



Xilinx WebPack and the Digilent D1le-DIO1

Dennis Silage, PhD
silage@temple.edu

Introduction

Xilinx WebPack is the computer-aided design (CAD) environment, which will compile and synthesize a Verilog behavioral description for a specific target field programmable gate array (FPGA) architecture. Xilinx WebPack is freeware and can be downloaded to your personal computer (www.xilinx.com). Xilinx WebPack is a Verilog compiler that synthesizes the appropriate *fuse-map* for the FPGA so that the device can be programmed and generate the desired function.

The target Xilinx FPGA, a Spartan XC2S200E, resides on a Digilent D1le hardware board, as shown below. The D11e board has an indicator light emitting diode (LED) and a 50 MHz master clock oscillator (OSC), as shown below. The process to be designed will also use the Digilent DIO1 input-output (IO) hardware board, which has 8 slide switches (SW), 5 pushbuttons (BTN), 4 numerical displays (seven-segment), and 8 light emitting diodes (LED), as shown in Figure 1 below.

Power Up

The D1le and DIO1 are mated at the C and D connector of the D1le. The programming cable is placed on the JTAG port with the clearly marked pin labeled VCC at the interior of the D1le, as shown in Figure 1. The other end of the programming cable is inserted into the parallel port of the Laboratory PC. Finally, a 3.3 VDC power supply plug is inserted at the matching jack (PWR) and the power LED (PWR LED) will indicate that the hardware boards are ready. If any of this does not seem correct to you, ask the Laboratory Assistant to verify the DIO1-D1le hardware setup.

Project Header File

The actual implementation of a Verilog behavioral compilation and synthesis design utilizing Xilinx WebPack and the Digilent D1le Spartan FPGA hardware board is quite intricate. A complete description of that lengthy process is available as the *Introduction to Xilinx WebPack*. Here we will simplify the process by providing a *Project Header* that contains the requisite structure for the design. The student merely has to enter the Verilog variables and source code for the

design, set the *User Constraints File* (UCF) for the IO ports in use, compile and synthesize the Verilog code, and download a programming *fuse-map* file to the FPGA.

Note that the Xilinx WebPack environment will not be used to simulate the design. Silos by Simucad has been used in the Laboratory to perform that useful function. Here we will observe the actual process on the IO ports, rather than by the simulated waveforms.

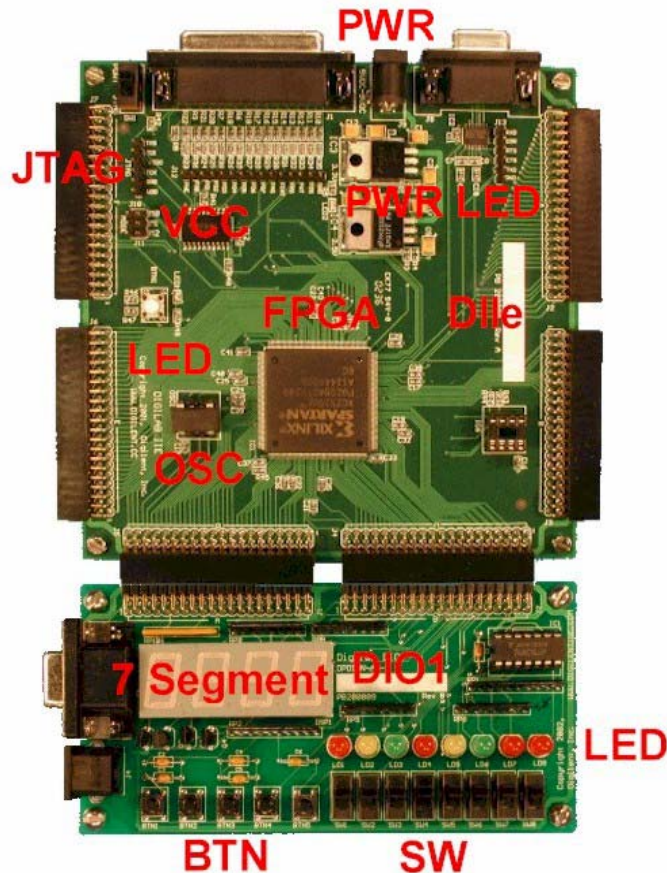


Figure 1

Project Entry

When the Xilinx WebPack *Project Navigator* is started, the design environment screen is opened, as shown in Figure 2. The *Project Header* file may already be available from prior use in the Laboratory, as shown below. If not, click on *File...New Project* and search for the project *EE157Lab* on the *PC Desktop* in the *EE157* folder.

Double-click on the file *EE157LabSource.v* in the *Sources in Project* window at the left of the screen. The *Project Header* Verilog source code appears in the edit window on the right of the screen, as shown in Figure 3.

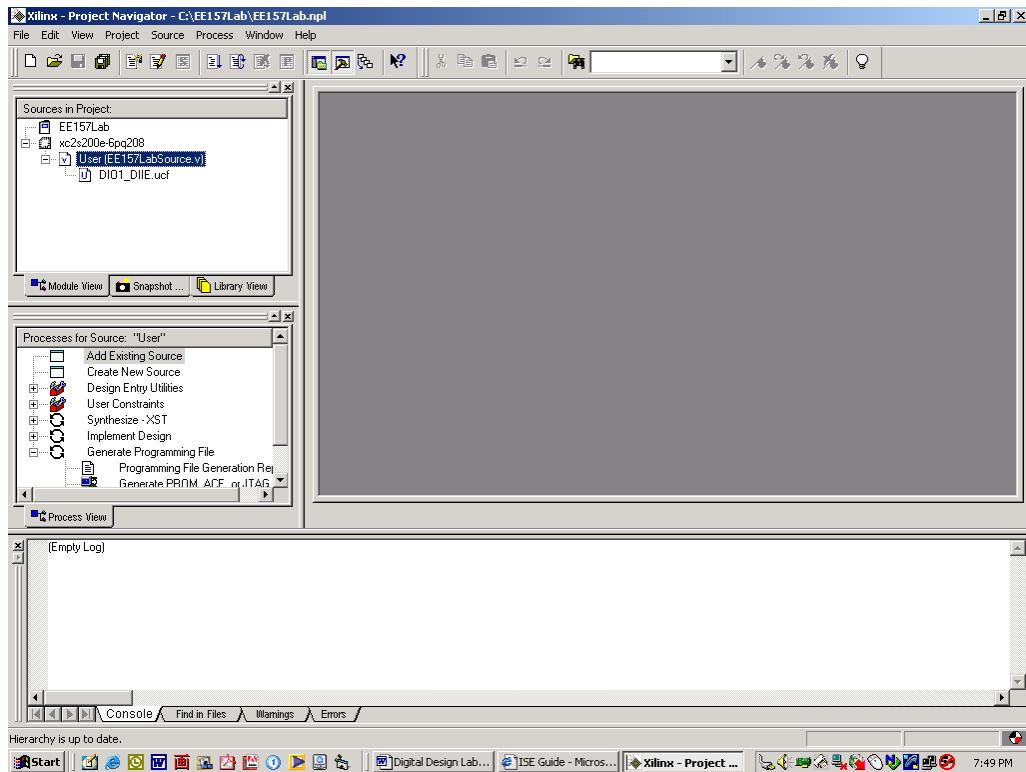


Figure 2

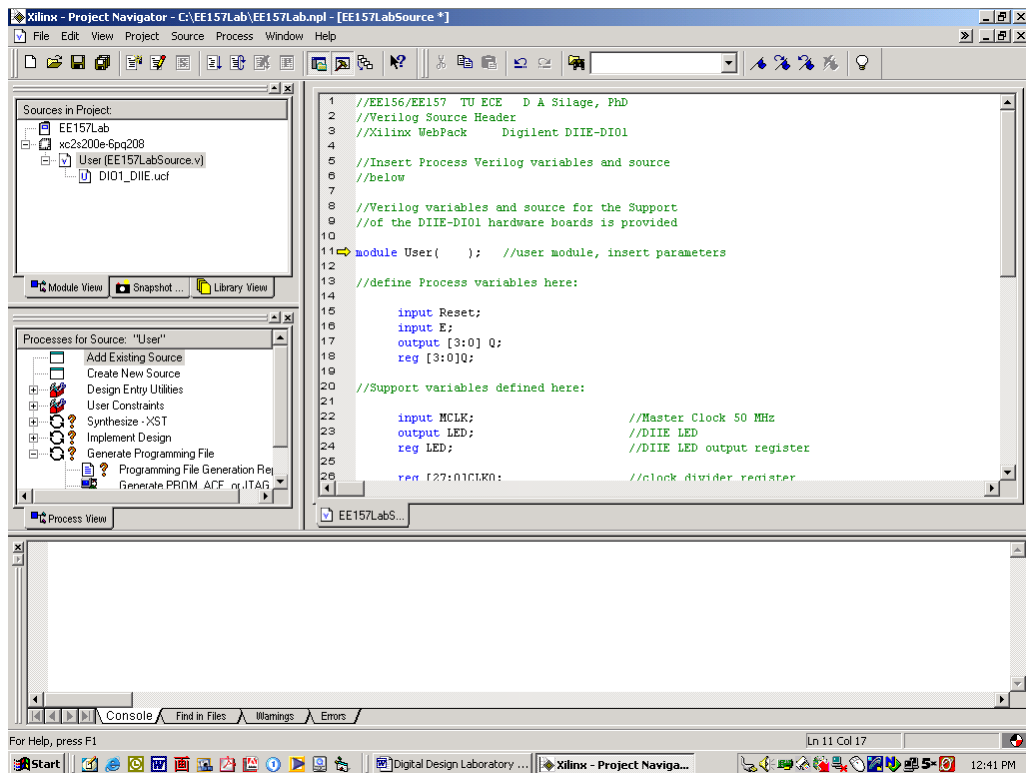


Figure 3

The complete *Project Header* Verilog source code is listed below. The Verilog module parameters, variables and source are inputted at the sections indicated. To illustrate the technique, Figure 7-56 on page 403 of your text, a four-bit up counter, will be compiled and synthesized.

The steps in the project entry of the design are as follows:

1. Enter your Verilog variables and source code in the appropriate areas of the *Project Header Source Code*, *EE157LabSource.v*. The variables and source code for the example of Figure 7-56 are listed below.
2. Enter the module parameters in the *Project Header Source Code*. The input and output parameters here include not only those of the your source code (Reset, E, and Q) but also those of the support code of the *Project Header* (MCLK and LED). The module parameters for the example of Figure 7-56 are listed below.

The completed Verilog source code for module *User* in the *Project Header* is shown in Figure 4.

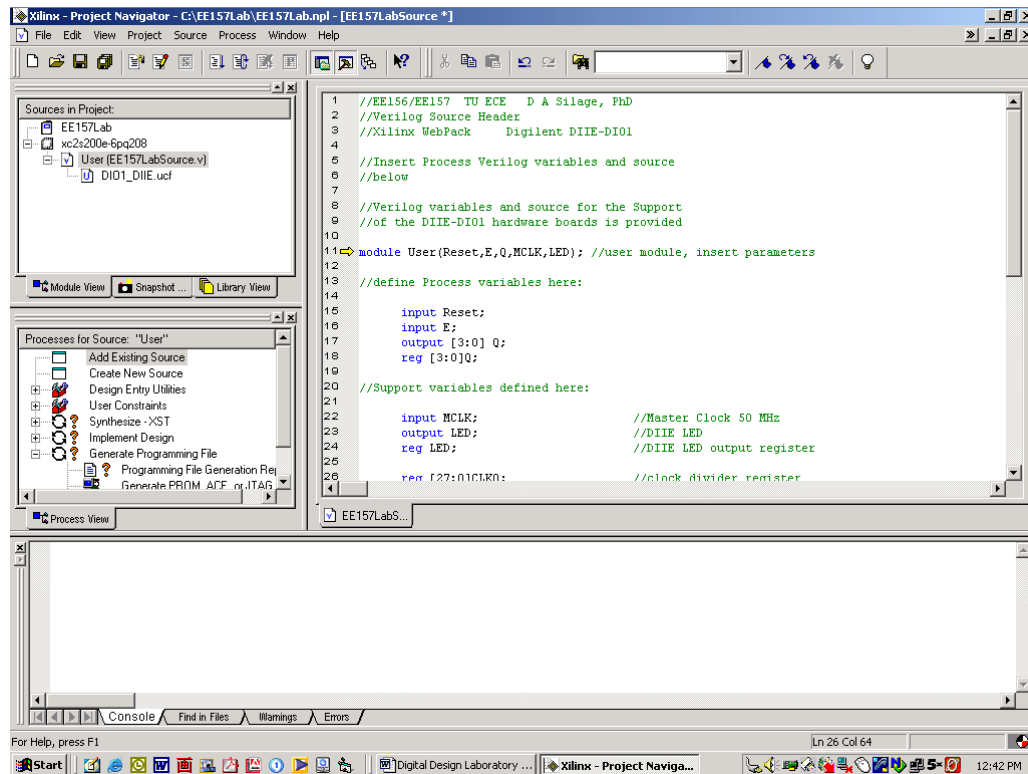


Figure 4

Project Header Verilog Source Code: *EE157LabSource.v*

```
//EE156/EE157 TU ECE      D A Silage, PhD
//Verilog Source Header
//Xilinx WebPack          Digilent DIIE-DIO1

//Insert Process Verilog variables and source below

//Verilog variables and source for the Support of the DIIE-DIO1 hardware boards is provided

module User([Parameters]);          //user module, insert parameters

//define Process variables here:      [

[Verilog Variables]

//Support variables defined here:

    input MCLK;                      //Master Clock 50 MHz
    output LED;                      //DIIE LED
    reg LED;                         //DIIE LED output register
    reg [27:0]CLKQ;                  //clock divider register
    reg Clock;                       //clock output register

//insert Process code here:

[Verilog Source Code]

//Support code for Digilent DIIE-DIO1 is here:

always@(posedge MCLK)
    begin
        CLKQ<=CLKQ+1;                //clock divider register
        if (CLKQ==25000000)          //1 Hz Output Clock: 50 MHz Master
            begin                    // Clock 25x10^6 half-cycle count
                Clock=~Clock;        //Output Clock
                CLKQ<=0;             //reset clock divider register
                LED=~LED;            //toggle DIIE LED ON/OFF
            end
        end
    end

endmodule
```

Verilog Parameters, Variables and Source Code for *Figure 7-56*

[Parameters]

Reset,E,Q,MCLK,LED

[Verilog Variables]

```
input Reset;
input E;
output [3:0] Q;
reg [3:0]Q;
```

[Verilog Source Code]

```
always@(posedge Reset or posedge Clock)
    if(Reset)
        Q<=0;
    else if(E)
        Q<=Q+1;
```

3. The User Constraints File (UCF) designates the FPGA hardware pins that will be used for the signals of the design. The complete UCF for the *Project Header*, *DIO1_DIle.ucf*, is listed below (the – character in a name is not allowed). Click on *File...Open* and select *DIO1_DIle.ucf* as shown in Figure 5. The UCF appears in the edit window on the right of the screen, as shown in Figure 6. The complete UCF is listed below.
4. The UCF file must be edited for your design entry. Most of the port pin assignments for the FPGA are commented out using the symbol #. The first two uncommented entries are port pins that are used on the DIle board for the master clock oscillator (MCLK) and an indicator (LED). Here the actual signal name from the source code (MCLK and LED) are entered after *NET* and before *LOC* (Location) with spaces in between, as shown:

```
NET MCLK LOC= P80;      #Clock on DIIE board
NET LED  LOC= P69;      #LED on DIIE board
```

P80, for example, indicates that pin 80 of the FPGA is the master clock oscillator input pin.

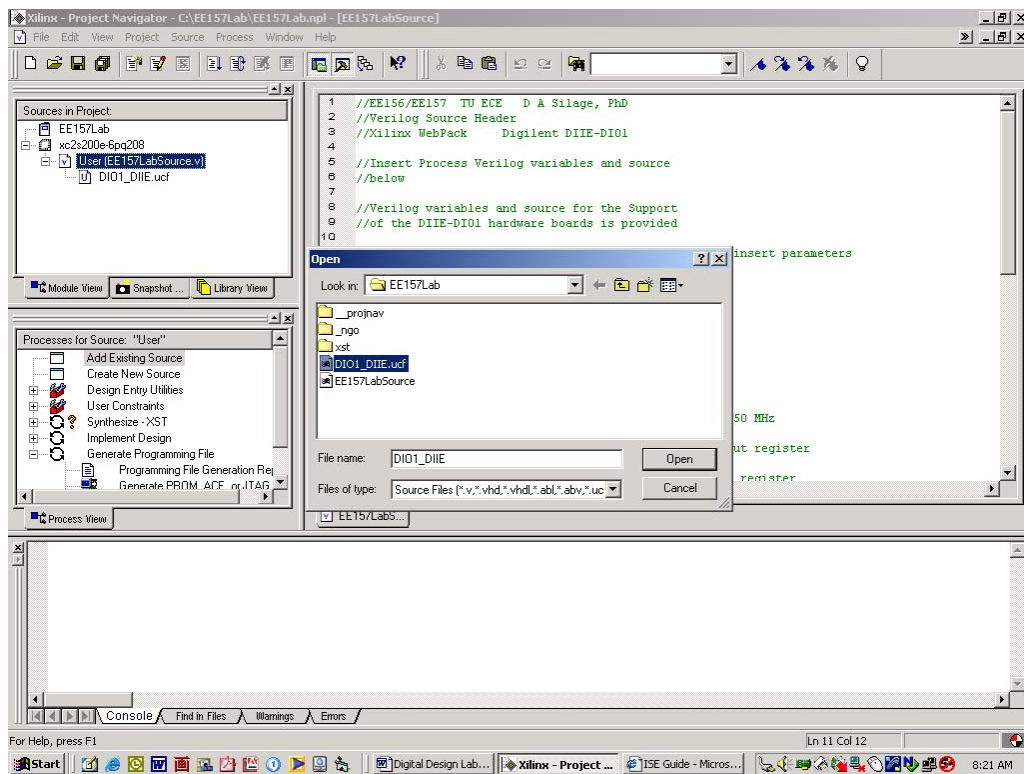


Figure 5

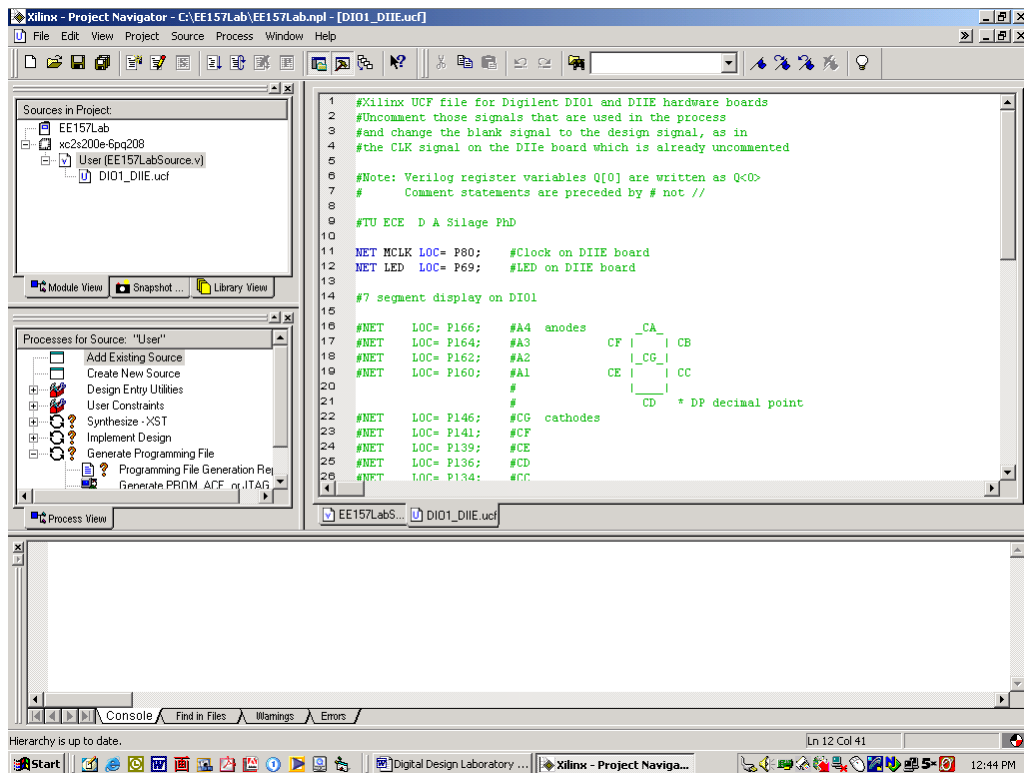


Figure 6

User Constraints File: *DIO1_DIle.ucf*

#Xilinx UCF file for Digilent DIO1 and DIIE hardware boards
#Uncomment those signals that are used in the process
#and change the blank signal to the design signal, as in
#the CLK signal on the DIle board which is already uncommented

#Note: Verilog register variables Q[0] are written as Q<0>
Comment statements are preceded by # not //

#TU ECE D A Silage PhD

NET MCLK LOC= P80; #Clock on DIIE board
NET LED LOC= P69; #LED on DIIE board
NET LDG LOC=P179; #LED gate

#7 segment display on DIO1

#NET	LOC= P166;	#A4	anodes	_CA_	
#NET	LOC= P164;	#A3		CF CB	
#NET	LOC= P162;	#A2		_CG_	
#NET	LOC= P160;	#A1		CE CC	
		#			
		#		CD	* DP decimal point
#NET	LOC= P146;	#CG	cathodes		
#NET	LOC= P141;	#CF			
#NET	LOC= P139;	#CE			
#NET	LOC= P136;	#CD			
#NET	LOC= P134;	#CC			
#NET	LOC= P132;	#CB			
#NET	LOC= P127;	#CA			
#NET	LOC= P148;	#DP			

#LEDS on DIO1

#NET	LOC=P176;	#LD8
#NET	LOC=P174;	#LD7
#NET	LOC=P169;	#LD6
#NET	LOC=P167;	#LD5
#NET	LOC=P165;	#LD4
#NET	LOC=P163;	#LD3
#NET	LOC=P161;	#LD2
#NET	LOC=P154;	#LD1

#Pushbuttons on DIO1

```
#NET LOC=P152;      #BTN4
#NET LOC=P151;      #BTN3
#NET LOC=P150;      #BNT2
#NET LOC=P149;      #BNT1
```

#Switches on DIO1

```
#NET LOC=P147;      #SW8
#NET LOC=P145;      #SW7
#NET LOC=P140;      #SW6
#NET LOC=P138;      #SW5
#NET LOC=P135;      #SW4
#NET LOC=P133;      #SW3
#NET LOC=P129;      #SW2
#NET LOC=P126;      #SW1
```

The input and output port signals names for your project must be entered by selecting an appropriate location. The input reset signal (Reset) is entered as pushbutton 1 (BTN1), the input enable signal (E) is pushbutton 2 (BTN2), and the count register output ([3:0]Q) must be individually entered for all four bits using LD1 through LD3, the DIO1 indicator LEDs. Note that register variables are entered as Q<0> and not Q[0] or some other variant.

5. Uncomment the appropriate line in the UCF by editing out the # symbol and entering the signal name for your project, as shown:

```
#NET Q<3> LOC=P165;      #LD4
#NET Q<2> LOC=P163;      #LD3
#NET Q<1> LOC=P161;      #LD2
#NET Q<0> LOC=P154;      #LD1

#NET E LOC=P150;      #BNT2
#NET Reset LOC=P149;    #BNT1
```

Save *All* the project files before proceeding, of course.

Compilation and Synthesis

Make sure that the source file is highlighted in the *Sources in Project* window on the upper left. Move the position bar in the window *Processes for Source* on the middle left. Double-click the last subentry under *Generate Program File*, which is *Configure Device (IMPACT)* as shown in Figure 7 below. The Verilog compilation and synthesis process begins. The *Console* window on the bottom shows the progress and any errors are highlighted with a red flag. If errors occur, edit the Verilog source file or the UCF as appropriate.

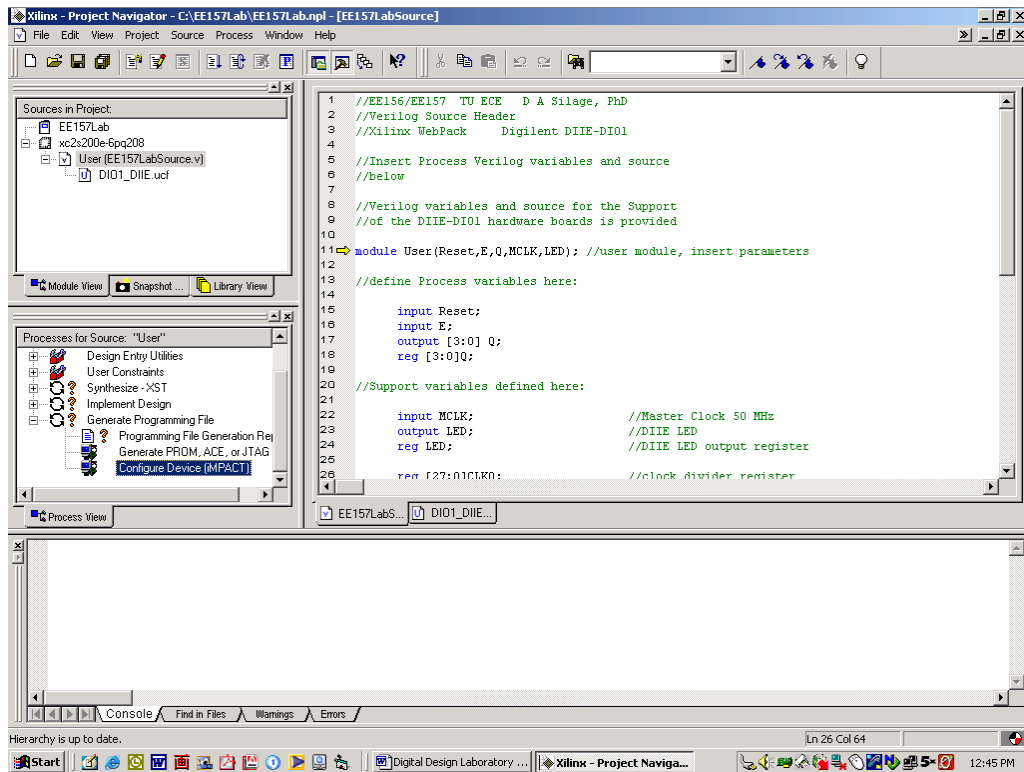


Figure 7

iMPACT Programming Tool

If the compilation and synthesis process is successful, the Spartan IIE FPGA *.bit* file or *fuse-map* is used to program the device. Note that the Spartan IIE is a static random access memory (SRAM) device (for high speed operation) and the device programming is volatile. If power is removed the program is erased. The iMPACT programming tool opens in a separate window, as shown in Figure 8 below. Click *Next* and accept the *Boundary Scan Mode* for programming.

Next, accept the *Automatically connect to cable and identify Boundary Scan chain* by clicking *Finish*, as shown in Figure 9 below. Downloading now occurs, as shown in Figure 10 below.

Click *OK* on the *Boundary Scan Contents Summary* screen (not shown) and the *Assign New Configuration File* screen appears, as in Figure 11 below. Select and *Open* the *user.bit* programming file, since *user* is the name of the project module here.

Ignore the warning message (not shown) concerning the assignment of XC2S200e *.bit* file to an XCV200e FPGA device by clicking *Yes*. Next, ignore the warning message (not shown) concerning reassignment of the StartUp Clock by clicking *OK*.

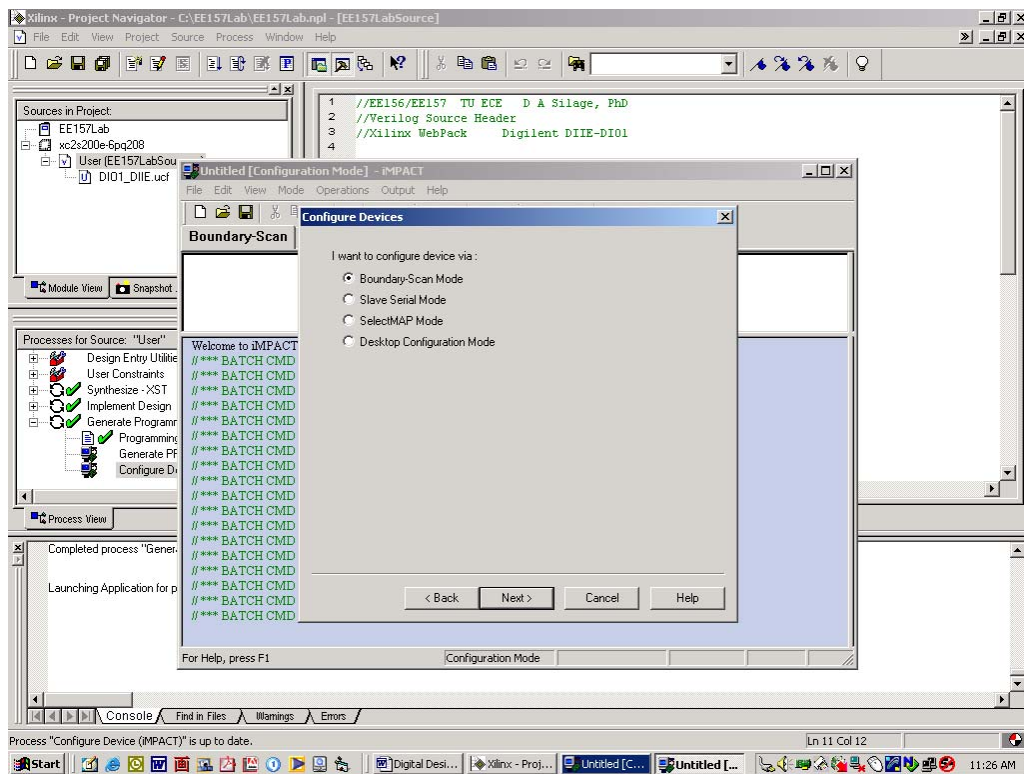


Figure 8

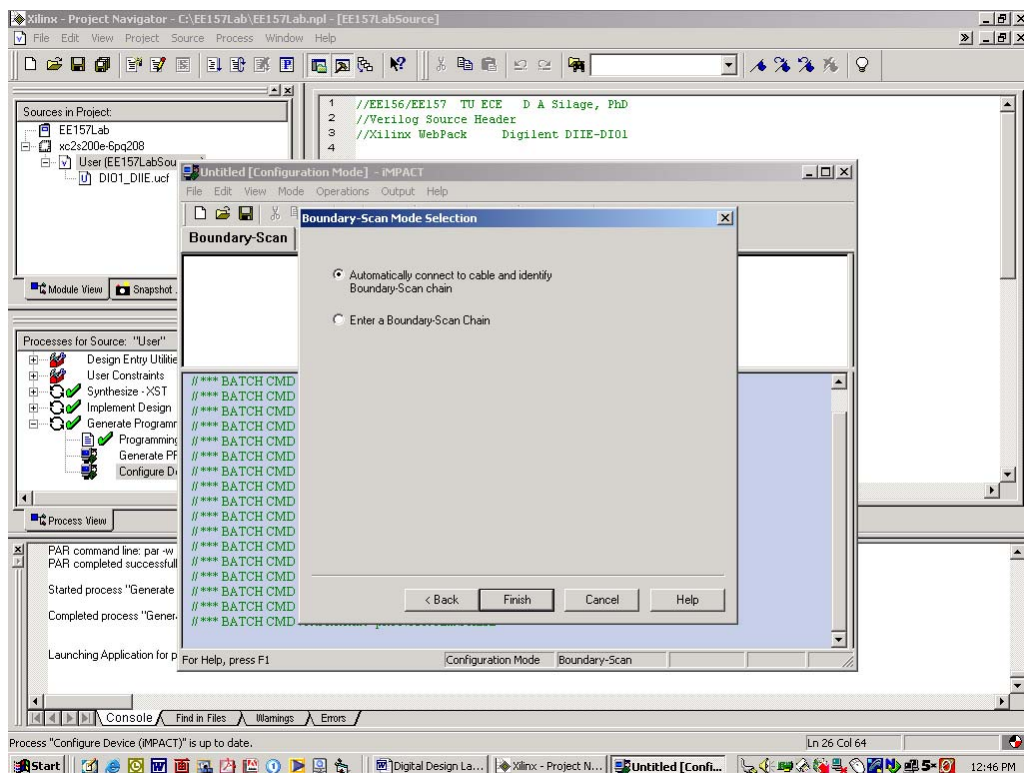


Figure 9

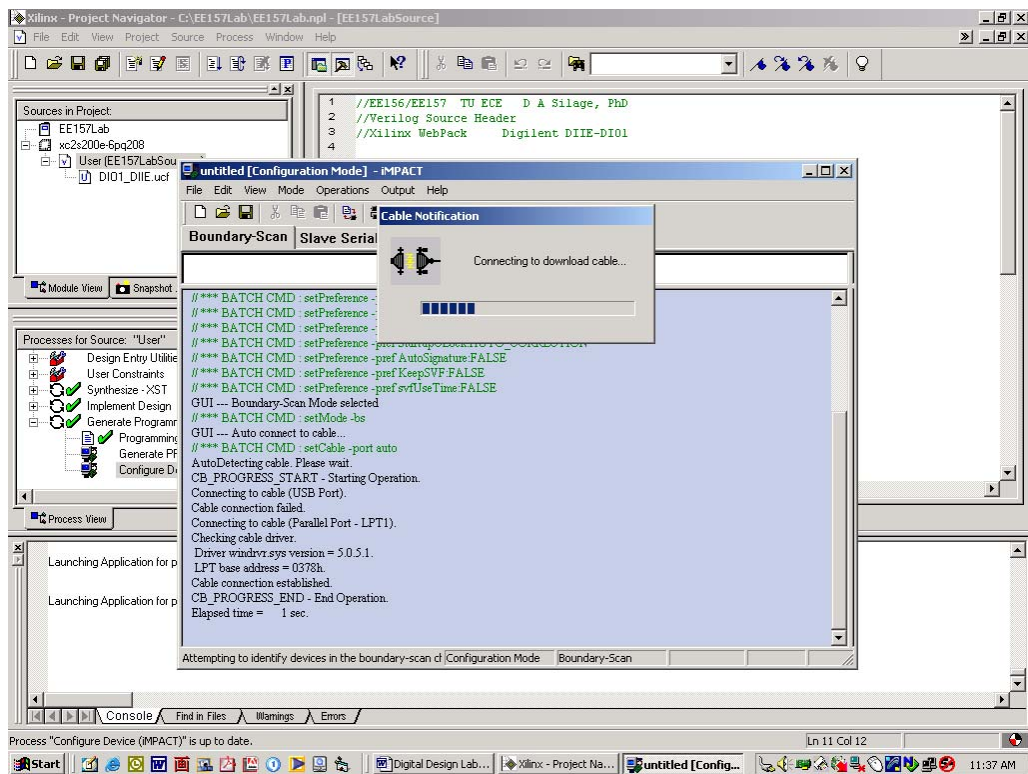


Figure 10

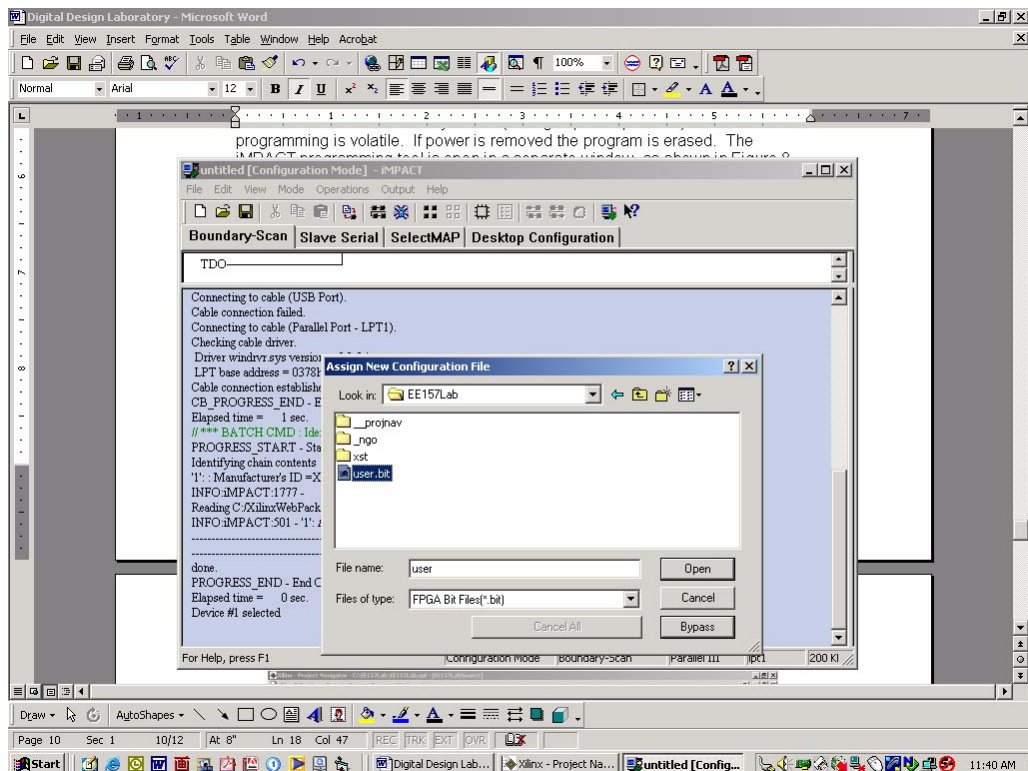


Figure 11

If you cannot identify the JTAG FPGA device in the upper (white) window of the iMPACT screen, actively pull the window section down, as shown in Figure 12 below. Right-click on the FPGA device (which turns green) and select *Program*, as shown in Figure 12. Make sure that *Verify* on the *Program Options* screen (not shown), is *not* selected and select OK.

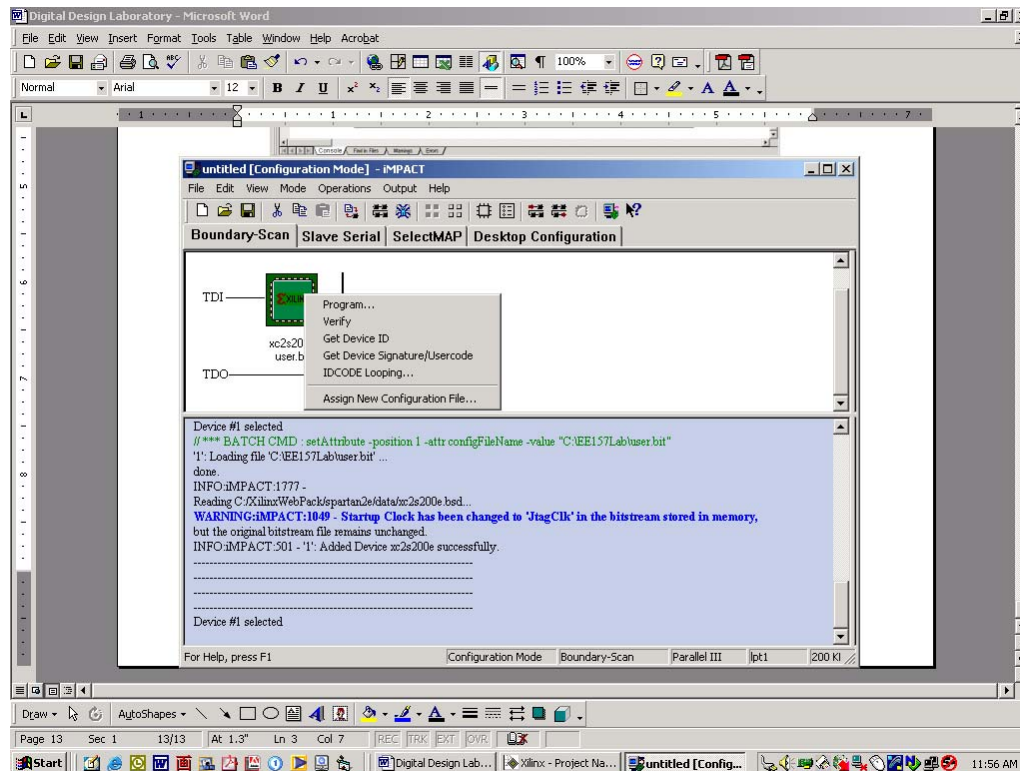


Figure 12

A *Programming Successful* prompt should then appear and the Spartan IIE FPGA device is programmed.

You should *close* the iMPACT programming tool window after use and not merely minimize it. A common mistake is that iMPACT is started each time it is evoked and the second occurrence will not successfully start with another occurrence open.

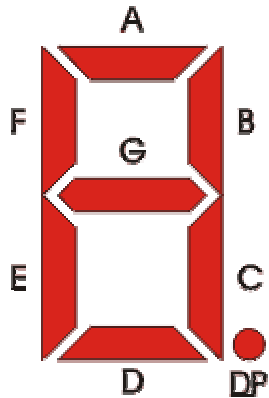
Modifications for Figure 7-56

You will notice that the Verilog code of Figure 7-56, a four-bit up counter, was modified slightly from that listed on page 403 of your text. This is because of the logic polarity of the pushbuttons (BTN) available on the DIO1 hardware board. The pushbuttons are normally logic 0, so the code was modified so that reset occurs on a logic 1 (Reset not !Reset) and counter enable occurs on a logic 0 (!E not E). Depressing BTN1, or Reset, zeros the four-bit count in LD1 through LD4. Depressing BTN2, or Enable, stops the count.

DIO1 Input-Output Ports

The eight LEDs available on the DIO1 board (LD) are directly available as indicators, as demonstrated in the example of Figure 7-56. The five pushbuttons (BTN) and the eight slide switches (SW) are used for program input. Placing the slide switch down (toward the edge of the DIO1 board) produces a logic 0. Placing the slide switch up produces a logic 1.

Note that the mechanical pushbuttons and slide switches can produce *bounce* or aberrant operation for time durations of about 2 msec. Since the *Project Header Source Code, EE157LabSource.v*, provides a *Clock* signal of 1 Hz (see the description below) the pushbuttons and slide switches could be effectively *debounced*. An event driven process (*always@(posedge Clock)*) probably would read the pushbuttons or slide switches after the *bounce* has occurred (Hopefully! There is about a 500:1 chance, the ratio of 1 sec to 2 msec, that it would not!).



The four seven-segment displays are common electronic devices. The seven segments can be used to display the numerical character 0 through 9. The cathode segments are labeled A, B, C, D, E, F, G and DP (decimal point), as shown. The DIO1 ports are actually driven on the display cathode (C) and are labeled CA, CB, CC, CD, CE, CF, CG and DP. The digits 0 through 9 can be formed by turning on various cathodes with a logic 1. The obvious bit pattern would be as follows:

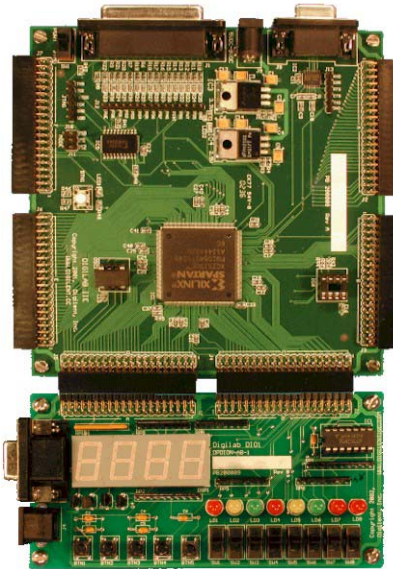
Digit	CA	CB	CC	CD	CE	CF	CG
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	1	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1

The four seven-segment displays are selected by turning on the appropriate anode signals A1, A2, A3 and A4 with a logic 1. However, note that the cathodes are shared by all four seven-segment displays, so the display must be multiplexed to show a multiple digit number. If the seven-segment display is

refreshed at a rate greater than about 30 Hz, the eye cannot see the actual flickering that occurs.

Dile Master Clock

The Dile master clock oscillator is a 50 MHz device (20 nsec period). This rate is so high that the four-bit up counter of Figure 7-56 would be a blur! To compensate for this, the support Verilog code of the *Project Header Source Code, EE157LabSource.v*, provides a clock divider so that the project clock is a more reasonable 1 Hz *Clock* output signal. The Dile LED is driven by this signal at a 1 Hz rate to verify the *Clock* operation.



Questions or Comments? email silage@temple.edu

Acknowledgement

The Xilinx University Program provided the Digilent Dile-DIO1 hardware board. The support of *Xilinx* for undergraduate ECE education is appreciated.

