Microprocessor-Based Systems (E155)

Harris — Fall 2004

Lab 2: Utility Board Assembly & FPGA Test — Due: Week of Sept 20

Introduction

In this lab you will assemble and test your utility board containing a Xilinx Spartan FPGA, EEPROM, PIC microcontroller, LEDs, switches, and a clock generator. You will be using the board for the remainder of the semester, so it is very important to assemble it properly. But don't worry! If you damage your board, you may obtain parts for another one from the stock room.

Objectives

- To learn how to solder;
- To assemble your utility board;
- To learn how to use the Xilinx software to program your design created in lab 1;
- To test and debug your board;
- To connect and drive a 7-segment hex display.

Requirements

Follow the steps in this guide to assemble your board. Modify your lab 1 to include input and output pads, and generate an EEPROM file. Finally, connect this output to the 7-segment display. Learn how to use the Xilinx software PROM programmer. Then "burn" the EEPROM chip that will program the FPGA and test the FPGA. Hook your board up to a 7-segment display and show that your display driver works. Finally, run a short program provided to test the PIC microcontroller.

Background

Historically, digital design projects have been built from a truckload of chips, each containing a few logic gates such as 74xx series logic gates or simple programmable logic chips (PALs). Such projects involve placing and wiring together dozens of chips on a breadboard. It is easy to make a wiring mistake or burn out a chip and spend hours tracking down the problem. Now you will be implementing all of your digital logic on a single field-programmable gate array (FPGA) to greatly reduce the necessary wiring and number of chips, and later in the course, you will use a PIC to write powerful programs that can interface with digital I/O, A/D converters, timers, and others.

You will need to connect your FPGA to the real world to get inputs and outputs. In particular, you will often find it useful to have a clock oscillator and switches and LEDs. Moreover, the FPGA comes in an 84-pin plastic leadless chip carrier (PLCC) package that does not plug directly into your breadboard. You will assemble a printed circuit board containing the FPGA, LEDs, switches, clock circuitry, and header pins to tap the FPGA signals onto the breadboard where you could interface them to other circuits you will build. The PIC will also be on your circuit board with all of its pins tapped out and some shared with the FPGA and LED inputs.

The FPGA you will use in this course is the Xilinx Spartan XCS10. It contains the equivalent of about 10,000 gates and 600 flip-flops. The PIC is a PIC18F452 with 1.5kB of RAM and 32 KB of Flash ROM. You can find the databook for these in the lab and on the class web page. You will need to become familiar with the internals as you progress in this class.

You will receive the Version 1.1a of the SMPS utility board. Version 1.0 was designed during the fall of 2002, and the subsequent revision was complete in Jan 2003.

The board was designed in a way to maximize your access to the capabilities of the FPGA and PIC from a breadboard. Therefore, as you will notice, the board has a single row of 58 header pins that give you access to most of the FPGA pins and all of the PIC I/O pins (except for reserved programming pins). There is an additional set of FPGA pins tapped out that are accessible with a ribbon cable. The other pins in the FPGA are power, ground or unused. Furthermore, the pins are labeled for your convenience. If interested, we advise you to look at the Spartan and PIC databook. It contains full explanation of the functions of each pin. The other characteristics of the utility board will be explained later, in the Discussion section of this guide.

Attached as an appendix of this lab are of the schematics for the utility board. It is advisable that you look and try to understand this schematics. It will help you figure out what goes on in the board. The parts in the schematics are labeled in the following way: R for resistors, C for capacitors, J for jumpers, and U for units (chips and other large parts).

Discussion

Figure 1 shows a schematic of the utility board. Major components include the power supply, header pins, clock generator, DIP switches, reset pushbuttons, the FPGA mode selector jumper, the debugger/programmer connectors, and LEDs.

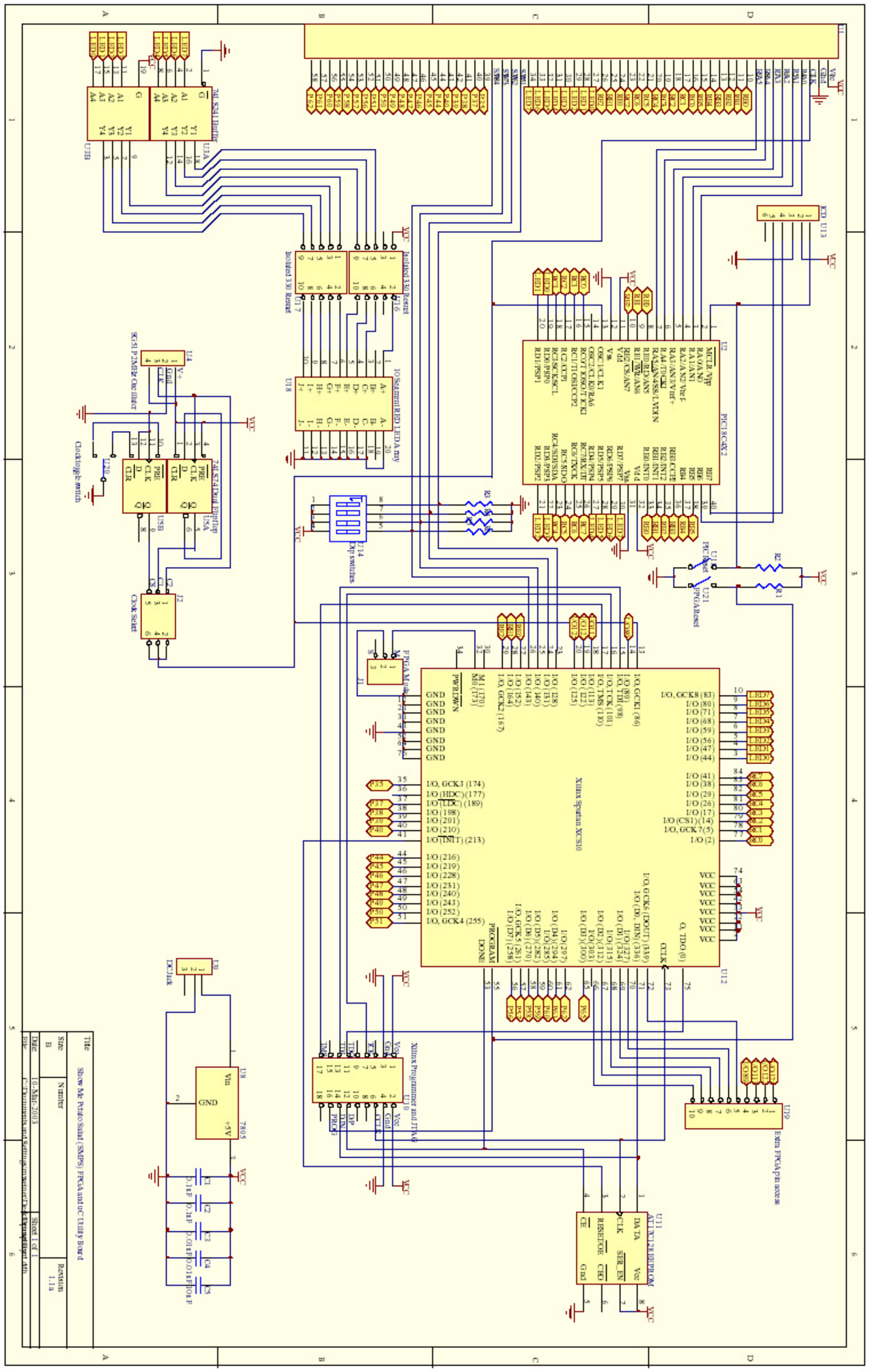


Figure 1 – Utility Board Schematics

Read through and understand this section that describes each of the board components. Identify the components in the schematic and think about how it operates.

The next section entitled “Assembling the Board” will guide you through assembling your board. The following section will describe changes you need to make to your Xilinx project from lab 1 to generate configuration files for the FPGA. To test your board, you will burn an EEPROM. You will connect an external 7-segment display and test your display driver. Finally, you will run a test program provided to check you’re your PIC microcontroller works. Your lab manual only needs to document the result of your testing and any problems you encountered as well as the design and test results of your external 7-segment display.

Power Supply

The utility board has two choices of where to supply power: two header pins, or a DC transformer input jack. The two pins on the header are for supplying a *regulated* +5 volts to the board (i.e., from the multiple output power supply in the lab). Alternatively, you may use a +9V DC transformer plugged into the power jack at the upper-right of the utility board (convenient if you aren’t working in the lab). This power is sent through a +5V regulator (7805), which then supplies power to the utility board and to the +5V pin. It is *very* important that you do not try to supply power to the +5V pin while using a DC transformer because you will create a short circuit.

Header Pins

The board contains space for one 58-pin row of male header pins that tap out signals from the FPGA and PIC, the clock, LEDs, switches, and power and ground. The header pins will be installed along the left side of the utility board.

Clock Generation

In digital electronics, the term clock refers to a square wave that oscillates between GND and V_{CC} (+5 volts on your board) Typically, clocks are used to sequence the action of a circuit. For instance, a flip-flop samples its input every time the controlling clock rises from logic 0 to logic 1. The clock signal is supplied onboard the board and is available at the MCLK (Master Clock) header pin. It is also sent to the PIC and the FPGA.

You can select a clock operating at 2 MHz, 1 MHz or may single step your clock using a debounced toggle switch. A jumper placed across one of three pairs of pins controls the frequency. Your board contains a 2 MHz clock oscillator. A divide-by-two circuit constructed using one D flip-flop provides the 1 MHz output. The large toggle switch is debounced using another flip-flop to provide a single-step clock.

Switches are mechanical devices. When you toggle an ordinary switch, the contacts have a tendency to bounce a few times, typically on a millisecond time scale. This bounce is unimportant for inputs to combinational circuits, but wreaks havoc on the clock input of sequential circuits because it might appear as several clock pulses rather than one. Your utility board has a clock debouncing circuit constructed from an SR latch that filters out these glitches. You are encouraged to study the debouncing circuit and figure out how it operates.

You may select your clock frequency by installing a jumper in one of the three positions at J2: 2Mhz, 1Mhz, or MAN (for manual clock generation via the toggle switch).

DIP Switches

The DIP switches provide a convenient source of inputs to circuits in your FPGA. They have a 1kΩ resistor in series with the output so the same pins of the FPGA can be used as an output, overpowering the switches if the switches are not needed.

Reset Pushbuttons

There are two reset pushbuttons situated near the FPGA and PIC. The pushbutton near the PIC resets the PIC, and the one near the FPGA resets the FPGA! Upon reset, the PIC will simply reset its program and begin as if it has just powered up. The FPGA, however, will have its memory cleared. If the EEPROM is installed and you are in master mode, it will immediately reprogram itself from the EEPROM. You will notice that these pushbuttons actually have 4 pins. However, only two pins are used, while the other two are left unconnected.

FPGA Mode Selector Jumper

The FPGA is configured by pattern of bits into internal SRAM specifying the logic and interconnections inside the chip. Configuration operates in one of two modes. In master mode, the FPGA generates a configuration clock on startup that drives a serial EEPROM. The serial EEPROM returns the data. This means configuration occurs automatically on startup but requires programming an EEPROM chip. In slave mode, the FPGA receives the configuration clock and data from a host computer over the Xilinx parallel programming cable. This avoids the need for the EEPROM but requires a connection to the PC. The configuration mode is set by jumper J1. To set the FPGA to master move the J1 header pin to the bottom two pins. Conversely, to set the FPGA to slave mode, place the jumper on the top two pins.

Debugger/Programmer Jacks

There are two programming connections at the top of the board. The double row of header pins are for programming the FPGA. The top row is for programming via the Xilinx Parallel III cable, and the bottom row is for programming via a JTAG connector (currently not available for lab use). The 6 pin jack (which resembles a telephone jack) is for programming the PIC microcontroller via the ICD 2 module. You will program and debug your PIC programs onboard in Lab 5.

LED array

There is a 10 segment LED array of red LEDs labeled ‘ON’ and 0 through 7. The ‘ON’ LED simply tells that the board is powered ON. The LED immediately to the right of the ON LED is unused. The other 8 red LEDs will be used for output of one byte, with the LSB on the right. Notice that there are 8 header pins labeled LED0 – LED7. These can be used in case you want the outputs of the LEDs for an external device. These 8 LEDs are normally on when the inputs are left floating. If a voltage is applied at one of the LED pins, the corresponding light is off for ground (GND) and on for voltage 5V (V_{cc}).

There are a few concerns that must be addressed whenever using LEDs. One is that LEDs turn on when given about 5 mA of current and may burn up when given more than roughly 20 mA. Therefore, it is important to include a current-limiting resistor to prevent LEDs from burning out. A 330 Ω resistor does the job nicely, allowing $\sim \frac{5V}{330\Omega} =$ 15 mA of current to flow. Not all chips can source this much current, so a buffer chip on the utility board isolates the LED from any digital circuits being constructed. The 74LS241 chip contains eight buffers capable of providing the 15 mA of current. A schematic of one of the LEDs appears in Figure 2:

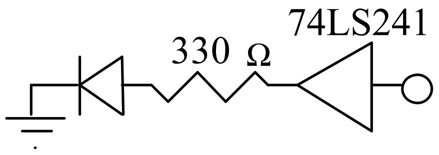


Figure 2 – Schematic for one LED

Note that, since the 74LS241 defaults to a high output when unconnected, all of the LEDs default to ON when not driven, or when the FPGA is reset. Also note that if you leave the appropriate pin of the FPGA floating, you can drive the LED from another circuit connected to that pin.

FPGA

The FPGA contains a large array of configurable logic blocks (CLBs) and programmable interconnections. The configuration information is stored in static random-access memory (SRAM) within the FPGA. Therefore, the information is lost when power is removed and the device must be reconfigured each time it is powered up.

The FPGA can be configured via a cable attached to your computer or may load its configuration information automatically from an EEPROM. In this class, you will usually do the latter. The utility board has a socket near the bottom edge where you may insert a 128 kilobit EEPROM into the breadboard. On powerup, the FPGA will serially configure itself from the EEPROM.

Assembling the Board

This section of the lab guide will give you tips on how to assemble the utility board. Following these tips will help you complete this lab quickly and in a logical way that places the flattest components to make soldering easier.

☐ Identify the Component Side of the Utility Board

The utility board has two sides. The component side, with the white silk screen markings indicating component placement, should go up. All components except the 58-pin header are placed on the component side. The parts (except the 58-pin header) are soldered on the reverse solder side.

Using a multimeter, check that there is nearly infinite resistance between the pins labeled +5V (or V_{CC}) and GND on the board. If the resistance is low, you have a short circuit on your bare board and should get a new one. As you assemble your board, occasionally check the resistance between power and ground. If it is ever less than 10 Ω, you've introduced a short and should debug it before continuing.

When you begin soldering, moisten the sponge. When the iron first heats up, tin the tip by applying a generous amount of solder directly to the tip and wiping off the excess on the sponge. Periodically retin as you work to keep the tip looking silvery. When you make a connection, touch the tip of the iron to the pad on the board at the same time it touches the lead of the component. Apply the solder to the joint, not the tip of the iron. The solder should smoothly adhere to both the pad and the part rather than balling up on the part. The connection should appear shiny; a gray color indicates a possible unreliable "cold solder" joint. If you are in doubt about the quality of your solder joints, ask early on rather than doing all of them first and discovering that many of your connections are intermittent or unreliable. Some of the components will be soldered close to vias. Be sure excess solder does not bridge to the via, creating a short circuit. When you are done, tin the iron one last time to protect the tip before turning it off.

You may wish to use safety goggles while soldering, especially if you don't wear glasses. Also, be sure to hold the ends of leads as you cut them after soldering so they don't fly into the eye of the person working beside you. Solder contains lead, so wash your hands afterward.

☐ Insert the Resistors and Resistor Networks

The first components to install in the utility board should be the resistors. They will sit flat on the board. The polarity of the resistors does not matter, i.e., you can insert them whichever way you'd like. However, it is convenient to install neighboring resistors in the same direction, making it easy to read them. Refer to the documentation in your lab notebook on reading resistor values. R1 – R6 are 1kΩ resistors.

Next you should install the (isolated) resistor networks. 'Isolated' means that it contains a set of totally independent resistors. These particular resistor networks are sets of 5 330ohm resistors and are installed at U16 and U17. Pins 1 & 2 are one resistor, pins 3 & 4 are another, and so on. Polarity does not matter. You may find it helpful to bend one pin at each end of the resistor network to hold it in place while soldering.

☐ Insert the EEPROM's Socket

At U11, labeled EEPROM, you should install an 8-pin socket to house the FPGA's serial EEPROM. Align the notch on the socket with the silkscreen notch on the board.

☐ **Insert the PIC's Socket**

The PIC's 40-pin socket should be installed next in U2. Align the notch of the socket with the silkscreen notch on the board.

☐ **Insert the Ics, Oscillator**

There are two ICs to be assembled in this step. They are the buffer chip driving the LEDs and the flip-flop used for switch debouncing. Chips are easily damaged by excess heat, so apply the soldering iron no longer than necessary to make the connections. The polarity of the dual inline package (DIP) chips is indicated with a notch at the top or a dot in the upper left near pin 1. The silk screen on the board also contain a notch and an indication of pin 1.

The 74LS241 chip, U3, contains 20 pins, and is located close to the LEDs. It may be useful to bend the legs with pliers before inserting the ICs. Pin 1 is the first pin to the left of the notch in the chip drawing on the board. Then add U5, the 74LS74 flip-flops.

The next step is to insert the oscillator. The oscillator is a 4-pin chip, and its location is in the opposite side of the board from the red LEDs. This location is marked with U4.

☐ **Install the Pushbutton Switches**

There are two small pushbutton switches that come with your kit, which will be installed in U15 and U21. These will be used to reset the PIC and the Xilinx. When installing one of these, you will have to align it in its holes and push until it snaps into location. Do not put your finger directly underneath as the sudden popping into position could stab your finger (I know this from experience!). A thin pair of pliers helps push the legs through. The switch will only align in two positions on the board; either alignment will work.

☐ **Insert the DIP Switches and LED array**

The DIP switches are to be placed at the bottom of the board at U14. Align the 'ON' label on the DIP switches with the 'ON' label on the board.

Finally, insert the LED Array at U18. If you look at the bottom of the LED array, you will see a faint arrow pointing to pin 1. Place pin 1 in the hole labeled '1' (just above the 'ON' label).

☐ **Insert the Capacitors**

There are two .01μF, two 0.1μF, and one 10μF capacitors to install. These will be used to supply power bypass to the whole board and near the FPGA and PIC. Install the 0.01μF caps in C2 and C3 (polarity does not matter). Install the 0.1μF's in C1 and C4 (again polarity does not matter). Now install the 10μF in C5 (polarity DOES matter). There are two ways to determine polarity on an electrolytic capacitor. If it is new and uncut, the longer of the two leads is the cathode (+) and goes in the + hole on the utility board. The other way is to look at the capacitor and find the stripe going down one side. This points to the anode (-) lead. If you install this with the incorrect polarity, it can leak or explode.

Small capacitors are labeled with an obscure three-digit code similar to resistors. For example, 223 would mean $22 \cdot 10^3$ pF = 0.022 μF.

☐ Install the Short Header Pins

Now install all headers except the 58-pin one. You will have to cut them from the long strips of pins. All of these should be installed on the component side (*top* of the board) with the short pins going through the hole. (The longer gold-plated pins should stay on the top of the board, as well as the black plastic.) Solder should be applied on the bottom side. A 9x2 row of header pins should be installed in U10. This will allow you to program the FPGA with the Xilinx Parallel III adapter (the JTAG port is also tapped out for those who want to play with it). Now, install the 3x2 row of header pins in U6. This will be for selecting the clock speed. Also put a jumper horizontally on two of the pins marked 2 MHz. Now install a 3x1 header at J1 and install a jumper on the two pins labeled M for Master mode. Finally, install a 10x1 female header at U19. This allows access to some more pins on the FPGA.

☐ Insert the FPGA Socket

Because of its many pins, the FPGA will take some time to be assembled. In this step you will install only the socket, leaving the chip itself to be installed only when the board is completed and debugged – make sure there are no short-circuits in the board or *between the FPGA pins* before inserting the FPGA chip in its socket, otherwise you may damage it. Before you begin soldering the socket, check for shorts between V_{CC} and GND on the board. If you find any, debug them before proceeding.

To assemble the socket, align its arrow (a vertical line in the middle of the outer edge of the socket) with the arrow on the board. Then insert the socket and solder all of the pins. Do not use excess solder; it may form a short under the socket and be very difficult to fix. When your board is complete and debugged, you will align the dot on the FPGA chip with the arrow of the socket. Be sure the orientation is correct; you will be very unhappy if you place the socket improperly and have to remove it later!

After you have soldered the socket, check again that there are no power/ground shorts. There is a hole on the board for pushing a screwdriver (or other prodding tool) through to pop the FPGA out of the socket. Do this carefully so you do not bend the pins on the FPGA. Check that the pins are aligned (and no two are touching each other) every time you insert the FPGA.

☐ Install the ICD connector and the DC Jack

The DC Jack is a 3-pin connector U9. The ICD connector looks like a telephone or Ethernet jack. When placing this part at U13, you will have to push gently until the jack is firmly in place. Then solder each of the 6 pins.

☐ Install the 58-pin Row of Header Pins

This is the *only* part that is installed on the bottom and soldered on the top (it will be inserted into the breadboard). You want to place the part so that the long pins extend below the board, with the black bumper up against the bottom of the board. The short pins go through the board and are soldered on top. Watch carefully that you do not use too much solder because it may bridge to one of the vias. When you place the header pins, try to keep all of them perpendicularly aligned so that placing the utility board in your breadboard isn't too much of a headache. As the header pins come in strips, you will need to use one long strip and cut one shorter strip.

☐ **Insert the Voltage Regulator**

U8 is a 7805 voltage regulator. It installs vertically with the metal heat sink closer to the row of header pins on the left.

☐ **Add the Clock Switch**

The clock switch is the large toggle switch. It was left for last because of its height. Inserting it first would make it difficult to insert the other components. Its location is just to the left of the DIP switches. Insert it and solder its pins under the board.

Now hopefully you should be done with assembling your board. Before proceeding with the lab, you should check all of your soldering. They should look reliable, filling their specific holes in the board. If some of your soldering looks more like a bubble on the top of the pin, you probably have a bad connection there. Also look for solder jumping across pins.

After you've checked your soldering and verified there is no power to ground short on the board, you should insert the FPGA and PIC chips. Remember to align the dot on the chip with the arrow in its socket. Your board is done!

☐ **Place the Utility Board on the Breadboard**

Now you have to place your board on the breadboard. It should fit in the right-most side of the breadboard.

After you've placed the utility board, connect a red wire from the V1 connector of the breadboard to the first power row. Then, from this row, connect a wire to the row of pins next to the +5V pin on the utility board. Then connect a black wire from the GND connector of the breadboard to the second row of the second power rows (right below the red wire). And connect this row to the Gnd pin of the utility board. Make sure these wires do not touch each other. You are now ready to proceed with the lab!

Generating the FPGA Configuration Files

Now you will learn how to program the circuit you created with Xilinx into the FPGA. To do so, you will need to "burn" the EEPROM with your circuit, which will then be placed EEPROM socket to load the FPGA.

Completing your FPGA Design

Your design from Lab 1 needs a few more tweaks before you can use it. Specifically, you need to generate a top-level Verilog module that contains both the LED circuits and the hex display decoder and you need to assign pin numbers that match the FPGA board.

First, open your lab1_xx file. Use the File • Save Project As command to save it in a lab2_xx directory as lab2_xx.npl. In the sources pane, the project will still be named lab1_xx. Double-click on the name to bring up the project properties dialog and rename it lab2_xx.

Now, create a top-level Verilog file named lab2.v. It should have the following inputs and outputs:

```
clk    input
s      input  3:0
led    output 7:0
seg    output 6:0
```

Instantiate copies of the led and sevenseg modules. The syntax for the led module is shown below. Replace the comment line with one more line of code for the sevenseg module.

```
module lab2(clk,s,leds,seg);
       input clk;
       input [3:0] s;
       output [7:0] leds;
       output [6:0] seg;

       led led1(clk, s, leds);
       // instantiate sevenseg here

endmodule
```

Next, assign pins. Be sure the lab2 source is selected. In the User Constraints suboption of the Design Entry Utilities process, select Edit Implementation Constraints (Constraint Editor). The Constraints Editor will open. Click on the Ports tab for a list of all the input/output pins in your design. In the Location column for the clk input, type P13 because pin 13 is the clock input (e.g. MCLK). Similarly, led[0] should be assigned P23. The pin numbers for the other LEDs and switches are on the silk screen of the FPGA board. For the outputs for the 7-segment display, you may select any I/O pins you'd like. For example, you could use pins 44 through 51, since they are contiguous.

It is wise at this point to simulate your lab2.v file to ensure you have not made any mistakes. Synthesize and implement the project and correct any errors you may encounter. Look at the place & route and pad reports and make sure they match your expectations.

Programming the EEPROM

With lab2 selected in the source window, double-click the Generate PROM File suboption of the Generate Programming File process. This invokes the PROM File Formatter tool which allows you to generate a file to burn into your EEPROM. The Xilinx FPGA reads the data from the EEPROM at startup to configure itself according to your design.

Use the File • PROM Properties menu to open the properties dialog. Check that the file format is MCS-86, type is Serial, and file is Single PROM. Click OK. Then use File • Create PROM to write the PROM file. Make sure you remember where the file was saved.

In order to complete programming the EEPROM, you will need to transfer the file you just created to a special computer in the E157 lab. However, since this computer is not connected to the engineering server, you will need to copy the file to a floppy disk. Thus, close the PROM File Formatter window and the Project Manager. Then open the explorer and go to the directory where the PROM File Formatter saved your file. There should be a file named xx_lab2.mcs in this directory. This is the file you need to copy to a floppy disk. You may find it handy to share floppy disks to transfer the .mcs files.

Remember, of course, to only use your own file. It is a serious honor code violation to pass somebody else's work as your own.

In the MicroP's lab you will find a computer connected to a device called "EMP-20," which is the device you will use to "burn" the EEPROM. This computer should be turned on already, and the EMP-20 software should be running. If not, turn on the computer, `cd emp20`, and then type "`EMP20`" to run the software. Also turn on the EMP-20 device programmer (the switch is on the back).

This software is simple to use. First, you need to specify the type of PROM you want to "burn." To do so, press "5" or open the "Select Device" option. You will be given a list of manufacturers and devices for that manufacturer. The manufacturer for the EEPROM you will use is option 6 (Atmel), and the device is option 118 (AT17C128).

Next, you will need to specify the source file that will be "burned" into the EEPROM and the file type. Press "U" to select the file type. Choose option 2 (INTEL HEX), which is compatible with the MCS-86 format you chose in the Xilinx PROM File Formatter. Now press "V" to enter the file name. Insert you floppy disk into drive B and then type the file name. The name should be something like "b:\xx_lab2.mcs." After this, when the EMP-20 software takes you back to the main window, press "8" or select the "Load File from Disk" option. This way the software will read your source file into the computer's memory. You will be notified if the file was loaded correctly or if there were any incompatibility problems.

Now you are ready to "burn" the EEPROM. Before proceding, though, you need to check if the "EMP-20" device has the correct Family Module installed. The device uses this module (21B in our case) to interface with the EEPROM. Make sure that the correct Family Module (21B) is installed. It should be inserted onto the left side of the "EMP-20" device. If the correct module is not installed, you will need to ask for the correct one in the stockroom (or may find them in a wallet sitting on the shelf above the programmer). You can learn more about installing the module by pressing [F10] in the EMP-20 software and choosing option 8 (Family Modules).

Place the EEPROM in the "EMP-20" device. It should be placed starting in the bottom of the socket – closest to the lever – with the notch on the chip facing away from the lever. Each pin should be in an individual slot. After placing the EEPROM, lowering the lever and making sure that the "EMP-20" device is on, press "1" or choose the "Program with Selected Algorithm" option. The EMP-20 software will "burn" the EEPROM and automatically check it for any problems. After this you should be informed that the operation was successful. Finally, check the reset polarity option by pressing B while the EEPROM is still in the programmer. The polarity should be active low; if it is not, press return to toggle it.

Testing Your Utility Board

Before testing the utility board, you will need to remove the EEPROM from the EMP-20 device and place it in the EEPROM socket. Insert it in the space between the header pins. The notch of the EEPROM should be aligned with the notch on the socket.

Be very careful when inserting or removing your EEPROM. They cost about $5 each and it is easy to bend or break the leads through careless handling. When removing your EEPROM from your breadboard, use a chip puller, pliers, or screwdriver to evenly lift the EEPROM out of the board. If you use your fingers, you will probably break the legs before the end of the semester.

Before connecting your board to the power supply, turn on the supply. Set the voltage knob to the 6-volt scale and adjust the output to exactly 5 volts. Then turn off the power supply and use red and black banana plugs to connect the 6V and COM outputs to V1 and GND terminals on your breadboard, respectively.

Turn on the power supply. Check that less than 100 mA of current is flowing from the 6 volt supply, as measured on the lower scale. Feel your FPGA and make sure it is not getting warm. The ON light on the LED array should also be lit. If anything is amiss, turn off the power supply immediately before damaging any parts. Look for solder bridges on your board.

Check that your test program properly controls all 8 LEDs. When you turn the power on, the FPGA should automatically load the data stream from the EEPROM and program itself. The circuit should be fully operational now, except for the 7-segment display, which has not yet been installed. Check that the 8 LEDs respond as expected to the switches and clock toggle switch.

Use a logic probe to check that the header pins going into your breadboard for the 8 LEDs and 4 switches have the same value as the LEDs and switches indicate, to ensure that there are no bad connections to those header pins. Use an oscilliscope to probe P13 / MCLK header pin on the board. Verify that the 1 and 2 MHz clock frequencies are correct.

If you have problems, the following checklist may help:

1. Borrow the known good E155 Board Test EEPROM from the teaching staff. This EEPROM should display different patterns on the LEDs based on the DIP switch settings. If your board works with it, your EEPROM is misprogrammed. If all the LEDs remain on when you use your EEPROM, the FPGA is probably not able to read your EEPROM. Check that it was programmed with Reset polarity "Active Low." Check that you generated the PROM .mcs file using the MCS-86 format in Xilinx and that you had implemented targeting the Spartan S10PC84 device. Check that the EMP programmer is set to the Atmel AT17C128 and the Intel Hex format. Check that the EEPROM is not rotated 180 degrees.
2. Check that the mode jumper is between the M and center pins (for master serial mode) and that the clock jumper is on the 1 MHz or 2 MHz row.
3. If the LEDs don't all remain on but your design does not function correctly, the problem is likely in your Xilinx schematics or implementation. Check the implementation report file that the pin mapping is what you want. Check for any logic bugs in your schematics.
4. Check that the power supply is hooked up correctly. Measure with the volt meter that you have 5 volts between the V_{CC} and GND pins on your FPGA board. Check that the current out of the power supply is less than 200 mA. If the current is high, look for power/ground shorts on your board. Otherwise, if the voltage is incorrect, adjust the power supply to produce the correct output.
5. If your board still does not work, refer to the appendix of this lab on the FPGA configuration sequence. Check on a scope that PROGRAM, LDC, and DONE

behave as expected. Look for bad solder joints or solder bridges on your FPGA board.

JTAG Programmer

When you are working in the lab, you can download code directly to your FPGA through the Xilinx Parallel Cable IV. The cable needs to be plugged into the parallel port of the PC. It has a power attachment that should be plugged into the keyboard or mouse port of the PC as well.

On your E155 board, be sure the mode jumper is set to slave serial mode (S) rather than master serial mode (M). Remove the EEPROM from its socket because it also uses the DIN pin and intereferes with the cable.

To download your design, use the following steps:

- Select Generate Programming File * Configure Device (iMPACT)
- Click on Slave Serial Tab.
- Right-click on FPGA and Assign New Configuration File
- Choose your .bit file
- Connect the following flying wires to the header pins on your board
    - VCC, GND, CCLK, DONE (to D/P), DIN, PROG.
    - Don't connect INIT
- Make sure power is turned on to your board.
- Output: Cable Autoconnect (should say cable connection established)
- Right-click on FPGA and choose Program (should say Programming completed successfully)

7-Segment Display Circuit

The 7-segment display will be used throughout the class for general output of numbers. In this lab assignment, though, it will be used to output the hexadecimal number entered by the user through the DIP switches.

Each segment of the display works as an independent LED. Therefore, the same current-limiting concern with the LEDs applies to the display. You can limit the current into each segment of the display the same way you did for the LEDs, adding a suitable resistor to provide roughly 5-20 mA of current. You can find resistors and other such components in the supply cabinet or in the stockroom.

The display you will be given in this lab can actually output two numbers and two dots (it is a dual 7-segment display). However, it works as two independent displays, each with one number and one dot. The following picture describes the pins of the display:

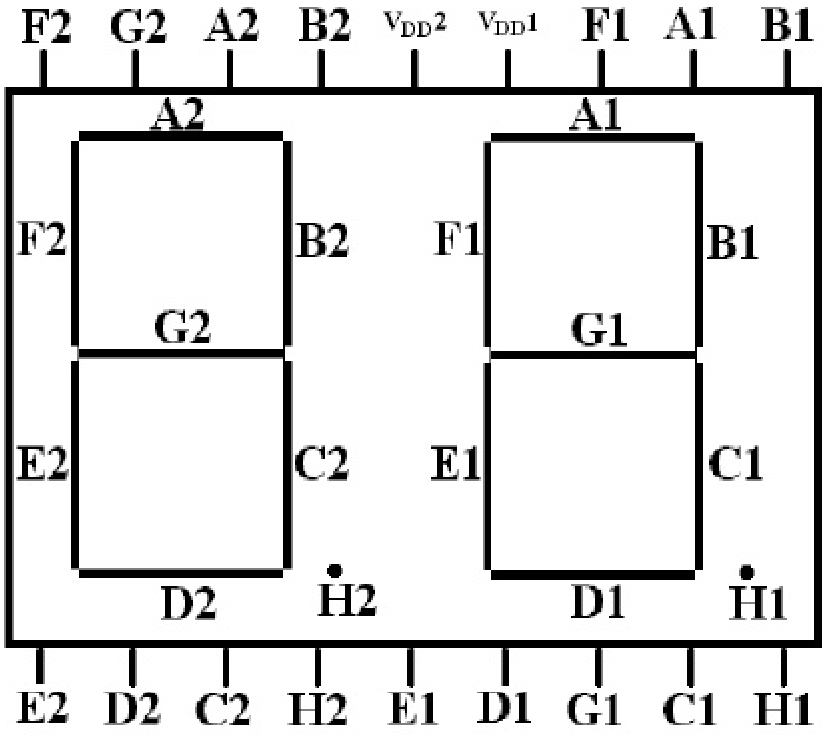


Figure 3 – 7-segment display pins

Be sure to turn power off before wiring circuits on your board. You can choose either side of the display to use in this lab. After deciding on which side to use, you will need to connect the V_{DD} pin of that side (either $V_{DD}1$ or $V_{DD}2$) to 5V. Then connect the input pins of the same side of the display to the header pins you chose in your schematics. Remember to add suitable resistor between each of the inputs to the display and the header pins.

After assembling the display, you may turn the power on and debug your circuit. You should be able to see the hexadecimal numbers on the display. Good luck!

You will test the PIC microcontroller in Lab 5.

What to Turn In

Turn in your lab notebook including:

- Notes about your PC board testing results. If anything did not work, document your debug process.
- Verilog code for your FPGA.
- Schematics of your breadboarded circuits.
- Test results for the 7-segment hex display.
- How many hours did you spend on the lab? This will not count toward your grade.

Have your lab checked off by the instructor. You will need to demonstrate that the board and 7-segment display operate correctly. You will also be asked a question about some part of the lab or your board. You should be thoroughly familiar with all of the lab and the components of your board to be able to answer the question. The oral exam is typically in the form of a "Fault-Tolerance

Question." What would happen if a particular wire is broken or a pin is shorted to V_{CC} or GND? Be prepared for any other questions about your lab, however.

Acknowledgments:

Revision 1.1a of the E155 utility board was developed by Marty Weiner in Fall 2002. This lab manual and previous utility boards have been developed by Elizabeth Reynolds '02, Marty Weiner '02, Fernando Mattos '00, and Eliot Gerstner '99. The name of the board was inspired by the writers for Family Guy in their ever pursuit of a great, cheap laugh.

Appendix: FPGA Initialization Sequence

If your FPGA board is assembled incorrectly or your EEPROM is programmed wrong, you may find that your FPGA fails to initialize. This sheet documents the initialization process to help you track down your problem. The initialization sequence is described in more detail in pages 4-25 through 4-34 of the Xilinx Data Book.

Test the initialization in Master Serial Mode by setting jumper J1 on the middle and M pins. The FPGA generates appropriate control signals to drive the EEPROM, which serially transmits the configuration information into the FPGA. Other configuration modes could be used if you were trying to configure multiple FPGAs at once or if you were trying to test your FPGAs using boundary scan.

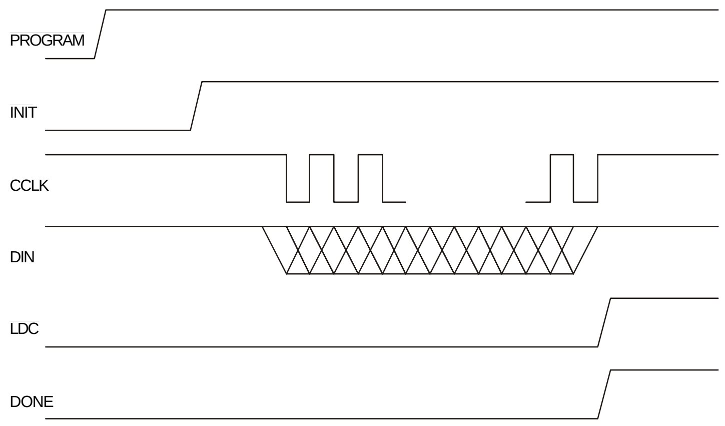
When you press the Xilinx reset switch (U21), the PROGRAMb[1] signal is driven low. This causes the FPGA to repeatedly clear its internal memory to prepare for configuration. While PROGRAMb is low, the FPGA drives INITb, LDCb (Low During Clear), and DONE low. It also drives CCK (Configuration Clock) high. The LDCb signal resets the serial EEPROM, which in turn drives DIN high.

When you release the Reset switch, PROGRAMb returns high. The FPGA finishes clearing the memory one last time (which should take about 700 µs), then lets INITb go back high. About 130 µs later, the FPGA begins togging the CCLK signal high and low at about 1 MHz. The EEPROM responds by sending appropriate data signals. The XCS10 FPGA has 94960 bits of internal state, so at 1 MHz the EEPROM will require approximately 100 ms to configure the FPGA. Note that the FPGA supports an express mode in which it generates an 8 MHz configuration clock. You can activate this mode from one of the Implementation options in Xilinx.

When configuration is complete, CCLK stops toggling and becomes high. The LDCb and DONE signals go high to indicate successful configuration. If these signals are not both high, the FPGA was not configured.

The configuration waveforms are shown below:

[1] Note that the lowercase b at the end of the signal name implies a bar over the signal name (i.e. the signal is active low).



If your board is acting up, connect the signals listed in the timing diagram to a logic analyzer. Trigger off the rising edge of PROGRAMb and look at the signals. If they are not toggling correctly, your EEPROM may be misconfigured (try somebody else's and see if it works) or your board may have a bug.

To track down EEPROM problems, first check that your .mcs file looks correct. It should follow the Intel HEX format, documented on the class web page. An example of a syntactically correct file from one of my FPGAs appears below. The first line should have the extended segment address record and the last line should include the end of file record.

```
:020000020000FC
:10000000FF04E88CF9EA7F7FFDF5D75F7FFDED7D8A
:10001000EFFEF6EE6FEFFEFAFBAFFFEFBFFFFEFB6A
:10002000EFBFFFFBDFDFFFFDFDFFDF7FFFFFE9FF2E
:10003000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD0
…
:102E5000F7D7777F7FF7B7F7FEFAEBAFBFFEFAEB56
:042E6000AF4FFEFF73
:00000001FF
```