Javaによる専用ハードウェア開発を夢見て
Synthesijerのこれまでとこれから

わさらぼ
みよし たけふみ
(miyox)

2015.04.11

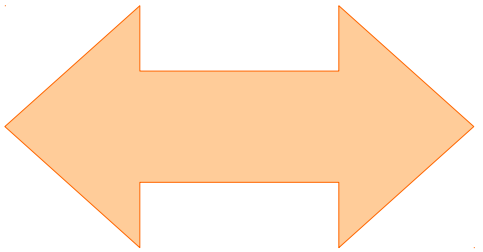
今日の話

- ✔ 何を作っているか
- ✔ どう作っているか

Javaによる専用ハードウェア？開発を夢見て
Synthesijerのこれまでとこれから

わさらぼ
みよし たけふみ
(miyox)

2015.04.11

専用  汎用

用途を特定しない
幅広く利用できる

専用 ⟷ 汎用

特定の用途を想定
得手不得手がはっきり

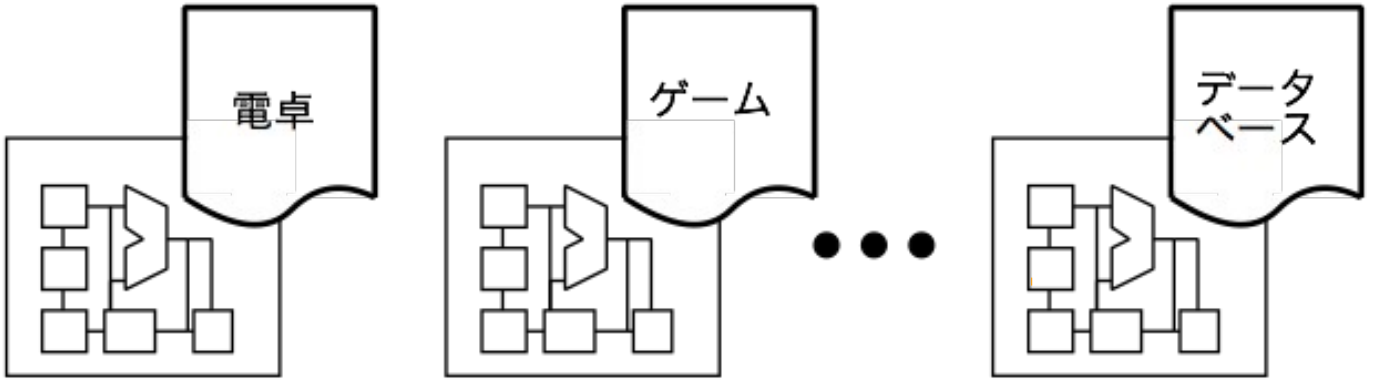
用途を特定しないHW
幅広く利用できるHW

専用HW ⟷ 汎用HW

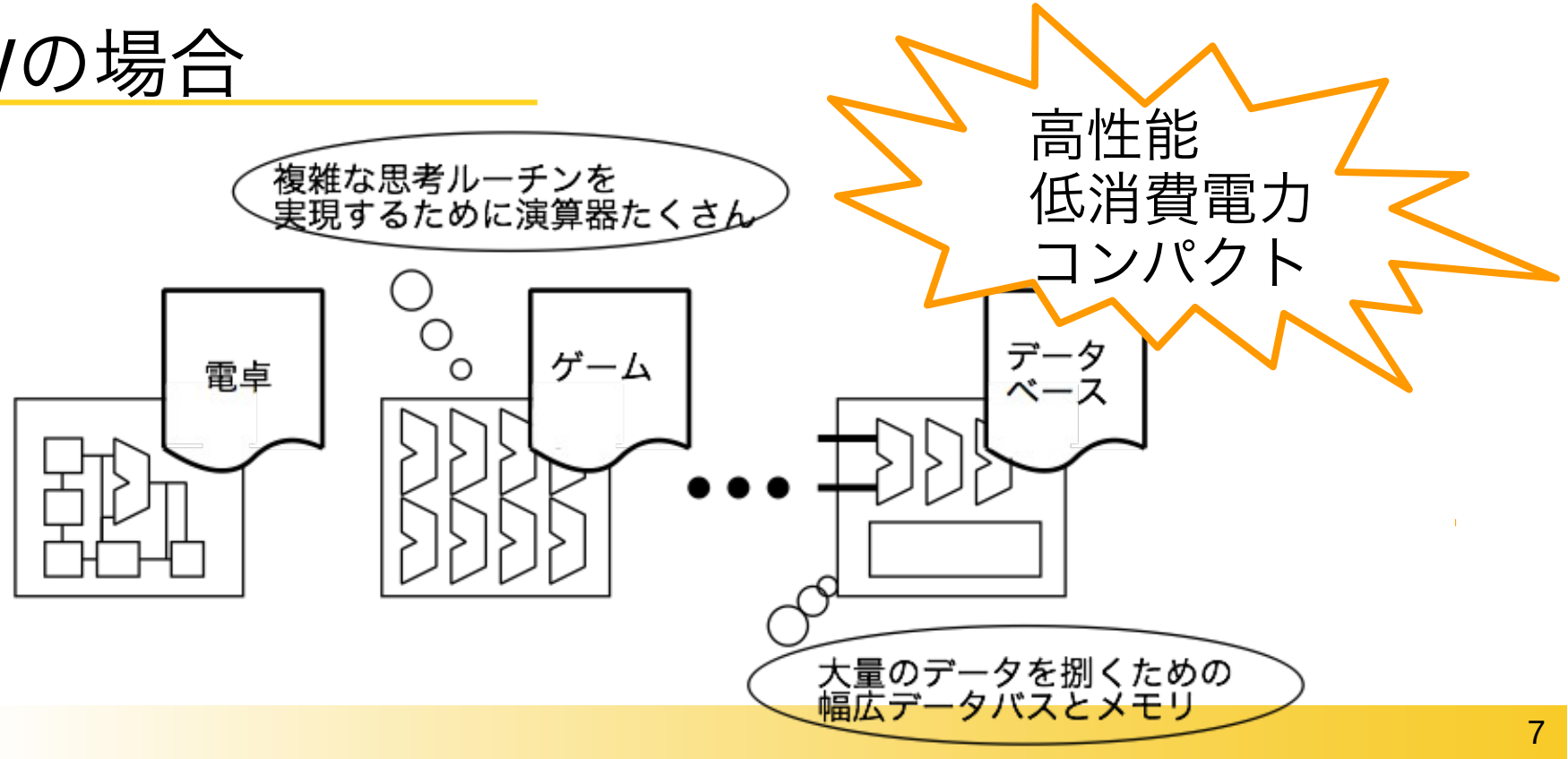
特定の用途を想定したHW
得意な処理は得意HW
不得意な処理は不得意/できないHW

専用にすると何が嬉しいか？

汎用HW(CPU)の場合

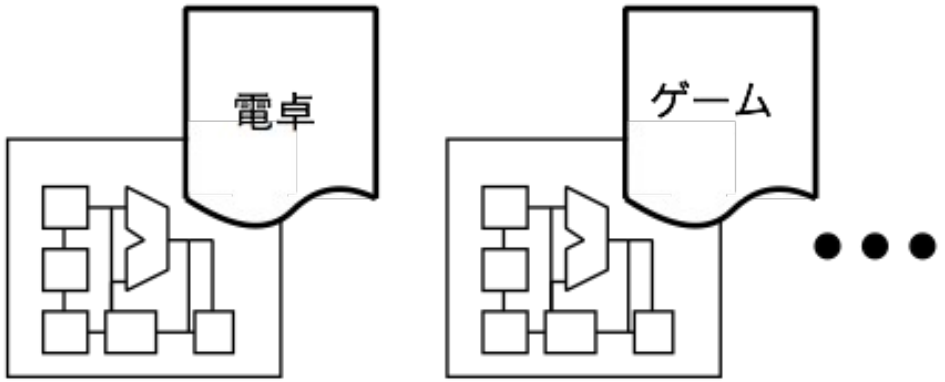


専用HWの場合



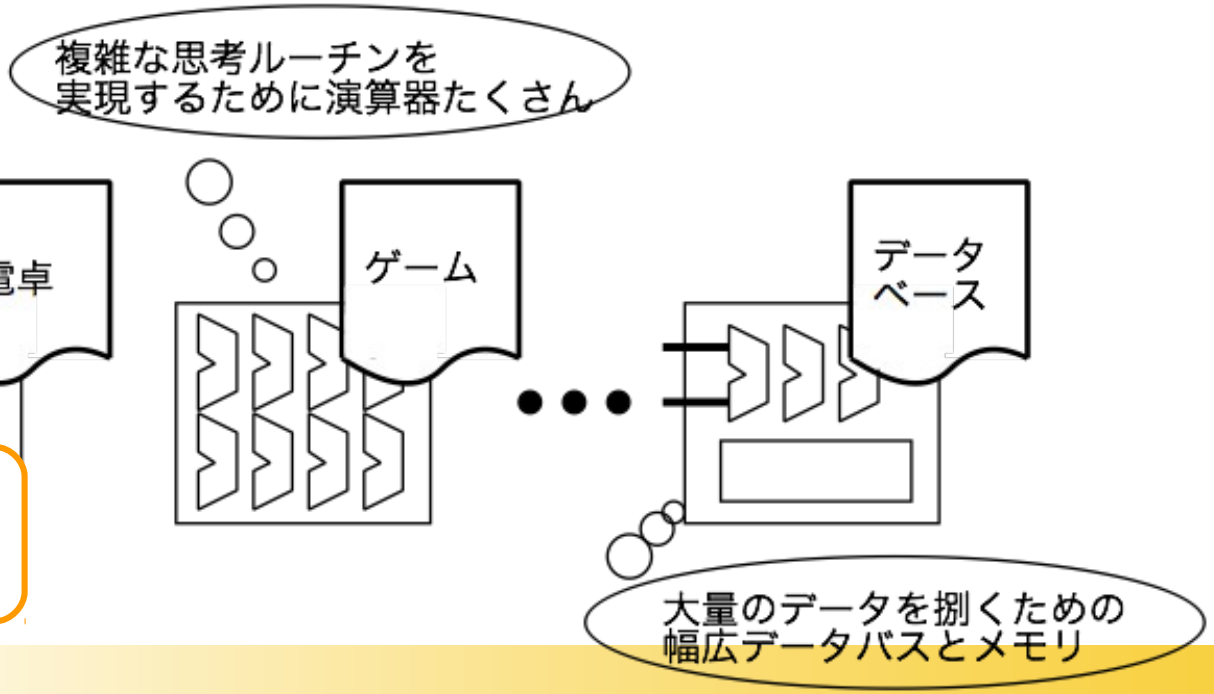
専用にすると何が嬉しくないか？

汎用HW(CPU)の場合



使い回しできる
= 安い．みんな使える

専用HWの場合



使い回しできない
= 高い．使うの大変

いつまでもCPUだけでいいの？

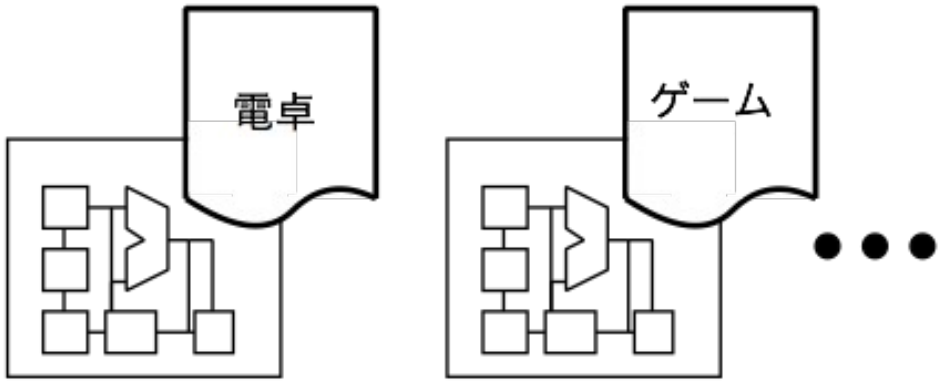
- CPU，これ以上速くなるの？
  - 動作周波数は頭打ち
  - 性能向上のためのトランジスタがのせられない
  - ~~ちょっと，この処理遅いけど来年のCPUなら大丈夫~~
- たくさんCPUならべると電気/熱が大変

…と，ここ何年も言われてきた

(と共に，ここ何年もスゴイ人たちが乗り越え続けている)

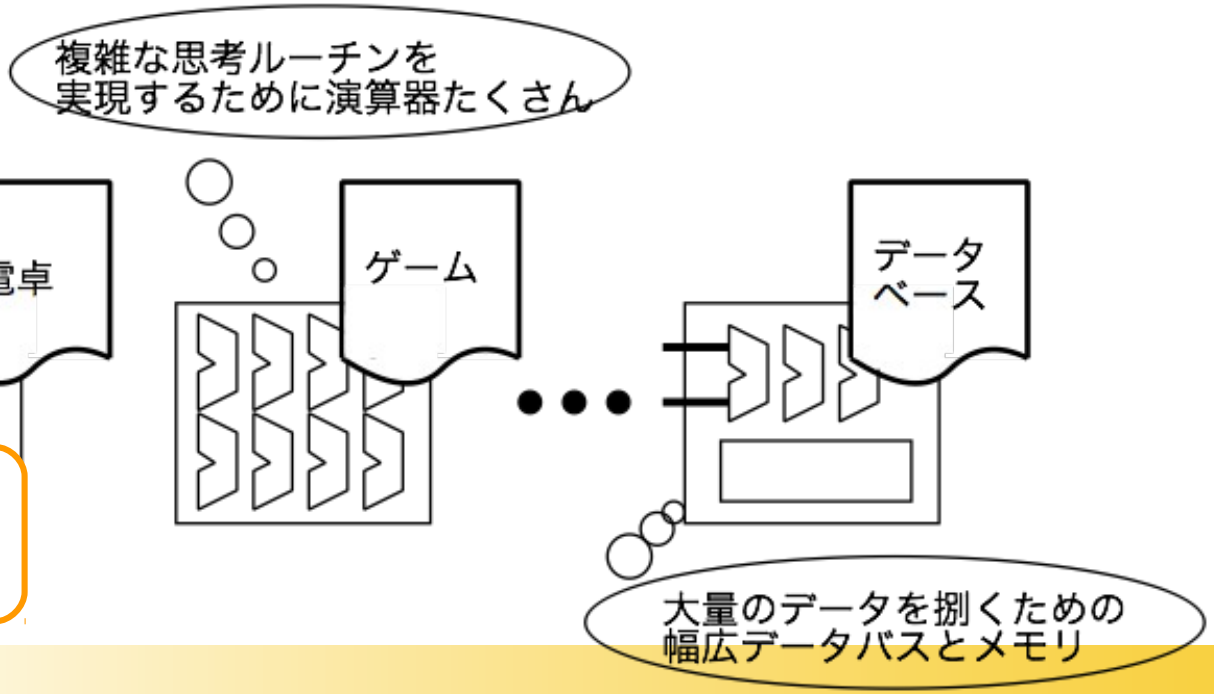
専用にすると何が嬉しくないか？

汎用HW(CPU)の場合



使い回しできる
= 安い．みんな使える

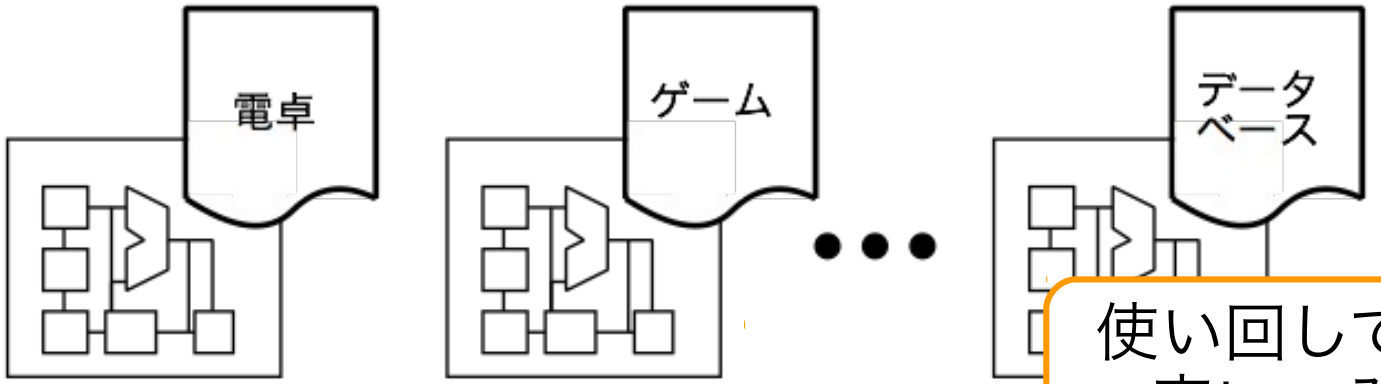
専用HWの場合



使い回しできない
= 高い．使うの大変

専用HWを作れる手軽なアイテム!!

汎用HW(CPU)の場合



使い回しできる
= 安い．みんな使える

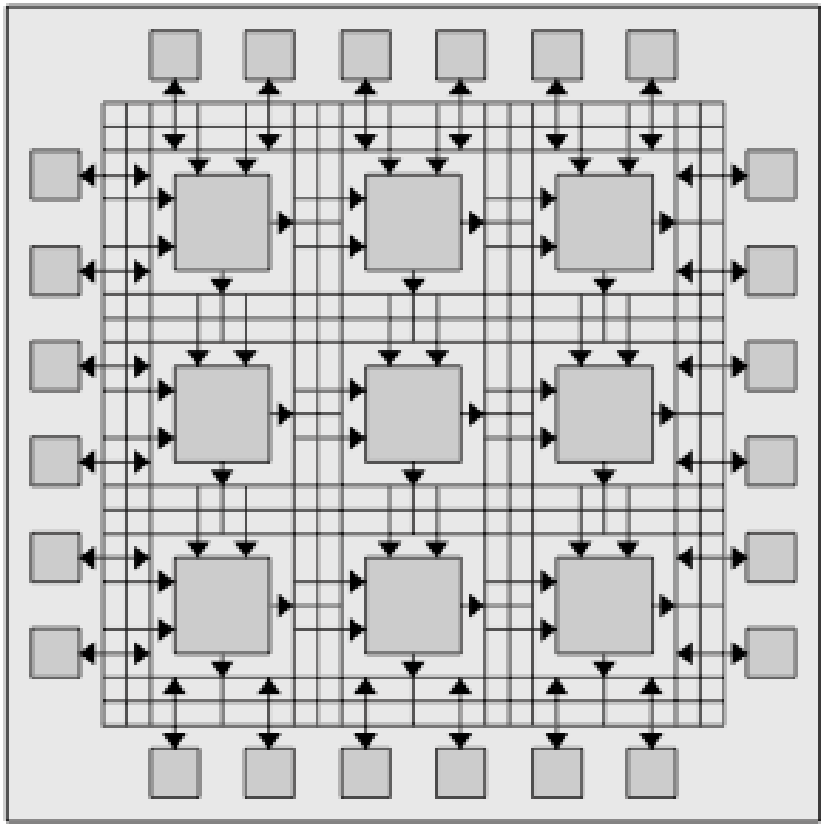
専用HWの場合

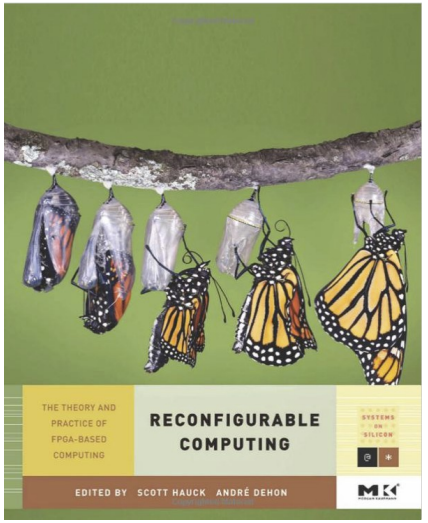
使い回しできない
= 高い．使うの大変

FPGAとは？

Field Programmable Gate Array

- ✔ 論理回路・データパスを自由に作り込めるLSI
- ✔ I/Oを自由に使える



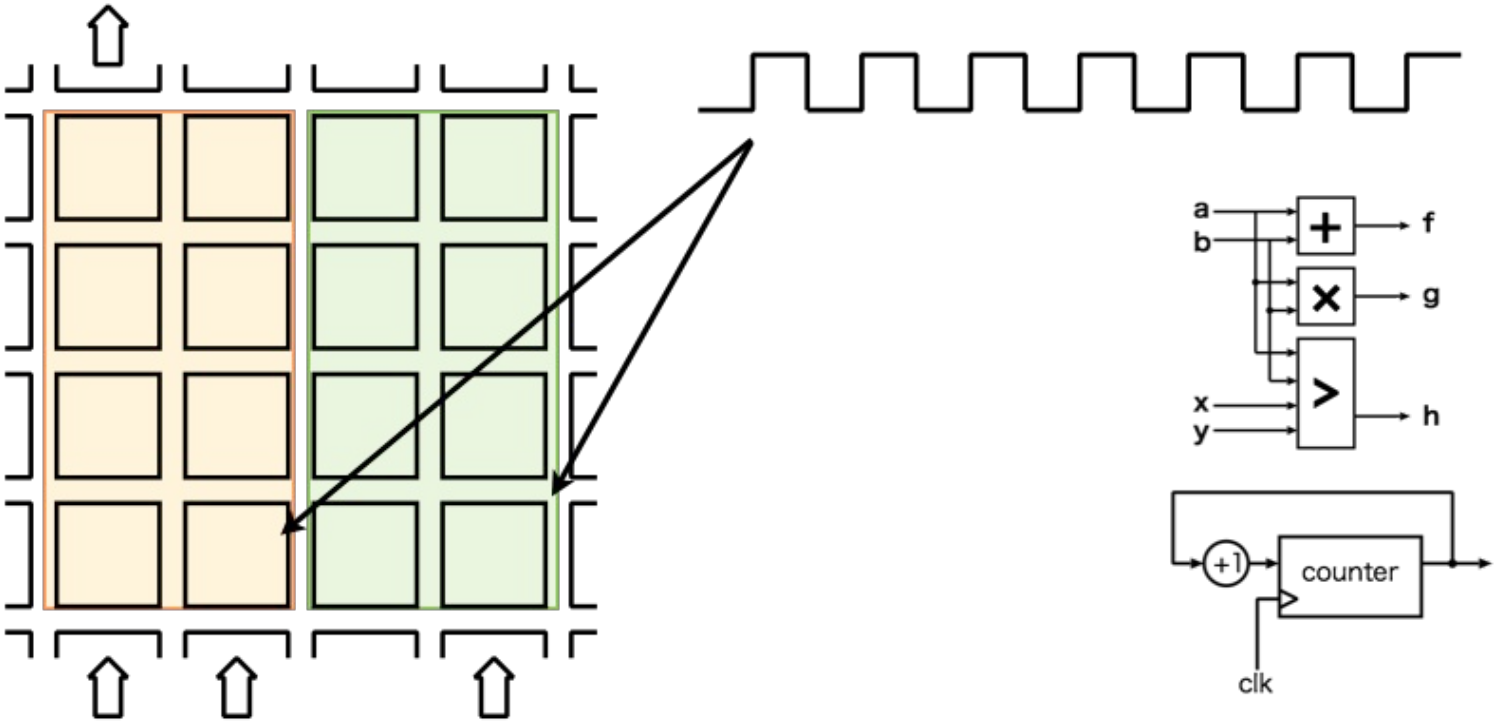


[6] S. Hauck and A. DeHon, Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

FPGAとは？

Field Programmable Gate Array

- ✔ 論理回路・データパスを自由に作り込めるLSI
- ✔ I/Oを自由に使える
- ✔ クロックレベルの同期と並列性を活用した処理を実現

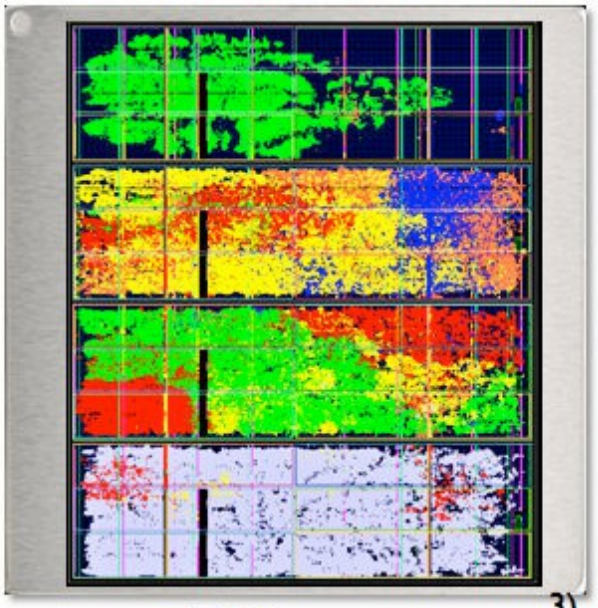


FPGAとは？

Device[1]	Logic Cells	Configurable Logic Blocks (CLBs)		DSP Slices[3]	Block RAM Blocks[4]			CMTs[5]	PCIe[6]	GTX	GTH	GTZ	XADC Blocks	Total I/O Banks[7]	Max User I/O[8]	SLRs[9]
		Slices[2]	Max Distributed RAM (Kb)		18 Kb	36 Kb	Max (Kb)									
XC7V585T	582,720	91,050	6,938	1,260	1,590	795	28,620	18	3	36	0					
XC7V2000T	1,954,560	305,400	21,550	2,160	2,584	1,292	46,512	24	4	36	0					
XC7VX330T	326,400	51,000	4,388	1,120	1,500	750	27,000	14	2	0	28					
XC7VX415T	412,160	64,400	6,525	2,160	1,760	880	31,680	12	2	0	48					
XC7VX485T	485,760	75,900	8,175	2,800	2,060	1,030	37,080	14	4	56	0					
XC7VX550T	554,240	86,600	8,725	2,880	2,360	1,180	42,480	20	2	0	80					
XC7VX690T	693,120	108,300	10,888	3,600	2,940	1,470	52,920	20	3	0	80					
XC7VX980T	979,200	153,000	13,838	3,600	3,000	1,500	54,000	18	3	0	72					
XC7VX1140T	1,139,200	178,000	17,700	3,360	3,760	1,880	67,680	24	4	0	96					
XC7VH580T	580,480	90,700	8,850	1,680	1,8											
XC7VH870T	876,160	136,900	13,275	2,520	2,8											

1)

2)

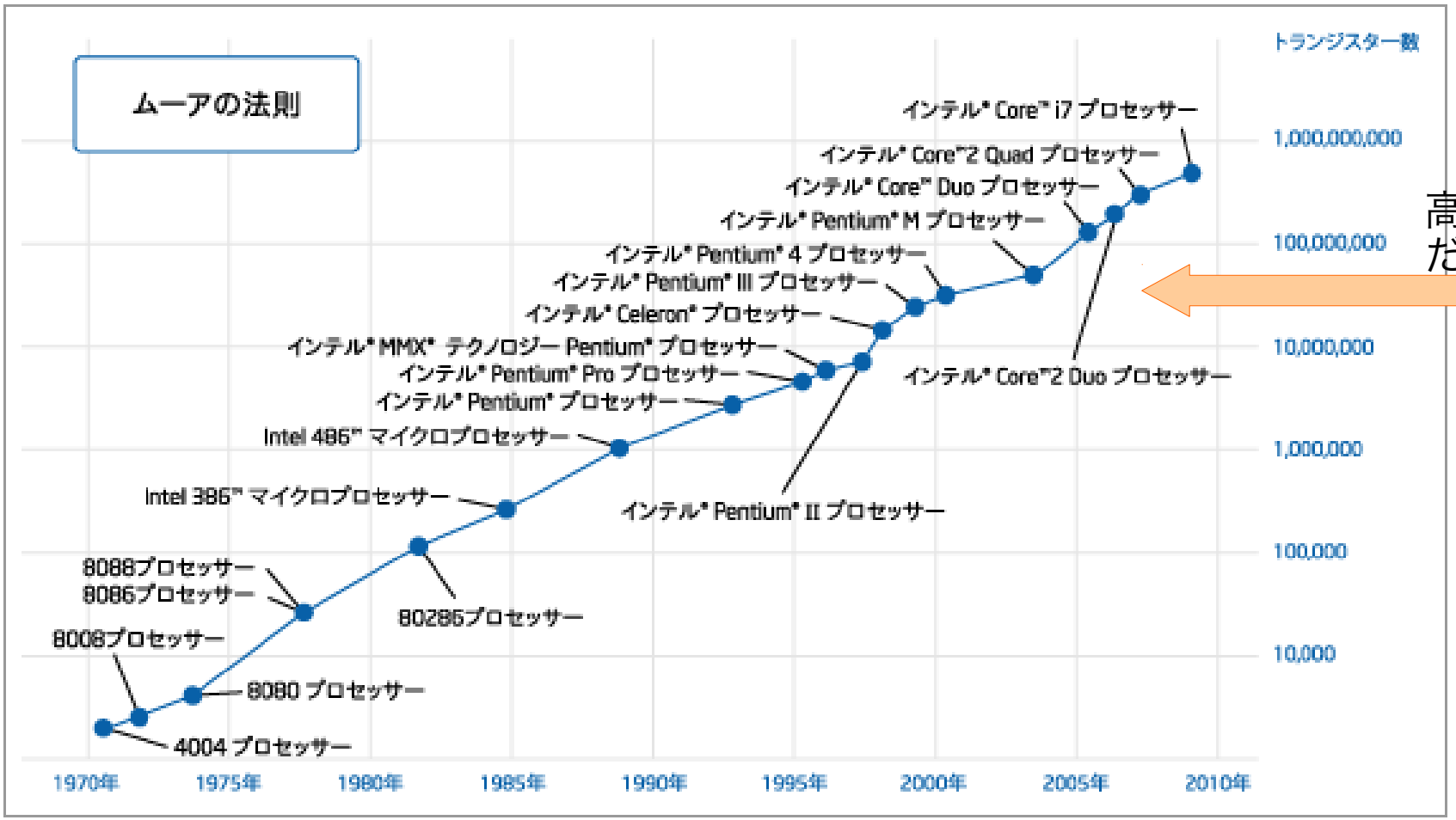


3)

1) http://japan.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf
2) http://hitechglobal.com/boards/virtex7_v2000t.htm
3) http://http://low-powerdesign.com/sleibson/2011/10/25/generation-jumping-2-5d-xilinx-virtex-7-2000t-fpga-delivers-1954560-logic-cells-consumes-only-20w/

FPGAとは？

Intelのプロセッサと比べてみてみると



高いFPGAが
だいたいこの辺

グラフは http://japan.intel.com/contents/museum/processor/index.html より

FPGA 美味しいの？

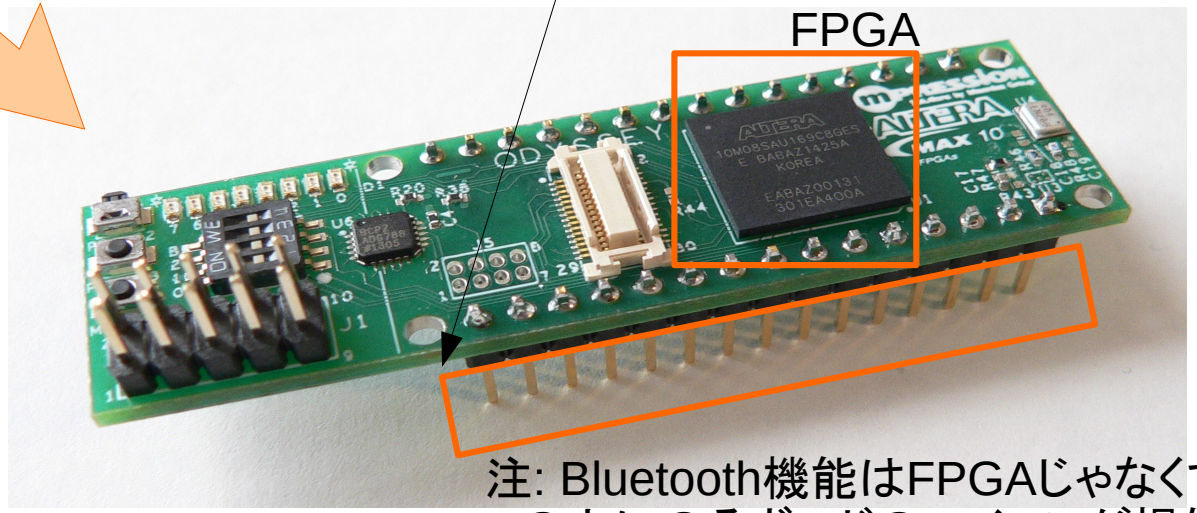
FPGA　○○○○○○○○○○

検索

○○○○○○○○○○にはコンピュータ界隈で
有名な会社の名前をいれてください

小さくてかわいいのも

例: Odyssey MAX10



ここに，LEDとかセンサとか
モーターとかつないで遊ぶ

注: Bluetooth機能はFPGAじゃなくて
この上にのるボードのマイコンが提供

https://cloud.altera.com/devstore/board/odyssey-max-10-fpga-kit/

JavaでIoTといえば...

http://www.snm.ethz.ch/snmwiki/Projects/SunSPOT

FPGAで
アプリを作ること

Javaによる専用ハードウェア開発を夢見て

Synthesijerのこれまでとこれから

なぜJavaなのか？

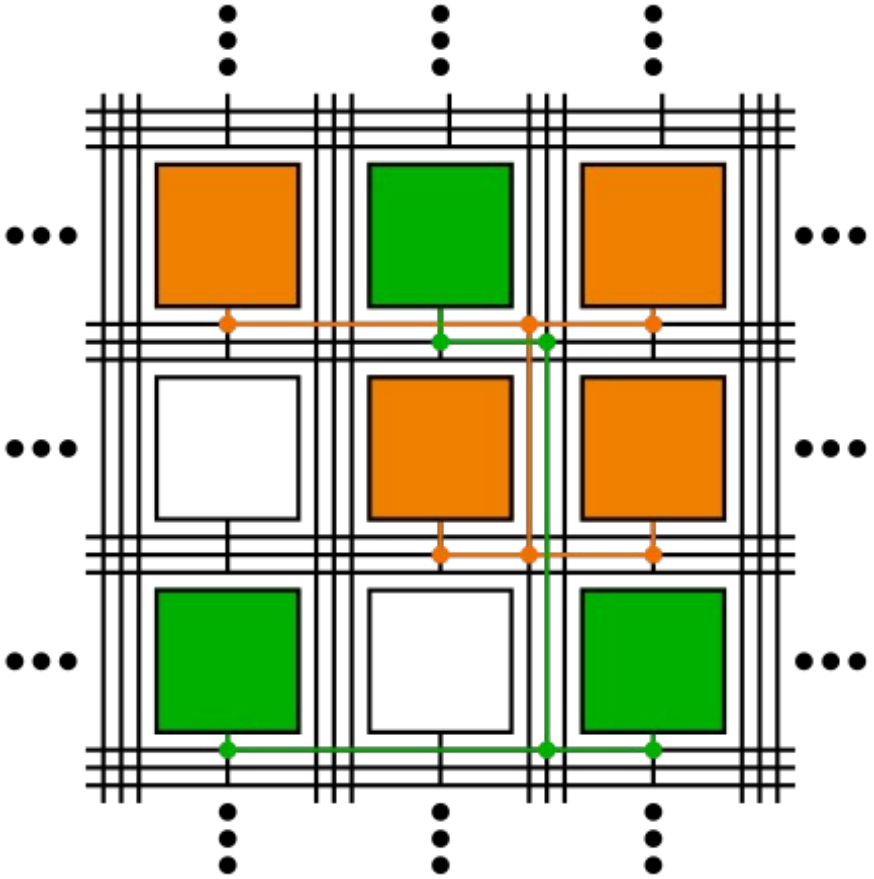
Write Once, Run Anywhere!!

Write Once, Run Anywhere!!

たとえCPUがなくても!!

一般的なFPGAアプリ設計

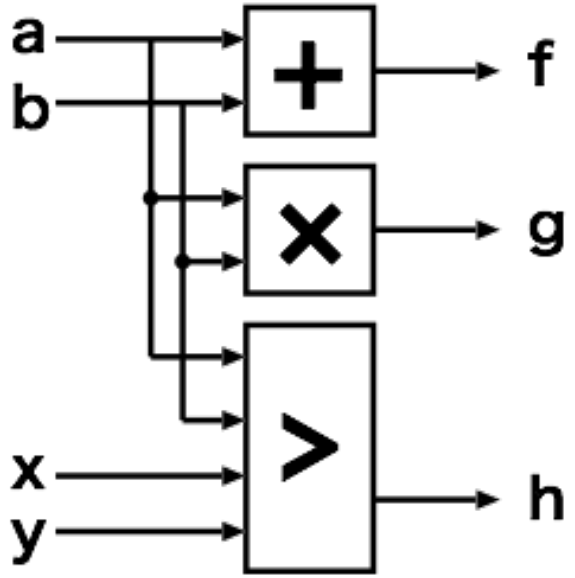
- ✔ 論理回路構成要素の内容を決める
- ✔ 論理回路構成要素同士の接続関係を決める



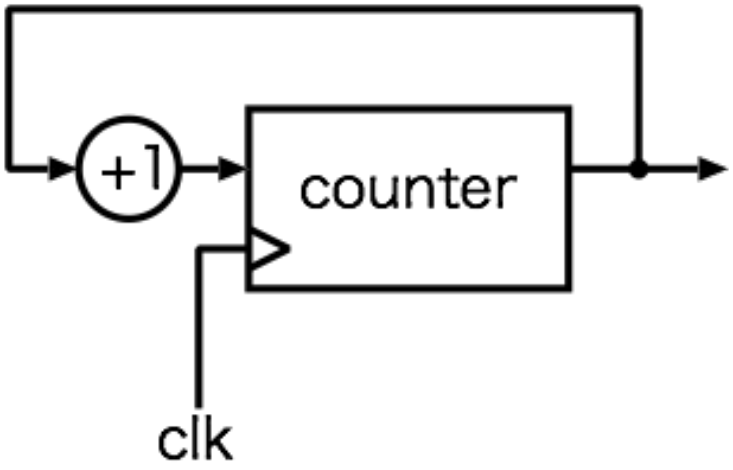
一般的なFPGAアプリ設計

HDL(VHDLやVerilog HDL)によるRTL(Register Transfer Level)設計

```
process (a, b, x, y)
begin
  f <= a + b;
  g <= a * b;
  if a > b then
    h <= x;
  else
    h <= y;
  end if;
end process;
```




```
process (clk)
begin
  if clk'event and clk = '1' then
    if reset = '1' then
      counter <= (others => '0');
    else
      counter <= counter + 1;
    end if;
  end if;
end process;
```



HDLによるRTL設計

メリット

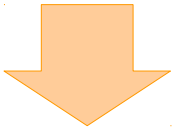
- ロジックを抽象化した式/構文で設計できる
- クロックレベルのデータ制御
- 細粒度の並列性の活用

デメリット

- 記述が煩雑
- “状態”を自分で管理しなければいけない
- デバッグ/動作検証が難しい

HDLによるRTL設計

```
a = 1;
b = 2;
c = a + b;
```

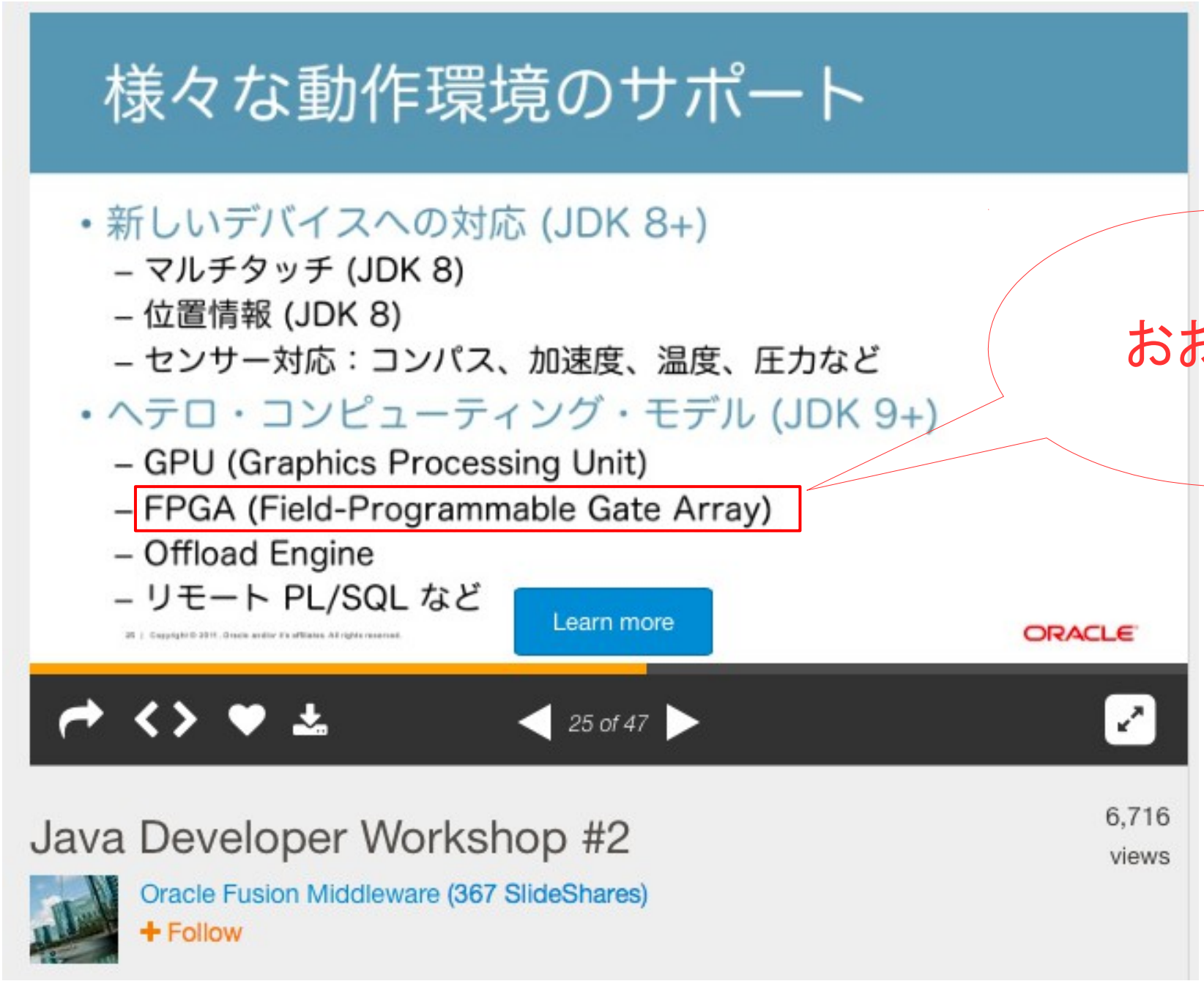


```
if clk'event and clk = '1' then
  case (s) is
    when S0 =>
      a <= 1;
      s <= S1;
    When S1 =>
      b <= 2;
      s <= S2;
    When S2 =>
      c <= a + b;
      s <= S3;
  end case;
end if;
```

RTL書きたくない!!

Javaで書きたい!!


Synthesijer



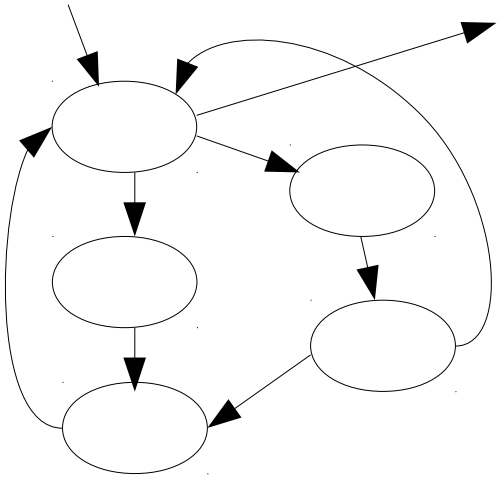
おおお！

http://www.slideshare.net/OracleMiddleJP/java-developer-workshop-2

残念ながら無関係です

Synthesijer とは

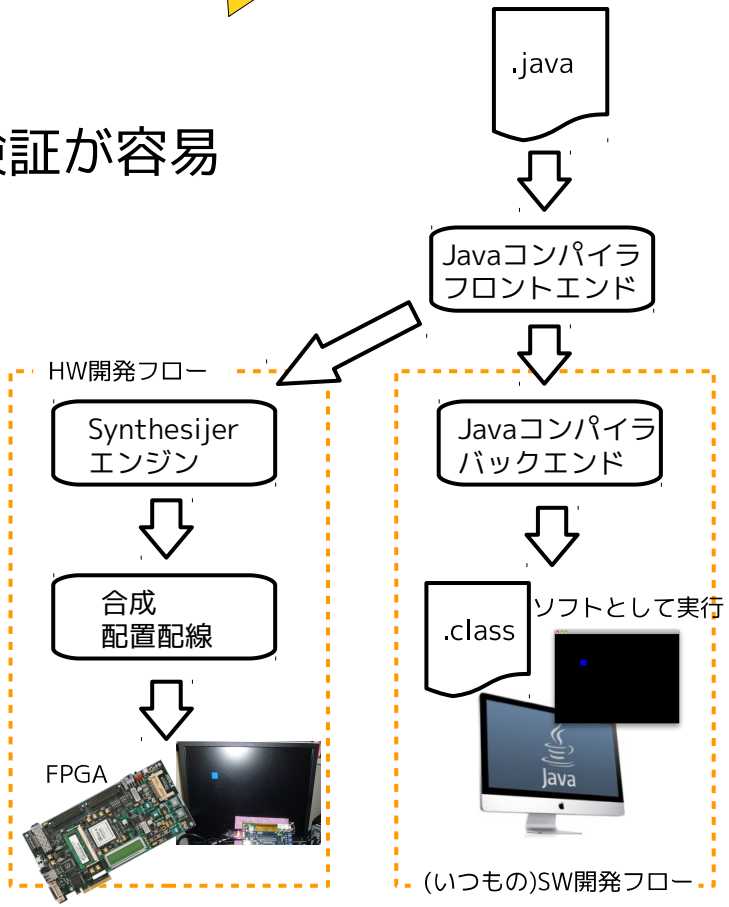
- JavaプログラムをFPGA上のハードウェアに変換
  - 複雑なアルゴリズムのハードウェア実装を楽に
  - オブクジェクト指向設計による再利用性の向上
- 特殊な記法，追加構文はない
  - ソフトウェアとして実行可能．動作の確認、検証が容易
  - 書けるプログラムに制限は加える
    (動的なnew，再帰は不可など)
- HDLモジュールのJavaからのインスタンス生成



```
while(){
if(...){
  …
}else{
  …
  …
}
….
}
```

複雑な状態遷移も，Javaの制御構文を使って楽に設計できる

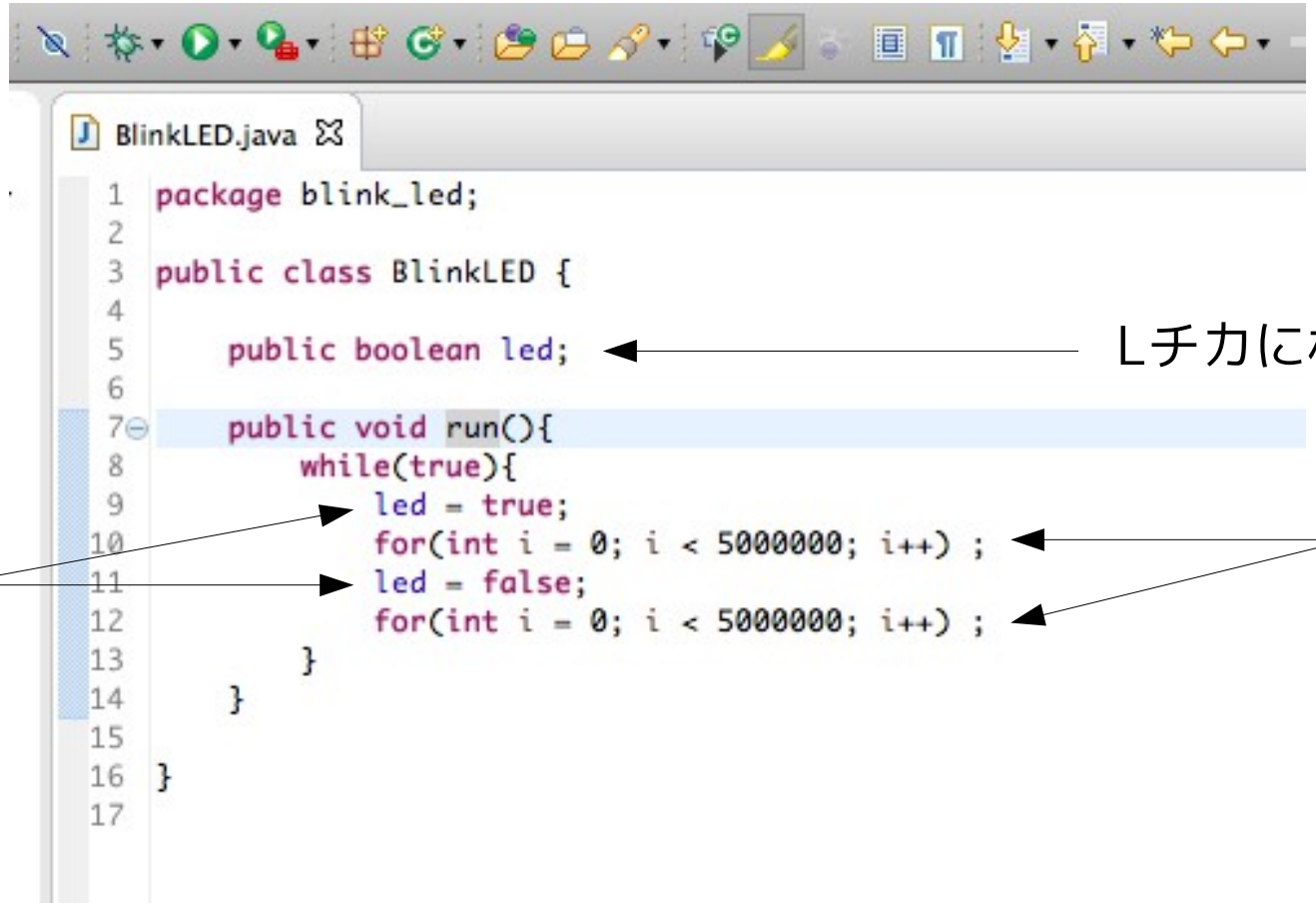
Open-source



同じJavaプログラムをソフトウェアとしても
FPGA上のハードウェアとしても実行可能

たとえば，Lチカ

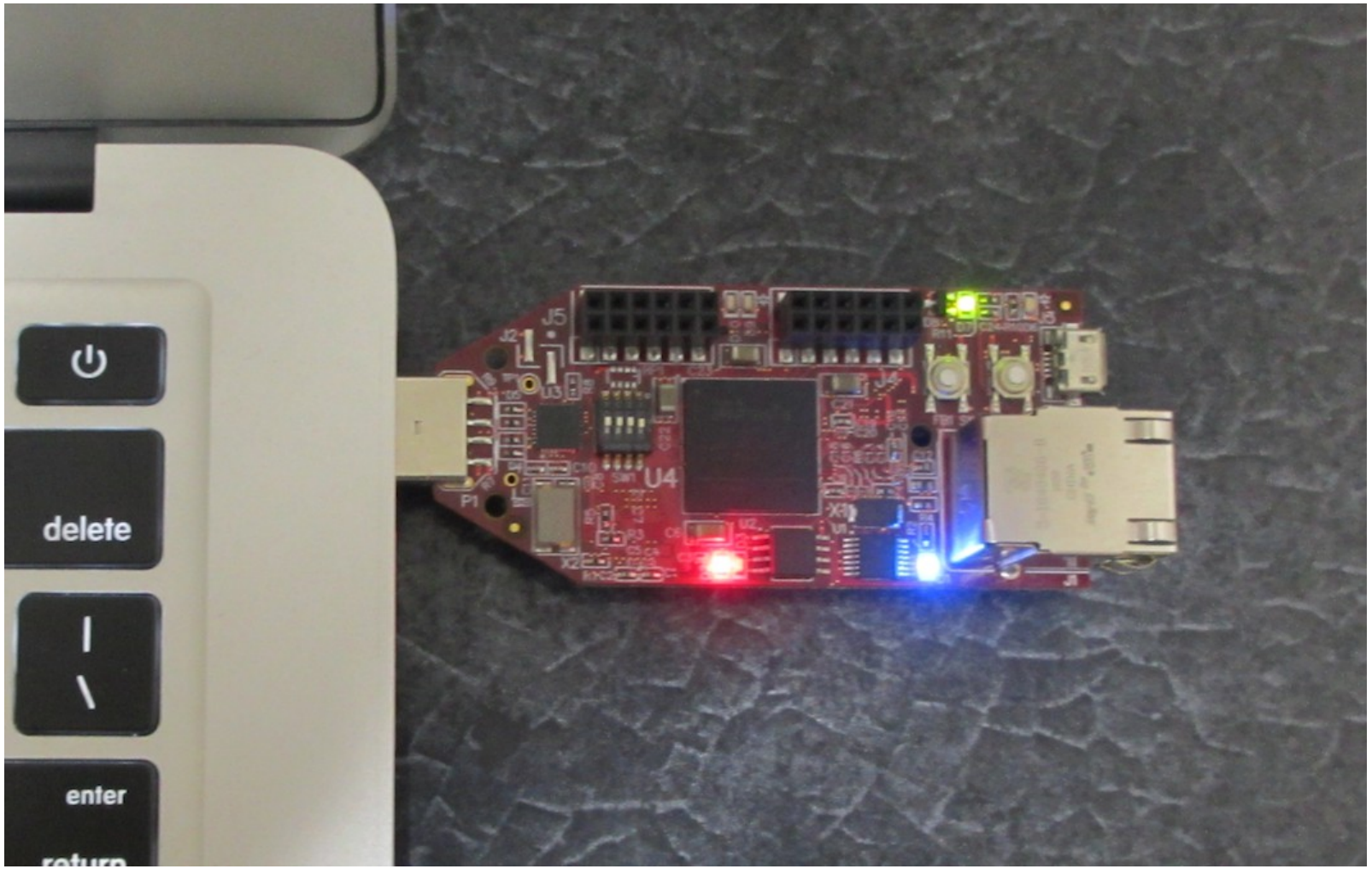
間隔をおいて変数ledをtrue/falseするプログラムを書く



自動コンパイルが裏で動くので，Javaコードとしての正しさは即座にチェックされる

たとえば，Lチカ

コンパイルしてFPGAにロードする



プロセッサ上で動くのと何が違うの？

	CPU	Synthesijerでできたもの
処理方式	読む→解釈→実行	演算器としてならべられる 適切なタイミングで確定
命令セット	決まっている	入力コードによって生成
I/O	決まっている	割と自由に追加できる
個数	決まっている	入力コードによって決まる

なぜJavaなのか？

Javaベースの高位合成処理系

- ✔ クラスによるオブジェクト指向設計言語

← HWのモジュール設計との親和性は高そう

- ✔ 言語仕様で並列処理をサポート

  - ✔ Thread，wait-notify
- ✔ (Cと違い)明示的なポインタの扱いが不要

  - ✔ 言語の想定するメモリ構造から解放され得るかも

- ✔ 動的な振る舞いは厄介そう

デモと応用事例の紹介

デモ(1) BrainF**k

- Javaといえば，スクリプト言語のホストとしても魅力的
  - JRuby，Scala，Clojure，and etc.
- BF: とても小さなスクリプト言語処理系

  +, -, >, <, [,], ., ,の記号からなるインタプリタ

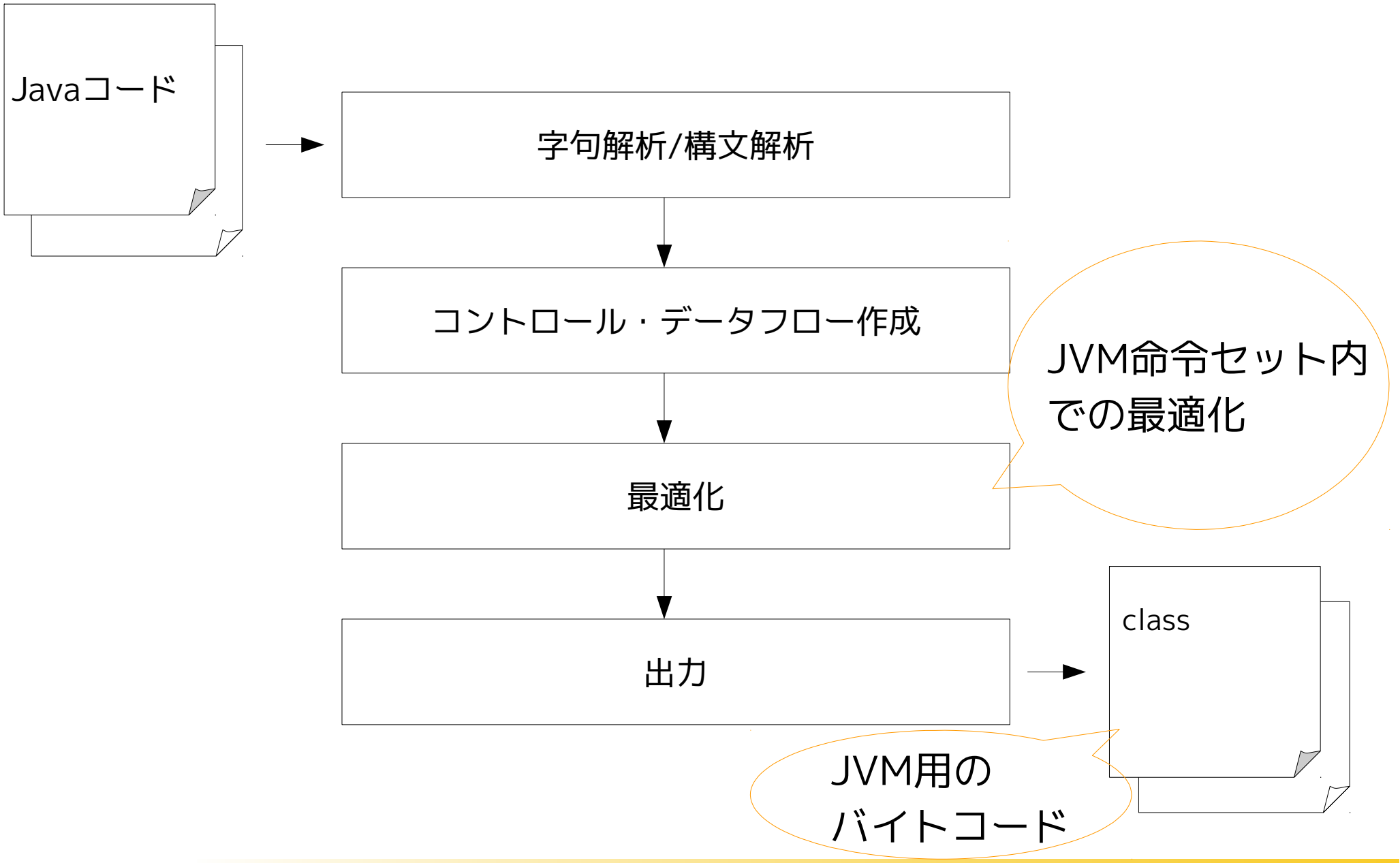
```
++++++++++[>++++++++++<-]>++++.+++++++.--------.--.
```

→ hoge

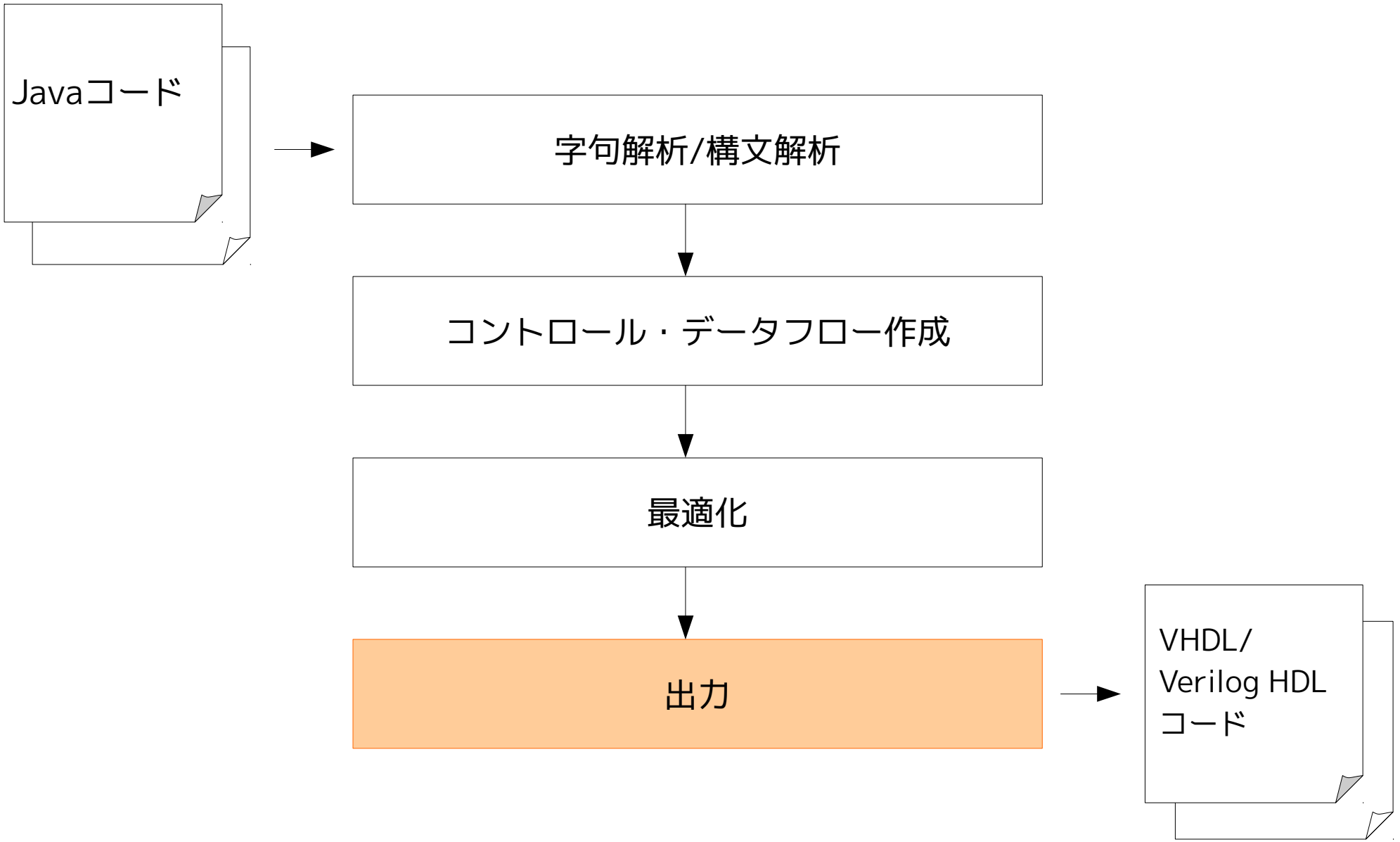
http://www.kmonos.net/alang/etc/brainfuck.php

どう作っているか？

Javaコンパイラ


Javaコード
字句解析/構文解析
コントロール・データフロー作成
最適化
出力
JVM命令セット内での最適化
class
JVM用のバイトコード

Synthesijerオーバービュー



OpenJDKすてき!!

- ✔ オープンソースなJavaの実装
- ✔ 勿論Javaコンパイラもオープンソース

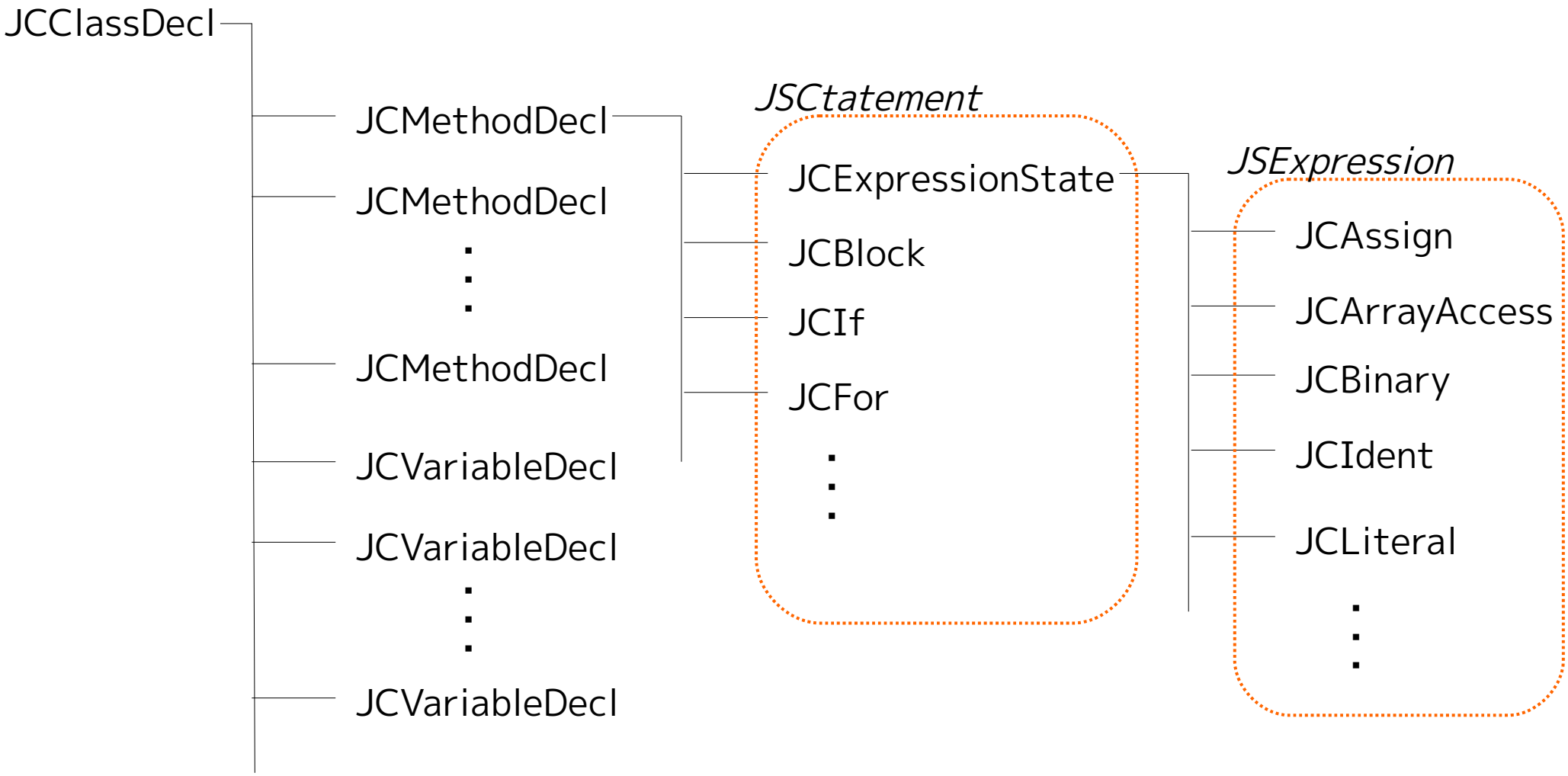
- ✔ openjdk/com/sun/tools/javac/main/JavaCompiler.java をフックすれば，解析/最適化済みの情報にアクセス可

OpenJDKすてき!!

JavaCompiler.javaの中身

```
/** Generate code and emit a class file for a given class
 *  @param env    The attribution environment of the outermost class
 *                containing this class.
 *  @param cdef   The class definition from which code is generated.
 */
JavaFileObject genCode(Env<AttrContext> env, JCClassDecl cdef) throws IOException {
    synthesijer.jcfrontend.Main.newModule(env, cdef); // add hook for synthesijer
    try {
        if (gen.genClass(env, cdef) && (errorCount() == 0))
            return writer.writeClass(cdef.sym);
    } catch (ClassWriter.PoolOverflow ex) {
        log.error(cdef.pos(), "limit.pool");
    } catch (ClassWriter.StringOverflow ex) {
        log.error(cdef.pos(), "limit.string.overflow",
                  ex.value.substring(0, 20));
    } catch (CompletionFailure ex) {
        chk.completionError(cdef.pos(), ex);
    }
    return null;
}
```

Javaコードの内部表現



この構造をたどりながら，すきな形(VHDL/Verilog HDL)に変換する

たとえばclass

```
public class JCTopVisitor extends Visitor{

    private final Module module;

...

    public  void visitClassDef(JCClassDecl that){
        for (JCTree def : that.defs) {
            if(def == null){
                ;
            }else if(def instanceof JCMethodDecl){
                def.accept(this);
            }else if(def instanceof JCVariableDecl){
                def.accept(new JCStmtVisitor(module));
            }else{
                System.err.printf("Unknown class: %s (%s)", def, def.getClass());
            }
        }
    }

...
```

たとえばmethod

```
    public void visitMethodDef(JCMethodDecl decl){
        String name = decl.getName().toString();
        Type type;
        if(JCFrontendUtils.isConstructor(decl)){
            type = new MySelfType();
        }else{
            type = TypeBuilder.genType(decl.getReturnType());
        }
        Method m = new Method(module, name, type);

        m.setArgs(parseArgs(decl.getParameters(), m));
...

        m.setPrivateFlag(JCFrontendUtils.isPrivate(decl.mods));
        m.setParallelFlag(JCFrontendUtils.isAnnotatedBy(decl.mods.annotations, "parallel"));
...
        m.setConstructorFlag(JCFrontendUtils.isConstructor(decl));
...
        for(JCStatement stmt: decl.body.getStatements()){
            JCStmtVisitor visitor = new JCStmtVisitor(m);
            stmt.accept(visitor);
            m.getBody().addStatement(visitor.getStatement());
        }

        module.addMethod(m);
    }
```

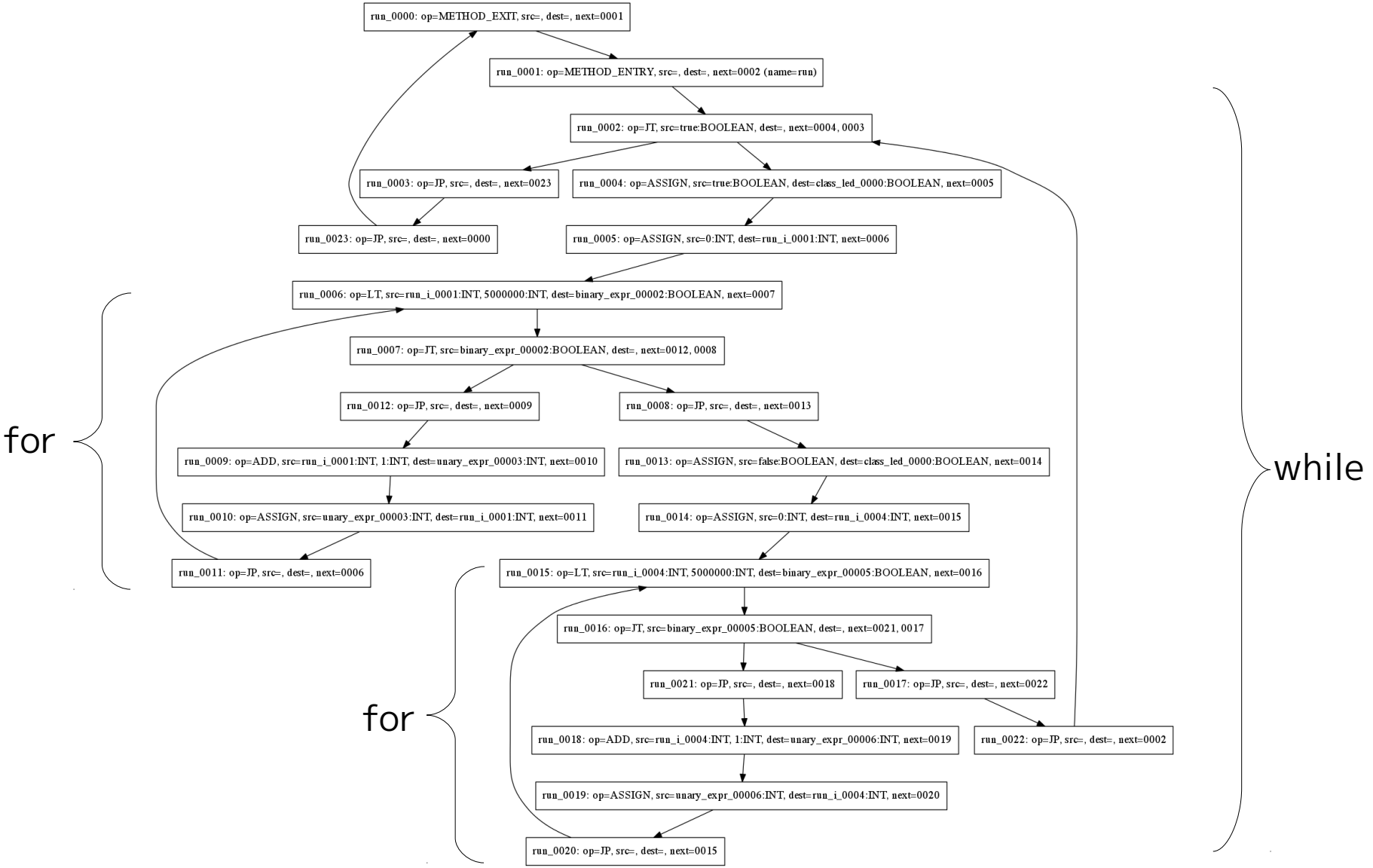
たとえば制御構文

```
public void visitIf(JCIf that){
    IfStatement tmp = new IfStatement(scope);
    tmp.setCondition(stepIn(that.cond, scope));
    tmp.setThenPart(wrapBlockStatement(stepIn(that.thenpart, scope)));
    if(that.elsepart != null){
        tmp.setElsePart(wrapBlockStatement(stepIn(that.elsepart, scope)));
    }
    stmt = tmp;
}

public void visitForLoop(JCForLoop that){
    ForStatement tmp = new ForStatement(scope);
    for(JCStatement s: that.init){
        //tmp.addInitialize(stepIn(s, scope));
        tmp.addInitialize(stepIn(s, tmp));
    }
    tmp.setCondition(stepIn(that.cond, tmp));
    for(JCStatement s: that.step){
        tmp.addUpdate(stepIn(s, tmp));
    }
    tmp.setBody(wrapBlockStatement(stepIn(that.body, tmp)));
    stmt = tmp;
}
```

Synthesijer でのHDL生成: 出力(状態遷移)

メソッド毎にスケジュール表を生成 → 状態遷移機械



Synthesijerのこれから

ロードマップ

手続き型言語的基本演算，操作	整数プリミティブ型変数の利用	算術，論理，シフト演算（除算・剰余算以外）	制御構文（分岐，ループ）	メソッド呼び出し	除算・剰余算	浮動小数点数演算		
オブジェクト指向的インスンタンス協調	finalでのインスタンスの生成	インスタンスメソッドの呼び出し	インスタンス変数へのリードアクセス	インスタンス変数へのライトアクセス	インスタンス間での配列読み書きのサポート	インスタンスの配列，配列の配列のサポート	コンストラクタのサポート	インスタンスのチェインアクセス
並列化パフォーマンス	基本ブロック内自動並列化	Threadによる明示的な処理の並列化をサポート	ループ内パイプライニング					
ライブラリ	整数プリミティブ変数の配列	AXI接続用のコンポーネントライブラリの提供	Stringクラスのサポート					
ユーザビリティ	コマンドラインでのコンパイル	HDLモジュールとのバインディング機構	FPGA合成ツールとの連携，統一的な開発フローの提供	スケジューリング可視化ツール	HDL生成ライブラリを活用したDSLの提供			

今後挑戦したいこと

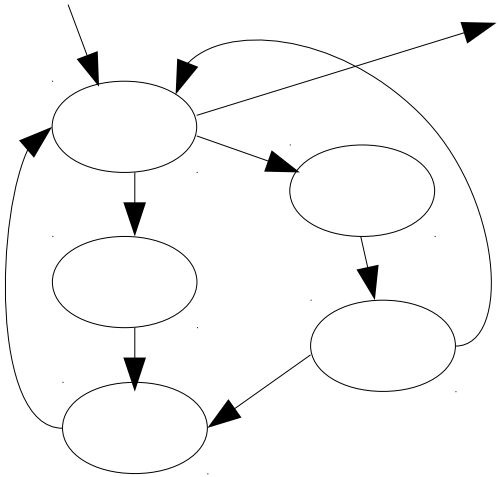
- ✔ メソッドのパイプライン化
- ✔ I/Oストリームへの対応

- ✔ コンストラクタに対応
  - ✔ コンパイル時のJavaプログラム実行
- ✔ lambda式対応
- ✔ Streamへの対応

- ✔ Erlangフロントエンド

Synthesijer とは

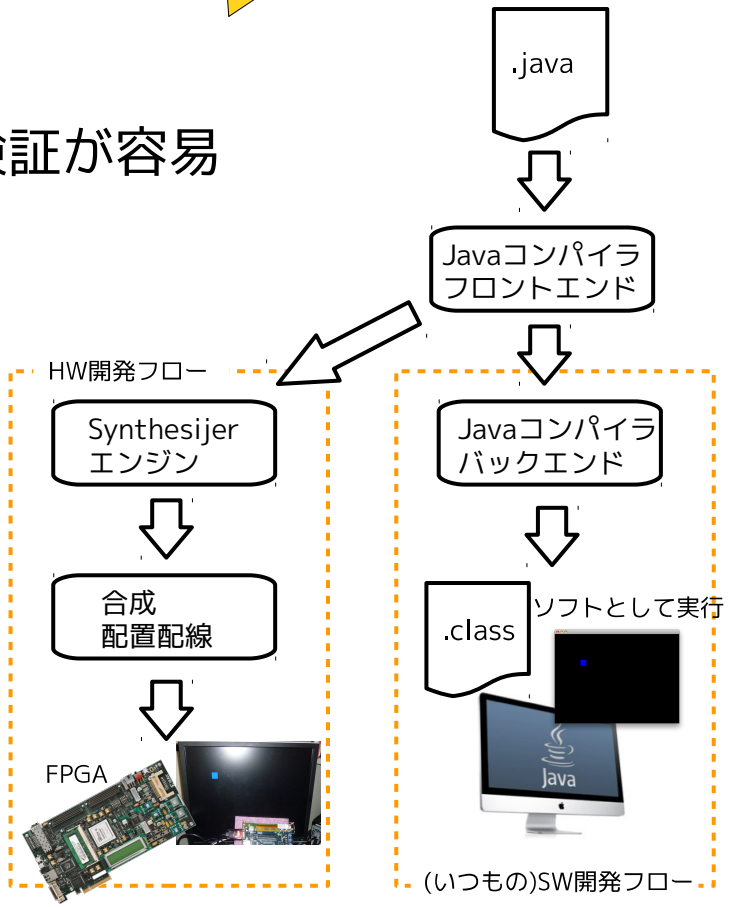
- JavaプログラムをFPGA上のハードウェアに変換
  - 複雑なアルゴリズムのハードウェア実装を楽に
  - オブクジェクト指向設計による再利用性の向上
- 特殊な記法，追加構文はない
  - ソフトウェアとして実行可能．動作の確認、検証が容易
  - 書けるプログラムに制限は加える
    (動的なnew，再帰は不可など)
- HDLモジュールのJavaからのインスタンス生成



```
while(){
if(...){
  …
}else{
  …
  …
}
….
}
```

複雑な状態遷移も，Javaの制御構文を使って楽に設計できる

Open-source



同じJavaプログラムをソフトウェアとしても
FPGA上のハードウェアとしても実行可能

参考

- http://synthesijer.sourceforge.net
  - リソース一式，クイックスタートガイドなど
- http://qiita.com/kazunori279/items/4951ca5f6164040878ce
  - Kazunori279さん: Synthesijer関連資料まとめ
- http://labs.beatcraft.com/ja/index.php?Synthesijer
  - ビートクラフトさん: Altera DE0-nanoでのサンプルの動作手順など
- http://marsee101.blog19.fc2.com/blog-category-116.html
  - FPGAの部屋さん: Synthesijerでラプラシアンフィルタを作ってみた
- http://cellspe.matrix.jp/parallella/synthesijer.html
  - Parallela Fan!さん: SynthesijerでJavaプログラムからHDLコードを自動生成する
- http://wasa-labo.com/wp/
  - わさらぼ ブログ – 開発状況，Tipsなど

この話の落としどころ

- ✔ もっとこんな風に開発したらいいんじゃない？
- ✔ Javaのこの機能をさっさと実装して欲しい/できないの？
- ✔ Javaでやってる処理をFPGAにオフロードしてみたい
- ✔ そんなことやってる人もいるのね，とか…