができればと思います。

著者 藤原 氏

1980年代後半から現在まで、RISC-V/LINEX/自動運転/AI/コードに関する知識をベースに、現在、初めてのマルチコア開発

測定結果(まとめ)

処理内容	コア数	実行時間 (秒)	性能向上率 (%)
逐次処理	1	10.0	100%
2コア並列化	2	5.0	50%
4コア並列化	4	2.5	25%
8コア並列化	8	1.25	12.5%

● OpenMPは並列化が容易(ソースを書き換える必要がほとんどない)
● ソフトウェアの書き換えが容易(ソースを書き換える必要がほとんどない)
● ソフトウェアの書き換えが容易(ソースを書き換える必要がほとんどない)
● ソフトウェアの書き換えが容易(ソースを書き換える必要がほとんどない)

並列処理の不具合と対策 · 2021/10/13

(4) ライブロック(livelock) [並列処理の不具合と対策]

ライブロックとは“動いているのだが進まない”という状況を指します。
たとえば、道を歩いていて向こうから来た人を選びようとするとき、相手が自分と同じ方向に動いてしまう。これを繰り返すような状況もライブロックとなります。

並列処理の不具合と対策 · 2022/03/16

(8) 並列処理問題をソースからを見つけるためのヒント [並列処理の不具合と対策]

これまでのブログであげた並列処理問題の解決方法を考えてみます

■並列処理動作の問題は超レアケースで再現性なし

ライブロック・デッドロック、リエントラントの問題など、並列処理で発生する多くのものはある条件下で発生し、その条件は絶妙のタイミングで起

問題の種類	発生条件	再現性	対策
ライブロック	互に相手のリソースを保持している状態	再現性なし	タイムアウトを設定する
デッドロック	互に相手のリソースを保持している状態	再現性なし	タイムアウトを設定する
リエントラント	同一リソースを同時に複数回アクセスする	再現性あり	互斥セマフォを使用する

マルチコアの基本 · 2024/06/20

第7回 オーバーヘッドと並列性能

前回、コア数を増やしてもコア数ほどには性能が上がらないという話をしました。その理由の一つとしてオーバーヘッドの話をしました。
例えば、一つのプロセッサで計算量が100の処理を2つのプロセッサで実行させようとしたとき、相手のプロセッサに処理依頼をする、結果を受け取るという処理がオーバーヘッドになります。その処理の計算量がそれぞれ20とすると、全体で処理量が140になってしまい、処理が理想的に2分割できたとしても各プロセッサの計算量は70になり、処理が2倍になることはありません(下図左)。もとの処理量が10倍の1000であれば、オーバーヘッドの計算量は変わらないため、処理はよくなります(下図右)。



動画コンテンツ (YouTubeチャンネル https://www.youtube.com/@EMC_multicore)

●第一弾「マルチコアって必要なの？」

EMC 会長 枝廣先生
EMC 副会長 梅藤さん

前回は話題になりましたが、これについてお二人のコメントを聞きたいと思っております。

EMC MULTICORE CONSORTIUM

マルチコアで資産性・再利用性を確保できる"王道"とは？

マルチコア上で動作するソフトウェアのデッドロック動作をわかりやすく解説



その他 : MCA (Multicore Association) 仕様書

- MCAの解散に伴い、一部仕様書がEMCに移譲
 - MCAPI
 - MRAPI
 - MTAPI
 - MPP
 - SHIM
- EMC WWWサイト “The Multicore Association Specifications”メニューから
- MPP、SHIM1.0に対してはコンソーシアムで和訳作成

MCA仕様書

[To Engl

2020年にMCA (Multicore Association)が解散した後、一部の仕様書類は組込みマルチコアコミュニティページではそれらの仕様書類をWikipediaにおける解説と共に公開しています。

Wikipedia: https://en.wikipedia.org/wiki/Multicore_Association (referred on June 9, 2021).

● MCAPI V2.015

2008年、Multicore Communications API (MCAPI) ワーキンググループは、MCAPI と呼ばれる MCAPIはメッセージパッシングAPIであり、組込みシステム内の比較的近距離に分散する要素 (チップ) 間に必要な通信と同期の基本APIを提供します。MCAPIは複数次元の不均一性 (heterogeneity) は、プロセッサコア、インターコネクティブファブリック、メモリ、オペレーティングシステム、プログラミング言語などがあります。

2011年、MCAPIワーキンググループはMCAPI2.0をリリースしました。拡張バージョンでは、追加されています。MCAPI2.0は、「ドメイン」の導入により、ノードネットワークに対してドメインは、例えば特定チップ上のすべてのコアを表現する、あるいはトポロジをパブリックエクスポートするためのさまざまな用途に使用できます。また、MCAPI 2.0は新しく3種類の初期化パラメータ (

組込みマルチコアコンソーシアムが見る 組込み産業の課題

「Software-Definedに向けてのソフトウェア課題」より抜粋
(組込みマルチコアコンソーシアム イベントページ
<https://www.embeddedmulticore.org/events/>
から入手可能)



目次

1. CASEとSoftware Defined Vehicle

- CASEとは
- CASEはグローバルな政策
- CASEによるビジネス機会
- Software Defined Vehicle (ソフトウェア定義型自動車) とは
- Tesla社の例

2. ハードウェア/ソフトウェアの共通プラットフォーム化

- E/E (電気/電子) アーキテクチャの変化
- Vehicle OS
- ソフトウェアプラットフォームの変化

3. 現状のソフトウェア開発の課題

- ビジネスモデル変更の必要性
- 開発方針変更の必要性
- 開発プロセス/開発手法変更の必要性

4. おわりに

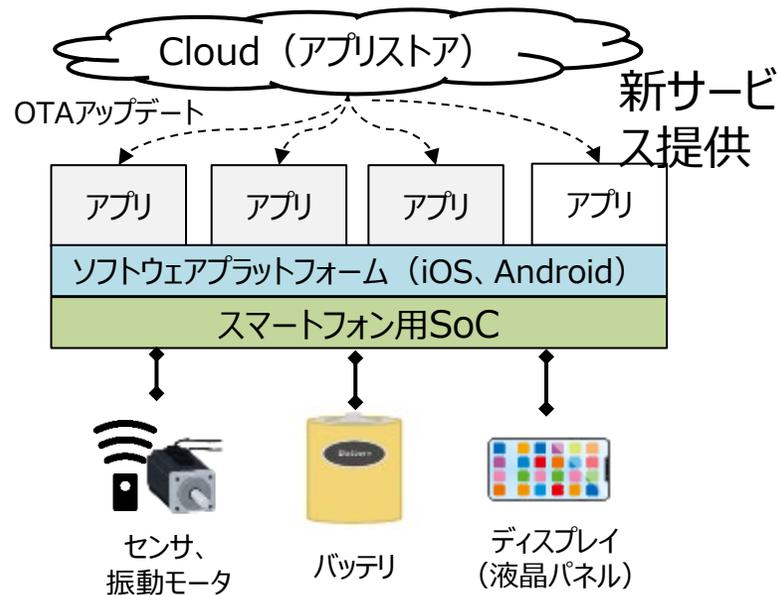
- 組込みマルチコアコンソーシアムとのかかわり
- まとめ

1. (CASEと) Software Defined Vehicle (ビジネス観点のトレンド)

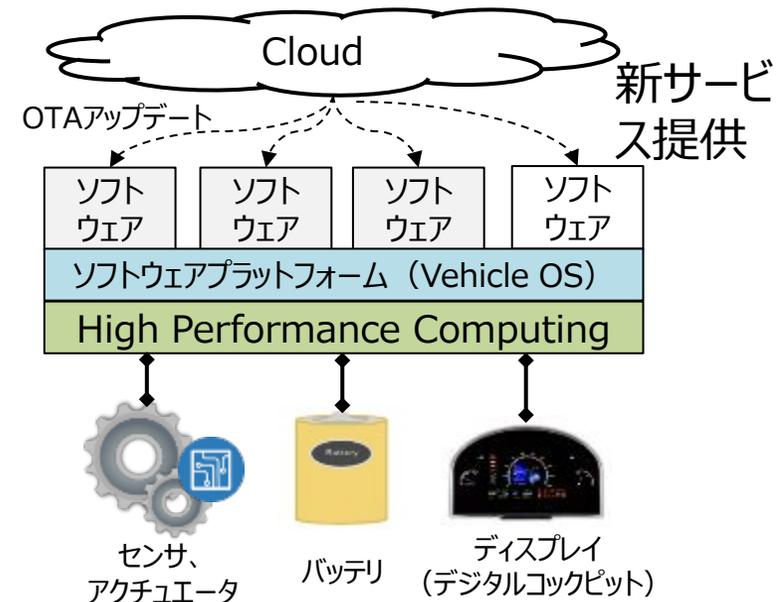
Software Defined Vehicle (ソフトウェア定義型自動車)

とは

- お手本は「スマートフォン」
- 機能実装の中心がハードウェアからソフトウェアへシフト
- 無線通信（OTA : over-the-air）により、継続的なアップデートや最新サービスを提供
- UI（user interface）などの機能やコンテンツのカスタマイズ、パーソナライズが可能に



スマートフォンのシステム構成



Software Defined Vehicleのシステム構成

Tesla社の例

- いち早く**Software Defined Vehicle**を市場に投入
- 1カ月単位でソフトウェアを更新 (バグ修正は24時間以内)

無線通信でソフトが更新される
● スマホやインターネットと接続するテスラ「モデル3」

新たな機能や
コンテンツの追加

システム変更
バグの修正

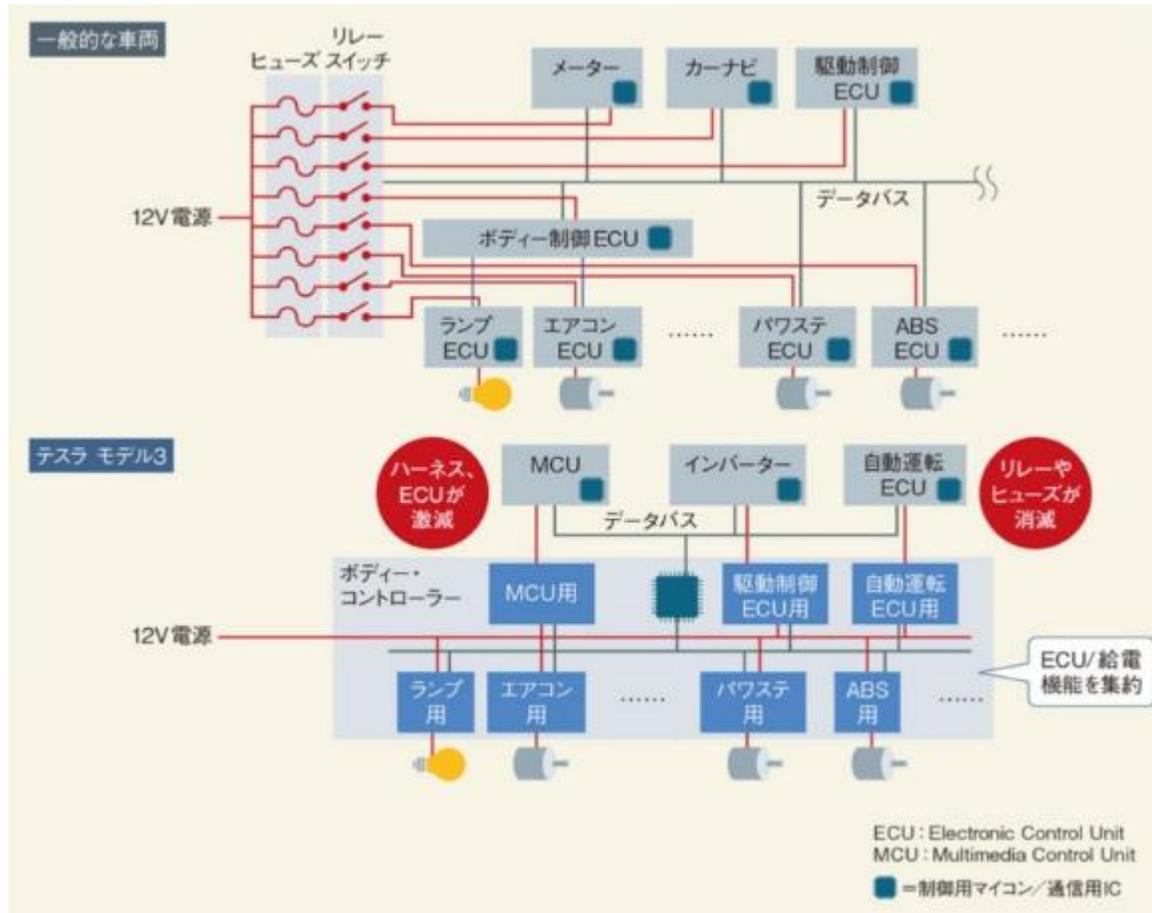
車両性能の
向上



出典：橋本 真美；「クルマ電腦戦(1) 風雲児テスラ、ソフトを主役に『進化する車』で自動車業界揺さぶる」、日経ビジネス、2022年3月7日。

Tesla社の例

- 分散したECUを集中・統合



- 一般的な自動車の車両では、機能ごとにECUが用意される。
- 各ECUは、相互通信のために、CANやLINのインターフェース、ECUに接続されるアクチュエータへの電力供給を制御するためのマイコンや電源回路が搭載される。
- また、12V系の電源も別の系統で提供される。

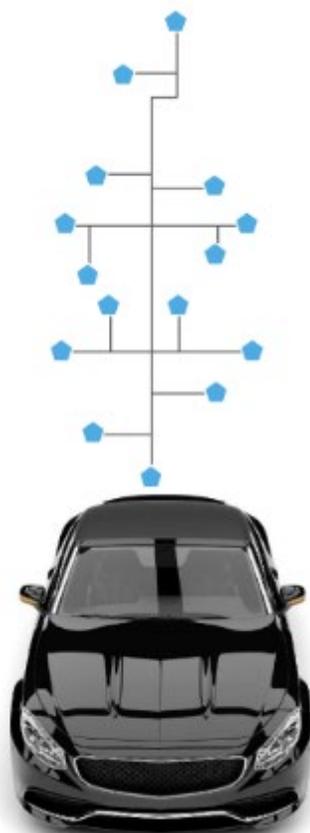
- テスラ モデル3では、分散していたECUを電源供給も含めてボディコントローラに統合した。

出典：野々村 洸；「テスラが常識外の“集中ECU”。部品が激減、ヒューズは消滅」、日経クロステック、2020年4月27日。

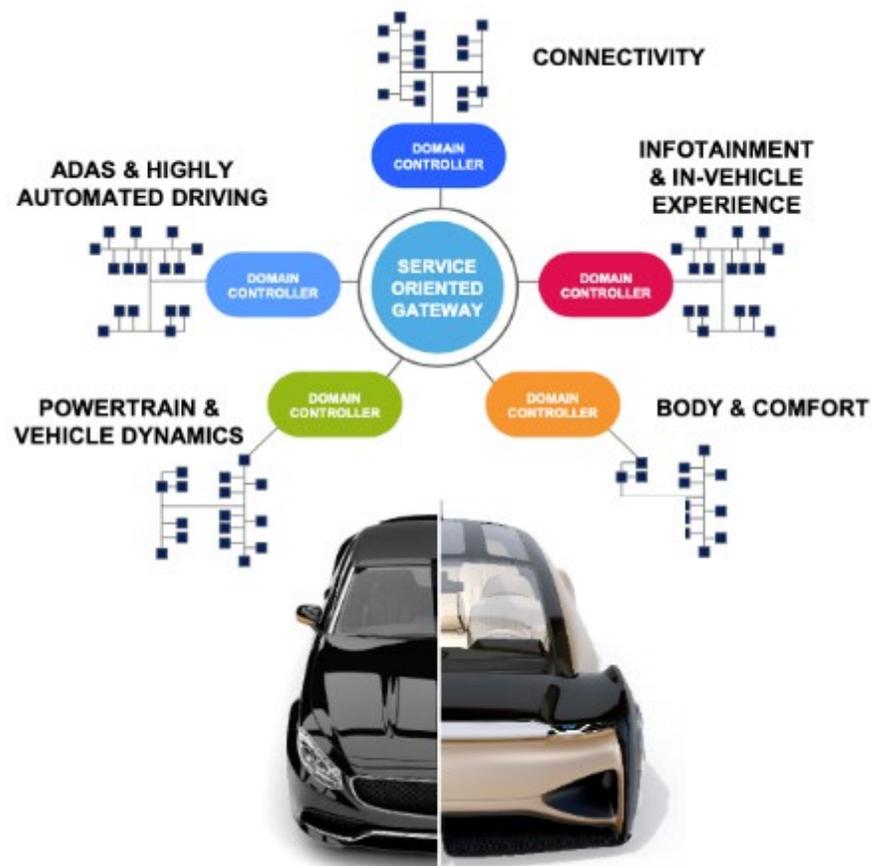
2. ハードウェア/ソフトウェアの共通プラットフォーム化 (技術観点のトレンド)

E/E (電気/電子) アーキテクチャの変化

- フラットなアーキテクチャ (分散型) からドメインアーキテクチャ (機能別の集中制御) へ



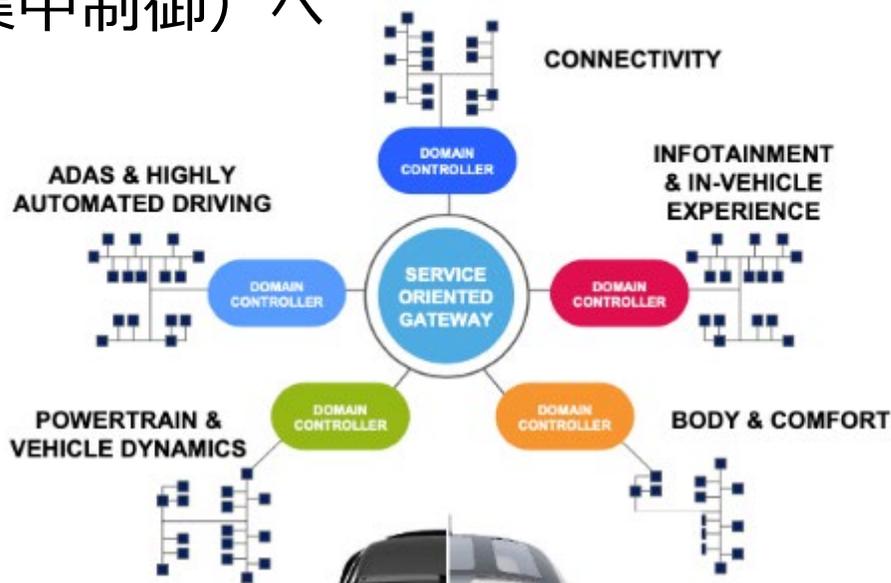
(a) 現在 | フラットなアーキテクチャ



(b) 論理的再構築 | ドメインアーキテクチャ

E/E (電気/電子) アーキテクチャの変化

- ドメインアーキテクチャ (機能別の集中制御) からゾーンアーキテクチャ (物理配置を考慮した一元的な集中制御) へ



(b) 論理的再構築 | ドメインアーキテクチャ (c) 物理的再構築 | ゾーンアーキテクチャ

Vehicle OS

- 自動車メーカーが、みずから「Vehicle OS」を開発

Vehicle OSの例

自動車メーカー	開発したOS/プラットフォーム
独国Volkswagen社	vw.OS
独国Mercedes-Benz Group AG社	MB OS
北欧Aktiebolaget Volvo社	VolvoCars.OS
米国Tesla社	Tesla OS
トヨタ自動車	Arene (アリーン)

- Vehicle OSの役割

- ハードウェアとソフトウェアの分離

- OTAによるアップデート（機能の追加・修正）の実現
- ハードウェアを意識しないアプリケーション開発の実現

- 開発環境（ツール類）の提供

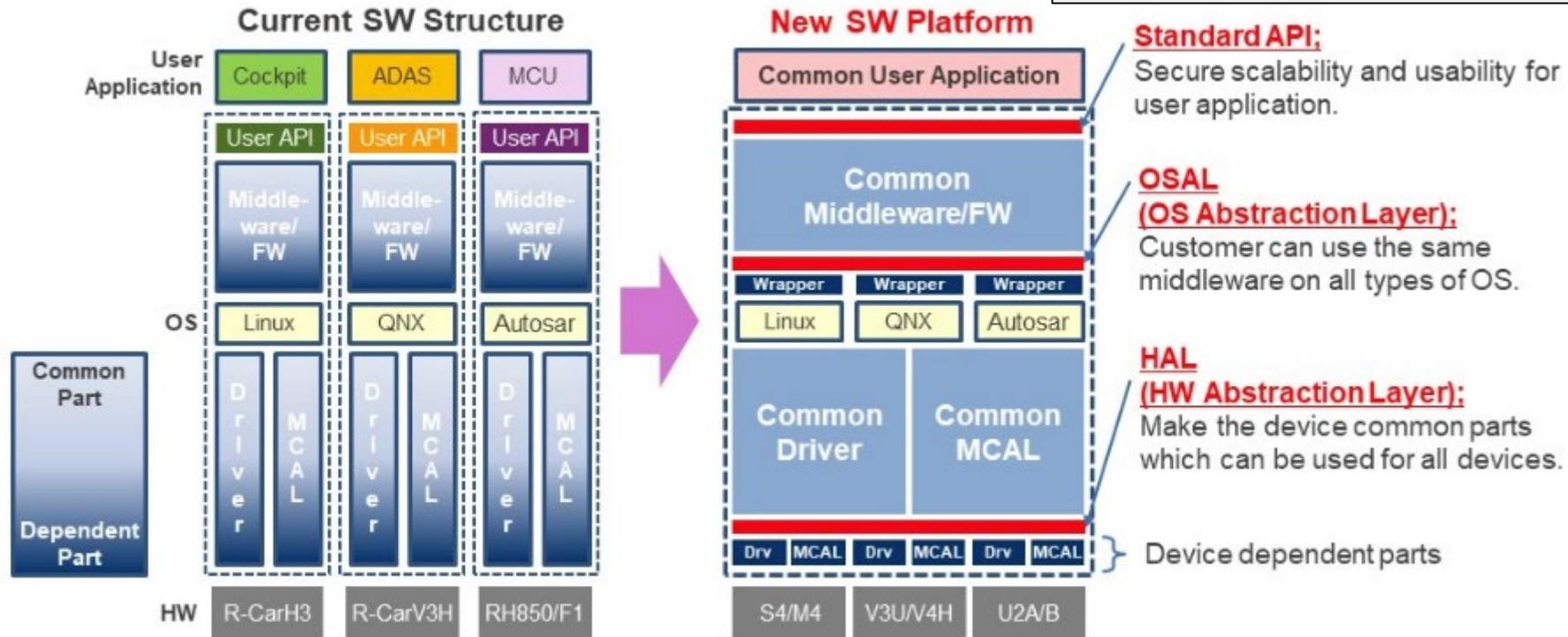
- Vehicle OS上で動作するアプリケーションの開発を効率化

[AndroidスマートフォンやiPhoneと同一スキームをVehicle OSで実現](#)

ソフトウェアプラットフォームの変化

- 複雑化・多層化が進むソフトウェアプラットフォーム
- 大規模ソフトウェア/アプリケーションの実行を支える
マルチコアベースのHigh Performance Computing

- 標準API、OSAL、およびHALの三つの抽象層を導入。
- セグメント、OS、およびデバイスに依存しないソフトウェアの再利用性を実現。



出典：Tsuyoshi Tsumuraya；「E/Eアーキテクチャの進化を支えるR-Car/RH850向けソフトウェアプラットフォーム」、ルネサス エレクトロニクス ブログ。

3. 現状のソフトウェア開発の課題 (変化の必要性)

ビジネスモデル変更の必要性

- 従来は、「ハードウェア（ECU/メカ等）ありき」のビジネスモデルだった
 - 販売価格は、主にハードウェア部品や製造コストをもとに算定
 - 付随するソフトウェアの価格/価値を、きちんと設定できていない
- **Software Defined Vehicle**の時代は、ソフトウェアが自動車の価値の源泉に
 - 今後は、「継続的なソフトウェアの供給」を前提としたビジネスモデルへ
 - 時間の経過とともにユーザ体験やユーザ価値が変化
 - 「ソフトウェアの適正な価格/価値」に基づいたビジネス展開が必要
 - ただし、「ソフトウェアの適正な価格/価値」のスタンダードモデルが、現状ない



個別最適化から全体最適化へ

- 従来のソフトウェアの開発方針は、個別最適化がすべてだった
 - ドメイン別の開発体制
 - 製造コストや部品コストを優先したハードウェア設計
 - ソフトウェアは、その上でいかに最適動作を実現するか、が重要だった
- **Software Defined Vehicle**の時代は、プラットフォームの共通化・標準化が極限まで進む可能性も...
 - ソフトウェアは共通プラットフォーム上で動作。限りなくハードウェア非依存に
 - 極端な「個別最適化」よりも、バランスのよい「全体最適化」が求められる
 - ドメインをまたいだソフトウェアの開発や最適化も必要に



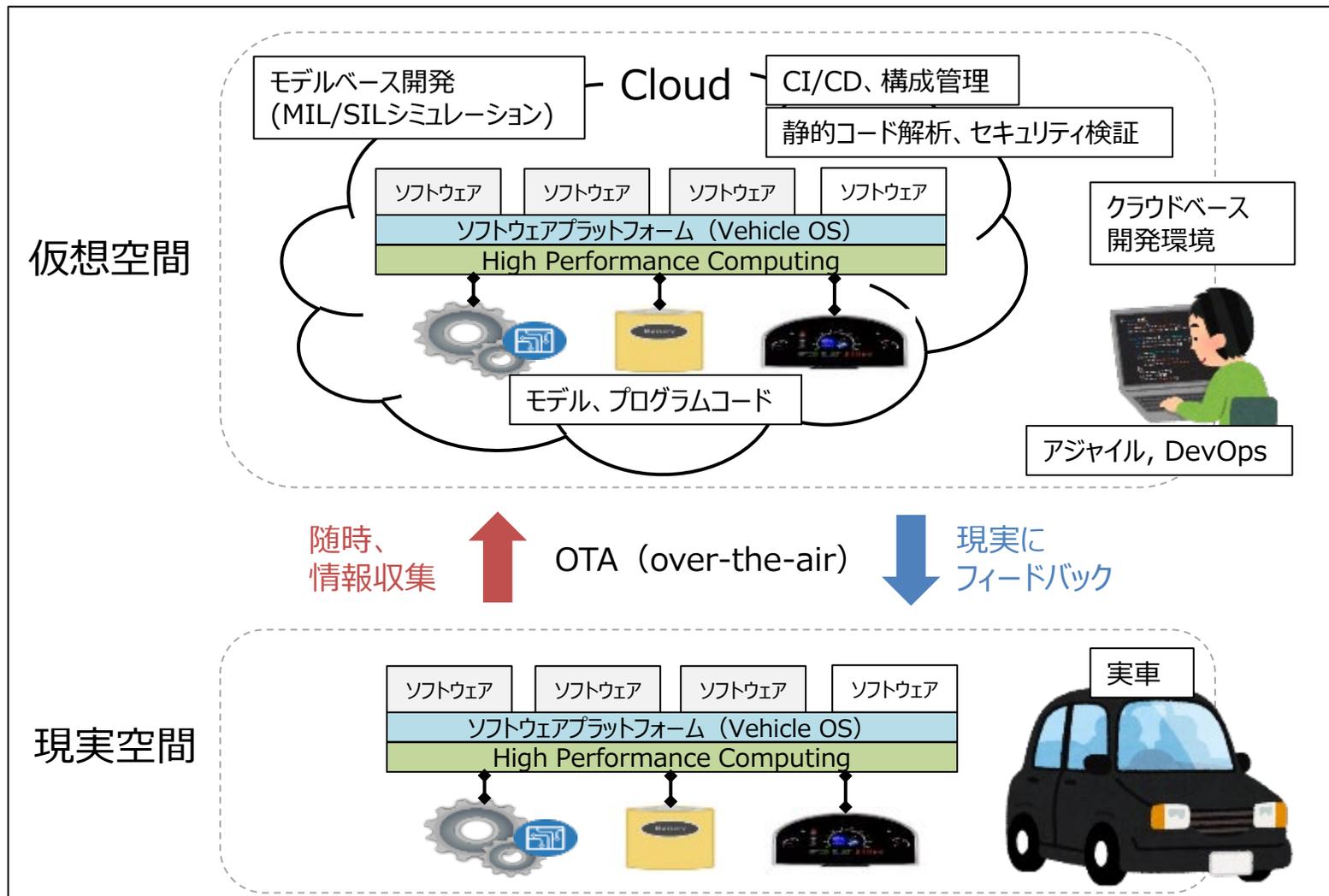
開発プロセス/手法の見直しの必要性

- 従来は納期までに「完璧な」ソフトウェアを完成させることが前提だった
 - バグなし。実現する機能は固定的
 - 既存のソフトウェア資産を徹底的に再利用
 - 派生開発のプロセスを確立して、高品質と短期開発を両立
- 今後はOTAによるソフトウェアアップデートを前提とした開発プロセス/開発手法へ
 - 短いサイクルで新機能や修正パッチを継続的にリリース
 - クラウドサービス設計
 - セキュリティ・バイ・デザイン
 - アジャイル/DevOps、CI/CD（継続的インテグレーション/継続的デリバリ）
 - 開発環境はデジタルツイン（モデルベース開発、クラウドベース開発環境 etc.）



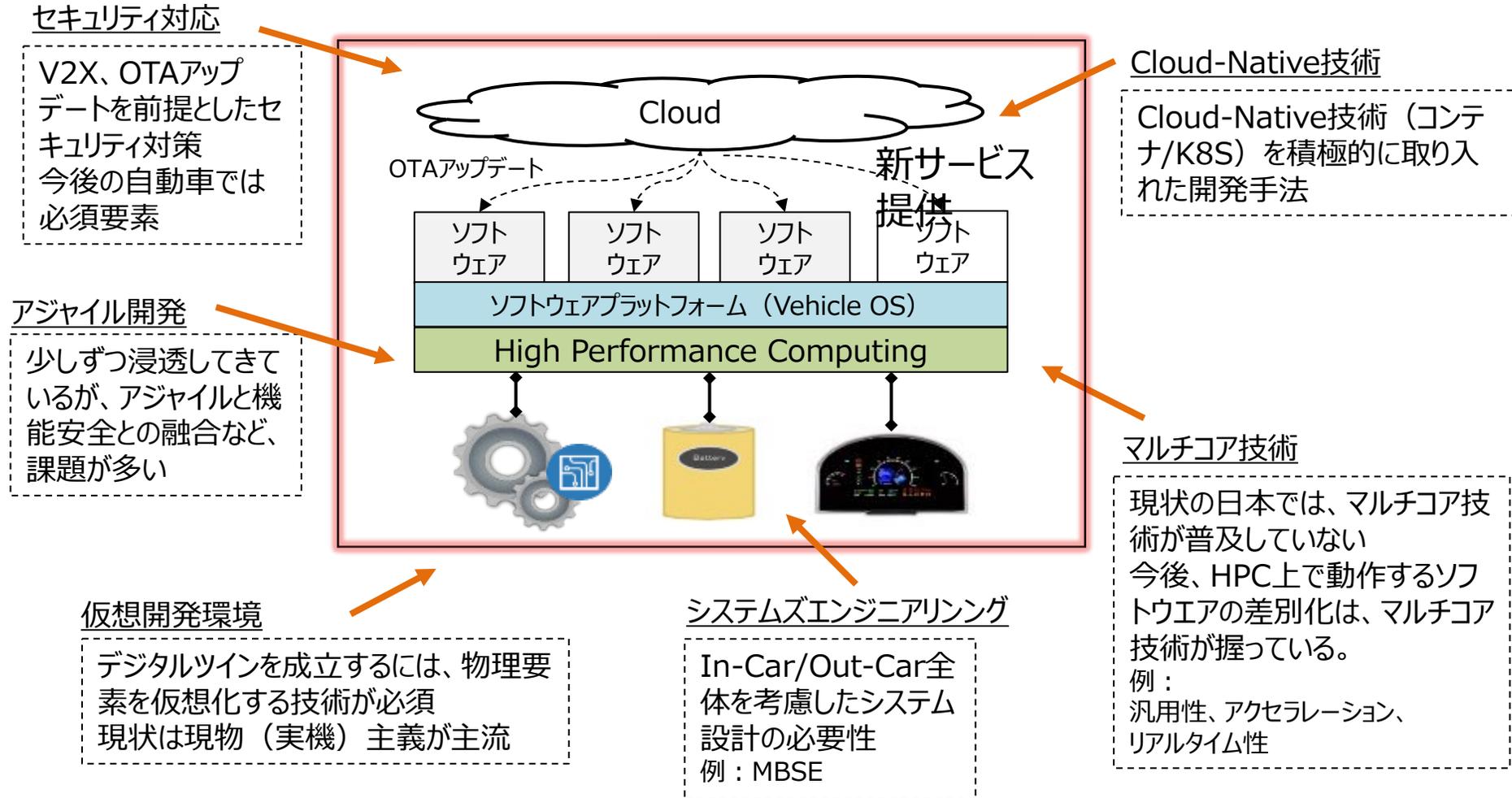
今後求められる仮想空間における開発プラットフォーム（デジタルツイン）

- デジタルツインとは、現実の世界から収集した様々なデータを、まるで双子であるかのように、コンピュータ上で再現する技術（下図は、車載ソフトウェアを対象としたデジタルツインのイメージ）



新しい技術導入の必要性

- 従来とは異なる時代の到来とともに、新しい技術導入が必須



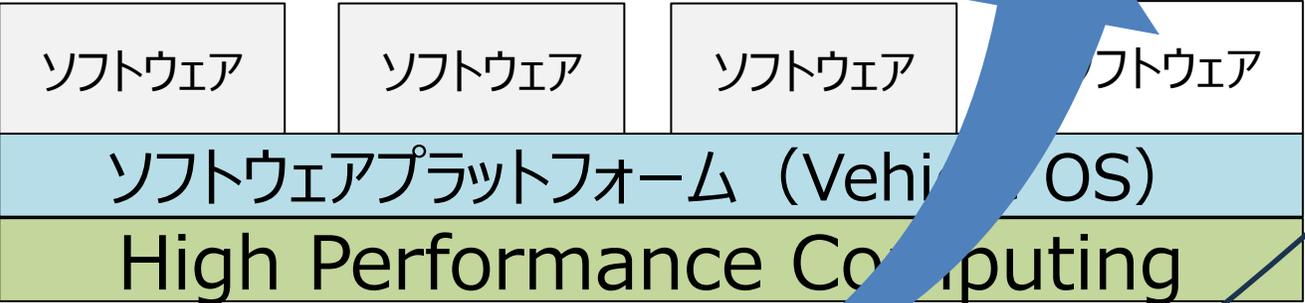
EMCでの取り組み

モデルベース並列化 (MBP)

システム設計開発初期段階からのソフトウェア性能見積 (SHIM)

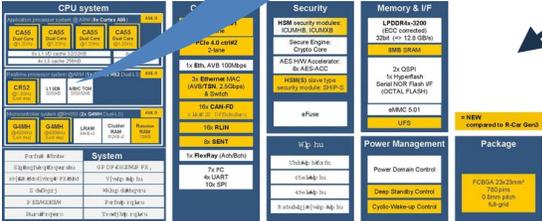
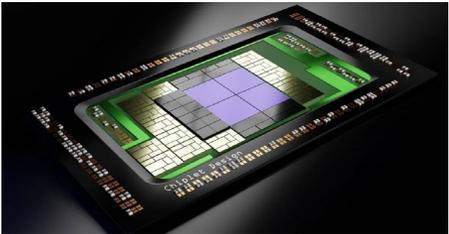
ハードウェア・ソフトウェア
コデザイン

モデルベース開発 (MIL/SILシミュレーション)



ハードウェアを抽象化したモデルでマルチコア上で動作するソフトウェアを協調設計。**Chiplet時代に重要**

マルチコア利用 / 開発技術普及 (マルチコア適用)



4. おわりに

まとめ

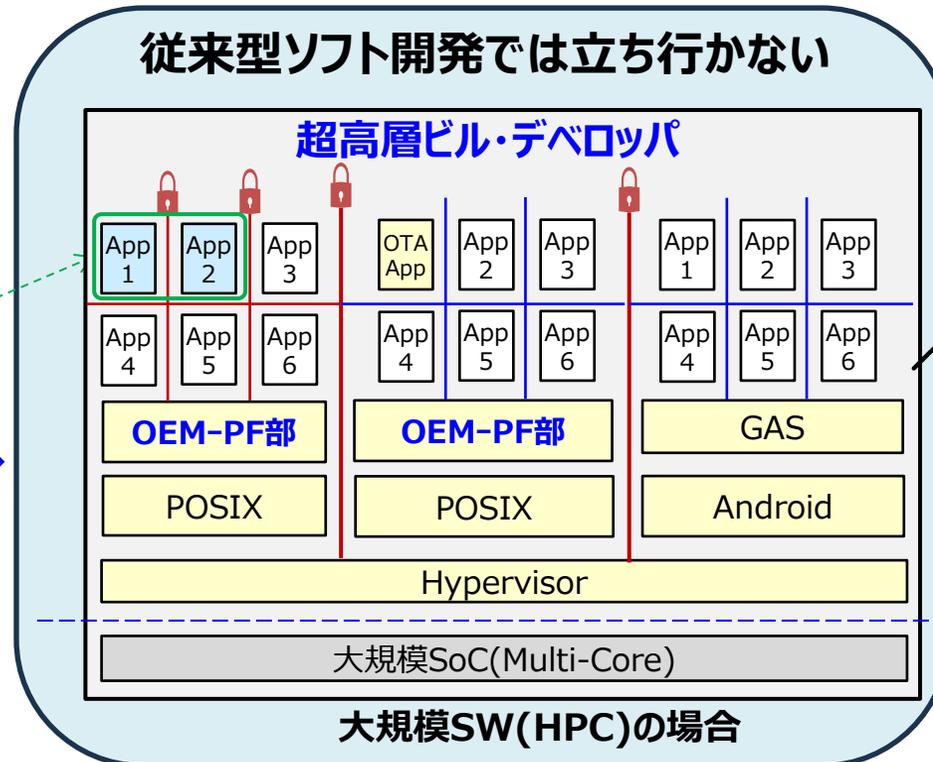
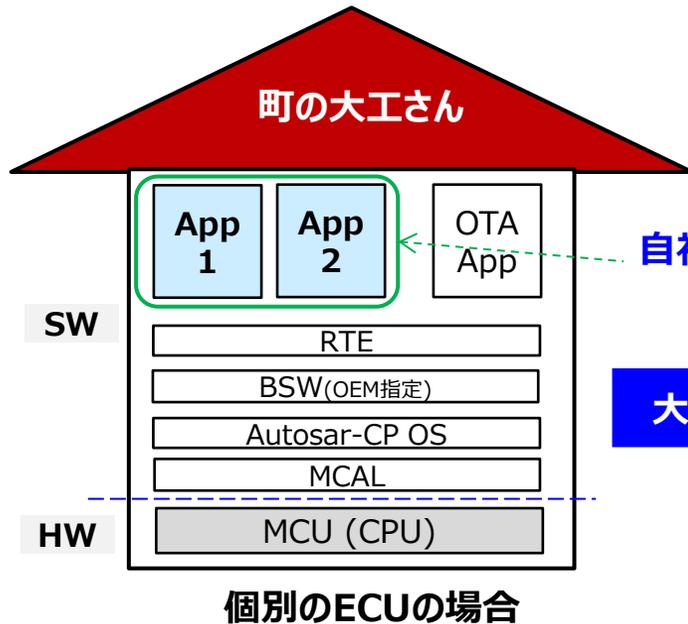
- 自動車産業を取り巻く環境がダイナミックに変化している
- 従来の自動車の概念を覆す「**Software Defined Vehicle**」が登場
 - スマートフォンをお手本としたシステム構成
 - 継続的な機能のアップデート、最新サービスの提供、パーソナライズが可能
 - Tesla社がいち早く製品を出荷
- これに伴って、自動車のE/Eアーキテクチャやソフトウェアプラットフォームが変化
 - ドメインアーキテクチャ、ゾーンアーキテクチャへ
 - 自動車メーカーが、みずからVehicle OS開発に取り組む
 - 大規模なソフトウェアはマルチコアベースのHigh Performance Computingで実行
- ビジネス環境や技術の変化を認識し、あらためてビジネスモデルやソフトウェアの開発方針、開発プロセス/開発環境を見直す必要があるのではないのでしょうか。

今後のEMC

2025年6月24日（火）
組込みマルチコアサミット
（TOPPERSカンファレンスと
同時開催（予定））

- 「Software-Defined」への課題解決に向けた仲間作り
 - 従来のソフト開発では、立ち行かない変革期
 - 日本の組込み産業を支える上で、多種の課題を解決する仲間作りを模索
- EMCが考える課題に共感頂ける方は、ぜひ、一緒に切り開いていきましょう!!

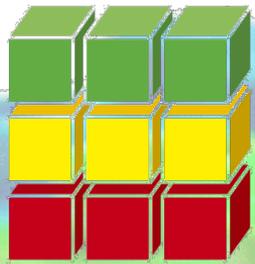
多種の技術要素を
実現するための仲間作り



- 必要となる新規領域
- ・ システムズエンジニアリング
 - ・ セキュリティ対応
 - ・ アジャイル開発
 - ・ OTA
 - ・ 仮想開発環境 (デジタルツイン)
 - ・ Cloud-Native
 - ・ マルチコア
 - ・ Hypervisor
 - ・ ソフト中心のビジネスモデル
 - ・ 新開発プロセスと品質方針
 - ・ 開発効率化/自動化

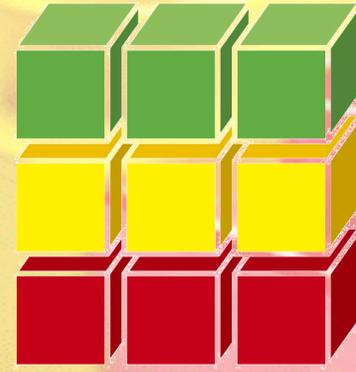
メンバーシップ

- 会員（2024年11月現在14団体）
 - アイシン、ルネサス エレクトロニクス、デンソー（予定）、eSOL、ガイオテクノロジー、萩原エレクトロニクス、三菱電機、大阪大学、埼玉大学、名古屋大学、早稲田大学アドバンスドマルチコアプロセッサ研究所、他
 - 相互協力：JASA、MCA(Multicore Association)
- メンバーシップ構成
 - 正会員（入会金なし、年会費20万） 準会員、特別会員
 - 詳細は <http://www.embeddedmulticore.org/>
- (参考) SHIM WG Primary Contributing Members
 - Cavium Networks, CriticalBlue, eSOL, Freescale, Nagoya University, PolyCore Software, Renesas, Texas Instruments, TOPS Systems, Vector Fabrics, and Wind River.



Embedded
Multicore
Consortium

www.embeddedmulticore.org



Embedded
Multicore
Consortium

www.embeddedmulticore.org