

TUTORIAL ON USING XILINX ISE DESIGN SUITE 14.6: Behavioral Simulation of a “Half Adder” Circuit

Shawki Areibi

September 8, 2024

1 Introduction

The simulator is a tool that allows us to test a circuits using software. Traditionally, a digital circuit had to be built on a breadboard using TTL chips in order to test for logical correctness. This had many drawbacks. It was costly to equip a lab with enough hardware for a class full of students. The cost was augmented further because many chips where hooked up incorrectly and destroyed. Hookup errors often occurred which were hard to detect on a breadboard. Software design solves these problems and allows designs to be re-used and built upon.

The objective of this tutorial is to familiarize the student with the Xilinx ISE Design Suite 14.6 ISim Simulator. In this tutorial you will learn the following topics:

1. How to use the different capabilities of the simulator.
2. How to create bench test (VHDL) to be used by the simulator.
3. How to verify the functionality of your design (Half Adder) using the Xilinx ISim Simulator.

2 Getting Started

The following sections outline the requirements for performing behavioral simulation.

Required Files: The behavioral simulation flow requires Design Files, A Test Bench File, and Xilinx Simulation Libraries.


1. **Design Files** (VHDL or Schematics): This tutorial assumes that you have completed the design entry tutorial in either **Schematic-Based Design** or **VHDL-Based Design**. After you have completed one of these, your design includes the required design files and is ready for simulation.
2. **Test Bench File:** To simulate the design, a bench file is required to provide stimulus to the design. A VHDL test bench file is available with this tutorial. You may also create your own test bench file.
3. **Simulation Libraries:** Xilinx simulation libraries are required when a Xilinx primitive or IP core is instantiated in the design. Even though we do not use any IP cores or any Xilinx primitives (i.e., digital clock manager) it is useful to know that these libraries are essential for future simulations since you might be using some IP core in your design.

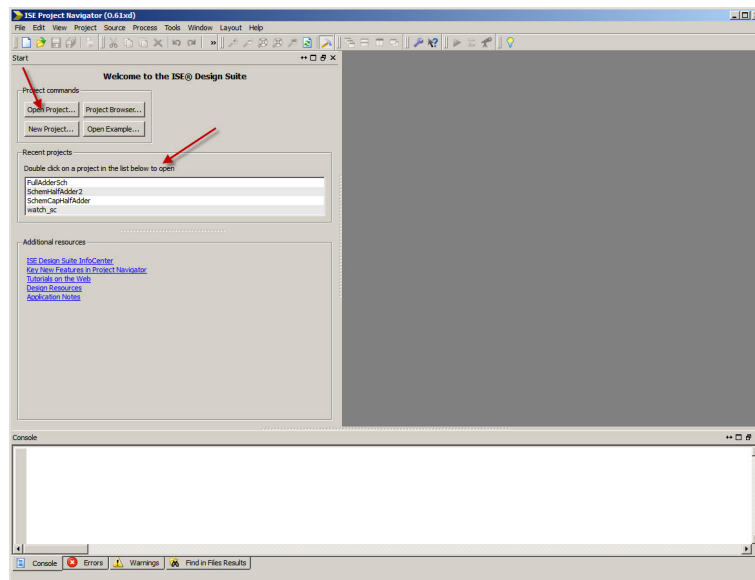
3 Behavioral Simulation Using ISim

In previous tutorials you learned how to enter your design using schematic capture for both a Half Adder and a Full Adder circuit. In both cases a schematic is available in the project. Here are the steps to be taken to simulate your design.

1. Opening your existing design project (Half Adder in our case).
2. Switching mode to Simulation
3. Adding an HDL Test Bench to your existing design.
4. Locating the Simulation Processes.
5. Specifying Simulation Properties.
6. Performing Simulation
7. Adding Signals
8. Analyzing the Signals

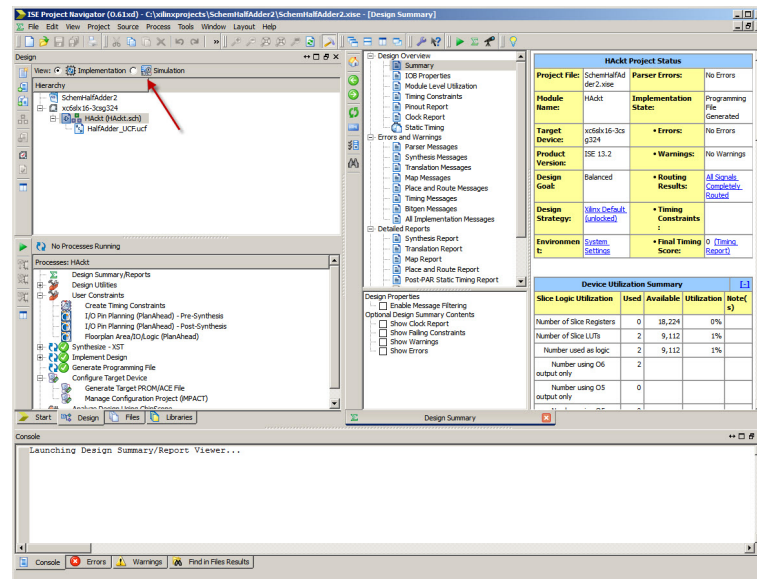
3.1 Opening your Existing Project

1. Load the Project Navigator from the  **Start** → **All Programs** → **Xilinx ISE Design Suite 14.6** → **ISE Design Tools** → **Project Navigator**.
2. The Project Navigator window will appear.



3. Click on **Open Project** or double click on the **existing project** on the screen.

4. A new screen will appear that shows your previous Half Adder design.

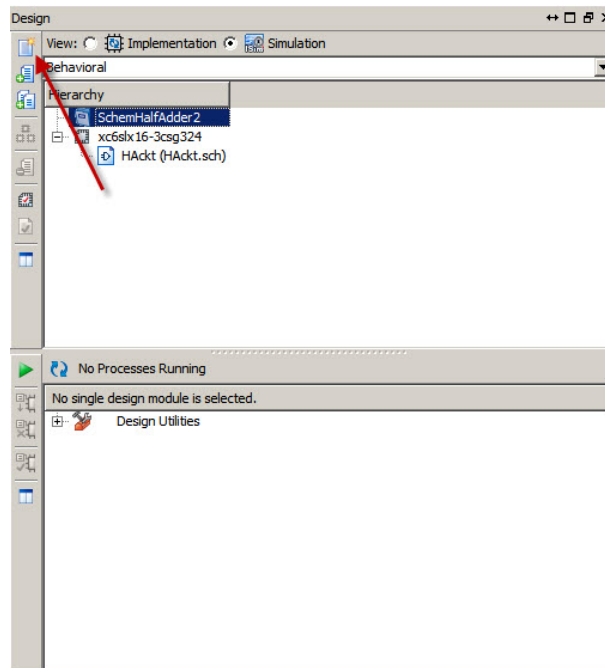


5. Notice in the view you have either **Implementation** or **Simulation** highlighted.
6. You should use the **Implementation** mode when you want to design a circuit using either schematic capture or VHDL. On the other hand you should switch to **Simulation** mode when you want to simulate your design.
7. When you switch to the **Simulation** mode you have several options:
 - (a) **Behavioral** Simulation (which is performed before you synthesize your circuit).
 - (b) **Post Translate** Simulation (which is performed after synthesis but before you implement the design).
 - (c) **Post Map** Simulation (which is performed after implementing the design, i.e., mapping).
 - (d) **Post Route** Simulation (which is performed after mapping, placement and routing).
8. In this tutorial we will only look at and use Behavioral Simulation (i.e., prior to synthesis).

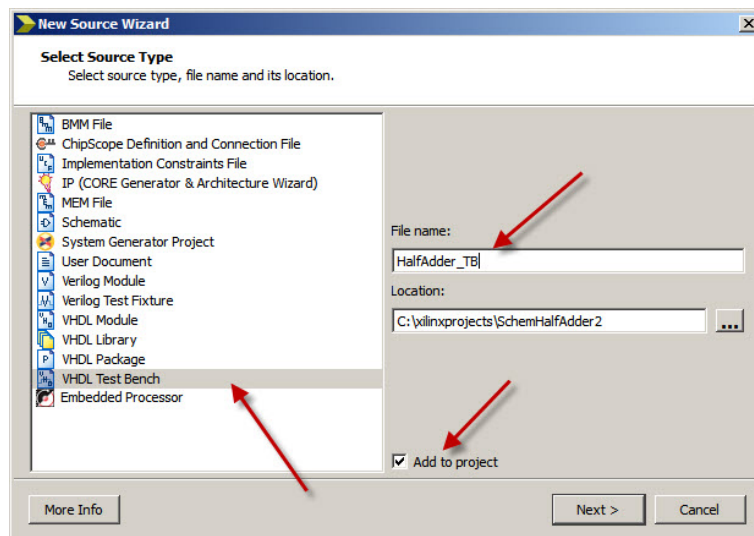
3.2 Adding an HDL Test Bench

To add an HDL test bench to your design project, you can either add a test bench file provided with this tutorial, or create your own test bench file and add it to your project. This section demonstrates how to create a new test bench file and modify it by editing it with statements from an existing test bench found on the web site. Follow these steps:

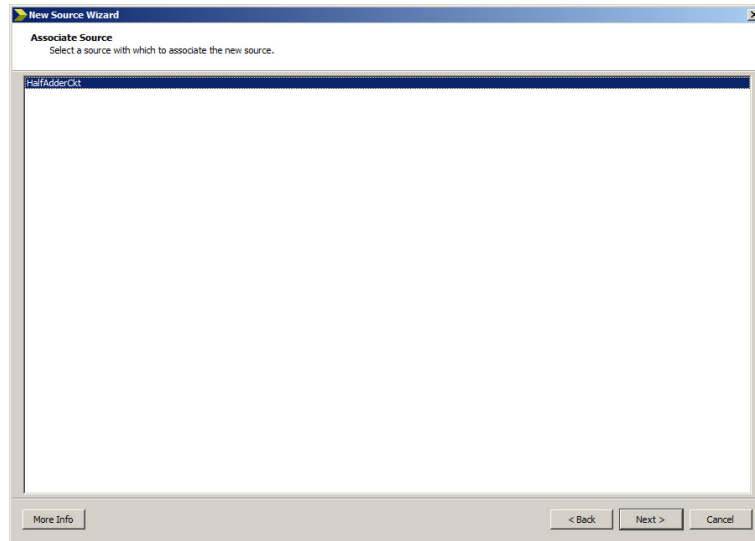
1. Highlight your project as seen in the figure below and then click on the  **New Source** button.



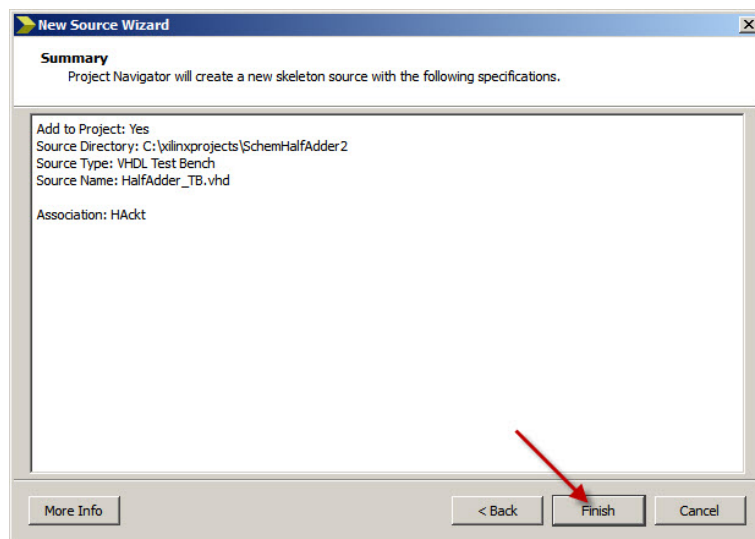
2. In the new dialog box that appears, select **VHDL Test Bench** from the list of file types and enter “HActk_TB” or “HalfAdder_TB” as the file name (should match your schematic if possible).



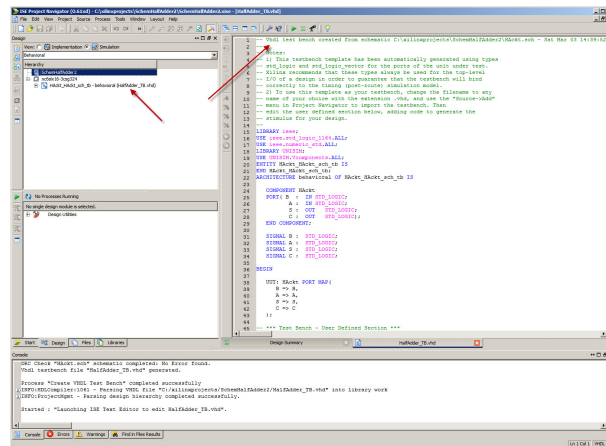
The default location is the current project directory and can be left as is. Ensure the **Add to Project** box is selected and click the **Next** button. You will then see a screen with your file name associated to the simulation (as seen in the figure below).



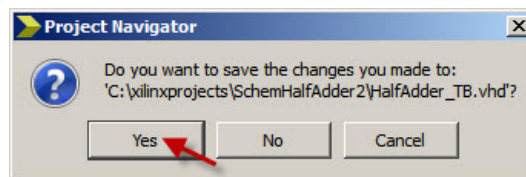
Verify the information in the next few dialog boxes and click **Next** then **Finish**.



3. A new file will be associated with your project in the **Hierarchy** pane (HActt_TB.vhd) and a skeleton VHDL file with new information will be shown in the **Workspace** panel on the right hand side of the screen.



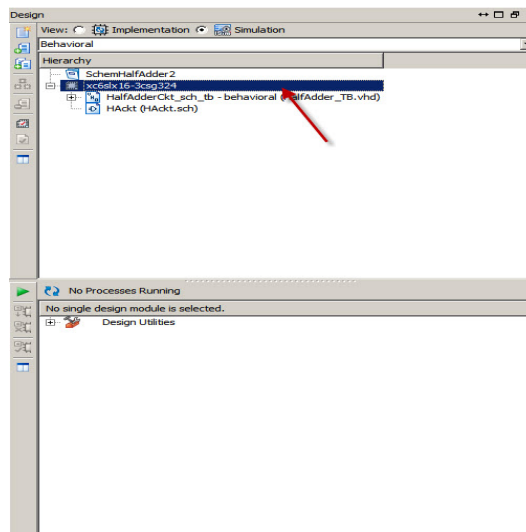
4. You will edit the VHDL code by using the “Test Bench used in Behavioral Simulation for Half Adder Design” which is found on the web page of the course (LAB2). A Copy of this test bench is found in Appendix A of this tutorial. **It is important to make sure that the name in the COMPONENT Part of the Architecture matches the name of your Schematic (Hackt.sch or HalfAdderCkt.sch).**
5. If you attempt to close the testbench file in the **Workspace** panel on the right hand side of the screen a new window will pop as seen below. Press **Yes** to ensure you save the file.



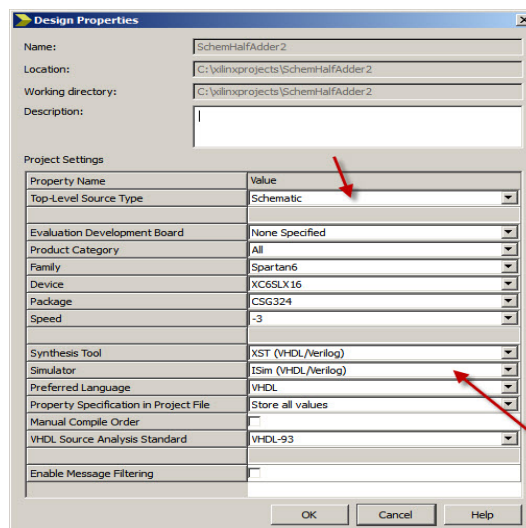
3.3 Behavioral Simulation Using ISim

Now that you have a test bench in your project, you can perform behavioral simulation on the design using ISim. The ISE software has full integration with ISim. The ISE software enables ISim to create the work directory, compile the source files, load the design, and perform simulation based on simulation properties. Here are the basic steps that you have to follow:

1. **Selecting ISim:** To select ISim as your project simulator, do the following:
 - (a) In the Hierarchy pane of the Project Navigator Design panel, right-click the device line (xc6slx16-3csg324), and select **Design Properties**.

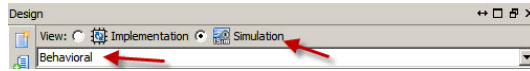


- (b) In the Design Properties dialog box, set the Simulator field to **ISim(VHDL/Verilog)** and press **OK**.



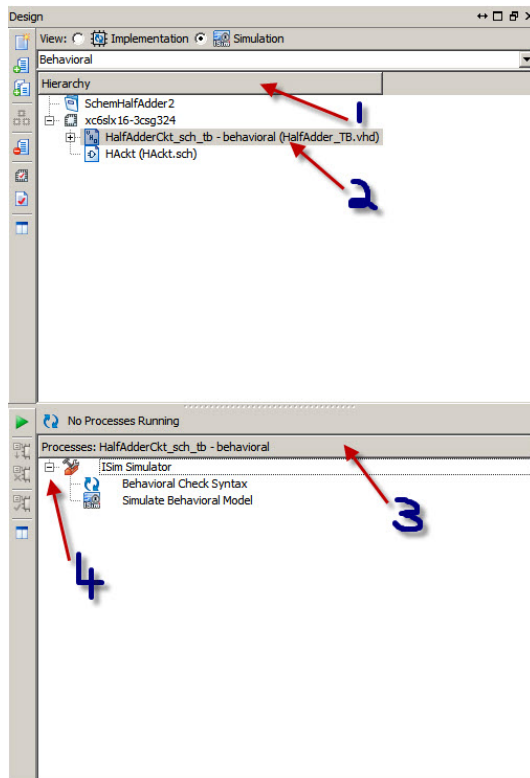
2. **Locating the Simulation Processes:** The simulation processes in the ISE software enable you to run simulation on the design using ISim. To locate the ISim processes, do the following:

- (a) In the View pane of the Project Navigator Design panel, select **Simulation**, and select **Behavioral** from the drop-down list.

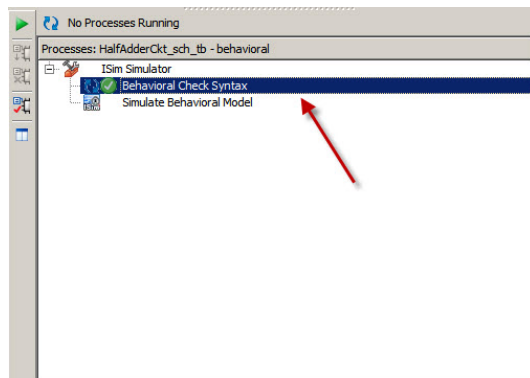


- (b) In the Hierarchy pane, select the test bench file (HActk_TB).

- (c) In the Processes pane, expand **ISim Simulator** to view the process hierarchy.

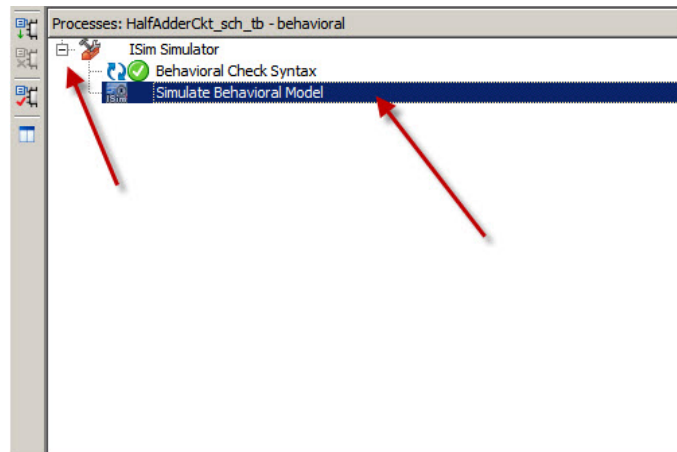


- (d) The following simulation processes are available: **Check Syntax** and **Simulate Behavioral Model**. Use the **Check Syntax** to make sure that your benchmark is free of any syntax errors. If all goes well you will see the following message (Process “Behavioral Check Syntax” Completed successfully).

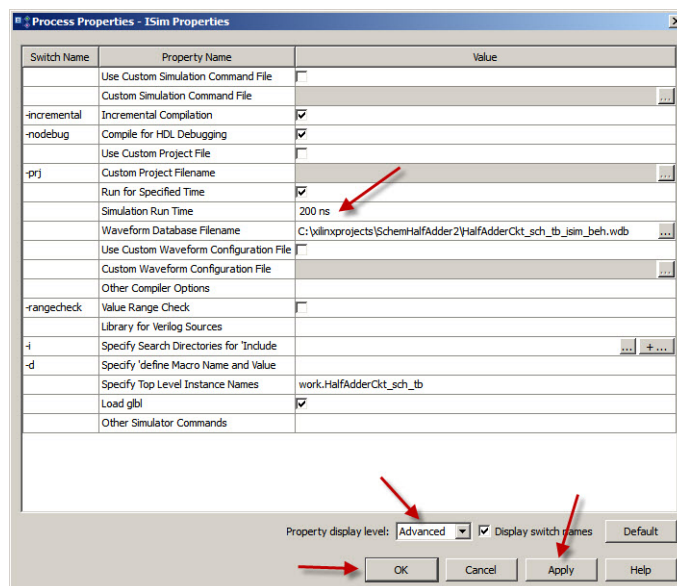


3. **Specifying Simulation Properties:** Now you will perform a behavioral simulation on the Half Adder circuit design after you set process properties for simulation. The ISE software allows you to set several ISim properties in addition to the simulation net-list properties. To see the behavioral simulation properties and to modify the properties for this tutorial, do the following:

- (a) In the Hierarchy pane of the Project Navigator Design panel, select the test bench file (HAcKt_TB).
- (b) In the processes pane, expand **ISim Simulator**, right-click **Simulator Behavioral Model**, and select **Process Properties**.



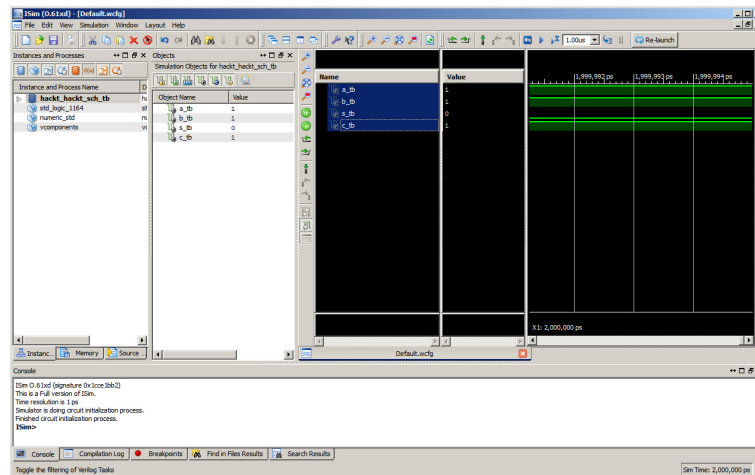
- (c) In the Process Properties dialog box, set the Property display level to **Advanced**. This global setting enables you to see all available properties.



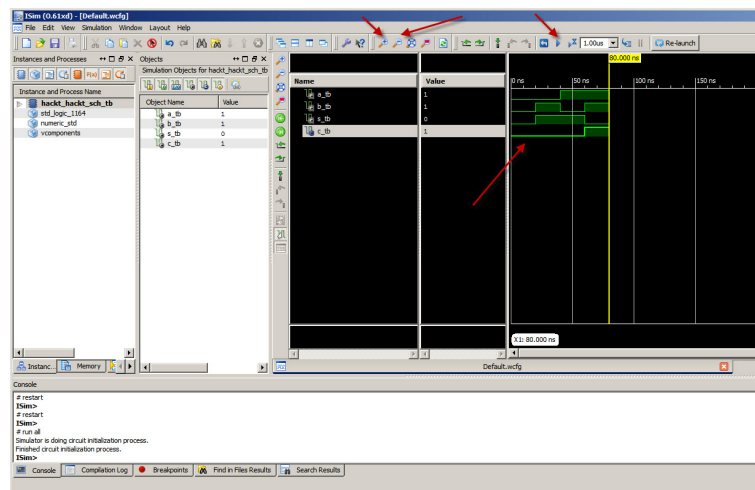
- (d) Change the Simulation Run Time to say **200 ns**.
- (e) Click **Apply** and then click **OK**.

4. **Performing Simulation:** After the process properties have been set, you are ready to run ISim to simulate the design. Follow the steps below to get waveforms for your design:

- To start the behavioral simulation, double-click **Simulate Behavioral Model**.
- ISim creates the work directory, compiles the sources files, loads the design, and performs simulation for the time specified.
- The following screen will pop:



- To display the correct waveforms expected you will need to use the zoom tools as shown below (zoom in or out):



- If you need to rerun the simulation, click the **Restart Simulation** icon



5. Your behavioral simulation is **complete**. You can **print the waveforms** and submit them along with your report by moving the mouse to the file section and then print or print preview.

4 Appendix A - VHDL Test Bench Information

A testbench is a program used to verify a design's functionality and/or its timing. Since a testbench is not synthesized, it can be written using any of the constructs and features of VHDL. In terms of coding, the effort expended to write an appropriate testbench may exceed that required to write the design description. A simple testbench consists of three basic constituents – the Unit Under Test (UUT), stimulus generate, and response monitor.

1. **Unit Under Test:** A testbench is top-level VHDL program that includes an instantiation of the design to be verified (in our case Half Adder circuit). This instance is labeled UUT (unit under test).
2. **Stimulus Generator:** A testbench includes code that applies a sequence of predetermined stimulus values to the UUT's input. This code comprises a stimulus generator.
3. **Response Monitor:** In response to each stimulus, the UUT's output values must be checked to verify that they are identical to the expected output values.

In test-benches for very simple designs, verification can be accomplished by visual inspection of the UUT's outputs in the simulator's waveform editor window. However, a more efficient approach is to include code in the testbench to automatically check the actual UUT output values against the expected values. Such a testbench is referred to as being self-checking. The code below is a testbench for a Half Adder circuit. Notice that the component part of the code below reflects the name of your Schematic or VHDL code.

The assertion statement checks whether a specified condition is true. If it is not true, a message is displayed. The report and severity clauses are optional.

```
-- A Sample Vhdl Test Bench
--
--
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;

ENTITY HalfAdderCkt_sch_tb IS
END HalfAdderCkt_sch_tb;

ARCHITECTURE behavioral OF HalfAdderCkt_sch_tb IS

    COMPONENT HalfAdderCkt
    PORT( A : IN STD_LOGIC;
          B : IN STD_LOGIC;
          C : OUT STD_LOGIC;
          S : OUT STD_LOGIC);
    END COMPONENT;

    SIGNAL A_tb : STD_LOGIC;
    SIGNAL B_tb : STD_LOGIC;
    SIGNAL C_tb : STD_LOGIC;
    SIGNAL S_tb : STD_LOGIC;
```

```

BEGIN
-- Create an instance of the circuit to be tested (Unit Under Test port map)
  UUT: HalfAdderCkt PORT MAP(
A => A_tb,
B => B_tb,
C => C_tb,
S => S_tb
  );

-- *** Test Bench - User Defined Section ***
  tb : PROCESS
constant period: time := 20 ns;
  BEGIN
A_tb <= '0'; -- apply input combination 00 and check outputs
B_tb <= '0';
wait for period;
assert ((S_tb = '0') and (C_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 ((S_tb = '1') and (C_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 ((S_tb = '1') and (C_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 ((S_tb = '0') and (C_tb = '1'))
report "test failed for input combination 11" severity error;

WAIT; -- will wait forever
  END PROCESS;
-- *** End Test Bench - User Defined Section ***


END;

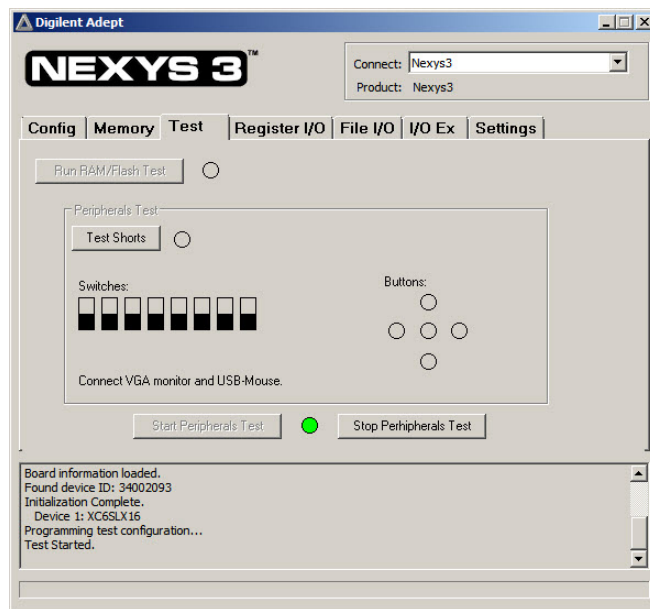
```

5 Appendix B - Setting-up and Testing the NEXYS3 board

This is intended to allow the student to quickly set up the NEXYS 3 board for this tutorial. It does not attempt to explain the configuration and is in no way a substitute for the documentation provided with the board. It will allow you to use the slide switches as input and the LEDs as outputs.

1. Connect the USB cable to the NEXYS 3 board.
2. Connect the host computer to your USB cable.
3. When the power switch to the board is on a small yellow LED labeled *Done* should glow.

You can test if the Digilent NEXYS3 Board is operational by using the Digilent Adept Tool. Double click on the  icon and you will see the Digilent Adept GUI on your screen. Press the **Test** icon. A new menu will appear. Press the “Start Peripherals Test”.



The test will display different values on the 7-segment display. You can also test the switches and light emitting diodes by sliding the switches to the on-off position. Once a switch is turned on the corresponding LED will glow. You will also notice that the switches on the Digilent Adept tool will change value. You can also test the push buttons by pressing on them. You will see the color of the corresponding button on the Adept tool change from transparent to black. Once you are satisfied that the FPGA board is operational you can press the “Stop Peripherals Test”. By pressing the “Reset Button” on the FPGA you will reset the board to the factory setting where it tests all other modules on the PCB board. Power off the board using the slide switch found at the top left part of the board.

6 Appendix C - LEDs, 7-Segments and Switches

The following sections explain the connection and location of the DIP switches and LEDs of the Digilent NEXYS 3 Board.

6.1 LEDs

The Digilent NEXYS 3 Board provides a series of eight LEDs (LD0–LD7) for use. All of these LEDs are **Logic Active High** meaning that an LED segment will glow when a logic-high is applied to it. The following table show the connection from the NEXYS 3 Board to LEDs expressed as UCF constraints.

—Description	—Location
NET LD0	LOC=U16
NET LD1	LOC=V16
NET LD2	LOC=U15
NET LD3	LOC=V15
NET LD4	LOC=M11
NET LD5	LOC=N11
NET LD6	LOC=R11
NET LD7	LOC=T11

Table 1: NEXYS 3 (Light Emitting Diodes) LEDs

6.2 Seven Segment Displays

The Digilent NEXYS 3 Board provides four multiplexed 7-segment displays for use. The following tables show the connection from the NEXYS 3 Board to the 7-segment displays expressed as UCF constraints.

—Description	—Location
NET CA	LOC=T17;
NET CB	LOC=T18;
NET CC	LOC=U17;
NET CD	LOC=U18;
NET CE	LOC=M14;
NET CF	LOC=N14;
NET CG	LOC=L14;
NET DP	LOC=M13;
NET AN0	LOC=N16;
NET AN1	LOC=N15;
NET AN2	LOC=P19;
NET AN3	LOC=P17

Table 2: NEXYS 3 (7-Segment display)

6.3 Slide Switches

The Digilent NEXYS 3 board has a bank of eight slide switches which are accessible by the user.

When closed or ON, each DIP switch pulls the connected pin of the NEXYS 3 Board to ground. When the DIP switch is open or OFF, the pin is pulled high through a $10K\Omega$ resistor.

The table below shows the connections from the Digilent NEXYS 3 Board to the switches expressed as UCF constraints.

—Description	—Location
NET SW0	LOC=T10
NET SW1	LOC=T9
NET SW2	LOC=V9
NET SW3	LOC=M8
NET SW4	LOC=N8
NET SW5	LOC=U8
NET SW6	LOC=V8
NET SW7	LOC=T5

Table 3: NEXYS 3 (Slide Switches)

6.4 Push Buttons

The Digilent NEXYS 3 board has five pushbuttons (labeled BTNS through BTNR) which are accessible by the user.

When pressed, each pushbutton pulls the connected pin of the NEXYS 3 Board to ground. Otherwise, the pin is pulled high through a $10K\Omega$ resistor. The table below shows the connections from the the Digilent NEXYS 3 Board to the push buttons expressed as UCF constraints.

—Description	—Location
NET BTNS	LOC=B8
NET BTNU	LOC=A8
NET BTNL	LOC=C4
NET BTND	LOC=C9
NET BTNR	LOC=D9

Table 4: NEXYS 3 (Pushbuttons)