

Lab 2 – Behavioral Simulation

CS 2052 Computer Architecture

Dept. of Computer Science and Engineering, University of Moratuwa

Learning Outcomes

In this lab, we will learn how to simulate a logic circuit using the software. After completing the lab, you will be able to:

- design and develop a simple logic circuit using schematics
- verify its functionality via simulation
- verify its functionality on the development board

Introduction

A logic simulator allows us to observe the outputs of a circuit in response to all possible combinations of inputs before the circuit is implemented in hardware. Simulating a circuit is perhaps the best technique an engineer can use to ensure that all required features are present, and there are no unintended behaviors. For larger circuits, simulation is less error-prone, time-saving (due to simplicity in debugging), and far cheaper option than designing and testing a hardware prototype. If errors are observed in the simulator's output, the circuit can easily be corrected and re-simulated as often as necessary.

The Xilinx XSim simulator allows simulations to be run from the Xilinx Project Navigator. This process involves two steps. In step one, we create the desired logic circuit. In step two, the circuit is tested by providing a set of stimulus values that define all input logic.

Building the Circuit

In this lab, we will build a logic circuit that can indicate whether all generators of a power station are correctly functioning. Consider a power station with 3 generators. Suppose the three generators are providing power to a city and any 2 generators out of the 2 can provide enough power. The third generator is added for redundancy in case one generator fails. However, a single power station cannot provide enough power for the city.

Suppose working status (i.e., on and off) of each power station is indicated using a switch, where switches **A** (SW2), **B** (SW1), and **C** (SW0) represent the three generators. Green light should be on when all three generators of functioning correctly. Whereas Amber light should be on only when 2 generators are functioning correctly. Red light should come up if less than 2 generators are functioning correctly. As LEDs on BASYS 3 do not support multiple color combinations, let us assume LD0 to LD2 indicate the Green, Amber, and Red lights.

Step 1: Logic Equation

Use a suitable mechanism find the simplified Boolean representations for the state of the 3 LEDs.

Step 2: Building the Circuit

Create a new project in Xilinx Vivado. Name the project as **Lab 2**.

Tips Review the steps given in Lab 1, if you do not remember how to perform some of the tasks in building a circuit or what parameters to use.

Create the logic circuit using suitable components. Name the circuit as **Circuit2**.

You may use VHDL operators such as AND, OR, NOT, (), etc. You may also use Boolean variables to maintain intermediate values.

Create a Design Constraints File based on **Basys3Labs.xdc** that defines how to connect the switches and LEDs to the correct pins.

Synthesize and make sure there are no errors in your design.

Behavioral Simulation Using XSim

While the task looks relatively simple, you may have already realized corresponding Boolean equations are not much simplified. Therefore, this is a good case to simulate the behavior of the circuit before being implemented on the BASYS 3.

Step 1: Generating Simulation File

To add a new or existing source file to a project, click on the **+** button or press **Alt + A** in the **Sources** pane of the **Project Manager** panel.

This will open up the Add Sources wizard dialog box. Select **Add or create simulation sources** and click **Next >** button.

Then in the next dialog box click on the **Create File** button. From the popup select **VHDL** as the **File type** and set **Circuit2Sim** as **File name**. Then click **Ok** button to close the popup, and then click **Finish** button to close the dialog box.

While creating the Simulation file we do not need to indicate the inputs and outputs. Just click **Ok** button. Click **Yes** from the Message box.

You can open the new VHDL file by double-clicking **Circuit2Sim(Behavioral) (Circuit2Sim.vhd)** by expanding **Simulation Sources** under Project Manager – Sources.

Step 2: Modify Test Bench File

Simulation file is called the Test Bench File. The VHDL test bench has the same structure as any VHDL design source code.

First, we need to indicate the test bench file to use our Circuit2. For that add the following to the test bench file (add it after the line which says `architecture Behavioral of Circuit2Sim is`):

```

COMPONENT Circuit2
  PORT(  g0, g1, g2 : IN STD_LOGIC;
        g, a, r    : OUT STD_LOGIC);
END COMPONENT;

```

Note that the generators (i.e., inputs) are labeled as *g0*, *g1*, and *g2* and LEDs (i.e., outputs) are labeled as *g*, *a*, and *r*. Instead of these labels, you need to use the appropriate input and output names/labels from your *Circuit2*. Exact alignment of terms is not needed. It is just added to make it easier to read.

Let us define internal variables with the same name as follows (add after `END COMPONENT;` line):

```

SIGNAL g0, g1, g2    : std_logic;
SIGNAL g, a, r      : std_logic;

```

Then we need to map the inputs and outputs used in our simulation to input and output ports on *Circuit2*. For that add the following lines after the `begin` keyword:

```

UUT: Circuit2 PORT MAP(
  g0 => g0,
  g1 => g1,
  g2 => g2,
  g  => g,
  a  => a,
  r  => r
);

```

UUT stands for unit under test.

During the simulation, we can simulate a specific set of input combinations and observe the corresponding output. However, as we have only 3 inputs, it will result in only 8 possible input combinations. Therefore, let us simulate and observe the outputs for all possible input combinations (in later labs may be asked to simulate only a subset of the input combinations to test your circuit).

Add the following to the test bench file after defining UUT:

```

process
begin
  g0 <= '0';          -- set initial values
  g1 <= '0';
  g2 <= '0';

  WAIT FOR 100 ns;   -- after 100 ns change inputs
  g2 <= '1';

  WAIT FOR 100 ns;   --change again
  g1 <= '1';
  g2 <= '0';

  WAIT FOR 100 ns;   --change again
  g2 <= '1';

  WAIT FOR 100 ns;   --change again
  g0 <= '1';
  g1 <= '0';
  g2 <= '0';

```

```

WAIT FOR 100 ns;    --change again
g2 <= '1';

WAIT FOR 100 ns;    --change again
g1 <= '1';
g2 <= '0';

WAIT FOR 100 ns;    --change again
g2 <= '1';

WAIT; -- will wait forever
end process;

```

Save the file.

Step 4: Simulating Circuit

Click on **Run Simulation** under SIMULATION in Flow Navigator.

After a few seconds Vivado will open up a window similar to Fig. 1.

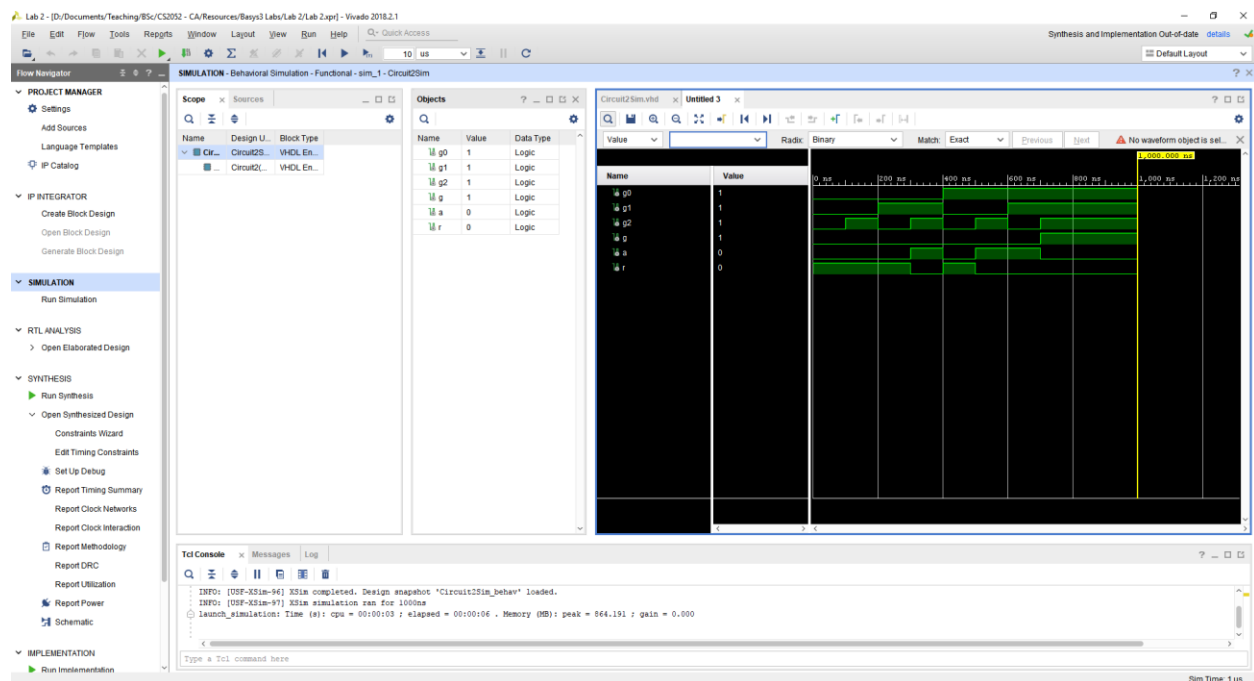


Figure 1 – XSim window.

XSim window has four components:

1. Source Files panel – use to select source files to be viewed
2. Objects panel – use to add different signals to the simulation
3. Simulation panel – use to observe the state of signals
4. Console panel – accept commands from the user

Use the Zoom buttons on the toolbar (especially **Zoom to Full View** button) to zoom the timing diagram. XSim by default simulates the circuit in 1 ps intervals. Hence, zooming is essential to see the changes in inputs and outputs clearly. You may also use **Go To Time 0** and **Go to Latest Time** buttons to navigate

further. Note how the inputs and the corresponding output changes. Do all the outputs confirm the correct functionality of your circuit? If not, double check your Boolean equations. Then re-synthesis and simulate again.

Step 5: Test on BASYS 3

Generate the programming file (i.e., bitstream) and load it to the BASYS 3 board. Change the switches (SW0 – SW2) on the board and verify the functionality of your circuit (check the outputs on LED LD0 – LD2). Demonstrate the circuit to the instructor and get the Lab Completion Log signed.

Step 6: Lab Report

You need to submit a report for this lab. Your report should include the following:

- Student name and index number. Do not attach a separate front page
- State the assigned lab task in a few sentences
- Steps showing how you derived the simplified Boolean representation
- Full VHDL design source code
- Schematic circuit from Vivado
- Test bench code
- Timing diagram from XSim showing all possible inputs and outputs
- Conclusions from the lab

Submit the lab report at the beginning of the next lab.

Tips If you do not have access to a printer while working on Vivado, you may save/print schematics and timing diagrams as PDF files and later print them.

Bibliography

- Jim Duckworth, Using Vivado to create a simple Test Bench in VHDL.
- Xilinx, “Vivado Tutorial – Lab Workbook,” 2015.

Prepared By

- Dilum Bandara, PhD – Feb 04, 2014.
- Last Updated on Sep 27, 2018

Note This lab is not compatible with the previous version of this document as this replaces BASYS 2 with BASYS 3, ISE with Vivado, and schematic design with VHDL.