Lab1: Introduction to Digilent Basys2 board and Xilinx ISE

Leonardo Fusser (1946995)

Objectives:

- Familiarize with features and function of Digilent Basys2 board.
- Familiarize with Xilinx ISE Design Suite.
- Enter a gate level logic circuit design in Verilog HDL form.

Material: Basys2_manual.pdf, Basys2_sch.pdf, Xilinx ISE

To hand in:

- This sheet with answers and screenshots.
- Commented Verilog code modules: mux module and fixture module.

All documents must be uploaded to **Teams Assignments**.

Xilinx ISE installation:

Most of the subsequent labs of this course will focus on FPGA programming based on Digilent Basys2 board, which uses the Xilinx Spartan-3E FPGA.
Download both schematic and reference manual Basys2 from Teams.

You **must** download necessary document about Xilinx ISE Design Suite 14.6.
To install Xilinx ISE WebPACK (free software) at home, get the following file from Teams – files tab - and follow the steps:
Xilinx ISE Installation_v4.pdf

It takes a minimum of 60 minutes to download and 30 minutes to install.

Lab Work:

Part 1: Familiarize with Basys2 board

- Read through the reference manual of the Basys2 board and understand the basic features and functions.
- Obtain the **board schematic** and identify the following components by **circling and labeling** them on the photo below. Write the reference designator in the spaces following the components in Figure 1.

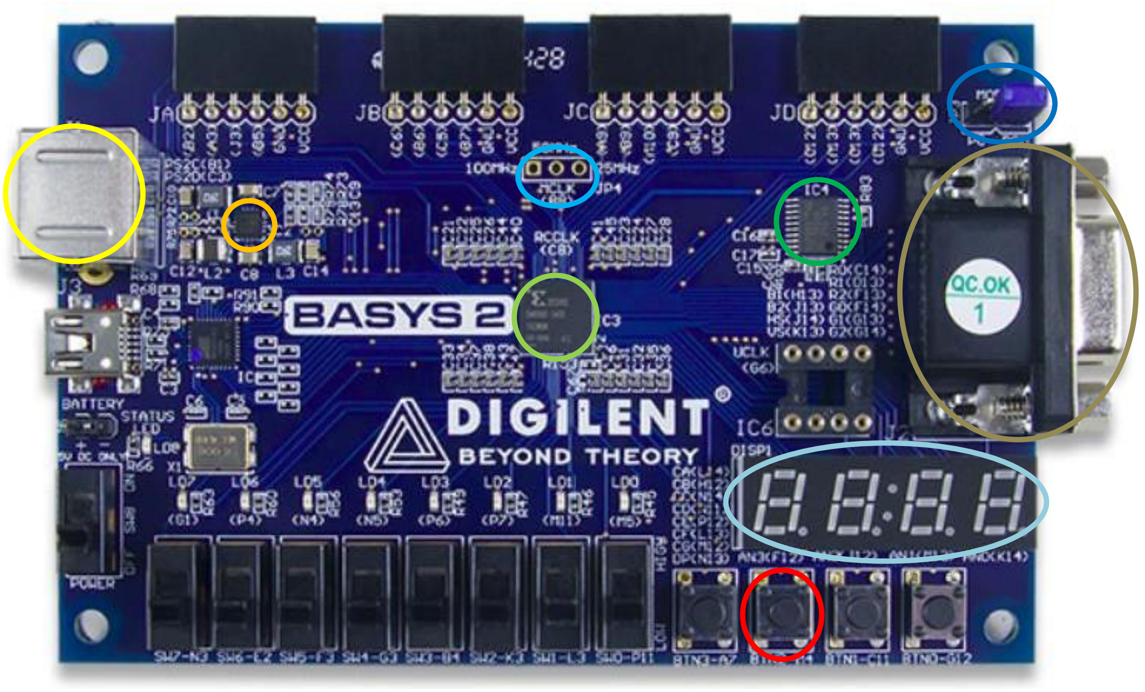


Figure 1 Basys2 board overview

Item	Reference designator
Push button no. 2	See red circle above.
3.3 V voltage regulator	See orange circle above.
PS/2 port	See yellow circle above.
The FPGA	See light green circle above.
The platform flash ROM	See green circle above.
Oscillator MCLK	See light blue circle above. MCLK chip is behind board.
The mode select jumper	See blue circle above.
Any LED	See aqua circle above.
VGA port	See tan circle above.

Using the schematics, answer the following questions:

Are the SPDT switches (SW) connected to pull-up or pull-down resistors? If neither of them, explain how they are connected.
The SPDT switches (SW0 to SW7 in schematics) are neither connected to pull-up nor pull-down resistors. Instead, they are connected to both depending on the position of the switch. For each of the switches, one terminal is connected to a resistor that goes to the FPGA (any of the Rx resistors – R30, R31, R33, R34, R36, R37, R39, R40 in schematics), another terminal is connected to Vcc and the other terminal is connected to GND. Depending on the position of the switch, the resistor could behave like a pull-up (if connected to Vcc) or could behave like a pull-down (if connected to GND). In other words, the switches are placed in an active-low and active-high configuration depending on their connected position in the circuit.

Are the BTN signals at the output of IC8A to IC8D active high or low? Explain
The BTNx signals (BTN0 to BTN3 in schematics) looks to be active low at first glance, but because there are inverting Schmitt triggers in the circuit (IC8A to IC8D in schematics), they are instead active high. If the inverting Schmitt triggers were not present, a press on any of the pushbuttons (SW-P8008: BTN0 to BTN3 in schematics) would cause the corresponding BTNx signal to go low and when released, the corresponding BTNx signal would go high. Since there are inverting Schmitt triggers in the circuit, the behavior of the BTNx signals when the pushbuttons are pressed/depressed is the opposite of what was just explained. In other words, because there are inverting Schmitt triggers in the circuit, the BTNx signals connected to the pushbuttons are active high.

Are the AN signals active high or low? Explain.
The ANx signals (AN0 to AN3 in schematics) are active low. This is because of the BJT configuration the ANx signals connect to. Each of the ANx signals connect to the base of a PNP transistor (Q1A, Q1B, Q2A and Q2B in schematics). For each BJT configuration, the emitter is connected to Vcc and the collector is connected to the anode of the 7-segment display (A1 to A4 in schematics). In order for the display to work, the base of the PNP transistor must be brought low. This is because the emitter-base junction must be properly forward biased in order for the display to work (emitter has to be positive and base has to be 0V). In other words, the PNP transistors have to be operating in the "closed" position in order for the display to light up properly, so the ANx signals must be active low.

Give the crystal value:
When using the onboard crystal, the user can select three different frequencies by manipulating the JP4 header (located directly above the FPGA chip). The three different frequencies that can be chosen are 25MHz, 50MHz or 100MHz. The default frequency used is 50MHz (there is no header by default on the Basys2 board).

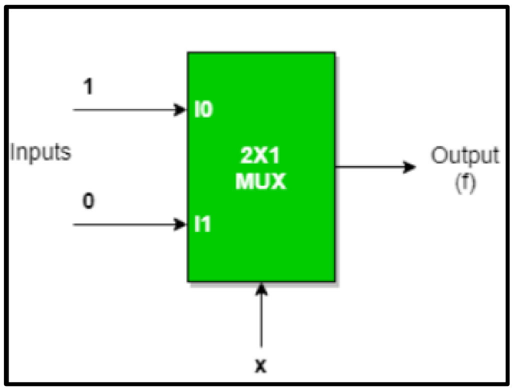
On the schematics, how many rectangles are needed to represent the FPGA?
A total of 6 rectangles are needed to represent the FPGA (IC3A to IC3F in schematics). Due to the massive amount of pins the FPGA has, it is broken down into smaller rectangles that span across multiple pages in the schematics. The FPGA has a maximum of 108 user I/Os. The manufacturer chip number for the Basys2 FPGA is XC3S100E-CP132.

Give the manufacturer chip number for the flash ROM:
The manufacturer chip number for the Basys2 flash ROM is XCF02S.

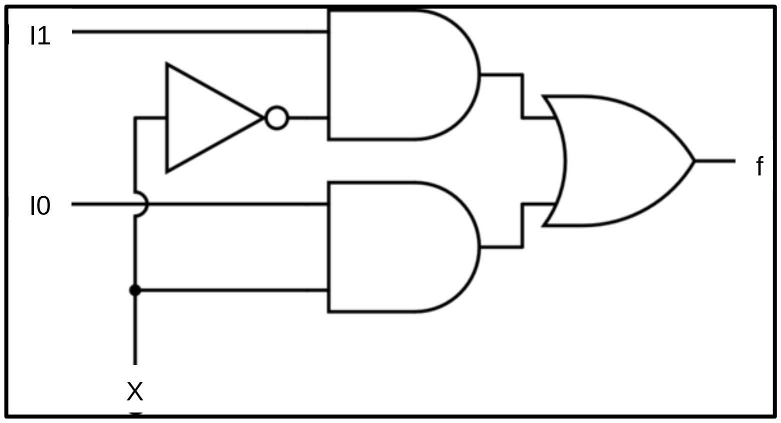
Part 2: gate level logic circuit design in Verilog

Write a Verilog module for a 1 bit 2-1 MUX.

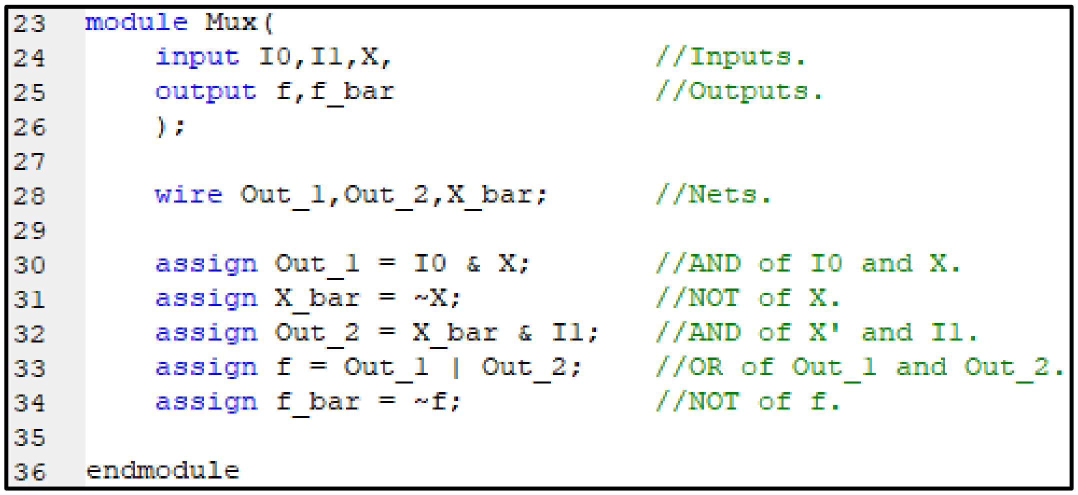
a) Draw the MUX symbol with its I/Os.



b) Draw the gate level circuit (structural)



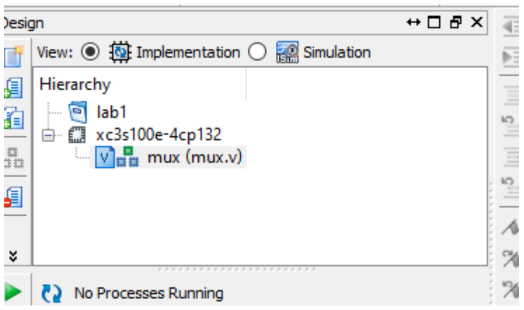
c) Write the Verilog module corresponding to the circuit (structural)



```
module Mux(
    input I0,I1,X,                  //Inputs.
    output f,f_bar                  //Outputs.
    );

    wire Out_1,Out_2,X_bar;         //Nets.

    assign Out_1 = I0 & X;          //AND of I0 and X.
    assign X_bar = ~X;              //NOT of X.
    assign Out_2 = X_bar & I1;      //AND of X' and I1.
    assign f = Out_1 | Out_2;       //OR of Out_1 and Out_2.
    assign f_bar = ~f;              //NOT of f.

endmodule
```

- Implement the logic of your mux circuit in Verilog HDL.
- Create an ISE project named Lab1 for your board. Choose “HDL” as your top-level source type.
- In the project settings you will need to select details that related to Basys2 board as follows:

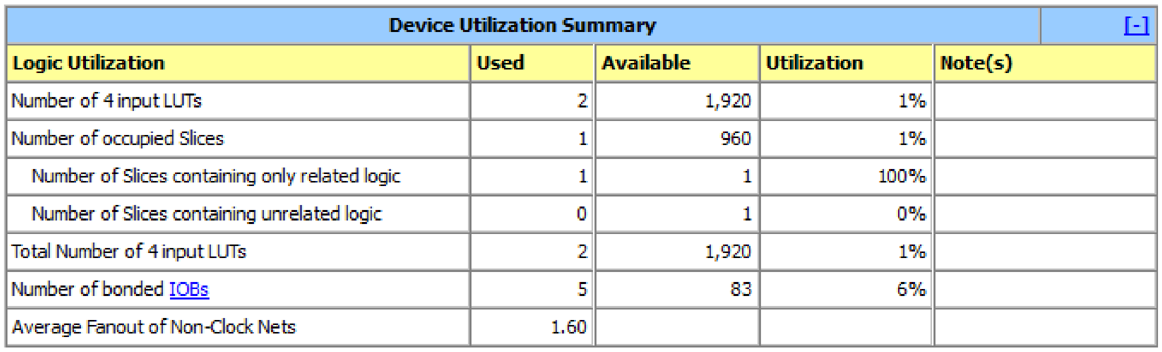
  - Family: Spartan 3E
  - Device: XC3S100E
  - Package: CP132
  - Speed: -4 or -5



Synthesis and Implement the project by double-clicking on “Implement Design”.

From the Design Summary/Report, find out how many resources that have been used by your design, including number of LUTs, IOBs, Slices etc. Comment on the resources used. Are they as expected?

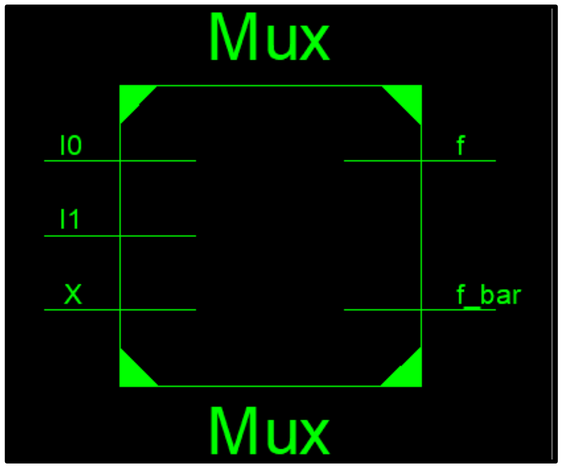
Looking at the design summary/report, 2x 4 input LUTs, 5x IOBs and 1x Slice are the partial elements of resources being used. Complete elements of recourses being used can be found in the table below.



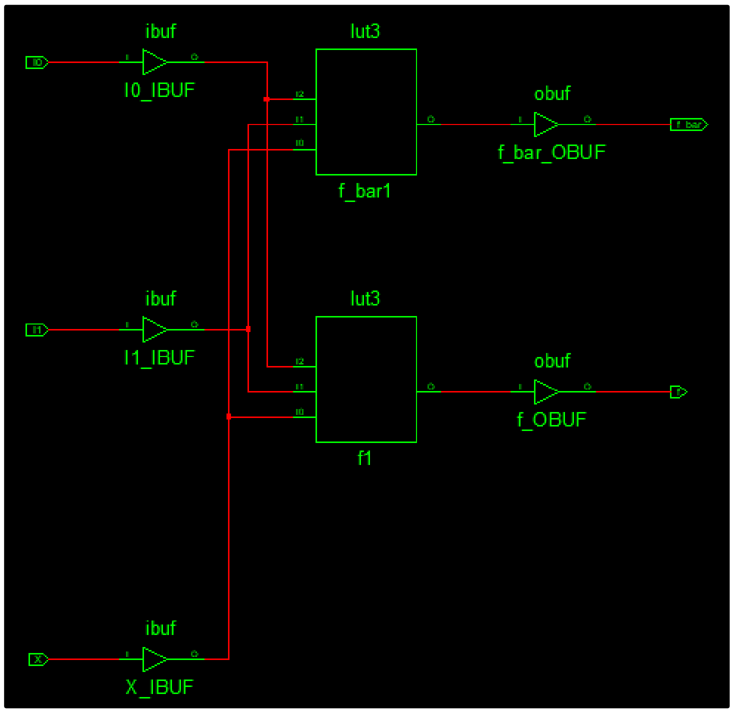
Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	2	1,920	1%	
Number of occupied Slices	1	960	1%	
Number of Slices containing only related logic	1	1	100%	
Number of Slices containing unrelated logic	0	1	0%	
Total Number of 4 input LUTs	2	1,920	1%	
Number of bonded IOBs	5	83	6%	
Average Fanout of Non-Clock Nets	1.60			

Screenshot taken from Xilinx ISE. Table above shows complete list of resources being used.

- Click on "View Technology Schematics" to create a technology schematic. **Take a screenshot.** Also, you must include a screenshot of at least one LUT's internal gates (right click on the LUT to see the content)



Top-level schematic view of the 2x1 MUX implemented in Xilinx ISE shown above.

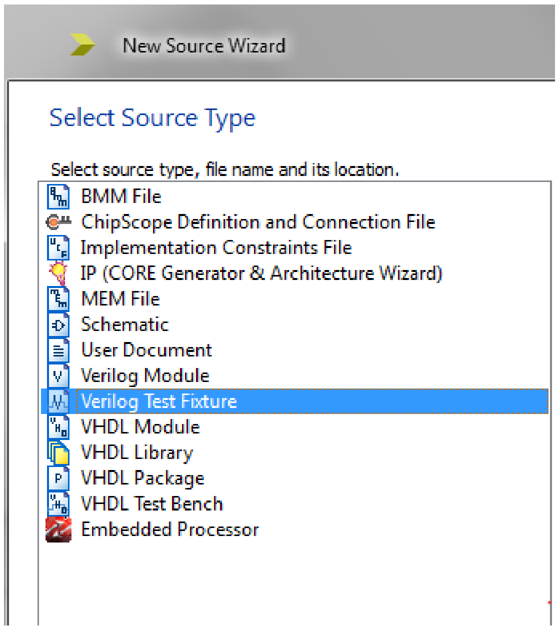


Detailed view of FPGA internals in schematic view of the 2x1 MUX implemented in Xilinx ISE shown above.

Part 3: Test Bench

- Create a Verilog testbench for your design

  a) Right-click on the project and select “`New Source…`”. Choose “`Verilog Test Fixture`” as the source type, as shown below.



New Source Wizard

Select Source Type

Select source type, file name and its location.

- BMM File
- ChipScope Definition and Connection File
- Implementation Constraints File
- IP (CORE Generator & Architecture Wizard)
- MEM File
- Schematic
- User Document
- Verilog Module
- Verilog Test Fixture
- VHDL Module
- VHDL Library
- VHDL Package
- VHDL Test Bench
- Embedded Processor

  b) Fill in necessary information via the wizard, and a testbench template will be created.

c) Edit your testbench accordingly. Ensure you have the following sessions:

i. Instantiate the circuit under test and connect all ports accordingly. Example:

```
// Instantiate the Unit Under Test (UUT)
mux uut (
      .A(A),
      .B(B),
      .S(S),
      .Y(Y)
);
```

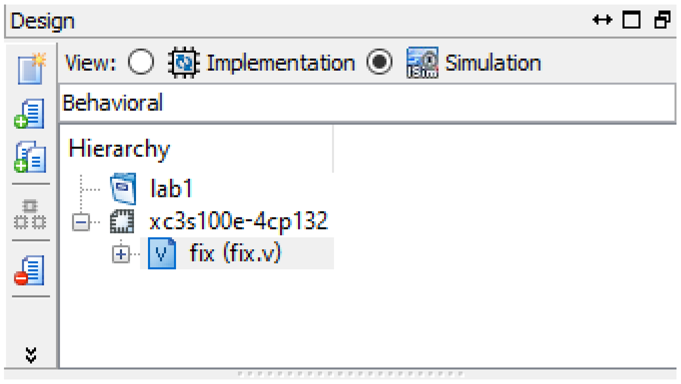
Generate 8 input stimuli to cover all the test cases.

```
#1; // delay 1 nS
A=0;
B=0;
S=0;
#1;
A=1;
B=0;
S=0;
#1;
A=0;
B=1;
S=0;
#1;
A=1;
B=1;
S=0;
#1;
A=0;
B=0;
S=1;
#1;
A=1;
B=0;
S=1;
#1;
A=0;
B=1;
S=1;
#1;
A=1;
B=1;
S=1;
```

Apart from using delay command, you may also use a *for* loop to generate continuous input bit pattern for simulation.

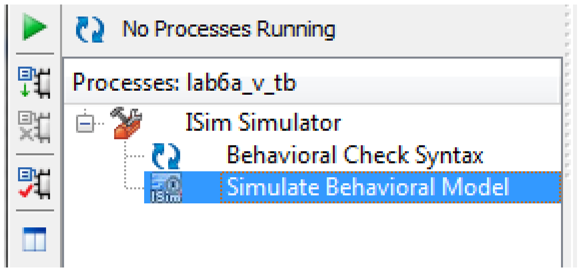
- Now your design is ready for behavioral simulation using ISim.

a) In Design panel, select Simulation, and select Behavioral from the drop-down list.

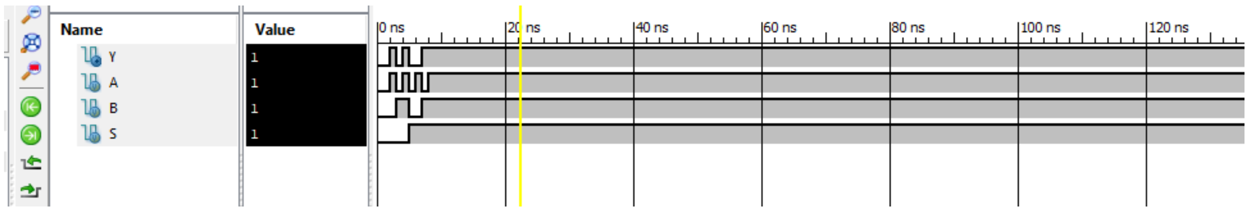


b) In the Hierarchy pane, select your testbench file.

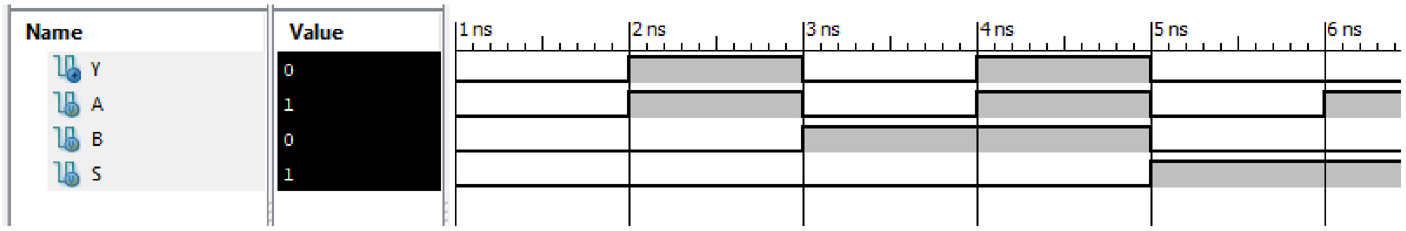
c) In the Processes pane, expand **ISim Simulator**. To start the behavioral simulation, double-click `Simulate Behavioral Model`.



d) An ISim (a separate application) window will open looking like this:

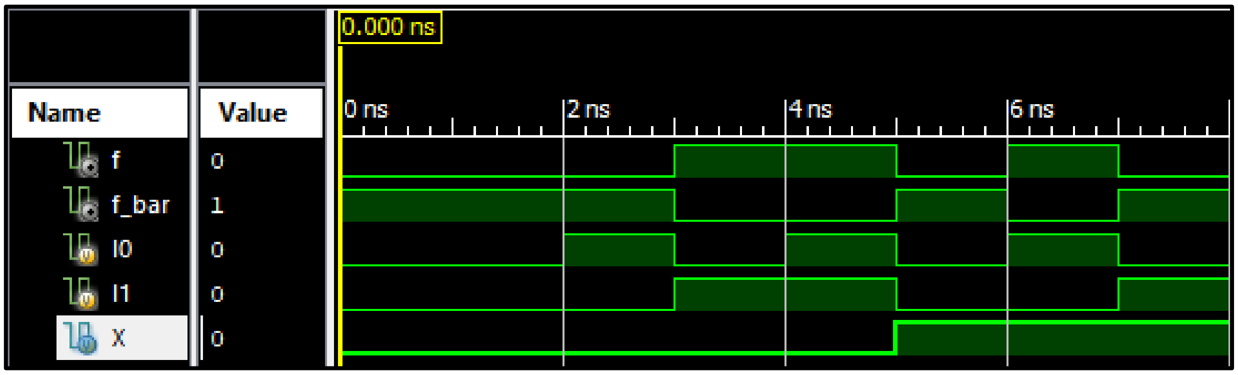


e) Click on the zoom to full view button and the view will show the details of the waveform. Expand any bus signals if necessary. Example :



Verify your results of your simulation. Is this the same as your truth table?

Looking at the results from the simulation, they match the 2x1 MUX truth table that was implemented above. Below are the truth table and timing diagram for the 2x1 MUX that was implemented above.



Screenshot taken from Xilinx ISim. Timing diagram shows expected behavior for a typical 2x1 MUX.

Inputs			Outputs
X	I1	I0	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Complete truth table for a typical 2x1 MUX shown above.

After lab questions:

Explain what you learned in this lab. What went wrong and what did you learn from your mistakes.

The primary takeaway from this lab was that I now know how the Basys2 board works (after reading manual and schematics) and also how to program for the Basys2 FPGA in Verilog (after reading notes and other references). I also learned how to use the Xilinx ISE and Xilinx ISim software for programming the Basys2 FPGA in Verilog (along with learning basic Verilog programming concepts). The main issue encountered was understanding the functioning of some of the Basys2 peripherals (switches, pushbuttons, etc…), but after some closer examination, this was resolved. Other issues encountered were with the Xilinx ISim software (strange error would always appear upon startup of ISim), but this was due to a technical difficulty and wasn't caused by any of the procedures I conducted in this lab.

Comment on the "View Technology Schematics"

Upon launching the schematic for the 2x1 MUX design, there were various options to choose from to produce my schematic. For example, there were options to view it as top-level or very detailed FPGA view (with LUTs being visible). There is a lot of flexibility for the user to configure how their schematic would look like. For my particular schematics, I choose to view both the top-level and detailed FPGA view. In the top-level view, only the basic overview of how the 2x1 MUX is shown (simple view). On the other hand, in the detailed FPGA view, more details about the 2x1 MUX are shown (LUTs, for example, are shown). See the two 2x1 MUX schematics under Part 2 for more details.