
計算機科学実験及演習3 ハードウェア 「CADツールを用いた設計フロー」

京都大学 工学部情報学科 計算機科学コース

計算機科学実験及演習 3

ハードウェア担当

le3hw@kuis.kyoto-u.ac.jp



本講義の概要

- CADツールQuartus Primeを用いた設計フロー
 - HDL設計における使い方
 - FPGAへの書き込みと実機上での実行
- 進め方
 - 加算器を練習として作成してもらいます
 - HDL設計についてよく知っているのであれば練習は飛ばして、導入課題の7SEG LED駆動回路とカウンタの設計をどんどん進めていって構いません

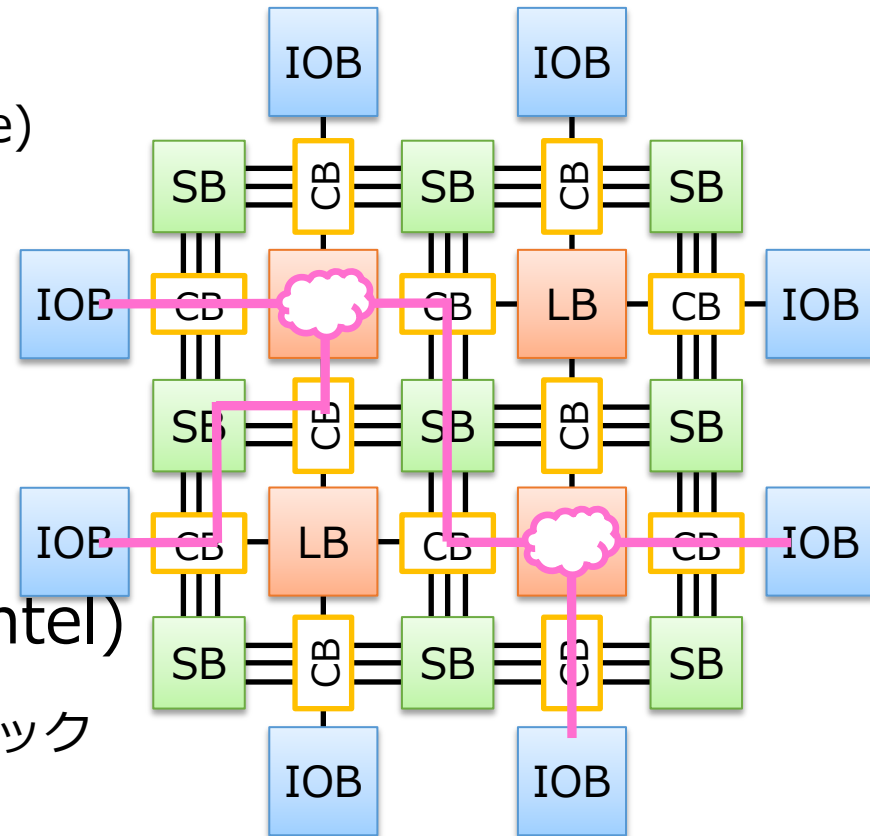
2023年度以降の注意点

- 本資料は新演習室環境を想定して作成しています
 - Ubuntu 22.04 LTS
 - Quartus Prime 20.1 Standard Edition
 - ModelSim Intel FPGA Starter Edition
- 各自のPC環境で作業する場合は適宜読み替えてください
 - 基本は大きく変わらないはずですが
 - 環境違いによる詰まりどころが多いようなら、Slack等で随時お知らせします

FPGAとは？

- Field Programmable Gate Array

- 中身を改変可能なLSI
(PLD: Programmable Logic Device)
- ハードウェアそのものの振る舞いを変えられる
- 独自のデジタル回路を自由に何回でも形成できる
- FPGA二大ベンダ：
AMD(旧Xilinx) , Altera(Intel)



LB 論理ブロック SB スイッチブロック

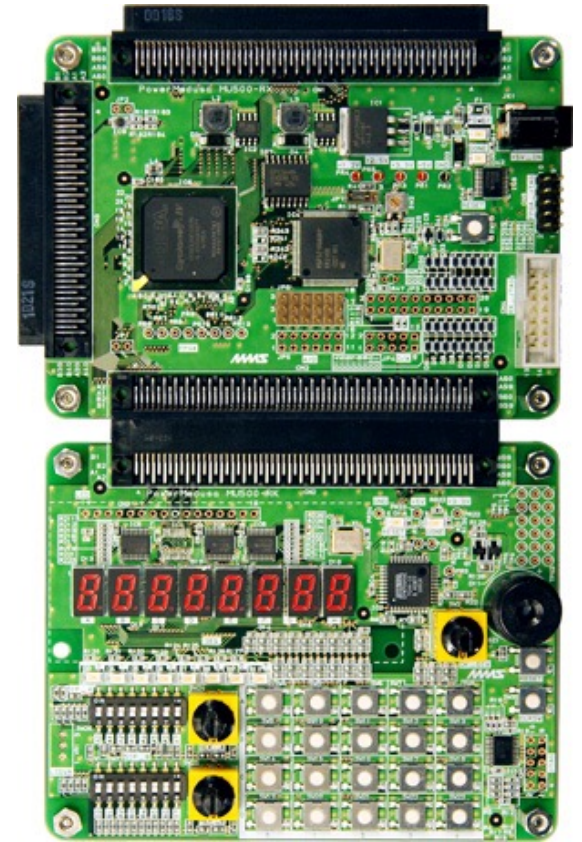
IOB 入出力ブロック CB コネクションブロック

FPGAの用途

- ASICの設計・検証用LSIに活用
 - ASICの機能を製造前に検証できる
 - HWの完成前からSWの開発が始められる
- 最終製品に組み込むLSIに活用
 - 民生品での実用例は枚挙に暇がないほど多い
 - 出荷後のデータ書き換え機能を持たせることも
 - 最近の実用例
 - ✓ 金融取引市場
 - ✓ ロボット
 - ✓ データセンター
 - ✓ 機械学習
 - ✓ オートモーティブ(ADAS)

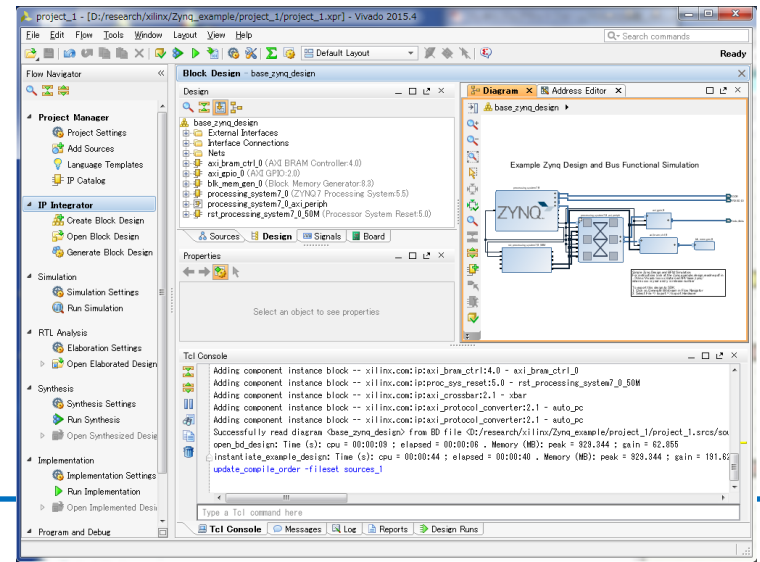
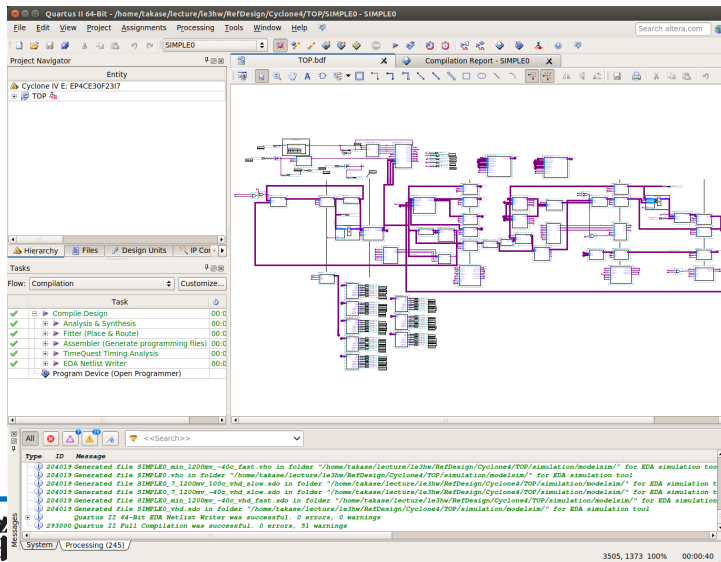
実験環境：FPGAボード

- PowerMedusa MU500-RX/RK
 - FPGA：Altera社Cyclone IV
EP4CE30F23I7N (28,840LE)
 - マイコン：ルネサス社RX210
 - クロック：20MHz発振器
 - 入出力ユーザインタフェース
 - ✓ プッシュSW x20, ロータリSW x2,
8bit DIP SW x2, クロックSW x1
 - ✓ 7SEG LED x8, LED x8, ブザー x1
 - 拡張ボード MU500-7SEG
 - ✓ 7SEG LED x64, LED x64



CADツール

- Computer-Aided Design:
論理回路を設計，合成するツール
 - 環境設定，回路の設計と記述，回路合成，ピンアサイン，シミュレーション，プログラミング（FPGAへの書き込み）までを行う
 - EDA(Electronic Design Automation)ツールとも呼ばれる



実験環境：CADツール（実験2と同じ）

- Intel Quartus Prime 20.1を用いる
 - 最新版ではない！
 - Google先生に聞く時にはバージョン番号に注意！
 - ✓ 公式マニュアルやドキュメントも別バージョンはアテにしない
- 各自のPCにもインストール可能
 - 機能制限有りの Lite Edition なら無償利用可能
 - ✓ シミュレータはModelSim-Intel FPGA Starter Editionを入れる
 - ▣ Starter無しIntel FPGA Editionは有償ライセンスが必要
 - ✓ 次ページの補足も参照
 - Windows/Linuxのみ（macOSは不可）

補足 : Quartus Primeのインストール

- 各自PCへのQuartus Primeの導入
 - 機能制限有りの Lite Edition なら無償利用可能です
 - シミュレータ ModelSim-Intel Starter Edition は無償利用可能です
 - ✓ Starter無しの ModelSim-Intel Editionは有償ライセンスが必要
- エディションの違いについて
 - Standard Editionだと30日間限定になります
 - Pro Editionは使用できません (ボード搭載のCyclone VIが未対応)
- 合成性能や最適化結果がエディションで異なってくる可能性があります
 - レポート記載の際には使用した Edition を明記してください



補足 : Quartus Primeのインストール

- **Quartus Prime 20.1 Lite Edition**のインストール方法
 - Quartus Primeのダウンロードサイト
<https://www.intel.co.jp/content/www/jp/ja/products/details/fpga/development-tools/quartus-prime/resource.html>
 - ✓ “Legal Disclaimer” で最新版じゃなくて良いの？の警告が出ることもあるが、無視して “I Agree” すればよい
 - ✓ 「一式ファイル」の場合は 5.9GB (Linux版 6.4GB)
 - ✓ 「個別ファイル」の場合は下記を同場所にダウンロード
 - Quartus Prime (includes Nios II EDS)
 - ModelSim-Intel FPGA Edition (includes Starter Edition)
 - Cyclone IV device support
 - (次ページに続く)

補足 : Quartus Primeのインストール

- **Quartus Primeのダウンロード**

1. QuartusPrimeのダウンロードページにアクセス。
2. Versionから"20.1"を選択
3. 「Multiple Download」タブから一式ファイルをダウンロード。または「Individual Files」タブから個別ファイル（前ページ参照）をダウンロード。

- **必要なソフトウェアのインストール**

- Quartusが動作するためにインストールする必要があるソフトウェアがある。以下にアクセスして、右上の"Download"からPDFドキュメントを取得。
 - ✓ <https://www.intel.com/content/www/us/en/docs/programmable/683472/20-1/introduction.html>
- PDFドキュメントの"2.3 Software Requirements"に記載のパッケージをインストールする



補足 : Quartus Primeのインストール

- **Quartus Prime 20.1 Lite Edition**のインストール方法
 - 「個別ファイル」の場合は, 3 ファイルを同位置に配置してください
 - インストーラを起動し, 適宜チェックを入れて実行してください
 - ✓ Installation directory はデフォルトで良いでしょう
(本資料等での説明はデフォルト位置を想定しています)
 - ✓ Select Components では, 下記のチェックが必須です
 - Quartus Prime (includes Nios II EDS)
 - Devices > Cyclone IV
 - ModelSim - Intel FPGA **Starter** Edition

Windowsの場合は, Quartus本体のインストール終了後に表示される USB Blaster II device driverもインストールしてください

- ✓ FPGAボードのUSBケーブルがうまく認識しない場合は,
[こちらのページ](#)を参考にしてください.

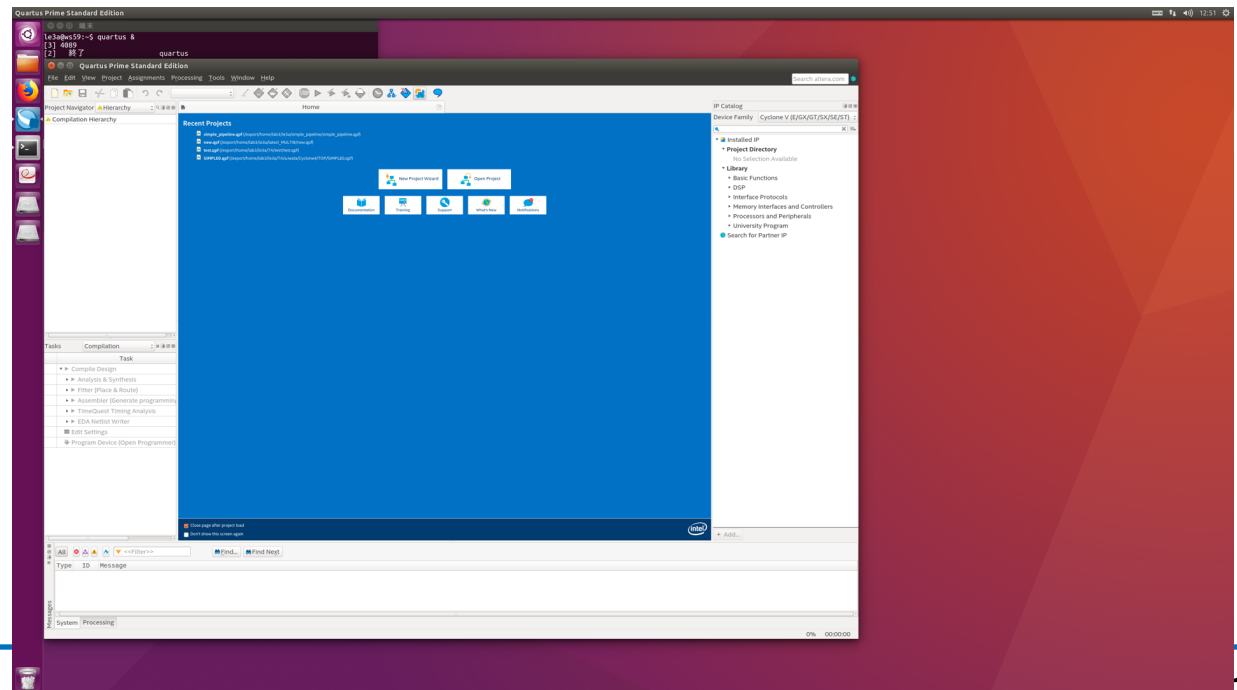
Quartusの起動

- ターミナル上で

```
$ /opt/intelFPGA/20.1/quartus/bin/quartus &
```

– 最後に&を入れるとバックグラウンドプロセスとして実行

✓ <https://ja.wikipedia.org/wiki/バックグラウンドプロセス>



PATHを通す方法

- 環境変数（\$PATH）の記述を変更し，コマンドの検索パスを追加する
- ターミナル上で

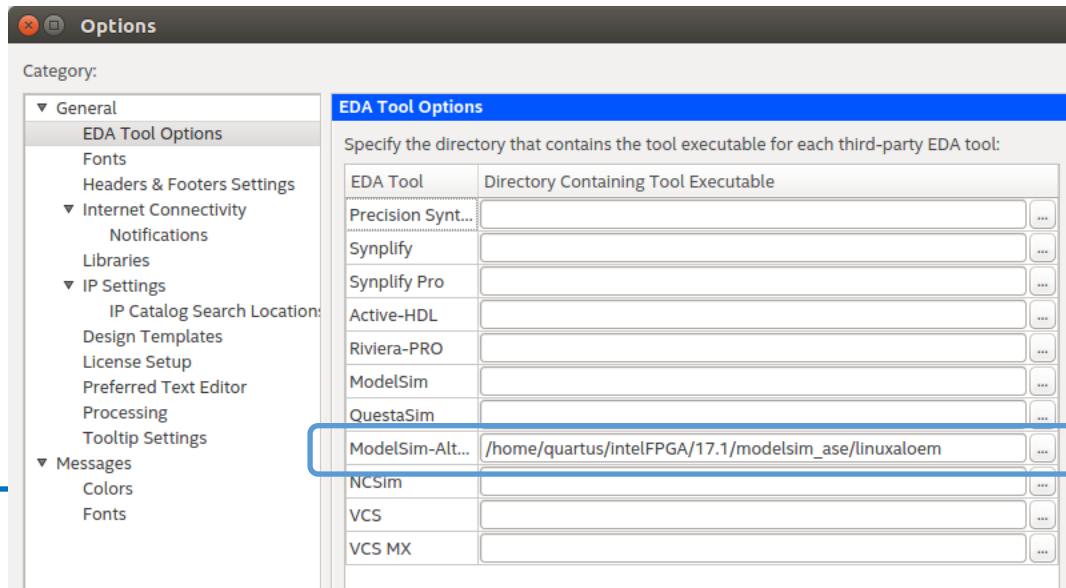
```
$ export PATH="$PATH:/opt/intelFPGA/20.1/quartus/bin"
```
- こうすると，単にquartusと打つだけで起動できるようになる
- ~/.bashrcに上のコマンドを書いておくと，シェル（Bash）を起動する度に実行されるので，毎回設定する必要がなくなる

設定の確認

- ライセンス認証
 - 初回起動時, If you have a valid license file, specify the location of your license fileにチェックしてOKをクリック
 - “License file: ” に “27003@is-fpg-10” を入力するか “LM_LICENSE_FILE variable:” にチェックして入力
 - 起動後, Tools -> License Setup..からも設定可能
- Webブラウザの設定
 - Tools -> Options... の Internet Connectivity で, “Web browser:” に好みのブラウザへのパスを設定
 - 何も好みがないければ “/export/share/bin/firefox”
 - ✓ エラー調査時に便利です

設定の確認

- Quartusからmodelsim-aseを呼び出すための設定をする
 - Tools -> Options... -> EDA Tool Options:
ModelSim-Altera: /opt/intelFPGA/20.1/modelsim_ase/linuxaloem
 - ✓ 's' が抜けていることがあるので注意
 - ✓ いったん空白にしてもよい



ツール利用と設計の流れ

- a. 新規プロジェクトの作成と環境設定
- b. HDLコードの作成
- c. コンパイル
- d. タイミング制約の設定と検証
- e. シミュレーションによる動作確認
- f. トップレベル回路への入出力ピンの
- g. FPGAへの回路の書き込み

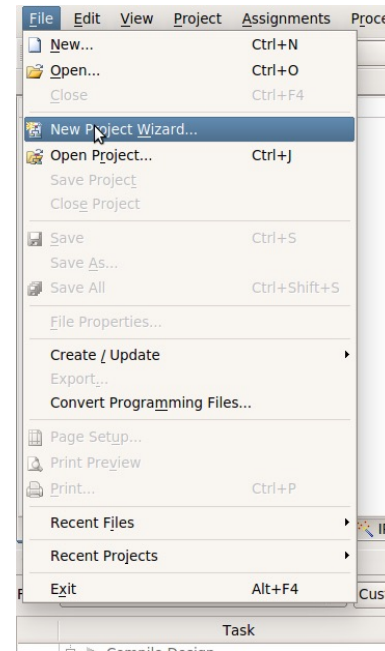
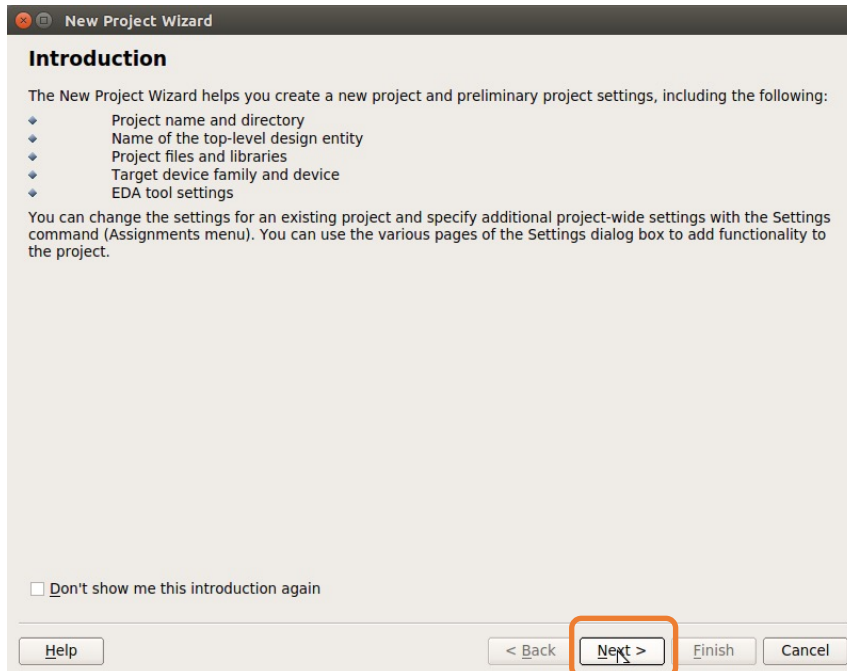
練習は組合せ回路なので無し

4ビット加算器をお題として設計フローを説明します

練習：4bit加算機的设计

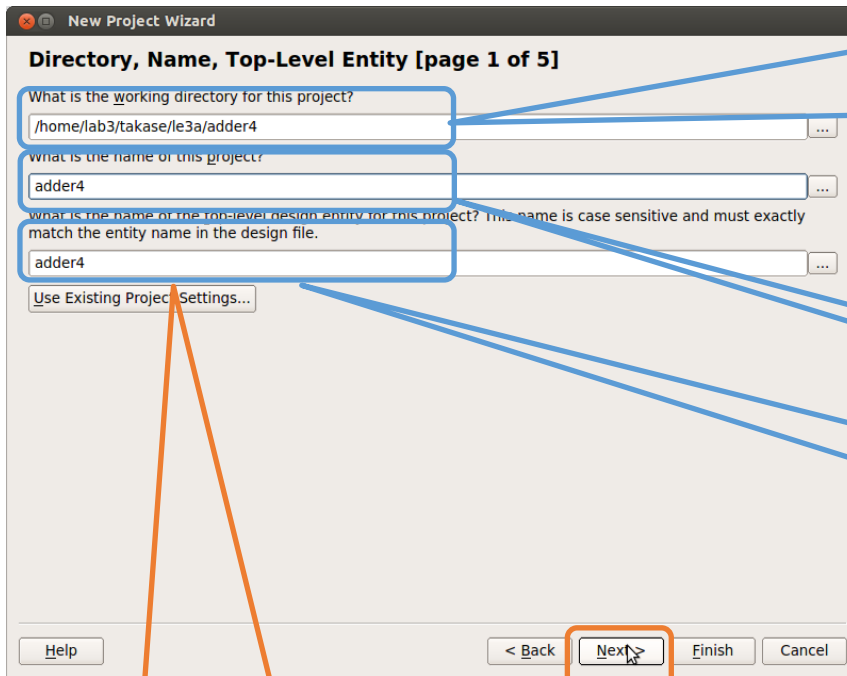
a. 新規プロジェクトの作成と設定 (1/4)

- File -> New Project Wizard...
- Introduction “Next”



a. 新規プロジェクトの作成と設定 (2/4)

• Directory, Name, Top-Level Entity

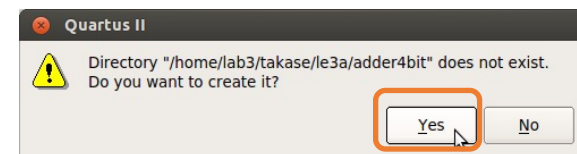


作業ディレクトリ：
ホームディレクトリそのものを指定せず、
必ず専用のディレクトリを作ること。そう
しないとツールが固まったり落ちたりする
原因になる。
また、プロジェクト毎にディレクトリを
作ったほうがよい（名前は自由）

プロジェクト名

トップレベルの回路名：
なにも気にしなければプロジェクト名と
同一（自動入力される）

数字から始まる、
または-(ハイフン)を含む
プロジェクト名や回路名は
随所でエラーが出るので避ける。



作業ディレクトリが存在しない
場合は自動作成してくれる

a. 新規プロジェクトの作成と設定 (3/4)

- Project Type : Empty projectを選択
- Add Files : 作成済みの回路をプロジェクトに追加する
 - 今回はまだ何も無いので空白のまま"Next"
- Family & Device Settings
 - Family:
"Cyclone IV E"
 - Available devices:
"EP4CE30F23I7"

選択が面倒な場合は
フィルタ検索も可能

Available devices:から
EP4CE30F23I7を選択

Family & Device Settings [page 3 of 5]

Select the family and device you want to target for compilation.
You can install additional device support with the Install Devices command on the Tools menu.

Device family

Family: Cyclone IV E

Devices: All

Target device

Auto device selected by the Fitter

Specific device selected in 'Available devices' list

Other: n/a

Show in 'Available devices' list

Package: Any

Pin count: Any

Speed grade: Any

Name filter: EP4CE30F

Show advanced devices HardCopy compatible only

Available devices:

| Name | Core Voltage | LEs | User I/Os | Memory Bits | Embedded multiplier 9-bit |
|--------------|--------------|-------|-----------|-------------|---------------------------|
| EP4CE30F23I7 | 1.2V | 28848 | 329 | 608256 | 132 |
| EP4CE30F23I7 | 1.2V | 28848 | 329 | 608256 | 132 |

Companion device

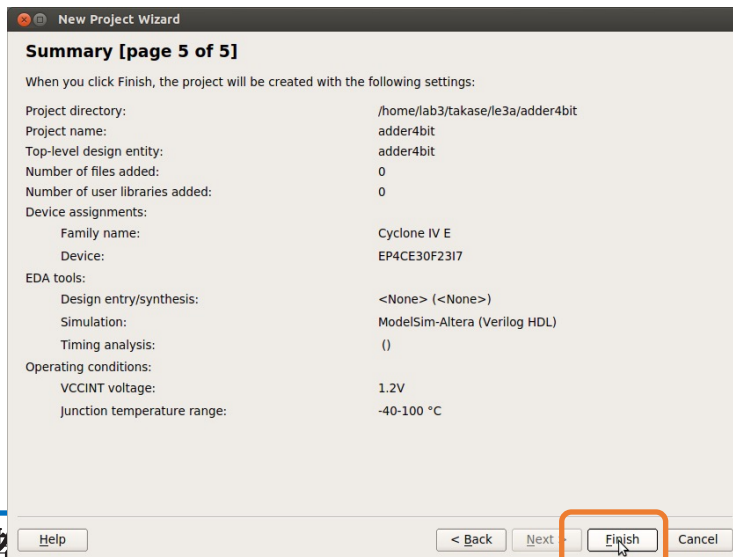
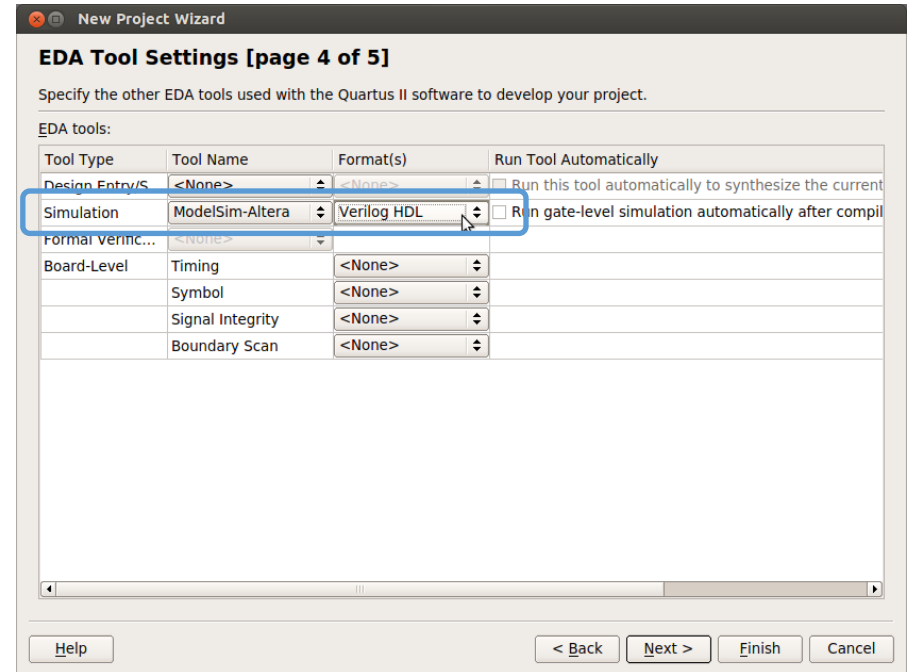
HardCopy:

Limit DSP & RAM to HardCopy device resources

Help < Back Next > Finish Cancel

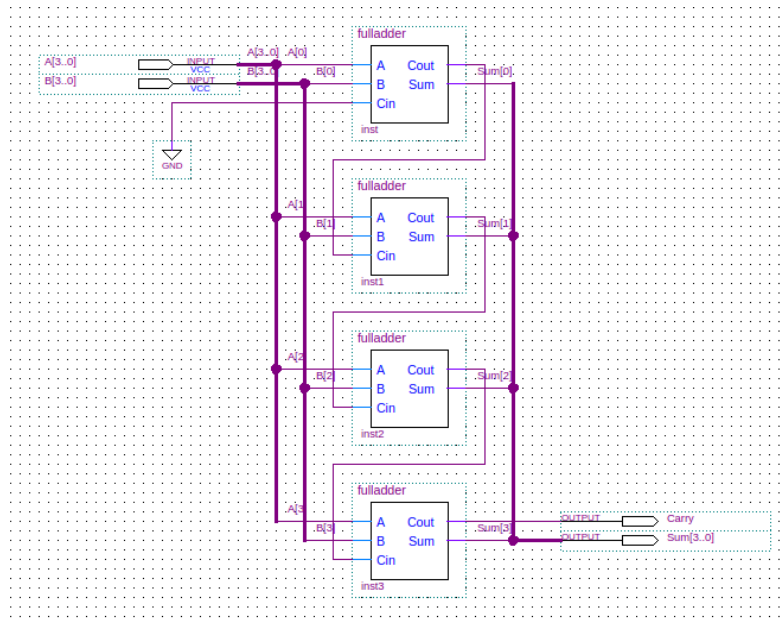
a. 新規プロジェクトの作成と設定 (4/4)

- EDA Tool Settings
 - 外部の設計ツールを使用／変更する場合に指定する
 - “Simulation”にて使用するHDLを指定する
- Summary
 - 設定を確認して“Finish”



実験2では…

- ブロック図によって加算器を設計した
 - これだけでも手間がかかった
- 実験3ではハードウェア記述言語を用いることで飛躍的に生産性が高まる

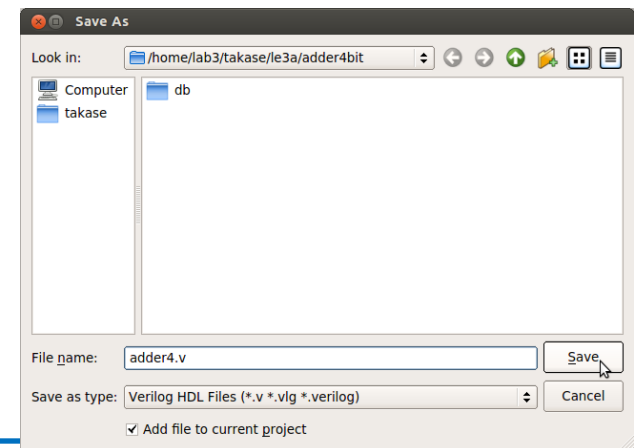
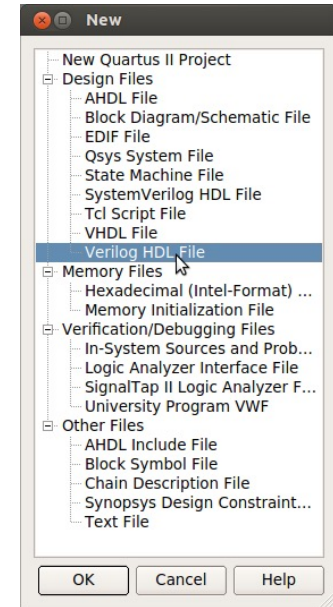


b. HDLコードの作成 (1/3)

本実験はVerilog-HDL前提

他の言語だと教員・TAもわからない可能性大

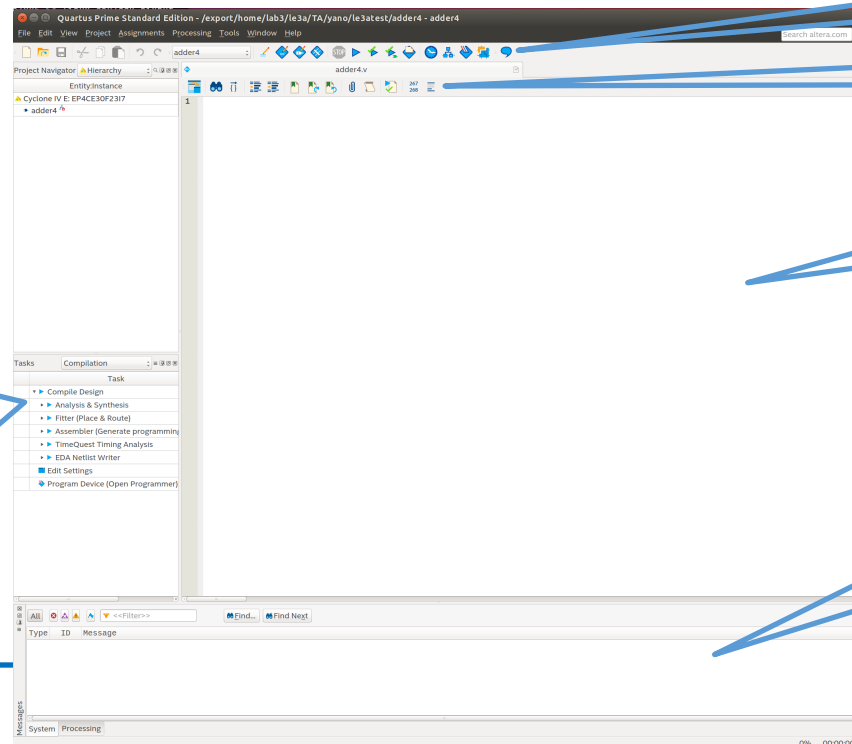
- HDLエディタを使用する
 - File -> New -> Verilog HDL File または File -> New -> VHDL File
 - ✓ 自分の使いたいHDLに合わせて選択する
 - 実験3ではハードウェア記述言語による設計を存分に楽しんでください
 - ✓ 回路図との組合せもOKです
 - ✓ トップレベルは回路図, 部品はHDLとすると全体の見通しが良くなります
- まずは(泣く前に)HDLファイルを保存
 - File -> Save As...
 - “adder4.v”
 - ✓ VHDLの場合は “adder4.vhd”
- Tasks窓のFlow:を “Compilation”にする



数字から始まる
ファイル名は避ける

b. HDLコードの作成 (2/3)

- (初期)ウィンドウの構成
 - 使いやすいようにレイアウトやサイズ等を適宜調整してよい
View -> Utility Windows など
 - エディタ機能は貧弱なので、お好みのエディタを使用してもよい
(外部ツールとのファイル同期は自己責任で)



ファイルの一覧や
呼び出し関係

ツール操作アイコン

エディタ操作アイコン

エディタ画面

メッセージウィンドウ
処理メッセージや
エラー報告など

タスク(コンパイル等)の
進行状況の表示や
レポートの選択
Flow: "Compilation"とする

b. HDLコードの作成 (3/3)

- Verilog HDLの例

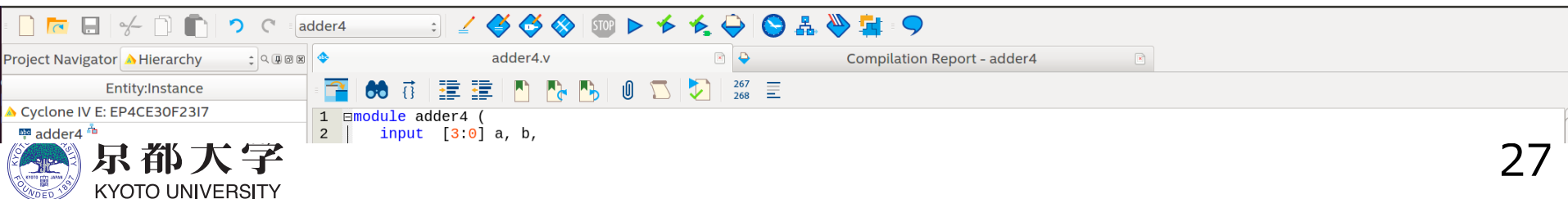
```
adder4.v
1 module adder4 (
2     input  [3:0] a, b,
3     input  cin,
4     output [3:0] sum,
5     output cout );
6
7     assign {cout, sum} = a + b + cin;
8 endmodule
```

- VHDLの例

```
adder4.vhd
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.std_logic_unsigned.all;
4
5  entity adder4 is
6  port (
7      a, b : in  std_logic_vector(3 downto 0);
8      cin  : in  std_logic;
9      sum  : out std_logic_vector(3 downto 0);
10     cout : out std_logic );
11  end adder4;
12
13  architecture adder4_body of adder4 is
14  signal tsum : std_logic_vector(4 downto 0);
15  begin
16     tsum <= ('0' & a) + ('0' & b) + cin;
17
18     cout <= tsum(4);
19     sum  <= tsum(3 downto 0);
20  end adder4_body;
21
```

c. コンパイル (1/4)

- FPGA内の論理ブロックの構成情報に変換する
 - 回路図をレジスタ転送レベルに変換する
 - 論理合成, テクノロジマッピング, 配置配線を行う
 - ✓ RTLで記述された回路をFPGA上の論理ブロックに割り付け, 論理ブロックの配置や配線を行う
 - ビットストリームを生成する
 - ✓ 配置/配線の情報をバイナリ記述の構成情報ファイルに変換する
- コンパイル手順
 - Processing -> Start Compilation または
ツール操作アイコンのStart Compilation または Ctrl+L
 - ✓ Start Analysis & Synthesis で入力チェックのみも可能



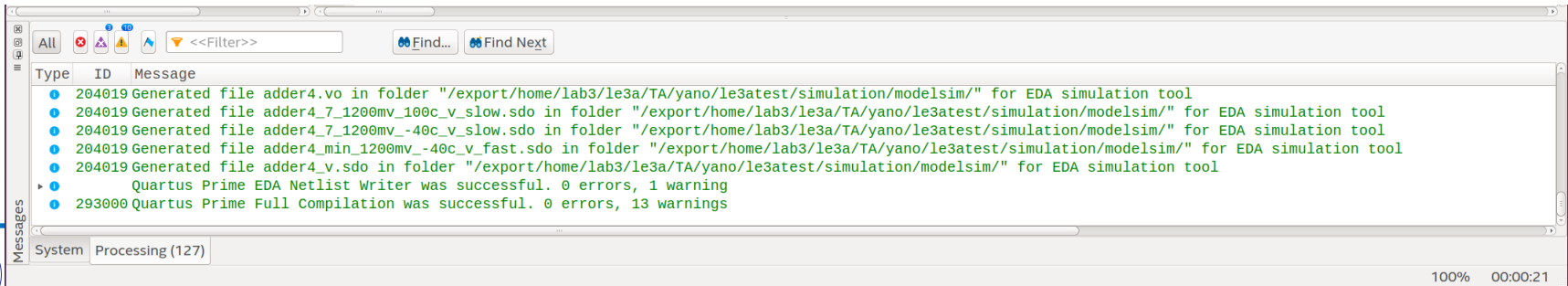
The screenshot shows the Quartus II IDE interface. The top toolbar contains various icons for file operations and compilation. The Project Navigator on the left shows the project hierarchy. The Entity: Instance pane shows the selected entity 'adder4'. The code editor displays the following RTL code:

```
1 module adder4 (  
2 |   input  [3:0] a, b,
```

The bottom status bar shows the target device as 'Cyclone IV E: EP4CE30F2317' and the compilation report for 'adder4'.

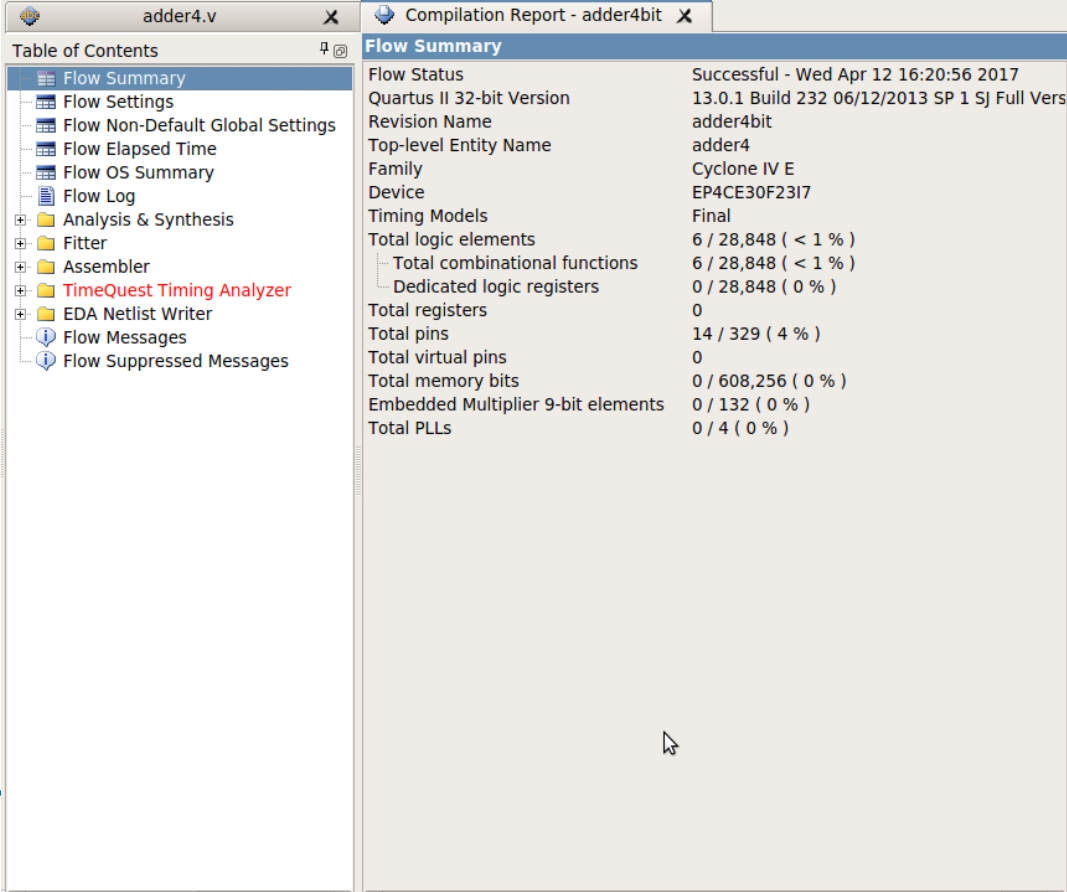
c. コンパイル (2/4)

- Tasks窓の“Compilation”がall green✓になったら正常にコンパイル成功
 - 黄字?はそのフローで再コンパイルが必要なことを表す
 - 赤字×はそのフローでエラーが発生していることを表す
- 正常終了しなかったら、下部のメッセージウインドウ または “Compilation Report -> Flow Messages” などを確認し、メッセージをもとにデバッグを行う
 - 右クリック Help でメッセージのWeb検索可能
 - Critical Warning / Warning も要チェック
 - ✓ CADツールでは警告がシビアに出力されがちだが、無視する場合も警告の意味を理解しておくこと（設計ミスに気づくことも多い）



c. コンパイル (3/4)

- Compilation Report で合成結果を確認する
 - Processing -> Compilation Report または Ctrl+R
 - 各項目の詳細は各自で調査すること



The screenshot shows the Quartus II software interface with the 'Compilation Report - adder4bit' window open. The 'Flow Summary' tab is selected, displaying the following data:

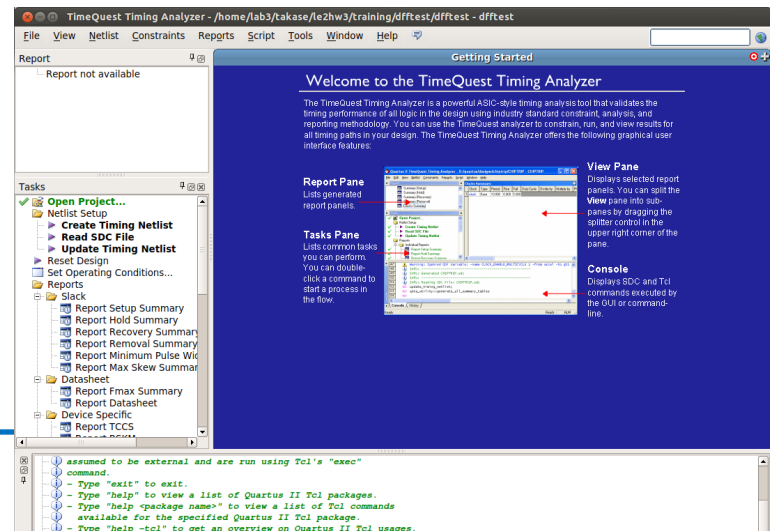
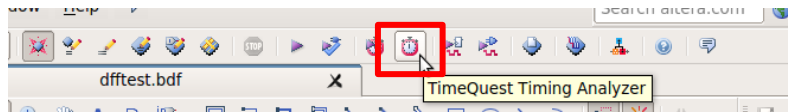
| Item | Value |
|------------------------------------|---|
| Flow Status | Successful - Wed Apr 12 16:20:56 2017 |
| Quartus II 32-bit Version | 13.0.1 Build 232 06/12/2013 SP 1 SJ Full Vers |
| Revision Name | adder4bit |
| Top-level Entity Name | adder4 |
| Family | Cyclone IV E |
| Device | EP4CE30F2317 |
| Timing Models | Final |
| Total logic elements | 6 / 28,848 (< 1 %) |
| Total combinational functions | 6 / 28,848 (< 1 %) |
| Dedicated logic registers | 0 / 28,848 (0 %) |
| Total registers | 0 |
| Total pins | 14 / 329 (4 %) |
| Total virtual pins | 0 |
| Total memory bits | 0 / 608,256 (0 %) |
| Embedded Multiplier 9-bit elements | 0 / 132 (0 %) |
| Total PLLs | 0 / 4 (0 %) |

c. コンパイル (4/4)

- 回路サイズ : Fitter -> Summary
 - FPGA上の各ブロックの使用量
 - 100%を超える項目があるとFPGAデバイスに収まらない
- 動作速度と遅延 : Timing Analyzer
 - SDCでset_max_delay, set_min_delayを設定すればそれぞれの入力端子から出力端子への伝搬遅延時間 (単位はns) を表示できる (FAQ参照)
 - 順序回路の設計で入力クロックを指定すれば, 動作周波数なども表示できるようになる
 - 赤字の項目がある場合はタイミング制約を満たしていない
 - ✓ **Unconstrained Paths**は set_max_delay, set_min_delayを適切に設定すれば消える
順序回路設計では特に気にする必要がある

d. タイミング制約の設定(参考)

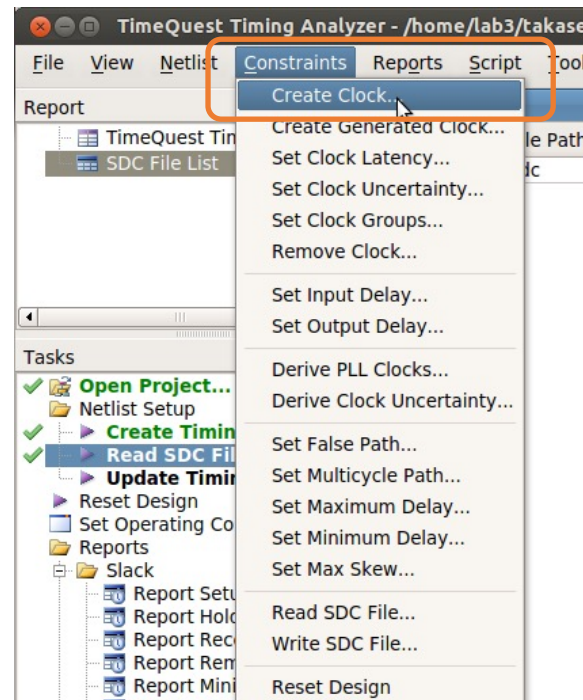
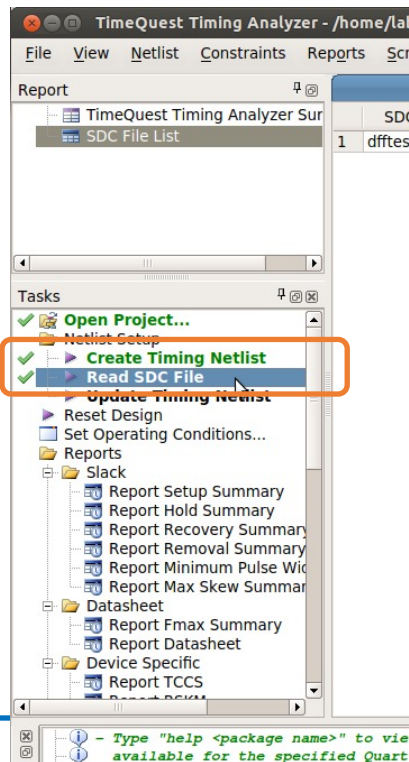
- 練習の加算器は組み合わせ回路なので設定の必要はないが、クロックを使用する順序回路では必ず設定する。
- クロックの指定やタイミング制約（レジスタ入力へのセットアップ時間やホールド時間，伝搬遅延など）の設定を行う
 - 基本的なクロックの指定のみ解説，詳細は各自で調査すること。
- Timing Analyzerを使用する
 - Tools -> Timing Analyzer または，
ツール操作アイコンの“Timing Analyzer”



d. タイミング制約の設定(参考)

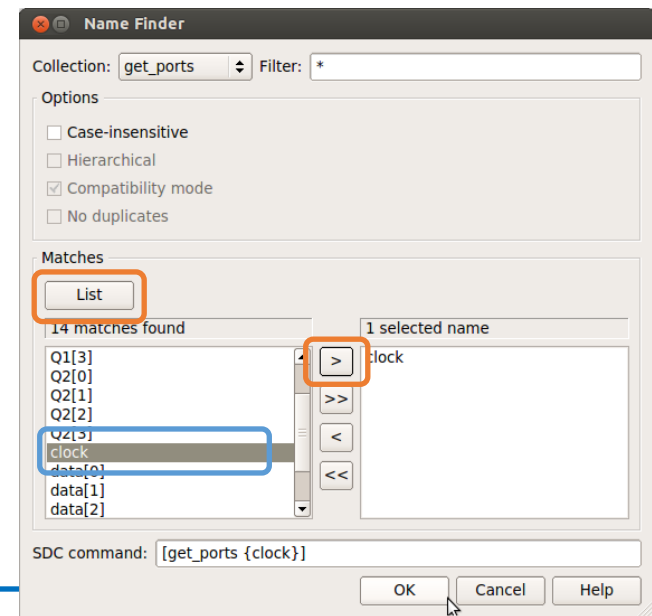
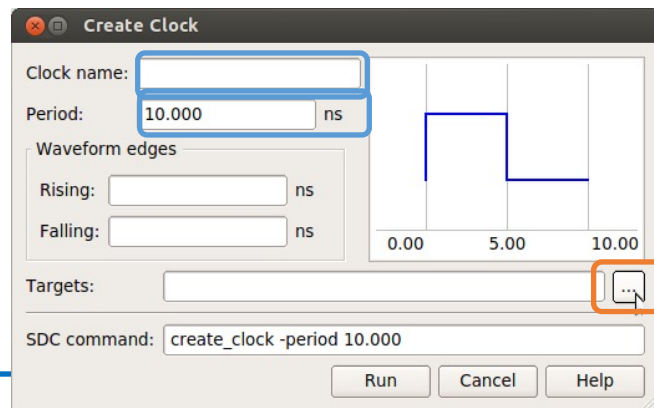
- Timing Analyzer

- Tasks窓の “Netlist Setup -> Create Timing Netlist” を実行
- Tasks窓の “Netlist Setup -> Read SDC File” を実行
- Constraints -> Create Clock... からクロックを設定



d. タイミング制約の設定(参考)

- Timing Analyzer
 - Constraints -> Create Clock... からクロックを設定
 - ✓ Clock name: クロック名の定義 未記入でも可 (例: clock)
 - ✓ Period: クロック周期の設定
 - ✓ Waveform edges: 立ち上がり/立ち下がり時間の設定
 - ✓ Targets: 設計回路上のクロックを選択
 - “List”から表示して”>”で選択
 - ✓ SDC commandに生成される制約が表示される



d. タイミング制約の設定(参考)

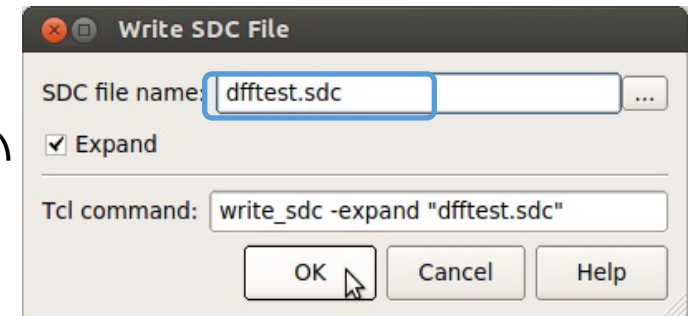
- Timing Analyzer

- Reports/ 以下でタイミングに関する種々のレポートが報告されるようになる

- ✓ 満足していないものがあれば制約の追加などで対処する
- ✓ Compilation Report からも確認できるようになる
- ✓ Fmax Summary: 動作可能な最大周波数 (与える制約によって変わる)
- ✓ Clocks: 現在設定値での設計回路のクロック周波数

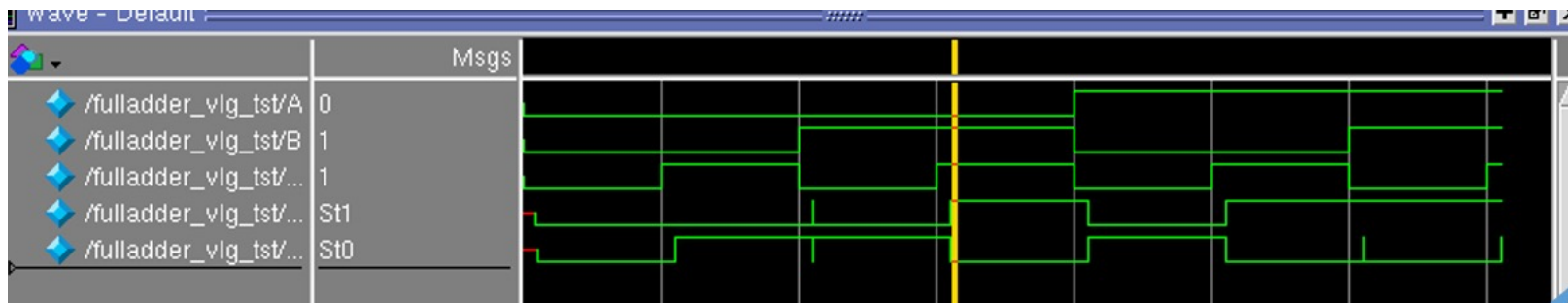
- Tasks窓の “Write SDC File...” で制約ファイルを保存

- ✓ <module>.sdc とするとプロジェクトに自動認識される (.outを取ったほうがよい)
- ✓ 慣れてきたらテキストベースで制約ファイルを作成して編集してもよい



e. シミュレーション (1/12)

- まずはテストベンチファイルを作成する
 - テストベンチファイルとは、設計した回路が期待通りの動作をするか確認するためのファイルのこと。
 - ハードウェア記述言語Verilog HDLで記述する（実験3では文法の理解もすること）
- 回路に任意の入力を与え、シミュレータModelSimで出力を観測する



e. シミュレーション (2/12)

- テストベンチファイルを作成する手順
 - Processing -> Start -> Start Test Bench Template Writer
 - ./simulation/modelsim/*.vt が作成される
- *.vt を編集 もしくは コピーして新しいファイルを作る
 - ✓ .vにするとエディタによってはverilogモードが使える
 - ✓ 時間単位を変更する
 - ``timescale 1 ns / 1ps`
 - 1タイムステップが1ns、丸め精度（最小単位）が1psという意味
 - ✓ 以下の記述の直下に初期設定を記述する
 - `// code that executes only once`
 - `// insert code here --> begin`
 - ✓ 以下の記述の直下に所望の値（波形）を記述する
 - `// code that executes for every event on sensitivity list`
 - `// insert code here --> begin`
 - **eachvec信号は使わないので@eachvec;はコメントアウトか削除する**
- 慣れてきたらTemplate Writerを使わなくても良い

e. シミュレーション (3/12)

- テストベンチファイルの記法例

- 代入 :

```
A <= 0;  
B <= B + 1;  
Cin <= ~Cin; //反転  
As <= 4'b10_01; // 4ビット  
の2進数 (binary)
```

- ✓ (遅延が無ければ)並列に
代入される

- 遅延 :

```
#1000
```

- 繰り返し :

```
always begin  
    #100  
    clock <= ~clock;  
end
```

- ✓ クロック波形の生成に便利
 - ✓ initial begin - end や
always begin - end の外側,
module - endmodule の
内側に記入すること
 - ✓ alwaysブロック同士は
並列に実行される
 - ✓ **eachvecは未使用のため
@eachvec;があると繰り返
されない**

Verilog-HDLで書かれています。
何も指定しないと数字は十進数となります。

e. シミュレーション (4/12)

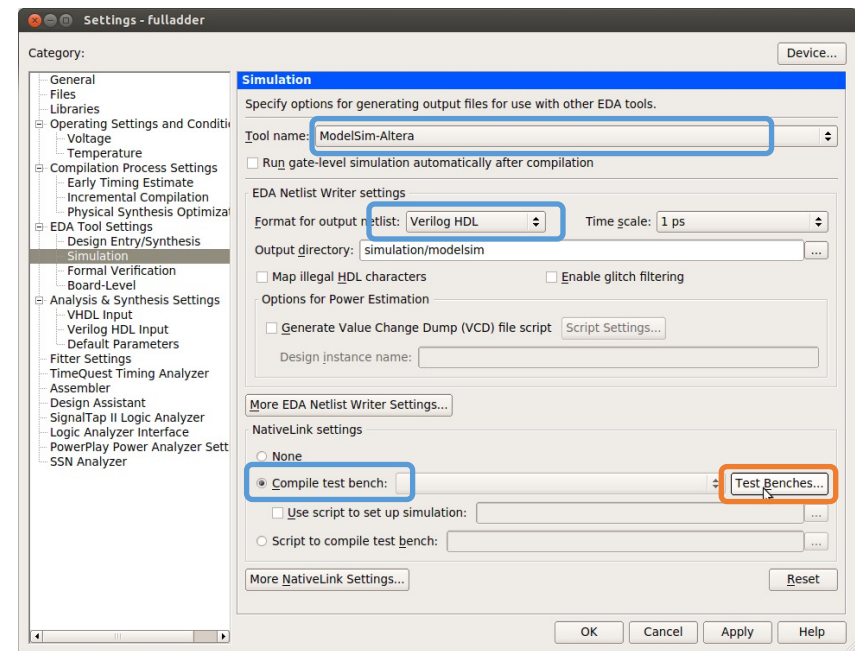
- テストベンチファイルの記述例
 - 時間単位も変更済み
 - ✓ 1タイムステップが1ns、丸め精度 (最小単位) が1ps
 - この例は安直な真理値表の網羅なので、工夫してみてください
 - 単位や遅延はよく考える
 - ✓ あまりに小さいと??

```
fulladder_test1.vt (/home/lab3/takase/le2hw3/training/fulladder)
開く 保存 元に戻す
fulladder_test1.vt x
);
initial
begin
// code that executes only once
// insert code here --> begin
A <= 0;
B <= 0;
Cin <= 0;

// --> end
$display("Running testbench");
end
always
// optional sensitivity list
// @(event1 or event2 or .... eventn)
begin
// code executes for every event on sensitivity list
// insert code here --> begin
#100
Cin <= 1;
#100
B <= 1;
Cin <= 0;
#100
Cin <= 1;
#100
A <= 1;
B <= 0;
Cin <= 0;
#100
Cin <= 1;
#100
B <= 1;
Cin <= 0;
#100
Cin <= 1;
@eachvec;
// --> end
end
endmodule
```

e. シミュレーション (5/12)

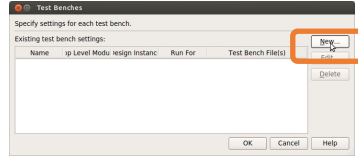
- Quartusからmodelsim-aseを呼び出すための設定をする
 - Assignment -> Settings -> EDA Tool Settings -> Simulation
 - ✓ Tool name: ModelSim-Altera
 - ✓ Format for output netlist:
Verilog HDL
(VHDLでの設計時はVHDL)
 - ✓ NativeLink settings:
Compile test bench
 - ✓ Test Benches... をクリック



e. シミュレーション (6/12)

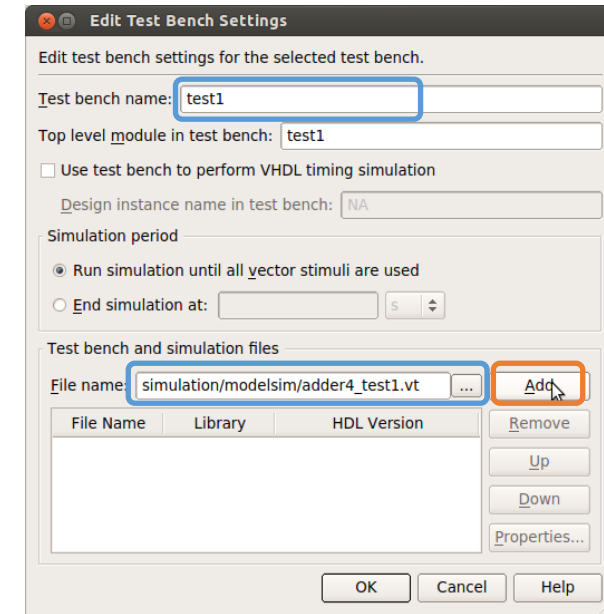
– Test Benches

- ✓ New... をクリック

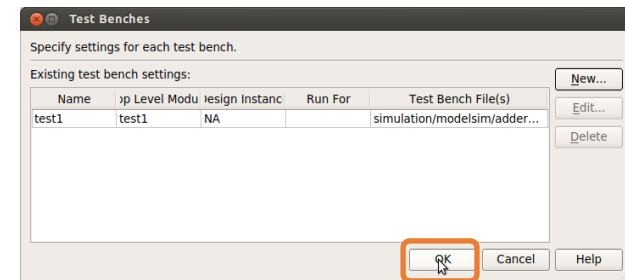


– New Test Bench Settings

- ✓ Test bench name: 任意の名前
- ✓ Top level module in test bench: を
実際のテストベンチモジュール名にする
(Template writerを使った場合, *_vlg_tst
になっている)
- ✓ Test bench and simulation files:
simulation/modelsim/*.vt
(...から前ページで作成したファイルを選択)
- ✓ Add をクリック



– Test Benchesに戻ってOK, Applyをクリック



e. シミュレーション (7/12)

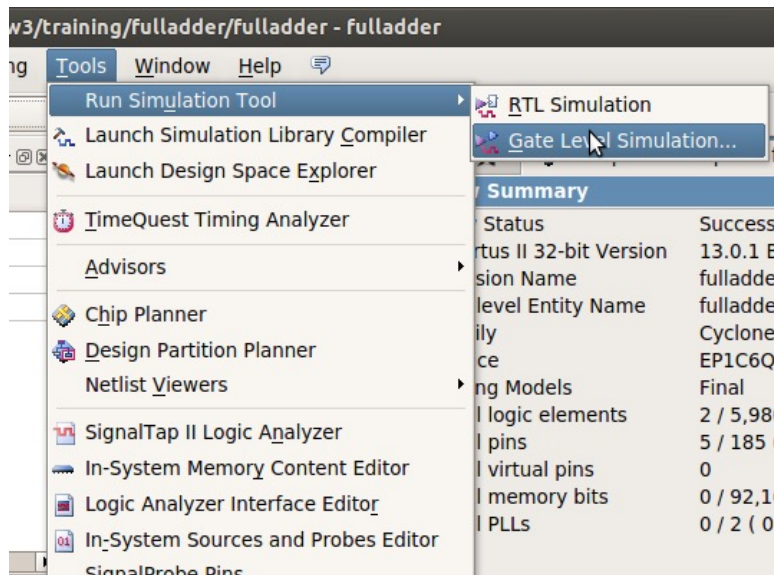
- テストベンチファイルの記述例
 - 時間単位も変更済み
`timescale 1 ns / 1ps
 - 単位や遅延はよく考える
 - ✓ あまりに小さいと??

```
);  
initial  
begin  
// code that executes only once  
// insert code here --> begin  
a <= 4'b0000;  
b <= 4'b0000;  
cin <= 0;  
  
// --> end  
$display("Running testbench");  
end  
always  
// optional sensitivity list  
// @(event1 or event2 or .... eventn)  
begin  
// code executes for every event on sensitivity list  
// insert code here --> begin  
#100  
a <= 4'b1100;  
b <= 4'b0010;  
cin <= 0;  
  
#100  
a <= 4'b0110;  
b <= 4'b1010;  
cin <= 0;  
  
#100  
a <= 4'b0110;  
b <= 4'b0011;  
cin <= 1;  
  
#100  
a <= 4'b1001;  
b <= 4'b0100;  
cin <= 1;  
  
//@eachvec;  
// --> end  
end  
endmodule
```

@eachvec;は
コメントアウト

e. シミュレーション (8/12)

- 設計を再コンパイルする (タスク窓が ? になっているはず)
- シミュレータを起動する
 - Tools -> Run Simulation Tool -> Gate Level Simulation...



e. シミュレーション (9/12)

Objectsウィンドウ
観測したい信号を
Waveにドラッグ

Restart

時間単位の指定

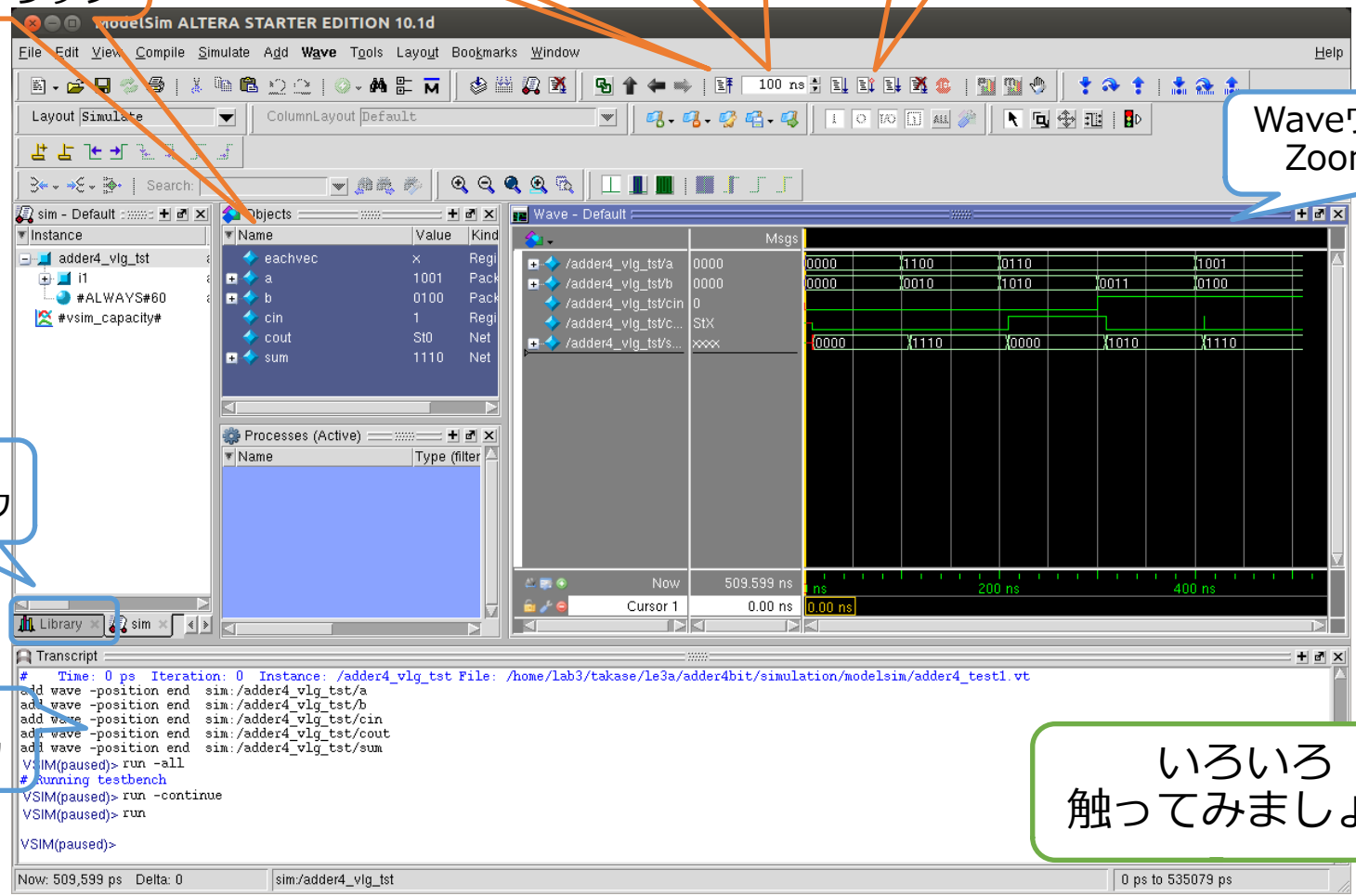
Run等

Waveウィンドウ
Zoom変更可

Library
ウィンドウ

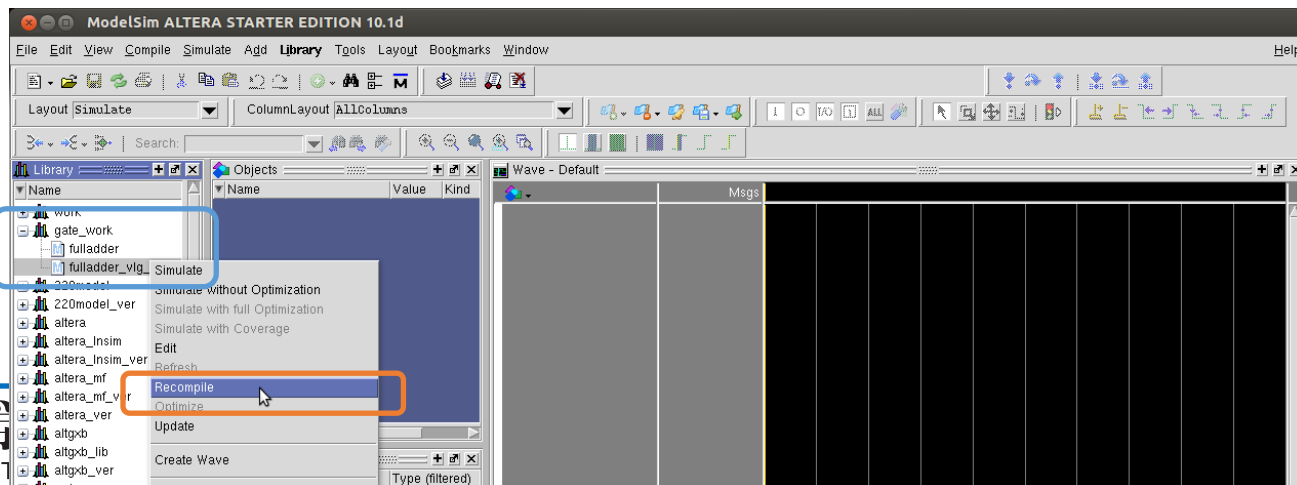
Transcript
ウィンドウ

いろいろ
触ってみましょう



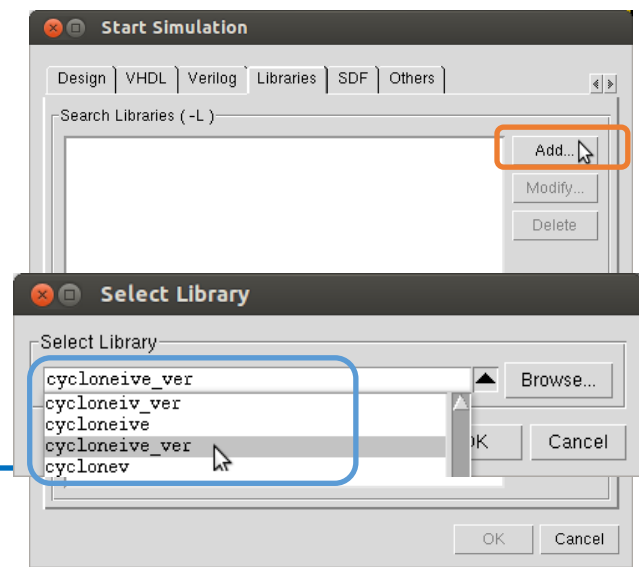
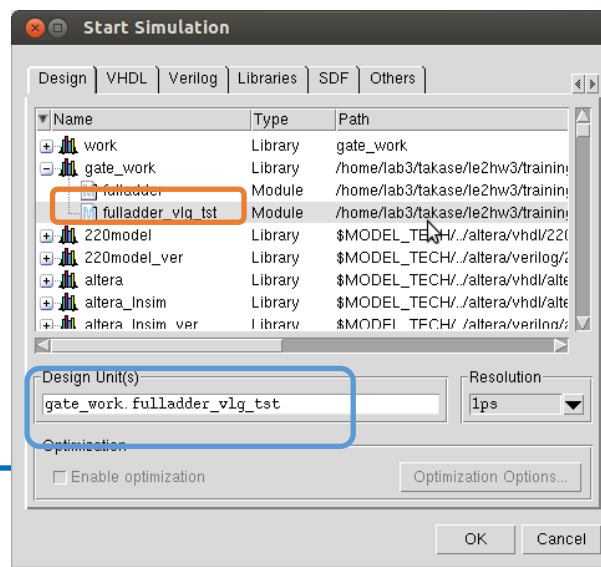
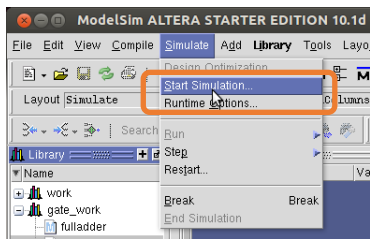
e. シミュレーション (10/12)

- 波形が表示されない場合
 - テストベンチファイルを念のため再コンパイルしよう
 - ✓ “Library” ウィンドウの gate_work -> <module>_vlg_tst を右クリックして “Recompile”
 - Transcriptionウィンドウに# Error loading design”が出ているか確認
 - ✓ 再コンパイルしても出ている場合, テストベンチファイルにミスがあるか, ファイルの指定が間違っているなどの可能性がある



e. シミュレーション (11/12)

- 波形が表示されない場合 (その2)
 - Simulate -> Start Simulation...
 - Design タブの "Design Unit(s)" にて gate_work.<module>_vlg_tst を選択
 - Libraries タブの "Search Libraries" にて Add... して必要なライブラリを追加する ("cycloneive_ver"のみ選択でよい)
 - ✓ "altera_ver", "gate", "gate_work" が自動的に入っていることがあるが、どちらでもよい



e. シミュレーション (12/12)

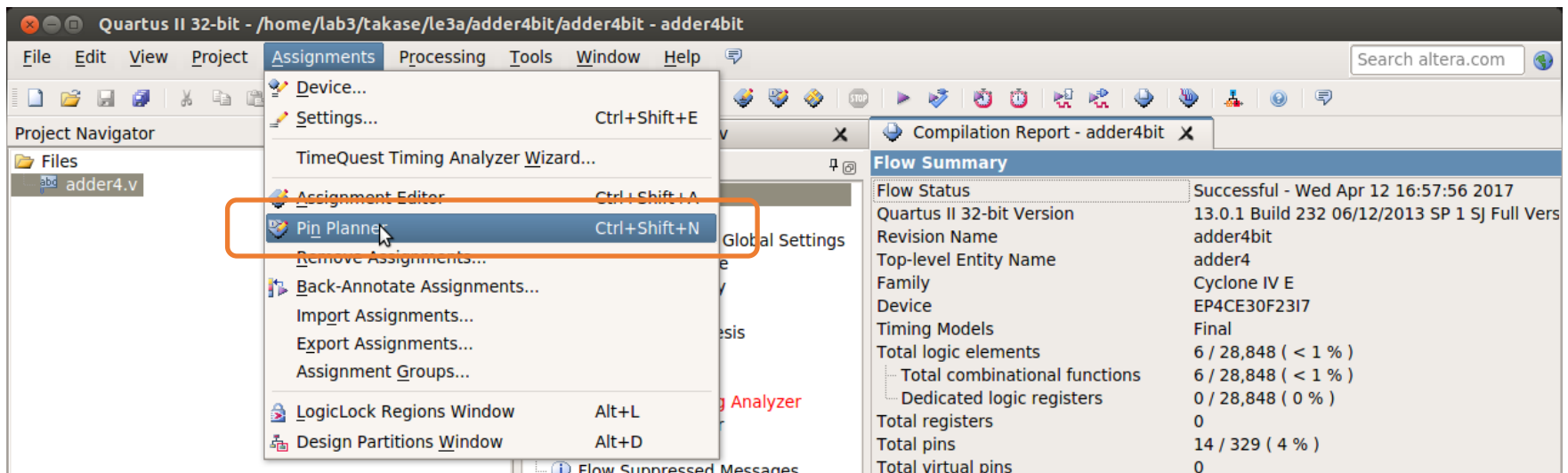
- その他の操作
 - “Objects” ウィンドウから観測したい信号線を “Wave” ウィンドウにドラッグする
 - “Run Length” を所望の大きさにする (例 : 100ns)
 - ✓ 大きすぎるとシミュレーション時間が掛かる
 - “Run” や “Run - All” などをクリックする
 - ✓ “Transcript” ウィンドウで直接入力も可能 (慣れてきたらこっちのほうが早いかも)
 - 波形の表示は右クリックまたはキー入力で Zoom In/Out/Fullなどが変更可能
 - 波形はテストベンチで指定済みだが, 直接指定も可能
 - ✓ 信号線の右クリックで “Force...” など
 - テストベンチを変更した場合は, “Library” ウィンドウからRecompileして “Restart”

f. 入出力ピンの割当て (1/3)

- 回路の信号線をFPGAの入出力ピンに割り当てる
 - 割当ての指定に応じて、ボード上のスイッチから入力を与えたり、LEDに信号を出力したりできるようになる
 - どのモジュールがどの入出力ピンに対応しているかはユーザーズマニュアルとピンアサイン表を参照すること
 - ✓ ピンアサイン表まとめが非常に便利
 - https://panda.ecs.kyoto-u.ac.jp/access/content/group/2024-110-9084-000/HWマニュアル/ピンアサイン表_rev.xlsx
- FPGAの入出力ピンと各種装置との対応は、取扱説明書およびピンアサイン表を精読する
 - 今回の例では、2つの入力値をディップSWのA,B, キャリーインをテンキーのプッシュSW SW4, 和とキャリーアウトをLEDとしている
 - 上記ピン割当ては使い勝手に応じて自由に変更してみましょう

f. 入出力ピンの割当て (2/3)

- Pin Plannerを起動する
 - “Assignments” -> “Pin Planner” または Ctrl+Shift+N



f. 入出力ピンの割当て (3/3)

- 入出力ピンを割り当てていく
 - Node NameとLocationを対応づける
 - ピンの名前が出てこない場合は設計を再コンパイルする
 - 別のピン名が出る場合は、Top-levelになっているか確認する

The screenshot shows the Pin Planner interface for a Cyclone IV E - EP4CE30F2317. The top view displays the FPGA pin grid with various pins highlighted. A table at the bottom lists the pin assignments for various nodes.

| Node Name | Direction | Location | I/O Bank | VREF Group | Filter Location | I/O Standard | Reserved | Current Strength | Slew Rate | Differen |
|-----------|-----------|----------|----------|------------|-----------------|----------------|----------|------------------|-------------|----------|
| a[3] | Input | PIN_D10 | 8 | B8_N0 | PIN_C8 | 2.5 V ...fault | | 8mA (default) | | |
| a[2] | Input | PIN_C10 | 8 | B8_N0 | PIN_H11 | 2.5 V ...fault | | 8mA (default) | | |
| a[1] | Input | PIN_B10 | 8 | B8_N0 | PIN_B6 | 2.5 V ...fault | | 8mA (default) | | |
| a[0] | Input | PIN_A10 | 8 | B8_N0 | PIN_B5 | 2.5 V ...fault | | 8mA (default) | | |
| b[3] | Input | PIN_B13 | 7 | B7_N3 | PIN_A5 | 2.5 V ...fault | | 8mA (default) | | |
| b[2] | Input | PIN_A13 | 7 | B7_N3 | PIN_C7 | 2.5 V ...fault | | 8mA (default) | | |
| b[1] | Input | PIN_F11 | 7 | B7_N3 | PIN_A4 | 2.5 V ...fault | | 8mA (default) | | |
| b[0] | Input | PIN_E11 | 7 | B7_N3 | PIN_E9 | 2.5 V ...fault | | 8mA (default) | | |
| cin | Input | PIN_E15 | 7 | B7_N1 | PIN_G10 | 2.5 V ...fault | | 8mA (default) | | |
| cout | Output | PIN_A8 | 8 | B8_N0 | PIN_F10 | 2.5 V ...fault | | 8mA (default) | 2 (default) | |
| sum[3] | Output | PIN_A9 | 8 | B8_N0 | PIN_H10 | 2.5 V ...fault | | 8mA (default) | 2 (default) | |
| sum[2] | Output | PIN_F8 | 8 | B8_N3 | PIN_C6 | 2.5 V ...fault | | 8mA (default) | 2 (default) | |
| sum[1] | Output | PIN_C8 | 8 | B8_N1 | PIN_B8 | 2.5 V ...fault | | 8mA (default) | 2 (default) | |
| sum[0] | Output | PIN_B8 | | | | | | 8mA (default) | 2 (default) | |

デバイス名を確認
違う場合は設定を見直す

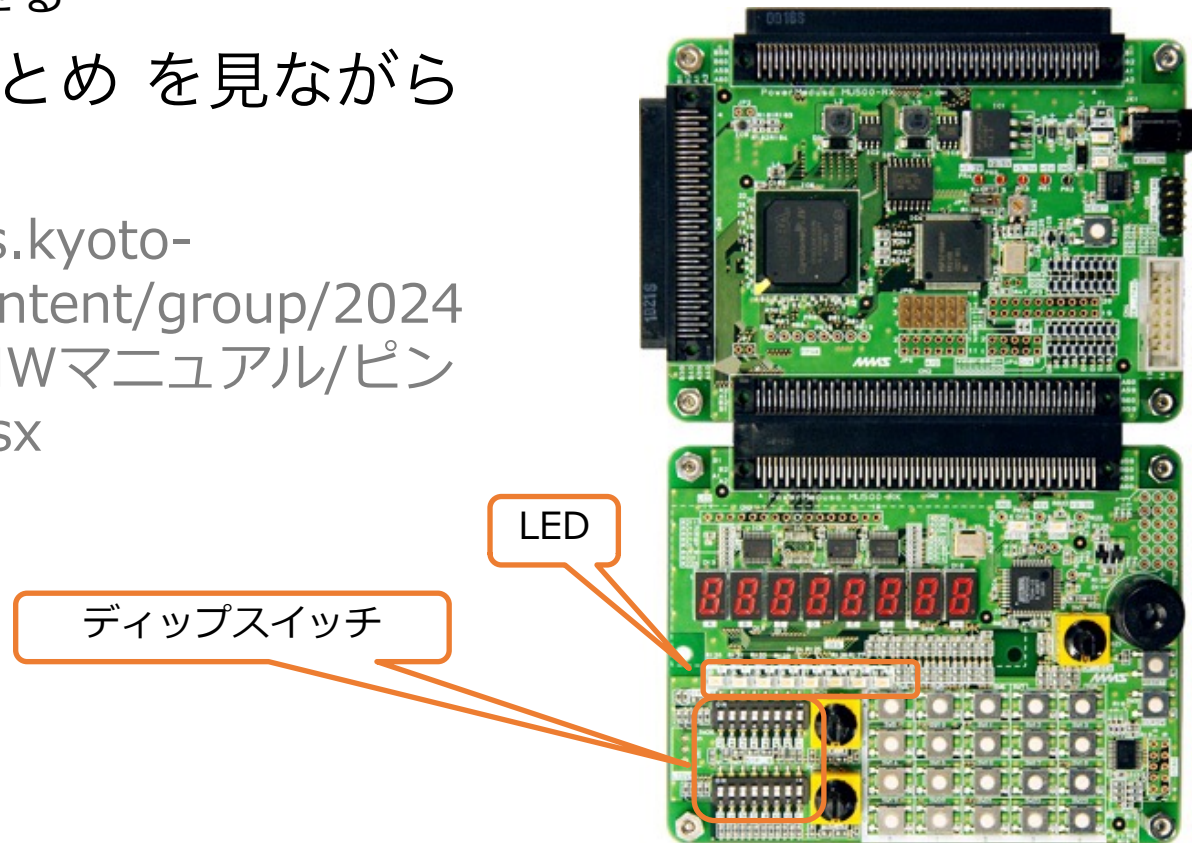
Location列でFPGAのピンを指定する
("PIN_"を省略した)直接入力も可能
他の列は自動入力される

この例ではどこに何が
割り当てられている??

ピン名と入出力方向
一覧がおかしい場合は
コンパイル状況や
Top-levelを確認する

f. 入出力ピンの割当て (3/3)

- 今回の割当て例
 - ディップスイッチで入力を与える
 - 結果をLEDで表示させる
- ピンアサイン表まとめ を見ながらやってみよう
 - https://panda.ecs.kyoto-u.ac.jp/access/content/group/2024-110-9084-000/HWマニュアル/ピンアサイン表_rev.xlsx



g. FPGAへの回路の書き込み

- 合成した回路イメージをFPGAに書き込み，実機上で動作を検証する
 - 実機で動作して初めて設計の完了！
 - Tasks窓が黄字？の場合は再コンパイル，赤字×の場合はエラーなのでHDLコード等をデバッグする

g. FPGAへの回路の書き込み

- FPGA書き込みツールを起動する
 - “Tools” -> “Programmer” または
Tasks窓の“Program Device (Open Programmer)”

The screenshot displays the Quartus II 32-bit software interface. The 'Tools' menu is open, showing various options. The 'Programmer' option is highlighted with a blue box. In the 'Tasks' window, the 'Program Device (Open Programmer)' task is also highlighted with a blue box. The 'Compilation Report - adder4bit' window is visible on the right, showing a successful compilation status.

| Flow Summary | |
|------------------------------------|---|
| Flow Status | Successful - Wed Apr 12 17:51:01 2017 |
| Quartus II 32-bit Version | 13.0.1 Build 232 06/12/2013 SP 1 SJ Full Vers |
| Revision Name | adder4bit |
| Top-level Entity Name | adder4 |
| Family | Cyclone IV E |
| Device | EP4CE30F2317 |
| Timing Models | Final |
| Total logic elements | 6 / 28,848 (< 1 %) |
| Total combinational functions | 6 / 28,848 (< 1 %) |
| Dedicated logic registers | 0 / 28,848 (0 %) |
| Total registers | 0 |
| Total pins | 14 / 329 (4 %) |
| Total virtual pins | 0 |
| Total memory bits | 0 / 608,256 (0 %) |
| Embedded Multiplier 9-bit elements | 0 / 132 (0 %) |
| Total PLLs | 0 / 4 (0 %) |

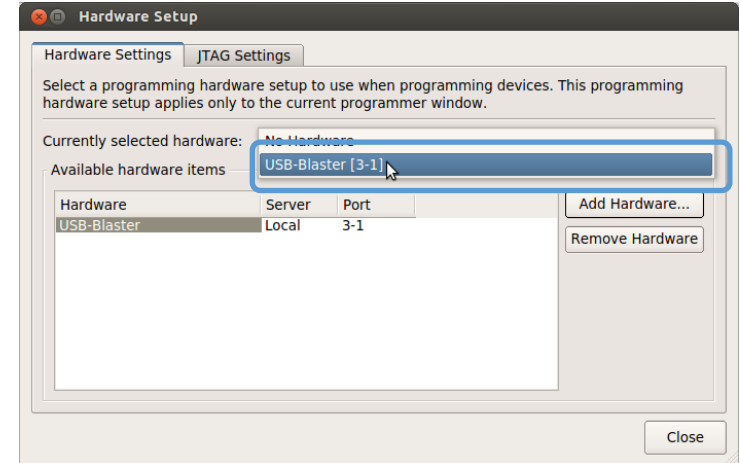
g. FPGAへの回路の書き込み

- FPGAボードをPCに接続
 - 電源に接続
 - USBを接続

g. FPGAへの回路の書き込み

- Programmerの設定を行う
 - Hardware Setupをクリックして、Currently selected hardware: を “USB-Blaster [X-X]” とする
 - ✓ Available hardware itemsにUSB-Blasterが出てこない時は、PCとUSB接続し直す。それでも出てこない場合はデバドラの問題も考えられるのでTAまで。
 - Modeを“JTAG”にする
 - リストにファイルが表示されていない場合は、Add File... から書き込むファイルを選択する
 - ✓ ./output_files/
<proj_name>.sof
 - リスト上のProgram/Configureにチェックを入れる

数字は環境によって異なる



g. FPGAへの回路の書き込み

- Programmerの設定を行う

USB-Blaster を
選択

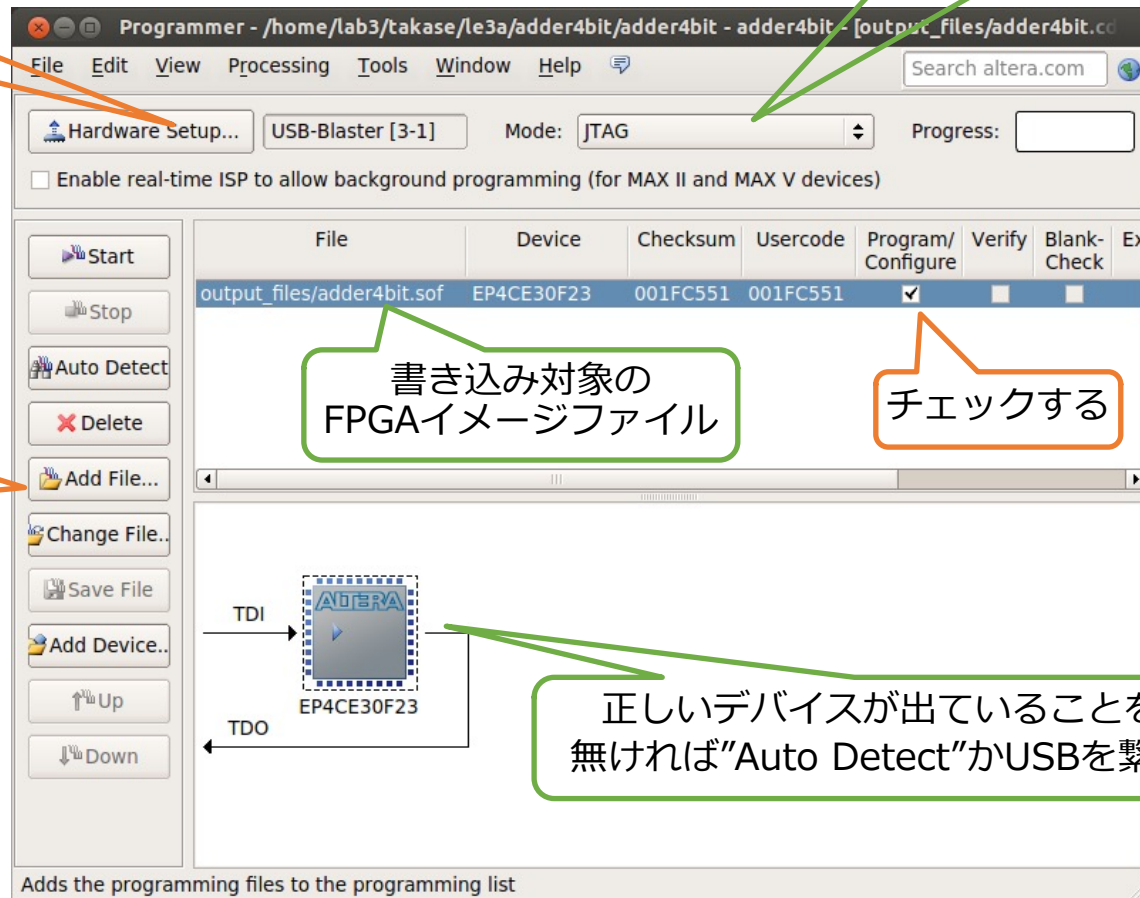
JTAGになっている
ことを確認

.sofを追加する

書き込み対象の
FPGAイメージファイル

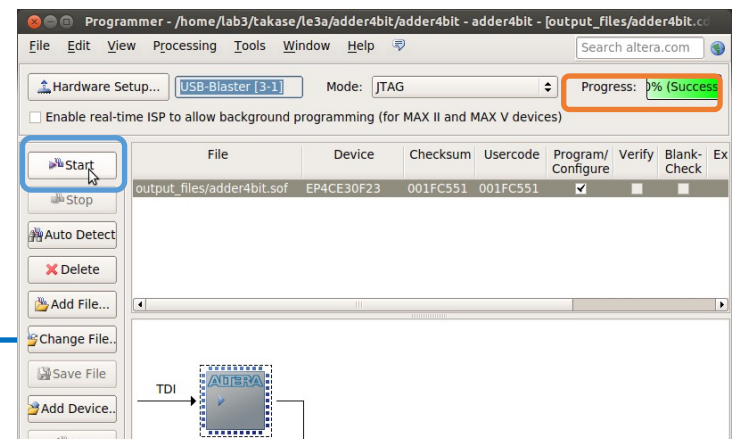
チェックする

正しいデバイスが出ていることを確認
無ければ"Auto Detect"かUSBを繋ぎ直す



g. FPGAへの回路の書き込み

- FPGAに回路を書き込む
 - Startをクリック
 - Process: が緑の"100% (Success)"になったら書き込み完了
 - FPGAボードが認識されているのに、書き込みがうまくいかない(Failedになる)場合、以下のコマンドを実行 (jtagdを再起動) してもう一度試す
 - ✓ `killall -9 jtagd`
 - ✓ `/opt/intelFPGA/20.1/quartus/bin/jtagconfig`
 - エラーが出ないことを確認
 - ✓ 詳細は https://isle3hw.kuis.kyoto-u.ac.jp/faq.html#programmer_failed を参照
- 実機上で動作を検証する
 - 正しく所望の動作ができていれば完了！
 - **ディップスイッチは負論理なので注意**
 - ✓ ONで0になる



TIPS

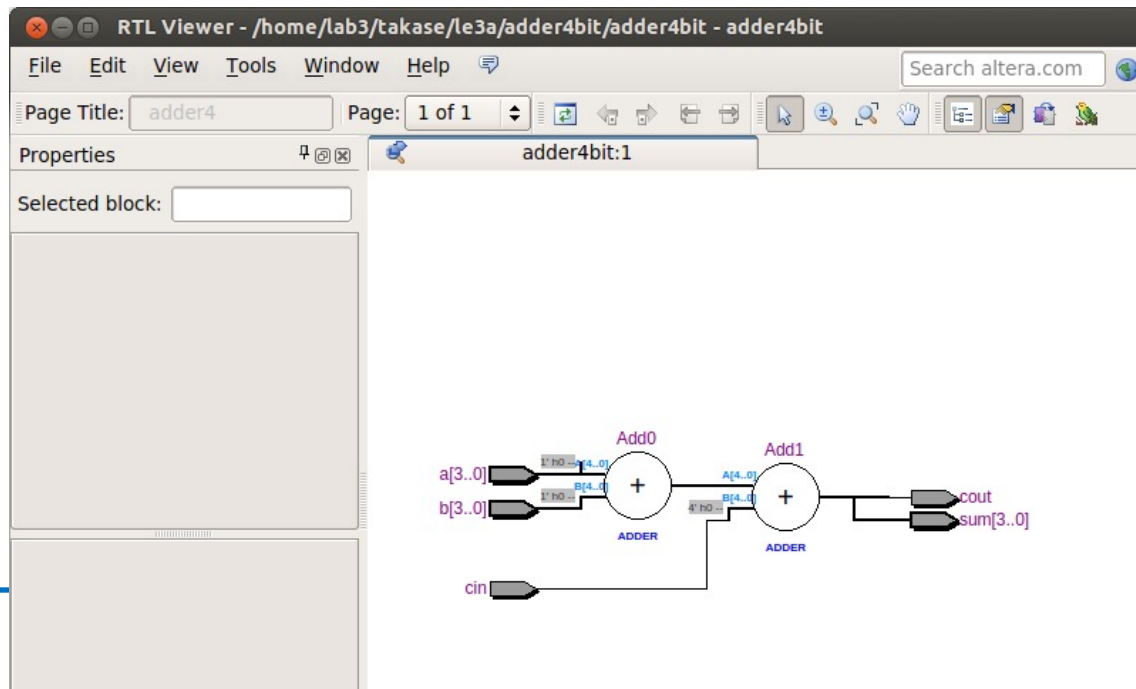
- うまくいかない場合はまずFAQを確認。
 - 過去にあった様々な質問とそれに対する回答がまとめられています。
- 実機で思った通りに動かない
 - まずはシミュレーションで論理的に正しく設計できているか、きちんと確認しましょう
 - シミュレーションが上手くいっていても実機で動かないこともよくあります。多くはタイミング制約を満足できていないことに起因します。FAQを参考にしてください。
- 入出力装置が思った通りに動かない
 - もちろん設計が正しいかは当然として、入出力装置にはそれぞれ正論理と負論理があります
 - 7SEG LEDはセレクトタ信号の設定も必要です
 - 未割り当てのLED(ピン)が光るのは無視して構いません
 - ✓ きちんと考慮するのが綺麗ですが, , ,
 - 詳しくは要するに取扱説明書を精読してください

TIPS

- 実機上で色々な情報を一度に出力したい
 - 拡張ボードMU500-7SEGを使いましょう
 - ✓ 使い方の詳細は取扱説明書を参照してください
 - 拡張ボードを使いこなせば、プロセッサのPC現在値だったりを実機上で確認できるようになってデバッグが非常に捗ります
- HDLの回路を回路図エディタから呼び出したい
 - 対象ファイルを選択した上で “Files” -> “Create / Update” -> “Create Symbol Files for Current File” を選択する
 - ✓ 回路図エディタから部品インスタンスとして配置できる
 - ✓ 逆もできます(回路図をHDLから下位モジュールとして呼び出し)
 - プロセッサ設計では、全体のアーキテクチャとデータパスは回路図エディタで設計し、各部品はHDLで設計するのも良いでしょう
 - ✓ 全体の見通しがよくなります

TIPS

- HDLからどんな回路が合成されるか知りたい
 - “Tools” -> “Netlist Viewers” -> “RTL Viewer” で合成回路をなんとなく可視化できます
 - 自分のHDLがどのような回路に合成されるのか, HW設計とはなんたるかを知るのに有用な機能です
 - ✓ 今回の例題だとまったく面白くないですが, , ,



TIPS

- シミュレーション
 - modelsim-aseの Transcript サブウィンドウに直接コマンドを入力して実行できる。慣れてきたらメニューをクリックするより早い。コマンドヒストリも使える。
 - 実行履歴は “./simulation/modelsim/msim_transcript” に残るので、編集して別名で保存しておくとうい。Transcriptから do (ファイル名) で実行できる。
- モジュールについて
 - 回路全体ではなく一部だけを合成したりシミュレーションしたい場合は、Project Navigator の該当ファイルを右クリックして “Set as Top-Level Entity” を設定するとよい
- バージョン Quartus Prime 20.1 / ModelSim ASE 10.5b
 - Google先生に聞く時はツールのバージョンに注意すること。古い／新しいために通用しないTIPSや解決策がある。