

Digital Logic Design ECEN 3233

Xilinx[®] Software Introduction and Tutorial

Objective:

This is an introductory exercise intended to teach you how to use some of the capabilities of the Xilinx[®] ISE 4.2i software. This software will be used every week to complete your laboratory assignments. No prior knowledge of digital logic is necessary. This document uses the design of a full adder to illustrate the use and operation of the Xilinx software.

We will be using the Xilinx[®] ISE software as a design and simulation tool throughout the semester. You will be responsible for being able to use this software on your own to greater degrees as the semester progresses. This is a very capable software package and we'll only use a small fraction of the functions that are available.

The Xilinx ISE software is included in your textbook on two CDs; Version 4.2i is in the most recent printing of the textbook and Version 2.1i is in older copies – we will use Version 4.2i. This is the same software as will be used in the lab. You can install it on your own PC, and it should be available in the CEAT PC labs (Engineering South, Engineering North, Kerr-Drummond, Cordell, and possibly other locations). If you install this software on your own PC, be sure to read the instructions that are included with the CDs. A highly recommended tutorial for installing the software can be found at <http://xup.msu.edu/>. Please view this tutorial before trying to install the software on your own PC – it provides valuable information that you may not find elsewhere.

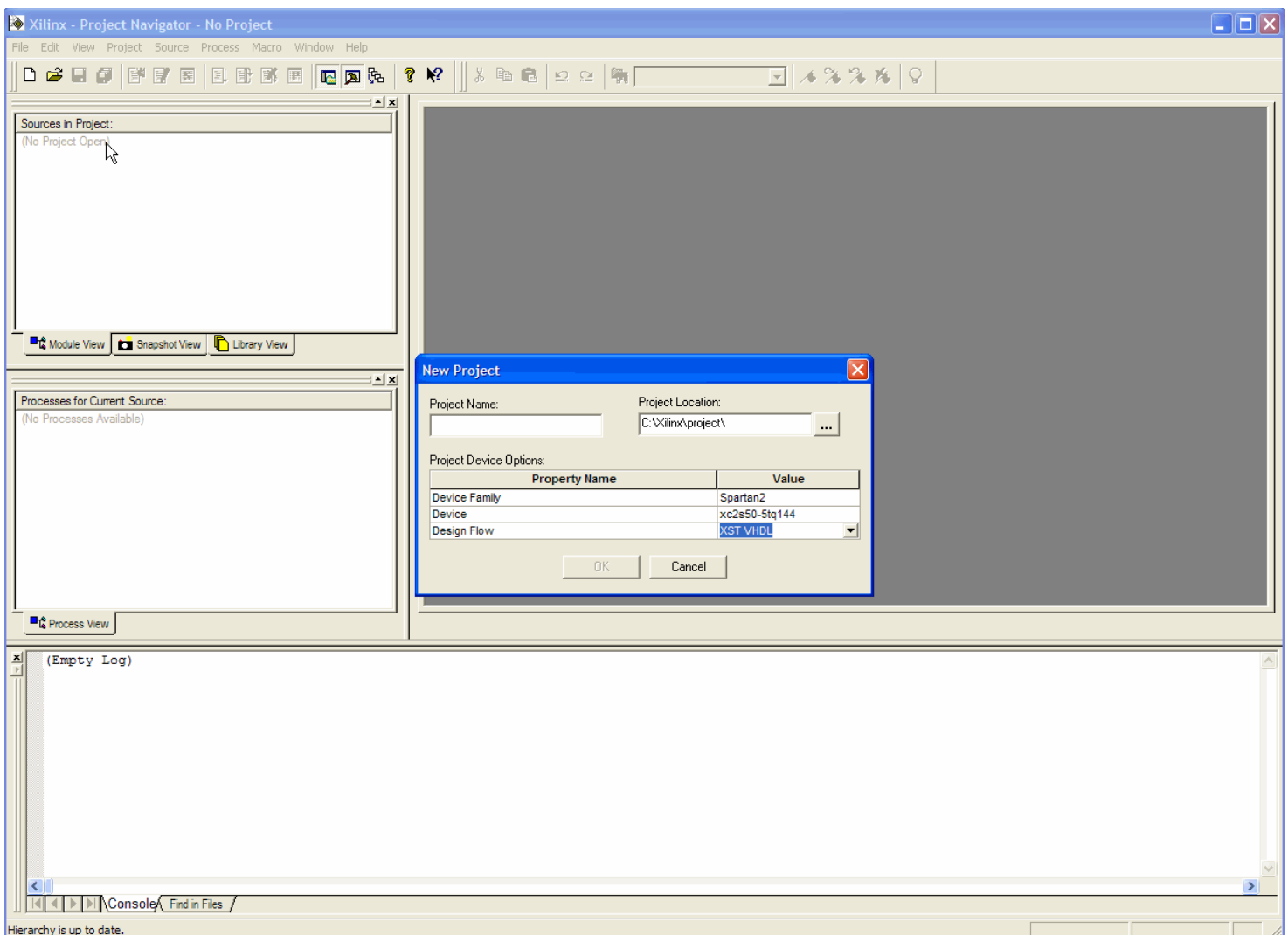
When installing the Xilinx ISE software, it's easiest to select everything to install. An exception is the two download cable drivers, which aren't needed unless you are connecting a Xilinx FPGA development board to your PC. This is a large software package – be sure that you have sufficient disk space before proceeding with the installation.

The class website (<http://ecen3233.okstate.edu>) has links to a great deal of information related to the Xilinx devices and software – see the Xilinx Info link in particular. Much useful information is available on the Xilinx university website (<http://www.xilinx.com/univ/index.htm>), at the Xilinx University Resource Center (<http://xup.msu.edu>), and as part of the ISE on-line documentation. The *MSU* site contains design examples, tutorials, and other useful resources.

The exercises below assume you are using the computers in the digital logic lab (ES 301). The locations of the program and program icons may be different depending on how you choose to install them on your own PC or if you are using the PCs in one of the CEAT computer labs, but the software should behave similarly.

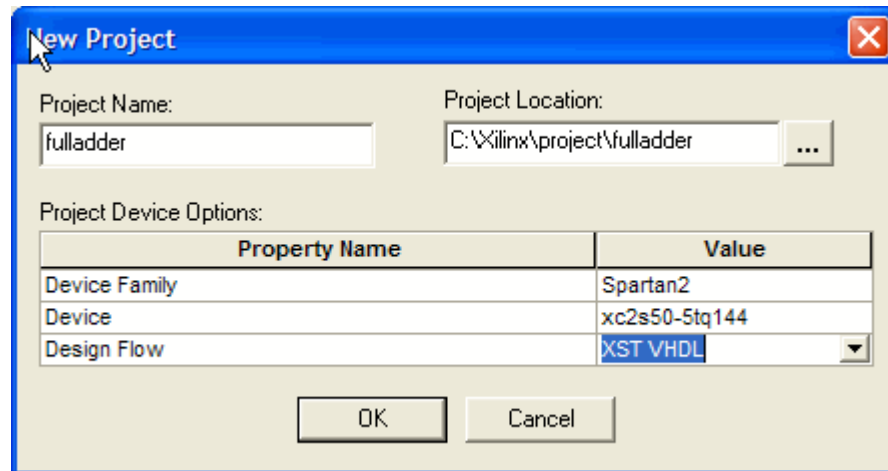
Prelab:

1. Open up the Xilinx® ISE program by clicking on the Xilinx Project Navigator icon that is on the desktop or in the Start menu. This will bring you to the “New Project” screen shown below. If the New Project dialog does not appear, go to File – New Project to open it. If a project is already open, indicated by a list of items appearing in the Sources in Project window, then select File – Close Project first.



2. You should now be in the “New Project” window – a typical example is shown below. Type in the name “fulladder” or another name of your choice, but don’t hit “OK” yet.

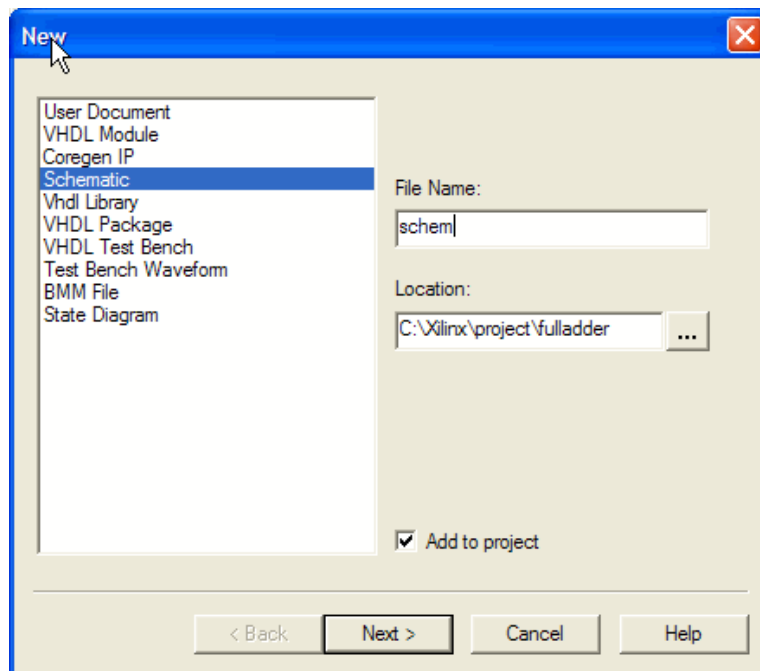
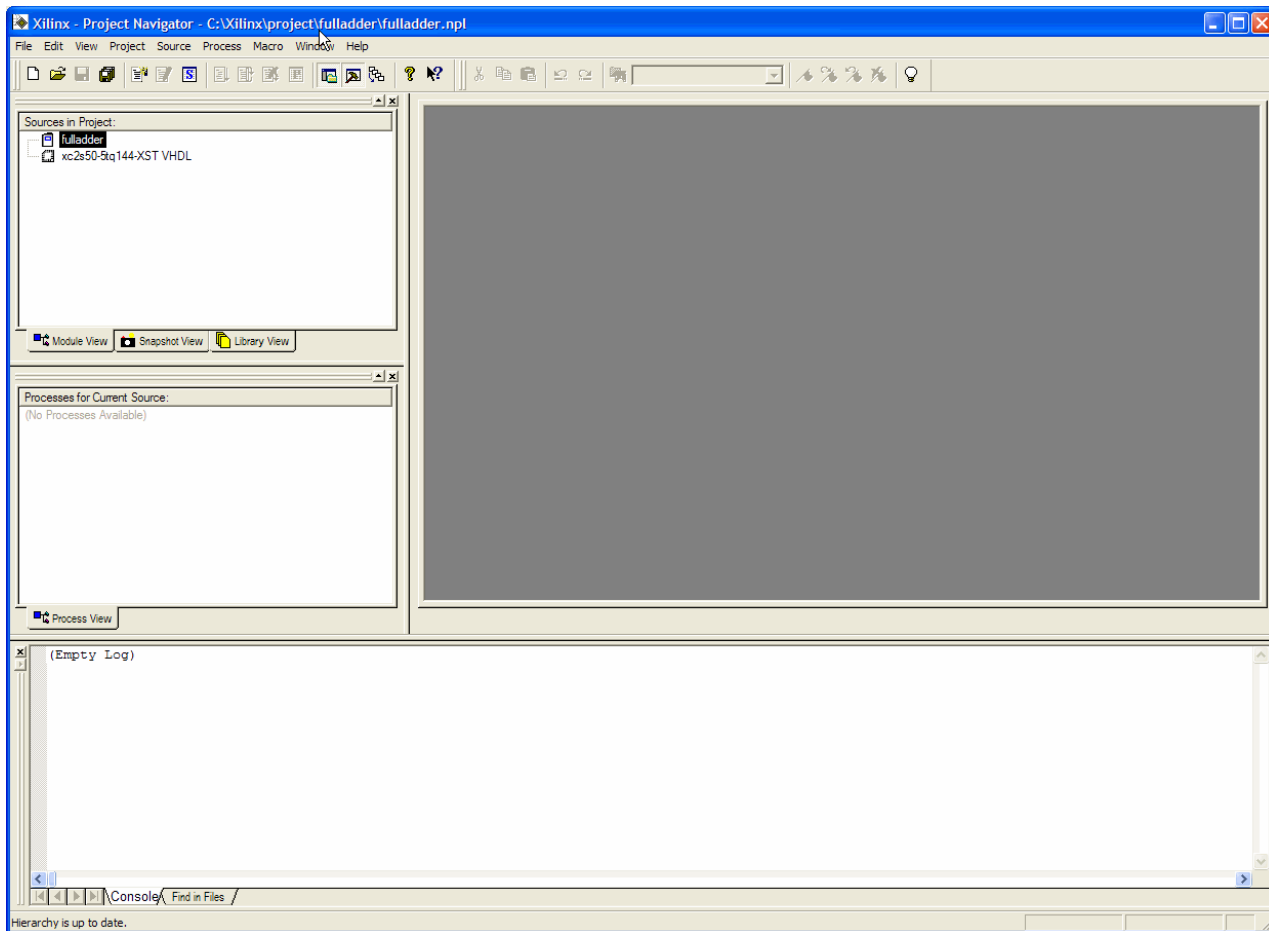
Make sure that the path to the project location is set, select the Spartan2 device family, enter **xc2s50-5tq144** for the device, and select **XST VHDL** for design flow. Now hit “OK”.



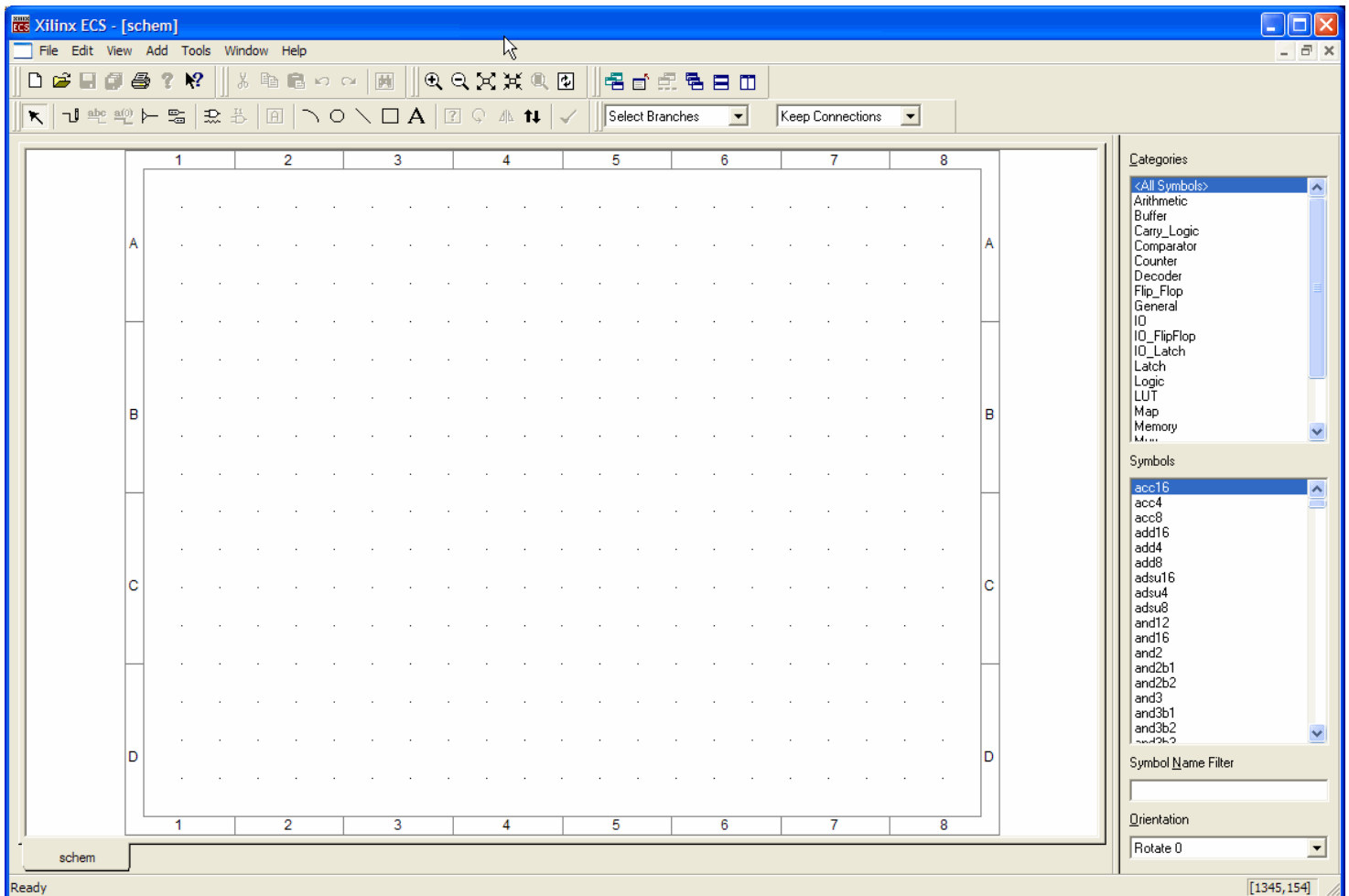
The last selections you made in this step specify a particular programmable logic device part number (the Spartan II xc2s50) having a 144-pin package (tq144) and a speed grade of “5”. This detail is not important now, but it will become important in a few weeks when we begin using Field Programmable Gate Arrays (FPGAs) to implement complex logic designs. It’s best to get in the habit of entering the correct specifications now! The design flow choice XST VHDL tells the software to generate its intermediate device and circuit models using the VHDL hardware description language.

3. You should now be in the “Project Navigator” window. On the left is the initial framework of the project structure that you are beginning to create. To proceed, you will insert into the project a schematic of the full adder you are going to design.

Click on the project name or on the device in the Sources in Project window, then right click to open the list of available sources to add. Select Schematic as shown below. Enter a name for your schematic, make sure Add to Project is selected, and select NEXT and then FINISH. The Xilinx ECS window should open after a few seconds. This is where you will actually draw your schematic diagram.



- The Xilinx ECS screen – also known as the Schematic Editor – will appear as shown below. Along the right hand side of the screen is a list of the library symbol categories along with the actual symbols corresponding to the various digital functions that are available. These consist of functions ranging from very simple logic operations (AND, OR, NOT, for example) to complex high-level compositions of functions. These library functions do not correspond to actual physical logic devices – they are merely functions that Xilinx understands.

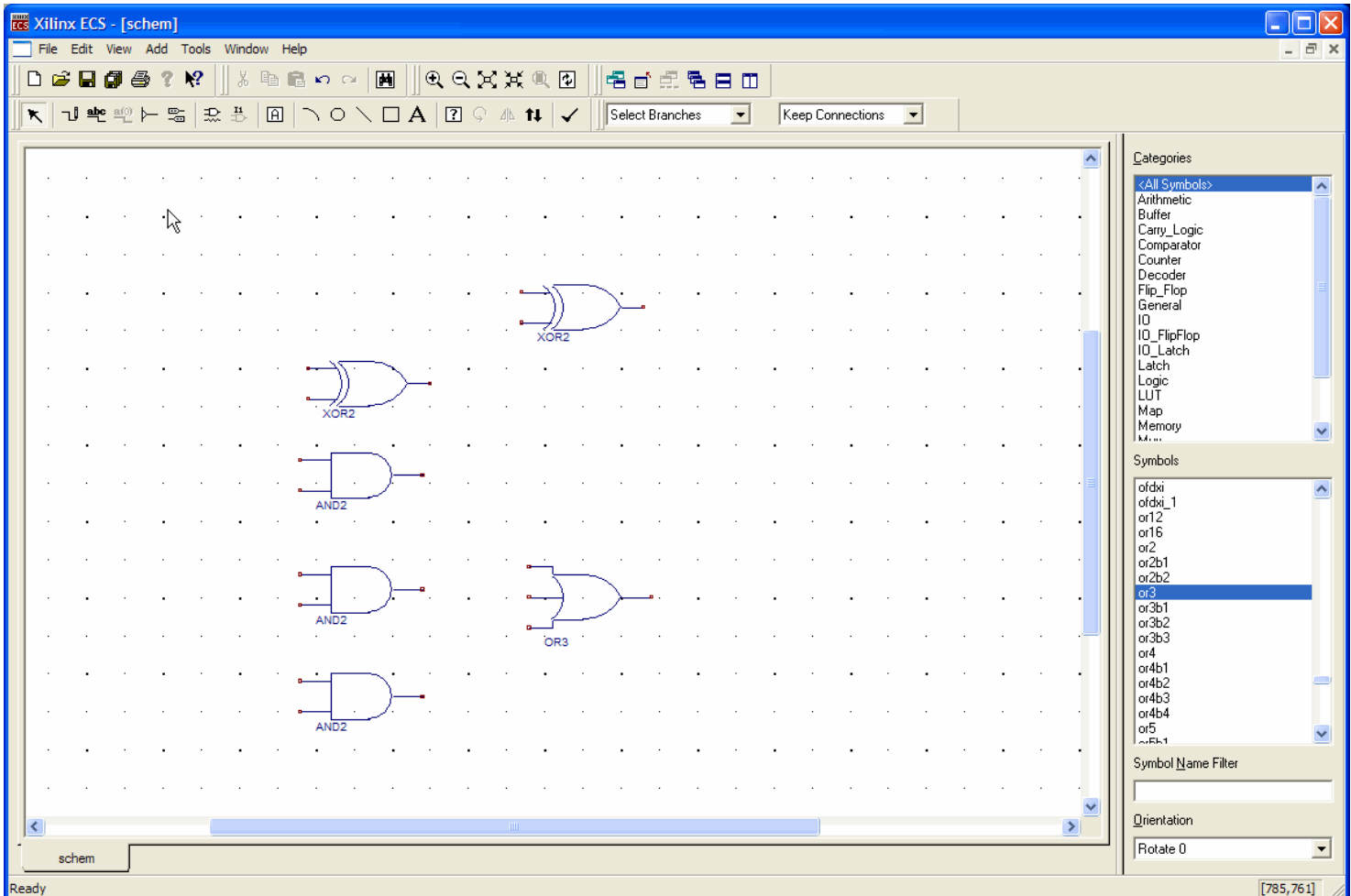


Now you are going to start placing parts to build a circuit. The circuit you are going to build is a very simple binary arithmetic circuit called a Full Adder, shown on the following pages. At this point, it's not important if you don't fully understand what a full adder does, although we discussed it in class earlier. We just want to illustrate the steps of implementing a circuit using this software since it's a task you will repeat many times this semester.

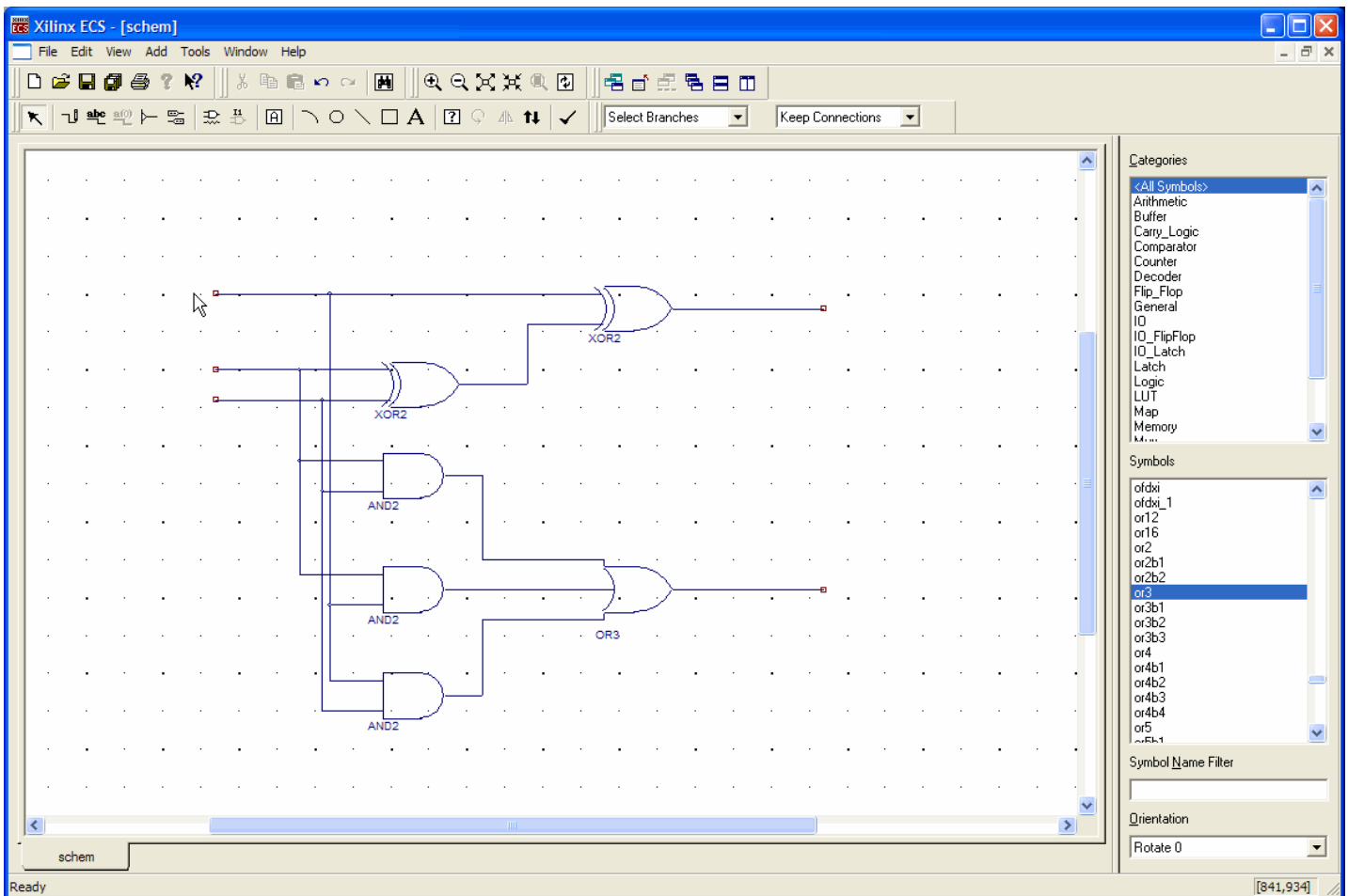
- We will be using three different kinds of logic operations/functions and a few other parts. Scroll through the list, click on each of the parts listed below (one by one), and place them onto the work area. Do this by clicking on a part in the list and then

moving the mouse into the work area and clicking again. You will be drawing the schematic shown below, so place the parts in the general locations where they will be needed. Pressing the escape key will clear the selection. An example is shown below – note that you can zoom in and out to make the schematic easier to work with.

3 - AND2	(2 Input AND Gate)
1 - OR3	(3 Input OR Gate)
2 - XOR2	(2 Input EXCLUSIVE-OR Gate)

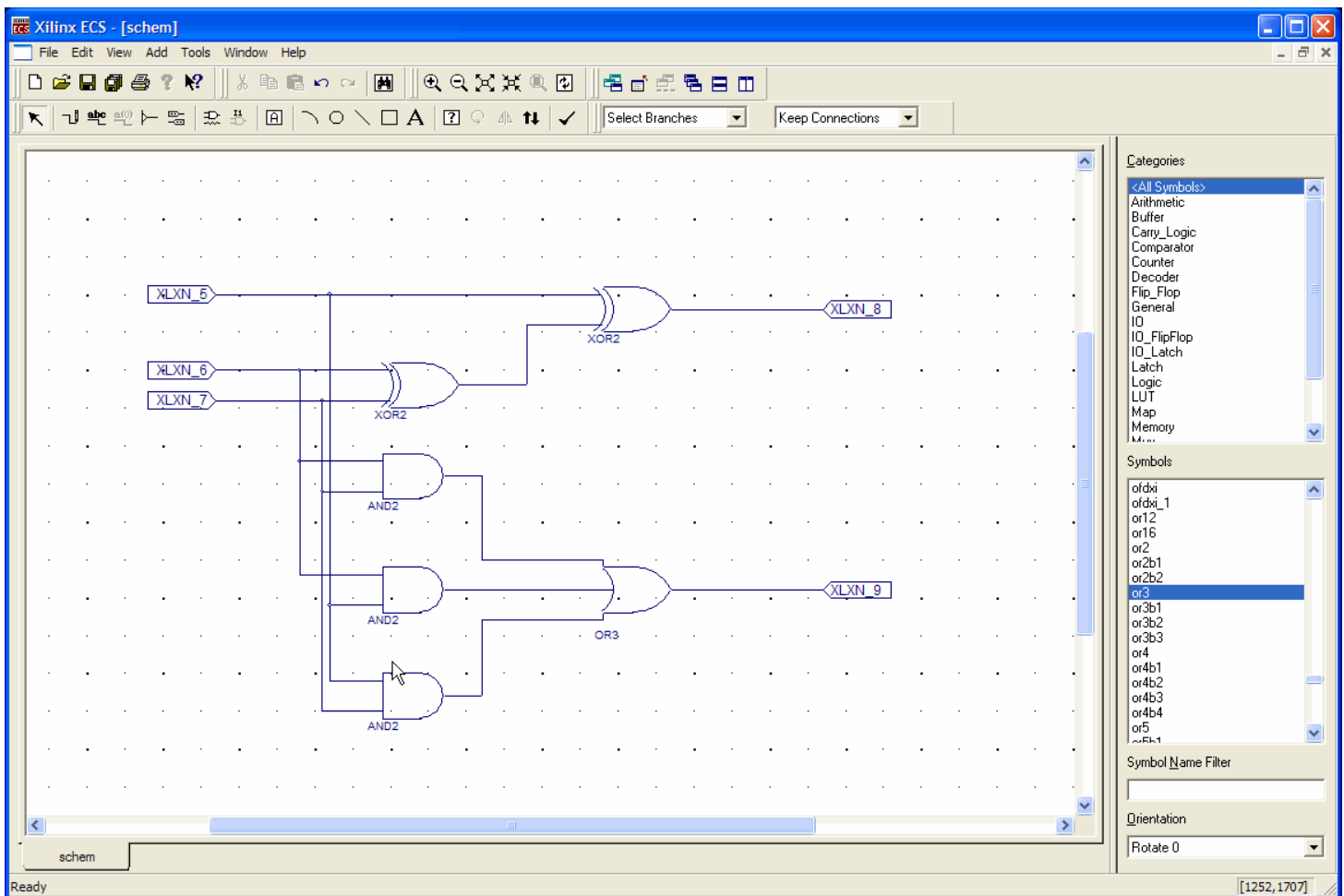


- To draw wires between symbols, click on the second button from the left on the lower toolbar – the one with a pencil. Or, you can press control-w. This selects the wire drawing tool. To draw a wire, click on the terminals of the parts you want to connect. To move parts, click on the arrow button, click on the part you want to move, and drag it. To delete an item, select it by clicking on it and then pressing the delete key. Try these operations a few times to become familiar with them. As before, the escape key will clear the selection and mode. Your schematic with all the parts connected should now look as shown.

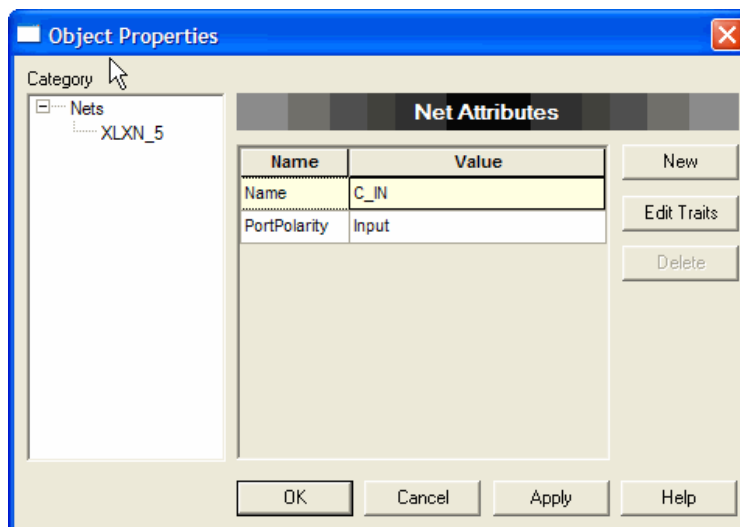


Sometimes two objects may appear to be connected when they're not. If a wire is connected to a part, when you move the part the wire will move with it. Unconnected wires are not uncommon and will result in a circuit that won't simulate correctly.

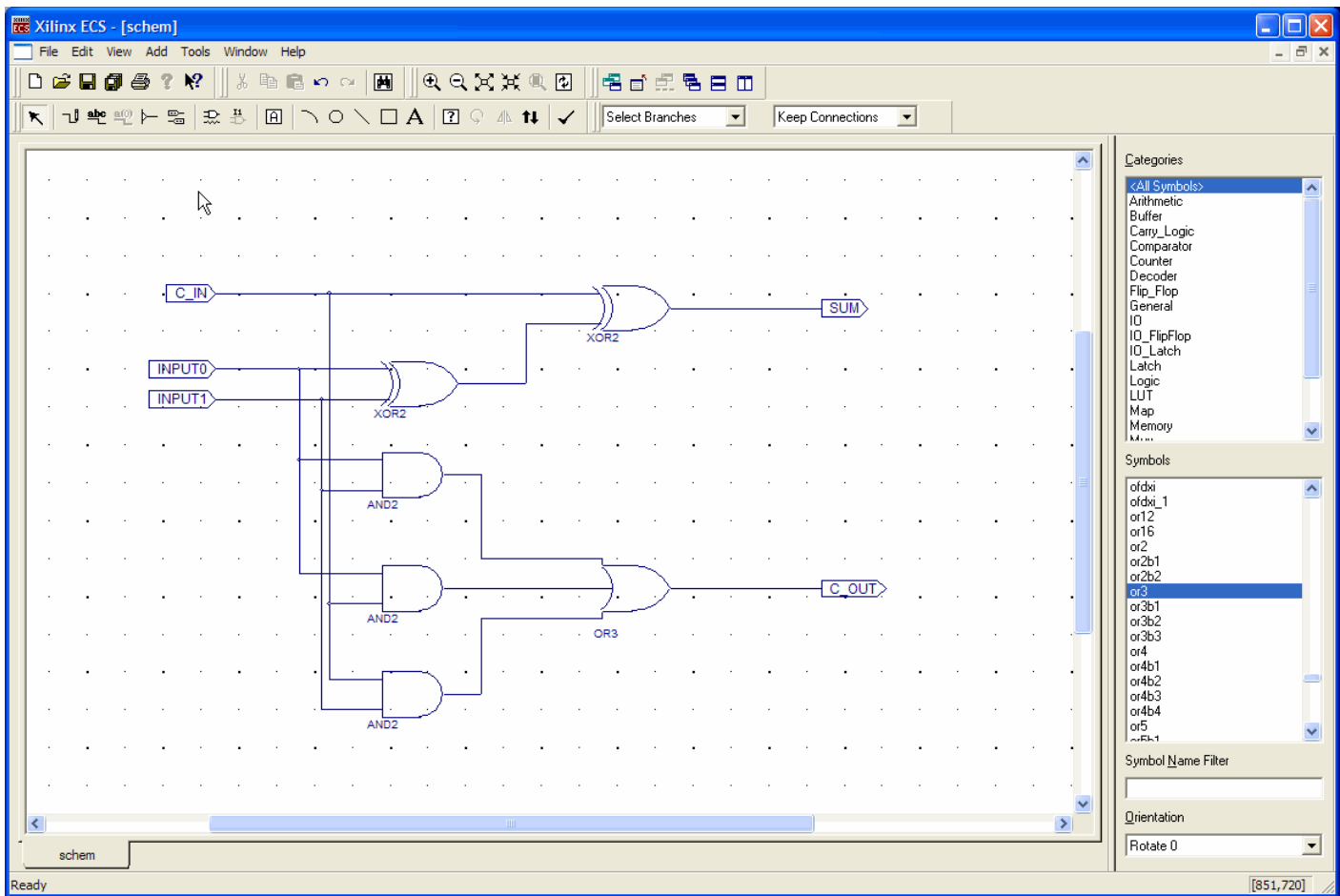
7. To complete the schematic, we need to tell Xilinx about the circuit inputs and outputs. Select I/O Marker mode by pressing the sixth button from the left on the lower tool bar – the one with the two opposing tabs. Click on each input and output (all five). Xilinx will give each input and output an arbitrary name. At this point the names are not important. Note that the I/O markers are oriented differently for the inputs (on the left) and the outputs (on the right). Your circuit should look like the one below.



8. Now that the schematic diagram is complete, we will name the inputs and outputs to correspond to the actual signals. This is called adding net names. Double-click one of the I/O markers and a dialog box (below) will appear. Type in the desired Name and make sure the polarity is properly selected (input or output).



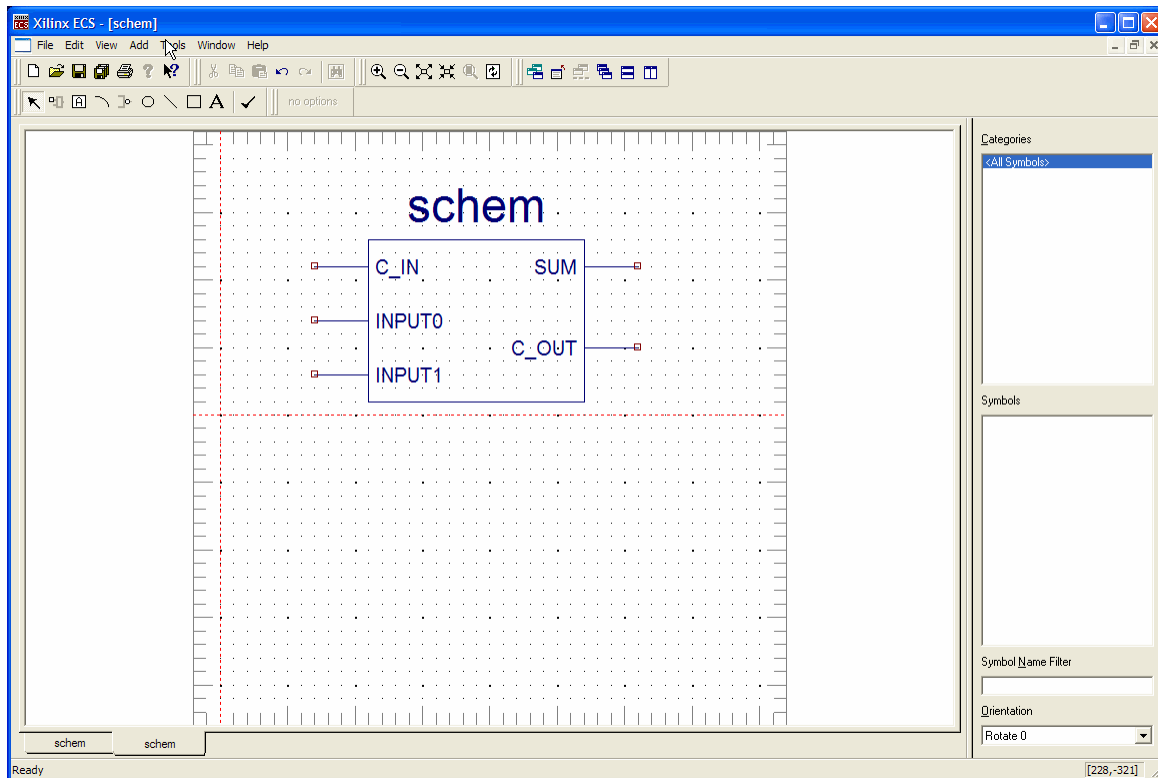
Repeat this procedure for all five I/O terminals. Name the remaining two inputs INPUT0 and INPUT1, and name the two outputs SUM (upper) and C_OUT (lower). Your final schematic should look like the diagram below.



- Often it's useful to create a new symbol to represent a circuit we have designed. This is because we may be more interested in what a circuit does (the input-output relationship or function performed) rather than precisely how the circuit is constructed internally. Or, the circuit may be very large and we might prefer to not see the details. Or, maybe we simply want to make the circuit easily reusable in other designs. Such a symbol that encapsulates a circuit is sometimes called a *macro*. Select Tools – Create Symbol, and click OK. Your circuit will now be represented as a block, shown below.

Your new symbol will now appear as one of the functions in the parts library and can be selected and used just like any of the other parts in the library.

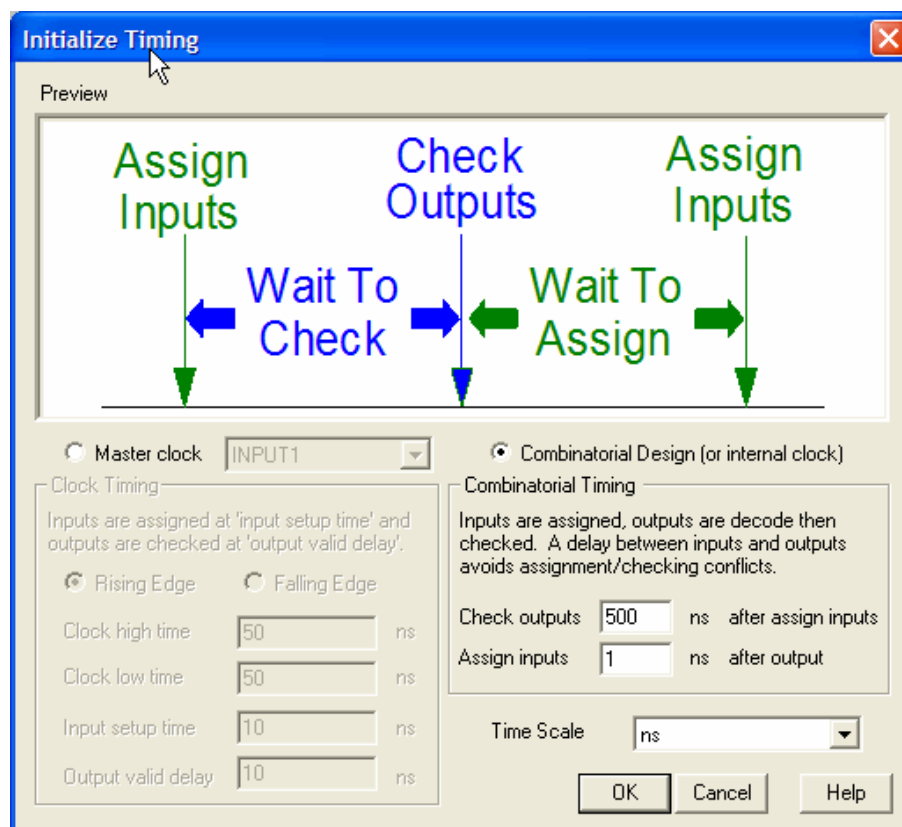
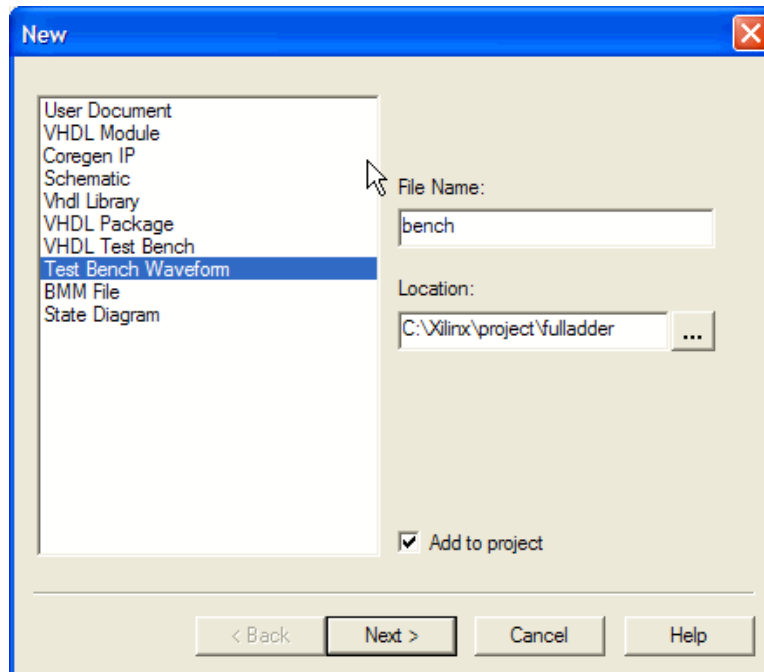
Close the schematic editor, being sure to save changes to your schematic. Return to the Project Navigator.



10. Now that the schematic has been drawn, we can use the Xilinx software to simulate the operation of the circuit and check to see if it functions as intended. We do this by using a timing diagram, which will be discussed in more detail later in lecture.

We will use the software to define and generate the desired input waveform(s) for the circuit simulation. The software will apply our input waveform(s) to the circuit and produce simulated output(s) in graphical form. In the Project Navigator, right click on the current part in the Sources window and select New Source. Then select Test Bench Waveform, and supply a name as shown below.

In a few seconds, the HDL Bencher window will appear. This window is used to define/describe a test waveform for the simulation. Specify 500 ns for Check Outputs and 1 ns for Assign Inputs. Respond OK. Note that the word “Combinatorial” in this window should really be “Combinational”. A combinational circuit is one that is constructed from a collection of simple logic functions such as the full adder, as opposed to a sequential circuit which we will discuss in detail later.



You should now see the window shown below. In the upper portion are the input and output variables that you specified for your schematic along with a time axis corresponding to your prior data inputs. In the lower portion is the VHDL code that

describes your project – the VHDL code can be ignored as it is beyond the scope of the class. It's sufficient to note that the Xilinx software uses the VHDL hardware description language internally to represent your circuit.

The blue markers indicate inputs; the yellow markers indicate outputs. Initially, both inputs and outputs are blank.

The screenshot displays two windows from the HDL Benchers software. The top window, titled "BENCH.TBW - HDL Benchers(tm)", shows a timing diagram. The vertical axis is labeled "Time (ns)" and has markers at 0, 501, 1002, 1503, 2004, 2505, 3006, 3507, 4008, 4509, 5010, and 5511. The horizontal axis represents time in nanoseconds. There are three input signals: C_IN (blue), INPUT0 (blue), and INPUT1 (blue). There are two output signals: C_OUT (yellow) and SUM (yellow). A green text box in the waveform area reads: "Waveform created by HDL Benchers 4.1i Source = schem.vhf Sat Jan 18 23:20:22 2003".

The bottom window, titled "schem.vhf - View HDL Source", shows the VHDL source code for the circuit. The code is as follows:

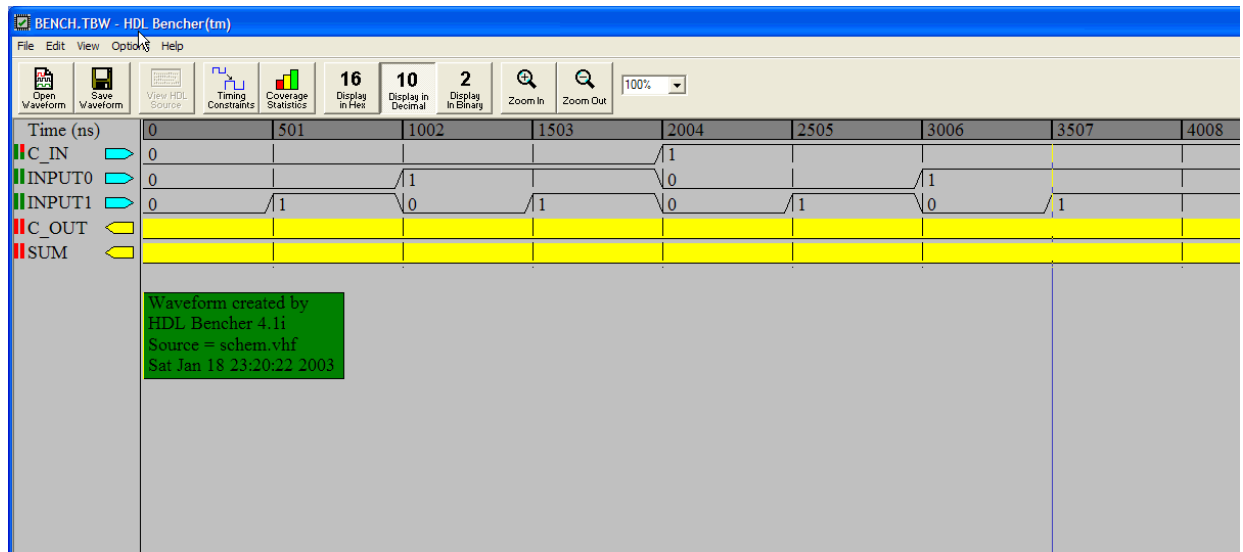
```

9  ENTITY schem IS
10     PORT ( C_IN : IN STD_LOGIC;
11           INPUT0 : IN STD_LOGIC;
12           INPUT1 : IN STD_LOGIC;
13           C_OUT : OUT STD_LOGIC;
14           SUM : OUT STD_LOGIC);
15
16 end schem;
17
18 ARCHITECTURE SCHEMATIC OF schem IS
19     SIGNAL XLXN_1 : STD_LOGIC;
20     SIGNAL XLXN_2 : STD_LOGIC;
21     SIGNAL XLXN_3 : STD_LOGIC;
22     SIGNAL XLXN_4 : STD_LOGIC;
23
24     ATTRIBUTE fpga_dont_touch : STRING ;
25     ATTRIBUTE fpga_dont_touch OF XLXN_1 : LABEL IS "true";
26     ATTRIBUTE fpga_dont_touch OF XLXN_2 : LABEL IS "true";
27     ATTRIBUTE fpga_dont_touch OF XLXN_3 : LABEL IS "true";
28     ATTRIBUTE fpga_dont_touch OF XLXN_4 : LABEL IS "true";
29     ATTRIBUTE fpga_dont_touch OF XLXN_5 : LABEL IS "true";
30     ATTRIBUTE fpga_dont_touch OF XLXN_6 : LABEL IS "true";
31
32 BEGIN
33
34     XLXN_1 : AND2

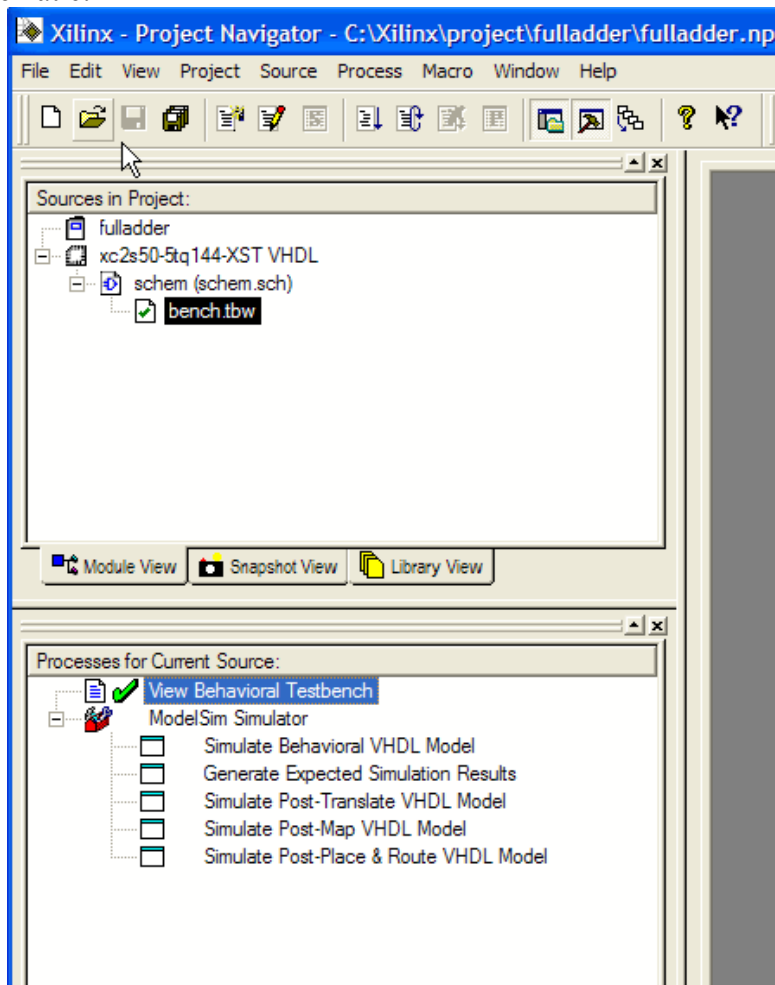
```

Click on the time intervals in each of the three input waveforms to set logic-0 or logic-1 as shown below. There should be eight combinations since there are three input variables, counting from 000 to 111 (zero to seven). This defines the input waveforms to be used in the simulation.

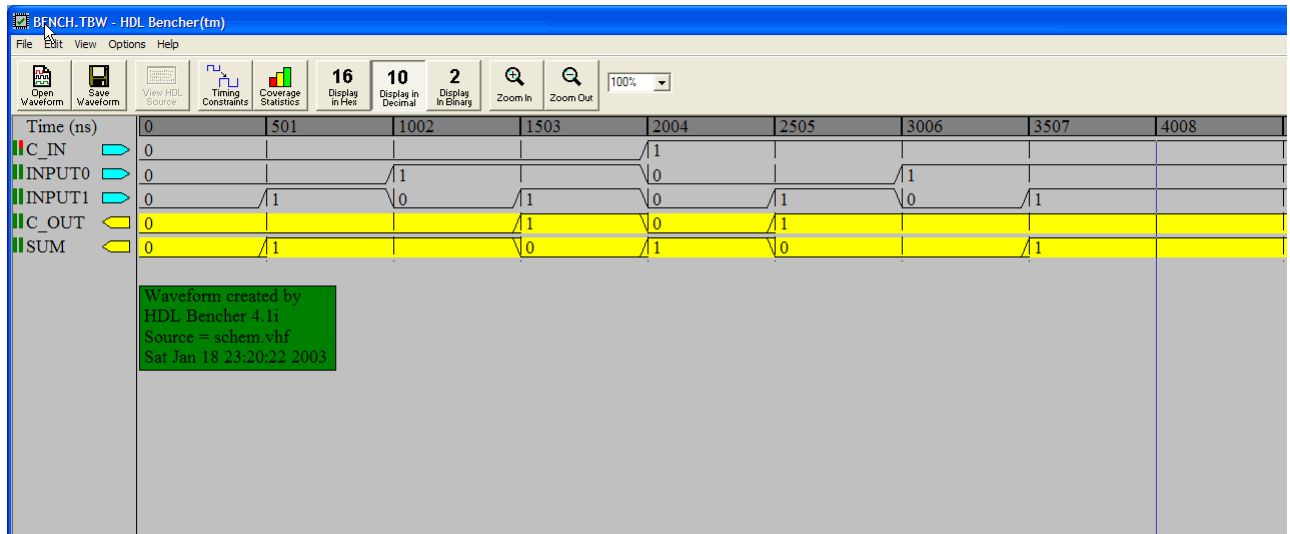
Save the waveform and close the window, returning to the Project Navigator.



- Expand the Processes for Current Source corresponding to ModelSim. ModelSim is the simulator software used to simulate your design. ModelSim will take the input waveform(s) you created and generate the simulated output waveform(s) based on your schematic.



12. Double-click on Generate Expected Simulation Results. In a few seconds the HDL Bencher window should open again with the simulated outputs filled in. You should see a result that looks like the waveform below.



Examine the timing diagram that HDL Bencher has produced. If you compare the resulting simulated waveforms with the expected output of a full adder from lecture, you will see that this design behaves as expected. Through simulation we have verified exhaustively that this full adder behaves as expected. This is *exhaustive* because we applied all the possible input combinations that can be formed from three variables and observed the output in each case. The schematic and the simulation waveform should be printed to document the design and its correct operation.

Logic simulation is an excellent method to verify the correct functional operation of a circuit you have designed. This is an important design step that should be performed prior to constructing a circuit in hardware.

Note also that there are other methods for specifying the input waveforms. Manually specifying each input condition is tedious and time consuming. We will explore other techniques later.

13. When you come to lab, you will construct the full adder circuit you just simulated, using electronic components from your parts kit. Each of the logic gates that you used in your design (AND, OR and XOR) is available as an integrated circuit (IC) (an electrical device consisting of one or more logic functions in a single plastic or ceramic package typically having either fourteen or sixteen pin connections).

You will be using the 7400-series of ICs, a family that consists of parts implementing many different logic functions – some of which are quite complex. For example, the 7408 or 74LS08 device implements four identical 2-input AND gates in a single 14-pin dual-inline package (DIP).

To use the devices to construct your circuit, you must properly wire the logic gates together to implement your circuit. This is accomplished by connecting the inputs and outputs of each logic gate as described by your schematic. In addition to the signal connections (inputs and outputs), each IC must also be connected to power (+5V) and ground (0V). In order to do this, you must know how the gates inside each IC are connected to the pins on the IC.

Look on the class web site and find the link for Data Sheets and Pin-out Diagrams (under Resources) – follow this link. There are links on this page to data sheets for each of the ICs you are likely to use during the semester – and a few you aren't. A data sheet provides operational and physical information about a part. Click on the link for the 74LS08. For now we'll concentrate on the first page, shown below.

FAIRCHILD
SEMICONDUCTOR™

March 1998

DM74LS08

Quad 2-Input AND Gates

General Description
This device contains four independent gates each of which performs the logic AND function.

Features

- Alternate Military/Aerospace device (54LS08) is available. Contact a Fairchild Semiconductor Sales Office/Distributor for specifications.

Connection Diagram

Dual-In-Line Package

Order Number 54LS08DMQB, 54LS08FMQB, 54LS08LMQB, DM54LS08J, DM54LS08W, DM74LS08M or DM74LS08N
See NS Package Number E20A, J14A, M14A, N14A or W14B

Function Table

$Y = AB$

Inputs		Output
A	B	Y
L	L	L
L	H	L
H	L	L
H	H	H

H = High Logic Level
L = Low Logic Level

DM74LS08 Quad 2-Input AND Gates

This page identifies the 74LS08 as a *Quad 2-Input AND Gate*, meaning that it contains four such AND gates. It illustrates the layout and pin assignments of the gates within the IC by means of a top view of the device – this picture also illustrates the general appearance of the part. Below this are a Boolean formula and a truth table further illustrating the function performed by each gate. The truth table shows that only when all inputs are logic-1 (high or true) will the output be logic-1 – an AND function.

In order to label the IC pin numbers on your schematic, you will have to print out your full adder schematic from Xilinx and label it by hand with appropriate pin numbers for each gate. All the gates in a given part are identical, so it doesn't matter which gates you select. After drawing and simulating your circuit, you should always bring your labeled schematic to lab with you as an aid in constructing your circuit.