2014年度後期　システム設計演習I（PBL）「FPGAボードを操作するANDROIDアプリ開発プロジェクト」

宇都宮大学　大学院工学研究科
情報システム科学専攻
大川　猛

2014年度後期　システム設計演習

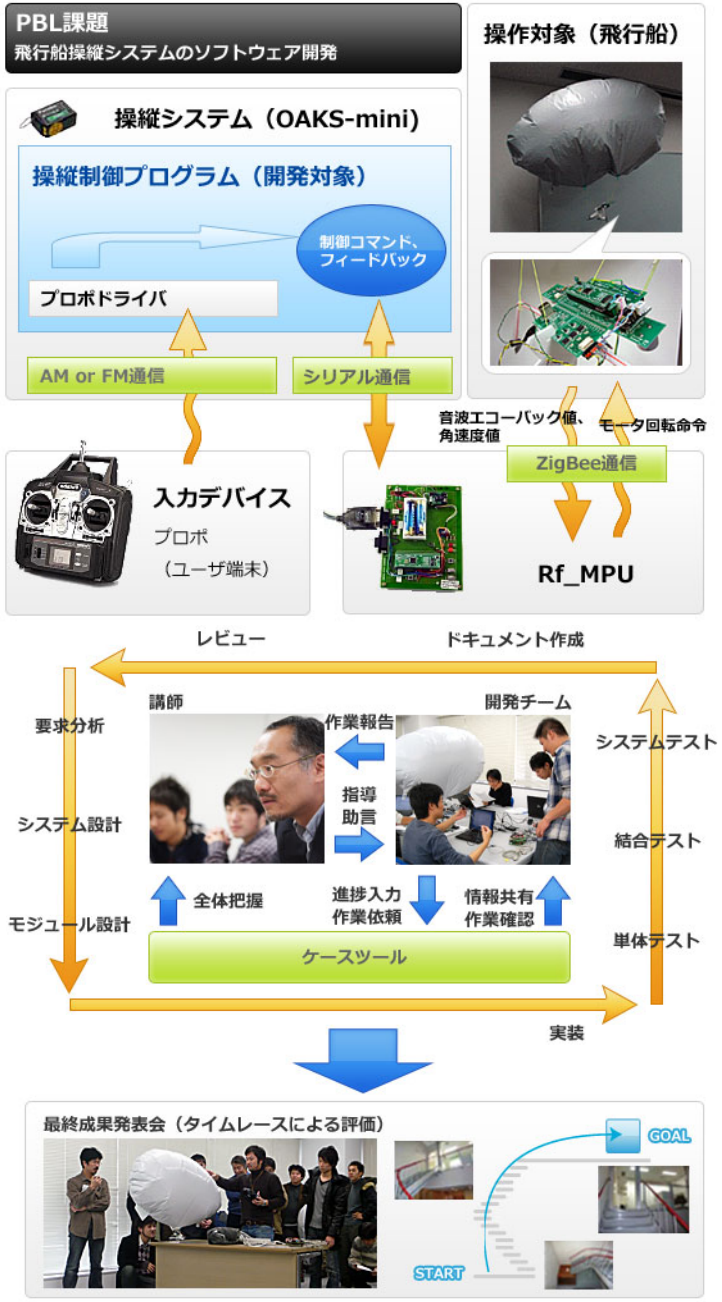
- 対象学年
  - 情報工学科３年次
- 【授業の到達目標】（シラバスより）
  - ある程度の規模のソフトウェアやハードウェアの設計や試作，解析等を行い，情報処理システムの計画から完成までの一連の過程を経験することにより，情報処理システムの基本的な原理・構成を把握し理解する。
- 【前提とする知識，関連する科目等】（シラバスより）
  - プログラミング入門 I，II，プログラミング演習 I，II，III，および情報工学実験 I，IIを履修していること。

2013年度後期　システム設計演習

- 【授業の具体的な進め方】　（シラバスより）
  各教員からテーマが設定される。受講者は，希望のテーマに沿ったプロジェクトを具体的に計画し，製作に取り組む。
- 本科目は **Project Based Learning**形式をとっており，具体的な内容の設定，事例調査，問題点の洗い出し，解決策の検討，実験と検証，考察と改良などを経て，情報処理システムの設計から完成までに取り組むことになる。
- また成果のプレゼンテーションとレポートの作成を通じて，コミュニケーション能力と自分の意見を的確に伝える能力の向上を図る.

PBLとは （１）

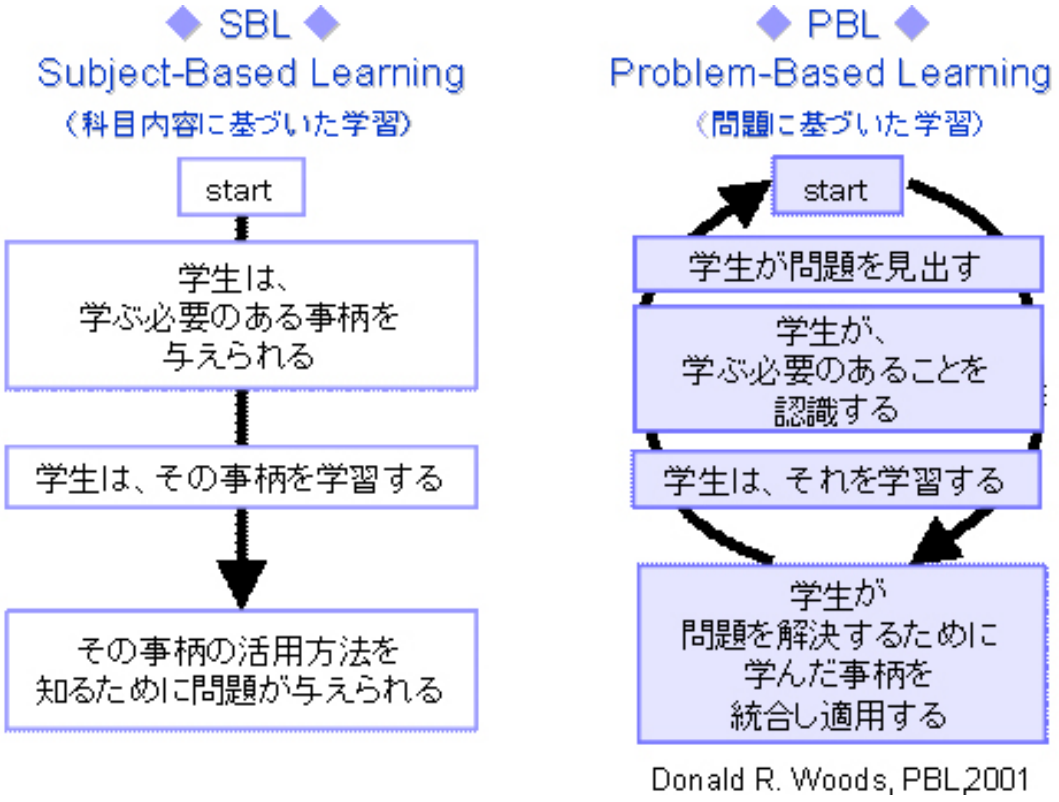
- PBL(Project-Based Learning)は、プロジェクト（演習）を通した学習の方法論で、実践的な知識の獲得や体験による対象の深い理解を目的としています。
- プロジェクトは、デザイン・問題解決・意思決定・調査を含むものです。学習者には長時間自律して課題に取り組む機会が与えられ、最終的にその成果物（プレゼンテーション）を生み出すことが求められます。
- 指導者は「先生」ではなく、「ファシリテータ」と呼ばれ、知識の提供は最小限にとどめられます。



出典：http://www.ocean.is.nagoya-u.ac.jp/Course/PBL/

PBLとは （2）

- PBLは，Problem-Based Learningの略であり，「問題に基づいた学習」という意味です。



出典：http://www.pbl.toyama-nct.ac.jp/?page_id=3

課題

- 「FPGAボードを操作するAndroidアプリ開発プロジェクト」
  - AndroidとFPGAを繋いで何か面白いものを作りましょう
- システムを完成させるのに必要な知識（必ずしも全部が必要なわけではない）
  - Java
  - Android
  - (FPGA)

2014年度
システム設計演習１
（PBL）

課題名：
FPGAボードを操作する
Androidアプリ
開発プロジェクト

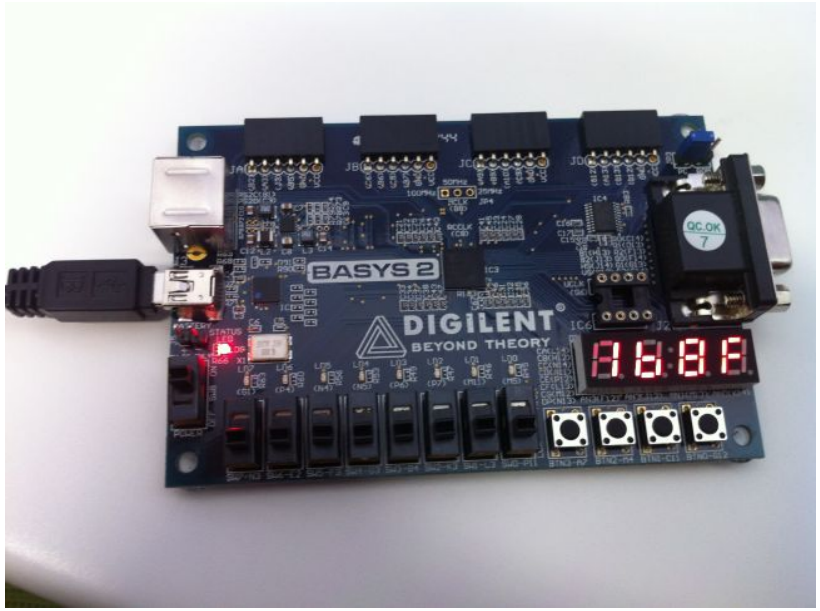
- 自分で企画して、Androidアプリ開発を行います
  - Androidアプリ開発に必要なJava言語を勉強しましょう
- Androidアプリから FPGAボードを操作します！
  - 希望者はFPGAの開発も出来ます

担当：大川
受け入れ可能人数　1～４人

操作対象のFPGAボード（1）

- Digilent Basys™2 Spartan-3E FPGA Board

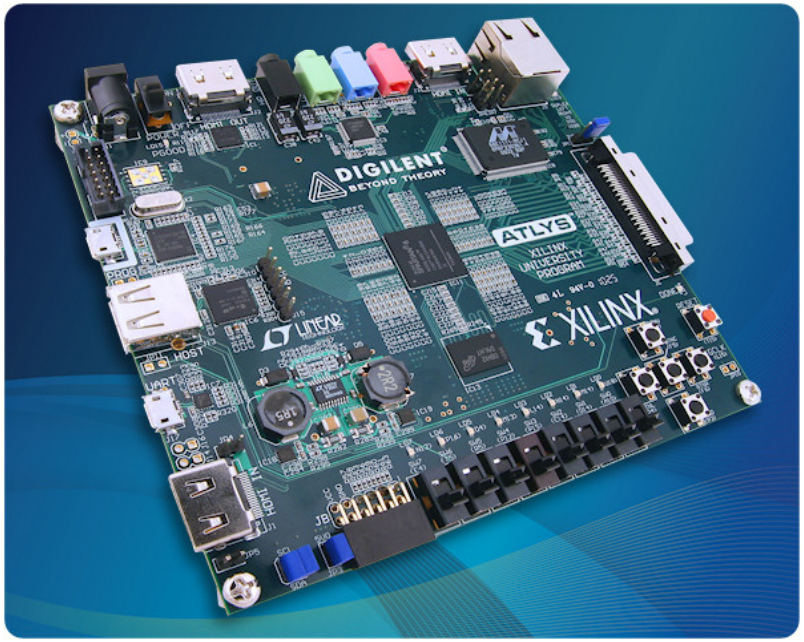
- 情報工学実験2で使用したもの



操作対象のFPGAボード（2）

- Digilent Atlys™ Spartan-6 FPGA Development Board

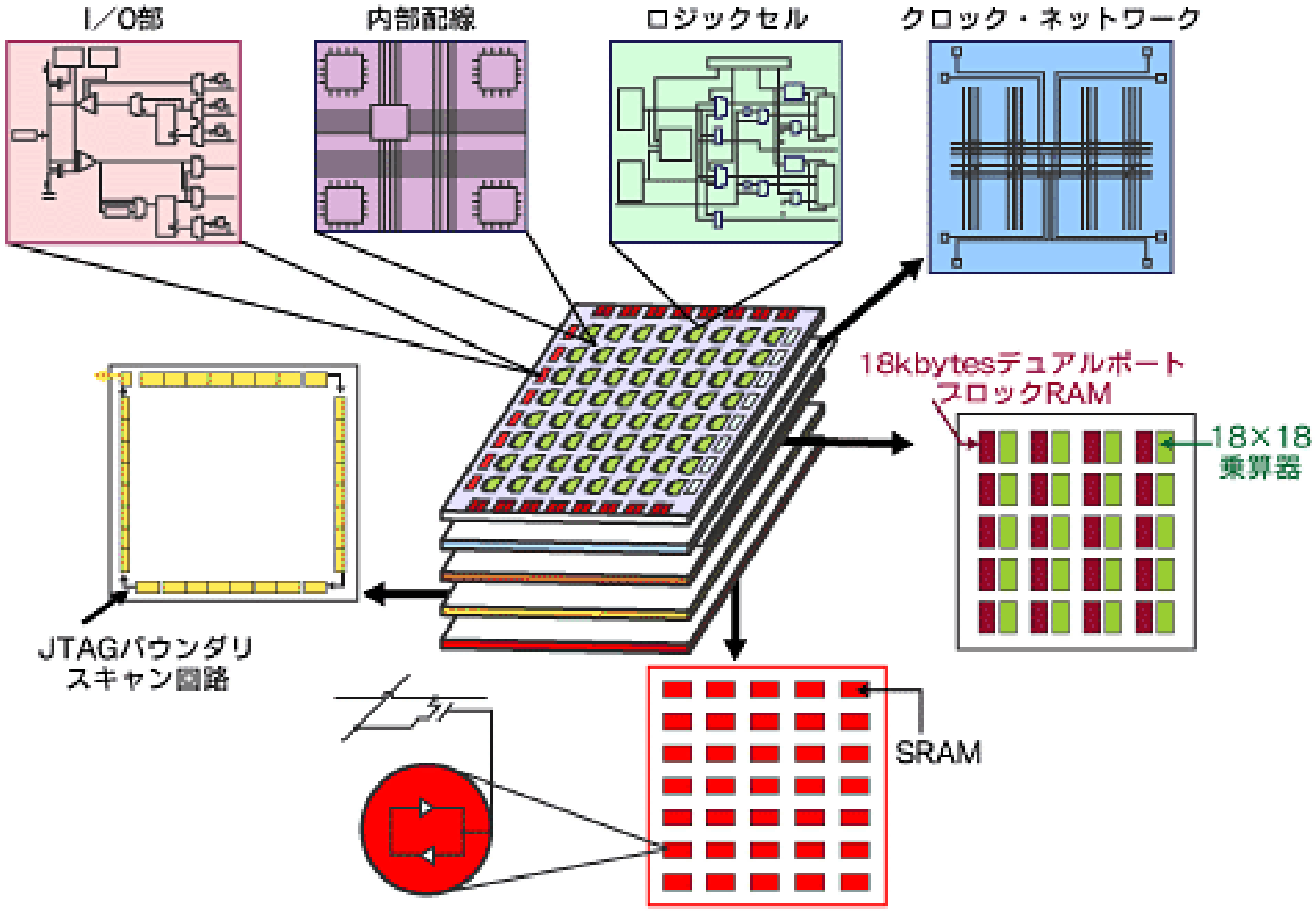
- 機能例
  - 1、スイッチ入力・LEDが光る
  - 2、音声入出力（音色シンセサイザ）
  - 3、画像入出力
  - 4、モータ制御
  - 5、イーサネット通信
  - 他にも色々可能



FPGAとは

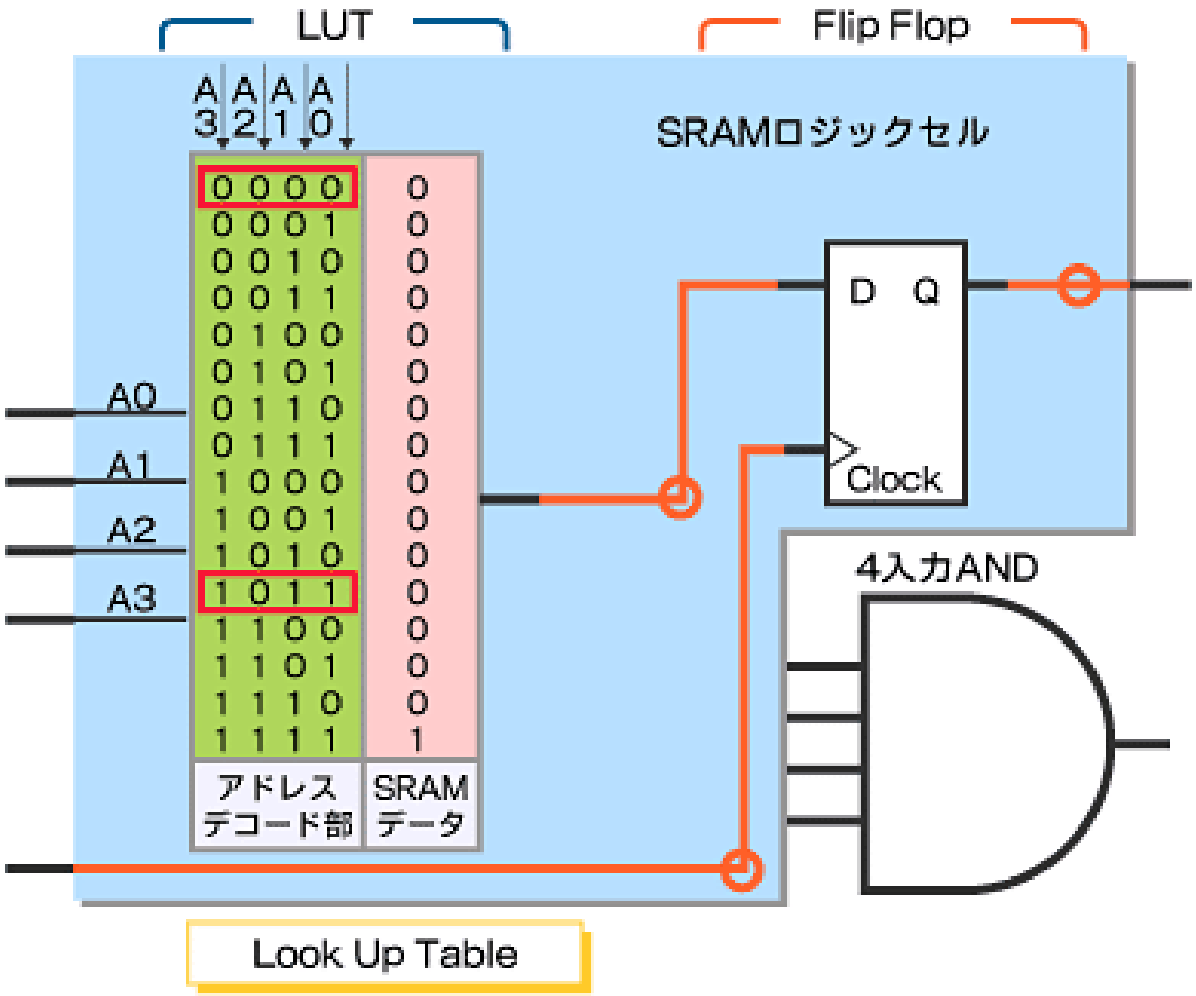
- FPGA：Field Programmable Gate Array
- 現場（フィールド）で（機能を）プログラム可能なゲートアレイ
  - ゲートアレイとは：　トランジスタ（ゲートアレイ）を作ってある半導体基板に、配線層だけをオーダーメイドで作る製造手法
- フィールドでプログラム可能にする方法は？
  - 配線＋スイッチをたくさん用意しておく
  - スイッチのON/OFFで、配線を（ある程度）自由につなぎかえる
  - スイッチ＝トランジスタ

FPGAアーキテクチャ概念図



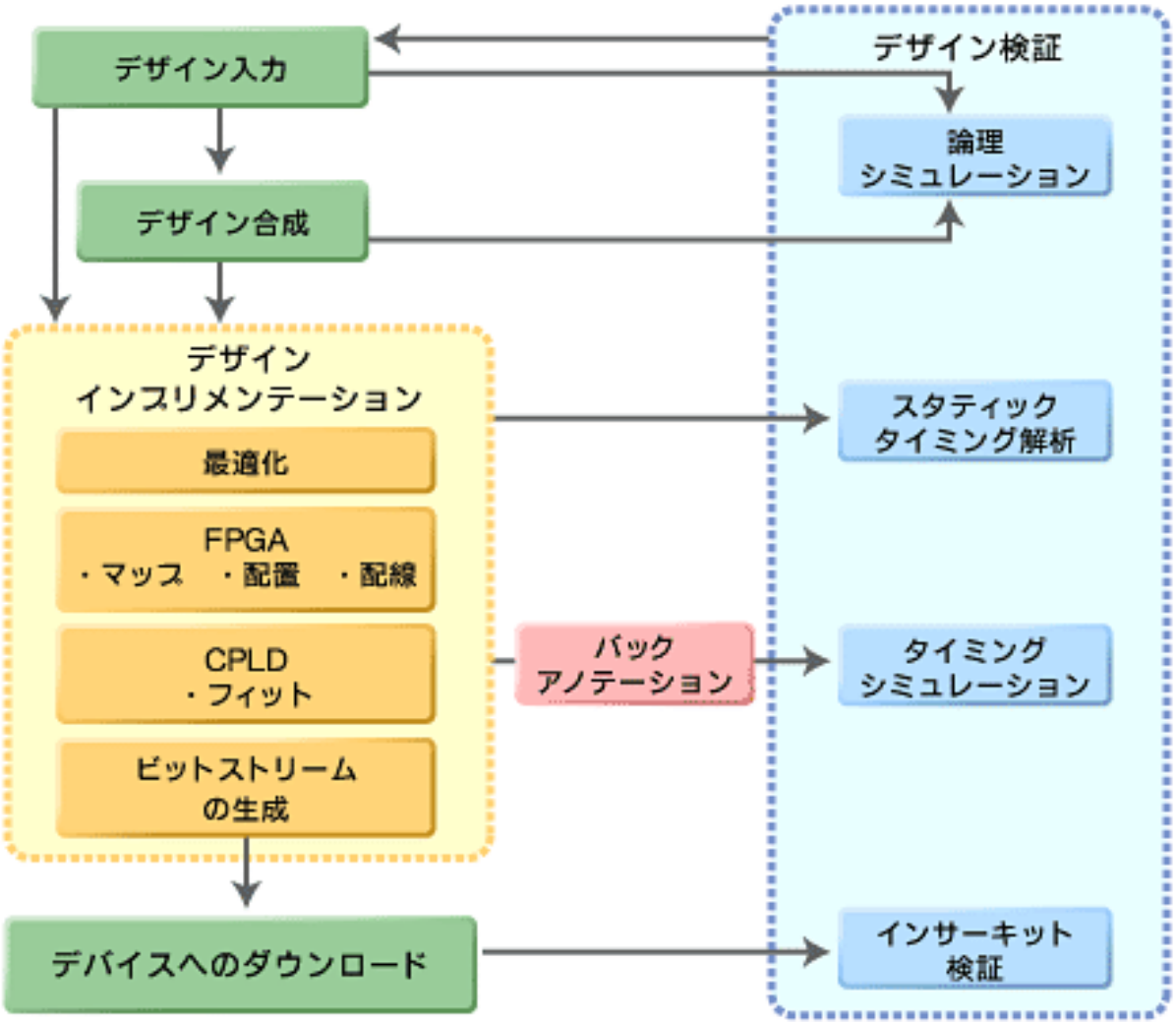
出典: http://monoist.atmarkit.co.jp/mn/articles/0609/20/news118_2.html

FPGAにおけるロジックセルの中身（例）



出典: http://monoist.atmarkit.co.jp/mn/articles/0609/20/news118_2.html

FPGAのデザインフロー（例）



出典: http://monoist.atmarkit.co.jp/mn/articles/0609/20/news118_3.html

RTLの例　（Verilog-HDL）

```
module bcount4 ( RSTb, clk, Q );
output [3:0] Q;
input  RSTb, clk;

always @ (posedge clk)
begin
        if (!RSTb)
                Q <= 4'h00;
        else
                Q <= Q + 4'h01;
end

endmodule
```

クロックが立ち上がる度に

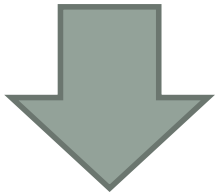
ここに書かれたことをする
（レジスタQを書き換える）
→Register Transfer Level

RTLからの回路合成後のネットリスト例

```
module bcount4 ( clk, RSTb, Q );
output [3:0] Q;
input  clk, RSTb;
wire N7, N8, N9, N10, N4, N5, N6, n1, ¥add_12/carry[2] , ¥add_12/carry[3] ;
                AND2 I11 ( .A(n1), .B(RSTb), .Y(N10) );
                AND2 I6 ( .A(Q[1]), .B(Q[0]), .Y(¥add_12/carry[2] ) );
                XOR2 I7 ( .A(Q[1]), .B(Q[0]), .X(N4) );

                AND2 I4 ( .A(Q[2]), .B(¥add_12/carry[2] ), .Y(¥add_12/carry[3] ) );
                XOR2 I5 ( .A(Q[2]), .B(¥add_12/carry[2] ), .X(N5) );
                AND2 I8 ( .A(RSTb), .B(N4), .Y(N9) );
                DF ¥Q_reg[0]  ( .CP(clk), .A(N10), .X(Q[0]), .Y(n1) );

                XOR2 I3 ( .A(Q[3]), .B(¥add_12/carry[3] ), .X(N6) );
                AND2 I9 ( .A(N5), .B(RSTb), .Y(N8) );
                DF ¥Q_reg[1]  ( .CP(clk), .A(N9), .X(Q[1]) );

                AND2 I10 ( .A(N6), .B(RSTb), .Y(N7) );
                DF ¥Q_reg[2]  ( .CP(clk), .A(N8), .X(Q[2]) );

                DF ¥Q_reg[3]  ( .CP(clk), .A(N7), .X(Q[3]) );
endmodule
```

FPGA導入時の障壁

- FPGAを使うときには (RTL)記述
  - クロックごとのレジスタへの書き込みを並列動作で記述する
  - 逐次動作はステートマシンで記述する
    - 記述量はC言語の7倍[*1]という報告もある



- Javaの動作記述からHDLを生成することができる「Synthesijer」を使用する

*1:若林一敏: “ソフトウェアプログラムからハードウェア記述を合成する高位合成技術-プロセッサ以外の汎用プログラム実行機構,” 電子情報通信学会基礎・境界ソサイエティ, Fundamentals Review Vol.6,No.1, pp.37-50, 2012 年7 月.

高位合成ツール「Synthesijer」

- SynthesijerはJavaの動作記述からVHDLを生成する高位合成ツール
  - Javaで機能レベルでのデバッグが可能
  - 動作記述からVHDLを生成するので記述するコード量が減る
  - プログラムの変更が容易

Javaのと20行程度で記述可能

```
1:  import net.wasamon.javarock.rt.*;
2:
3:  @javarockhdl
4:  public class Timer {
5:    static int counter = 0;
6:
7:    @auto
8:    public void run() {
9:      while(true) {
10:       counter++;
11:     }
12:   }
13:
14:   public static int getCounter(){
15:     return counter;
16:   }
17:
18:   public static void resetCounter(){
19:     counter = 0;
20:   }
21: }
```

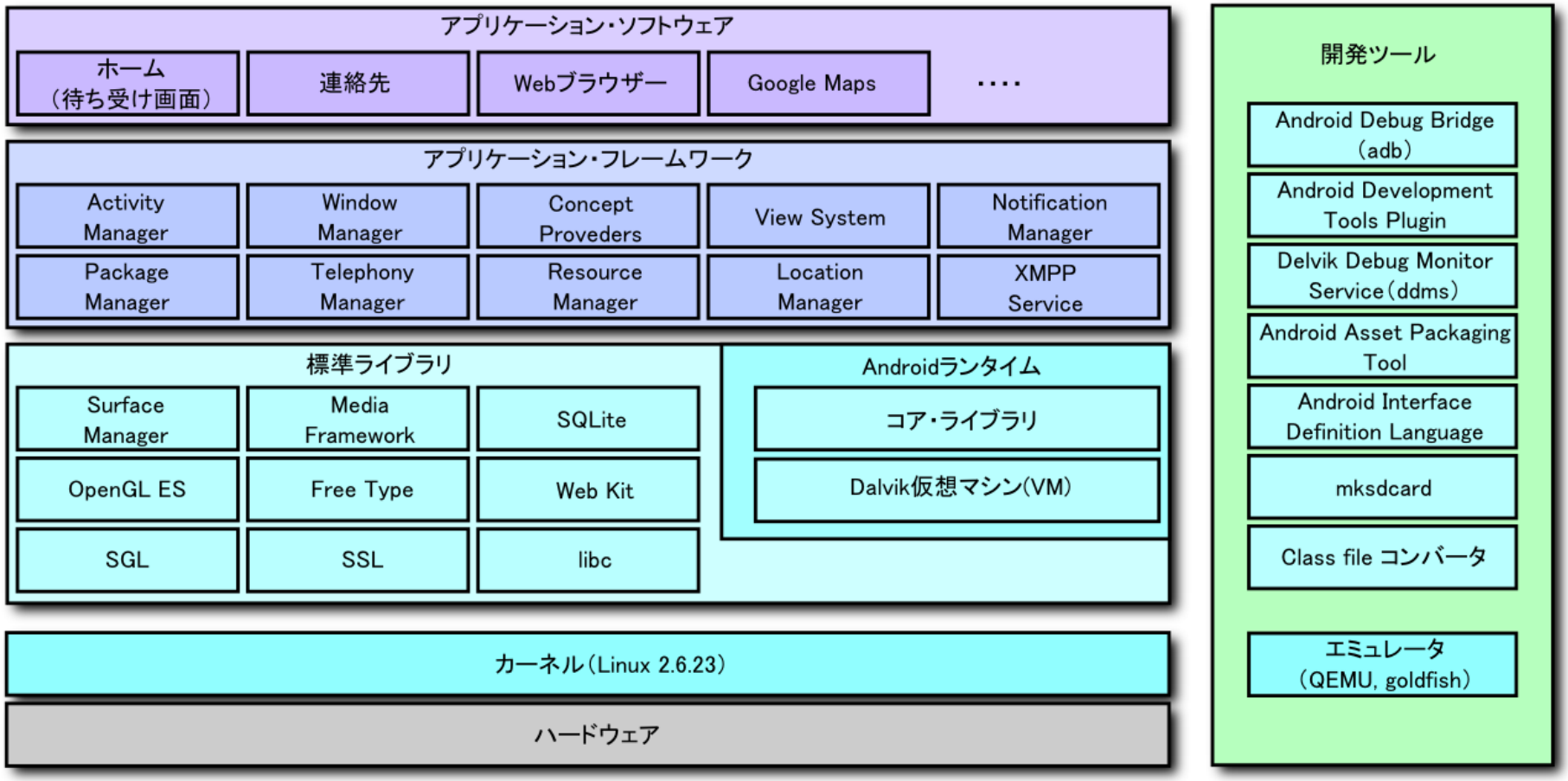
HDL
生成

JavaRockで生成したVHDL
全体で300行のうち一部記載した

```
127:      begin
128:        if (clk'event and clk = '1') then
129:          if (reset = '1') then
130:            state_counter_getCounter_getCounter_0 <= (others => '0');
131:            this_getCounter_method_busy <= '0';
132:            getCounter_method_state <= (others => '0');
133:            output_port_getCounter <= (others => '0');
134:          else
135:            case conv_integer(getCounter_method_state) is
136:              when 0 =>
137:                if(this_getCounter_method_request = '1') then
138:                  this_getCounter_method_busy <= '1';
139:                  getCounter_method_state <= getCounter_method_state + 1;
140:                else
141:                  this_getCounter_method_busy <= '0';
142:                end if;
143:              when 1 =>
144:                if(this_getCounter_method_request = '0') then
145:                  getCounter_method_state <= getCounter_method_state + 1;
146:                end if;
```

Androidアプリの構成

Androidのアーキテクチャ



15回の流れ

- 【授業計画】（シラバスより）
  各課題担当教員により詳細は異なるが，大筋では以下の授業計画に沿って実施される。
  - 1.オリエンテーション，本講義の目的，テーマの解説
  - 2.グループディスカッションによる課題の設定と到達目標の設定
  - 3.**課題実施計画**の作成
  - 4.インターネット検索，文献調査，フィールドワーク等による事例調査，関連技術の調査
  - 5.調査結果の検討，考察と再調査
  - 6.目的を実現するための基礎となる理論の学習
  - 7.学習結果を課題に適用するためのグループディスカッション
  - 8.課題製作の実施
  - 9.課題製作の実施と中間発表準備
  - 10.**中間発表**と問題点の討論
  - 11.課題製作の改良
  - 12.課題製作の仕上げ
  - 13.実施結果のまとめと発表準備
  - 14.**成果発表**と相互評価
  - 15.レポート作成

日程案

		内容
第1回	10月2日	オリエンテーション、目的・課題の説明、開発テーマアイディアだし
第2回	10月16日	開発テーマアイディアだし2、テーマ絞り込み、開発計画作成開始
第3回	10月23日	開発計画の作成（パワーポイント） ※Androidアプリ(Java)開発環境インストール
第4回	10月30日	【計画発表】 パワーポイントによるプレゼン、一人5分発表＋5分Q&A ※Androidアプリ(Java)開発環境ガイダンス ※Java開発最初の一歩
第5回	11月6日	※GitHubインストール・ガイダンス ※Java開発　次のステップ
第6回	11月13日	
第7回	11月20日	
第8回	12月4日	
第9回	12月11日	
第10回	12月18日	
第11回	12月20日	【中間発表】 パワーポイントによるプレゼン、一人10分発表＋10分Q&A 発表と問題点の検討
第12回	12月25日	
第13回	1月8日	
第14回	1月15日	
第15回	1月22日	【最終発表】 パワーポイントによるプレゼン、一人15分発表＋10分Q&A

評価基準　および　単位取得要件

- 成果物　（80%）
  - ソフトウェア：　GitHub上で提出　(※応相談)
  - プレゼンテーション3回分（計画・中間・最終）：　PDFで提出
- 出席　（20%）
  - 15回中10回の出席が必須　※最低で2/3以上の出席が必須

- 成果物とは
  - Androidアプリ
    - ソフトウェア　および　ハードウェア（FPGA設計）
    - GitHubにてオープンソースとして公開・管理　（応相談）
      - 成果物は自分(受講生)の作品として公開する
  - プレゼンテーション
    - 計画発表　1回
    - 中間発表　1回
    - 最終成果発表　1回