

FPGAを使った システム開発について考えよう

大川 猛 (宇都宮大学)

安藤友樹 (名古屋大学)

自己紹介

- 大川
 - 宇都宮大学工学研究科情報システム科学専攻 助教
 - 主な研究テーマはHW/SW協調設計（分散並列処理）
 - 普段はザイリンクスのFPGAを利用
 - Xilinx Zynqを対象とした研究を実施しています
- 安藤
 - 名古屋大学 組込みシステム研究センター研究員
 - 主な研究テーマはSW/HWコデザイン
 - 普段はアルテラのFPGAを利用
 - Altera SoCを対象とした研究を実施しています

いきなりですが質問です

- Q1. FPGAを利用したことがありますか？
- Q2. SWエンジニア or HWエンジニア？

背景

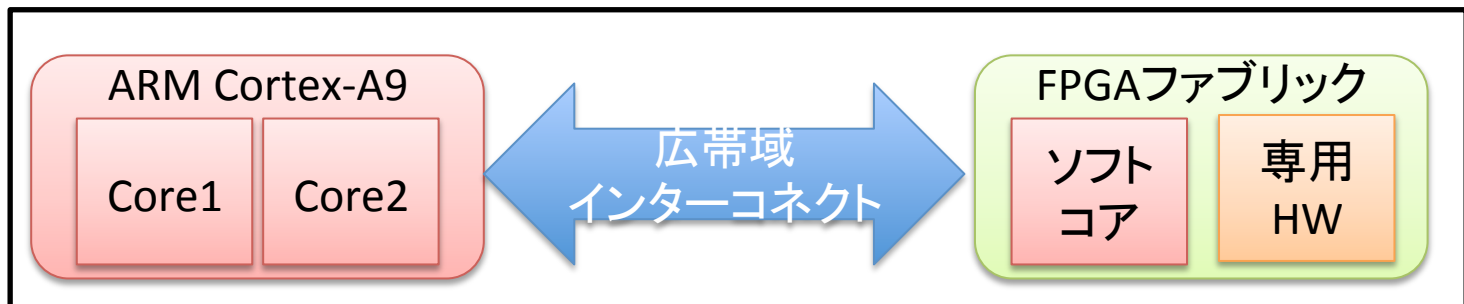
- 少品種大量の時代から多品種少量の時代へ
 - 一部の製品を除き、ASICの利用が困難に
 - 医療、宇宙、通信設備でのFPGA利用率は高い
 - 車載システムでのFPGA利用が始まっている
- FPGAでSoCを実現したい
 - ARM+FPGAの登場
- ハードウェアのアップデートをしたい

FPGAとは

- Filed Programmable Gate Arrayの略
- 書き換え可能な（再構成可能な）ハードウェア
- 利点
 - 再構成可能
 - 修正可能（SWのバージョンアップのように）
 - 少量多品種の製品開発が可能
- 欠点
 - 専用チップと比較すると性能では劣る
 - 消費電力
 - 周波数
 - 面積効率

最近ホットなFPGA：ARM+FPGA

- ARMコアとFPGAロジックを1チップに統合
- 特徴
 - ARMコアとFPGAロジックは広帯域バスで接続
 - 高性能
 - ARM：800MHz
 - FPGAロジック：200MHz
 - ヘテロマルチコアの構成が可能
 - ARM+ソフトコア（Nios II/MicroBlaze）+専用HW



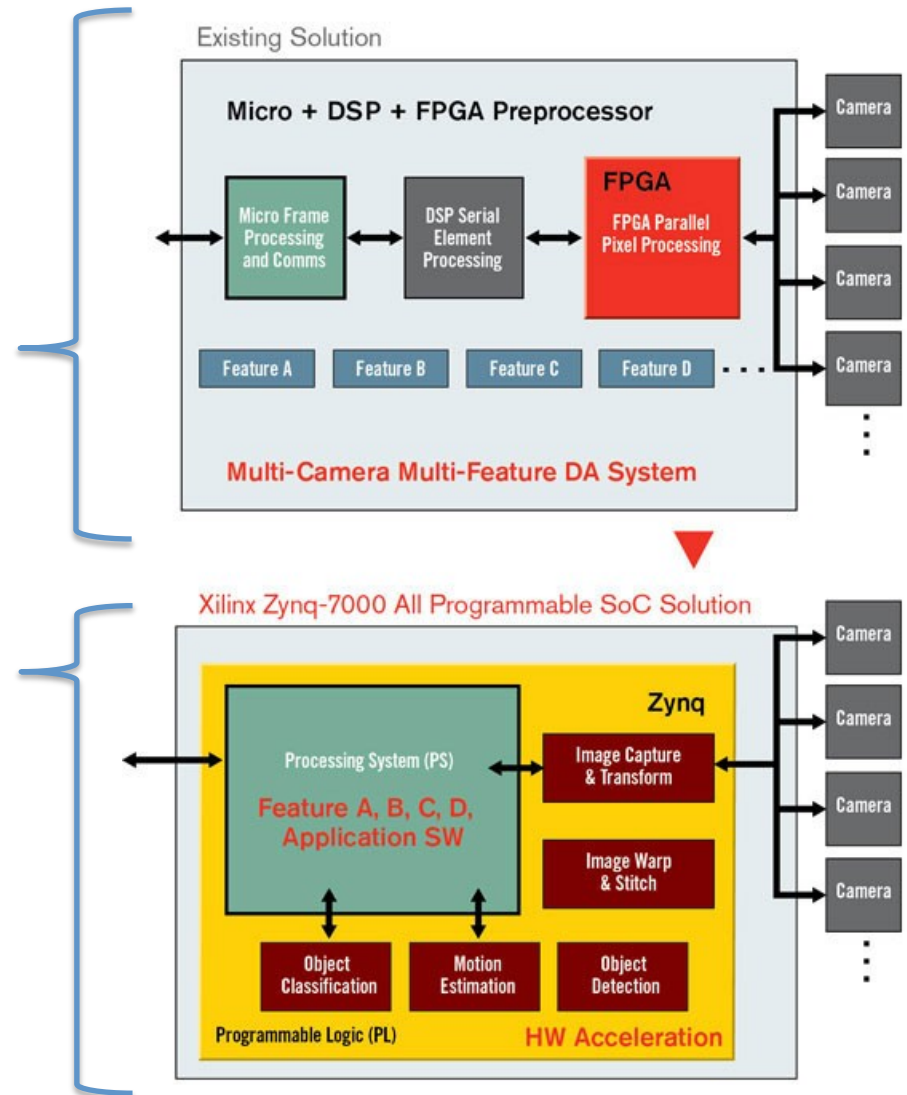
ARM+FPGAの利用事例①

- フェノクス (Phenox)
 - Webサイト : <http://phenoxlab.com/>
 - 超小型自立飛行ドローン
 - Xilinx Zynqを利用
 - ARM : 専用APIを用いたLinuxアプリケーション開発
 - OpenCVを利用可能
 - FPGA : 画像処理、制御処理
 - FPGAのみで飛行制御可能
 - 認識処理、飛行方法などをLinuxアプリとして実装



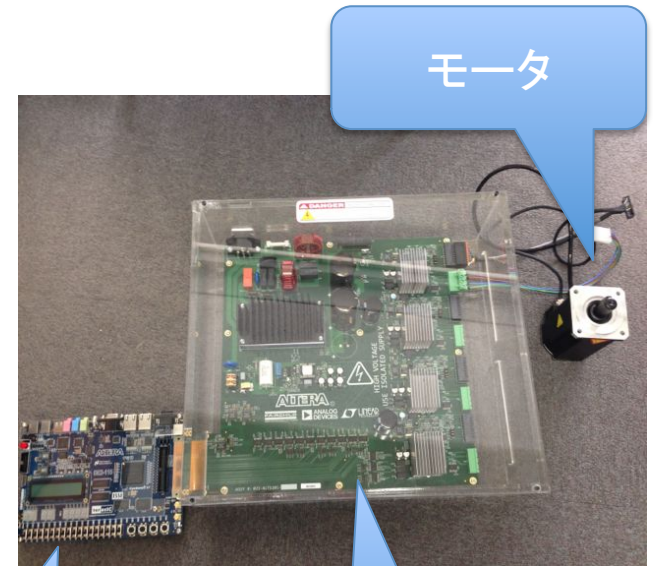
ARM+FPGAの利用事例②

- 運転支援システム
 - Xilinx Zynqを使用
 - 従来：3チップで構成
 - プロセッサ
 - DSP
 - FPGA（画像処理）
 - Zynq1つで構成可能
 - ARM：SWとDSP
 - FPGA画像処理



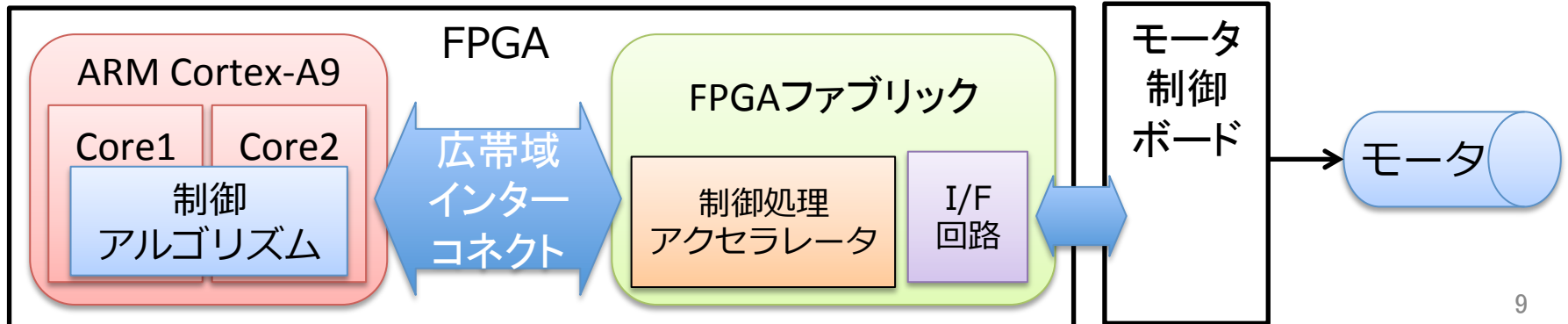
ARM+FPGAの利用事例③

- モータ制御システム
 - Altera SoCを利用
 - ARM
 - モータからの割り込み処理
 - 入出力処理
 - FPGA
 - 制御処理のアクセラレータ
 - モータ制御ボードとのI/F回路



FPGA

モータ
制御ボード



FPGAを利用したシステム設計フロー：概要

- STEP1：アーキテクチャ構成の決定
 - コアの種類と数（ARM/ソフトコア）
 - バスの種類と数
 - メモリの種類と数 など
- STEP2：ソフトウェア&ハードウェア設計
 - SW設計
 - HW設計
 - 通信設計
- STEP3：コシミュレーション
- STEP4：合成と実行（テスト）

STEP1：アーキテクチャ構成の決定

- FPGAベンダのGUIツールが充実している
 - Altera社：Qsys
 - Xilinx社：Vivado
- 各社ともに基本的なIPが用意されており、選択して接続するだけでSoC開発が可能
 - ソフトコアプロセッサ（マルチコア可能）
 - タイマ
 - メモリ
 - IF回路
 - PLL
 - DMAなど

Altera社のツール：Qsys

割込み
の設定

バスの
接続設定

ユーザ
作成の
IP

Altera提供のIP
ライブラリ

IPのベース
アドレス設定

ユーザ作成
のIPの設定

The screenshot displays the Altera Qsys tool interface. The 'System Contents' pane on the left shows a project hierarchy with 'user' and 'sp_hw' highlighted. The 'Connections' pane in the center shows a network diagram of components. The 'Project Settings' pane on the right shows a table of components and their base addresses.

Name	Clock	Base	IRQ
clk_50	exported	0x0000_1900	
altpll_sys	clk_50	0x7000_0000	
sdram	altpll_sys	0x0a00_0000	
sram	altpll_sys	0x1000_0000	
clock_crossing_io	altpll_sys	0x0000_0030	
led	altpll_sys	0x0000_0040	
sw	altpll_sys	0x0000_0060	
sysid	altpll_sys	0x0801_0000	
cpu	altpll_sys	0x0801_0800	
timer	altpll_sys	0x0801_0820	
jtag_uart	altpll_sys	0x0800_0000	
uart_0	altpll_sys	0x0600_0000	
sp_hw_0	altpll_sys	0x0500_0000	
clk_1	altpll_sys	0x0500_1000	
avalon_S1	altpll_sys	0x0500_2000	
clk_m_1	altpll_sys	0x0500_3000	
avalon_M1	altpll_sys	0x0500_4000	
clk_m_2	altpll_sys		
avalon_M2	altpll_sys		
clk_m_3	altpll_sys		
avalon_M3	altpll_sys		
clk_m_4	altpll_sys		
avalon_M4	altpll_sys		
clk_m_5	altpll_sys		
avalon_M5	altpll_sys		
clk_m_6	altpll_sys		
avalon_M6	altpll_sys		
clk_m_7	altpll_sys		
avalon_M7	altpll_sys		
clk_m_8	altpll_sys		
avalon_M8	altpll_sys		
avalon_S1_irq	altpll_sys		
onchip_memory2_0	altpll_sys		
onchip_memory2_1	altpll_sys		
onchip_memory2_2	altpll_sys		
onchip_memory2_3	altpll_sys		
onchip_memory2_4	altpll_sys		

デザイン
フロー

モジュール
階層

Xilinx社のツール : Vivado

Zynq全体

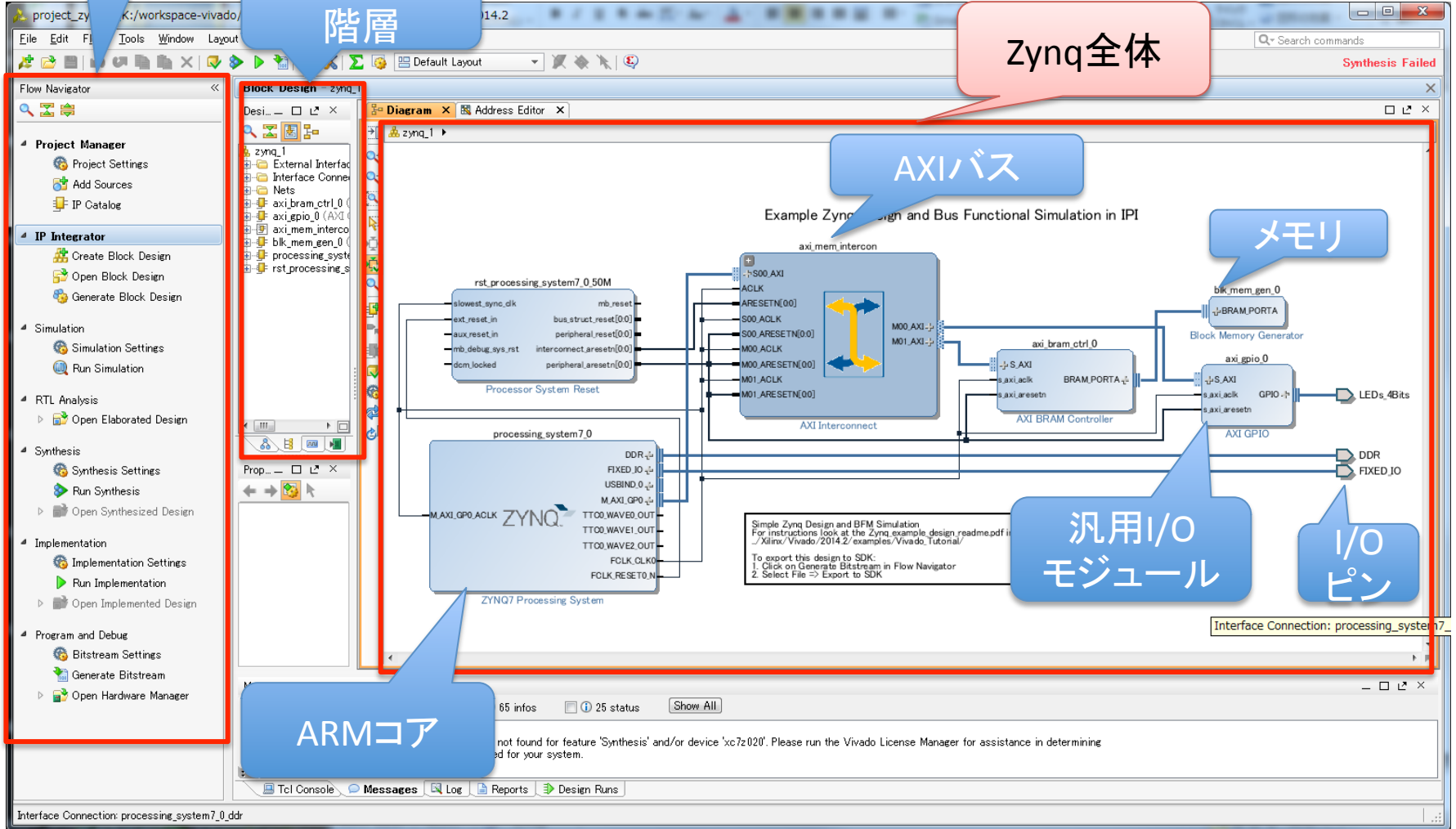
AXIバス

メモリ

汎用I/O
モジュール

I/O
ピン

ARMコア



STEP2：ソフトウェア&ハードウェア開発

- ソフトウェア開発
 - OSを利用
 - Linux
 - リアルタイムOS
 - ベアメタル
- ハードウェア開発
 - HDL設計
 - 高位合成ツール（C言語→HDLコンパイラ）
 - 最近の高位合成ツールは実用に近づいている
 - Xilinx社：Vivado HLS
 - Altera社：高位合成ツールはないが、OpenCLに対応

STEP2 : ソフトウェア&ハードウェア開発

• 通信設計

– SW-SW間通信

- シングルコア : グローバル変数
- ホモマルチコア : OSのAPIを利用
- ヘテロマルチコア : OSはサポートできない

– アプリレベルでの通信設計が必要

– 共有メモリ、コア間割込みがあれば可能

– SW-HW間通信

- HW : I/F回路の設計
- SW : HW側のI/Fへアクセスするドライバが必要
- ドライバとI/F回路生成をするツールも存在する

従来 : SW-HW間通信が課題



現在 : ヘテロコア間の通信が課題

大きな課題

STEP3 : コシミュレーション

- SWのシミュレーション
 - 所謂シミュレータを利用
 - 命令セットシミュレータなど
- HWのシミュレーション
 - HDLシミュレータを利用
- SWシミュレータとHWシミュレータ間の通信
 - Verilog PLIなど、他のプログラムと通信する仕組みを利用する

STEP4 : 合成 & 実行

- SW
 - コンパイラを利用
- HW
 - FPGAベンダの合成ツールを利用
 - Altera社 : Quartus II
 - Xilinx社 : Vivado
 - シミュレーションはOKだが実機で動作しない場合も
 - クロック周波数の制約
 - ピンアサイン
 - IPのベースアドレス設定
 - FPGAだかこそ生じる問題

FPGAシステム設計の課題

- SW設計、HW設計、通信設計を行う必要がある
 - 部分部分ではツールのサポートがある
 - 全体を網羅したツールはない
 - 設計者の技術に依存する部分が多い
- ヘテロマルチプロセッサ構成が可能
 - ARM+FPGAで、ハードコア+ソフトコアを実現
 - ヘテロマルチコア向けの通信設計は新たな課題

ワーク

- チーム作成 5分
- Q1 3分
- Q2 3分
- Q3 4分
- ポスター作成 10分
- ポスター発表とQ&A 15分
- 予備 5分

チーム作成 (5分)

- FPGA経験年数
- 組み込み経験年数

に基づいて、チームを作ります。
(シャッフル)

→作成したら所属・名前・経験年数を自己紹介

ワークのルール

- アイディアの否定は禁止

Q1) FPGAに期待すること (3分)

- 自由にデジタル回路を作れるFPGAを使って何をしたいですか？ソフトウェアではできないような、どんな事ができると思いますか？
- 以下のようなキーワードを5個以上あげる
 - 要素部品
 - システム・アプリケーション
 - 抽象的なイメージ

Q2) FPGAでは出来ないこと (3分)

- FPGAで設計実装するのは難しい、ソフトウェアで作った方がよい、と思われるシステムってどんなものですか？
- (Q1同様) 以下のようなキーワードを5個以上
 - 要素部品
 - システム・アプリケーション
 - 抽象的なイメージ

Q3) ARM+FPGAで何をする？ (4分)

- Q1のアイデアをFPGAで
- Q2のアイデアをARMプロセッサで

分担してやらせれば、今までにない新しいものが出来るはず！

(Q1,Q2の新しいアイデアも追加OK)

ポスター作成 (10分)

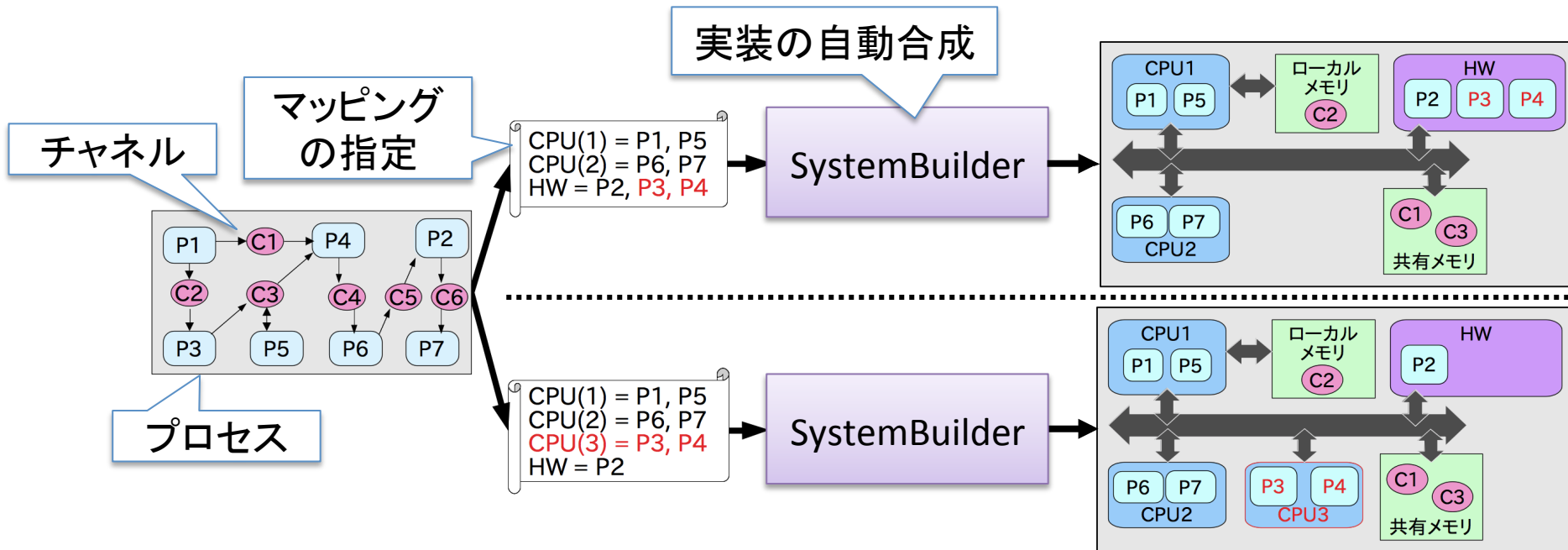
- Q3のアイデアを使って、ARM+FPGAを使った新製品のポスターをつくってみましょう！

ポスター発表とQ&A (15分)

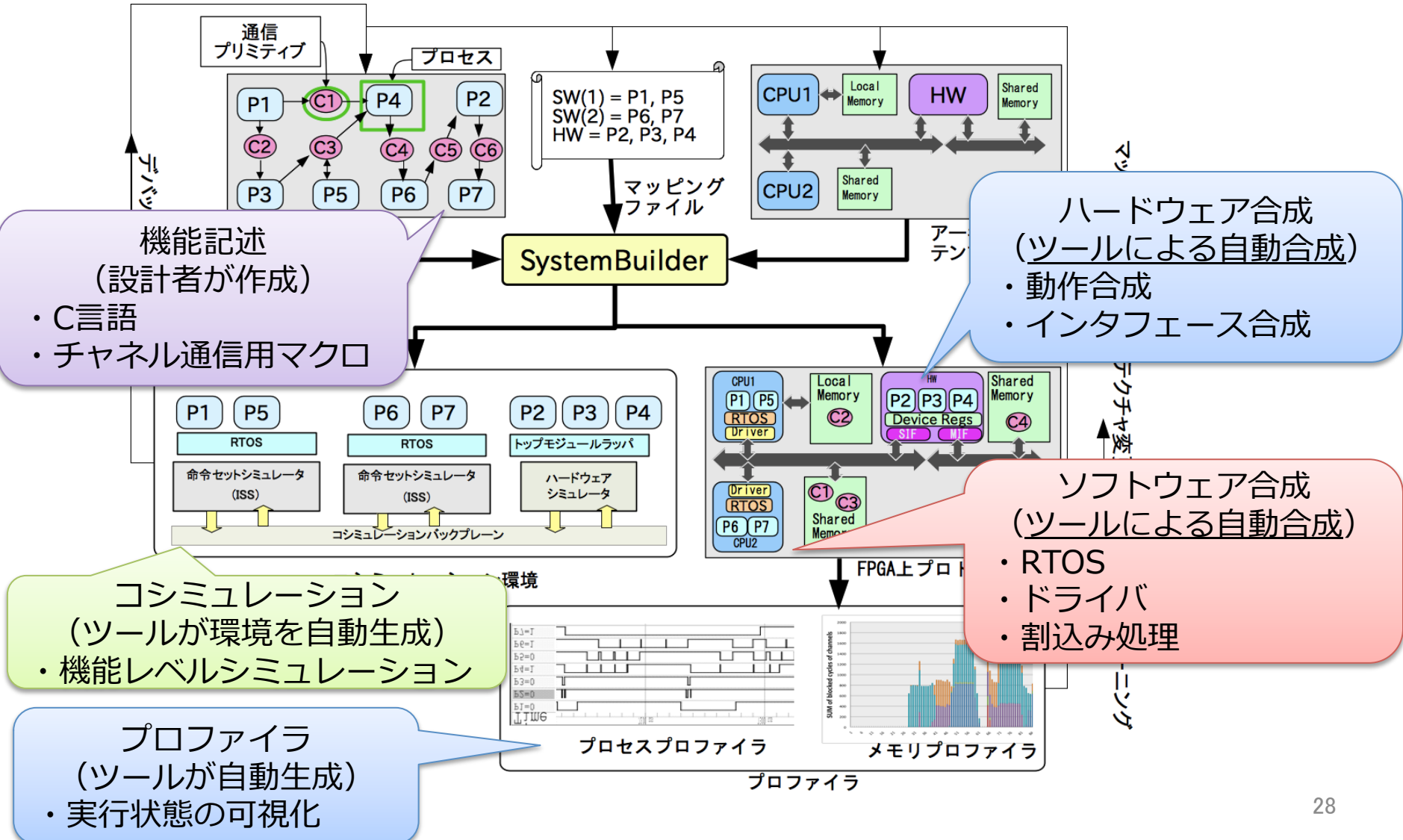
- 新製品のポスター発表を聞いて、Q&Aしましょう
- 実際の開発にはどんな課題がありますか？
- 課題を解決する開発環境はどのようなものですか？

1つの解：システムレベル設計ツール

- SW/HWを意識せず、システムをプロセス（機能）とチャネル（通信）の集合として記述
- プロセスのマッピング（SW/HW分割）に従った実装を自動合成
- 人手による実装と比べ、様々なマッピングを**短期間に評価可能**



ツールを用いた設計フロー



機能記述 (設計者が作成)

- C言語
- チャネル通信用マクロ

ハードウェア合成 (ツールによる自動合成)

- 動作合成
- インタフェース合成

ソフトウェア合成 (ツールによる自動合成)

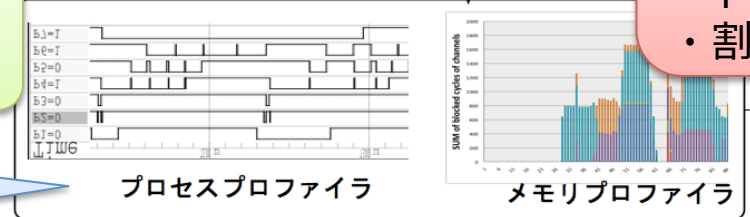
- RTOS
- ドライバ
- 割り込み処理

コシミュレーション (ツールが環境を自動生成)

- 機能レベルシミュレーション

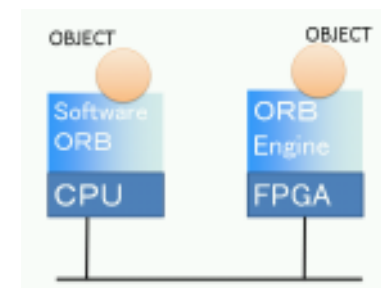
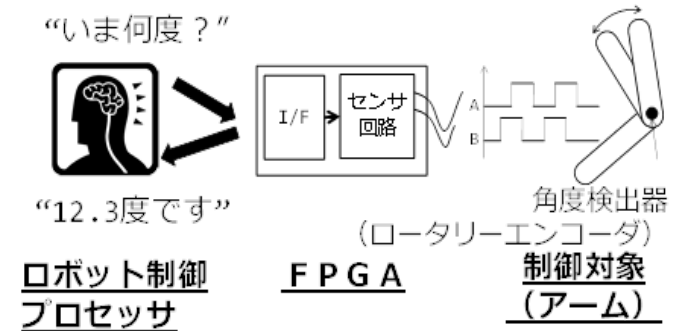
プロファイラ (ツールが自動生成)

- 実行状態の可視化



高抽象度で操作可能に： FPGAを分散オブジェクト化

- 遠隔のオブジェクト間で、メソッド呼び出しを記述
 - Client: メッセージ生成・送信
 - Server: メッセージ受信・解釈
 - 回路動作 (例: センサ値取得)
 - 返信メッセージ生成・送信
 - Client: メッセージ受信・解釈
- 共通のメッセージを使えば、実装言語は何でも良い！
(Java、C++、Ruby・・・)



CORBA分散オブジェクトを用いた設計事例 倒立振り子ロボット

倒立振り子ロボット

– 直立するよう制御

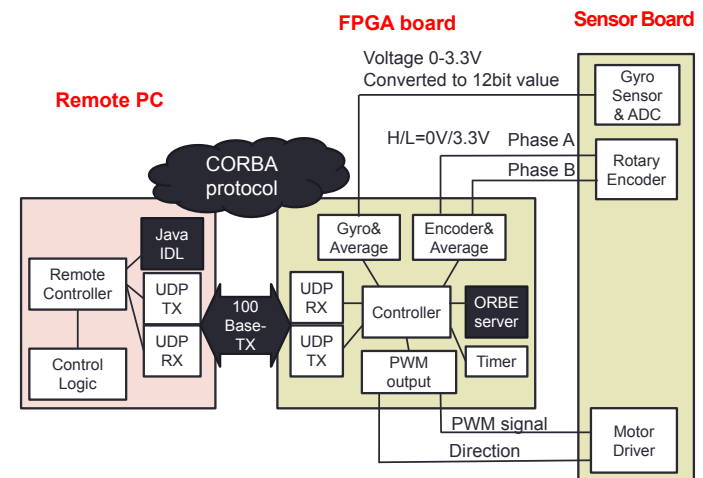
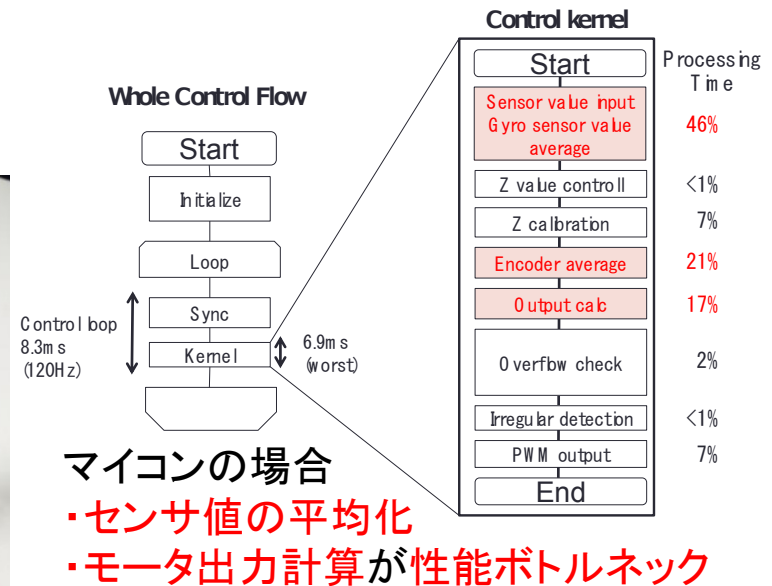
- ジャイロセンサ
(角加速度検出)
- エンコーダ
(回転検出)
- モータ出力



– センサ・アクチュエータを
オブジェクトとして扱う

– 制御論理を

- ソフトウェアにしてデバッグ
- **FPGA化して高速化**



まとめ

- ARMハードコアの性能 + FPGAの柔軟性
 - 必要に応じてHWを実現し、処理を並列実行可能
 - 性能向上が期待できる
- SWだけでなくHWも変更可能なFPGAの課題
 - プアな開発環境
- なんでもできる夢のデバイス
 - マイコンだけでない魅力的なチップ
 - どのような製品を開発しますか？
 - 研究で利用
 - 新しいアルゴリズムをいち早く高速化