第12回 CEST技術セミナー

組込みシステム開発技術の今後10年のチャレンジ

2009年2月17日

高田 広章
名古屋大学 大学院情報科学研究科 教授
附属組込みシステム研究センター長
NPO法人 TOPPERSプロジェクト 会長
Email: hiro@ertl.jp　URL: http://www.ertl.jp/~hiro/

目次

組込みシステム開発の現状と課題

Hiroaki Takada

組込みシステム開発を取り巻く状況

半導体技術の進歩

従来からの組込みシステムの大規模化・複雑化

- 機器の複合化・デジタル化・ネットワーク化
- 制御要素に情報処理要素が複合
- コンピュータ制御による高機能化・高付加価値化

組込みシステムの適用分野が拡大

- コンピュータの小型化・低価格化により広がる適用分野

開発期間の短縮やコストダウンに対する要求

- 製品の早期の市場投入が収益を大きく左右
- QCDの3つを同時に満たすことは困難．どれかが犠牲になっている（どれが犠牲になっているかは分野による）

（単一の）プロセッサの高速化の限界

- 消費電力（＝発熱量）が最大の制約条件に

組込みシステム開発のアチーブメント

レクサス LS460

- 100個以上のECU（車載組込みコンピュータ）
- 組み込まれたソフトウェアの総ライン数が7,000,000行（*A4の紙にプリントすると約10万ページに*）

（いずれも雑誌の報道による数値）

http://www.lexus.jp/

- 日本の（ないしは，トヨタグループの）組込みシステム開発能力の高さを実証

! 次のモデルを同じ手法で開発することは不可能との声も

組込みシステムソフトウェア開発の課題

設計品質の確保／向上

- システムの複雑化により，ディペンダビリティ（信頼性，安全性，セキュリティ，…）の確保が困難に

設計の効率化（生産性の向上）

- システムの大規模化や品質要求により，設計効率が低下

その他にも多くの技術的課題（主なもの）

- 新しいハードウェア技術（例: マルチコア）への対応
- これまで以上に高い信頼性・安全性の達成
- 低消費電力（エネルギー）システム技術
- 組込みセキュリティ技術

その背景にある深刻な問題

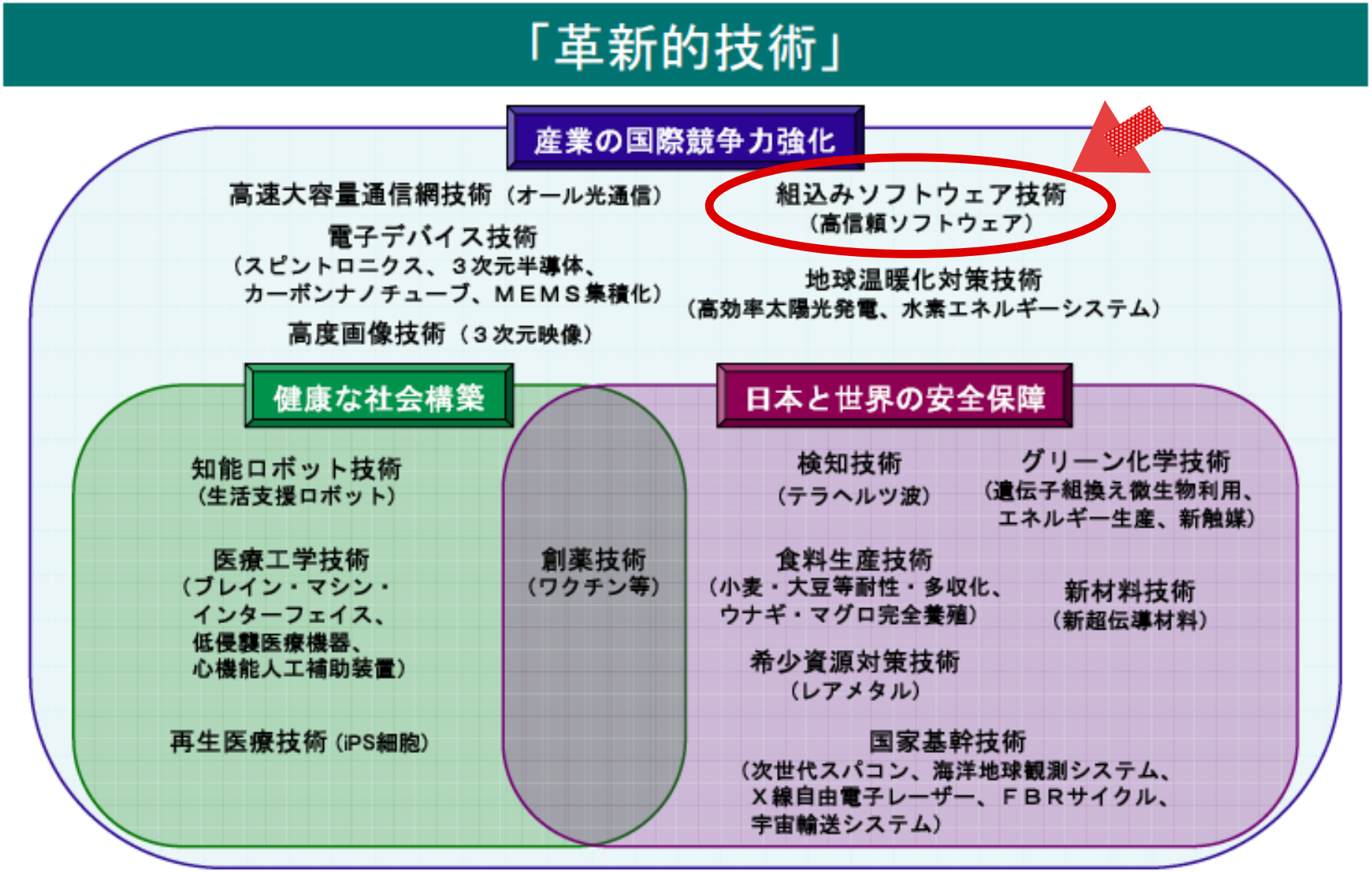
組込みシステム技術者（人材）不足

組込みソフトウェアに関する政府の動き

- 政府においても，2003年頃から組込みシステム/ソフトウェアの重要性の認識が広がり，経済産業省や文部科学省において積極的な取組み

総合科学技術会議

- 第3期科学技術基本計画の分野別戦略（情報通信分野）で，次のテーマが重要な研究開発課題に挙げられた
  - 高信頼・高安全・セキュアな組込みソフトウェア設計開発技術
  - System-on-Chip技術と組み込みソフトウェア技術
- 本年5月に採択された「革新的技術戦略」において，「組込みソフトウェア技術（高信頼ソフトウェア）」が，我が国の産業の国際競争力強化に必要な革新的技術として挙げられている



総合科学技術会議:「革新的技術戦略」より

文部科学省

- 科学技術振興調整費（新興分野人材養成）「組込みソフトウェア技術者の人材養成」… 名古屋大学で実施
- CRESTの研究領域「実用化を目指した組込みシステム用ディペンダブル・オペレーティングシステム」

経済産業省

- 2003年度より組込みソフトウェア産業実態調査を実施
- ソフトウェアエンジニアリングセンター（2004年10月設立）は，組込みソフトウェア開発力強化を3本柱の1つに
- 戦略的基盤技術高度化支援事業において，組込みソフトウェア分野が設定（2006年度～）
- 産業技術研究開発委託費（産学連携ソフトウェア工学実践事業（高信頼組込みソフトウェア開発））を，JasParが受託（2007年度～）

組込みシステムソフトウェアの多様化と分類

組込みシステムの多様性と分化

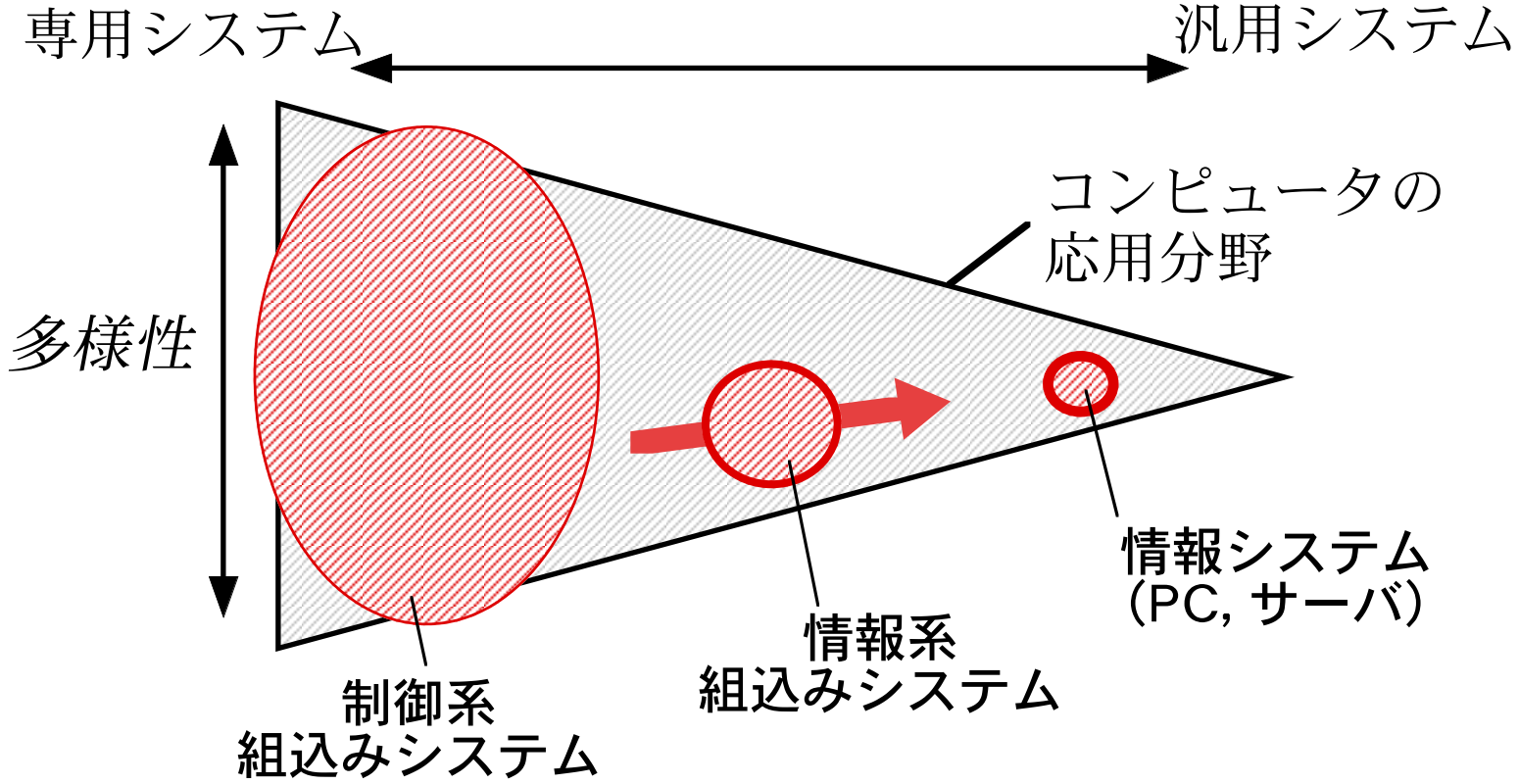
- 組込みシステムは，システムの規模・性質いずれの面からも極めて多様である
- 携帯電話機やカーナビなどのモバイル情報機器は，従来の組込みシステムと汎用システムの中間的な性質

組込みシステム/ソフトウェアの大分類

- 組込みシステム/ソフトウェア技術は，少なくとも次の3つの分野に分けて考えることができる
  - 制御系組込みシステム
  - 情報系組込みシステム
  - システムLSIへの組込みソフトウェア
- ただし，実際の組込みシステム/ソフトウェアは，複数の分野に跨がっていることが多い

専用システム技術の多様性と汎用システム技術の適用

- 専用システム技術は，アプリケーション特有の要求事項に対して特化して設計されるため，必然的に多様になる
- 汎用システム技術の適用可能性について常に注意が必要



技術分類毎の要求の違い

制御系組込みシステム

- 信頼性, 安全性, リアルタイム性が重視される
- そのためには，過度な複雑性は持ち込みたくない
- 処理性能は (多くの分野で) ほどほどで良い

情報系組込みシステム

- 多くの機能の実現が要求される．必然的に大規模なソフトウェアに
- セキュリティ確保が重要
- マルチメディア処理以外は，処理性能はほどほどで良い

システムLSIへの組込みソフトウェア

- 限られた消費電力下で高い性能が求められる
- それを達成するために専用のハードウェアとソフトウェア

近付く組込みシステム開発の限界

これで良いのか？ 組込みソフトウェア開発

- 人海戦術的な開発体制
- 開発管理にかかるオーバヘッドが大きい
- レベルの高い技術者が開発管理だけやっていてよいのか？
  - ソフトウェア開発は，子会社・協力会社の社員や，派遣社員が実施していることがほとんど
- ソフトウェア技術者の能力をちゃんと活かせているか？
  - ソフトウェア作成能力が飛び抜けて高い人材が埋没していないか？

- 品質を確保し，開発期間を厳守する（納期を守る）ためにはやむをえない面があることはわかるが…

次に起こると思われること

- **現状**：組込みソフトウェアの開発期間は，製品の出荷時期から，ハードウェア（メカ，エレキ）の開発期間を考慮して決定されている

- **次には**：組込みソフトウェアの開発期間が，製品の開発期間（＋製品の出荷時期）を決定するようになる

! エンタプライズソフトウェアでは，普通の現象

例）東京三菱UFJ銀行の経営統合

- 合併する両銀行のシステム接続の遅れにより，銀行合併を3ヶ月遅らせる
- システム統合（最大6000人の技術者，11万人月の開発規模）も半年遅延
- 移行に伴うトラブルも発生

不況下の組込みシステム開発

不況に強い組込みシステム（一般論としては）

- 組込みシステム開発は，製品開発の一部であるため，不況下でも開発を止めるわけにはいかない
- とは言え，勝ち組（力のある企業）と負け組の格差が拡大

車載組込みシステムのこの1年の変化

- QcDからQCDへ
  - 開発コスト削減要求が厳しく，開発の効率化が重要な話題に
  - 部品コスト削減のために，機能を厳選するケースも
- 安全から省エネルギーへ
  - 世界的なエネルギー価格高騰・エネルギー危機から，省エネルギーに対する要求が高まる

組込みシステム開発技術の方向性と今後10年のチャレンジ

Hiroaki Takada

組込みシステムに対する要求の変化

! 組込みシステムの社会インフラ化

汎用・多機能 vs. 専用・単機能

- それぞれのシステムの複雑化はさらに進むと思われる
- すべての機器がネットワーク接続されれば，すべての機器が汎用・多機能である必要はない
  - パラダイムチェンジの時期の見極めが難しい

制御と情報処理の統合

- ディペンダビリティ要求の深化
  - セキュリティと安全性（セーフティ）の両立
- 組込みシステムと情報システムの統合（例：ITS）
  - 異なる開発文化の擦り合わせ

消費電力あたりの性能の向上

- 新しいハードウェア技術の導入が必要

組込みシステムの開発効率化と品質確保のために

! *QCD要求を満たすシステム/ソフトウェア開発のために*

設計抽象度を上げる

- 設計物を少ないライン数で記述できるようにする
- ! 一人の技術者が一定期間に記述できるソフトウェアのライン数は大きくは変わらないと言われている

設計資産の再利用の促進

- 過去の設計資産を再利用することで新規開発分を減らす
- 設計資産の質の向上が重要

応用分野毎のプラットフォームの構築・活用と共通化

- 応用分野毎に，それに向いたハードウェア上に，リアルタイムOSと必要なミドルウェアを載せたプラットフォームを構築し，広範に活用する

設計抽象度を上げる

ハードウェア設計

- 動作レベルでの設計（プログラミング言語で記述し，そこからRTLを生成）

ソフトウェア開発

- ソフトウェアのモデルベースの設計（モデルを記述し，そこからプログラムを生成）
- ソフトウェアのモデル化には，状態遷移図（UMLのステート図など）／状態遷移表やブロック線図（MATLAB/Simulinkなど）を用いる（向き不向きの見極めが重要）
- モデルからのプログラム生成技術へのより一層の取り組みが必要（アスペクト指向的な考え方の導入が必要）

システムレベル設計

- ハードウェアとソフトウェアを一体で設計 → 後述

設計資産の再利用の促進

設計資産の質の向上

- 良質な設計資産の蓄積（社内で，業界レベルで）が重要
  - 設計資産の再利用は重要だが，レガシー化は避けなければならない
- 設計ドキュメントとして残すことは言うまでもなく重要だが，設計根拠を残すべき

再利用を前提とした開発プロセス

- 設計資産の品質を維持するためには，開発プロセスや組織から考えることが重要

  → プロダクトライン型ソフトウェア開発

ソフトウェアの部品化の促進 → 後で紹介

プラットフォーム化の促進 → 次に紹介

プロダクトライン型ソフトウェア開発

- 製品を系列（上位機種・下位機種, 仕向け地の違い, バージョンアップ）で開発する場合に着目して，ソフトウェアの再利用性を向上させるための開発手法

プロダクトライン開発の流れ

- 製品群と開発方法を分析して，再利用範囲を決定
- 共通に使えるソフトウェア開発の資産（アーキテクチャ, コンポーネント, ドキュメント, 開発・テストツール, …）を事前に開発．これをコア資産と呼ぶ
- コア資産を組み合わせて製品を開発
- 製品開発の結果を，コア資産にフィードバック

プロダクトライン適用のポイント

- 単なる開発手法の見直しに留まらず，（必要に応じて）マネジメントや組織の見直しに踏み込むこと

ソフトウェア開発効率化のための一提案

基本的な考え方

- 製品のtime-to-marketを短縮するためには，製品開発の開始前に開発できるソフトウェアは，先行して開発しておくこと（先行開発）が有力な方法
- 先行開発フェーズにおいては，製品開発のような厳密な工程管理は必要なく，開発効率向上の余地が大きい

先行開発の課題

- アプリケーションの要求から開発するソフトウェアの仕様を決めるというトップダウンの開発手法は取れない
  ➡ ボトムアップ的な開発になる
- 将来のアプリケーションの要求と使用できるハードウェア技術を予測することが必要
- その時点で使用できるソフトウェア技術を動員

先行開発できる部分

- OSやミドルウェアは，製品仕様にはそれほど影響されない．おおよその製品特性がわかっていれば，開発できる
- プロダクトライン開発におけるコア資産

先行開発の特性とその活用

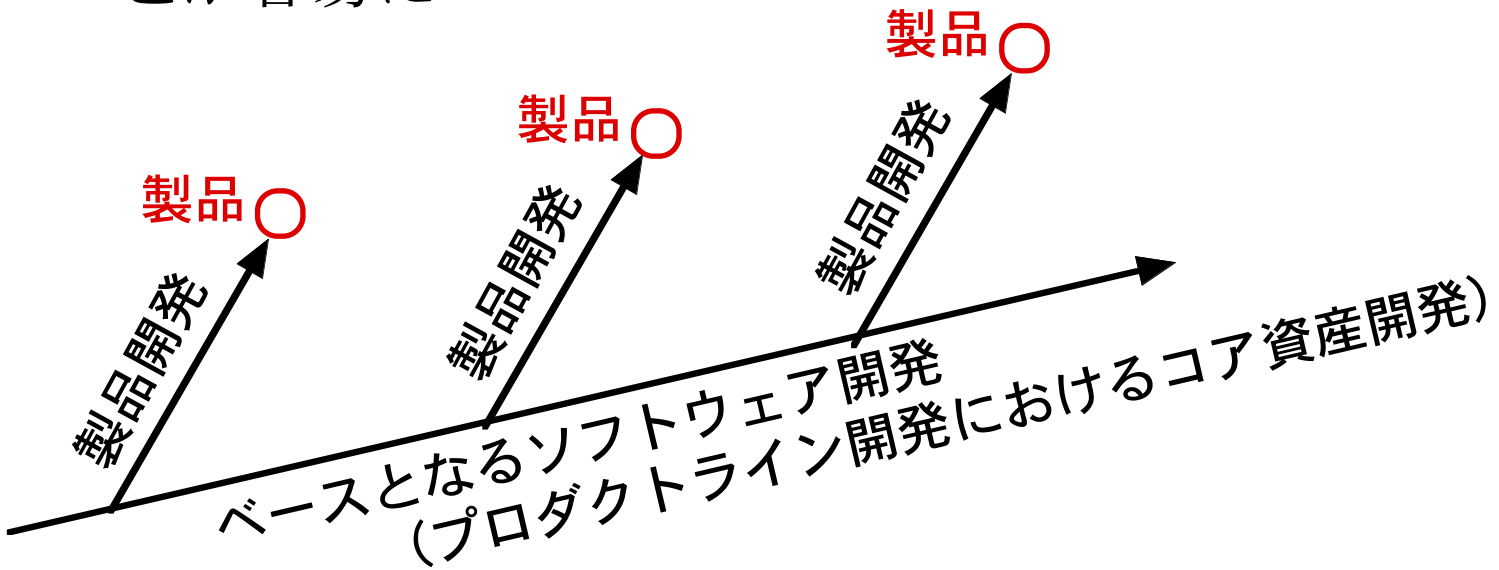
- 開発期間（納期）を守ることは，強くは求められない
- 品質の高いソフトウェアが開発できれば，価値が高い

- 人海戦術的な開発体制を取る必要がなく，レベルの高い技術者だけで開発することが可能
- 職人的/芸術的なソフトウェア開発が許容できる
- アジャイル開発などの管理オーバヘッドとリスクが低い開発プロセスを採れる

ベースとなるソフトウェアの開発進度を考慮した製品企画

- 製品企画において，要素技術（主にハードウェア技術）のトレンドを見るのは当然
- ベースとなるソフトウェア開発（コア資産の開発）の進度も見るようにする

  ➡ 小さいオーバヘッドで開発期間（納期）を守ることが容易に



プラットフォームの構築・活用と共通化

応用毎のプラットフォーム構築

- 多様な組込みシステムを，1つのプラットフォームでカバーするのは不可能
- 適切な範囲（これが難しい）の応用ドメインを設定し，それに向いたプラットフォームを構築

プラットフォーム活用によるコスト低減と品質向上

- 多くのアプリケーションで同一のプラットフォームを用いることで，（アプリケーションあたりの）プラットフォーム開発コストを削減
- システムの品質確保のために鍵となる部分であり，開発リソースを集中させて高品質なプラットフォームを構築することで，システムの品質向上につながる
- プラットフォームの共通化・標準化が重要に

プラットフォームの共通化・標準化の例

社内共通プラットフォーム

例) パナソニックのUniPhier (ユニフィエ)

- プロセッサとビデオコーデックなどを含むシステムLSIと，ミドルウェアやOSなどのソフトウェアからなるデジタル家電用の統合プラットフォーム

業界内でのプラットフォーム標準化活動

例) AUTOSAR → http://www.autosar.org/

例) JASPAR → http://www.jaspar.jp/

- 自動車制御システム向けのプラットフォームやツールの標準化

例) CE Linuxフォーラム → http://www.celinuxforum.org/

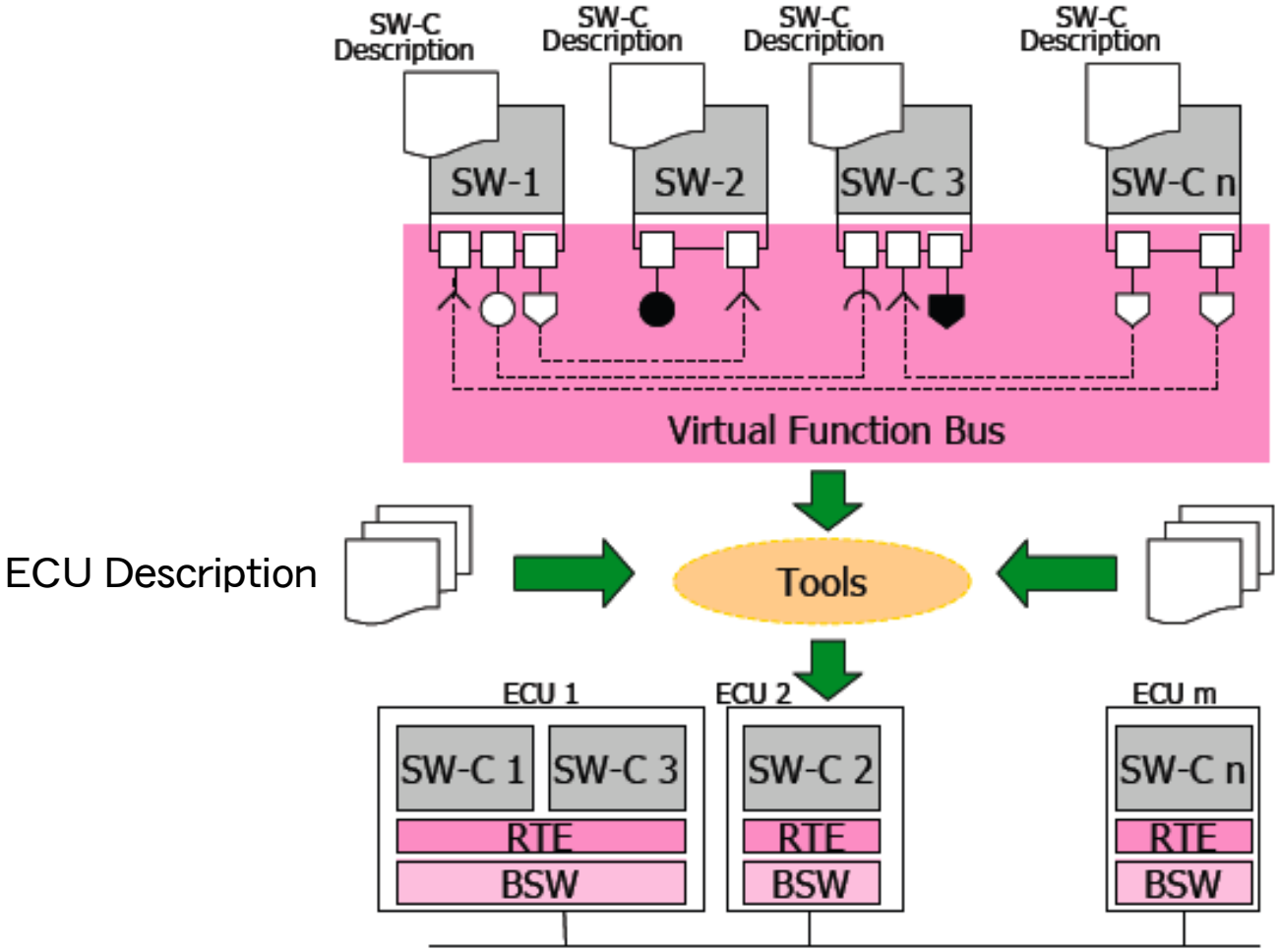
- デジタル家電機器向けのLinuxの仕様を定義

業界内でのプラットフォームのデファクト標準

AUTOSAR (AUTomotive Open System ARchitecture)

- 自動車, 自動車部品, エレクトロニクス, 半導体, ソフトウェア企業によるグローバルパートナーシップ (2003年に設立)
  - ソフトウェアの複雑性を軽減するために，ソフトウェア基盤 (infrastructure) の業界標準を作成
- コアパートナー
  - BMW Group
  - Bosch
  - Continental
  - Daimler
  - Ford
  - Opel
  - PSA Peugeot Citroen
  - トヨタ自動車
  - Volkswagen
- 最初の仕様書シリーズが2006年春から順次発行
- 改訂された仕様書シリーズ (バージョン3) が2008年春に発行．ウェブサイトからダウンロード可能

AUTOSARのアプローチ

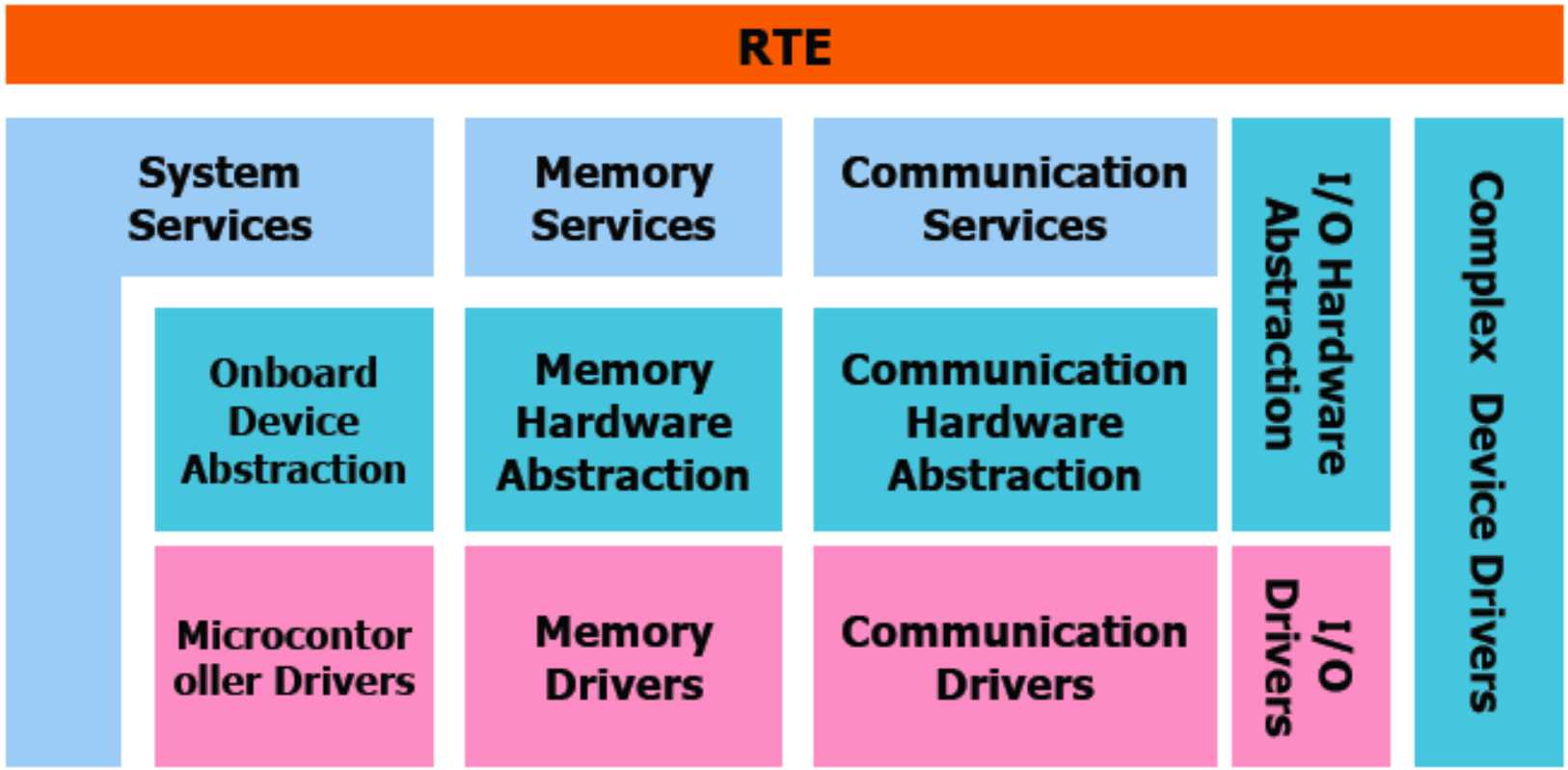


SW-C
Software Component
RTE
Runtime Environment
BSW
Basic Software

! ウェブサイトからダウンロードできる仕様書を元に紹介

Basic Softwareの構造

- Basic Software + RTE がプラットフォームを構成
- サービス層, ECU抽象化層, マイクロコントローラ抽象化層の3つの階層で構成



ハードウェア技術の変化

(まずは) マルチコアプロセッサ

- 消費電力あたりの性能向上のためには不可欠な技術
- ただし，普通に使いこなせるようになるには，もう少し時間 (技術開発と人材育成) が必要

その先は？(専用 vs. 汎用)

- メニーコア？
  - ハードウェア的には実現可能なレベルに来ているが，ソフトウェア開発がついていけるか？
  - 適用される分野は限定的か？
- (リ)コンフィギュラブルロジックの活用
  - ハードウェアの援用による性能向上
  - FPGA: 汎用チップ上に専用回路 (過剰な汎用性の排除)

今後の10年に向けての
我々の取組み事例

Hiroaki Takada

取組み事例1：システムレベル設計技術

システムレベル設計技術とは？

- システム設計のなるべく遅い段階まで，ハードウェアとソフトウェアを区別せずに設計を進める手法
  - ハードウェアとソフトウェアの境界を最適に決めること／変更することが容易に
- C言語のようなプログラミング言語レベルからハードウェアを合成する動作合成技術の進歩により，システム全体をプログラミング言語で記述することが現実的に
- アナログ回路やメカまで含んだシステムレベル設計も考えられるが，まだ研究段階
  - メカを含まない電子部分だけという意味で，最近では，ESL (Electronic System Level) という用語が使われる場合も

システムレベル設計技術の必要性

ハードウェア技術者とソフトウェア技術者の間の溝

- システム（ハードウェア＋ソフトウェア）の最適設計を進める上で，両者の間のコミュニケーションミスが大きな障害に

コミュニケーションミスの原因

- 使っている言葉が違う　典型例）テストと検証
- （定義により）設計している対象が違う
- 闘っている問題が違う → 何を重要視するかが違う
  - ハード技術者 … 性能（コストパフォーマンス）＋物理現象の不確実性
  - ソフト技術者 … 複雑性，大規模化 → 設計効率化
- 組込みソフトウェア技術者の被害意識

→ ***お互いに相手の問題意識を理解していない***

コミュニケーションミスによる問題点

- ハードウェア設計者とソフトウェア開発者のコミュニケーションミスにより，ムダな工数が発生
- 性能もムダになっていると思われる

解決に向けての方策

- ハードウェアと，それに密着したソフトウェア（デバイスドライバなど）は，一体で設計した方が効率的

  ➡システムレベル設計

  - ハードウェアとソフトウェアを一体で同一の言語（システムレベル言語）で記述する
  - そこから両者を自動生成（または，手設計）する
- ハードウェアとソフトウェアの両方のスキルを持ったシステムレベル設計を行う技術者／チームが必要

  プラットフォーム設計者/設計チーム

システムレベル設計環境：SystemBuilder

着眼点

- ハードウェア（デバイス）とそれを直接扱うソフトウェアを同一の言語で一体で記述（システムレベル記述）することで，設計効率化が図れる

  設計効率化のためのコデザイン

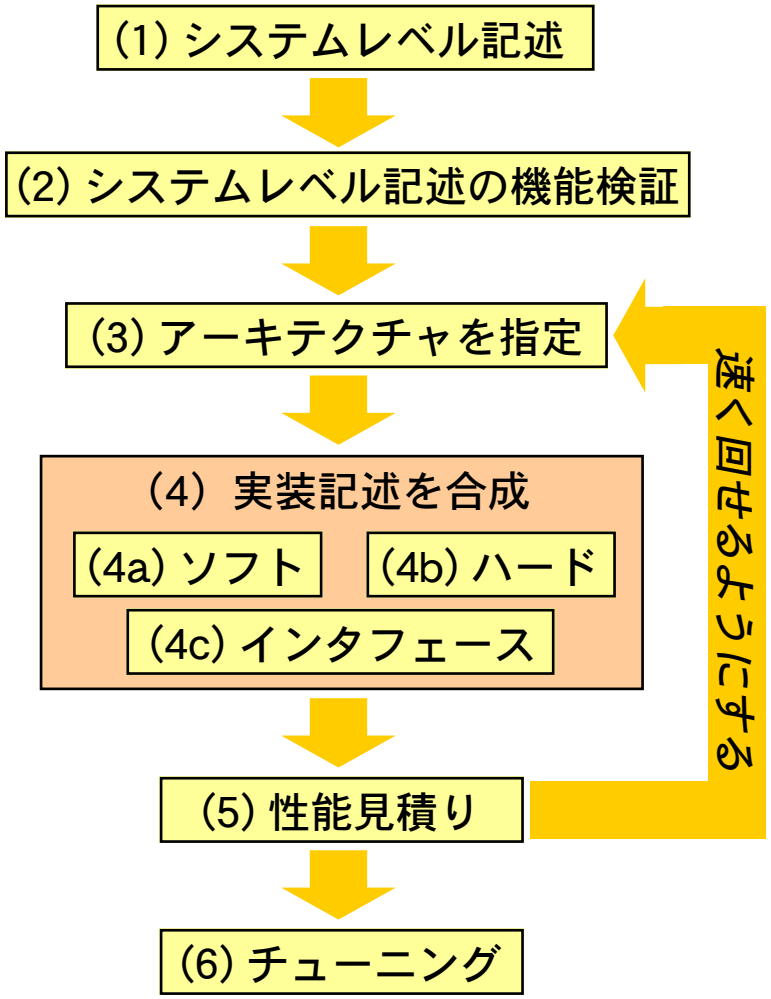
- ハードウェアとソフトウェア間のインタフェースを自動生成することで，インタフェース設計の抽象度を上げることができる

  抽象度の高いインタフェース記述

- アーキテクチャ選択の自動化は，現在の技術では困難．アーキテクチャ探索を容易に/迅速に行えることが重要

設計手法の流れ

- システムレベル言語により，ハードとソフトを一体で記述(システムレベル記述)
- システムレベル記述に対して機能検証を実施
- アーキテクチャを指定して，実装記述(ソフト，ハード，インタフェース)を合成
- 実装記述の性能見積り(コシミュレーションなどを利用)
- 性能見積り結果を評価し，必要ならアーキテクチャを変更して再試行
- 必要ならチューニングを実施

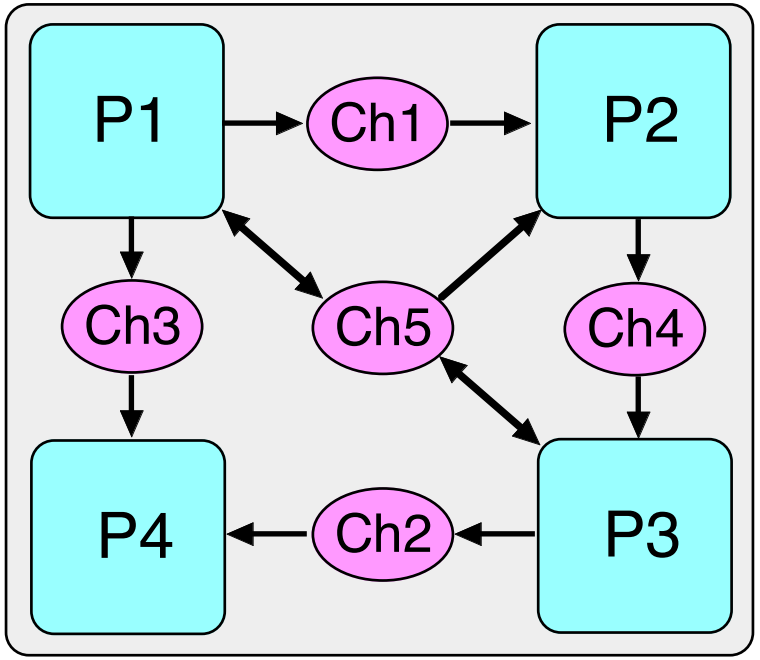


SystemBuilderの主な特徴

- C言語（ANSI C準拠）とシステム構成記述により，アプリケーションを記述
- ソフトウェア／ハードウェア分割は設計者が指定
- RTOS上で動作するソフトウェアを生成
- 動作合成ツール（市販のものを利用）により，論理合成可能なハードウェア記述を生成
- ソフトウェアーハードウェア間のインタフェースを生成
- 生成したソフトウェア／ハードウェアは，すべてのFPGA上で実行可能（ソフトコアプロセッサを利用）
- マルチプロセッサシステムをサポート（ただし，現状では最も単純なアーキテクチャのみ）
- 各種の精度でのソフトウェア／RTOS／ハードウェアの協調シミュレーション環境

SystemBuilderにおけるアプリケーション記述

- プロセス (P*i*)
  - 並行実行の単位
  - ソフト／ハード分割の単位
  - ソフトでは，タスクまたはスレッドに相当
  - ハードでは，モジュールまたはビヘイビアに相当
- チャネル (Ch*i*)
  - ノンブロッキング通信 (NBC) または レジスタ
  - ノンブロッキング通信 (BC) または FIFO
  - メモリ (MEM)



SystemBuilderにおけるシステム構成記述

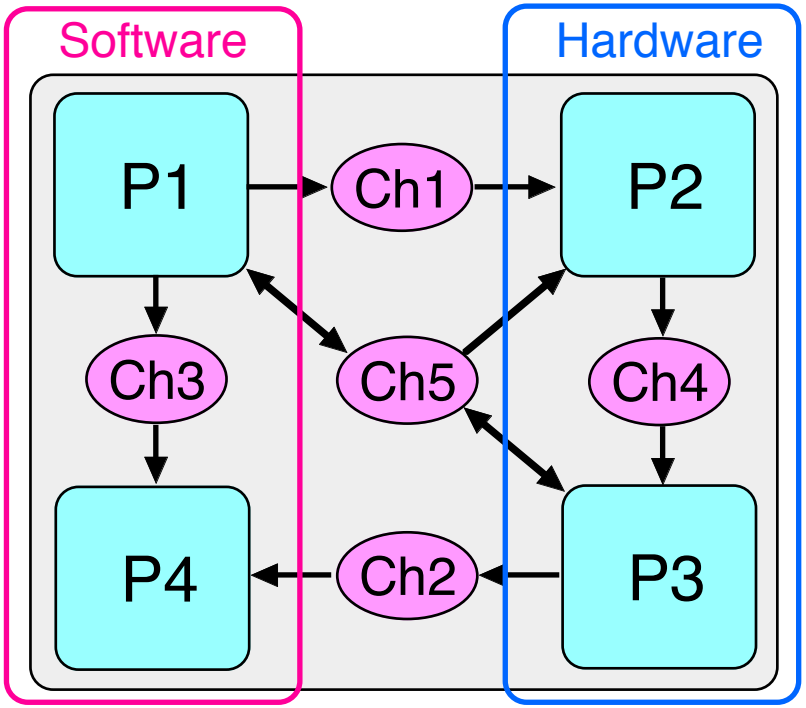
```
SYS_NAME = test
SW = P1, P4
HW = P2, P3

BCPRIM   ch1, SIZE = 32
BCPRIM   ch2, SIZE = 32
NBCPRIM  ch3, SIZE = 32
MEMPRIM  ch4, SIZE = 32
NBCPRIM  ch5, SIZE = 16

BEGIN_PROCESS
 NAME   = P1
 FILE   = "process1.c"
 USE_CH = ch1(OUT),
          ch3(OUT),
          ch5(INOUT)
END
```

```
BEGIN_PROCESS
 NAME   = P2
 FILE   = "process2.c"
 USE_CH = ch1(IN),
          ch4(OUT), ch5(IN)
END

BEGIN_PROCESS
 NAME   = P3
 PROCESS= "process3.c"
 USE_CP = cp2(OUT),
          cp4(IN), cp5(INOUT)
END

BEGIN_PROCESS
 NAME   = P4
 FILE   = "process4.c"
 USE_CP = cp2(IN), cp3(IN)
END
```

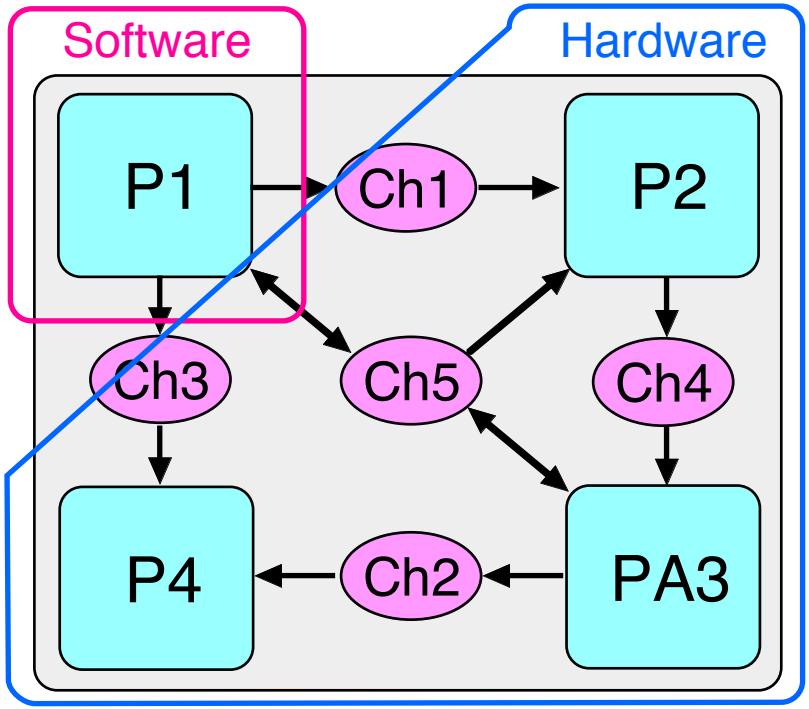
SystemBuilderにおけるソフト／ハード分割

- 設計者は，システム構成記述中に，ハード／ソフト分割を記述する

SW = P1, P4
HW = P2, P3

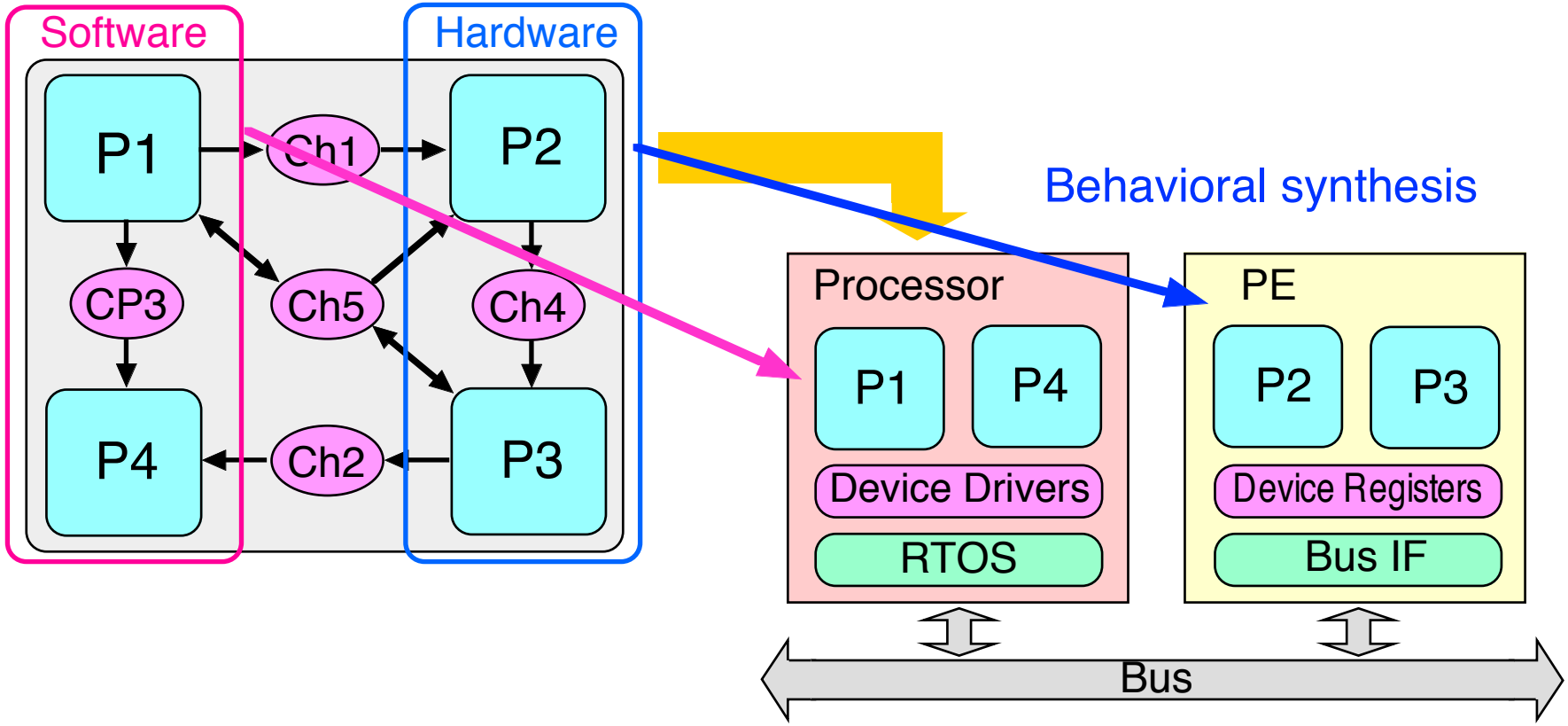


SW = P1
HW = P2, P3, P4



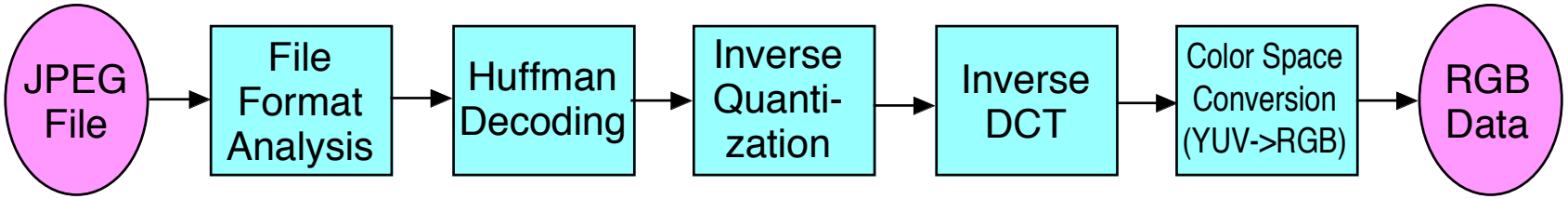
SystemBuilderにおける実装記述の合成

- インタフェース（ソフトーハード間の場合はデバイスレジスタとデバイスドライバ）を合成
- RTOSやBus IFは設定したものを利用



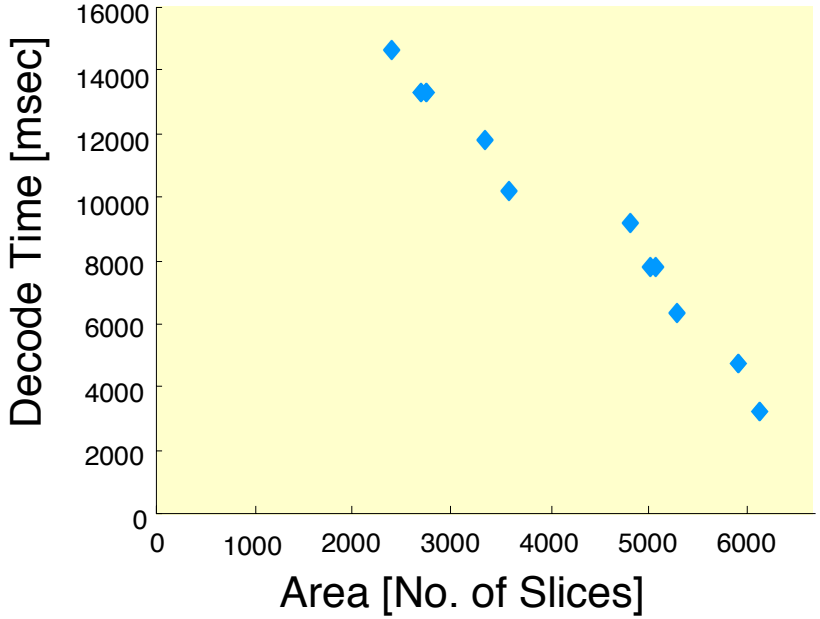
SystemBuilderにおける設計例

- JPEGデコーダを5つのプロセスに分割して記述



- FPGA上に実装して評価（ソフトコアプロセッサを利用）
- 右のグラフは，約10通りのソフト／ハード分割で面積と性能を評価

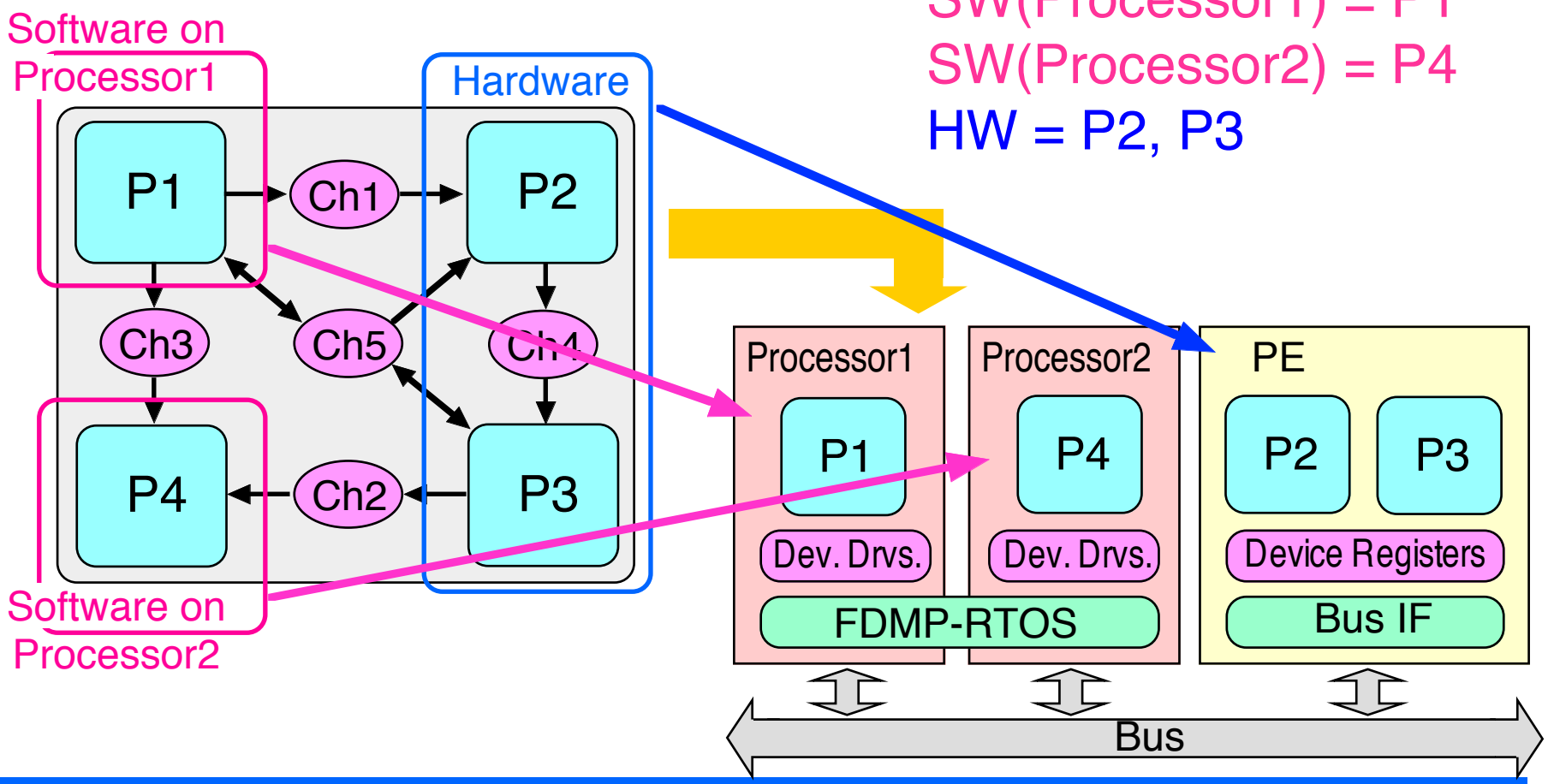
! 数時間で実施可能



SystemBuilderのマルチプロセッサ拡張

- ソフトで実装されるプロセスに対して，マッピング先のプロセッサを指定

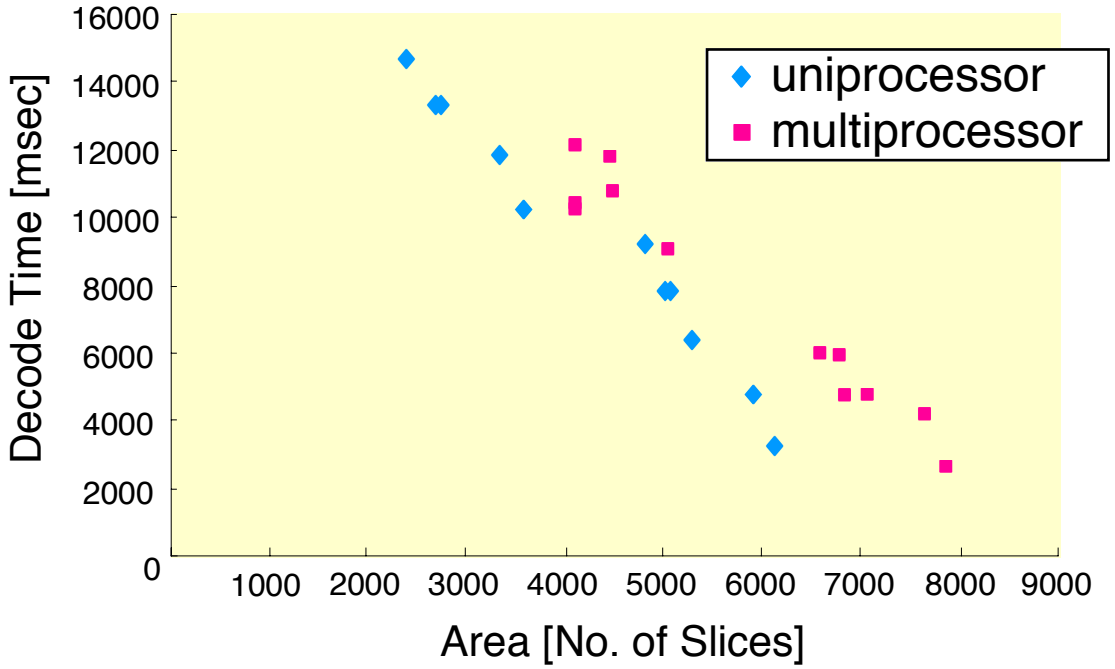
SW(Processor1) = P1
SW(Processor2) = P4
HW = P2, P3



マルチプロセッサシステムの設計例

- ソフトコアプロセッサを2つ用いてFPGA上に実装
- 約10通りのソフト／ハード分割で面積と性能を評価

! こちらについても，数時間で実施可能



システムレベル通信インタフェース

通信チャネルとインタフェースの定義

- 抽象度が高いがデータの流れを表現できる，標準的な通信チャネルと，そのインタフェースを定義
  - 状態変数チャネルとFIFOチャネル (基本)
  - チャネルの種類毎に複数種類のインタフェースを定義

例) FIFOチャネルのパケットタイプインタフェース

書込み側	読出し側
Ch.acquireRoom(tmout)	*Ch.acquireData(tmout)*
Ch.tryAcquireRoom( )	*Ch.tryAcquireData( )*
Ch.store(&variable)	*Ch.load(&variable)*
Ch.getPointer(&p)	*Ch.getPointer(&p)*
Ch.releaseData(tmout)	*Ch.releaseRoom(tmout)*

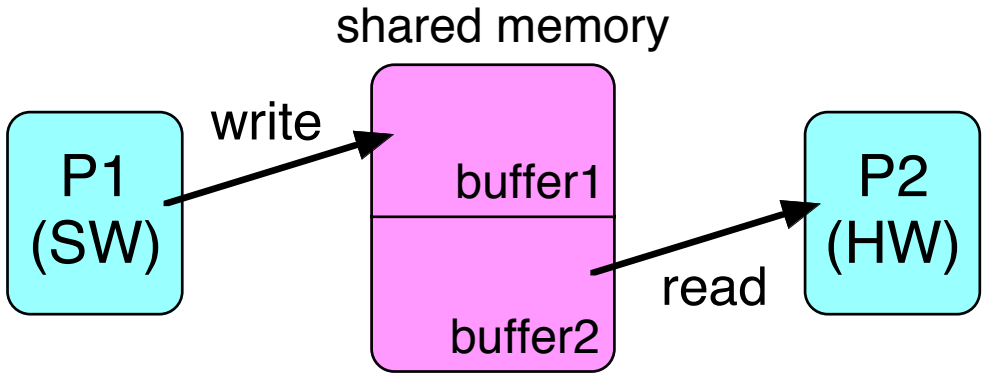
- 設計資産の再利用性向上のためには標準が重要

インタフェース合成手法

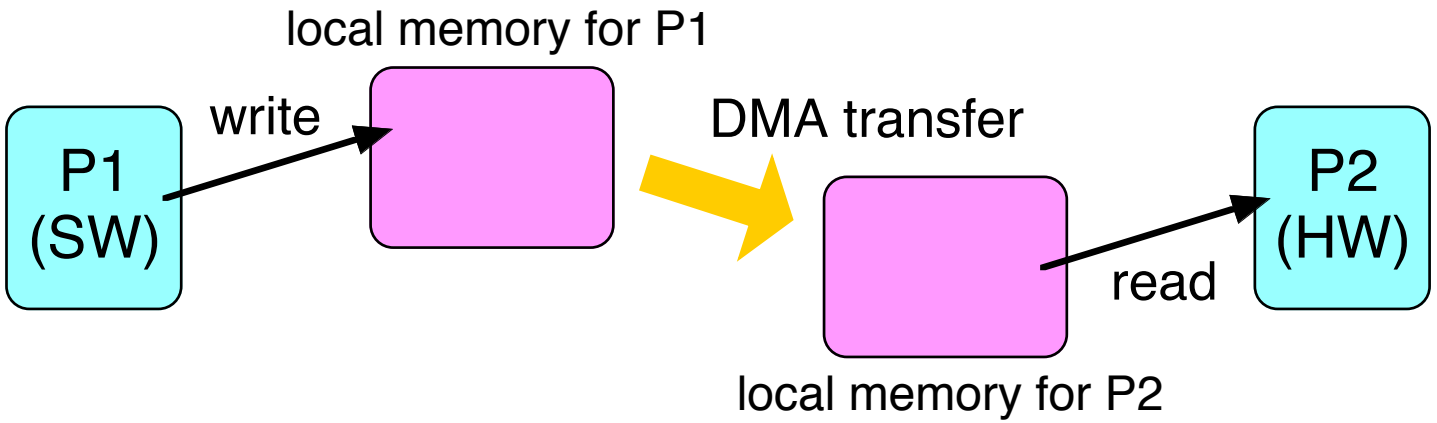
- 定義した通信チャネル・インタフェースから，効率のよい実装を合成する手法
- 接続先のプロセスの実装形態に応じて，3通りのインタフェース形態
  - SW－SW間通信インタフェース
  - HW－SW間通信インタフェース
  - HW－HW間通信インタフェース
- さらに，プロセス間の接続形態（アーキテクチャ）に応じて，最適なインタフェースを合成
  - 共有メモリによる通信インタフェース
  - DMAによる通信インタフェース

例）FIFOチャネルのパケットタイプインタフェースを実現する2種類のSW－HW間通信インタフェース

- ダブルバッファによる通信インタフェース



- ローカルメモリ+DMAによる通信インタフェース



取組み事例２：TECS

TECS（TOPPERS組込みコンポーネントシステム）とは？

- 組込みシステムを構成するソフトウェアモジュールを，コンポーネント（部品）として扱うための技術
  - WindowsのCOM（およびDCOM）に対応する技術
  - コンポーネント間を静的に結合．ツール（インタフェースジェネレータ）が最適なインタフェースを生成
  - 分散システムにおけるRPCにも対応予定
- コンポーネントWGにおいて，仕様の検討と，サポートツールおよびカーネル対応の開発を実施中

開発状況

- TECSのインタフェースジェネレータと，ASPカーネルのシステムログ機能やシリアルドライバ等をTECS対応にしたものを，会員向けに早期リリース中

組込みシステム向けにコンポーネント技術の必要な背景

- ソフトウェア部品の流通性・再利用性向上に対する要求
  - ミドルウェアを外部から導入することが多くなっているが，ポーティングにかかるコストが大きく，導入するメリットが低い
  - デバイスドライバが流通していない

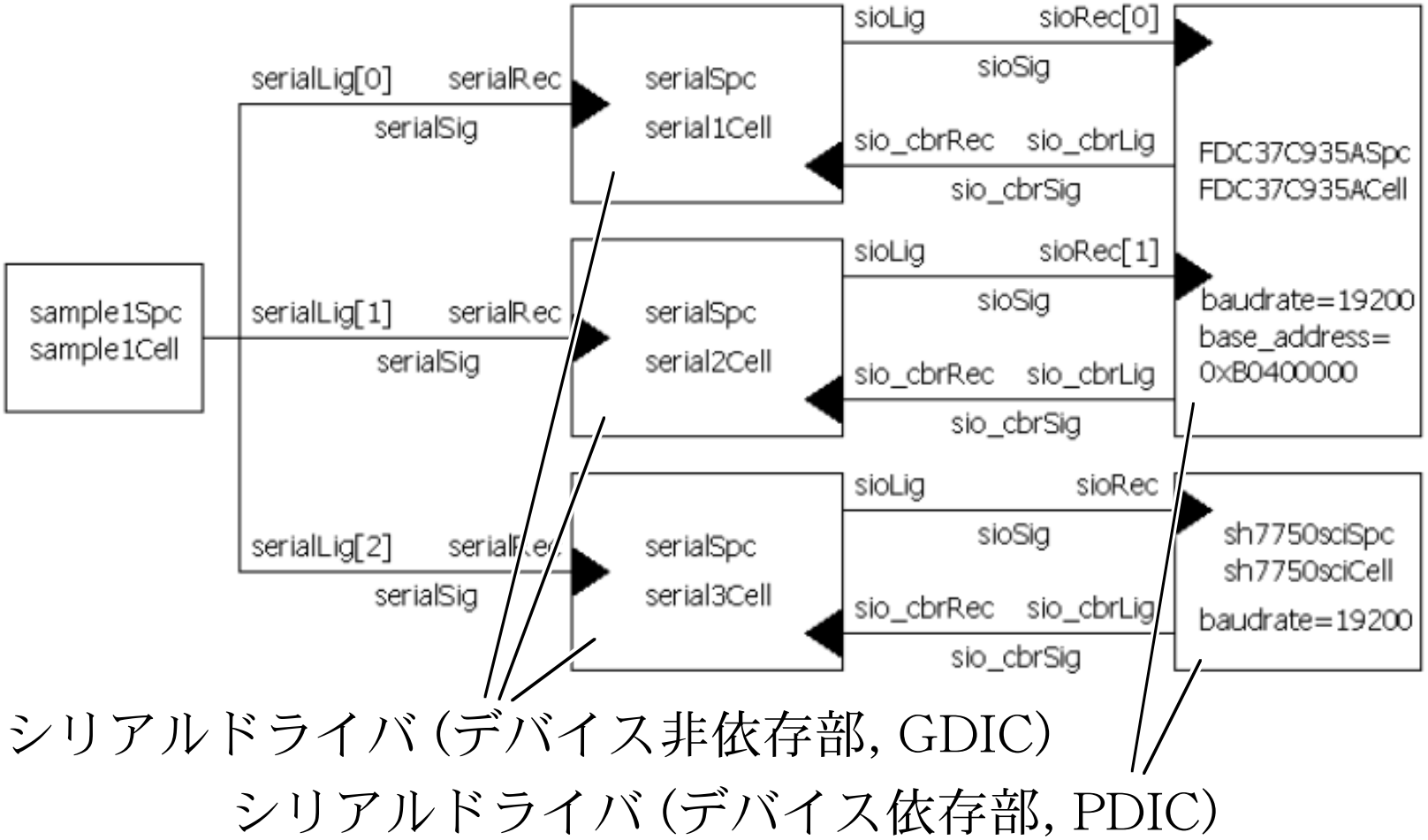
  → *μITRON仕様のような組み上げ型のRTOSには難しい状況*
- 複数のプロセッサを使用するシステムの増加
  - プロセッサ間通信を有効にサポートするプラットフォーム（分散フレームワークやマルチプロセッサOS）がなく，ソフトウェア開発が非効率

なぜ「組込みコンポーネント仕様」なのか？

- 汎用システム分野では，コンポーネント技術 (MS COM, JavaBeans など) が普及しているが，組込みシステム分野で広く使われている技術はない
- 既存のコンポーネント仕様を組込みシステムに適用するための問題点
  - オーバヘッドが大きい
  - コンポーネント間の結合を動的に行う
  - オブジェクト指向言語をベースにしている (組込みシステム分野では，オブジェクト指向言語が普及していない)
  - 組込みシステム分野で使われている多様なネットワーク (例えば，CANなどの制御用ネットワーク) に対応できない

組込みコンポーネント仕様の適用イメージ(1)

- JSPカーネルのシリアルドライバの構造

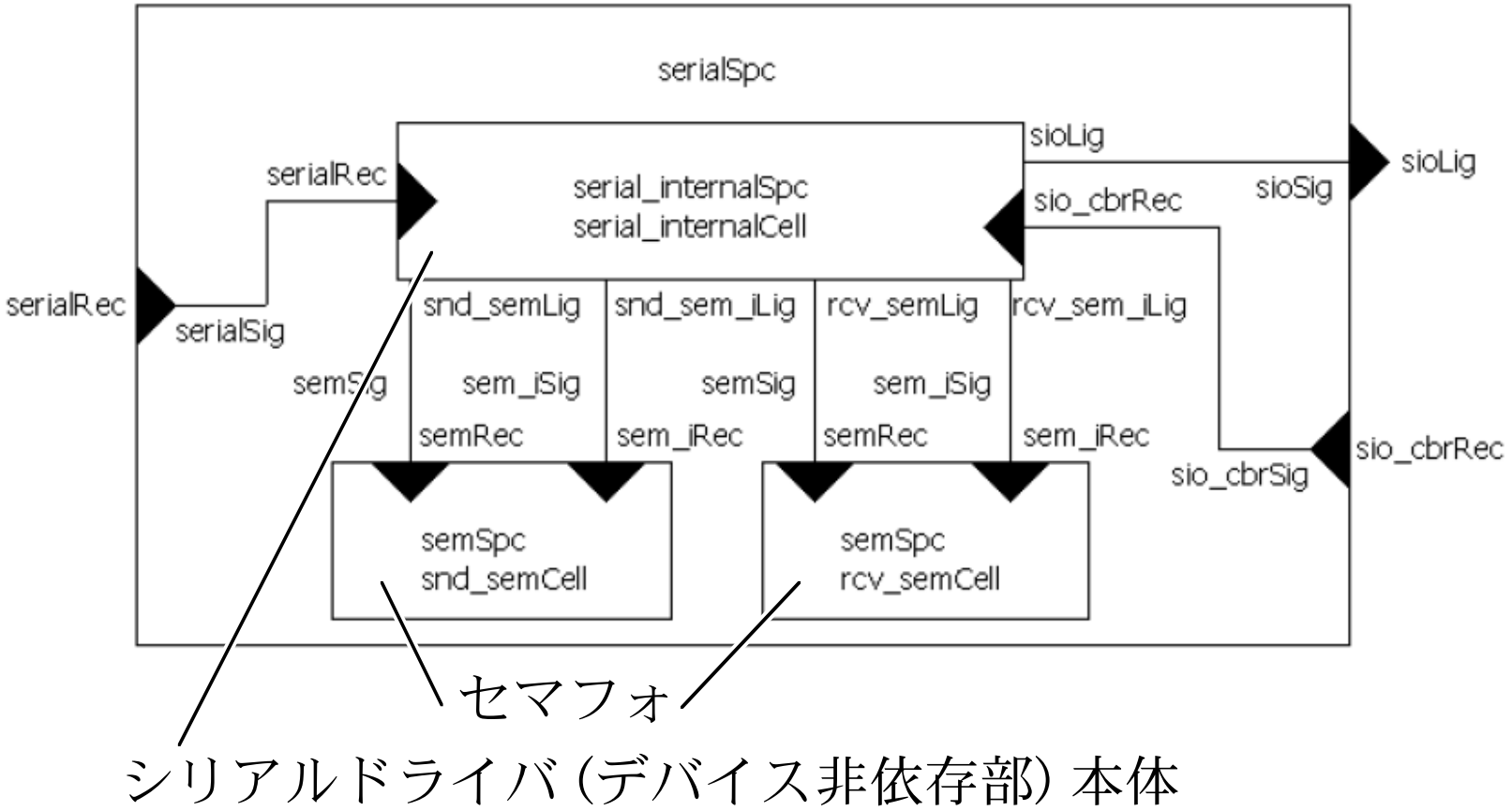


シリアルドライバ(デバイス非依存部, GDIC)

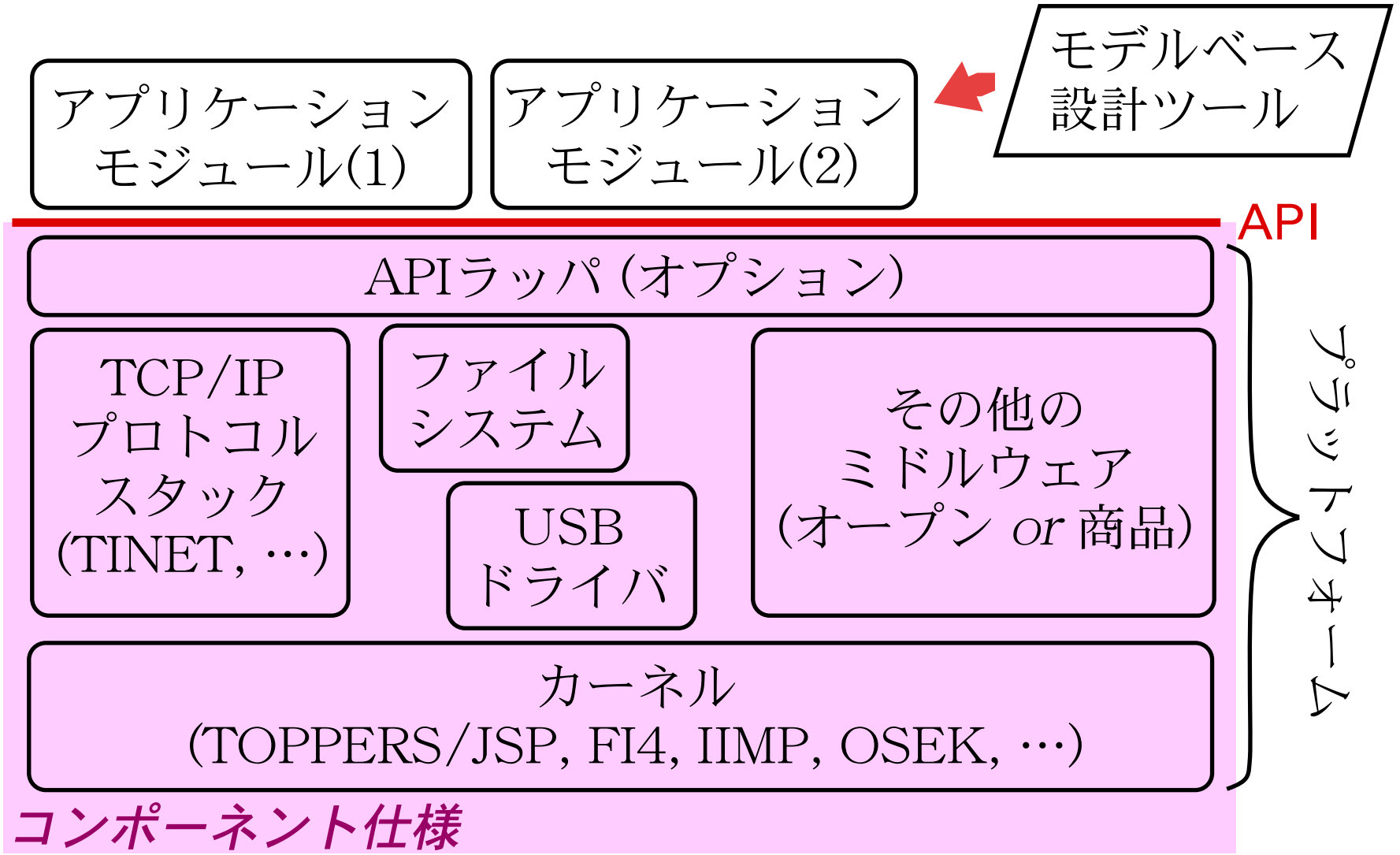
シリアルドライバ(デバイス依存部, PDIC)

組込みコンポーネント仕様の適用イメージ(2)

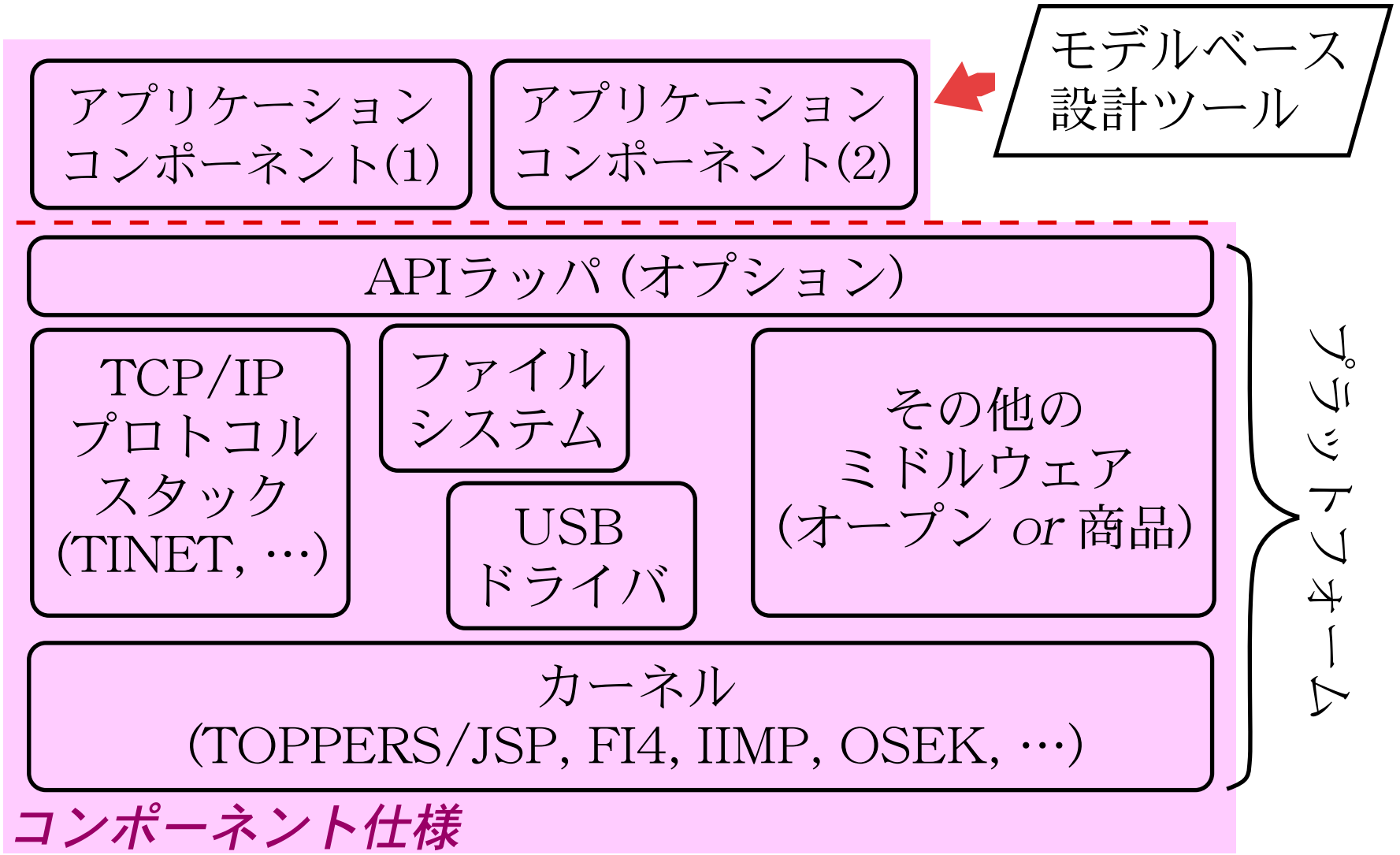
- JSPカーネルのシリアルドライバ(デバイス非依存部, 複合コンポーネント)の内部構造



組込みコンポーネント仕様の適用イメージ(3)



組込みコンポーネント仕様の適用イメージ(4)



TECSの特徴とアプローチ

- コンポーネント間の結合を静的に最適化する
  - コンポーネント間は静的に結合するのが基本
  - 状況に応じた最適なコンポーネント間インタフェースをツールにより生成（インタフェース生成）
  - オーバヘッドが小さく，粒度の小さいコンポーネントを扱える
- システム内のすべてのソフトウェアをコンポーネントとして扱う
  - プラットフォームも含めて構築が可能
  - 多様なプラットフォーム（プロセッサ, OS, ネットワーク）に対応可能
- ネットワークを超える遠隔呼出し（RPC）のためのコンポーネントをツールにより生成（RPC生成）

取組み事例３：消費エネルギー最適化

- JST CREST「情報システムの超低消費電力化を目指した技術革新と統合化技術」領域の採択テーマ
- 九州大学, 東芝との共同研究

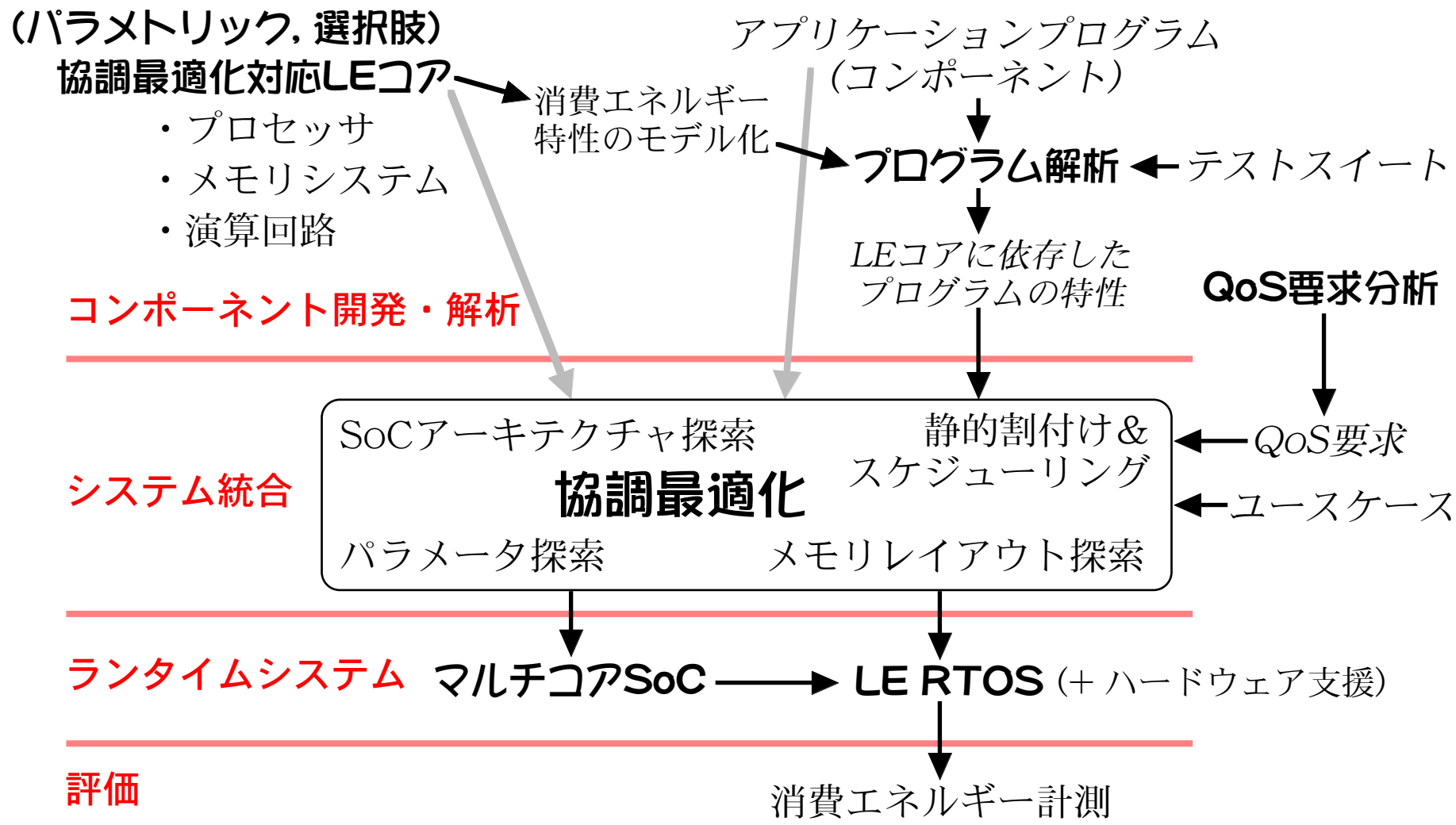
研究の目標

- ソフトウェアとハードウェアの協調により，アプリケーションに必要なQoS（性能, 計算精度, 信頼性）を保証しつつ，消費エネルギーを100分の1に低減する技術を開発

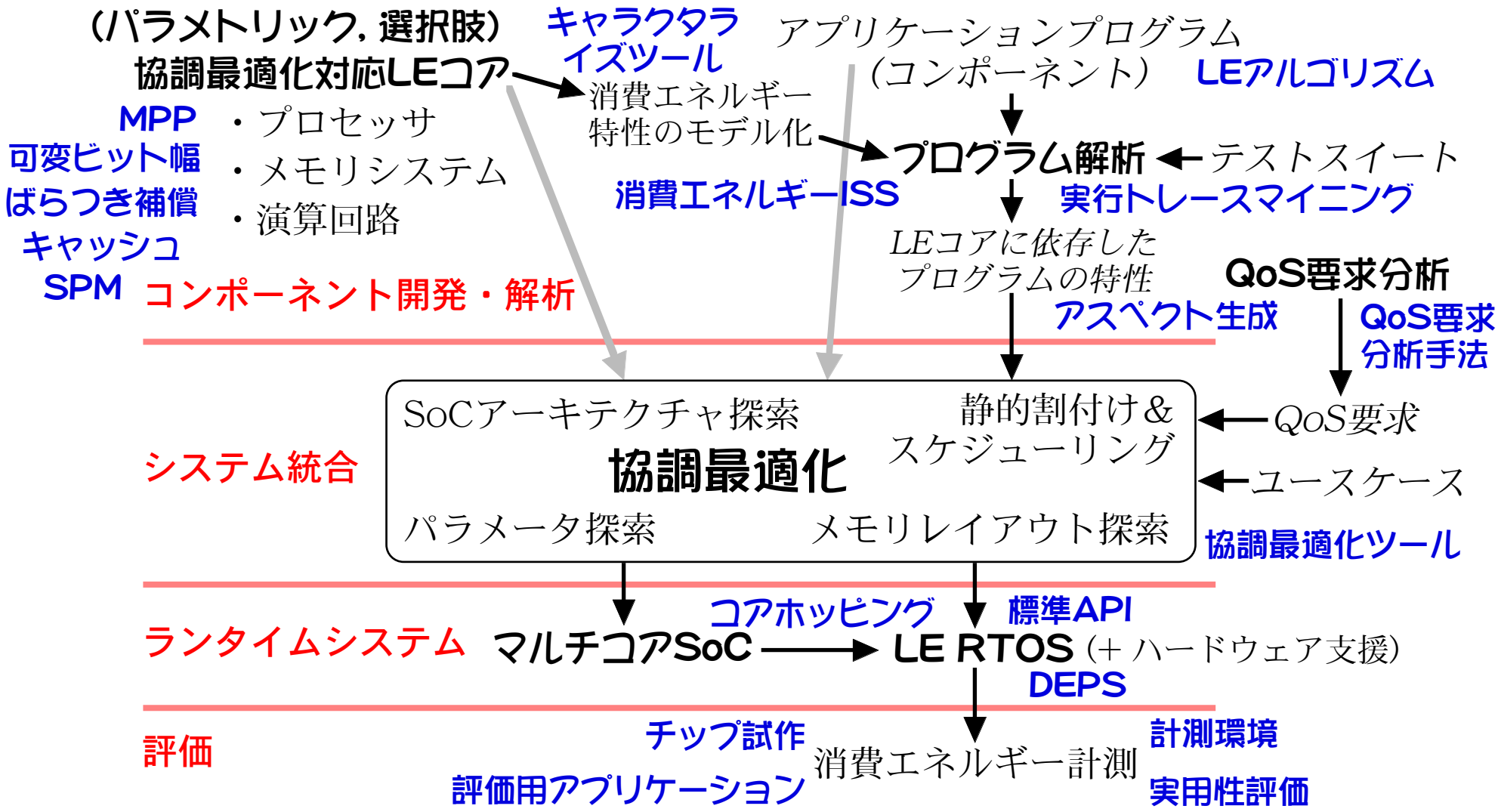
研究のアプローチ

- 組込みシステム設計技術を対象にするため，アプリケーションの性質を最大限に活用
- 広範囲な設計階層（ソフトウェア設計からハードウェアの回路設計まで）に跨る統合的・体系的な最適化
- 必要なQoSを保証しつつ消費エネルギーを最小化

研究プロジェクトの全体像



研究プロジェクトで取り組んでいる要素技術



研究中の要素技術例

マルチパフォーマンスマルチプロセッサ

- プロセッサの回路面積が2倍（消費電力も同じトレンド）になっても，性能は1.4倍程度にしかならない
- 性能の異なるプロセッサを1チップ上に集積し，必要十分な性能を持ったプロセッサを使う

DEPS (Dynamic Energy Performance Scaling)

- DVFS (Dynamic Voltage Frequency Scaling) の一般化
- 必要十分な性能を持ったハードウェアを活用するためのRTOSのスケジューリング機構

実行トレースマイニング

- プログラムの実行トレースから，プログラムの性質を自動抽出するための技術

メモリレイアウト最適化

おわりに

- 組込みシステム開発技術の今後10年のチャレンジと我々の取組み事例について紹介
- 現在の経済状況は，組込みシステム開発にも大きな変化をもたらす可能性がある
  - 適切なレベルの機能とコストのバランス
  - 開発効率化がますます重視される
- この時期に，次に向けての仕込みを行うことで，ピンチをチャンスに変えたい
  - 設計資産の棚卸し（リファクタリングの実施）
  - 人材育成
  - 地道な技術開発，改善活動

組込みシステム技術に関する高度な研究開発人材の養成

Hiroaki Takada

組込みシステム研究センター（NCES）

設立目的

☞ http://www.nces.is.nagoya-u.ac.jp/

- 組込みシステム分野の技術と人材に対する産業界からの要求にこたえるために，組込みシステム技術に関する研究・教育の拠点を，名古屋大学に形成
- 産業界が必要とする技術課題を分析・抽出し，大学における基礎研究に反映

活動領域（スコープ）

- 組込みシステムに関する以下の活動に，***産学連携***の枠組みで取り組む
  - 大学の持つ技術シーズを実現／実用化することを指向した研究（第二種基礎研究）
  - プロトタイプとなるソフトウェアの開発
  - 組込みシステム技術者の教育／人材育成

組込みソフトウェア技術者人材養成プログラム

プログラムの概要

- 社会人（主に企業の技術者）を対象に，組込みソフトウェア技術に関する実践的な教育プログラムを提供
- 科学技術振興調整費により2004年度より5年計画で実施
- 2～4日間程度の短期集中コース（座学および実習，または座学のみ）を年間15回程度開講

高い実績と評価

- これまで実施したいずれのコースも定員以上の申込み
- これまでの4年間に約1,000名が修了（のべ人数）
- 3年目の中間評価においてもA評価を獲得
- 企業からのアドバイザリ委員からも高く評価される

2008年度が最終年度

組込みシステム技術に関する高度な研究開発人材の養成

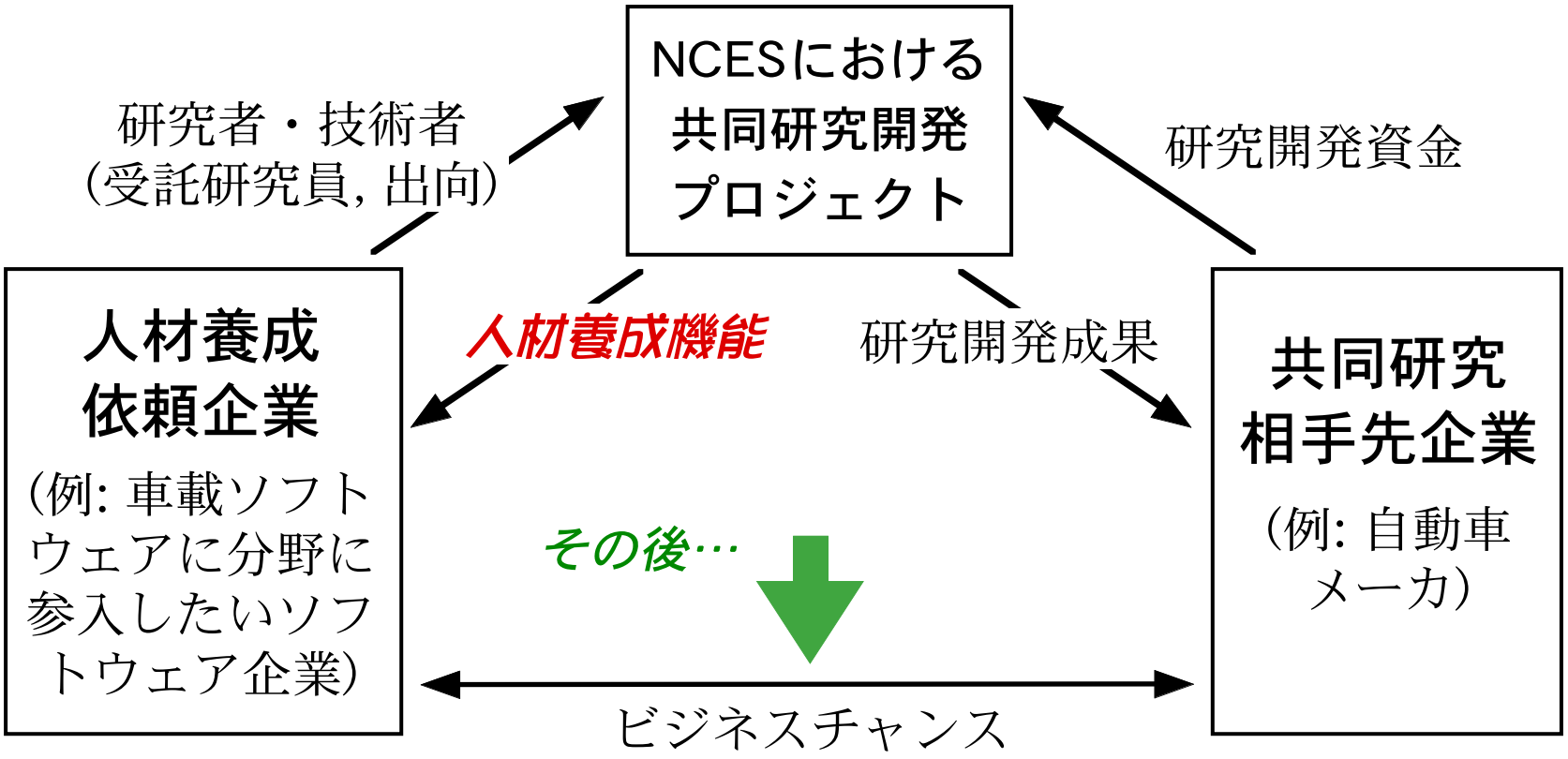
事業の趣旨

- NCESにおける研究開発プロジェクトを活用して，より高度な研究開発人材を養成

実施内容

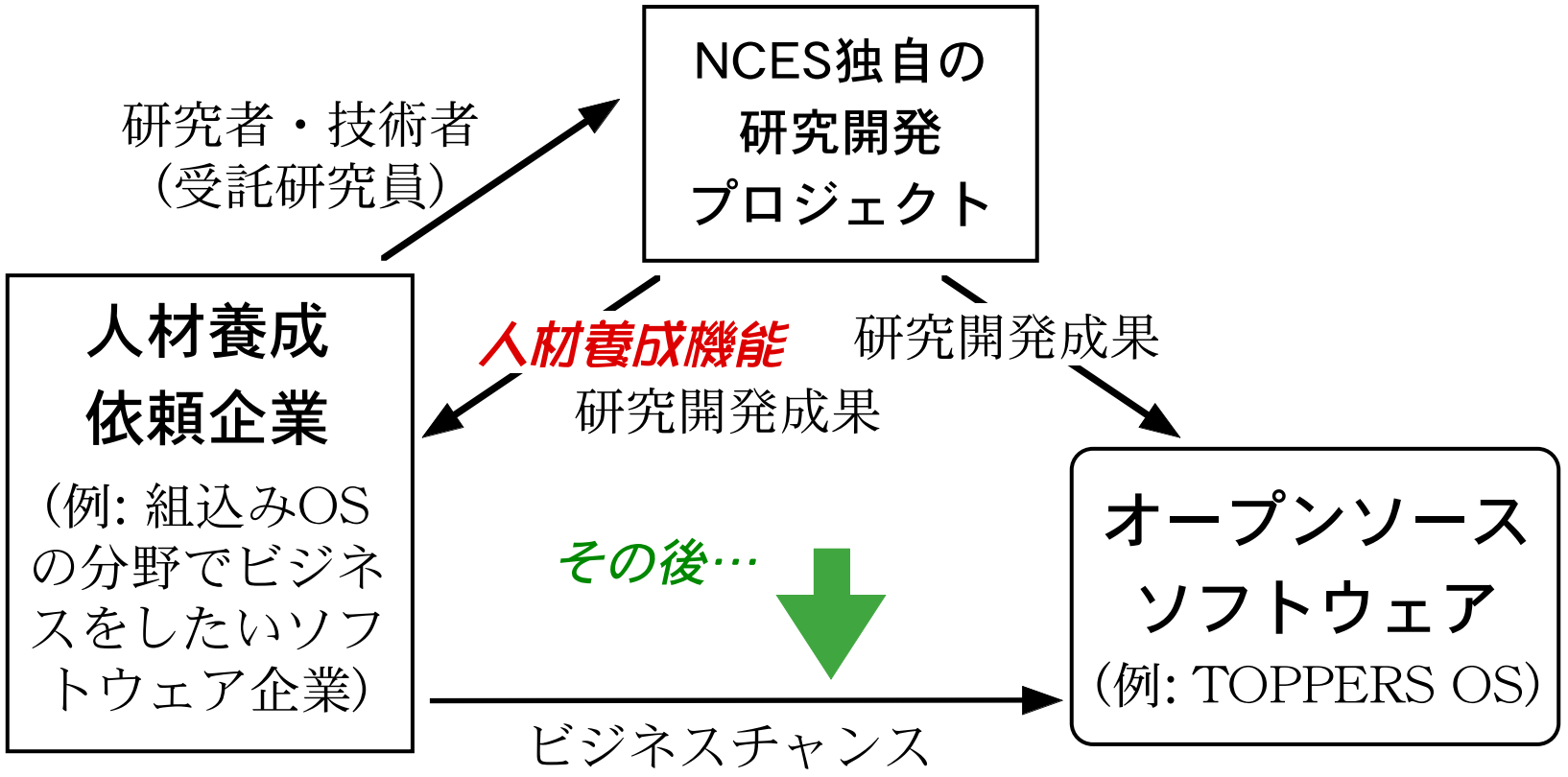
- 民間企業等からの養成対象者を，NCESにおける研究開発プロジェクトに参加させることで，研究要素を含む先端的な開発業務をリードできる高度な組込みシステム技術者（高度研究開発人材）を養成（5年で80名程度）
- 社会人と大学院生を対象とした公開講座の開講等（5年で1,000名程度）
- NEXCESSで開発してきた教材を継続的に改良・改版
- 技術者の養成カリキュラム等に関して関係機関と連携
- 他の教育機関や産業界に教材と人材養成ノウハウを提供

高度研究開発人材養成の実施形態例（1）



企業と共同の研究開発プロジェクトを活用した
人材養成スキームの例

高度研究開発人材養成の実施形態例（2）



大学独自の研究開発プロジェクトを活用した
人材養成スキームの例

マルチプロセッサ向けRTOSに関するコンソーシアム

プロジェクトの位置付け

- 名古屋大学独自に実施してきたマルチプロセッサ向けRTOSの研究開発プロジェクトに企業からの参加を募集
- このプロジェクトを，高度な研究開発人材の育成事業に活用（人材育成スキームの例 (2) に該当）

研究開発の内容

- 名古屋大学で開発を進めているTOPPERS/FMPカーネルをコアとする
- マルチプロセッサ向けRTOSに関する以下のようなテーマに関して研究開発
  - 検証手法，性能評価手法（検証スイートなど）
  - 開発環境（トレースログの可視化ツールなど）
  - 仕様策定・実装技術（動的負荷分散手法など）