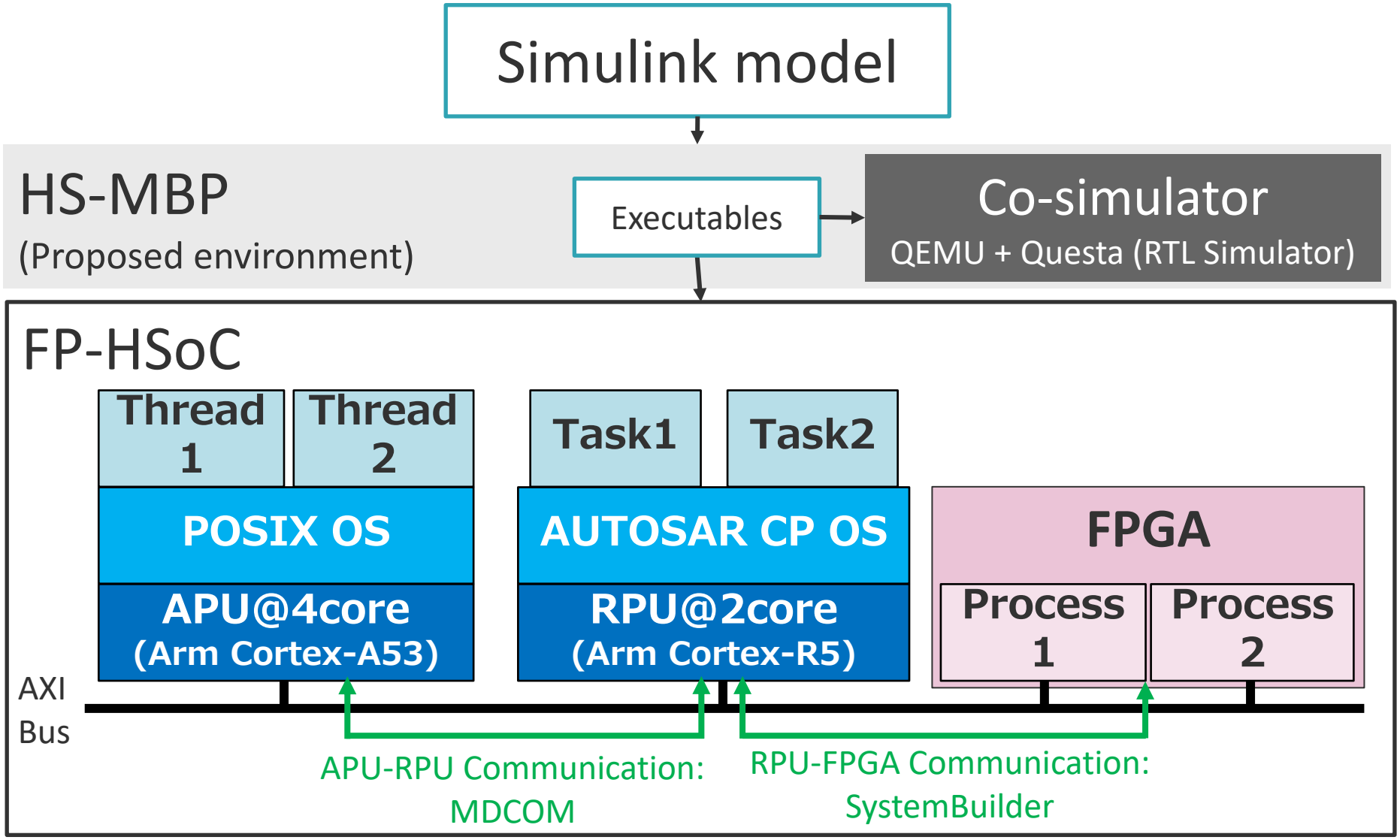
HS-MBP: FPGA混在ヘテロジニアスマルチコア向けモデルベース開発環境の紹介

名古屋大学大学院情報学研究科 研究員
山本 椋太

謝辞 本研究の一部は，国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）の委託業務（JPNP16007）の結果得られたものです.

今日の発表の概要: HS-MBP



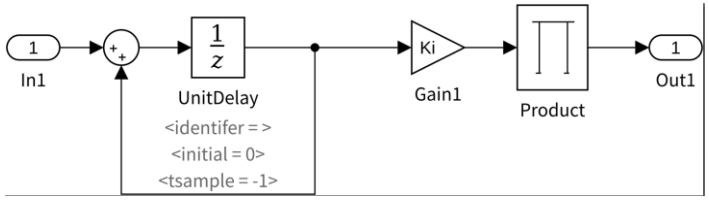
背景: 現状と問題

- 組込み制御システムへの要求
  - 大規模化
  - 複雑化
  - Time-To-Market (TTM) の短縮
  - 処理負荷の高い処理の活用に対する性能要求
    - 例: 自動運転や機械学習など

- 要求に対して生じる問題
  - 複雑化・大規模化による開発期間の増大
    → TTM要求を満たせない
  - マルチコア環境であっても性能が不十分
    → 性能要件や消費電力要件を満たせない

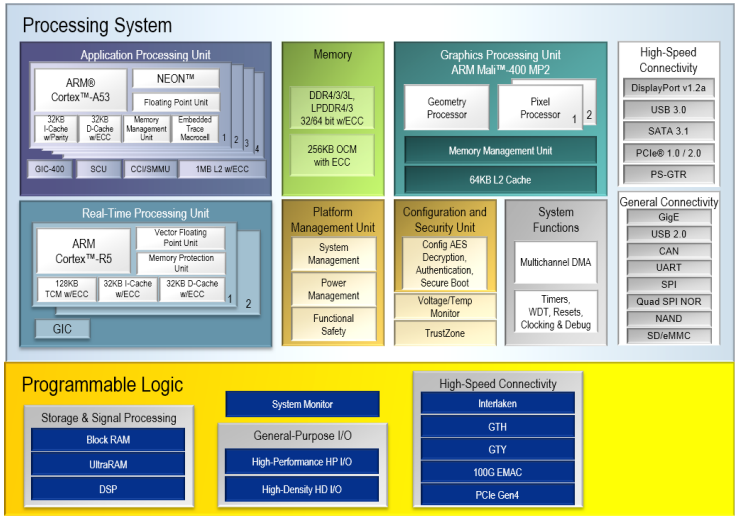
背景: 問題に対する解決方法

- 複雑化・大規模化に伴う開発期間の短縮
  - モデルベース開発 （MBD）環境の利用
    - 例) MATLAB/Simulink
  - アルゴリズムレベル検証が容易
  - 並列化コードの生成は未サポート



Simulinkモデルの例

- 高性能化
  - ヘテロジニアスマルチコアSoC（FP-HSoC）の利用
    - FPGA混在のFP-HSoCを想定
  - 異なるPE間での通信・同期機構の実装，PE向けの最適化が必要

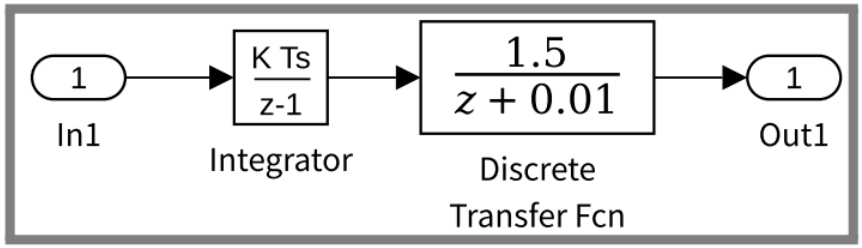


FP-HSoC の構成 [1] より引用

[1] Xilinx: Zynq UltraScale+ MPSoC, https://japan.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html （2020 年 7 月 30 日参照）.

MBD (Model Based Development)

- 数学的なアルゴリズムの設計開発を支援
  - 最も有名なツール: MATLAB/Simulink



- MATLAB/Simulink はCコード生成を支援
  - しかし，逐次コードのみである

逐次コードを並列コードに書き換えるためにはどうすればいいのだろう？

HW/SW 協調設計

- HWとSWでは，記述が違う
  - HW は，高位合成（HLS）技術を使えば，Cで記述可能
  - しかし，SW向けのCでは，HWで動作させることが困難
    - 厳しい型制約
    - 再帰関数は利用不可
    - ポインタ変数の利用方法，etc.

- HW-SWの通信機構の開発は高難度である
  - アーキテクチャをよく理解する必要がある.

ソフトウェア開発者は，どうすれば簡単に HW/SWの協調設計ができるようになる？

目的

開発期間短縮とFP-HSoCの容易な利用のための モデルベース並列化（MBP）環境 HS-MBP の開発

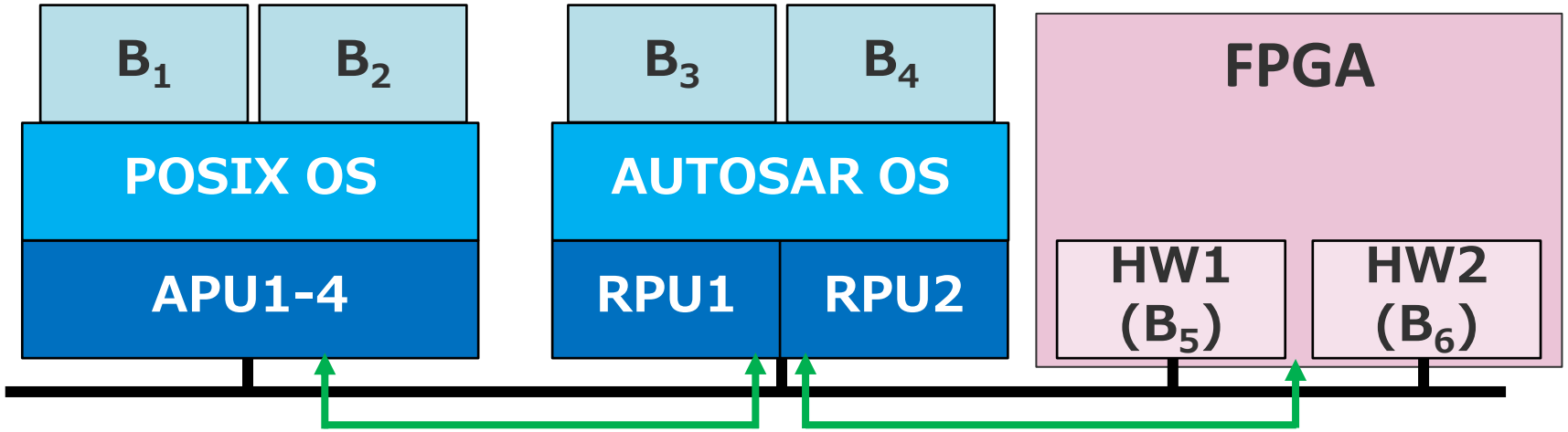
本発表での貢献

- FP-HSoC向けの実行可能ファイルの生成までを支援するHS-MBPのフローの提案
- 並列化時に異なる種類のPE間の通信機構の実装

FP-HSoCの実装時の構成

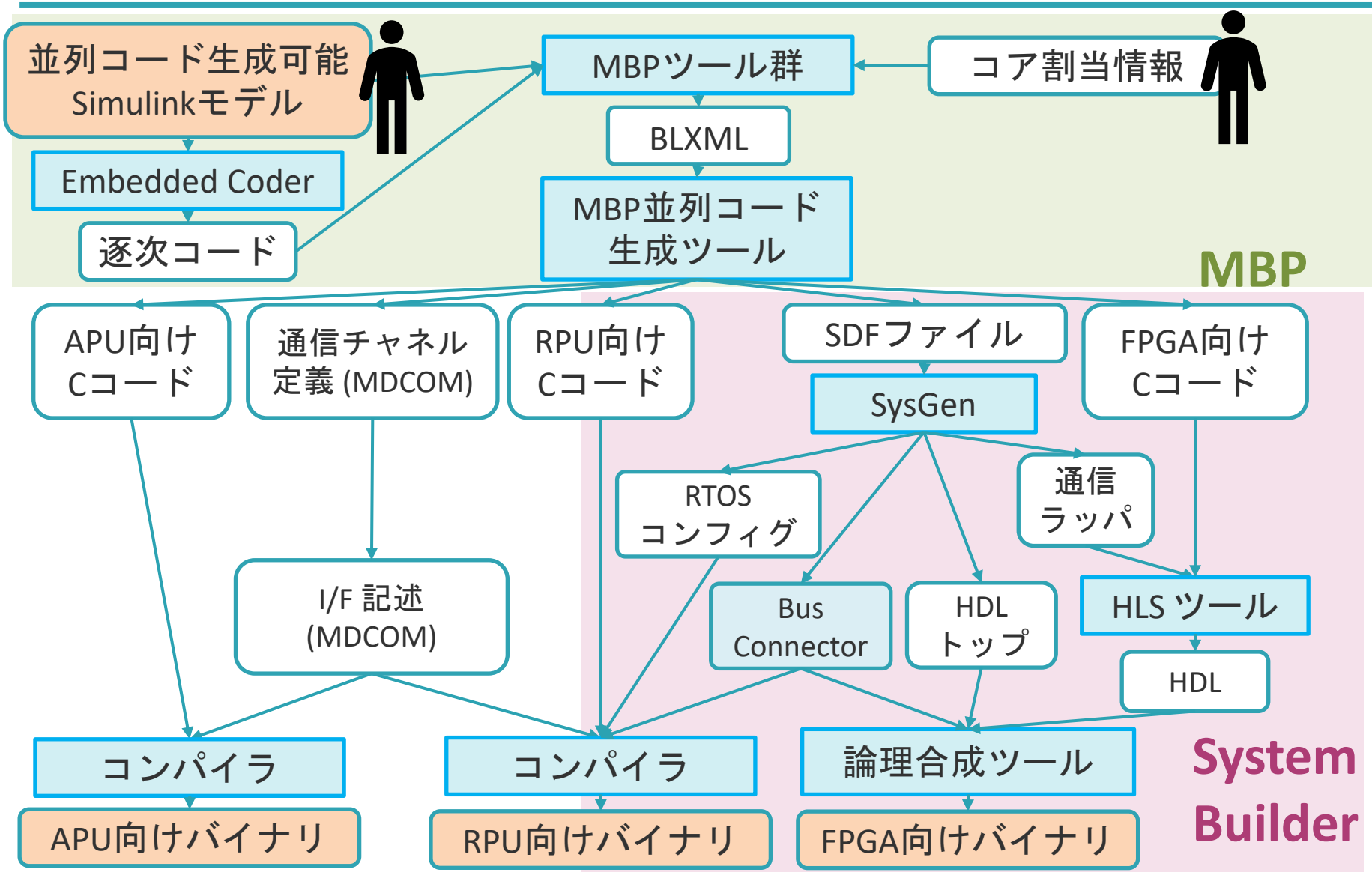
Zynq UltraScale+ MPSoC (Xilinx社, ZynqMPと呼ぶ)

- APU (Application Processing Unit): ARM Cortex-A53 4コア@最大1.5GHz
  - L2キャッシュやパイプライン等の演算高速化機構
- RPU (Real-time Processing Unit): ARM Cortex-R5 2コア@最大600MHz
  - 割込みやローカルRAMへのアクセスが低レイテンシ
  - ロックステップを搭載，機能安全系での利用も想定
- FPGA



※ 図中のB_iは，
Simulinkモデルの
ブロックのIDとする．

HS-MBPの全体像


並列コード生成可能
Simulinkモデル
Embedded Coder
逐次コード
MBPツール群
BLXML
MBP並列コード
生成ツール
コア割当情報
MBP
APU向け
Cコード
通信チャネル
定義 (MDCOM)
RPU向け
Cコード
SDFファイル
SysGen
FPGA向け
Cコード
RTOS
コンフィグ
通信
ラッパ
I/F 記述
(MDCOM)
Bus
Connector
HDL
トップ
HLS ツール
HDL
コンパイラ
コンパイラ
論理合成ツール
APU向けバイナリ
RPU向けバイナリ
FPGA向けバイナリ
System
Builder

HS-MBP の構成

- MBP 環境 [2]
  - MATLAB/Simulink で作成されたSimulinkモデルから並列化コードを生成
- SystemBuilder [3]
  - 高位合成を利用したC言語HW/SW協調設計ツール
  - **RPU-FPGA間通信**をサポート
    - APU-FPGA間の通信は現在検討中
- MDCOM（Multi Domain Communication）[4]
  - 異なるOS・アーキテクチャのCPU間通信ライブラリ
  - **RPU-APU間通信**をサポート

[2] 鍾 他: 組込み制御システムに対するマルチコア向けモデルレベル自動並列化手法，情報処理学会論文誌，Vol. 59, No. 2, pp. 735–747 (2018).
[3] 本田 他:システムレベル設計環境: SystemBuilder，電子情報通信学会論文誌，Vol. 88,No. 2, pp. 163–174 (2005).
[4] 大竹 他: ヘテロジニアスプロセッサ向け通信ライブラリ MDCOM，情報処理学会研究報告，Vol. 2017-EMB-44, No. 34, pp. 1–6 (2017).

SystemBuilder

- C言語によって，HW/SWの協調設計が可能である.
- HW 側は，高位合成によって HDL を生成する.
  - CyberWorkBench (NEC，以下CWB)
  - eXCite（YXI）
  - Vivado HLS （Xillinx）

- SW 側は，ITRONまたは車載 RTOS 上で動作するアプリケーションとなる.

- HDLや，RTOSの利用方法への理解度が低くとも利用可能である.

高位合成（High Level Synthesis）

- CやC++などの逐次コードを用いてHW設計し，ハードウェア記述言語（HDL）を生成する技術
  - HWや並列化の知識は多少なりとも必要

- HLSの難度が高い点
  - 通常のコンパイラよりも型やポインタの扱いに厳しい.
    - CPU向けに書かれたソースコードが，そのままHLSをパスするとは考えられない.
  - HWや並列化の知識が必要となる
    - 無償ツールでは，最適化を自動で適用してくれない.

SystemBuilderの通信

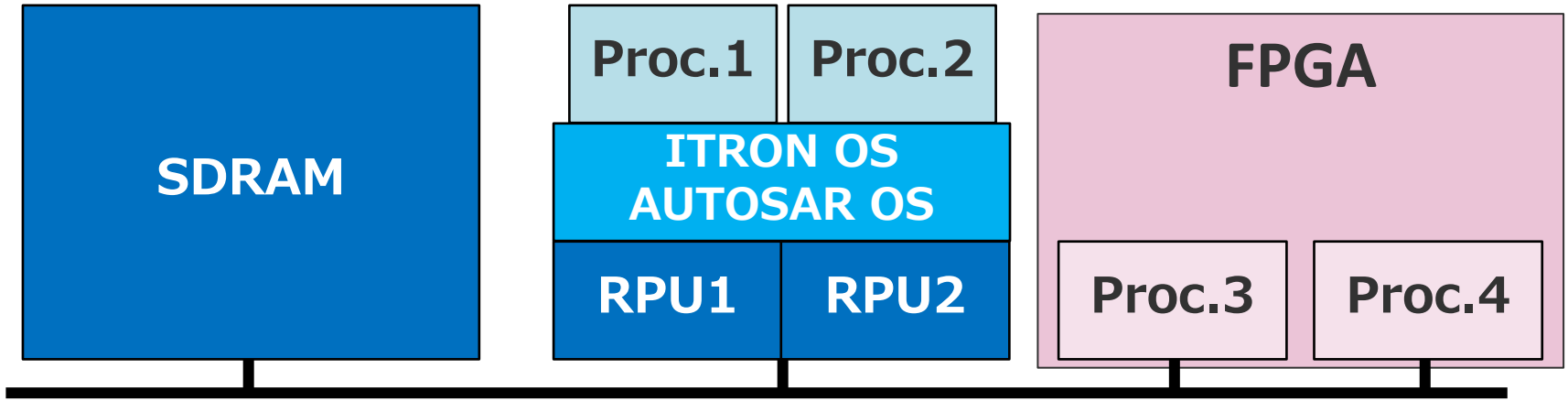
SystemBuilderは，HW-HW, HW-SW, SW-SW 間の以下の通信をサポートしている.

- BC (Blocking Channel)
  - FIFO
- NBC (Non-Blocking Channel)
  - レジスタ（単なる変数）
- MEM (MEMory Access)
  - アクセス先のメモリを静的に指定できる.
  - 実行時は，オフセット値をから特定アドレスのメモリにアクセスできる.

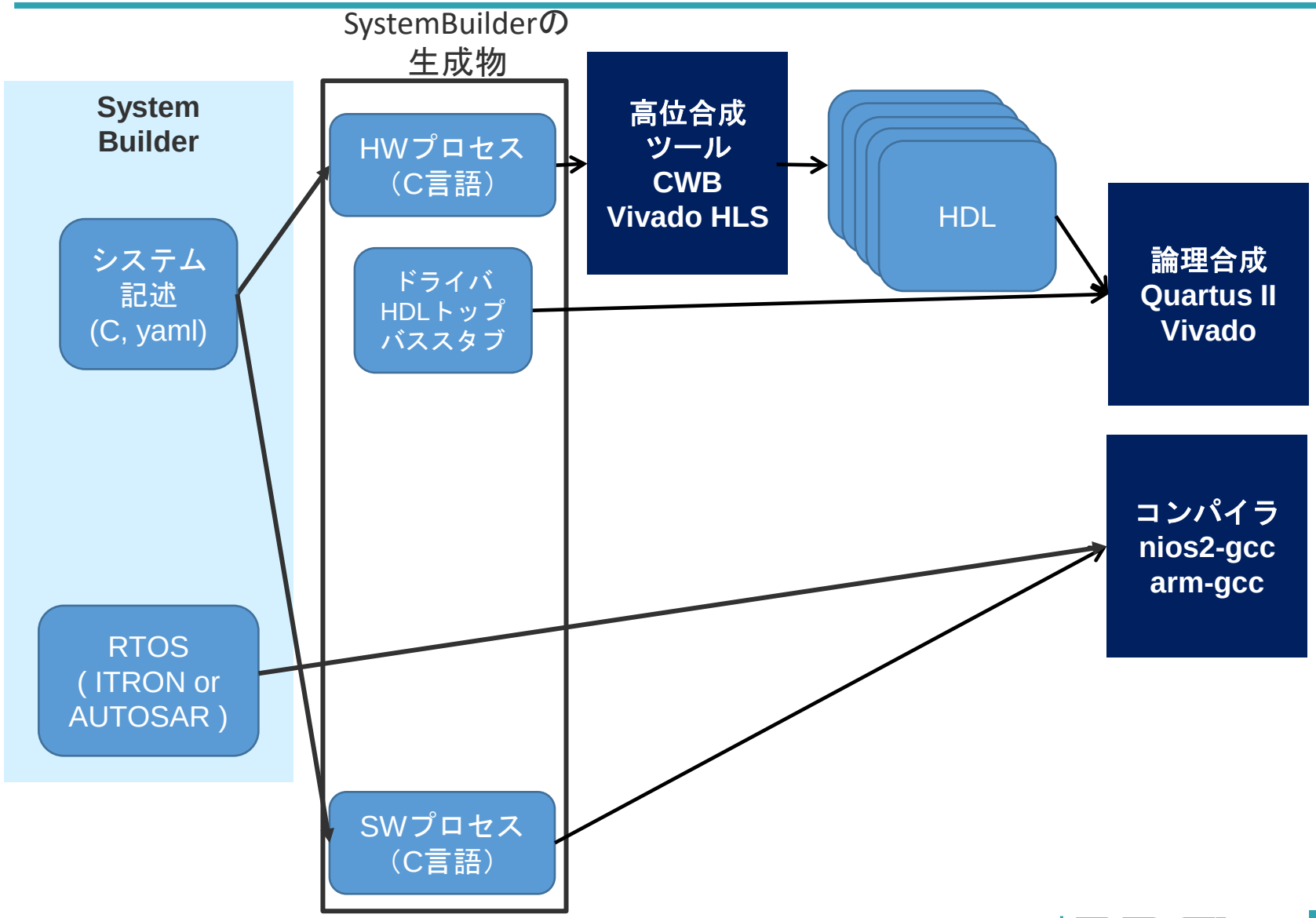
SystemBuilder の通信とターゲット環境例

SystemBuilderは，　HW-HW, HW-SW, SW-SW 間の以下の通信をサポートしている.

- BC (Blocking Channel) : FIFO
- NBC (Non-Blocking Channel): レジスタ
- MEM (MEMory Access) : アクセス先のアドレスを静的に指定



SystemBuilder の開発フロー



MDCOM

- POSIX-OS と TOPPERS/ATK2カーネル（車載RTOS）間通信ライブラリである.

- 通信チャネル
  - FIFOチャネル
  - SMEMチャネル
    - 共有メモリを用いたドメイン間通信
    - アクセス時にロックを取得して排他制御を実現

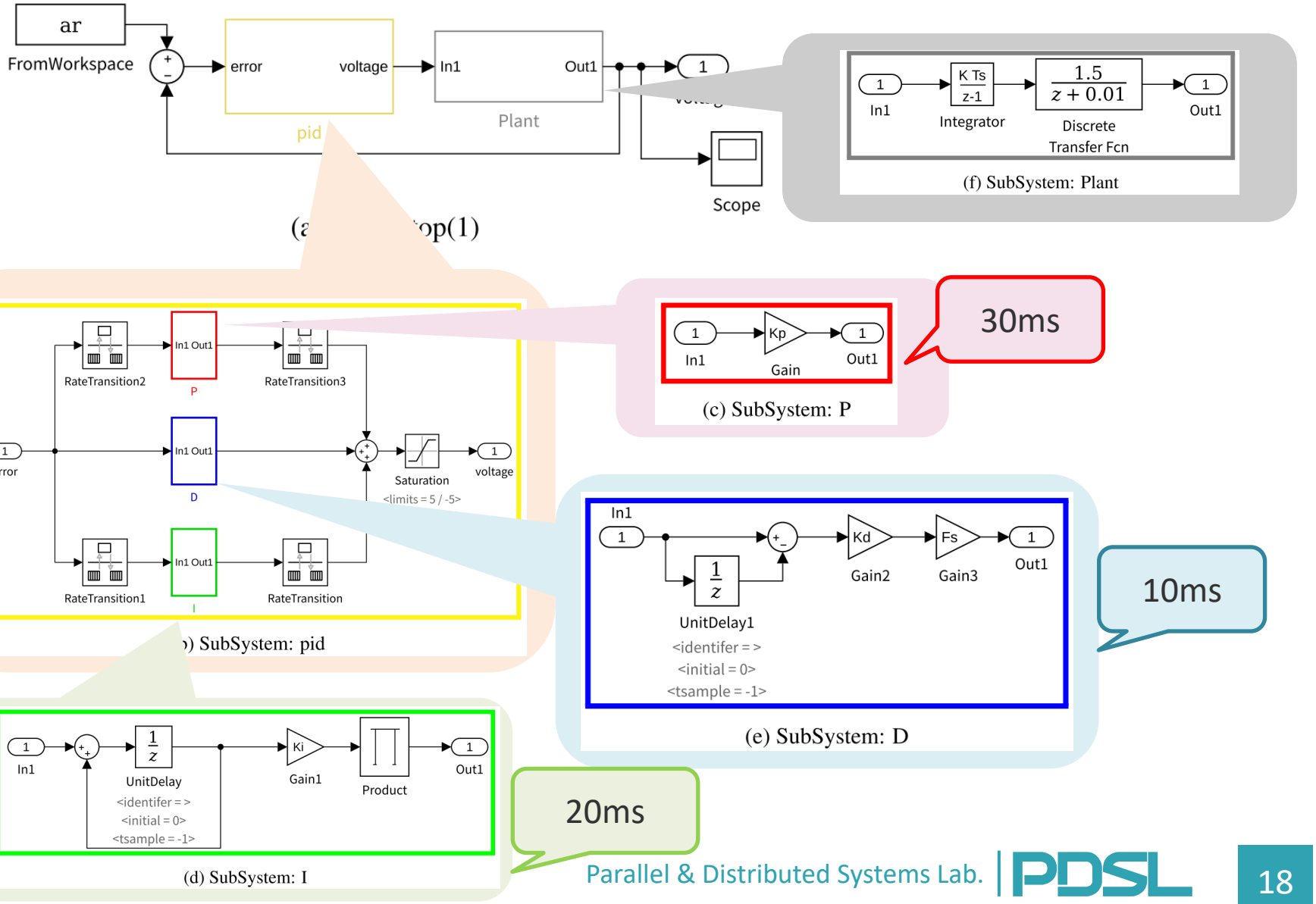
ケーススタディ

- Case study1: マルチレートの Simulink モデルを様々なパターンで各PEに配置してみる (Table1).
- Case study2: HWプロセスのチューニングをする

Table 1: Execution Patterns

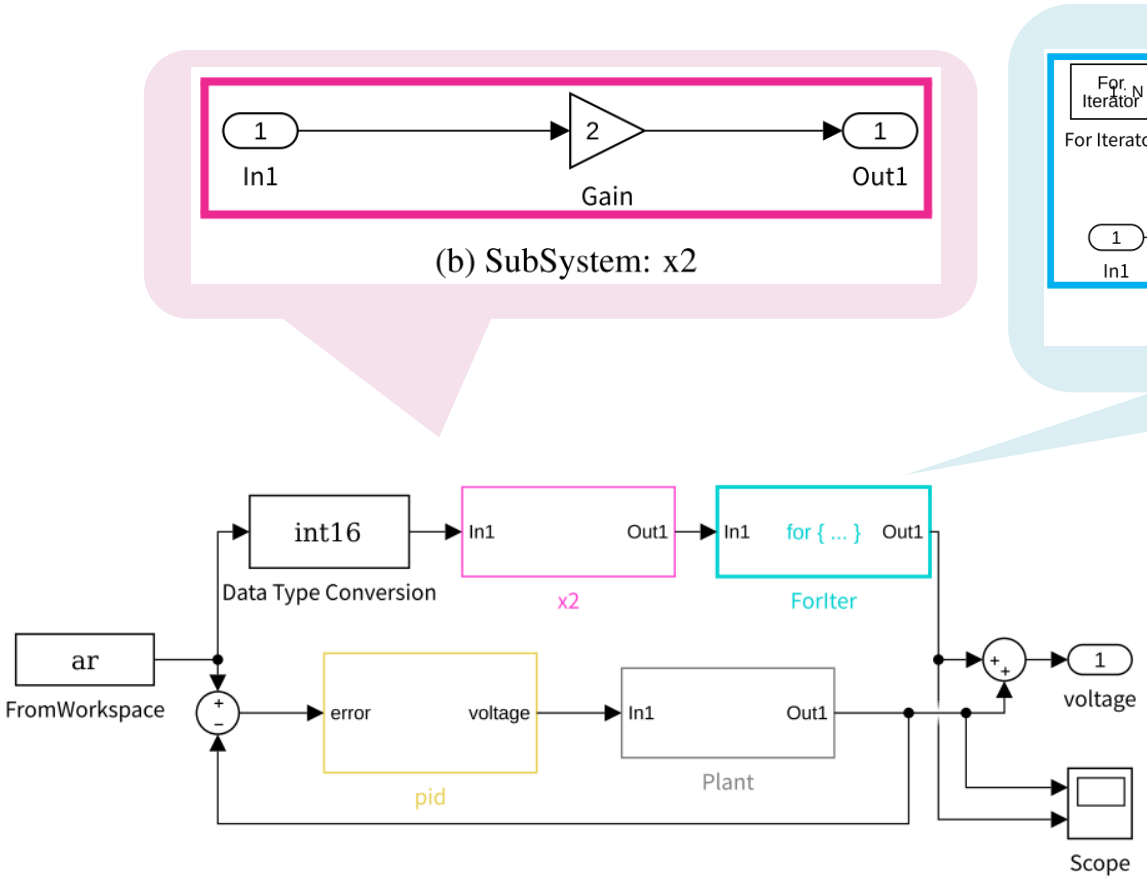
ID	Control Period		
	10ms(D)	20ms(I)	30ms (P)
C1	RPU0	RPU0	RPU0
C2	RPU0	RPU1	RPU1
C3	RPU0	APU0	APU0
C4	RPU0	APU0	APU1
C5	APU0	RPU0	RPU0
C6	RPU0	HW0	HW0
C7	RPU0	HW0	HW1
C8	HW0	RPU0	RPU0
C9	RPU0	APU0	HW0

Case Study1: Target Model1



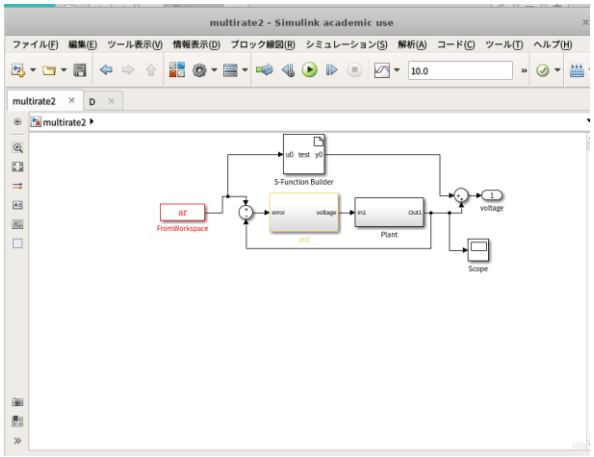
Case Study2: Target Model2

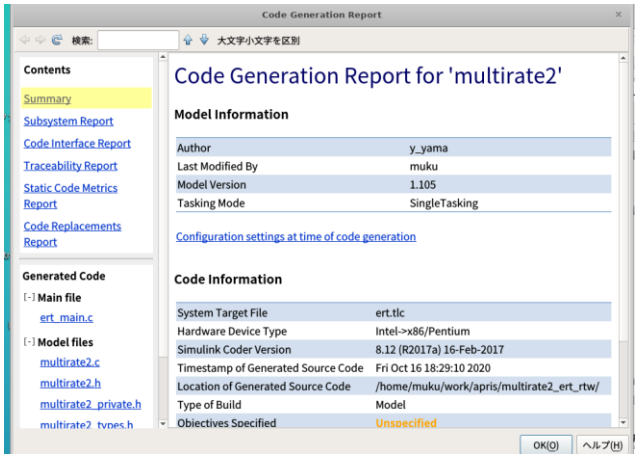
Target Model 1 にFPGA向けのブロックを追加

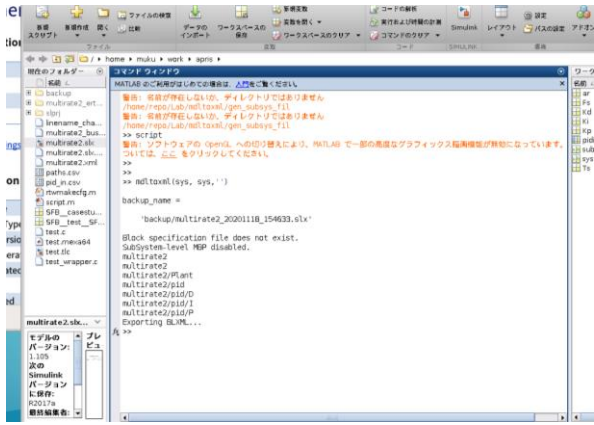


利用手順 (1/)

- モデルを作成後，Embedded Coder で逐次Cコードを生成
- スクリプトを用いて BLXML（ブロックレベル構造情報）を抽出

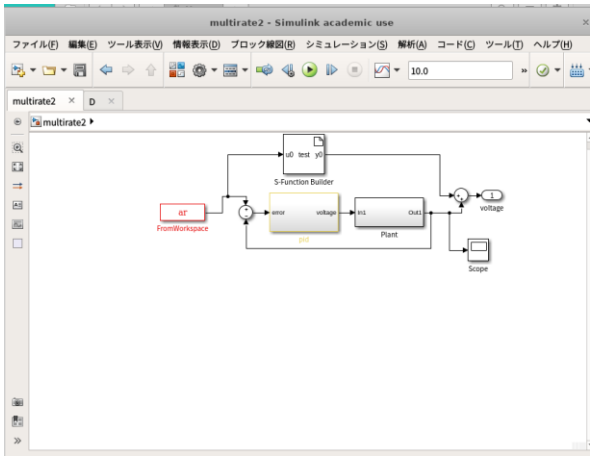


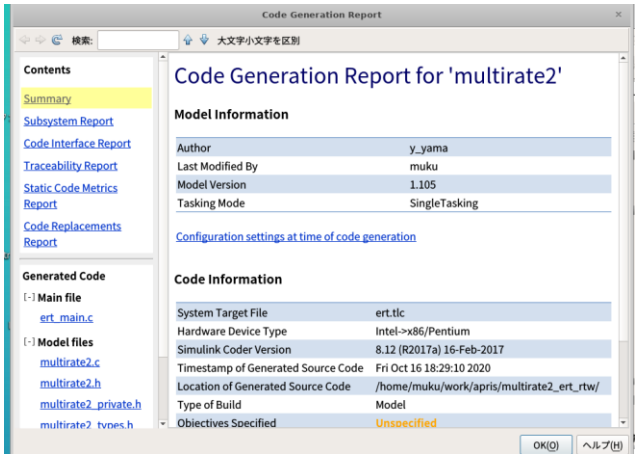


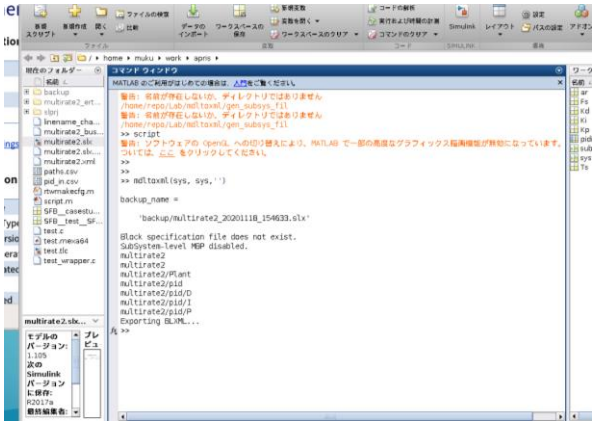


利用手順 (1/2)

- モデルを作成後，Embedded Coder で逐次Cコードを生成
- スクリプトを用いて BLXML（ブロックレベル構造情報）を抽出







利用手順 (2/2)

- MBPツールのコマンドでコード情報を埋め込む
- コア割当を（今回は）手動で行う.
- 並列コードを自動生成
  - コマンド例 (一部オプションを省略)

    csp_translator -s multirate2_ca.xml -T hsmbp
- ビルド（ビルドもコマンドラインでOK）
  - コマンド例

    hsmbp-builder -s multirate2 -T arf --all

Case Study1: Result

- 開発環境
  - CPU: AMD Ryzen 9 3950X 16-Core Processor @ 2.2GHz
  - RAM: 64GB
  - HLS tool: CWB 8.1
- すべてのパターンで，Simulinkモデルの
  シミュレーション結果と出力が同じになった.
- 作業時間は,
  - FPGAなしの場合，3分以下であった.
    - コンパイル時間と手動PE割当の時間が支配的
  - FPGAありの場合，10分程度であった.
    - HLSと論理合成の時間が支配的

Case Study2: Result

- ケーススタディの環境:
  - HLS ツール: CWB 8.1
  - CLK: 100MHz
- 最適化手法:
  loop folding
  - ループレベルの
    パイプライン化
  - 書き換え時間は
    1分未満
- 2665ns から 1530ns へ
  レイテンシが現象

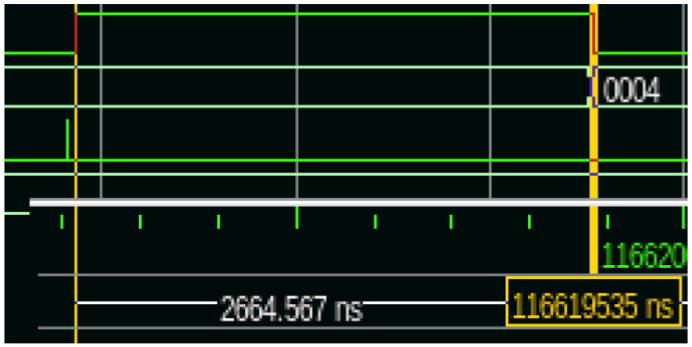


Fig. 12: HW latency without folding: 2664.567ns

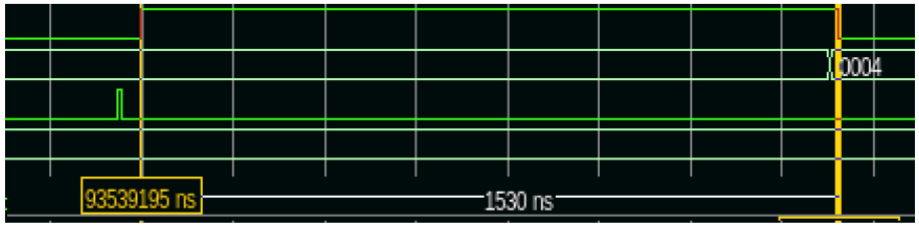


Fig. 13: HW latency with folding: 1530ns

考察: Case study1

- PE上での実行およびPE間の通信機構
  - もし，データ通信が不整合である場合，
    シミュレーション結果が一致しなくなる.

→ 通信機構をマルチレート向けに実装できており，
各PE上で正しくブロックを実行できている.

- 性能
  - マルチレート周期は守られている.
    - これも，結果が正しいことによる.

考察: Case study2

- チューニング結果
  - HWレイテンシを 2665ns から 1530ns に低減

- 作業時間
  - 1分未満で作業ができた
    - しかし，作業者はHLSに慣れている.
  - 小シミュレーションは，性能測定を支援した
    - However, low-price RTL simulator is slow compared to expensive one.
  - Tuning target has only simple “For iterator”.

ここまでのまとめ

- FP-HSoC向けのMBD環境を構築した
  - 通信機構と各PE上での実行をサポート
  - マルチレート対応

- HWのレイテンシ低減については今後も検討が必要
  - シンプルなターゲット以外でも試す必要がある

ヘテロジニアス向け開発フロー

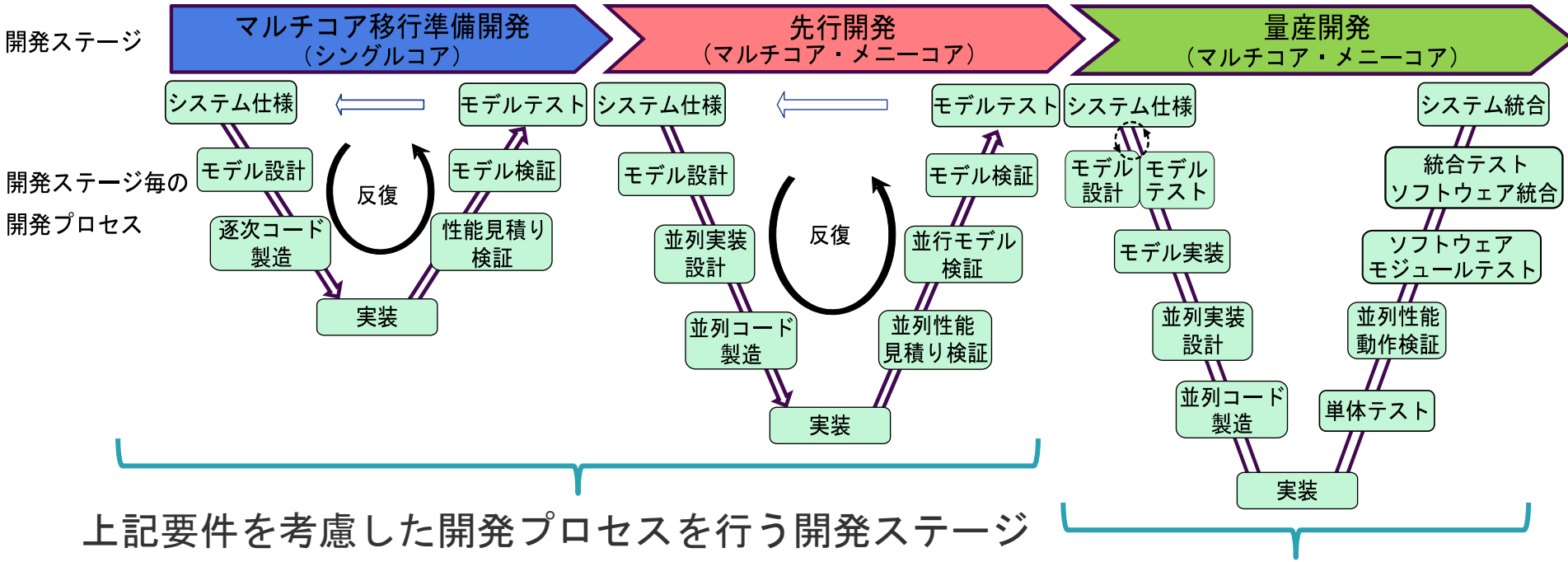
- モデルから，ヘテロジニアス向けの設計環境は完成しつつあるが…
  - ホモジニアス向けの開発と比べて何が違う？
  - FPGAを含めた開発と比べて何が違う？

- このツールを，実際の開発ではどのように利用するのか？

ホモジニアス向け開発フロー [1]

開発フローへの要求

- 開発の早い段階で性能見積が可能なこと
- 性能見積にもとづいた並列化性能設計が可能なこと
- 並列化特有の問題が生じることなく動作すること

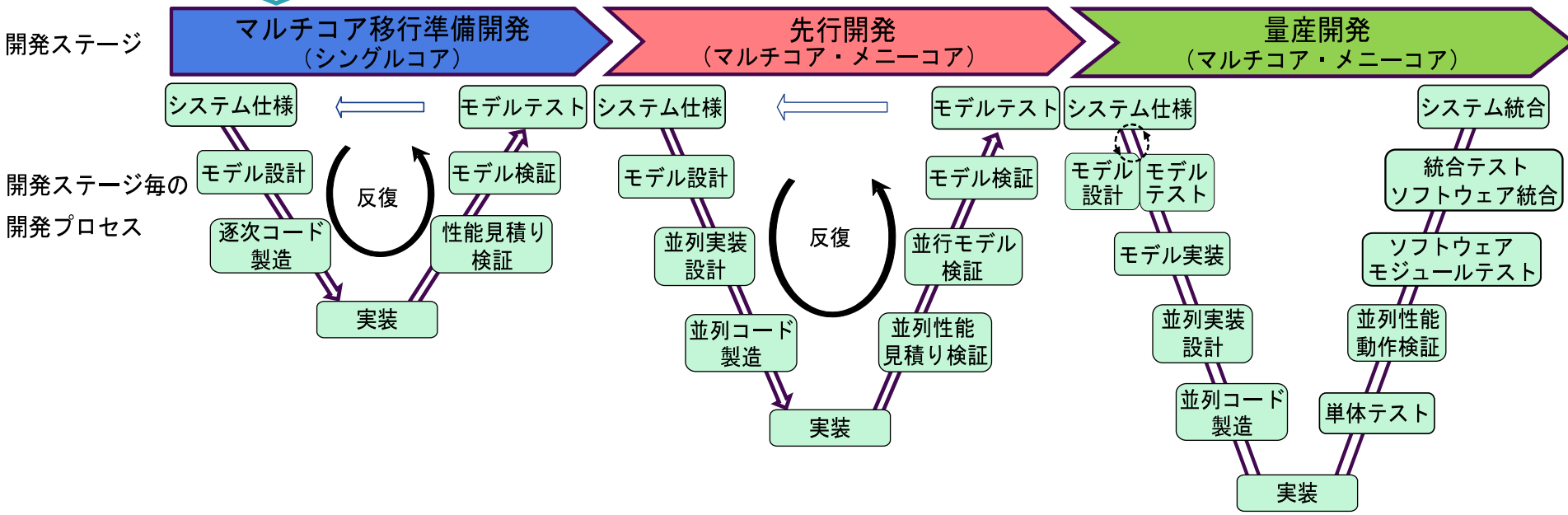


上記要件を考慮した開発プロセスを行う開発ステージ

課題1から課題4を解決した状態で行う開発ステージ

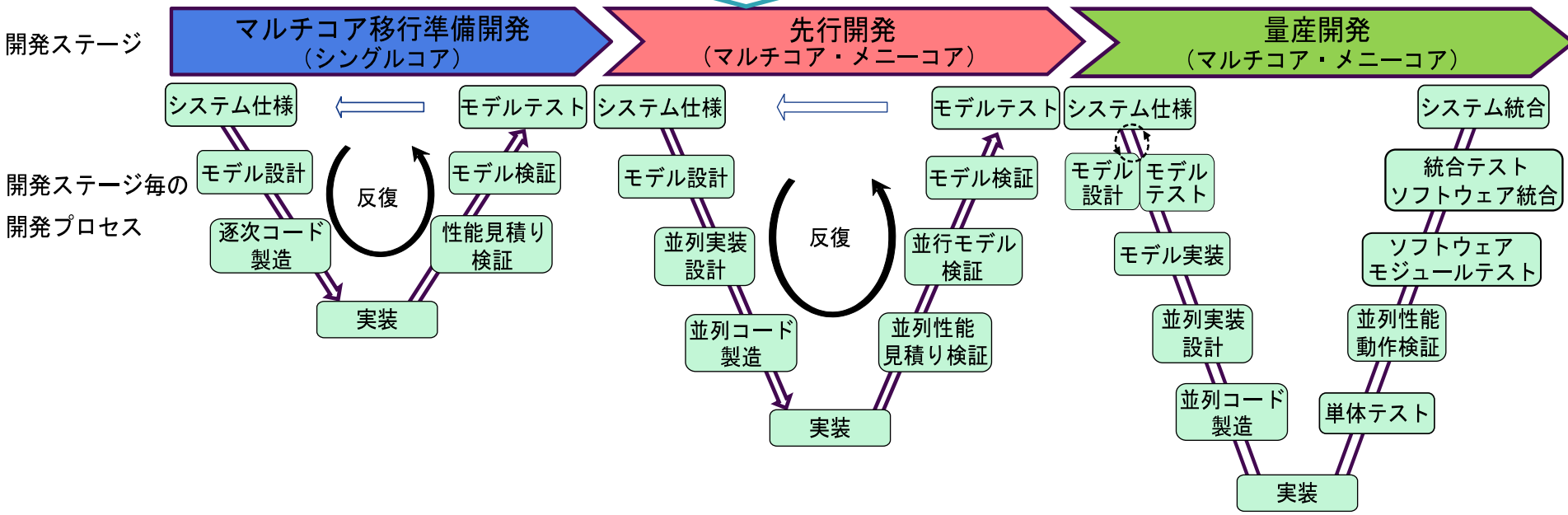
ヘテロジニアス向け(APU-RPU)開発フロー

「動くものを最初に作る」
という意味では,
このステージは変わらない.

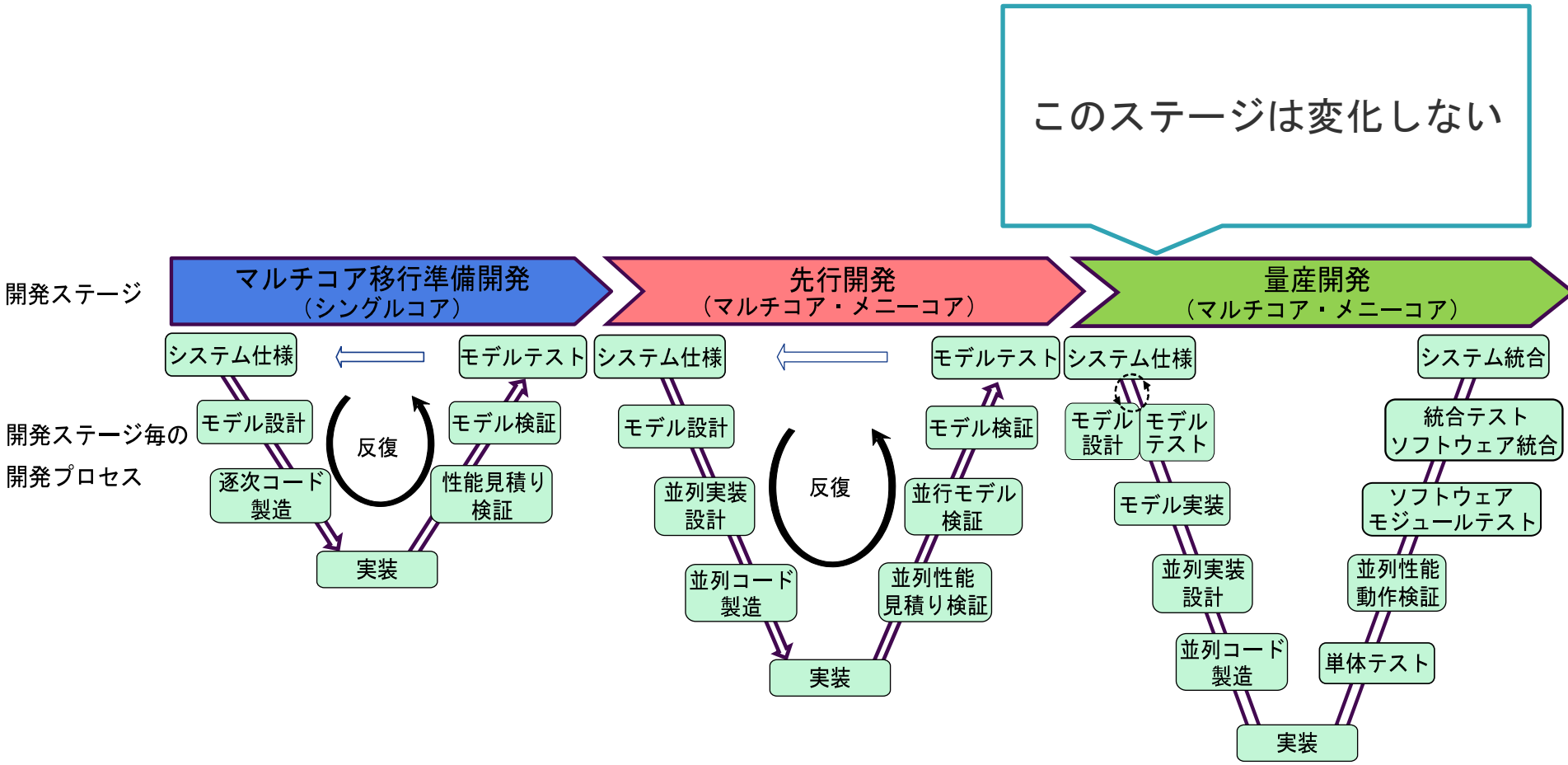


ヘテロジニアス向け(APU-RPU)開発フロー

どの種類のコアにどの実装を割り当てるかを
検討する必要があり，それも含めて
反復回数が増加する可能性がある



ヘテロジニアス向け(APU-RPU)開発フロー



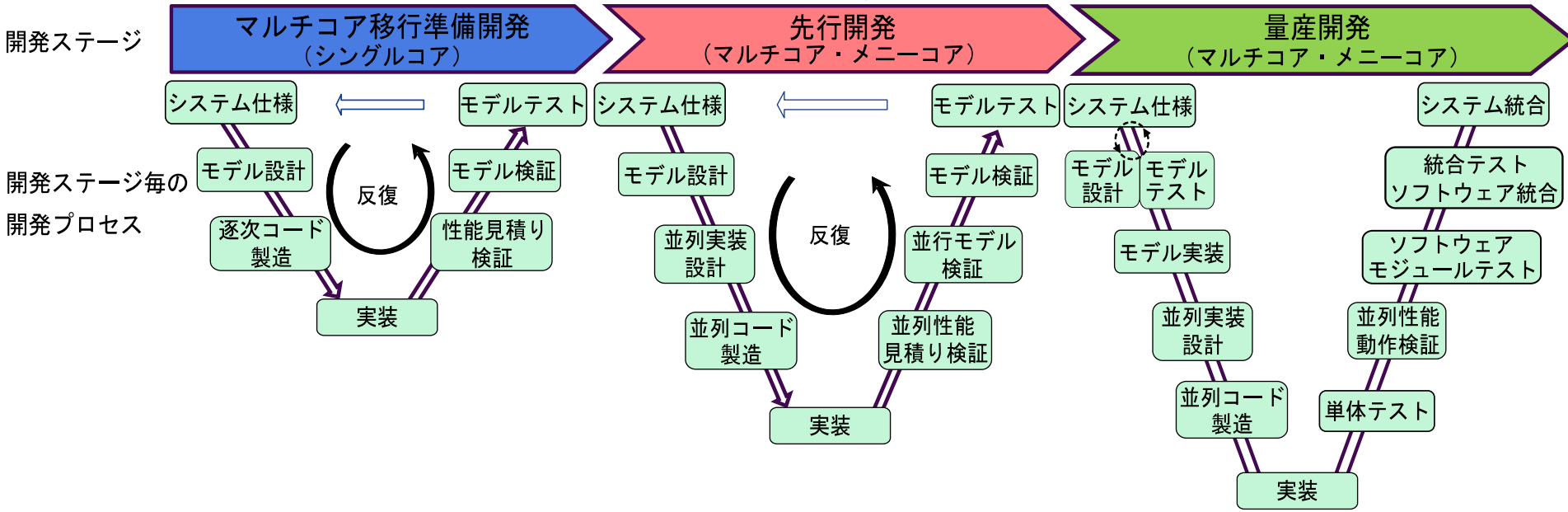
ヘテロジニアス向け(FPGA込)開発フロー

- FPGAの設計は，HLSを用いる前提

- そもそも，HLSを用いた開発フローとは？
  - 私の慣習としては，
  - 先に，CPUで動作するCを作成し，
  - HLSでエラーを生じないように書き換え，
  - 高速化するようにコードをチューニングする

ヘテロジニアス向け(FPGA込)開発フロー

このステージで，様々な割当を試行する必要があり，
実機による性能見積もりでは限界があるため，
コシミュレーションなどの工夫が重要．
イテレーション回数は多くなる．



開発フロー全般

- 様々な種類の演算ユニットをもつ場合は，開発プロセスを詳細化・具体化する必要がある．

- 定義をしたものの，現場で実証実験を行うことができていない．
  - ご協力いただける企業様を探しております．

まとめと今後の課題

- 通信機構を実装し，FP-HSoCの各PE向けに実行可能ファイルを生成し実行できた.
  - 通信機構の動作確認をすることができた.
  - 各PEでの実行の動作確認をすることができた.

- 性能や作業量の面では調査するべきことや，課題が残っている.
  - より詳細な実行時情報の調査・測定
  - シミュレーション環境の拡充
  - FPGAのチューニングのためのフローの検討