

Spring 24, Ken Short

## Laboratory 02a: VHDL/PLD Design Flow - Synthesis and Post-Synthesis Simulation

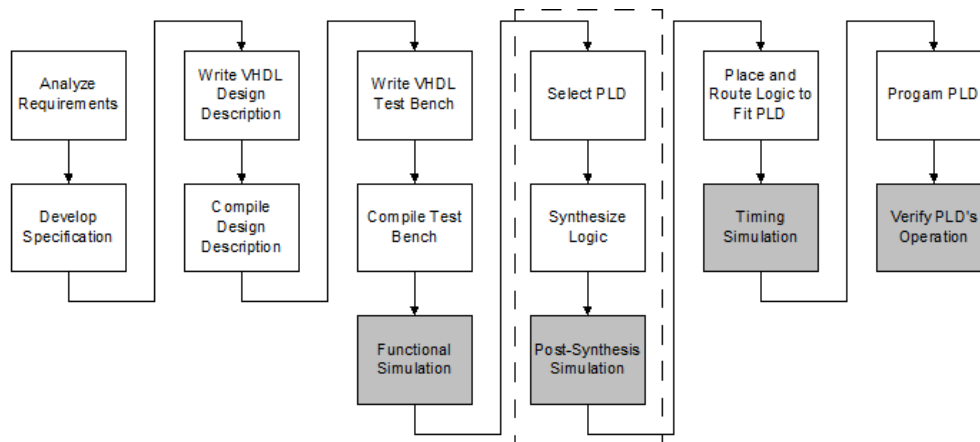
This laboratory is to be performed the week starting Feb. 4th.

### Prerequisite Reading

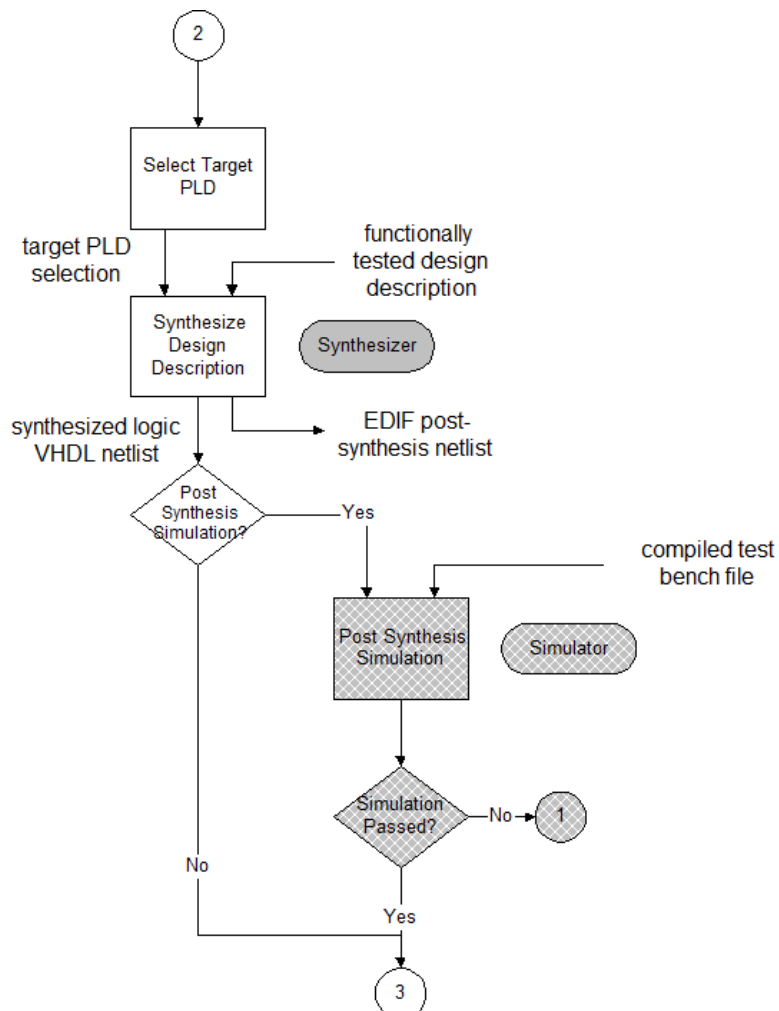
1. Sections 1.7 through 1.9 of the text.
2. ispGAL22V10C-10 Data Sheet.

### Purpose

The purpose of this laboratory is to provide you with experience in how a VHDL design description is synthesized and how the functionality of the synthesized logic is verified. The focus is on that portion of the VHDL/PLD design flow from the selection of a PLD to post synthesis simulation (Inside dashed rectangle in figure that follows). The half-adder VHDL design description from Laboratory 01 is to be synthesized and the VHDL netlist generated by the synthesizer is to be simulated. The purpose of this post synthesis simulation is to verify the functionality of the logic synthesized from the VHDL design description by the synthesizer. **This post-synthesis simulation is often skipped, and the timing simulation is instead used to verify both the functionality of the synthesized logic and its timing when mapped to the target PLD.**



The portion of the design flow that is of interest in this laboratory is shown in more detail in the following figure.



The synthesizer tool used is Synplicity's Synplify Pro. Synplify Pro is a CPLD and FPGA logic synthesis tool. Synplify Pro accepts design descriptions written in VHDL and creates netlists based on the selected target CPLD or FPGA. The netlists generated are: a VHDL netlist used for post synthesis simulation and an EDIF netlist used for programming the PLD.

## Design Tasks

The first step in this sequence of phases is selection of a target PLD. This requires you to have a knowledge of the types of PLDs available, their architectures, capabilities, and performance. Since we have yet to cover PLD architectures in any detail, the target PLD, an ispGAL22V10C-10, has been selected for you. The ispGAL22V10C-10 is one of the simplest commercially available SPLDs.

Other than selection of a target PLD, there are no other design aspects to this part of the design flow.

## Laboratory Tasks

### *Overview*

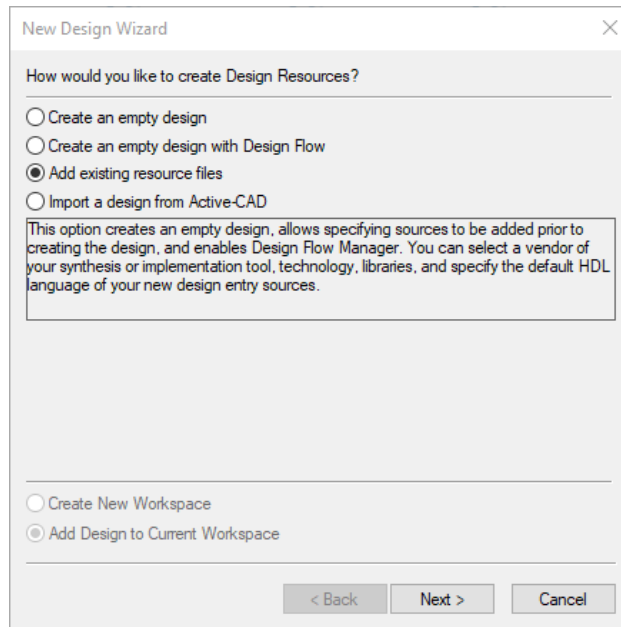
In the laboratory you will go through several phases in the VHDL/PLD design flow to produce and simulate a VHDL netlist model from a design description. This is the VHDL model you simulate to perform a post synthesis simulation. The phases you will carry out are:

1. Create a new project named lab02. Import the design files for the design description and the testbench from Laboratory 1.
2. Use Aldec's Active-HDL simulator to compile and functionally simulate this design, just as you did last week. Save the waveforms from this simulation.
3. Use Synplicity's Synplify synthesizer tool to synthesize logic corresponding to your VHDL design description.
4. Replace the original VHDL design description, instantiated as the UUT in the testbench, with the VHDL netlist generated by the synthesizer. This is done by removing the design description file from the Aldec project and replacing it with the VHDL netlist file.
5. Simulate the VHDL netlist model to produce a post-synthesis functional simulation.
6. Compare the simulator outputs from these two simulations to determine if the logic generated by the synthesizer is functionally correct.

### *Create a new project*

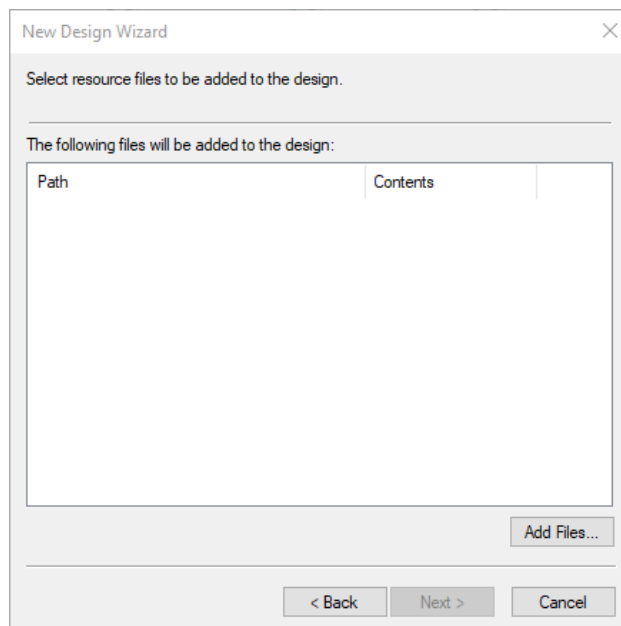
1. Follow the procedures from Laboratory 1 to launch Active-HDL and create a new workspace and a new project, both named lab02. Import the design description and testbench for the half

adder from Laboratory 1 into this project by selecting Add existing resource files in the New Design Wizard.

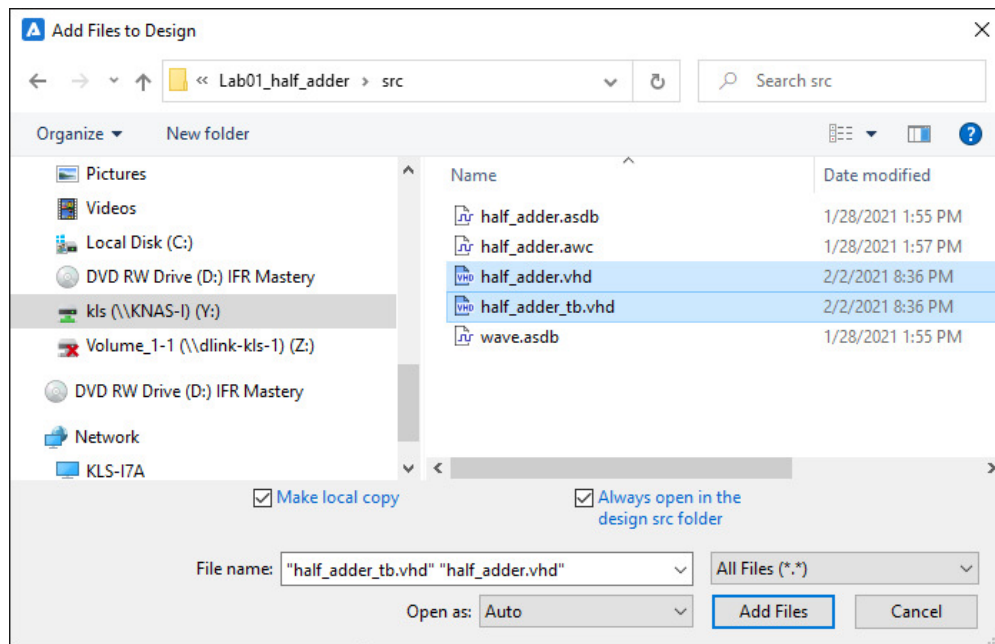


Click Next.

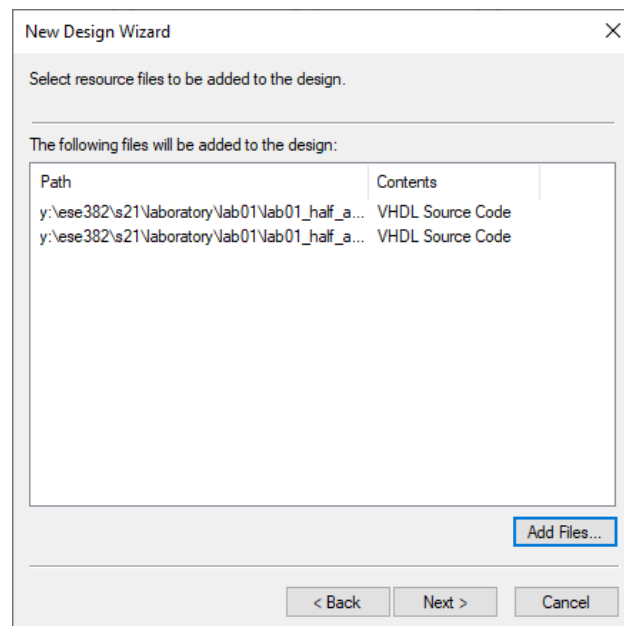
The New Design Wizard will ask you to Select the resource files to be added to the design.



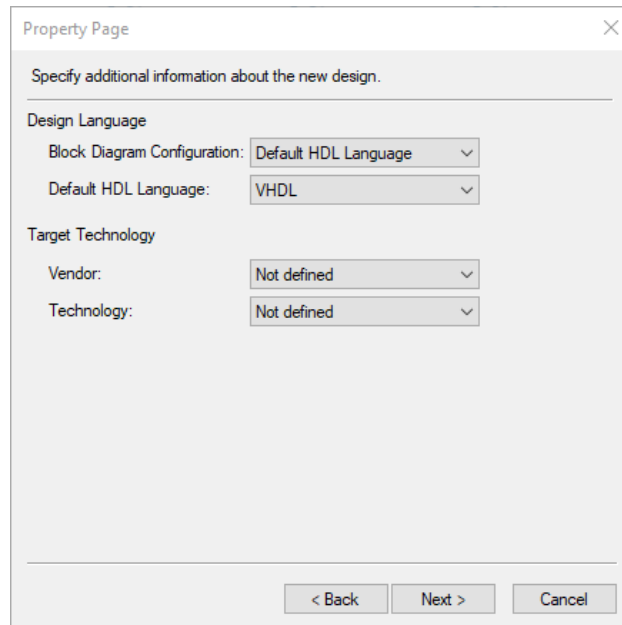
Click on Add Files. An ADD Files to Design window will open and you can browse to the folder for Lab01 and select the half\_adder and half\_adder\_tb files.



**Make sure the Make local copy box is checked.** Click on Add Files. The following window will open.



Click on Next.



Property Page

Specify additional information about the new design.

Design Language

Block Diagram Configuration: Default HDL Language

Default HDL Language: VHDL

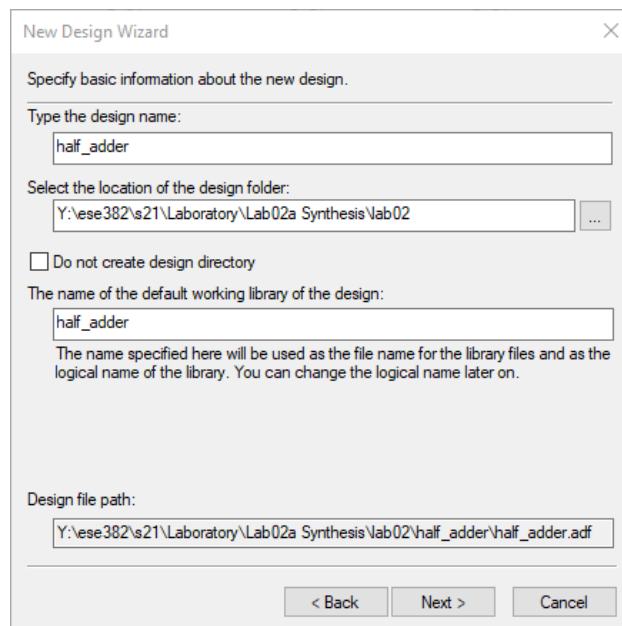
Target Technology

Vendor: Not defined

Technology: Not defined

< Back Next > Cancel

Click on Next.



New Design Wizard

Specify basic information about the new design.

Type the design name:

half\_adder

Select the location of the design folder:

Y:\ese382\s21\Laboratory\Lab02a Synthesis\lab02 ...

☐ Do not create design directory

The name of the default working library of the design:

half\_adder

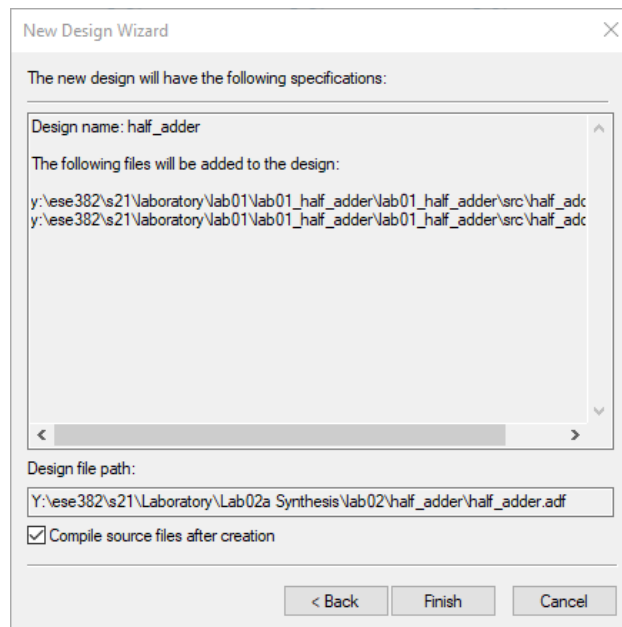
The name specified here will be used as the file name for the library files and as the logical name of the library. You can change the logical name later on.

Design file path:

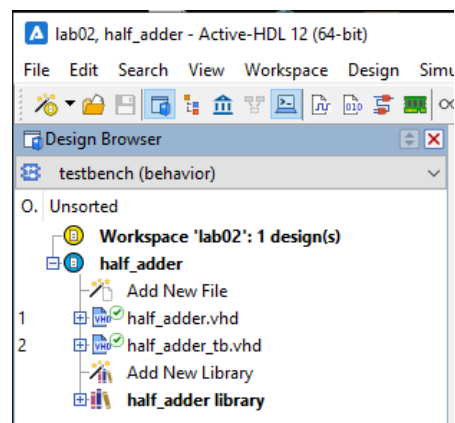
Y:\ese382\s21\Laboratory\Lab02a Synthesis\lab02\half\_adder\half\_adder.adf

< Back Next > Cancel

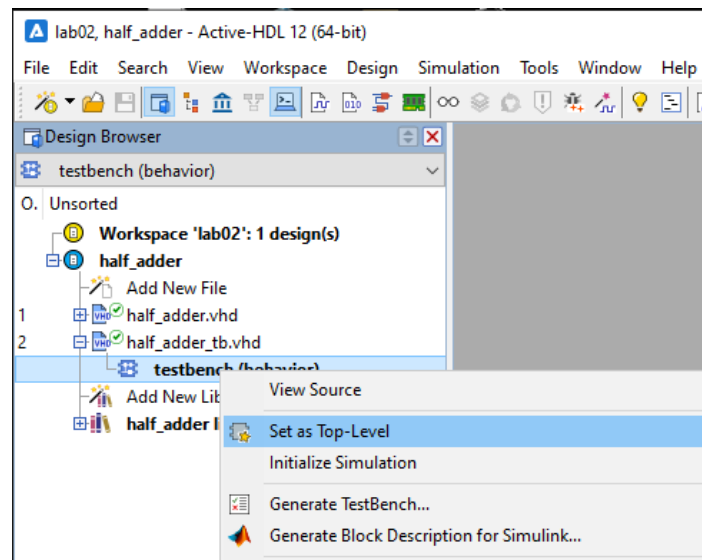
Type in the name of the design, half\_adder. Click Next.



Click Finish.



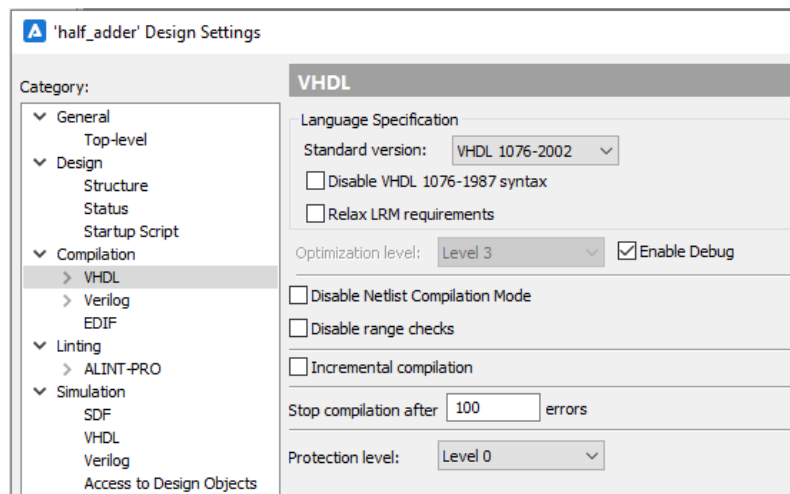
In the Design Browser window you should see that a new design has been created that includes the files half\_adder and half\_adder\_tb.



Expand the tree for half\_adder\_tb. Right click on the on the testbench(behavior) entity/architecture icon and select Set as Top-Level.

### ***Compilation and Functional Simulation***

**Before starting a simulation check that the VHDL standard to be used is 1076-2008 and that Enable Debug is selected. Go to Design > Setting > Compilation > VHDL. Use the drop down selection window to select 1076-2008 and click on Enable Debug, then click on OK.**



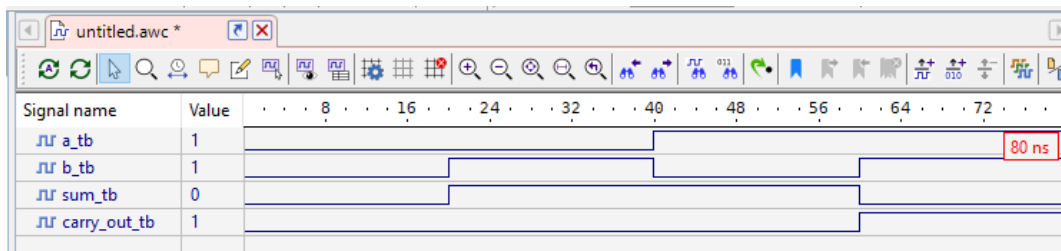
Click on Apply and OK at the bottom of the window.



1. Follow the procedures in Laboratory 1 to compile and functionally simulate your half-adder design description. Set the simulation to run for 80 ns. You can click on the **Compile All** icon to compile both design files in one operation.

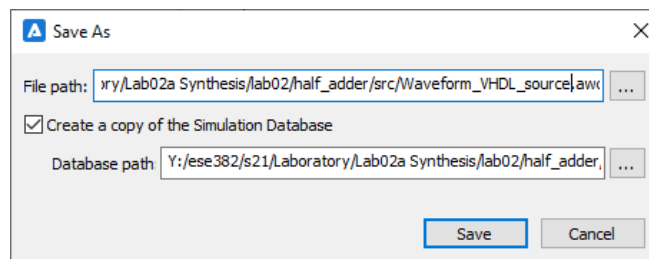


Make sure that the signals that you select to appear in the waveforms are a\_tb, b\_tb, sum\_tb, and carry\_out\_tb:

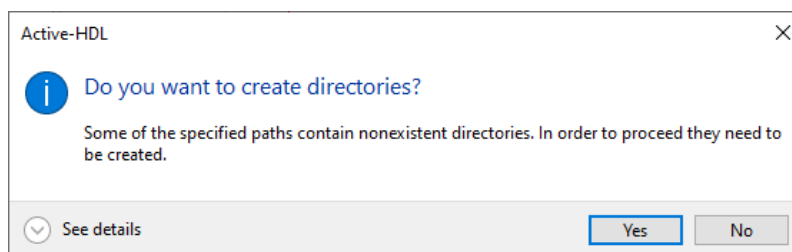


2. After the simulation is complete, Click the red box to stop debugging.

Save the timing simulation waveforms by selecting **Save as** from the **File** menu. Give the file the name Waveform\_VHDL\_source.



In response to a pop window may ask if you want to create directories. Click on Yes.



3. Save your design by clicking **Save All** and then **Close Workspace** from the **File** menu.

4. Close Active-HDL by selecting **Exit** from the **File** menu.

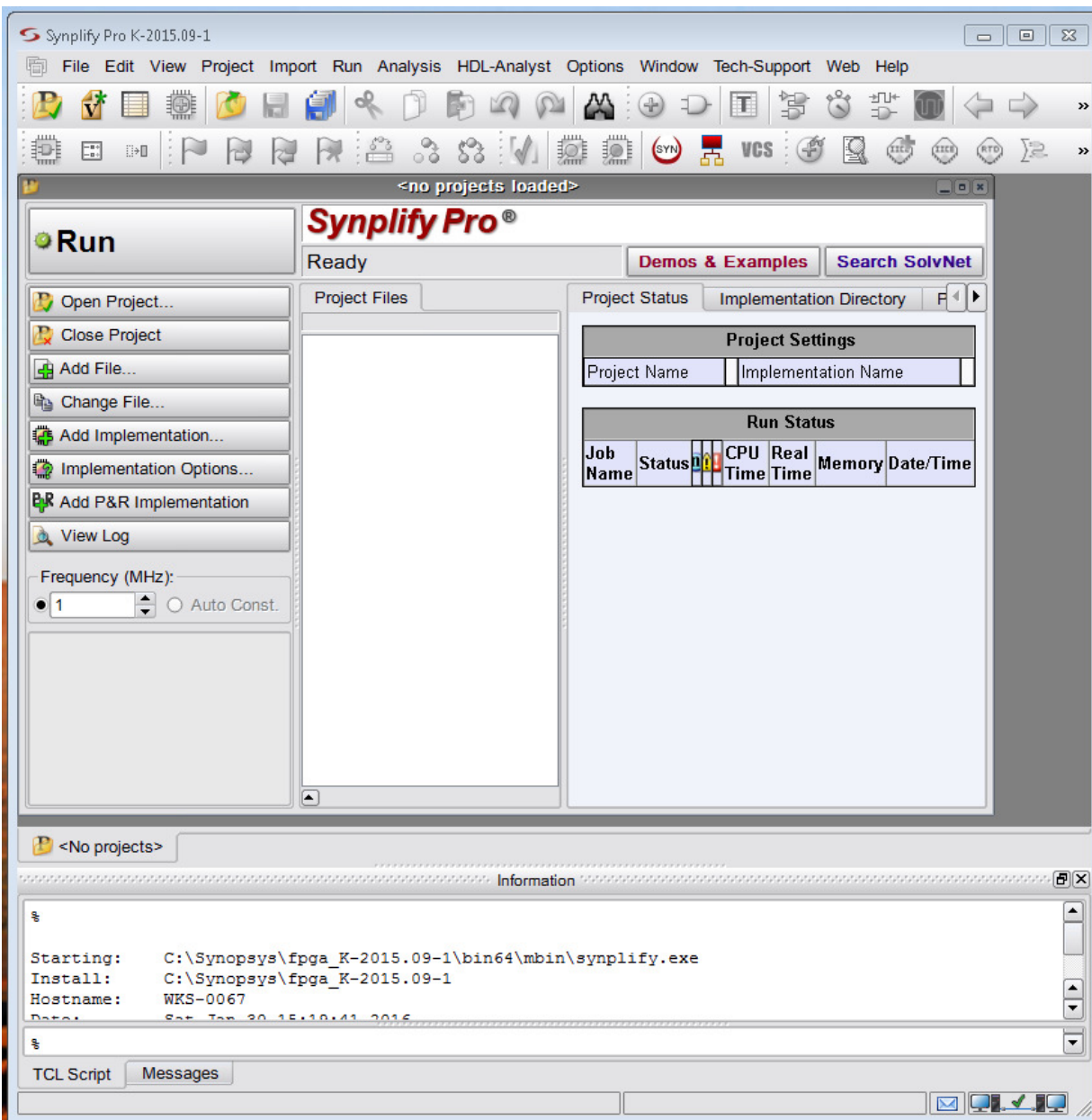
## *Synthesizing Logic and Creating an VHDL netlist file*

You will now use Synplify to synthesize logic from your half-adder design description.

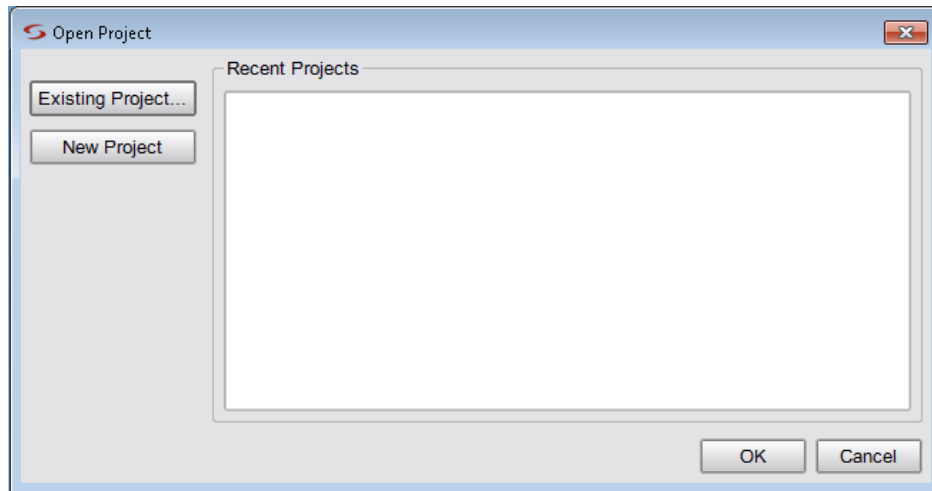
1. Double click the desktop icon to launch the Synplify synthesis tool.



The Synplify main window appears.

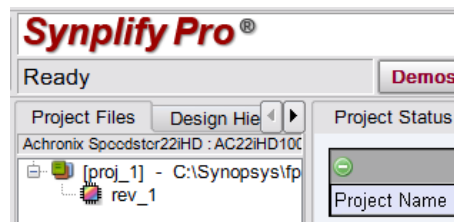


Click on the **Open Project** button on the left.

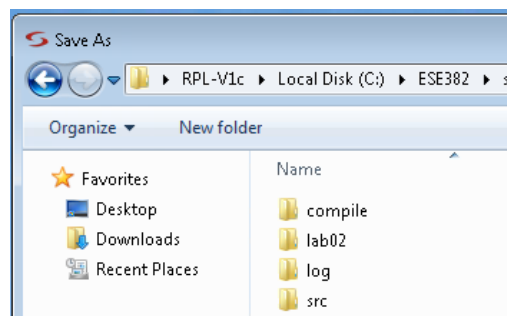


Click on **New Project**.

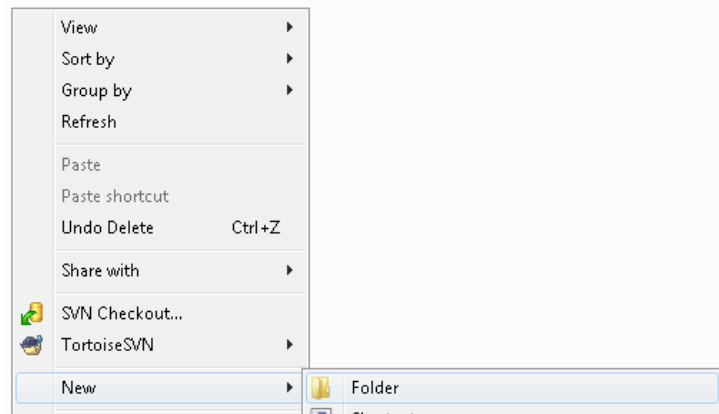
A project contains information about your synthesis run, including design file names, constraint file name (if used), and other options you set for the synthesis run. Clicking on New Project creates a new project, as can be seen in the Project Tree View (window).



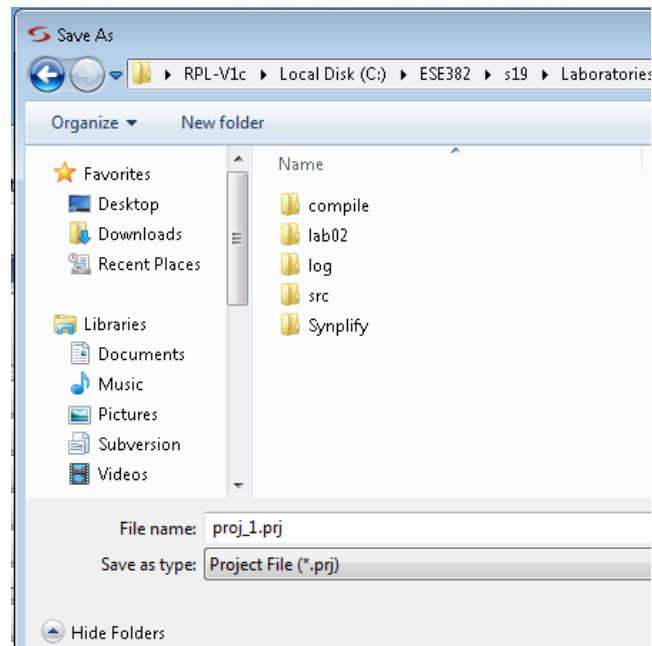
Before continuing we want to save the project. From the **File** menu select **Save As**. This opens the **Save As** window.



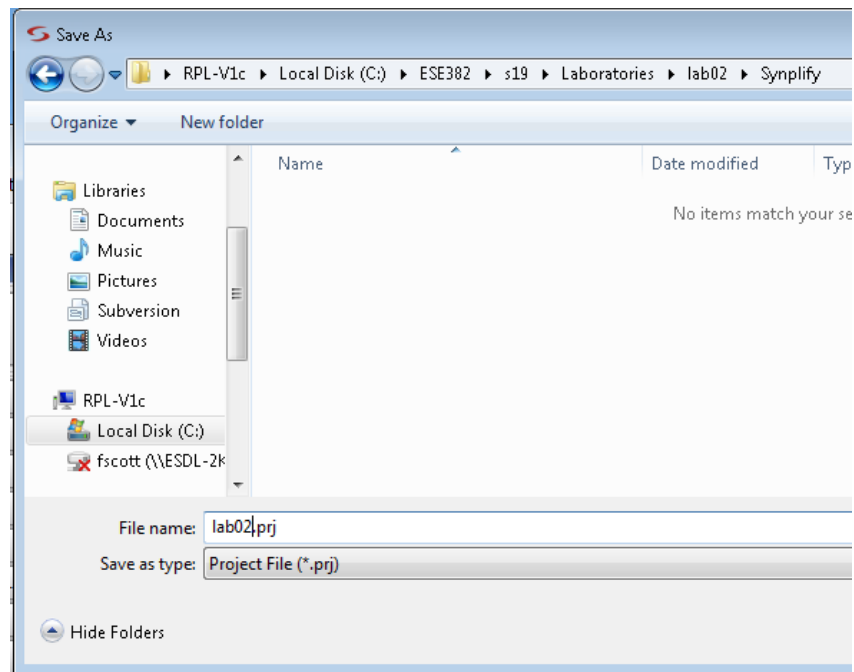
Browse to and open the folder lab02. Right click in the open space below the folders in the Save As window. A menu will open. Select New > Folder.



Name the folder Synplify.



Double click on the newly created Synplify folder to open it. Type lab02.prj in the **File name** box



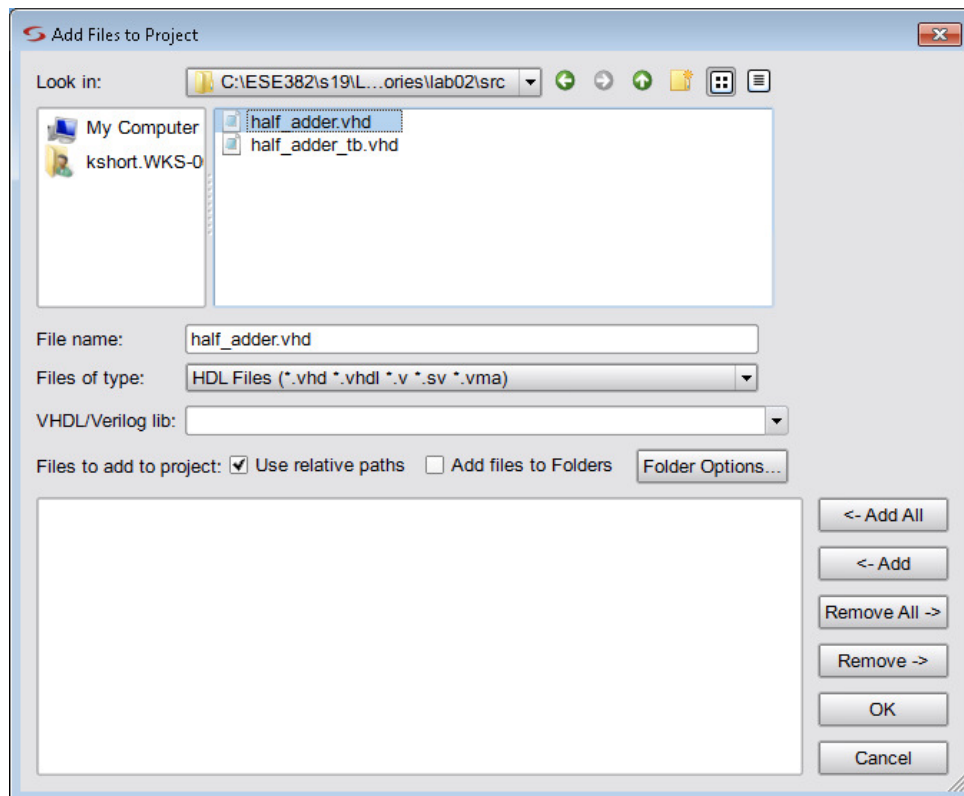
Click **Save**.

When you save a project, the information is written out to a project file in Tcl (Tool command language - an industry standard scripting language).

Click on the **Add File** button. This opens the Select Files to Add to Project window. In the Files of Type drop down menu select HDL Files. Using the Parent Directory button in the window and the

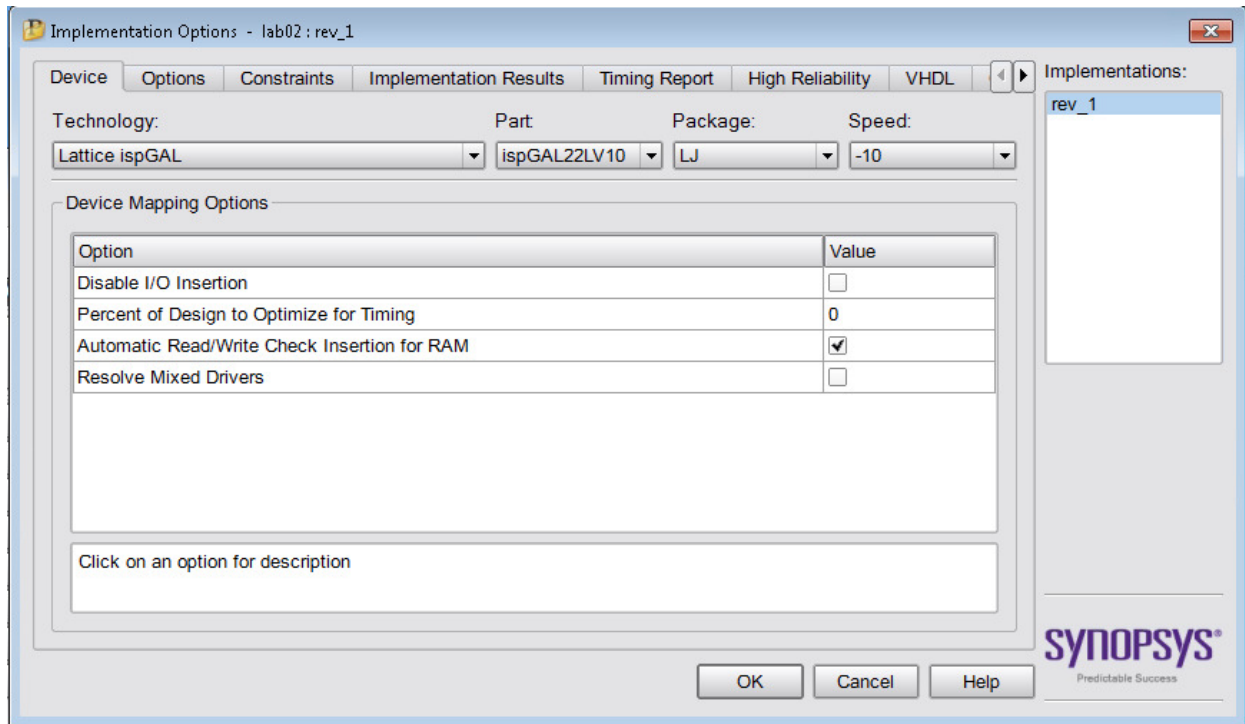


window's drop-down menu, browse to the location of your file half\_adder.vhd (in the src folder).



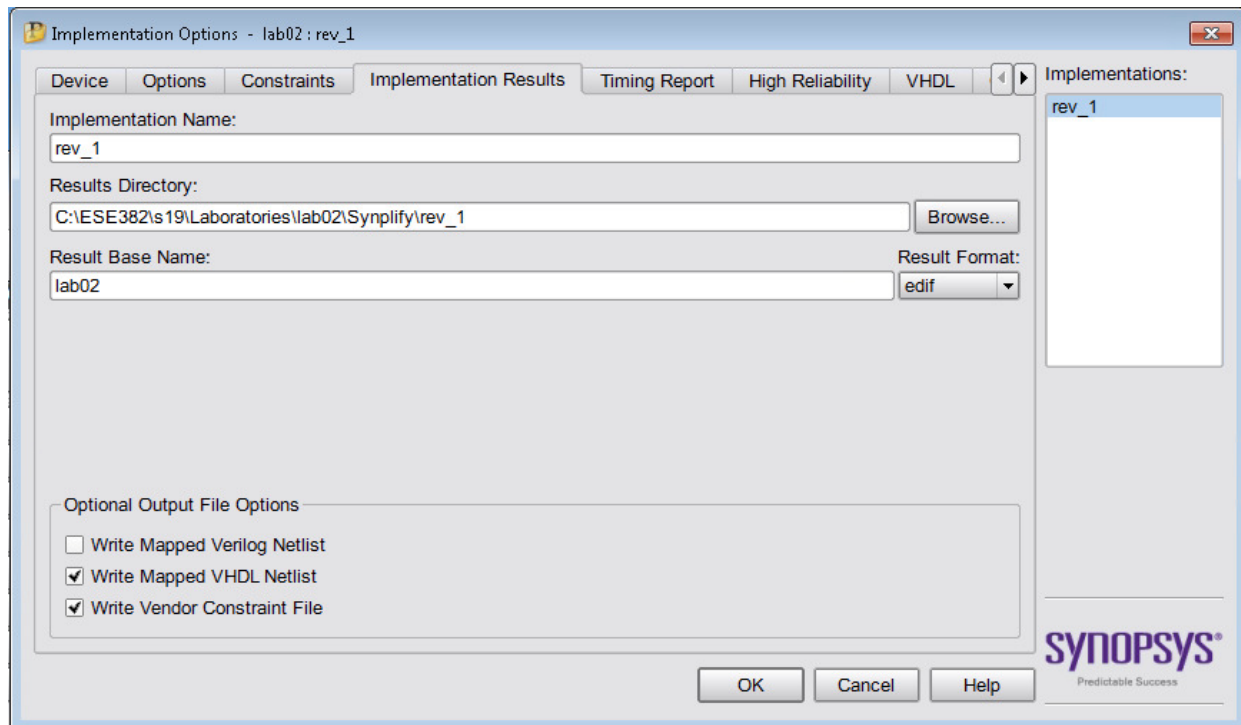
Select the file then click **Add**. This adds the file to the project. *The file `half_adder_tb` is not to be added to this project.* Click **OK** to close this window

Click on Implementation Options button on the left. This opens the **Implementation Options** window.

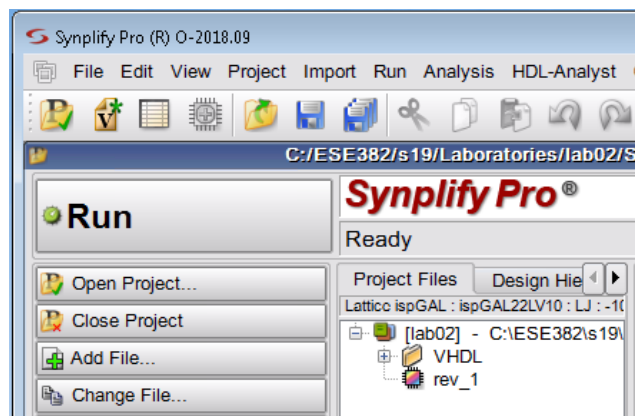


In the text boxes under the **Device's Tab**, select the information shown that specifies that the device is a Lattice ispGAL22V10C-10 in a LJ package.

Click on the **Implementation Results** tab. Put a check in the **Write Mapped VHDL Netlist** checkbox. Uncheck the Write Mapped Verilog Netlist. Specify the Result Base Name as lab02. Click **OK** to close the window.

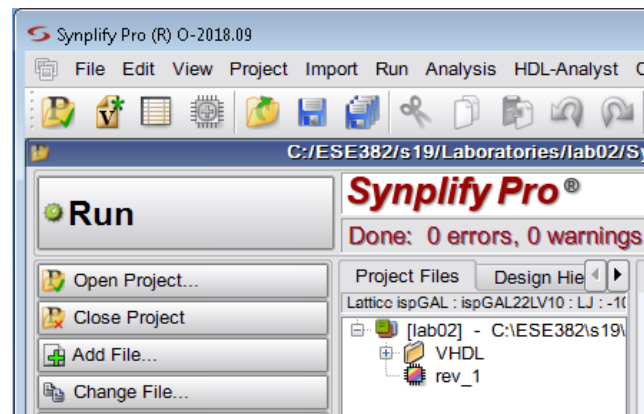


Click the large **RUN** button in the main Synplify window. When synthesis is completed, “Done!” will appear in the main window.

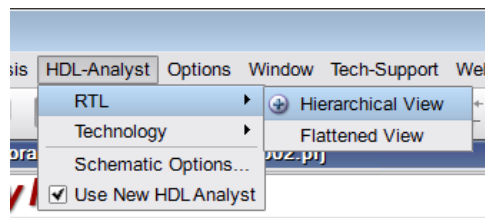


Synplify contains its own compiler. If there are synthesis errors in your source code, they will be detected. Synthesis errors can occur in your design description from using some of the VHDL constructs that can't be synthesized.

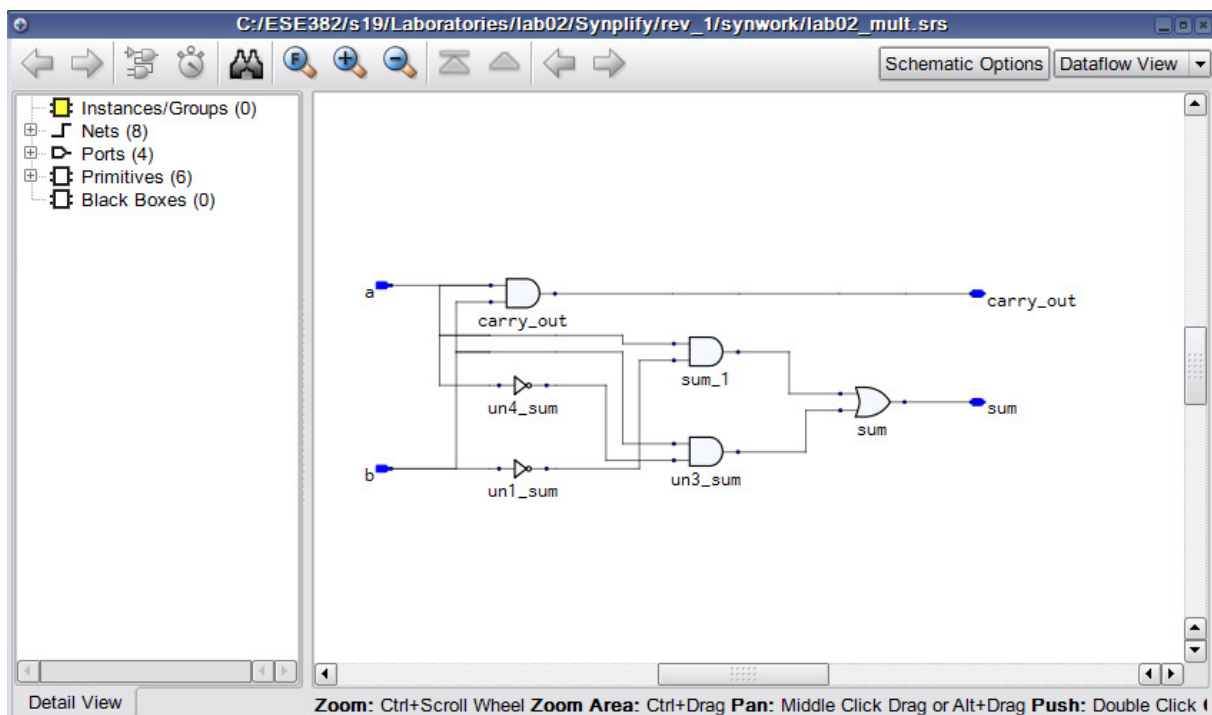




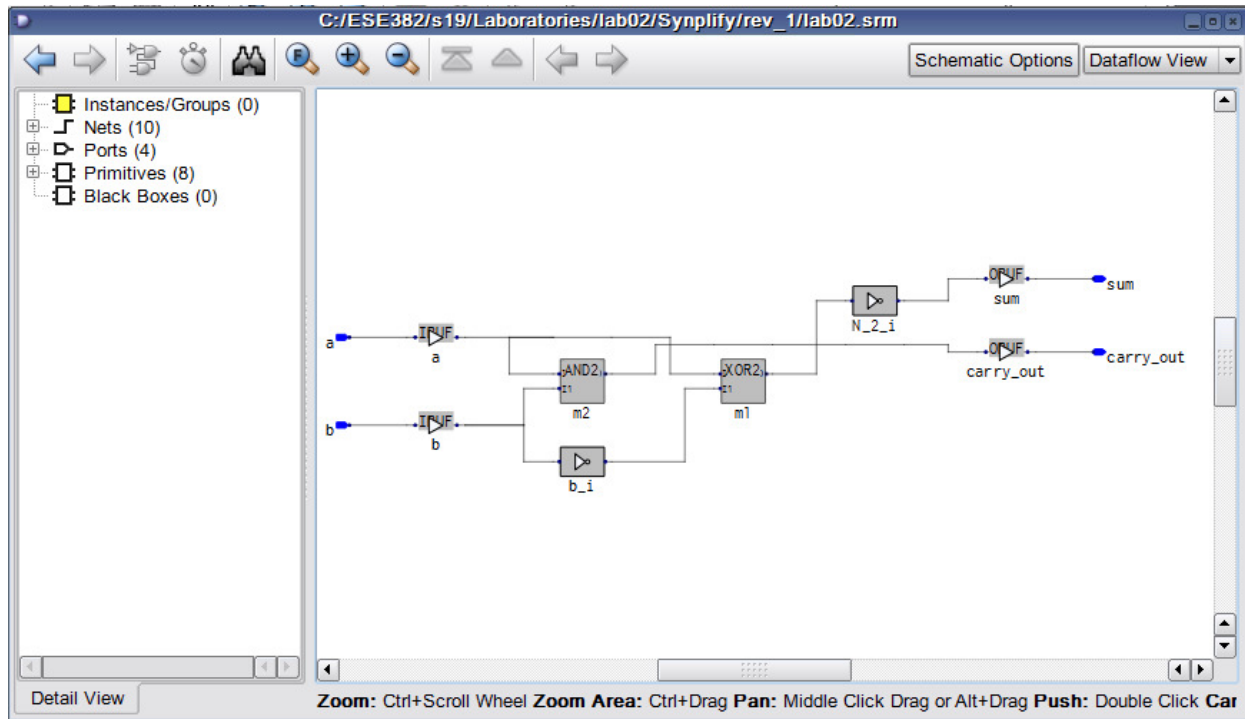
From the **HDL Analyst** menu, select **RTL**, and then click on **Hierarchical View**.



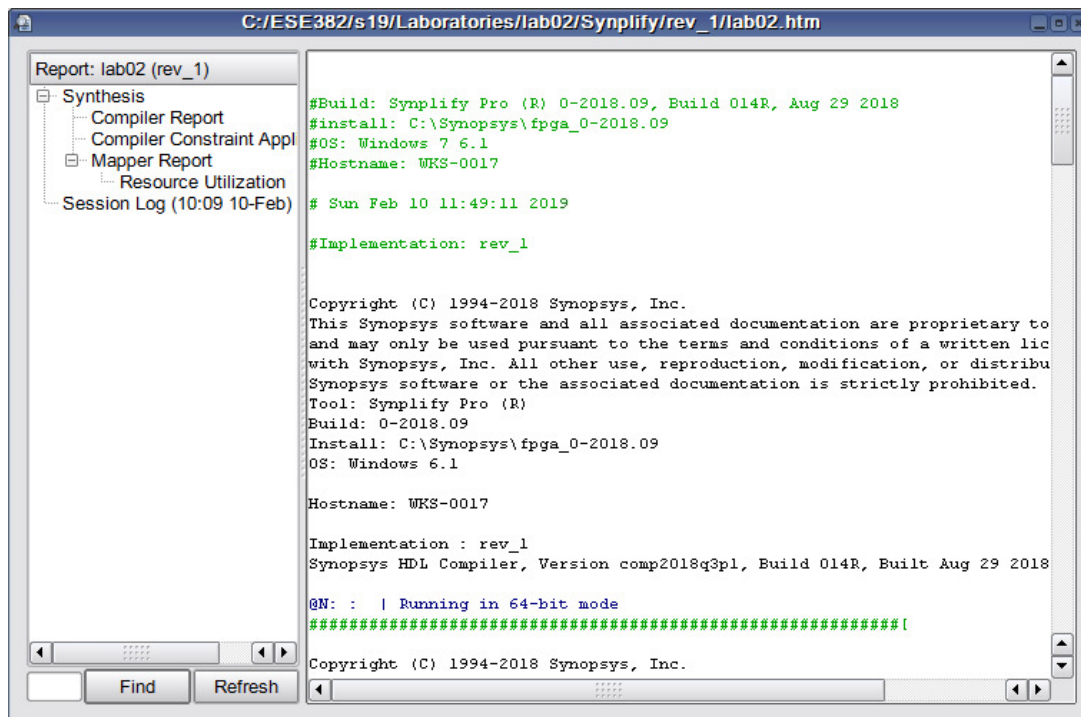
You will see the hierarchical diagram of the RTL synthesized logic. This is a technology independent RTL level schematic. Print this schematic.



Next, from the **HDL Analyst** menu, select **Technology**, and then click on **Flattened to Gates View**. You will see the gate-level diagram of the synthesized logic. This is a schematic of the design after it is mapped to the target PLD architecture. Print this schematic.



From the selections under the big Run button, select **View Log File**. The log file of the synthesis run is displayed. Print this file. Note the Resource Usage Report portion of this file.



The screenshot shows a web browser window displaying the Synplify log file. The left sidebar shows a tree view with 'Report: lab02 (rev\_1)' selected, and 'Session Log (10:09 10-Feb)' is the active report. The main content area displays the following text:

```
#Build: Synplify Pro (R) 0-2018.09, Build 014R, Aug 29 2018
#install: C:\Synopsys\fpga_0-2018.09
#OS: Windows 7 6.1
#Hostname: WKS-0017

# Sun Feb 10 11:49:11 2019

#Implementation: rev_1

Copyright (C) 1994-2018 Synopsys, Inc.
This Synopsys software and all associated documentation are proprietary to
and may only be used pursuant to the terms and conditions of a written lic
with Synopsys, Inc. All other use, reproduction, modification, or distribu
Synopsys software or the associated documentation is strictly prohibited.
Tool: Synplify Pro (R)
Build: 0-2018.09
Install: C:\Synopsys\fpga_0-2018.09
OS: Windows 6.1

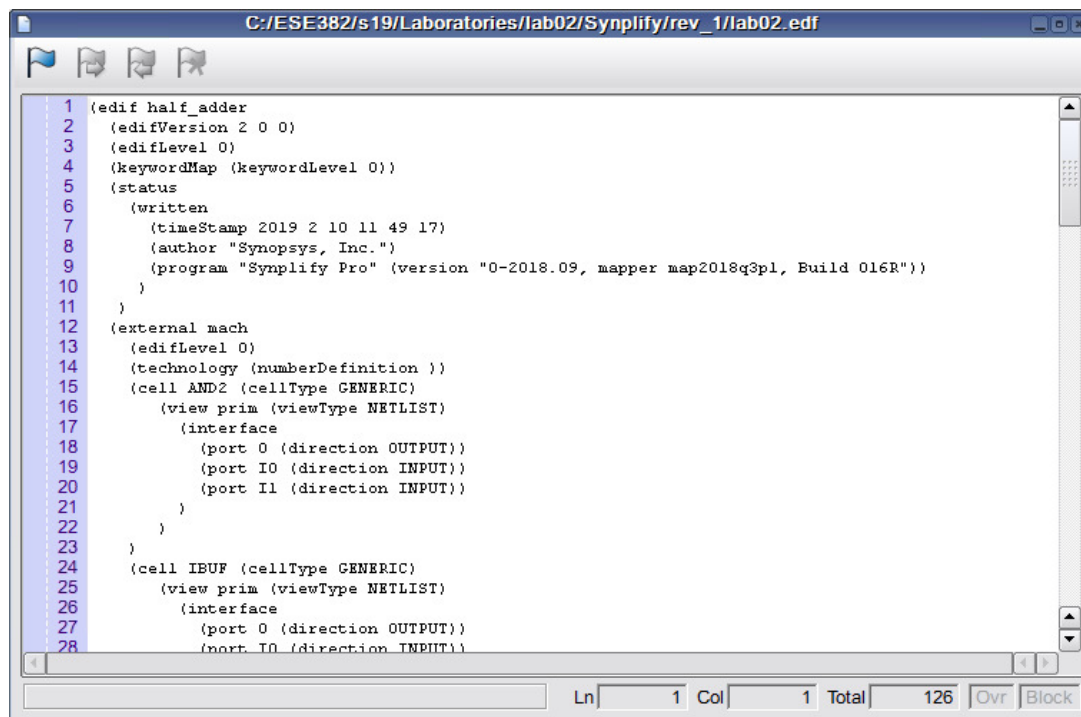
Hostname: WKS-0017

Implementation : rev_1
Synopsys HDL Compiler, Version comp2018q3p1, Build 014R, Built Aug 29 2018

@N: : | Running in 64-bit mode
#####

Copyright (C) 1994-2018 Synopsys, Inc.
```

From the **View** menu select **View Result File**. The EDIF netlist that is used by the place and route tool is displayed. Print this file.



The screenshot shows a web browser window displaying the Synplify EDIF netlist file. The left sidebar shows a tree view with 'Report: lab02 (rev\_1)' selected, and 'lab02.edf' is the active report. The main content area displays the following EDIF code:

```
1 (edef half_adder
2   (edefVersion 2 0 0)
3   (edefLevel 0)
4   (keywordMap (keywordLevel 0))
5   (status
6     (written
7       (timeStamp 2019 2 10 11 49 17)
8       (author "Synopsys, Inc.")
9       (program "Synplify Pro" (version "0-2018.09, mapper map2018q3p1, Build 014R"))
10    )
11  )
12  (external mach
13    (edefLevel 0)
14    (technology (numberDefinition {}))
15    (cell AND2 (cellType GENERIC)
16      (view prim (viewType NETLIST)
17        (interface
18          (port 0 (direction OUTPUT))
19          (port I0 (direction INPUT))
20          (port I1 (direction INPUT))
21        )
22      )
23    )
24    (cell IBUF (cellType GENERIC)
25      (view prim (viewType NETLIST)
26        (interface
27          (port 0 (direction OUTPUT))
28          (port I0 (direction INPUT))
```

From the **File** menu select **Save All**.

Close the project by selecting **Close Project** from the **File** menu.

Exit Synplify by selecting **Exit** from the **File** menu.

Use Windows Explorer to go to the rev\_1 folder in the Synplify folder in folder lab02. Use Notepad to open the file **lab02.vhm** or (**lab02**). This is the VHDL netlist file generated by the synthesizer. It is a VHDL structural description of the logic synthesized by the synthesizer. Print this file.

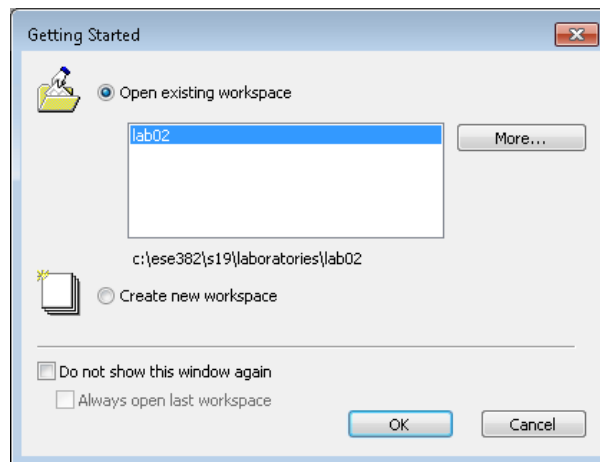
## *Post Synthesis Simulation*

To perform a post synthesis simulation, the UUT used by the testbench must be changed. The UUT in the functional simulation at the beginning of this laboratory was the design entity `half_adder` contained in the file `half_adder.vhd`. This file contained our original VHDL design description of the half adder.

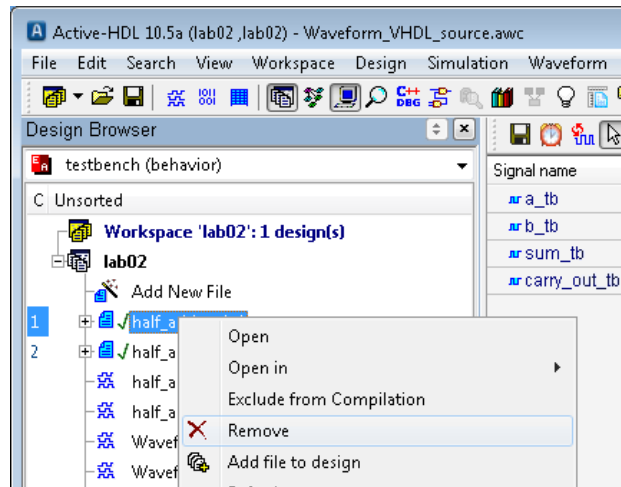
For the post synthesis simulation, we need to use the VHDL netlist model of the design entity `half_adder` that was automatically produced by the synthesizer. This design entity is contained in the file **lab02.vhm** that was automatically generated and stored in the folder `rev_1` in the folder `Synplify`.

We must put the file **lab02.vhm** into our original Active-HDL project in place of the file `half_adder.vhd`.

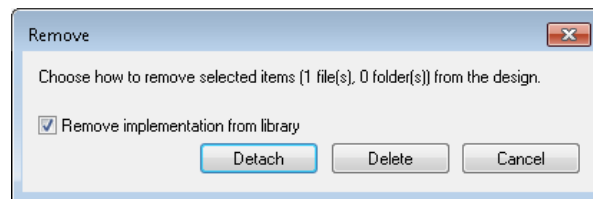
Launch Active-HDL. In the **Getting Started** window check **Open existing design** and select `lab02`. Click **OK**



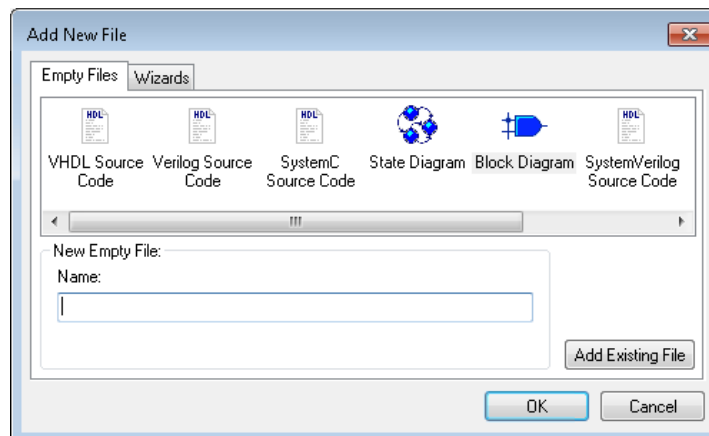
In the **Design Browser**, right click on the file **half\_adder.vhd**. In the pop-up window click **Remove**.



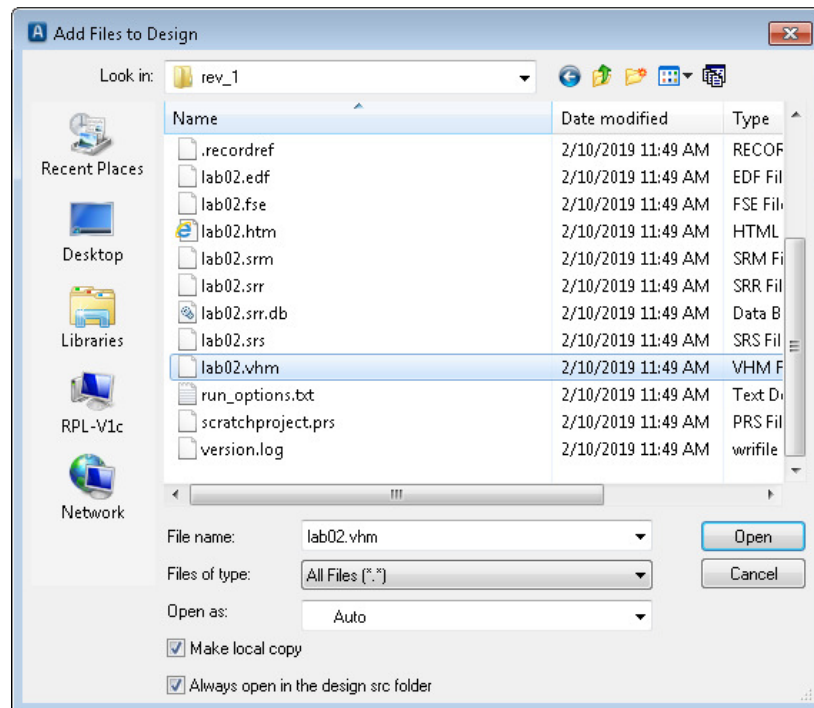
The Remove window opens. **Do not** click Delete. We want to *detach* the file **half\_adder.vhd** from the project, click **Detach**.



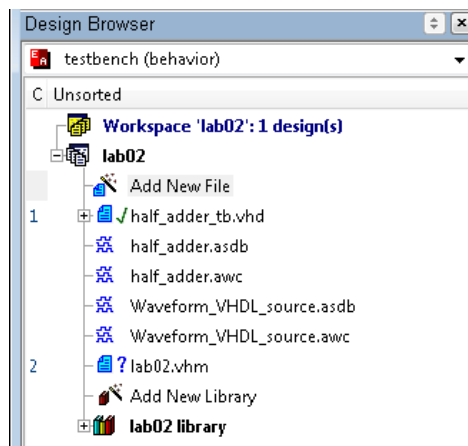
In The Design Browser, double click **Add New File**. The **Add New File** window opens. Click on **Add Existing File**.



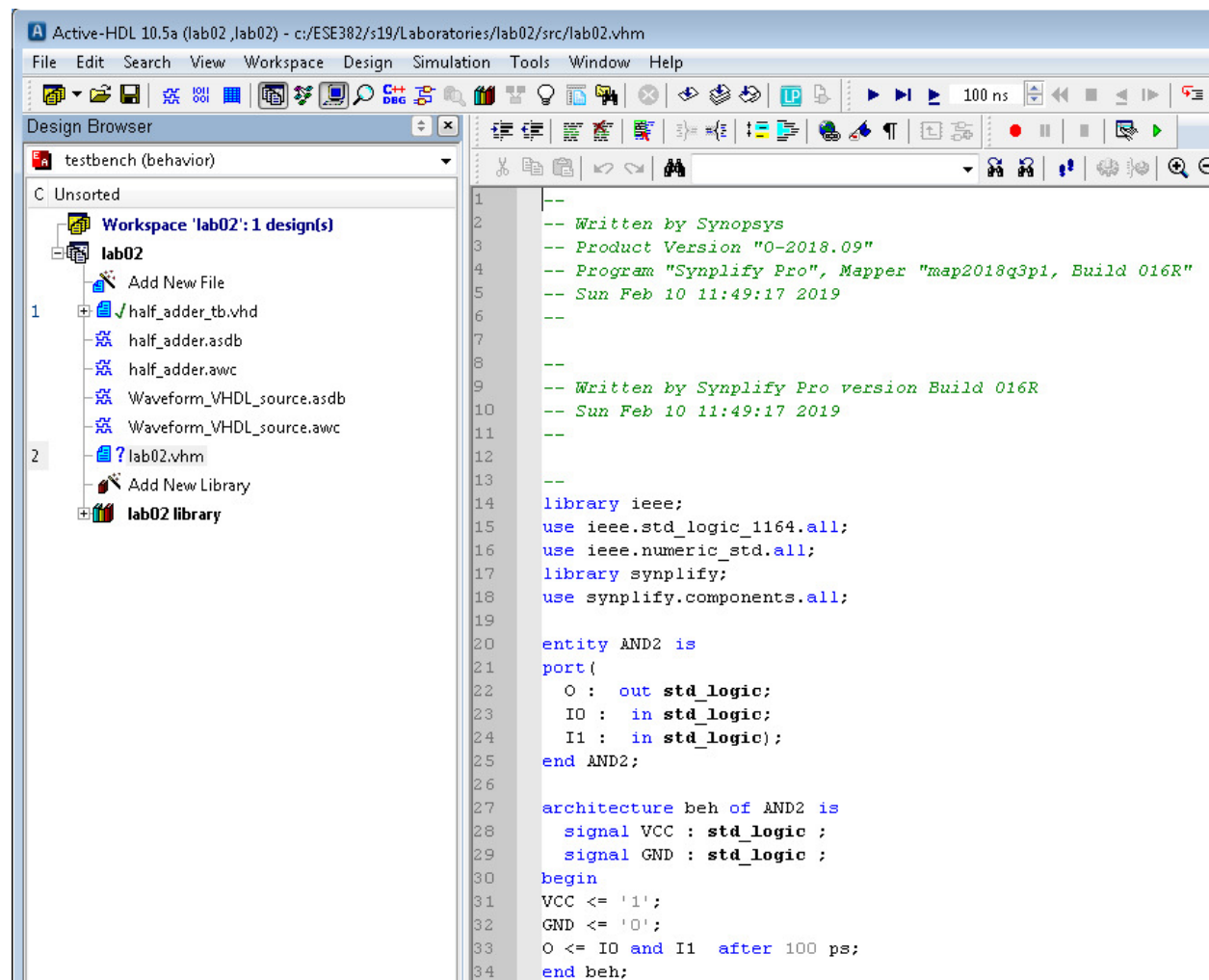
Browse to the folder **rev\_1** in the folder **Synplify** and click on **lab02.vhm**. Then click **Open**.



The file **lab02.vhm** is added to the project.



Double click on the file lab02.vhm. The HDL editor will open. Examine this complete file. This file is a VHDL structural style netlist description of the logic synthesized for the half-adder using logic particular to the ispGAL22V10C-10. r.

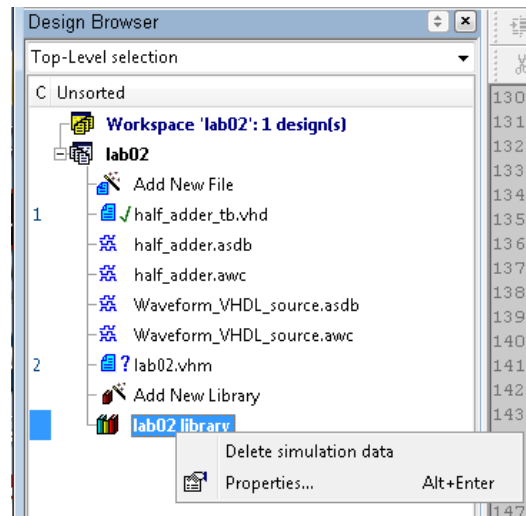


First you will see component entities AND2, IBUF, INV, OBUF, XOR2, defined. These are functionally: a two-input AND, input buffer, inverter, output buffer, and two-input exclusive OR, respectively. The components' architecture bodies use concurrent signal assignment statements. And are, therefore, what we have called dataflow style. At the end of the file you will see the top-level entity half\_adder defined. It has a structural style architecture and uses the form component instantiation where components are declared in the declarative part of the architecture.

Compare the Technology Flattened to Gates schematic with the VHDL netlist (half-adder.vhm) and the Log File, try to match each gate primitive in the schematic to gates listed in the Log File and to the gates instantiated in the VHDL netlist

Right click on the icon labeled lab02 library, at the bottom of the Files tab of the Design Browser. Then click on Delete simulation data. This removes the contents of the working library from the previous compilation.





Click on the Compile All icon

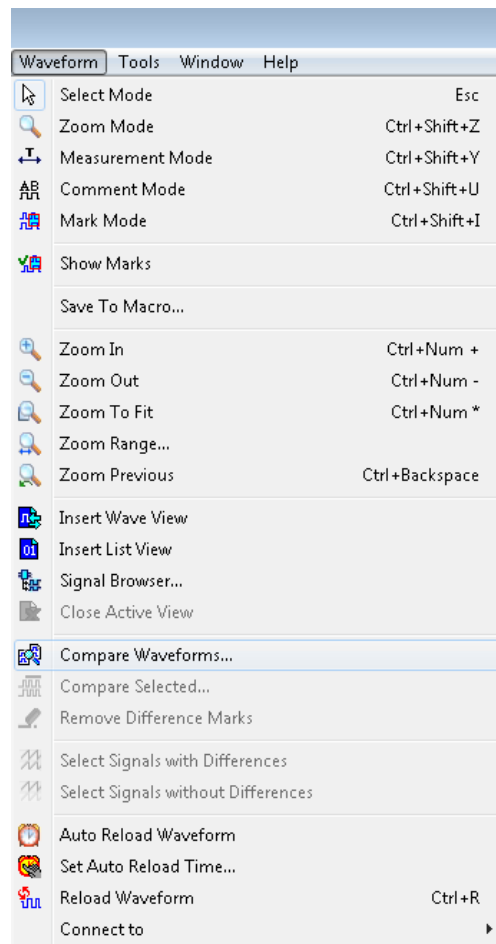


to compile the file **lab02.vhm** and recompile the testbench.

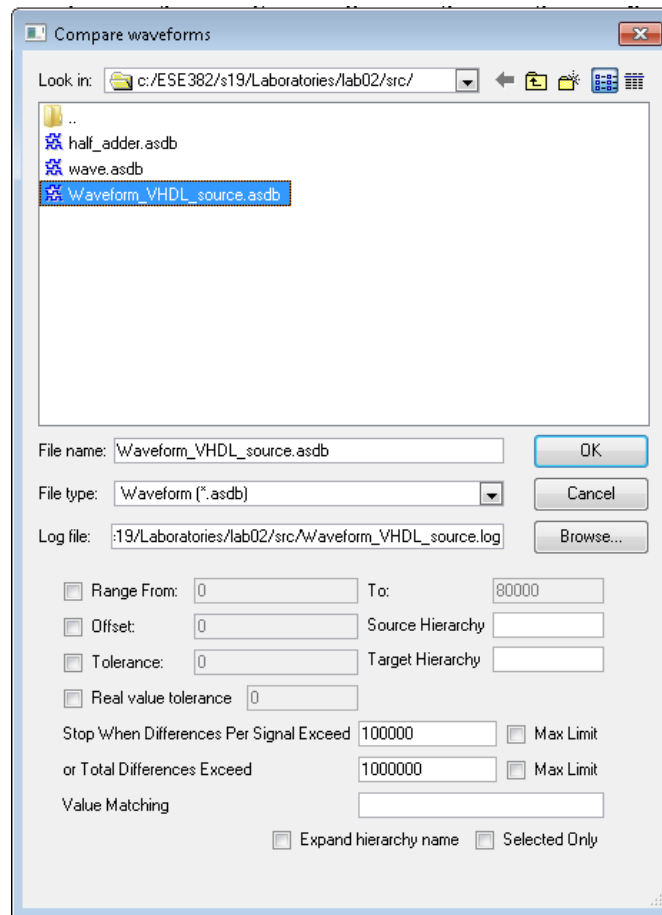
After the compilation completes, click on the + symbol to the right of **half\_adder\_tb.vhd** to expand the tree. Right click on its entity/architecture pair and then click on Set as Top-Level.

Next, select the **Structure** tab at the bottom of the **Design Browser**. Follow the procedure to perform a simulation. Make sure that the signals that you select to appear in the waveforms are **a\_tb**, **b\_tb**, **sum\_tb**, and **carry\_out\_tb**. Set the Run For time to 80 ns before running the simulation. After this simulation is complete, from the **Simulation** menu select **End Simulation**. Because the model of **half\_adder** that we are simulating is the VHDL netlist model generated by the synthesizer, this simulation was a post-synthesis simulation.

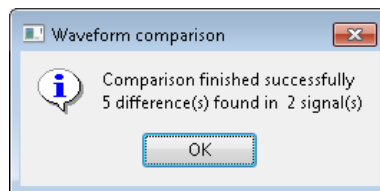
From the **Waveform** menu select **Compare Waveforms**. This feature allows waveforms from two different simulations to be compared.



The **Compare waveforms** window open. Browse to and select waveform Waveform\_VHDL\_source.asdb. This is the waveform you saved from the earlier functional simulation. This waveform should be in the src folder. Click the **OK** button.



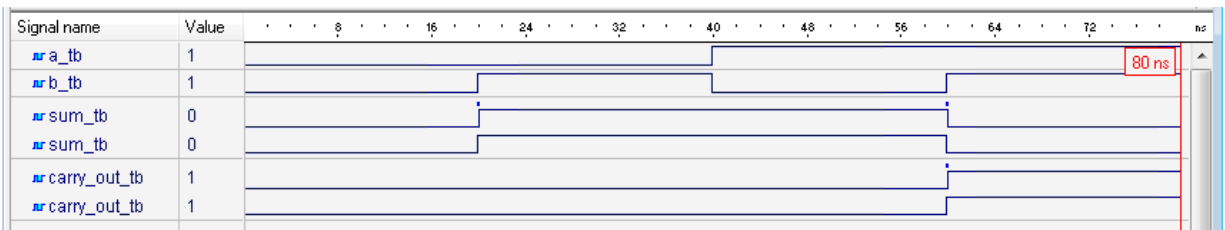
The following window opens.



Click the OK button.

Differences between the output waveforms from the post synthesis simulation and the earlier waveforms from the functional simulation are displayed.

Aldec refers to waveforms displayed in the waveform window just after a completed simulation as the original waveforms. The waveforms that these waveforms are compared with are referred to as the pattern waveforms.



Differences are highlighted by blue bars over the waveforms where they differ. The original waveforms (the post-synthesis simulation waveforms) are displayed in blue and the pattern waveforms (the functional simulation waveforms) are shown in blue. The intervals where two waveforms differ are indicated by a blue bar over the waveforms. There are three differences indicated visually by blue lines for the length of the differences over the edge of the signal transitions. These appear at the start of the transitions of each of the output signals. These differences are not normally expected between a functional simulation and a post-synthesis simulation. They appear here because Synplify has put a 100 ps delay in the model of each of the gates in the file half\_adder.vhm. These are not the actual values of the delays for the gates in the isp22V10, but only nominal delays. Usually post synthesis models do not include delays.

A tolerance can be set in the Open window that appears after a selecting Waveform > Compare Waveforms. This tolerance specifies the minimum time two signals must differ to be considered as an actual difference. If the tolerance were set to 1 ns before the previous waveform comparison, no differences would be indicated.

The Console window also indicates there are differences. However, the Console window says that there are five differences, not the three visible on the waveform.

```

Console
# wavecompare -tolerance 0 -df 1000000 -ds 100000 -offset 0 -o c:/ESE382
c:/ESE382/s19/Laboratories/lab02/src/wave.asdb c:/ESE382/s19/Laboratorie
# 4 signal(s) compared, 5 difference(s) found in 2 signal(s)
# Differences detected, double click on this line to view the log file '
#
>

```

You can click on the line # Differences detected ... to see a listing of the differences.

```

1 Comparing waveforms: "wave.asdb" - "Waveform VHDL_source.asdb"
2 Maximum number of differences per signal: 100000
3 Maximum number of differences: 1000000
4 Ignore glitches: true
5 /testbench/sum_tb from 0fs to 100ps
6 /testbench/sum_tb from 20ns to 20100ps
7 /testbench/sum_tb from 60ns to 60100ps
8 /testbench/carry_out_tb from 0fs to 100ps
9 /testbench/carry_out_tb from 60ns to 60100ps
10 -----
11 4 signal(s) compared, 5 difference(s) found in 2 signal(s)
12

```

How can you explain the discrepancy?

You can learn more about comparing waveforms in Active-HDL by clicking on the Help menu, selecting Active-HDL On-line Documentation, and reading the section on Comparing Waveforms.

This verifies that the synthesized logic is functionally equivalent to the function described in our original design description, because these waveforms are essentially (within the tolerance) identical.

**Have the TA verify and sign off on your simulation.**

## Questions

1. What is the purpose of a post synthesis simulation?
2. What model for the half-adder is used as input in post synthesis simulation and where does it come from.
3. Is the ispGAL22V10C-10 a SPLD, CPLD, or FPGA? Explain the reason for your answer.
4. Does the synthesizer have its own compiler or does it use the compiler from Aldec Active-HDL? Explain the reason for your answer.
5. Is the same testbench used for both the functional simulation of Laboratory 1 and the post synthesis simulation of this laboratory?
6. Do you expect the results from the functional simulation and the post-synthesis simulation of the same design entity to give the same exact results?
7. What coding style is used in the automatically generated post synthesis model?
8. Why are the RTL - Hierarchical view and the Technology - Flattened to Gates schematics produced by HDL Analyst for the half-adder different? Are they logically the same?
9. When the waveforms from the simulation of the original design description and the synthesized logic were compared there were slight differences. Explain the reason for these differences.