Digital System Design using Altera Quartus II and ModelSim For EECE 353

by Eddie Hung
(with updates by Guy Lemieux and Steve Wilton)
University of British Columbia

January 2014

The purpose of this document is to very briefly describe how to use the Altera Quartus II and ModelSim software packages to design digital systems. It is targeted at students of EECE 353 at the University of British Columbia.

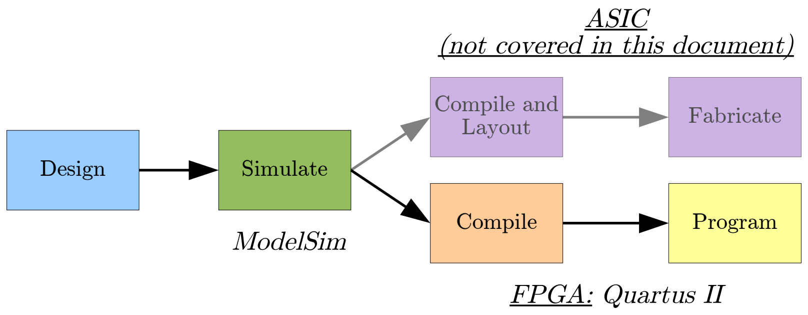


Figure 1: A typical digital design flow

A typical design flow may look like Figure 1. First, the designer will describe their circuit using a Hardware Description Language (HDL) such as VHDL. In this tutorial, we have provided this description for you. Next, the designer will simulate their VHDL to ensure that it functions correctly. We will be using the industry-standard ModelSim software package (produced by Mentor Graphics) to perform this logic simulation stage. It is worth pointing out that the circuit description will likely not be perfect first time, and only through simulation will many errors be discovered. Hence, it may take several iterations between design and simulation before all such errors are ironed out.

The third stage is to compile the circuit, from a text-based description into physical hardware, such as logic gates. In this course, we will be exclusively targeting Field-Programmable Gate-Arrays, or FPGAs, as the implementation platform. For reference, and to highlight its similarity, the flow to build a custom ASIC chip is also shown. We shall be using another industry-standard tool: Altera Quartus II to compile VHDL into a format compatible with the Altera FPGAs that are used in this course. Finally, we will then use Quartus II to program the FPGA chip with this output to actually realize the circuit in hardware on the DE2 board.

Section 1 of this document covers how to install the Quartus II and ModelSim software packages, whilst Section 2 covers how to simulate a design using ModelSim. Compiling the design, and programming it onto an FPGA board using Quartus II is covered in Sections 3 and 4.

This tutorial was originally produced in January 2012 using Quartus II version 11.1sp1 and ModelSim-Altera version 10.0c. The following screenshots may contain some cosmetic differences if you are using any other versions, though the core functionality will remain the same.

Note: The departmental lab computers may have multiple versions of Altera Quartus II installed. Please ask the course instructor which version you should use.

1 Installing Quartus II and ModelSim

In this section, we shall quickly walk through the steps required to install the Quartus II and ModelSim software onto your personal computer. You can choose from either Windows or Linux versions of the software. If you have a Mac, you will need a software product like Boot Camp, VirtualBox, Wine, Parallels, etc. Skip this section if you are using the departmental computers, which will have this software pre-installed.

The web page below describes which Altera tools, and which version, to download and install. `http://help.ece.ubc.ca/Altera` The following instructions can be read in conjunction with that web page.

The easiest way to download the software is at `http://dl.altera.com/?edition=web` From this web page, choose the version 13.0sp1 from the dropdown box (*not* Version 13.1). If you forget to select 13.0sp1 and download Version 13.1 instead, you will not be able to compile your design to the DE2 board, and will have to re-install. Choose the appropriate operating system (Windows for most of you), and the Akamai DLM3 Download Manager.

Then, choose the Combined Files tab, and click Download. You will be asked to create an Altera account; if you prefer not to create one with your own email address, ask the TA in the lab for help. Follow the prompts, and download a .tar file. Extract the files in this tar file, and run the setup.bat file (inside the directory you have just created).

Again, follow the prompts. You will need to select which components you wish to install. You need the Quartus II web edition, the Cyclone II device family files, and Modelsim-Altera Starter.

Note: ModelSim-Altera Starter is free, but ModelSim-Altera is not.

On your own computers, you should be able to find Quartus II and ModelSim under your Start menu. On the departmental computers, you may have to look in a Software folder on the desktop.

If you have problems installing the software, please contact your TA in the first week of Lab 1.

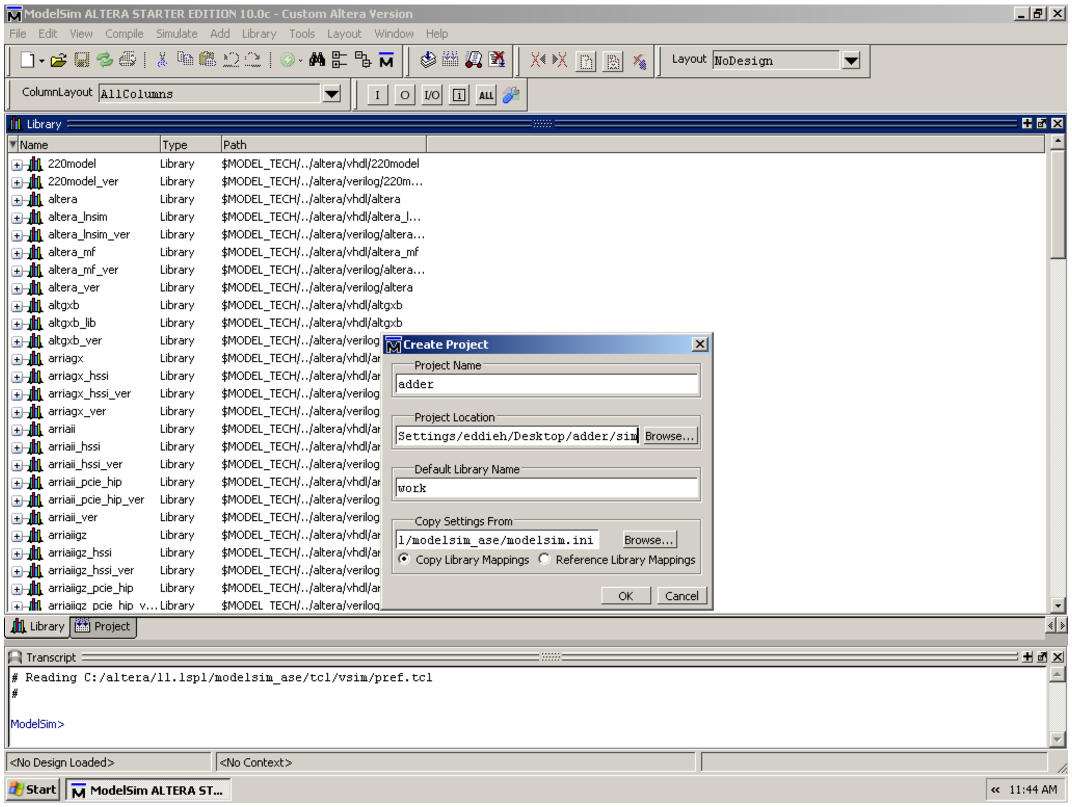
2 Simulation using ModelSim

In this section, we will look at how to simulate a hardware design using ModelSim.

Login to Connect at `http://connect.ubc.ca/` and download the `adder.vhd` and `adder_tb.vhd` files from the Lab 1 folder if you have not already done so.

`adder.vhd` describes a very simple combinational adder. Don't worry about how this circuit is described by the VHDL code right now, all you need to know is that the circuit inputs are two sets of sliding switches on the board, and that the output is a set of red LEDs. Switches 0 to 3 is used to describe the 4-bit binary value that is added to switches 4 to 7, and the 4-bit binary output will be shown on the red LEDs 0 to 3.

`adder_tb.vhd` describes the "test-bench" for this design. As its name implies, the job of this file is to thoroughly test the adder to make sure that it does work as expected — before it is inserted as a component inside a microprocessor, for example. This is a hugely important part of hardware design — of course, designs can still be tested after they are built, but with modern circuits costing millions of dollars to manufacture, it is vitally important to make sure that they are as correct as possible. Test-benches like this one will virtually stimulate the inputs of any design-under-test in order to fully exercise its behaviour — in our adder example, it would apply different binary numbers onto the input switches. Usually, it would then be up to the designer to interpret the outputs, which are shown as waveforms, in order to verify that their circuit is performing correctly. In some more advanced test-benches, it is possible to automate this checking procedure.


ModelSim ALTERA STARTER EDITION 10.0c - Custom Altera Version
File Edit View Compile Simulate Add Library Tools Layout Window Help
Layout NoDesign
ColumnLayout AllColumns
Library
Library Project
Transcript
Start ModelSim ALTERA ST...
11:44 AM

Name	Type	Path
220model	Library	$MODEL_TECH/../altera/vhdl/220model
220model_ver	Library	$MODEL_TECH/../altera/verilog/220m...
altera	Library	$MODEL_TECH/../altera/vhdl/altera
altera_lnsim	Library	$MODEL_TECH/../altera/vhdl/altera_l...
altera_lnsim_ver	Library	$MODEL_TECH/../altera/verilog/altera...
altera_mf	Library	$MODEL_TECH/../altera/vhdl/altera_mf
altera_mf_ver	Library	$MODEL_TECH/../altera/verilog/altera...
altera_ver	Library	$MODEL_TECH/../altera/verilog/altera
altgxb	Library	$MODEL_TECH/../altera/vhdl/altgxb
altgxb_lib	Library	$MODEL_TECH/../altera/vhdl/altgxb
altgxb_ver	Library	$MODEL_TECH/../altera/verilog
arriagx	Library	$MODEL_TECH/../altera/vhdl/ar
arriagx_hssi	Library	$MODEL_TECH/../altera/vhdl/ar
arriagx_hssi_ver	Library	$MODEL_TECH/../altera/verilog
arriagx_ver	Library	$MODEL_TECH/../altera/verilog
arriaii	Library	$MODEL_TECH/../altera/vhdl/ar
arriaii_hssi	Library	$MODEL_TECH/../altera/vhdl/ar
arriaii_hssi_ver	Library	$MODEL_TECH/../altera/verilog
arriaii_pcie_hip	Library	$MODEL_TECH/../altera/vhdl/ar
arriaii_pcie_hip_ver	Library	$MODEL_TECH/../altera/verilog
arriaii_ver	Library	$MODEL_TECH/../altera/verilog
arriaiigz	Library	$MODEL_TECH/../altera/vhdl/ar
arriaiigz_hssi	Library	$MODEL_TECH/../altera/vhdl/ar
arriaiigz_hssi_ver	Library	$MODEL_TECH/../altera/verilog
arriaiigz_pcie_hip	Library	$MODEL_TECH/../altera/vhdl/ar
arriaiigz_pcie_hip_v...	Library	$MODEL_TECH/../altera/verilog

Create Project

Project Name: adder

Project Location: Settings/eddieh/Desktop/adder/sim

Default Library Name: work

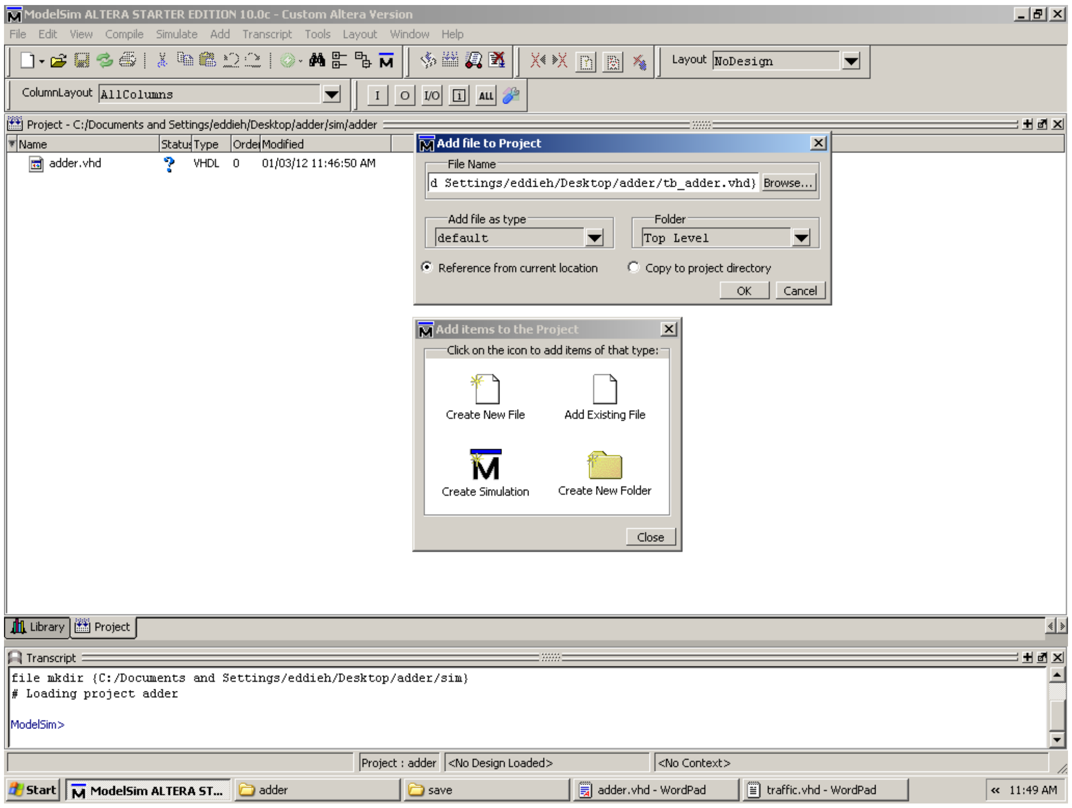
Copy Settings From: l/modelsim_ase/modelsim.ini

[✓ Copy Library Mappings] Reference Library Mappings

```
# Reading C:/altera/11.1spl/modelsim_ase/tcl/vsim/pref.tcl
#

ModelSim>
```

<No Design Loaded> <No Context>

Figure 2: Creating a new project in ModelSim


ModelSim ALTERA STARTER EDITION 10.0c - Custom Altera Version
File Edit View Compile Simulate Add Transcript Tools Layout Window Help
Add items to the Project
Click on the icon to add items of that type:
Create New File
Add Existing File
Create Simulation
Create New Folder
Close
Browse...
OK
Cancel
Library
Project
Start

Layout: NoDesign
ColumnLayout: AllColumns

Project - C:/Documents and Settings/eddieh/Desktop/adder/sim/adder

Name	Status	Type	Order	Modified
adder.vhd	?	VHDL	0	01/03/12 11:46:50 AM

Add file to Project
File Name: d Settings/eddieh/Desktop/adder/tb_adder.vhd}
Add file as type: default
Folder: Top Level
(●) Reference from current location ( ) Copy to project directory

Transcript
```
file mkdir {C:/Documents and Settings/eddieh/Desktop/adder/sim}
# Loading project adder

ModelSim>
```

Project : adder | <No Design Loaded> | <No Context>

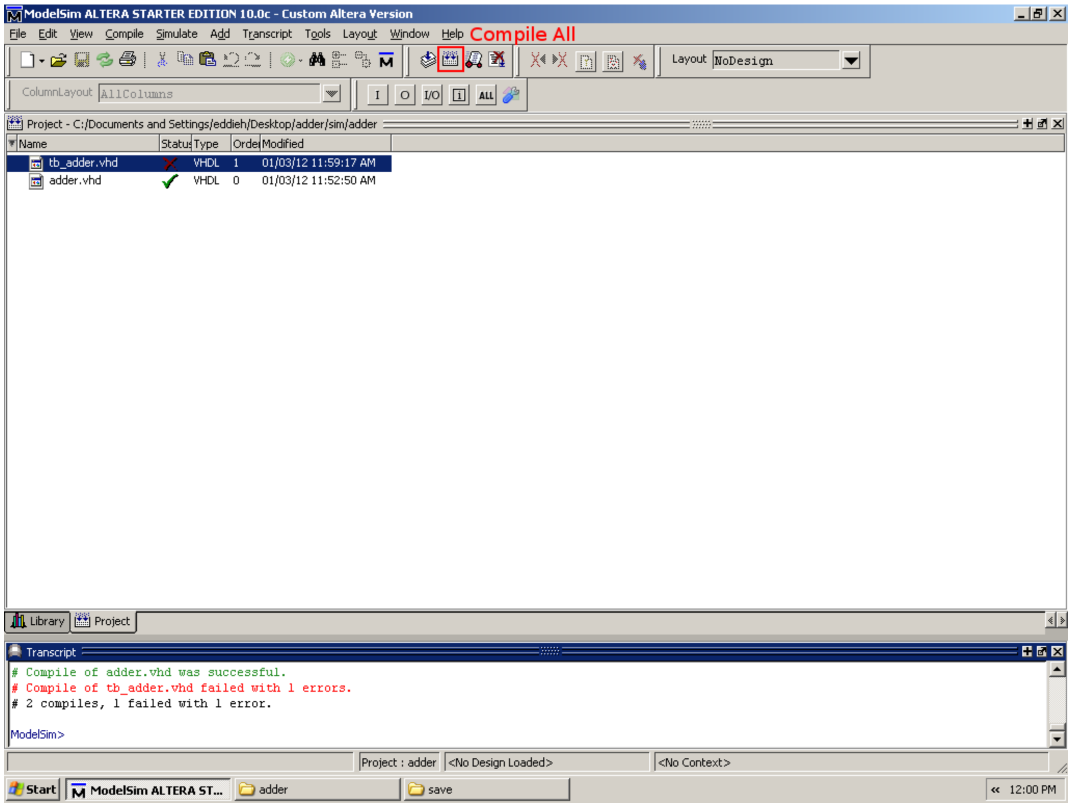
ModelSim ALTERA ST... | adder | save | adder.vhd - WordPad | traffic.vhd - WordPad | 11:49 AM

Figure 3: Adding files to a ModelSim project

First, start up ModelSim and select File — New — Project. Select a Project Name and Location, but leave the other settings as-is, like in Figure 2. Next, select “Add Existing File” and select both `.vhd` files that you previously downloaded as in Figure 3.

Select Compile — Compile All to compile all the source files in your project. Alternatively, click on the button highlighted in Figure 4. You should now see that one of the source files, `adder.vhd` will have compiled correctly (as indicated by a green tick in its status column, and a green “successful” message in the transcript window at the bottom). However, `adder_tb.vhd` was not so successful, with one error[s].

Double-click on the failed message in the transcript, and then double-click again on the first error message in the subsequent window. This will open the code up to the right of your Project window and helpfully take you to the vicinity of the error, as shown in Figure 5. The error message here is complaining about a semicolon, and by looking at the source code it should be clear that the previous line, numbered 23, is missing one. Add a semicolon back to the end of the line, after the last double-quote, and compile again. Both files should now compile successfully.

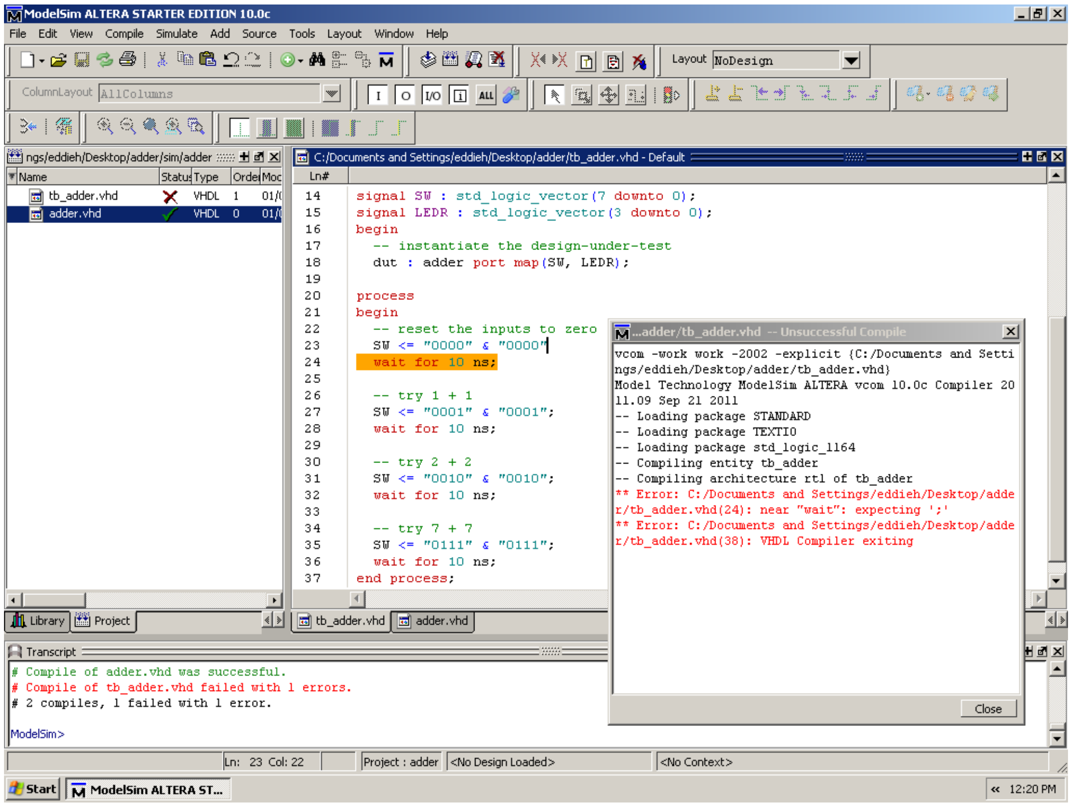

ModelSim ALTERA STARTER EDITION 10.0c - Custom Altera Version
File Edit View Compile Simulate Add Transcript Tools Layout Window Help
Compile All
Layout NoDesign
ColumnLayout AllColumns
Project - C:/Documents and Settings/eddieh/Desktop/adder/sim/adder
Library Project
Transcript
Start ModelSim ALTERA ST... adder save 12:00 PM

Name	Status	Type	Order	Modified
tb_adder.vhd	✗	VHDL	1	01/03/12 11:59:17 AM
adder.vhd	✓	VHDL	0	01/03/12 11:52:50 AM

```
# Compile of adder.vhd was successful.
# Compile of tb_adder.vhd failed with 1 errors.
# 2 compiles, 1 failed with 1 error.

ModelSim>
```

Project : adder <No Design Loaded> <No Context>

Figure 4: Compile All in ModelSim


ModelSim ALTERA STARTER EDITION 10.0c
File Edit View Compile Simulate Add Source Tools Layout Window Help
Layout NoDesign
ColumnLayout AllColumns
ngs/eddieh/Desktop/adder/sim/adder
C:/Documents and Settings/eddieh/Desktop/adder/tb_adder.vhd - Default
Library Project
tb_adder.vhd adder.vhd
...adder/tb_adder.vhd -- Unsuccessful Compile
Close
Transcript
Start ModelSim ALTERA ST... 12:20 PM

Name	Status	Type	Order	Mod
tb_adder.vhd	✗	VHDL	1	01/(
adder.vhd	✓	VHDL	0	01/(

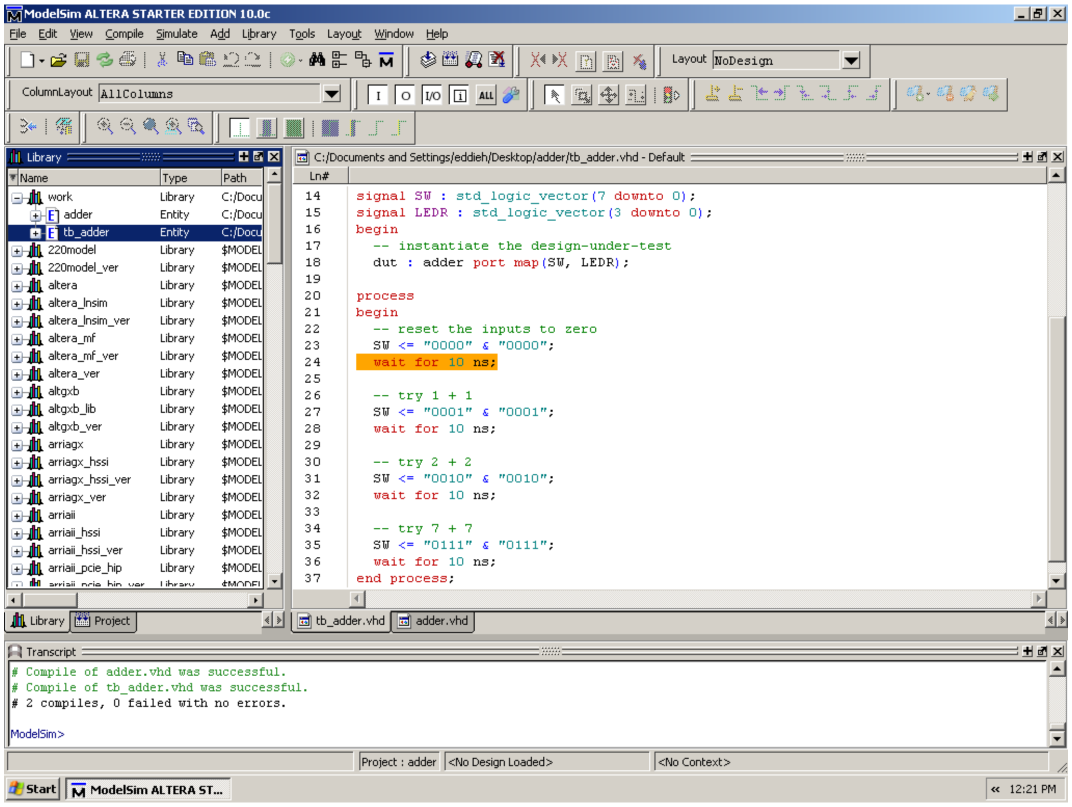
```
Ln#
14  signal SW : std_logic_vector(7 downto 0);
15  signal LEDR : std_logic_vector(3 downto 0);
16  begin
17    -- instantiate the design-under-test
18    dut : adder port map(SW, LEDR);
19
20  process
21  begin
22    -- reset the inputs to zero
23    SW <= "0000" & "0000"
24    wait for 10 ns;
25
26    -- try 1 + 1
27    SW <= "0001" & "0001";
28    wait for 10 ns;
29
30    -- try 2 + 2
31    SW <= "0010" & "0010";
32    wait for 10 ns;
33
34    -- try 7 + 7
35    SW <= "0111" & "0111";
36    wait for 10 ns;
37  end process;
```

```
vcom -work work -2002 -explicit {C:/Documents and Setti
ngs/eddieh/Desktop/adder/tb_adder.vhd}
Model Technology ModelSim ALTERA vcom 10.0c Compiler 20
11.09 Sep 21 2011
-- Loading package STANDARD
-- Loading package TEXTIO
-- Loading package std_logic_1164
-- Compiling entity tb_adder
-- Compiling architecture rtl of tb_adder
** Error: C:/Documents and Settings/eddieh/Desktop/adde
r/tb_adder.vhd(24): near "wait": expecting ';'
** Error: C:/Documents and Settings/eddieh/Desktop/adde
r/tb_adder.vhd(38): VHDL Compiler exiting
```

```
# Compile of adder.vhd was successful.
# Compile of tb_adder.vhd failed with 1 errors.
# 2 compiles, 1 failed with 1 error.

ModelSim>
```

Ln: 23 Col: 22 Project : adder <No Design Loaded> <No Context>

Figure 5: Locating errors in ModelSim


ModelSim ALTERA STARTER EDITION 10.0c
File Edit View Compile Simulate Add Library Tools Layout Window Help
Layout NoDesign
ColumnLayout AllColumns
Library
C:/Documents and Settings/eddieh/Desktop/adder/tb_adder.vhd - Default
Library Project
tb_adder.vhd adder.vhd
Transcript
Start ModelSim ALTERA ST...
12:21 PM

Name	Type	Path
work	Library	C:/Docu
adder	Entity	C:/Docu
tb_adder	Entity	C:/Docu
220model	Library	$MODEL
220model_ver	Library	$MODEL
altera	Library	$MODEL
altera_lnsim	Library	$MODEL
altera_lnsim_ver	Library	$MODEL
altera_mf	Library	$MODEL
altera_mf_ver	Library	$MODEL
altera_ver	Library	$MODEL
altgxb	Library	$MODEL
altgxb_lib	Library	$MODEL
altgxb_ver	Library	$MODEL
arriagx	Library	$MODEL
arriagx_hssi	Library	$MODEL
arriagx_hssi_ver	Library	$MODEL
arriagx_ver	Library	$MODEL
arriaii	Library	$MODEL
arriaii_hssi	Library	$MODEL
arriaii_hssi_ver	Library	$MODEL
arriaii_pcie_hip	Library	$MODEL

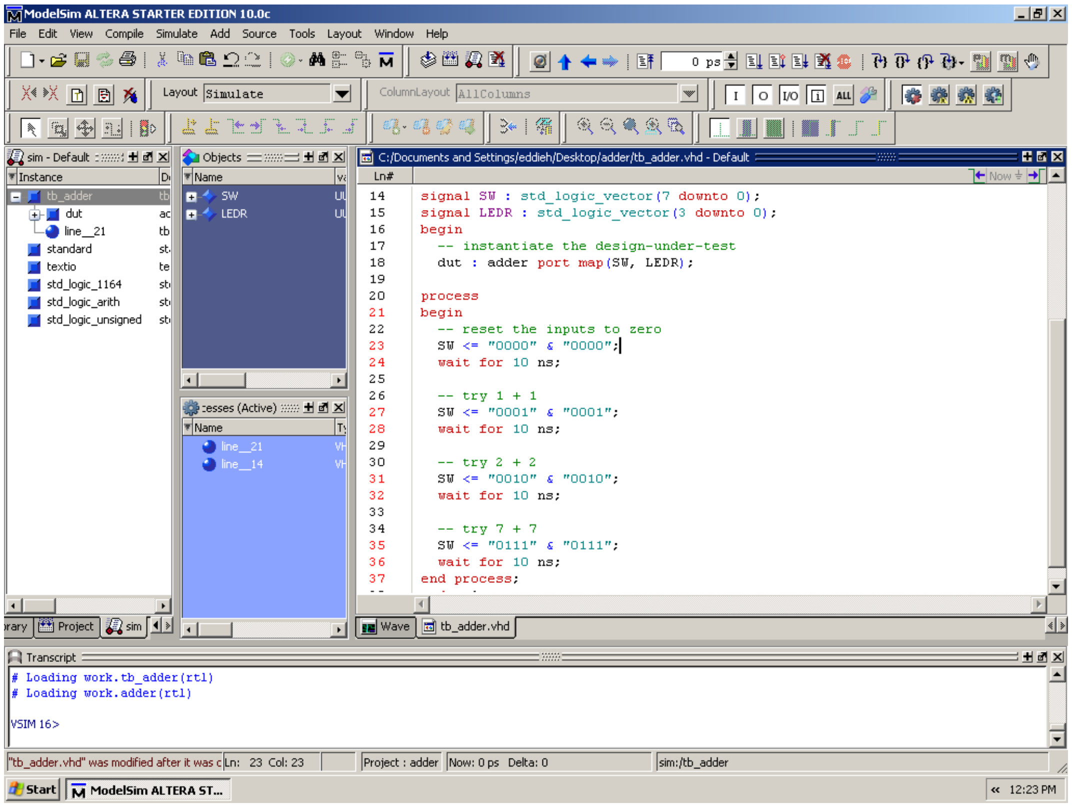
```
signal SW : std_logic_vector(7 downto 0);
signal LEDR : std_logic_vector(3 downto 0);
begin
  -- instantiate the design-under-test
  dut : adder port map(SW, LEDR);

process
begin
  -- reset the inputs to zero
  SW <= "0000" & "0000";
  wait for 10 ns;

  -- try 1 + 1
  SW <= "0001" & "0001";
  wait for 10 ns;

  -- try 2 + 2
  SW <= "0010" & "0010";
  wait for 10 ns;

  -- try 7 + 7
  SW <= "0111" & "0111";
  wait for 10 ns;
end process;
```

```
# Compile of adder.vhd was successful.
# Compile of tb_adder.vhd was successful.
# 2 compiles, 0 failed with no errors.

ModelSim>
```

```
Project : adder   <No Design Loaded>   <No Context>
```

Figure 6: Starting a simulation in ModelSim

Now comes the fun part, actually simulating your design. Select the Library tab of your Project window, open up the work library, and double-click on "adder_tb" as shown in Figure 6, after which you should see something that resembles Figure 7. Select the signals that you want to see during simulation from the (dark blue) "Objects" window by right-clicking inside and choosing Add — To Wave. For now, select all signals by choosing "Signals in Region", which should bring you to Figure 8.

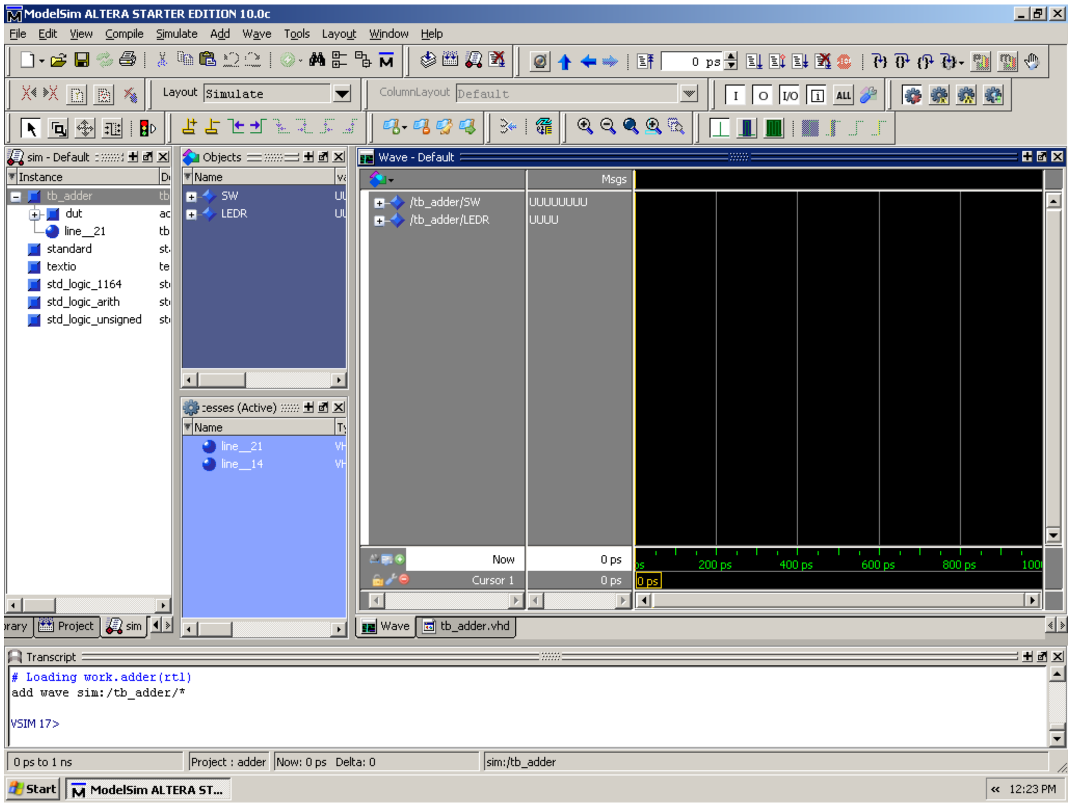


```
signal SW : std_logic_vector(7 downto 0);
signal LEDR : std_logic_vector(3 downto 0);
begin
  -- instantiate the design-under-test
  dut : adder port map(SW, LEDR);

process
begin
  -- reset the inputs to zero
  SW <= "0000" & "0000";
  wait for 10 ns;

  -- try 1 + 1
  SW <= "0001" & "0001";
  wait for 10 ns;

  -- try 2 + 2
  SW <= "0010" & "0010";
  wait for 10 ns;

  -- try 7 + 7
  SW <= "0111" & "0111";
  wait for 10 ns;
end process;
```

```
# Loading work.tb_adder(rtl)
# Loading work.adder(rtl)

VSIM 16>
```

```
"tb_adder.vhd" was modified after it was c Ln: 23 Col: 23   Project : adder   Now: 0 ps  Delta: 0   sim:/tb_adder
```

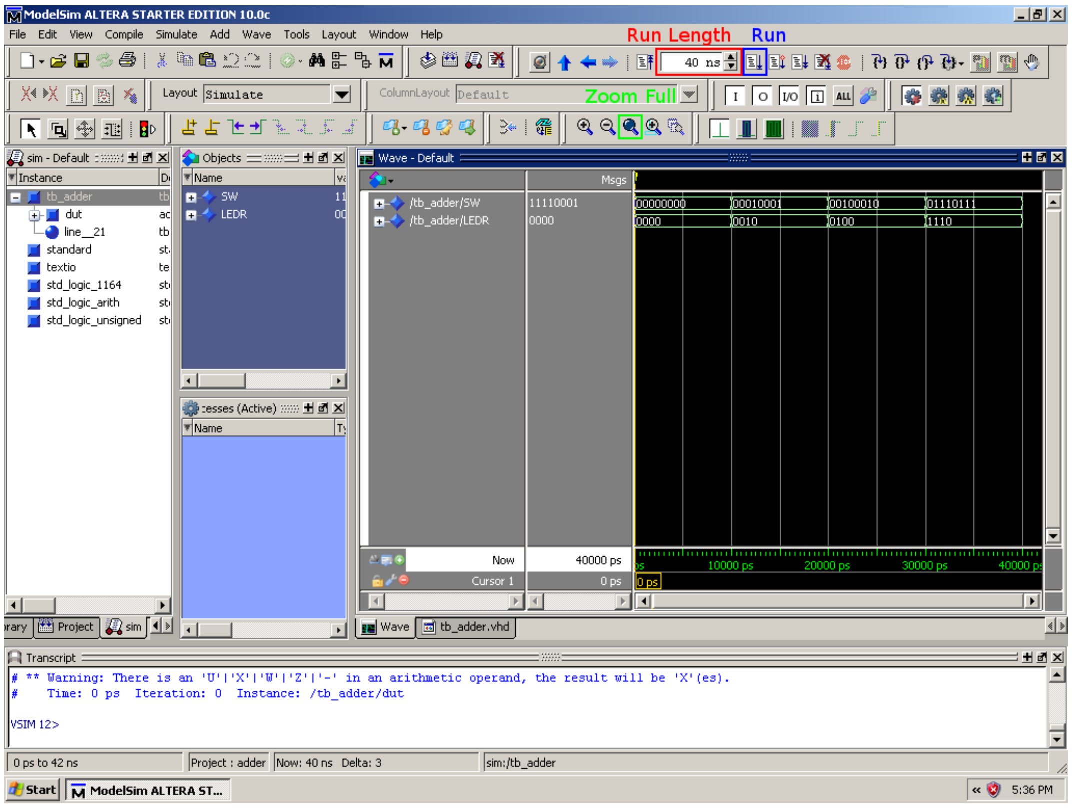
Figure 7: Selecting the signals for observation in ModelSim



```
# Loading work.adder(rtl)
add wave sim:/tb_adder/*

VSIM 17>
```

```
0 ps to 1 ns   Project : adder   Now: 0 ps  Delta: 0   sim:/tb_adder
```

Figure 8: Waveform viewer in ModelSim



```
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
#    Time: 0 ps  Iteration: 0  Instance: /tb_adder/dut

VSIM 12>
```

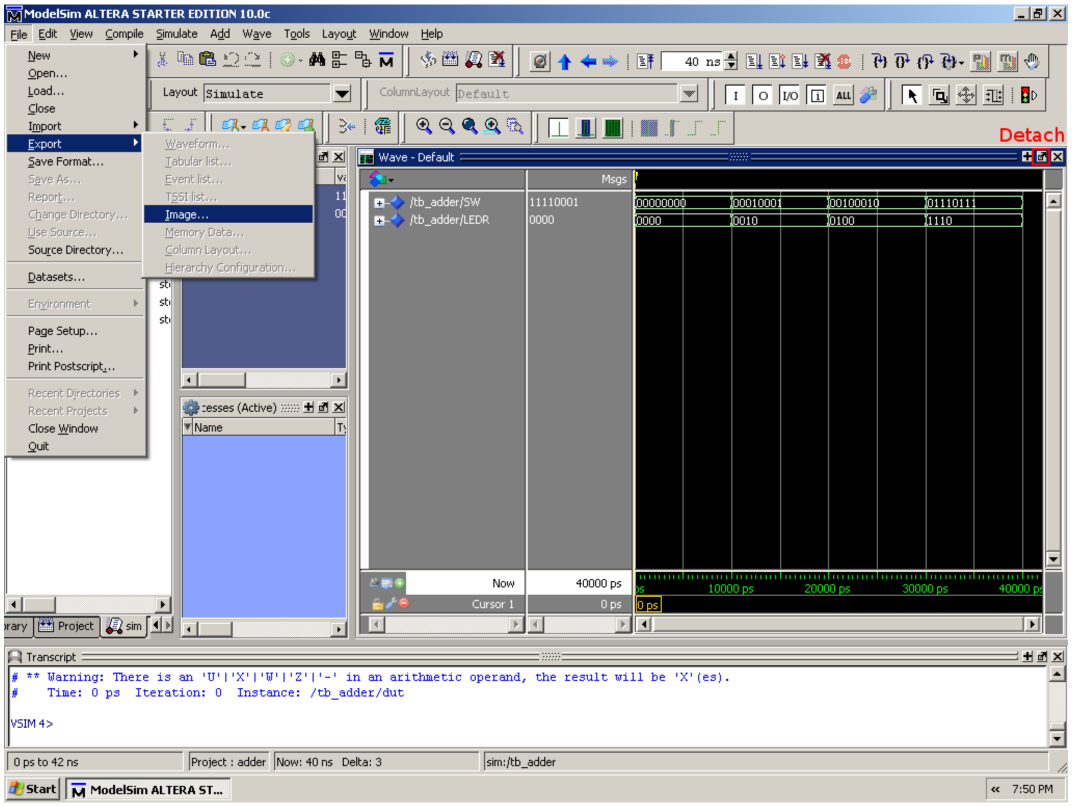
Figure 9: Running a simulation in ModelSim

Change the Run Length to "40 ns", and then click the "Run" button to its right. Now select the "Wave" window by clicking anywhere in the waveform viewer, and then click on the "Zoom Full" button to view the whole waveform at once, as in Figure 9. Study the waveforms carefully — initially, the switch inputs (SW) are all zero, and hence "0000" + "0000" gives a zero output on the LEDR. Check that the other test cases are also correct. Run the simulation again for another 40 ns, and "Zoom Full" again. What do you see?

Another way to run your simulation is by using the command-line interface. You may have noticed that clicking on the "Run" button will result in a `run` command issued in the transcript window. Try it for yourself by entering `run 100ns` onto the command line.

A very useful feature in ModelSim is to be able to restart a simulation, whilst keeping the signals selected in the waveform viewer intact. The restart button is located to the left of the "Run Length" text box – try it to make sure that it works. In the future, when you come to write your own VHDL, you will find that you can edit your VHDL code, re-compile it, and be able to restart the simulation using this button without having to go through the whole setup procedure that we just went through. Similarly, this can also be achieved using the `restart` command.

Many, many other commands exist for use in this ModelSim Tcl (pronounced "tickle") interface: almost everything you can do using the graphical interface can be done with the command line, which will even allow you to automate sequences of tasks. In time, you'll come to learn more commands and embrace some as your favourites.



```
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
#    Time: 0 ps  Iteration: 0  Instance: /tb_adder/dut

VSIM 4>
```

Figure 10: Saving the simulation waveform in ModelSim

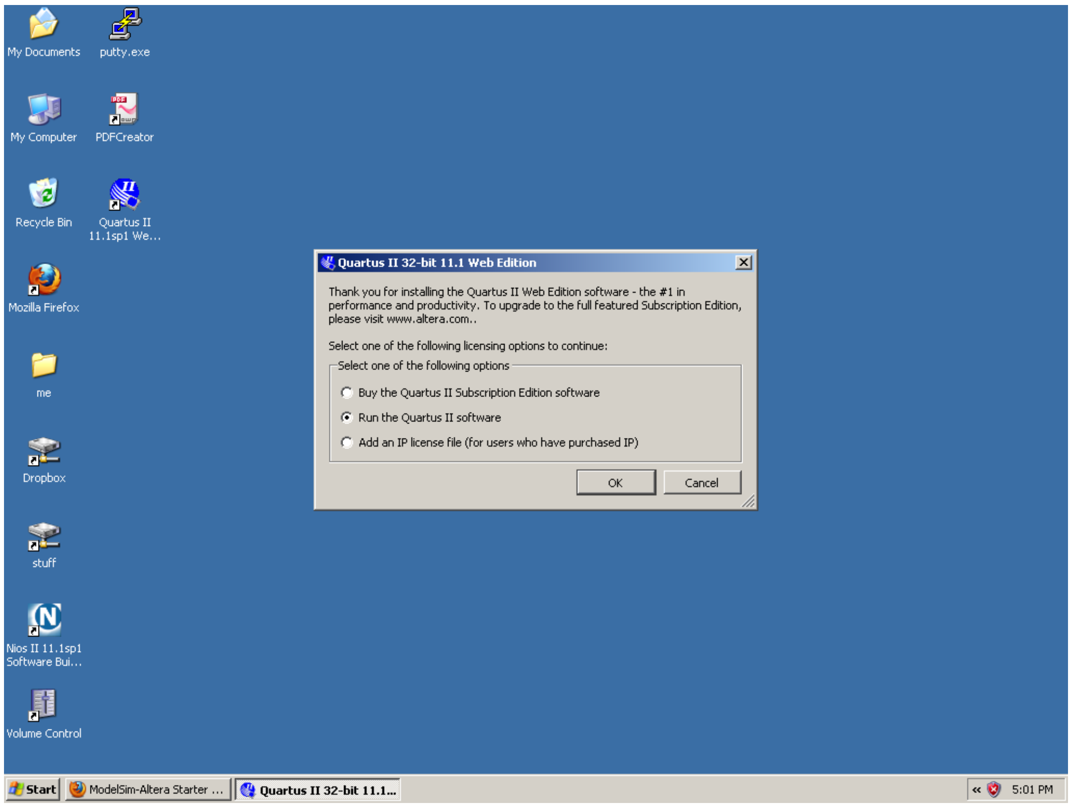
To save the simulated waveforms, make sure you're happy with what is currently displayed in the "Wave" window (all signals names are clearly displayed, and all values are clear — expand signals and adjust the zoom if necessary; you may also choose to detach the Wave window by clicking on the button left of the 'X' close button, and feel free to use multiple images to show your results). Select the window by clicking anywhere inside, and then choose File — Export — Image, as shown in Figure 10.

3 Compilation using Quartus II

Now that you have fully simulated your design and are (fairly) confident that it works, it is now time to compile it into real hardware to be implemented onto an FPGA.

Start up the Quartus II software. If you are prompted with the dialog in Figure 11, select "Run the Quartus II software". Once inside Quartus II, choose "Create a New Project". Click "Next" to advance on to the second page, and you should end up at Figure 12. Feel free to choose any working directory, but **it is crucially important that you enter the top-level design entity name as shown**. This top-level design entity represents the topmost entity or module (i.e. the one which contains all other entities) that is to be implemented in hardware, and its name must match the entity name inside its VHDL file. In this document, this entity is "adder".

In the following menu, add *only* the `adder.vhd` file, which contains our circuit description. The test-bench is not necessary for hardware implementation because we will be able to physically change the inputs of the actual circuit (recall, a test-bench must be used to *virtually* stimulate a circuit description inside ModelSim). Furthermore, the VHDL used in test-benches is *unsynthesizable*. Don't worry if you don't understand what this term means right now, all will become clear in time (but only if you go to the lectures). Make sure you click the "Add" button after browsing for the file, otherwise it will not be added. You should now see something resembling Figure 13.


Quartus II 32-bit 11.1 Web Edition

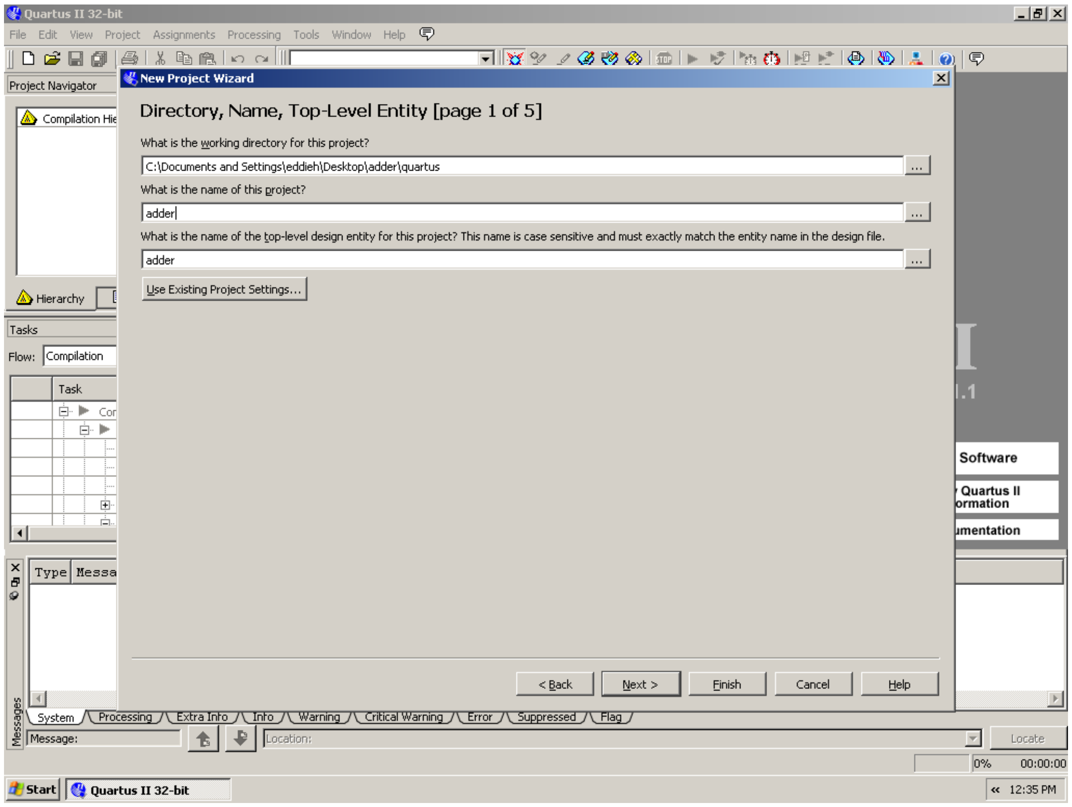
Thank you for installing the Quartus II Web Edition software - the #1 in performance and productivity. To upgrade to the full featured Subscription Edition, please visit www.altera.com..

Select one of the following licensing options to continue:

Select one of the following options

- Buy the Quartus II Subscription Edition software
- [✓ Run the Quartus II software]
- Add an IP license file (for users who have purchased IP)

Figure 11: License dialog for Quartus II



Directory, Name, Top-Level Entity [page 1 of 5]

What is the working directory for this project?

C:\Documents and Settings\eddieh\Desktop\adder\quartus

What is the name of this project?

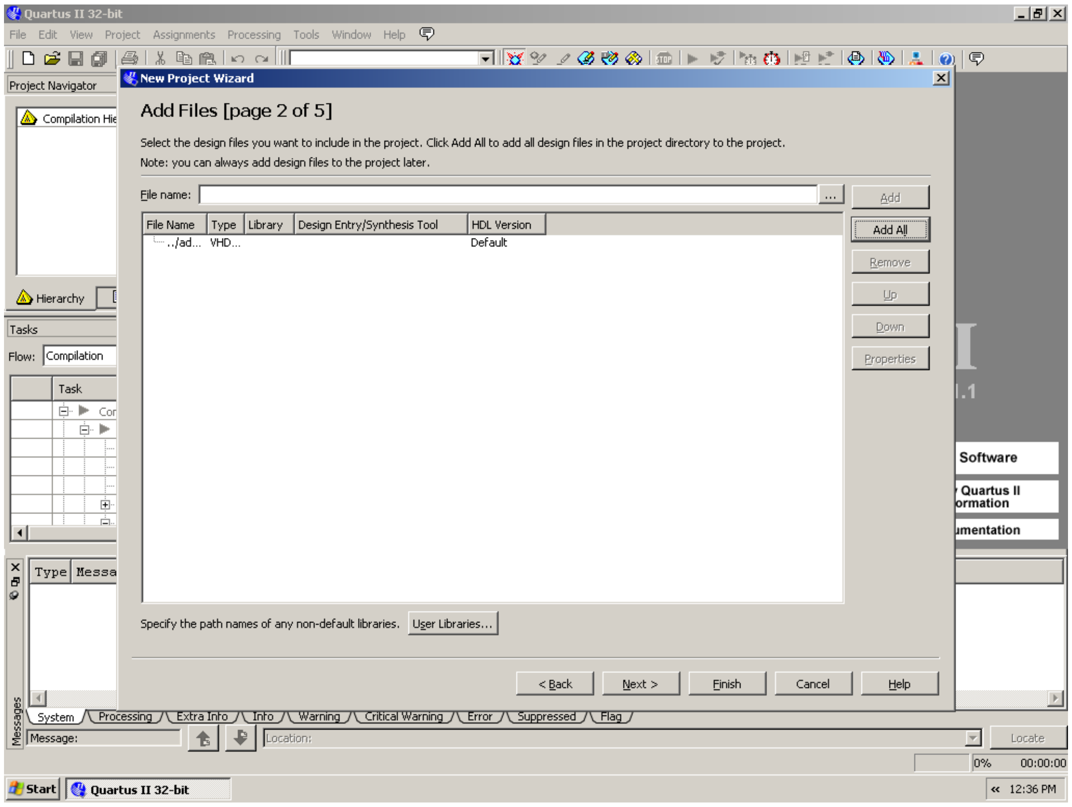
adder

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

adder

Use Existing Project Settings...

Figure 12: Starting a new project in Quartus II



Add Files [page 2 of 5]

Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.

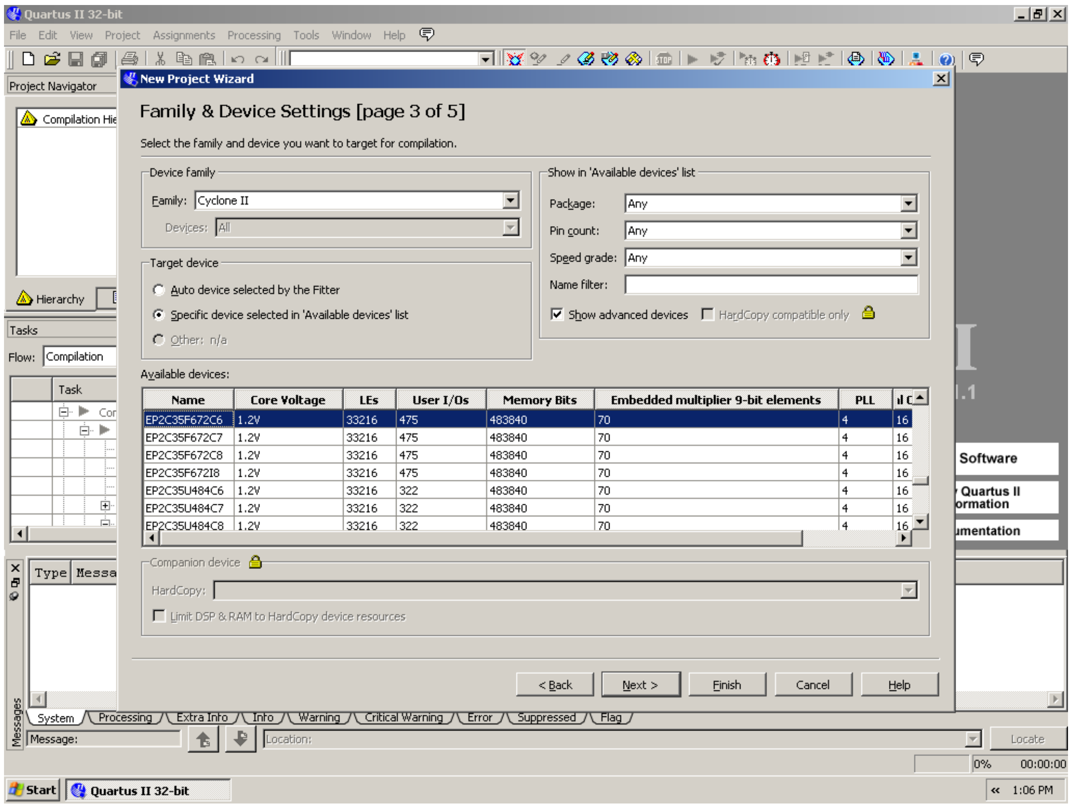
Note: you can always add design files to the project later.

File name:

File Name	Type	Library	Design Entry/Synthesis Tool	HDL Version
../ad...	VHD...			Default

Specify the path names of any non-default libraries. User Libraries...

Figure 13: Adding source files to a Quartus II project



New Project Wizard

Family & Device Settings [page 3 of 5]

Select the family and device you want to target for compilation.

Device family
Family: Cyclone II
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:
[✓] Show advanced devices [ ] HardCopy compatible only

Available devices:

Name	Core Voltage	LEs	User I/Os	Memory Bits	Embedded multiplier 9-bit elements	PLL	ıl C
EP2C35F672C6	1.2V	33216	475	483840	70	4	16
EP2C35F672C7	1.2V	33216	475	483840	70	4	16
EP2C35F672C8	1.2V	33216	475	483840	70	4	16
EP2C35F672I8	1.2V	33216	475	483840	70	4	16
EP2C35U484C6	1.2V	33216	322	483840	70	4	16
EP2C35U484C7	1.2V	33216	322	483840	70	4	16
EP2C35U484C8	1.2V	33216	322	483840	70	4	16

Companion device
HardCopy:
[ ] Limit DSP & RAM to HardCopy device resources

Figure 14: Selecting the FPGA device in Quartus II

The subsequent menu will ask you to select the FPGA device that the project is compiling the hardware for. Different FPGAs have different ways of implementing circuit logic, and different numbers of pins so make sure you select the right model:

Family: Cyclone II

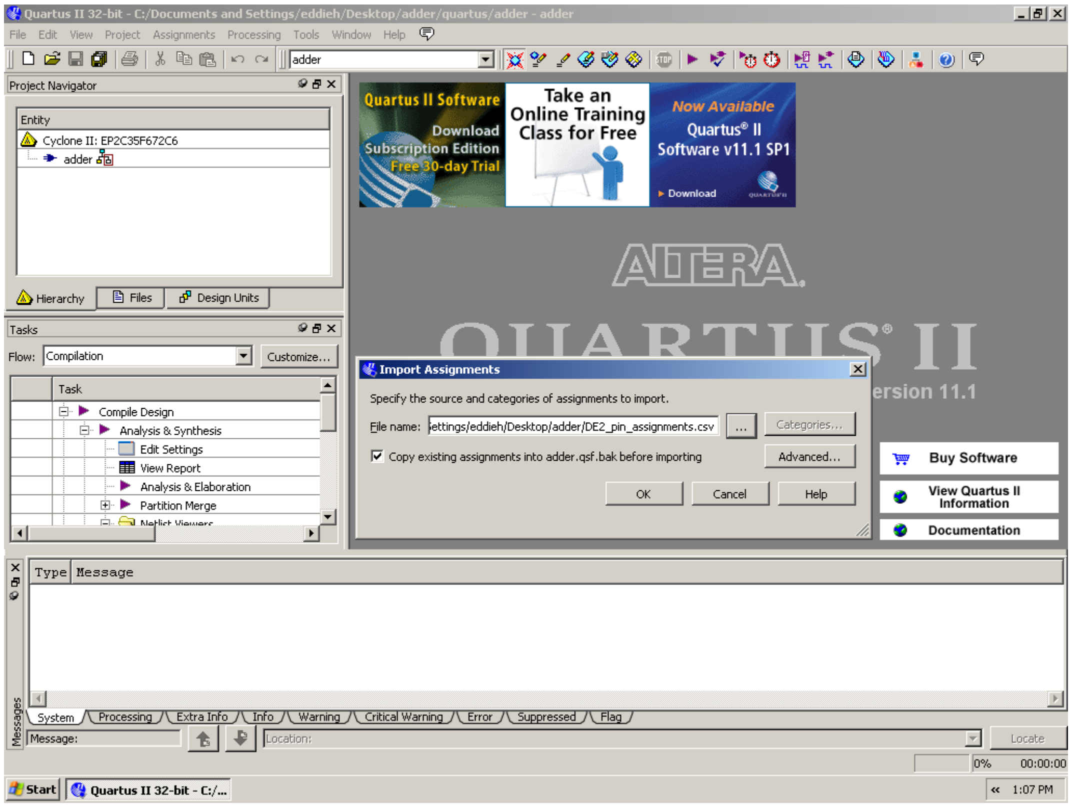
Device: EP2C35F672C6

If you're ever stuck for this number during the lab, you can find this model number is printed on the actual FPGA chip itself – look at the largeset chip on your DE2 board. Make sure your screen looks like Figure 14 before clicking "Finish".

Next, you'll need to tell Quartus II exactly which I/O pins of the FPGA are connected to which signals in the adder. Recall that we have used the "SW" signal as the input for example, and "LEDR" as the output. We need to tell Quartus exactly which pins those signals map to on the DE2 PCB.

This is a critically important step. Forgoing this task, or entering an incorrect assignments file, can damage the FPGA!

Download the `DE2_pin_assignments.csv` file from the Connect website at: `http://connect.ubc.ca`. In Quartus II, select Assignments — Import Assignments, and select the downloaded file as shown in Figure 15. Click OK. Ensure that a "Import Completed. 425 assignments were written (out of 425 read)" message is visible in the output window, as in Figure 16.



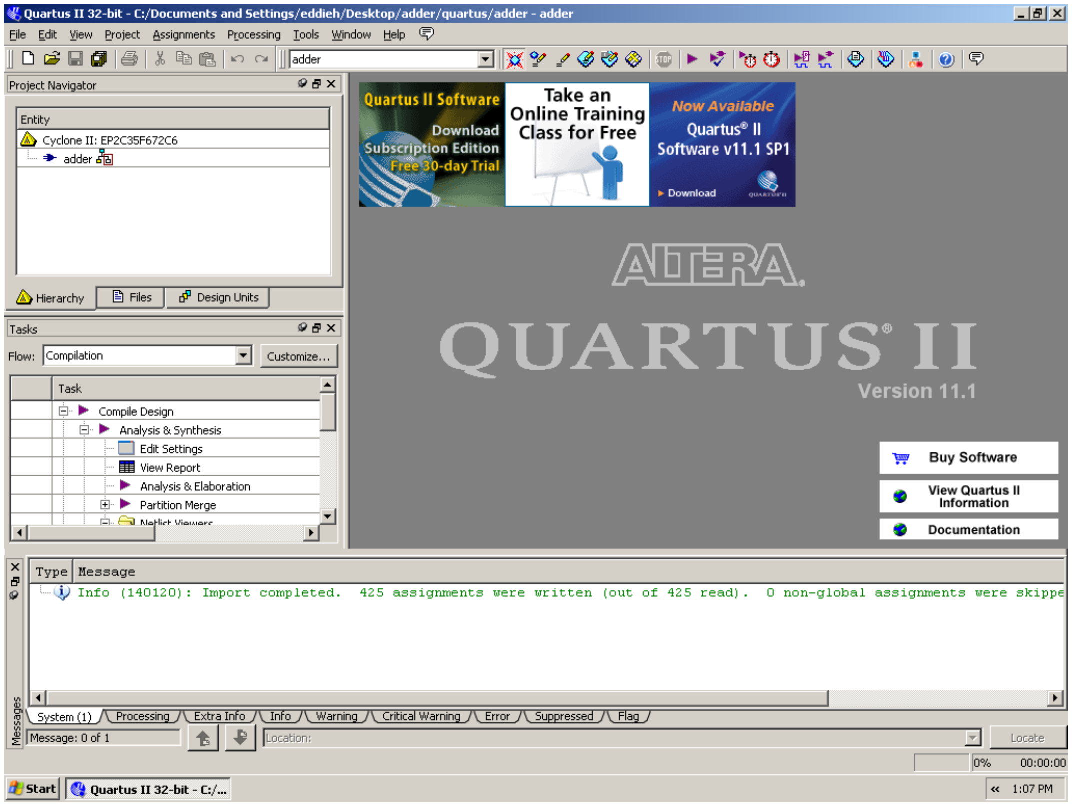
Import Assignments

Specify the source and categories of assignments to import.

File name: settings/eddieh/Desktop/adder/DE2_pin_assignments.csv

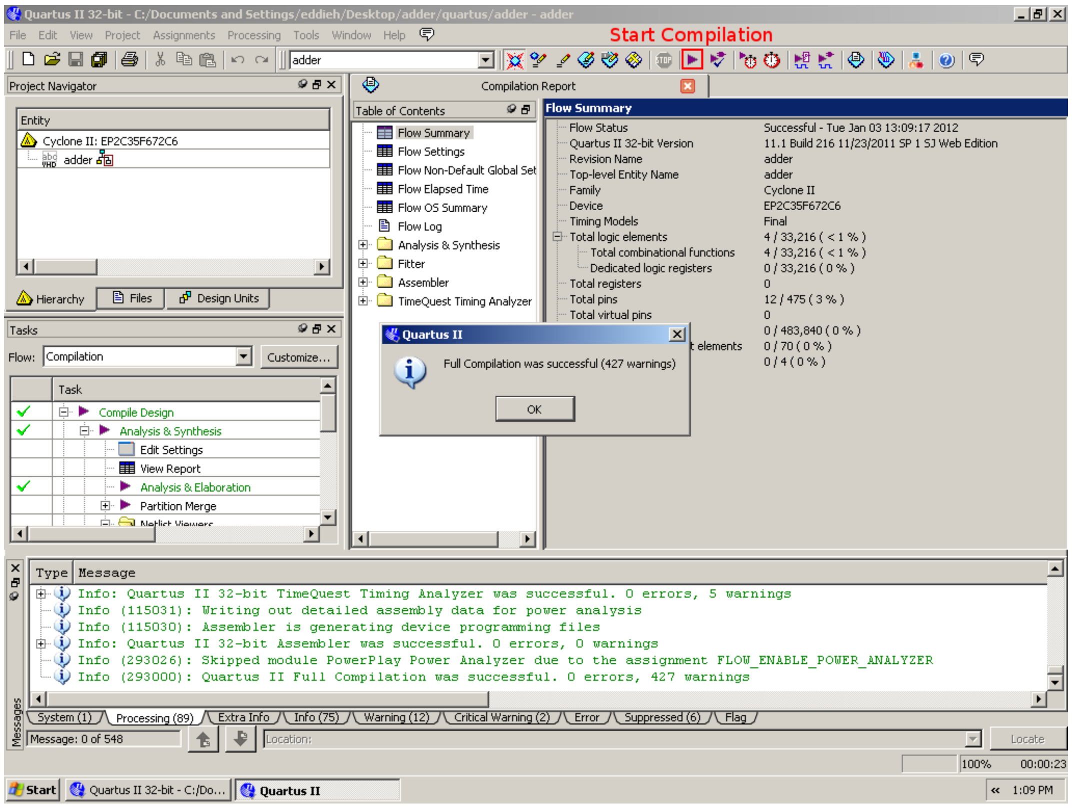
☑ Copy existing assignments into adder.qsf.bak before importing

Figure 15: Importing pin assignments in Quartus II



```
Info (140120): Import completed.  425 assignments were written (out of 425 read).  0 non-global assignments were skippe
```

Figure 16: Pin assignment successfully imported into Quartus II



Full Compilation was successful (427 warnings)

Flow Summary	
Flow Status	Successful - Tue Jan 03 13:09:17 2012
Quartus II 32-bit Version	11.1 Build 216 11/23/2011 SP 1 SJ Web Edition
Revision Name	adder
Top-level Entity Name	adder
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	4 / 33,216 (< 1 %)
Total combinational functions	4 / 33,216 (< 1 %)
Dedicated logic registers	0 / 33,216 (0 %)
Total registers	0
Total pins	12 / 475 (3 %)
Total virtual pins	0
	0 / 483,840 (0 %)
t elements	0 / 70 (0 %)
	0 / 4 (0 %)

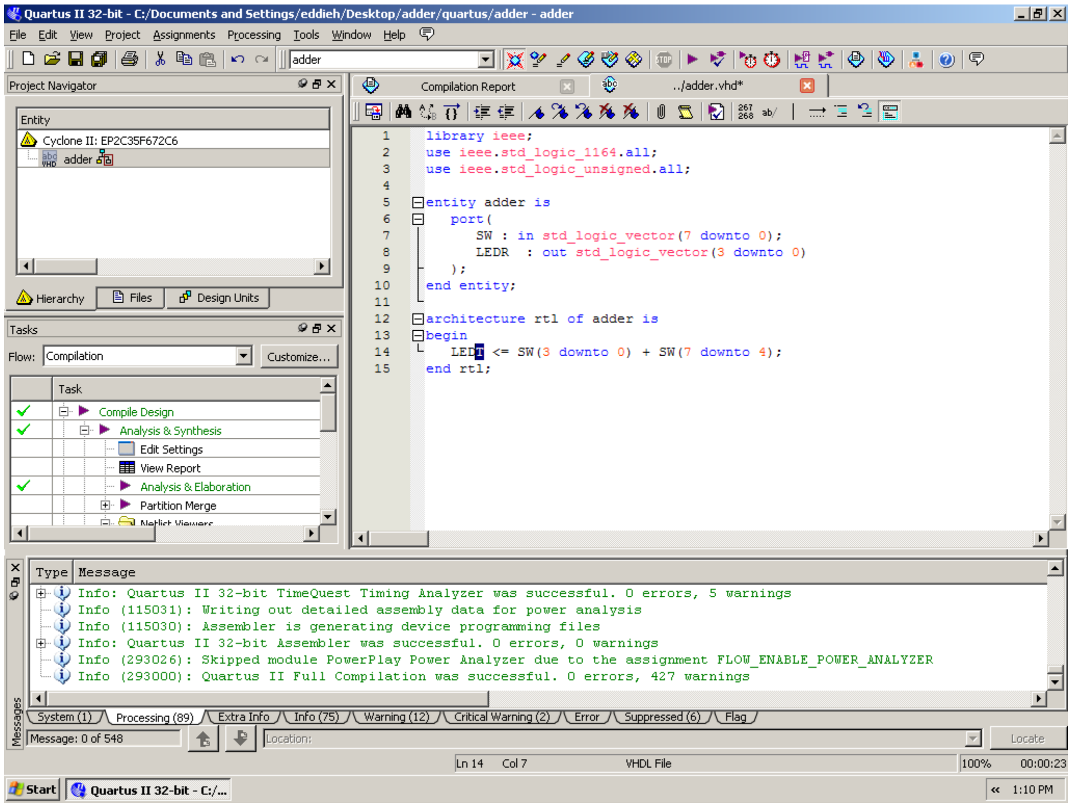
```
Type Message
Info: Quartus II 32-bit TimeQuest Timing Analyzer was successful. 0 errors, 5 warnings
Info (115031): Writing out detailed assembly data for power analysis
Info (115030): Assembler is generating device programming files
Info: Quartus II 32-bit Assembler was successful. 0 errors, 0 warnings
Info (293026): Skipped module PowerPlay Power Analyzer due to the assignment FLOW_ENABLE_POWER_ANALYZER
Info (293000): Quartus II Full Compilation was successful. 0 errors, 427 warnings
```

Figure 17: Successful compilation in Quartus II

Now you are ready to compile your design, so click on the aptly-named "Start Compilation" button. Alternatively, this is available under the "Processing" menu option. Wait a little while for it to do its thing, after which you should see a successful prompt as in Figure 17; if you don't, check that the VHDL source file was added successfully in Assignments — Settings — Files. However, things are not always this rosy.

Let's say we mis-spelled the `LEDR` signal as `LEDT`. Double-click on "adder" in the top-left entity window, and change line 14 as in Figure 18. Compiling the design would now be unsuccessful, as shown by Figure 19. Any errors would appear in the messages window, and as with ModelSim, double clicking on the error message would take you to the vicinity of the error. Revert this change and re-compile again.

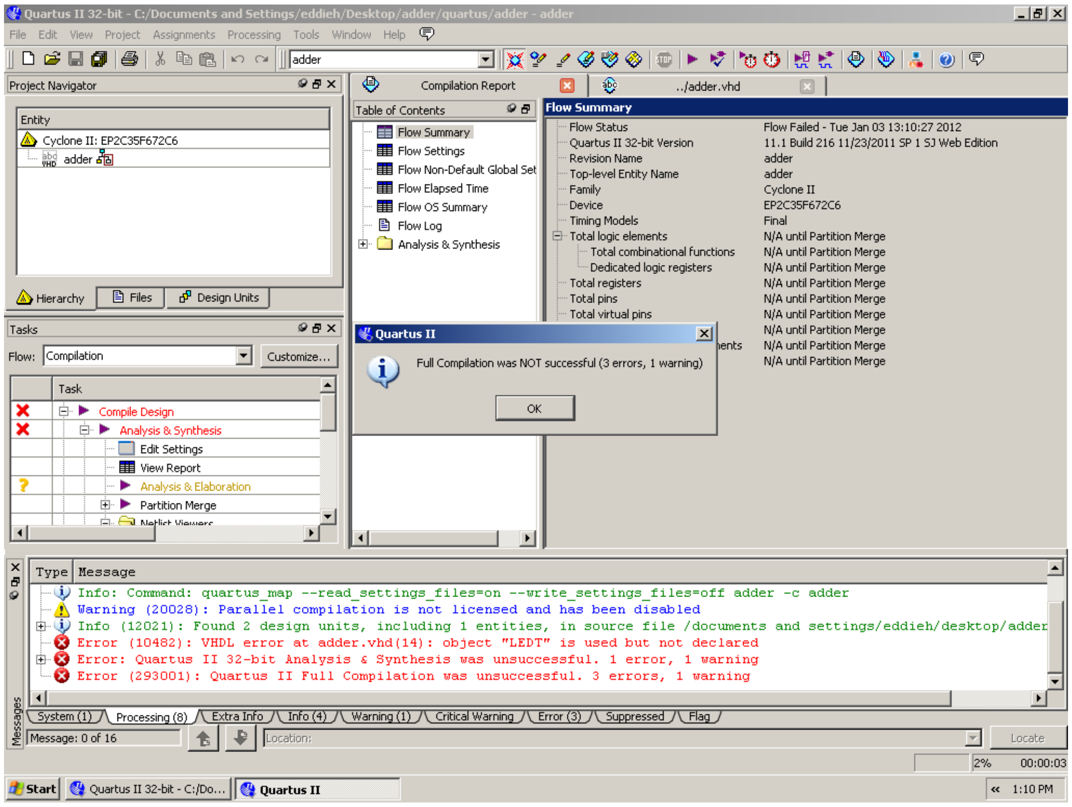
You may notice that even in an successful compilation, a number of info and warnings messages are still generated. Some of those messages are actually quite important, but aren't strictly errors (e.g. any "inferring latches" messages almost always indicate that you are writing unsynthesizable VHDL) and over time and with experience you will come to learn which ones to look out for and what they mean!


Quartus II 32-bit - C:/Documents and Settings/eddieh/Desktop/adder/quartus/adder - adder

```
 1  library ieee;
 2  use ieee.std_logic_1164.all;
 3  use ieee.std_logic_unsigned.all;
 4
 5  entity adder is
 6      port(
 7          SW : in std_logic_vector(7 downto 0);
 8          LEDR  : out std_logic_vector(3 downto 0)
 9      );
10  end entity;
11
12  architecture rtl of adder is
13  begin
14      LEDT <= SW(3 downto 0) + SW(7 downto 4);
15  end rtl;
```

```
Type Message
Info: Quartus II 32-bit TimeQuest Timing Analyzer was successful. 0 errors, 5 warnings
Info (115031): Writing out detailed assembly data for power analysis
Info (115030): Assembler is generating device programming files
Info: Quartus II 32-bit Assembler was successful. 0 errors, 0 warnings
Info (293026): Skipped module PowerPlay Power Analyzer due to the assignment FLOW_ENABLE_POWER_ANALYZER
Info (293000): Quartus II Full Compilation was successful. 0 errors, 427 warnings
```

Figure 18: Editing source files in Quartus II


Quartus II 32-bit - C:/Documents and Settings/eddieh/Desktop/adder/quartus/adder - adder

Flow Summary

Flow Status	Flow Failed - Tue Jan 03 13:10:27 2012
Quartus II 32-bit Version	11.1 Build 216 11/23/2011 SP 1 SJ Web Edition
Revision Name	adder
Top-level Entity Name	adder
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	N/A until Partition Merge
Total combinational functions	N/A until Partition Merge
Dedicated logic registers	N/A until Partition Merge
Total registers	N/A until Partition Merge
Total pins	N/A until Partition Merge
Total virtual pins	N/A until Partition Merge
	N/A until Partition Merge
…ents	N/A until Partition Merge
	N/A until Partition Merge

Quartus II

Full Compilation was NOT successful (3 errors, 1 warning)

```
Type Message
Info: Command: quartus_map --read_settings_files=on --write_settings_files=off adder -c adder
Warning (20028): Parallel compilation is not licensed and has been disabled
Info (12021): Found 2 design units, including 1 entities, in source file /documents and settings/eddieh/desktop/adder
Error (10482): VHDL error at adder.vhd(14): object "LEDT" is used but not declared
Error: Quartus II 32-bit Analysis & Synthesis was unsuccessful. 1 error, 1 warning
Error (293001): Quartus II Full Compilation was unsuccessful. 3 errors, 1 warning
```

Figure 19: Failed compilation in Quartus II

Here's a handy checklist for any Quartus II-related problems:

- Are you using the correct version of Quartus II?
- Is your Z: drive full?
  (If it is, Quartus II will not be able to create any new files and will crash. Delete a few things off it and try again. *Hint*: the "db" and "incremental_db" folders in old projects take up considerable space and can be safely deleted.)
- Is your top-level entity set correctly?
  (You can change this in Assignments — Settings — General)
- Have you set the FPGA model to: Cyclone II EP2C35F672C6?
  (You can change this from Assignments — Device)
- Have you added all source files into your project?
  (Check in Assignments — Settings — Files)
- Have you correctly imported your pin assignments?
  (You can import them again, and compile again, if you are not sure)

4 FPGA Programming using Quartus II

The last step of this tutorial is to program the FPGA chip on the DE2 board. The output of Quartus II from the previous section is a file known as a "bitstream". This bitstream describes the digital logic, in a format specific to the selected FPGA model, that is used to implement the described circuit.

First, attach the power cable to the connector in the top-left of your DE2 board, and the USB cable to the left "BLASTER" port closest to the power socket. Do not connect the USB cable to the right "DEVICE" port. Ensure that the programming switch (below the red power switch, and left of the LCD screen) is set to the "RUN" mode. This enables the FPGA to be temporarily programmed, which is sufficient for this course, as opposed to storing the bitstream permanently into the flash memory. Power-on the DE2 board by pressing the big red power button underneath the power socket. Your board should now look like Figure 20.

If you are using the departmental computers, the DE2 USB-Blaster drivers will have been pre-installed for you. However, if you are using your own computer, you should see that it will prompt you to install a driver for the new USB device that you have just attached. Tell it to search for the driver in a folder inside the Quartus II install directory, such as:

```
C:\altera\13.0sp1\quartus\drivers\usb-blaster
```

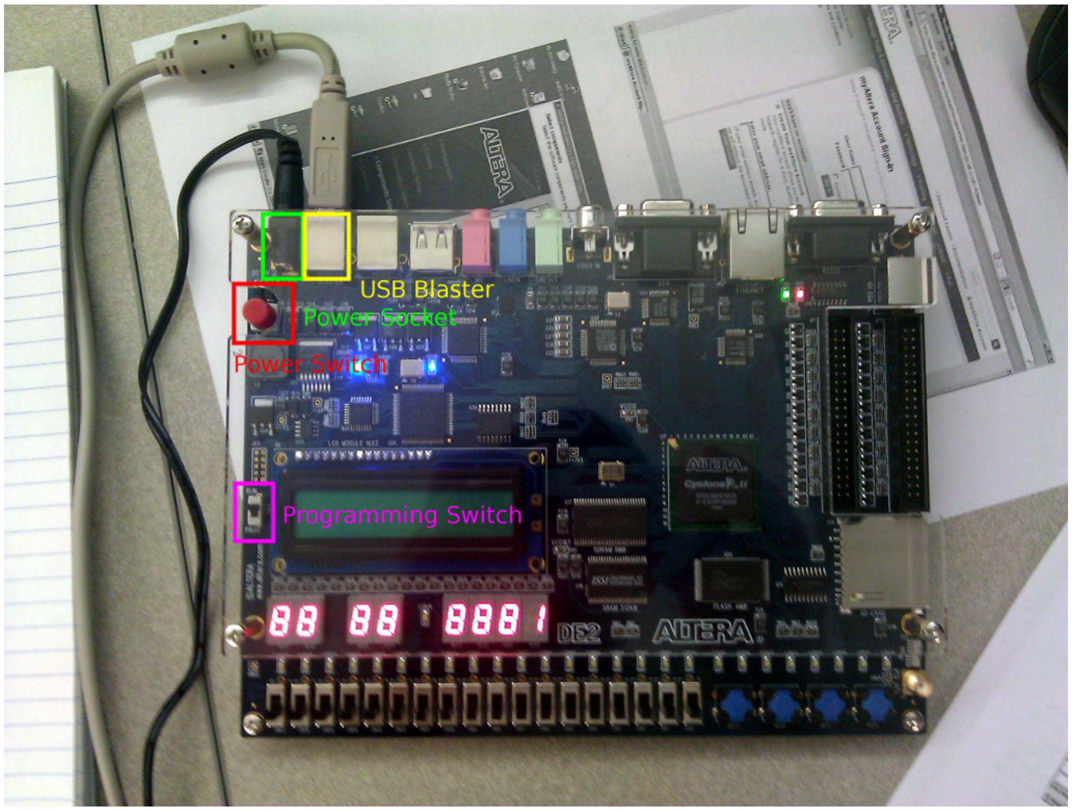
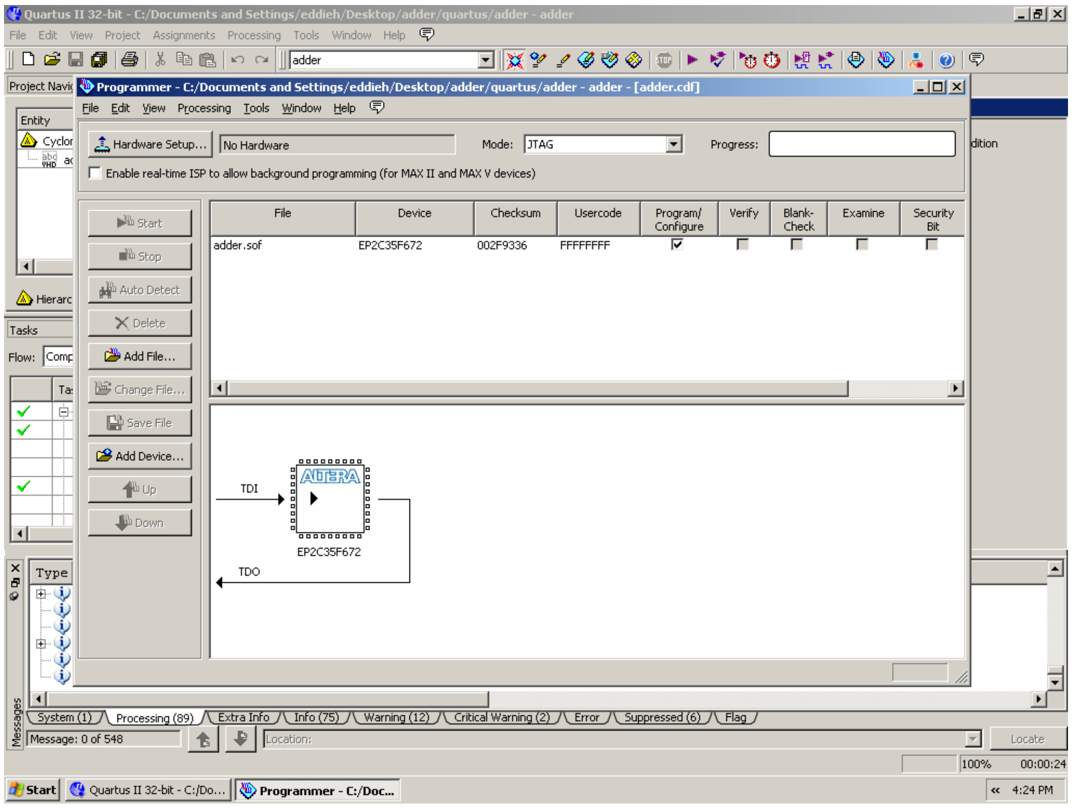


Figure 20: A working DE2 board


Quartus II 32-bit - C:/Documents and Settings/eddieh/Desktop/adder/quartus/adder - adder
Programmer - C:/Documents and Settings/eddieh/Desktop/adder/quartus/adder - adder - [adder.cdf]
File Edit View Processing Tools Window Help
Start
Stop
Auto Detect
Delete
Add File...
Change File...
Save File
Add Device...
Up
Down
TDI
TDO
EP2C35F672
System (1) Processing (89) Extra Info Info (75) Warning (12) Critical Warning (2) Error Suppressed (6) Flag
Message: 0 of 548
Location:
Locate
100% 00:00:24
Start Quartus II 32-bit - C:/Do... Programmer - C:/Doc... 4:24 PM

Hardware Setup... No Hardware   Mode: JTAG   Progress:

Enable real-time ISP to allow background programming (for MAX II and MAX V devices)

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit
adder.sof	EP2C35F672	002F9336	FFFFFFFF	☑	☐	☐	☐	☐

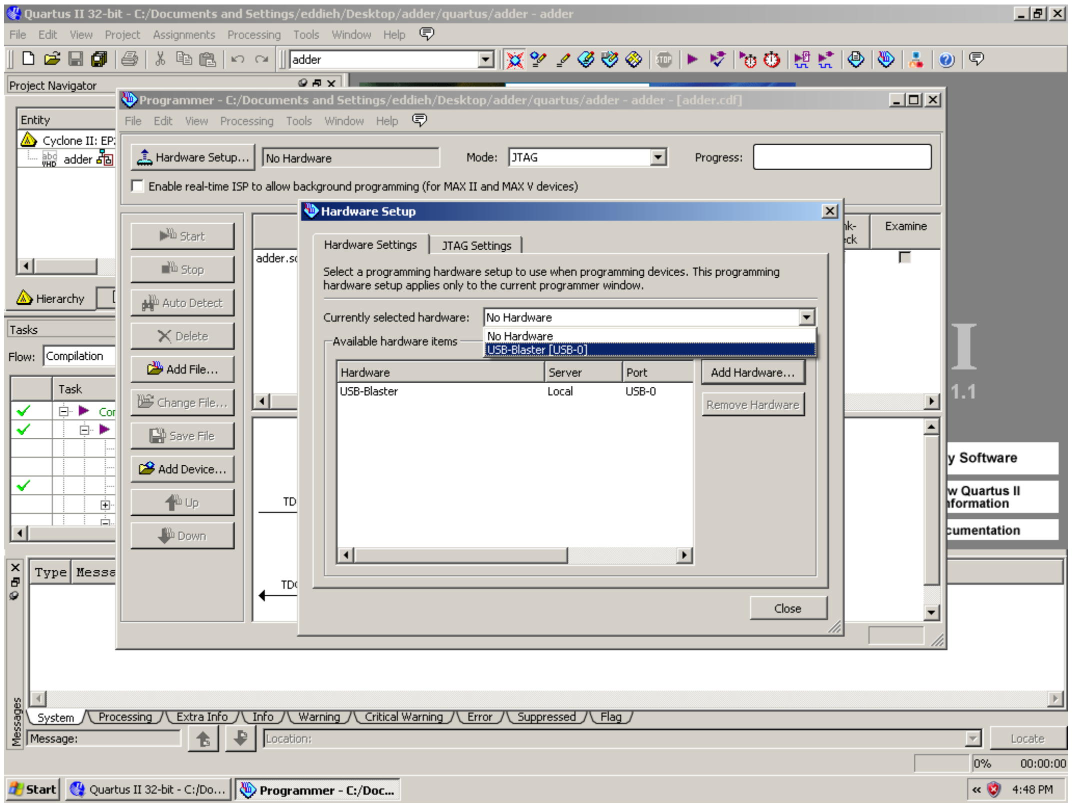
Figure 21: Programmer from Quartus II

If this does not work, you may try the following. Visit `http://threadingbuildingblocks.org/download` Download the latest stable Windows version of the Thread Building Blocks. You will download a zip file. Save the zip file to your hard drive. Look into this zip file, and navigate to bin\intel64\vc9 . Copy everything in that folder to your installation directory which might look something like altera\13.0sp1\quartus\bin64 Some people have reported that the files in v9 do not work, but that the files in v10 do work. I installed this on two computers, and in one case, I found I had to use the files in v9, and in the other, I had to use the files in v10. Just use v9 for now, and later, if your device programmer can not find your USB download hardware, come back and replace the files with those in v10. If you have problems with this, your TAs can help you.

Note: You are more than welcome to use your own computer to program the DE2 boards during the lab sessions, but please bring your own USB cable. The USB cables that are available in the labs are attached to the computers and should not be removed. Also, be forewarned that the TAs will not be able to help you with any driver-related issues.

To transfer the bitstream onto the FPGA, we will be using the Quartus II Programmer tool. Select it from Tools — Programmer, or click on its button from the same toolbar where you found "Start Compilation", which should bring up a window a little like something in Figure 21.

Click on the "Hardware Setup" button in the top left of the Programmer. From the "Currently selected hardware" drop-down box, select "USB-Blaster", as in Figure 22. If USB-Blaster option is not present, then you may not have installed the drivers correctly: double-check the Device Manager. Additionally, ensure that the USB cable is plugged in to the BLASTER port on the DE2 board, the other end is in a working port on your computer, and that the board is powered on.



Programmer - C:/Documents and Settings/eddieh/Desktop/adder/quartus/adder - adder - [adder.cdf]

Hardware Setup... No Hardware Mode: JTAG Progress:

Enable real-time ISP to allow background programming (for MAX II and MAX V devices)

Hardware Setup

Hardware Settings | JTAG Settings

Select a programming hardware setup to use when programming devices. This programming hardware setup applies only to the current programmer window.

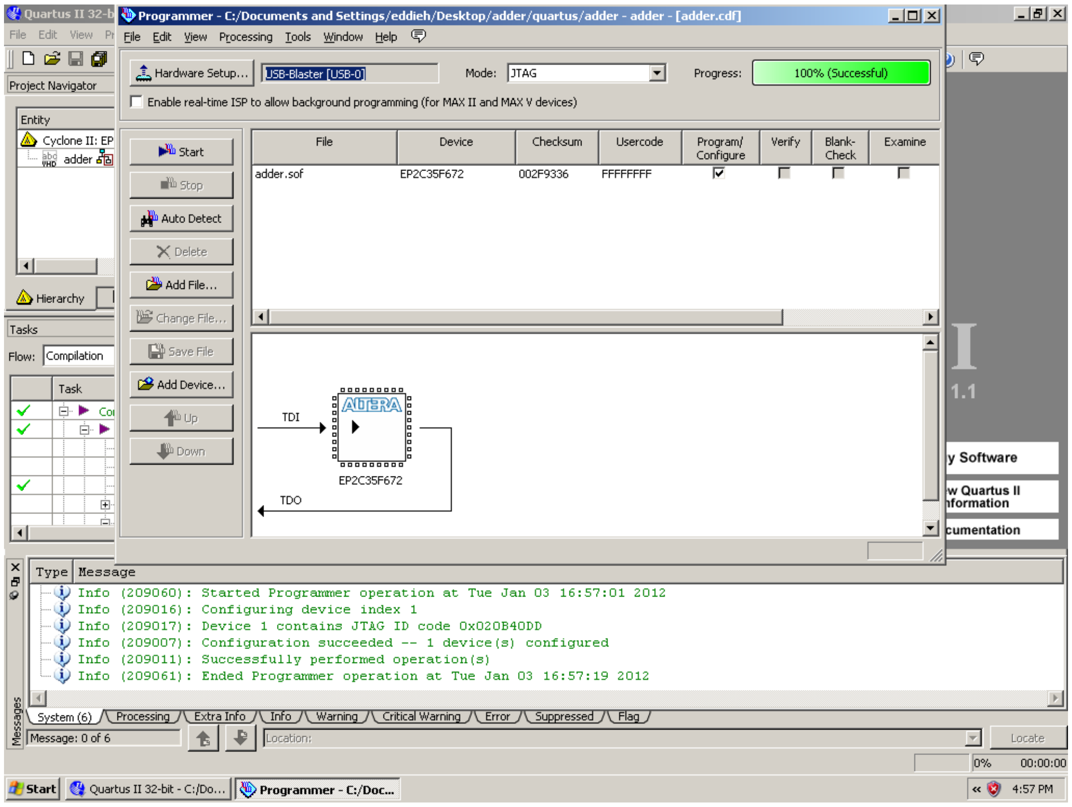
Currently selected hardware: No Hardware

Available hardware items: No Hardware / USB-Blaster [USB-0]

Hardware	Server	Port
USB-Blaster	Local	USB-0

Add Hardware... Remove Hardware Close

Figure 22: Hardware Setup for Programmer



Programmer - C:/Documents and Settings/eddieh/Desktop/adder/quartus/adder - adder - [adder.cdf]

Hardware Setup... USB-Blaster [USB-0] Mode: JTAG Progress: 100% (Successful)

Enable real-time ISP to allow background programming (for MAX II and MAX V devices)

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine
adder.sof	EP2C35F672	002F9336	FFFFFFFF	✓			

```
Type Message
Info (209060): Started Programmer operation at Tue Jan 03 16:57:01 2012
Info (209016): Configuring device index 1
Info (209017): Device 1 contains JTAG ID code 0x020B40DD
Info (209007): Configuration succeeded -- 1 device(s) configured
Info (209011): Successfully performed operation(s)
Info (209061): Ended Programmer operation at Tue Jan 03 16:57:19 2012
```

Figure 23: Successful programming operation

The bitstream file, `adder.sof`, should have been automatically added to the Programmer for you (if not, re-run the compilation procedure, check that there are no errors, and that you're Z: drive is not full. If that doesn't work, choose "Add File" and browse to "adder.sof" (which may be in the "output_files" directory). Additionally, the "Mode" drop-down box should say "JTAG". If not, check that the programming switch is set to "Run", and that the FPGA model is set correctly (Quartus II will not let you program a bitstream intended for another model onto the DE2).

Click on the "Start" button in the top left corner of the Programmer. You should now arrive at Figure 23.

Now play around with the SW0-7 switches on your DE2 board to ensure that the adder is working correctly.

Congratulations! You have now successfully simulated a VHDL design using the ModelSim software package to ensure that it functions correctly, compiled (or synthesized) this design into a hardware bitstream using Quartus II, and now programmed this bitstream onto the FPGA. Give yourself a pat on the back.

In case it's not quite so plain-sailing in the future, here's a handy checklist for any Programmer-related problems:

- Are you using the correct version of Quartus II?
- Is your Z: drive full?
  (If it is, Quartus will not be able to create any new files and will crash. Delete a few things off it and try again)
- Have you set the FPGA model to: Cyclone II EP2C35F672C6?
  (You can change this from Assignments — Device)
- Have you attached the USB cable to the "BLASTER" port on the DE2?
- Is the programming switch set to "RUN"?
- Is the board powered on?
- Is the USB-Blaster driver installed correctly?
- Is "USB-Blaster" displayed beside the "Hardware Setup" button?
  (If not, press the button and select it. If it is not listed, check that the board is turned on, the cable is connected properly, and the USB-Blaster drivers are installed.)

Good luck!