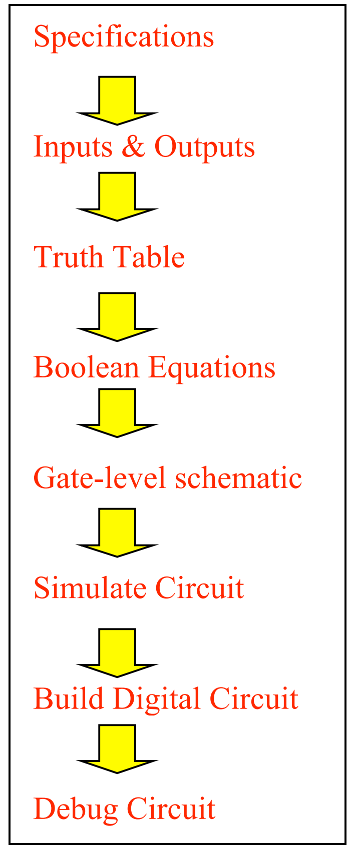
Programmable Logic Design Techniques I

The design of digital circuits is a multi-step process. It starts with specifying what the circuit must do followed by defining what a circuit receives as inputs and it generates as outputs. Next the designer has to create a truth table, which lists the output values for all possible combinations of input values. Once the truth table is written down, the designer has to derive Boolean equations that describe how each binary output can be computed from the binary inputs using logical operations such as AND, OR, NOT etc. Next, the Boolean equations are transformed into a gate-level circuit schematic drawing. Each AND, OR etc. operation in a Boolean equation is replaced with a corresponding AND gate, OR gate etc. in the schematics. Inputs and outputs of these gates are wired as needed according to the specifications of the logic. Before building the circuit it is a good thing to make sure that all previous steps have been completed correctly. This is done by manual or computer simulation. If successful, a physical circuit is built. Real inputs are applied to the circuit and possible bugs, if any, are fixed.



You have already built and tested some simple digital circuits using transistor-transistor logic (TTL) devices, and a breadboard where the connections were made using wires. Please, recall Lab 11 and Lab 12 for more details. As you have probably realized, this method of building circuits is not without problems: not every type of TTL logic circuit may be immediately available; wires are often plugged into the wrong place and a lengthy check must be made to find errors. Once the circuit works it has to be taken apart to make room for the next circuit; and last but not least, relatively complex digital circuits of more than 10-20 gates are practically impossible to build and test. These problems may be eased if a different approach is followed and more elaborate tools employed. The design of digital circuits may still begin by describing the truth table. The details of the logic circuit needed to realize the truth table are, however, worked out by a logic-synthesis program and not by hand. The operation of the "virtual" circuit built is checked using a simulation program. If the circuit simulates correctly, the gates and wires are mapped into a Field Programmable Gate Array (FPGA) using specialized place & route programs. The FPGA contains logic gates and reusable interconnections within a single Integrated Circuit (IC). The programmed FPGA can be used independently or placed into a larger circuit where it will perform its functions. In this Lab you will be using the XILINX Foundation Series 4.2i software tools to create and test logic designs that can be downloaded into the XC4003E FPGA. *To be successful in this Lab you should review Labs 11 &12, Chapters 11 & 12 of DH and the Guides to the XILINX software and hardware posted on the PHY440 web site.*

Problem 1. Create a digital circuit of a single AND gate and verify its truth table.

To begin, click on the "Project Manager" icon. This will bring up the **Project Manager** Window. Click on “Create a New Project” and then “OK”. Enter the project name (your choice), project directory (keep the default one), type of design flow (Foundation Series 4.2i; Schematic), chip family (XC4003E), chip part number (4003EPC84) and device speed (default). Click **OK** to return to the Project Manager Window.

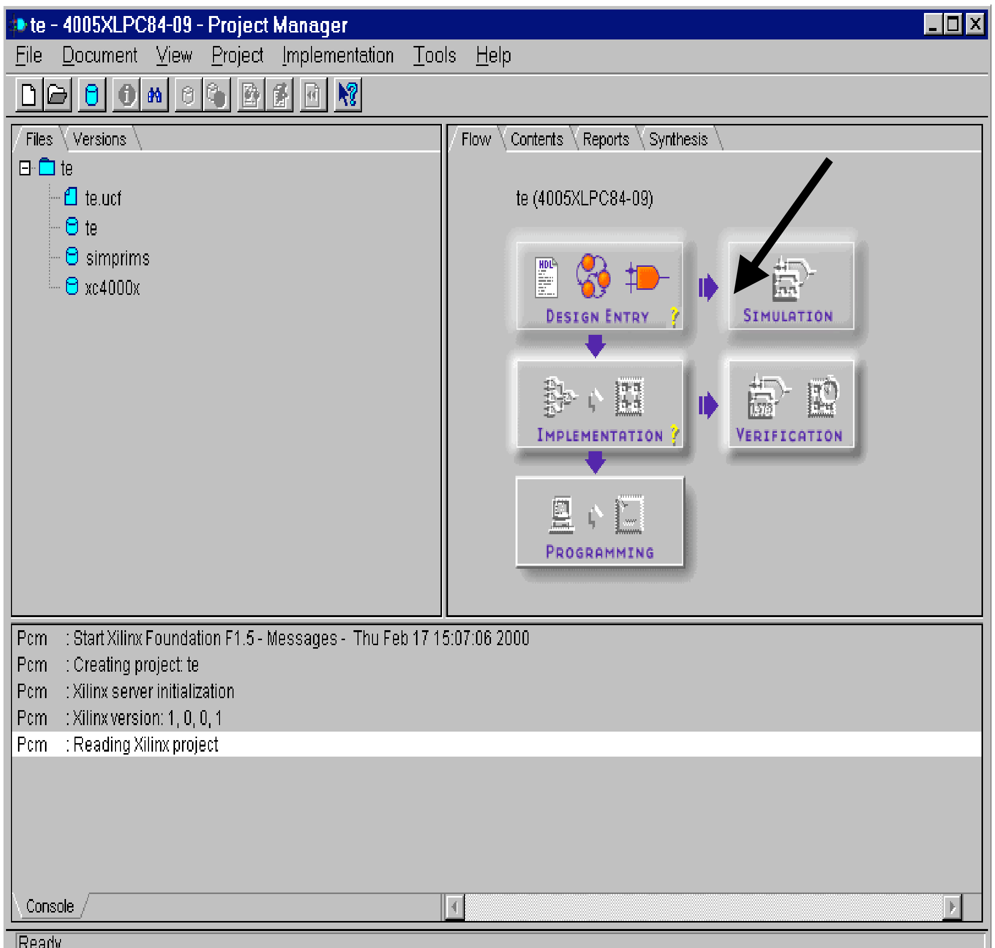


Figure 1: Project Manager home page

In the **Project Manager** window click on the **Schematic Editor** button (see the screenshot above) and a schematic editor window will appear (see below).

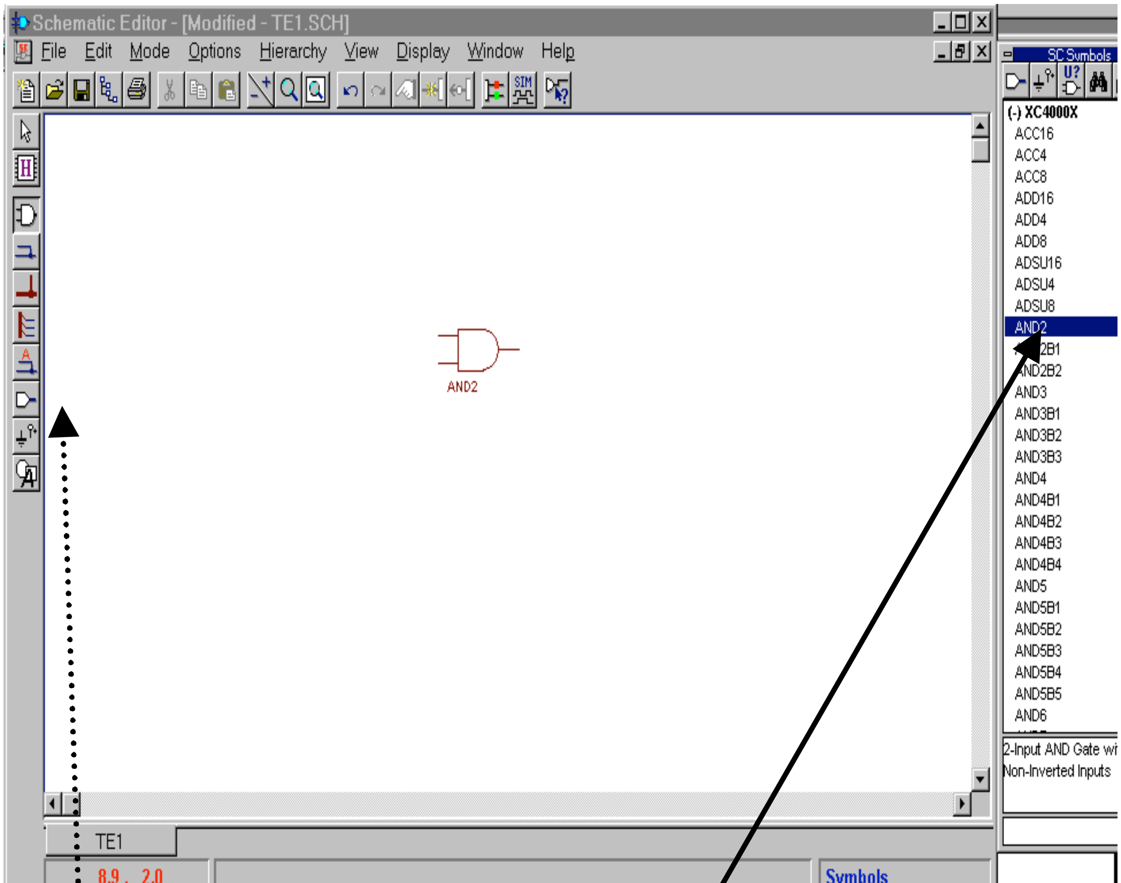


Figure 2: Design Entry window with 2 input AND gate

Select the **Mode** --> **Symbols** menu item and the **SC** window will appear with a list of logic components that we can use. Scroll the list of components in the **SC** window, click on AND2 (two-input AND gate), move the cursor into the drawing area and drop the AND2 gate there. We need to get our inputs and outputs into the circuit as well. To do this double-click on the **Inputs** button. A dialog window (see below) will appear in which you have to type the name and the type of each input and output.

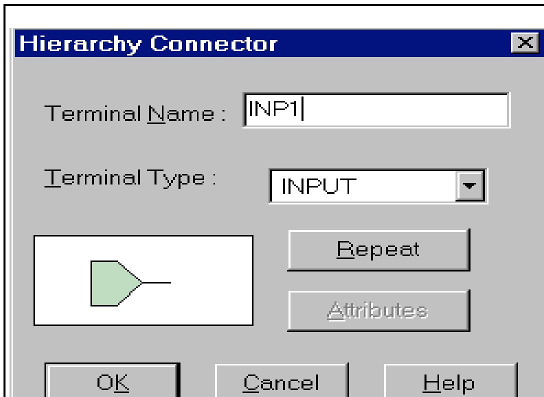


Figure 3: Connector Window

While we have entered the inputs and output terminals, we still need to add buffers between the terminals and the logic gate. The buffers indicate that the signals attached to them will actually enter and exit the FPGA chip via its I/O pins. To add input and output buffers we select **IBUF** and **OBUF** symbols, respectively, from the **SC symbol** window and drop them in the drawing area. The next step is to connect (wire) the inputs and outputs to the AND gate. Select the **Mode** --> **Draw Wires** menu and do the wiring. A line will appear connecting the inputs/outputs to the gate as shown below.

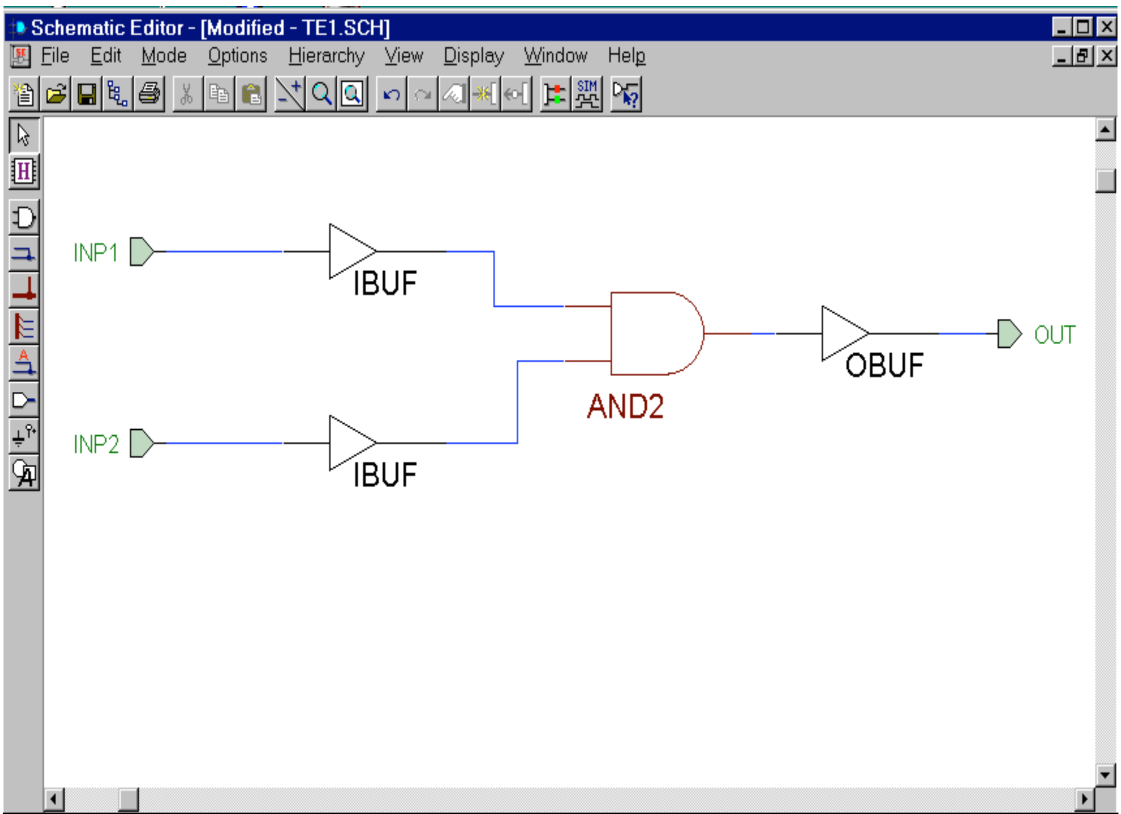


Figure 4: Window with wired up and gate

Now that the schematic is done, we need to check it for errors. First, select **Options** --> **Create Netlist.** This will activate a program that examines the schematic drawing and generates a machine-readable netlist which describes what type of gates are used (only one in our case) and how they are connected. Once it is done, select **Options** --> **Integrity test** to initiate an error check. The check should indicate that there are no errors. Then save the schematics using the **File** --> **Save** menu item. Also, the netlist created must be exported in a format that other XILINX tools understand. First click on the **Options** --> **Export Netlist** and then on the **Open** button. Finally, select **File** --> **Exit** to close the **Schematic editor** and return to the **Project Manager** window. Next we will use the functional simulator to see if what we have entered performs as intended. In this case we wish to verify the truth table for an AND gate.
Start the functional **Simulator** by clicking the corresponding button in the **Project Manager** window. This brings up the **Logic Simulator** window. The first thing to do is add the inputs and outputs of the logic circuit to the **Waveform Viewer** so we can see

what happens as the circuit is simulated. Do this by selecting the **Signal --> Add Signals** menu item. The **Component Selection for Waveform Viewer** window will appear. Click on one of the inputs to highlight it and then click on the **Add** button. Repeat for all the other terminals and then click on the **Close** button.
Now the inputs and the output are displayed but nothing happens since the two inputs are set to logic 0. We need to apply a stimulus to the circuit, so we select the **Signal --> Add Stimulators** menu item. This brings up the **Stimulator Selection** window. We are interested in the 16-bit counter labeled **Bc** (see below).

During a simulation, Bc counts up from with the bit values shown. We can test the response of our circuit (a single AND gate) to every possible combination of inputs by attaching the two inputs of our logic circuit to two of these bits, say B0 and B1. We do this by clicking on the name of the input in the **Waveform viewer** and then clicking on the corresponding bit-circle in the **Bc** section of the **Stimulator Selection** window. Once the two inputs are attached to the counter bits, we click **Close** to leave the **Stimulator Selection** window.

Next we set up the parameters that control the speed of the simulation. **Select**

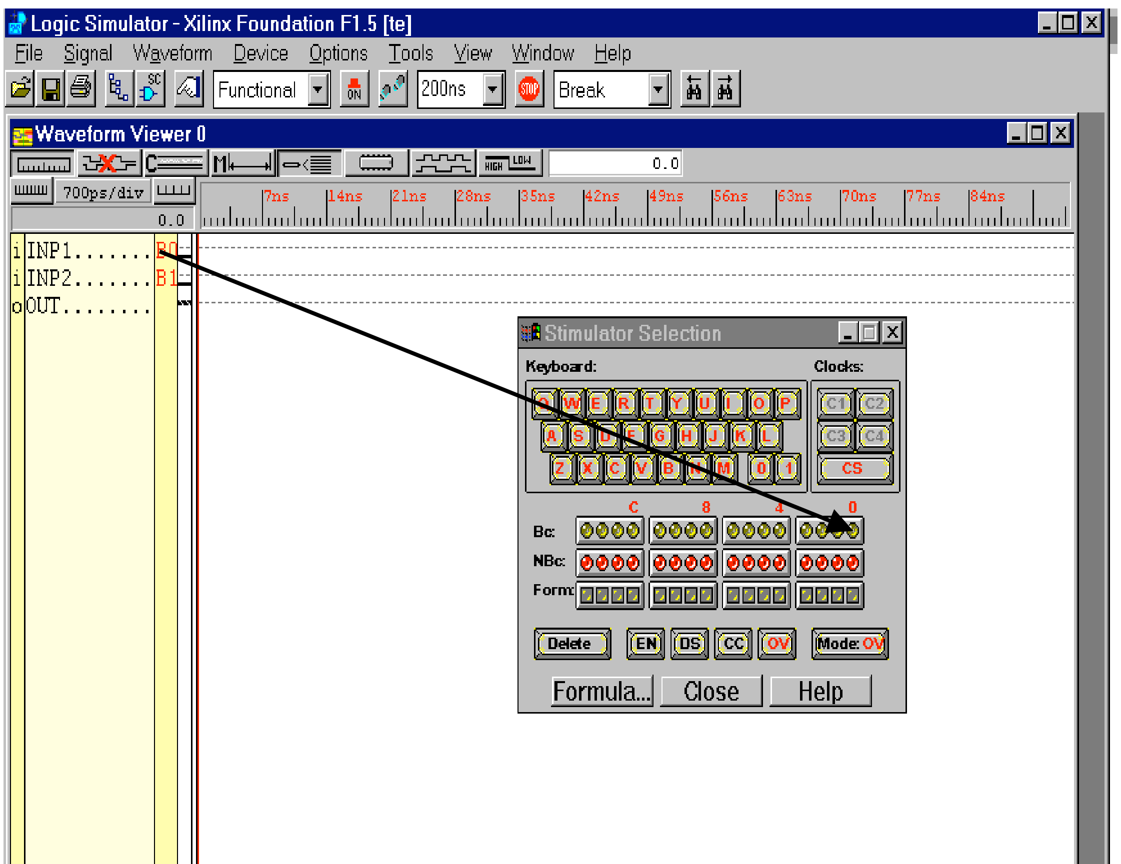


Figure 5: Attaching signals to inputs

Options --> Preferences and set the frequency to 50 MHz or the pulse length to 20 ns. The simulation precision setting determines the length of each simulation step. Choose a number that reflects the level of detail that interests you. Too small a step increases the

simulation time and makes the graphical delay too long. In this case a precision of 100 psec is more than adequate. Click **OK** when you have made your choice. Next go to Options and Start Long Simulation. Pick a time (one second is plenty) and push start. Almost instantly, the results of the simulation appear as shown below.
If your circuit is properly constructed, you will note that the output goes high only when the two inputs are high. This exactly matches the truth table for the AND gate.
Following the procedure described above verify the truth tables for OR and NAND gates.

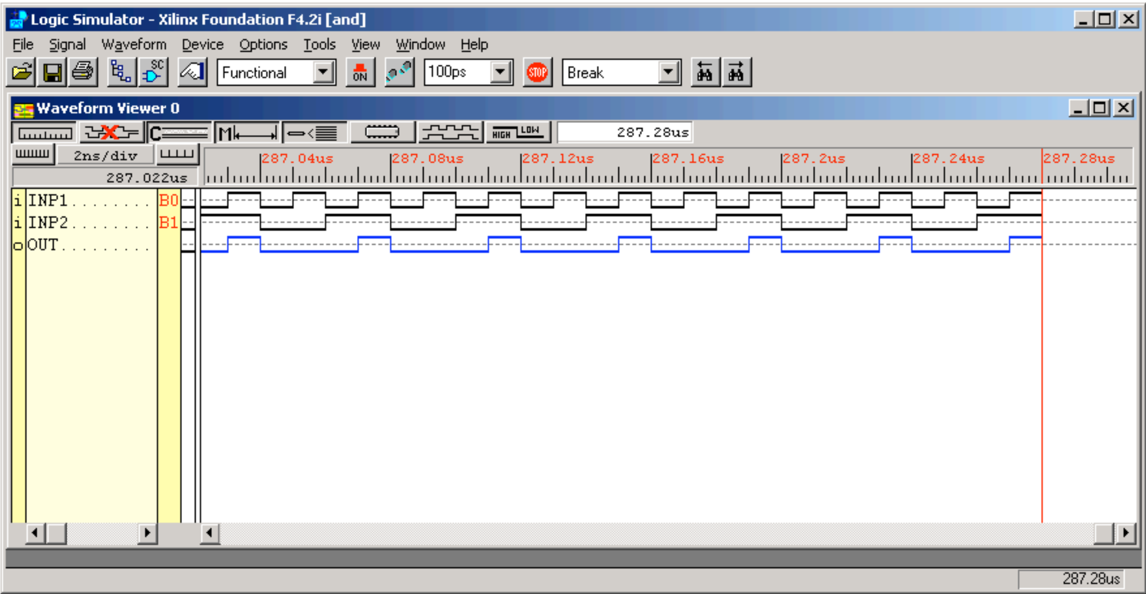


Figure 6: Results of simulation

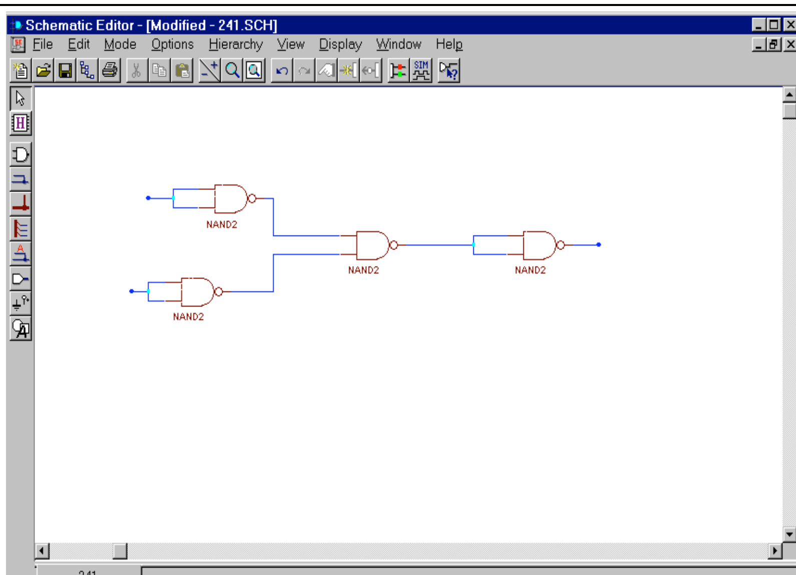


Figure 7: A NOR gate from 4 NAND's

Problem 2. Design the circuit in Fig. 7 and show (by simulation) that 4 NAND gates can make a single NOR gate (recall Experiment 8, Lab 11).

Problem 3. Design a 2-to-4 Decoder using 6 NOR gates (Recall Experiment 4, Lab 12). Compile the logic design and download it to the Xilinx FPGA (XC4003) demo board. Test the Decoder using one of the 7-segment LED's on the board.

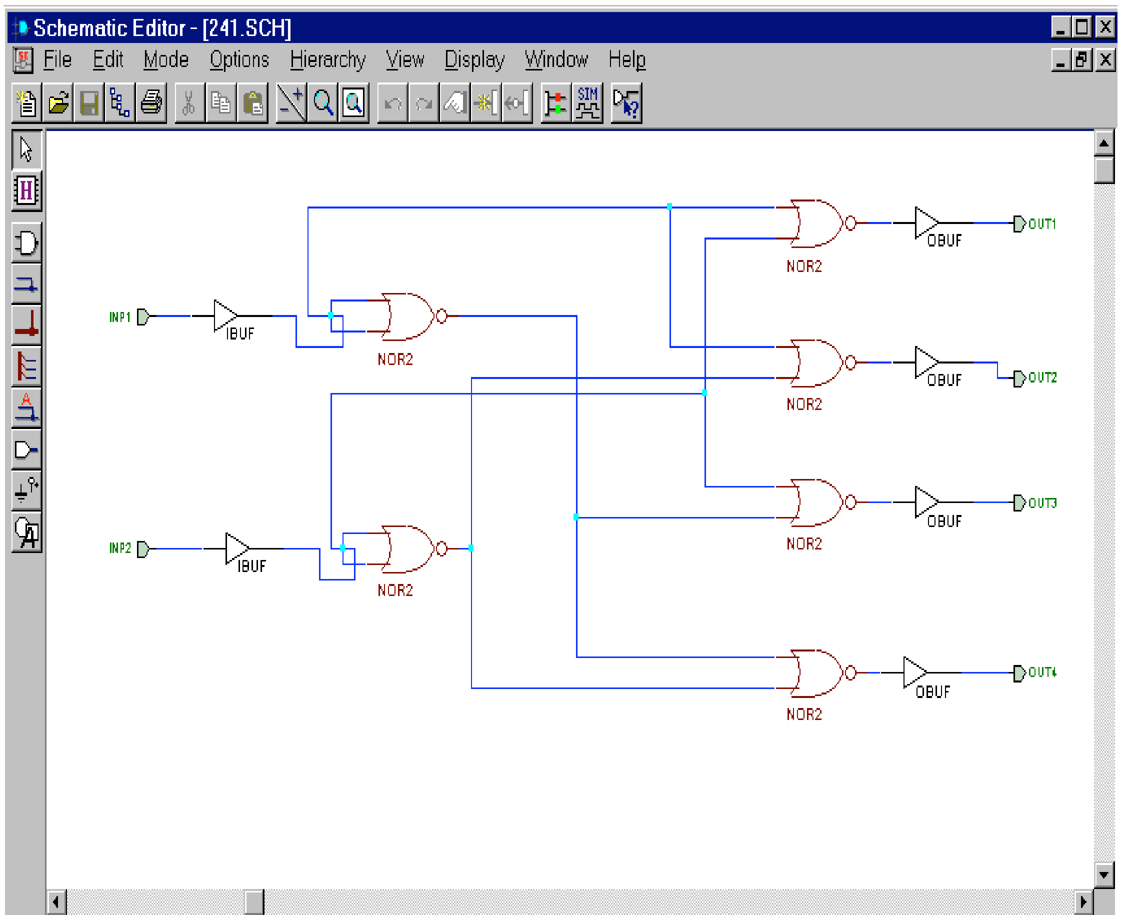


Figure 8: A 2 to 4 decoder

If you have designed and simulated the circuit in Fig. 8 successfully, we can proceed with compiling the design into a bit-stream that can be loaded into the FPGA chip and tested. The test is done by applying signals to certain pins of the FPGA that correspond to the inputs of the circuit. We also have to hook an LED to the pins that carry the outputs so we can visually observe if the circuit is operating correctly. First, we have to assign the two inputs of the circuit to two pins of the XC4003 FPGA which can be driven by a corresponding switch (labeled SW3) incorporated in the board. The pins we can use are # 19, 20 and 23 to 28. To do it, double-click on any IBUF to set its attributes. The **Symbol Properties** window will appear (see below). Click in the **Name** box and type **LOC**. Click in the **Description** box and type **p19**. Click the **Add** button and then OK. Let us assign the INP1 and INP2 inputs (the buffers!) to pins 19 and 20, respectively.

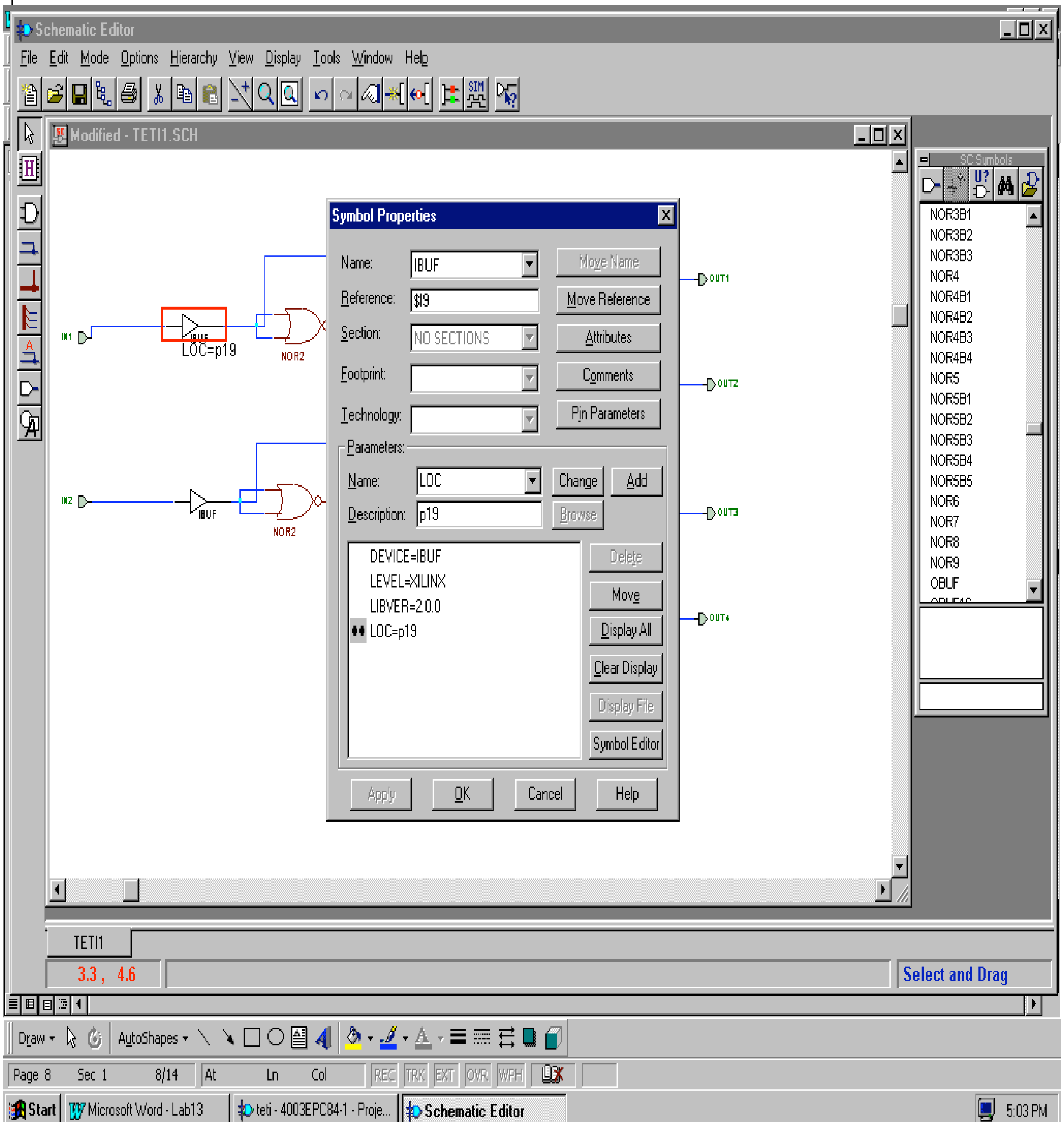


Figure 9: Assigning input/output pins

Let's assign the OUT1, OUT2, OUT3 and OUT4 outputs (the buffers!) to pins 49, 48, 47 and 46. Pins 49, 48, 47, 46, 45, 50 and 51 drive the 7-segment LED (labeled U8) on the board. Pin 41 drives the decimal point on U8. So by associating the output of the logic circuit with pins 49 to 46 we could observe the results coming out on the LED. Hopefully you ended up with a circuit like the one below:

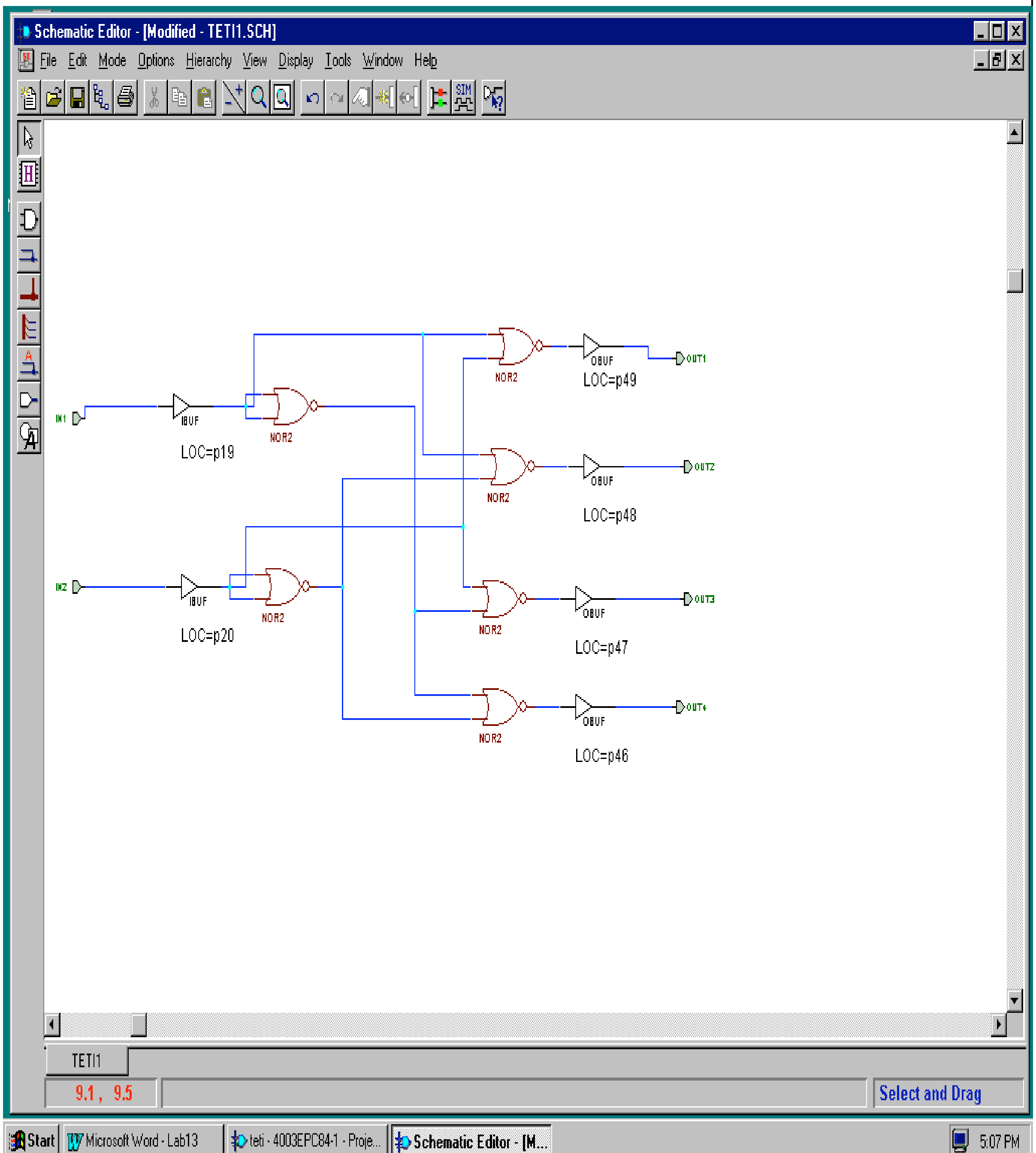


Figure 10: Schematic of 2 to 4 decoder

Once all inputs and outputs are assigned to pins do not forget to create and export a netlist! Save your design and **Exit** back to the **Project Manager**.

Now it is time to compile your design. Do it by selecting the **Implementation menu** item. When it appears Click on **Run**. The **Flow engine** (see below) will start up.

The compilation goes through the following steps: (they do not require any actions from you)

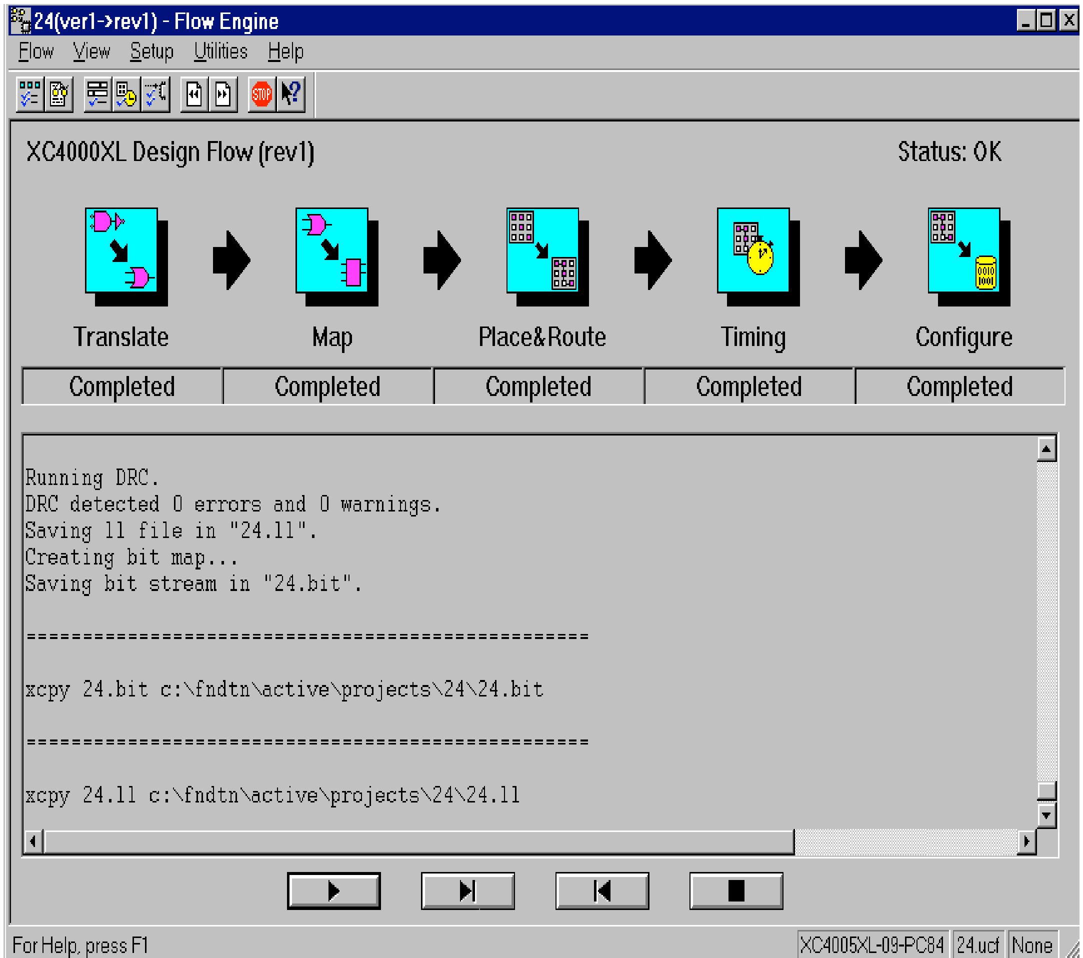


Figure 11: Flow Engine

Translate: Converts the *netlist* to an appropriate internal format.
Map: Optimizes the logic circuit.
Place&Route: The gates in the *netlist* are assigned to particular programmable switch matrices in the FPGA.
Timing: Computes the delay of the signal while it propagates through the circuit.
Configure: Stream of bits is generated which can be downloaded into the FPGA chip to configure it to carry out the logic functions described in the schematic file.

The Xilinx FPGA demo board (see Figure 14 and Figure 16) should be set up on your desk. It is connected to the parallel port of your computer by means of a Parallel Cable IV, Model DLC7. The power to the board is provided by a +5 V supply. The cable gets its power by a dongle connected to the mouse/keyboard cable at the back of the computer and also from the +5 V supply mentioned above. If the cable is properly installed, a green status light is illuminated on the connector near the demo board. The cable is attached to the demo board by flying wires that you should not have to touch.

The next step is to download the bit-stream file to the demo board and program the FPGA. Go back to the **Project Manager** window and click on the **Programming** button. Make sure the IMPACT button is pressed on the popup window and then press OK. When the IMPACT window comes up, select "Slave Serial" from the available tabs, right-click the mouse in the window and select Add Xilinx Device from the popup window. Select your "design.bit" file in the window and click Open. This should produce a screen like the one below.

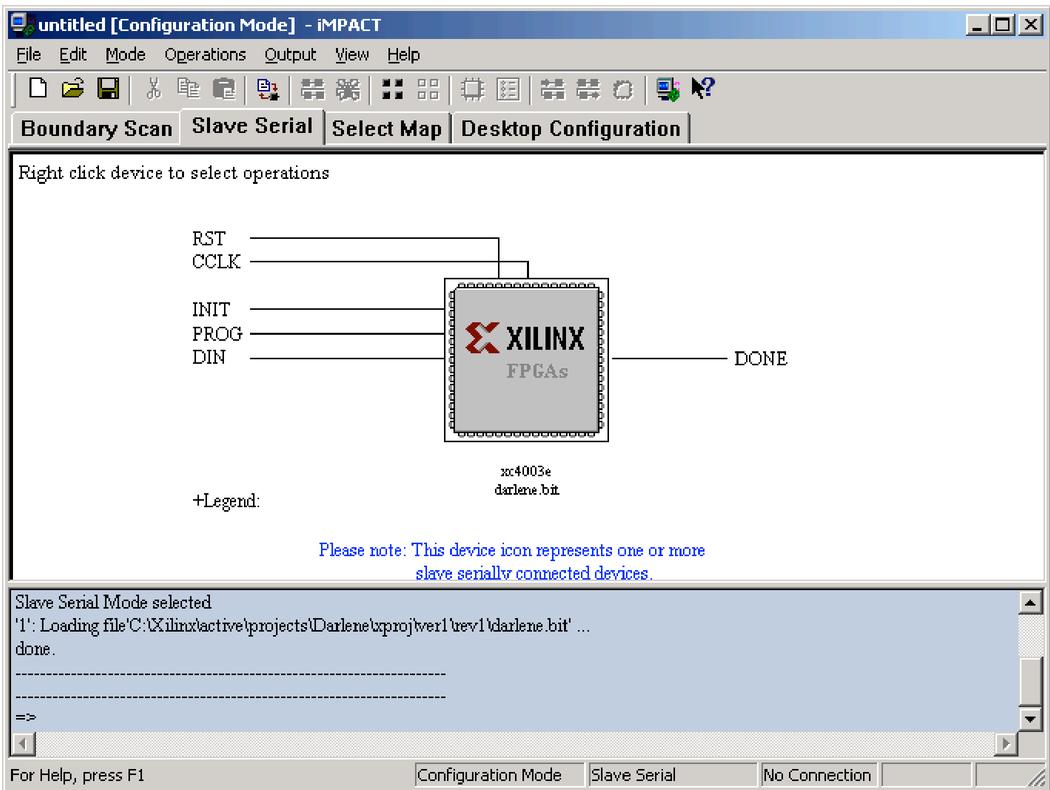


Figure 12: Impact Page

Right-click in an empty space on this window and select Cable Autoconnect. In the ensuing dialog you should see that the connection was successfully made to LPT1. Next right-click on the Xilinx icon in the window and select Program. If you are successful you should get the screen shown in Figure 13. If instead, you get a message on a red field that your programming failed, push the rightmost of three push buttons on your Xilinx Demo board to reset the chip and then repeat the programming steps. If this still fails you will need help from your instructor.

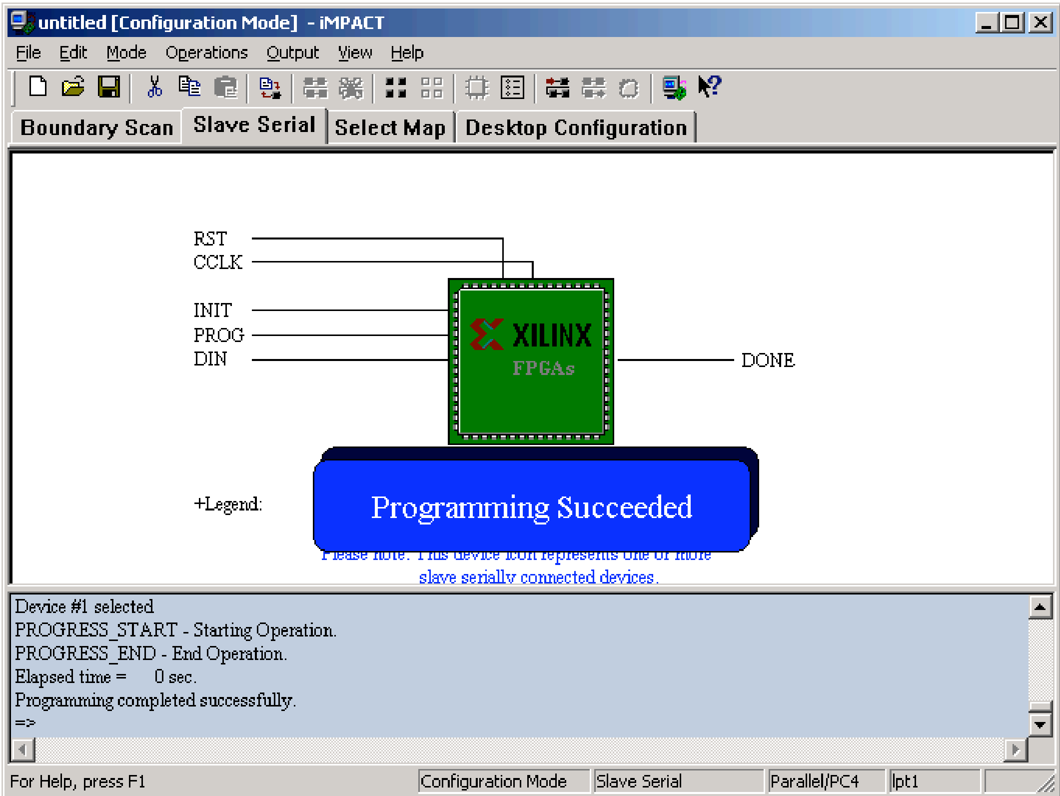


Figure 13: IMPACT page showing successful programming

Exercise the functions of your design by providing logic inputs to the FPGA you just programmed. Use the SW3 switch(es) on the board to set the associated input pin(s) of FPGA to logic “1” or “0”. Close(“1”)/open(”0”) the corresponding switch(es) and examine the response of the LED. Does your circuit function properly? Please, note that each LED segment is turned on by driving the corresponding FPGA output pin “LOW” with logic “0”.

Problem 4. Design a 3-to-8 Decoder. (Recall Experiment 5, Lab 12). Compile the logic design and download it to the demo board. Test the decoder using the7-segment LED on the board. (Hint: use the D3_8E component available in the SC library see Figure 14)

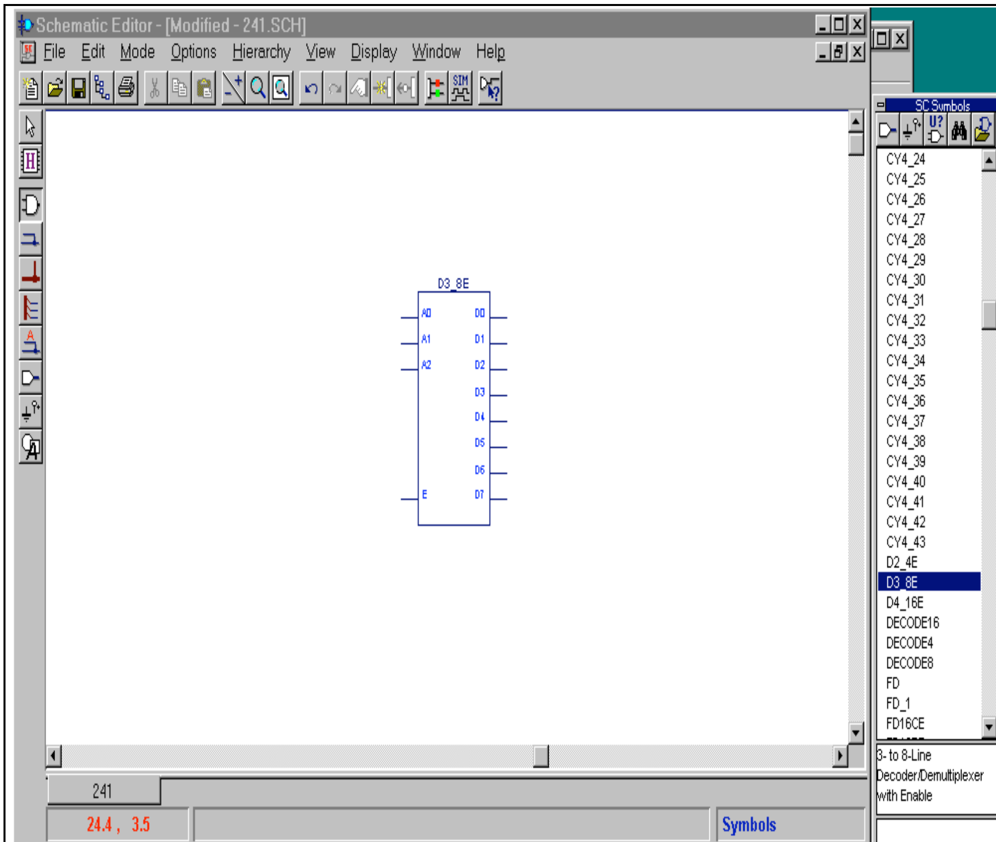


Figure 15: Three to 8 Decoder

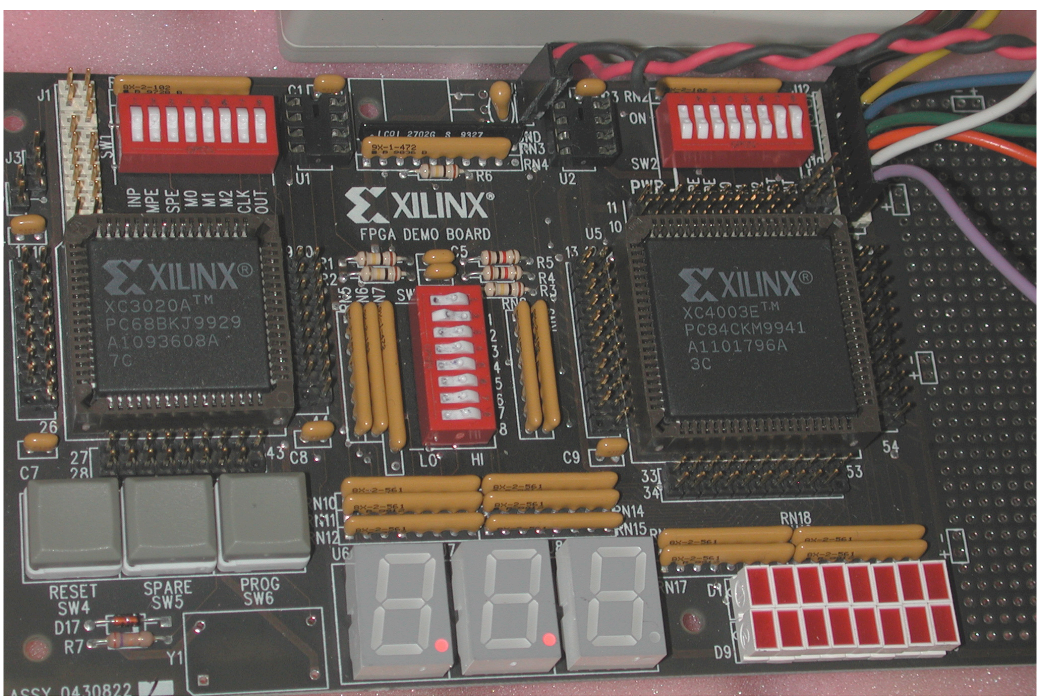


Figure 14: Xilinx FPGA Demo Board

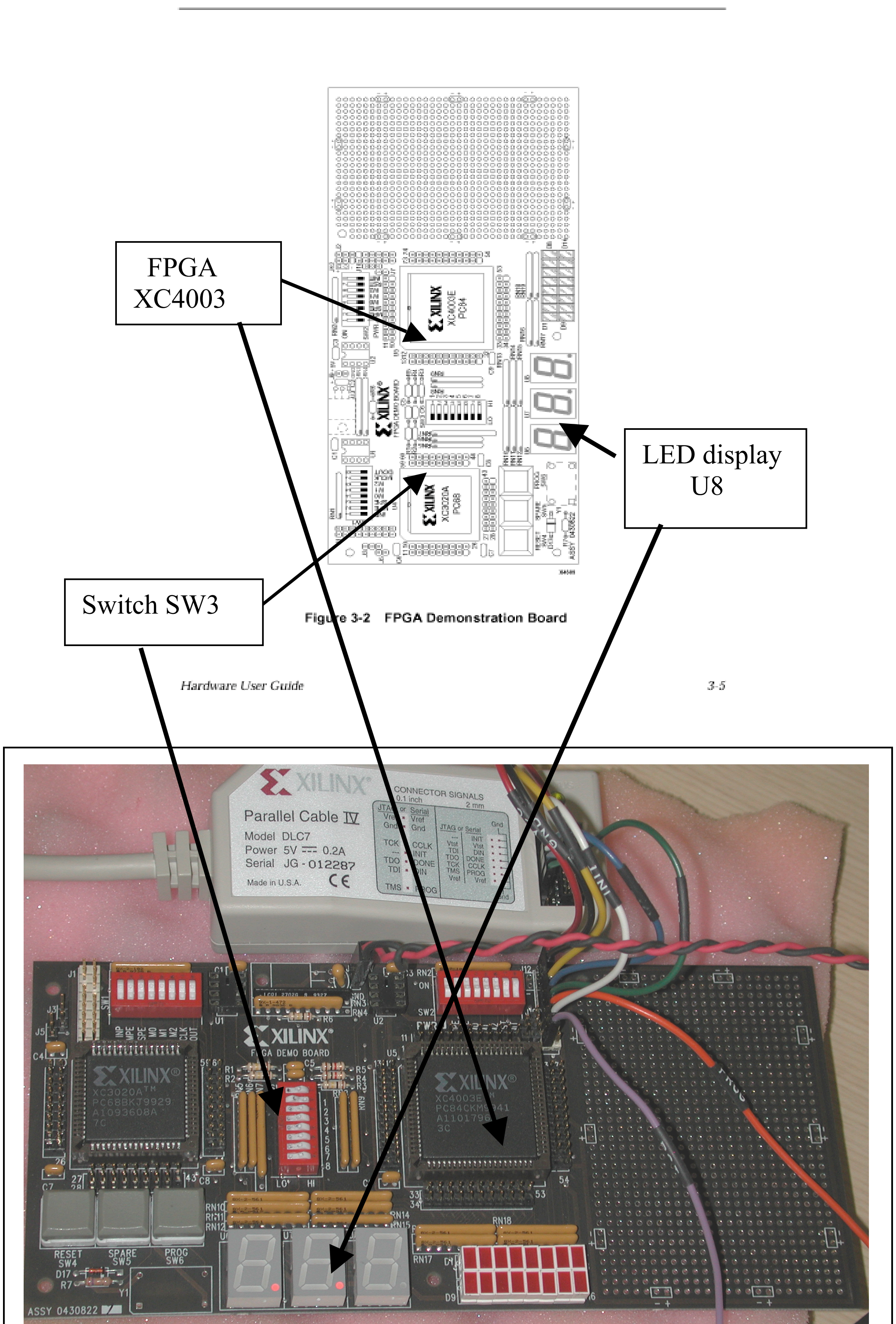


Figure 16: Xilinx FPGA Demo Board with Parallel Cable