Hardware and Software Laboratory
Project 3 (Hardware)

“Design Flow Using CAD Tools”

Hardware and Software Laboratory Project 3

In Charge of Hardware

Computer Science Course, School of Informatics and Mathematical Science, Faculty of Engineering, Kyoto University

Lecture Outline

- Design flow using Quartus Prime (a CAD tool)
  - How to use it for HDL design
  - Writing to the FPGA and execution on an actual device
- Lecture structure:
  - I will go give a semi-hands-on presentation using an adder as an example.
  - If you get stuck on adder design, be proactive and ask a TA!
  - If you can do it on your own, you can go ahead and get started on your design for a 7SEG LED drive circuit and counter for the introductory assignment.
- For those who forgot everything over spring break...
  - http://www.lab3.kuis.kyoto-u.ac.jp/~takase/le3a/le2hw3-2019.pdf

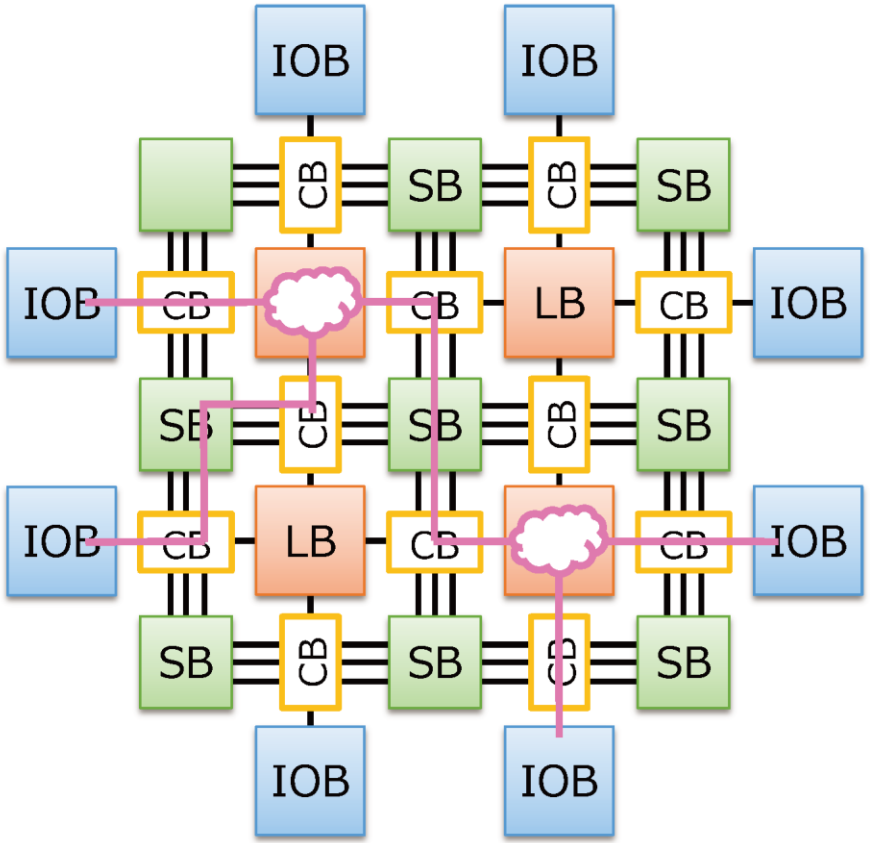
Notes for AY2021

- This document has been prepared to be compatible with the laboratory PC environment.
  - Ubuntu 16.04 LTS
  - Quartus Prime 17.1 Standard Edition
  - ModelSim Intel FPGA Starter Edition 10.5b
- Please replace descriptions according to your own PC environment.
  - The basics should not change much.
  - If there are a lot of issues due to different PC environments, I'll let you know via Slack, etc. as required.

I'll use balloons to list the points that I know are important to keep in mind.

Review: What is an FPGA?

- Field Programmable Gate Array
  - An LSI whose contents can be modified (PLD: Programmable Logic Device)
  - Can change the behavior of the hardware itself
  - Allows you to form your own digital circuits freely as many times as you like
  - The two major FPGA vendors: Xilinx, Altera (powered by Intel)



LB Logic block

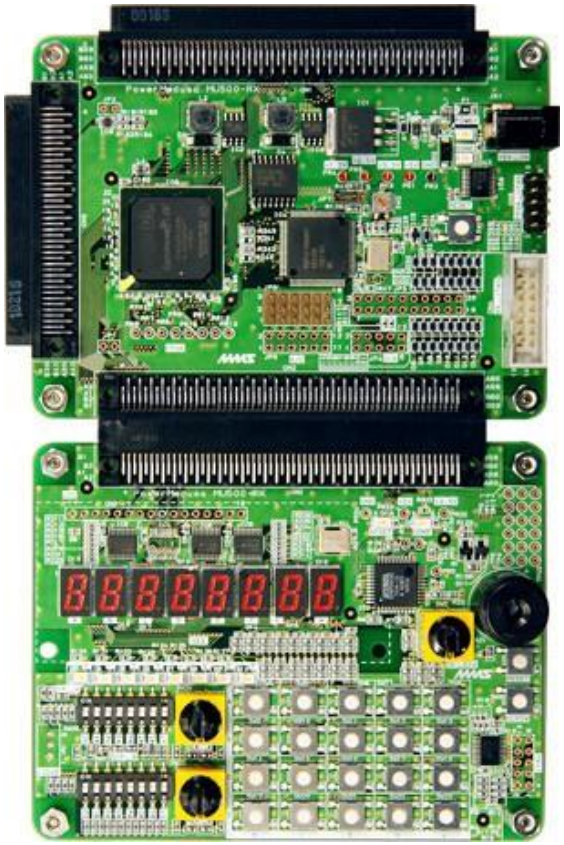
SB Switch block

IOB Input/output block

CB Connection block

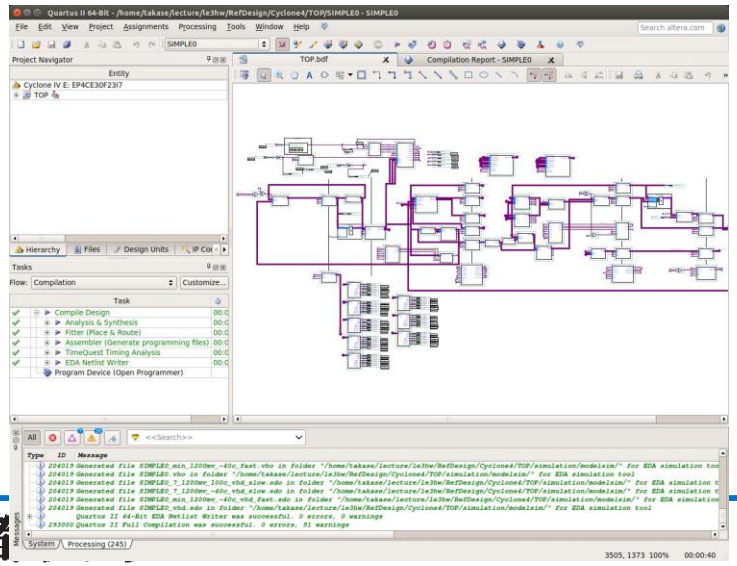
Experimental Environment: FPGA Board

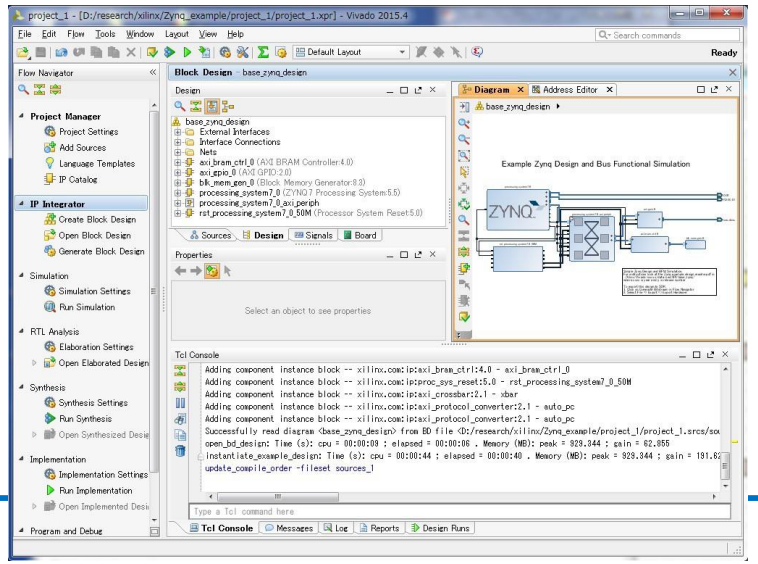
- PowerMedusa MU500-RX/RK
  - FPGA: Altera Cyclone IV EP4CE30F23I7N (28,840 LE)
  - Microcontroller: Renesas RX210
  - Clock: 20 MHz oscillator
  - I/O User Interface
    - ✓ Push SW x20, Rotary SW x2, 8bit DIP SW x2, Clock SW x1
    - ✓ 7SEG LED x8, LED x8, Buzzer x1
  - Expansion Board: MU500-7SEG
    - ✓ 7SEG LED x64, LED x64



Review: CAD Tools

- Computer-Aided DesignTools to design and synthesize logic circuits
  - They can do environment setup, circuit design and description, circuit synthesis, pin assignment, simulation, and programming (writing to FPGA).
  - Also referred to as EDA (Electronic Design Automation) tools.





Experimental Environment: CAD Tools

- Use “Intel Quartus Prime 17.1”
  - Not the latest version (19.1)!
  - The “Standard Edition” is already installed on the lab PCs.
  - When searching Google for help, make note of the version number!
    - ✓ Don't rely on the official manuals and documentation for other versions.
- You can install it on your own PC.
  - The Lite Edition with limited functionality is available for free.
    - ✓ Use ModelSim-Intel FPGA Starter Edition for the simulator.
      - ▫ The non-starter edition of Intel FPGA requires a paid license.
    - ✓ See the supplement on the next slide for more details.
  - Windows/Linux only (Not compatible with macOS)

Supplement: Installing Quartus Prime

- How to install **Quartus Prime 17.1 Lite Edition**
  - Download the installer for 17.1 Lite Edition.
    https://fpgasoftware.intel.com/17.1/?edition=lite
    - ✓ You need to create an Intel account.
    - ✓ You will sometimes get a “Legal Disclaimer” warning you that it is not the latest version. Ignore the warning and just click “I Agree.”
    - ✓ The “batch file” is 5.8 GB.
    - ✓ If you're downloading the “individual files,” download the following to the same location on your drive (total 3.2 GB).
      - Quartus Prime (includes Nios II EDS)
      - ModelSim-Intel FPGA Edition (includes Starter Edition)
      - Cyclone IV device support
  - (continued on next page)

Supplement: Installing Quartus Prime

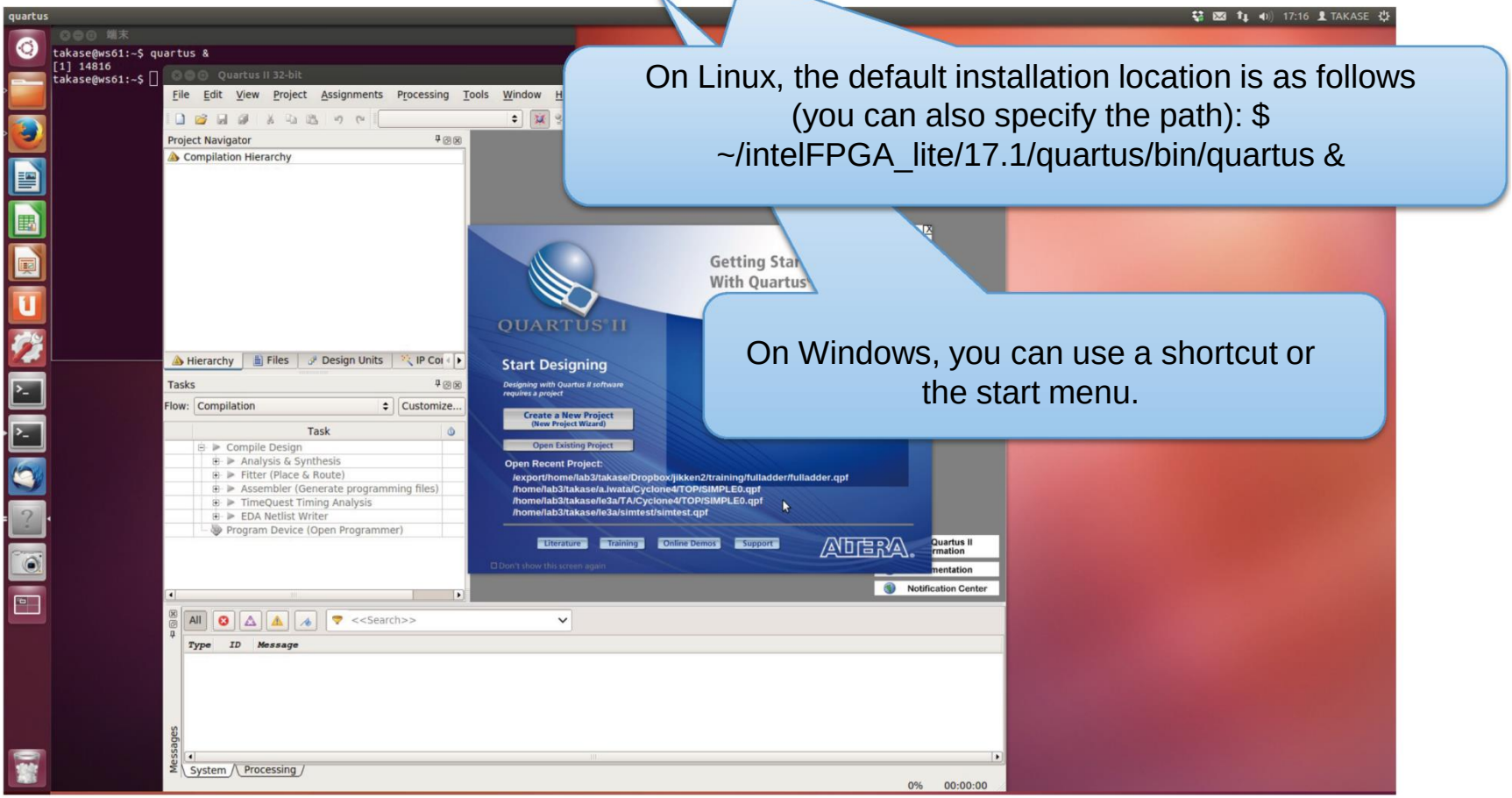
- How to install **Quartus Prime 17.1 Lite Edition**
  - In the case of “individual files,” please place the three files in the same location.
  - Start the installer and run it with the appropriate checkboxes checked.
    - ✓ The installation directory should be the default (the explanation in these slides assumes the default location).
    - ✓ In Select Components, the following must be checked.
      - ¤ Quartus Prime (includes Nios II EDS)
      - ¤ Devices > Cyclone IV
      - ¤ ModelSim - Intel FPGA **Starter** Edition
        - » NOT ModelSim - Intel FPGA Edition!!
  - If you are using Windows, please also install the USB Blaster II device driver that appears after the Quartus installation is complete.
    - ✓ If the USB cable of the FPGA board is not recognized properly, please refer to this page.

Supplement: Installing Quartus Prime

- Installation of Quartus Prime on your PC
  - The Lite Edition with limited functionality is available for free.
  - The ModelSim-Intel Starter Edition is a simulator that is available for free.
    - ✓ The non-starter edition of ModelSim-Intel requires a paid license.
- What is the difference between editions?
  - The Standard Edition, which is the same as the one installed on the lab PCs, is a 30-day trial.
  - Pro Edition is not available (Cyclone VI equipped on board is not supported).
  - There is no significant difference at the level of use in this exercise, except that Lite does not have “multi-processor support.”
    - ✓ Multi-processor support can reduce the compile time for synthesis. Other differences
- Synthesis performance and optimization results may differ between editions.
  - When writing your report, please specify the edition used.

Boot up Quartus

- In the terminal, type $ quartus &



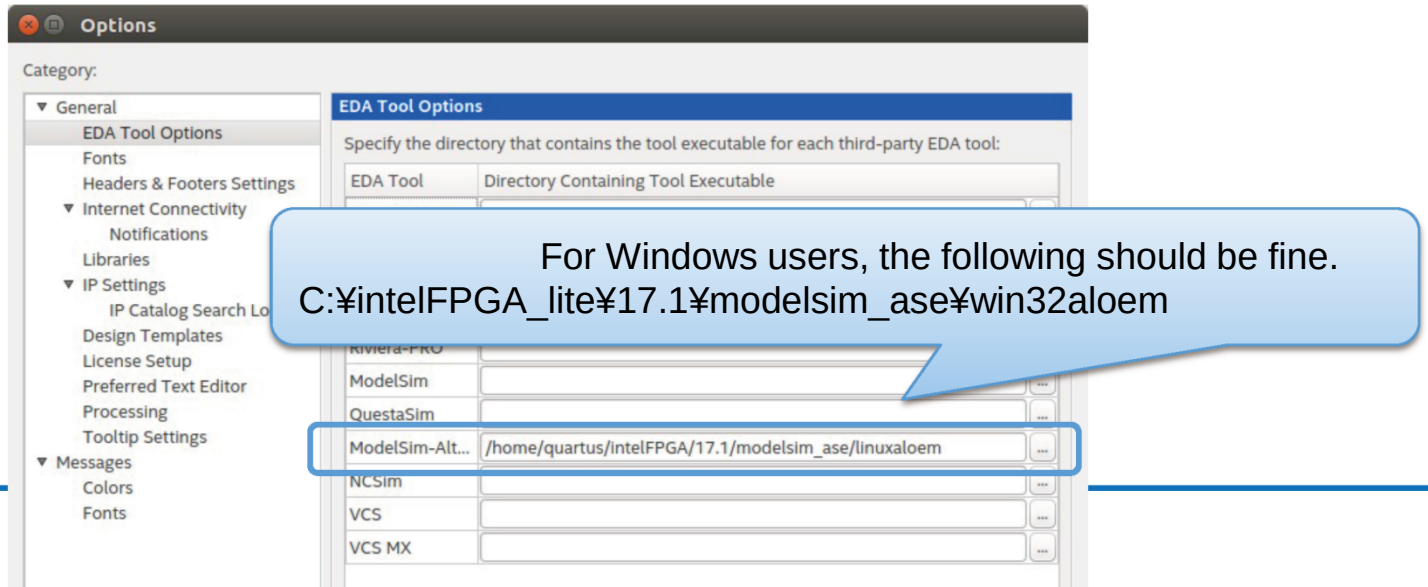
Check Your Settings

The Lite Edition does not require any license settings. Select 'Run the Quartus Prime software' when prompted at the first startup.

- Activation
  - Tools -> License Setup... Make sure you are not getting a "Not found" message.
  - If it says "Not found" or "License Setup Required" appears at startup, enter "27000@is-fpg-00" in the "License file:" field, or check "LM_LICENSE_FILE variable:" and enter it.
- Web browser settings
  - Navigate to Tools -> Options, then Internet Connectivity... and set the path to your preferred browser in "Web browser:" field.
  - If you don't have any preference, use "/export/share/bin/firefox."
    - ✓ This is useful during error checking.

Check Your Settings

- Configure the settings to call modelsim-ase from Quartus.
  - Tools -> Options... -> EDA Tool Options: ModelSim-Altera:
    /home/quartus/intelFPGA/17.1/modelsim_ase/linuxaloem
    - ✓ Note that the ‘s’ may be missing.
    - ✓ You can leave it blank for the moment.



Tool Use and Design Flow

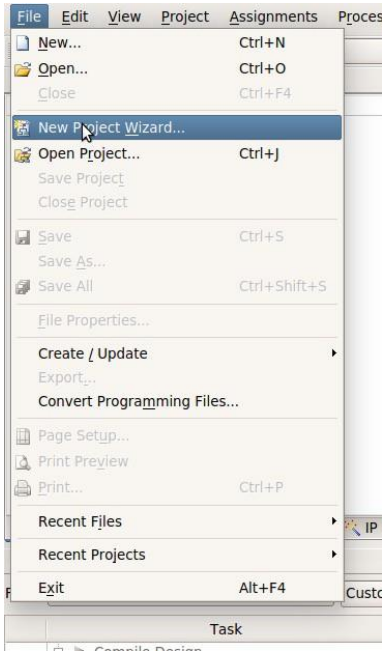
a. Creating a New Project and Configuring the Environment
b. Logic Circuit Design
c. Creating HDL Code
d. Compilation
e. Setting and Verifying Timing Constraints
f. Verifying Operation via Simulations
g. Assignment of I/O Pins to Top-Level Circuits
h. Writing the Circuit to the FPGA

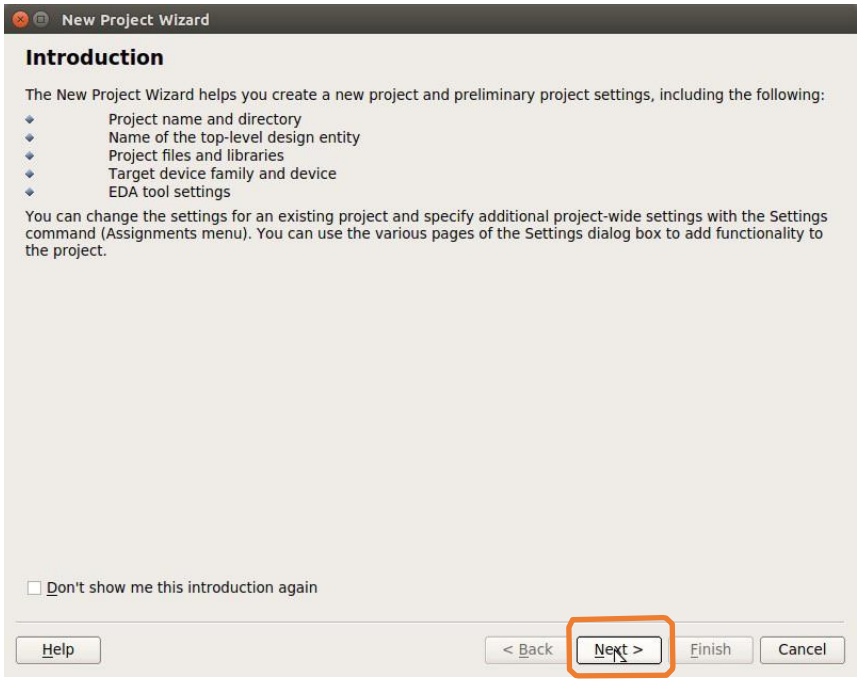
This project is a combinational circuit, so this step is not needed.

The design flow is explained using a 4-bit adder as an example.

a. Creating and Configuring a New Project

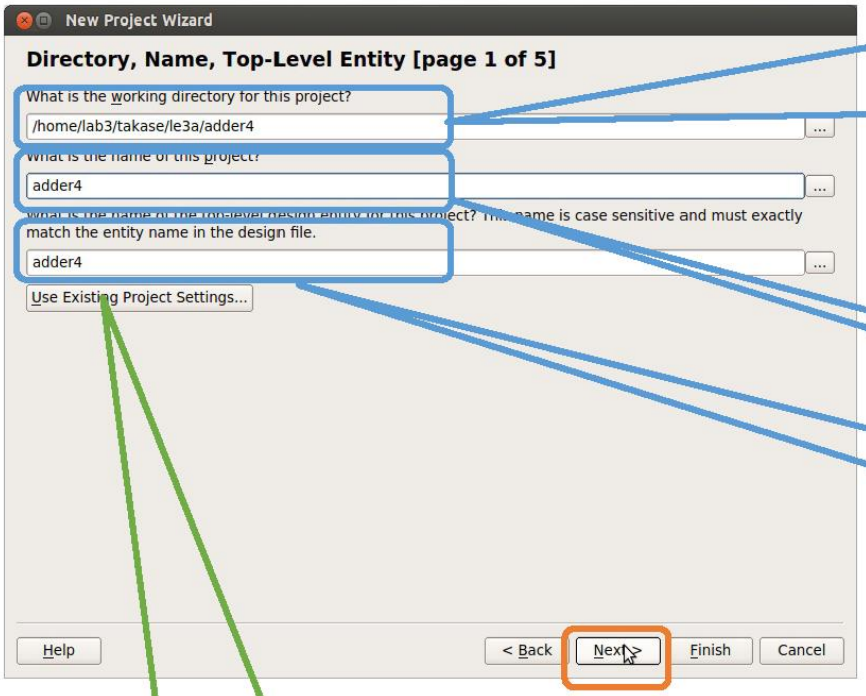
- File -> New Project Wizard...
- Introduction “Next”





a. Creating and Configuring a New Project

- Directory, Name, Top-Level Entity



Working directory:
Make sure to create a dedicated directory; do not use your home directory. If you do not do it this way, the tool may freeze or crash. Also, it is best to create a directory for each project (you can name it whatever you want).

Project name

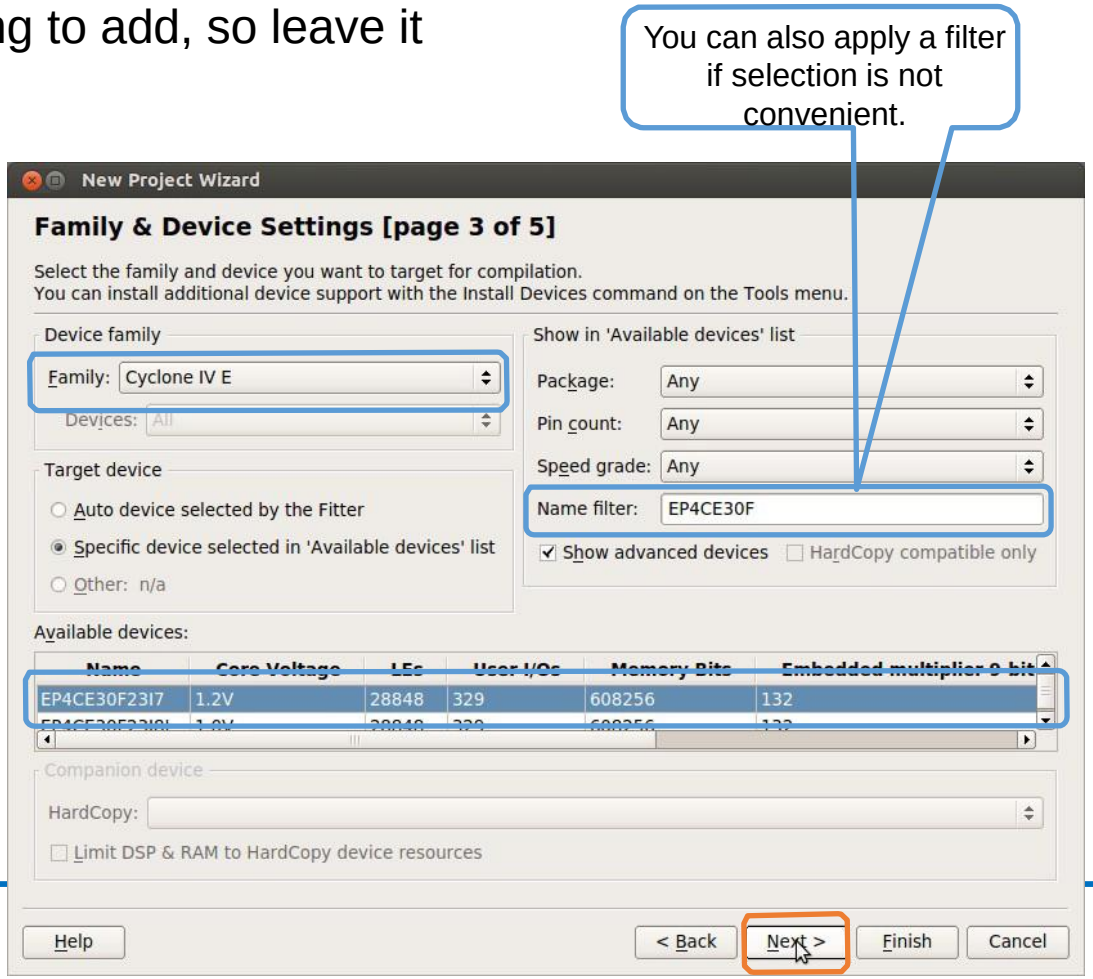
The name of the top-level circuit:
Same as the project name if you don't change it to something specific (autofilled).

Avoid project names that start with numbers.

If the working directory does not exist, it will be created automatically.

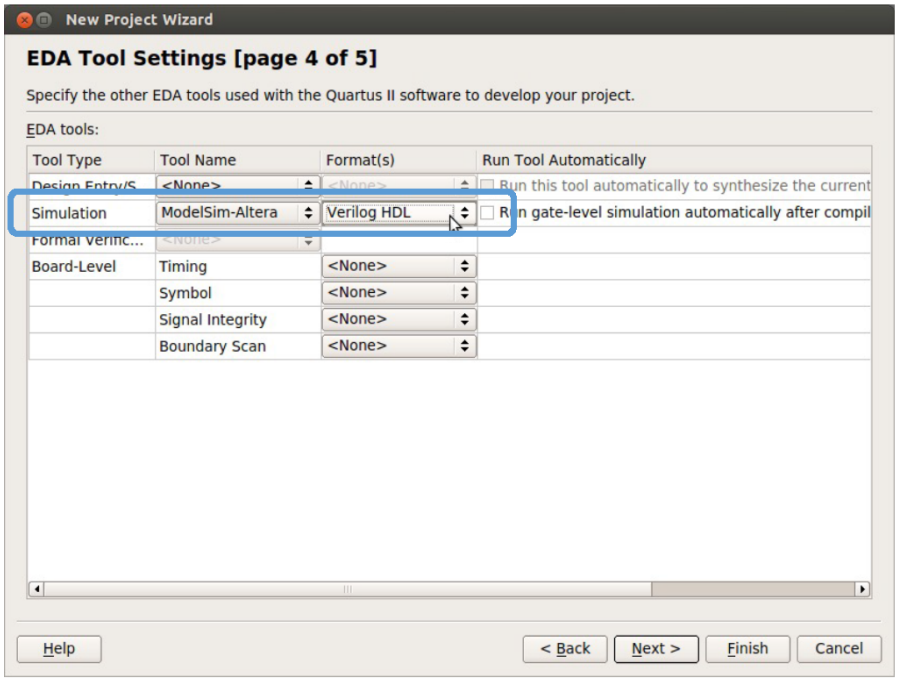
a. Creating and Configuring a New Project

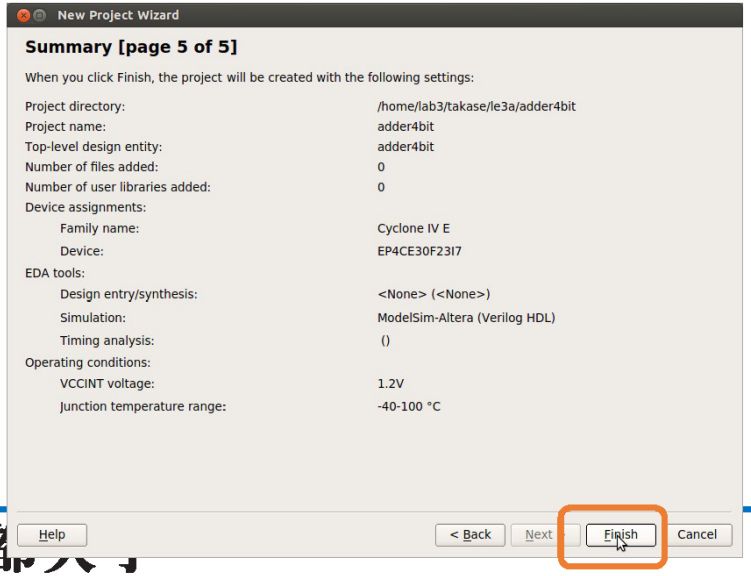
- Project Type: Select “Empty project”
- Add Files: Add already created circuits to the project.
  - This time, there is nothing to add, so leave it blank and click “Next.”
- Family & Device Settings
  - Family:
    “Cyclone IV E”
  - Available devices:
    “EP4CE30F23I7”



a. Creating and Configuring a New Project

- EDA Tool Settings
  - Specify if you want to use/modify external design tools.
  - Specify the HDL to be used in the “Simulation” step.
- Summary
  - Confirm the settings and click “Finish.”





b. Logic Circuit Design

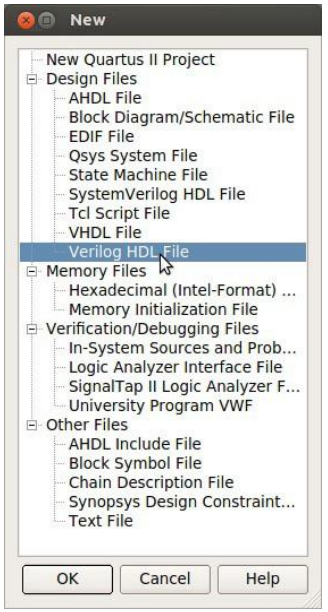
- Think about the logic circuit you want to design.
  - External specifications: I/O, functions
  - Internal specifications: Circuit configuration
    - ✓ Truth tables, Karnaugh maps, state transition diagrams, etc. → Logic formula expression
      - → There is no need to go into this much detail in HDL design, but you should think long and hard about what kind of HDL code you want to write.

First think about the design, then start using the tool!

- External specifications of 4-bit adder
  - Input: 4-bit additive data, 1 bit of carry-in
  - Output: 4-bit sum, 1-bit carry-out

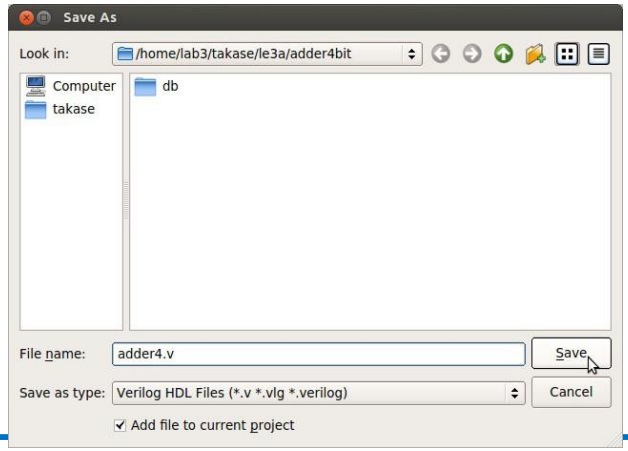
c. Creating HDL Code

- Use the HDL editor.
  - File -> New -> Verilog HDL File or File -> New -> VHDL File
    - ✓ Select it according to the HDL you want to use.
  - Have fun designing with hardware description languages in Experiment 3!
    - ✓ You can also combine with circuit diagrams.
    - ✓ If you make top level circuit diagrams and components in HDL, it will improve the overall visibility.



- First (before you end up crying), save the HDL file.
  - File -> Save As...
  - “adder4.v”
    - ✓ For VHDL: “adder4.vhd”

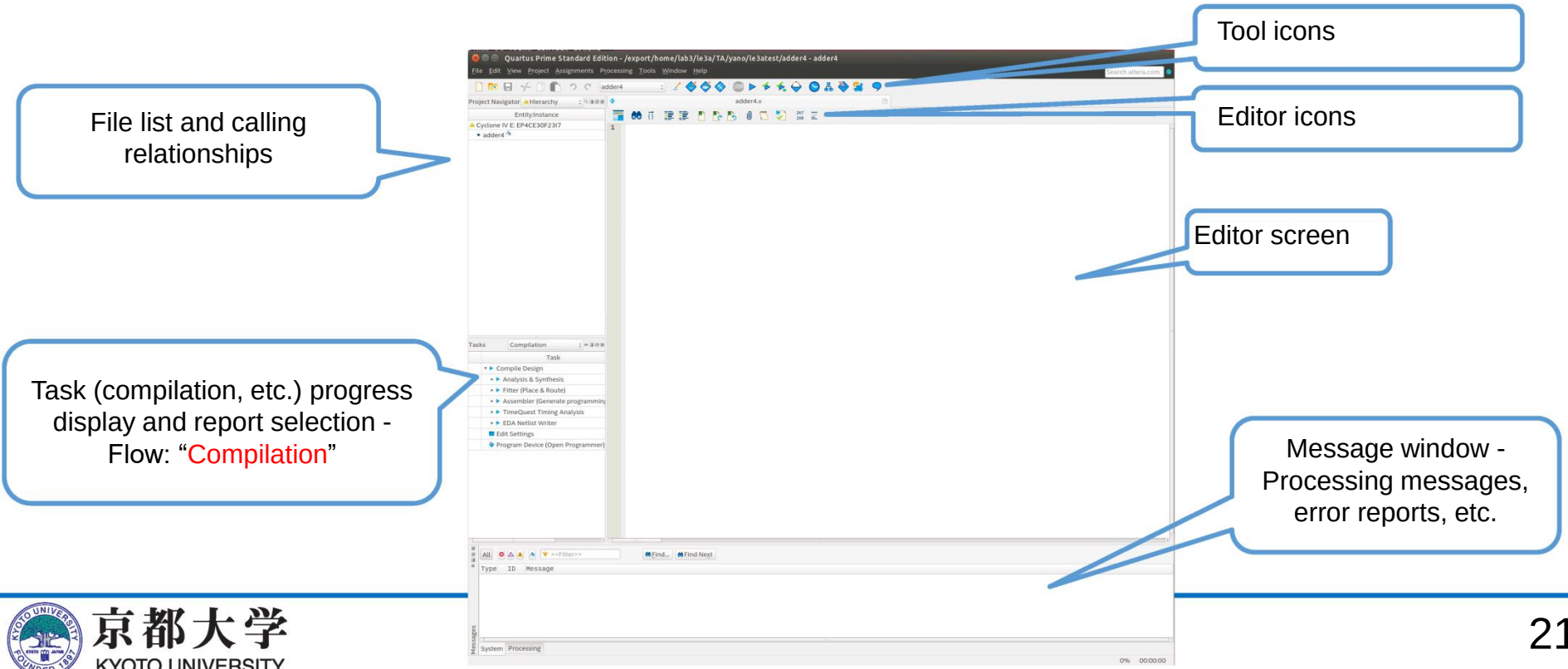
Avoid file names that start with numbers.



- Set the “Flow:” in the Tasks window to “Compilation.”

c. Creating HDL Code

- Configuration of the (initial) window
  - The layout and size of the windows can be adjusted as appropriate to suit your needs via View -> Utility Windows, etc.
  - The editor function is lacking, so you may use your preferred editor (synchronize files with external tools at your own risk).



c. Creating HDL Code

- Example for Verilog HDL

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
```

- Example for VHDL

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
```

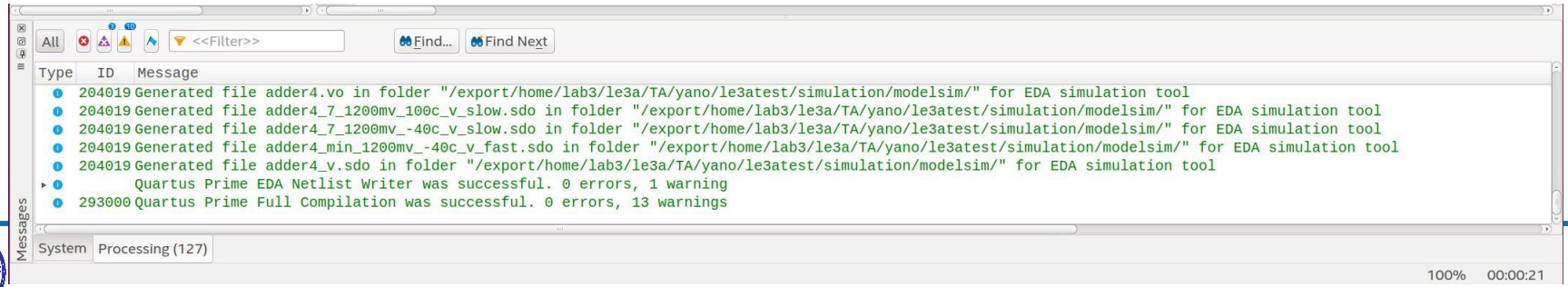
d. Compile

- Convert to configuration information of logic blocks in FPGA (equivalent to C compilation).
  - Convert circuit diagrams to register transfer level.
  - Perform logic synthesis, technology mapping, and placement/wiring.
    - ✓ Assign RTL-described circuits to logic blocks on the FPGA, then place and wire the logic blocks.
  - Generate a bitstream.
    - ✓ Convert placement/wiring information into a binary configuration information file.
- Compilation procedure
  - Processing -> Start Compilation or Start Compilation on Tool Icons or Ctrl+L
    - ✓ Only an input check is possible with Start Analysis & Synthesis.

d. Compile

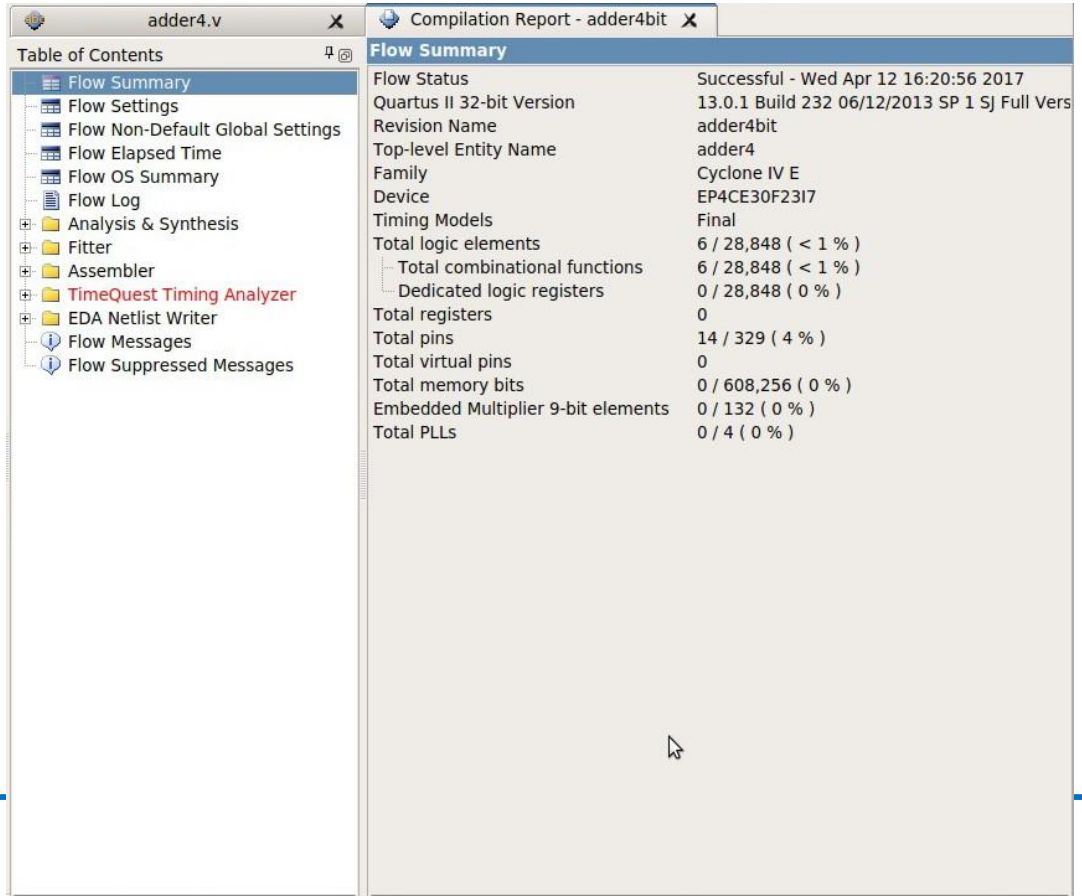
- If all of the ✓ under “Compilation” in the Tasks window are green, the compilation is successful.
  - A yellow ? indicates that the flow requires recompilation.
  - A red × indicates that an error has occurred in the flow.
- If the flow does not terminate normally, check the message window at the bottom or messages found at “Compilation Report -> Flow Messages” and debug based on the messages.
  - Right-click Help to allow online access to messages.
  - You also need to check Critical Waning/Warning.

  ✓ Warnings tend to be quite relentless in CAD tools, but only ignore warnings if you understand the meaning (often they help to discover design mistake or the like).



d. Compile

- Check the result of synthesis with Compilation Report.
  - Processing -> Compilation Report or Ctrl+R
  - Investigate the particulars of each item on your own.



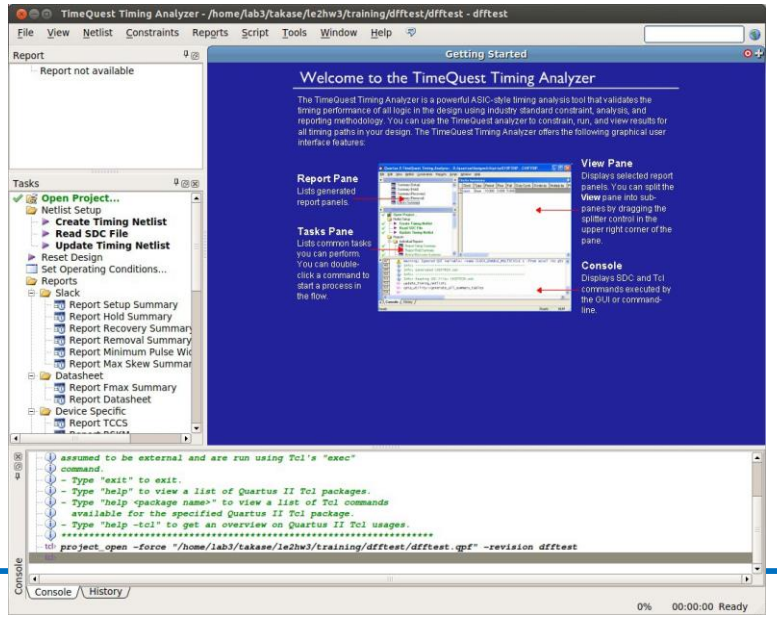
d. Compile

- Circuit size: Fitter -> Summary
  - Usage of each block on the FPGA
  - If any item is over 100%, it will not fit on the FPGA device.
- Operating speed and delay: TimeQuest Timing Analyzer
  - If you set set_max_delay and set_min_delay in the Synopsys Design Constraints (SDC), you can display the propagation delay time (in ns) from each input pin to each output pin (refer to the FAQ).
  - If the input clock is specified in the design of the sequential circuit, the operating frequency, etc. can also be displayed. (We are working on a combinational circuit now, so more on that later.)
  - If there are items in red, timing constraints have not been met.
    - ✓ Unconstrained Paths will disappear if set_max_delay and set_min_delay are set appropriately.

      In sequential circuit design, it is necessary to pay special attention to this.

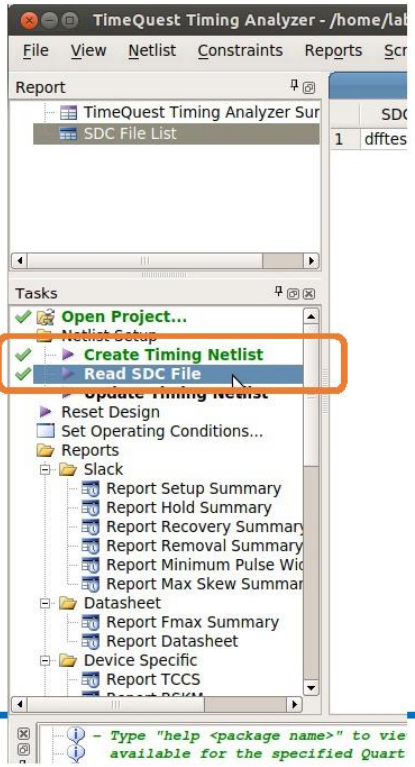
e. Setting Timing Constraints (for reference)

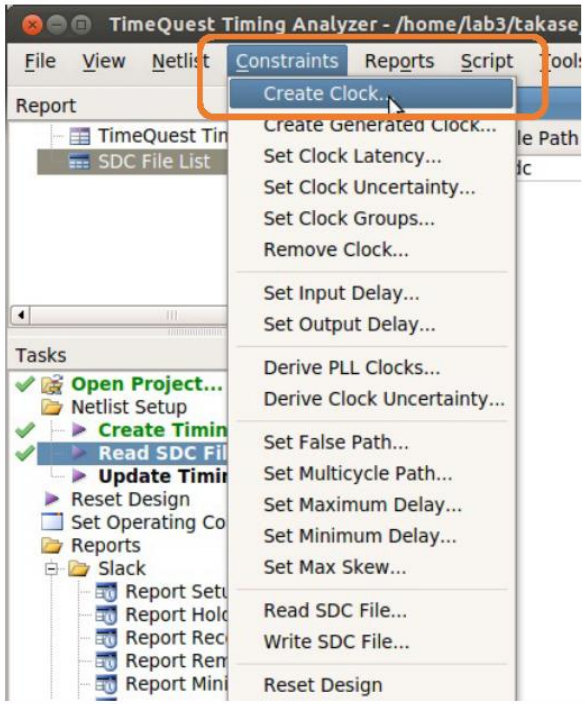
- Specify the clock and set timing constraints (setup time to register input, hold time, propagation delay, etc.).
  - I will only explain the basic clock specification; please look into the details on your own.
- Use “TimeQuest Timing Analyzer”
  - Navigate to Tools -> TimeQuest Timing Analyzer or “TimeQuest Timing Analyzer” on Tool Icons



e. Setting Timing Constraints (for reference)

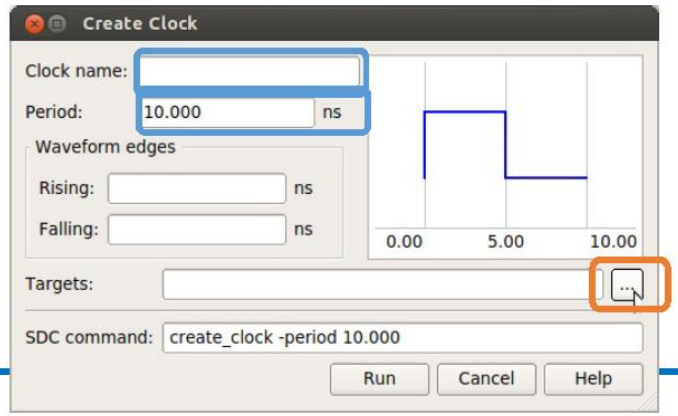
- TimeQuest Timing Analyzer
  - Execute “Netlist Setup -> Create Timing Netlist” in the Tasks window.
  - Execute “Netlist Setup -> Read SDC File” in the Tasks window.
  - Set the clock from Constraints -> Create Clock...

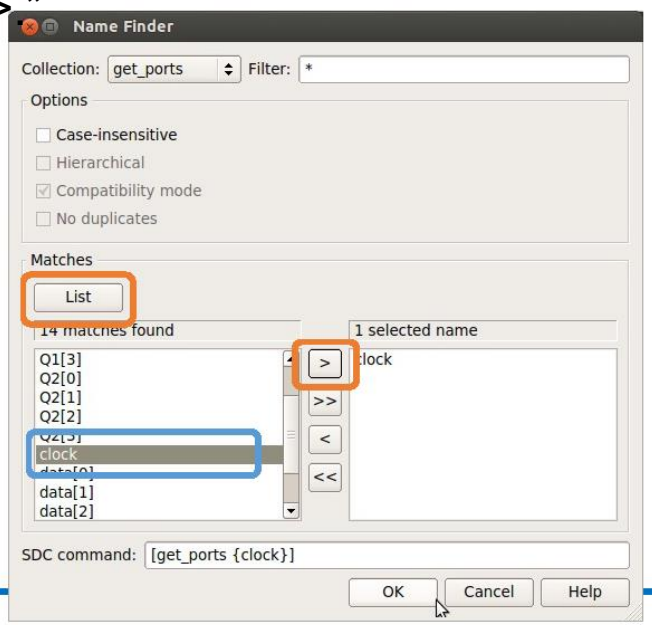




e. Setting Timing Constraints (for reference)

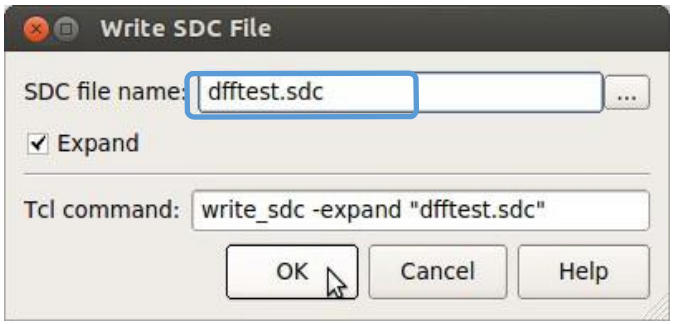
- TimeQuest Timing Analyzer
  - Set the clock from Constraints -> Create Clock...
    - ✓ Clock name: Define clock name, can be left blank (e.g. clock).
    - ✓ Period: Set clock period.
    - ✓ Waveform edges: Set rise/fall time.
    - ✓ Targets: Select the clock on the design circuit.
      - ¤ Display from “List” and select with “>”
    - ✓ The constraints generated are displayed in SDC command.





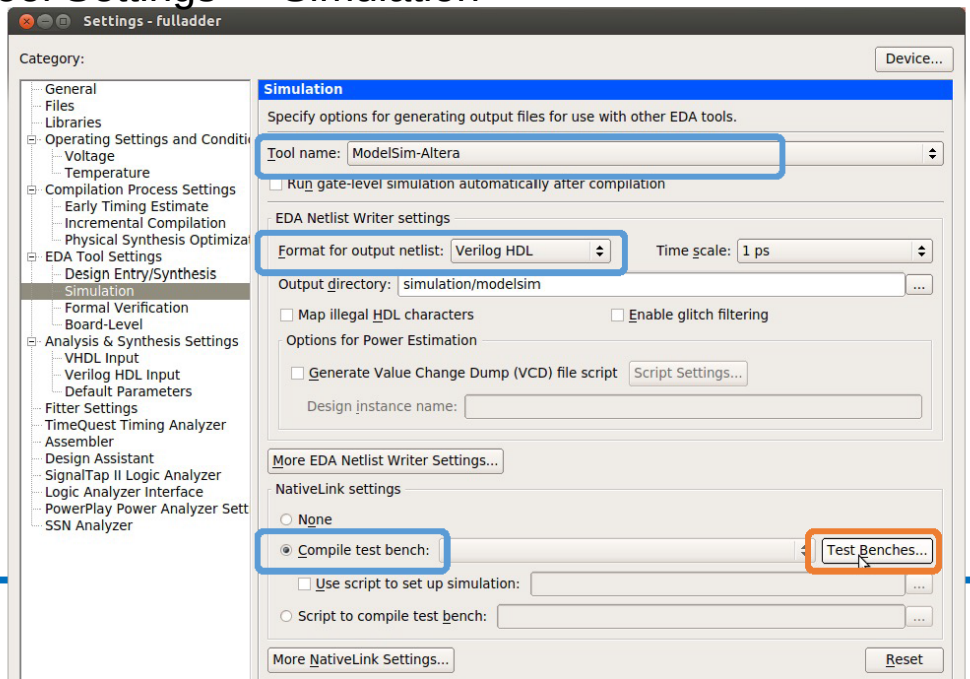
e. Setting Timing Constraints (for reference)

- TimeQuest Timing Analyzer
  - Various reports on timing will be reported under Reports/.
    - ✓ If you are not satisfied with any of them, you can add constraints.
    - ✓ You can also check the reports in the Compilation Report.
    - ✓ Fmax Summary: The maximum frequency at which the system can operate (depending on the constraints you provide)
    - ✓ Clocks: Clock frequency of the design circuit at the current setting
  - Save the constraint file by clicking “Write SDC File...” in the Tasks window.
    - ✓ If you name it <module>.sdc, it will be automatically recognized by the project. (It is better to remove the .out.)
    - ✓ Once you get used to it, you can create a text-based constraint file and edit it.



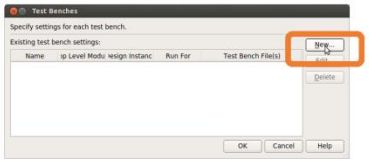
f. Simulation

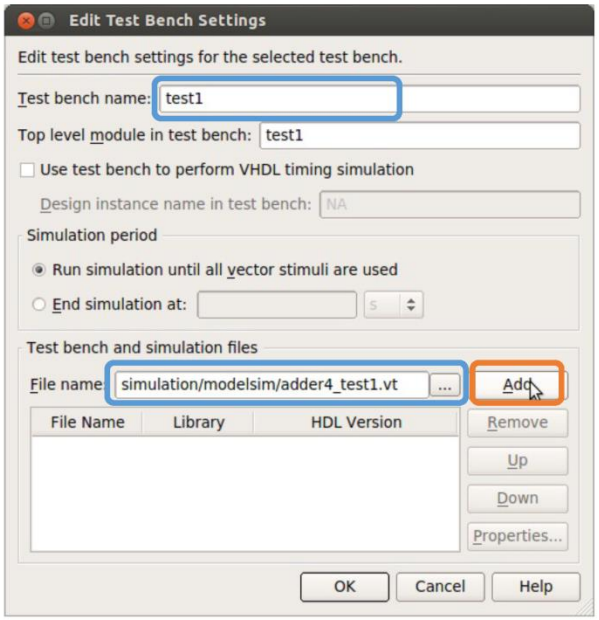
- Use a waveform editor to check if the logic circuit you designed works correctly.
  - Use “ModelSim-Intel FPGA Starter Edition (modelsim-ase)”
    - ✓ Note that modelsim-ae is also available (it does not work due to licensing constraints).
- First, configure the setting so that modelsim-ase is called from Quartus.
  - Assignment -> Settings -> EDA Tool Settings -> Simulation
    - ✓ Tool name: ModelSim-Altera
    - ✓ Format for output netlist: Verilog HDL
      - ▫ Select VHDL when designing with VHDL.
    - ✓ NativeLink settings: Compile test bench
    - ✓ Select “Test Benches...”



f. Simulation

- First, configure the setting so that modelsim-ase is called from Quartus.





  - Test Benches
    - ✓ Select “New...”
  - New Test Bench Settings
    - ✓ Test bench name: Give it a name and create a new one.
    - ✓ If you set the “Top level module in test bench:” to the actual test bench module name (<module>_vlg_tst if you use Template Writer), you will not get the “Error loading design” message on the first run of ModelSim.
    - ✓ Test bench and simulation files: simulation/modelsim/<module>_test1.vt (.vht for VHDL)
      - ▫ Create an empty file from the terminal.
    - ✓ Select “Add”
  - Return to Test Benches and select OK.

f. Simulation

Once you get used to it, you don't need to bother creating a template.

- Create a test bench file.
  - Processing -> Start -> Start Test Bench Template Writer
  - ./simulation/modelsim/<module>.vt will be created.
  - Copy it to ./simulation/modelsim/<module>_test1.vt, etc.$ cp ./simulation/modelsim/adder4.vt ¥

    ./simulation/modelsim/adder4_test1.vt ¥
  - Edit <module>_test1.vt so that you have the desired input waveform.
    - ✓ Change the time unit (default is ps, which is too fast).

      `timescale 1 ns / 1 ps
    - ✓ Describe the default settings directly under the following description.

      // code that executes only once

      // insert code here --> begin
    - ✓ Describe the desired value (waveform) directly under the following description.

      // code that executes for every event on sensitivity list

      // insert code here --> begin

f. Simulation

- Example of test bench file notation

  - Assignment:

```
// 1 bit
A <= 0;
B <= B + 1;
// bitwise operation
Cin <= ~Cin;
// 4-bit binary number
 As <= 4’b10_01;
// 4-bit binary number
As <= 4’b10_01;
```

  ✓ Assigned in parallel (if there is no delay).

  - Delay:

```
#1000
```

  - Iteration:

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

  ✓ Useful for generating clock waveforms.

  ✓ Must be written outside of “initial begin - end” or “always begin - end,” and inside of “module - endmodule.”

  ✓ always blocks are executed in parallel with each other.

  ✓ Conditional statements can be written using @().

Notation is the same with Verilog HDL.The explanation for VHDL is omitted.

f. Simulation

- Example of test bench file description
  - The time unit has also been changed.
    `timescale 1 ns / 1ps
  - Think carefully about units and delays.
    - ✓ If it is too small?

adder4.vt (/home/lab3/takase/le3a/adder4bit/simulation/mo

開く 保存 元に戻す

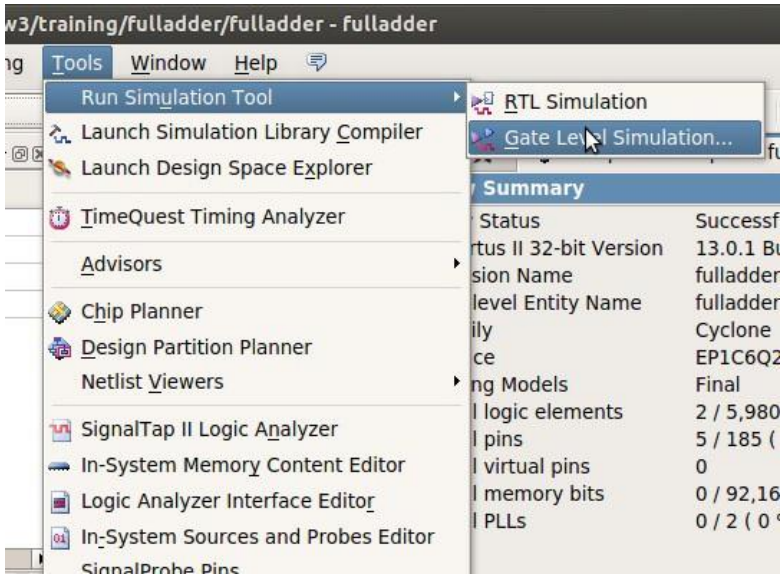
adder4.vt

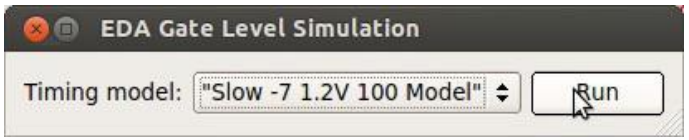
```
);
initial
begin
// code that executes only once
// insert code here --> begin
a <= 4'b0000;
b <= 4'b0000;
cin <= 0;

// --> 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
```

f. Simulation

- Recompile the design (the task window should show a ?).

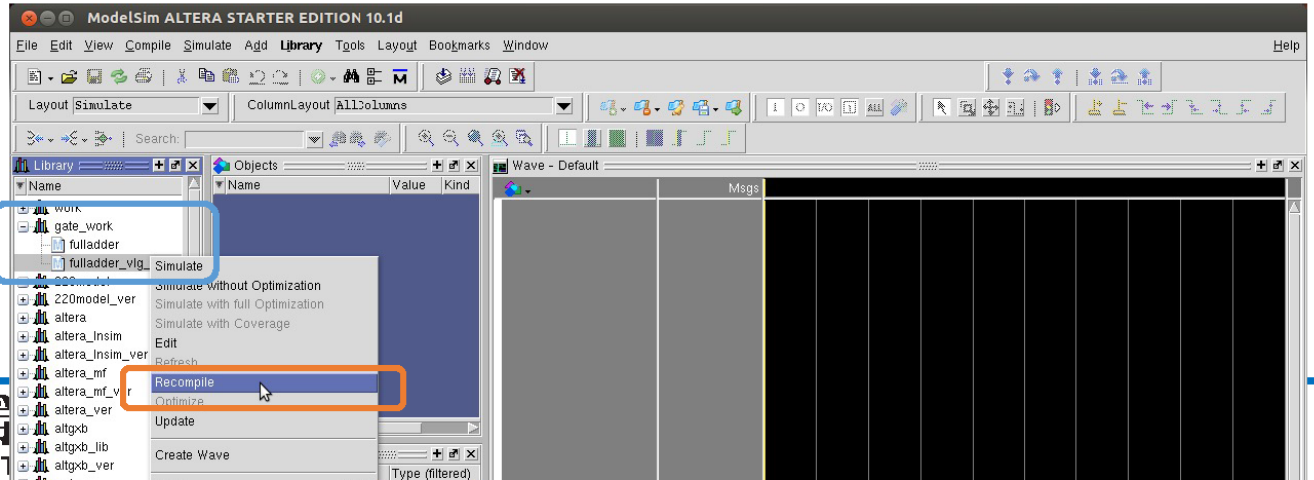
- Start the simulator.
  - Tools -> Run Simulation Tool -> Gate Level Simulation...
  - Timing model: Run “Slow -7 1.2V 100 Model”





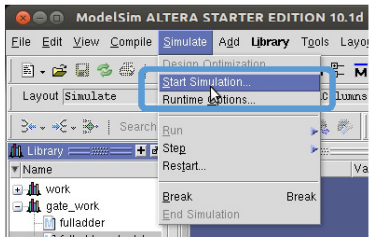
f. Simulation

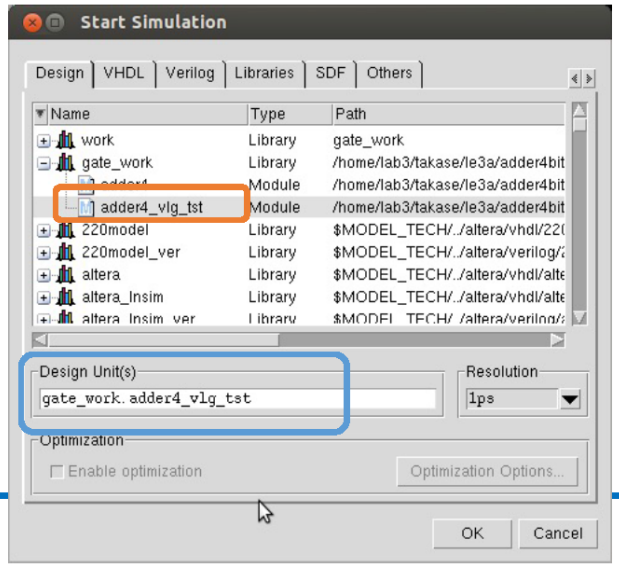
- Start the simulator.
  - The circuit you have designed is located under gate_work in the Library.
    - ✓ A link to gate_work will be under “work.”
- Recompile the test bench file just in case.
  - Right-click on gate_work -> <module>_vlg_tst in the “Library” window and click “Recompile.”
    - ✓ You will get “# Error loading design” right after startup, but don't worry about it. It will go away after recompiling.
    - ✓ You need to recompile every time you rewrite the test bench.

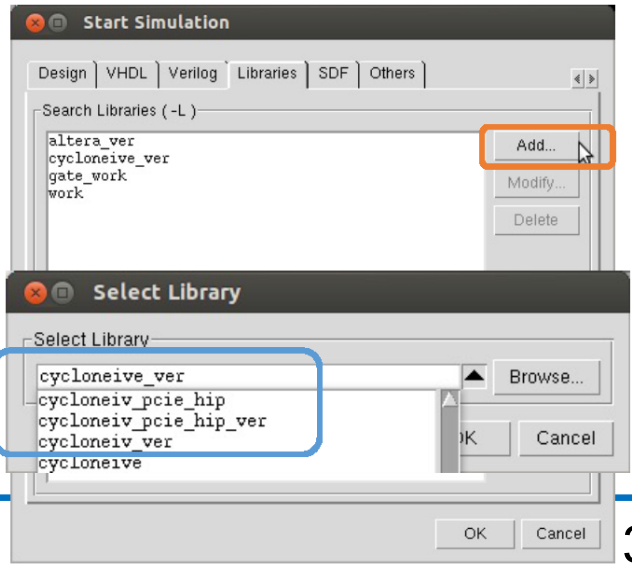


f. Simulation

- Starting simulation
  - Simulate -> Start Simulation...
  - In the Design tab, under “Design Unit(s),” select gate_work.<module>_vlg_tst.
  - In the Libraries tab, under “Search Libraries,” click Add... to add the necessary libraries.
    - ✓ Select “cycloneive_ver” and “altera_ver” in advance.







f. Simulation

- Running simulation
  - Drag the signal line you want to observe from the “Objects” window to the “Wave” window.
  - Set the “Run Length” to the desired size (e.g. 100 ns).
    - ✓ If it is too large, the simulation time will increase.
  - Click “Run,” “Run - All,” etc.
    - ✓ You can also type directly in the “Transcript” window (this way may be faster once you get used to it).
  - Zoom In/Out/Full etc. of the waveform display can be changed by right-clicking or typing.
  - Waveforms are already specified in the test bench, but can also be specified directly.
    - ✓ Right-click on a signal line and select “Force....,” etc.
  - If you have changed the test bench, recompile it from the “Library” window and “Restart.”

f. Simulation

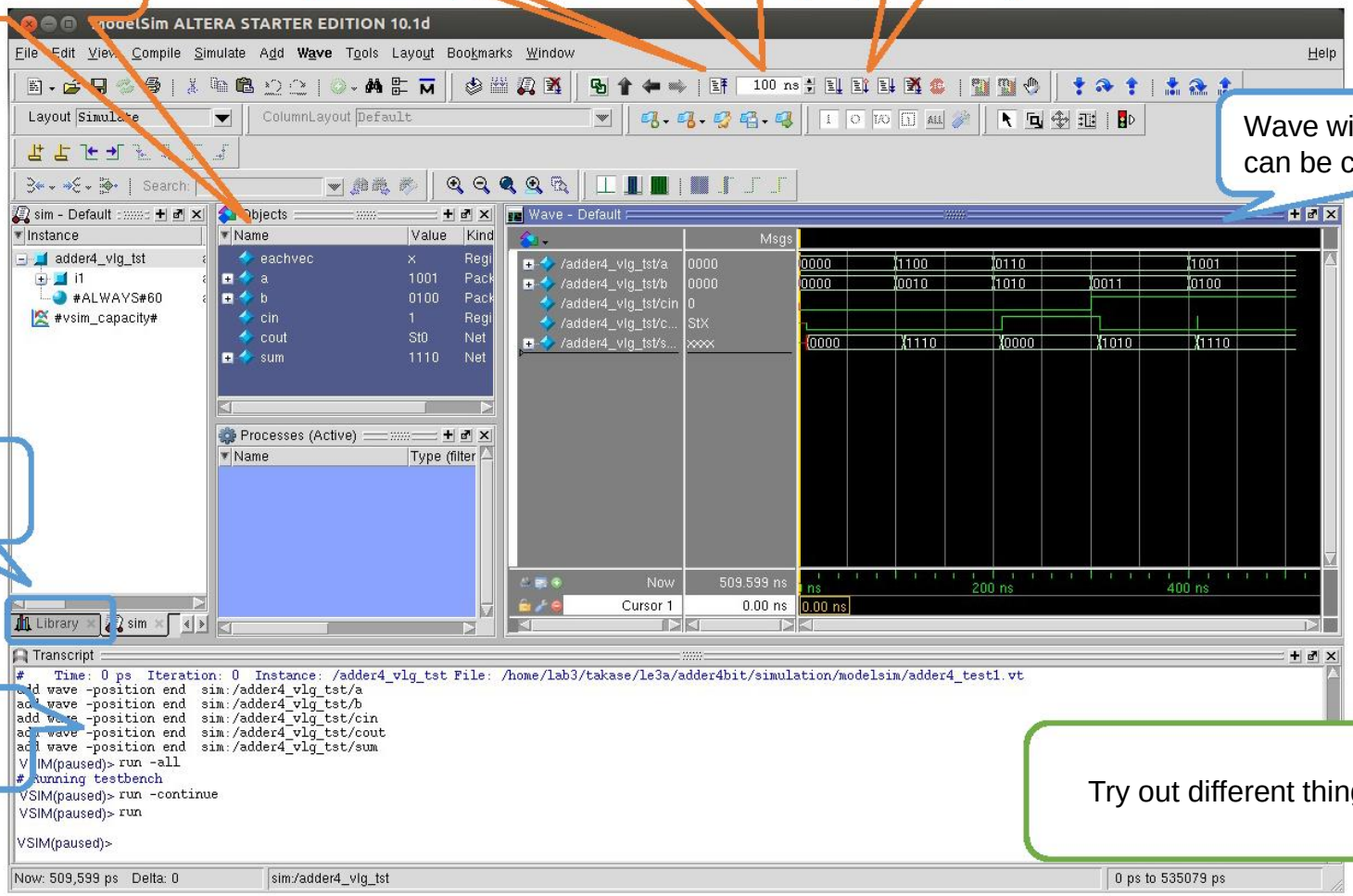
Objects windowDrag the signal you want to observe to the Wave.

Restart

Specify the time unit.

Run, etc.

Wave windowZoom can be changed.



Library window

Transcript window

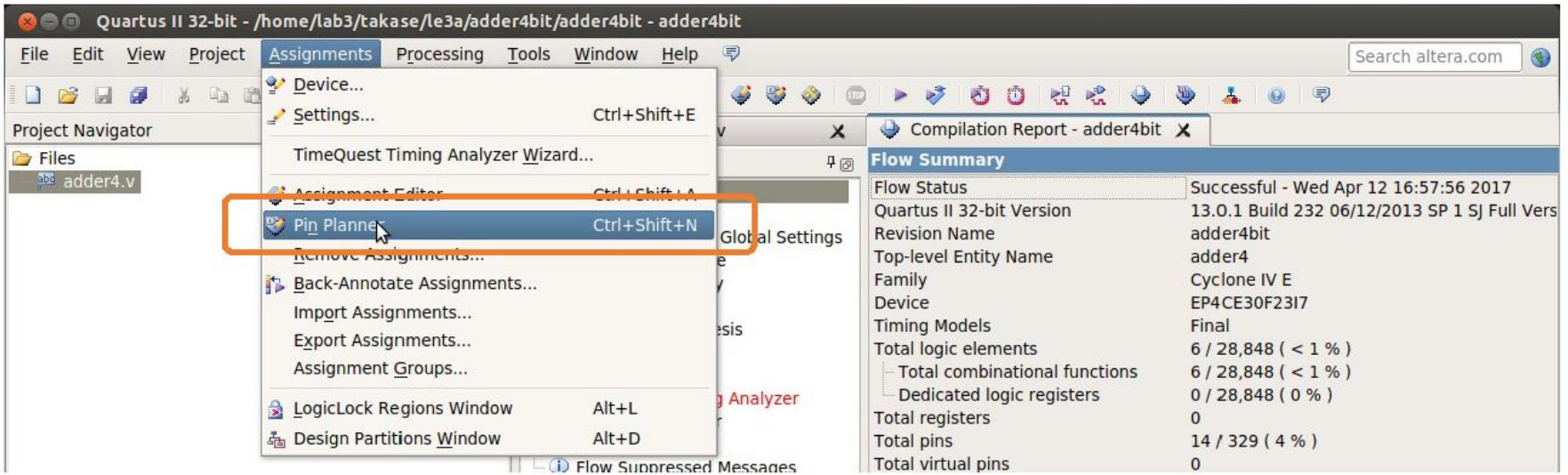
Try out different things.

g. Assignment of I/O Pins

- Assign the signal lines of the circuit to the I/O pins of the FPGA.
  - Depending on the assignment, you can provide input from switches on the board or output signals to LEDs.
  - Refer to the user's manual and pin assignment table to see which module corresponds to which I/O pin.
- For the correspondence between the I/O pins of the FPGA and various types of devices, read the user's manual and pin assignment table carefully.
- In this example, the two input values are DIP SW A and B, the carry-in is the numpad push SW SW4, and the sum and carry-out are LEDs.
- Feel free to change the above pin assignments in any way that makes it easier for you.

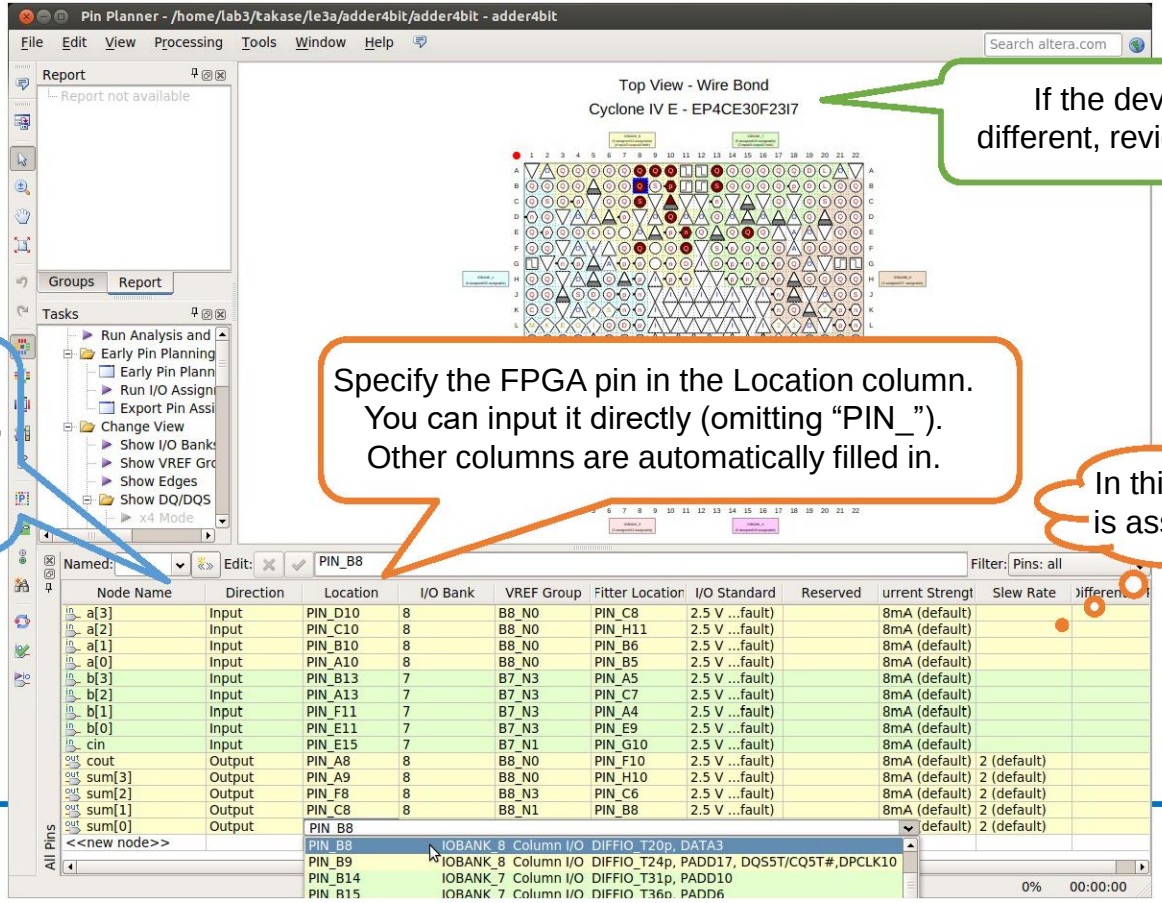
g. Assignment of I/O Pins

- Boot up “Pin Planner”
  - “Assignments” -> “Pin Planner” or Ctrl+Shift+N



g. Assignment of I/O Pins

- Assigning I/O pins
  - Recompile the design if the name of the pin does not appear.
  - If another pin name appears, check if it is top-level.



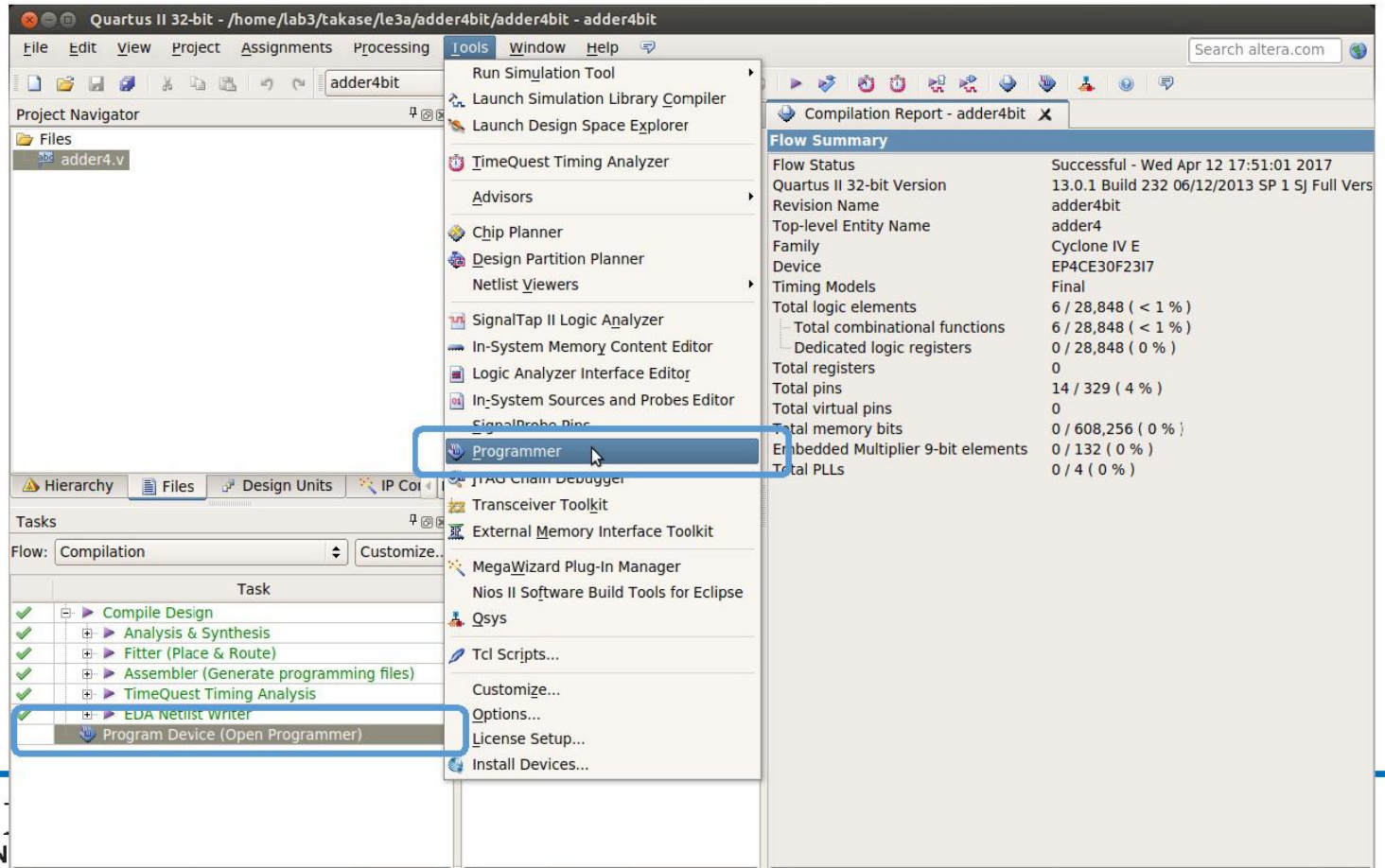
h. Writing the Circuit to the FPGA

- Write the synthesized circuit image to the FPGA and verify the operation on the actual device.
  - The design is not complete until it works on the actual device!
  - If the Tasks window shows a yellow ?, recompile; if it shows a red ×, debug the HDL code, etc. since it is an error.

- Preparing the FPGA board
  - Connect the power supply cable.
  - Connect the USB cable to the PC.
  - Turn on the board (switch on extension cable).
  - If the USB cable is not recognized, please refer to this FAQ.

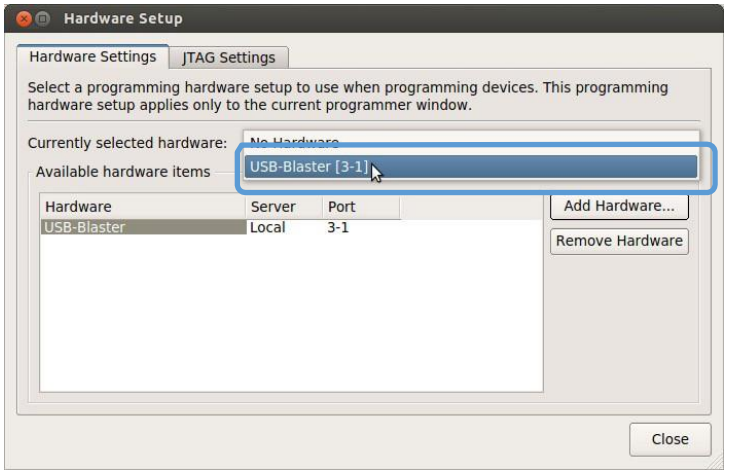
h. Writing the Circuit to the FPGA

- Launching the FPGA writing tool
  - “Tools” -> “Programmer” or “Program Device (Open Programmer)” in the Tasks window



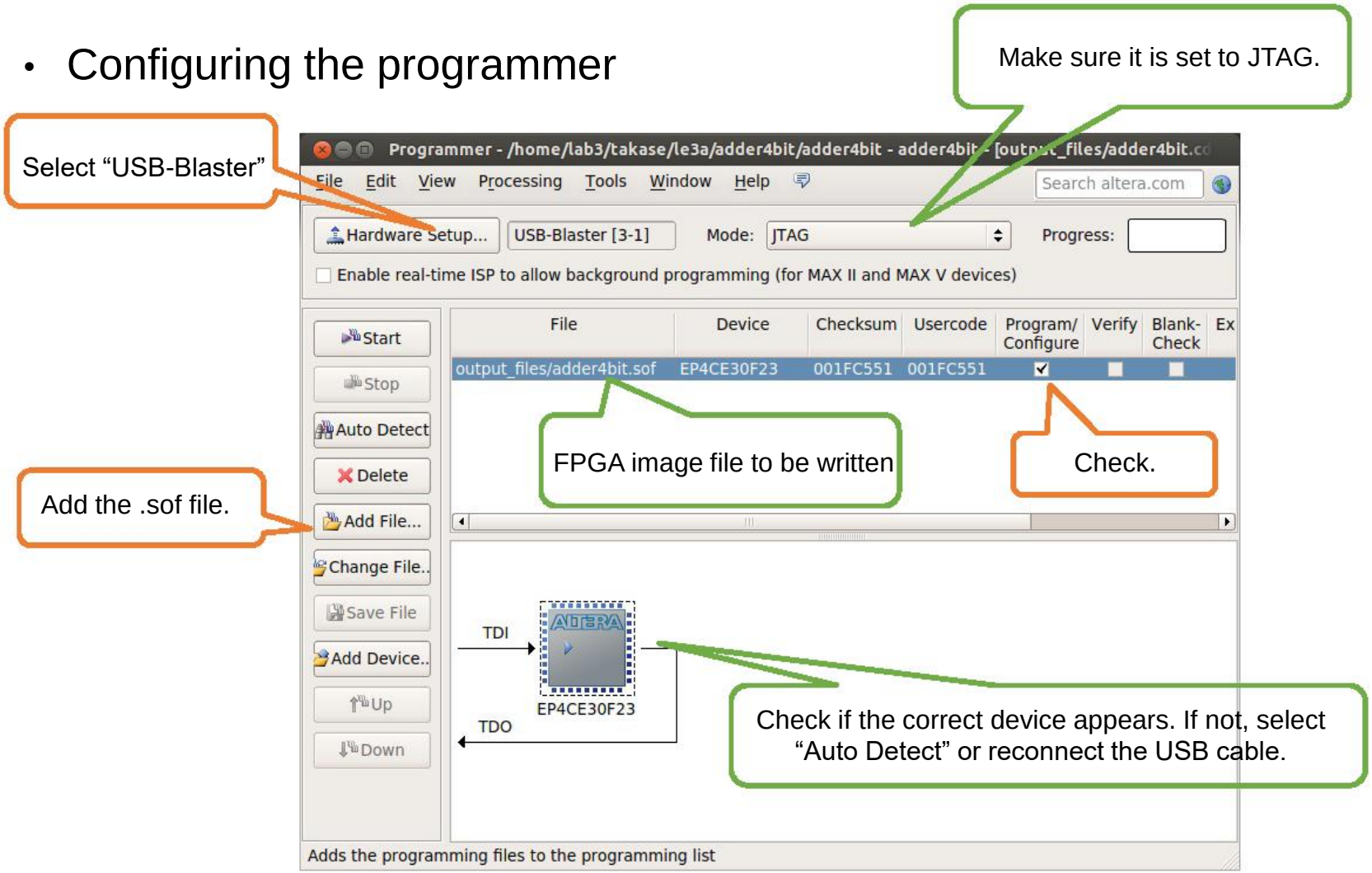
h. Writing the Circuit to the FPGA

- Configuring the programmer
  - Click “Hardware Setup” and select “USB-Blaster [1-1]” under “Currently selected hardware:”
    - ✓ If USB-Blaster does not appear in the Available hardware items, reconnect the PC to USB. If USB-Blaster still doesn't appear, you may have a device driver problem, so ask a TA.
  - Set the Mode to “JTAG.”
  - If the file is not displayed in the list, select the file to be written from the Add File...
    - ✓ ./output_files/<proj_name>.sof
  - Check Program/Configure on the list.



h. Writing the Circuit to the FPGA

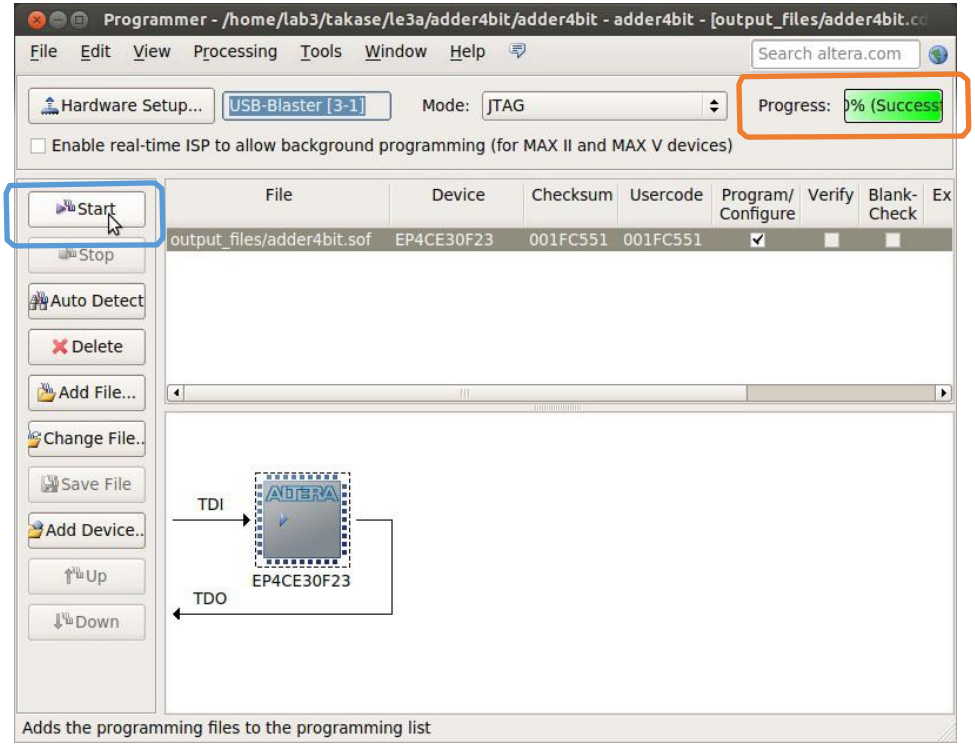
- Configuring the programmer



h. Writing the Circuit to the FPGA

- Writing the circuits to the FPGA
  - Click “Start”
  - When “Process:” shows “100% (Success)” in green, the writing process is completed. achine

- Verifying the behavior on the actual machine
  - If the desired behavior is correct, you are finished!



TIPS

- If the actual machine doesn't work as expected:
  - First of all, make sure that your design is logically correct by using simulations!
  - It's not uncommon for something to not work on the actual device even if it works well in simulation. Many such issues are due to failure to satisfy timing constraints. Please refer to the FAQ.
- If I/O device does not work as expected:
  - Of course, the design must be correct. Then, consider that each I/O device has positive and negative logic.
  - The 7SEG LED also requires a selector signal setting.
  - Unassigned LEDs (pins) lighting up can be ignored.
    - ✓ It is of course great to properly consider these...
  - After all, it is best to read through the instruction manual for details.

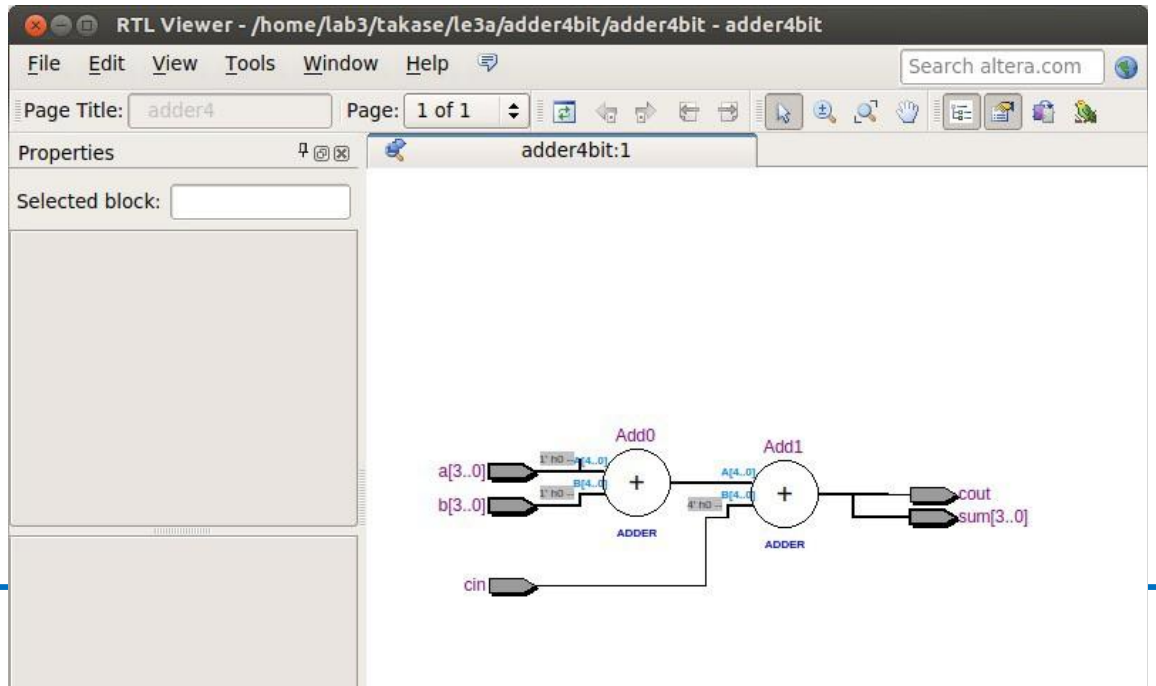
TIPS

- I want to output various information all at once on the actual machine:
  - Use the expansion board MU500-7SEG.
    - ✓ For details on how to use the MU500-7SEG, refer to the instruction manual.
  - Once you acquire a command of how to use the expansion board, you will be able to check the current PC value of the processor on the actual device, which will significantly help with debugging.

- I want to call HDL circuits from the circuit diagram editor:
  - To do this, select the file you wish to use and then select “Files” -> “Create/Update” -> “Create Symbol Files for Current File.”
    - ✓ You can place it as a component instance from the circuit diagram editor.
    - ✓ You can also do the reverse (call the circuit diagram from HDL as a lower-level module).
  - In processor design, it is also good to design the overall architecture and data paths in a circuit diagram editor, while designing the individual components in an HDL.
    - ✓ This will improve visibility of the project as a whole.

TIPS

- I want to know what circuits are synthesized from the HDL:
  - You can use “Tools” -> “Netlist Viewers” -> “RTL Viewer” to somehow visualize the synthesized circuits.
  - This is a useful feature to know what kind of circuit your HDL will be synthesized into and what HW (hardware) design is all about.
    - ✓ The following example is not very interesting at all, but here it is:



TIPS

- Simulation
  - You can directly input and execute commands in the Transcript sub-window of modelsim-ase. Once you get used to it, it is faster than clicking through menus. A command history is also available.
  - Execution history is stored in "./simulation/modelsim/msim_transcript," so you can save it under a different file name and edit it.

    You can execute it from the Transcript window with do (file name).
- About modules
  - If you want to synthesize or simulate only a part of the circuit instead of the whole circuit, right-click on the file in the Project Navigator and select "Set as Top-Level Entity."
- Version: Quartus Prime 17.1/ModelSim ASE 10.5b
  - Note the version of the tool when Googling things.

    There are some tips and solutions that do not work because they are for older/newer versions.