

# XILINX FOUNDATION CPLD SOFTWARE TUTORIAL

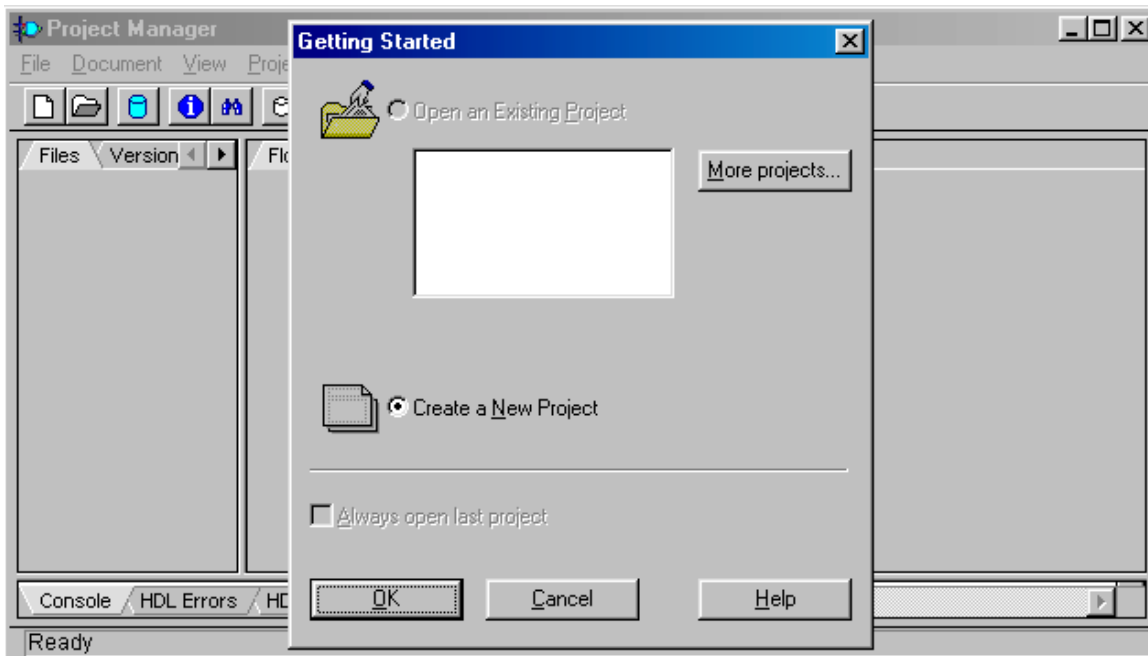
*Note: This material was developed by William Kleitz for inclusion in his textbook Digital Electronics: A Practical Approach 6<sup>th</sup> edition, (Prentice-Hall 2002).*

<http://vig.prenhall.com/catalog/academic/product/1,4096,0130896292,00.html>

*The software used for this CPLD development was downloaded from the Xilinx website ([www.xilinx.com](http://www.xilinx.com))*

We will learn the basics of CPLD design and simulation by building a solution to the Boolean equation  $X = AB(C+D)$

Start the Xilinx Foundation program. The **Getting Started** screen is shown in Figure E-19.



**Figure E- 19**

## Create a New Project

1. Check the box labeled **Create a New Project**, then press **OK**

Use the following entries in the **New Project** window:

Name: *Boolean1*

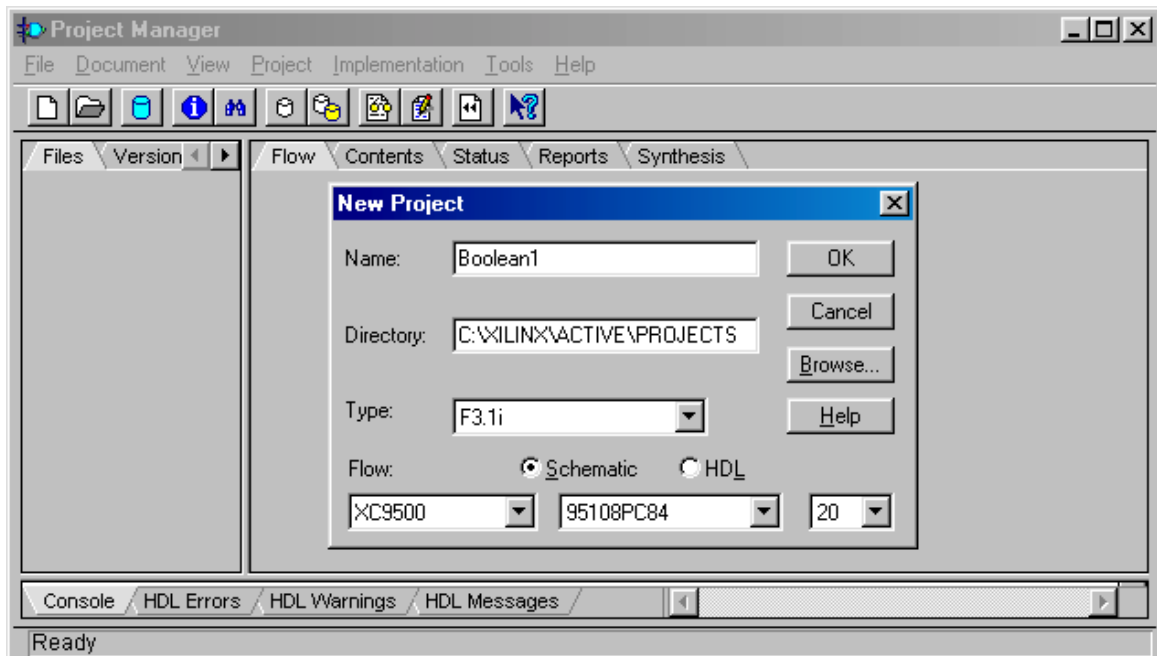
Directory: *C:\XILINX\ACTIVE\PROJECTS*

Type: *F3.1i* (Or your current software version number)

Flow: *Schematic*

Then choose the following device numbers: *XC9500, 95108PC84, 20*

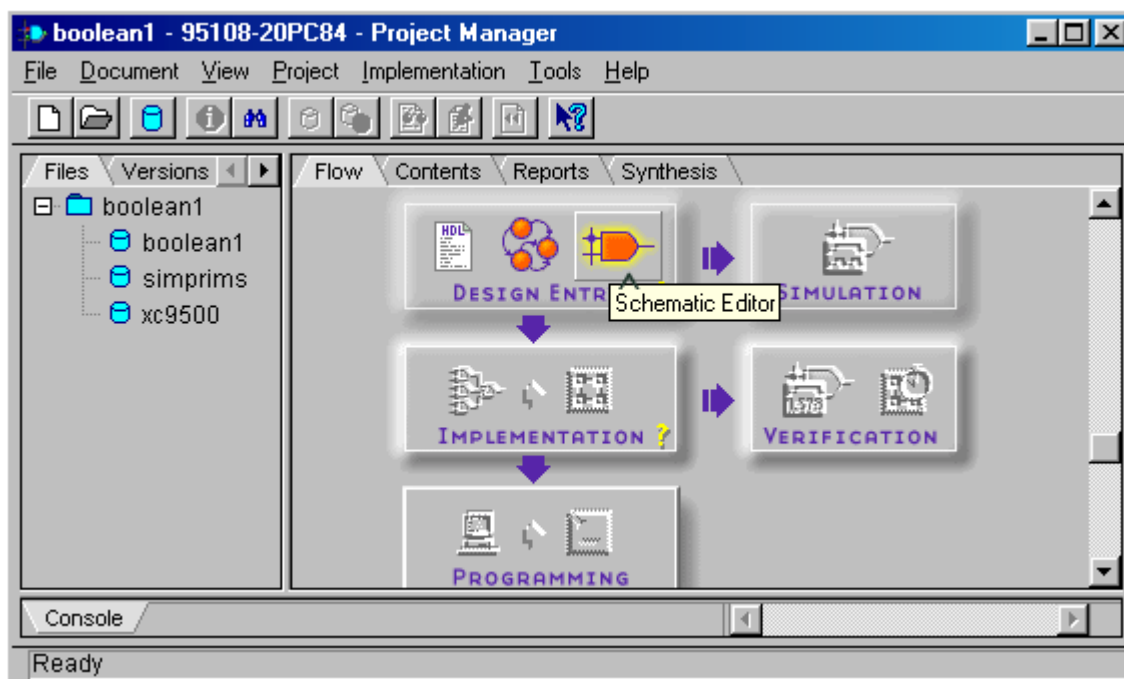
Then press **OK** (See Figure E-20)



**Figure E-20**

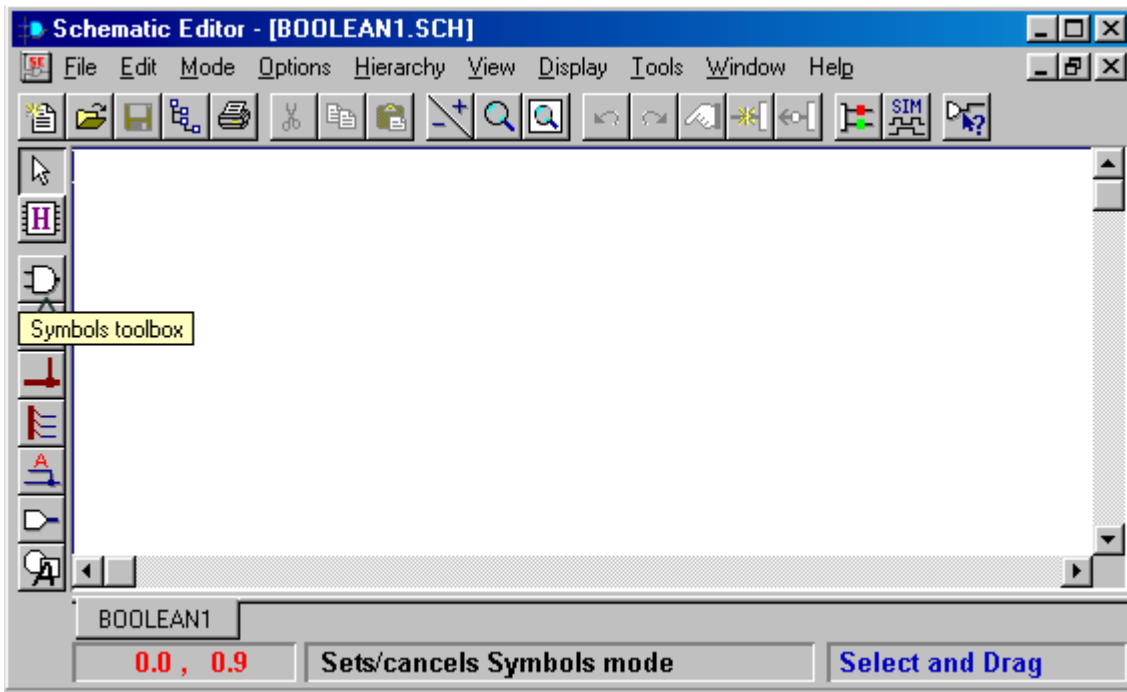
## Create a Schematic File (\*.scf)

2. To draw the logic circuit for our Boolean equation we will use the Schematic Editor to create a Schematic File (\*.scf). Start the Schematic Editor by pressing the AND gate symbol in the **Design Entry** button. (See Figure E-21)



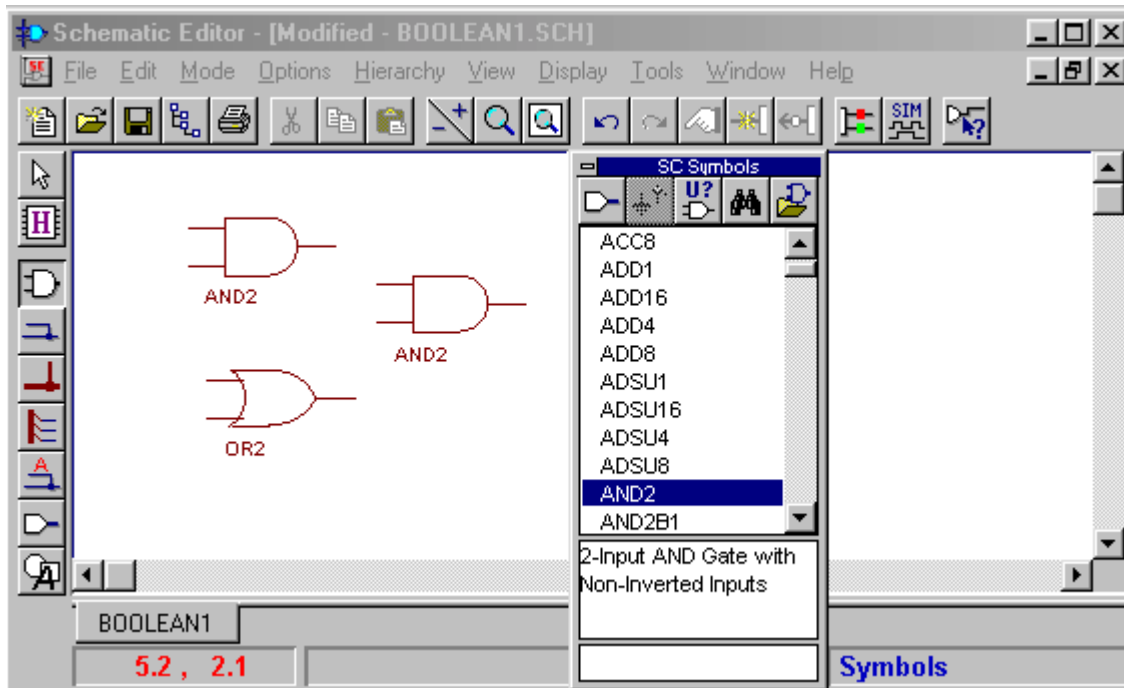
**Figure E-21**

3. The logic symbols that we need to complete our design are located in the **Symbols Toolbox**. To see the toolbox, press the AND gate symbol on the left column of the Schematic Editor. (See Figure E-22)



**Figure E-22**

4. To complete the logic circuit we will need two AND gates and one OR gate. To obtain the 2-input AND gate, click on *AND2* in the **SC Symbols** window (or type *AND2*), then move the mouse pointer to an empty area in the schematic editor workspace and drop the gate by clicking once. Repeat for the next AND gate and the OR gate. (Wires connecting these gates will be added later.) (See Figure E-23)



**Figure E-23**

5. Every input and every output to a logic circuit requires a buffer. An input buffer (IBUF) is a circuit that passes the logic level (1 or 0) from the outside world to the input of the logic gate it is supplying. The same for the output buffer (OBUF) but in the opposite direction. Add four input buffers and one output buffer to the workspace.

6. The final symbols that are required are the input and output terminals (called **Hierarchy Connectors**). These are used to describe the I/O connection points on the actual PLD IC. This connector is the symbol located in the upper-left of the **SC Symbols** window.

Click on **Hierarchy Connector** symbol and specify Name: *A* and Type: *INPUT*, then press **OK**. Drop this terminal symbol in front of the first IBUF. Repeat for the other three inputs and the one output named *X*. (See Figure E-24)

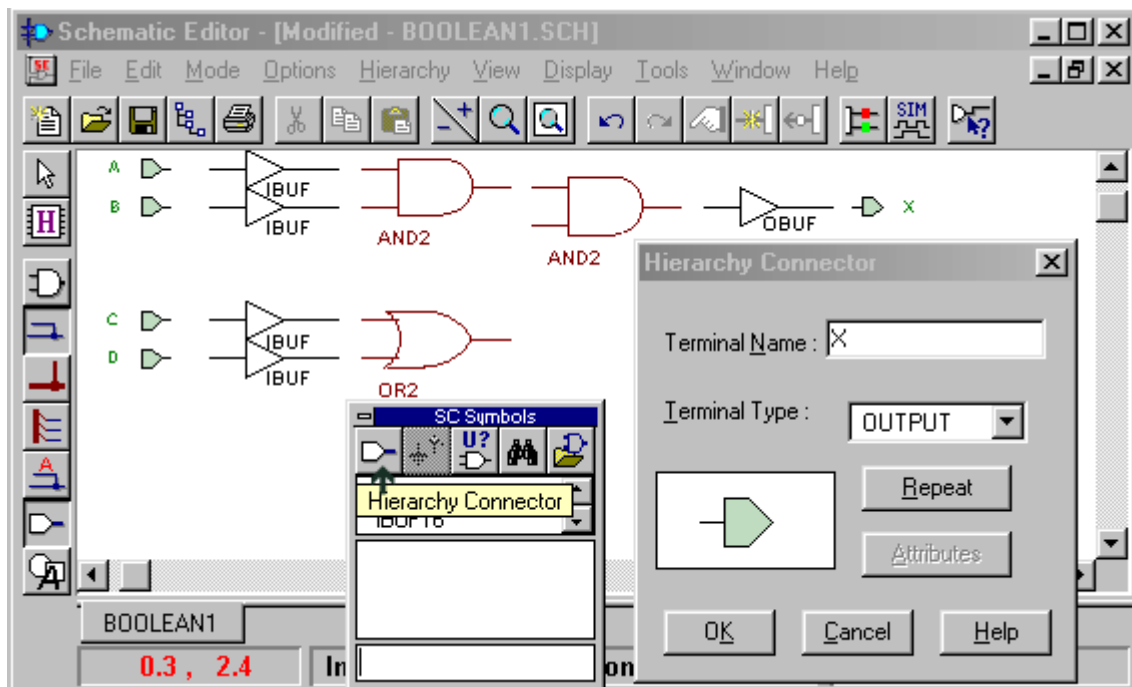


Figure E-24

7. Now you are ready to add the necessary wires to connect all of the symbols. Click on the **Draw wires** icon on the left column of the Schematic Editor.

(Or you can choose **Options** > **Draw wires**.)

8. Using the left mouse button, draw all of the connecting wires to complete the circuit. (See Figure E-25)

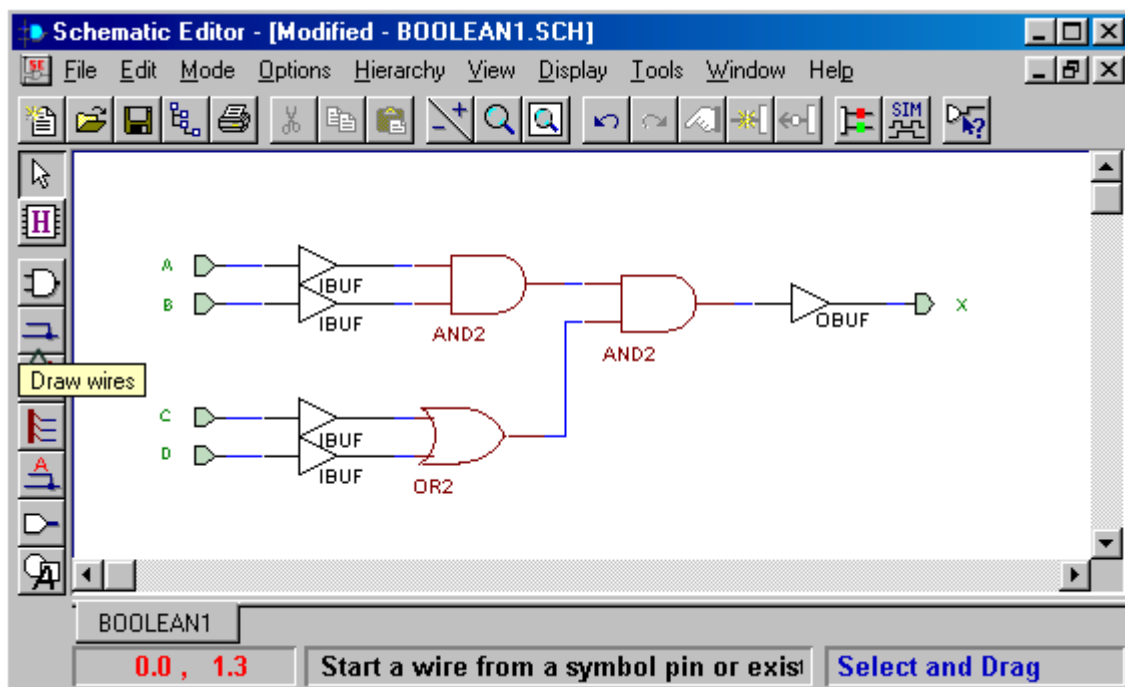


Figure E-25

## **Create a Netlist of the Logic Design**

In the next few steps we will create a file from our schematic called a Netlist file. It represents the previously designed logic connections and gates as a binary file that can be used for circuit simulation. If the simulation proves to be valid, we will then use the Netlist file to program an actual CPLD chip.

9. To create a netlist from our schematic design, in the Schematic Editor window:

Choose **Options** > **Create Netlist**

Next, to check for errors in the netlist:

Choose **Options** > **Integrity Test**

This should come back with no errors (Warnings may be issued. They usually won't affect the operation of your design, but they are worth reading anyway.)

Now that we have a valid netlist, we want to save our schematic file:

Choose **File** > **Save as** > *boolean1.sch* > **OK**

(See Figure E-26)

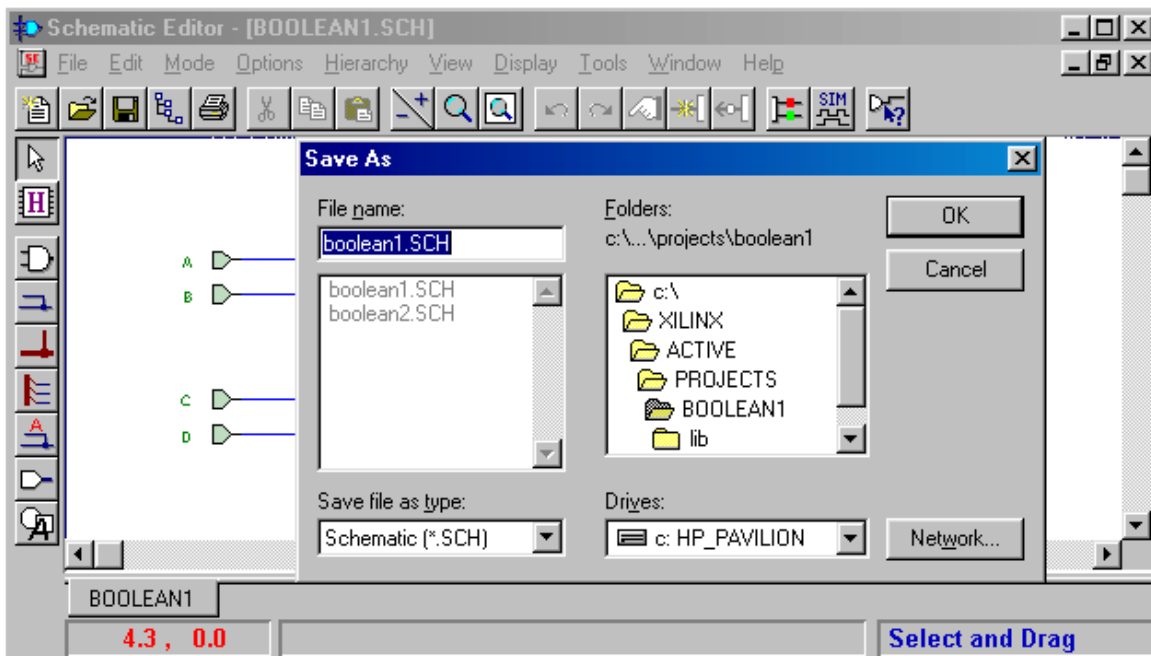
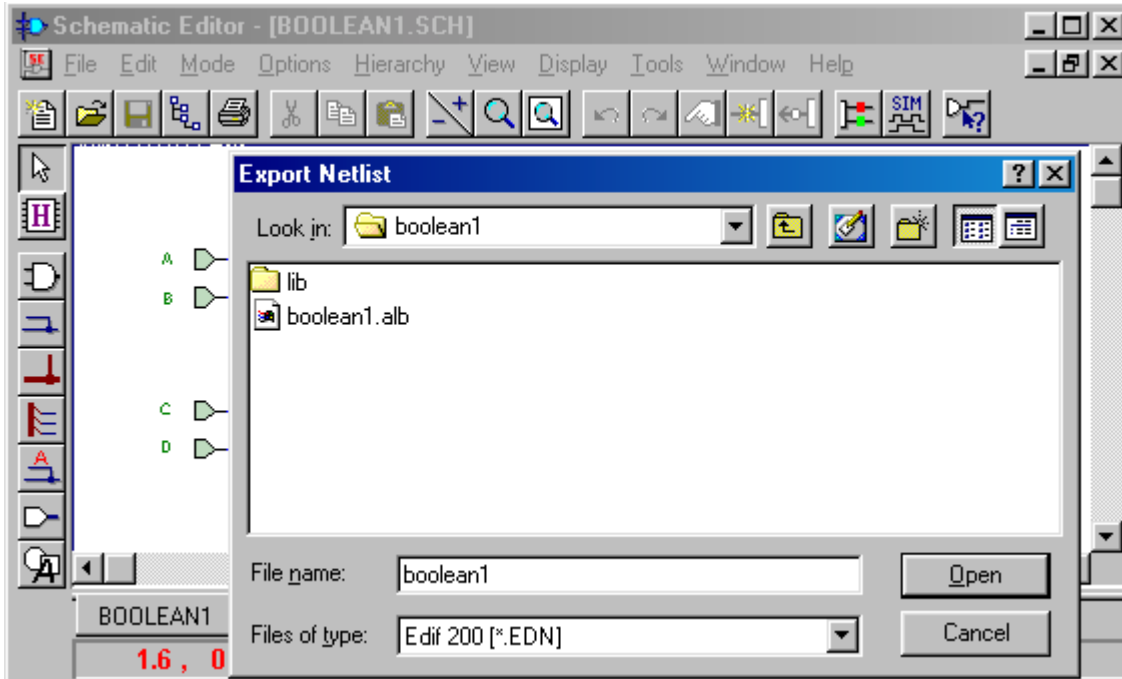


Figure E-26

10. We must now export the netlist so that it is accessible to the simulator and the device programmer. To do this:

Choose **Options** > **Export Netlist**

In the **Export Netlist** window, type the file name: *boolean1* and be sure the file type is set at: *Edif 200*, then press **Open**. (See Figure E-27)



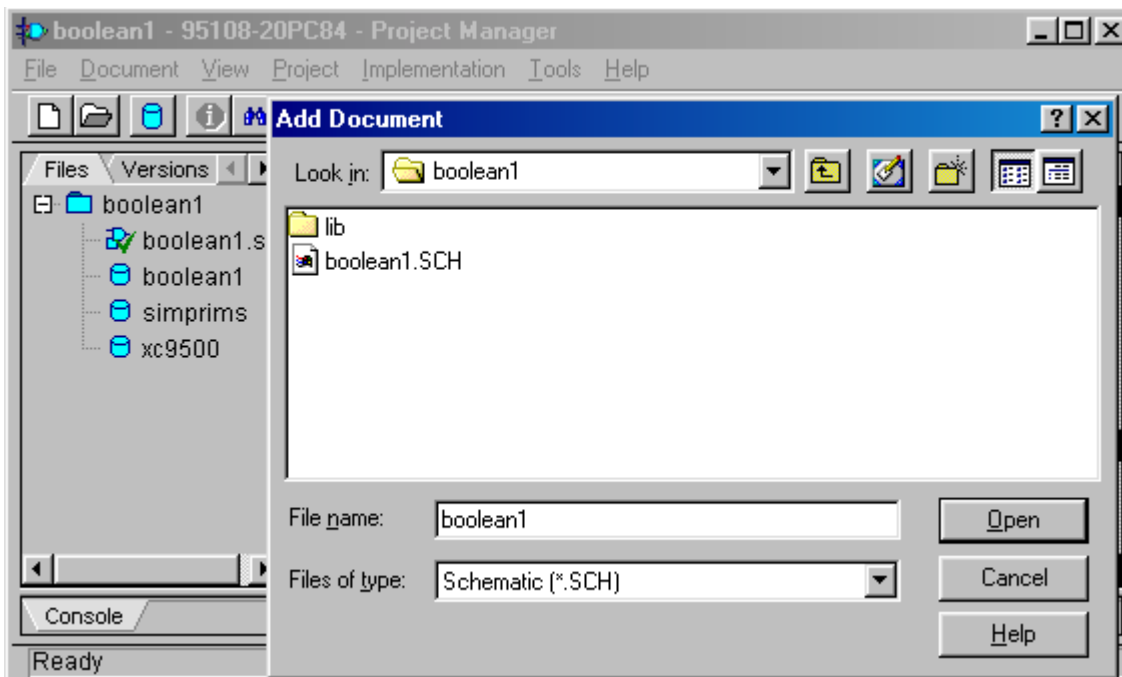
**Figure E-27**

11. We can now return to the **Project Manager** window and add the schematic to the project. In the Schematic Editor window:

Choose **File** > **Exit** Then in the Project Manager window:

Choose **Document** > **Add**

In the **Add Document** screen, type the file name: *boolean1* and be sure the file type is set at: *Schematic (\*.sch)*, then press **Open**. (See Figure E-28)



**Figure E-28**

## Project Simulation

There should be a check mark in the Design Entry box to indicate that the design is complete. Now we are ready to simulate our design. A simulation is accomplished by creating sample waveforms at the logic circuit inputs to exercise all possible combinations of input conditions. The software uses those inputs and determines the output response that will be produced by our logic design. We then look at the output waveform to be sure that it responded as we would expect for our logic design.

12. To start the simulation design process, press the **Simulation** button in the Project Manager screen. The **Logic Simulator** window will appear. (See Figure E-29)

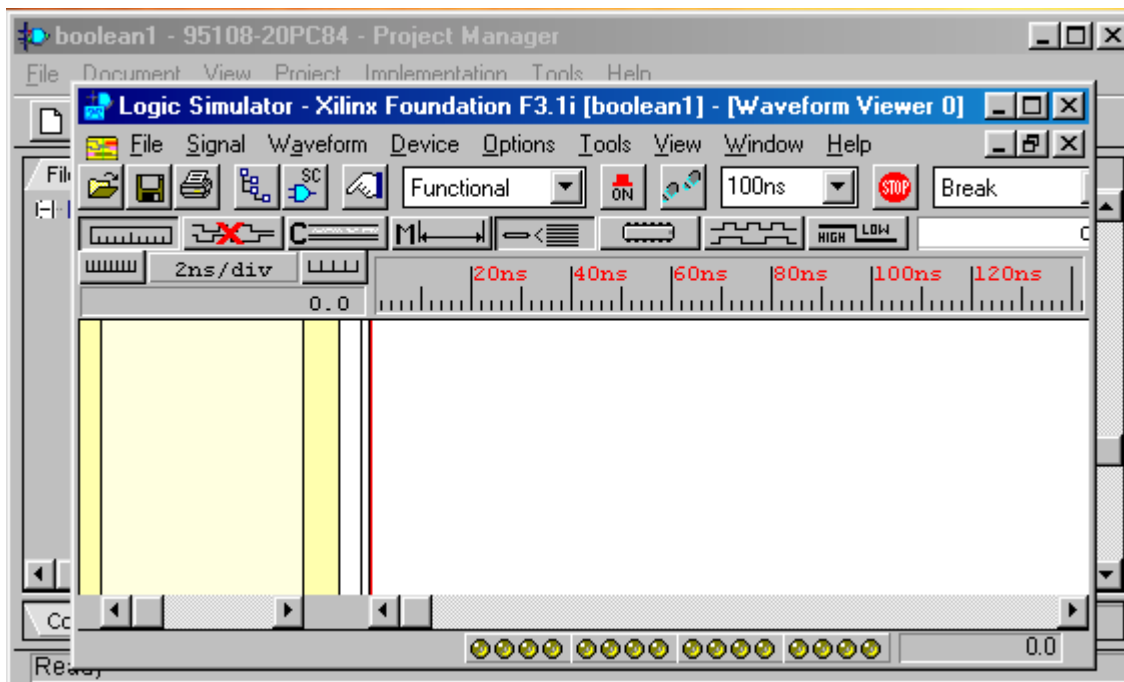


Figure E-29

13. To add the input signals to the screen:

Choose **Signal** > **Add Signals**

You will see the input and output names listed in the left column. To add input **A**:

Click on **A** then press **Add**. Repeat for the inputs (**B**, **C**, and **D**) and the output (**X**). (See

Figure E-30)

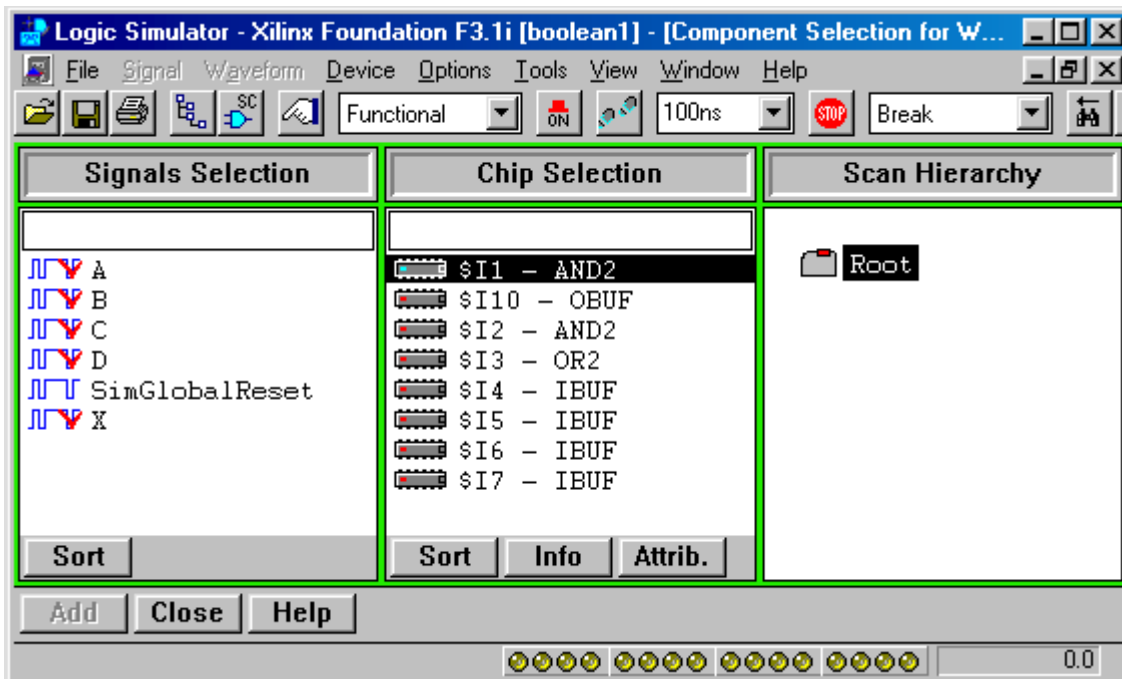


Figure E-30

14. After all signals have been added, press **Close** to return to the Logic Simulator window. Notice that all of our signal names now appear in the window. (See Figure E-31)

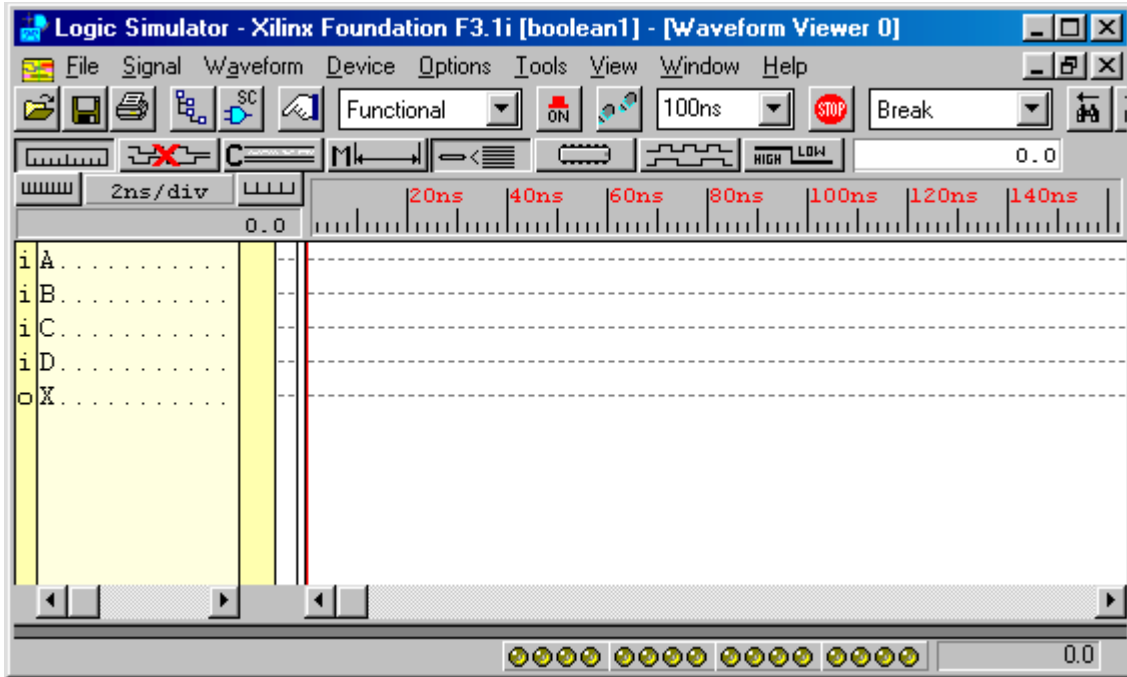


Figure E-31

15. Now we must create a waveform for each of the inputs. To insure that we provide every possible combination for the four inputs, let's draw waveforms that simulate a counter. It will count 0000, 0001, 0010, . . . up to 1111, just like a truth table with 16-combinations:

Choose **Signal** > **Add Simulators**

You will see the **Simulator Selection** window overlaid on the Logic Simulator.

To define the waveform for *A*, highlight the *A* then press the button that corresponds to **Bc:0**. Repeat for **B=Bc:1**, **C=Bc:2** and **D=Bc:3**. Notice the letter B0 now appears next to *A*; B1 appears next to *B*; and so on. (See Figure E-32)

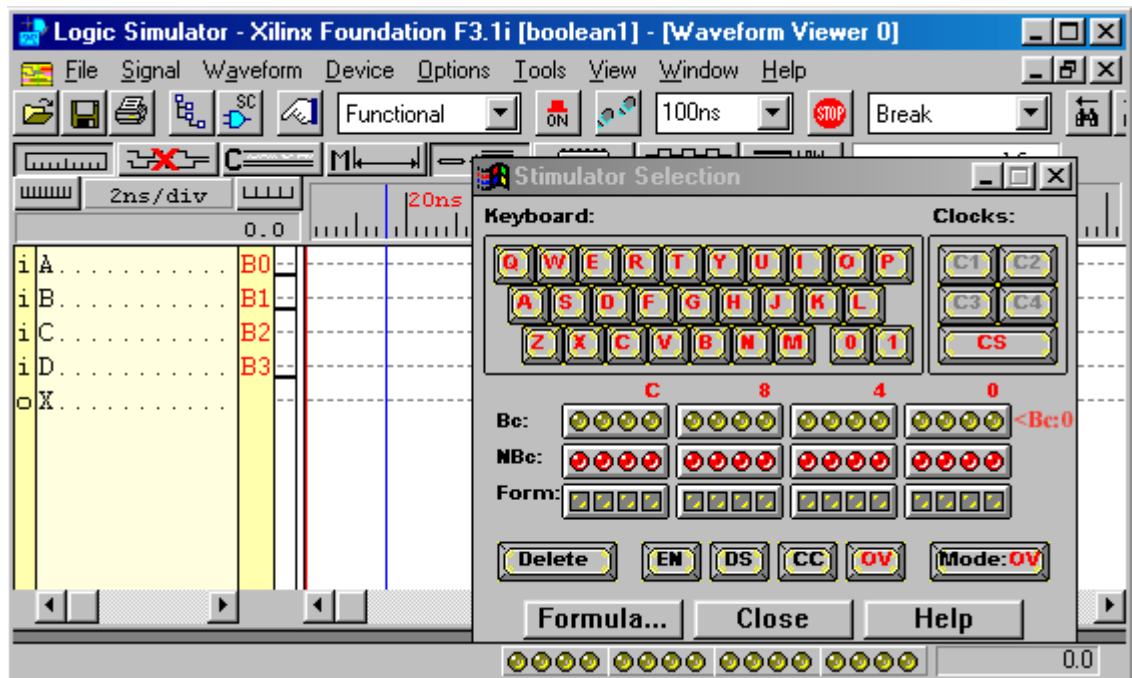
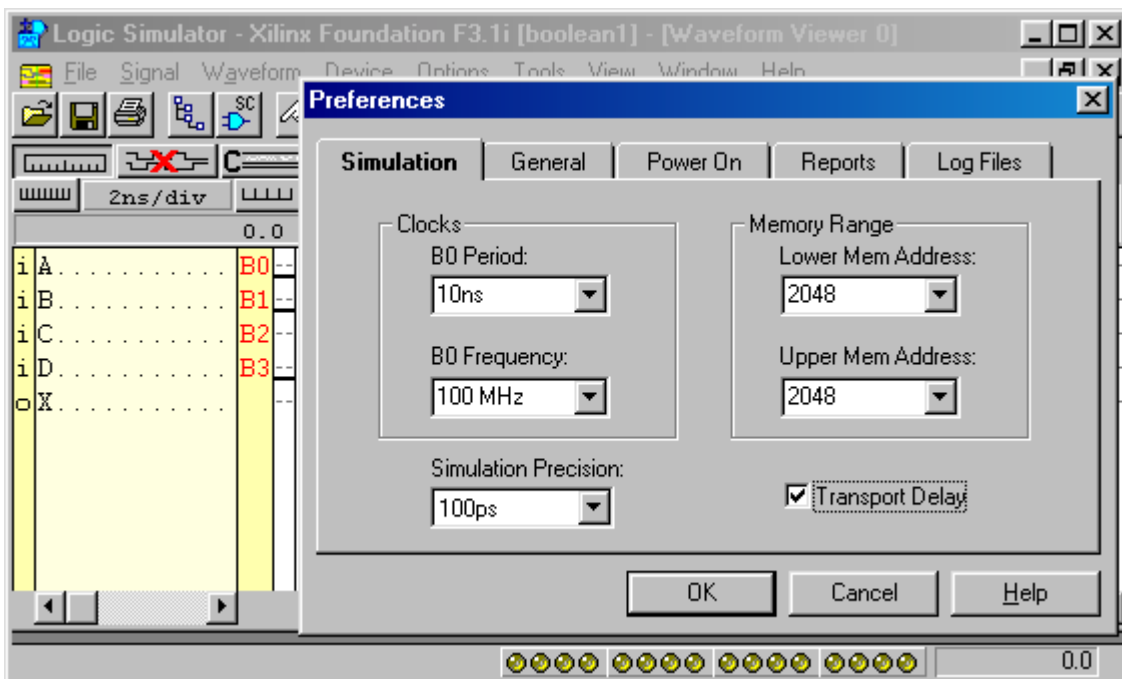


Figure E-32

16. Now that we are done defining the four inputs, press **Close** to return to the Logic Simulator window. To draw the waveforms, the following steps must be performed:

- a). Choose **Options** > **Preferences** > **B0 Frequency=100Mhz**  
> **OK**

[This makes the time period of B0 equal to 10ns. (See Figure E-33)]



**Figure E-33**

b). Back in the Logic Simulator window, choose a step size of 100ns by selecting **100ns** in the drop-down **step size** window.

[This will allow us to see 10 time periods of B0 within this one step. (See Figure E-34)]

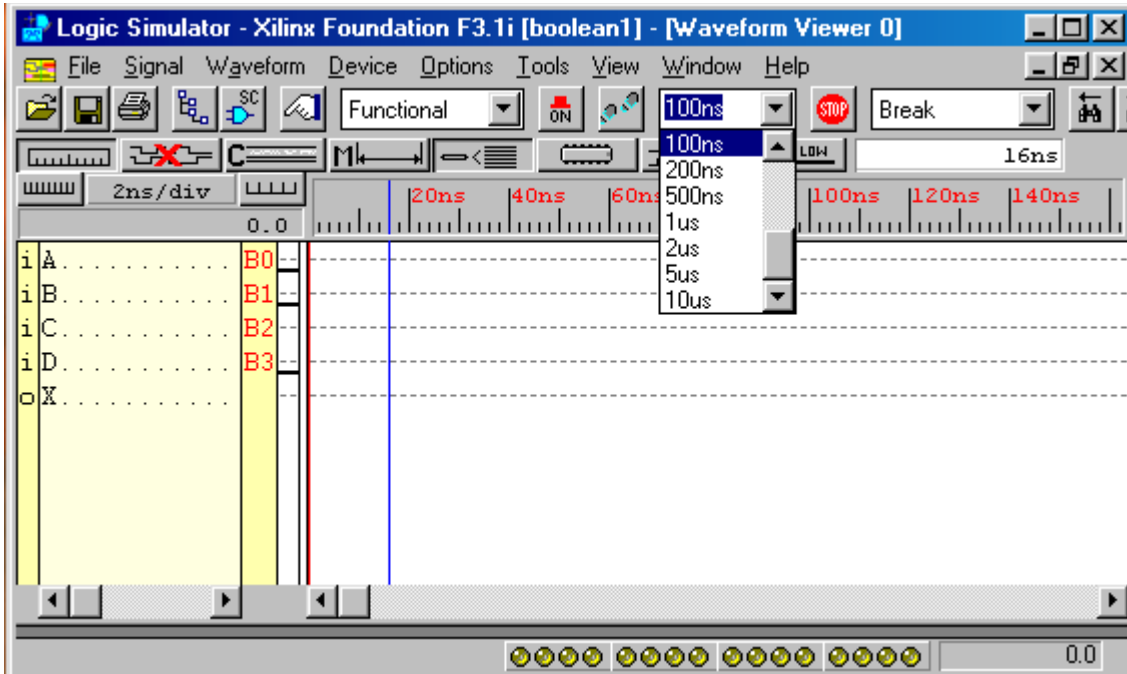
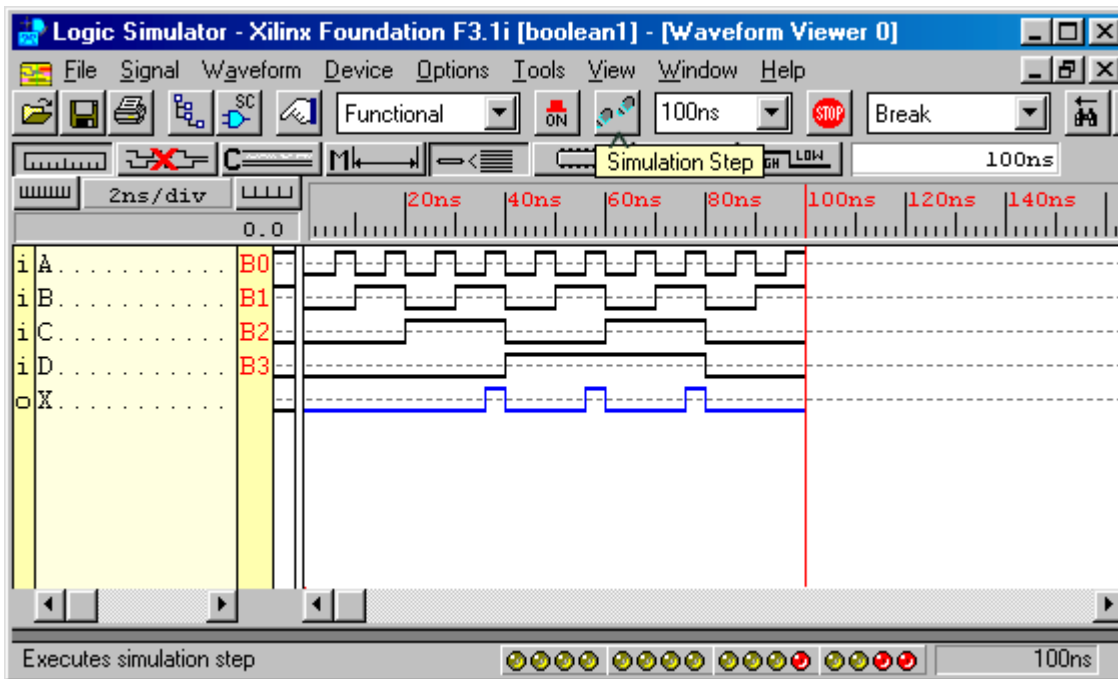


Figure E-34

- c). Press the button labeled **ON (Power On)** to reset the waveform starting point to 0ns.
- d). Press the **Simulation Step** button once to draw the waveforms from 0ns to 100ns.
- Since the time period of B0 is 10ns and one step size is 100ns, then we should see 10 periods of B0. (100ns/10ns per period = 10 periods.) [You may need to adjust the **Zoom-in** and **Zoom-out** controls (located just above the input variable names) to see the entire 100ns.]
- e). If you make a mistake and want to re-run the waveforms, Press the **Delete Waveforms** button (**X**) and repeat steps (c) and (d). (See Figure E-35)



**Figure E-35**

17. Keeping in mind that our original Boolean equation was  $X=AB(C+D)$ , prove to yourself that the circuit operates correctly by studying the resultant output at  $X$ .
18. After exiting the Logic Simulator, if you want to view the waveforms again you can either re-build the waveforms as we just did, or you can open the previous simulation file [called the Test Vector file (\*.tve)] by pressing the Simulation button in the Project Manager, then:

Choose **File** > **Load waveform** > **Boolean1.tve** > **OK**

(See Figure E-36)

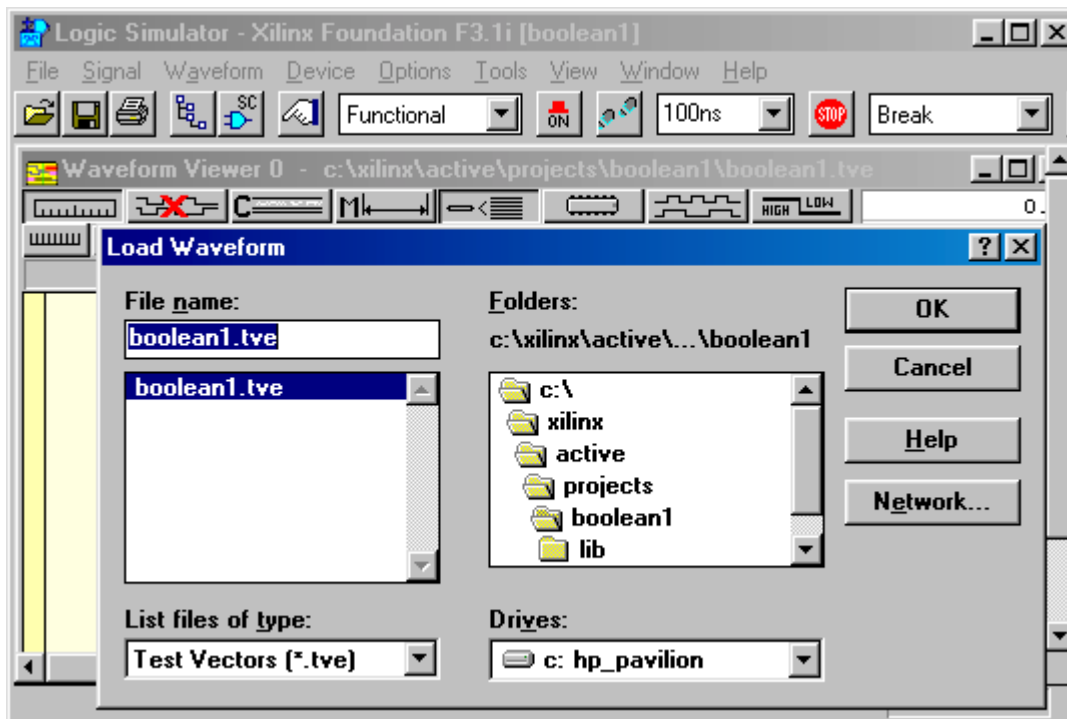


Figure E-36

## Testing the Logic Design on an Actual CPLD

Once you are satisfied that the simulation produced a valid response, you can create a Bitstream file that can be downloaded to a CPLD. The CPLD programming example shown here assumes that you are using an **XS95** CPLD target board with an **XStend** prototyping board from XESS Corporation ([WWW.XESS.COM](http://WWW.XESS.COM)). The XS95 is an educational board that can be used for testing our CPLD designs. The heart of the board is the Xilinx XC95108 CPLD. It also has on-board RAM memory and an 8051 microcontroller for more advanced operations. The XStend board extends the prototyping area of the XS95 and provides pre-wired push buttons, switches, LEDs and LED digit displays. Instruction manuals for both products are available from the XESS web site. Educational boards for Xilinx CPLDs are also available from other manufacturers. You would need to refer to their instruction manuals if using their product.

The following steps are required to implement the design and download a bitstream file to the XESS CPLD board.

19. The XStend board has eight pre-wired switches that are connected directly to specific pins on the XC95108 CPLD. The XStend manual defines the pins as shown in Table E-1. (Moving the switch to the ON position puts a LOW at the corresponding CPLD pin.)

Switch #	1	2	3	4	5	6	7	8
CPLD pin #	6	7	11	5	72	71	66	70

**Table E-1**

The XStend board also has eight active-LOW LEDs connected to specific CPLD pins as shown in Table E-2. (A LOW output to the CPLD pin turns ON the corresponding LED.)

LED #	1	2	3	4	5	6	7	8
CPLD pin #	44	43	41	40	39	37	36	35

**Table E-2**

To take advantage of these pre-wired switches and LEDs, we need to have the Foundation software lock the inputs and output to specific pin numbers on the CPLD instead of letting it choose pins on it's own.

20. Let's use switch numbers 2-3-4-5 on the XStend board to represent the A-B-C-D inputs to our logic design. Table E-1 shows that those switches are connected to pins 7-11-5-72 on the CPLD.

To define and lock the pin numbers, in the Schematic Editor window, double click on the IBUF connected to the A input. This brings up the **Symbols Properties** window for that IBUF. To lock pin 7 to that IBUF, enter the following parameters:

**Name:** *LOC*

**Description:** *P7*

Press **ADD** to add the lock, then press **Move** to close the window and allow you to move the text *LOC = P7* in front of the A input. (See Figure E-37.)

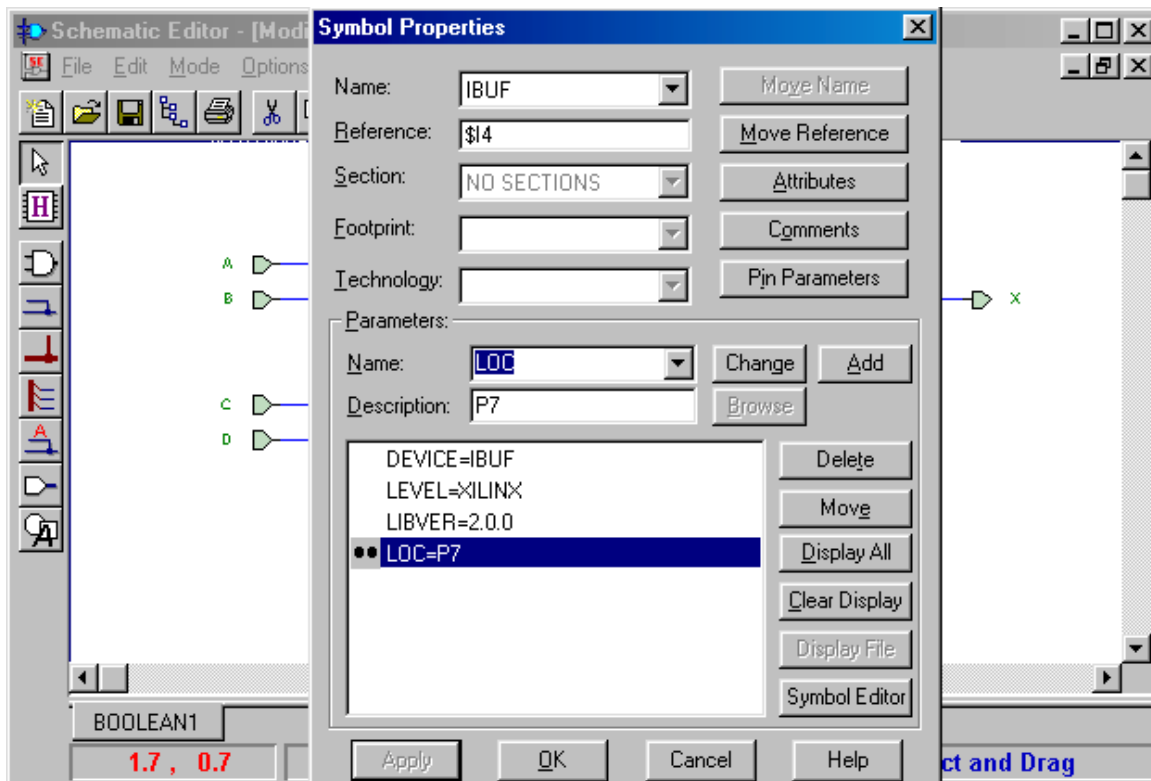


Figure E-37

21. Repeat step 20 for all four IBUFs and the single OBUF so that the schematic ends up looking like Figure E-38. (Use LED1 connected to pin 44 for the OBUF at X.)

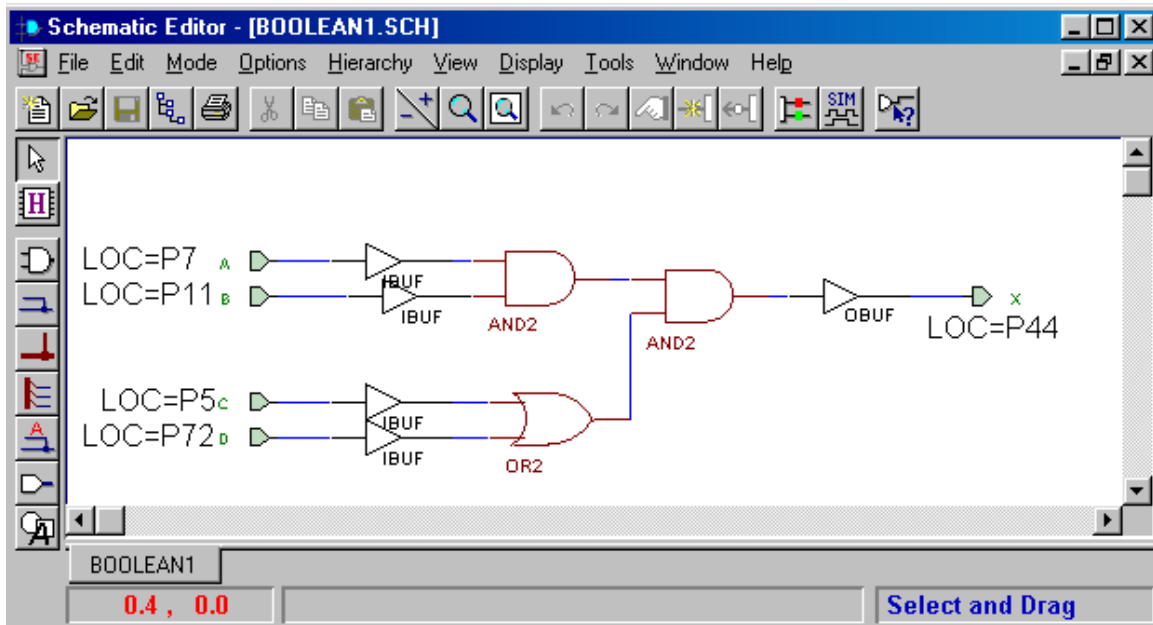


Figure E-38

22. Because we have modified the schematic, we have to re-do the steps to export a new netlist. Re-do steps 9, 10 and 11 to create and export the new netlist to be used in the programming steps to follow.

23. The next step is to implement the design and create a bitstream file to be downloaded to the CPLD. In the Project Manager window, press the **Implementation** button. In the **Implement Design** window make the following selections as shown in Figure E-39:

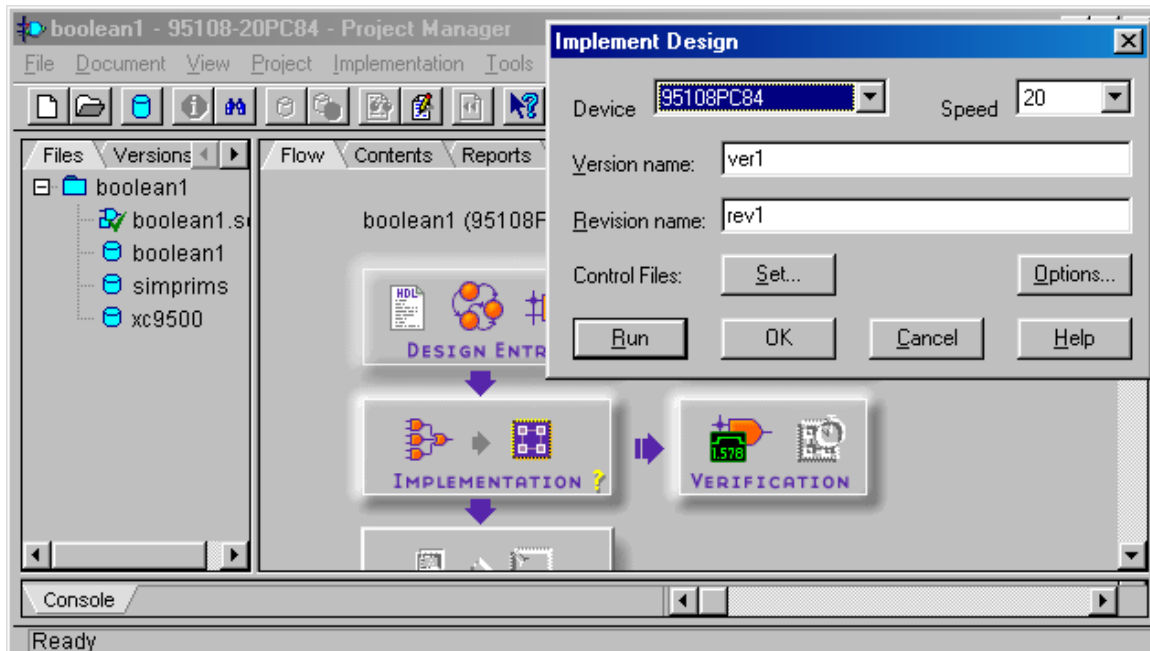
**Device:** *95108PC84*

**Speed:** *20*

**Version:** *ver1*

**Revision:** *rev1*

**Run**



**Figure E-39**

24. After the Run key is pressed, you will see the **Flow Engine** window. It will take several seconds as it completes the following steps: **Translate > Fit > Timing > Bitstream**. At this point, assuming that there were no errors, you will have a bitstream file that can be used to program the CPLD. You should see a window stating “Flow Completed Successfully”. Press **OK** to return to the Project Manager window.

25. The XC95108 CPLD on the **XS95** board will be used to test our design. Before using the **XS95** and **XStend** prototyping boards, you must follow the instructions provided in the XESS manuals for setting jumpers (See Table E-3) and connecting the boards to your PC.

Jumper #	16	17	11	8	7	4	13
	ON	OFF	OFF	ON	ON	ON	ON

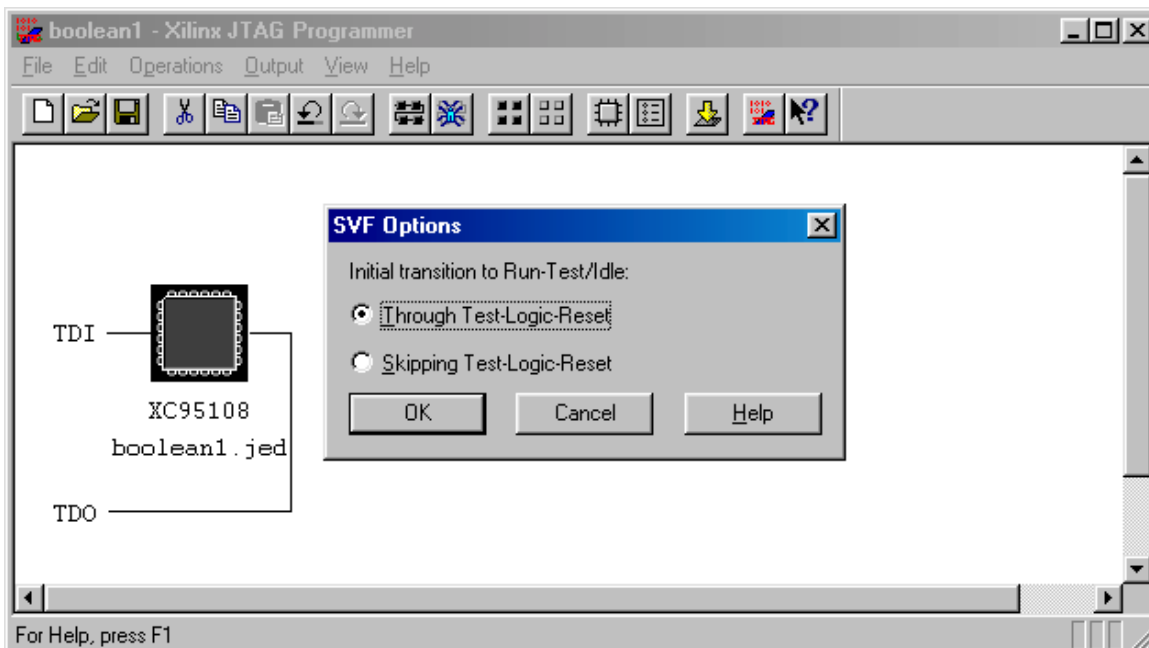
**Table E-3**

26. To start the programming process, press the **Programming** button in the Project Manager window. A window called **JTAG Programmer** will appear:

Choose **Output** > **Create SVF file**

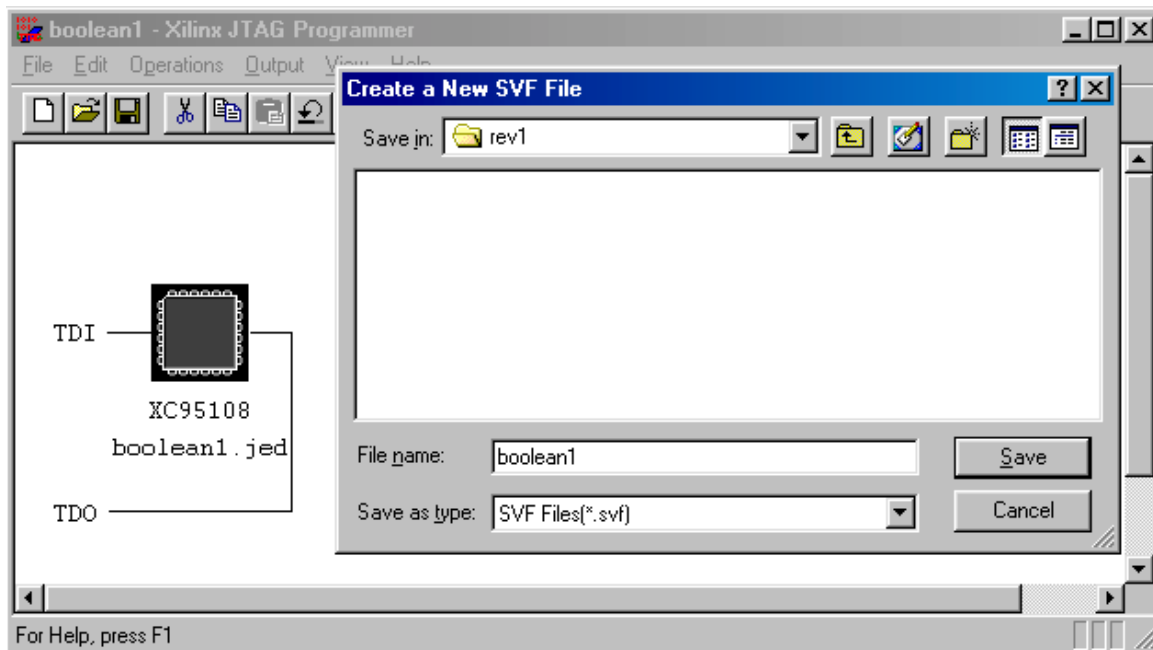
Then in the SVF Options window:

Choose **Through Test-Logic\_Reset** > **OK** (See Figure E-40)



**Figure E-40**

27. A window will appear requesting a name for your new *SVF* file. Use a **File Name** of *Boolean1* then press **Save**. (See Figure E-41.)



**Figure E-41**

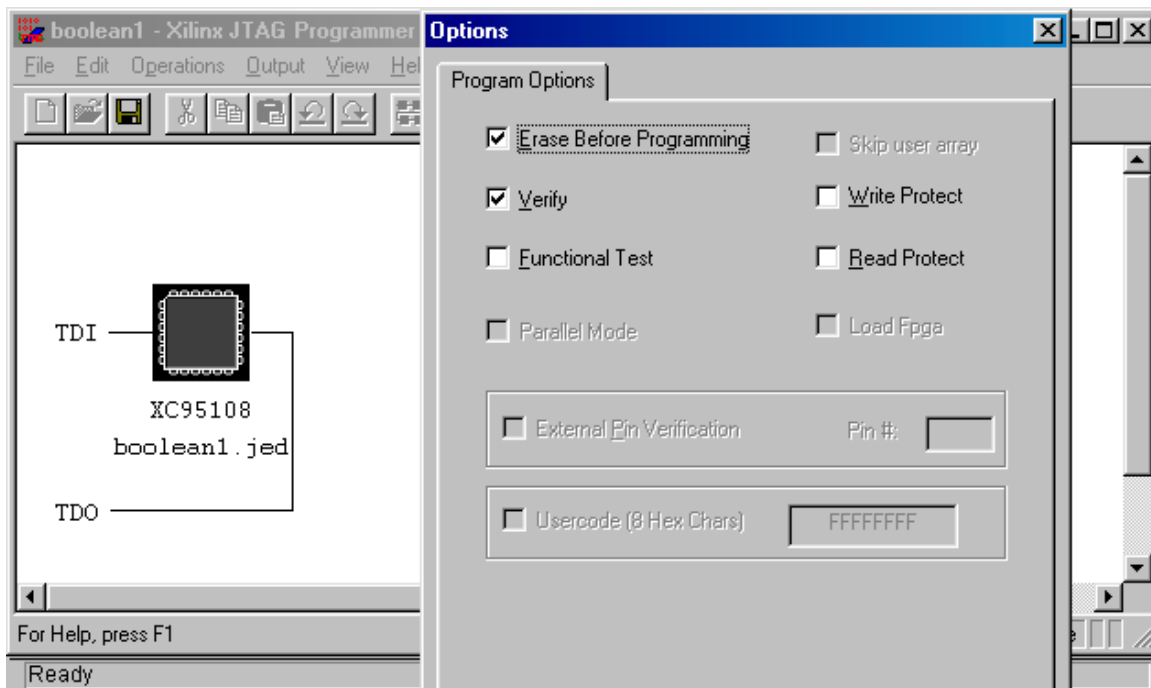
28. Next, we need to generate SVF vectors for the CPLD. In the JTAG Programmer window :

Choose **Operations** > **Program**

This will display the Options window.

Choose **Erase Before Programming** > **Verify** then press **OK**

(See Figure E-42)



**Figure E-42**

29. After the software generates SVF vectors, the **Operations Status** window shows “All Operations Completed Successfully”. Press **OK**.
30. You are now ready to download the Serial Vector Format file to the XS95 prototyping board. A software program called **XSLOAD** was provided on the XESS XS95 software CD. This program takes your file called ***Boolean1.svf*** and downloads it to the XC95108 CPLD. Run the program **XSLOAD**.

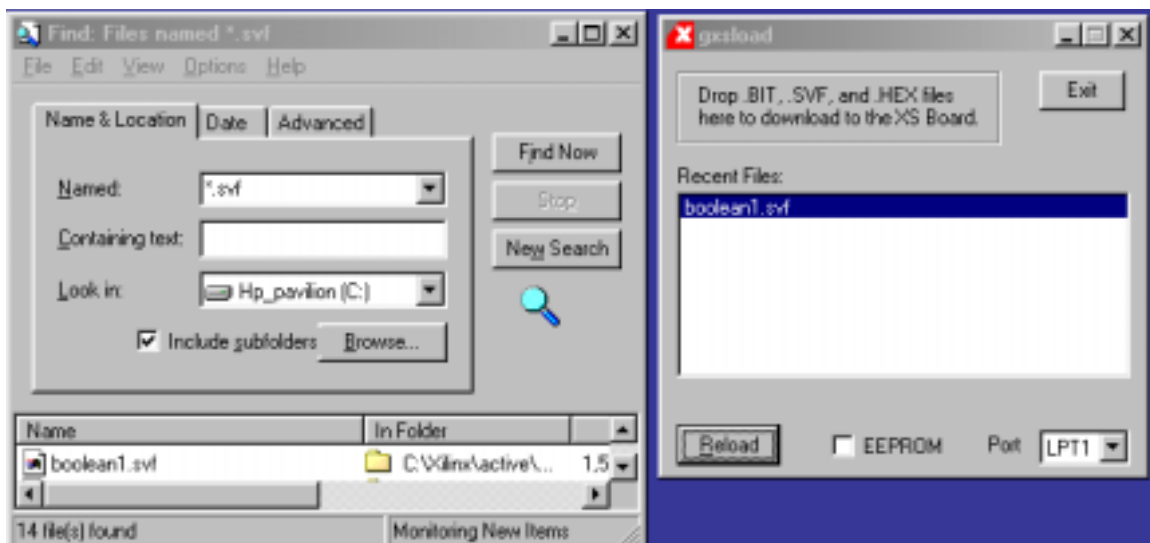
31. The XSLOAD program asks you to drop your SVF file onto it. This file is buried deep in the Xilinx subdirectory. Find it using Windows Explorer then drag and drop it into the XSLOAD program window.

*Note:* If you have trouble finding your SVF file, use the **Find** feature provided in

Windows. From the Windows toolbar:

Choose **Start** > **Find** > **Files or Folders** >  
**Named: \*.svf** > **Find Now**

This will list all files ending with the *svf* extension. Find your file in the list. Drag and drop it into the XSLOAD window. (See Figure E-43)



**Figure E-43**

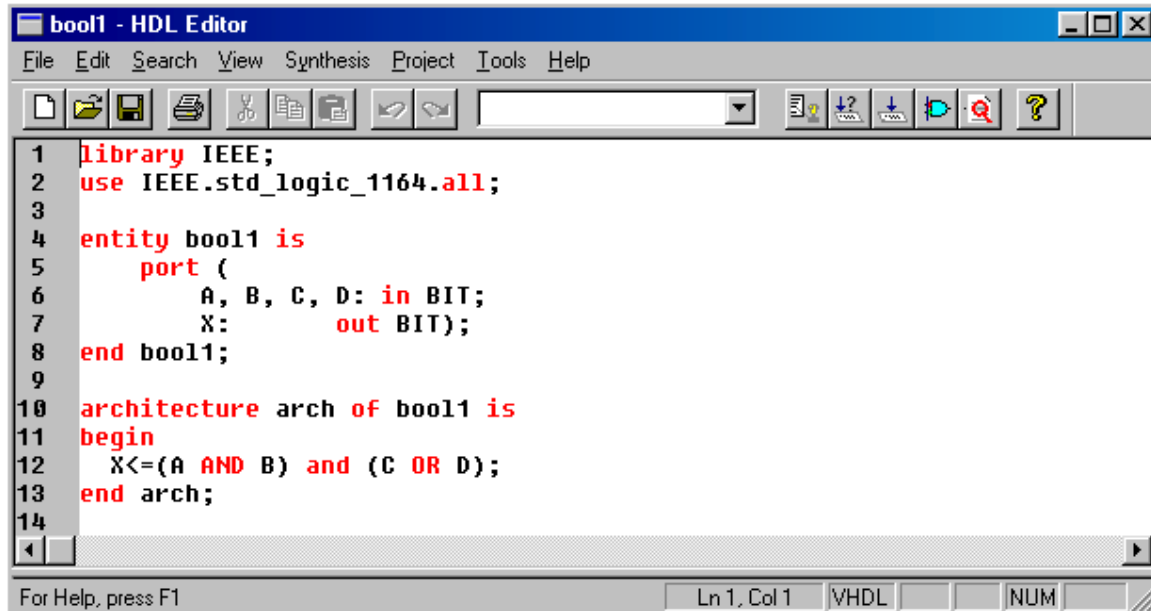
32. After downloading for several seconds, the XSLOAD program will display “The XS95 Board is programmed!!” Your program *Boolean1* is now programmed into the EEPROM section of the XC95108 CPLD. Use the switches (2-3-4-5) on the XStend board to exercise the logic to prove the equation  $X=AB(C+D)$  is indeed programmed into the CPLD. (You may want to refer to the simulator timing waveforms that we produced in Figure E-35. Step through all of those input conditions to thoroughly test the logic.) Keep in mind that the OFF position on each switch provides a logic 1, and the active-LOW LED turns ON with a logic 0 as stated in the XESS instruction manual.

## Optional VHDL Design Entry

In this section, instead of designing Boolean1 using the Schematic Editor, we will use the VHDL Text Editor.

1. Run the Xilinx Foundation program. In the Getting Started window, select **Create a New Project > OK**.
2. In the **New Project** window, use a **Name** like *Bool1* and fill in everything else like we always have except select **HDL** instead of **Schematic**. Press **OK**.
3. In the **Project Manager** window, press the **HDL** symbol on the **Design Entry** button.
4. The **HDL Editor** window now asks you if you want to use **the HDL Design Wizard** or **Create Empty**. Select **Create Empty**. Then press **OK**. (Try the Design Wizard on your next project to see if you like it better.)

5. You should now be at the top line in the HDL Editor window. This is simply a text editor that allows you to enter your VHDL statements. Enter the VHDL program shown in Figure E-44.



The screenshot shows a window titled "bool1 - HDL Editor". The menu bar includes "File", "Edit", "Search", "View", "Synthesis", "Project", "Tools", and "Help". The toolbar contains icons for file operations and editing. The main text area contains the following VHDL code:

```
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity bool1 is
5     port (
6         A, B, C, D: in BIT;
7         X:         out BIT);
8 end bool1;
9
10 architecture arch of bool1 is
11 begin
12     X<=(A AND B) and (C OR D);
13 end arch;
14
```

At the bottom of the window, there is a status bar with the text "For Help, press F1" on the left and "Ln 1, Col 1" followed by "VHDL" and "NUM" on the right.

**Figure E-44**

6. After typing the VHDL program, in the **HDL Editor** window:

Choose **Synthesis** > **Check Syntax**

If the check returns errors correct them until it returns “Check Successful”

7. Save your error-free file;

Choose **File** > **Save**

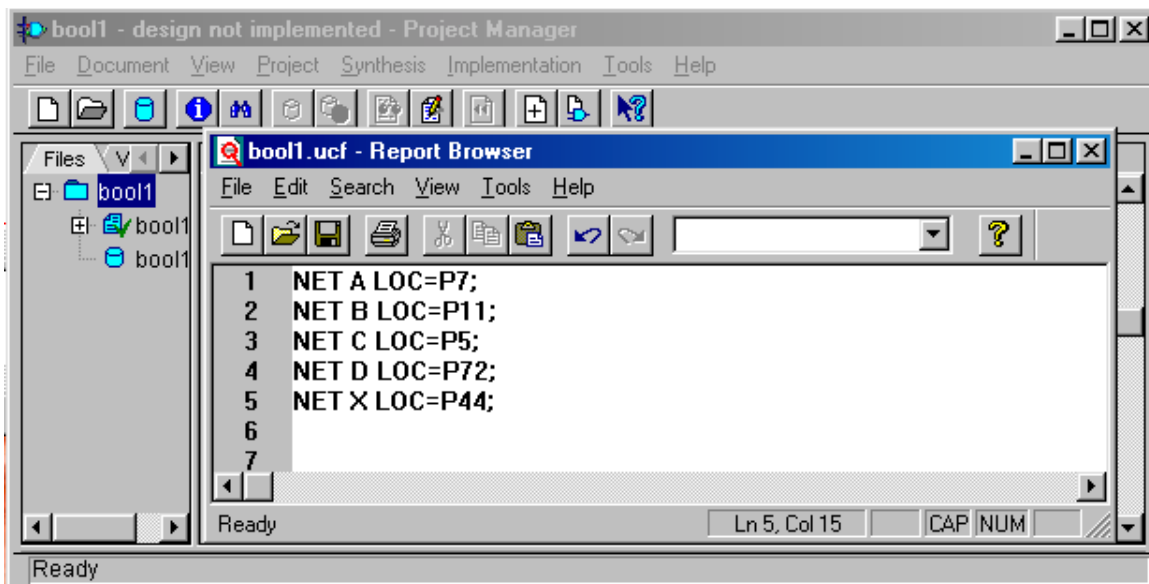
8. To add this file to your current project:

Choose **Project** > **Add To Project** Then save your file again:

Choose **File** > **Save** Then return to the Project Manager window:

Choose **File** > **Exit**

9. Assuming that we are going to eventually download this design to a CPLD chip, we need to define the I/O pins on the CPLD. We need to follow a different procedure than we followed on the schematic design. VHDL designs use a new file called a User-Constraint file (*ucf*) to define pins. In the Project Manager: Choose **View** > **Files** Then highlight the *ucf* file that you need to modify (*bool1.ucf* in this case) then press **Open**.
10. Delete all of the existing statements in the file (they are all comments) and enter the statements shown in Figure E-45. (These statements will define the I/O to use the same pins as shown back in Figure E-38.)



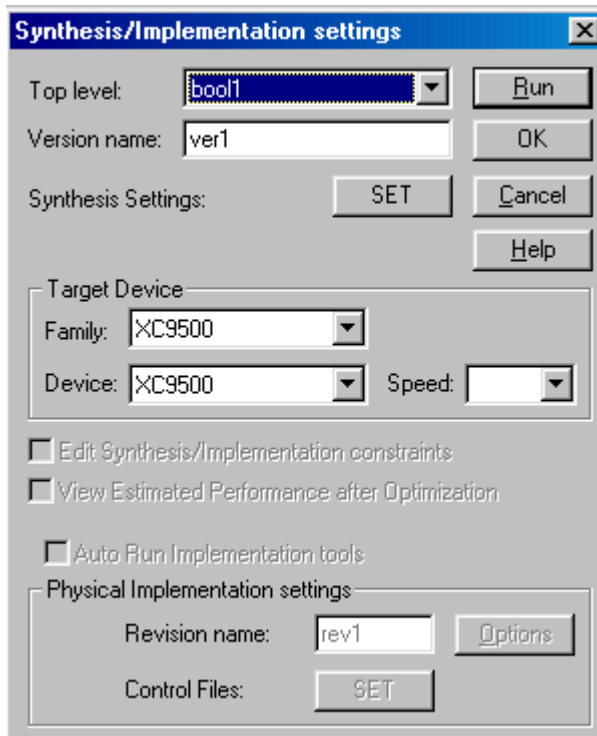
**Figure E-45**

11. Save this file and return to the Project Manager:

Choose **File** > **Save** > **File** > **Exit**

12. As you can see in the flow window, in order to perform a **Simulation** or an **Implementation**, we need to first perform a **Synthesis** of our design:

Press the **Synthesis** button and fill in the settings as shown in Figure E-46.



**Figure E-46**

13. When the synthesis is complete you will see a check mark in both the **Design Entry** button and in the **Synthesis** button. From here you should perform a **Simulation** of the circuit using the same steps outlined previously to be sure the design acts like it should. Finally, you can execute the **Implementation** and **programming** steps that were outlined previously to program the actual chip on the XESS board.

*Note:* VHDL can be a very powerful design tool. The example shown above was a simple implementation of a Boolean equation. Typically, VHDL is used as the primary design tool for all of the more complex logic found in modern digital designs.

An extremely powerful feature of this HDL Editor is that it is capable of writing your VHDL program for you. Program modules such as comparators, flip-flops, counters and shift registers can be automatically generated by the software. To see an example of this, in the HDL Editor window:

Choose **Tools > Language Assistant** Double click **Synthesis templates** Then click on any of the templates listed to see the corresponding VHDL code for that function. Press the **Use** button to insert it into your program.