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

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
  - 加算器設計で詰まったら，積極的にTAを捕まえましょう
  - 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
- 春休みで全て忘れてしまった方は，，，
  - http://www.lab3.kuis.kyoto-u.ac.jp/~takase/le3a/le2hw3-2019.pdf

2023年度の注意点

- 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
  - 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

分かっている気を付けどころを
吹き出しで記載します

2023年度の注意点

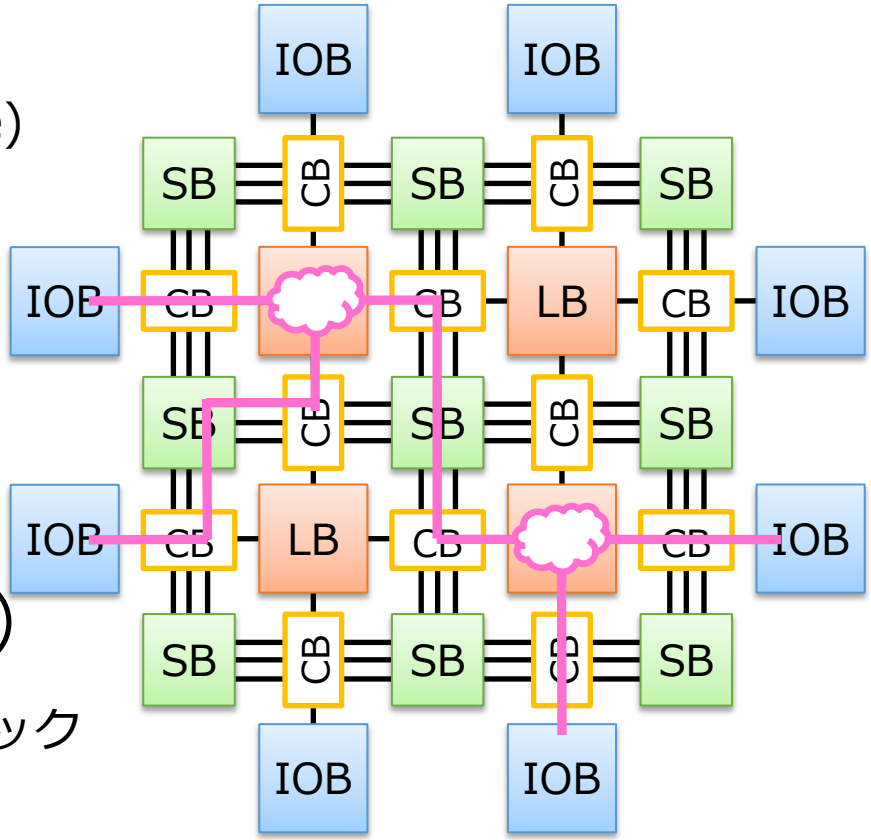
- 今年度から計算機室のPCが新しくなりました。昨年度までのデータは旧ホームディレクトリにありますので、必要なものは新ホームディレクトリに移動させてください。

自分の入学年度

  - 新ホームディレクトリ: /export/home/021/<ECS-ID>
  - 旧ホームディレクトリ: /old_export/home/021/<ECS-ID>
- 旧ホームディレクトリは1年後に完全に削除します。

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：
    Xilinx(AMD)，Altera(Intel)



LB 論理ブロック　SB スイッチブロック

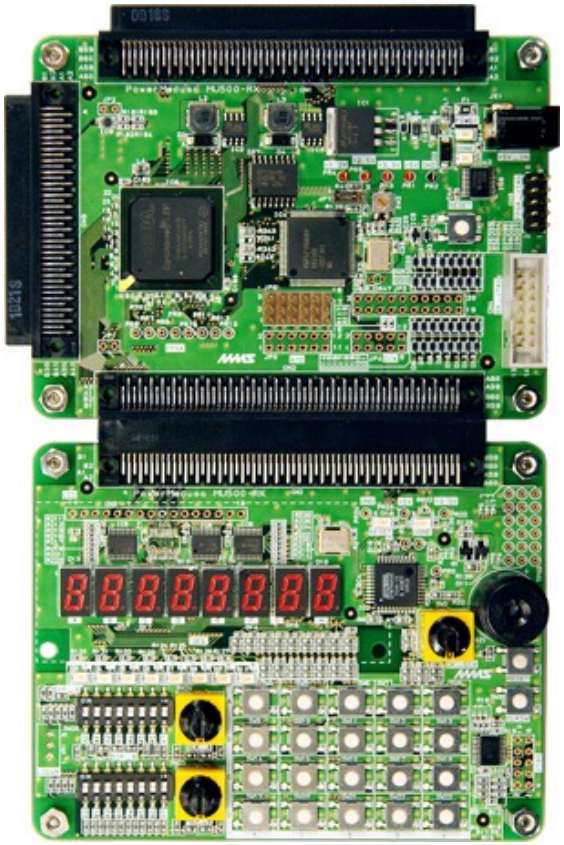
IOB 入出力ブロック　CB コネクションブロック

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)

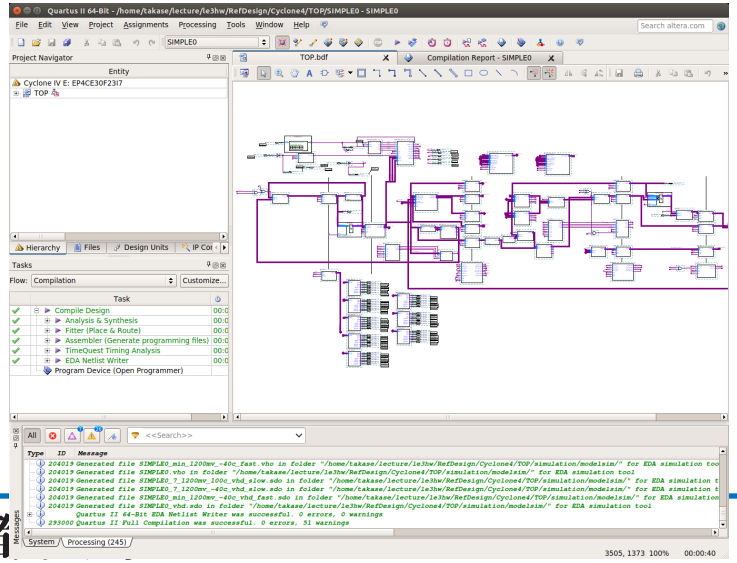
実験環境：FPGAボード

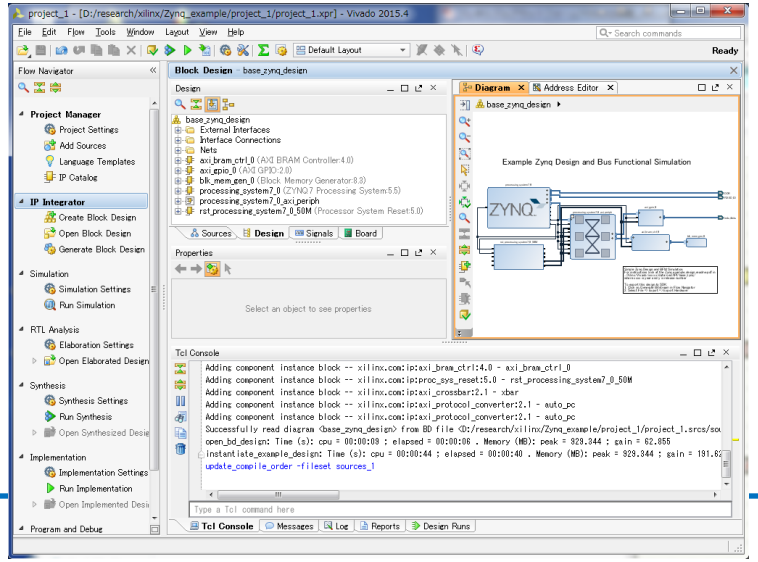
- PowerMedusa MU500-RX/RK
  - FPGA：Altera社Cyclone IV
    EP4CE30F23I7N (28,840LE)
  - マイコン：ルネサス社RX210
  - クロック：20MHz発振器
  - 入出力ユーザインタフェース
    - ✓プッシュSW x20, ロータリSW x2, 8bit DIP SW x2, クロックSW x1
    - ✓7SEG LED x8, LED x8, ブザー x1
  - 拡張ボード MU500-7SEG
    - ✓7SEG LED x64, LED x64



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

- 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？の警告が出ることがあるが，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

      計 3.2GB (Linux版 3.7GB)
  - (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

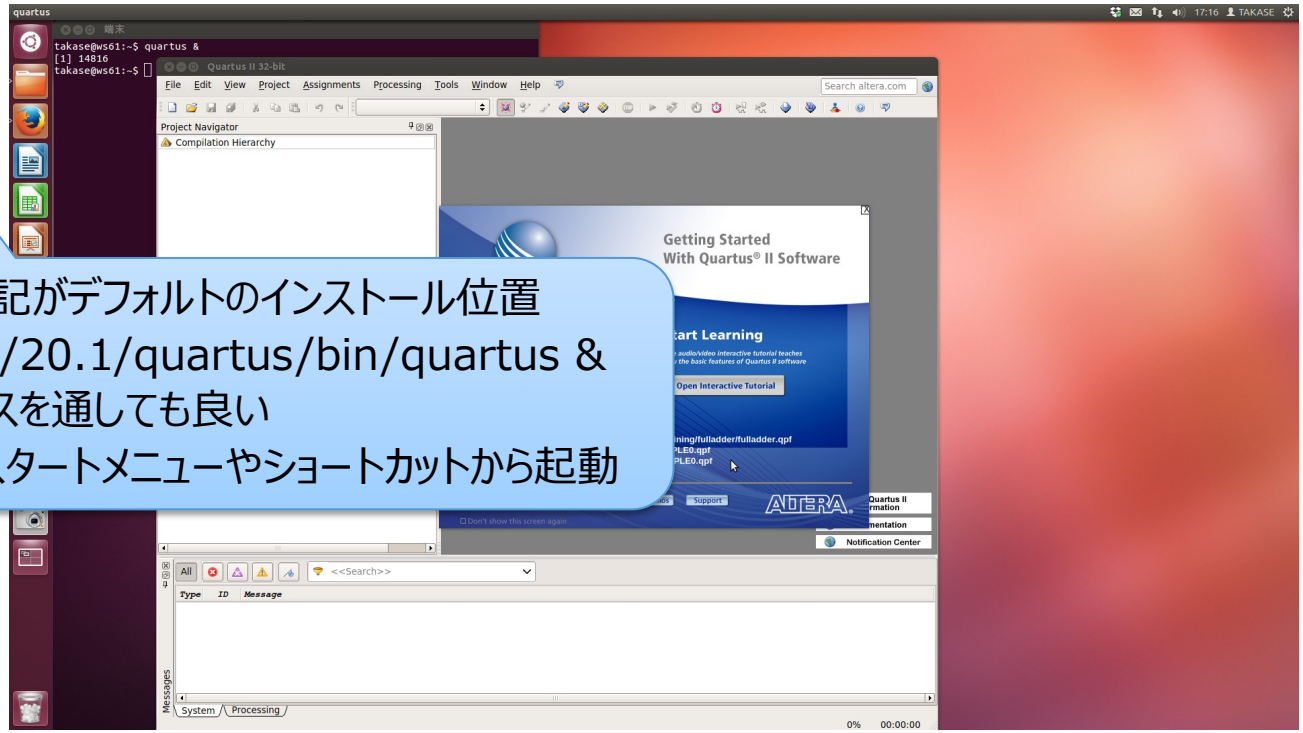
Windowsの場合は，Quartus本体のインストール終了後に表示される
USB Blaster II device driverもインストールしてください

- ✓ 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 &



Lite Editionは下記がデフォルトのインストール位置
$ ~/intelFPGA_lite/20.1/quartus/bin/quartus &
パスを通しても良い
Windowsの場合は、スタートメニューやショートカットから起動

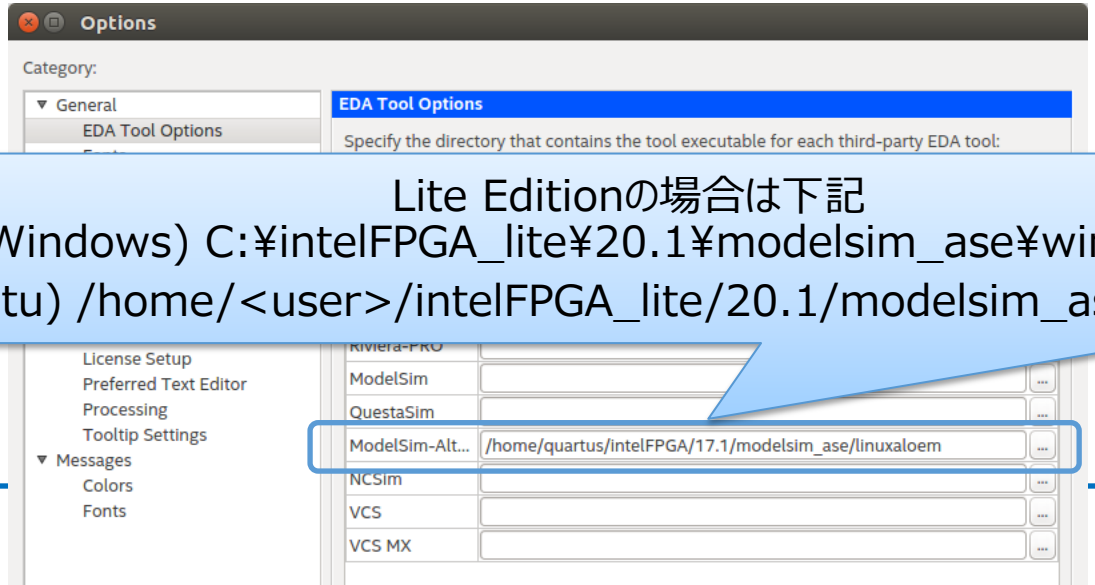
設定の確認

Lite Edition 20.1はライセンス設定は不要
初回起動時になにか聞かれたら
‘Run the Quartus Prime software”を選択

- 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



Lite Editionの場合は下記
(Windows) C:¥intelFPGA_lite¥20.1¥modelsim_ase¥win32aloem
(Ubuntu) /home/<user>/intelFPGA_lite/20.1/modelsim_ase/linuxaloem

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. 論理回路の設計
c. HDLコードの作成
d. コンパイル
e. タイミング制約の設定と検証
f. シミュレーションによる動作確認
g. トップレベル回路への入出力ピンの割当て
h. FPGAへの回路の書き込み

練習は組合せ回路なので無し

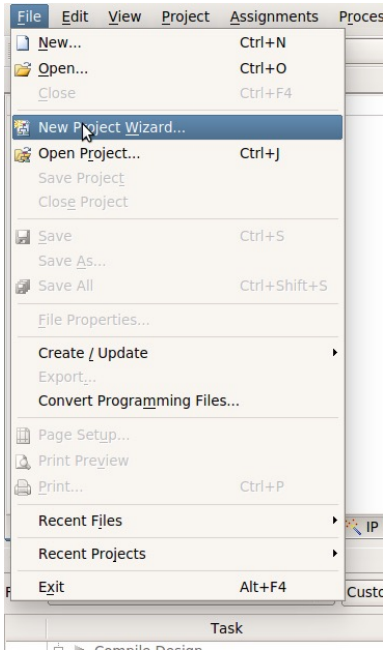
4ビット加算器をお題として設計フローを説明します

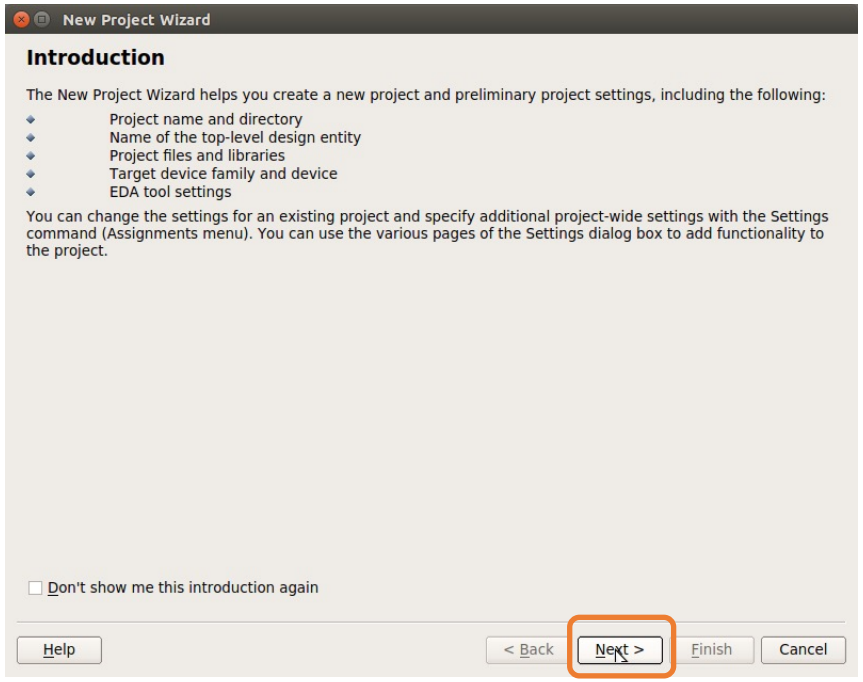
練習：4bit加算機の設計

a. 新規プロジェクトの作成と設定

- File -> New Project Wizard...

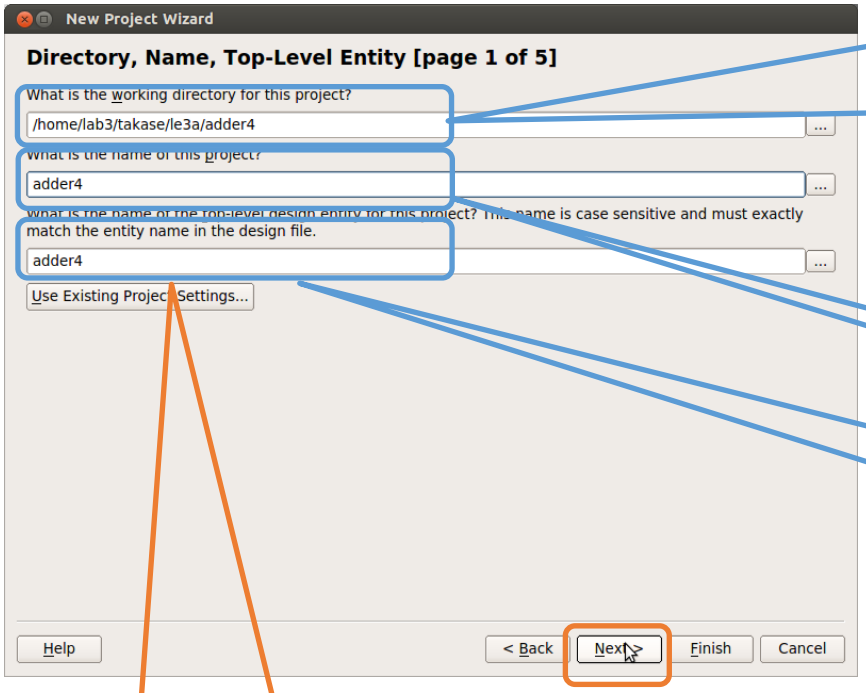
- Introduction “Next”





a. 新規プロジェクトの作成と設定

- Directory, Name, Top-Level Entity

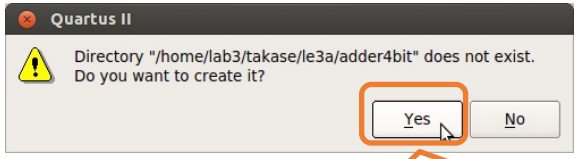


作業ディレクトリ：
ホームディレクトリそのものを指定せず，必ず専用のディレクトリを作ること．そうしないとツールが固まったり落ちたりする原因になる．
また，プロジェクト毎にディレクトリを作ったほうがよい（名前は自由）

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。

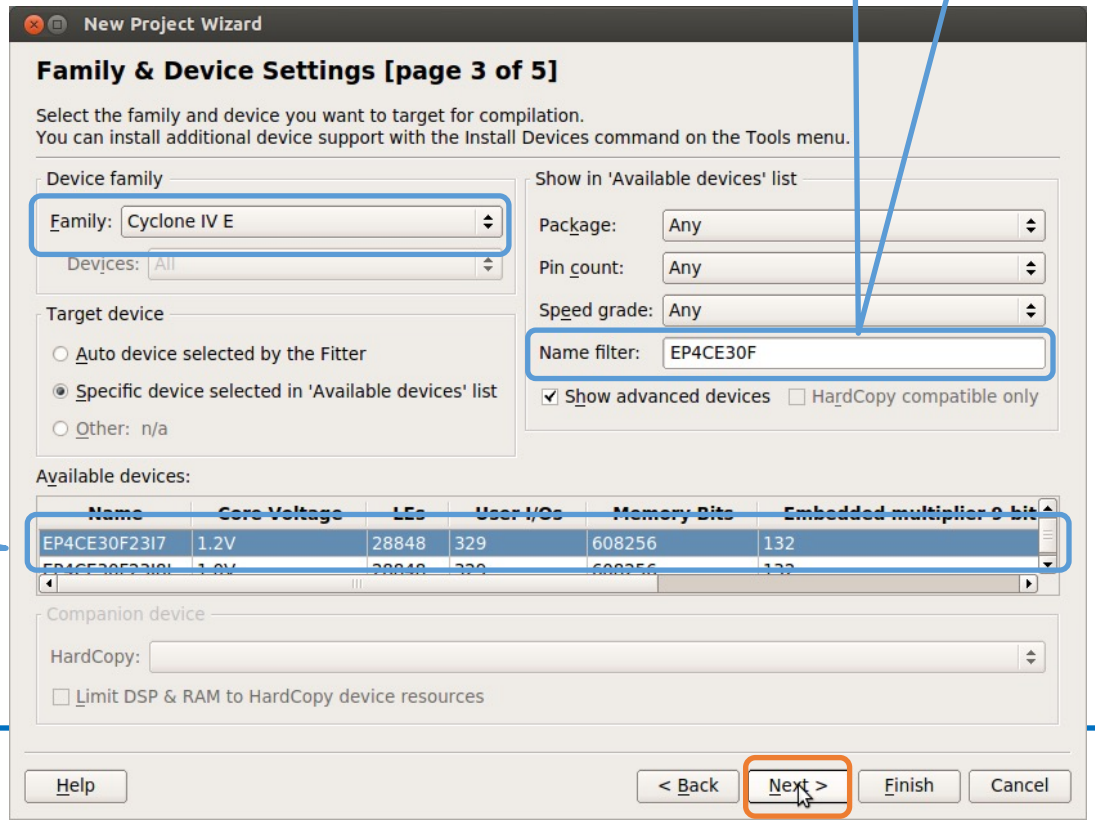


作業ディレクトリが存在しない
場合は自動作成してくれる

a. 新規プロジェクトの作成と設定

- 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”

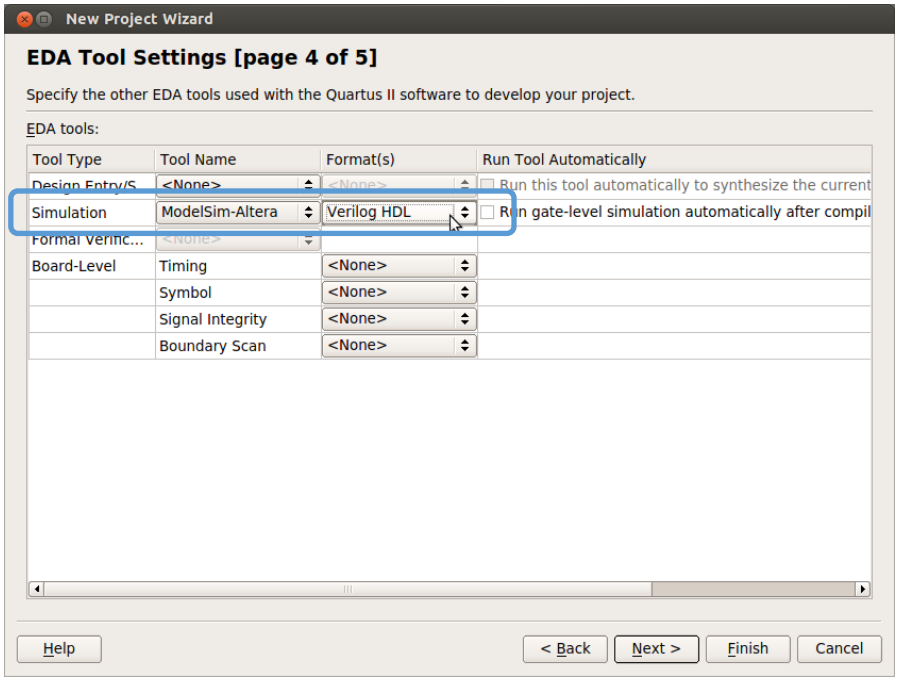
選択が面倒な場合は
フィルタ検索も可能

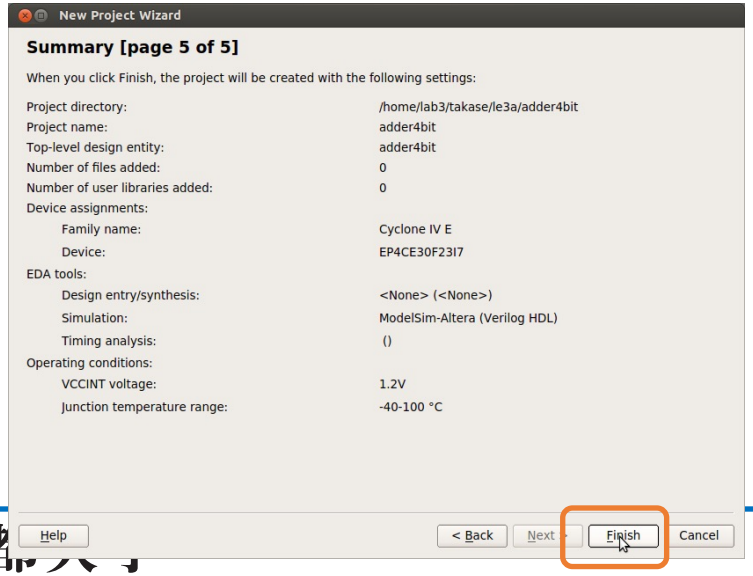


Available devices:から
EP4CE30F23I7を選択

a. 新規プロジェクトの作成と設定

- 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”





b. 論理回路の設計

- 設計したい論理回路を考える
  - 外部仕様：入力と出力，機能
  - 内部仕様：回路構成
    - ✓ 真理値表，カルノー図，状態遷移図，，，　→ 論理式表現
      → HDL設計ではここまで細かくやる必要はないが，
      どんなHDLコードを書きたいか深く考えること

まずは設計を考えてからツールを使い出すこと！

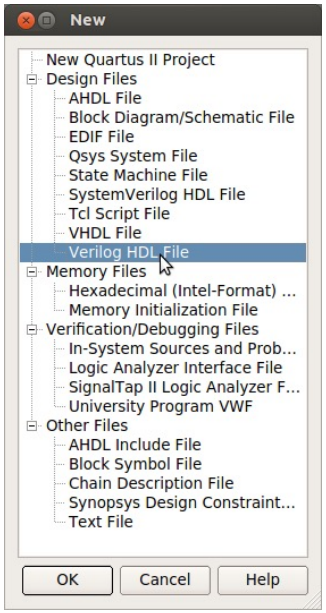
- 4ビット加算器の外部仕様
  - 入力：4ビットの加算データ，1ビットのキャリーイン
  - 出力：4ビットの和，1ビットのキャリーアウト

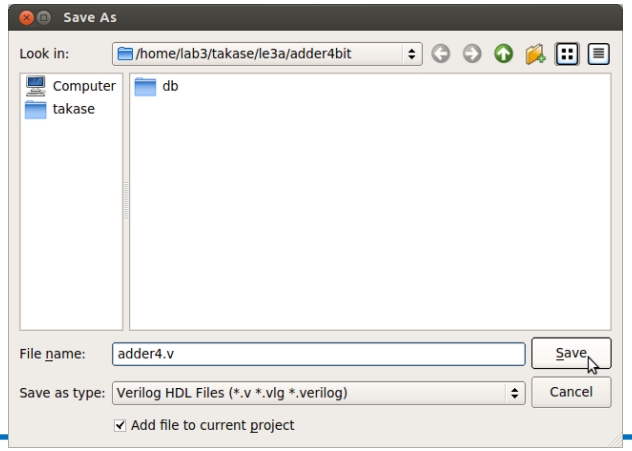
c. HDLコードの作成

本実験はVerilog HDL前提
VHDL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

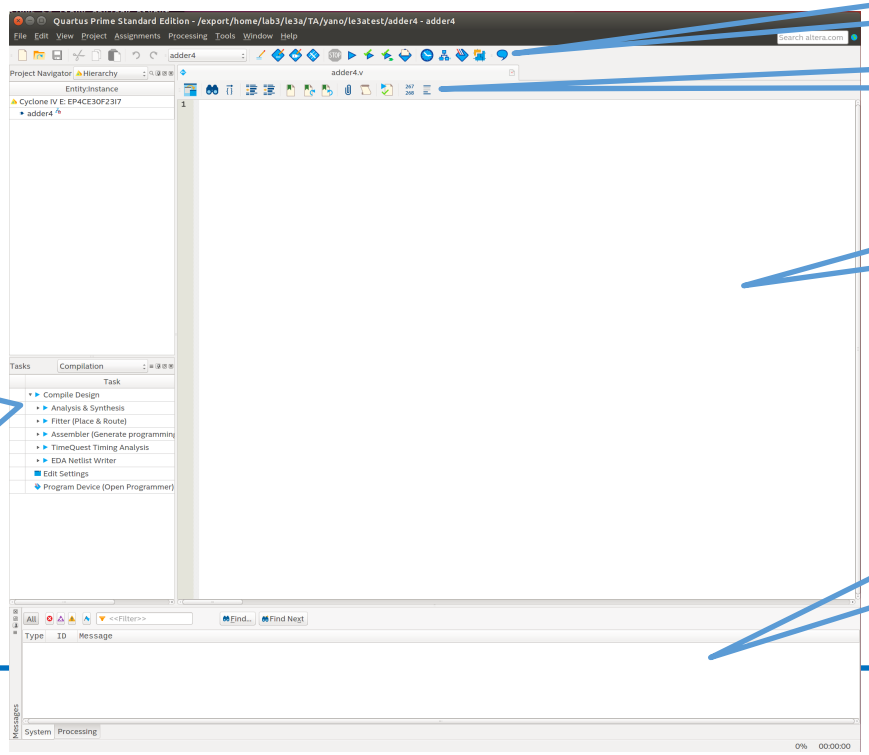
数字から始まる
ファイル名は避ける





c. HDLコードの作成

- (初期)ウインドウの構成
  - 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)



ツール操作アイコン

エディタ操作アイコン

ファイルの一覧や
呼び出し関係

エディタ画面

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

メッセージウインドウ
処理メッセージや
エラー報告など

c. HDLコードの作成

- 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

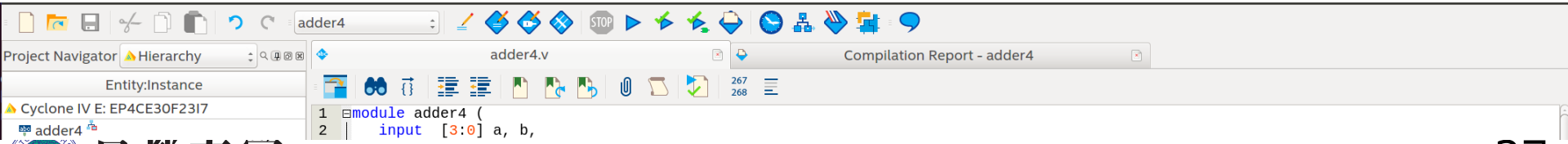
adder4.vhd

```
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
```

d. コンパイル

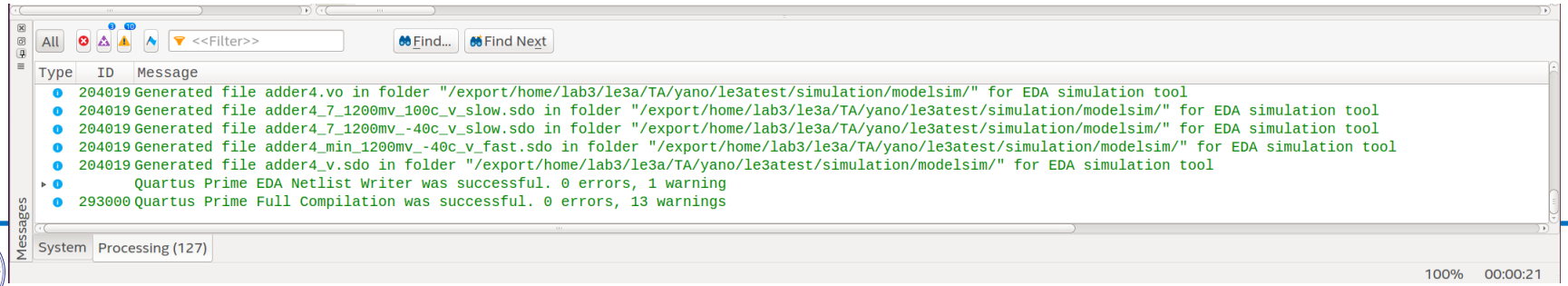
- FPGA内の論理ブロックの構成情報に変換する（Cのコンパイルに相当）
  - 回路図をレジスタ転送レベルに変換する
  - 論理合成，テクノロジマッピング，配置配線を行う
    - ✓ RTLで記述された回路をFPGA上の論理ブロックに割り付け，論理ブロックの配置や配線を行う
  - ビットストリームを生成する
    - ✓ 配置／配線の情報をバイナリ記述の構成情報ファイルに変換する

- 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



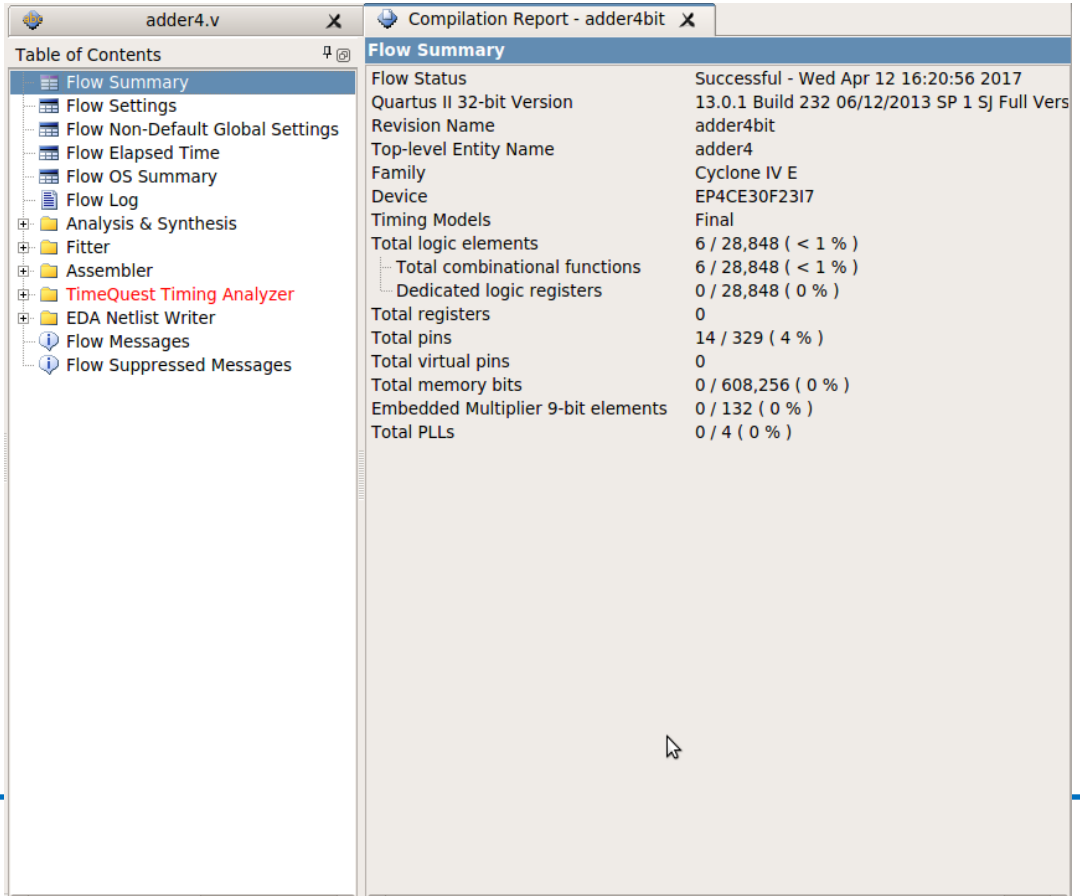
d. コンパイル

- 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アクセス可能
  - Critical Waning / 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，警告の意味を理解した上で無視すること（設計ミスとかに気づくことも多い）



d. コンパイル

- 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

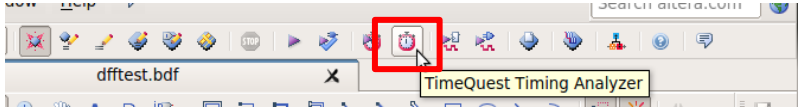


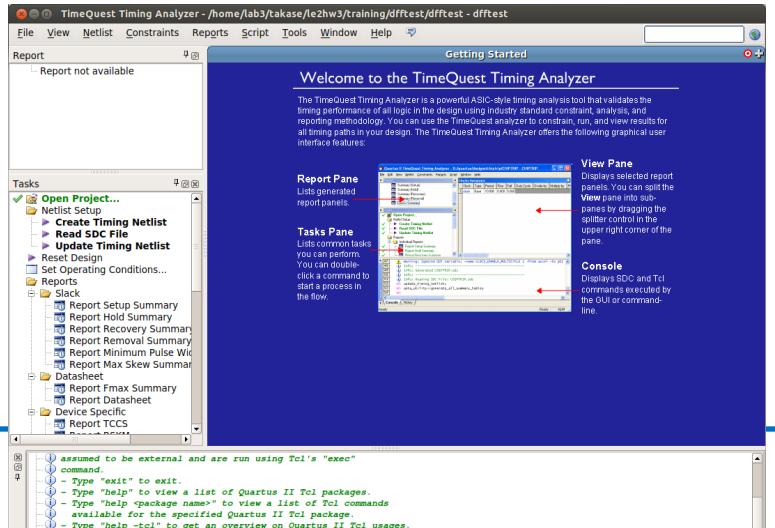
d. コンパイル

- 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：TimeQuest 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（今は組合せ回路なので詳細は後ほど）
  - 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

e. タイミング制約の設定(参考)

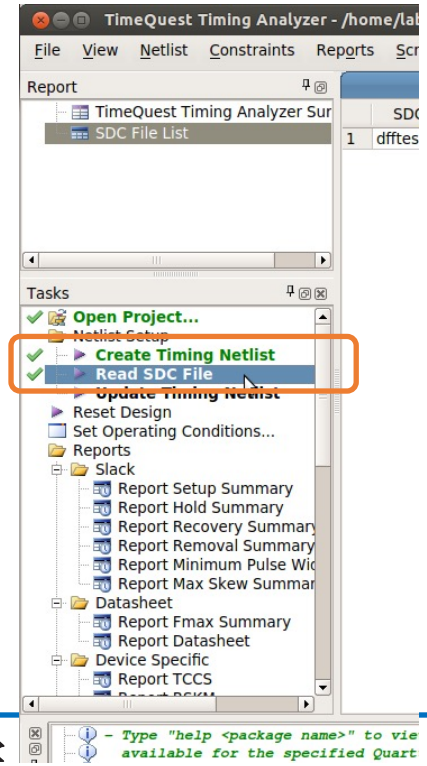
- 練習の加算器は組み合わせ回路なので設定の必要はないが、クロックを使用する順序回路では必ず設定する。
- クロックの指定やタイミング制約（レジスタ入力へのセットアップ時間やホールド時間，伝搬遅延など）の設定を行う
  - 基本的なクロックの指定のみ解説，詳細は各自で調査すること.
- TimeQuest Timing Analyzerを使用する
  - Tools -> TimeQuest Timing Analyzer または，
    ツール操作アイコンの “TimeQuest Timing Analyzer”

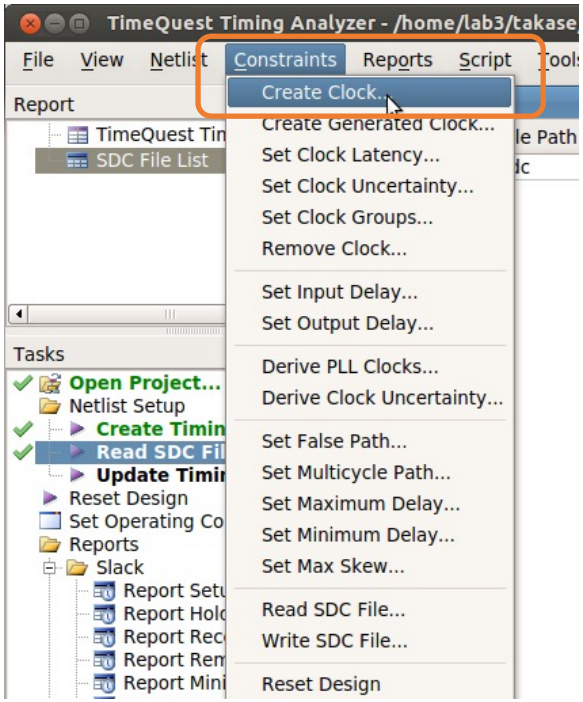




e. タイミング制約の設定(参考)

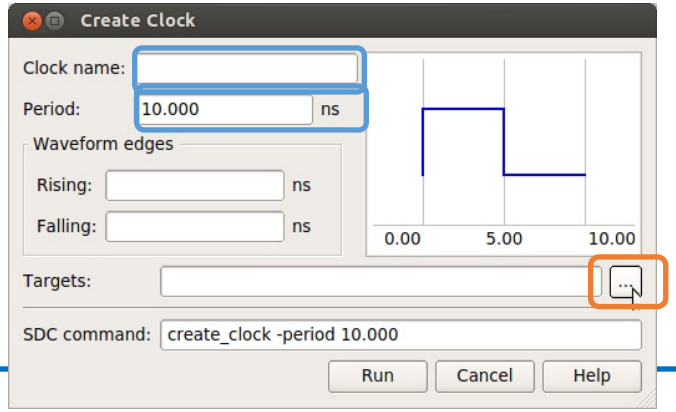
- TimeQuest Timing Analyzer
  - Tasks窓の “Netlist Setup -> Create Timing Netlist” を実行
  - Tasks窓の “Netlist Setup -> Read SDC File” を実行
  - Constraints -> Create Clock... からクロックを設定

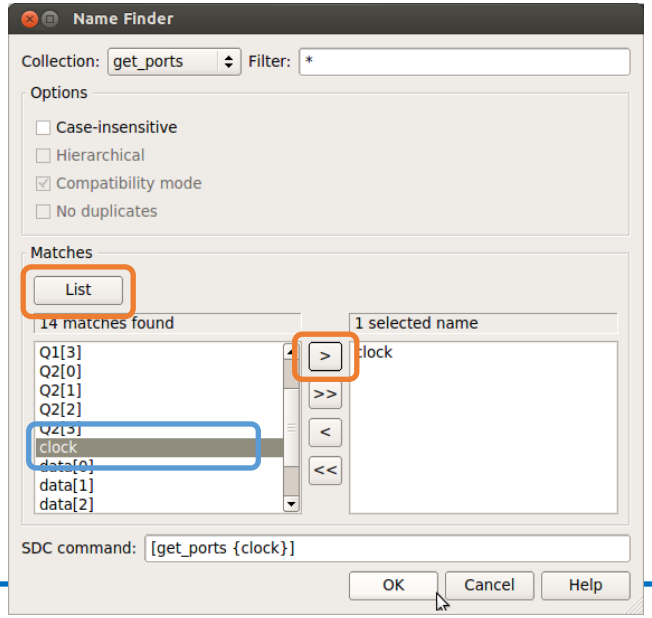




e. タイミング制約の設定(参考)

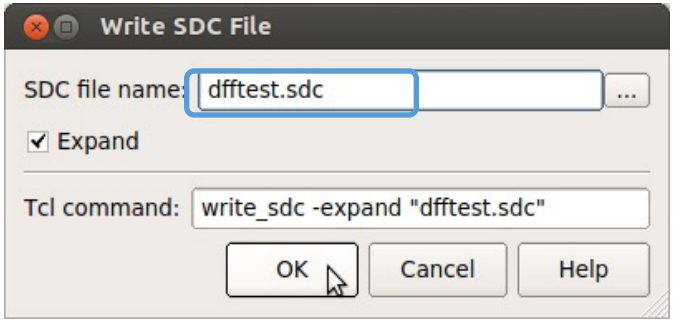
- TimeQuest Timing Analyzer
  - Constraints -> Create Clock... からクロックを設定
    - ✓ Clock name: クロック名の定義　未記入でも可（例：clock）
    - ✓ Period: クロック周期の設定
    - ✓ Waveform edges: 立ち上がり／立ち下がり時間の設定
    - ✓ Targets: 設計回路上のクロックを選択
      - ◘ “List”から表示して”>”で選択
    - ✓ SDC commandに生成される制約が表示される





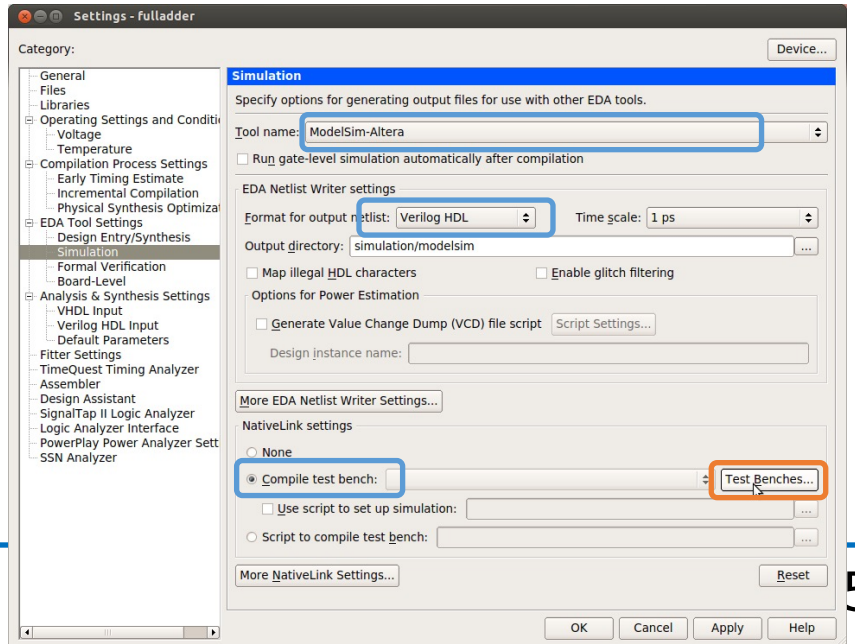
e. タイミング制約の設定(参考)

- TimeQuest 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



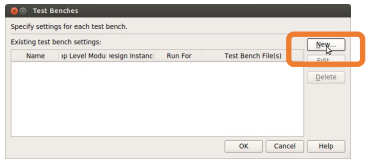
f. シミュレーション

- テストベンチファイルを作成する
  - Processing -> Start -> Start Test Bench Template Writer
  - ./simulation/modelsim/<module>.vt が作成される
  - ./simulation/modelsim/<module>_test1.vt などにコピー
- 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

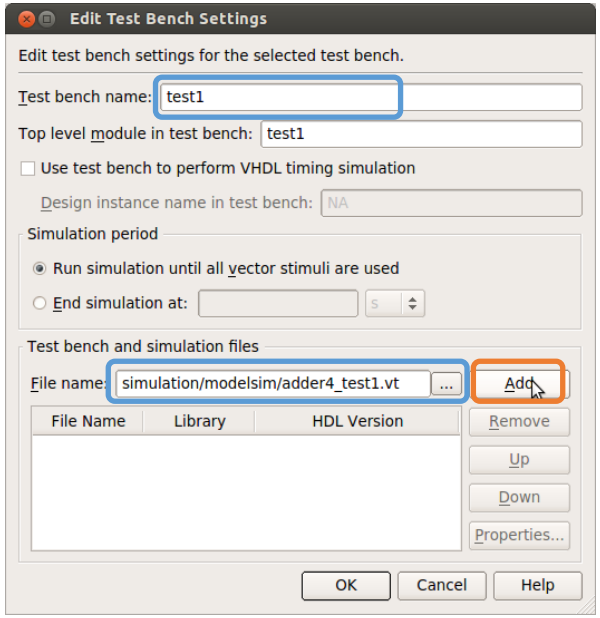


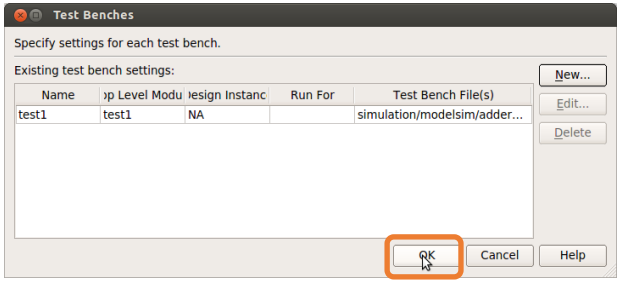
f. シミュレーション

- 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 (Template Writer を使うなら <module>_vlg_tst) にしておくと，ModelSim起動の一発目で Error loading design が出なくなる
  - ✓ Test bench and simulation files: simulation/modelsim/<module>_test1.vt
    (...から前ページで作成したファイルを選択)
    (VHDLの場合は .vht)
  - ✓ Add をクリック
- Test Benchesに戻ってOKをクリック





f. シミュレーション

- テストベンチファイルの編集
  - 所望の入力波形が得られるように <module>_test1.vt を編集
    - ✓ 時間単位を変更する（デフォルトは ps なので速すぎる）
      `timescale 1 ns / 1ps
    - ✓ 以下の記述の直下に書いたものは1回だけ実行される
      // code that executes only once
      // insert code here --> begin
    - ✓ 以下の記述の直下に書いたものは繰り返し実行される
      // code that executes for every event on sensitivity list
      // insert code here --> begin
    - ✓ デフォルトで記述されている”@eachvec;”はコメントアウトする
      （これがあるとシミュレーションがそこで止まってしまう）

慣れてきたらわざわざ
テンプレート作らなくてもよい

f. シミュレーション

- テストベンチファイルの記法例
  - 代入：

```
// 1ビット
A <= 0;
B <= B + 1;
// ビット演算
Cin <= ~Cin;
// 4ビットの2進数
As <= 4'b10_01;
// 4ビットの2進数
As <= 4'b10_01;
```

    - ✓ (遅延が無ければ)並列に代入される

Verilog HDLと記法は一緒
VHDLは説明省略

  - 遅延：

```
#1000
```

    - ✓ 単位は前ページで設定した単位(ns)
  - 繰り返し：

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

    - ✓ クロック波形の生成に便利
    - ✓ 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
    - ✓ @()で条件の記述も可能

f. シミュレーション

- 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？？

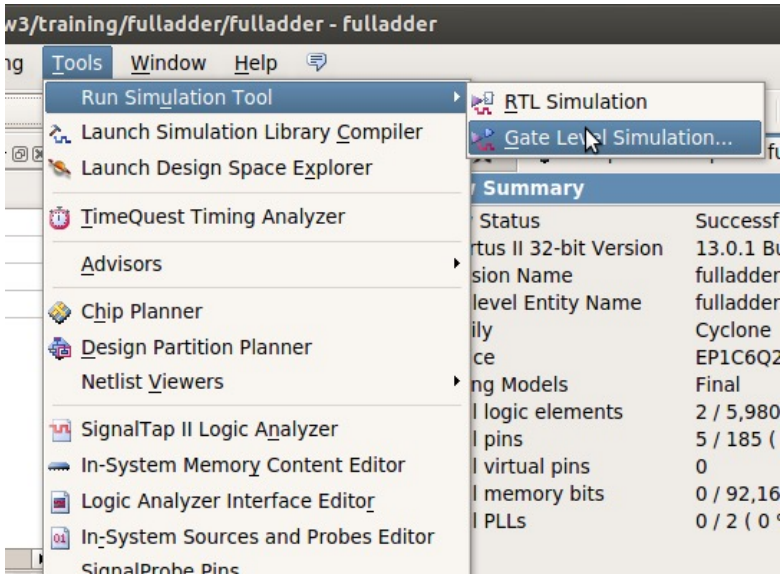
```
);
initial
begin
// code that executes only once
// insert code here --> begin
a <= 4'b0000;
b <= 4'b0000;
cin <= 0;

// --> 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

f. シミュレーション

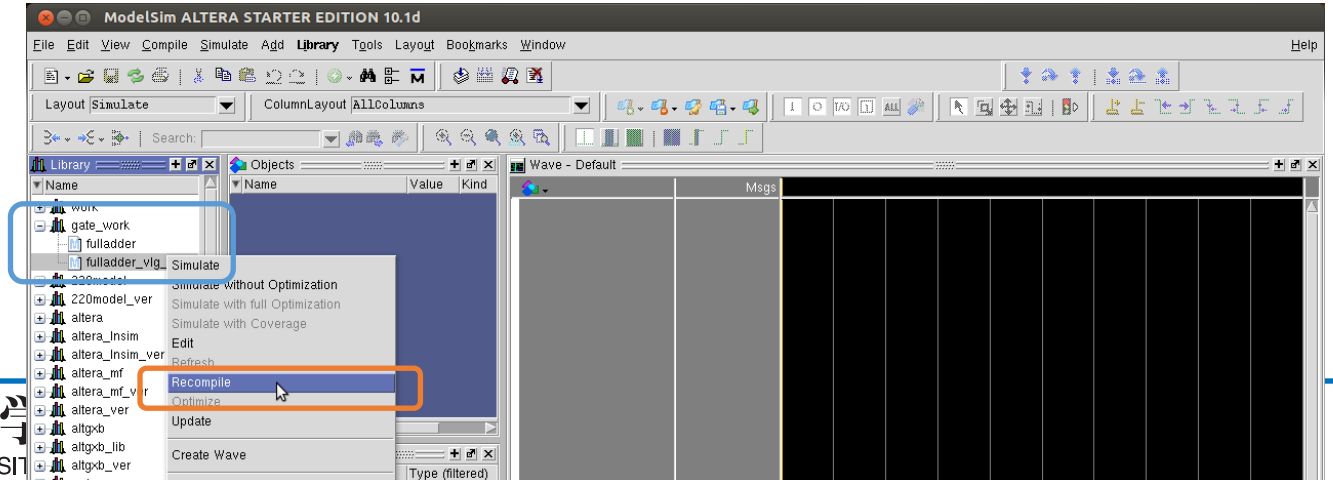
- 設計を再コンパイルする（タスク窓が？になっているはず）

- シミュレータを起動する
  - Tools -> Run Simulation Tool -> Gate Level Simulation...



f. シミュレーション

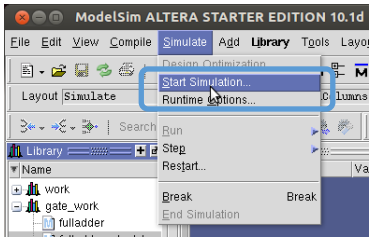
- シミュレータを起動する
  - 設計した回路は Library の gate_work 以下にある
    - ✓ work 内は gate_work へのリンク
- テストベンチファイルを念のため再コンパイル
  - “Library” ウインドウの gate_work -> <module>_vlg_tst を右クリックして “Recompile”
    - ✓ 起動直後は”# Error loading design”が出るが再コンパイルで解消されるので気にしなくてよい
    - ✓ テストベンチを書き換える都度で再コンパイルが必要となる

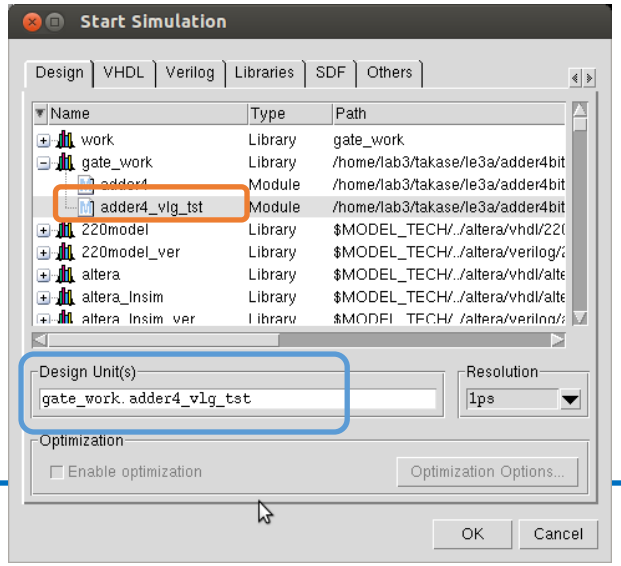


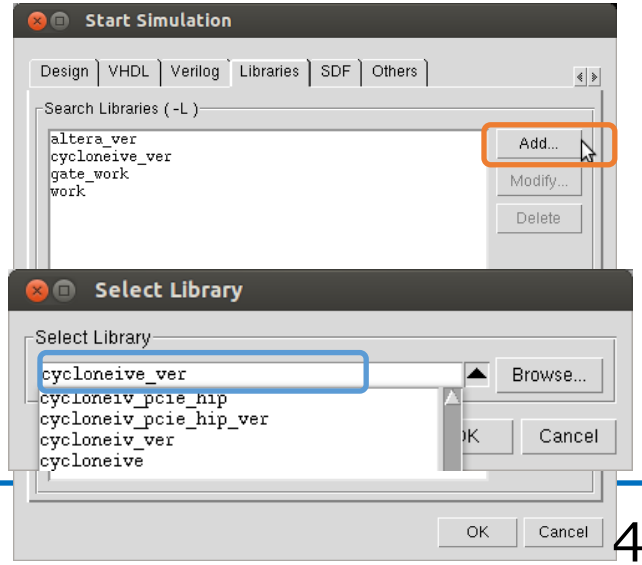
f. シミュレーション

- シミュレーションの開始
  - 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“ と “altera_ver” は選択しておく
    - ✓ 基本的にはデフォルトで追加されている。

ない場合は、テストベンチにエラーがあるか、設定が間違っている。エラーメッセージを確認。







f. シミュレーション

- シミュレーションの実行
  - “Objects” ウインドウから観測したい信号線を
    ”Wave” ウイ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” ウインドウで直接入力も可能 (例：run 100ns)
      (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” ウインドウからRecompileして “Restart”

f. シミュレーション

Objectsウインドウ
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ウインドウ
Zoom変更可

Library
ウインドウ

Transcript
ウインドウ

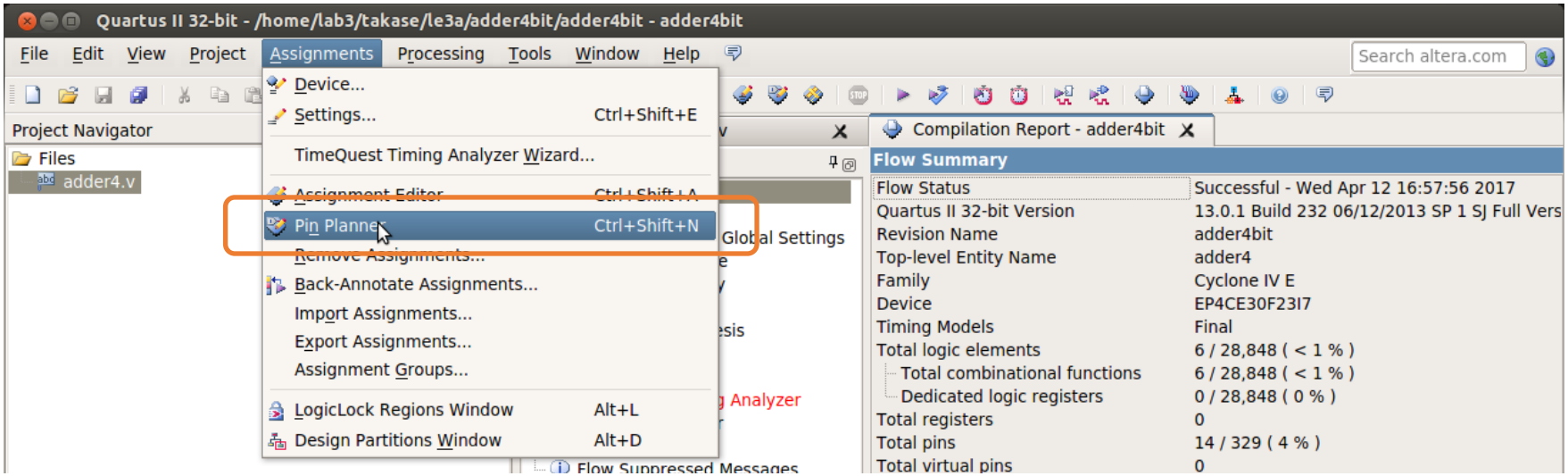
いろいろ
触ってみましょう

g. 入出力ピンの割当て

- 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
- 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えてみましょう

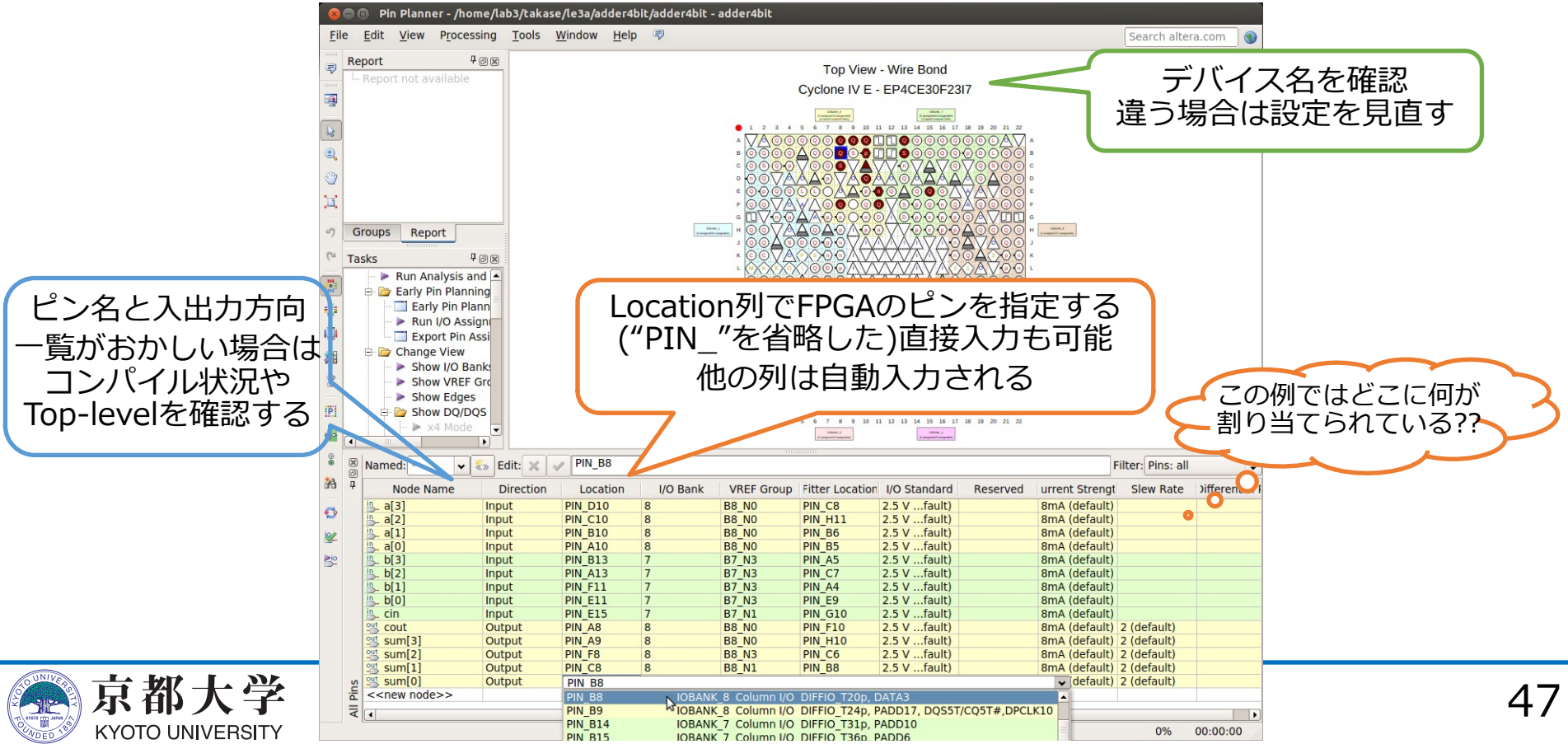
g. 入出力ピンの割当て

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



g. 入出力ピンの割当て

- 入出力ピンを割り当てていく
  - 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

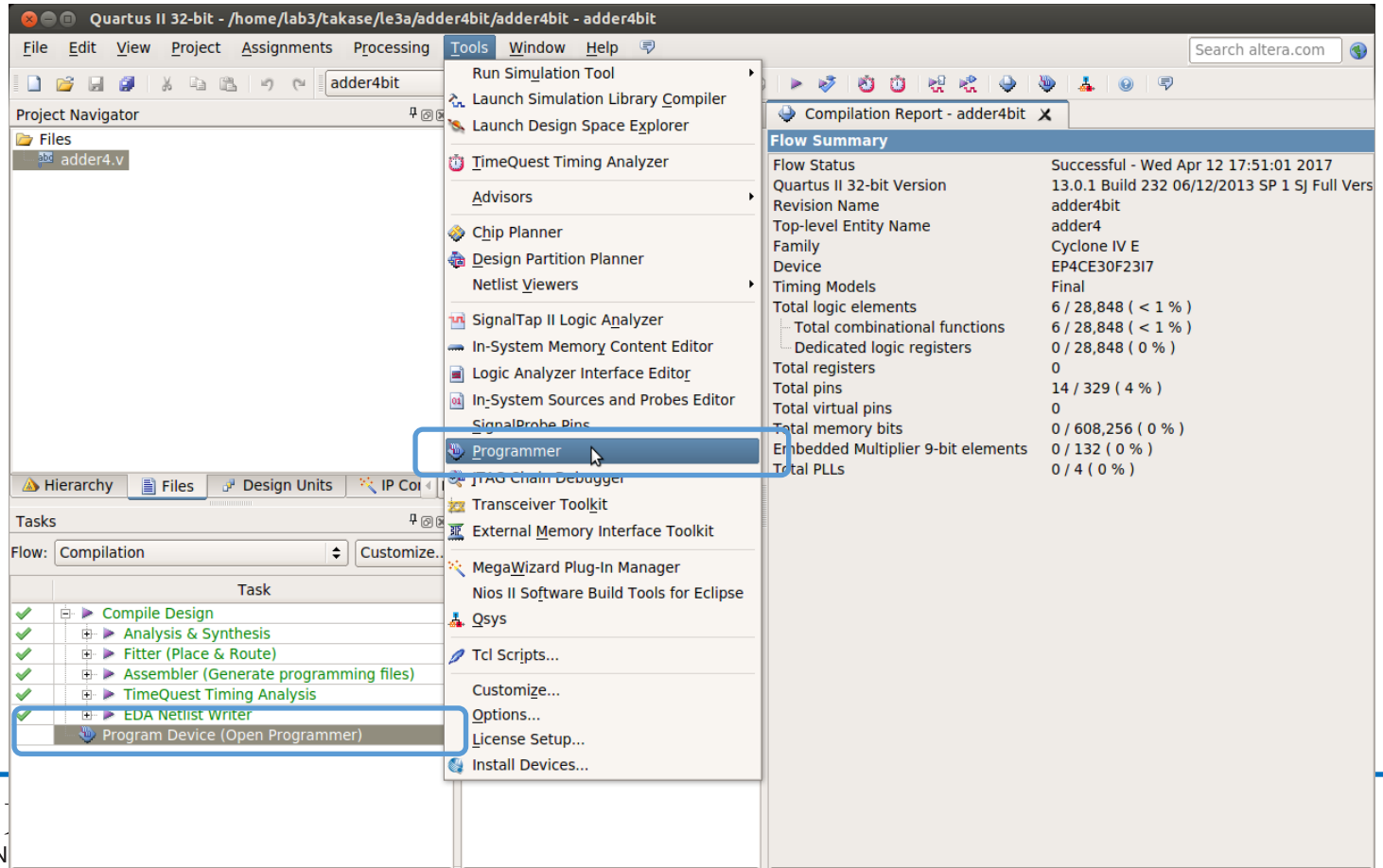


h. FPGAへの回路の書き込み

- 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

h. 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)”



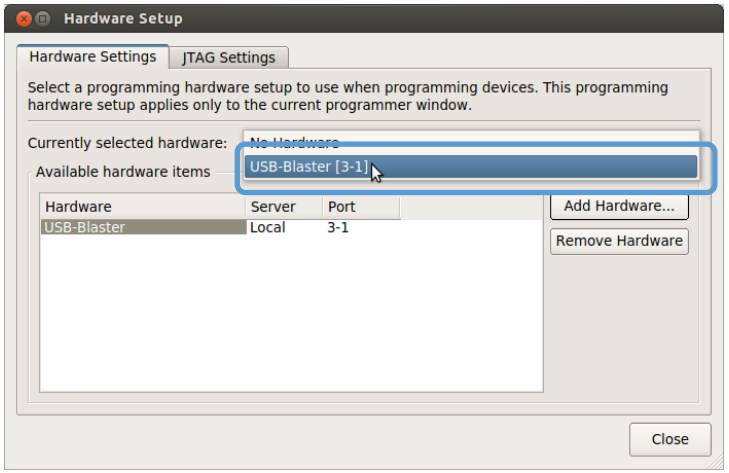
h. FPGAへの回路の書き込み

- FPGAボードをPCに接続
  - 電源に接続
  - USBを接続

h. FPGAへの回路の書き込み

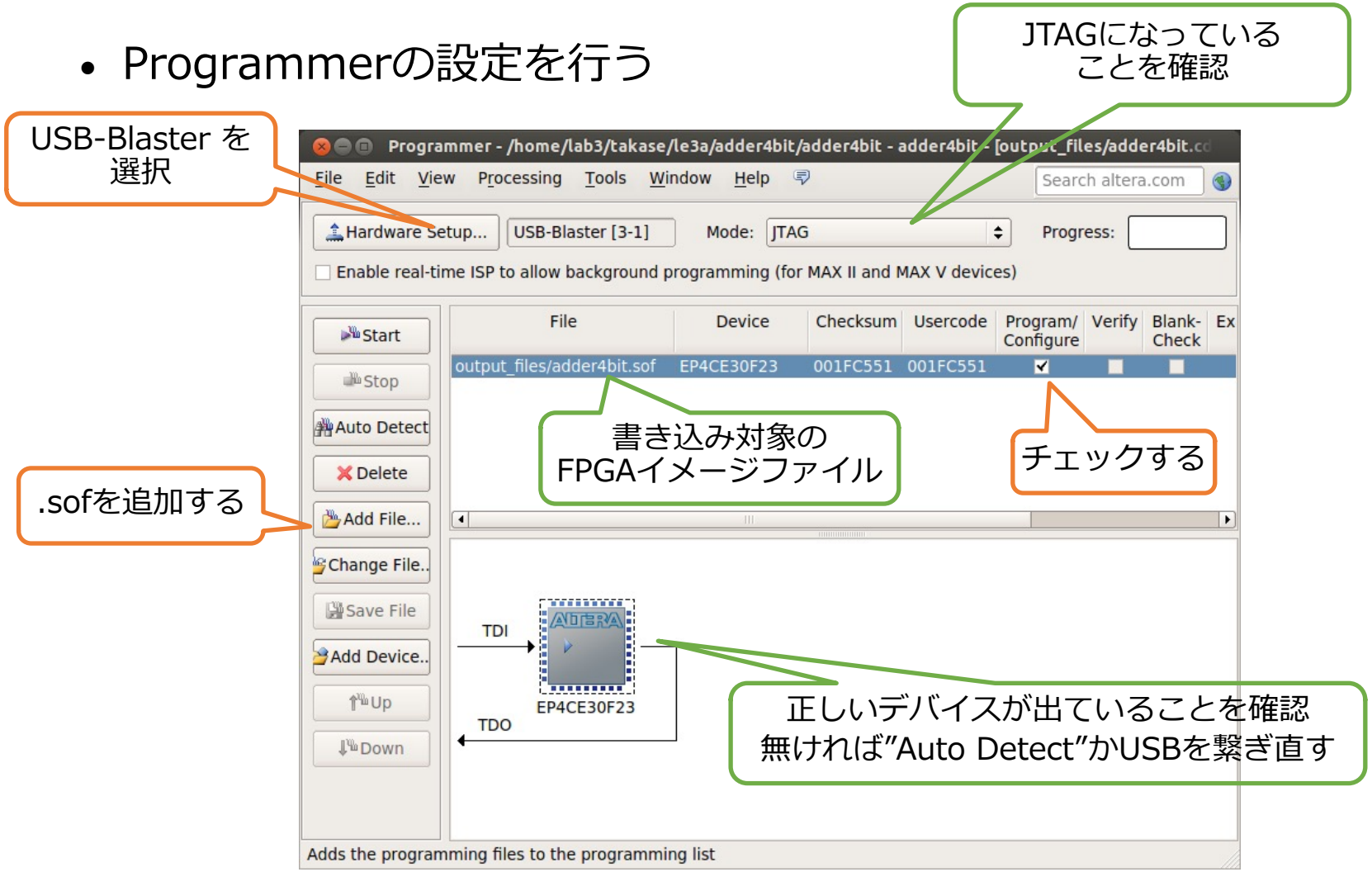
数字は環境によって異なる

- 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



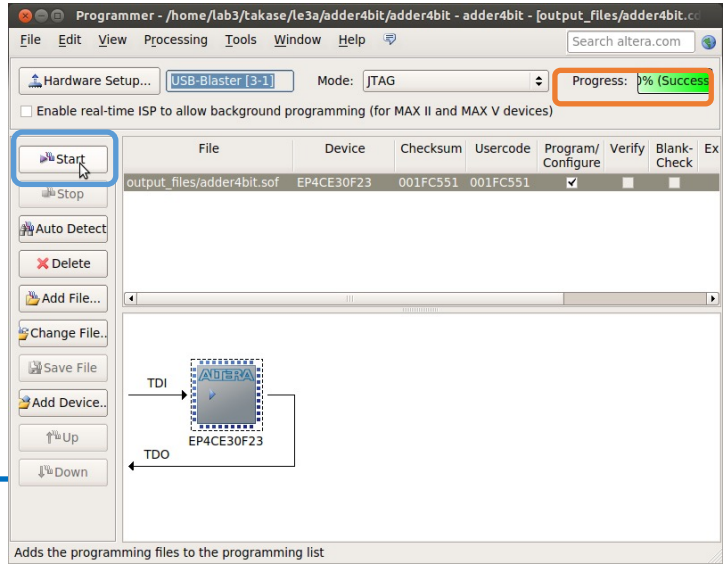
h. FPGAへの回路の書き込み

- Programmerの設定を行う



h. 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
- 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！



TIPS

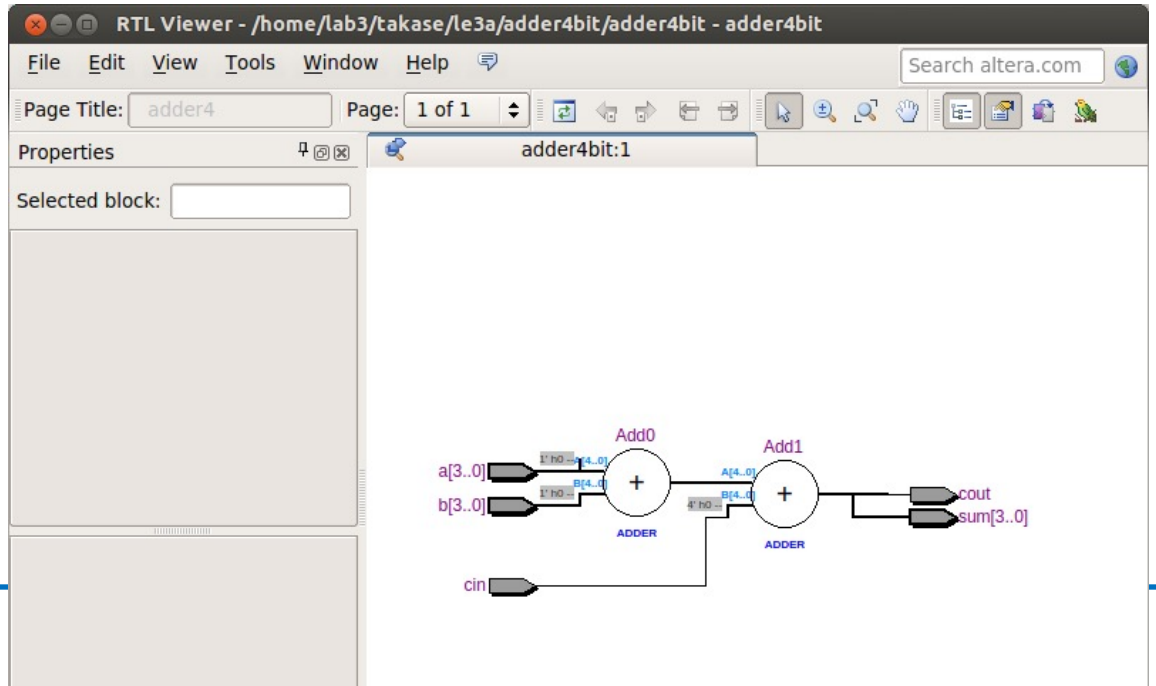
- うまくいかない場合はまずFAQを確認。
  - 過去にあった様々な質問とそれに対する回答がまとめられています。
- 実機で思った通りに動かない
  - まずはシミュレーションで論理的に正しく設計できているか，きちんと確認しましょう
  - シミュレーションが上手くいっていても実機で動かないこともよくあります．多くはタイミング制約を満足できていないことに起因します．FAQを参考にしてください.
- 入出力装置が思った通りに動かない
  - もちろん設計が正しいかは当然として，入出力装置にはそれぞれ正論理と負論理があります
  - 7SEG LEDはセレクタ信号の設定も必要です
  - 未割り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