Programmable Logic Design – I

Read through each section completely before starting so that you have the benefit of all the directions.

Introduction

During the last lab you built simple logic circuits on breadboards using TTL logic circuits on 7400 series chips. This process is simple and easy for small circuits. With increasing complexity of the logic circuitry the possibility of wiring errors grows and it becomes increasingly difficult to debug the circuit. Another problem is the difficulty in finding all the needed logic circuitry on available chips. To address these problems, the electronics industry has developed the concepts of Programmable Logic Devices (PLD's) and of Field Programmable Gate Arrays (FPGA's). The basic idea behind these devices is the notion that logic circuitry of arbitrary complexity can be constructed from simple gates connected with appropriate links and the technical advance that has made this possible is the development of large gate arrays with computer programmable links. The design process then consists of specifying the logic design by means of a logic design language such as VHDL or by entering it on a schematic layout. A computer program then turns this design into a series of instructions that are downloaded into the chip to establish the desired logic circuitry. Facilities are provided to specify the pin out of the logic, to control placement of logic circuits on the chip and to impose timing constraints.

Design Tools

For our designs we will be using the Xilinx Corporation (www.xilinx.com) ISE Foundation 9.2i. We will start by entering our design in the form of a circuit schematic but, in later stages of the labs, we will also use a high level design language, VHDL, to benefit from its power and flexibility. In our designs we will sample only a few of the features and capabilities of this software package which is widely used in the electronics industry today.

Hardware

We will download our designs into a Digilab D2XL board connected to a Digilab Digital I/O board (DI01) shown in Figure 1 below. The FPGA chip on the D2XL board is a member of the Spartan II family, the XC2S30, with package type tq144, embodying 972 logic cells with a total of 30,000 gates. While this size of device was state-of-the-art a few years ago, rapid advances in technology have pushed the largest device sizes to many millions of gates.

The D2XL board's I/O resources are limited to a single pushbutton and one LED for use with a test program to verify proper operation. A large variety of I/O devices, however, are available on the DI01 board attached to the D2XL by means of two 40 pin connectors. Our two experiments will exploit the features of the D2XL/DI01 combination to design a number of circuits that will demonstrate the usefulness of the discussed procedure.

Getting Started

Hooking up the Hardware

The circuitry is extremely delicate and can easily be destroyed if handled improperly. Static electricity which is easily generated is particularly dangerous and care must be taken to wear a grounded wrist strap when handling the circuitry. Your instructor will show you how to use it properly. Your two boards should be connected to one another, with power cord installed and with a programming cable from the parallel port of the PC to the JTAG connector of the D2XL attached. Ask your instructor for help if this is not the case.

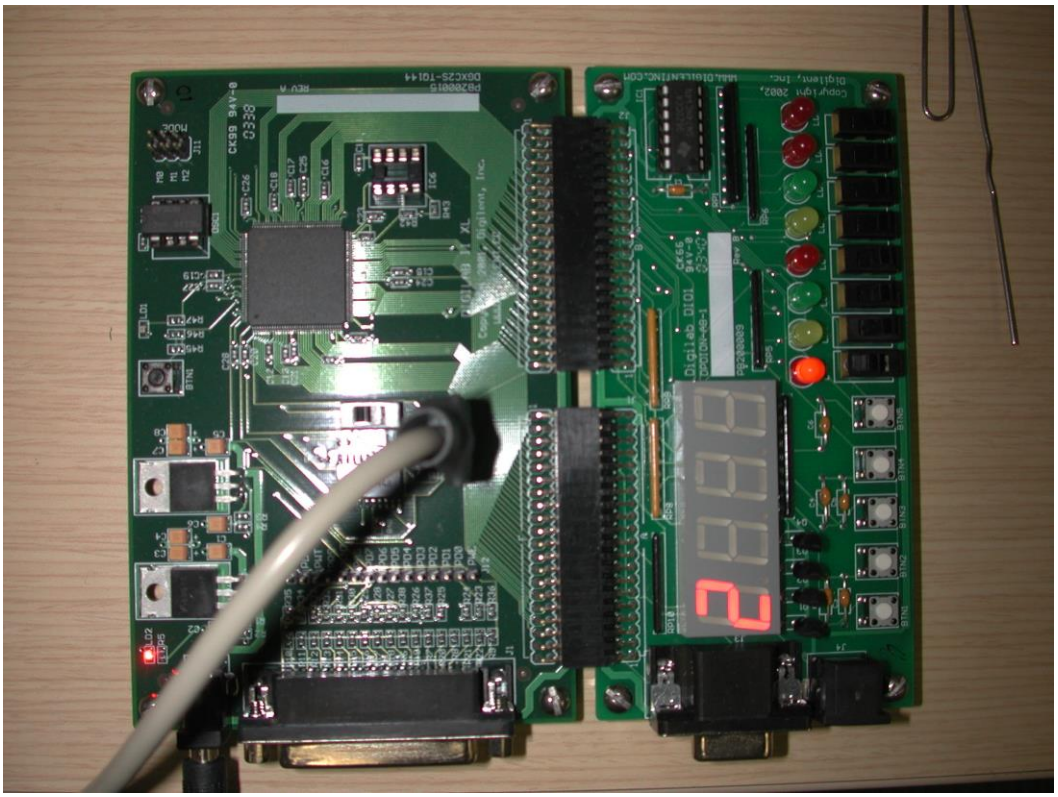
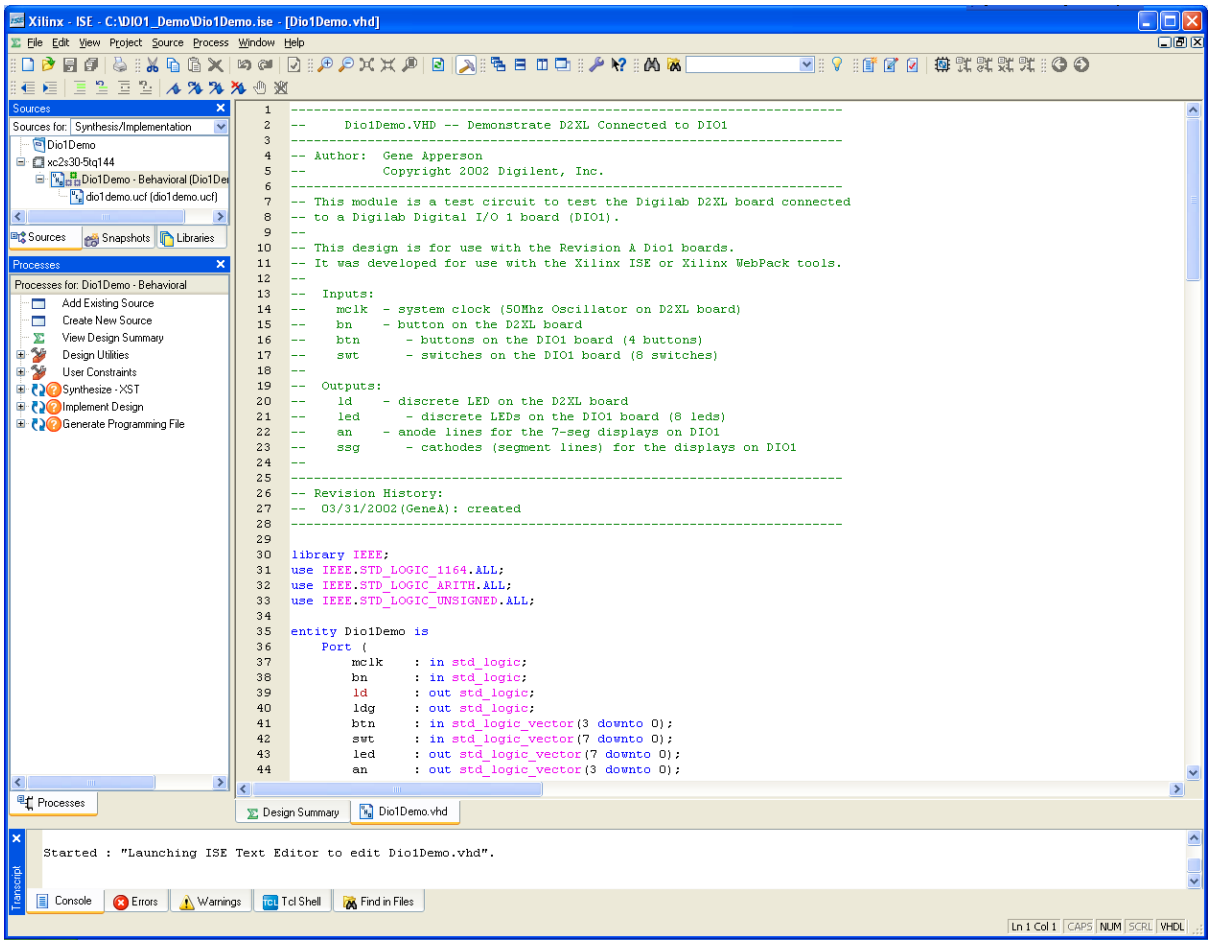


Figure 1: The D2XL and DI01 boards with programming cable attached.

Testing the D2XL board

You should have a Xilinx ISE 9.2i icon on your screen. Double click on it to open the program. As a first program to download we want to use “Di01Demo”(C:/Digilent/ Di01Demo.ise) to test the integrity of the D2XL board and the attached Digital I/O 1 board. If another project automatically opens up, close it from the “File” menu and use




```
1  ----------------------------------------------------------------------------
2  --     Dio1Demo.VHD -- Demonstrate D2XL Connected to DIO1
3  ----------------------------------------------------------------------------
4  -- Author:  Gene Apperson
5  --          Copyright 2002 Digilent, Inc.
6  ----------------------------------------------------------------------------
7  -- This module is a test circuit to test the Digilab D2XL board connected
8  -- to a Digilab Digital I/O 1 board (DIO1).
9  --
10 -- This design is for use with the Revision A Dio1 boards.
11 -- It was developed for use with the Xilinx ISE or Xilinx WebPack tools.
12 --
13 --  Inputs:
14 --    mclk  - system clock (50Mhz Oscillator on D2XL board)
15 --    bn    - button on the D2XL board
16 --    btn      - buttons on the DIO1 board (4 buttons)
17 --    swt      - switches on the DIO1 board (8 switches)
18 --
19 --  Outputs:
20 --    ld    - discrete LED on the D2XL board
21 --    led      - discrete LEDs on the DIO1 board (8 leds)
22 --    an    - anode lines for the 7-seg displays on DIO1
23 --    ssg      - cathodes (segment lines) for the displays on DIO1
24 --
25 ----------------------------------------------------------------------------
26 -- Revision History:
27 --  03/31/2002(GeneA): created
28 ----------------------------------------------------------------------------
29
30 library IEEE;
31 use IEEE.STD_LOGIC_1164.ALL;
32 use IEEE.STD_LOGIC_ARITH.ALL;
33 use IEEE.STD_LOGIC_UNSIGNED.ALL;
34
35 entity Dio1Demo is
36     Port (
37         mclk    : in std_logic;
38         bn      : in std_logic;
39         ld      : out std_logic;
40         ldg     : out std_logic;
41         btn     : in std_logic_vector(3 downto 0);
42         swt     : in std_logic_vector(7 downto 0);
43         led     : out std_logic_vector(7 downto 0);
44         an      : out std_logic_vector(3 downto 0);
```

Figure 2: Project Navigator

“Open Project” from the same menu to open “Di01Demo”. If you are successful you should get a screen display like Figure 2.

You actually need to double click on the 3rd line in the top left window to get the display in the top right window. The above four windows represent the design environment for the project with “Sources” in the top left window, the “Processes” for a given Source below it, the contents of selected files in the right hand window and text files below. On top of the “Sources” window is a tab to select the type of sources to be displayed. In this instance the sources are those associated with “Synthesis/Implementation” but we will also be interested with sources for “Behavioral Simulation” there.

Before proceeding we need to check that we have selected the proper chip and simulation software. If you right-click the xcs230-tq144 icon in the "Sources" window and select "Properties" you should trigger a popup window such as shown below:

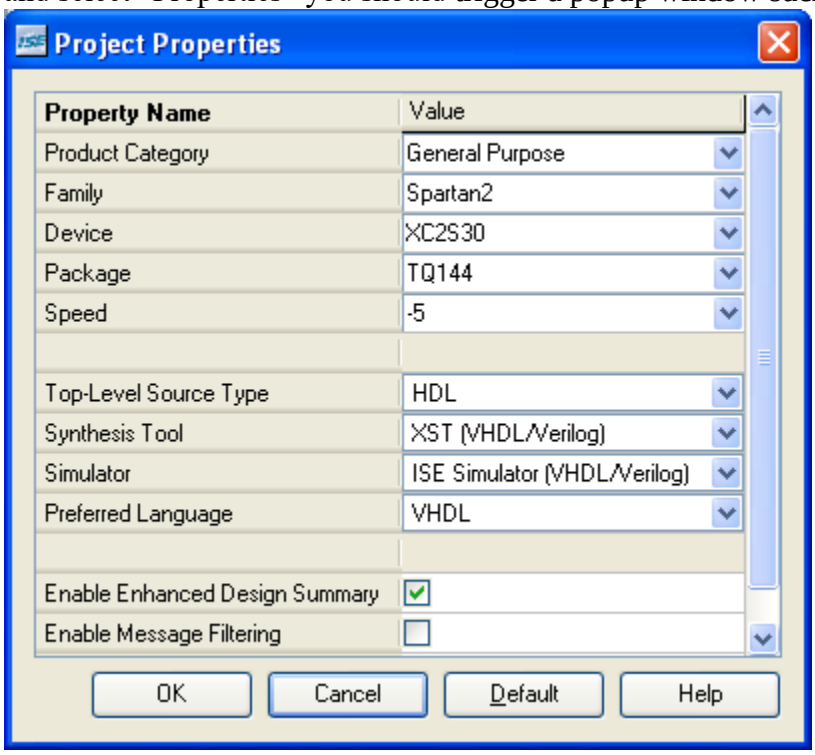


Figure 3: Project Properties

Check that the Device and Package types are correctly selected, i.e. such as above. Specifically, under "Simulator" choose "ISE Simulator (VHDL/Verilog)" and under "Preferred Language" choose "VHDL". When you are satisfied, close the window.

Extensive help files are available online and at this time it would be a good idea to go to the "Help" window, select "Help Topics, FPGA Design, FPGA Design Flows" and read the chapter "FPGA Design Flow Overview".

To test our board we are going to select the "Synthesis/Implementation" in the "Sources" window, and click on "Di01Demo-Behavioral". Expand the "Generate Programming File" under the "Processes" window and then double click on "Configure Device (iMPACT)". Doing this puts the program through all of the necessary steps to generate the file for programming the FPGA and to initiate the downloading (to FPGA) process.

*If you are prompted to do so, choose the Configure devices using Boundary Scan bullet and select "Automatically connect a cable and identify a Boundary Scan chain". You will then be asked to assign a new configuration file, choose "dio1demo.bit" [in other instances in the following experiments where you will create your own programs, choose the new *.bit file created by you to initiate the device]*. If all goes well you should wind up with a page such as shown in Figure 4.

Click on the boundary scan tab at the bottom of the window. Following instructions, right-click on the device and select "Program" and then press "OK" in the

popup window, if there are no problems success will be signaled by a “Program Succeeded” announcement. At this time the 7-segment display should be cycling through the numbers 0 to 9, the slide switches should control their corresponding LED’s and depressing the push buttons should interrupt the display on the 7-segment chip opposite the button. Also, the push button on the D2XL card, when depressed, should light up the LED “LD1”. **If the “Program Succeeded” message came up and the board did not respond properly, unplug the board for 15 seconds, plug it back in and select program” again.**

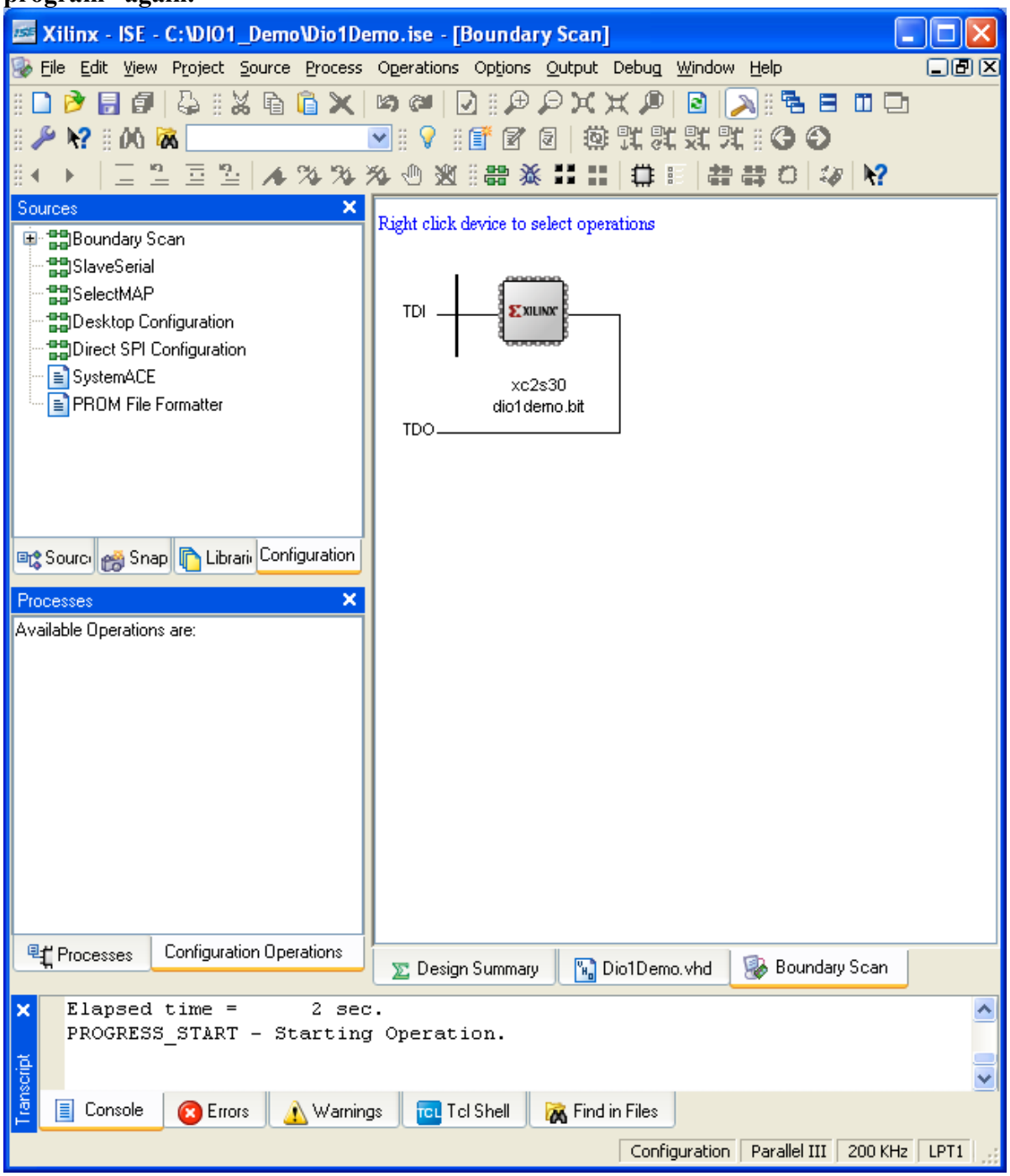


Figure 4: Programming page

If this is so, then the D2XL board and the DI/O1 board are in proper working order and we can go on to our first project.

Important Xilinx design notes:

1. ***Create your own directory in the D drive, D:\. Anything put into the root directory C:/ will be wiped out as soon as you log off. All of your work must be saved in your new D-drive directory or in its subdirectories. As a back-up, it is a good idea to copy your directory contents onto a flash drive before logging off.***
2. ***Your directory and subdirectories CANNOT contain any spaces.***
3. ***DO NOT name any of your projects or design elements any reserved logical names such as, AND, OR, COUNTER, BUS, OUTPUT, etc …***
4. ***If you ignore any of these three notes, your project will not work. You will have to start the project over from the beginning. Depending on the project this could cost you a considerable amount of time. In addition, if one of your projects fails for one of these three reasons points will be deducted from your lab score.***
5. ***If implementing your design to your D2XL and D101 boards does not work, try restarting the board by disconnecting it from the power supply.***

Design Project I

For our first project we will use "Schematic Entry" to design a circuit with a single AND gate. We will use "ISE Simulator" to test its proper operation, attach push buttons to the two inputs and an LED to the output, download it into the FPGA and test its operation.

Schematic Entry

Go to the "Help" window, select "Help Topics" and then read the chapter FPGA Design/Design Entry/Schematics. **Close the current project and do not save the configuration file.** Select "File" "New Project" to get to the Project Wizard. Choose a Project Name, Project Location on the D drive and use "Schematic" for Top-Level Source Type. Check that the appropriate hardware is selected (see before) and that "ISE Simulator" is chosen. Do not create a new source nor add an old one at this time but continue until the process finishes and displays the Design Environment again. Select the "Synthesis/Implementation" tab; click on your device name and in the "Processes" window double click "Create New Source". From the choices given select "Schematic" and give it a name of your choosing When the Wizard finishes, a schematic entry window is created representing your schematic file with an extension .sh. Open that window. Click on the "Symbols" tab in the "Sources" window; choose "Logic" from the categories in the "Sources" window. (If your "Sources" window is too small and you cannot get the symbols or categories scrollbars to work, increase the length of the "Sources" window.) Select "and2" under "Symbols" and place it (by left-clicking the mouse) in the schematic entry window. You may wish to magnify the drawing for better viewing. Next add input and output buffers to the two inputs and one output using the "buf" symbol. (The buffers are needed to interface the logic devices to the input/output components on the board; they can be found in the "Buffer" category.) You will need the

wiring tool to make the connections. Each input needs an input marker and the output needs an output marker, which can be generated by clicking on the I/O Marker icon. Edit the default names to change them to IN_1, IN_2 and OUT_1 respectively. Save your design and go to "Tools" "Check Schematic" to verify your design. If there are no errors, you can go on to the next step. Your design should look like Figure 5. **Print out the window with your schematic.**

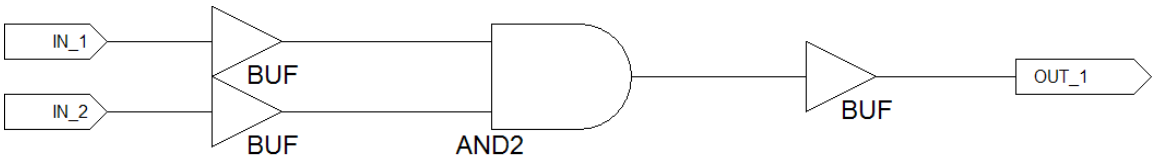


Figure 5: AND Logic Design

Modeling the Design

We must now simulate the design to see whether it satisfies our design goals. Select the "Behavioral Simulation" tab under "Sources", then click on the sch file, and double click on "Create New Source" under the "Processes" window. In the popup, select "Test-bench Waveform" and assign an appropriate name. In the "Timing Window" of Figure 6 select "Combinatorial" for the clocks and leave everything else as it is. The next window, shown in Figure 7, gives you the opportunity of setting a train of input pulses to test your logic design. Note that the two inputs should be chosen to exhaust all possible ways of feeding the circuit. Save your file and go back to the "Sources" window with the "Behavioral Simulation" tab depressed.

Under "Processes" window, select "Xilinx ISESimulator", and then double click on "Simulate Behavioral Model" and you should be rewarded with Figure 7. Check it over to assure yourself that the simulation outcome indeed represents what you expect from your circuit. **Print out the window with results of your simulation.**

Assigning Pins

Next we must assign pins to our device. If you are interested, you may check the manual of DI01 provided by the manufacturer, where the eight LED drive signals are set to be active high (p. 2). We will attach the output of our AND to the drive signal of LED 1. We will also want to tie our inputs to the pushbuttons 1 and 2, which, when physically pressed, connect their outputs to Vdd or logic high (p.4). We check the DI01/D2XL pin correspondence chart and note that LD1 is connected to FPGA pin 93 and that BTN1 and BTN2 are connected to pins 84 and 85.

To implement the pin assignments, select the .sch file in the "Sources" window with tab set to "Synthesis/Implementation" and then double click on "Assign Package Pins" in the "Processes" window under "User Constraints". Answer "Yes" that you do want to create an .ucf file. In the .ucf file enter p84, p85 and p93 in the "Loc" column for I/O components "In_1", "In_2" and "Out_1" respectively and then save the file choosing "Synplify Verilog Default: []" as the bus delimiter.

Implementing the Design

Select the .sch file in the "Sources" window. Then under the "Processes" tab go to "Generate programming" and double click "Configure Device [iMPACT]" to generate the appropriate files and download them to the FPGA. Follow the same steps you did in the initial tests of the boards. Ignore any warnings about clocks. You should once again see "Programming Succeeded" as the indication that no errors were found in your design and that the download was successfully accomplished.

Testing the Design

Manually verify that BTN1 and BTN2 are inputs to an AND circuit whose result is displayed in LD1.

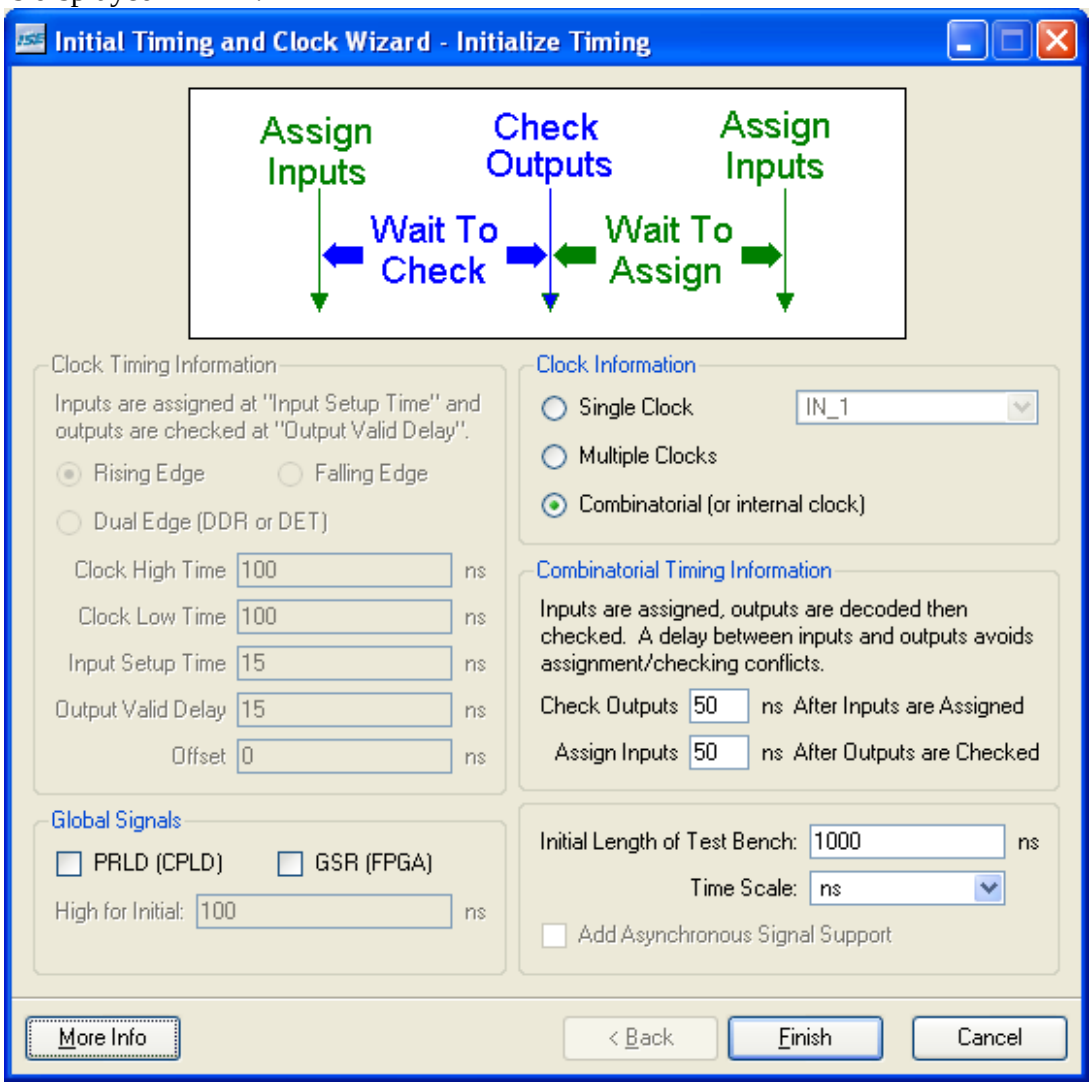


Figure 6: Timing choices for ISE simulator

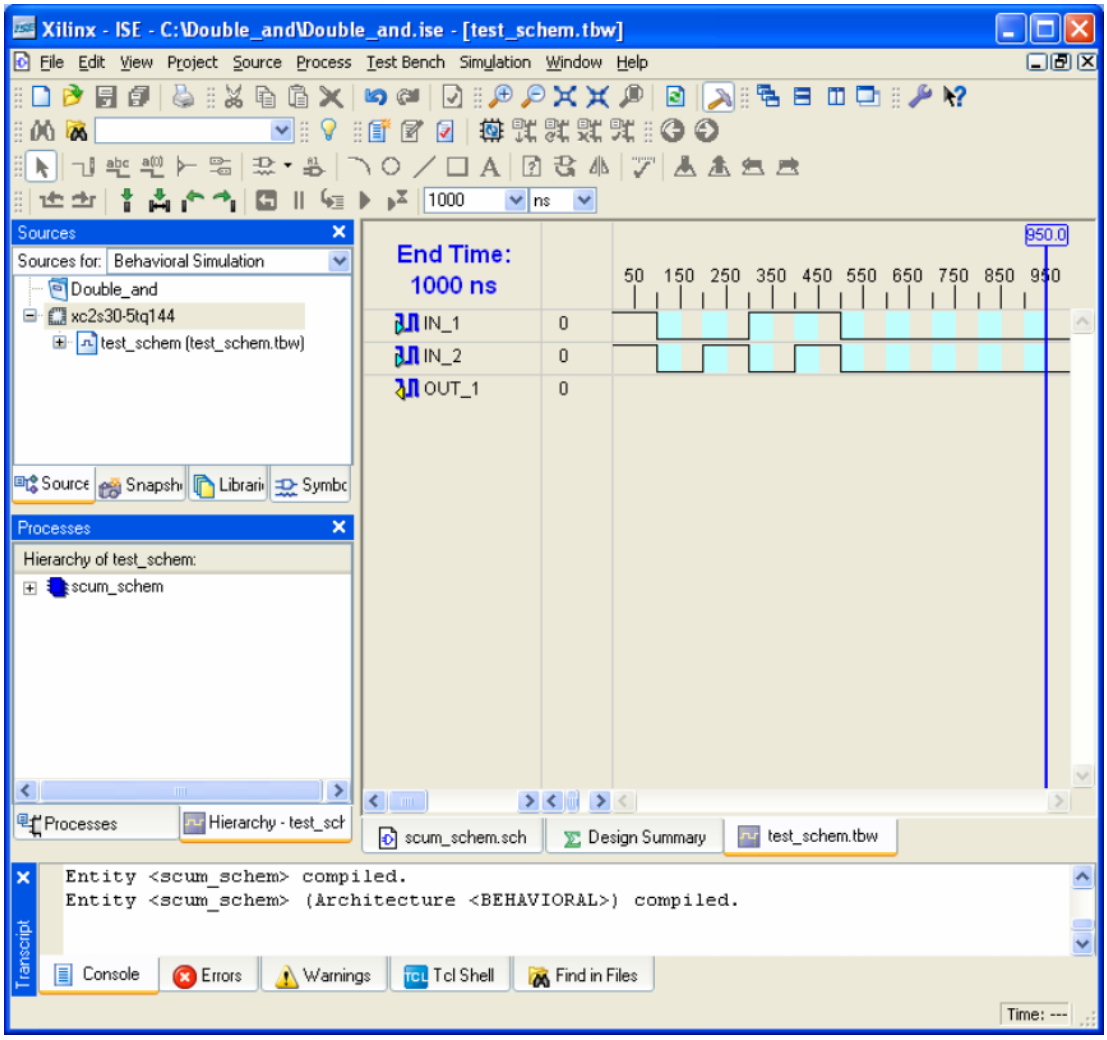


Figure 7: Simulation of logic train.

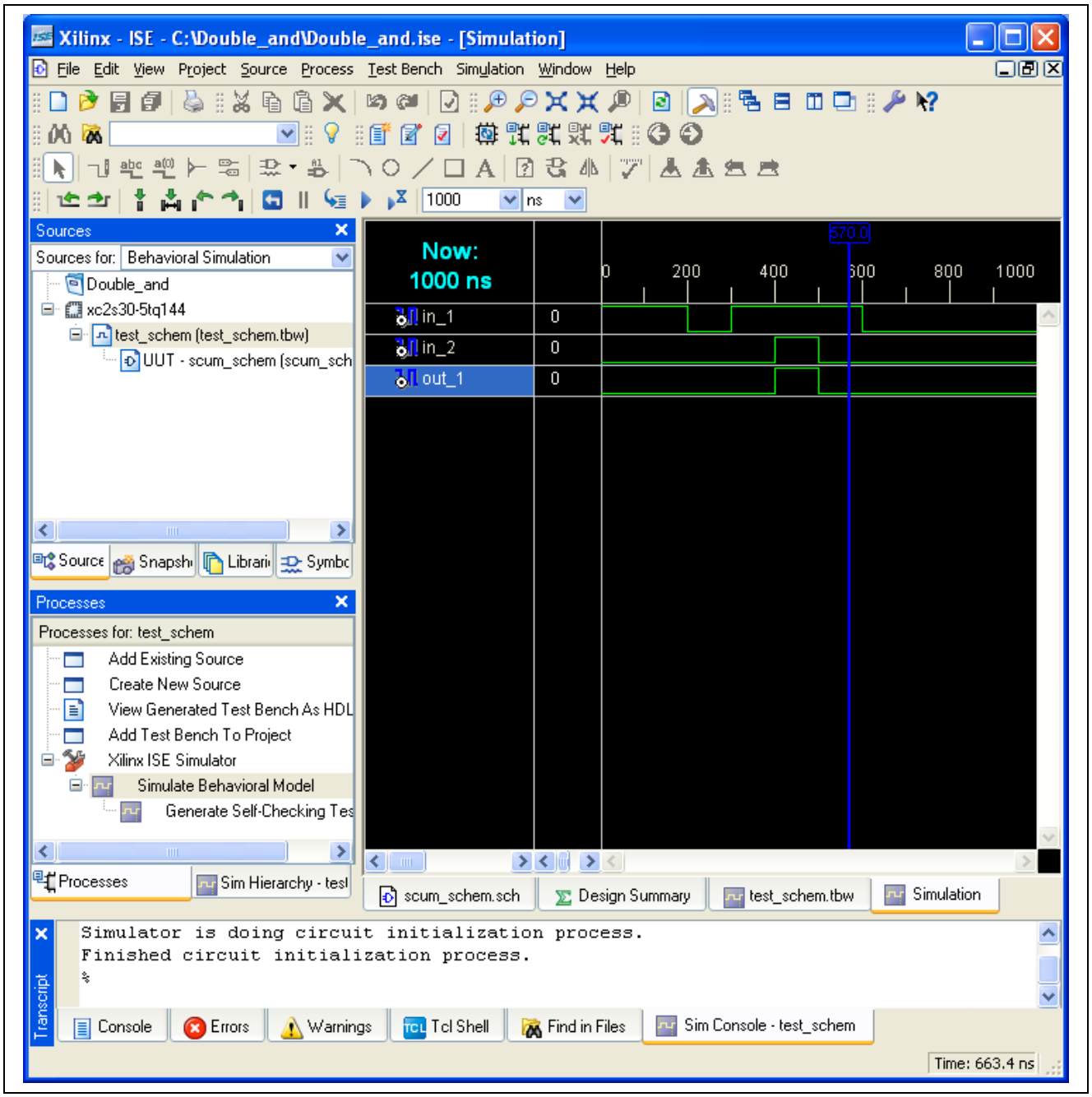


Figure 7: The Test-Bench with input and output logic levels

Design Project II

Following the same procedure above, implement, simulate, download and test the circuit in Figure 8 which creates a NOR gate from 4 NAND's. **Be sure to "Close Project" before start working on project II.** Note that wires can be connected to each other, but only one wire can be connected to a pin**. Finish the circuit by adding input and output (I/O) buffers** and giving appropriate labels to the I/O lines. You can once again use BTN1 and BTN2 for inputs and LD1 for the output. Verify that the truth table for a two-input NOR gate is satisfied. **Print out the windows with your schematic and results of your simulation.**

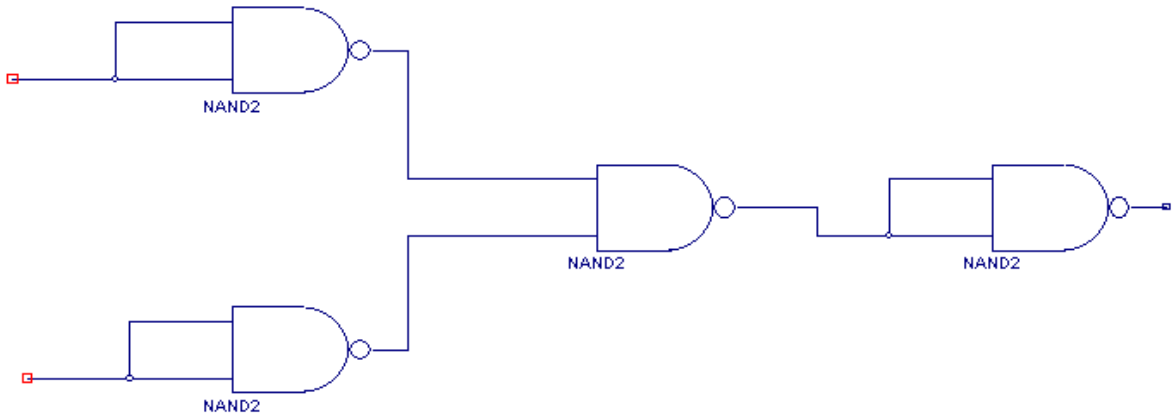


Figure 8: A two-input NOR circuit constructed from NAND's

Design Project III

One of the huge advantages of this form of design is that macros of arbitrary complexity can be constructed, stored and reused in future designs. Our collection of symbols contains many such macros. For our next design we will use one of them, CB4RE, a 4 Bit Cascadable Binary Counter with Clock Enable and Synchronous Clear, to construct a counter that will count from 0 to 5, then reset and continue counting. We will then save it as a new macro to be used in our next experiment. The design in Figure 9 is an implementation of such a counter.

Below, you will follow the same procedure above, implement, simulate, download and test the circuit in Figure 10. Work out a scenario for testing the counter with now three inputs, the clock (clk), counter enable (cen), and clear (clr). Provide a sufficient number of clock cycles through the "clk" to show the full, repetitive operation of the counter. The "cen" needs to be logic high for the counter to start counting. If the "cen" is logic low, it pauses the counting. The "clr" clears the counter, resetting it to zero.

What is meant by “synchronous clear”? You may wish to consult the “Symbol Info” tab for the CB4RE counter.

A new wrinkle in this design is the presence of a bus which is a collection of individual signals. It is automatically created when a wire, drawn with the wiring tool, is labeled as a vector, e.g. btout(3:0). Individual elements of the bus are selected via bus

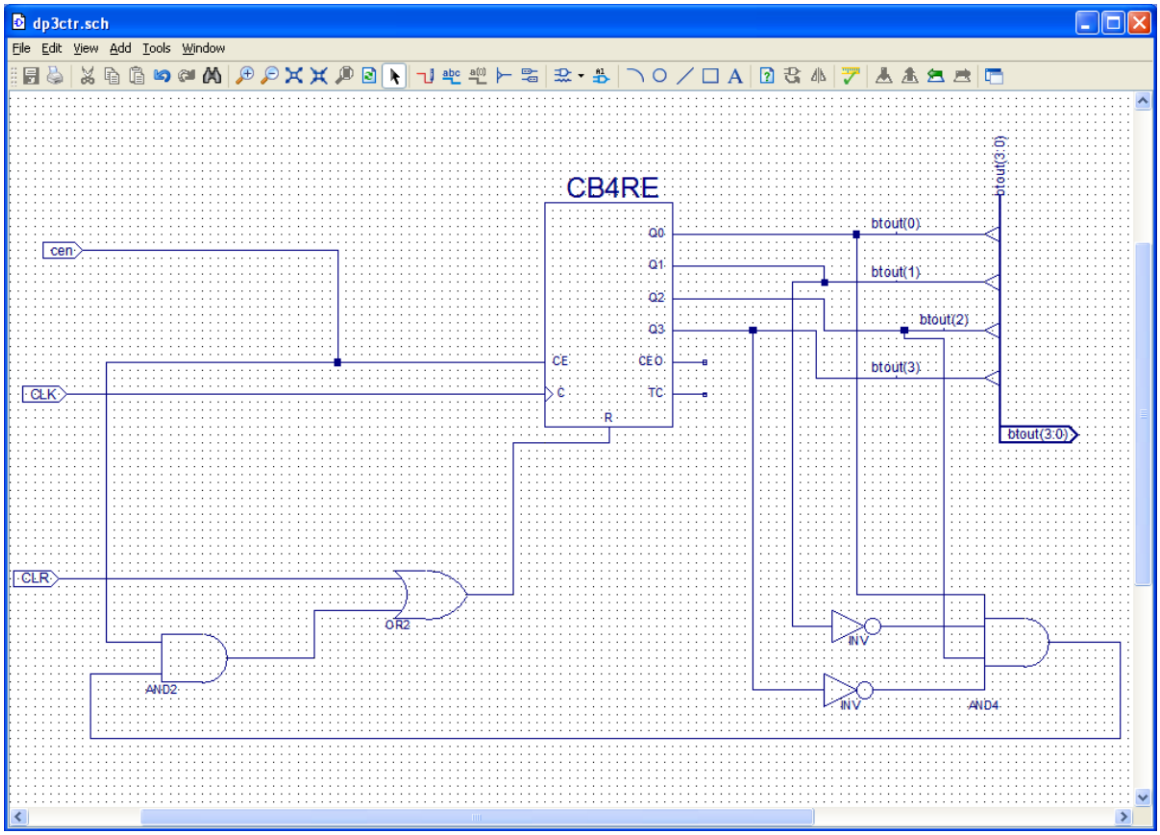


Figure 9: A 0 to 5 counter

taps with their names specified by using the netname tool. In our case these names are btout(0) to btout(3). You may need to use the “Rotate” command located in the “Edit” menu to orient the bus tabs properly.

Create the schematic entry in Figure 9; check for errors and save the file. The device that we wish to build will have 3 inputs, marked “ce” for Clock Enable, “clr” for Clear and “clk” for Clock. Outputs will consist of a bus 4 bits wide, btout(3:0). Start a new project with schematic entry at the highest level. Enter your design in the now usual fashion naming the inputs ce, clk and clr. To generate a bus for output draw a line using the wiring tool and give it a name btout(3:0). The program will change the line into a thicker bus-line in accordance with the name. Attach the bus taps to the bus. For that, click on the “Add Bus Tab” icon, possibly correct your tap’s orientation with the “Rotate” icon, and then click on the tap attached to the cursor when close to the bus line. Depending on program settings, the tap may be named automatically, in the order they

are added from btout(0) to btout(3). Otherwise, you will need to name the bus tap within its "Object Properties". There should be two names there. Vector for the bus as a whole, and scalar name for the tap. You only need to change the scalar name. That will also become the name for any attached wire. Another way to add the bus tap is to click, after the "Add Bus Trap" icon, onto the pins of the component that needs to be connected to the bus, as in adding I/O markers. However, there is even less control using tap addition than in the procedure above. In any case, you need to end up with the named connections to the bus such as btout(0) to btout(3). Names can also be changed using the "net name" under the "Add" menu. Be sure to click directly on the wire for the chosen name to apply. You can use the "increase the name" feature to ease the amount of typing. An output marker needs to be attached to the bus.

Be sure to save the design and to check for errors. Most often the errors are in poorly connected wires such as to the bus traps. Within the schematic, individual wires can be clicked to highlight the connections and find the names assigned to the connections that may be mentioned in error messages. If no errors are found, go back to "Project" and create a new source of type "Testbench Waveform" using a single clock. Provide changes in the logic states of "ce" and "clr" to fully test your design. Save this file and go to "Simulate Behavioral Model". Verify that your logic is working properly by examining the outputs on the "Wave" plot below (Figure 10). You may have to extend the simulation time to see a full clock cycle. In the toolbar, make sure the simulation ran for at least 10000ns. If not, change it on the drop down menu in the toolbar and hit the "run for specified time" under the "Simulation" menu. Note that the "btout" bus is given numerically after each clock transition and that it can be expanded to look at the individual bit states as well. Does the output agree with your expected performance? **Print the window with sample results of your simulation.**

Next select your .sch file in "Sources". **Print the window with the schematic.** Double click "Create Schematic Symbol" in "Processes" under "Design Utilities". In your schematic entry page, verify that the new symbol is available for use. It should carry the name of your schematic file.

Design Project IV

Today's lab is the introductory part of a two lab series that will result in the construction of a counter to count from 0 to 60 with the counts displayed on our 7-segment displays. If you have arrived this far with time remaining you may wish to continue with next week's lab and construct the 0 to 60 counter that is at the heart of our stopwatch design.

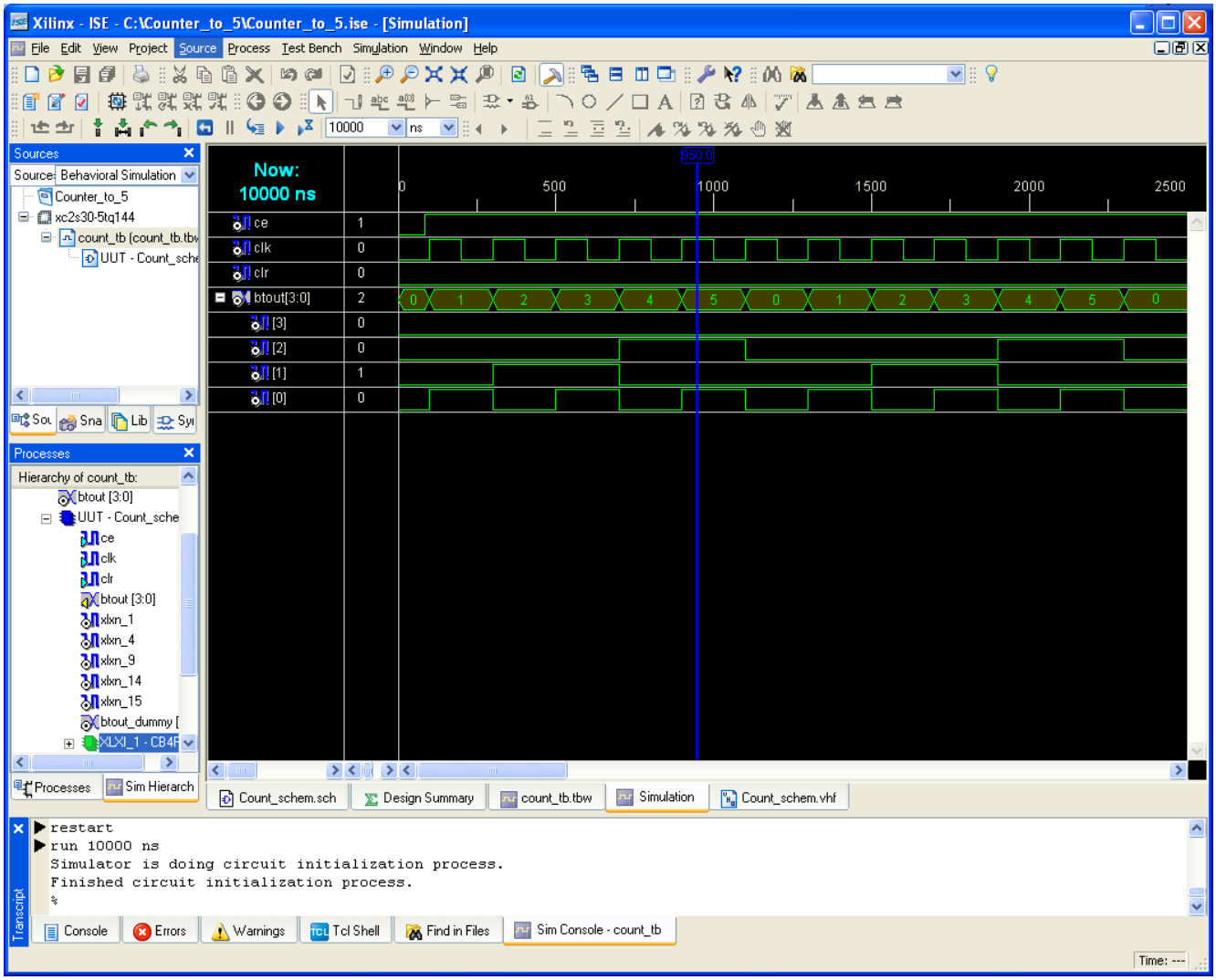


Figure 10: Testbench Waveforms for 0 to 5 Counter