Lab-200 Guide

Welcome to COE-200 Digital Logic Design Laboratory. This lab will introduce you to the exciting world of digital system design.

The field or rather art of designing digital systems has brought about profound changes in every sphere of our lives. Most of us are aware of the importance of some pervasive digital devices like digital-watches, calculators, personnel-computers, digitalnotebooks etc. But more interestingly, digital devices have made deep inroads in nearly every electronic device we are aware of (rewrite). Digital systems are used in broad spectrum of areas ranging from industry, military, household automation & control to home multimedia devices (rewrite). Such devices incorporate mini digital processing machines and are said to have embedded (invisible) systems.

Due to the widespread and complex use of digital devices, there always existed a constant drive to cut the design time and increase their design efficiency. The classic paradigm of designing using individual ICs (Integrated Circuits) is far lagging the pace and thus (in terms of design time) increasingly being replaced by automated design procedures. Today, rapid prototyping and development of embedded systems have become a reality through the use of FPGAs (Field Programmable Gate Arrays) and front-end tools.

The purpose of this lab is at one end to introduce students to digital logic, a hands-on-practice for digital designing and at the other end to familiarize them with the high-level design process.

We hope that this would be an interesting journey.

Xilinx

Xilinx is leader of programmable ICs, FPGAs, provider www.xilinx.com/

Front-End Tools GUI Based tools that streamline digital design process using either schematics or Hardware Description Languages (HDLs)



Figure 1.1: Design Flow In this lab, you would be designing digital systems applying your design knowledge acquired through course work. You will be using Xilinx's front-end tool, Xilinx Foundation Series 2.1i software (XF), to enter the design using its schematic drawer. To verify that your design is correct and that you have implemented (connected) it properly, you would then be simulating the design using internal simulator of XF 2.1i and check for functional correctness. After on-screen verification of the design, you are required to download it on Digilent Board and do the hardware verification. For downloading, you would be using software provided in the suit of Xilinx ISE 4.2i.

The above two Xilinx softwares are available as shortcuts in the program menu of your PCs and are in the folder **Start-Menu/Programs/Xilinx**. Their names are the *Project-Navigator* and the *Program*. *Project Navigator* is the Xilinx 2.1i interface and *Program* is Xilinx 4.2 design downloader. The two softwares are used because of incompatibility problem associated with older 2.1 editions while downloading the design and inability of 4.2 editions to support schematic entry for Spartan devices (Spartan device is the FPGA chip on the Digilent board).

Thus your design process involves interaction with Xilinx software and Digilent hardware that contains Xilinx FPGA. In a few labs, you would also be interfacing a few external ICs with the board using the accompanying breadboard.

The design outline is also pictorially presented in Figure 1.1.

2. The Software

Info

Xilinx Foundation Series offers a wealth of online documentation and multimedia tutorials.

FPGA

The FPGA device has an array-like architecture and is volatile (SRAM based). It is good to realize complex logic functions that contain both combinational and sequential circuits. Its capacity is usually limited by the number of input/output pins and not by its complexity.

CPLD

Complex Programmable Logic Device has a PALlike architecture and is non-volatile. It gives relatively good performance (up to 250 MHz) and is well suited for combinational logic circuits and control logic of medium complexity (up to about 10,000 logic gates).

HDL

A hardware description language allows you to describe the behavior of a system rather than specifying individual gates. There are several popular hardware description languages such as VHDL, Verilog and ABEL. The Xilinx Foundation© CAE (Computer Aided Environment) system is a development tool that consists of an integrated set of programs to create, simulate and implement digital designs in a FPGA or CPLD target device. All the tools use a graphical user interface (GUI) that allows all programs to be executed from toolbars, menus or icons. On-line help is available.

Designs can be entered in two basic modes: *Schematic* and *HDL* (Hardware Description Language) mode. This lab requires you to specify the designs using schematic mode. Schematic entry flow is presented in Figure 2.1.



Figure 2.1: Schematic Flow

1.1 Project Manager

Xilinx Foundation Series project manager is an integrated development environment (IDE) that provides coordination among the different tools that come with the package.

Project manager can be invoked from the Start menu. Start the project manager with a new project command (also available in the file menu) and enter the design name and the directory to store it. (Hint: It is a good practice to make a new directory under \Projects with its name being the same as that of your current experiment). Also specify the name of the project. Choose other options as shown in Figure 2.2.

New Proje	ct		×
Name:			OK.
Directory:	D:\FNDTN\ACTIVE\PI	ROJECTS	Cancel
-			Browse
Туре:	F2.1i	•	Help
Flow:	Schematic	C HDL	
Spartan	▼ S10PC84	•	3 💌

Figure 2.2: New Project Settings

The Digilab boards on which we are going to download our designs have a Spartan^M family FPGA whose device number is S10. The remaining part of the string (S10PC84), which is PC84 in the device names, indicates that it has a pincount of 84. The '3' in the last box represents the speed grade of the device. You can also confirm these markings from the FPGA installed on the board provided to you.

Once you have specified the above information, project manger opens and your screen would look like the snapshot shown in Figure 2.3.

When you create a new project e.g. **myproj**, the Foundation tools will create some extra files whose description follows:

- A project configuration file (PDF), called Project Description File (**myproj.pdf**)
- The project library file (**myproj**), which contains information of the design
- The simulation library file (simprims), which contains information regarding simulation
- Device library file, which is the library of the device being used. In the present case it would show 'spartan'

The libraries are shown in the left window-pane (called the Hierarchy browser) of the Project Manager. A project must always have one or more top-level design files. In case the top-level cell is a schematic, the file will be shown in the hierarchy browser with an extension .sch (not yet present in our diagram). The foundation tools will create additional folders and files during different stages of the project design/implementation.



Figure 2.3: Project Manager Window

The right window-pane in the Project Manager has several tabs. The Flow tab graphically shows the different steps involved in the design of a project as was shown schematically in Figure 2.1. You can click on the icons to access a particular tool, as we will be seeing later. The bottom window-pane (status pane) gives status and error or warning messages.

In case a library is missing (e.g. the device library), the tools will give you a warning. This will also show up when you try to open the schematic, as symbols will be missing or blanked out. If that happens, you can add the library to the project. In the Project Manager, go to File\Project-Libraries. This will open the Project Libraries window, shown in Figure 2.4. The left side panel shows the "Attached Libraries" and the right side windows the Project Libraries. If any of these are missing, you can add them from the list in the left side window. For using Digilab board, which contains a Spartan device, we would only be concerned with the three libraries that are being displayed in the project libraries list in Figure 2.4. Add any library if missing.

Project Libraries			x
Attached Libraries:		Project Libraries:	
flash myproj simprims spartan	Add >>	myproj simprims spartan	
spartanx virtex	Info		
xabelsim xc3000	Lib Manager		
Attach	Close	Help	

Figure 2.4: Project Libraries

1.2 Design Entry using Schematics Editor

Schematics editor can be invoked by clicking on its icon in project manager. Schematics Editor's window appears (Figure 2.5). Throughout the lab, we would be using schematics editor to design our systems. We will now see how to use various features of schematics editor to design our labs.

;• \$	Sche	matic l	Editor -	[EASYP	R01.SCH]						_ 🗆 ×
S.	<u>F</u> ile	<u>E</u> dit	<u>M</u> ode	<u>O</u> ptions	<u>H</u> ierarchy	⊻iew	<u>D</u> isplay	<u>T</u> ools	<u>W</u> indow	Hel <u>p</u>	_ & ×
1	e	- E. (5 %	ēn 😢]		${\bf n} \propto$	∕ 3] ₩[(↔	0 15 13	1 Dig		
R I		Hierarch	/ Push/P	op	Î		/ Simulation T	Toolbox			
D		Symbol 1	Foolbox		Zoom Tools			Simu	ulator		
Ī		DrawW	ire								
4		Draw bu	ses								
		Draw bu	ıs taps								
큰	h~	Add Net	or Bus n	ame							
붋	<u>├</u> -	Hierarchy	Connec	tor							
<u>_744</u>	0	raphics 1	Foolbox								-
	•										
	E	ASYPR	101								
		4.2,	2.1						5	Select a	nd Drag

Figure 2.5: Schematic Editor

Placing Symbols

You can add the logic symbols by clicking on the Symbol Toolbox (Symbols icon) in the toolbar on the left. An SC Symbols Toolbox will pop up (Figure 2.6). You can scroll down the list and select your symbol or type the symbol name at the bottom box of the list. Notice that a brief description of the selected symbol appears at the bottom. You can now place the gate with your cursor on the schematic by clicking the mouse. Tip

To place a symbol that is already present in the schematic sheet, simply click on the previous instantiation and click anywhere you like to place the new symbol.

Drawing and Naming Wires

To connect one gate to the other, use the Draw Wire feature available in the toolbox. This can be done by clicking on the wire symbol just below the Symbol Toolbox icon of the left side toolbar. The shape of the cursor will change into a pencil-like image upon its selection. To connect two symbols through a wire, click on the first symbol's input/output pin after entering the draw-wire mode and then click on the pin you want to connect to. All symbols must be connected with wires (Figure 2-7).



You can name a wire by clicking on the "Name Wire Icon" just below "bus" icon on the vertical toolbar. Type in a name in the Net Name window and put the cursor over the location of the wire. Make sure you point to the wire that you want to name; otherwise the Tip

A shortcut to naming nets is double clicking on the net

name will not be connected to anything. Always fill out the net name between the IO pads and buffers. Net names should appear in blue (green names indicate that the name is not connected to a net).



Figure 2.6: SC Symbols

Figure 2.7: Connecting the wires

Тір	IIIV
To redraw the screen, press the F10 function key.	The SC Symbols Toolbox provides library primitives of the selected device. In other words, it provides components that directly map to the device architecture. Make sure that Spartan device is selected as you place the symbols.

Info

Adding IO Pads and Buffers

For taking outputs or applying inputs externally through the boundaries of FPGA, you need to route your input signals through an input pad and buffer before applying it to the logic. Similarly the output signals have to be taken through output buffer before applying to the output pad.

IO-Pads

IPAD: Input Pad OPAD: Output Pad IOPAD: Input/Output Pad You can add buffers and IO-pads in the same manner as you place individual components. All device pins MUST be represented with one of these I/O pads. Pads should be given a name (label) and possibly a pin number (pin location).

IO Buffers

IBUF: Input Buffer OBUF: Output Buffer

Info

You need to route clock inputs through a special global clock buffer having name 'BUFG'

Drawing and Using Buses

Schematic editor allows you to include buses in your design. A bus can be added by using 'Draw Buses' icon in the Symbols toolbox. Click on any part of the schematic where you want to introduce bus. Double clicking





Figure 2.8: Bus Properties

will terminate the bus along with a pop-up screen asking you to define parameters for the bus you just made. The parameters defined in Figure 2.8 define a bus IN of type Input having 8 bit-width, MSB is bit 7 and LSB is bit 0. Similarly output and bidirectional buses can be created. Once a bus has been created, you need to put input or output buffers depending on the direction of data, the bus is transmitting. After that, connect the buffers with the bus using 'bus taps' tool found in the toolbox just below the bus tool. Rename the connections so they reflect the bit they are carrying. An example of 7-bit input bus (IN[6:0]) is shown in Figure 2.9.

Pin Assignment

The pins on the FPGA are connected on specific locations on the prototype board. Pin assignment is necessary while taking input from a particular location, for example input buttons. Similarly the outputs have to be routed on particular pins to get them displayed on sevensegment displays or LEDs that are hardwired to some pin location on FPGA (See pin assignment detail of Digilent board later in the document).

You can assign pin numbers to each input and output pad. If you don't do this, the tool will automatically assign the pin numbers for you.

There are several ways you can assign pin numbers. First, you can place the pin numbers on the schematic using the 'LOC' property. This is done by double clicking on the PAD symbol. In the pop-up Symbol Properties window (see Figure 2.10),



1.1.1.1.1.1.1 Figure 2.10:

enter (or select from drop down list) Parameter Name: LOC in the parameters section, and pin number as P#, where # represents the pin number as shown in the figure (the letter P is required in front of the number). Click on the ADD button. Two diamonds will appear next to the description just added in the list. Click OK when finished.

An alternative way is to assign pin numbers using Constraints Editor (Tools\Implementation\Constraints Editor from Project Manager main window). For editing constraints like pin assignment, first finish your schematic and exit it after saving and generating its netlist (more on saving and generating netlist follows). If the option of constraints editor appears in gray in the project manager window then compile the project once and then use the constraints editor (See later pin assignment using constraints editor).

Adding Designer's Name, Project Title and Date

It is good practice to specify a name to your design. There is a standard way of doing so. Go to the bottom of the page and fill out the small rectangle. If the box has a predefined name, you can change this by going to the File\Table Setup. You can now change the Address, Name, Description, Date, etc.

Netlist and Integrity Test

You will need to generate a netlist, which is in a format that is readable by the compiler. This is done by going through **Options****Create Netlist**. When finished, it is always a good idea to check that the schematic has no electrical design rule errors. This is done through invoking **Options****Integrity-Check**. A message indicating the success of integrity test appears as shown in Figure 2-13.

Debugging

If there are any errors/warnings while performing the integrity test or during compilation, they will get displayed in the status pane of the project manager. То easily locate anv nets causing error/warning, you may change the mode into 'Query' (Mode\Query) in the schematics editor. A SC Query/Find window will appear as shown in Figure 2.11. You can now click on the various nets and components in schematics editor to locate the problem (Figure 2.12). Alternately, you can also put the name of the net that is causing the problem in the text box of SC Query window with proper selection of type of device.

•	SC Query / Find
\$NET00010_	
-Search for: -	Search in:
Signal/Bus	C current sheet only
Cinstance	Opened sheets
○ Pin	·
C Symbol	
Not	hing selected

Figure 2.11: SC Query/Find Window

Tip

Most of the warnings are due to improper connections. Make sure you have removed the problem by performing integrity check before proceeding to compile the design.

Save your schematic

Use File\Save option to save your schematic with '.sch' extension. When finished with the schematic, exit the Schematic capture program, which will bring you back to the Foundation Program Manager window.



Figure 2.12: Debugging using Query Mode

Adding the schematic to the project

If the created schematic is not listed in the Project Manager window under the Project you created (e.g., myproj.sch), add it manually using Document\Add in the Project Manager window.



Figure 2.13: An Integrity Tested Schematic

1.3 Creating Macros

Often, you will use a circuit (consisting of different logic gates) again and again, for example, a seven-segment decoder or a full adder. Instead of drawing the full logic again, it would be more efficient to make a module or macro of it with its own symbol. This macro can then be added to library and can be used like other available devices. A macro hides all the underlying logic and simplifies the design by providing only the interface signals for use in the design. The macro then gets added to your symbol library. You can use it like you use ordinary symbols of the library.

A macro's creation is essentially the same as creating the schematic with just two main differences.

Since the output of the macro need not be routed outside the FPGA, you don't need to add input/output buffers (IBUF, OBUF).

You do not use IPADs or OPADs for the input and output ports. Use I/O terminals instead to indicate the input and output I/Os (More on this later).

A macro can be created by three different methods

Using Symbol Wizard

By using this method, the wizard automatically adds I/O terminals and adds the symbol in the library for you. What remains is to draw the logic of the macro.

From an Existing Schematic

You might face a situation where you want to convert an existing schematic into macro. Here you will need to manually replace the I/O Pads and Buffers with I/O terminals and add the macro to the symbol library.

Creating Schematic First

This is essentially the same as creating macro from the schematic. The only difference is instead of replacing I/O Pads and Buffers from the schematic with I/O terminals, you add I/O terminals directly.

1.3.1 Using Symbol Wizard

Open the Schematic Entry tool by clicking on the Schematic Entry icon in the Project Manager Window.

In the Schematic Entry window, go to the Tools\Symbol Wizard. A Wizard will guide you through by asking several questions.

- For the Symbol Name, type in the name you want to give, e.g. 7Decoder (for seven segment decoder). Select the Schematic for Contents.
- Then click the NEXT button. Specify, by filling in, the names of the input and output ports. When all input and output ports have been defined, click the NEXT button. Follow the instructions.
- When finished click the FINISH button.

The symbol will now be created and added to the project library. An underlying design file will also be created that has the defined port names. The schematic editor will open showing an empty schematic page for the macro, except for the input and output pins. Then, you draw the logic for the macro in the same way as you draw a new schematic.

1.3.2 Using an existing schematic as a macro

In case you created a schematic as part of another project and would like to use this schematic as a macro in a new project you can do so. The steps are as follows:

First you will have to add the old schematic to the current (new) project. In the Project Manager window, go to the File\Add Document menu. Use the browser to locate the schematic of interest. When done, you will see that the schematic has been added to the project in the left window page (Files tab). Let's assume that the schematic is named TEST1.sch.

- You may want to modify this schematic, e.g., to change the pin names, replace input PADS (IPADS) by I/O terminals and remove buffers (BUF). Remember that for macros you do not use IPADs or OPADS (these indicate the physical pins of the device) but I/O terminals
- Next you need to create a macro from the schematic. This is done as follows: go to Hierarchy\Create Macro Symbol from Current Sheet. A window will pop up. You can change the name of the macro, and add a brief description in the comment line. Let's call the macro MACRO1. You can also check the input and output signals. Click OK.
- When you convert the schematic **TEST1.sch** into the macro MACRO1, there is no longer any relationship between the schematic TEST1.sch and the project, even though you can still open the schematic TEST1.sch. You can remove the schematic from the project directory using the Windows Explorer.

1.3.3 I/O Terminals

You can add I/O terminals (pins) by clicking on the I/O Terminal icon (hierarchy connector) on the left-hand side of the toolbar or at the top of the SC Symbols window. A pop-up Terminal window will open. Fill out the name of the terminal (ex. X, Y or Z) and indicate whether it is an Input or Output port. Be careful to indicate the right direction for the inputs and outputs.

1.4 Simulation

Design verification is an important aspect of project design. Powerful simulators are available that help check each and every design unit before its download and chip verification.

Simulation is of two types, *functional* and *timing* simulation. The position for either type of simulation is shown in Figure 2.1. Functional simulation is concerned with logic verification of the design so as to ascertain the correct operation of the required logic. Apart from this, timing simulation also utilizes gate level delays of the device type selected in showing the output. Thus, timing simulation requires information of the chip as well as the design for its working. Hence it is carried out after the chip has been compiled for a particular device. The method of performing either type of simulation is pretty much the same with some extra steps included in the latter for timing analysis. Both types of simulations are discussed in this guide though our major focus in labs would be to simulate functionally.

1.4.1 Functional Simulation

Invoke the simulator either from the project manager or through its icon in the schematics editor. The simulation window appears as shown in Figure 2.14.



Figure 1.14: Simulator Window

Below are the steps you need to perform simulation.

Adding Signals

To simulate the circuit and probe if it is working correctly, you need to add the signals you need to view. The signals added are displayed in the signals pane and their simulation waveforms in the waveform pane. We proceed inputting the signals by clicking on the component selector button in the toolbox. The component selector window appears as shown in Figure 2.15. Select the signals to view and while selected right click to add the signals in the signals pane.

Component Selection for W	Component Selection for Waveform Viewer					
Signals Selection	Chip Selection	Scan Hierarchy				
N V BUZZER N V ELECTRICITY N V HEAT N V LIGHT N V SimGlobalReset	<pre>\$\[\begin{aligned} \$\leftyleftyleftyleftyleftyleftyleftylefty</pre>	T Root				
Add Close Help						

Figure 2.15: Component Selector

An alternate way to add signals is to add probe in the schematics window at the wires you wish to include in the signals pane. You can invoke schematics editor within simulator by clicking on the schematic capture icon in the toolbox. Schematics editor will open along with schematic probes (SC probes toolbox). You can add probes on any wire in the schematic by selecting the probe tool in the SC probes toolbox and clicking on the nets you wish to be included in the simulator (Figure 2.16).



Figure 2.16: Adding Probes in Schematic Editor

Adding Stimuli

After the signals have been added, you can apply stimuli to input signals and view the behavior of the circuit on your chosen set of inputs.

Xilinx Foundation Series in-built simulator provides many ways to apply the stimuli. One of the best ways is to use 'stimulator selection'. Click on the stimulator icon in the toolbox to view the 'stimulator selection' window (Figure 2.17).



Figure 2.17: Stimulator Selection

A key on the keyboard of your PC can be also assigned to any signal using the stimulator shown in Figure 2.17. This will cause the selected signal to get associated with that keyboard key so that it toggles whenever you press that key. You can also permanently ground or high a signal by assigning 0 or 1 key in the keyboard section of the stimulator.

Underneath the keyboard section are two rows of round LEDs and one row of square ones. The round LEDs represent the bits of a free-running 16-bit binary counter. This counter can be used to generate clock signals of various frequencies with a 50% duty cycle (See more in the following section on how to define frequency of the counter).

The top row of LEDs, labeled Bc: gives the output in true format and the 2nd row, labeled NBc: gives it in the inverted format. The first four outputs from the far right are labeled B0 to B3; the next four outputs are labeled B4 to B7, etc. The outputs will go through the following sequence:

.....

The outputs of the inverted counter, NBc, will be

An alternate way of applying stimuli is to use 'Logic State Descriptor' available in the toolbox. For applying the stimuli through this method, select the signals that have to be assigned value after opening up the logic state descriptor window and click on the appropriate button.

Defining the Frequency of the Free-Running Counter

One can define the clock frequency of the free-running counter. This can be done from **Options****Preference** menu in the Waveform Viewer. Under the Simulation section, click on the B0 period drop-down list and select the required period. The frequency will automatically adjust. The precision depends on the type of simulation. A functional simulation requires less precision (e.g. 1ns) while a timing simulation requires higher precision (e.g., 100ps or less).

Simulating and Viewing the Waveforms

To conduct functional simulation, choose "Functional" from the pull down menu at the top of the simulator window. Then click on the Simulation step icon at the top of the window to initiate the simulation. The inputs and corresponding output waveform are displayed, as shown in Figure 2.18. To change the size of the step, use the pull down menu next to the step icon. You can also change the scale of the time axis by clicking on the zoom icons on top of the Signal list. An alternative way is to click and drag on the time axis on top of the waveforms pane. The time scale is displayed as well.

You can switch back to the schematics window by clicking on the SC icon in the Simulator window. The schematic window will show the actual logic levels (0 or 1) of each of the signals added to the Waveform Viewer window. The results will be shown on the schematic for the signals you have added to the Waveform Viewer window. You can add additional signals by clicking on the Probe icon in the SC Probes window. Click on the nets that you want to display. Figure 2.18 shows the results of the simulation in both the Simulator and Schematic corresponds to the last time step displayed in the Simulator window or to the position of the cursor (47ns in Figure 2.18). One can also view the values of the logic signals on the schematic diagram. This is another convenient way to check the operation of the circuit.

Now you can verify if the circuit performs as expected. If problems occur, check your logic or the schematic.

To clear the waveform and start the simulation over again, go to the Waveform\Delete\All Waveforms-With-Power-on menu item.

You can also save the waveform from File\Save Waveform.



Figure 2.18: Schematic and its Simulation Output

1.4.2 Timing Simulation

Timing simulation is performed in the same manner as functional simulation. It is invoked by clicking on the 'Verification' button (First icon on the button). By this, the timing simulator will be loaded and is ready to be used. The only difference you might notice from Figure-2.14 is the selection of 'Timing' instead of 'Functional' in the drop down menu.

You can now proceed in selecting the signals in the waveform viewer and adding the stimuli in the same manner as you did for functional simulation. Alternatively, you can load an earlier created waveform by going to the File\Load Waveform menu.

In case you are using Flip-flops you have to assert the Global Reset. The global reset (GSR) must be pulsed at the beginning of all timing simulations. This sets or resets the flip-flops in the schematic. The Global Reset signal does not exist in the schematic but it exists in the device and timing simulation netlist.

When all signals have been added and stimuli been specified, make sure 'TIMING' is selected from the drop-down list on the horizontal toolbar. You can now run the simulation by clicking on the Step icon.

You may have to zoom in on the waveform in order to see the timing delays. Click on the time ruler and drag to zoom at that particular time reference. To measure the delay, go to the **Waveform**\Measurements\Measurements-On. Position the cursor over the edge of the signal of interest to indicate the beginning of the measurement. Click on another transition to complete the measurement.

For instance, consider Figure 2.19, and note the delay between application of the input and its propagation to the output of the buffer. Refer to Figure 2.20 for relating assignments with the actual nets of the schematic.

🗧 Waveform Viewer 0 📃 🗆 📃 🗵
100ps/div 100ps/
57.3ns duuluuluuluuluuluuluuluuluuluuluuluuluul
1 LIGHT B2
1 HEAT NO
lELECTRICITY. d
1 CI
Ins
1 INV1
1 INV2
2ns
101
102
10R2
Ins

Figure 2.19: Timing Simulator Output Diagram for Schematic of Figure 2.20



Figure 2.20: Schematic for Timing Simulation

2 Implementation of Design

After the design has been found to be functionally correct, the next step is its translation into the programmable device. This step is the 'Design Implementation' phase, which is mostly a push button operation with the software doing all the hard work.

Click on the implementation button in the project manager to start design implementation process. For the first time, a pop-up window (Figure 3.1) would inquire about the device and design information. Make sure the device and its speed grade match the one shown in the figure. These settings match with the device we have on Digilab Board. Click at 'Run' to proceed with the implementation process.

Implement Des	sign				X
Device S10P	C84	•	Speed	3	•
Version name:	ver1				
Revision name:	rev1				
Control Files:	Set			Option	ns
Run	ОК	Car	ncel	Hel	P

Figure 3.1: Device/Version Information

A window indicating the flow appears as shown in Figure 3.2

The first step is the translation of the design file in a proper format. This implies that the representation of your schematic (or HDL) is translated into FPGA elements corresponding to the target device (look-up tables, etc). The next step is mapping of the design to the specific target device. The mapper optimizes and maps the design in the targeted FPGA device. Next is the Place & Route operation, followed by the generation of the timing information for use by the timing simulator. The final step is the generation of the BitStream that is a configuration file that can be used to program the FPGA.

🚰 myproj (ver1->rev1) -	Flow Engine			
Flow View Setup Utilities	Help			
<u> </u>	?			
SPARTAN Design Flo	w (rev1)			Status: OK
	• 🎦 •			➡ ^{III} m
Translate	Мар	Place&Route	Timing (Sim)	Configure
Completed	Completed	Completed	Completed	Completed
Running DRC. DRC detected 0 er Saving 11 file in Creating bit map. Saving bit stream xcpy myproj.bit d xcpy myproj.ll d: *	rors and 0 warnings "myproj.ll". in "myproj.bit". :\fndtn\active\proj \fndtn\active\proje	s. jects\myproj\myproj ects\myproj\myproj	.bit 11	×
For Help, press F1				XCS10-3-PC84 myproj.ucf

Figure 3.2: Design implementation flow

A pop up window appears informing the success of design implementation phase. Click OK. In case errors are reported, refer to Implementation Log file in the Project Manager (click on the Reports tab on top of the right window pane)

2.1 Implementation Results

You can view the reports from the **Project Manager**. Click on the **Reports** tab in the design flow window-pane and double click on the **Implementation Report** Files. This will open the report browser window. You will see the Translation, Map, Place & Route, Pad reports and others. Click on the Pad report to see the assignments of the I/O pins. You will recognize the same names as the ones you constrained earlier. Another interesting report is the Map report, which informs if any logic has been removed (as part of the optimization) or added. The Place & Route report indicates how much of the device has been utilized. It gives also a rough estimate of the average interconnection delay.

In case of errors/warnings at any step in the design implementation phase, the report file of that process will transcript description of that problem. Check with the report file and work along the directions given in the description. Recompile after removing the cause of the problem.

2.2 Assigning pins with user constraint

There are two types of constraints: (1) location and (2) timing. In general location constraints allow you to control mapping and positioning of logic elements in the target device, such as the location of the pads (IO pins). Timing constraints inform the system which paths are critical and need short interconnections (high speed lines) in order to ensure that your design's performance functions properly under worst-case conditions. We will concentrate on the location constraints.

Constraint Editor

If you did not assign pin numbers to the input and outputs in your schematic, you should do it now, before compiling the design. If you don't assign the pins, the compiler will assign them for you. It is necessary to assign the pins by yourself to make use of on-board switches, LEDs and seven segments available on the Digilab board.

Constraints editor can be run after you have implemented your design. Invoke Constraints Editor from the project manager window (Tools\Implementation\Constraints-Editor). Constraints editor window opens as shown in Figure 3.3.

🍇 Constraints Editor - [Global - mypr	oj.ngd / myproj.ucf]			_ <u>8</u> ×
File View Window Help				
Clock Hot Home	Boriod	Pad to Sotup	Clock to Bod	 1
	Fendu		Clock to Pau	
Pad to Pad				
<u> </u>				
Global Ports	Advanced			
				[
Editable Constraints	Constraints (read-only)			
				1
				4
21				<u>ا</u> ح
Errors (Warnings) Info	Messages /			
For Help, press F1				NUM

Figure 3.3: Constraints Editor

Click on the 'Ports' Tab and assign pin locations in the location textbox next to the unassigned pins. See Figure 3.4.

ZZER OUTPUT P99 NA ECTROTY NPUT P27 NA AT NPUT P26 NA HT NPUT P28 NA HT NPUT P28 NA HT NPUT P28 NA HT NPUT P28 Select Group Erohbit I/O Locations Clobal Ports Advanced	Port Name	Port Direction	Location	Pad to Setup	Clock to Pad	
ECTRICITY NPUT P27 N/A AT NPUT P26 N/A NPUT P28 N/A IT NPUT P28 N/A Profile Group Name: Create Group Prohibit I/O Locations Select Group Global Ports Advanced	IZZER	OUTPUT	P69	N/A		
AT NA HT NA HT NA HT NA HD Configuration Options Frohibit I/O Locations Global Ports Advanced K	ECTRICITY	INPUT	P27		N/A	
HT P28 NA Pad Groups Group Name: Create Group Prohabit I/O Locations Global Ports Advanced Clobal Ports Advanced	AT	INPUT	P26		N/A	
 	7 1/O Configuration Options	Pad Groups Group Name:	[Create Group		
🔆 Editable Constraints 🎋 Source Constraints (read-only)	Prohibit I/O Locations Global Ports	Advanced		Select Group		

Figure 3.4: Assigning pin constraints

After editing the pin constraints you will need to save before exiting the constraints editor and recompile the design to cause the changes to take effect.

3 DigiLab Board

The Digilab is a handy board to implement digital circuits on a Xilinx FPGA. The board contains a Spartan XCS10 (5V) FPGA. The board also provides 4 seven-segment displays, 8 LEDs, 8 general-purpose switches, four button switches, a serial port, a parallel port, PS/2 mouse and keyboard connector, VGA port, an 1/8" audio connector, a BNC connector, 2 on-board clocks, a ROM, a regulated power supply, a breadboard and several connectors for easy access to the FPGA pins. A simplified schematic of the board is shown in Figure 4.1 below.



Figure 4.1: Digilab Board

3.1 General Purpose Slide Switches SW1-SW8

These switches (SW1-SW8) can be used to supply a "1" (Vdd) or a "0" (GND) to the pins of the FPGA. The switches connect both to the FPGA pins and the pins of the J1 connector as indicated in the Table 1 below. The switch will provide a "1" (Vdd) when the slider is in the down position (towards the edge of the board) and a "0" (GND) when in the up position.

Slide Switch #	Pin # on FPGA	Pin on J1 Connector
1	28	SW1
2	27	SW2
3	26	SW3
4	25	SW4
5	24	SW5

6	23	SW6
7	20	SW7
8	19	SW8

Table 4.1: Slide Switches Connections

3.2 Push Buttons BTN1-BTN4

The Digilab board provides four push buttons which are normally connected to GND ("0") and which can be momentarily connected to Vdd ("1"). The buttons are connected to both to the FPGA pins and the pins of the J1 connector as indicated in the Table 4.2.

Push Button #	Pin # on FPGA	Pin on J1 Connector
1	59	BTN1
2	58	BTN2
3	57	BTN3
4	56	BTN4

 Table 4.2: Push Buttons Connections

3.3 LEDs

Eight LEDs are provided to display the value of output signals. The LED will light up when a "1" (Vdd) is applied to its input (anode) and will be off when a "0" (0V, low or GND) is applied. The signals can come both from the FPGA and from the pins on the J1 connector, as shown in the table below. One should be careful not to drive the LEDs from both the FPGA and the J1 connector.

	LED drive signal LD		Gate Sig	nal LDG
LED#	Pin # on FPGA	Pin J1	Pin # on FPGA	Pin J1
1	69	LD1		
2	68	LD2		
3	67	LD3		
4	66	LD4	70	LDG
5	65	LD5		
6	62	LD6		
7	61	LD7		
8	60	LD8		

 Table 4.3: LED Connections

3.4 Seven-Segment Displays

The Digilab provides four seven-segment displays that can be used to display data from the FPGA or from other circuits through the J1 connector. The segments of each display are called A, B, up to G as is standard. In order to

reduce the number of connections needed to address each of the LEDs, the anodes of the LEDs of each seven-segment display have been connected together. The common anode for the first seven-segment display is called A1; A2 for the second display, etc. In addition, the cathode pins from each display have been connected together to form seven common terminals, called A, B, C, D, E, F and G, corresponding to the seven-segments. Thus to illuminate a segment of a particular display, e.g. segment E of the third display, one has to apply a "1" (High or Vdd) to anode A3 and a "0" (GND or Low) to cathode node E. In addition to the seven segments LED, there is also a decimal point available (DP). However, the decimal point is not connected to the FPGA but to a pin of the J1 connector. If one needs to drive the decimal point, one can use a general-purpose output of the FPGA and connect this output to the DP pin on the J1 connector.

Seven-segment Terminal	Pin # of FPGA	Pin on J1 connector
A1 (Anode digit 1)	44	A1
A2 (Anode digit 2)	40	A2
A3 (Anode digit 3)	39	A3
A4 (Anode digit 3)	38	A4
A (Cathode of segment A)	51	CA
В	50	СВ
С	49	CC
D	48	CD
Е	47	CE
F	46	CF
G	45	CG
Dec. point	-	DP

 Table 4.4: Seven segment displays' connections

3.5 Clocks

The Digilab board has two clocks, CLK1 and CLK2. CLK1 clock is the general system clock that has been connected to pin 13 (PGCK1) of the FPGA. It provides a clock of 25.175MHz. CLK2 is connected to pin 35 (PGCK2) of the FPGA as well as to pin CLK2 on the J1 connector. The second clock can come in handy for peripheral devices that need a clock.

We would be using CLK2 to provide manual pulses in many experiments. For doing so, we would be connecting CLK2 pin with some push-button like BTN1.

Clock	Pin # on FPGA	Pin on J1 connector
CLK1 (25.175 MHz)	13 (PGCK1)	No Connection
CLK2	35 (PGCK2)	CLK2

 Table 4.5: Clock Connections

3.6 Parallel Port (J7)

The parallel port can be used for two purposes: data communication and for programming the FPGA. The choice between the two modes are set by setting switch SW9 in the "PORT" (for parallel data communication) or in the "PROG" (programming) position. We would mostly be concerned with programming through the parallel port.

The parallel port can be used to program the FPGA. Make sure that switch SW9 is in the "PROG" position. The Xilinx Project Manager will automatically detect the parallel cable and enable programming over this port. However, for the first time it may be required to manually set the cable type in the Hardware Debugger Window (more on it in Section xx4.0).

Detailed Pin Description of the Digilab Board

Shaded boxes represent dedicated pins that are not available for use.

Pin#	Function	Pin#	Function	Pin#	Function
1	GND	29	01	57	BTN3
2	VDD	30	M1_NC	58	BTN2
3	PWE	31	GND	59	BTN1
4	PD0	32	MODE	60	LD8
5	PD1	33	Vdd	61	LD7
6	PD2	34	M2_NC	62	LD6
7	PD3	35	CLK2	63	Vdd
8	PD4	36	O2	64	GND
9	PAS	37	O3	65	LD5
10	PRS	38	A4	66	LD4
11	Vdd	39	A3	67	LD3
12	GND	40	A2	68	LD2
13	CLK1	41	O4	69	LD1
14	PDS	42	Vdd	70	LDG
15	PWT	43	GND	71	O5
16	PD5	44	A1	72	RXD
17	PD7	45	CG	73	CCLK
18	PD6	46	CF	74	Vdd
19	SW8	47	CE	75	PINT
20	SW7	48	CD	76	GND
21	GND	49	CC	77	R
22	Vdd	50	CB	78	G
23	SW6	51	CA	79	В
24	SW5	52	GND	80	HS
25	SW4	53	DONE	81	VS
26	SW3	54	Vdd	82	PS2C
27	SW2	55	PROG	83	PS2D
28	SW1	56	BTN4	84	PINT

4 Design Download

Once you have successfully simulated and implemented the design along with proper pin assignments, it is time to check the design on board.

Before proceeding, make sure that

The Digilab board is connected with the PC using parallel port cable It is being supplied with power. That SW9 is in PROG mode That LED is glowing

Go in Start-Menu/Programs/Xilinx and run *Program* shortcut. The 'Program' shortcut runs Design Impact software of Xilinx 4.2. A screen as shown in Figure 5.1 appears.

🛃 Untitled [Configu	ration Mode] - iMPACT	_ 🗆 🗙
File Edit Mode Opera	ations Output View Help	
Baundany Scan	Operation Mode Selection	
Boundary Scan	What do you want to do first?	
	Configure Devices	
	C Prepare Configuration Files	
	C Load Configuration File (.cdf, .pdr)	
=>	< Back Next > Cancel Help	
<		2
For Help, press F1	Configuration Mode	

Figure 5.1: Design Impact - Initial Setup - 1

Select Configuration Devices

🛃 untitled [Configur	ation Mode] - iMPACT	
File Edit Mode Opera	tions Output View Help	
	onfigure Devices	
Boundary Scan	I want to configure device via : C Boundary-Scan Mode Slave Serial Mode C Select Map Mode C Desktop Configuration Mode	
Driver Version = 503. Cable connection establ CB_PROGRESS_END Elapsed time = 1 sec =>	< Back Finjsh Cancel Help	 ()
For Help, press F1	Configuration Mode	

Figure 5.2: Deign Impact - Initial Setup - 2

In the ensuing dialog box shown in Figure 5.2, select Slave Serial Mode.

The Design Impact will try to find the hardware board attached to it. If all the connections are made properly, it will detect the board and show the successful status in the status portion (the light blue box)

The Design-Impact will now ask for the design file to be loaded as shown in Figure 5.3. This is the file that is generated when implementing the design. The default extension of this file is .bit. It is saved in the folder where the design is saved. The name of the file is the name of the project. For e.g if *abc* is the project name, the design file to be loaded is **abc.bit**.

📑 untitled [Configuration Mode] - iMPACT	
File Edit Mode Operations Outpu	View Help	
	:※ ::: ↓ Ⅲ # # 0 \$ №	
Boundary Scan Slave Serial	Select Map Desktop Configuration	
Add Device	2 🔀 🤉	
Look in: 🗲)FPGALab 💽 🔶 💼 🖝 🖽 -	
Reviewed		
Unfinishe	i i i i i i i i i i i i i i i i i i i	
test.bit		
File name:	test.bit Open	
Files of tupe:	All Design Elles	
r iles or ope.		
Driver Version = 503. Cable connection established		<u>^</u>
CB_PROGRESS_END - End Operation		
Elapsed time = 1 sec.		-
<.		2
For Help, press F1	Configuration Mode Slave Serial	

Figure 5.3: Bit File Loading

Design Impact is ready for design downloading. The ready setup is shown in Figure 5.4. Right click on the device and click on download to download the design.



Figure 5.4: Downloading Design

Design download success is displayed as shown in Figure 5.5.



Figure 5.5: Successful Download