UNIVERSITY OF CALIFORNIA, RIVERSIDE
Department of Electrical Engineering
Department of Computer Science
Written by Hyung-Jun Kim in 1999
Modified by Enoch Hwang
Used and distributed with the author's permission

EE/CS 120A : Logic Design Laboratory

Instructor: Dr. Enoch Hwang
Office: A303 Bourns Hall
Email: ehwang@cs.ucr.edu
Homepage: www.cs.ucr.edu/~ehwang

Lab Overview

It involves design, assembly and test of combinational and sequential logic circuits. Logic designs will be done using simple CAD tools and implemented using field-programmable gate arrays (FPGA). In the laboratory digital logic circuits will be designed and implemented using the Foundation Series Tools and FPGAs from Xilinx, Inc.

Grading Policy

Grading will be based on laboratory reports and in-lab performance (i.e., quizzes and laboratory-related questions). TAs and/or the instructor will ask pertinent questions to individual members of a group at random. The tentative grading policy is:

- Pre-lab Work: 20%
- In-lab Performance: 40%
- Laboratory Report: 40%

Lab Instructions

- Seven lab experiments and Projects are to be conducted, in addition to the lab orientation. Groups of 2 students will be formed. The lab report, *written individually*, is due one or two weeks after the day of the experiment at the *beginning* of the next lab. Every person must submit their own lab report for each lab *before* each session.

- You are excepted to read the lab description, and complete the pre-lab *before* each session. Read each week's assignment and complete any pre-lab calculations and/or design work *before* coming to the lab. Try to understand the objectives of the experiment and determine what information you should record in your laboratory notebook. If you have any questions, please see the instructor or your TA before the scheduled time of your lab.

- Be prepared to show your pre-lab work to your TA. If you have VHDL code, schematic and wiring diagrams ready before you come to lab, you will be able to finish quite easily during the lab time. If you are not prepared, you may not be able to finish during the lab period and consequently, your grade will suffer. Each lab session lasts three hours. TA or instructor is not required to give you extra time to complete your work.

- Reports that appear to be copied will be returned ungraded. No late report will be accepted.

Lab Reports

Please use the following format for your lab reports:

- *Cover:* Includes course number, topic of the experiment, section number, names of members, and your name and student id number.

- *Abstract:* Brief introduction to the experiment and a summary of results.

- *Analysis:* Design techniques, implementation details, e.g., K-maps, boolean equations, schematics, etc.

- *Records:* Simulation results, VHDL input and output, FPGA pin-maps, interconnections, etc.

- *Discussion:* Comments, circuit debugging and problem solving issues, and conclusions.

EE/CS 120A : Logic Design
Lab 1: Up/Down Counter with 7-segment display

Design Problems

In this lab, you will construct an Up/Down counter with 7-segment display as shown in Figure1.

The MC14029B Binary/Decade up/down counter consists of D-type flip-flops with a gating structure to provide toggle flip-flop capability. The counter can be used in either Binary or BCD operation. It also can be used either as a Down-counter (when you connect pin 10 to ground) or as an Up-counter (when you connect pin 10 to 5V) as shown in Figure 1.

The MC14511B BCD-to-seven segment decoder also has an output drive capability.
Figures 2 and 3 show the detail logic diagrams of the MC14029B (Binary/Decade Up/Down counter) and MC1451B (BCD-to-Seven segment Latch/Decoder/Driver), respectively. You may need the circuit for the final project.

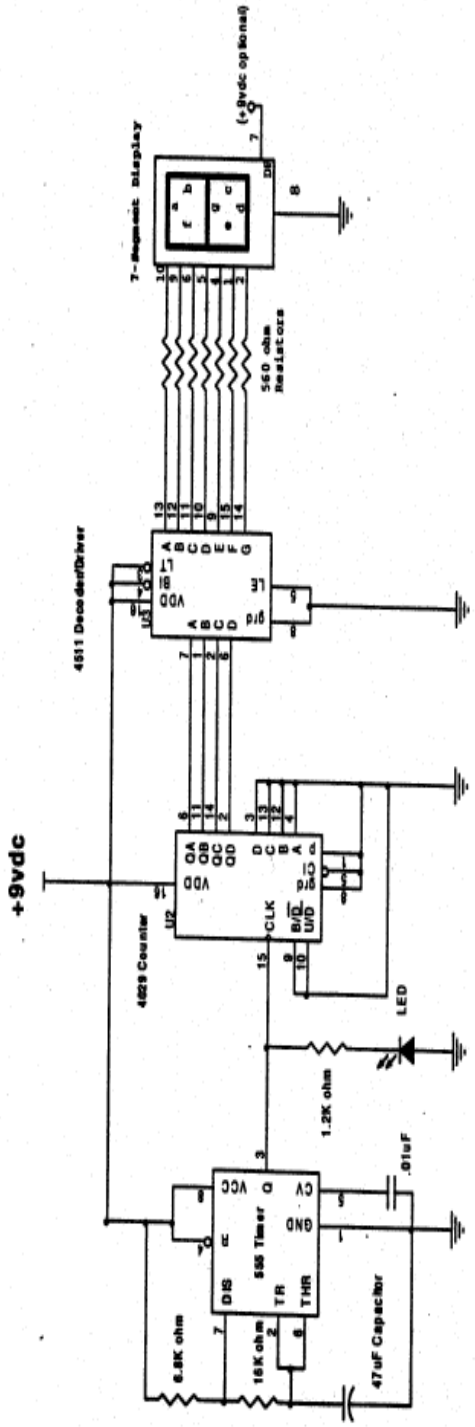


Figure 1: Up/Down counter with 7-segment display

4

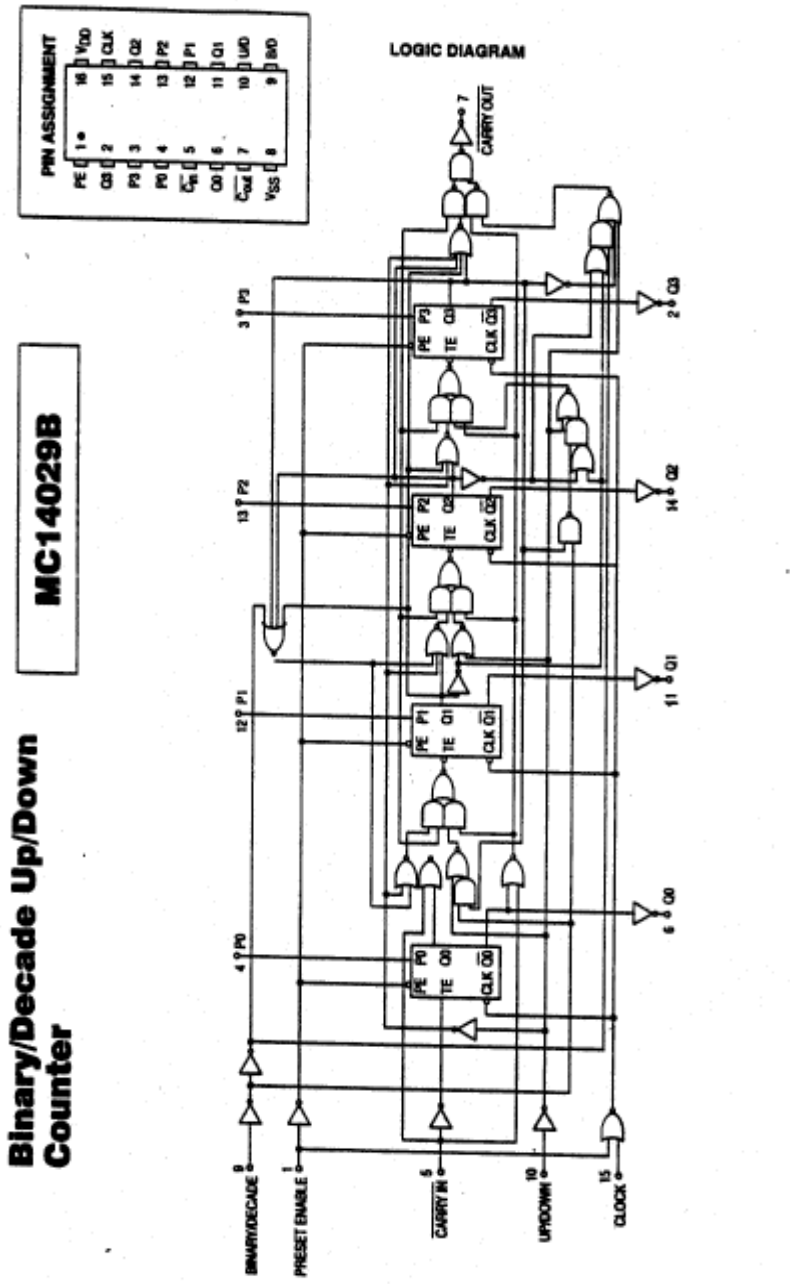
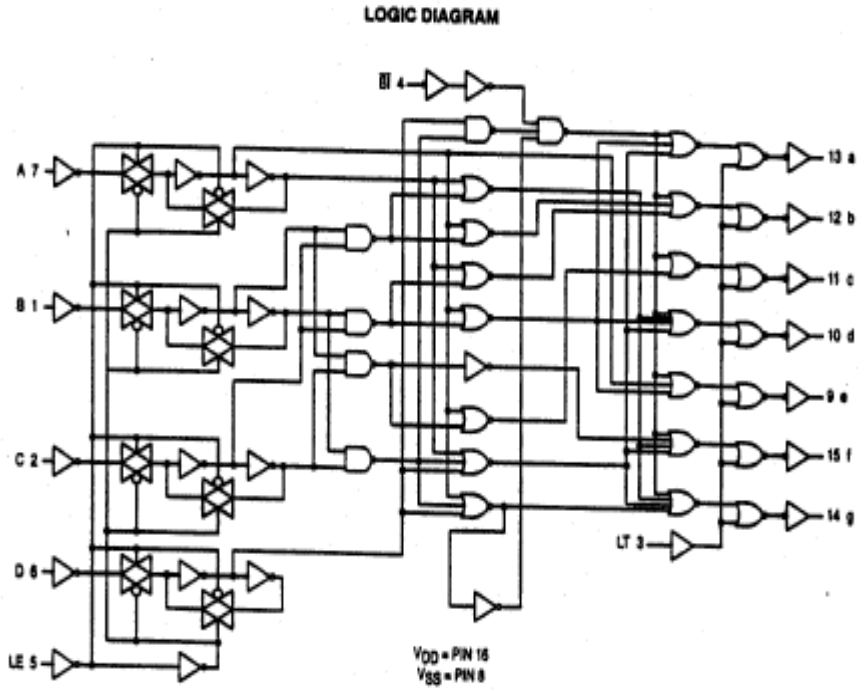


Figure 2: Logic diagram of MC14029B(Binary/Decade Up/Down counter)



BCD-To-Seven Segment Latch/Decoder/Driver

MC14511B

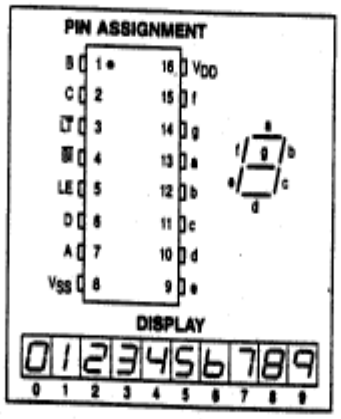


Figure 3: Logic diagram of MC14511B(BCD-to-Seven segment Latch/Decoder/Driver)

EE/CS 120A : Logic Design
Lab 2: Combinational Logic Design Using Xilinx Foundation Tools

Objectives

The objective of this laboratory assignment is:

- To get familiar with the Xilinx Foundation Series Tools to design logic circuits.
- To design and implement simple combinational logic circuits using Schematic editor and simulator.

Laboratory Instructions

- Create a directory with your name on the C drive of your lab PC. Use this directory to create your project, store your results, bitstreams, etc. during the lab session.
- You can bring complete project files on a floppy disk and then use the **Copy Project** command from the Project Manager menu to copy it into the directory you created above.
- Alternatively, you can create a new project in your directory on the C drive and then copy your files to that new project directory. Remember to **Add** your .SCH file to the project.
- Perform functional simulation of your design and have it checked by the lab instructor or your TA.
- In case you modify your source file, remember to copy it back to your floppy disk.
- Test and demonstrate your circuit to the lab instructor or your TA.
- Before you leave the lab please **remove** the files and directories that you created on your lab PC and leave your workplace clean and tidy.

1. Double-click on the **Xilinx Foundation Project Manager** icon.

2. You will get the **Getting Started** window. Click on **Create a New Project** and then click OK. You will get the **New Project** window. Enter your project name in the box **Name**: and click on Schematic, and then choose **XC9500** (lower left device box). Click OK after this.

3. You will see a screen that shows you the directory and some default files that have been created. There are three main panes in the Project Manager window. On the upper right-hand pane, you can find a box - **Design Entry**. Click on the rightmost icon which has the shape of an AND gate (icon for the schematic editor). You now see the schematic editor in a new window. This is where you will draw your schematic.

4. The first thing you need to do is add the gates for your project. On the left side of the screen you will see an icon in the form of an AND gate - called a symbol toolbox. Click on this and the **SC Symbols** window will appear with a list of all types of components you can use. You will get a list of components in the form of a drop down menu or alternately you can go to **Mode** $\rightarrow$ **Symbols** menu item on the options bar on the top and click on symbols to get the same menu. You now have to only type the name of the component that you need. For a 2-input AND gate case, an AND gate at the bottom of the menu or scroll down through the list in the components window till you see an AND2 (2-input AND gate) and select it from the list. You then simply move the mouse and position it to wherever you require it to be placed. To position the component simply left click on the mouse. You can attach another copy of the AND gate to your cursor by simply clicking on the AND gate you just dropped. Then you can drop the new AND gate in the schematic as well. In a similar manner you can get all the components that you require for your circuit. You are now ready to connect the gates for the circuit.

5. You now need to add input and output buffers to the circuit. This is commonly referred to as adding **IBUFs** and **OBUFs** (short forms for Input Buffer and Output Buffer, respectively). These buffers indicate that the signals attached to them will actually enter and exit the FPLD chip via the I/O pins. Select IBUF for input and OBUF for output from the symbol table. Add the buffer and give it the same name as the input (or output) signal. When this is done we add **Terminals**. To do this click on the terminal button on the upper left corner of the toolbar in the **SC symbols** window in which you can type the **Terminal Name** and **Terminal Type** of each input and output, e.g. terminal name is A and type is INPUT (you can select input or output from the drop down box). Click OK and an input (or output) terminal will be attached to your cursor. Simply click on the mouse to drop the terminal into your schematic. Then add all other inputs and outputs in the same manner. The terminal names must be unique. Make sure that IBUF's and OBUF's are between the terminals and the logic gates.

6. At this point we have all the components we need so you can double-click on the upper left corner of the **SC Symbols** window to get rid of it.

7. You might at some point want to rotate the component. Go into **Mode → Select and Drag** menu item. Select the component you want to rotate with the mouse. Right click on the mouse and select **Symbol Properties** from the pop up menu, which appears. A new window will appear. Click on **Attributes**. You will get a menu wherein you can rotate the component by the required degrees. Select the same and click OK.

8. The next step is to connect the gates. Select the **Mode → Draw Wires** menu item to begin the process. Say you want to connect the output of the AND gate to the input of the OR gate. Click on the output of the AND gate followed by clicking on the say upper input of the OR gate. Click on the output of the AND gate followed by clicking on the say upper input of the OR gate. A line will appear connecting the output to the input. You can continue in the manner until all the gates are connected as required by the Boolean equation.

9. Now that the schematic is done, we need to check it for errors. First, select **Options → Create Netlist**. This will check your schematic drawing and generates a machine-readable netlist, which describes what types of gates are used and how they are the connected. Next, select **Options → Integrity Test** to initiate an error check on the netlist. If the netlist has no errors, save the schematic using **File → Save As**… menu item. Now we must export the netlist. Go to the option **Options → Export Netlist** and click on it. An Export Netlist window will appear. Select Edit 200 [*.EDN] in the **Netlist Format** selection box. Click on the **Open** button. Now select **File → Exit** to close the schematic editor. On returning to the **Project Manager** we must make the filename (your file) .SCH as part of the project. Select the **Document → Add**… menu item and list items of type Schematic (*.SCH) in the dialog window. Highlight your file and click on OK. You should see your file as part of the project.

10. We are now ready for simulation. Click on the visible icon called **Simulation** at the Project Manager. This will bring up the **Logic Simulator Foundation Window** and a single, empty **Waveform Viewer** window.

11. The first thing to do is add the inputs and outputs of the circuit to the **Waveform Viewer** so that we can see what is happening as the circuit is simulated. Do this by selecting **Signal → Add Signals**… menu item. The **Component Selection for Waveform Viewer** window will appear. Click on your input name (say "A") to highlight it and then click on the **Add** button. A waveform labeled "A" will appear in the waveform viewer and a red checkmark will appear by the selected signal. We can repeat this procedure for all the inputs. Similarly do the same thing for all the outputs. Then click on the **Close** button.

12. Now the inputs and outputs are displayed, but nothing interesting is happening because all the inputs are set to logic 0. Now we need to apply a stimulus to the circuit, so naturally we select the **Signal → Add Stimulators**… menu item. This brings up the **Stimulator Selection** window. There are many number of buttons, but we are only interested in a single item: a 16-bit binary counter labeled **Bc**.

13. We now have to do a one to one mapping of all input signals to this circuit. During a simulation, the right-most 4 bits of this counter will go through every possible combination of inputs, from 0000 to 1111. Starting from the lowest input possible, we label that to the right-most bit of the counter by clicking on one of the bit circles. Do this by clicking on the name of an input in the **Waveform Viewer** window (the selected input is highlighted) and then clicking on one of the bit-circles in the **Bc** section of the **Simulator Section** window. The label of the counter bit attached to the circuit input will appear to the right of the input name in the waveform viewer. Once all the inputs are attached to the counter bits, we can click on **Close** to leave the **Simulator Selection** window. We now need to set up a parameter that controls the speed of the simulation. First, select the **Options → Preferences** menu item.

Then in the **Preference** window, set the frequency of the "BO" bit of the binary counter at 50 MHz. Then click on OK. Now you can run the simulation. In case you want to run the simulation again, delete the previous waveforms using **Waveform → Delete** menu.

14. We are now ready for simulation. Set the simulation mode to **Functional** in the drop-down menu in Logic Simulator window tool-bar. This indicates we are doing a functional simulation that checks only the logical operation of our circuit and ignores detailed physical simulation timing issues. You could go for **Step by Step simulation** or **Long simulation**. Step by Step simulation is done by pressing the Step simulator waveform on the menu bar at the top of the screen. This simulates the circuit for each clock pulse. Alternatively we could go for long simulation which simulates the circuit for a fixed period of time (time period decided by you) and display the results. To do this go to **Goto options→Start Long simulation** and set the running time to 1 sec. Click on start. This will test your circuit for 1 second and the waveform will appear on the screen. Check the waveform for all the input combinations and make sure that the output is correct for all conditions. Similarly check the other circuits and show the grader the complete *one cycle of waveforms* for all the required circuits. **Caution**: When you print the waveforms, check the **Page Setup** first. You have to select "Current Page" option before you send any file to the printer. To change the size of the schematic output, check the **Print option**.

15. When you finish the experiment, save your files on your diskette and delete all your files when you leave the lab. Failure to do so will result in marking down of the points for that lab.

Design Problems

Using the Xilinx Foundation Series Tools design, test and demonstrate circuits, which implement the following functions.

1. f(a,b,c,) = abc' + a'bc + a'b'c'

2. f(x,y,z) = y · z + x' · y · z' + x (y' · z + y · z')

3. f(w,x,y,z)= Σ(1,2,4,5,9,11,12,13,14,15)

4. f(w,x,y,z)= Π(1,2,6,7,8,10)

5. Design a circuit for the following truth table:

x	y	z	m
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

6. Design a circuit with four inputs (a,b,c,d) and one output (x) that produces a "1" at the output if there is an equal number of 0's and 1's at the inputs.

EE/CS 120A : Logic Design
Lab 3 : Digital Logic Gates

Objectives

The objectives of this laboratory assignment are:

- To get familiar with the Xilinx Foundation Series Tools to design logic circuits.
- To design and implement full adder and full subtractor circuits using Schematic editor and simulator.
- To download your circuit on the prototype board and test it.

Laboratory Instructions

- Create a directory with your name on the drive C of your lab PC. Use this directory to create your project, store your results, bitstreams, ect. during the lab sessions.
- You can bring complete project files on a floppy disk and then use the **Copy Project** command from the Project Manager menu copy it into the directory you created above.
- Alternatively, you can create a new project in your directory on the C drive and then copy your file to that new project directory. Remember to **Add** your .SCH file to the project.
- Perform functional simulation of your design and have it checked by the lab instructor or your TA.
- In case you modify your source file, remember to copy it back to your floppy disk.
- Test and demonstrate your circuits to the lab instructor or your TA.
- Before you leave the lab please **remove** and files or directories you created on your lab PC and leave your workplace at least as clean and tidy as you found it.

EE/CS 120A : Logic Design
Instruction to download the schematic to the board

1. Open a new project - one project for each of the files you will be working on.

2. Perform a functional simulation that checks the logical operation of your circuit. Check the waveform for all the input combinations and make sure that the output is correct for all conditions. If the output is correct, print your schematic and functional analysis output files. You have to select "Current Page" option before you send any file to the printer.

3. Open you schematic file. Delete the input and output terminals. Instead, add **IPAD**s (inputs) and **OPAD**s (outputs) in their places.

4. Now, you will have to map the input and output pins on the board to these pads. For this, select the components (IPAD or OPAD) and the click on the right mouse button. You will see a popup menu. Select symbol properties. You come to a square dialog box. In the space for name, type in capital letters, **LOC**, which stands for location. In the space for description, type the corresponding pin number, say p46 to an IPAD. Then, click add. You will see that the pin number you specified is added to the corresponding IPAD or OPAD. Do this for all inputs and outputs using the following tables which specify the input and output pin numbers. For example, $X_i \rightarrow$p48, $Y_i \rightarrow$p47, $C_i \rightarrow$p46, $C_{i+1} \rightarrow$p19, and $S_i \rightarrow$p23.

XC95108 pin	Xsport
p46	B0
p47	B1
p48	B2
p50	B3
p51	B4
p52	B5
p81	B6
p80	B7

XC95108 pin	LED seg
p21	S0
p23	S1
p19	S2
p17	S3
p18	S4
p14	S5
p15	S6
p24	S7

5. Save the file. Now, go back to the opening screen. You are ready for implementation.

6. Click on the **Implementation Design** in the dialog box. Select device name **95108PC84**. Then, click run. If an error message appears on the screen, then you need to find out the cause of the error. Click on verification and find out the cause of your error. Rectify it.

7. Now, you are ready for downloading the schematic on to the board. Click on **Programming**. You will come to a screen **JTAG Programmer**. This window lets us produce a stream of bits that is suitable for programming a set of XC9500 chips. Select the **Output → Create**

SVF File… menu item. Then an **SVF Options** pop-up window will appear. Select Through Test-Logic-Reset, then click on OK.

8. Do not save the file in the rev (revision) directory that appears on the popup box but instead to save the **.svf** file we have to go into the **C:\XSTOOLS\BIN** directory and save the file there as our executable format will be built there. Therefore go to that directory and save the file as **yourfilename.svf**.

9. Now turn on the power for the XS prototype board. Make sure that the power supply is in the **+6V** range and adjust the voltage to about 6V.

10. Select **Operation → Get Device ID** and then click on OK.

11. In the JTAG popup window that you are in go to **Operations → Program** and select "Erase Before Prog" option. In this step erase the program on the board.

12. As the board has a ROM you probably will need to erase the previous program stored on the board. Go to **Start → Run** and type **C:\XSTOOLS\BIN\Xstest.batxs95-108**. This will clear the old program from the board.

13. You are now ready to download your program. Go to **Start → Run** and type **C:\XSTOOLS\BIN\Xsload.exe yourfilename.svf.**

14. To test the program, we test it for all possible combinations of the input. For example, if we have 3 inputs the test vector shall range from 000 to 111. To test for each condition go to **Start → Run** and type **C:\XSTOOLS\BIN\Xsport.exe yourtestvector** where you have to manually change your test vector from 000 to 111 for a 3 input circuit and so on. The corresponding output will be displayed on the LED on the board. For 4 inputs case, you have to change the test vector from 0000 to 1111.

15. When you are ready for the experiment, let the TA check your results.

16. When you finish your lab experiment, delete all the .svf files under the BIN directory.

Design Problems

Using the Xilinx Foundation Series Schematic-based Tools design, test and demonstrate circuits based on the specifications given in the following table.

x_i	y_i	c_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

1. Using AND, OR, and XOR gates design the full adder that is specified by the above table.
2. Using NAND and OR gates design the full adder that is specified by the above table.
3. Using multiple-input NAND gates design the full adder that is specified by the above table.

x_i	y_i	b_i	b_{i+1}	d_I
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

4. Using the basic logic library implement the full subtractor that is specified by the above table.

5. Download your circuit on the prototype board and test it. You need to use one of the three Adder and Subtractor circuits you had designed. The output will be seen in the seven segments display and your input will be given via the keyboard to the input pins specified. There are the steps that you will be going through.

EE/CS 120A : Logic Design
Lab 4: Multiplexer and Decoder

Objectives

The objectives of this laboratory assignment are:

- To get familiar with the Xilinx Foundation Series Tools to design logic circuits.
- To design and implement Multiplexer and Decoder circuits using Schematic editor and simulator.
- To download your circuit on the prototype board and test it.

Laboratory Instructions

- Create a directory with your name on the drive C of your lab PC. Use this directory to create your project, store your results, bitstreams, etc. during the lab session.
- You can bring complete project files on a floppy disk and then use the **Copy Project** command from the Project Manager menu to copy it into the directory you created above.
- Alternatively, you can create a new project in your directory on the C drive and then copy your file to that new project directory. Remember to **Add** your .SCH file to the project.
- Perform functional simulation of your design and have it checked by the lab instructor or your TA.
- In case you modify your source file, remember to copy it back to your floppy disk.
- Test and demonstrate your circuit to the lab instructor or your TA.
- Before you leave the lab please **remove** any files or directories you created on the your lab PC and leave your workplace at least as clean and tidy as you found it.

Design Problems

Using the Xilinx Foundation Series Schematic-based Tools design, test and demonstrate circuits. Download your circuit on the prototype board and test it.

1. Use as few 4-to-1 multiplexer modules as possible to design a circuit for the following function. Do not use any other logic gates.
   $f(w,x,y,z) = \Sigma(1,3,5,12,15)$

2. Design a 7-segment decoder that has a 4-bit, hexadecimal input ($I_3 I_2 I_1 I_0$) and seven outputs ($S_0, S_1, S_2, S_3, S_4, S_5, S_6$) one for each segment of the display. Your decoder should be designed to use the 7-segment display on your prototype board and should be capable of decoding all hexadecimal inputs.

EE/CS 120A : Logic Design
Lab 5 : Hierarchical Combinational Design Using MSI Macros

Laboratory Instructions

- Create the VHDL source file(s) for your designs before coming to the lab. You can use the Xilinx HDL editor or any text editor to create your files. Remember to bring these files to the lab on a floppy disk.

- Create a directory with your name on the drive C of your lab PC. Use this directory to create your project, store your results, bitsteams, ect. during the lab session.

- You can bring a complete project (i.e. *project.pdf* file and *project* directory) on a floppy disk and then use the **Copy Project** command from the Project Manager menu to copy it into the directory you created above.

- Alternatively, you create a new project in your directory on the C drive and then copy your VHDL to that new project directory. Remember to *Add* your VHDL file to the project.

- Perform functional simulation of your design and have it checked by the lab instructor or your TA.

- In case you modify your VHDL source file, remember copy it back to your floppy disk.

- If the circuit works as excepted, implement it using the prototyping board assigned to you.

- Use keyboard and LEDs available to apply input stimuli and observe the outputs. Disconnect the XSPORT (parallel port) when you apply input stimulus from the workbench.

- Test and demonstrate your circuit to the lab instructor or your TA.

- Before you leave the lab please remove any files or directories you created on your lab PC and leave your workplace at least as clean and tidy as you found it.

EE/CS 120A : Logic Design
Xilinx VHDL Simulation Procedures

VHDL stands for "VHSIC Hardware Description Language." VHSIC, in turn, stands for "Very High Speed Integrated Circuit," which was a U.S. Department of Defense program to encourage research on high-performance IC technology.

The only real difference between the schematic and VHDL design flows is the method by which the netlist is extracted from the design description. Therefore, this procedure will only discuss the steps of entering the VHDL code for the circuit design and the synthesis of the netlist from the VHDL code. The steps which follow (functional simulation, compiling into a bitstream, downloading the bit stream to an XS board, and testing the downloaded design) are performed exactly as they were in the schematic design flow discussed in the previous section.

1. Double-click on the **Xilinx Foundation Project Manager** icon.

2. You will get a screen **Getting Started**. Click on **Create a New Project** and then click OK. You will get a screen - **New Project**. Enter your project name in the box **Name:** and click on HDL. Click OK after this.

3. You will come to a screen that shows you the directory and some default files that have been created. There are three main panes in the Project Manager window. On the upper right-hand pane, you can find a box-**Design Entry**. Click on the leftmost icon, which has the shape of a paper with HDL (icon for the HDL editor). You now see the HDL editor in a new window.

4. In the window that appears, click on **Use HDL Design Wizard** and then click on OK.

5. Then click on Next in the **Design Wizard** window. Here, select the VHDL and click Next to move to the **Design Wizard-Name** window. Enter your file name and click Next.

6. Now you should be in the **Design Wizard - Ports** window where you specify inputs and outputs for your design. To add an input, click on New and then type your input variable in the Name field. Click on Input to see the port to be an input. Repeat these steps for all your inputs. To add your output of your circuit, click on New, enter your output variable and click on the Output. Repeat these steps for all your outputs. In the case your input or output is a vector, you can use Bus option. Now all the inputs and outputs are defined so click on the Finish.

7. If you are familiar with HDL editor, you can click on the **Advanced** button to bring up the **Advance Port Settings** window. The drop-down menu in this window lets you set the type of any of the inputs or outputs. The default setting is std-logic.

8. At this point, an **HDL Editor** window will appear with the skeleton program. The first line of the file uses the LIBRARY keyword followed by the names of the libraries you want to use in your design. Libraries are used to encapsulate functions

that are generally useful in a wide variety of designs. You can access the macros, definitions, and functions from the IEEE library. A library can be subdivided into package, which further encapsulate features useful in a certain area of type of application. The USE keyword in line 2 indicates that our design will have access to ALL the features found in the std-logic-1164 package of the IEEE library. The IEEE library and the std_logic_1164 package are standards, which are support by the VHDL tools.

9. Following the library access control lines define the interface to your circuit. A VHDL entity is simply a declaration of a module's inputs and outputs, while a VHDL architecture is a detailed description of the module's internal structure or behavior. The interface declares the inputs and outputs which an external circuit can use to gain access to the features and functions of the circuit.

10. You will use the std_logic type which allows logic signals to take on the standard 1 and 0 Boolean states as well as the undefined, high-impedance, and other states.

11. The interface definition is followed by an architecture definition. The VHDL statement in the architecture section describe how the circuit actually carries out the operations on the input/output values passed through the interface. You should enter your statement here. Like other high-level programming languages, VHDL generally ignores spaces and line breaks, and these may be provided as desired for readability. Comments being with two hyphens and end at the end of a line.

12. Now we have to check to make sure we have not made any mistakes. Select the **Synthesis → Check Syntax** menu item. A small window below the HDL editor will appear informing you that the VHDL code is being examined for errors. Within a few seconds, it will state Check Successful if there is no error. Click OK in the pop-up window. If there are errors, each error will be highlighted in the **HDL Editor** window and an error message will appear at the bottom of the window. You can get error-free VHDL code examples for reference by selecting the **Tools → Language Assistant** menu item.

13. VHDL is not synthesized within the HDL Editor window so the **Synthesis → Synthesize** menu item is blanked out.

14. Now that the design entry is complete, select **File → Save** in the **HDL Editor** window. Then select **File → Exit**.

15. Upon returning to the **Project Manager** window, you must make yourfilename.VHD file a part of your project. Select the **Document → Add…** menu item and list items of type **HDL** (yourfilename.VHD) in the dialog window. You should then see your VHDL file under your project name.

16. Now, you have to map your input and output variables to the input and output pins on the board. For this , you should double click on the .ucf file under your project name. You will see **Report Browser** window. Go down to the end of the file and type "NET your *input or output* LOC=p*number*,". For example, if you use "A" for your input (MSD), then type "NET A LOC=p48,". Add all other inputs and outputs in the same manner. Now the pin assignment is complete, select **File → Save** and the **File → Exit**.

17. Once the VHDL source file complete, you next need to extract its netlist. Select the **Synthesis → Force Analysis of ALL HDL Source Files** in the **Project Manger** window. This indicates a check of all the VHDL files to detect any errors. If there is no error, a green checkmark will appear by the .VHD file name under your project name.

18. Next, click on the SYNTHESIS button in the right-hand pane of the **Project Manager** window. This brings up the **Synthesis/ Implementation** window.

19. When the window first appears, the name of the interface for your file will be listed in the Top Level text box. The Version Name box shows ver 1 and this will be incremented each time you change the source code for the design.

20. In the Target Device are of the window, you will select the family, particular device type, and device speed in the drop-down menus. This lets the synthesis software know the type of chip architecture you are targeting so it can generate a netlist that takes advantage of the features of the chip. Select Family name **XC9500** and Device name **95108PC84**.

21. You can also use the controls in the Synthesis Settings are to direct the synthesis tools to emphasize high-speed or area-efficient circuitry. There is also an Insert I/O Pads checkbox, which controls whether input and output buffers will be placed on all I/O signals. This box should be checked if your .VHD file is at the top level of design. (This box would not be checked if the .VHD circuitry is included as a macro in a larger design.)

22. Clicking on the Run button starts the synthesis process. If the synthesis is successful, you should see a green mark in the SYTHESIS box of the Flow tab in the **Project Manager** window. You will see a red cross in event of a failure.

23. At this point, you have extracted a netlist from the VHDL code that describes your circuit. With this netlist you can do functional simulation, compile the netlist into a bitstream, and download and test the bitstream to an XS95 Board in exactly the same way as was shown in the previous lab on the schematic design.

24. Test a functional simulation.

25. Download your circuit on the prototype board and test it.

Design Problems

Using the Xilinx Foundation Series Tools design using VHDL, test and demonstrate circuits, which implement the following functions. Your circuits should be as small as possible.

1. Design the full adder using VHDL.

2. Design a 4-to-1 multiplexer as a module that can later be used as a building block for other circuits.

3. Use as few 4-to-1 multiplexer modules as possible design a circuit for the following function. Do not use any other logic gates.
   $F(w,x,y,z) = \Sigma(1,3,5,11,15)$

4. Design a 7-segment decoder that has a 4-bit, hexadecimal input ($I_3I_2I_1I_0$) and seven outputs (S_0, S_1, S_2, S_3, S_4, S_5, S_6) one for each segment of display. Your decoder should be designed to use the 7-segment display on your prototype board and should be capable of decoding all hexadecimal inputs.

EE/CS 120A : Logic Design
Lab 6 : Arithmetic and Logic Unit Design

Design Problems

Using the Xilinx Foundation Series Tools design using VHDL, test and demonstrate circuits, which implement the following functions. Your circuits should be as small as possible.

1. Design a 1-bit full adder module.

2. Design a 3-bit arithmetic unit using 2's complement number representation. The arithmetic unit has two 3-bit inputs ($A_{2\text{-}0}$, $B_{2\text{-}0}$), a mode input (m), and produce a 3-bit result ($S_{2\text{-}0}$), and a 1-bit overflow flag, *OVFL*. The circuit should perform 2's complement addition ($A_{2\text{-}0} + B_{2\text{-}0}$) when the module input is 1 and 2's complement subtraction ($A_{2\text{-}0} - B_{2\text{-}0}$) when the mode input is 0. Use the 7-segment decoder designed in Lab 5 as a module in this design to decode and display the output using the on-board 7-segment display.

3. Enhance the capability of the arithmetic unit designed above by making it perform bit-wise logical operations, *AND* and *OR* on its inputs *A* and *B*. This new circuit is called *Arithmetic* and *Logic Unit* (*ALU*). This ALU circuit has a 2-bit mode input ($m_{1\text{-}0}$) besides all the other inputs and outputs described above (Note: for logical operation output flag *OVFL* is ignored). The table below gives a list of functions performed by the ALU based on the mode input. Use the 7-segement decoder designed in Lab 5 as a module in this design to decode and display output using the on-board 7-segment display.

m_1	m_0	Function
0	0	bit-wise OR
0	1	bit-wise AND
1	0	Addition
1	1	Subtraction

EE/CS 120A : Logic Design
Lab 7 : Traffic Light Controller

Design Problems

Design, using the Xilinx VHDL editor, a street-intersection traffic light controller. Both traffic light units have three lights: Red (stop), Yellow (prepare to stop) and Green (go). Each unit should be triggered by either of the two inputs representing cars or pedestrians waiting at a red light. Before one unit changes from red to green, the other unit must change from green to yellow to red.

Due to I/O limits on the XS95 board, you may simulate one set of the red, yellow, and green pattern by lighting segments on the on-board display. (Segment #S2&S5 = G, S0&S6=Y, S1&S4=R). The other set of lights should be connected to the Red, Yellow and Green LED's. Use the Dip switch for the four inputs.

- Download and demonstrate to the TA.
- Conclusion (Write-up).
- How well did your design perform?
- Explain your choice of Flip-flops.
- What changes and/or improvements could you foresee to make your design closer to an actual traffic control system?

EE/CS 120A : Logic Design

Project 1: Simplified Vending Machine Control

Design, using the Xilinx Schematic and VHDL editors, a simplified vending machine. Assume that 5 and 10-cent coins are sequentially deposited in the machine slot until totaling 25-cents. When 25-cents has been totaled, the dispense signal is turned to one from zero. If 25-cents is exceeded, the dispense signal and the change signal will turn from zero to one. Output these signals to light separate LED's on your breadboard. Display the current number of 5-cents deposited on the on-board 7-segment display.

- Download and demonstrate to the TA by trying at least 6 different input sequences of 5 and 10-cent coins.
- Conclusion (Write-up).
- How well did your design perform?
- Explain your choice of Flip-flops.
- What changes and/or improvements could you foresee to make your design closer to an actual vending machine control?
- Extra credit: Add product selection input and two digit 7-segment display. Display total amount deposited.

EE/CS 120A : Logic Design

Project 2: Up/Down Counter

Design, using the Xilinx Schematic and VHDL editors, an up/down counter with clock (approx. 5Hz) and output to a 7-segment display. (You can use the 7-segment decoder designed earlier.) You can use the 12MHz clock on the XS95 (pin 9) as a source for your 5 Hz clock. You may want to test this in stages, substituting some of the hardware from your breadboard circuit in Lab 1.

- Download and demonstrate to the TA.
- Conclusion (Write-up).
- How well did your design perform?
- How did you derive the 5 Hz clock?
- Extra Credit: Add second digit using another 7-segment display.

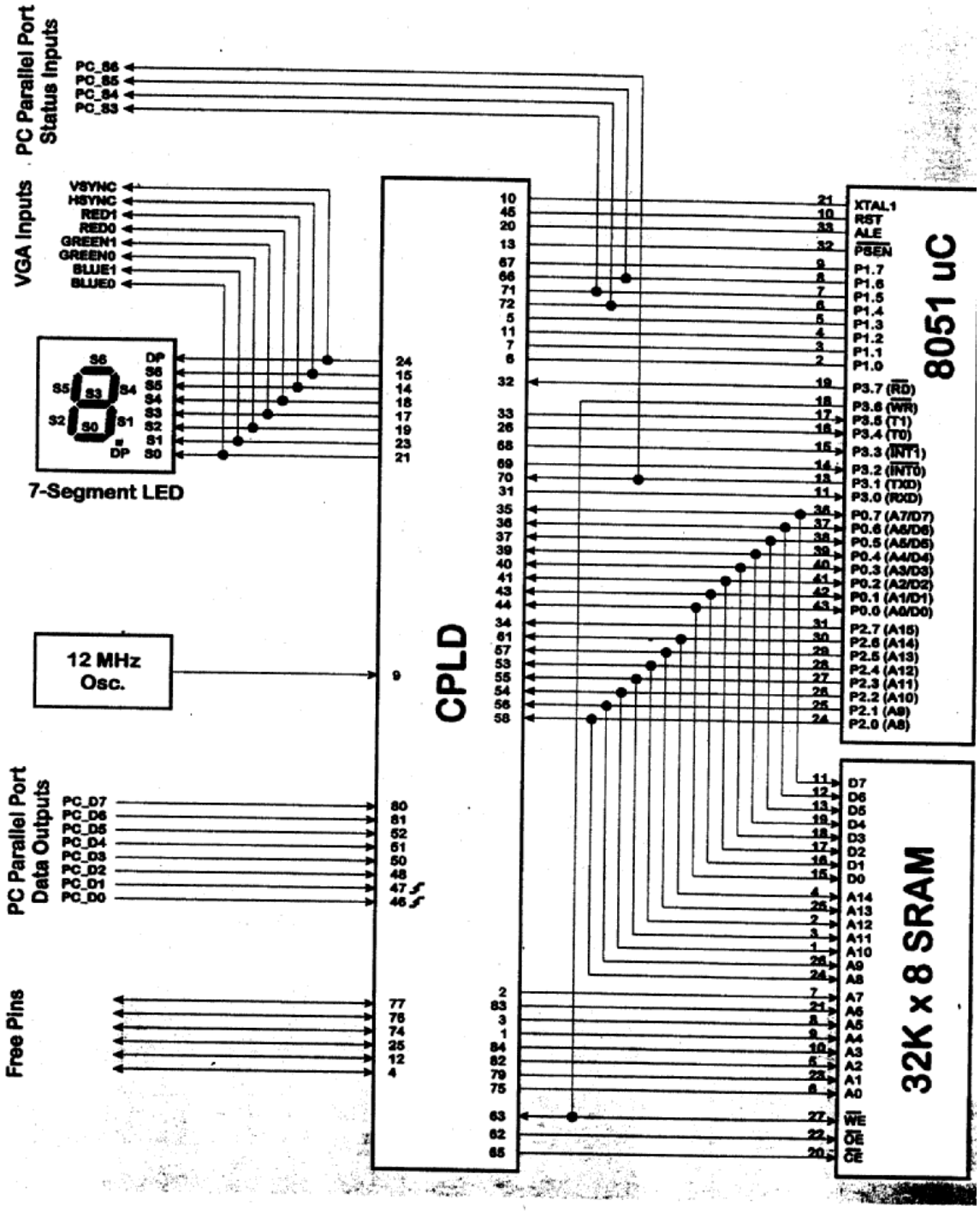
EE/CS 120A : Logic Design

Project 3 : Combination Lock

Design, using the Xilinx Schematic and VHDL editors, a combination lock that will display the input setting on the 7-segment display and an unlock signal to an LED when the correct code is set on the input Dip switch. Use a 4-bit BCD input code. Set the lock code to be 0101.

- Download and demonstrate to the TA.
- Conclusion (Write-up).
- How well did your design perform?
- Extra credit: Make the combination code programmable using the Xsport utility and make the display two decimal places using 8-bit input.

APPENDIX



XS95 CPLD Board Programmer's Model

XS95 Pin	Connects to...	Description
21	S0,BLUE0	These pins drive the individual segments of the LED display (S0-S6 and DP). They also drive the color, horizontal, and vertical sync signals for a VGA monitor.
23	S1,BLUE1	
19	S2,GREEN0	
17	S3,GREEN1	
18	S4,RED0	
14	S5,RED1	
15	S6,HSYNCB	
24	DP,VSYNCB	
9	CLK	An input driven by the 12 MHz oscillator.
46	PC_D0	These pins are driven by the data output pins of the PC parallel port. Clocking signals can only be reliably applied through pins 46 and 47 since these have additional hysterisis circuitry.
47	PC_D1	
48	PC_D2	
50	PC_D3	
51	PC_D4	
52	PC_D5	
81	PC_D6	
80	PC_D7	
10	XTAL1	Pin that drives the uC clock input
45	RST	Pin that drives the uC reset input
20	ALEB	Pin that monitors the uC address latch enable
13	PSENB	Pin that monitors the uC program store enable
6	P1.0	These pins connect to the pins of Port 1 of the uC. Some of the pins are also connected to the status input pins of the PC parallel port.
7	P1.1	
11	P1.2	
5	P1.3	
72	P1.4,PC_S4	
71	P1.5,PC_S3	
66	P1.6,PC_S5	
67	P1.7	
31	P3.0(RXD)	These pins connect to the pins of Port 3 of the uC. The uC has specialized functions for each of the port pins indicated in parentheses. Pin 63 connects to the data write pin of the uC and the write-enable pin of the RAM. Pin 70 connects to a status input pin of the PC parallel port.
70	P3.1(TXD), PC_S6	
69	P3.2(INTB0)	
68	P3.3(INTB1)	
26	P3.4(T0)	
33	P3.5(T1)	
63	P3.6(WRB), WEB	
32	P3.7(RDB)	
44	P0.0(AD0), D0	These pins connect to Port 0 of the uC which is also a multiplexed address/data port. These pins also connect to the data pins of the RAM.
43	P0.1(AD1), D1	
41	P0.2(AD2), D2	
40	P0.3(AD3), D3	
39	P0.4(AD4), D4	
37	P0.5(AD5), D5	
36	P0.6(AD6), D6	
35	P0.7(AD7), D7	
58	P2.0(A8), A8	These pins connect to Port 2 of the uC which also outputs the upper address byte. These pins also connect to the 7 upper address bits of the RAM.
56	P2.0(A9), A9	
54	P2.0(A10), A10	
55	P2.0(A11), A11	
53	P2.0(A12), A12	
57	P2.0(A13), A13	
61	P2.0(A14), A14	
34	P2.0(A15)	
75	A0	These pins drive the 8 lower address bits of the RAM.
79	A1	
82	A2	
84	A3	
1	A4	
3	A5	
83	A6	
2	A7	
62	OEB	Pin that drives the RAM output enable.
65	CEB	Pin that drives the RAM chip enable.
4	FREE0	These pins are not connected to other devices and can be used as general purpose I/O.
12	FREE1	
25	FREE2	
74	FREE3	
76	FREE4	
77	FREE5	