

Spring 20, Ken Short

January 31, 2020 4:20 pm

Laboratory 1: VHDL/PLD Design Flow - Compilation and Functional Simulation

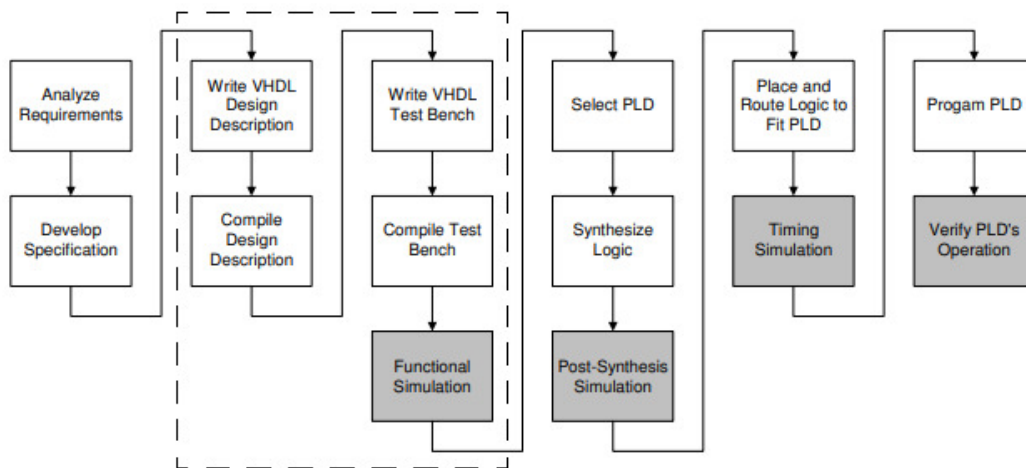
This laboratory is to be performed during your assigned laboratory section the week starting February 2nd.

Prerequisite Reading

1. Sections 1.1 through 1.6 of Chapter 1 of VHDL for Engineers.
2. Design description and testbench source files (Listings 1.3.1 and 1.5.1 of Chapter 1)

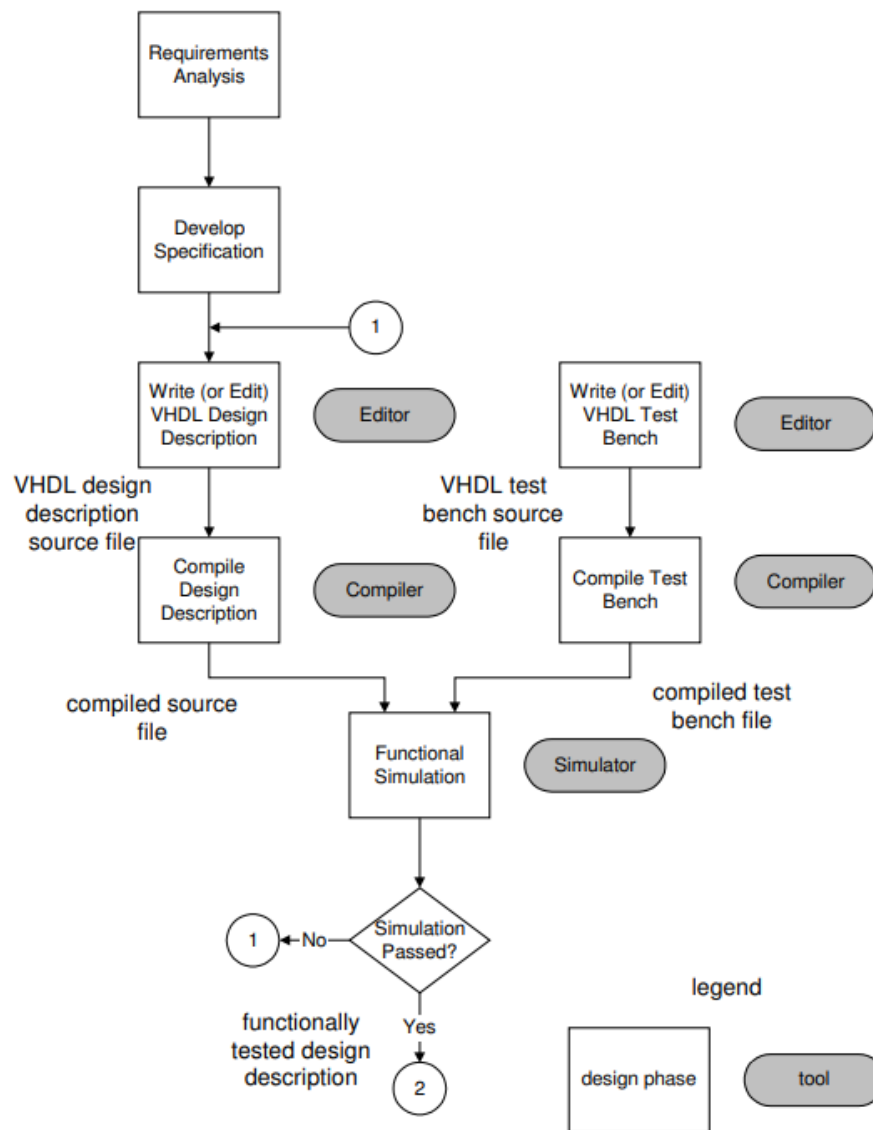
Purpose

The purpose of this laboratory is for you to become familiar with carrying out the steps in the portion of the VHDL/PLD design flow from writing the VHDL design description through its functional simulation. This portion of the design flow is shown inside the dashed rectangle in the figure that follows.



To become familiar with these steps, you will compile and functionally simulate a very simple VHDL design description. The focus is mostly on the design flow steps and not so much on what is being designed. It is assumed that requirements analysis and specification development have led to the need for a half-adder. The half-adder is to be designed and functionally simulated. The purpose of functional simulation is to verify the logical correctness of the VHDL design description. The simulator tool used is Aldec's Active-HDL.

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



After simulation of the half-adder is completed, you will create a second design and repeat the design flow. The second design is for a half-subtractor. You will modify the half-adder files to create the corresponding files for the half-subtractor.

Design Tasks

In future laboratory assignments you are required to carry out design tasks outside of the laboratory prior to your laboratory session. In general, these tasks involve writing a VHDL design description and a VHDL testbench and functionally simulating your design. However, in this laboratory the VHDL code for the half-adder and its testbench are provided for you. You will modify this code to create the half-subtractor.

The half-adder design requires two source files. The first file contains the VHDL design description of the half-adder. For Laboratory 1, you will create this file using Active-HDL's Design Wizard.

The second file is a VHDL testbench. It is the testbench that is actually simulated. This file provides stimulus (input values) to the unit under test (UUT) and checks the outputs produced by the UUT. The UUT in this case is the half-adder design description. Since this is your first laboratory, the testbench file is provided for you. You will simply add it to your design.

The VHDL files (files that you would normally be required to write prior to your laboratory) for this laboratory are Listings 1.3.1 and 1.5.1 of Chapter 1 of VHDL for Engineers.

Examine the design description for the half-adder. Determine what changes are required to make this a design description for a half-subtractor. Mark up a copy of the half-adder design description to produce a half-subtractor design description. The inputs to the half subtractor are *a* and *b*. The outputs are *diff* and *borrow*. You must also change the two concurrent signal assignment statements to compute the difference ($a - b$) and the borrow.

Examine the testbench file for the half-adder. Determine what changes are required to make this a testbench for a half-subtractor. Mark up a copy of the half-adder testbench to create a half-subtractor testbench.

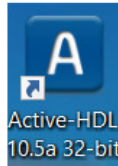
Laboratory Tasks

In the Digital System Rapid Prototyping Laboratory (DSRPL) you will create the VHDL design description file and add (import) the VHDL testbench design file. You will compile the design description and testbench source files and carry out a functional simulation. This simulation is device (PLD) independent. The Aldec Active-HDL Tutorial that follows will walk you through these steps.

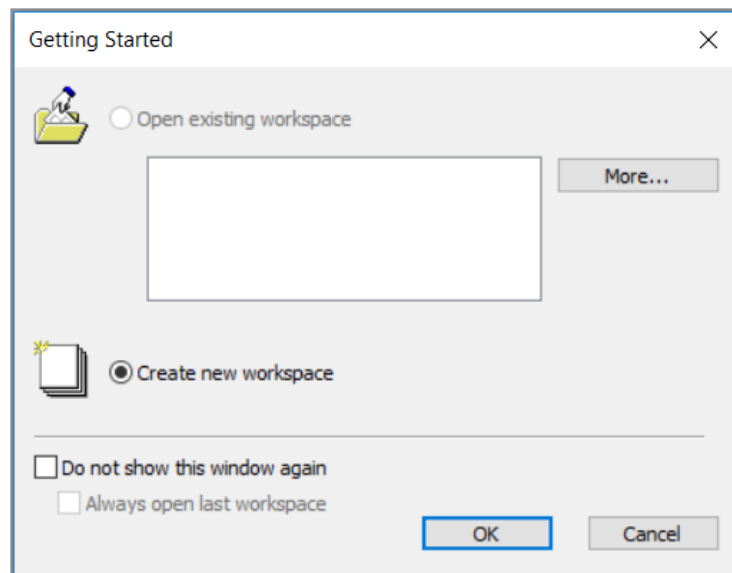
Take time to study the graphical user interface (GUI) for Active-HDL as you go through these design flow steps. Before continuing from one window to the next, carefully examine each window to see what options are available at each step in the design flow. After Laboratory 2, you will be expected to know how to perform all the basic processes in the design flow.

Aldec Active-HDL Tutorial

Log on to your assigned computer in the Digital System Rapid Prototyping Laboratory. From the desktop, start Active-HDL by clicking on the Active-HDL icon.

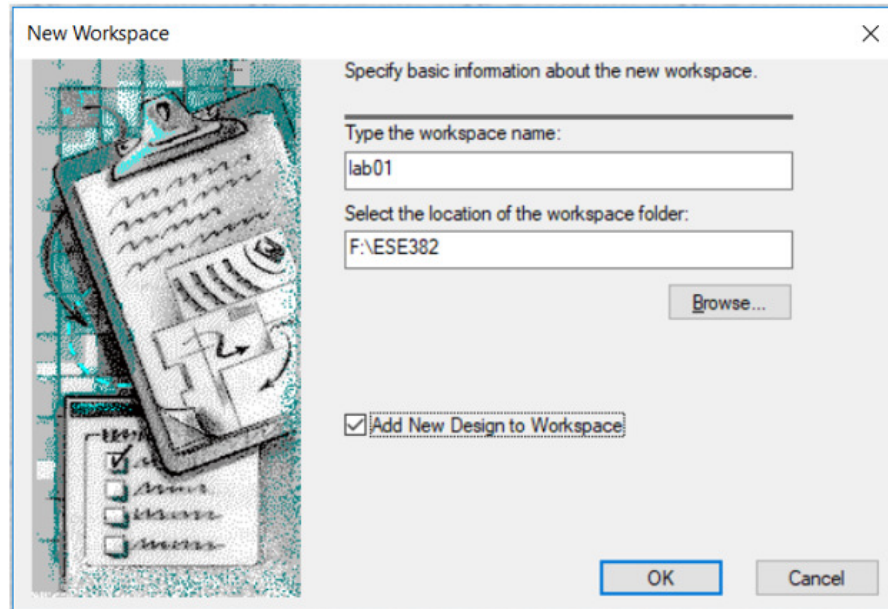


After the loading process completes, the **Getting Started** window appears:

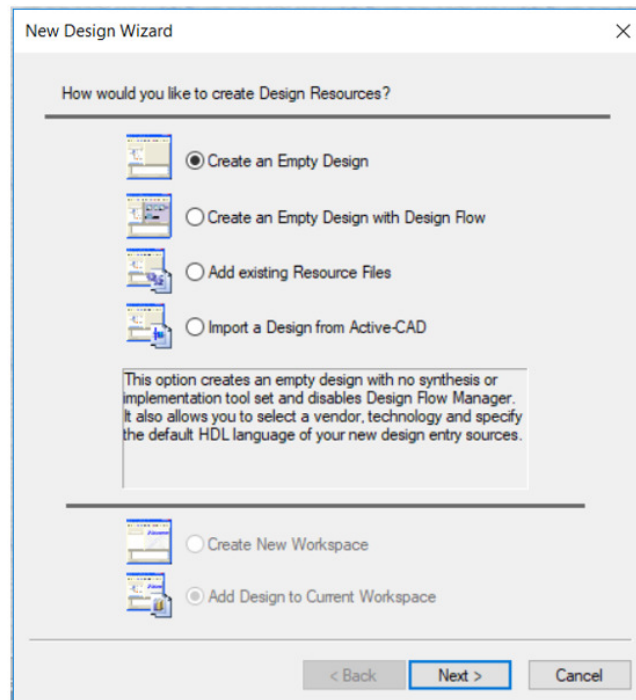


This window lets you either create a new workspace or open an existing one. The box with the list of designs may be empty the first time you log on. Click the **Create new workspace** option and then click the **OK** button.

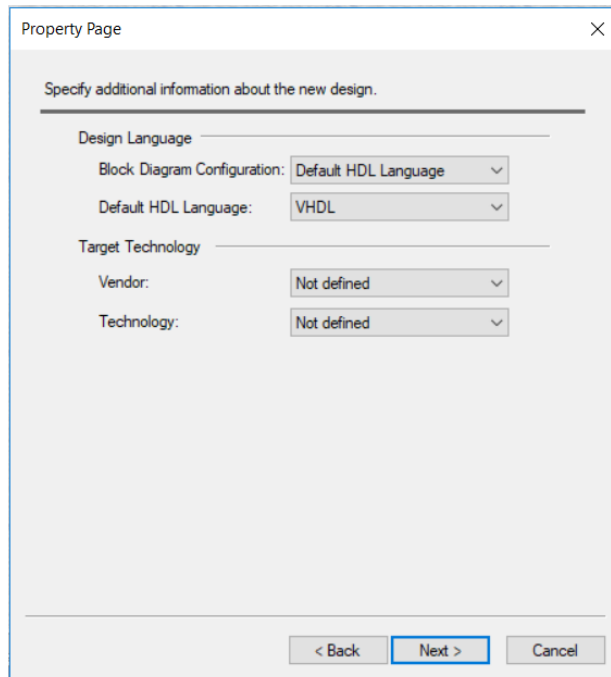
This brings up the **New Workspace** window. In this window, you must specify basic information about the workspace. Enter the workspace name “lab01” in the workspace name text box. In the workspace folder text box, type the location for the workspace folder. You will enter F:\ESE382” in the design folder text box, unless you are told in the laboratory to use a different path. Click the **OK** button.



This brings up the **New Design Wizard**. The **New Design Wizard** asks “How would you like to create design resources.” Check **Create an Empty Design** and click **Next**.

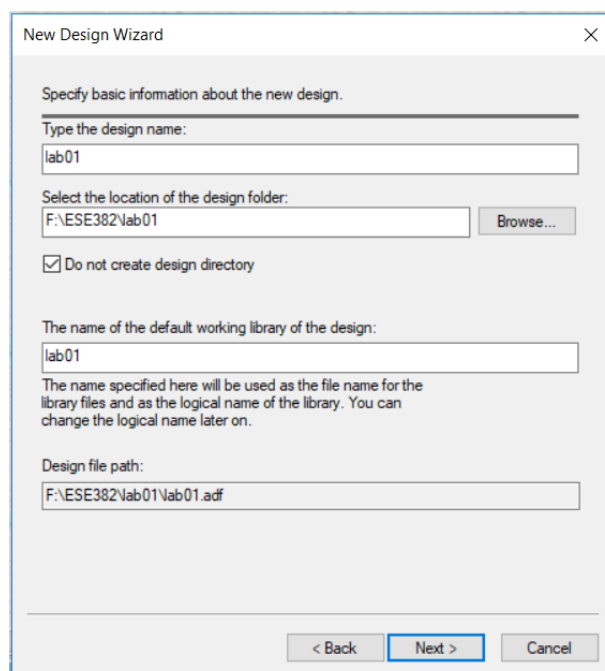


The next window, the Property Page, asks you to “Specify additional information about the new design.” Accept the default values shown and click **Next**.



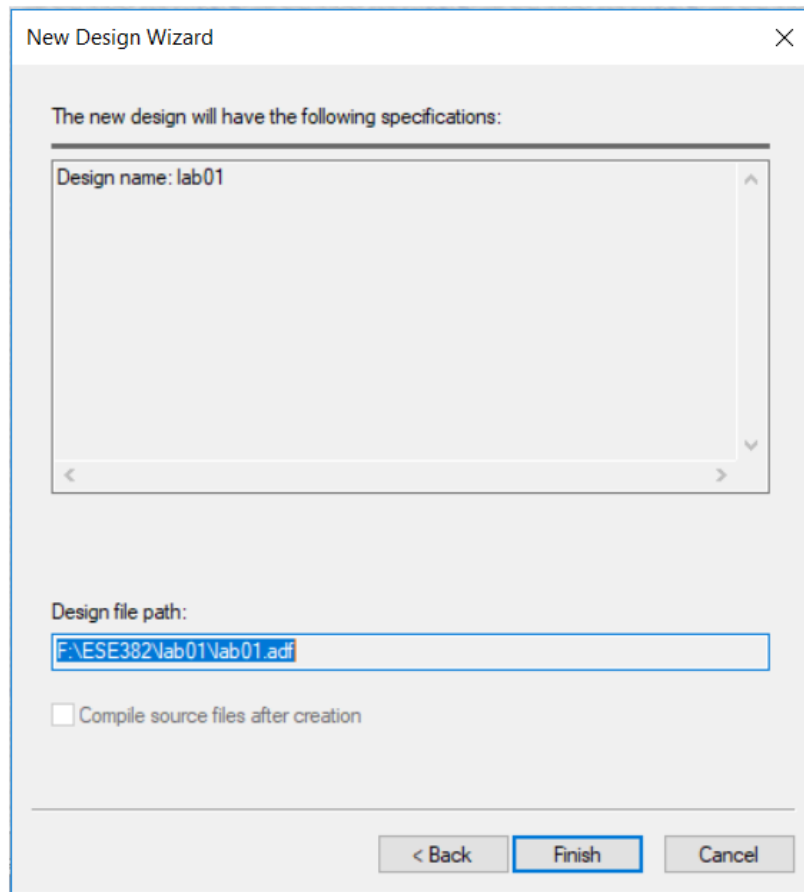
The Property Page dialog box is titled "Property Page" and contains the instruction "Specify additional information about the new design." It is divided into two sections: "Design Language" and "Target Technology". Under "Design Language", there are two dropdown menus: "Block Diagram Configuration:" set to "Default HDL Language" and "Default HDL Language:" set to "VHDL". Under "Target Technology", there are two dropdown menus: "Vendor:" set to "Not defined" and "Technology:" set to "Not defined". At the bottom, there are three buttons: "< Back", "Next >" (highlighted with a blue border), and "Cancel".

In the next window, you must specify basic information about the design. Enter the design name “lab01” in the design name text box. In the design folder text box, type the location for the design folder. You will enter “F:\ESE382\lab01” in the design folder text box. If you used a path different from F:\ESE382 in the New Workspace window on page 5, then use that path with \lab01 appended. Click on “Do not create design directory.” The library text box is automatically filled in. Click the **Next** button.

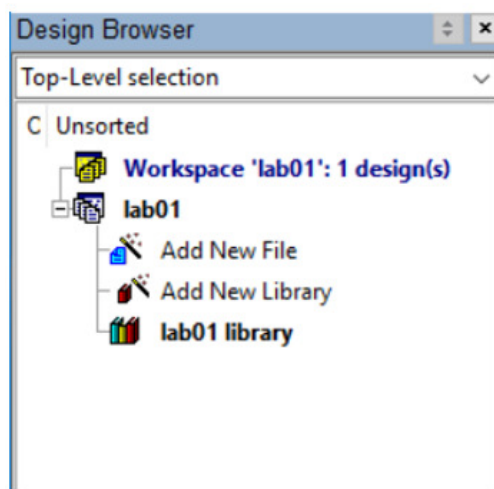


The New Design Wizard dialog box is titled "New Design Wizard" and contains the instruction "Specify basic information about the new design." It has several text input fields and a checkbox. The "Type the design name:" field contains "lab01". The "Select the location of the design folder:" field contains "F:\ESE382\lab01" and has a "Browse..." button next to it. Below this is a checked checkbox labeled "Do not create design directory". The "The name of the default working library of the design:" field contains "lab01", with a note below it stating: "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." The "Design file path:" field contains "F:\ESE382\lab01\lab01.adf". At the bottom, there are three buttons: "< Back", "Next >" (highlighted with a blue border), and "Cancel".

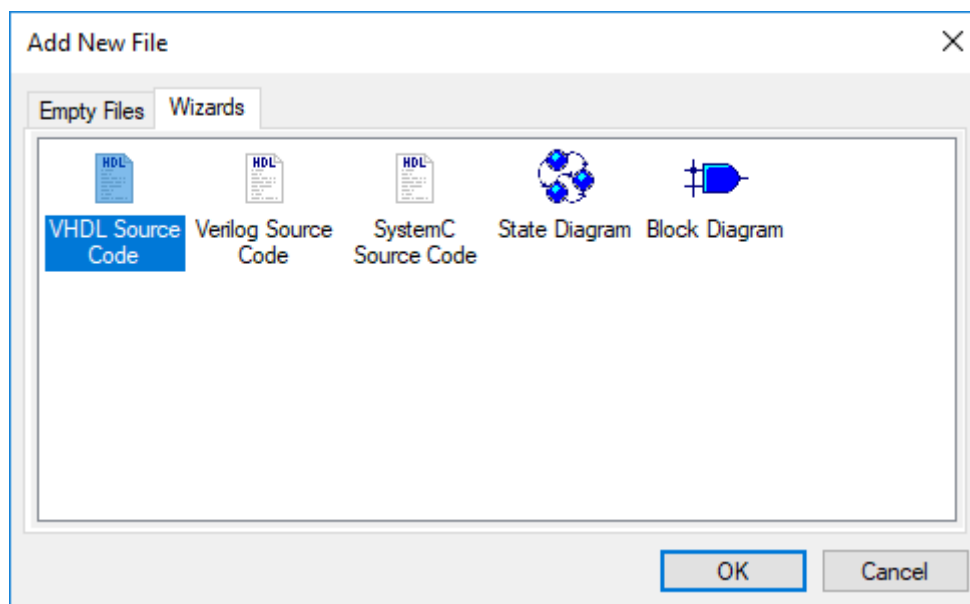
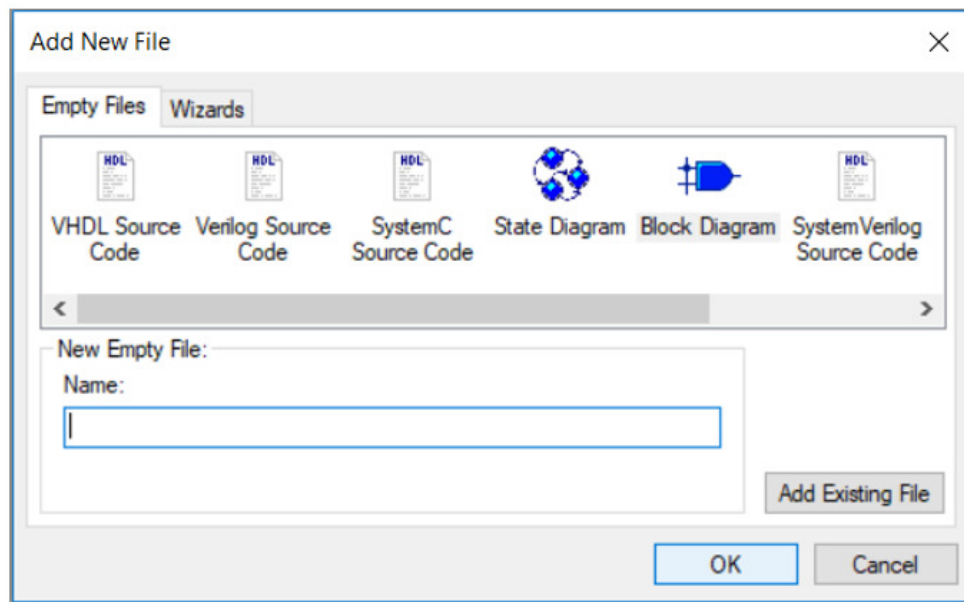
In the last window of the **New Design Wizard**, click **Finish**.



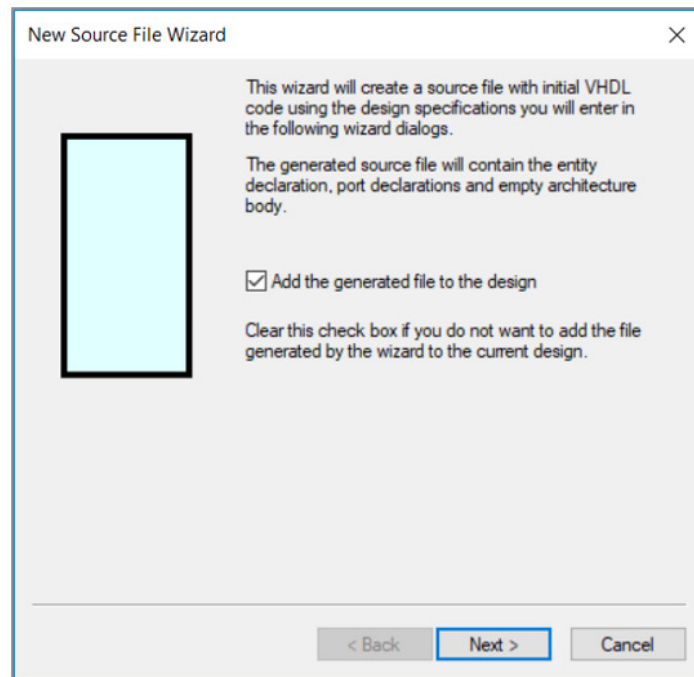
In the Design Browser window, you will see that a new empty design (lab01) has been created along with a working library named after the design (lab01 library).



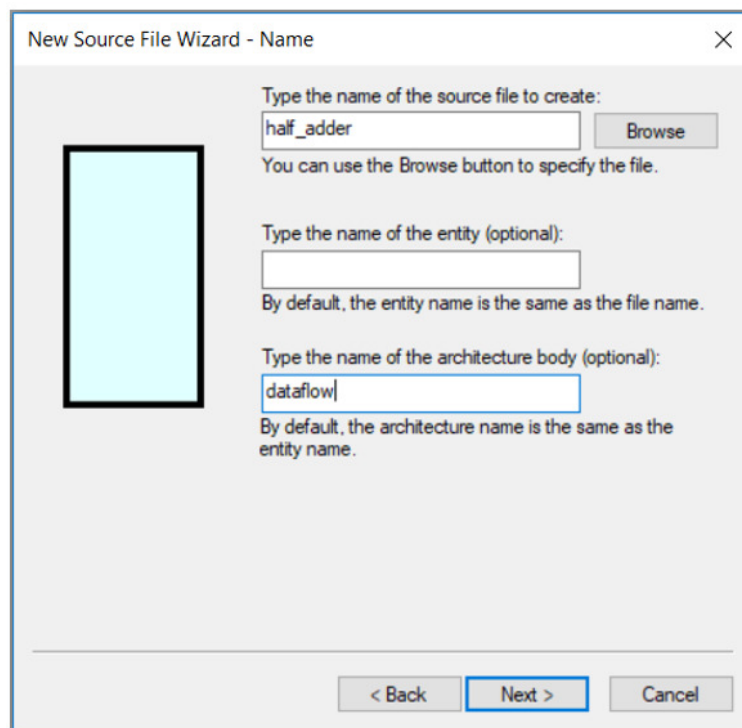
Now you are ready to create project resources. Double click the **Add New File** icon located in the **Design Browser** window. In the **Add New File** window, switch to **Wizards** by clicking on the **Wizards** tab and then double click on the **VHDL Source Code Wizard** icon.



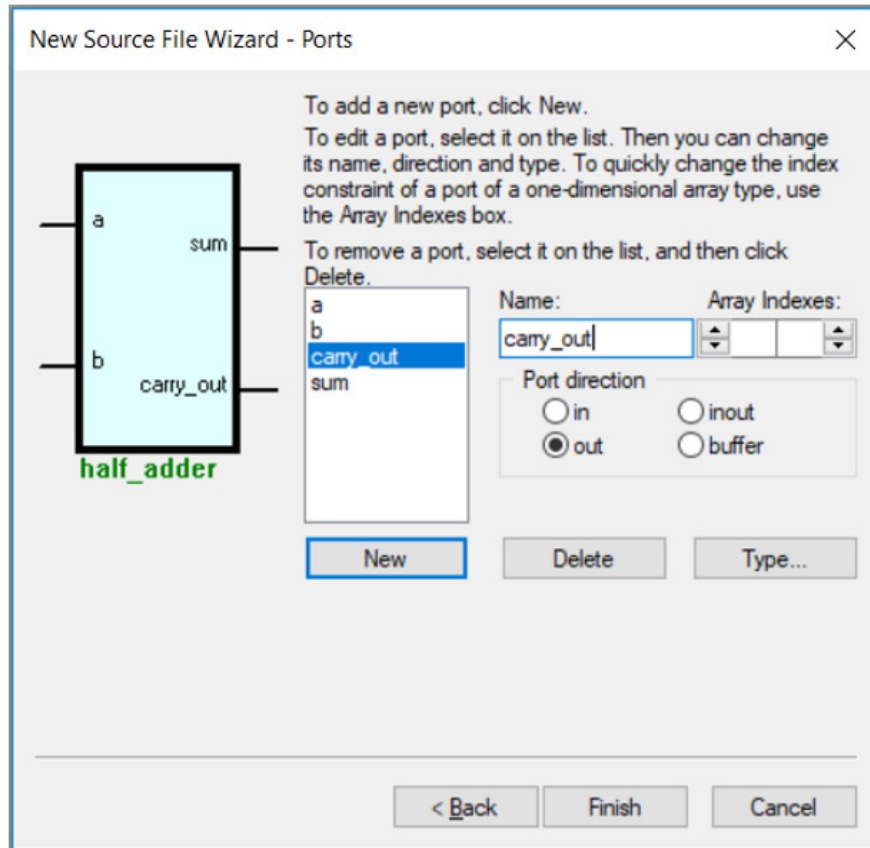
This opens the **New Source File Wizard**. This wizard will guide you through the process to create your design description file. Make sure that the **Add the generated file to the design** box is checked. Click the **Next** button.



In the **New Source File Wizard - Name** window you have to specify the file name for the new source file, and optionally an entity and architecture name. Type in the name `half_adder` for the source file (top text box). Leave the middle text box for the entity name empty. For the architecture body name (bottom text box) type `dataflow`. Click **Next**.



To add ports to the half-adder design entity, click the **New** button and type the port's name in the **Name** field. Select the direction in the **Port direction** box. Click **New** again to add the next port. The half-adder has input ports `a` and `b` and output ports `sum` and `carry_out`. After creating all the ports click **Finish**.



After clicking the Finish button, the automatically generated resource file is opened in the **HDL Editor**.

```
1  -----
2  --
3  -- Title       : half_adder
4  -- Design      : lab01
5  -- Author      : admin
6  -- Company     : SBU
7  --
8  -----
9  --
10 -- File        : F:\ESE382\lab01\src\half_adder.vhd
11 -- Generated   : Tue Jan  8 08:38:44 2019
12 -- From       : interface description file
13 -- By        : Itf2Vhdl ver. 1.22
14 --
15 -----
16 --
17 -- Description :
18 --
19 -----
20
21 --(( Section below this comment is automatically maintained
22 --   and may be overwritten
23 --(entity (half_adder) architecture (dataflow))
24
25 library IEEE;
26 use IEEE.std_logic_1164.all;
27
28 entity half_adder is
29     port(
30         a : in STD_LOGIC;
31         b : in STD_LOGIC;
32         sum : out STD_LOGIC;
33         carry_out : out STD_LOGIC
34     );
35 end half_adder;
36
37 --)) End of automatically maintained section
38
39 architecture dataflow of half_adder is
40 begin
41
42     -- enter your statements here --
43
44 end dataflow;
45
46
```

This file is a template file for the half-adder design description. The entity declaration is complete, but the architecture body has no concurrent statements.

You must complete the architecture body by typing in two signal assignment statements following the comment “-- enter your statements here --.” You may need to scroll down to see this part of the file. The signal assignment statements to be typed in are:

```
sum <= (not a and b) or (a and not b);
carry_out <= a and b;
```

```

library IEEE;
use IEEE.std_logic_1164.all;

entity half_adder is
    port(
        a : in STD_LOGIC;
        b : in STD_LOGIC;
        sum : out STD_LOGIC;
        carry_out : out STD_LOGIC
    );
end half_adder;

--)) End of automatically maintained section

architecture dataflow of half_adder is
begin
|
    sum <= (not a and b) or (a and not b);
    carry_out <= a and b;

end dataflow;

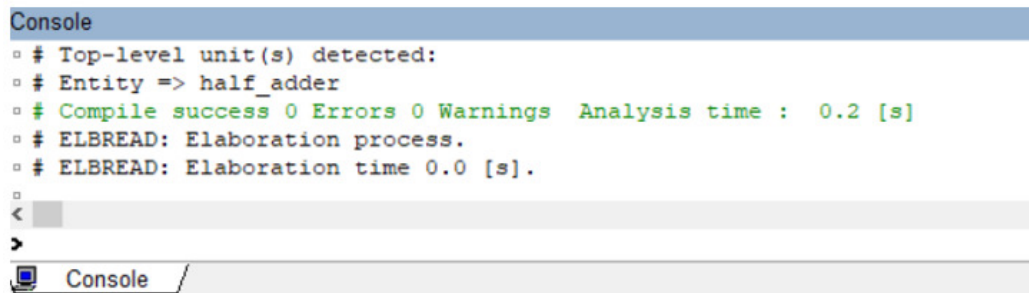
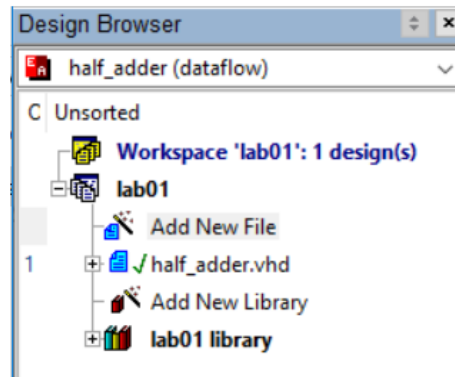
```

After typing in these statements save the file by selecting **Save** from the **File** menu, click on the compile button in the tool bar to compile the file.

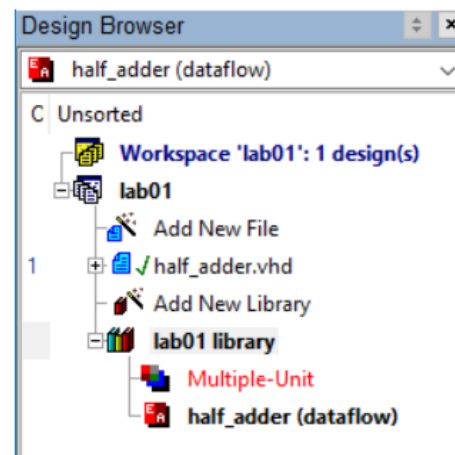


All the buttons in the tool bar have “fly by” capability. If you move the cursor over a button, a legend pops up, at the cursor’s location, stating the button’s function.

A successful compilation is indicated by a green check next to the file name and is appropriately reported in the **Console** window (bottom left of the GUI). There is also a plus sign next to the compiled file's name, allowing branch expansion of the tree structure to show the entity/architecture pair.

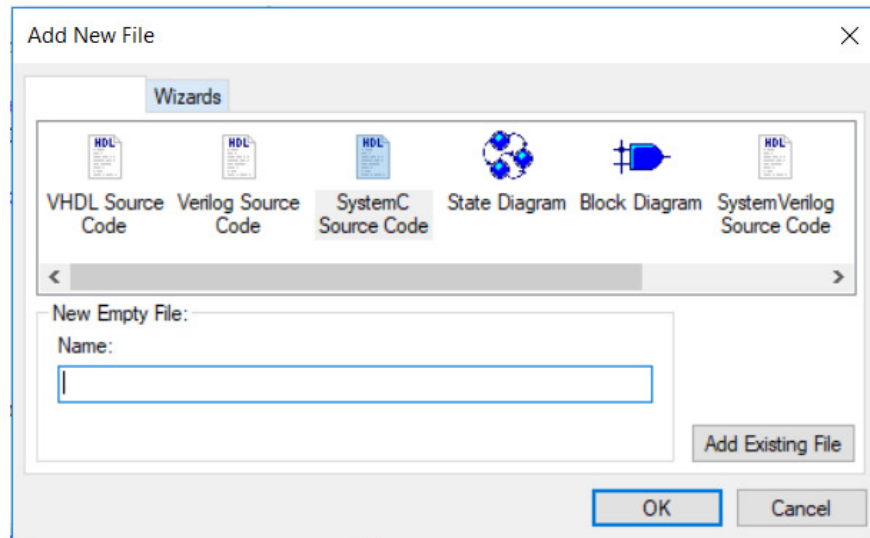


If you expand the “half_adder.vhd” branch you will note that the half-adder entity/architecture pair has been created. If you expand the “lab01” library branch you will note that the compiled half-adder has been automatically attached to the working library.

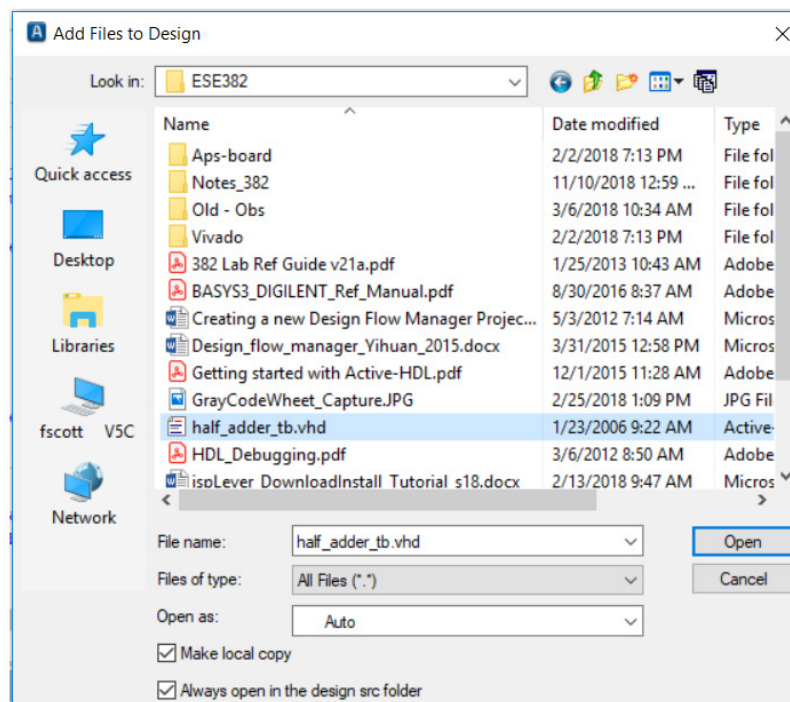


Adding an Existing File to the Project

You now need to add the testbench file to the project. In the **Design Browser** double click **Add New File**. Click the **Add Existing File** button.



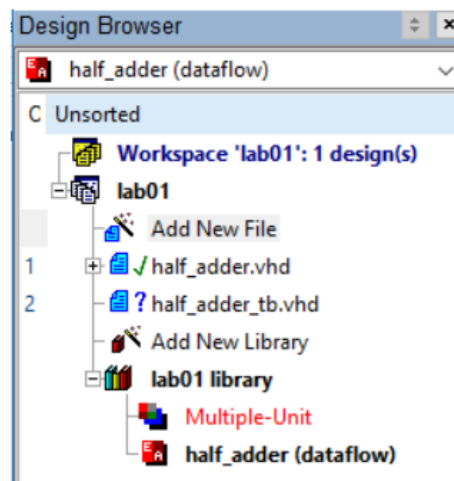
We want to add the file `half_adder_tb.vhd` which is a testbench file for the half-adder.



In the **Add Files to Design** window use the “go up one level icon” to browse to the location of the



testbench file. A copy of this file is located at \\nas_drive\public\ESE382. If you have a copy of this file on your Flash Drive from Blackboard, you can, alternatively, copy it from there. Make sure that **Make local copy** is checked in the **Add Files to Design** window. Select the file “half_adder_tb.vhd.” Click **Add**. This file’s name will appear in the **Design Browser** as part of your project.



Double click on the file “half_adder_tb.vhd” in the **Design Browser** to open the **HDL Editor** and view the contents of the file. Verify that you have added the correct testbench.

```

-- d:\Bkvhdl_progs\Chap01\half_adder\src\half_adder_tb.vhd

library ieee;
use ieee.std_logic_1164.all;

entity testbench is
end testbench;

architecture behavior of testbench is

    -- Declare local signals to assign values to and observe
    signal a_tb, b_tb, sum_tb, carry_out_tb : std_logic;

begin
    -- Create an instance of the circuit to be tested
    uut: entity half_adder port map(a => a_tb, b => b_tb,
        sum => sum_tb, carry_out => carry_out_tb);

    -- Define a process to apply input stimulus and test outputs
    tb : process
        constant period: time := 20 ns;

        begin
            -- Apply every possible input combination

            a_tb <= '0';    --apply input combination 00 and check outputs
            b_tb <= '0';
            wait for period;
            assert ((sum_tb = '0') and (carry_out_tb = '0'))
            report "test failed for input combination 00" severity error;

            a_tb <= '0';    --apply input combination 01 and check outputs
            b_tb <= '1';
            wait for period;
            assert ((sum_tb = '1') and (carry_out_tb = '0'))
            report "test failed for input combination 01" severity error;

            a_tb <= '1';    --apply input combination 10 and check outputs
            b_tb <= '0';
            wait for period;
            assert ((sum_tb = '1') and (carry_out_tb = '0'))
            report "test failed for input combination 10" severity error;

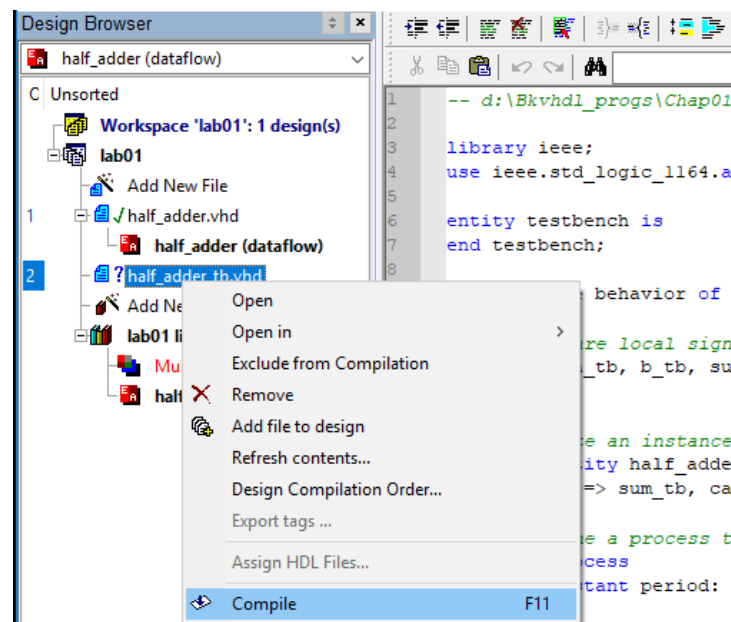
            a_tb <= '1';    --apply input combination 11 and check outputs
            b_tb <= '1';
            wait for period;
            assert ((sum_tb = '0') and (carry_out_tb = '1'))
            report "test failed for input combination 11" severity error;

            wait;    -- indefinitely suspend process

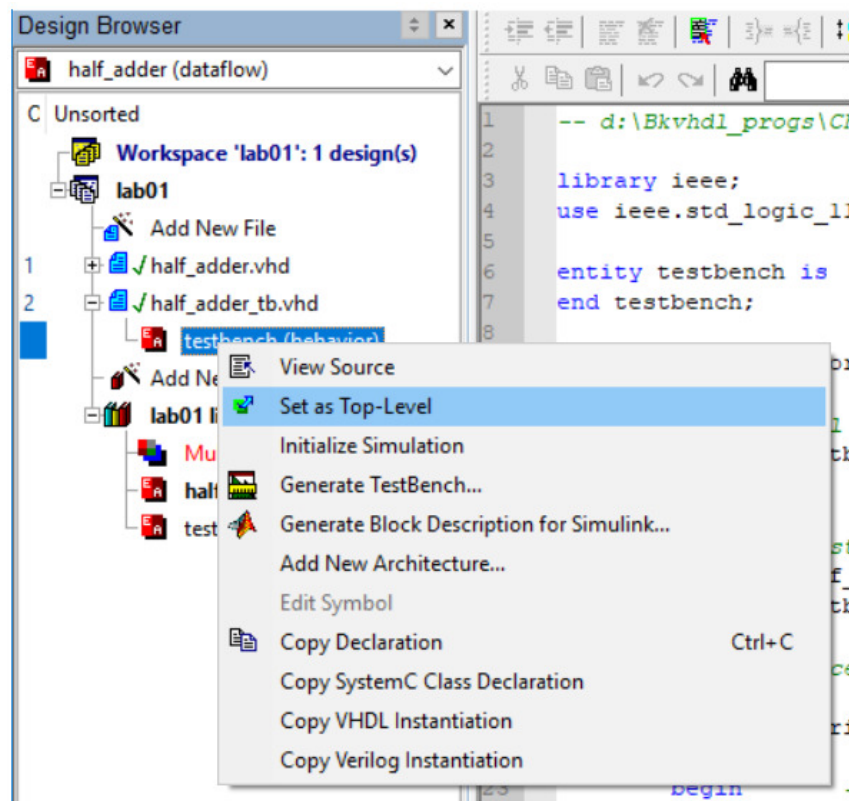
        end process;
end;

```

The question mark to the left of the file's name in the Design Browser indicates that the file has not been compiled. Right click on the file "half_adder_tb.vhd." and in the pop up menu, click **Compile**. This compiles the testbench.



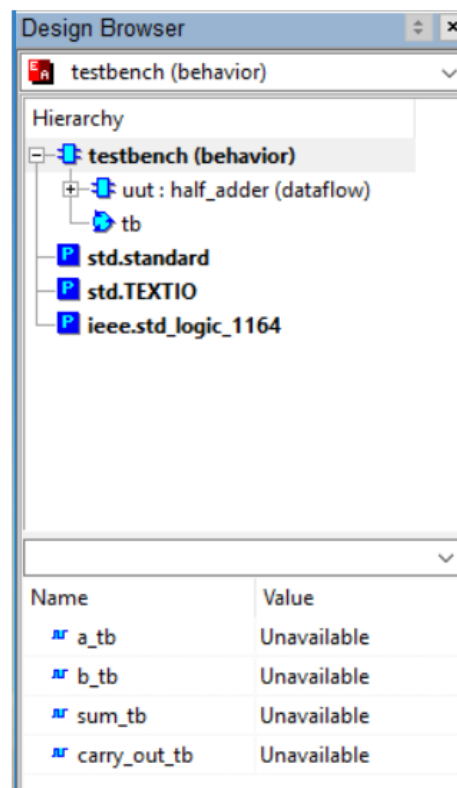
In the Design Browser, click on the + sign to the left of the file name “half_adder_tb.vhd.” This expands the tree to make the entity/architecture pair (E/A) “testbench(behavior)” visible. Right click on this E/A and in the context menu click **Set as Top-Level**.



By telling the simulator that the “half_adder_tb.vhd” is the top level you are specifying that the testbench is what is to be simulated. When the simulator builds the simulation model before the simulation it will start with the testbench entity. The testbench entity instantiates the half_adder entity and this results in the half_adder being simulated indirectly. If you don’t set the testbench as the top level, then when you try to run the simulation nothing will happen because no stimulus inputs will be generated.

Viewing Local Data

At the bottom of the **Design Browser** are three tabs. Click the **Structure (Stru...)** tab. This opens the structure window in the **Design Browser**. The structure of your project is displayed as a tree in this window. Click on the + sign next to “testbench (behavior)” to expand the branch to show “uut” and “tb.”



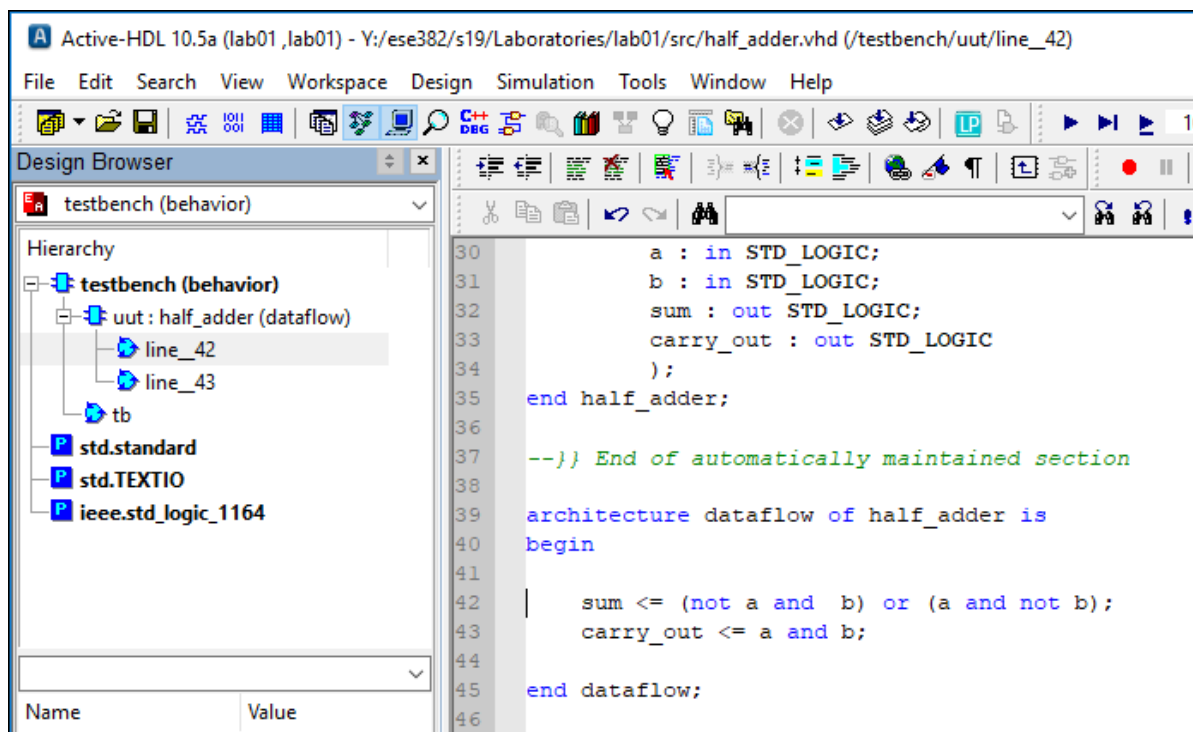
Read and examine the following carefully!

The blue rectangles represent the components of the project. The two components are the testbench(root) design entity and the UUT (half-adder design entity).

The file “half_adder_tb.vhd” that you added to the project contains the design entity (entity/architecture pair) named “testbench.” After adding this file, you made “testbench” the Top-Level of the design. The structure tree shows “testbench” as the Root or Top-Level. If you follow the vertical

line down from the component symbol for “testbench” you will see a circle symbol with three arrows. This symbol represents a process name `tb`. Simulation processes are created by the simulator from the design files. It is the simulation processes that are executed during a simulation. If you double click this symbol, the file “half_adder_tb.vhd” will be opened in the HDL Editor window (if it is not already opened) and the cursor will be at the start of the process `tb`.

The other component in this project is the UUT, the design entity `half_adder`. This design entity is defined in the file “half_adder.vhd” that you created initially. Click the + sign next to “half_adder” component. This expands the subtree under “half_adder” to reveal two simulation processes specified in terms of their line numbers in the file. Double click on the first of these two processes and the file “half_adder.vhd” is opened in the HDL Editor window with the cursor at the start of this simulation process. This simulation process corresponds to the single VHDL concurrent signal assignment statement to the right of the cursor, the assignment statement for `sum`. In the Design Browser window, double click on the second of the two simulation processes. In the HDL Editor window, the cursor moves to the start of this simulation process. This simulation process corresponds to the single VHDL concurrent signal assignment statement for `carry_out`.

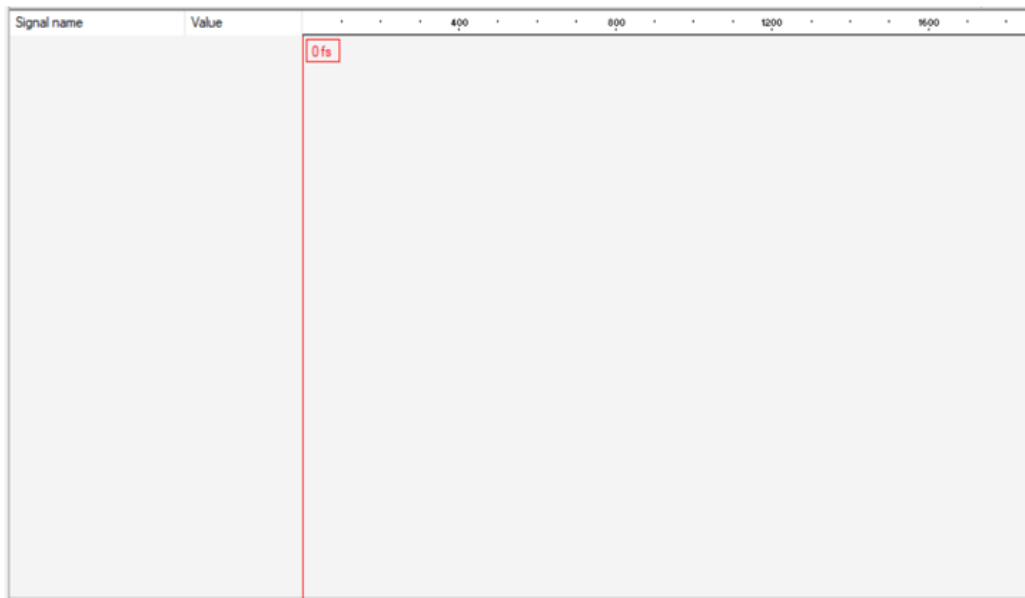


Functional Simulation

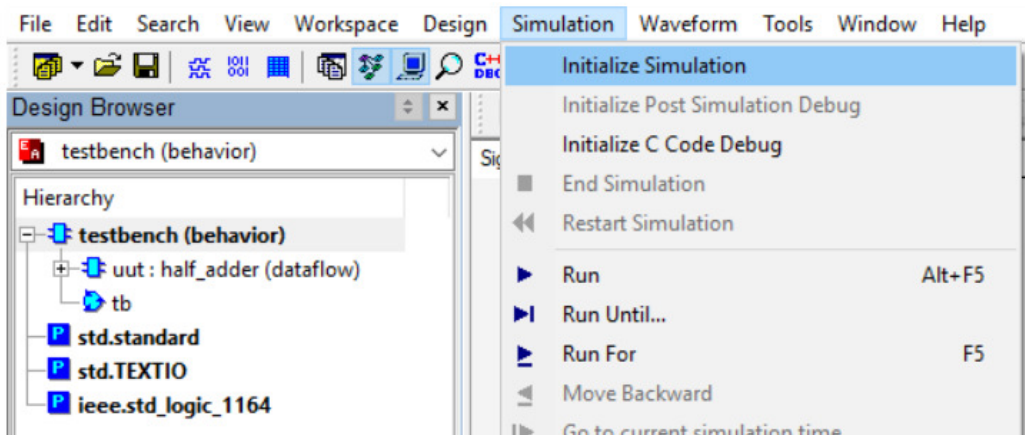
To set up a simulation, you have to open a new Waveform window. Click the **New Waveform** toolbar button .



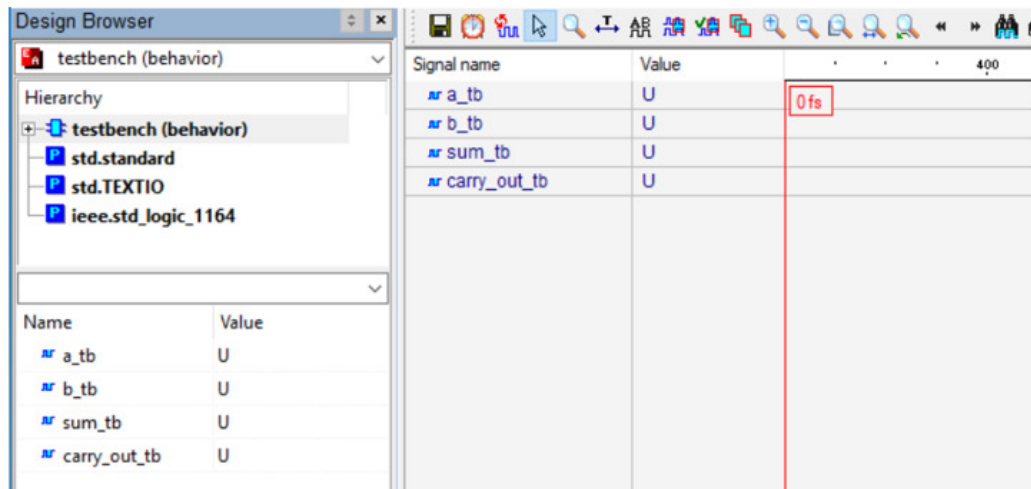
The new Waveform window appears. As previously stated, the icons in the tool bar have a “fly by” feature, so that, if you move the cursor over an icon and leave it there for an instant, a box will pop up with the icon’s name.



Before you start the simulation, you must initialize the simulation and add the signals that you wish to observe to the simulator window. From the menu select Simulation and click on Initialize Simulation.



In the **Design Browser**, select the Top-Level design with the Root attribute. With the cursor over the Root attribute, hold down the left mouse button and drag the cursor to the upper left side of the Waveform window, then release the mouse button. The names of all the signals associated with the testbench appear in the Waveform window.



To the right of the **Run For** button in the tool bar is a box where you can select a period of time for the simulation to run:



Change the value to 80 ns. Click on the **Run For** button:

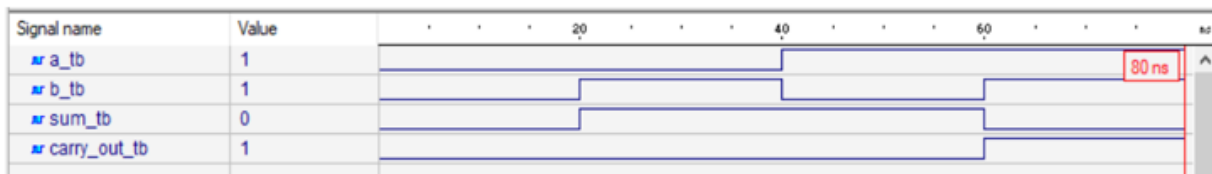


The simulation runs for 80 ns. The waveforms produced appear in the Waveform window.

Click on the **Zoom to Fit** button in the Waveform window tool bar. You will now see the complete



simulation waveforms in the Waveform window.



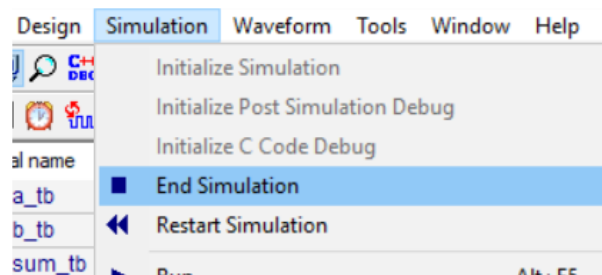
The signals displayed are the signals that connect the half-adder to the testbench. These signals were declared in the testbench architecture, as local signals, and each has the suffix `_tb`. Note that in the waveforms, when an input changes (either `a_tb` or `b_tb`) the output changes immediately (with no delay). This happens because the UUT that we are simulating is the VHDL design description and such a model does not specify any propagation delays for the half-adder.

Observe that the output waveforms correspond functionally to what you expect for a half-adder from the half-adder truth table. Using the mouse cursor, drag the vertical cursor in the Waveform window to time = 0. Slowly drag the vertical cursor to the right. Observe that the values in the Value column to the right of the signal names change to indicate the values of the signals at the particular point in time corresponding to the cursor position.

Since the testbench applied every possible input combination to the half-adder, and the corresponding outputs were observed to be correct, you have functionally verified your design for the half-adder.

Have the TA verify and sign that you have correctly simulated the half-adder.

From the **Simulation** menu select **End Simulation** to stop simulator execution.



Simulating an Incorrect Design File

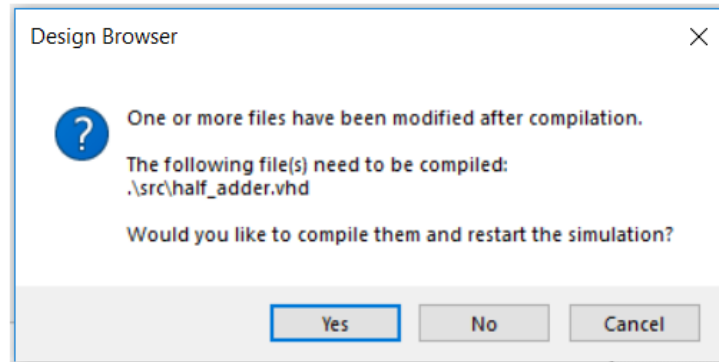
Click on the **Files** tab at the bottom of the Design Browser. In the Design Browser window double click the file “half_adder.vhd.” This opens the HDL Editor. Change the second assignment statement from:

```
carry_out <= a and b;
```

to

```
carry_out <= a or b;
```

Click on the **Waveform** tab at the bottom of the main window to bring the waveform viewer back. Initialize the simulation. A window will open indicating that a file has been changed. Click Yes to recompile and restart the simulation.



You may need to re-add the signals to be observed to the waveform viewer. If so, in the **Design Browser**, select the Top-Level design with the Root attribute. With the cursor over the Root attribute, hold down the left mouse button and drag the cursor to the upper left side of the Waveform window, then release the mouse button. The names of all the signals associated with the testbench appear in the Waveform window.

Run the simulation for 80 ns. From examination of the waveforms in the waveform viewer you can see that the design is not functionally correct. However, since the testbench was written to be a selfchecking testbench, you can also see the errors listed in the Console window.

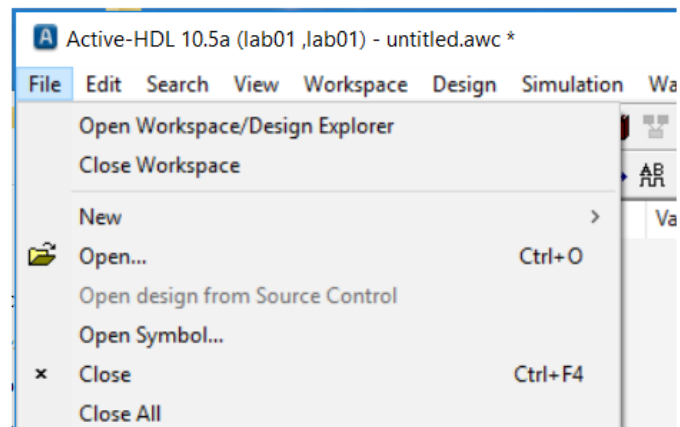
Examine the results in the Console window.

```
Console
▫ run 80 ns
▫ # EXECUTION:: ERROR : test failed for input combination 01
▫ # EXECUTION:: Time: 40 ns, Iteration: 0, Instance: /testbench, Process: tb.
▫ # EXECUTION:: ERROR : test failed for input combination 10
▫ # EXECUTION:: Time: 60 ns, Iteration: 0, Instance: /testbench, Process: tb.
▫ # KERNEL: Simulation has finished. There are no more test vectors to simulate.
▫
```

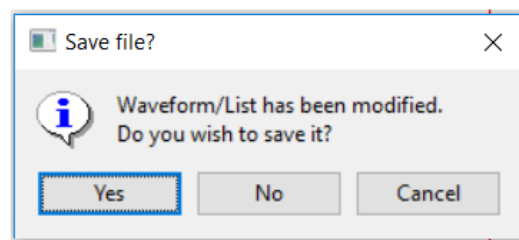
End the simulation.

Go back and correct the change that you made to the file `half_adder.vhd`. Carry out another simulation run to verify your correction. Stop the simulation.

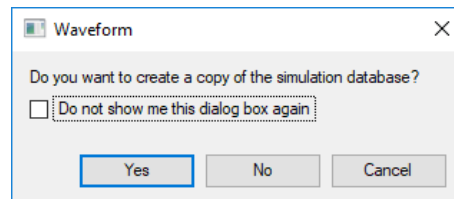
From the **File** menu select **Close Workspace** to save your design.



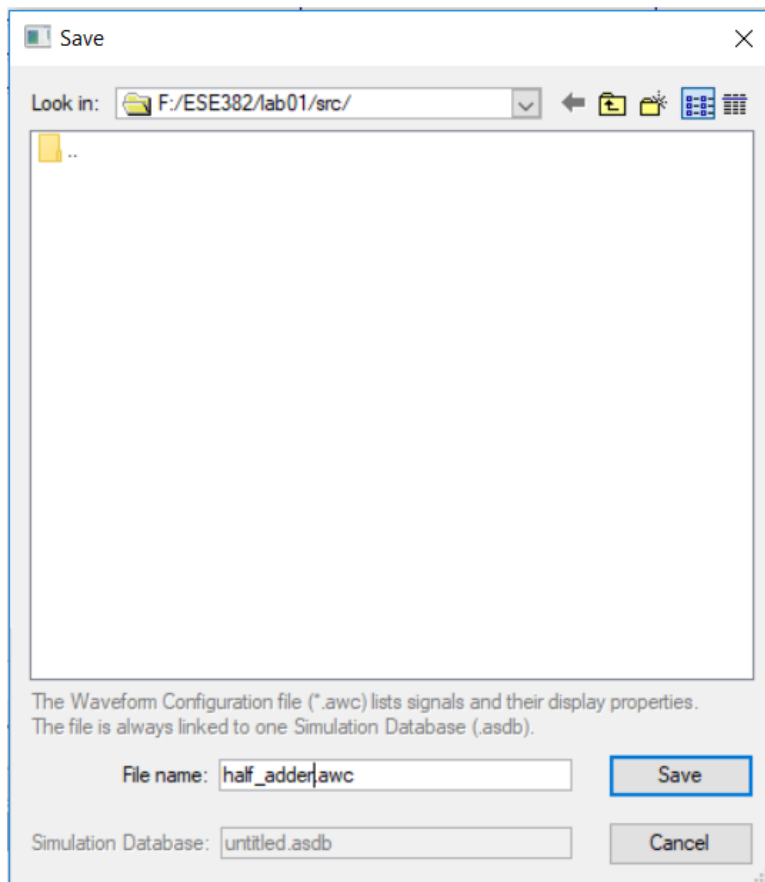
A window will open asking if you want to save the waveform file. Click on Yes.



A window will open asking if you want to create a copy of the simulation database? Click Yes.



The Save window will then open. For filename type `half_adder.awc`. Then click Save



Half-subtractor Design Entity

Create a new design named lab01b. You will do this by repeating the steps that you did for the half_adder design up to where in the **Add New File** window you used the VHDL Source Code Wizard to generate the file for the half_adder.

Instead, in the Design Browser click on **Add New File**. In the window that opens, make sure that **Make local copy** is checked. Then add both the design files from project lab01, these files will be found in the subfolder src in the folder lab01.

In the **Design Browser**, right click on the file half_adder.vhd. Click on **Rename**. Change the file's name to half_sub.vhd.

Use this same process to rename the file half_adder_tb.vhd to half_sub_tb.vhd.

Double click on the file half_sub.vhd in the **Design Browser**. This opens the **HDL Editor**. Edit the entity declaration and architecture body so that they correspond to a half-subtractor. The inputs remain a and b. The outputs' names must be changed to diff and borrow. You must also change the two concurrent signal assignment statements to compute the difference (a - b) and the borrow.

In a similar fashion modify the testbench file half_sub_tb.vhd to make it an appropriate testbench for a half-subtractor.

Compile both files.

Functionally simulate your half-subtractor design.

Have the TA verify and sign that you have correctly simulated the half-adder.

Questions

1. What is a design entity?
2. What is the purpose of an entity declaration?
3. What is the purpose of an architecture body?
4. List the name of each design entity in the file `half_adder.vhd`. Also give the names of the inputs and outputs declared in each entity declaration.
5. List the name of each architecture body in the file `half_adder.vhd`.
6. List the name of each entity in the file `half_adder_tb.vhd`. Also give the names of the inputs and outputs of each entity.
7. List the name of each architecture body in the file `half_adder_tb.vhd`.
8. How is the design entity `half_adder` used in the testbench `half_adder_tb.vhd`?
9. What kind of model of the half adder is used in the testbench for this simulation?
10. What is the purpose of a functional simulation?