



Laboratory Exercise #2

Basic Constructs in Verilog HDL

(Introduction to ModelSim*-Intel® FPGA Starter Edition)

Name: _____ Group: _____

Target Course Outcomes:

CO1: Create descriptions of digital hardware components such as in combinational and sequential circuits using synthesizable Verilog HDL constructs.

CO2: Verify the functionality of HDL-based components through design verification tools.

Intended Learning Outcomes:

- Familiarize software tool (ModelSim*-Intel® FPGA Starter Edition) and its basic features
- Create design entry of a logic circuit using Verilog HDL code
- Synthesize a circuit specified in Verilog HDL
- Simulate the designed circuit using a testbench file

Supplement:

To proceed with this laboratory exercise, make sure that you have already installed the **Intel® Quartus® Prime Lite Edition version 20.1 (or the later version, v20.1.1)** with **ModelSim*-Intel® FPGA Starter Edition**. Please refer to **Software Tools** in Canvas (*under Unit 0: Course Orientation in Modules*) for download and other related links.

This exercise requires a basic understanding of the Intel Quartus Prime design flow, as detailed in Unit 1 and Laboratory Exercise #1. It is also expected that Laboratory Exercise #1 is already completed before performing this exercise.

Instructions:

After completing Laboratory Exercise #1, proceed with the succeeding hands-on exercises. This laboratory exercise is intended to be done individually.

Exercise 2A: Getting Started

Launch the Quartus Prime Lite Edition software.

1. Open the project you created in Laboratory Exercise #1 (Exercise 1C: **HalfAdder.qpf**). This can be done by clicking the **Open Project** in the Home Tab or by navigating to **File > Open Project**. Locate the local directory where **HalfAdder.qpf** project file is located and click **OK**.
2. Specify the **EDA Tool Settings** by clicking **Tools > Options > General > EDA Tool Options**. See Figure 1 for reference.
3. In the same window as Figure 1, enter the ModelSim-Intel FPGA Edition executable path in **ModelSim-Altera EDA Tool**. For Lite Edition, search in the directory where Intel Quartus Prime Lite Edition is installed: **\\intelFPGA_lite\\20.1\\modelsim_ase\\win32aloem**. Click **OK**.

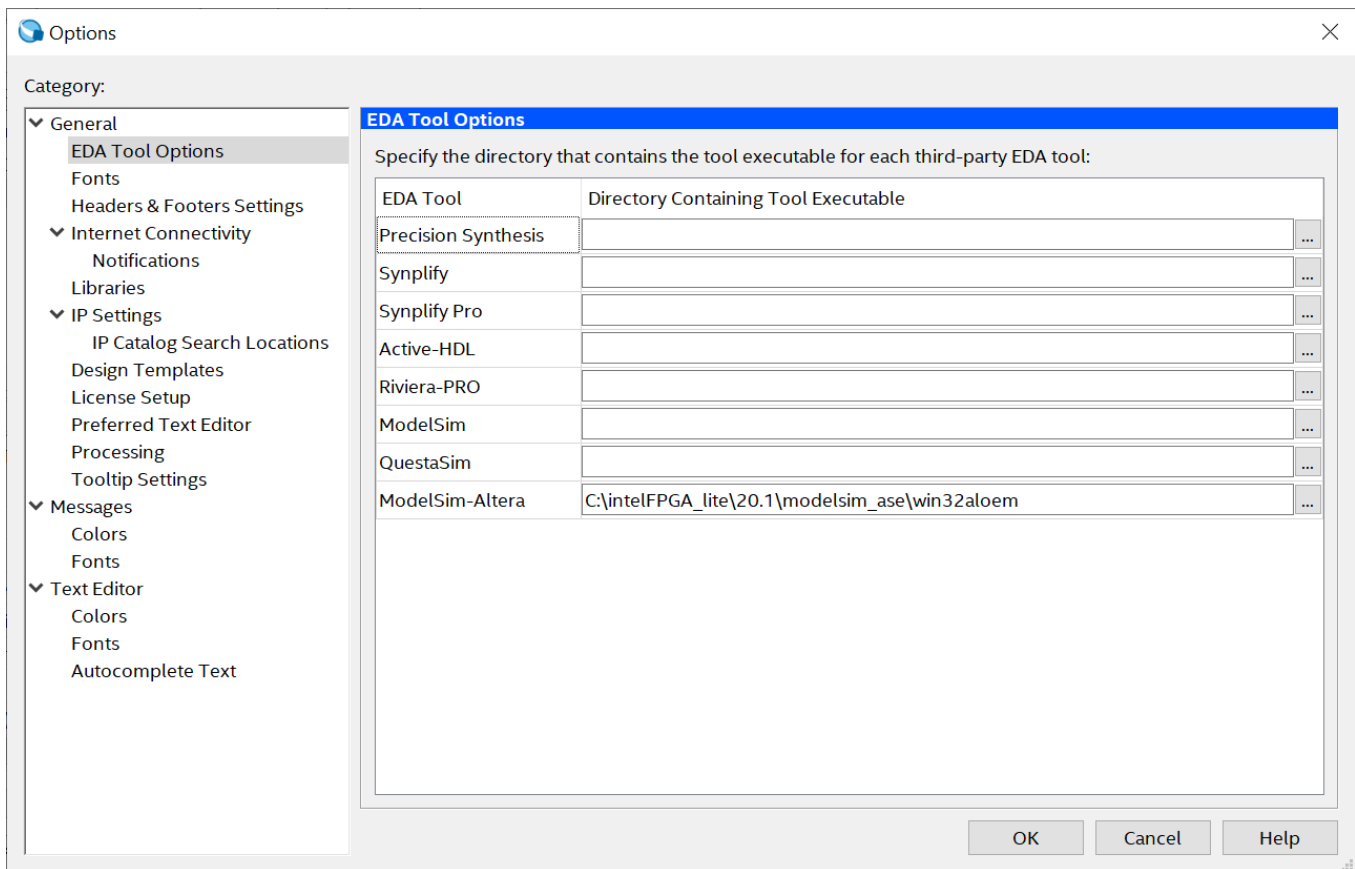


Figure 1. EDA Tool Options

4. To create the testbench file, open **File > New**, select **Design Files > Verilog HDL File**, and click **OK**.
5. Enter the testbench contents shown in Figure 2 to this new Verilog HDL File. Once finished, save the file as **tb_HalfAdder.v**. Take note that the file name of the testbench file should match the module name of the testbench. Then, double check that the option **Add file to current project** is selected before clicking **Save**.
6. Then, click **Assignments > Settings > EDA Tool Settings > Simulation**. Refer to Figure 3 or the information below to specify the values needed for the **Simulation Options**:

Tool name:	ModelSim-Altera
Run gate-level simulation automatically after compilation:	Disable checkbox
Format for output netlist:	Verilog HDL
Map illegal HDL characters:	Disable checkbox
Enable glitch filtering:	Disable checkbox
Generate Value Change Dump (VCD) file script:	Disable checkbox



```
1  /*****  
2  * File:          tb_HalfAdder.v  
3  * Author:       Antoniette Mondigo-Canete  
4  * Class:        CPE 3101L  
5  * Group/Schedule: Group 2 Thu 7:30-10:30 AM  
6  * Description:   Testbench file for HalfAdder.v  
7  *  
8  *****/  
9  
10 `timescale 1 ns / 1 ps  
11 module tb_HalfAdder();  
12  
13     // all inputs to UUT (HalfAdder.v) are declared as reg type  
14     reg x, y;  
15     // all outputs from UUT (HalfAdder.v) are declared as wire type  
16     wire C, S;  
17     // instantiate UUT with implicit port mapping  
18     HalfAdder UUT (x, y, C, S);  
19  
20     // generate stimuli  
21     initial  
22     begin  
23         x = 0;   y = 0;   #10  
24         x = 0;   y = 1;   #10  
25         x = 1;   y = 0;   #10  
26         x = 1;   y = 1;   #50  
27  
28         $stop;           //system task to end the simulation  
29     end  
30  
31 endmodule  
32
```

Figure 2. Testbench File for HalfAdder.v

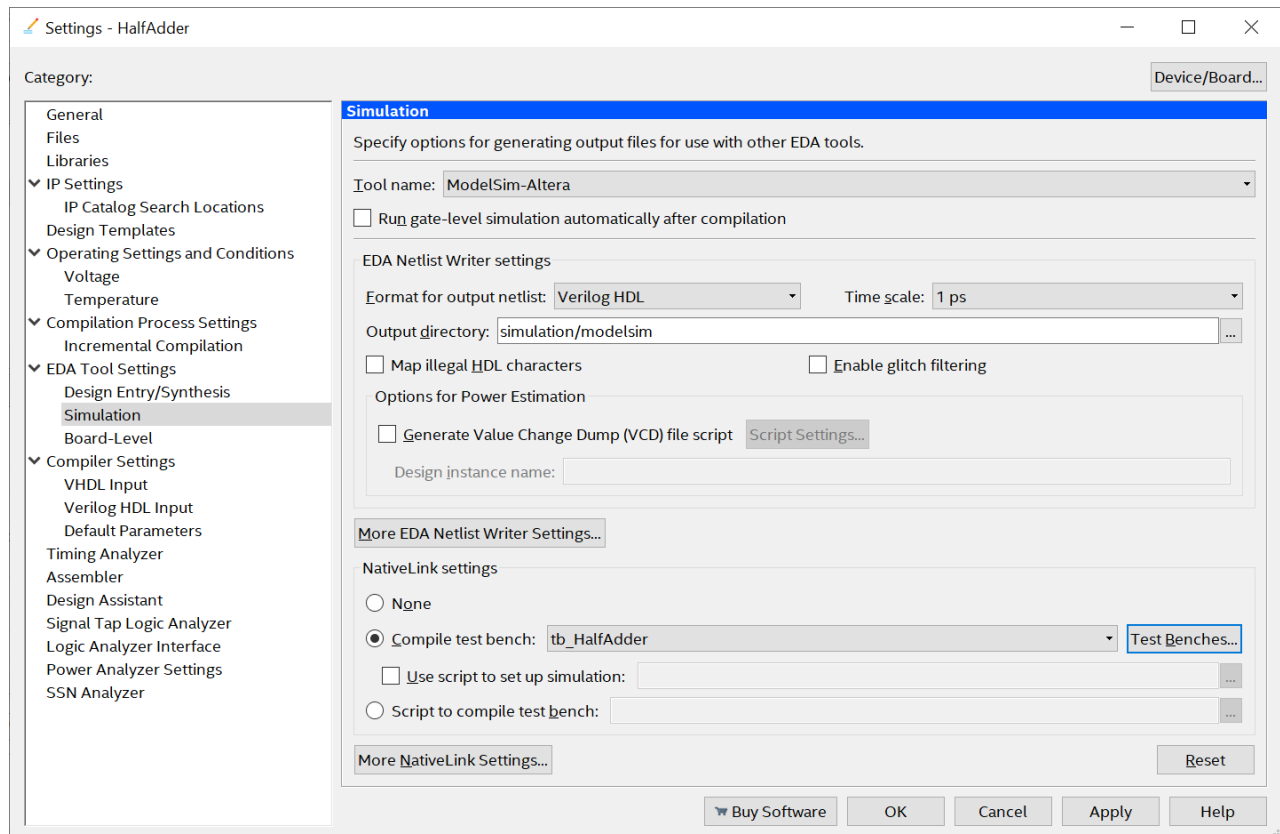
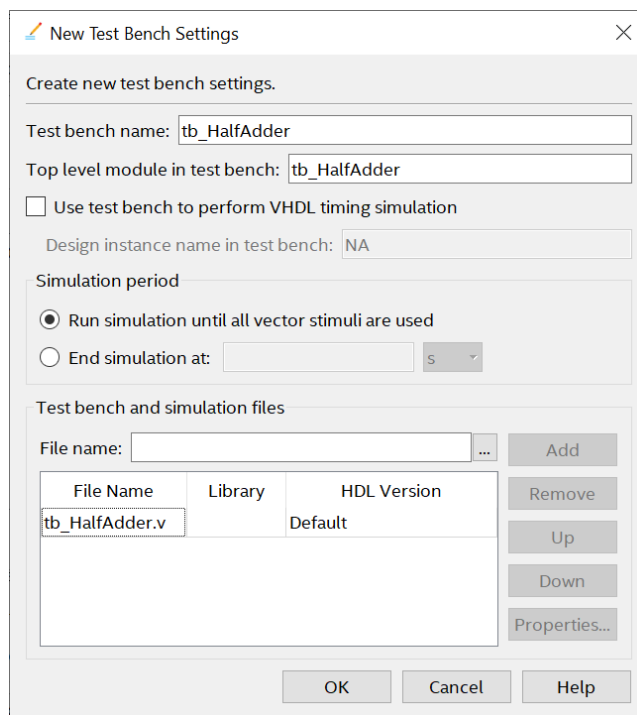


Figure 3. Simulation Options

7. Under **NativeLink settings**, select the option **Compile test bench** and click the **Test Benches** button.
8. Click **New** and specify **tb_HalfAdder** as the **Test bench name** and the **Top level module in test bench**, as shown in Figure 4. Keep the other default settings. Then, under **Test bench and simulation files**, select **tb_HalfAdder.v** file as the **File Name** and click **Add**. Once done, click **OK**.
9. The resulting **Test Benches** window will be displayed, as shown in Figure 5, and then, click **OK**. In the **Settings** window, click **Apply** and **OK**.



New Test Bench Settings

Create new test bench settings.

Test bench name:

Top level module in test bench:

☐ Use test bench to perform VHDL timing simulation

Design instance name in test bench:

Simulation period

☒ Run simulation until all vector stimuli are used

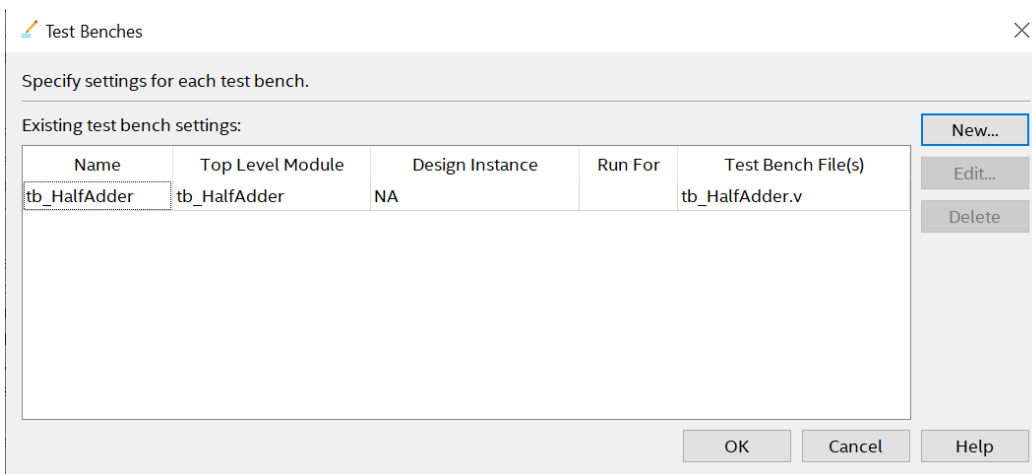
☐ End simulation at: s

Test bench and simulation files

File name: ...

File Name	Library	HDL Version
tb_HalfAdder.v		Default

Figure 4. New Test Bench Settings



Test Benches

Specify settings for each test bench.

Existing test bench settings:

Name	Top Level Module	Design Instance	Run For	Test Bench File(s)
tb_HalfAdder	tb_HalfAdder	NA		tb_HalfAdder.v

Figure 5. Test Benches Dialog Box

10. Before simulation, run **Analysis & Synthesis** under **Compile Design** in the **Tasks** window. Make sure this process has successfully completed. There should be no errors in the **Messages** window.
11. To generate and run the ModelSim-Intel FPGA Edition automation script from within the Intel Quartus Prime software, click **Tools > Run Simulation Tool > RTL Simulation**. This will launch the ModelSim-Intel FPGA Edition simulator, as shown in Figure 6, and will simulate the Half Adder design. The testbench file, **tb_HalfAdder.v** is displayed at the right side of the GUI.

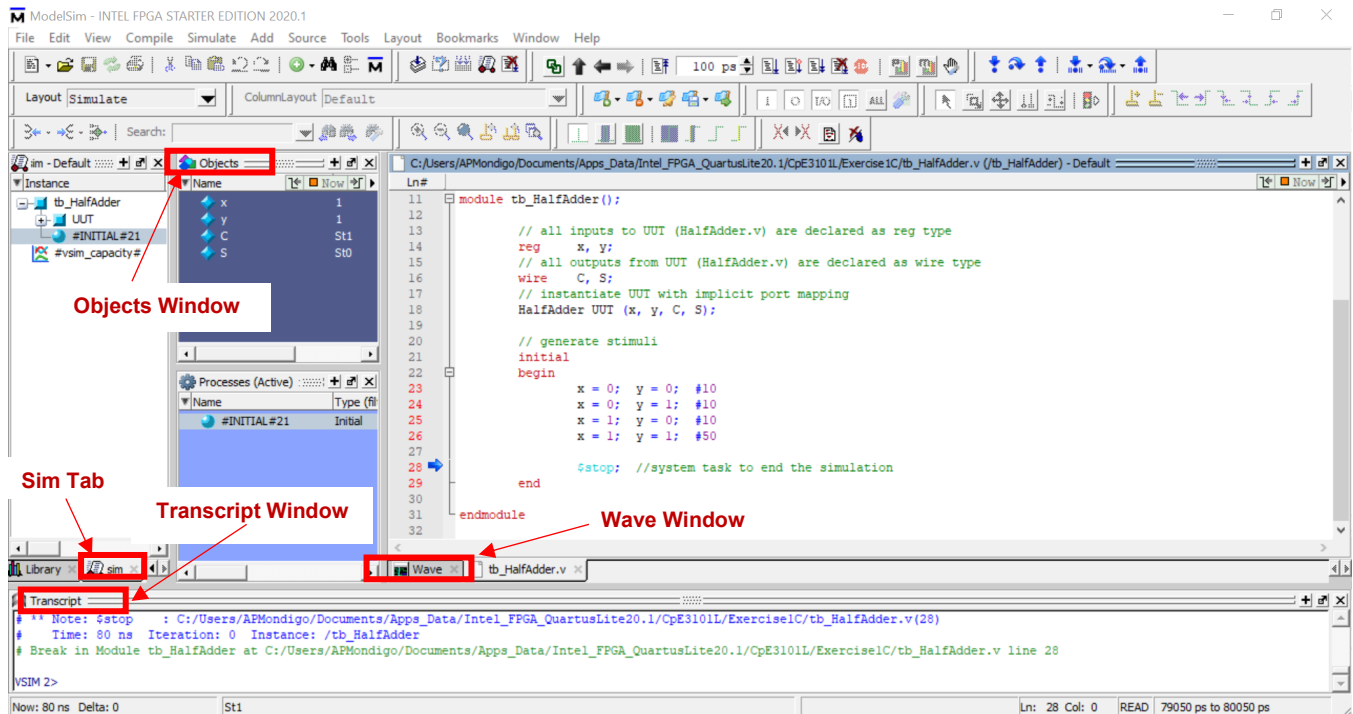


Figure 6. ModelSim - Intel FPGA Starter Edition GUI

12. Click the **Wave** window to view the signals in the simulation waveform, as shown in Figure 7. Click the **Zoom Full** icon to see the entire simulation time.
13. To rerun the simulation, click **Simulate > Restart**. Retain all default options and click **OK**. This will clear the waveforms and restart the simulation time. Then, click **Simulate > Run > Run -all** option. Click the **Wave** window again to see the simulation waveforms.

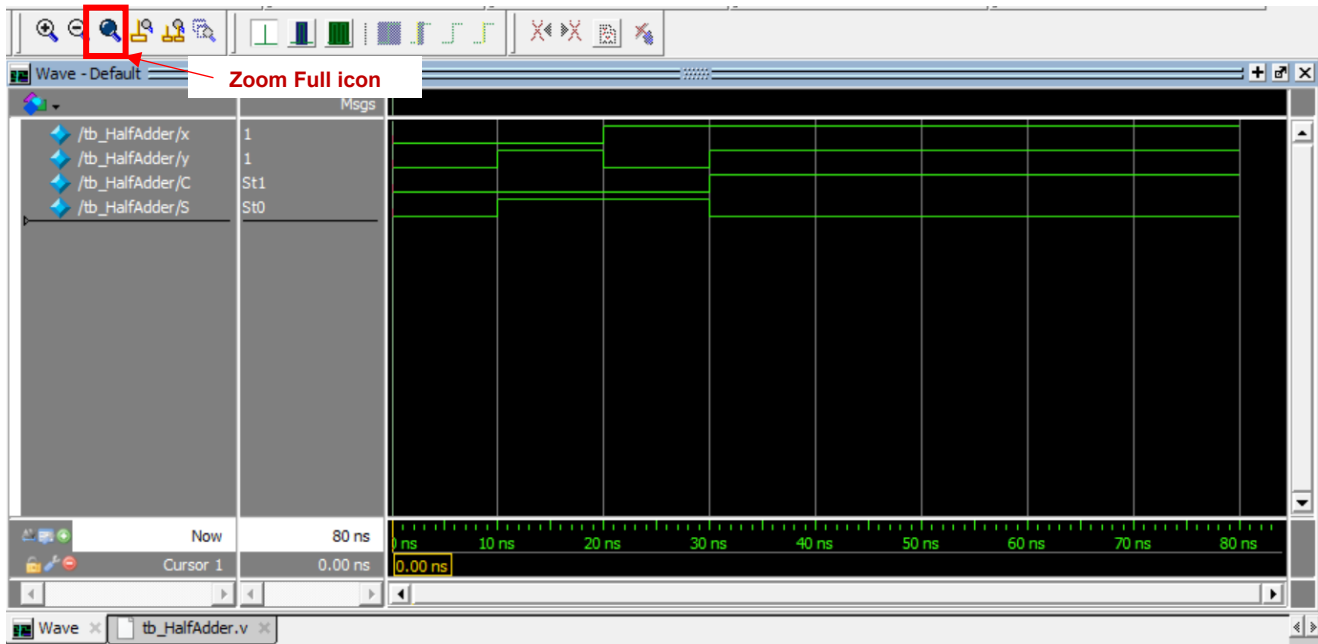


Figure 7. Wave Window

14. You may change the radix of the signals by **right-clicking** on the signals and selecting **Radix > Binary**. See Figure 8 for reference. You may also click on the different time steps in the simulation waveform to see the different input/output values of the Half Adder design.

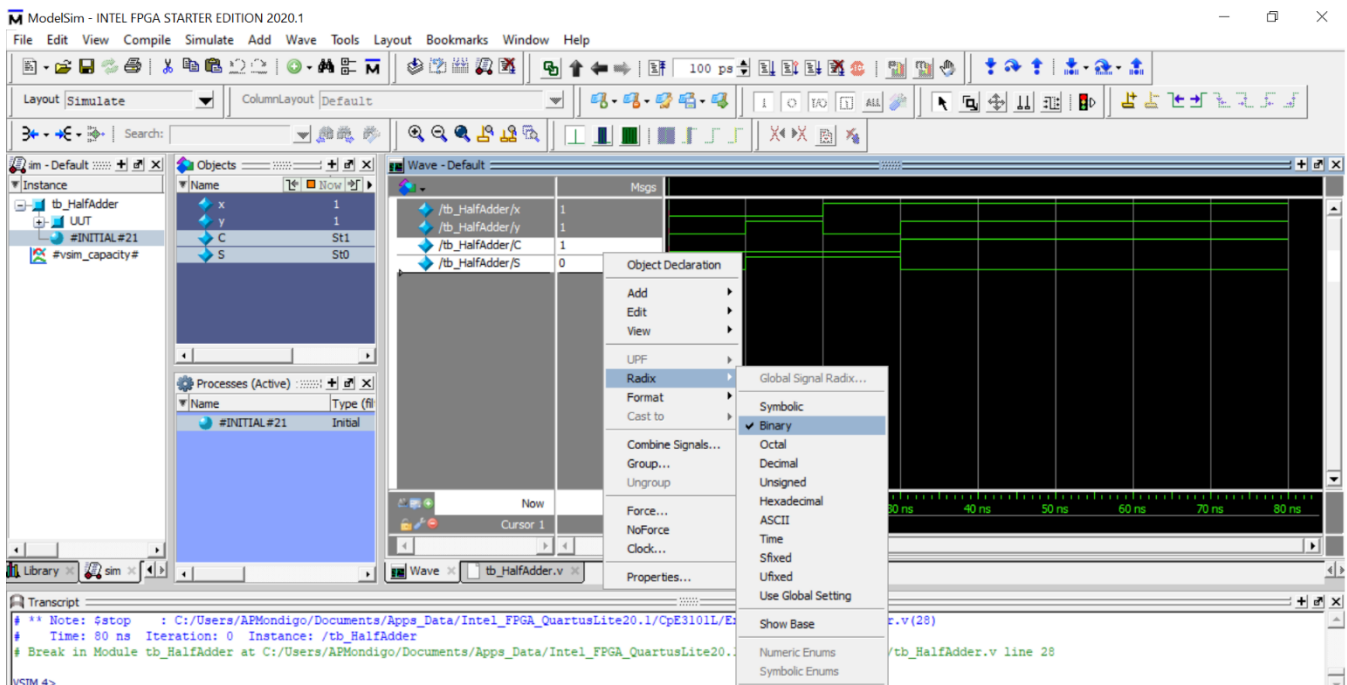


Figure 8. Changing Radix of Waveform Signals



15. The design is considered functionally verified if the waveforms follow the Half Adder functionality (*verify with its truth table*). Figure 9 shows an annotated version of the waveform screenshot to show that the results are functionally verified.

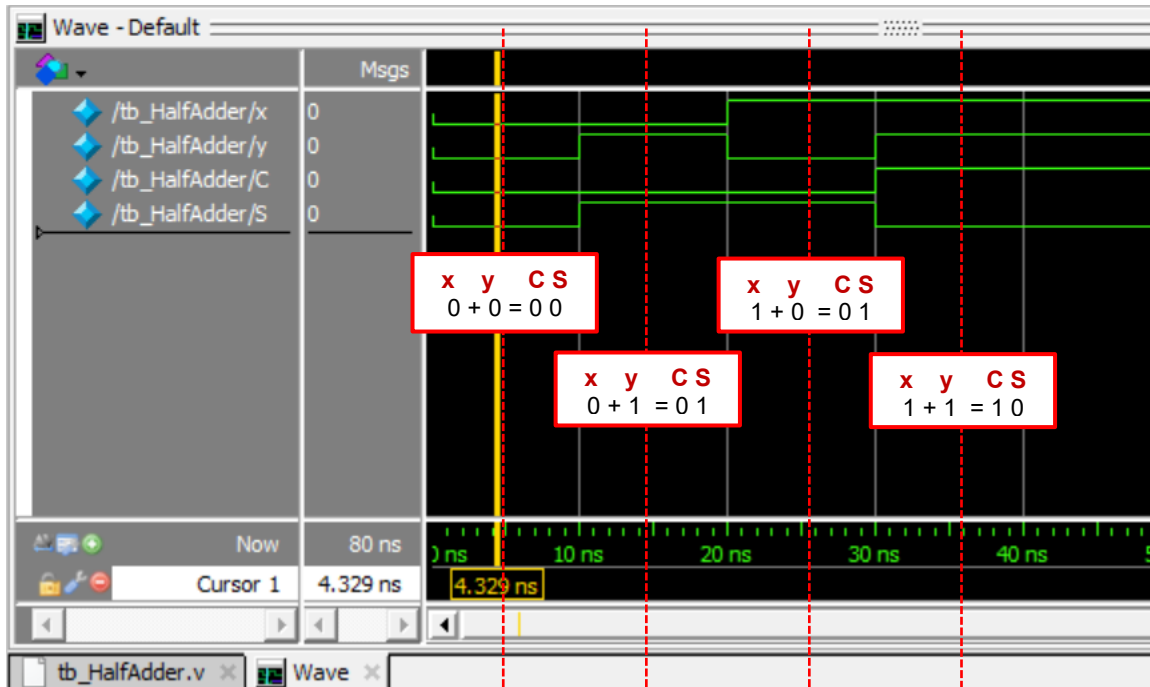


Figure 9. Testbench waveform for HalfAdder.v (with annotations)

Exercise 2B: Full Adder

Under a new project, create a Verilog HDL description of a **full adder** using **gate primitives** and **structural modeling**. Use the entity diagram below for reference (see Figure 10). For structural modeling, derive the Boolean functions and draw the logic diagram of the circuit. Analyze and synthesize the design.

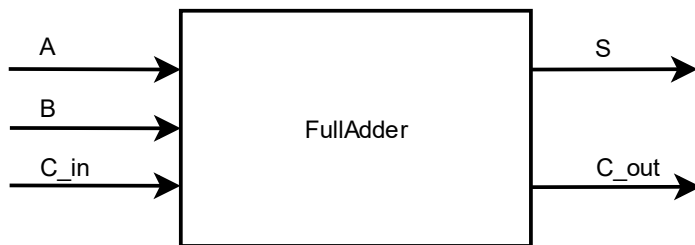


Figure 10. Entity Diagram of a Full Adder

Boolean Functions:

(You can use the back portion for the truth table, K-maps, etc.)

S = _____

C_out = _____

After the design synthesis, create and add a testbench file, as shown in Figure 11, to the current project. Simulate and verify the generated waveform using the procedures demonstrated in Exercise 2A.



```
1  /*****
2  * File:          tb_FullAdder.v
3  * Author:       <Your_name>
4  * Class:        CPE 3101L
5  * Group/Schedule: <Group_#> <Schedule>
6  * Description:   Testbench file for <design_file.v>
7  *
8  *****/
9
10 `timescale 1 ns / 1 ps
11 module tb_FullAdder();
12
13     // all inputs to UUT are declared as reg type
14     reg  A, B, C_in;
15     // all outputs from UUT are declared as wire type
16     wire S, C_out;
17
18     // instantiate UUT with implicit port mapping (you must supply this part)
19     FullAdder UUT ( ); // check the port list of your FullAdder.v file
20
21     // generate stimuli
22     initial
23     begin
24         A = 0; B = 0; C_in = 0; #10
25         A = 0; B = 0; C_in = 1; #10
26         A = 0; B = 1; C_in = 0; #10
27         A = 0; B = 1; C_in = 1; #10
28         A = 1; B = 0; C_in = 0; #10
29         A = 1; B = 0; C_in = 1; #10
30         A = 1; B = 1; C_in = 0; #10
31         A = 1; B = 1; C_in = 1; #30
32
33         $stop; //system task to end the simulation
34     end
35
36 endmodule
```

Figure 11. Testbench File for FullAdder.v

For the Laboratory Report, prepare and take screen shots of the following items for Exercise 2B:

- **Verilog HDL Design Entry** of the following:
 - Exercise 2B's Full Adder
 - Show your **name** and **course group with schedule** in the comments
- **Compilation Report for the Flow Summary**
 - Indicate how many logic elements and pins were used.
- **Schematic Diagram of Synthesized Circuit** using RTL Viewer
- **Verilog Testbench File**, showing your **name** and **course group with schedule** in the comments
- **Testbench Waveform with your annotations**, such as in Figure 9

Please see Canvas and Laboratory Template for other instructions.



Evaluation:

Level Criteria	1.0	2.0	3.0	5.0	Rating
	Outstanding	Competent	Marginal	Not Acceptable	
CO1: Verilog Design Entry	Verilog HDL description is correct and follows specified instructions.	--	Verilog HDL description is correct but DID NOT follow the specified instructions.	NO Verilog HDL description is presented or the design entry is INCORRECT.	
CO1: Design Synthesis	Verilog HDL description is synthesized and compiled with NO issues (with 0-1 warning). There is clear evidence in the lab report that supports this result.	Verilog HDL description is synthesized and compiled with MINOR issues on constructs (with 2-3 warnings). There is clear evidence in the lab report that supports this result.	Verilog HDL description is synthesized and compiled with MAJOR issues on constructs (with 4 or more warnings). There is evidence in the lab report that supports this result.	Simulation has FAILED or there is NO evidence showing a successful simulation.	
CO2: Verilog Testbench Entry	Verilog HDL testbench is correct and follows specified instructions. ALL possible input combinations or AMPLE test stimuli are included to generate sufficient and verifiable output results.	--	Verilog HDL testbench is correct but DID NOT follow specified instructions. Possible input combinations may be INCOMPLETE or test stimuli are NOT SUFFICIENT included to generate verifiable output results.	NO Verilog testbench is presented or INCOMPLETE.	
CO2: Functional Verification	Simulation of the synthesized design is functional with NO issues and shows ALL correct and expected results. An analysis is made by providing image annotations and/or discussions of the results. There is clear evidence in the lab report that supports this.	Simulation of the synthesized design is functional. Expected results may be INCOMPLETE but are ALL correct. An analysis is made by providing image annotations and/or discussions of the results. There is clear evidence in the lab report that supports this.	Simulation of the synthesized design is functional. Expected results may be INCOMPLETE or there may be INCORRECT results. NO analysis is made with NO image annotations and/or discussions of the results. There is clear evidence in the lab report that supports this.	Simulation has FAILED or there is NO evidence showing a successful simulation.	