大学でシステムLSI（アーキテクチャ）をどう教育するか？

村上, 和彰
九州大学大学院システム情報科学研究院 | 九州大学情報基盤センター

出版情報：SLRC プレゼンテーション, 2005-07-19. 九州大学システムLSI研究センター
バージョン：
権利関係：

大学でシステムLSI（アーキテクチャ）をどう教育するか？

村上和彰
九州大学　情報基盤センター長
murakami@i.kyushu-u.ac.jp

本発表の概要

- 企業における「システム・アーキテクトの不在，不足，渇望」が言われて久しい。このような状況に対して，我々大学人はどう応えるか？本課題に対して，国内外の実例を交えて議論を行いたい。

のはずだったが…

本発表の目的

- 大学／大学院におけるシステムLSI関連の教育カリキュラムの現状をサーベイしたり、あるいは、あるべき姿を提案するわけではない
- １大学院の１研究室（九州大学大学院システム情報科学研究院・村上研究室）における昨今のシステムLSI関連研究活動を通して、大学院生に対してシステムLSI（特に、アーキテクチャ、設計）を教育することの意義を再確認する

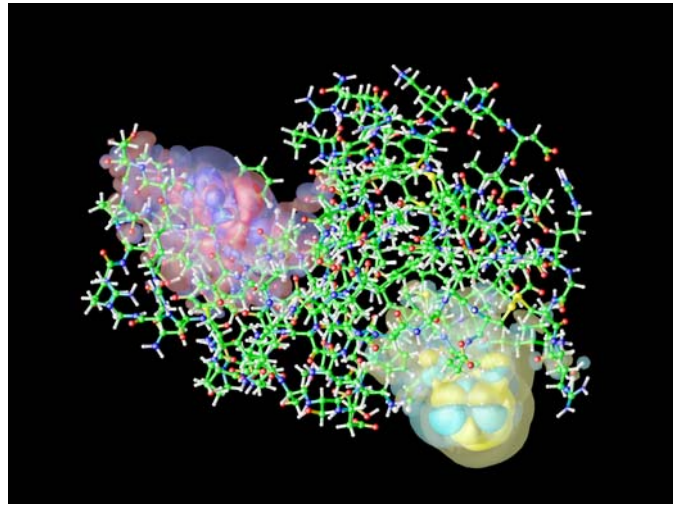
背景

- 企業側から大学側に要求する人材像の変化
  - 以前:「教育は二の次で良いから、とにかく(大学入学時の成績が)優秀な学生を！」
  - 現在:「システムの上から下まで分かるシステム・アーキテクトを！」
- 大学側(私個人)からの反論
  - システムアーキテクトは、本来は会社内で育成すべき！
  - とは言え、大学の使命として何らかの答は出すべき！

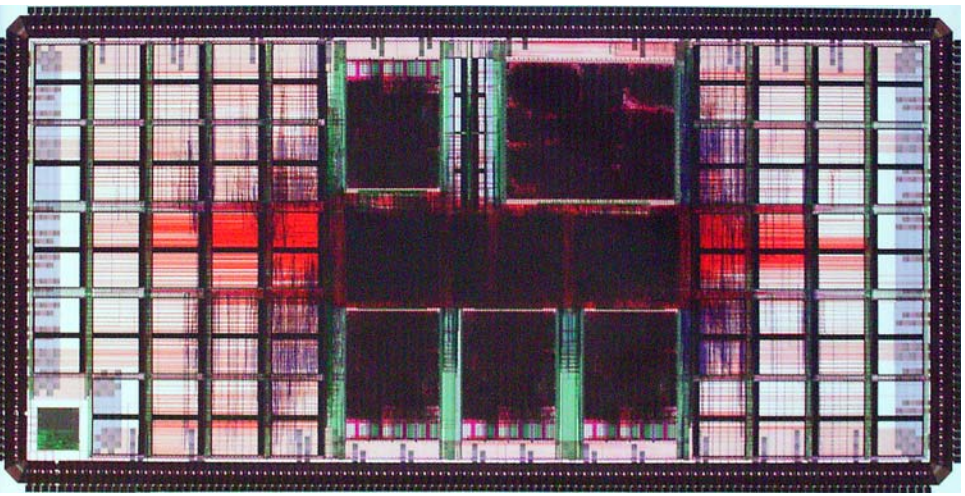
主張

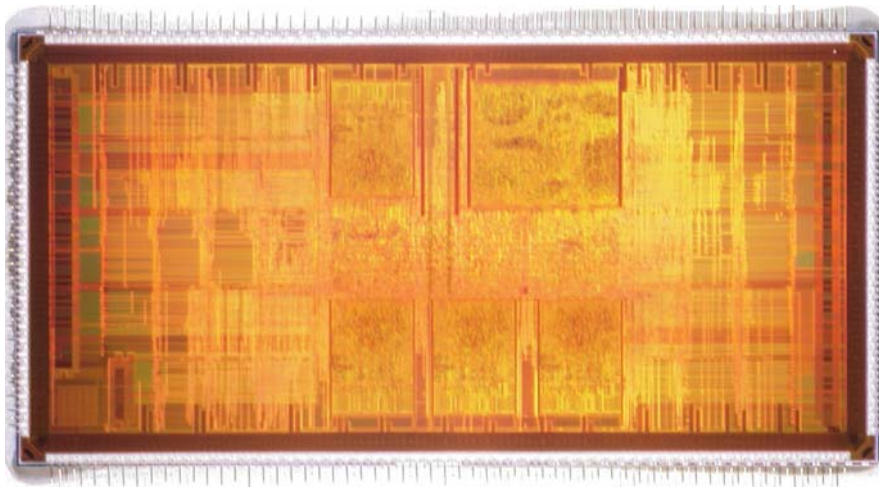
- 大学の研究室での標準的な3年間（卒業研究から修士課程修了まで）で「システムLSIの上から下まで」まで一通り「実践」させるのは困難
  - ６年間（博士後期課程修了まで）なら可能だが・・・
- 「設計実践」も重要だが、問題や技術の「分析力」、「分類能力」、「体系化／理論化能力」を習得することが大切
  - 「設計実践」は入社した後のOJTでも可能
  - 「分析」、「分類」、「体系化」、「理論化」はいわば大学の使命

Eric:二電子積分計算専用LSI

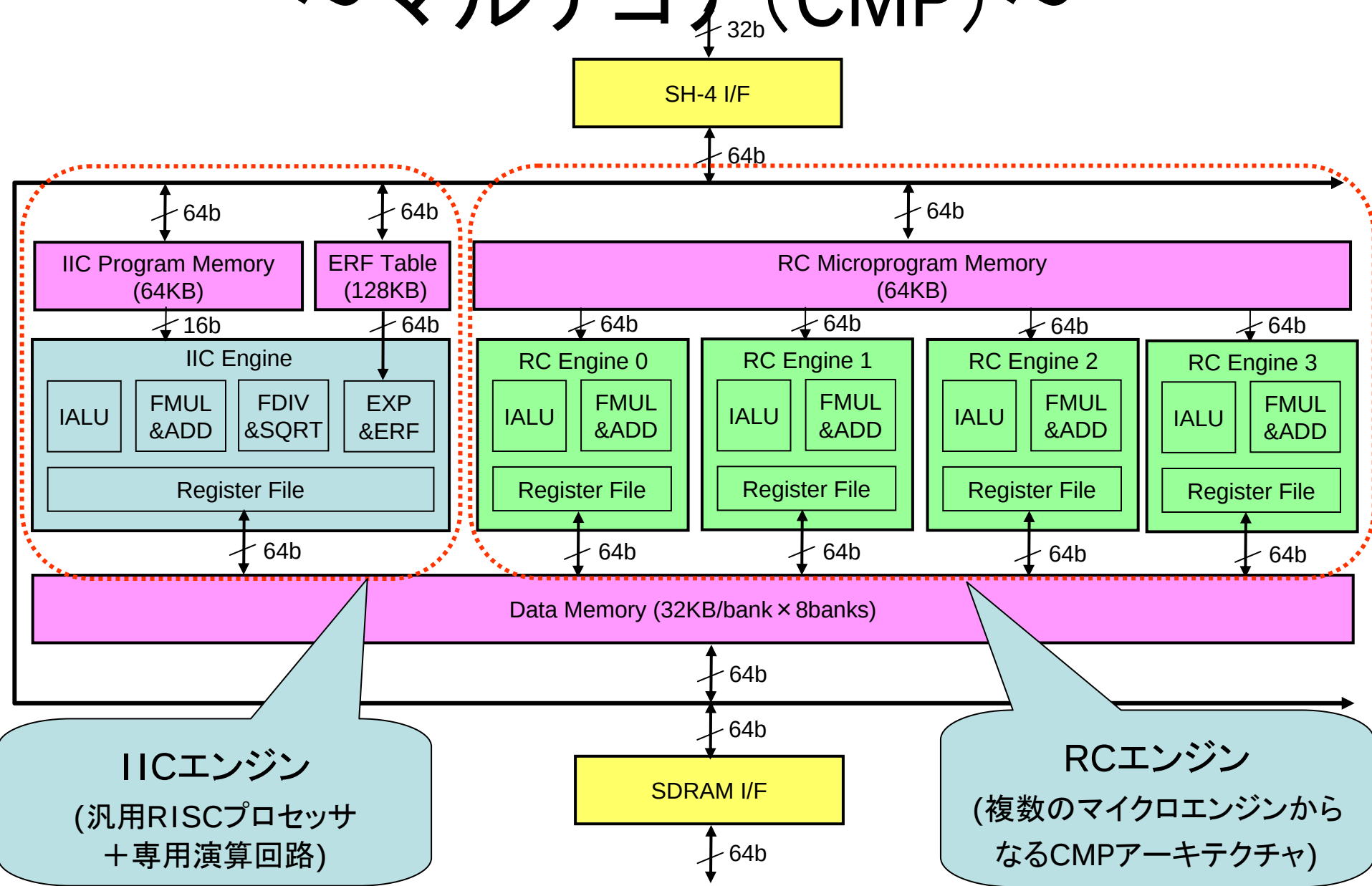


- 仕様
  - プロセス:TSMC 0.13um, 6層, Cu配線
  - チップサイズ:5 x 10mm^2
  - 論理ゲート:4M
  - メモリ:704KB
  - パッケージ:セラミックPGA 257PIN, 50.8mm□
- 動作クロック周波数
  - 200MHz
- 消費電力
  - 2.1W

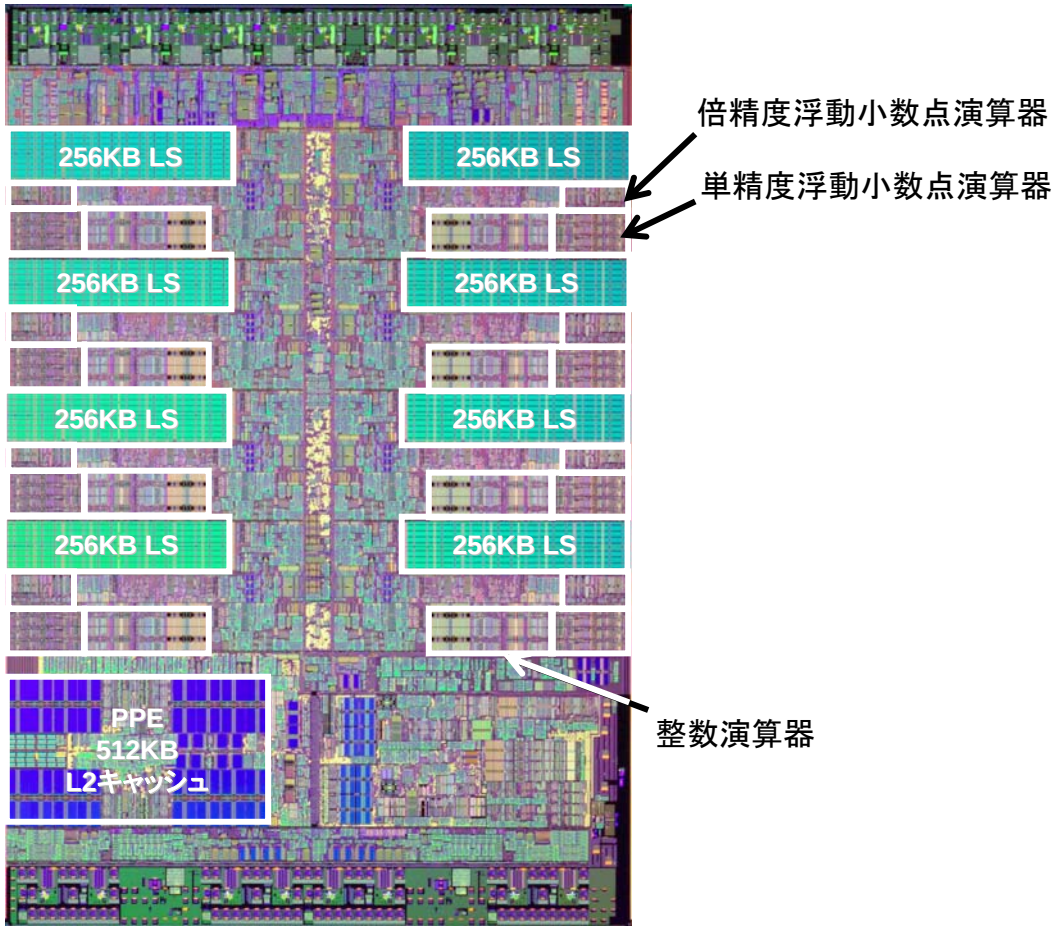




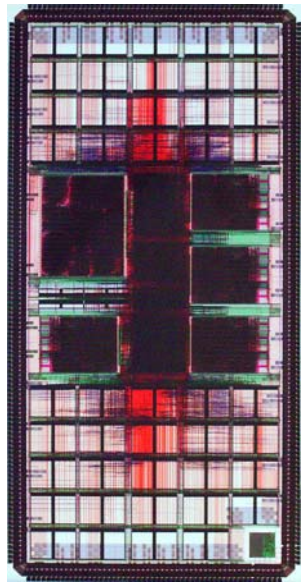
Eric
～マルチコア（CMP）～



マルチコア（CMP）アーキテクチャ ～CellとEric～



Cell (221mm^2 @90nm)



Eric (50mm^2 @130nm)

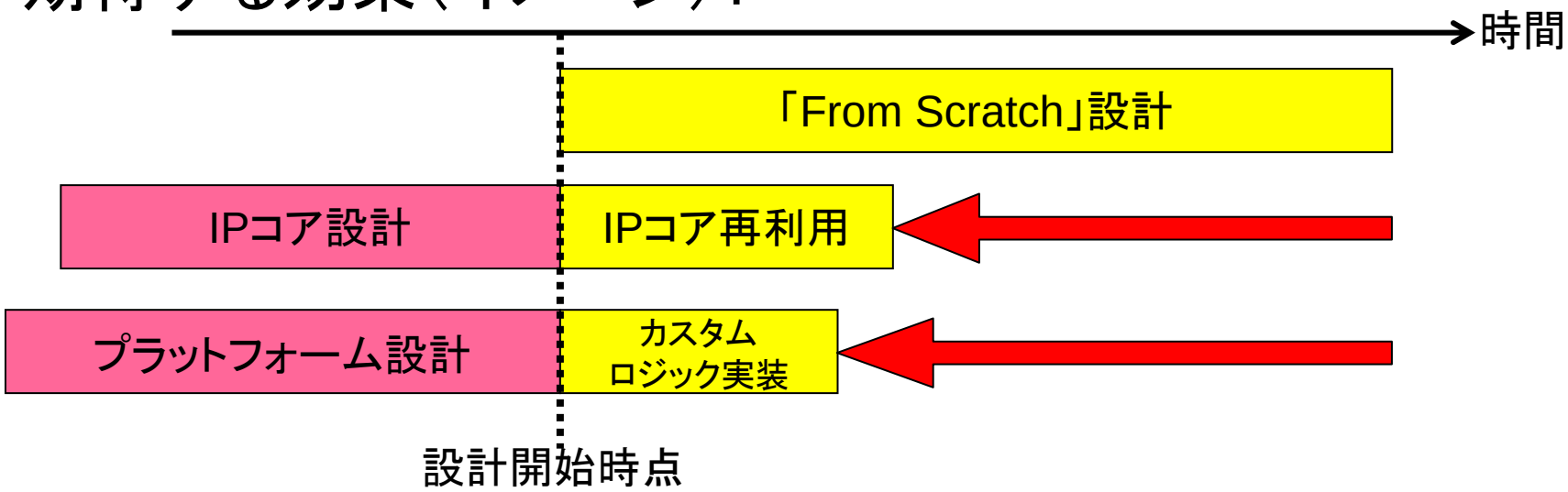
写真：SONY提供

発表の内容

- ケーススタディとして、システムLSI（SoC）の「プラットフォーム・アーキテクチャ」を題材に、その分類、体系化を研究室の学生に実践させたので、その結果を紹介する
- さらに、その結果は後進の学生の教育にも応用可！

プラットフォームとは？

- 異なる「カスタム・ロジック」を実現するのに利用可能な，「カスタム・ロジック」の違いに拠らない「共通の土台」
  - アナロジー：自動車の共通車台
- プラットフォーム・ベース設計：上記プラットフォーム上で「カスタム・ロジック」を設計する手法
  - 他の選択肢：「From Scratch」設計，IPコア・ベース設計
- 期待する効果（イメージ）：



カスタム・ロジックとは？

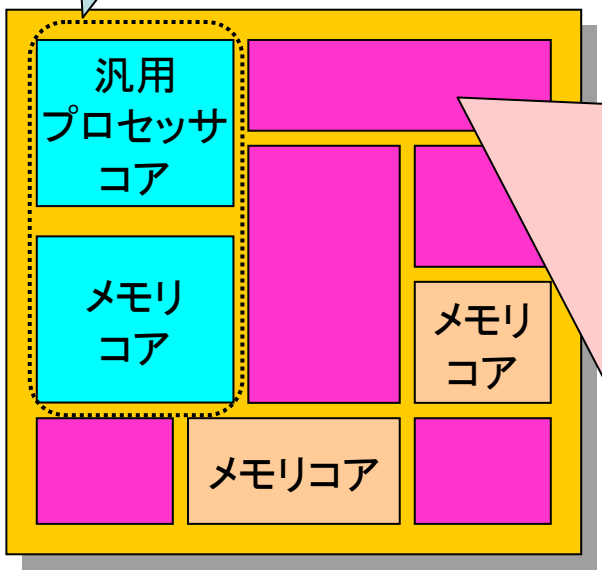
- システムを構成する各種機能のうち，「汎用プロセッサ」上では性能上の理由で実現しない／出来ない機能
  - 反対語：汎用機能，汎用ロジック
  - 同義語：専用ロジック
  - 類似語：専用ハードウェア，ハードウェア・ロジック
- 例：MPEG4の場合
  - 動き予測（ME），動き補償（MC）
  - DCT（離散コサイン変換），IDCT（逆変換）
- 専用機能の存在が「SoCがSoCたる」所以

プラットフォーム・ベース設計 vs. 他の設計手法

- 「From Scratch」
  - 毎回，ハードウェアの新規設計を実施
- IPコア・ベース
  - 既存の「ハードウェア設計（IPコア）」を再利用
  - SoC設計上の最重要課題
    - ①最適なIPコアの選択
    - ②同IPコアの特定半導体テクノロジ上での実装
  - IPコアの種類
    - ソフトIPコア
    - ハードIPコア
- プラットフォーム・ベース
  - 「プラットフォーム」上でカスタム・ロジックをハードウェアまたはソフトウェアとして実現
  - SoC設計上の最重要課題
    - ①最適なプラットフォームの選択
    - ②同プラットフォームの特定半導体テクノロジ上での実装
    - ③同プラットフォーム上でのカスタム・ロジックの実現
  - プラットフォームの種類
    - プロセッサ
    - コンフィギュラブル・プロセッサ
    - リコンフィギュラブル・プロセッサ
    - リコンフィギュラブル・ハードウェア

SoC設計への諸アプローチ：現象

汎用機能の実現法
- 汎用プロセッサ＋ソフトウェア

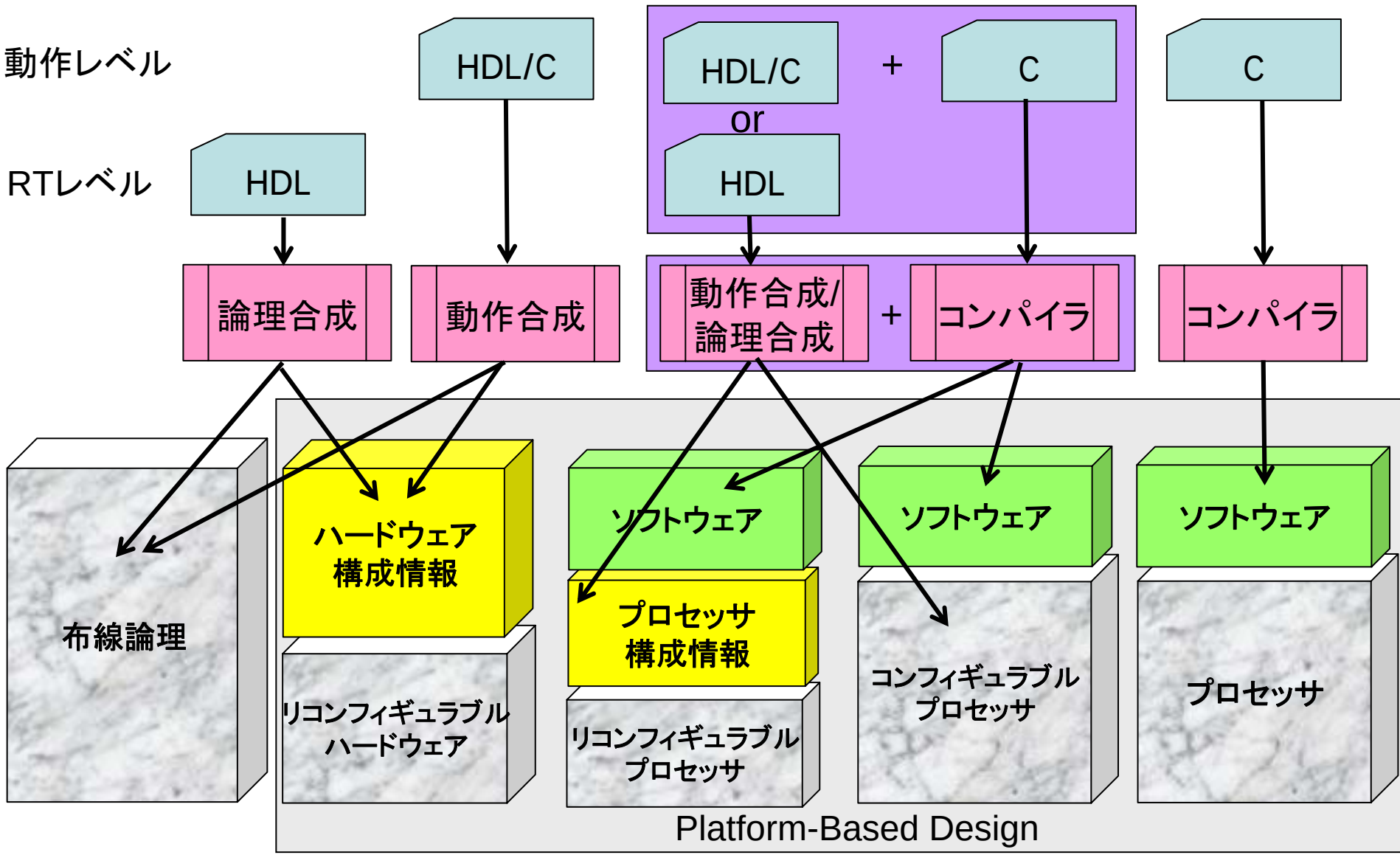


カスタム・ロジック（専用機能）の実現法
- 「From Scratch」
  - 新規設計のハードウェア
- IPコア・ベース
  - 既存設計のハードウェア
- プラットフォーム・ベース
  - プロセッサ＋ソフトウェア
    - SONY/Toshiba/IBM Cell
    - QuickSilver ACM
  - コンフィギュラブル・プロセッサ＋ソフトウェア
    - Tensilica Xtensa
    - PDI VUPU
  - リコンフィギュラブル・プロセッサ＋ソフトウェア
    - IP Flex DAP/DNA
    - Stretch
    - CLUSS Redefis
  - リコンフィギュラブル・ハードウェア
    - FPGA (Xilinx, Altera, etc.)
    - NEC DRP

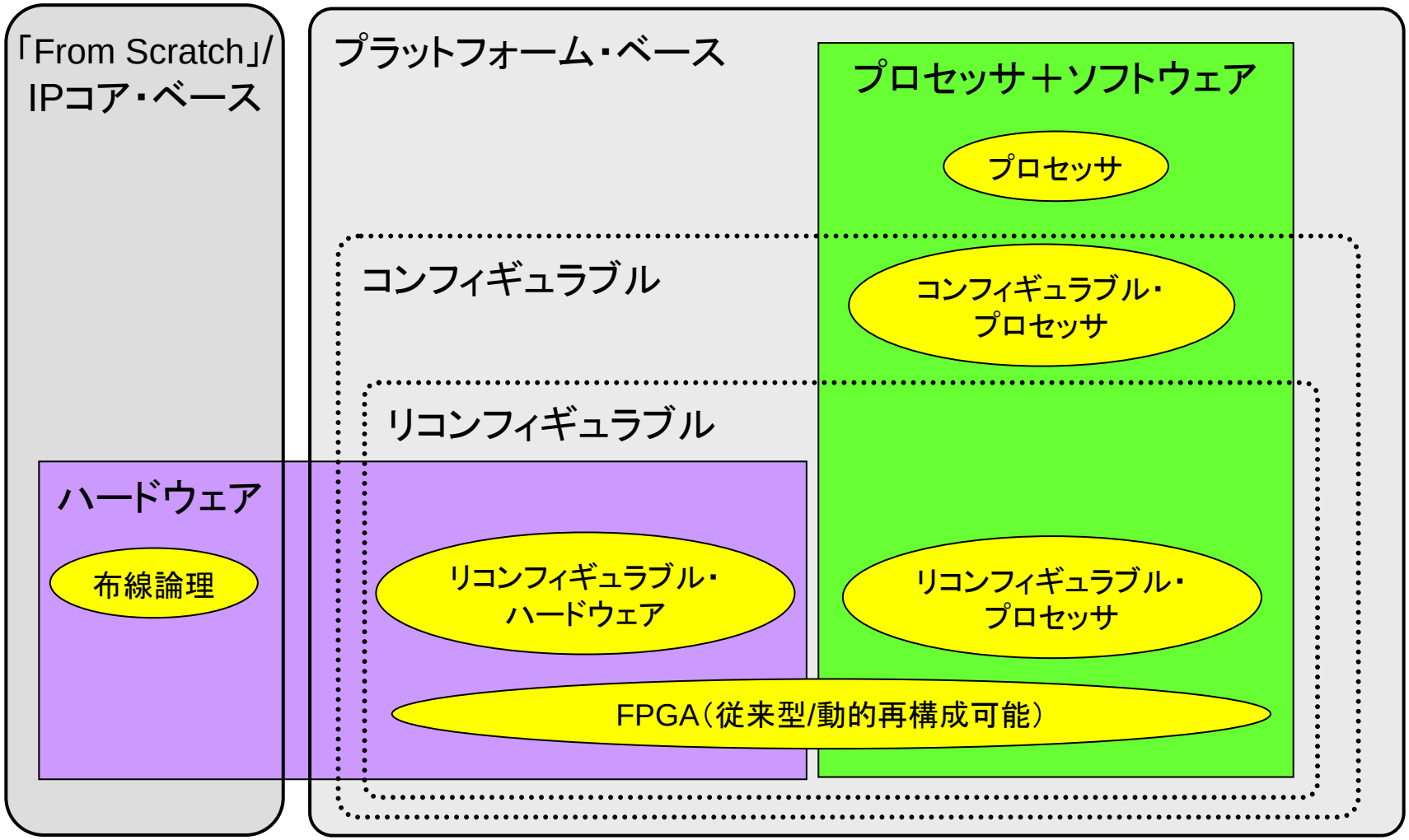
リファレンス・リスト

- プロセッサ＋ソフトウェア
  - 従来型の組込みプロセッサ/DSP
  - 均質型マルチプロセッサ
    - SONY/Toshiba/IBM Cell ➔
  - 非均質型マルチプロセッサ
    - QuickSilver ACM ➔ http://www.quicksilvertech.com/
- コンフィギュラブル・プロセッサ＋ソフトウェア
    - Tensilica Xtensa ➔ http://www.tensilica.com/
    - PDI VUPU ➔ http://www.pdi.co.jp/
- リコンフィギュラブル・プロセッサ＋ソフトウェア
    - IP Flex DAP/DNA ➔ http://www.ipflex.com/
    - Stretch ➔ http://www.stretchinc.com/
    - CLUSS Redefis ➔ http://www.fleets.jp/
- リコンフィギュラブル・プロセッサ
    - NEC DRP ➔ http://www.necel.com/ja/techhighlights/drp/

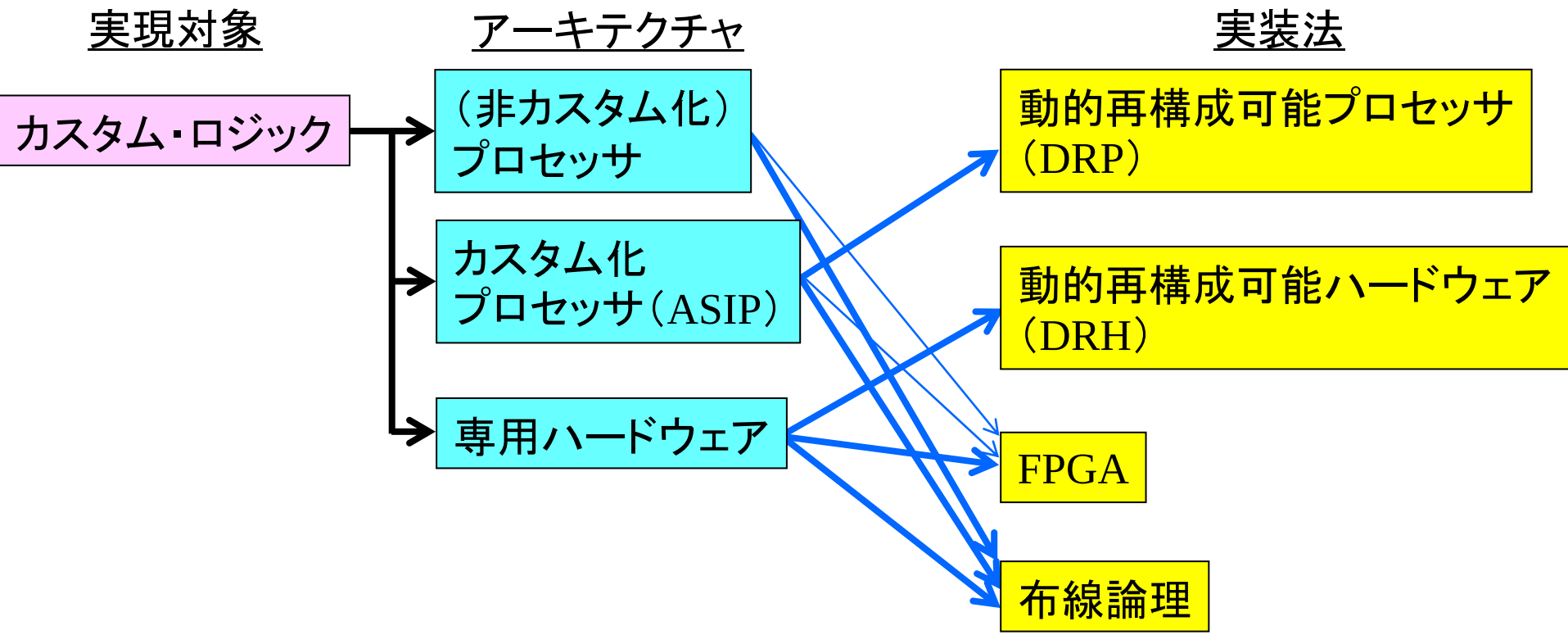
SoC設計への諸アプローチ：現象



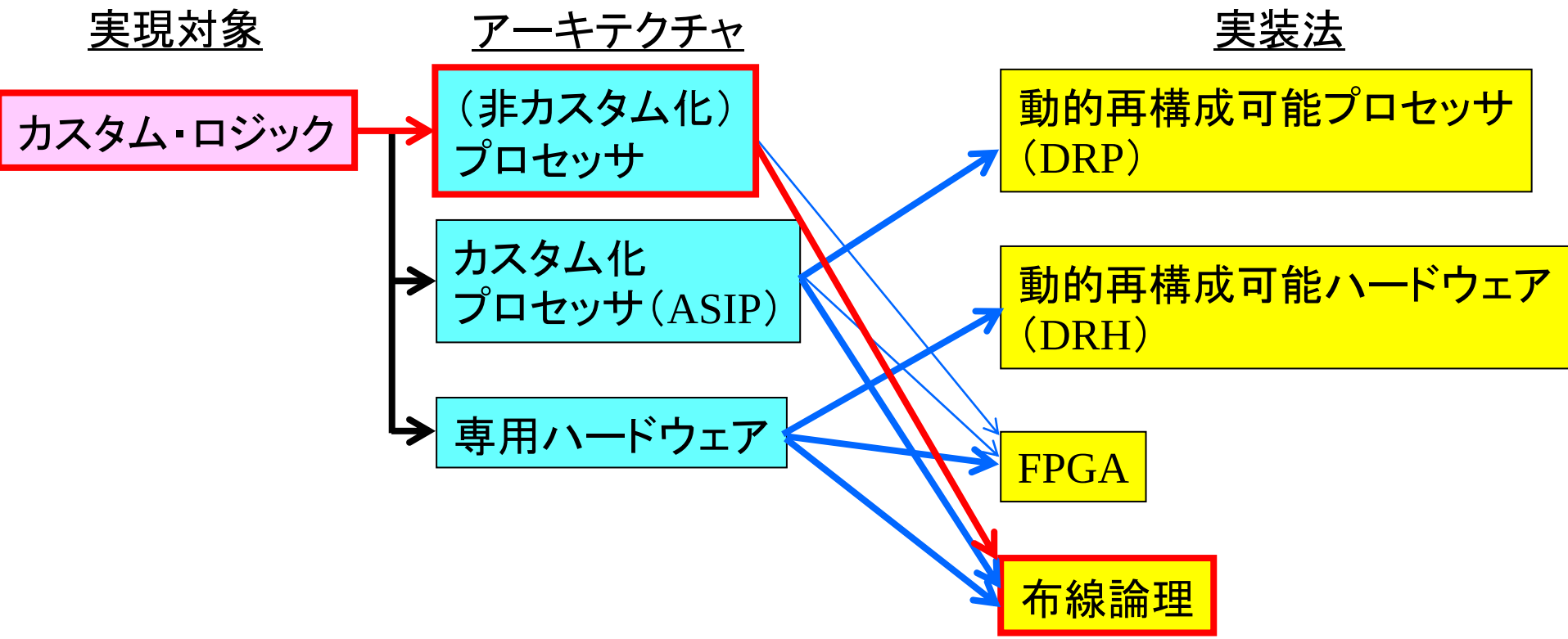
SoC設計への諸アプローチ
～1分類法～



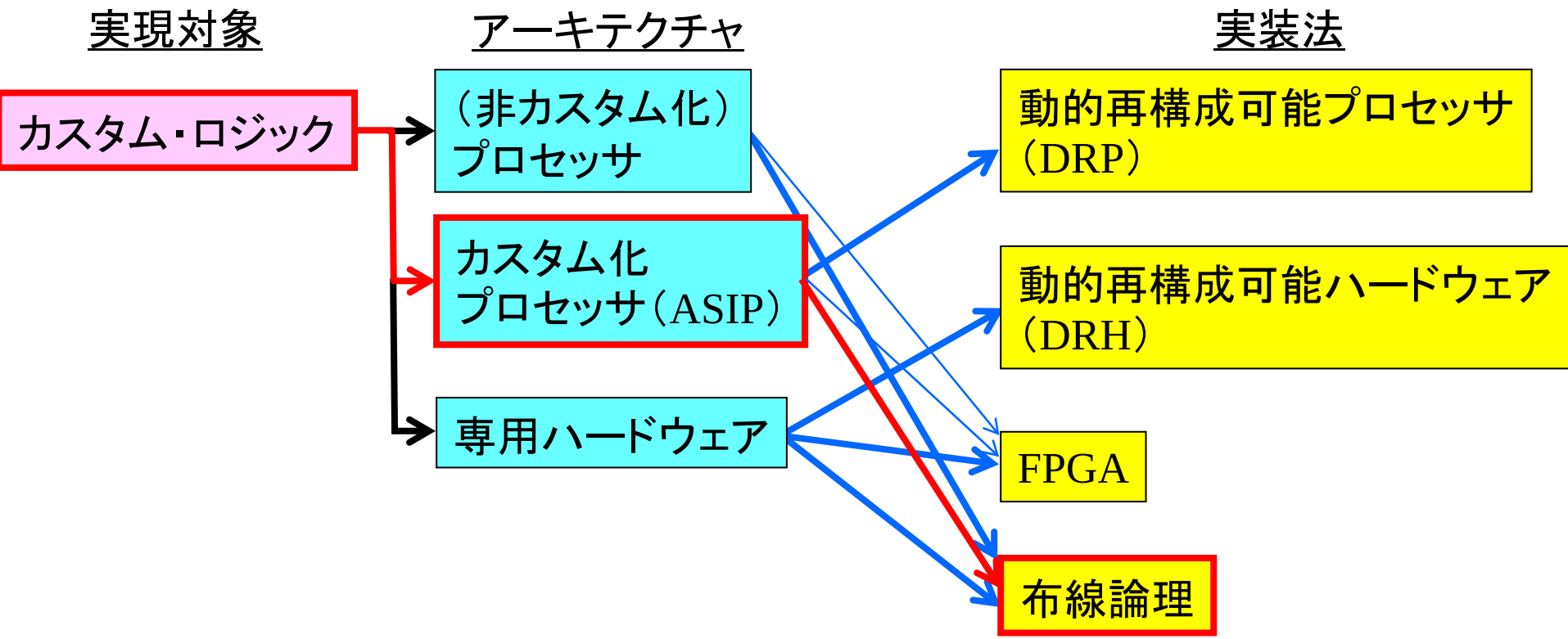
カスタム・ロジックの実現方法



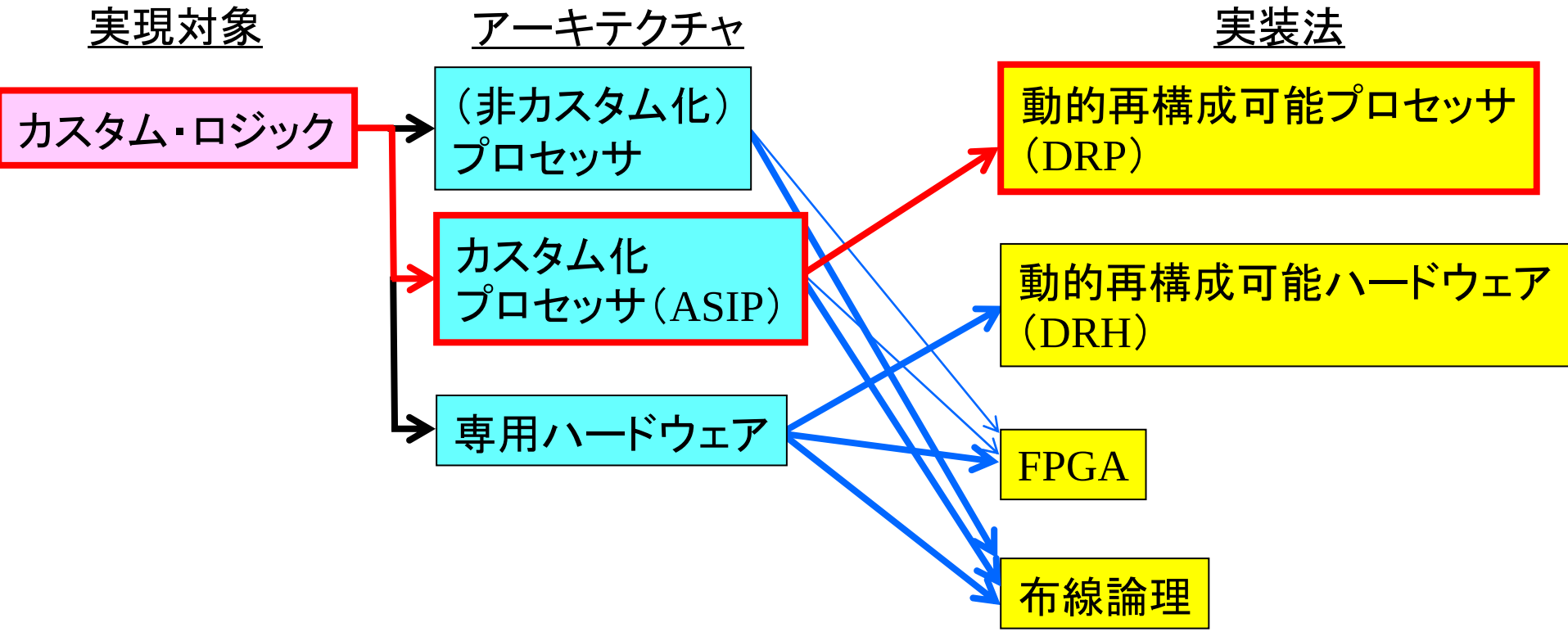
カスタム・ロジックの実現方法
～プロセッサ＋ソフトウェア～



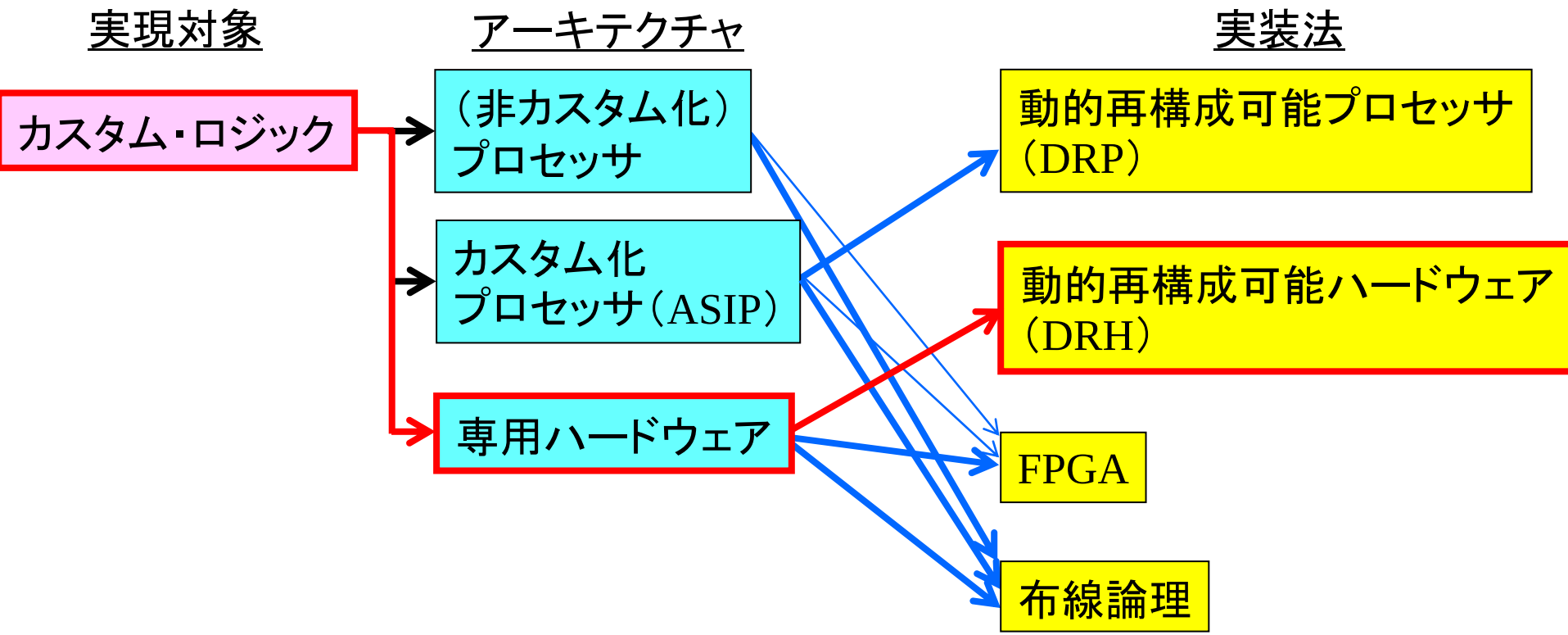
カスタム・ロジックの実現方法
～コンフィギュラブル・プロセッサ＋ソフトウェア～



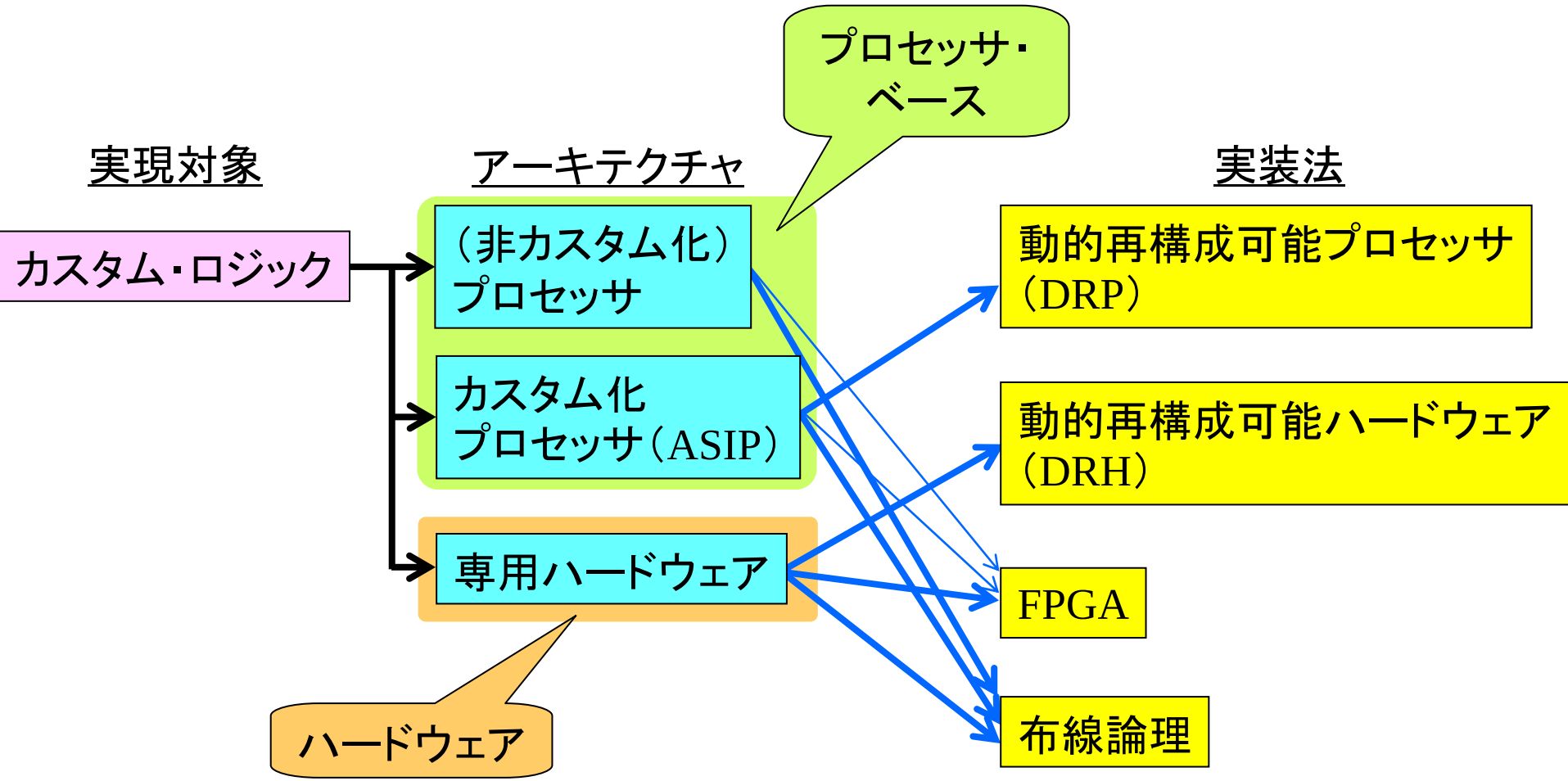
カスタム・ロジックの実現方法
～リコンフィギュラブル・プロセッサ＋ソフトウェア～



カスタム・ロジックの実現方法
～リコンフィギュラブル・ハードウェア～

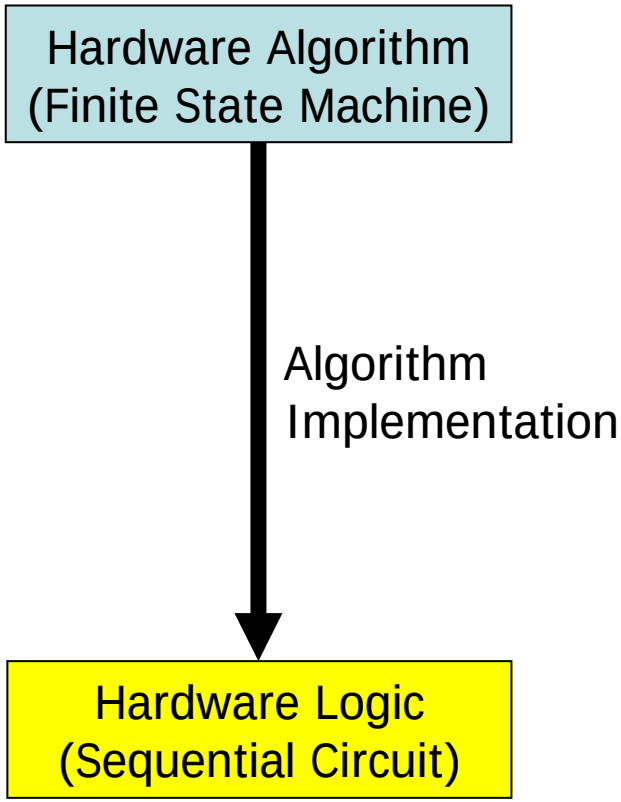


ハードウェア対プロセッサ・ベース

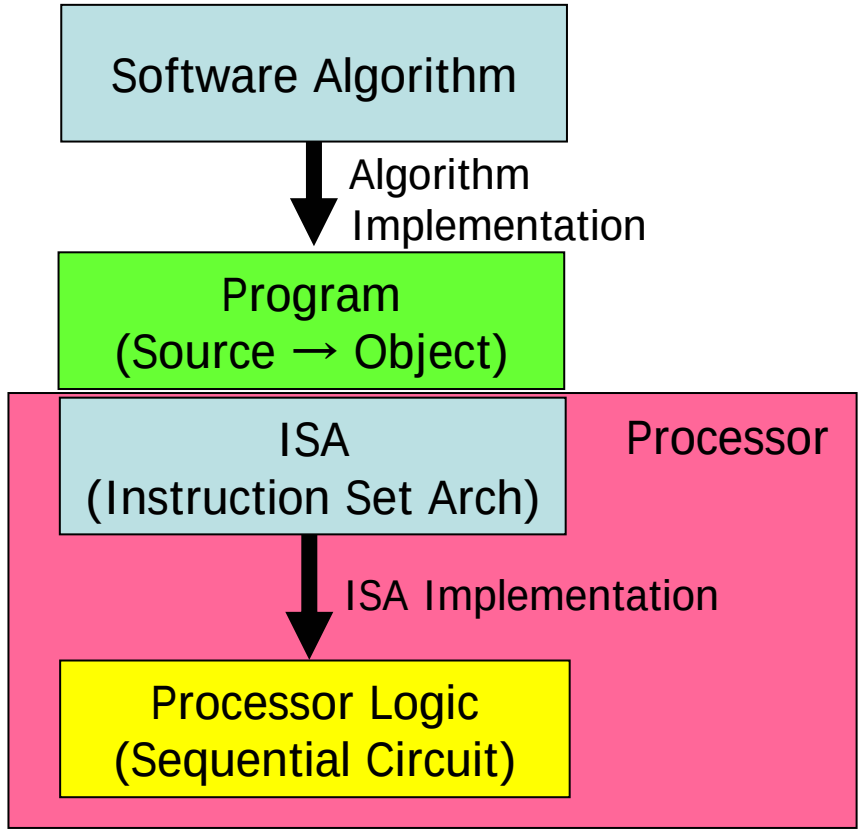


ハードウェア対プロセッサ・ベース

- ハードウェア

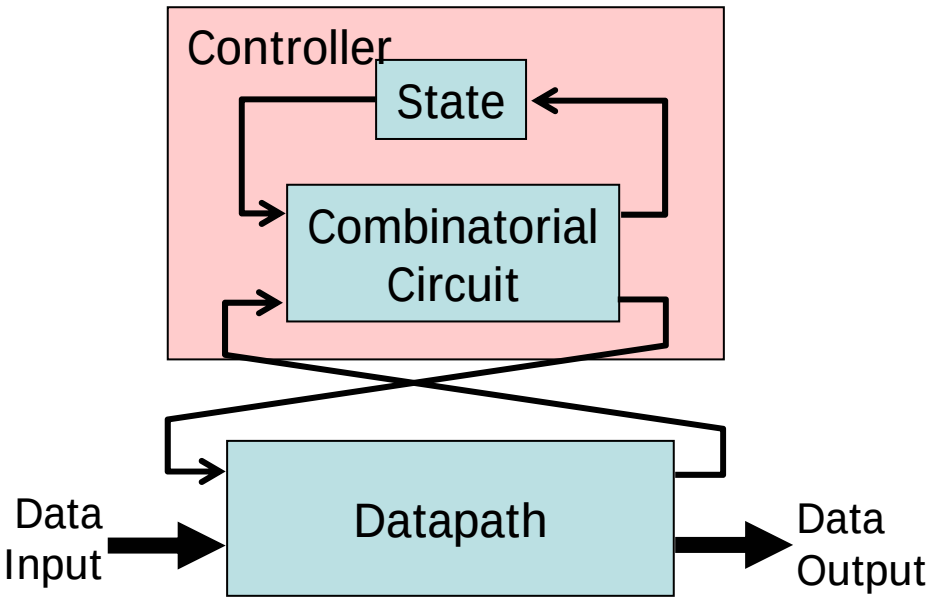


- プロセッサ＋ソフトウェア

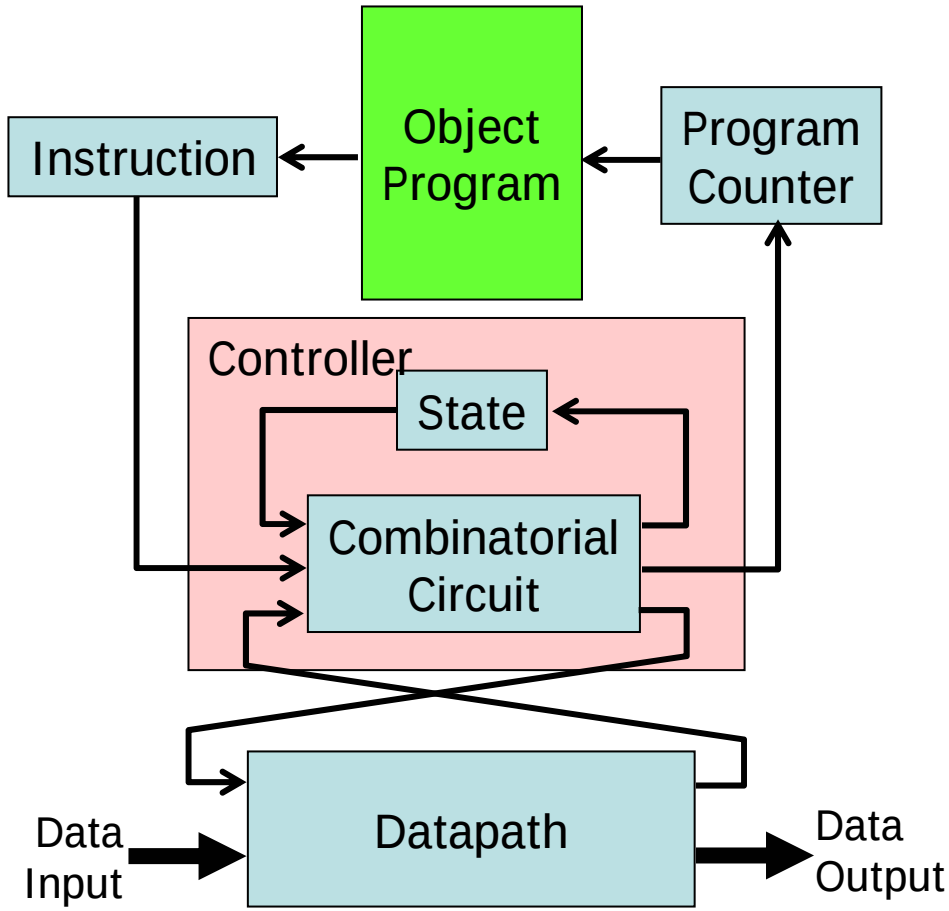


ハードウェア対プロセッサ・ベース

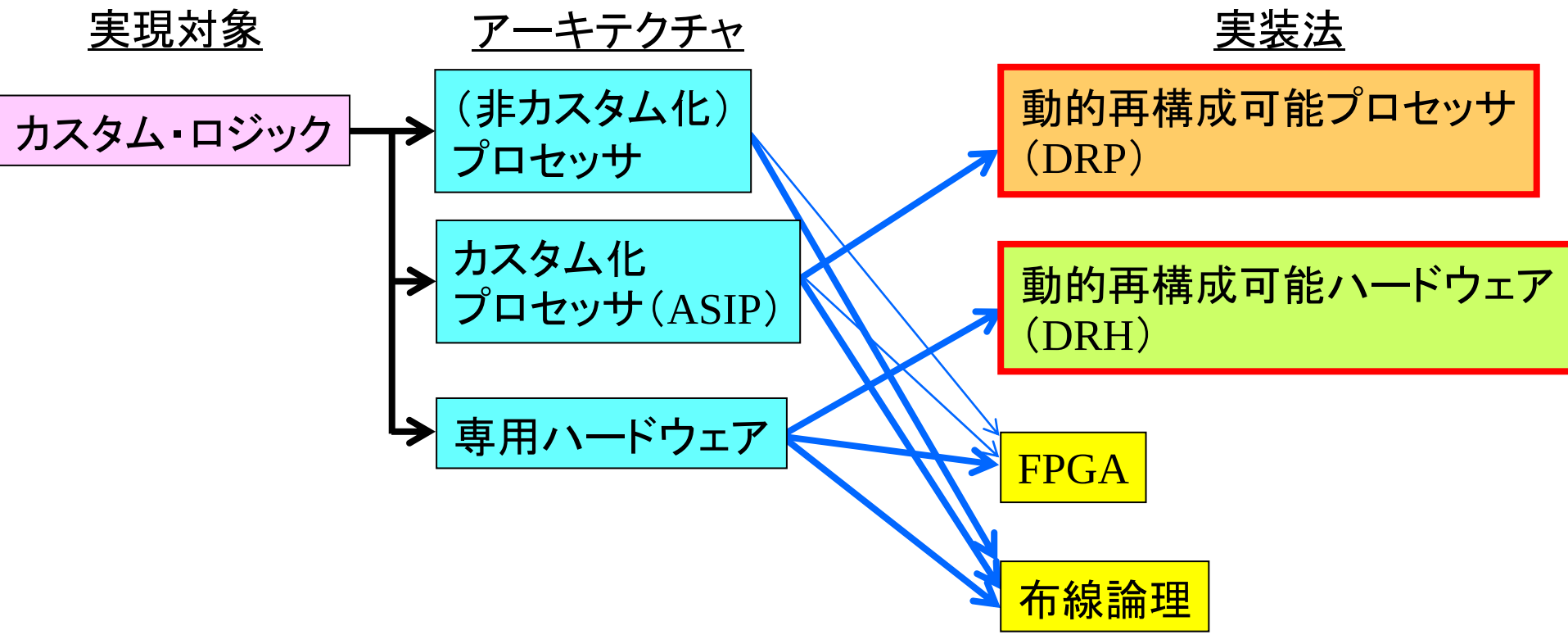
- ハードウェア



- プロセッサ＋ソフトウェア

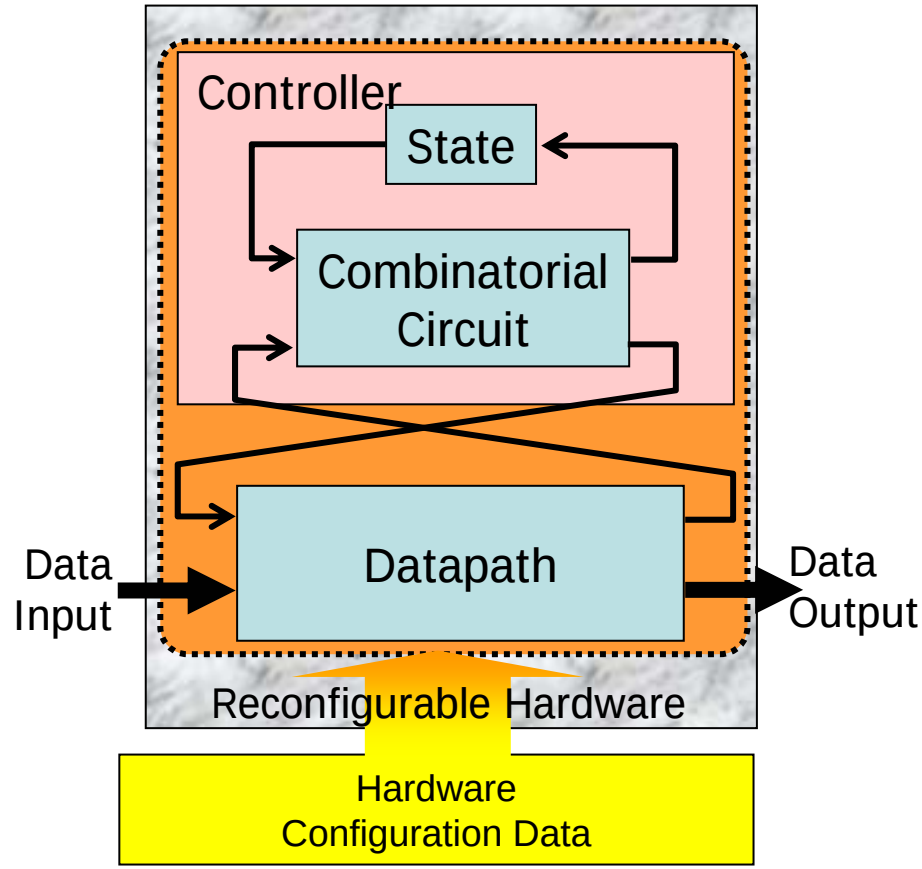


動的再構成可能ハードウェア対動的再構成可能プロセッサ

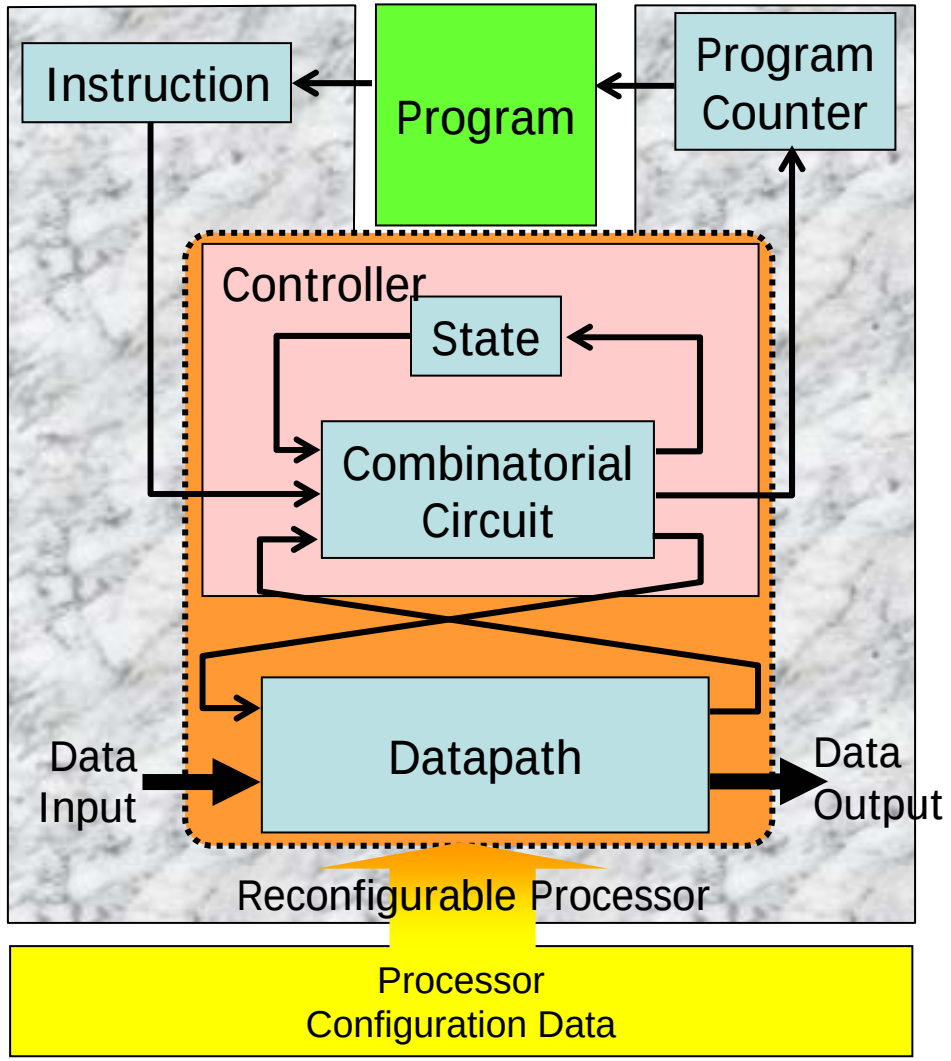


動的再構成可能ハードウェア対
動的再構成可能プロセッサ

- 動的再構成可能ハードウェア

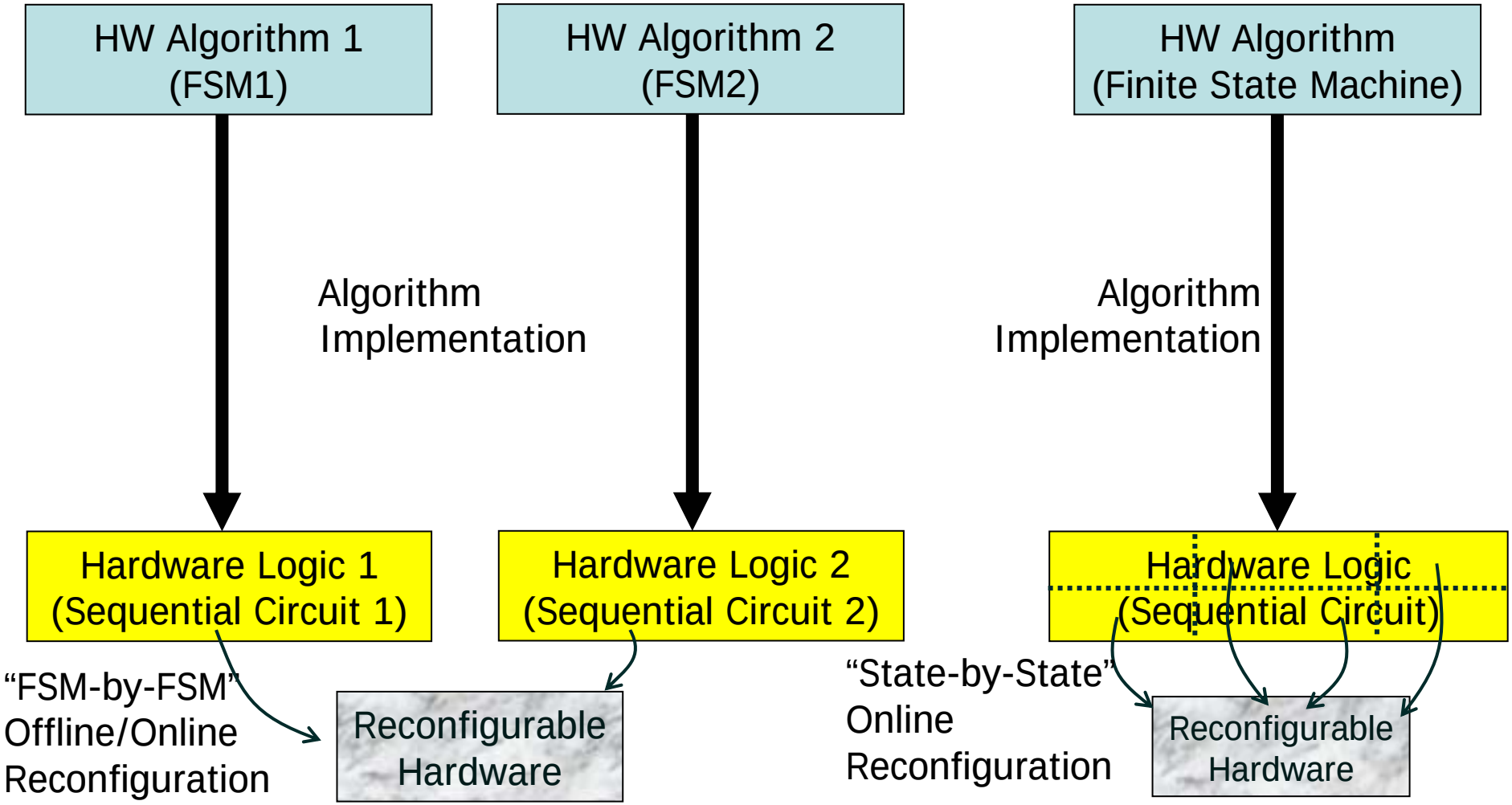


- 動的再構成可能プロセッサ



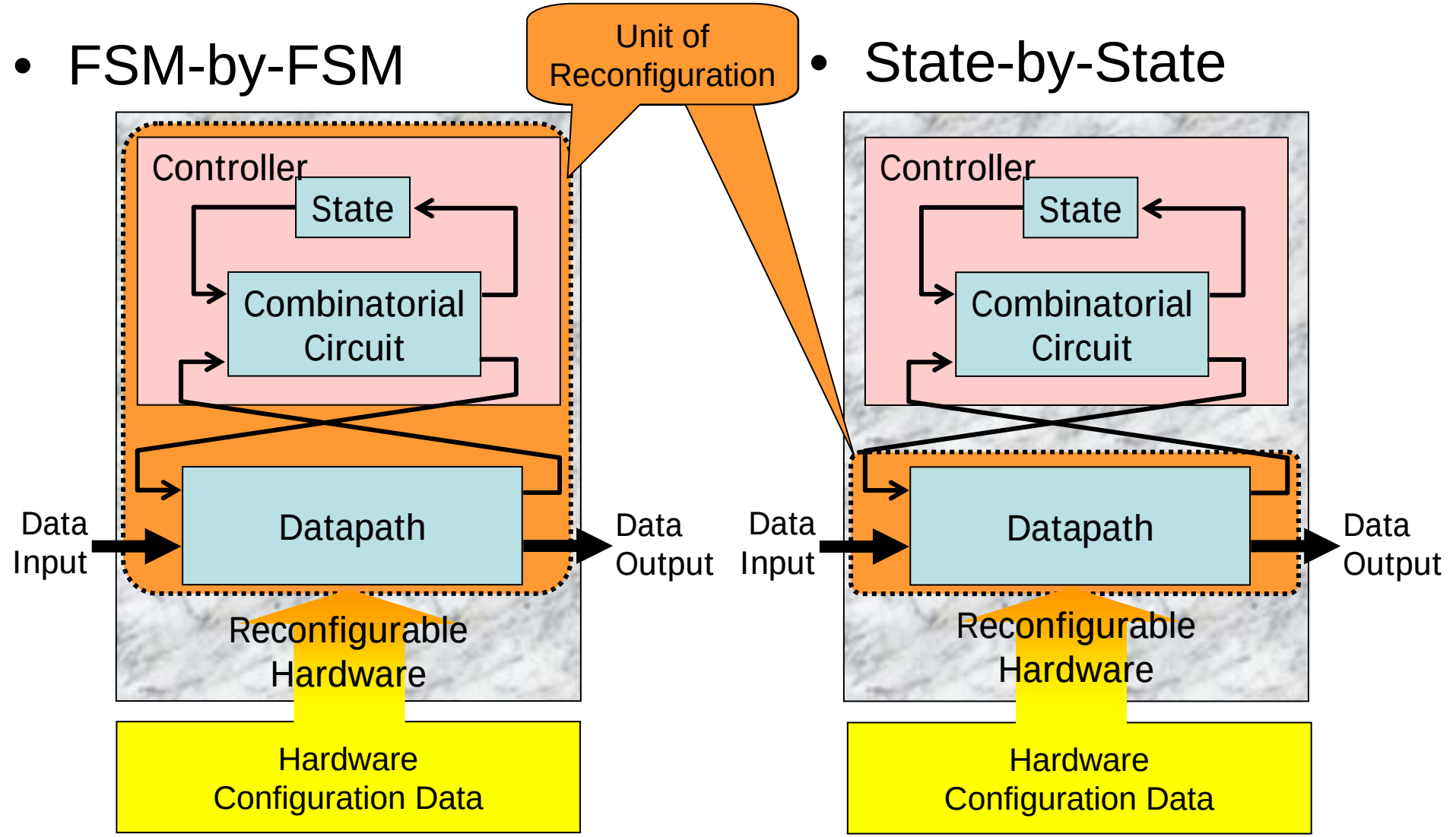
動的再構成可能ハードウェア
～時間的粒度～

- FSM-by-FSM
- State-by-State



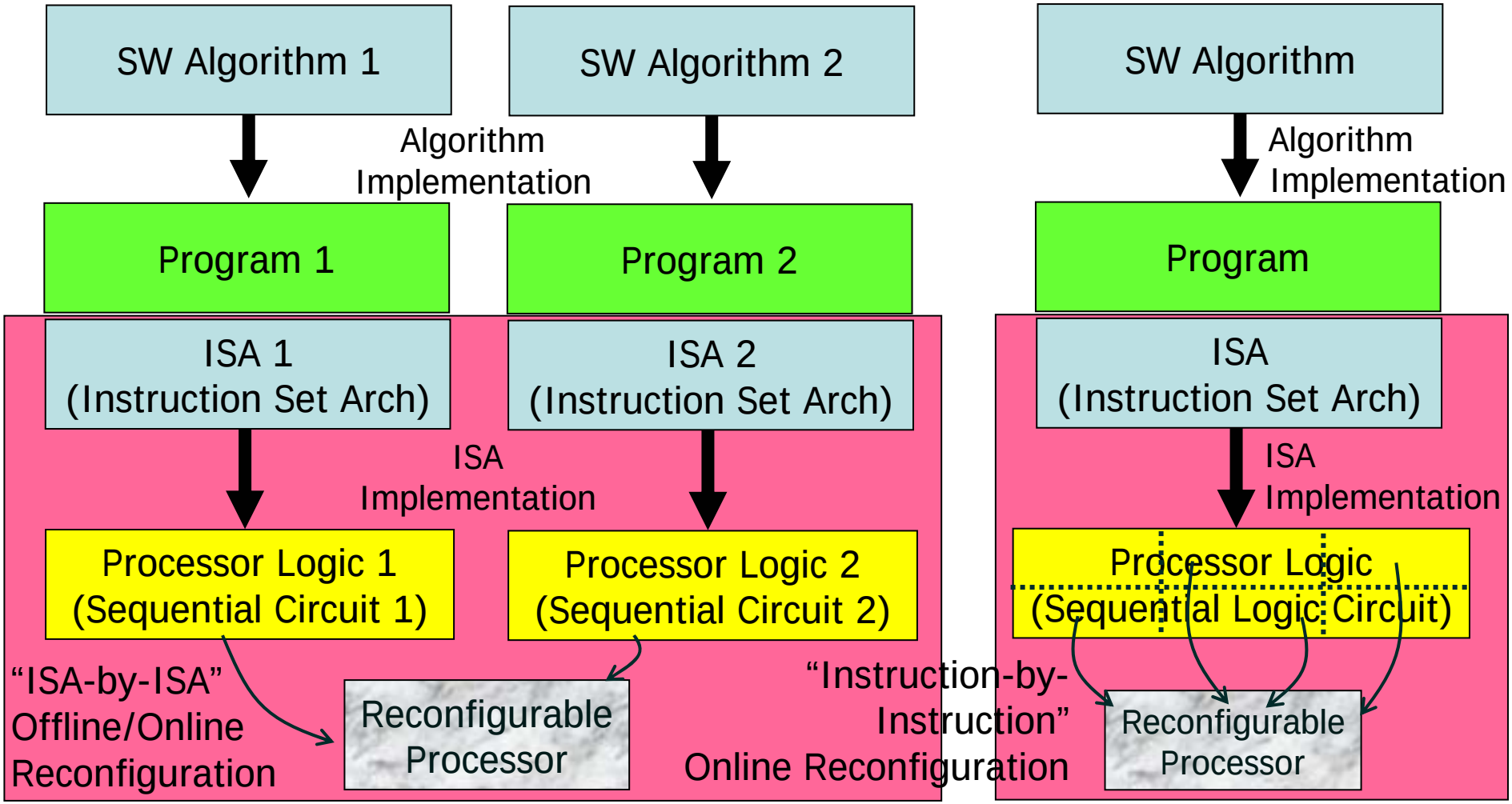
動的再構成可能ハードウェア
～時間的粒度～

- FSM-by-FSM
- State-by-State



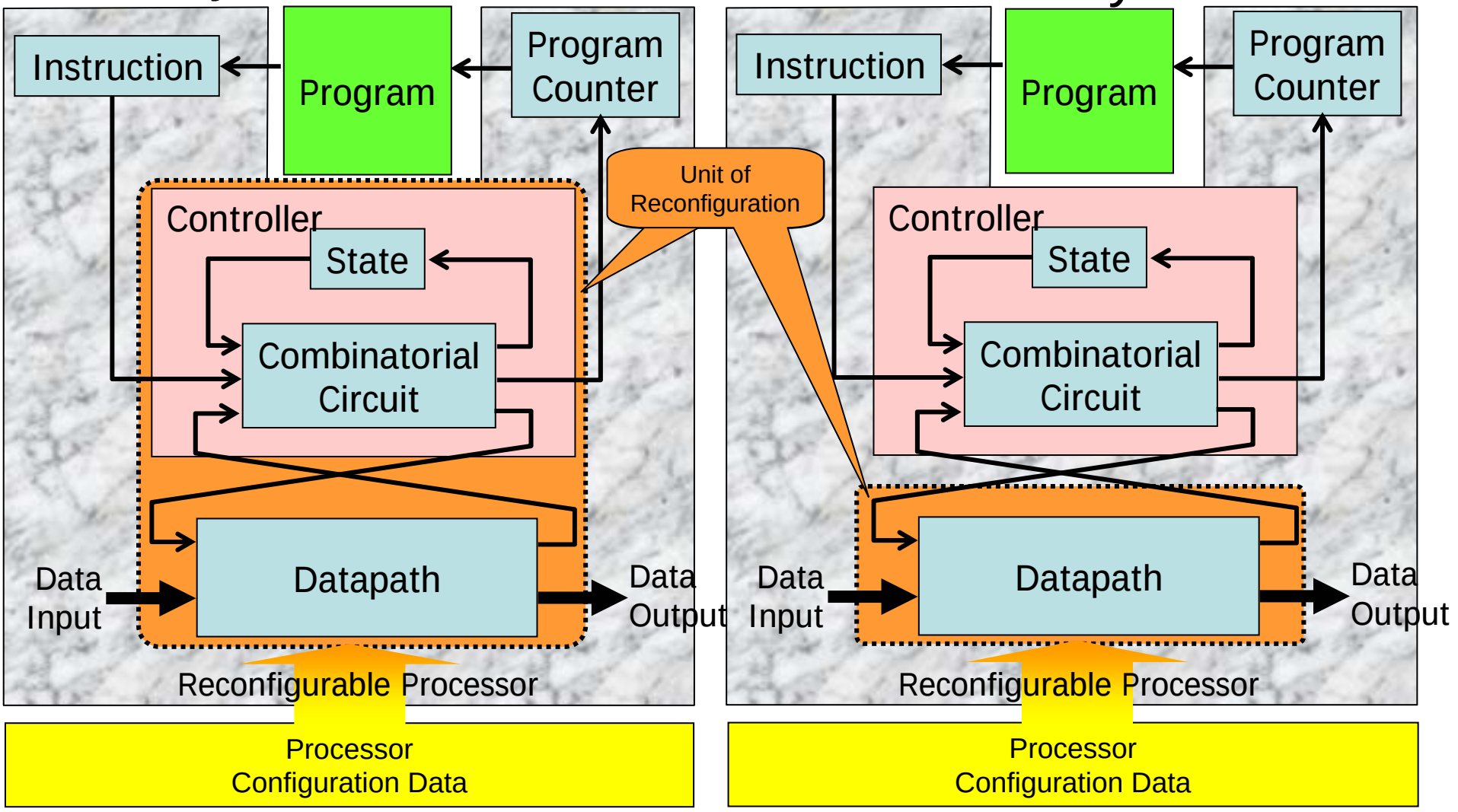
動的再構成可能プロセッサ
～時間的粒度～

- ISA-by-ISA
- Instruction-by-Instruction



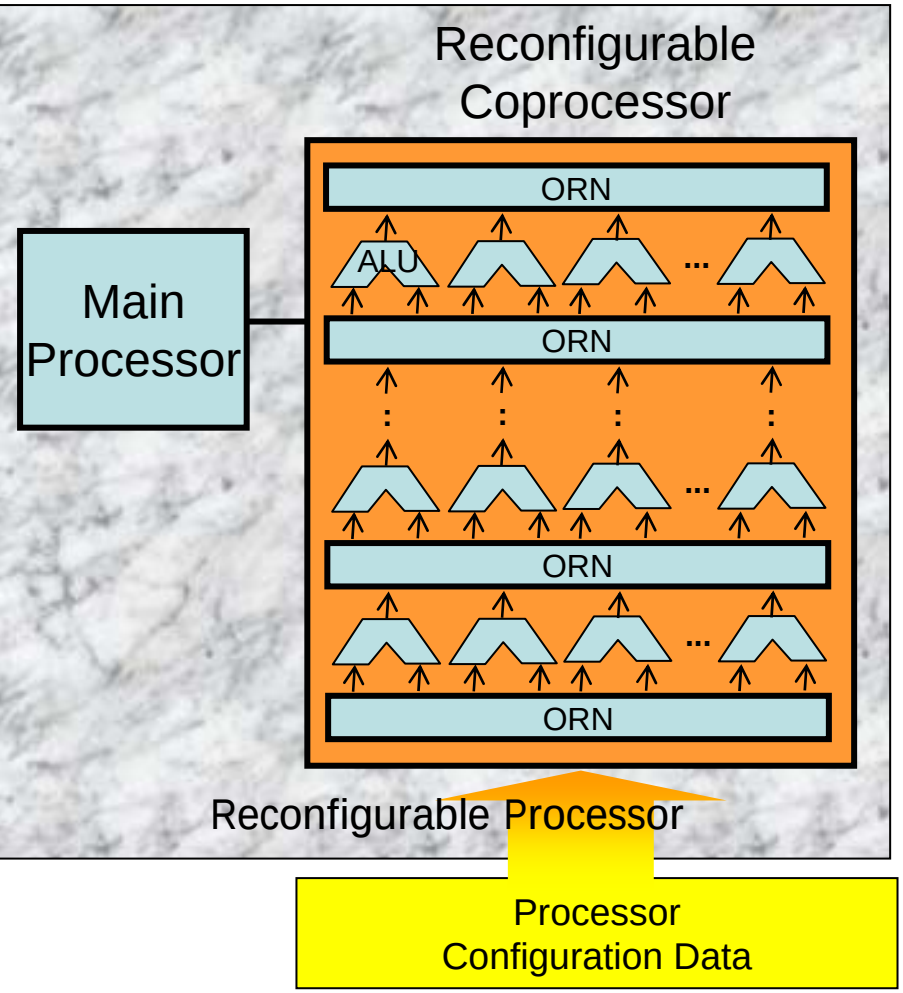
動的再構成可能プロセッサ
～時間的粒度～

- ISA-by-ISA
- Instruction-by-Instruction

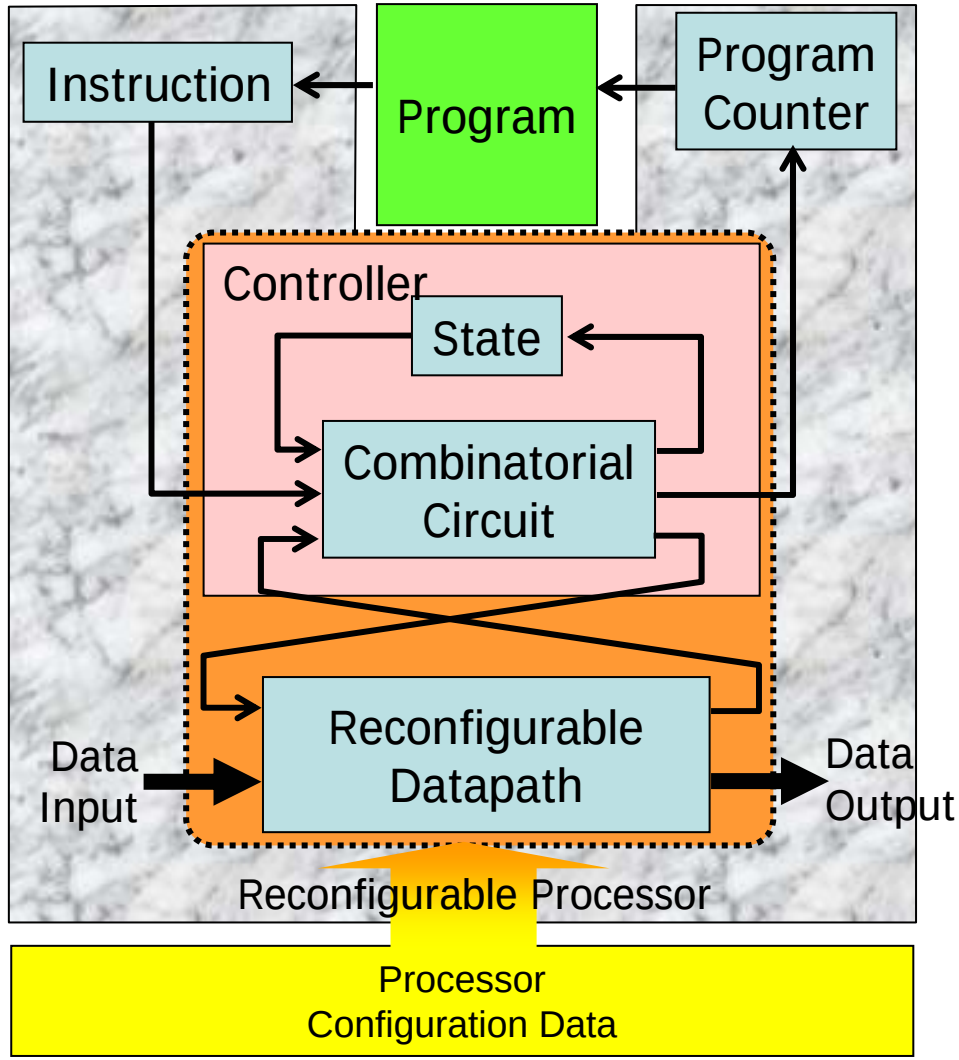


動的再構成可能プロセッサ・アーキテクチャ

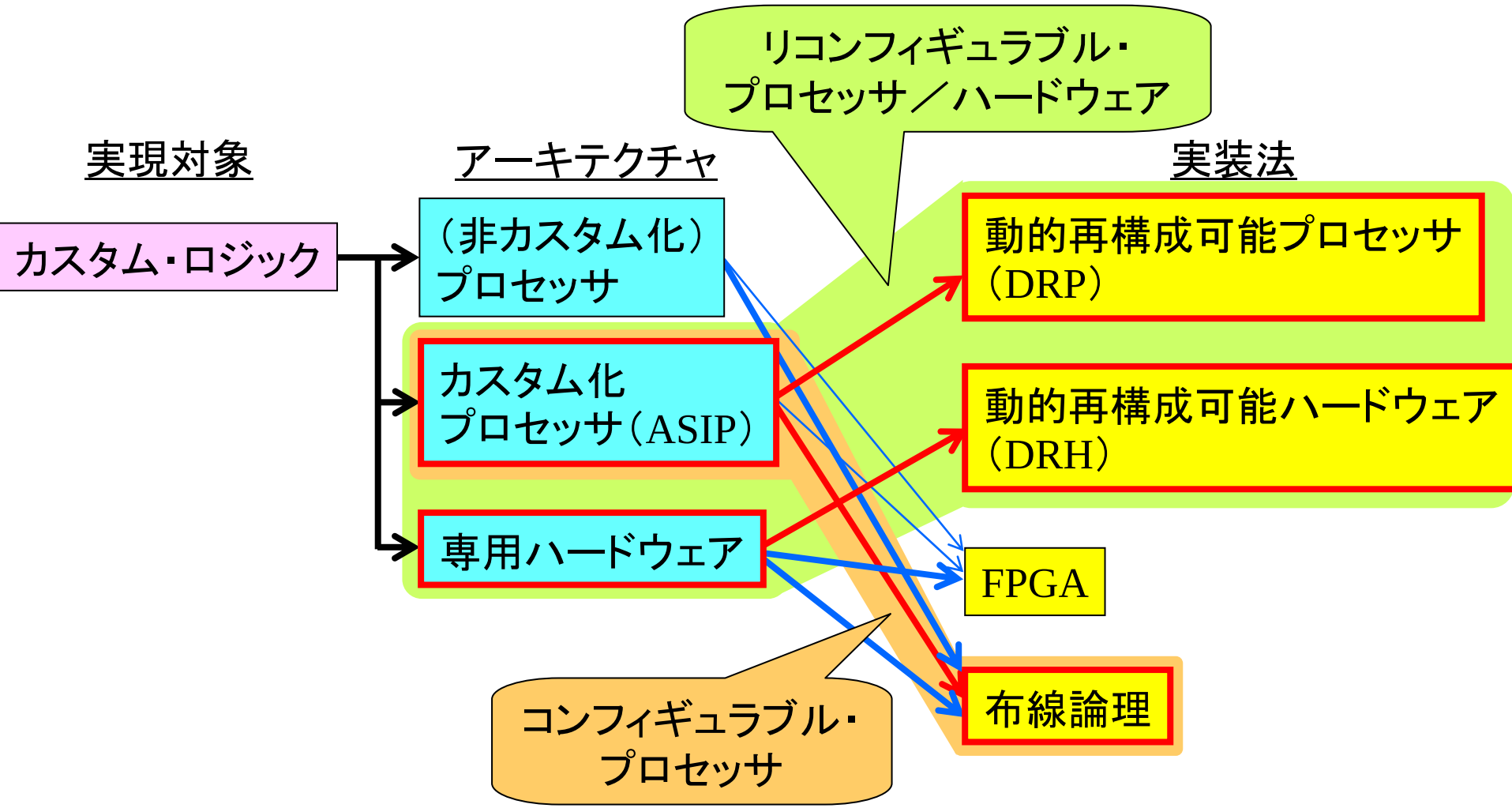
- コプロセッサ型



- プロセッサ内蔵型



コンフィギュラブル 対 リコンフィギュラブル

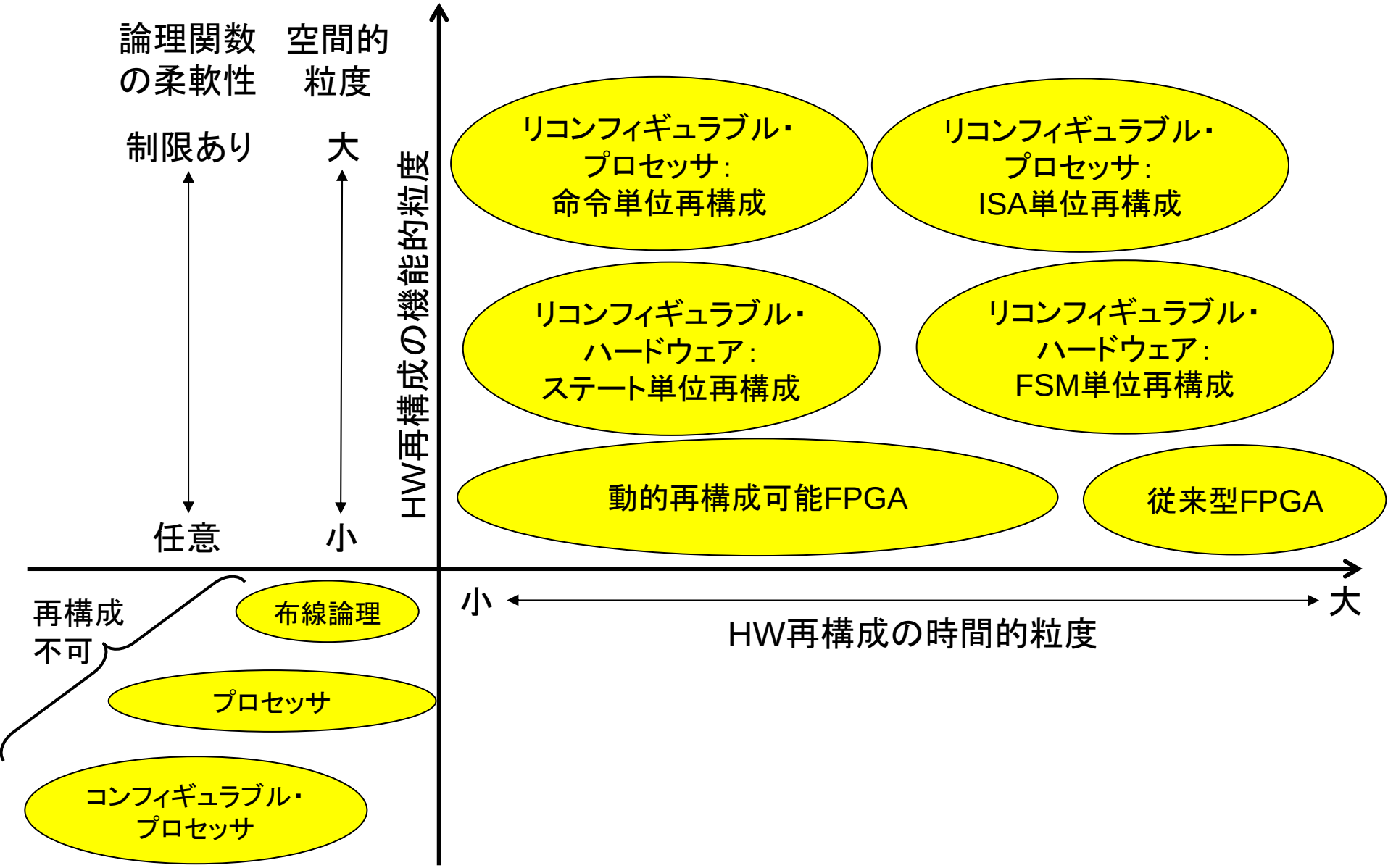


コンフィギュラブル 対 リコンフィギュラブル

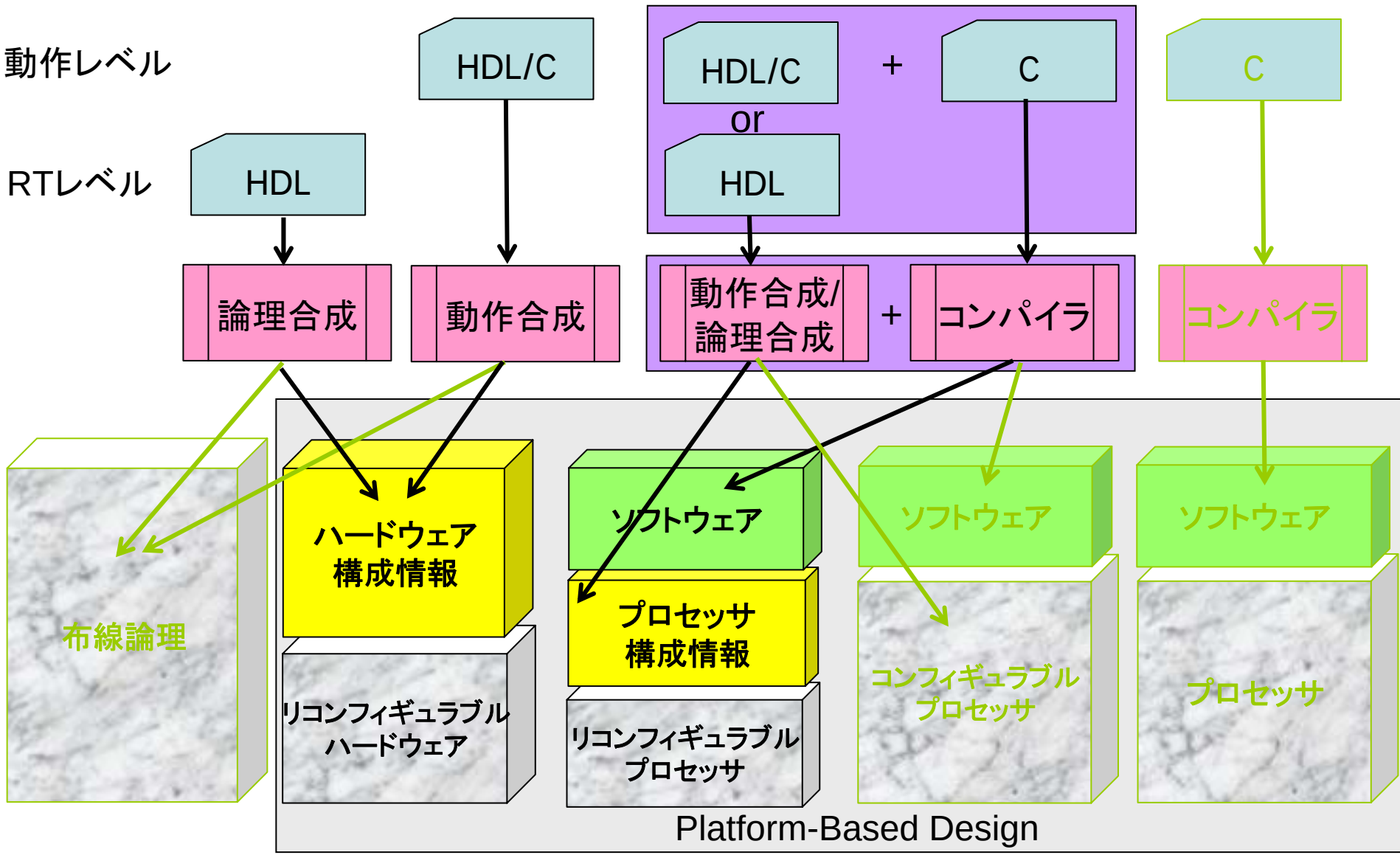
- 共通の性質：ハードウェアまたはプロセッサとして実現する機能が（SoC設計者によって）設定可能
  - Configurable（構成可能）：上記設定が「一度だけ」可能
  - Reconfigurable（再構成可能）：上記設定が「何度でも」可能

	ハードウェア	プロセッサ
非コンフィギュラブル	―	通常のプロセッサ
コンフィギュラブル	通常のハードウェア設計	コンフィギュラブル・プロセッサ
リコンフィギュラブル	リコンフィギュラブル・ハードウェア	リコンフィギュラブル・プロセッサ

HW再構成可能性に関する設計空間

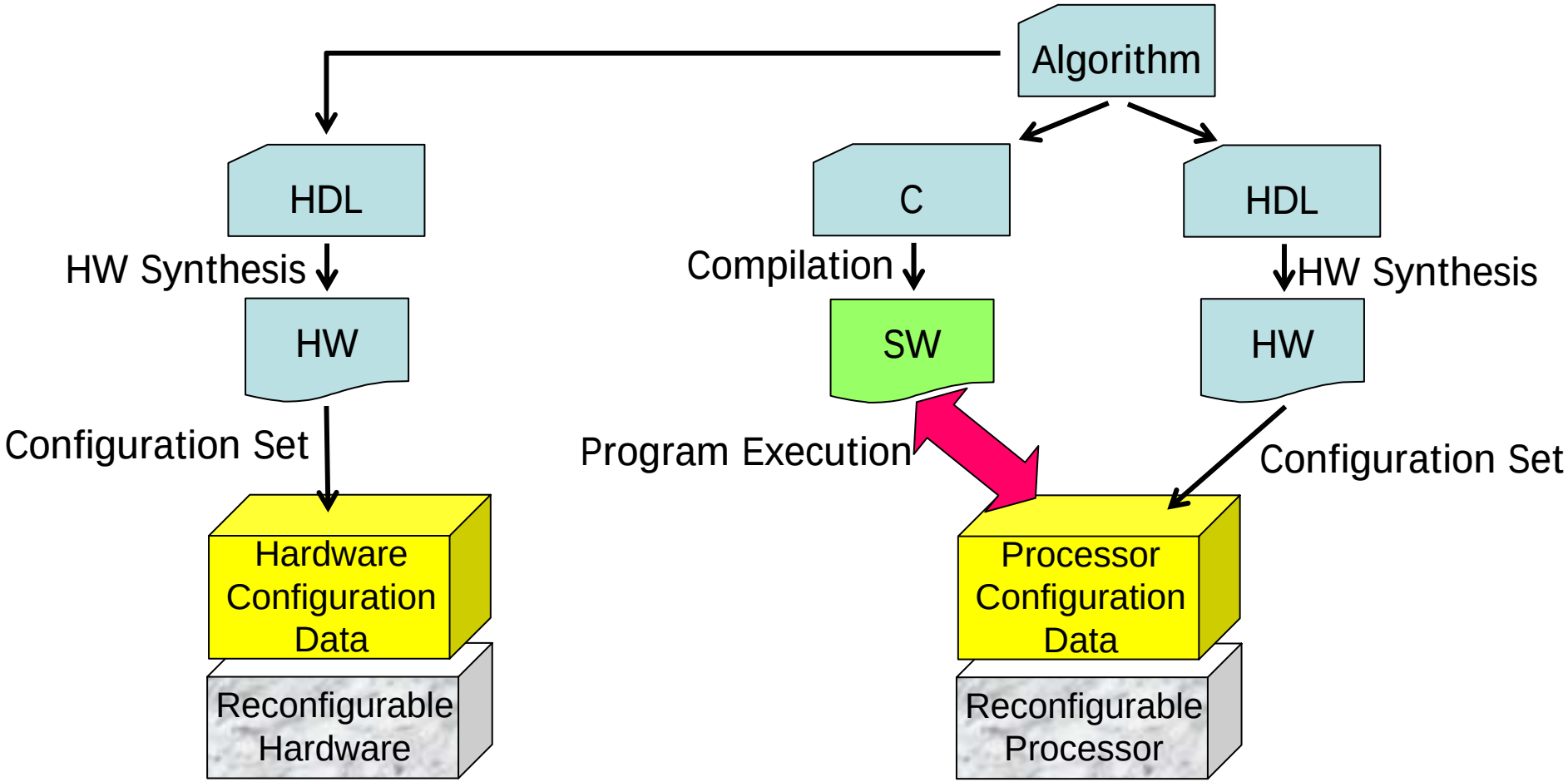


SoC設計への諸アプローチ

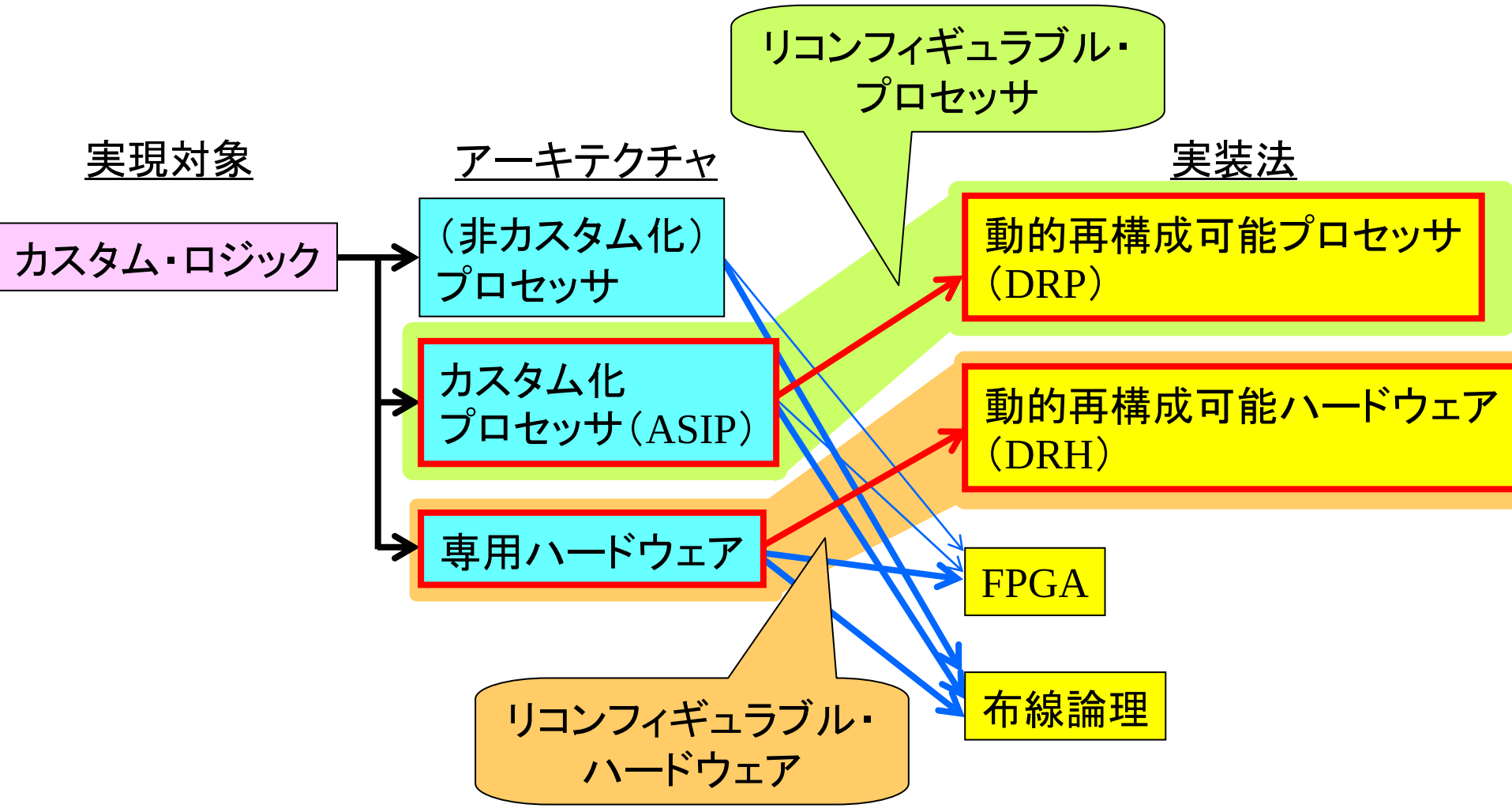


リコンフィギュラブル・ハードウェア 対 リコンフィギュラブル・プロセッサ

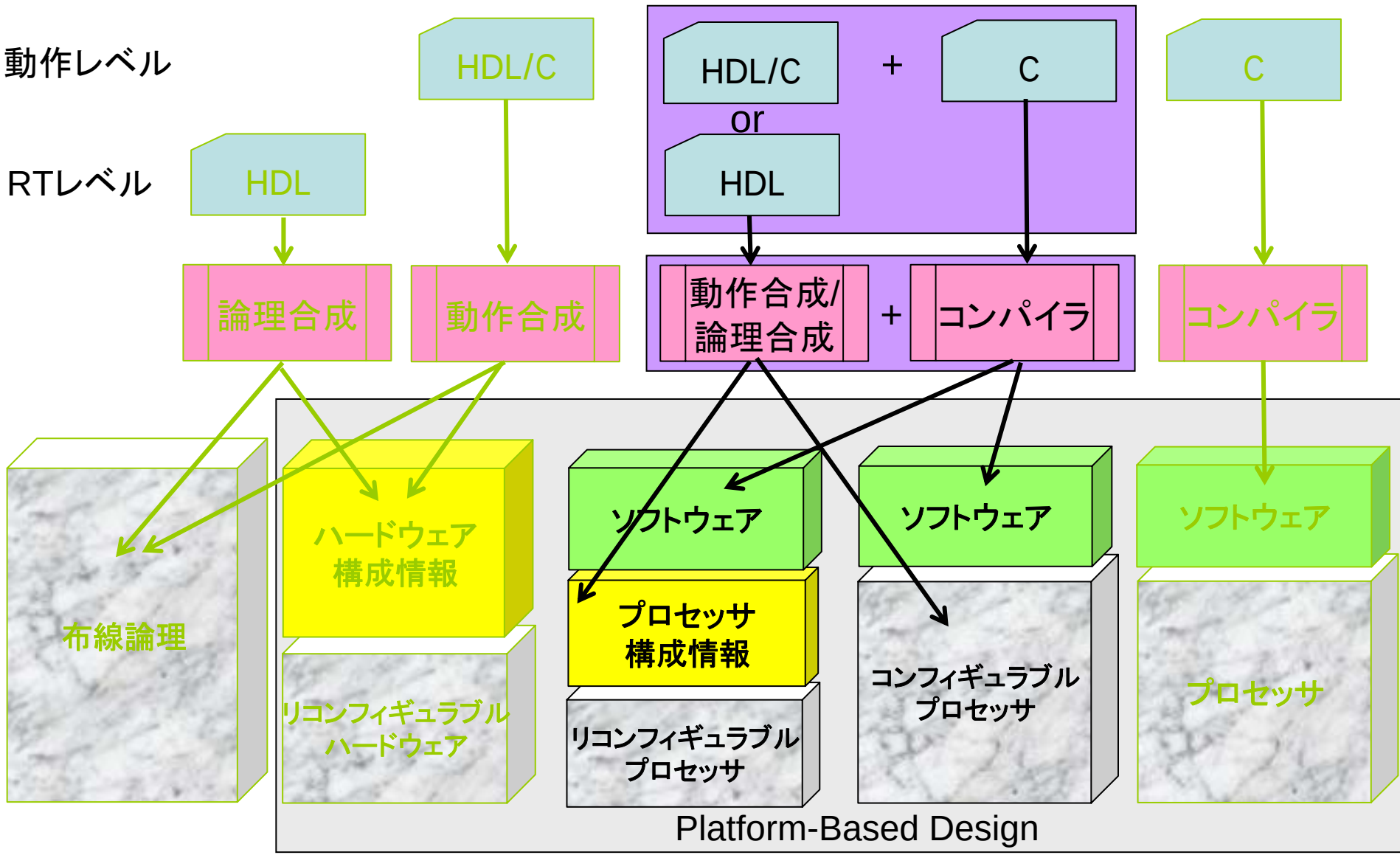
- リコンフィギュラブル・ハードウェア
- リコンフィギュラブル・プロセッサ



リコンフィギュラブル・ハードウェア 対 リコンフィギュラブル・プロセッサ

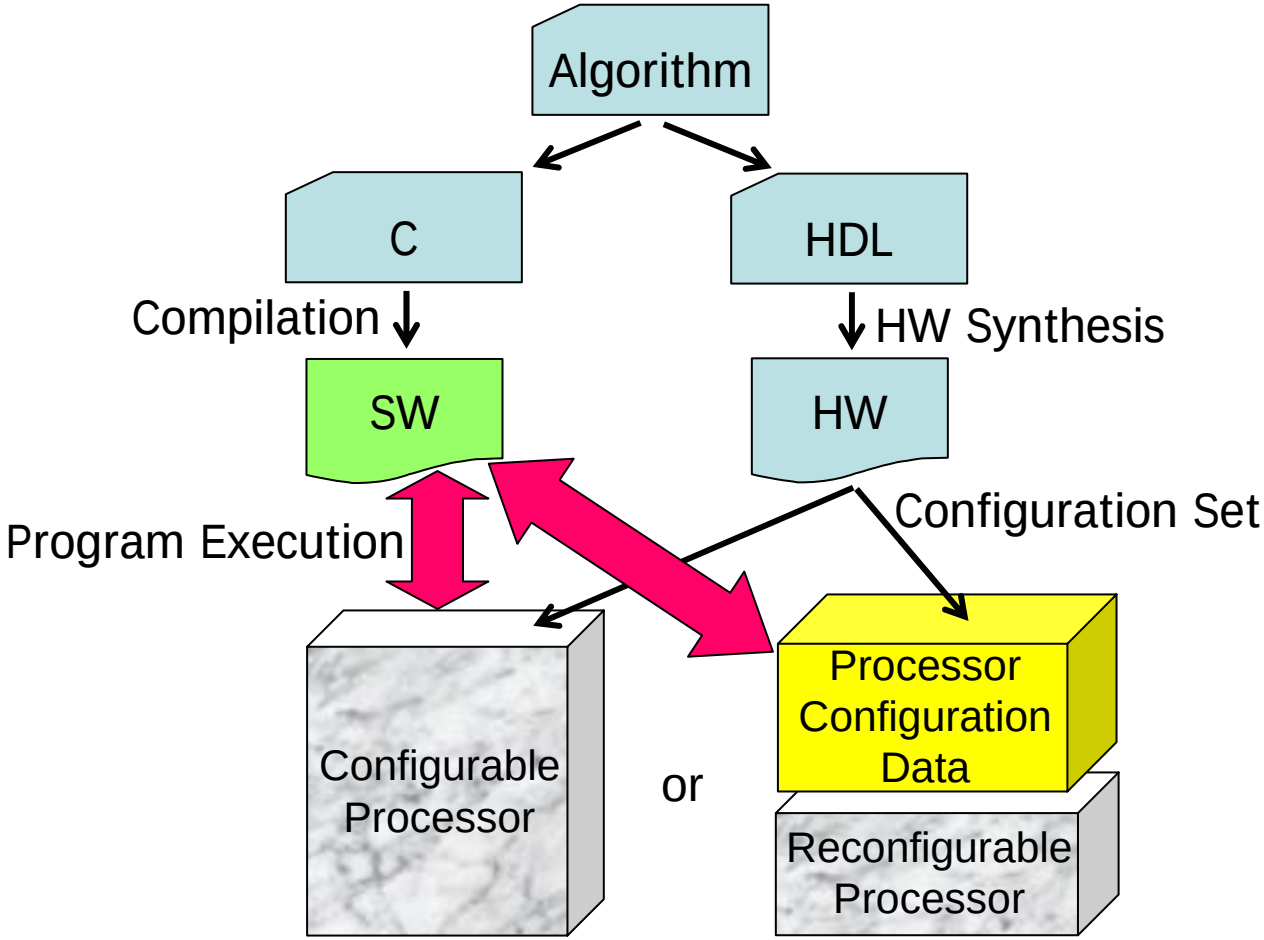


SoC設計への諸アプローチ

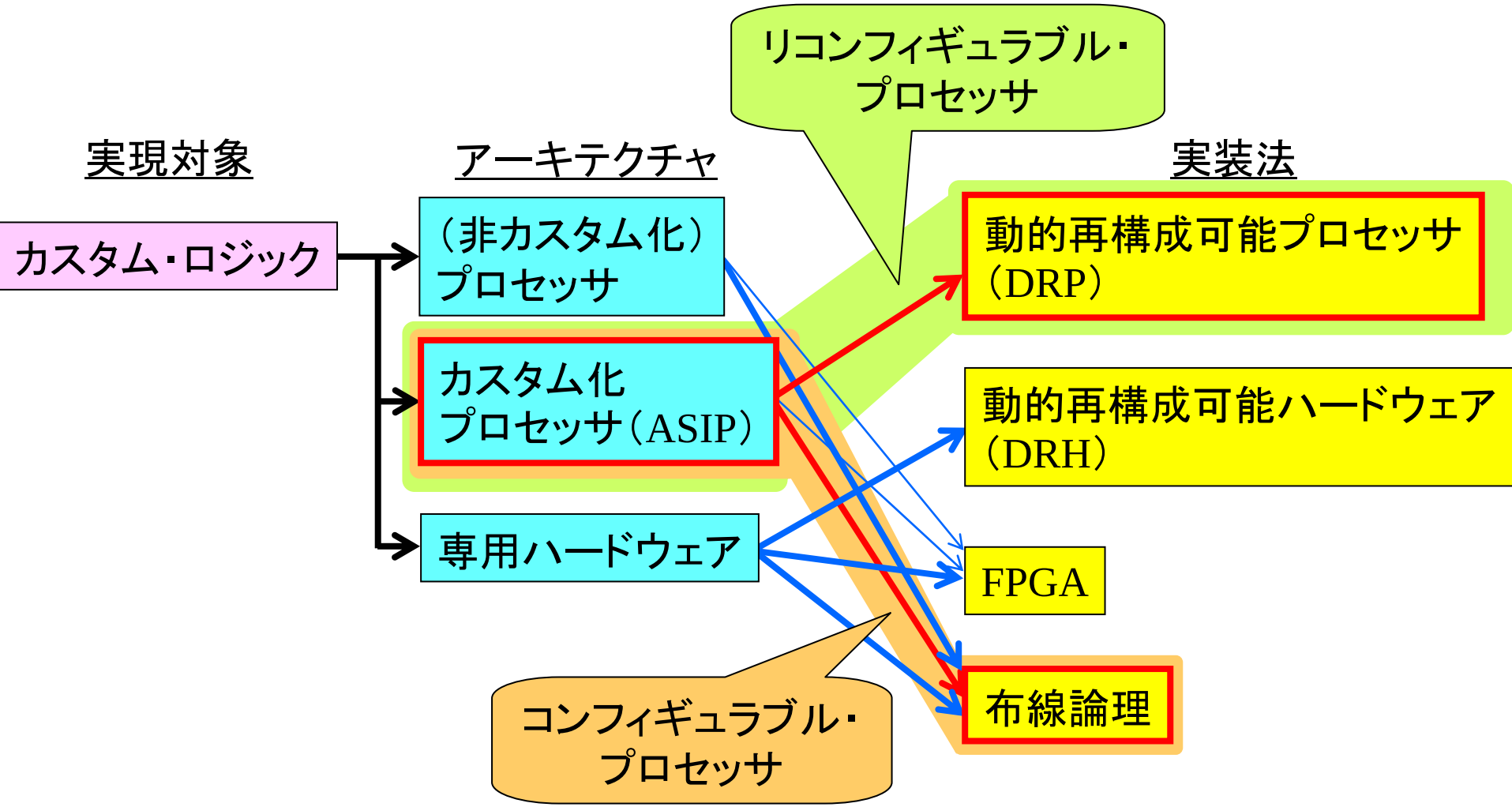


コンフィギュラブル・プロセッサ 対 リコンフィギュラブル・プロセッサ

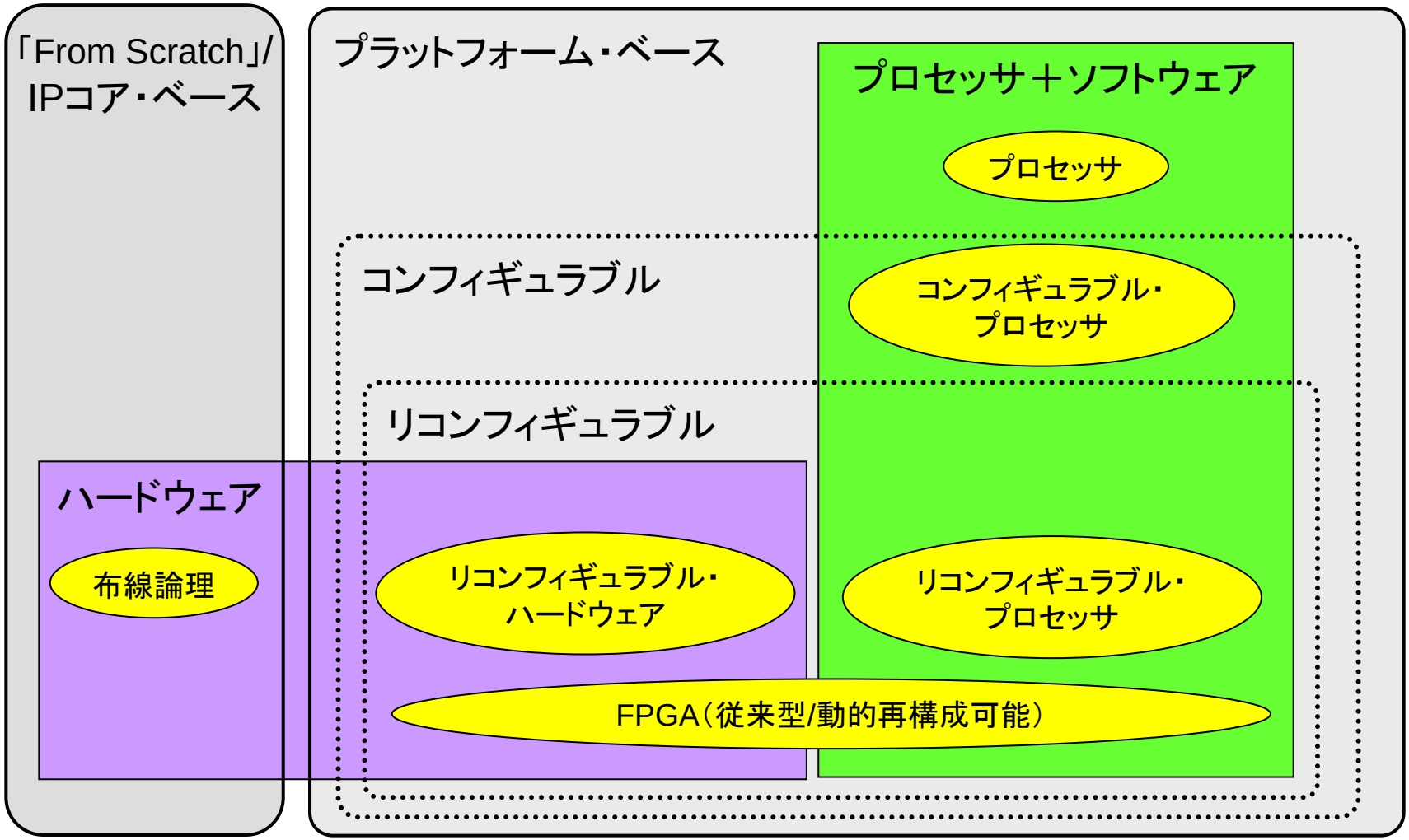
- コンフィギュラブル・プロセッサ
- リコンフィギュラブル・プロセッサ



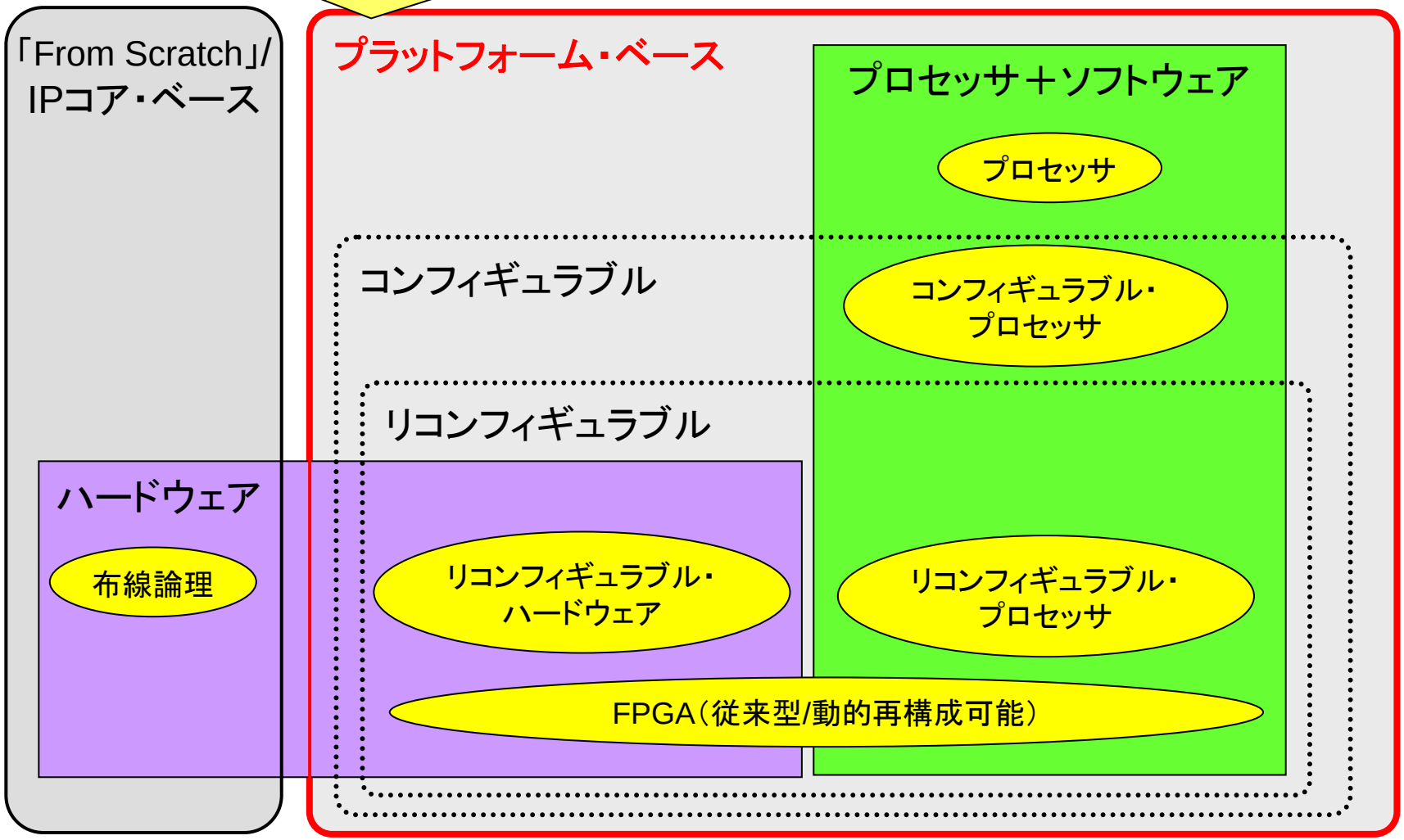
コンフィギュラブル・プロセッサ 対 リコンフィギュラブル・プロセッサ



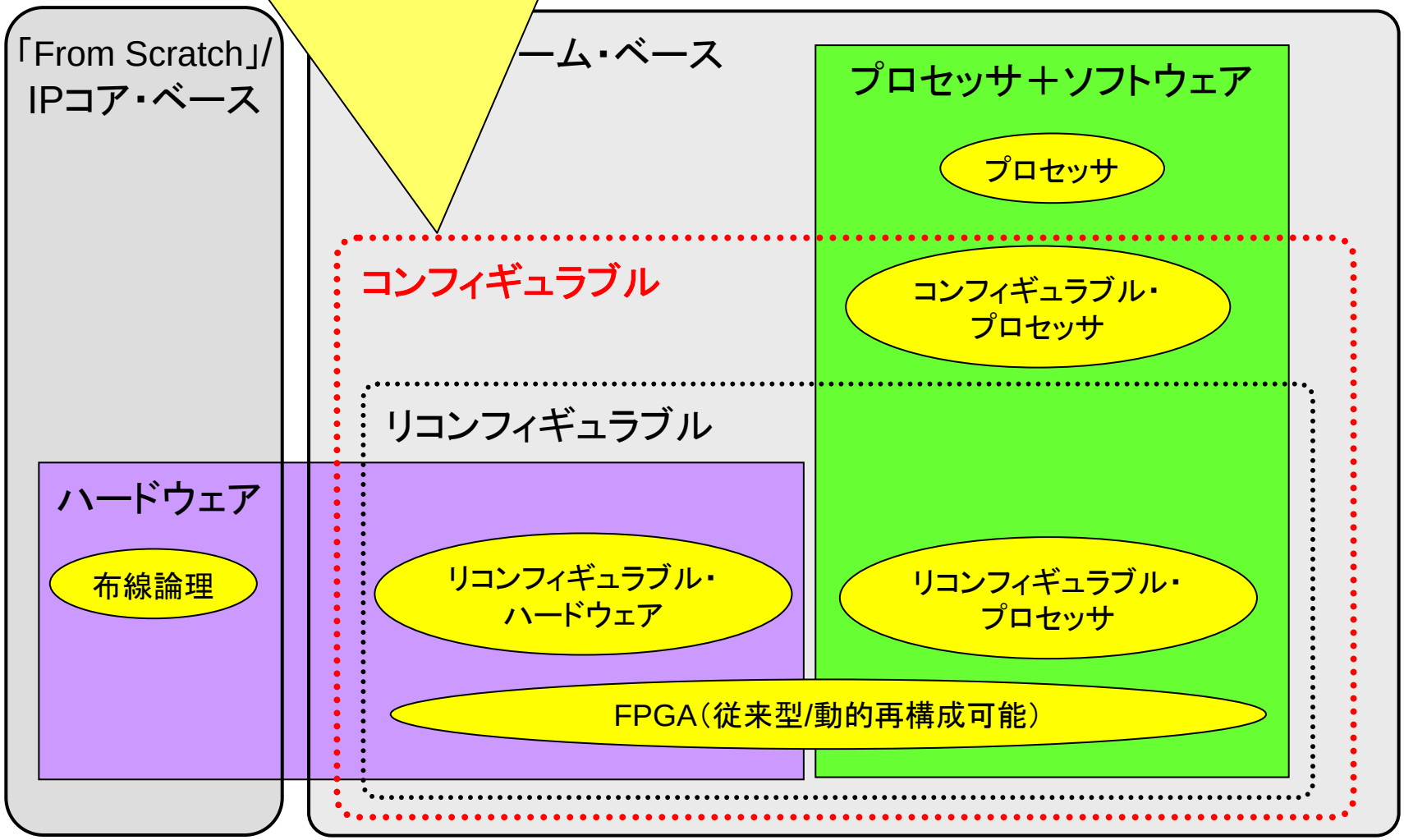
SoC設計への諸アプローチ



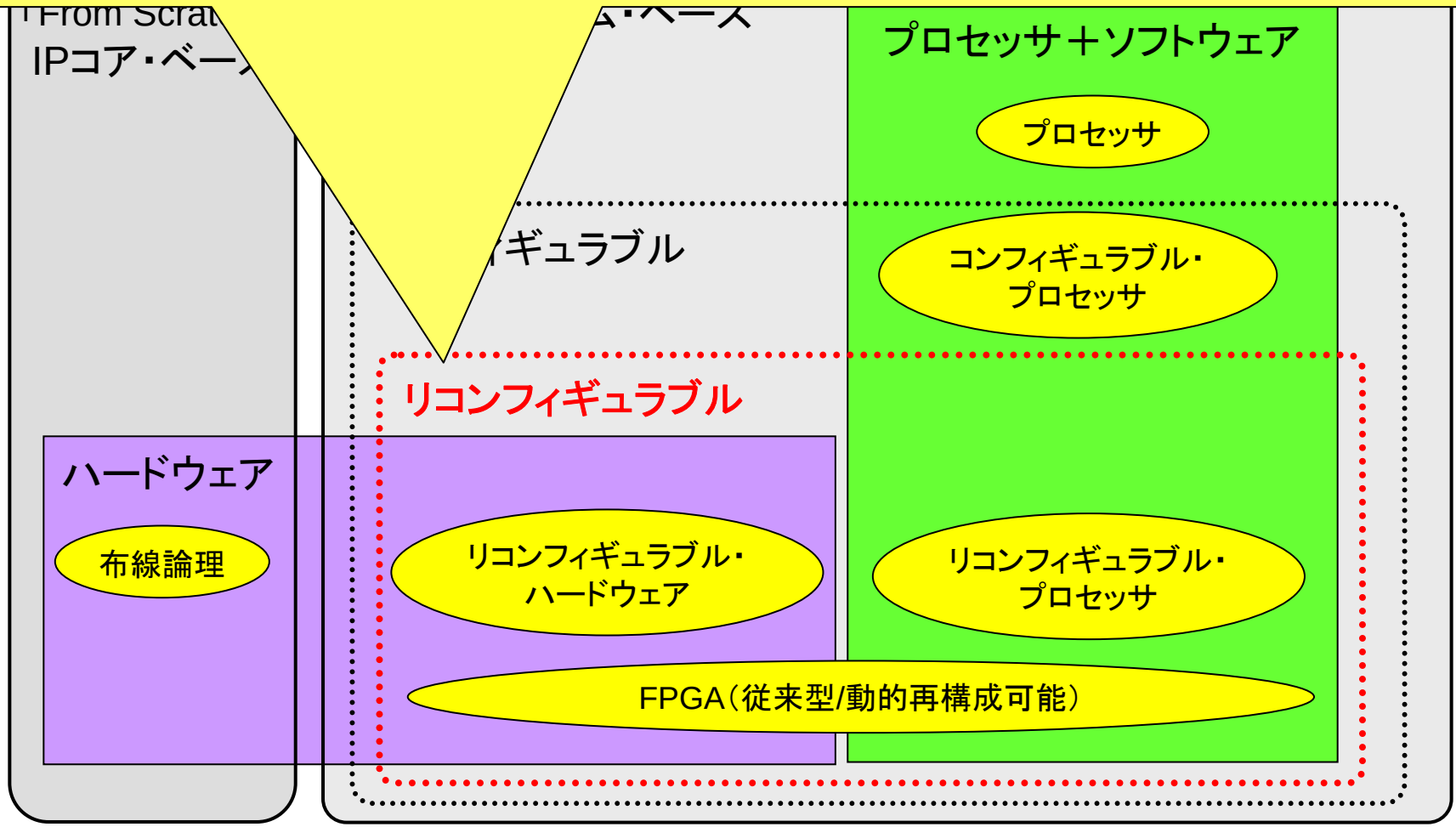
- SoC設計作業を①プラットフォーム実装，および，②その上での専用機能実現に分割！
- プラットフォームを再利用することで，SoC設計作業を②の「専用機能実現」に集中し，SoC設計期間を短縮！
- さらに，プラットフォーム上で多用な専用機能が実現可能なことから，SoC設計の共有＆汎用化が可能→少品種大量生産が可能に！



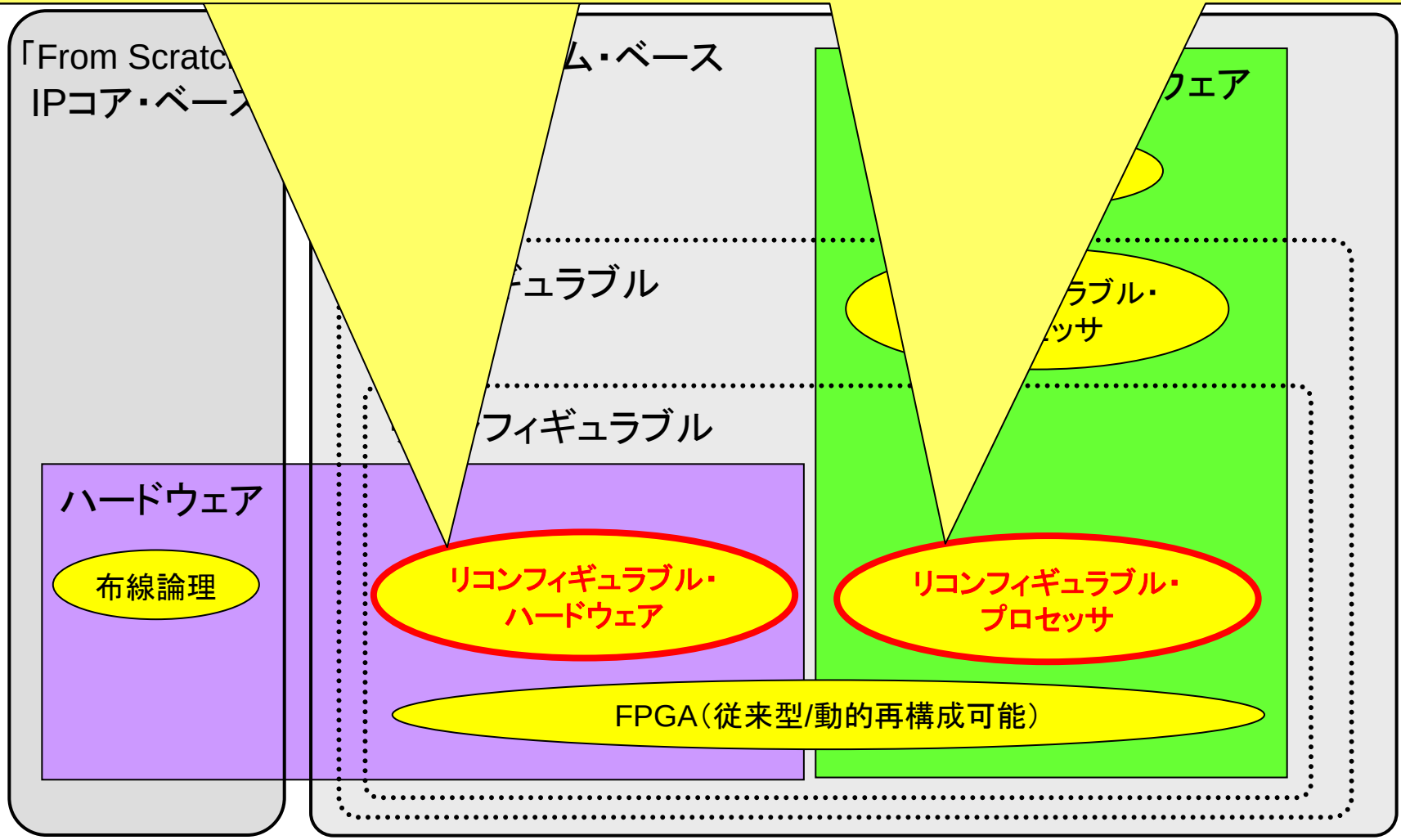
- 実現すべき専用機能に合わせてプラットフォーム（プロセッサ・ロジック）の構成，機能をカスタム化可能！
  ↗従来型プロセッサ/DSPに比べてコスト対性能比が向上
  ↘従来型プロセッサ/DSPに比べて，カスタム化のためにSoC設計期間が増大



- プラットフォーム（ハードウェア・ロジック，または，プロセッサ・ロジック）の構成，機能をカスタム化するタイミングをSoC設計終了後にまで延期可能！
  ↗コンフィギュラブル・プロセッサに比べてSoC設計期間を短縮
- 再構成のためのオーバヘッド（面積，時間）が不可避
- ただし，再構成可能性を活用することで，実装すべき機能を（布線論理や従来型プロセッサのように）ナイーブに実装するより，必要面積を削減することが可能
  – コンフィギュラブル・プロセッサに比べたコスト対性能比は？



- コスト対性能比：
  ＋リコンフィギュラブル・ハードウェア・ロジック
  －リコンフィギュラブル・プロセッサ
- 設計生産性，Cレベル設計との親和性：
  －リコンフィギュラブル・ハードウェア・ロジック
  ＋リコンフィギュラブル・プロセッサ



結論／まとめ

- 大学では「設計実践」もさることながら、問題や技術の「分析」、「分類」、「体系化」が出来る学生を育成する必要あり
  - 言い換えれば「問題の本質」を見極める能力を養成！
- 「現象論」のみならず「本質論」、そして、「本質と現象の関係」を議論したSoCに関する教科書（シリーズ）を早急に編纂すべし