Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Hands-on Workshop on DFT
Octubre-diciembre 2025

TEC | Tecnológico de Costa Rica

Laboratory 2

Importing Modules, Pin Assignment via QSF, and Signal Verification in Simulation and on the DE10-Standard FPGA

Introduction

This laboratory focuses on the integration, simulation, and hardware verification of a modular ALU design on the DE10-Standard FPGA. Students will learn to import SystemVerilog modules into a Quartus Prime project, assign pin locations using a .qsf file, and verify the design functionality both in simulation (using ModelSim) and on physical hardware. The lab emphasizes efficient design reuse, structured pin assignment, and functional validation through waveform analysis and FPGA outputs. Additionally, students will develop and apply test benches to simulate different ALU operations and compare results between simulation and hardware execution.

Objectives

General objective:

To verify the functional behavior of an ALU-based SystemVerilog design on the DE10-Standard FPGA through module integration, pin assignment via QSF, simulation in ModelSim, and on-board signal observation.

Specific objectives:

1. To import SystemVerilog modules into a Quartus Prime project and manage project files effectively.
2. To assign physical FPGA pins using a preconfigured .qsf file and validate the connections via the Assignment Editor.
3. To compile, program, and test the ALU design on the DE10-Standard board using switches, buttons, LEDs, and 7-segment displays.
4. To simulate the ALU module using ModelSim, apply test cases through a testbench, and visualize waveforms.
5. To compare simulation results with on-board outputs to ensure design correctness.
6. To identify and explain configuration-related errors that may occur during hardware deployment despite correct code syntax.

Materials

- Quartus Prime 24.1 Standard Edition.
- ModelSim-Intel® FPGAs Standard Edition 19.1
- DE10-Standard FPGA development board.

Procedure

Import Modules

1. Create a new folder named **LabFPGA2** in Desktop\LabFPGA_DE10St.
2. Create a new project based on the procedure learned in Laboratory 1. Save the project in the folder LabFPGA2 located at Desktop\LabFPGA_DE10St. The project name must be **LabFPGA2** (Device name: **5CSXFC6D6F31C6**).
3. In the top menu, select ***Project → Add/Remove Files in Project...***, as shown in Figure 1. A new window will appear.

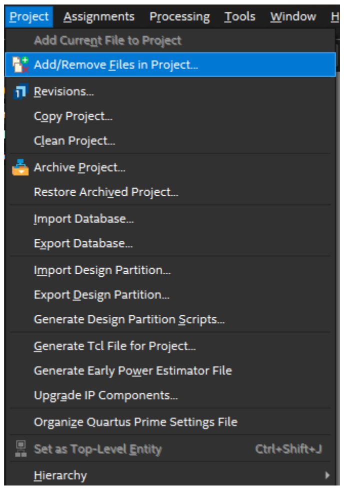

Project
Assignments
Processing
Tools
Window
Add Current File to Project
Add/Remove Files in Project...
Revisions...
Copy Project...
Clean Project...
Archive Project...
Restore Archived Project...
Import Database...
Export Database...
Import Design Partition...
Export Design Partition...
Generate Design Partition Scripts...
Generate Tcl File for Project...
Generate Early Power Estimator File
Upgrade IP Components...
Organize Quartus Prime Settings File
Set as Top-Level Entity Ctrl+Shift+J
Hierarchy

Figure 1. Add Files

4. Under *Settings - LabFPGA*, **click the three dots** next to *File name*, as shown in Figure 2, and browse to select the file **LabFPGA2.sv** located at Desktop\LabFPGA_DE10St\LABResources\LabFPGA2.

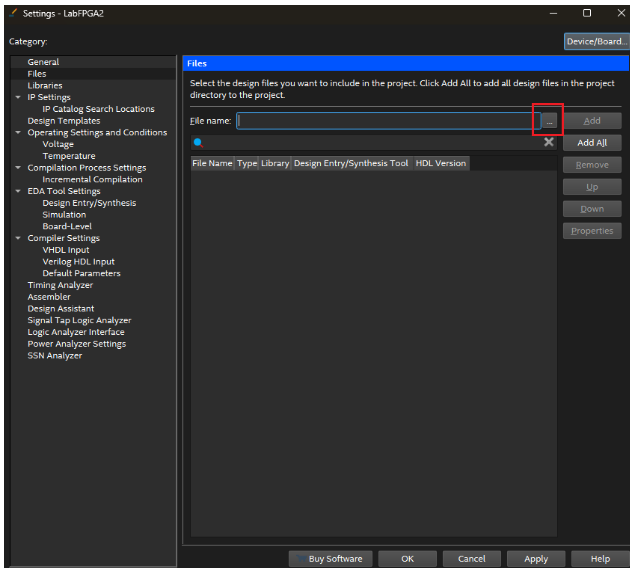


Figure 2. Select File

5. Click ***OK*** to add the new file to the project.

6. In the left panel of the software, **double-click** on LabFPGA2 (Figure 3) to view the added module.

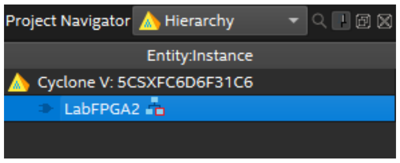


Figure 3. Show New Module

7. The LabFPGA2 module is a more advanced design than the previous one. It operates with 24-bit operands and supports 15 different operations. It also uses the 7-segment display to show data.

8. **Compile** the design as shown in Laboratory 1. Make sure there are no errors.

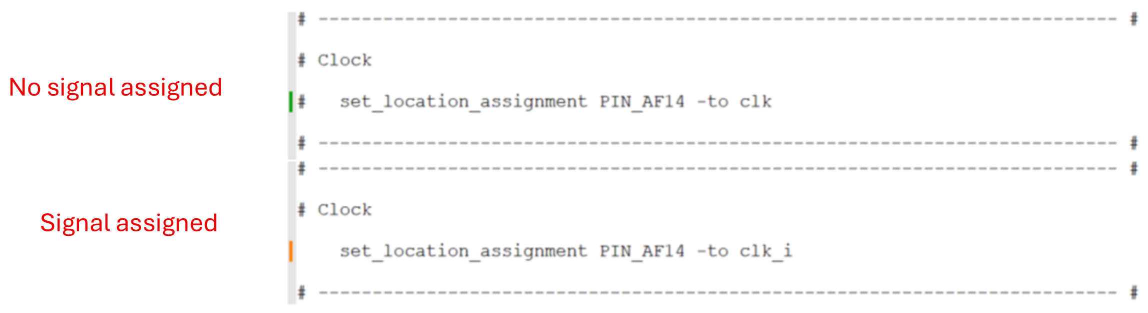
Pin Assignment via QSF

9. In the previous lab, we manually assigned FPGA signals to physical pins. However, this process can become tedious for designs with a large number of signals. Quartus allows you to import a .qsf file that already contains the pin assignments, which greatly simplifies the process.

10. Using the File Explorer, open the file **LabFPGA_pinout.qsf** located at Desktop\LabFPGA_DE10St\LABResources\LabFPGA2 using **Notepad or Notepad++**. This

.qsf file contains an organized list of the required peripherals and their corresponding FPGA pin assignments.

11. To assign a pin to a signal in your design, replace the generic placeholder name with your **signal name** and **remove the # charac**ter at the beginning of the line. This character is used to comment the line, so **removing it is necessary** for Quartus to recognize the assignment. Figure 4 shows an example of a signal assignment.



```
# ------------------------------------------------------------------------- #
# Clock
#   set_location_assignment PIN_AF14 -to clk
# ------------------------------------------------------------------------- #
# ------------------------------------------------------------------------- #
# Clock
    set_location_assignment PIN_AF14 -to clk_i
# ------------------------------------------------------------------------- #
```

Figure 4. Example of a Signal Assignment

12. **Assign all required signals** in the .qsf file. Table 1 lists the signals along with their corresponding FPGA peripherals.

Table 1. Pin Assignment

Signal	**Peripheral**
clk_i	Clock
op_i[0]	SW[0]
op_i[1]	SW[1]
op_i[2]	SW[2]
op_i[3]	SW[3]
sw_reg_b_i	SW[4]
cntrl_alu_i[0]	SW[5]
cntrl_alu_i[1]	SW[6]
cntrl_alu_i[2]	SW[7]
cntrl_alu_i[3]	SW[8]
rst_i	SW[9]
slct_op_i	KEY[0]
slct_byte_i	KEY[1]

slct_sb_i	KEY[2]
en_reg_i	KEY[3]
sb_op_o	LEDR[0]
counter_slct_byte_o[0]	LEDR[1]
counter_slct_byte_o[1]	LEDR[2]
counter_slct_op_o[0]	LEDR[3]
counter_slct_op_o[1]	LEDR[4]
flag_o[0]	LEDR[5]
flag_o[1]	LEDR[6]
flag_o[2]	LEDR[7]
flag_o[3]	LEDR[8]
display_o[0]	Display_hex0[0]
display_o[1]	Display_hex0[1]
display_o[2]	Display_hex0[2]
display_o[3]	Display_hex0[3]
display_o[4]	Display_hex0[4]
display_o[5]	Display_hex0[5]
display_o[6]	Display_hex0[6]
display_o[7]	Display_hex1[0]
display_o[8]	Display_hex1[1]
display_o[9]	Display_hex1[2]
display_o[10]	Display_hex1[3]
display_o[11]	Display_hex1[4]
display_o[12]	Display_hex1[5]
display_o[13]	Display_hex1[6]
display_o[14]	Display_hex2[0]
display_o[15]	Display_hex2[1]
display_o[16]	Display_hex2[2]
display_o[17]	Display_hex2[3]
display_o[18]	Display_hex2[4]
display_o[19]	Display_hex2[5]
display_o[20]	Display_hex2[6]
display_o[21]	Display_hex3[0]
display_o[22]	Display_hex3[1]
display_o[23]	Display_hex3[2]
display_o[24]	Display_hex3[3]
display_o[25]	Display_hex3[4]
display_o[26]	Display_hex3[5]
display_o[27]	Display_hex3[6]
display_o[28]	Display_hex4[0]
display_o[29]	Display_hex4[1]
display_o[30]	Display_hex4[2]

display_o[31]	Display_hex4[3]
display_o[32]	Display_hex4[4]
display_o[33]	Display_hex4[5]
display_o[34]	Display_hex4[6]
display_o[35]	Display_hex5[0]
display_o[36]	Display_hex5[1]
display_o[37]	Display_hex5[2]
display_o[38]	Display_hex5[3]
display_o[39]	Display_hex5[4]
display_o[40]	Display_hex5[5]
display_o[41]	Display_hex5[6]

13. **Save the changes** you made to the .qsf file.

14. To import the pin assignments into Quartus, go to the top menu and select ***Assignments* → *Import Assignments...***, as shown in Figure 5.

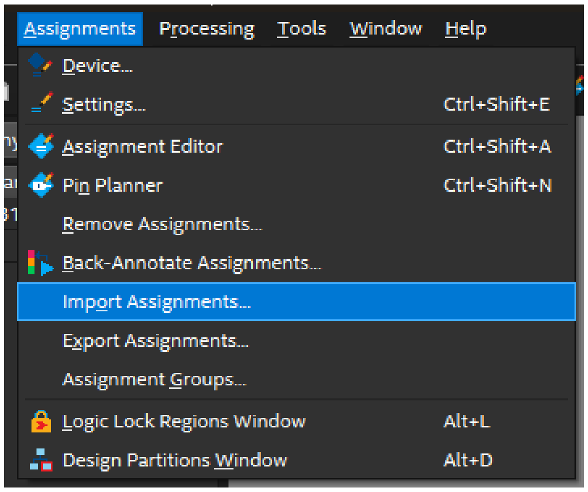
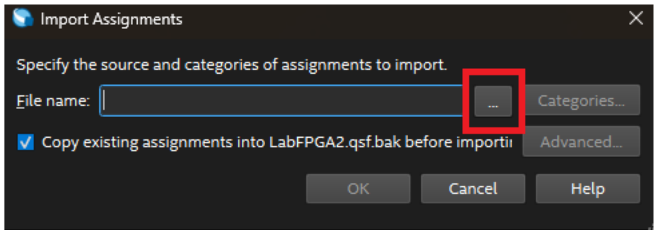


Figure 5. Import Assigments

15. Click the **three dots** next to File name, as shown in Figure 6. Then browse and select the **.qsf** file located at Desktop\LabFPGA_DE10St\LABResources\LabFPGA2. Click **OK** to finish.


Import Assignments

Specify the source and categories of assignments to import.

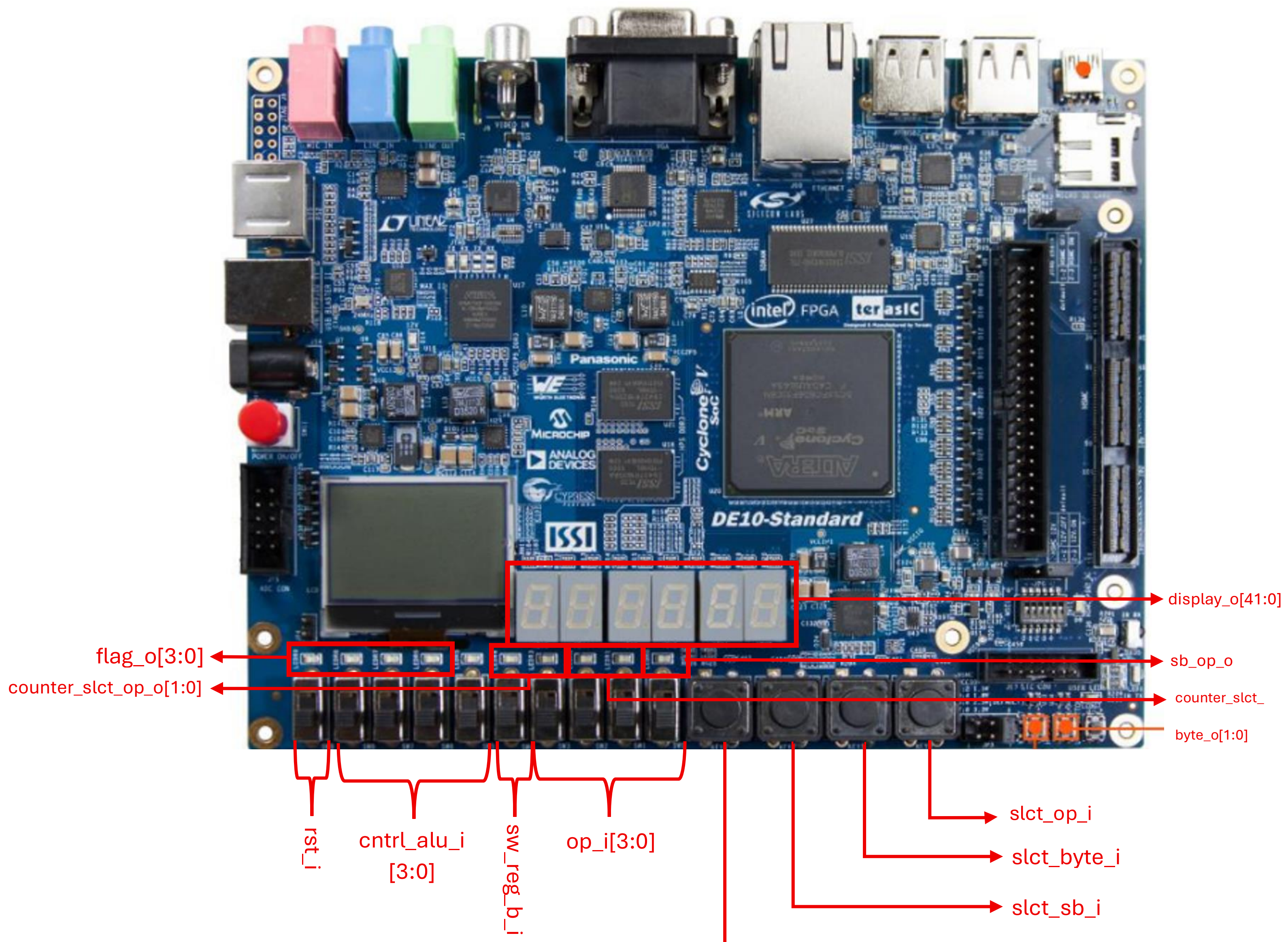
File name:

☑ Copy existing assignments into LabFPGA2.qsf.bak before importi

Figure 6. Select .qsf File

16. If you now open the **Assignment Editor** (as done in the previous lab), you will see that all signals have been correctly assigned to the FPGA's physical pins.
17. You must **recompile** the design.
18. **Program** the FPGA as done in the previous lab. Make sure to correctly place the ***SOCVHPS*** device in the chain before starting the programming process.

Peripherals



Signal	**Type**	**Description**
clk_i	Input	System clock. All synchronous logic is triggered on the rising edge.
rst_i	Input	System reset. Resets all internal registers and counters. **Active in low.**
cntrl_alu_i[3:0]	Input	ALU control code. Selects which operation the ALU will execute (e.g., ADD, SUB, AND, etc.).
op_i[3:0]	Input	4-bit input nibble. Loaded into the selected nibble of the selected byte of the current operand.
slct_op_i	Input	Selects active operand/result: 0 = Operand A 1 = Operand B 2 = Result
slct_byte_i	Input	Selects which byte of the operand is active: 0 = Byte 0 (LSB) 1 = Byte 1 2 = Byte 2 (MSB)
slct_sb_i	Input	Selects nibble within the byte: 0 = LSB (lower 4 bits) 1 = MSB (upper 4 bits)
en_reg_i	Input	Enables writing op_i into the selected nibble and byte of the current operand.
sw_reg_b_i	Input	Operand B source selector: 0 = input from op_i 1 = load result from ALU into Operand B Note: In both cases, you must load register B using en_reg_i
flag_o[3:0]	Output	ALU status flags: bit 0 = Z (Zero) bit 1 = S (Sign) bit 2 = O (Overflow) bit 3 = C (Carry)
counter_slct_op_o[1:0]	Output	Indicates current operand selected: 0 = Operand A 1 = Operand B 2 = Result
counter_slct_byte_o[1:0]	Output	Indicates current byte selected: 0 = Byte 0 1 = Byte 1 2 = Byte 2
sb_op_o	Output	Indicates nibble selection: 0 = LSB, 1 = MSB.

display_o[41:0]	Output	7-segment display output. Shows the value selected (A, B, or result) in hexadecimal digits.

ModelSim Simulation

19. Open **File Explorer** and go to the directory Desktop\LabFPGA_DE10St\LABResources \LabFPGA2. Open the file **tb_LabFPGA2.sv** using Notepad or Notepad++.

20. In this file, you will find a pre-written **testbench** for the module LabFPGA2.sv. Scroll to the end of the file and locate the comment **Continue your code here...**

21. Just **above** that comment, you will find an example test case that performs the addition of 0x5A5 and 0xA5A. Based on this example, write at least eight additional test cases using **different ALU operations** to **verify the correctness of your design against the results observed on the FPGA**. *(Make sure to save your changes after editing the file.)*

22. Table 2 shows all the operations supported by the ALU, along with their corresponding control code (in **decimal**).

Table 2. Code for ALU operations

Code	Operation
0	ADD
1	SUB
2	Increase (A + 1)
3	Decrease (A - 1)
4	AND
5	OR
6	NOT A
7	NOT B
8	NAND
9	XOR
10	XNOR
11	SLL (A << B)
12	SRL (A >> B)
13	Less than
14	Greater than or equal to

23. **Open ModelSim** and create a **new project**. To do this, go to ***File → New → Project...***, as shown in Figure 7.

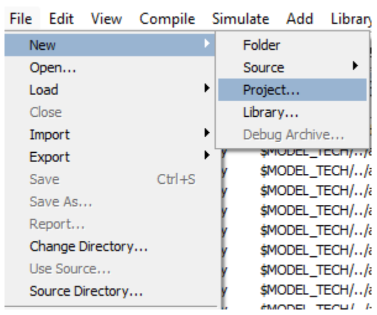
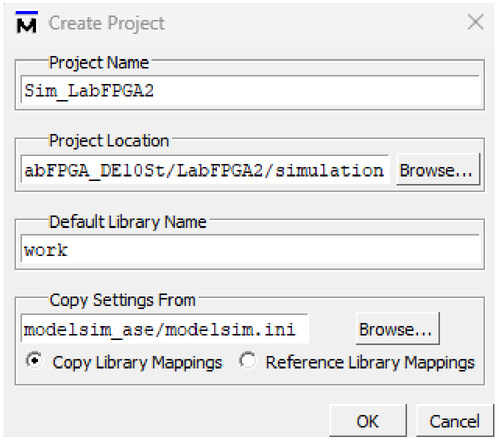


Figure 7. New project in ModelSim

24. Name the project **Sim_LabFPGA2** and set the project location to the simulation folder located at **Desktop\LabFPGA_DE10St\LabFPGA2\simulation**. Leave all other settings as default and click ***OK***. Figure 8 shows an example of the setup.



Project Name: Sim_LabFPGA2
Project Location: abFPGA_DE10St/LabFPGA2/simulation
Default Library Name: work
Copy Settings From: modelsim_ase/modelsim.ini
(● Copy Library Mappings ○ Reference Library Mappings)

25. *Figure 8. Create New Project*

26. A new pop-up window will appear. Click on ***Add Existing File***, then click the *Browse* button and navigate to Desktop\LabFPGA_DE10St\LABResources\LabFPGA2. Add the files **LabFPGA2_sim.sv** and **tb_LabFPGA2.sv**, then click ***OK*** and finally ***Close*** to complete the process.

27. In the *Project* tab, you will see the added files. **Right-click** on either of the two files and select ***Compile → Compile All***, as shown in Figure 9.

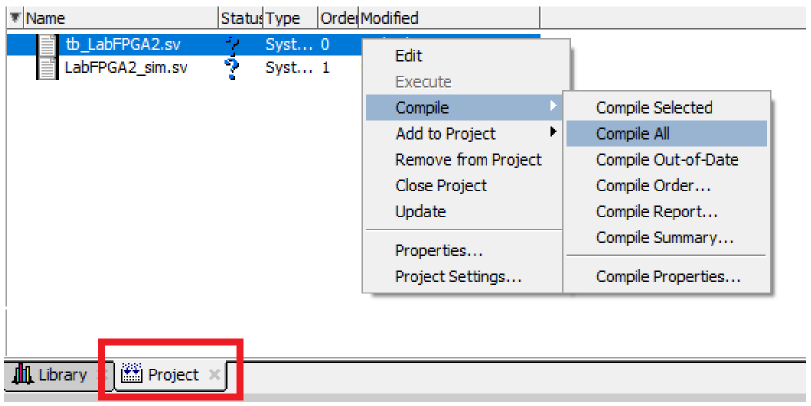


Figure 9. Compile Files

28. ModelSim will compile the files. If there are no syntax errors, the status icon will change from a question mark to a checkmark. Otherwise, an "X" will appear indicating a compilation error. Verify that both files show a **checkmark** under the Status column, as illustrated in Figure 10.

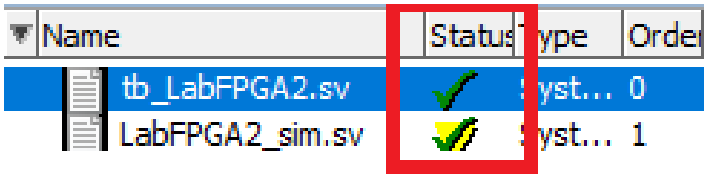


Figure 10. Compile Status

29. Click on the ***Library*** tab and expand the library named **work**. You will see all compiled modules from the project files, including the testbench, top-level module, and any submodules. **Right-click** on tb_LabFPGA2 and select ***Simulate***, as shown in Figure 11.

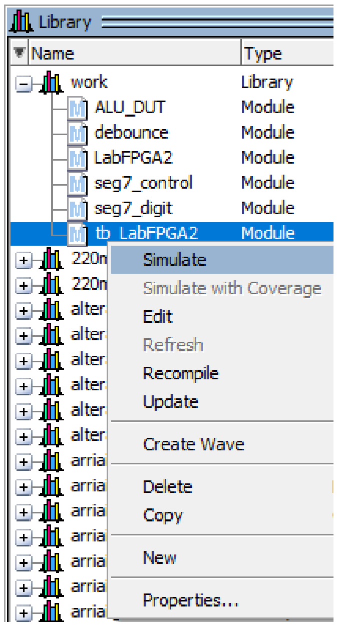
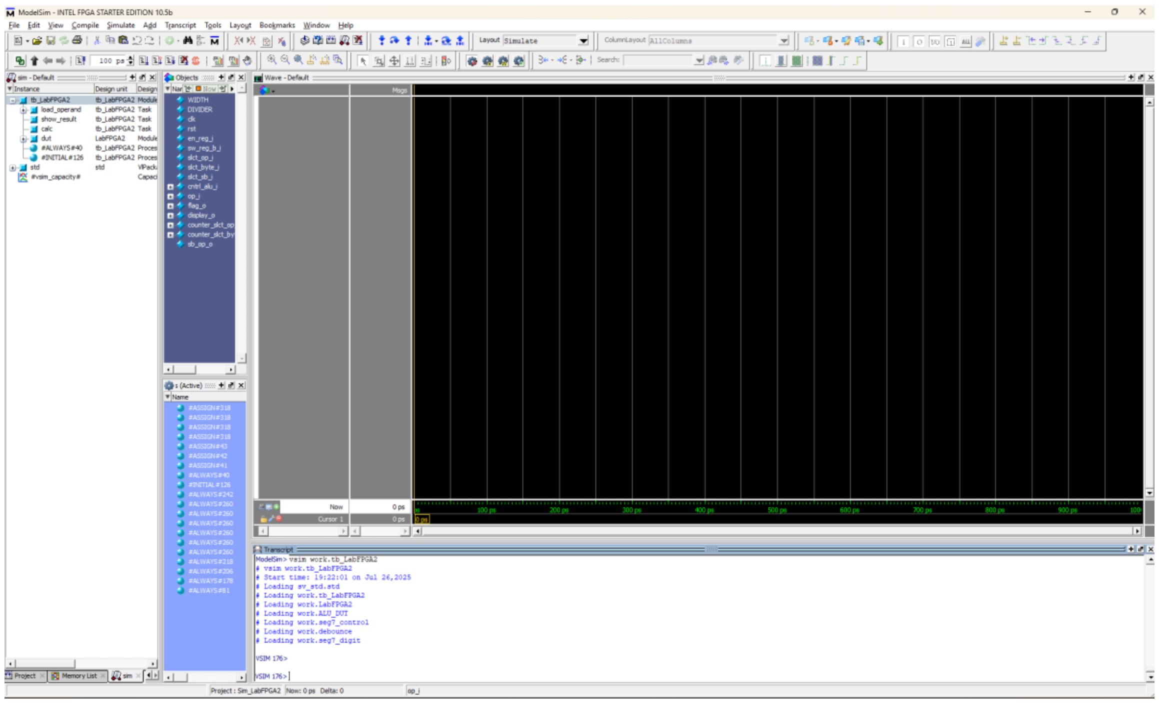


Figure 11. Start Simulation

30. A set of tabs will appear, as shown in Figure 12. You can drag signals from the *Objects* window into the *Wave* tab to visualize the signal waveforms. In the *Instance* tab, you can explore both testbench and DUT signals. Feel free to explore freely.


ModelSim - INTEL FPGA STARTER EDITION 10.5b
File Edit View Compile Simulate Add Transcript Tools Layout Bookmarks Window Help
sim - Default
Objects
Wave - Default
Transcript
Now
0 ps
Cursor 1
0 ps

```
ModelSim> vsim work.tb_LabFPGA2
# vsim work.tb_LabFPGA2
# Start time: 19:22:01 on Jul 26,2025
# Loading sv_std.std
# Loading work.tb_LabFPGA2
# Loading work.LabFPGA2
# Loading work.ALU_DUT
# Loading work.seg7_control
# Loading work.debounce
# Loading work.seg7_digit

VSIM 176>

VSIM 176>
```

```
Project : Sim_LabFPGA2   Now: 0 ps  Delta: 0   op_i
```

Figure 12. Simulation Tabs

31. In the ***Instance*** tab, expand the **dut** module and **click** on ALU. Then drag the following signals from *Objects to Wave*: **cntrl_alu_i, reg_a_i, reg_b_i, and result**. These are the minimum required signals to verify the operation. You may add any other signals from any instance you find relevant.

32. To run the simulation, **click the Run-All** button, as shown in Figure 13. The simulation will stop at the first stop command so you can review the results for the first test case. To continue, **click Run-All** again.

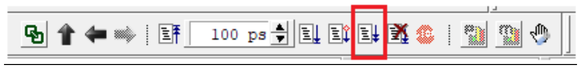


Figure 13. Run Simulation

33. To restart the simulation, click the Restart button, as shown in Figure 14.

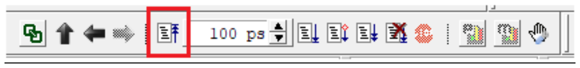


Figure 14. Restart Simulation

34. You may modify the testbench at any time. After making changes, simply recompile and repeat the simulation process starting from Step 26.

35. For easier reading, you can display the signals in hexadecimal format, as shown in Figure 15.

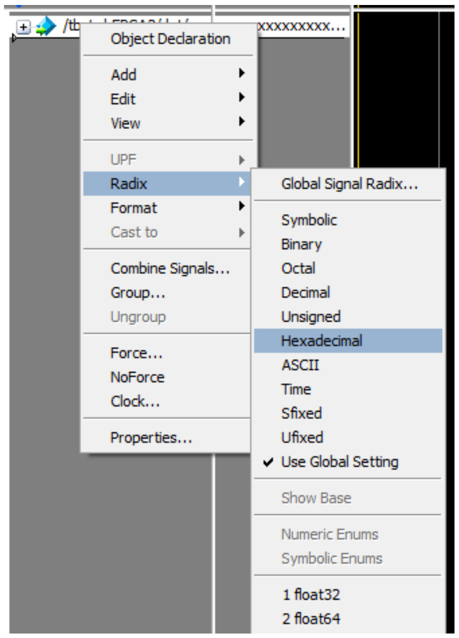


Figure 15. Hexadecimal Signal

36. Run at least **six test cases** as proposed in your testbench to validate the functionality of the ALU.
37. Verify that the results obtained from the FPGA match the simulation results obtained in ModelSim.

Additional Material

38. Create a new Quartus project as done in the first part of this lab. This time, only add the file **LabFPGA2_extra.sv** instead of LabFPGA2.sv.
39. Compile the design.
40. **Import the same .qsf file** you used previously to assign the pin locations.
41. Recompile the design and program the FPGA.
42. This version contains an error, but it is **not caused by a syntax or logic mistake in the code**. Identify the cause of the error and explain it.