

ECE561 Project 1

Introduction to *Xilinx v.10* for Combinational Design

Instructor Prof. Joanne E. Degroat

Introduction

This project assignment is intended to familiarize you with the essential elements of the *Xilinx* design environment. You may use departmental laboratory facilities or your own computer to complete this project (the *Xilinx* software is distributed with the ECE561 textbook and the latest version is available free in a WebPACK from *Xilinx*).

Your instructor spent several hours designing a seven-segment decoder. However, he was very, very lazy with the design. Your job, as a fledgling designer, is to optimize the design and simulate it within the *Xilinx ISE* environment. By the way, it is pronounced "zylinks".

Import and Open a Project

If you are working in the ECE PC laboratory, log in to a system and create an ECE561 sub-directory of your home directory (Z:). If you're working on your own machine, you can choose a special place for your *Xilinx* project files or just use the default. Please note that *Xilinx* cannot read blank space in directory names such as "My Documents" and do not save *Xilinx* project files on Desktop.

Retrieve ***561proj1.zip*** from the ECE561 web site and store it in the ECE561 sub-directory on your PC (the one you have just created). Unzip ***561proj1.zip*** using WinZip. The extracted contents include *561proj1* sub-directory (folder). Open sub-directory *561proj1*. Among the files that have been extracted, you will find a file named ***proj561.ise***.

Start up the *Xilinx* project by double-clicking on the file ***proj561.ise***. Or you can start *Xilinx ISE* by double-clicking the ISE Project Navigator icon on your desktop or select Start → Programs → Xilinx ISE Design Suite 10.1 → Project Navigator. The Project Navigator window opens up. Select File → Open Project. Now browse down to ***proj561.ise*** and open it. This will open the



Xilinx ISE 10.1

project ***proj561*** in the Project Navigator window.

Project Navigator is divided into four main sub-windows. The top left is the Sources window, which hierarchically displays the elements included in the project. Beneath the Sources window is the Processes window, which displays available processes. The third window at the bottom of Project Navigator window is the Message Console, which shows status, error, and warning messages. It is updated during all project actions. On the right, the fourth window is the Editor. From this window you can edit source files and view various reports that are generated by the processes.

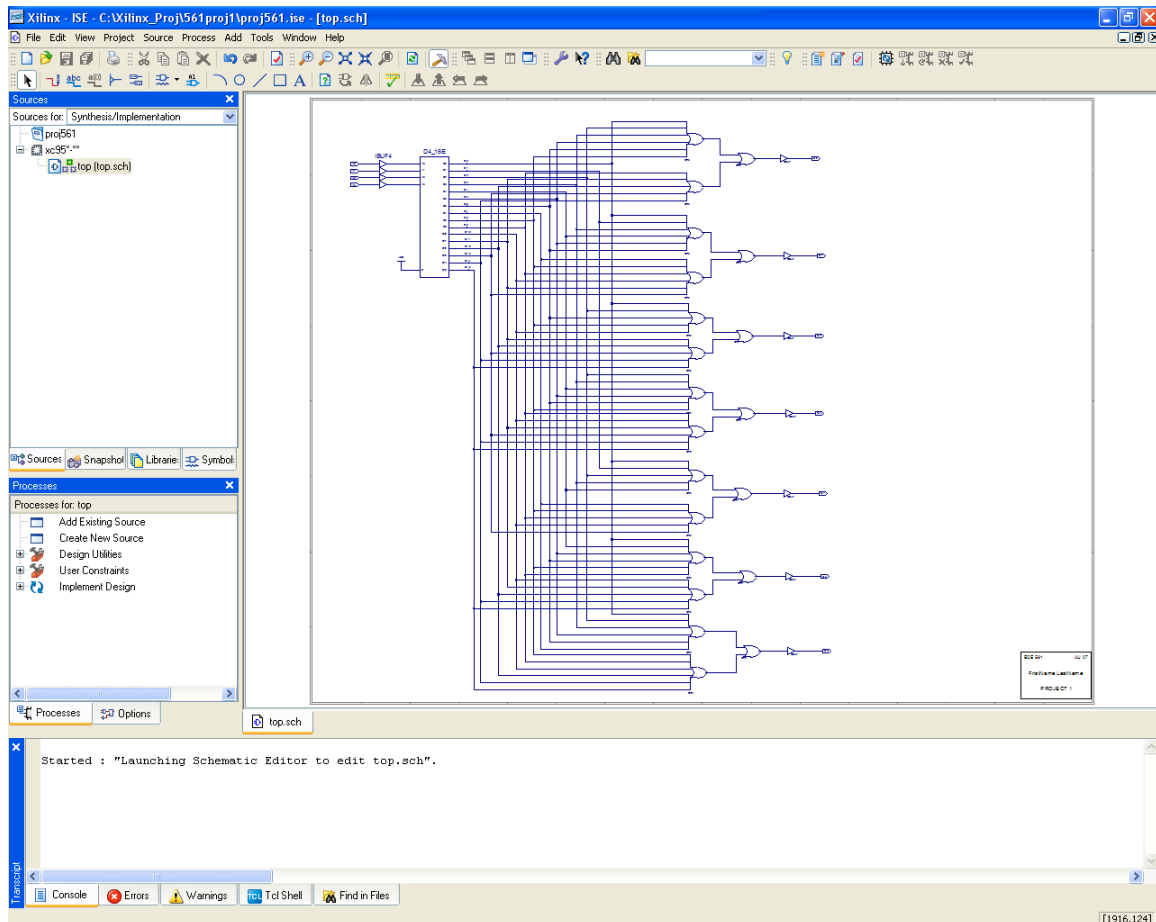


Figure 1: Project Navigator window

Invoke the Schematic Editor

To invoke the Schematic Editor, select *Sources for Implementation* in the Sources window. Double-click on the source named *top (top.sch)* (shown highlighted in the above picture). This will load the seven-segment design shown in Figure 1. There are toolbars across the top of the window. The buttons in the toolbars have popup balloon help strings that appear if you leave the mouse pointer atop them for a few seconds. Explore.

This design consumes a lot of space so the first thing you may want to do is zoom out. Click the *Zoom Out* button from the toolbar. The schematic view will expand to show the entire design sheet. You can use the *Zoom In* button to get a detailed view of the schematic and use the scroll bars to view a particular portion of the schematic. Try it out!

This circuit contains three main pieces of combinational logic, all of which should be familiar from ECE261.

- There is a 4-to-16 decoder (**D4_16E**, a *Xilinx* library part). Four bits **A0** — **A3** go in — they specify which of the output lines **D0** — **D15** to assert. Exactly one of those lines is asserted at any time EXCEPT when the enable line **E** is not asserted. If **E** is not asserted, none of the outputs is asserted. Your instructor tied **E** to **Vcc** (high, logic 1), thus asserting it permanently, to avoid this silly situation. Note: all of the signals in the **D4_16E** part are active high, so "asserted" means "1" in this case. You should remember from ECE261 that the decoder outputs **D_j** are all **AND** functions of inverted and non-inverted versions of the **A_i**.
- There are several **OR** gates used to build seven output signals **S0** — **S7** from the **D_j** outputs of the decoder. Hence, each of these signals is a sum-of-products of the **A_i**.
- There are buffers attached to the circuit inputs and outputs. These are necessary for a physical realization of the circuit to be constructed. They implement the "identity" function — what goes in comes out unchanged.

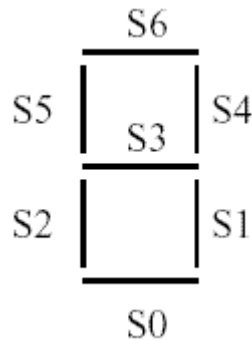


Figure 2: 7 Segment Display

If (for example), 0000 appears on the inputs **A0** — **A3** and **E** is asserted, 1 will appear on **D0** and 0 will appear on **D1** — **D15**. If you follow the wires on the schematic (note: wire bonds are indicated by circles — no circle means no connection. You have to zoom in to see this), you'll see that **D0** is fed to six of the seven **OR** functions, and that **S0** — **S2** and **S4** — **S6** will be asserted as a result. **S3** will not be asserted.

If you don't understand this, refresh your memory on decoders and sums of their outputs. Different values applied to $A_0 — A_3$ will cause different patterns to appear on the $S_0 — S_6$. The figure shows a seven-segment display with the signals S_k corresponding to each segment labeled. If $S_k = 1$ for a particular k , the corresponding segment will be illuminated; if not, the segment will be dark.

Problem 1: For each of the sixteen possible input combinations of the A_i , draw a picture showing what will appear on the seven-segment display.

Functional Simulation

Xilinx ISE is optimized for a Hardware Description Language (HDL) approach to design, and the simulation of circuits is carried out using testbenches. But for simulation we don't require the knowledge of any HDL as *Xilinx ISE* has a GUI based testbench generator called HDL Bencher. After setting the input stimulus in HDL Bencher, we can do a functional simulation of the circuit using ModelSim, simulation software from *ModelTech*.

In your Project Navigator window, click on your schematic file *top (top.sch)* to make it active. Now select Project → New Source. In the window that opens up select the option *Test Bench Waveform*. Specify a name for the waveform in the *File Name* field and click on *Next*. In the following window click on *Next* and then finally click *Finish*.

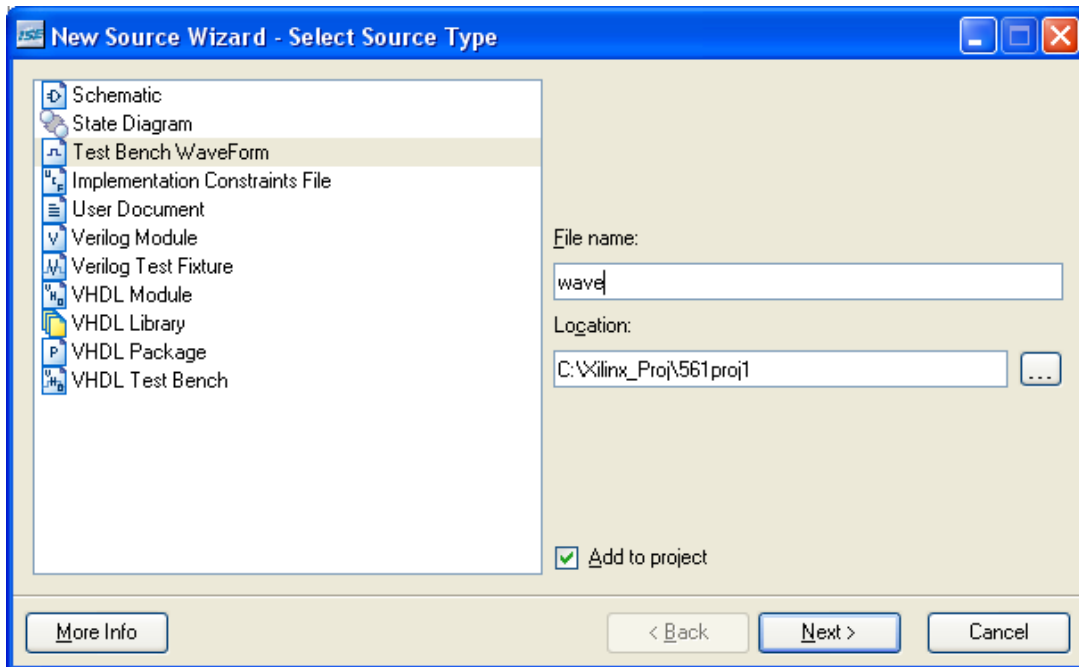


Figure 3: New Source window

This will open the HDL Bencher. In the Initialize Timing window, select the option *Combinatorial (or internal clock)* design. In the input boxes after *Check outputs* and *Assign Inputs*, enter the value 1, set *Initial Length of Test Bench* to 32 ns, set the time scale to *ns*, and deselect *GSR(FPGA)* as shown below and click *OK*.

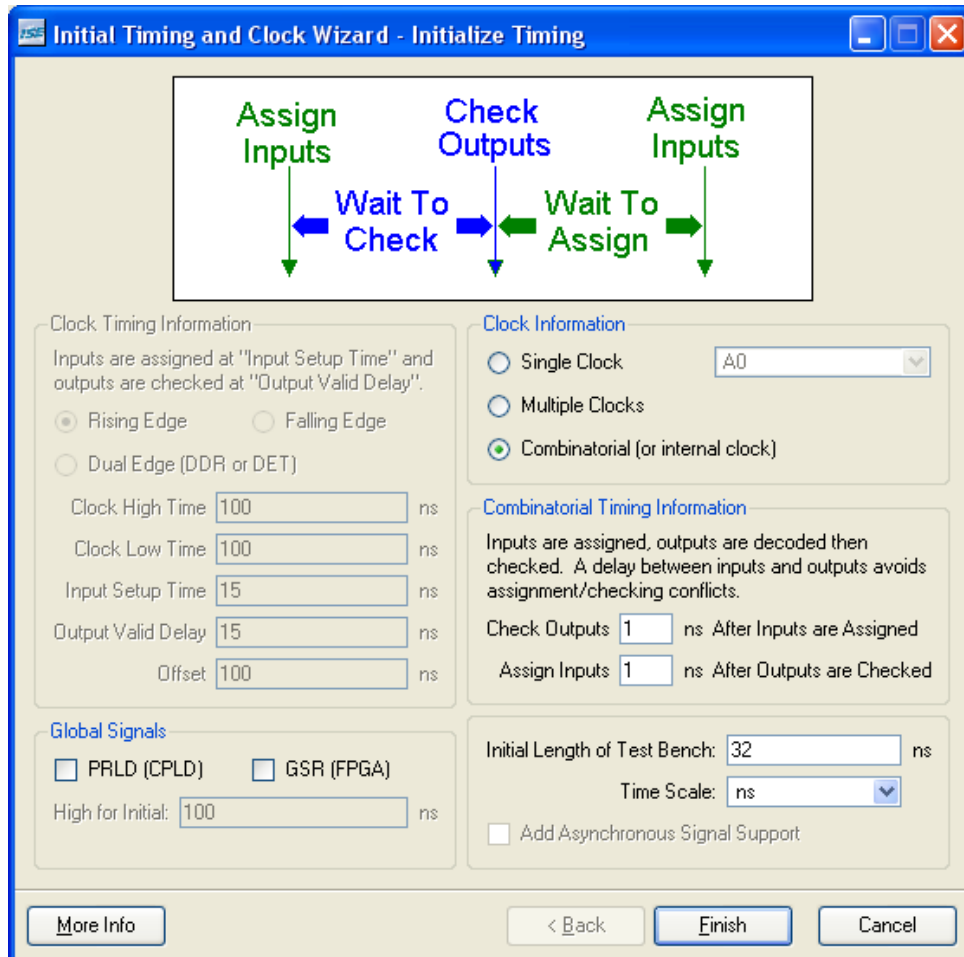


Figure 4: Initialize Timing window

Assigning Input Values

We have four primary inputs to our circuit **A0**, **A1**, **A2**, **A3**. Now to functionally simulate the circuit, we should simulate all the 16 possible combinations of the inputs. To do this, we are going to assign a square waveform (toggles between 1 and 0 in a regular pattern) to **A0** and then assign similar waveforms to **A1**, **A2**, **A3** but the time period of successive waveforms will be twice that of the former. So the time period of **A1** is twice that of **A0**. The time period of **A2** is four times that of **A0** and the time period of **A3** is eight times that of **A0**. Now if we simulate the circuit for 8 time periods of **A0**, we will get all the 16 possible combinations.

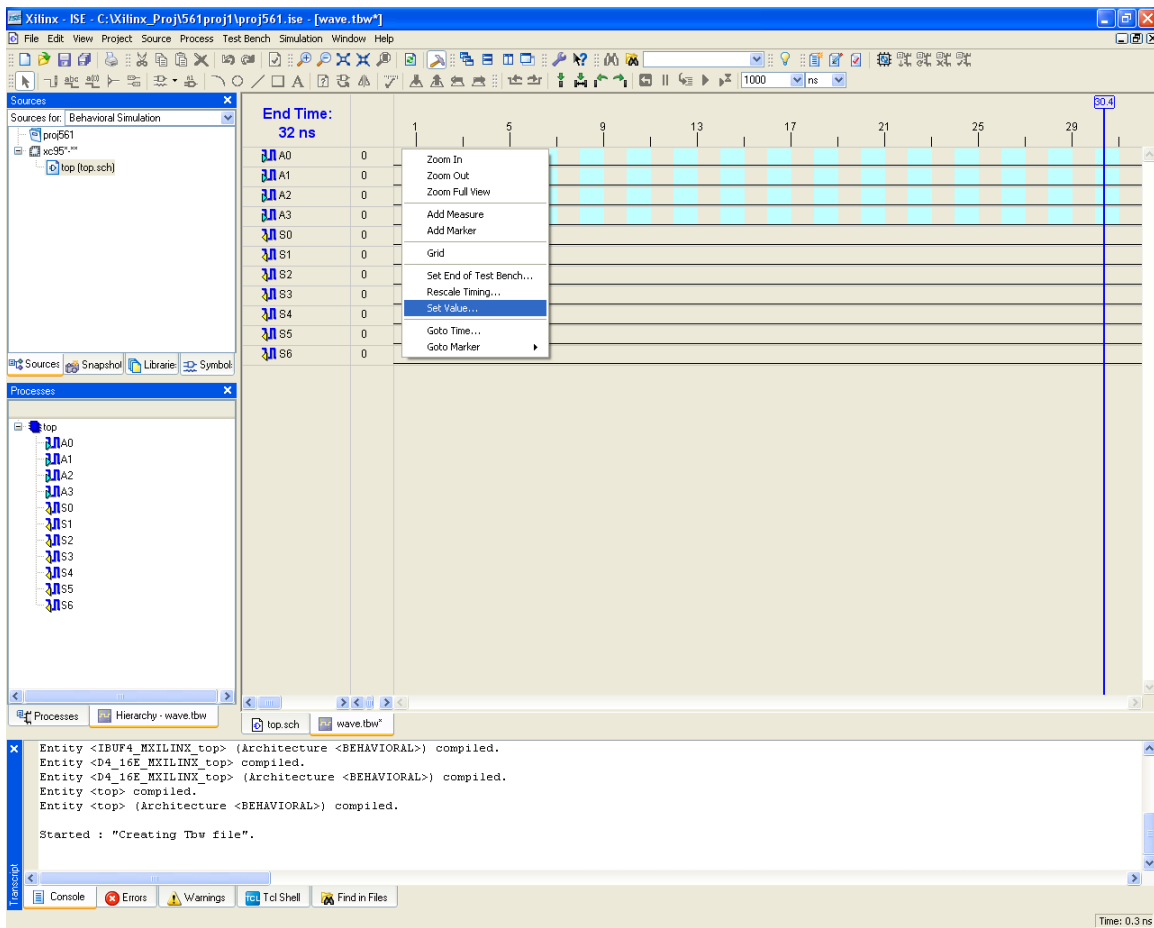


Figure 5: HDL Benchmer window

When you click on any of the blue areas in the row of inputs, it will toggle the values from 0 to 1 and vice versa. To assign a square waveform, right-click on the beginning part (0-1 ns) of the waveform for the input A0. Select *Set Value* from the drop-down menu.

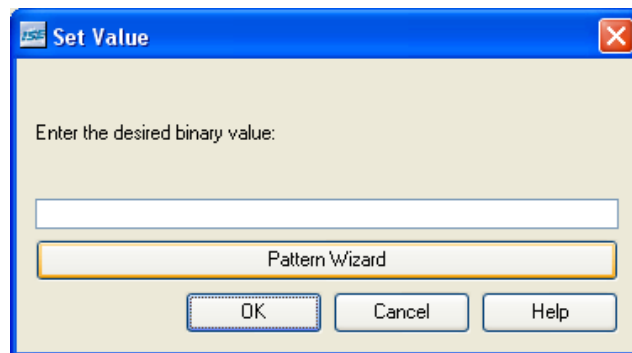


Figure 6: Set Value window

Now click on the tab named *Pattern Wizard* in the Set Value window. This will open up the Pattern Wizard window. Set the pattern as *Toggle*, *Number of Cycles* as 16, *Initial Value* as 0 and *Other Value* as 1 and *Toggle Every* as 1 as shown below.

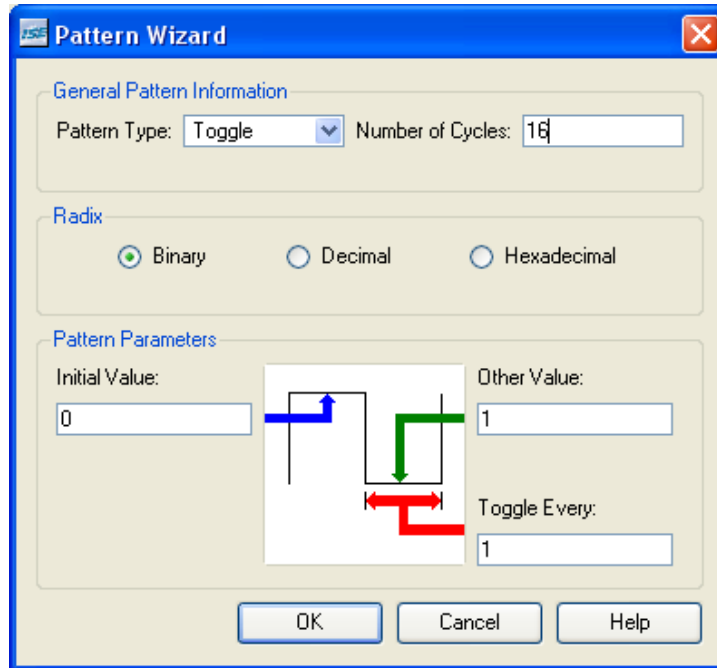



Figure 7: Pattern Wizard

Repeat this process for **A1**, **A2**, and **A3**. Only two values have to be changed – the *Number of Cycles* field and the *Toggle Every* field. For **A1**, set the *Number of Cycles* value as 8 and *Toggle Every* value as 2. For **A2**, set the *Number of Cycles* value as 4 and *Toggle Every* value as 4. For **A3**, set the *Number of Cycles* value as 2 and *Toggle Every* value as 8. Figure 8 shows the input waveform pattern. In the Project Navigator window, click on the *Save Waveform* icon  .

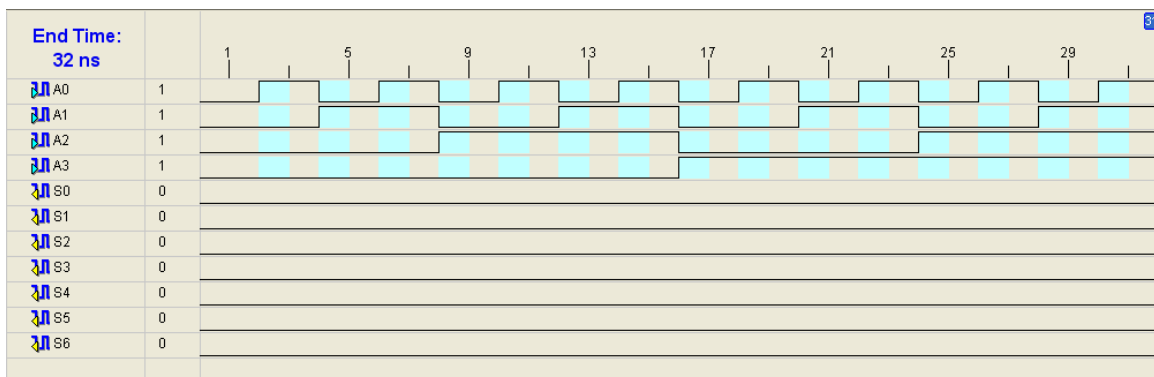


Figure 8: Simulation Waveform

Now select *Sources for Behavioral Simulation*, and click on the waveform file in the Sources sub-window (Sources Tab). In the Processes sub-window (Process tab), click on (+) in front of *ModelSim Simulator*. Then right click on *Simulate Behavioral Model* and select *Properties*. Then change the *Simulation Run Time* to *32ns*. To simulate the circuit, double-click on the option *Simulate Behavioral VHDL Model*. This will run ModelSim and open up the waveform with the outputs. Print out the waveform output (Zoom Full) by using the print command in the waveform window or Print Screen on the keyboard.

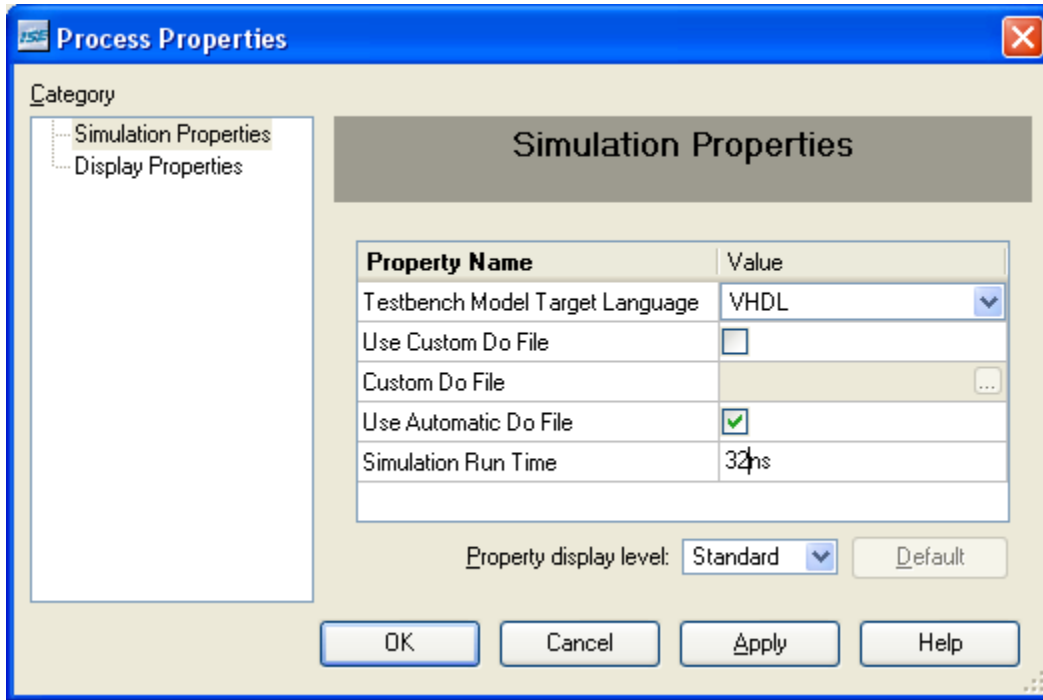


Figure 9: Process Properties window

Problem 2: Based on your answer to Problem 1 and the simulation results, is A0 the least significant bit of the binary number being displayed by the seven-segment display? Explain why.

Optimize 1: Minimization

The decoder-OR network in this design implements seven straightforward sum-of-products for the segment signals. It occurs to you that you could compare this design (which is functional, but brute force and perhaps not ‘elegant’) with an equivalent combinational design obtained from minimization with Karnaugh maps.

Problem 3: From the answer to Problem 1, obtain seven four-variable k-maps for the S_k signals. Using ECE261 techniques, minimize each function independently and give the minimized function in sum-of-products form. Show your work (draw contours in the k-map, etc.).

Problem 4: Repeat Problem 3, except treat the inputs 1010 through 1111 (binary numbers 10 through 15) as don't care entries in the k-maps.

Optimize 2: Implementing your own design

Create a new Project

Invoke the Project Navigator, and create a new project by selecting **File** → **New Project** with the name *XXXproj1*, where *XXX* is replaced with your initials. In the New Project window, specify *Schematic* as the *Top-Level Module Type*. After clicking *Next*, specify *XC9500 CPLDs* as the *Device Family* and *XST (VHDL/Verilog)* as your *Synthesis Tool* from the pull-down menus, as shown in the figures.

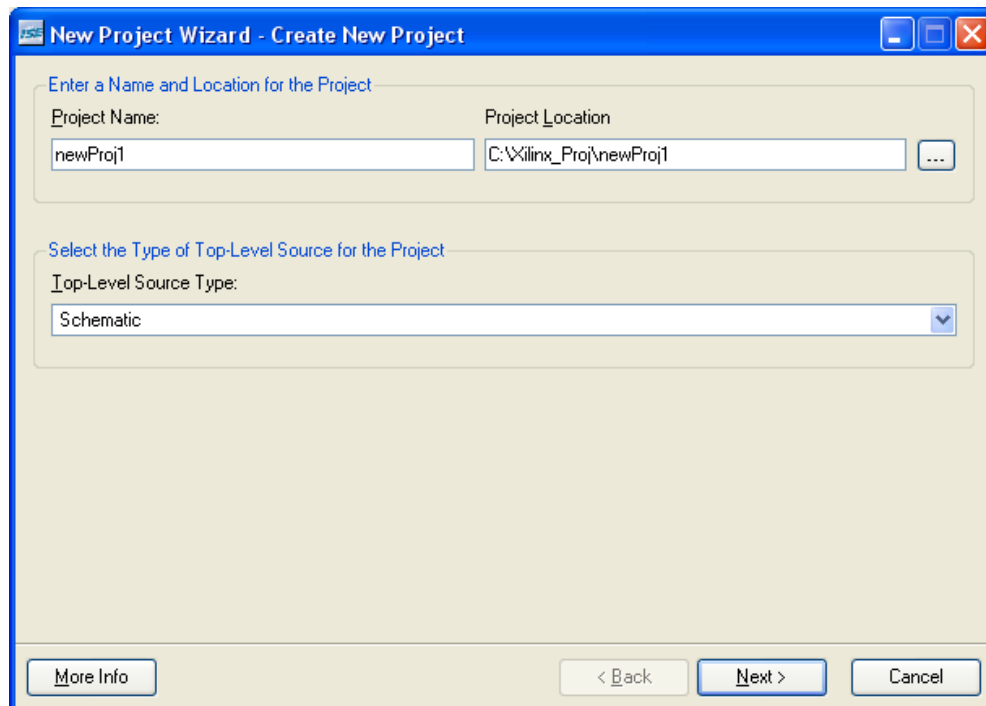


Figure 10: New Project window

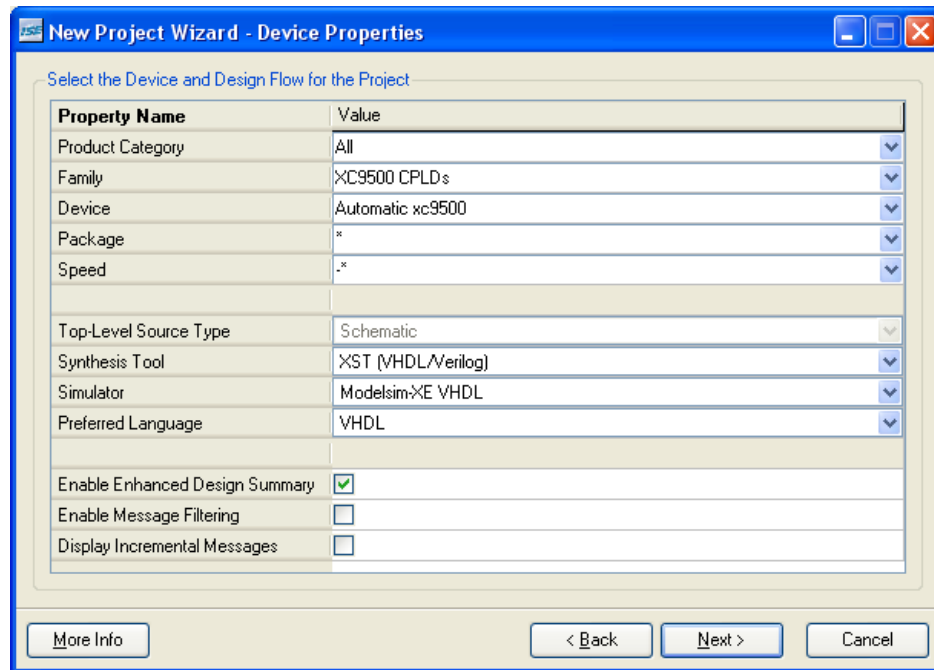


Figure 11: New Project window (continued)

Continue clicking *Next* and then *Finish*. The Project Navigator will create a new project and you will see two entries in the Sources window – *XXXproj1* and *Auto xc9500-.**.

Now click on Project → New Source. In the window that opens up, select the entry *Schematic* and give any name you like in the *File Name* field as shown in the figure.

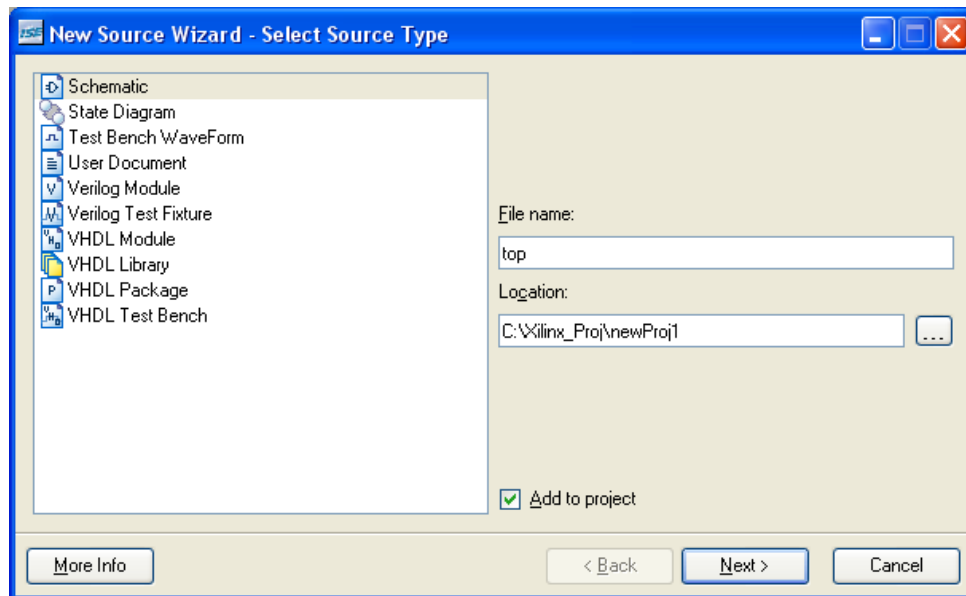


Figure 12: New Source window

Click *Next* and in the next window click *Finish*. Once you do that a blank schematic will open up in the Xilinx ECS schematic editor.

Inputs and Outputs

The default schematic sheet size is C (22 x 17 in). If at any point you think you need more space for your schematic, you can change the sheet size. This can be done by clicking *Edit* → *Object Properties*. This opens the Schematic Properties window. Highlight the entry *Sheets* under *Category* and click on the entry **C =22*17**. This will open up the drag down menu for different sizes. Choose a bigger size **D** or **E** and click *OK*.

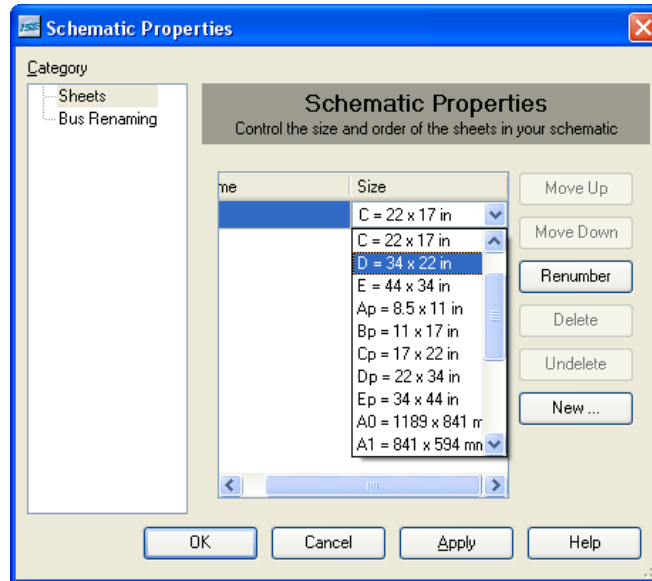


Figure 13: Schematic Properties window

Start by adding eleven I/O Markers to the design. Click on *Tools* → *Create IOMarkers*. A window will pop up, requesting you to specify the input and output port names. Specify inputs as A0, A1, A2, A3 and outputs as S0, S1, S2, S3, S4, S5, S6. Separate the IOMarker names by commas as shown below.

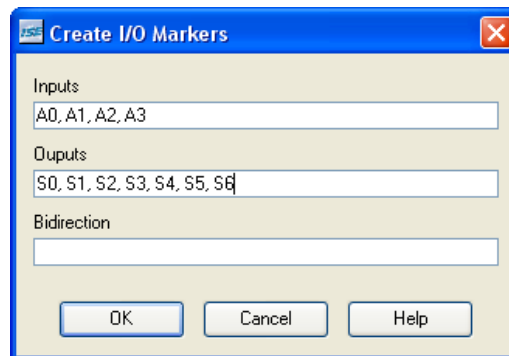



Figure 14: Create IO Markers window

This will add 4 inputs and 7 outputs with wires attached to them in your schematic. Now add input buffers corresponding to each input. In the left side of your Project Navigator window, click on the *Add Symbol* button . The Module View window changes to the Symbols window. Under the sub-window called Categories, click on the entry *IO*. Now click on the entry *ibuf* under the Symbols sub-window. This will select an input buffer. To place the buffer on the schematic, just click the mouse in the schematic sheet to place the part. Repeat this four times.

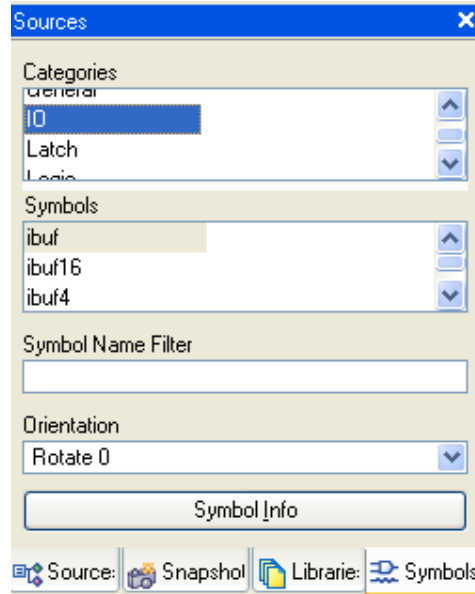


Figure 15: Symbols window

The schematic editor is now still in symbol placement mode. To get back into selection (or pointer) mode press the *ESC* key and the mouse pointer changes back. You can select symbols and move them around in this mode only. To move a symbol, click on the symbol and move it to the desired location while keeping the left mouse key pressed down.

Add the output buffers to the outputs. Click on the *obuf* entry under the Symbols sub-window and repeat the same procedure as you did for the input buffers. If you have made any mistakes you can delete any schematic item, like a component or wire by clicking on it (when in the selection mode) and pressing delete.

Place some devices

Start with your equation for S_0 . It is a sum of products. The products use inverted and non-inverted versions of the A_i , so the first thing to do is sending the A_i through inverters. Select an inverter part (*inv*) as follows. Click on the entry *Logic* under

Categories and type *inv* in the *Symbol Name Filter* for quick search, or scroll down to the *inv* entry in the *Symbols* list and click it. Then click the mouse in the schematic sheet to place the part. Repeat as necessary. A possible layout is shown below.

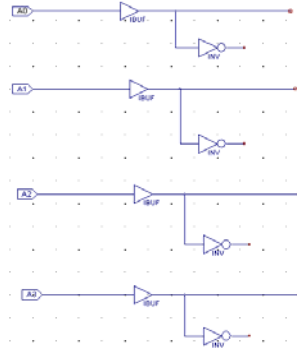



Figure 16: Possible initial layout of the Schematic

The next task is to implement some of **S0**'s product terms. Choose **AND** gates from the device selection window that you got the *inv* parts from. If you need a 6-input **AND** gate, choose *and6*, etc. The symbol names are mnemonic, so a 2-input **OR** gate will be named *or2* and so on. If the exact **AND** gate you need isn't there (not likely now, but perhaps later), build a two-level design. You can use the selection mode to reposition and delete items. To make the schematic look better you may at times want to rotate a symbol (e.g., to get an inverter oriented vertically instead of horizontally) or mirror a symbol (e.g., to make the inverter point left instead of right). You can do this by selecting the symbol and clicking on the *Rotate* or *Mirror* toolbar button.

Using this technique, lay out the gates (**AND** and **OR**) for the sum-of-products realization of the **S0** function. If you wish, you may use a two-level **NAND** design.

Wire it up

Now you will 'wire up' the circuit. Click the toolbar button  to enter the wiring mode. Begin a wire by clicking on one of the ends of a device in the sheet, and terminate the wire by double-clicking on its destination. Wire the logic using this technique. To route a single signal to multiple destinations, complete one connection, and then click on this wire to start the other connections. Try it! You can delete a wire by selecting it and pressing the delete key. You can reroute a wire (sometimes with difficulty) by clicking and dragging it around.

Add constant signals

Note: you probably won't need to do this in this project, but it is included for completeness. The remaining dangling inputs need to be connected to ground (logic 0) or high (logic 1) signals. This is done by clicking on the *General* item in *Categories* in your Schematic Editor window. To place a High (logic 1) click on the *vcc* entry in the *Symbols* list and place it where you need in the schematic. For ground (logic 0), click on the entry *gnd*. You can put as many of these terminals as you wish in your design, but be neat about it.

Checking Errors

Click on Tools → Check Schematic. If there are errors, they will be displayed in the *Transcript* window. If there are no errors, then the *Console* message will display *No error or warning is detected*.

Problem 5: Implement and simulate your answer to Problem 4. Each of the S_k should be implemented and the simulation should consider all input combinations. Print your design and the simulation window output. In the spirit of Problem 1, draw the seven-segment display appearance for inputs of 1010 through 1111.

Report

The report should be typed and follow the report format provided. Be sure to include:

- Answers to the questions, etc. in Problems 1-5
- A trace output from the simulation of the first circuit
- The schematic of the modified circuit
- A trace output from the simulation of the modified circuit