組込み向け SoC を支える RISC-V と AI 技術・その実例

HIDEKI SUGIMOTO
CTO NSITEXE, INC.

自己紹介

杉本 英樹 - NSITEXE CTO

- 8086 互換 MPU（V30, 33…）の回路設計～評価～量産移管～製品担当
- V850（初代）の ISA, μArch 設計, 製品企画・論理設計
  以降 CPU の内部構造改善数回
- V850E1 CPU コアの ISA拡張, μArch, S/W model…
  以降 MCU 製品, ASIC（PDC とか IJ Printer とか）多数
- V850E2 の企画途中で MIPS 系分門へ移動
- V_R シリーズ向け オリジナル μArch CPU コア開発
  Navi, DTV, DVR 等に使われた
- 制御系アーキ改革のため？に MCU 向け CPU コア開発に戻る
  仮想化, SMT 等を実装するも、その時は日の目を見ず…（早すぎた？）
- 半導体業界を見限り？、デンソーへ転職
  半導体からは足を洗ったつもりだったが…
- 半導体業界再生？のため NSITEXE 立上げ

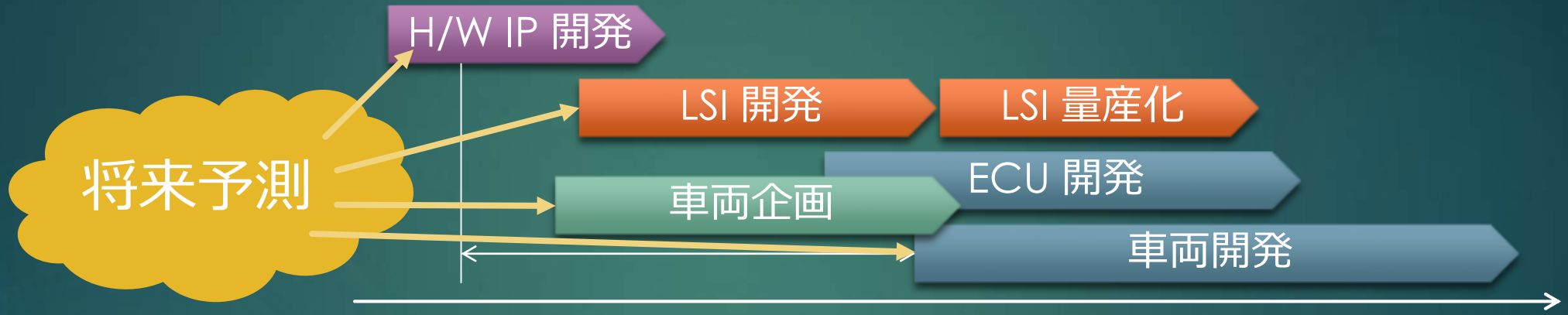
組込／車載領域を取り巻く状況

システム価値とその要素の変化

トップダウン設計による 対象システムへの最適化 → 高性能・高効率・高品質	→	市場トレンドへ変化への 追従の速さと多様性対応 → フレキシビリティ
実現手段が価値を律速 → 専用化、最適化が価値 の源泉	→	実現時間が価値を律速 → 手段の汎用化や事前準 備が価値の源泉
先を読み、十分な時間を かけて確実に開発・製品化 →「固い」開発プロセス	→	先読みが困難な中で必要な 開発時間をとって品質確保 →「柔い」開発プロセス
製造コストの低減中心 （大量生産でコスト低減） → 開発費配賦＜＜製造	→	開発コストの低減も課題 （生産数量増＜開発費増） → 開発費配賦＜＞製造？

SDA (Software Defined Architecture) 化の流れ

何が予測出来て、何が予測困難か？



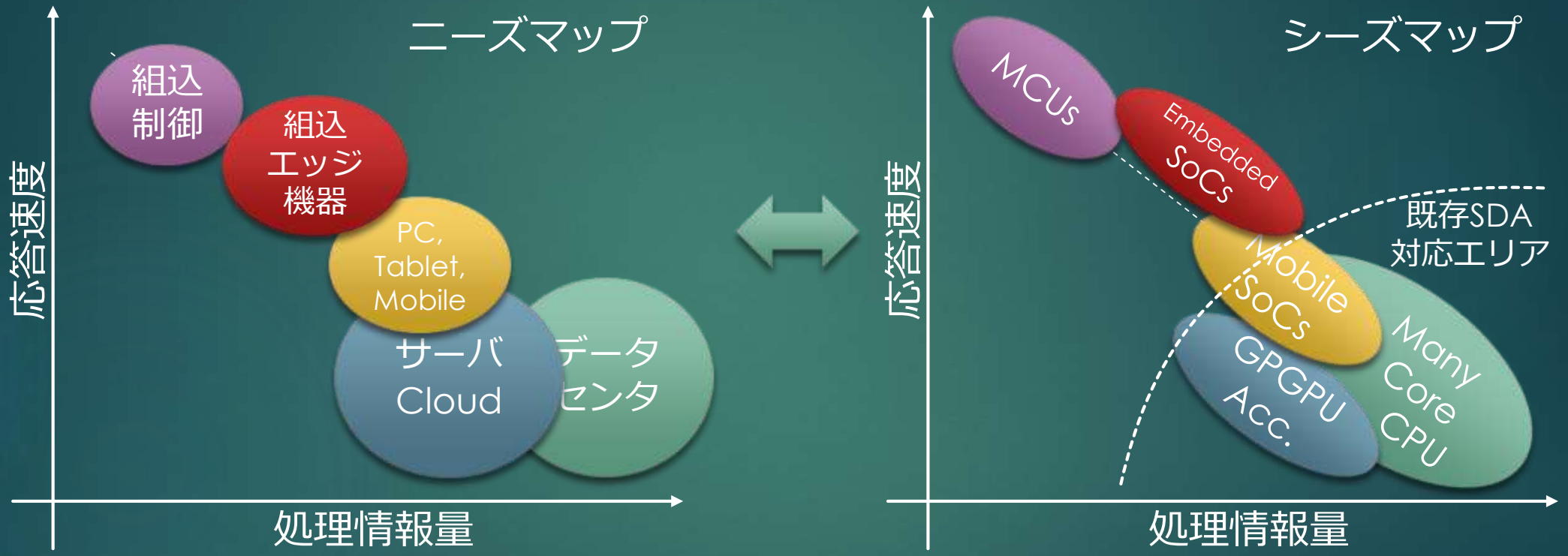
これまでは、
将来予測は明確でなくても
ある程度暗黙の了解があった
→ それぞれ先を予測して開発していれば、最終的に車両（システム）を開発する際のトップダウンフローの中ですり合わせ可能であった

これからは、
変化の速さと多様化で
将来予測は困難になり、
暗黙の了解は成立せず
機能とその実現手段の分離（独立化）必要に
→ SDA が必要とされる大きな理由の一つ

では、単に組込分野に既存の SDA を持ってくれば解決するのか？

プロセッサ特性（応答時間／情報量）

簡単に SDA 化とは言うけれど…

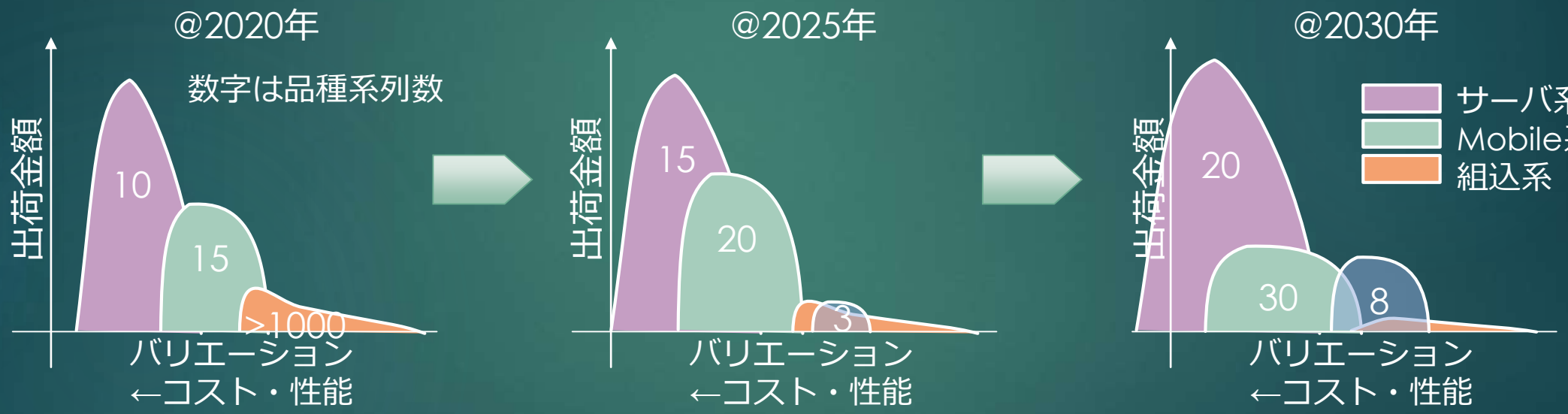


SDA だけでハードウェアを意識しなくてよくなるわけではない！
→ 本質的な処理特性の差は埋められない

SoC ビジネス環境と課題

では独自 SoC を作ればよい？

→ 将来を見据えて？ H/W の共通化が進められてきたサーバ, Mobile 領域に対し、個別最適化を進めてきた組込系は SoC のビジネス成立性が低い！



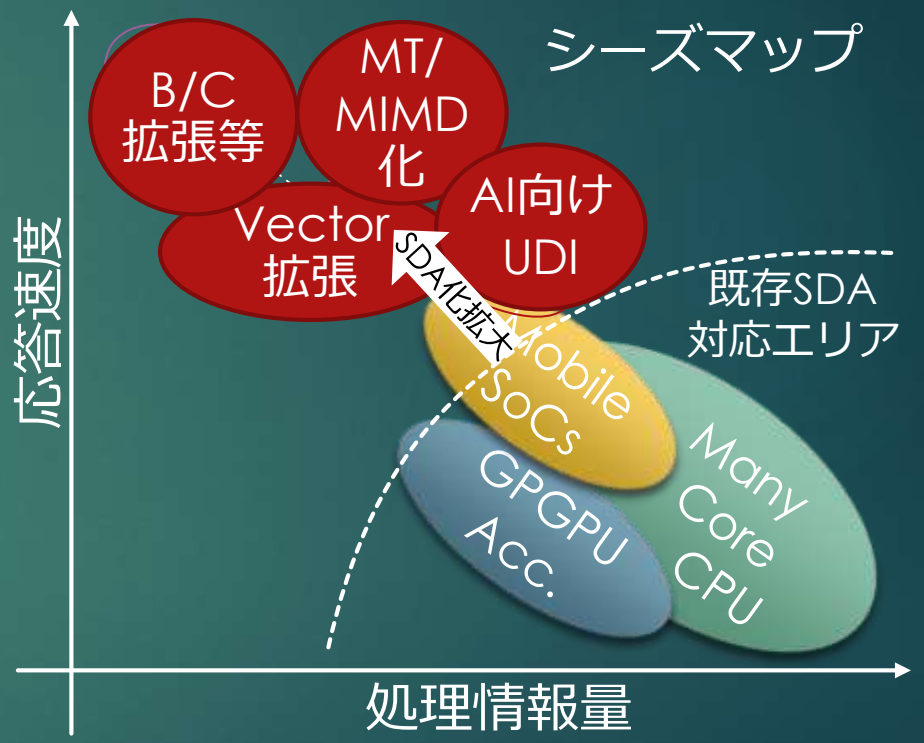
作らねばシステムが成立しない or 非常に効率が悪いが、簡単には作れない → RISC-V は如何にしてこの状況を救えるのか？

RISC-V への期待と現状

どうやってアーキテクチャの穴を埋めるか？

SDA 対応 H/W の要件

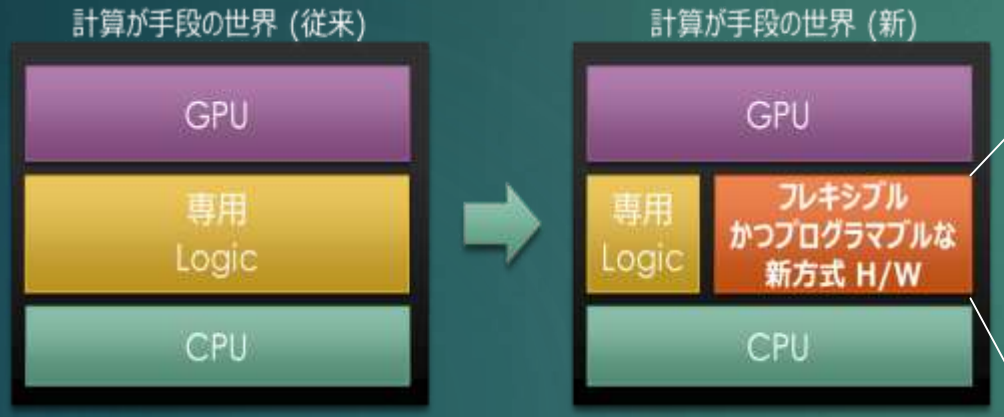
- ✓ 将来予測される処理特性をカバー
  → アーキ管理下での各種拡張
- ✓ 処理特性（≒プロセッサ方式）に依存しないプログラミングモデル
  → 標準エコシステムへの準拠
- ✓ 性能スケーラビリティ
  → 各社得意領域を活かした協調
- ✓ 十分なアプリ・カバレッジ
  （SoC ビジネス成立するための条件）
  → 統一リファレンスシステム適用と SiP 等 SoC H/W 技術の進化を利用したバリエーション展開容易化

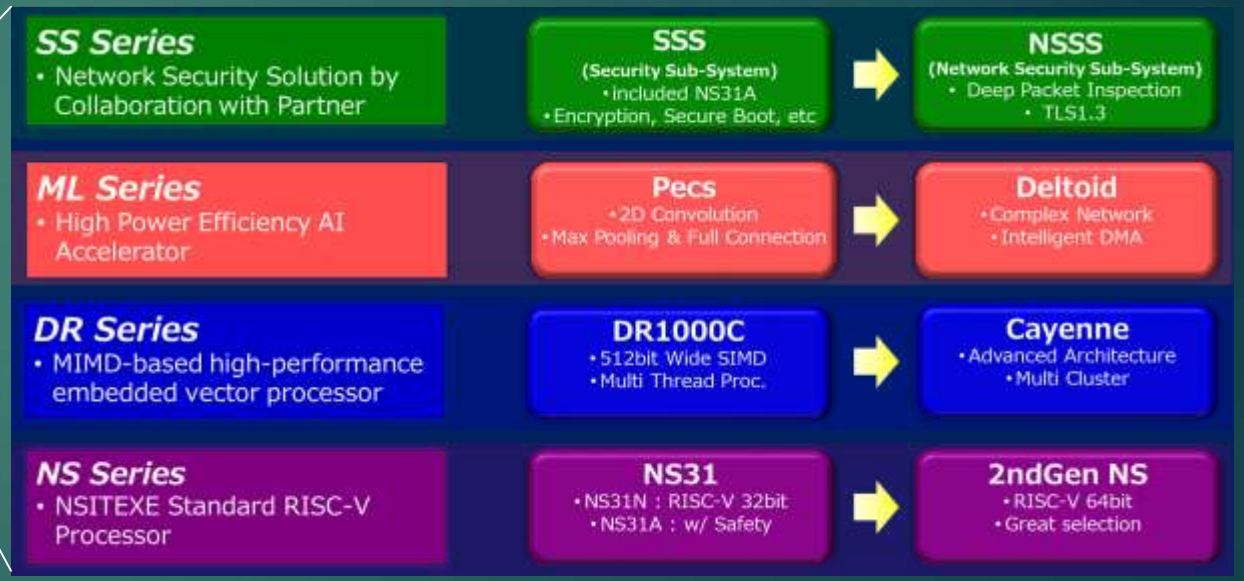


SoC応用の動向と，我々が目指すところ

エコシステムの穴を埋める（例え儲けにくくても…）

→ 既存領域の延長線上の領域は他社にお任せし、特に専用 Logic からプロセッサへの移行が難しい部分をシステム・プロセッサ両知見を活かし推進

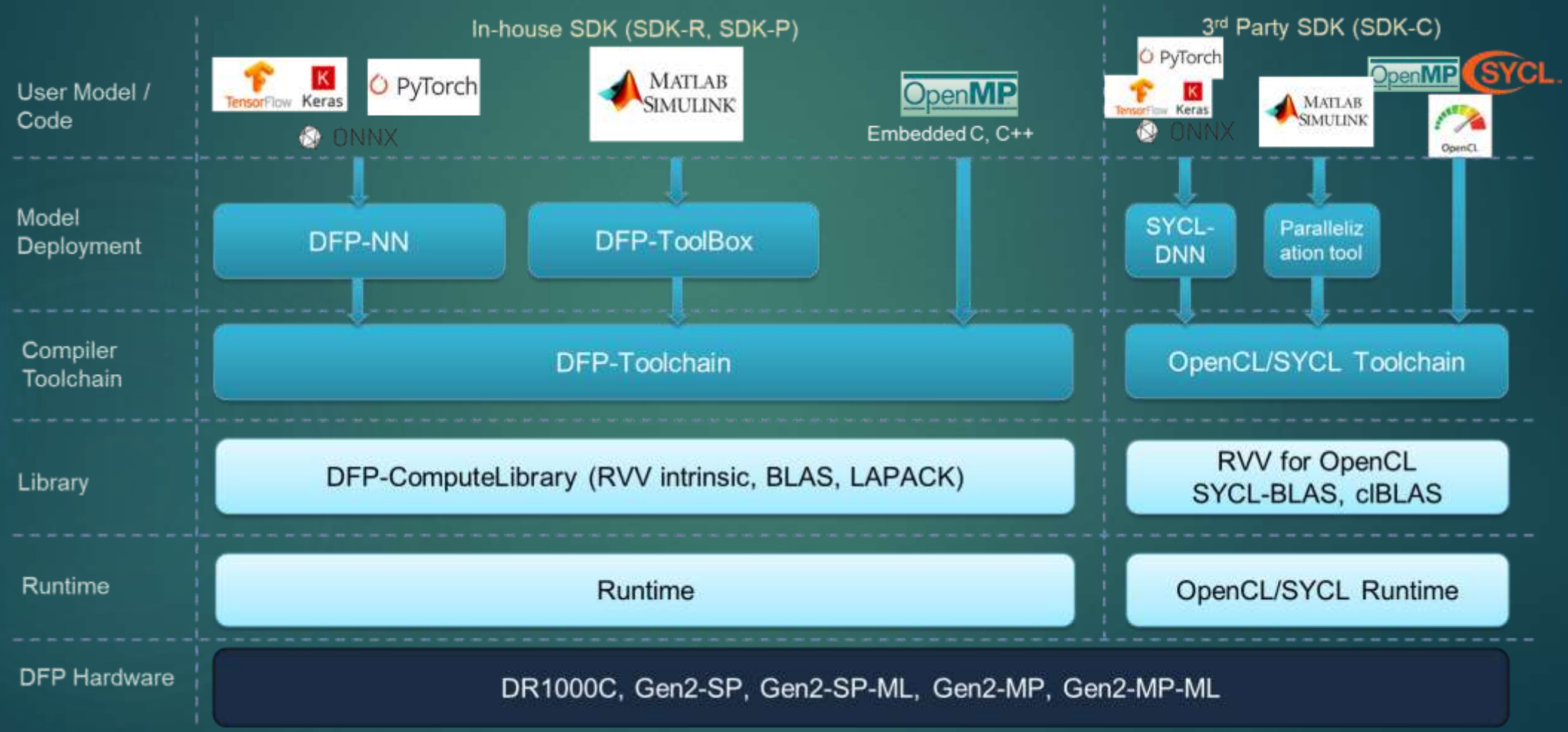




特定の機能に結びつかず
将来ないと困ると思われる
（が今必須な性能ではない）
→ 正直売りにくいが、だからこそやる意義がある！

我々が目指すところ（続き）

統一されたプログラミングモデル



RISC-V, AI の応用例 – 制御系

ハイエンドだけの AI ではない！

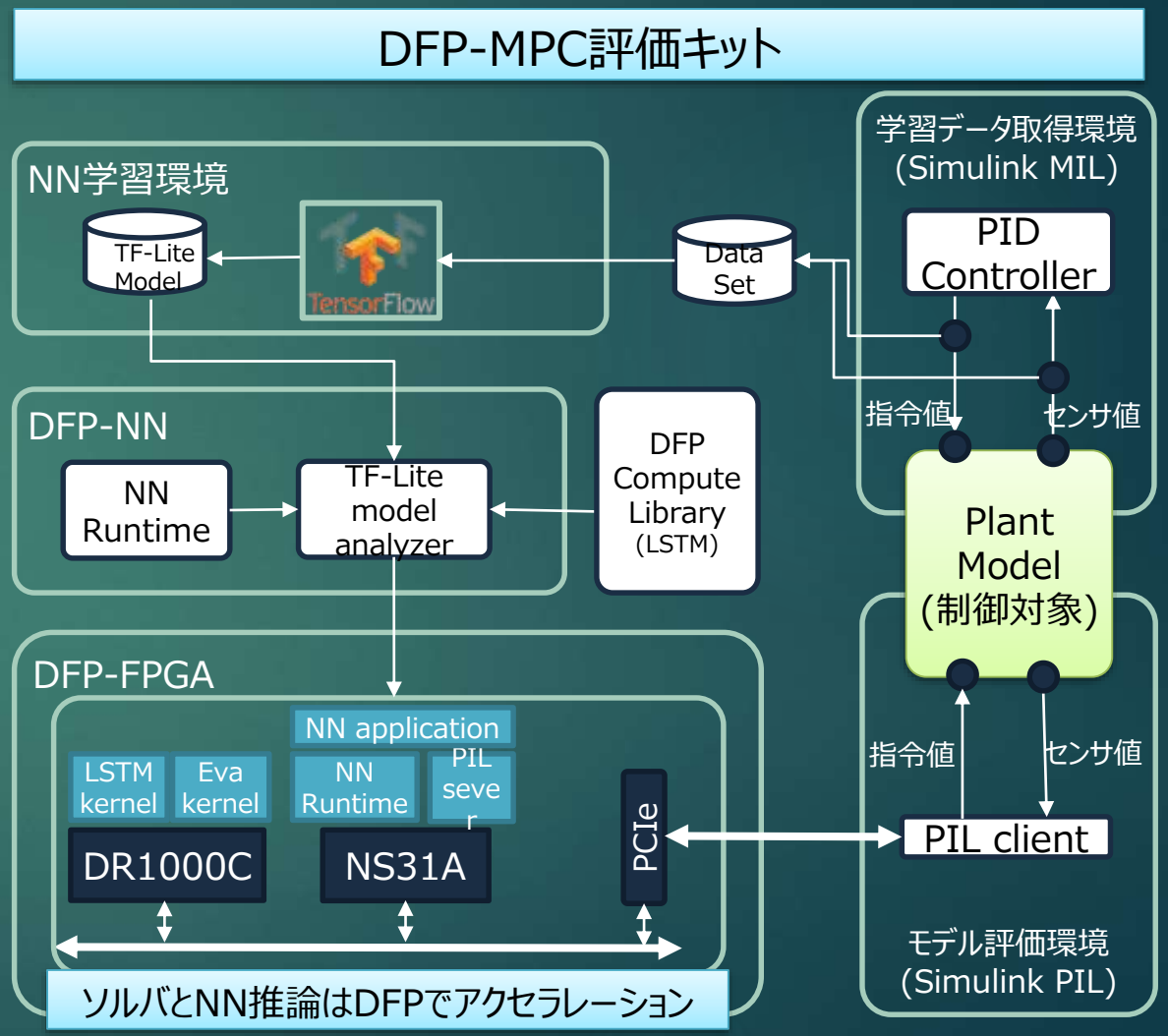
- PID制御の課題
  - 所望の制御特性を得るため実機を用いた試行錯誤的なパラメータチューニングが必要
- 従来MPC制御の課題
  - ソルバとプラント出力予測演算負荷が重く制御周期に間に合わない
  - ソルバ動作のためプラントモデル、評価関数に制約あり
- DFP-MPCの特徴
  - メタヒューリスティックソルバ(ABC)採用
    - 様々なプラントモデル、評価関数へ対応が可能
  - DNNによるプラント出力予測
    - 学習により高精度モデルを自動的生成
    - 再学習により個体ばらつき、 経年劣化に追従可
  - DFP（RISC-V Vector拡張）による演算処理の高速化

DFP-MPC	ユーザーメリット
評価関数を最小化する最適操作量を出力	制御性能向上: • FA: スループット向上 • パワートレイン: 燃費、電燃向上 • 車両制御: 乗り心地向上
メタヒューリスティックソルバにより 任意の評価関数の最適化が可能	制御目的に合わせた任意の評価関数を設計出来るため、制御特性のチューニング工数を削減可: • 複数の制御目的を同時に取り扱える • トレードオフ関係にある制御目的のバランスを調整可
NNを活用したプラントモデル構築	• 高精度なプラントモデルを自動生成できるのでモデル作成工数を削減可 • 再学習により、個体ばらつき、経年劣化に追従しモデル精度を確保

RISC-V, AI の応用例 – 制御系 (続き)

利用イメージ

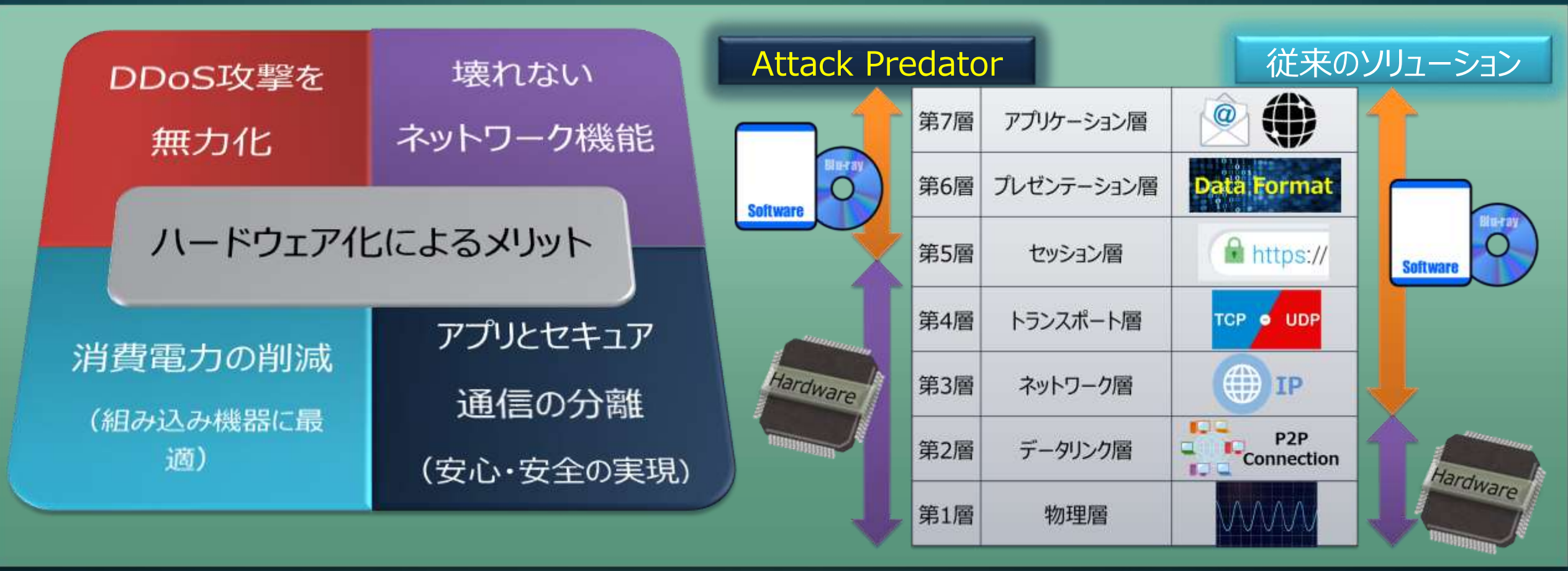
- Step1：Simulinkモデルの仮学習、制御特性評価
  - 制御対象プラントモデルを既存手法で制御し学習データを取得
  - Tensor Flow上で学習
  - 同じプラントモデルをDNN予測モデルによるMPCで制御しモデルの妥当性を評価
- Step2：実機によるモデル再学習、制御特性評価
  - 実機データに基づいてモデルを再学習させ精度を向上させる



RISC-V, AI の応用例 – データ処理系

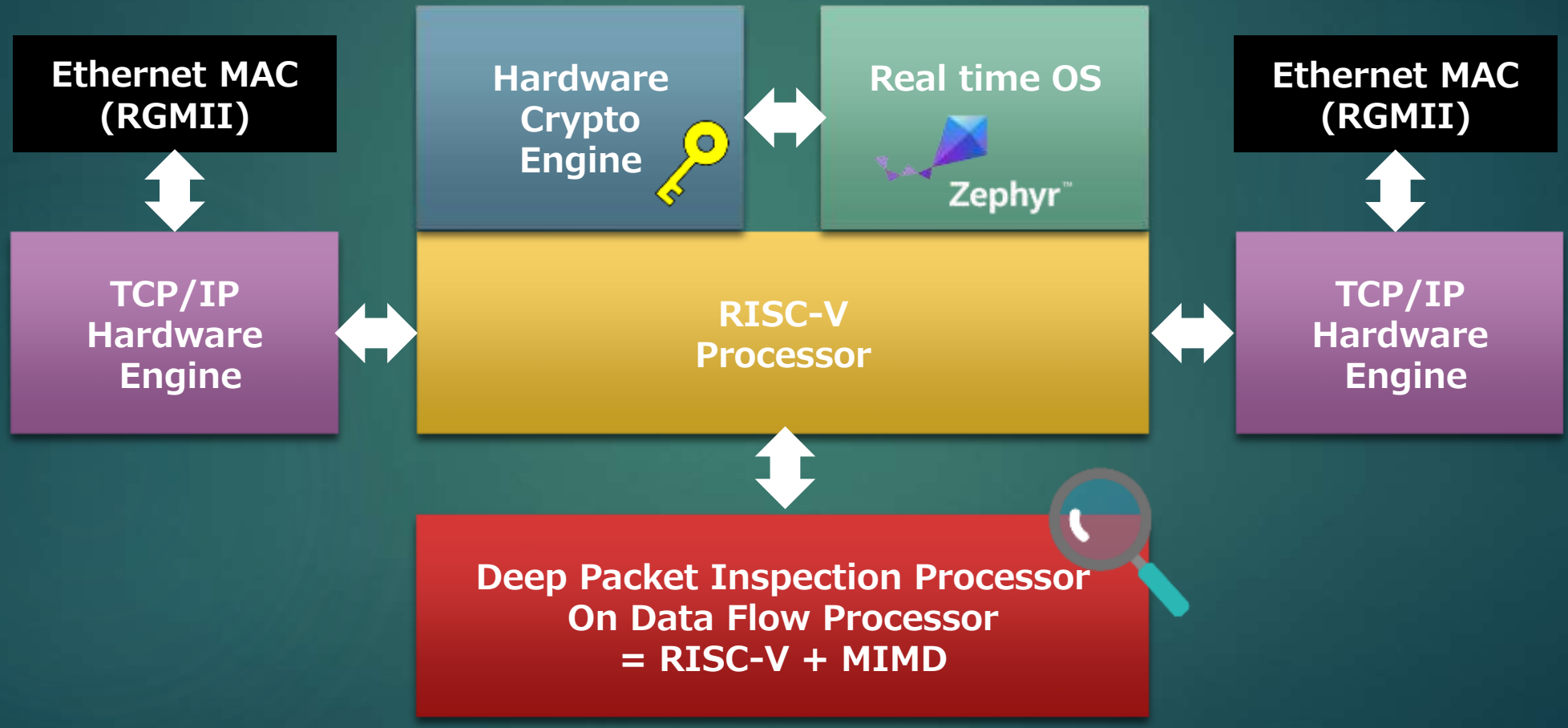
想定外を想定内に！

Attack Predator は**ネットワーク第５層の高負荷処理までハードウェアにて実現（安心・安全・堅牢）**



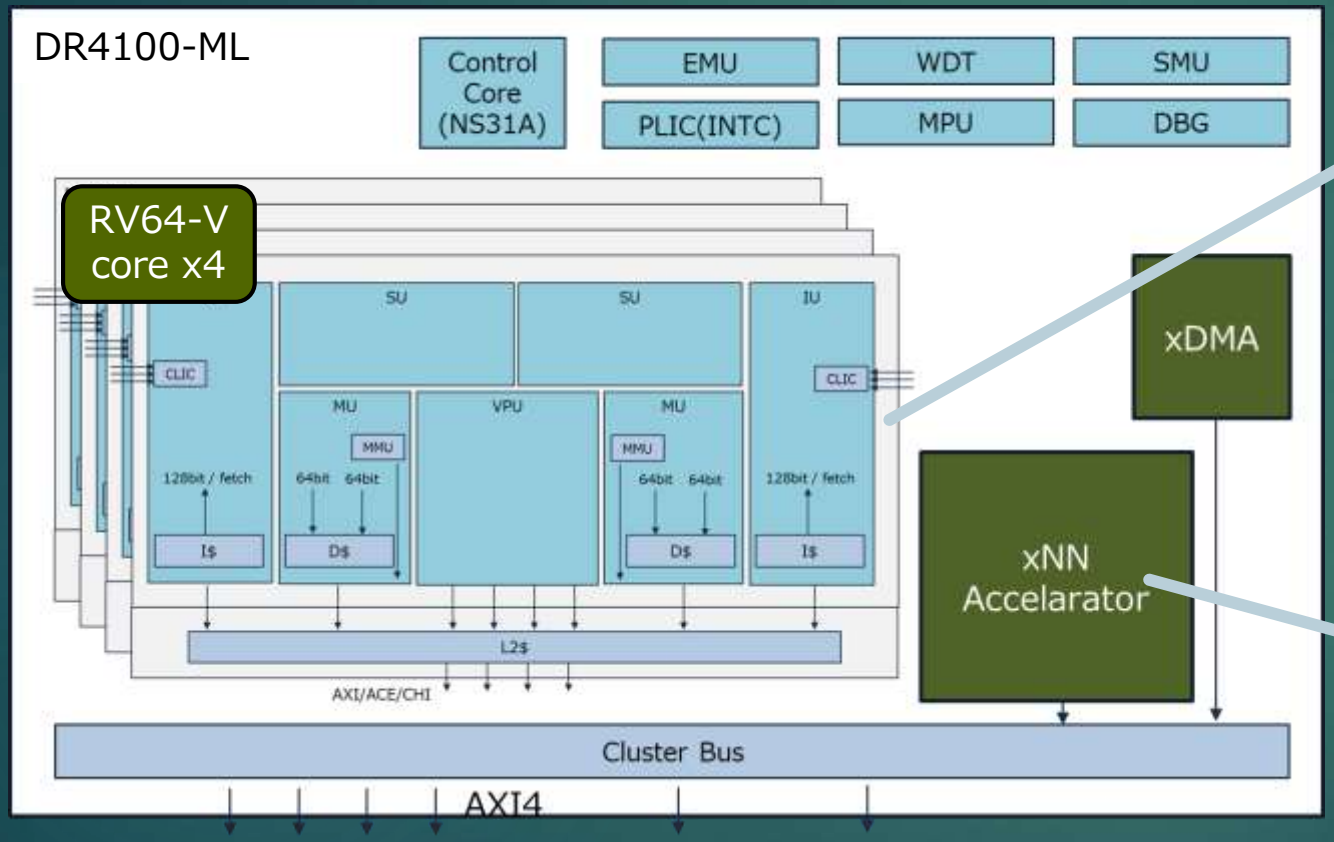
RISC-V, AI の応用例 – データ処理系 (続き)

内部構成図

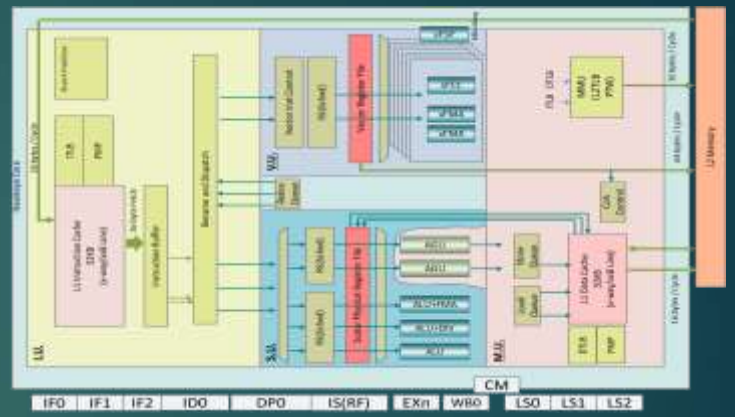


RISC-V, AI の応用例 – アクセラレータ

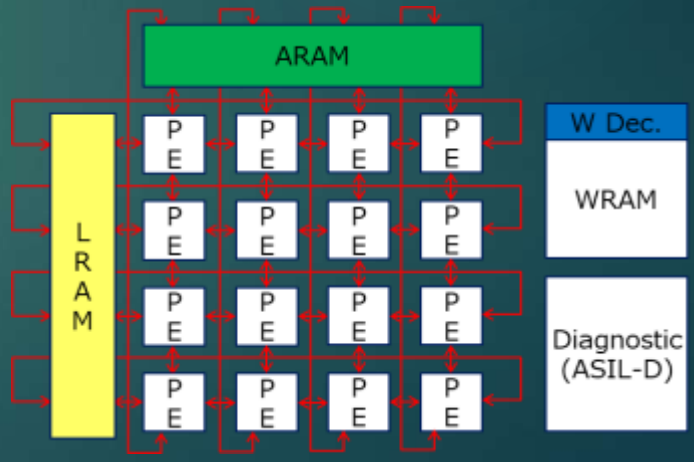
組込の特性に向けたアクセラレータも



Next Gen. DFP w/ AI Accererator



RV64 Processor Core

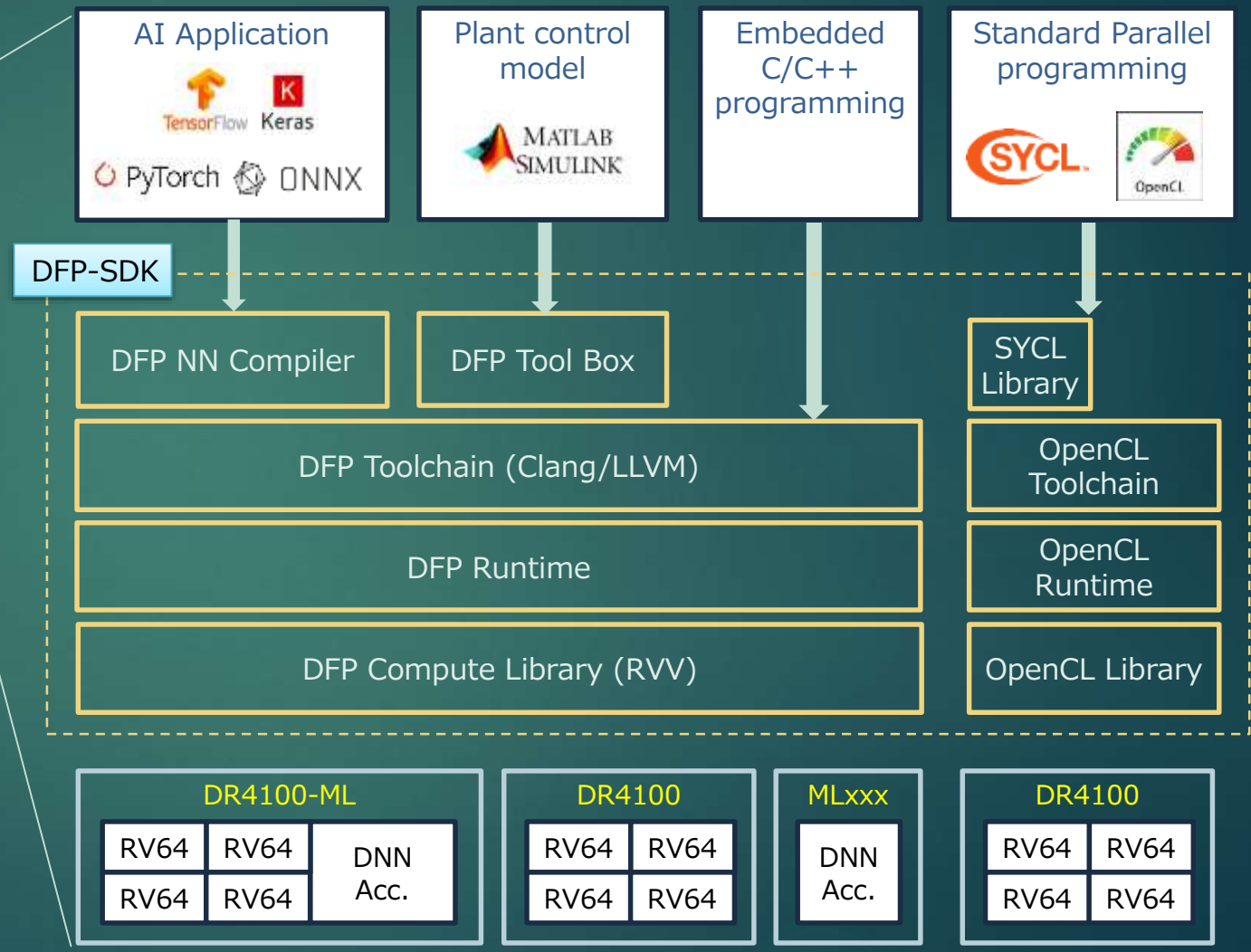


DNN accelerator

RISC-V, AI の応用例 – S/W 構成

当然 統一 SPF へ組込み

- Highly functional and easy to implement industry-standard AI and plant control models in DFP
- Seamlessly enable PC-based application development with DFP
- Compiler for efficient NN layer fusion
- High performance and reliable real-time thread control
- Automotive quality and functional safety



まとめと今後の展望

- 組込領域でもシステムの開発トレンドは SDA へ
- SDA 時代に合った、かつ組込み特性を意識した SoC が求められる
- RISC-V はこの時代の要請に合ったソリューションである
- より広く使える、共通的なソフトウェアフレームワークが必要

➔ 今後、ISA のみでなく HAL 層（e.g. Cahce 制御）に関する標準化や、サブシステム（OS Timer, 割込み系等）のリファレンス・デザインの提供等、周辺活動の充実がされるとさらに普及に拍車がかかるはず