CS152B Lab 0 – ISE TUTORIAL

Required Equipment

FPGA Board	FPGA Power Supply	Micro USB Cable connected to DIGILENT port	Locker	Access Card

Introduction
The purpose of this lab is to make you comfortable with the Xilinx programming environment and the steps needed to download your design to the board and run it.

Instructions
a. Setting up ISE
Open ISE (the 64 bit edition from either the shortcut on the desktop or from the start menu). Type in a project name and make sure Top-level source type says HDL. Make sure the Xilinx Virtex5 platform is selected in the project hardware configuration.

B. Adding Source Files
Right click on the FPGA in the "hierarchy window" with the "simulation" radio button selected and choose New Source. Choose a Verilog Module named "counter" as shown below. Then enter the following Code:

Figure 1: counter.v and its testbench from asic-world.com

```
module counter (
clock , // Clock input of the design
reset , // active high, synchronous Reset input
enable , // Active high enable signal for counter
counter_out // 4 bit vector output of the counter
); // End of port list
//-------------IO Ports-----------------------------
input clock ;
input reset ;
input enable ;
output [3:0] counter_out ;
//-------------Input ports Data Type-------------------
// By rule all the input ports should be wires
wire clock ;
wire reset ;
wire enable ;
//-------------Output Ports Data Type------------------
// Output port can be a storage element (reg) or a wire
reg [3:0] counter_out ;

// We trigger the below block
// with respect to positive edge of the clock.
always @ (posedge clock)
begin : COUNTER // Block Name
  // At every rising edge of clock we check if reset is active
  // If active, we load the counter output with 4'b0000
  if (reset == 1'b1) begin
    counter_out <= #1 4'b0000;
  end
  // If enable is active, then we increment the counter
  else if (enable == 1'b1) begin
    counter_out <= #1 counter_out + 1;
  end
end // End of Block COUNTER
endmodule // End of Module counter
```

```
module first_counter_tb();
// Declare inputs as regs and outputs as wires
reg clock, reset, enable;
wire [3:0] counter_out;

// Initialize all variables
initial begin
  $display ("time\t clk reset enable counter");
  $monitor ("%g\t %b   %b     %b      %b",
              $time, clock, reset, enable, counter_out);
  clock = 1;       // initial value of clock
  reset = 0;       // initial value of reset
  enable = 0;      // initial value of enable
  #5 reset = 1;    // Assert the reset
  #10 reset = 0;   // De-assert the reset
  #10 enable = 1;  // Assert enable
  #100 enable = 0; // De-assert enable
  #5 $finish;      // Terminate simulation
end

// Clock generator
always begin
  #5 clock = ~clock; // Toggle clock every 5 ticks
end

// Connect DUT to test bench
counter U_counter (
clock,
reset,
enable,
counter_out
);

endmodule
```

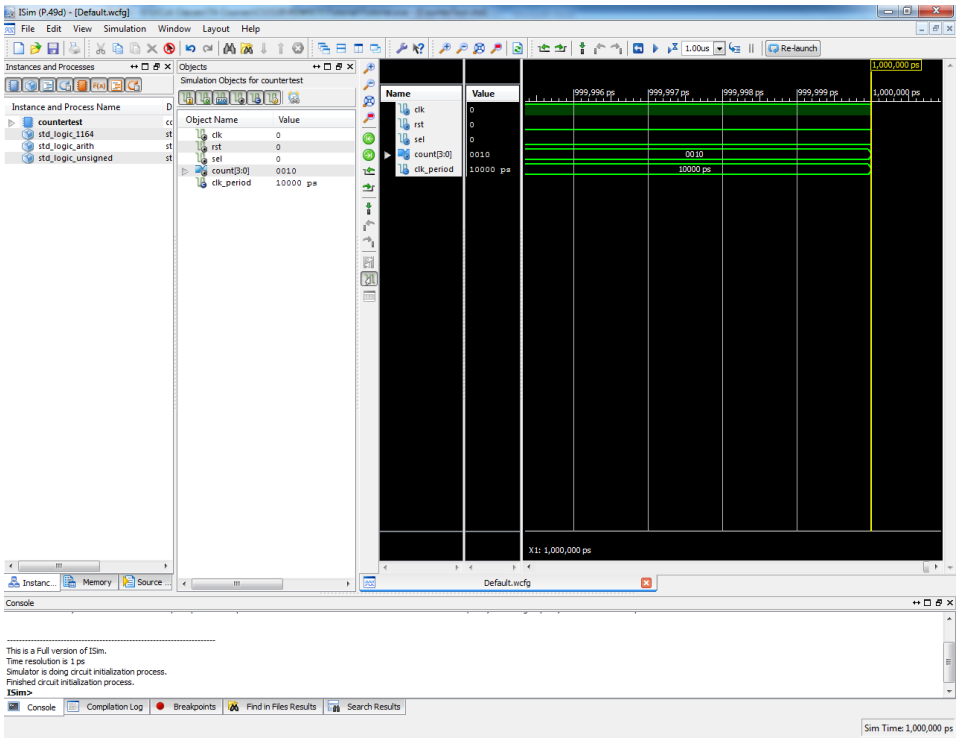
C. Checking Syntax and Simulating Counter
The steps taken to design and implement hardware are as follows: Add Code, Check Syntax, Synthesize, Simulate, User Constraints, Implement Design, Generate Programming File, and Configure Target Device.

In order to simulate, we have to design a test bench to drive the design under test. Right click in the hierarchy to design a Verilog Test Bench. Name it first_counter_tb. Hit next. Select the counter for test design. Hit next. Hit Finish. This will create a test bench based upon the counter design.

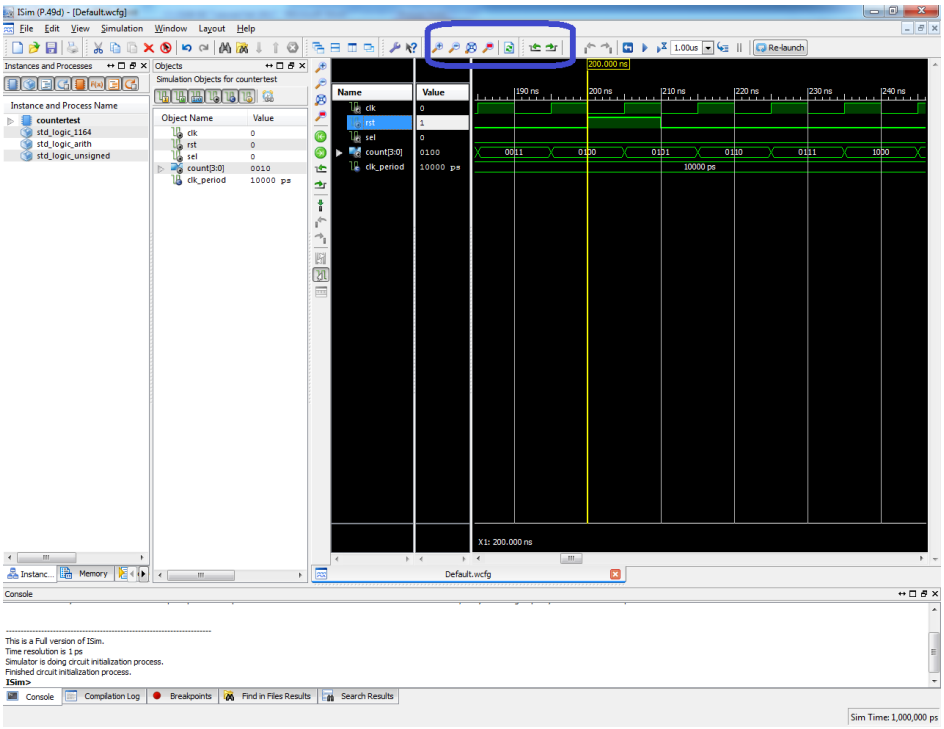
This creates a stub waveform generator where you can drive the input signals and check the output signals. Notice several key points. First, the creation of the test bench that looks like a piece of hardware and the internal signals that represent the ports of our counter. After that is the creation of the piece of hardware being tested the Design under test (or Unit under test). Finally, notice the test bench determines the clock signal and generates a square waveform, along with a stub for designing the test vectors. Please enter the code in Figure 2 to use as your testbench.

Next, you are asked to modify the given testbench file to test more complicated use cases. In other words, change the timing of the signal transitions that are inputs to the counter module.

Now we are ready to run our simulation. We select three things in order. First, we select the Simulation radio button. Then we click on the test bench (NOT the uut). The window below will change to show ISim, the hardware simulator. We expand the options, check the syntax, then select "Simulate Behavioral Model." The simulator window and waveform should launch:

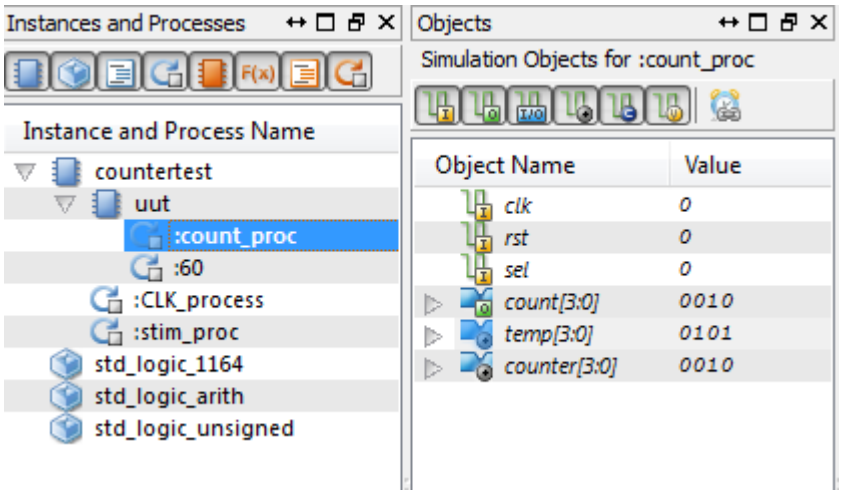


Notice the circled area in the image below:

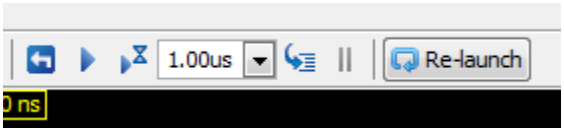


Select in the waveform the beginning of the signal, press the + magnifying glass to zoom in until you can read the signal, then click on rst, and press the green right arrow to move forward until this point in time.

****NOTE** The screenshots in this document are just examples: your list of signals may be different than those shown.**

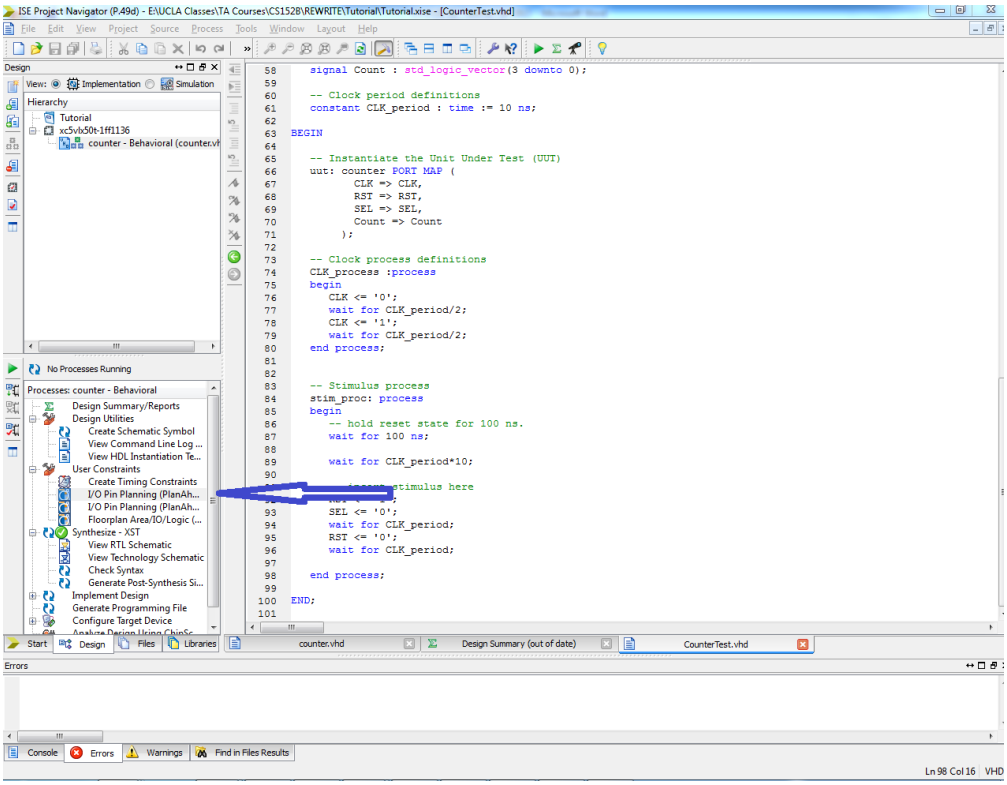


Now, you can expand the counter test on the left, and the uut under it to choose the particular design or processes. Then notice the Objects windows shows not only the ports but the internal signals as well. Notice there are no waveforms for the new signals, because they weren't being monitored while running the simulation initially. Thus use the following controls found at the top of the screen to restart and re-run the simulation.



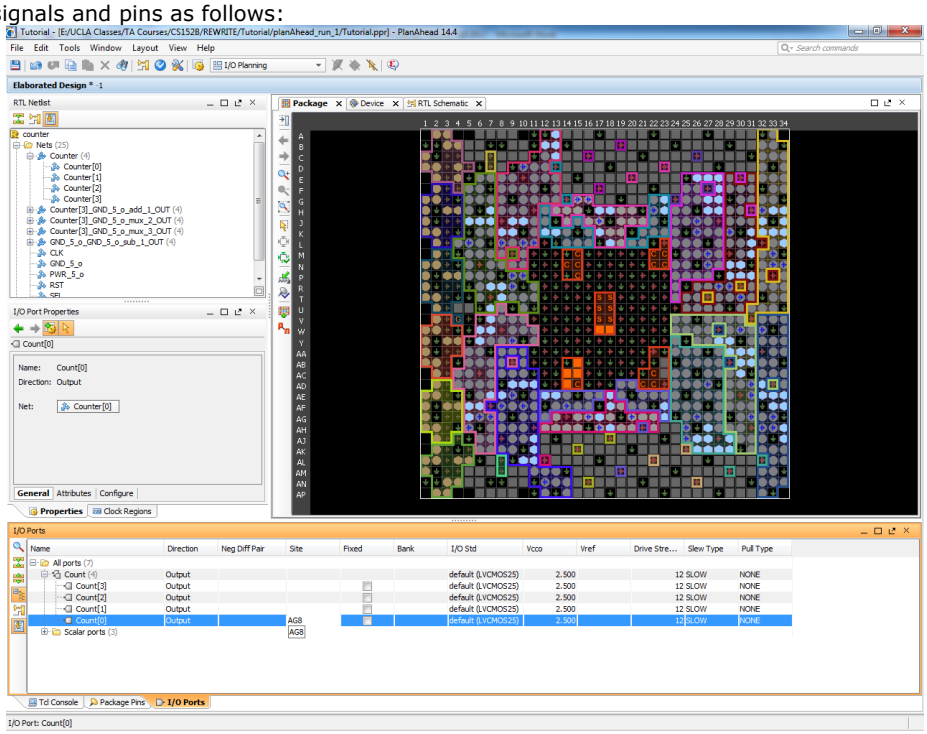
E. Pins

Switch back to implementation from simulation. Open the user constraints option and select I/O Pin Planning - Pre-Synthesis. This will create your user constraints file and allow you to connect the FPGA pins on the board to the clock and the desired switches or push buttons for select and reset.



When you select this, press yes to create your first .ucf file. Wait a bit as it takes the next program some time to launch. We are going to connect the system clock as well as use LD0-3 for the counter, SW0 for RST and SW1 for SEL.

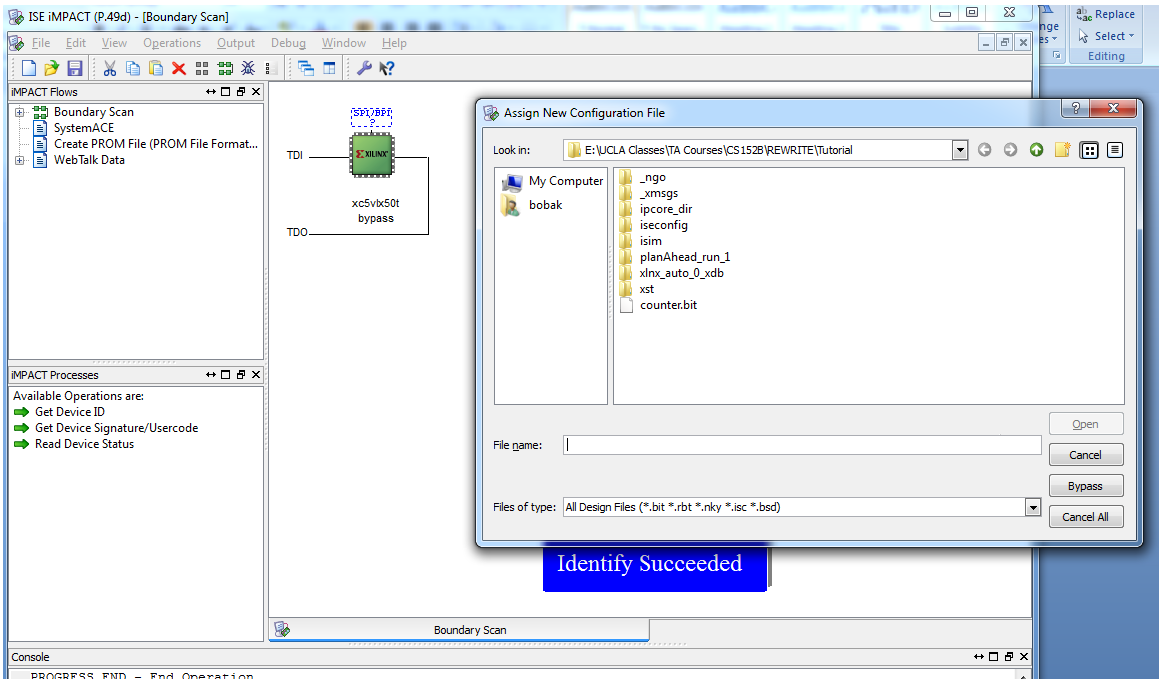
Select the appropriate signals and pins as follows:



Select for Count 0 AG8, Count 1 AH8, Count 2 AH9, Count 3, AG10
Select for RST J19, Select for CLK AG18.
Save your constraints. Exit Plan Ahead. Open the .ucf file in the project to see if the pinning has been assigned.

F. Download Design

Select Implement Design and Generate Programming File. Make sure your Xilinx board is powered on, the USB cable plugged in to the Xilinx USB, and the mode jumper set to JTAG Configuration. Then expand the Configure Target Device option and choose Manage Configuration Project (iMPACT). When impact launches, choose new project. Configure device using Boundary Scan (JTAG). Assign Configuration Files. It should look something like this:



Select Counter.bit for download. It will prompt you about FLASH Proms. Select "no", and hit "ok". Right click on the FPGA and select "Program".

A checkoff is not necessary for this lab. However, please submit a 1-2 page lab report (see syllabus for details) describing your experience with this lab, any problems you encountered, etc.