# Microprocessor-Based Systems (E155)

**D. Money Harris** **Fall 2008**

Lab 1: Utility Board Assembly

## Introduction

In this lab you will assemble and test your utility board containing a Xilinx Spartan 3 FPGA, PIC microcontroller, configuration PROM, LEDs, switches, and a clock oscillator. 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 parts on your board, you may obtain parts for another one from the stock room.

## Objectives

- Learn how to solder
- Assemble your utility board
- Write a Verilog module to control LEDs and a 7-segment display
- Program the FPGA with your Verilog code
- Test and debug the utility board
- Interface the 7-segment display to the utility board

## Requirements

Follow the steps in this guide to become familiar with and assemble your board. Write some Verilog code to exercise the FPGA using the switches, LEDs, and a 7-segment display to ensure your board is operational. Simulate and synthesize your code, then burn it onto the Flash memory and test the board. Hook up a 7-segment display and demonstrate that it works.

## Background

In the 1980's and 1990's, digital design projects were built from a truckload of chips, each containing a few logic gates such as 74xx series logic gates or simple programmable array logic chips (PALs). Such projects involve placing and wiring together dozens of chips on a breadboard. It was easy to make a wiring mistake or burn out a chip and spend hours tracking down the problem. Now you can perform all of your digital logic on a single field-programmable gate array (FPGA) to greatly reduce the necessary wiring and number of chips. Later in the course, you will use a PIC microcontroller to write programs in assembly language and C that can interface with external hardware and the FPGA.

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, switches, and LEDs. Moreover, the FPGA comes in a 144-pin thin quad flatpack (TQFP) package. This is a surface mount (SMT) part so the pins of the chip do not go through the printed circuit board (as with many of the other through-hole parts you'll be working with). SMT components are mounted on pads on top of the PCB. Their pins are so small that special soldering skills and tools are needed to reliably attach them to the board. Your board comes with the FPGA and its Flash memory already attached, but you will get to solder the rest, including a few easier SMT components.

The FPGA you will use in this course is the Xilinx Spartan XC3S400. It contains the equivalent of about 400,000 gates and about 7,000 flip-flops. The microcontroller is a Microchip PIC18F452 with 1.5KB of RAM and 32 KB of Flash ROM and many I/O ports. You can find the databook for these in the lab notebook and on the class web page. You will need to become familiar with the internals as you progress in this class.

The board was designed 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 (on one of the long sides of the board) 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.

Figure 1 shows the schematic for the utility board. It is advisable that you study this schematic to understand what goes on in the board because you will need this information to use and debug your board and are likely to be asked about some subsystem during your checkoff. The parts in the schematics are labeled using 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.

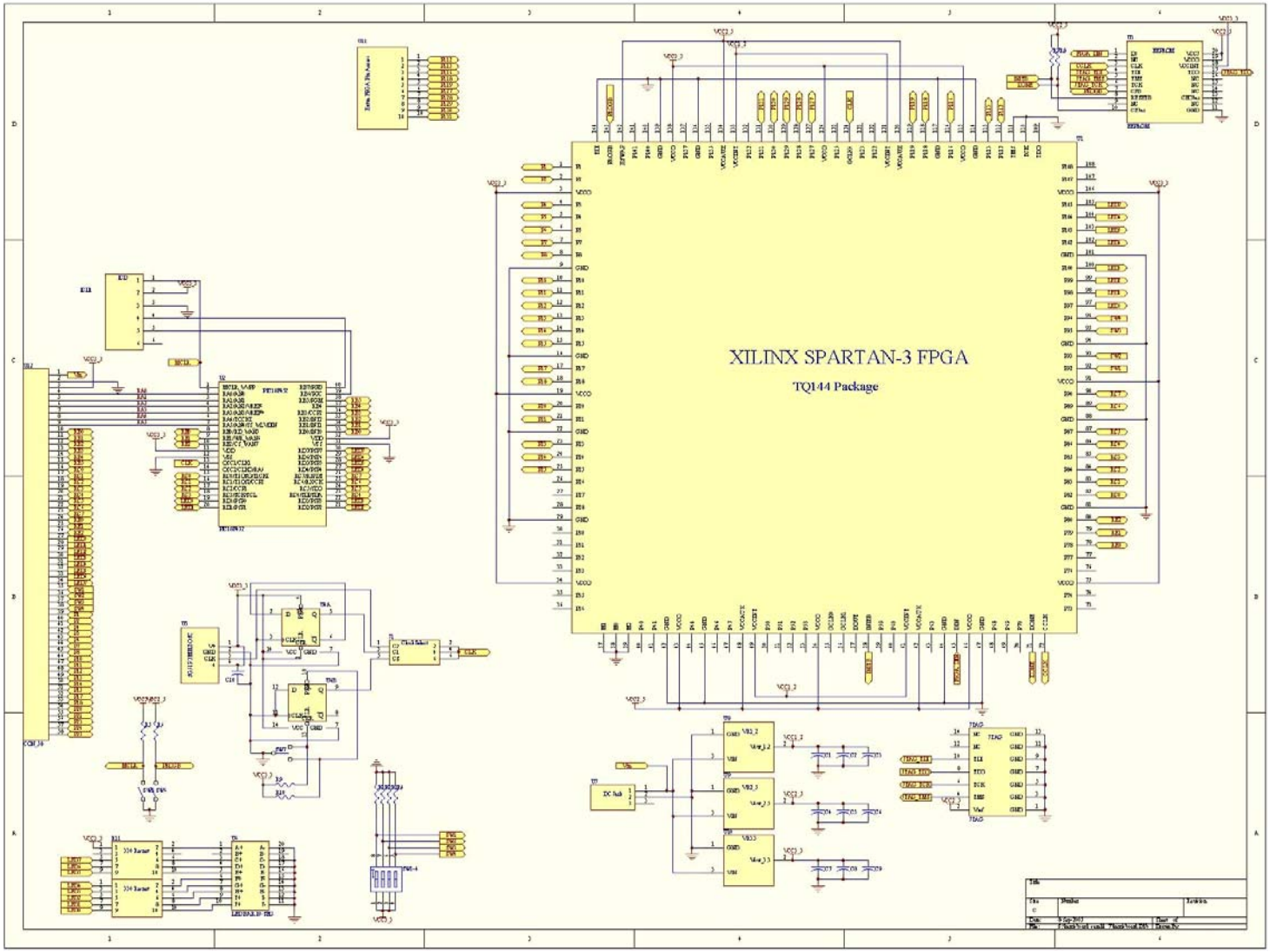



**Figure 1 – Utility Board Schematics**

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 subsequent section will guide you through the assembly process.

**Power Supply**

The utility board has two choices of where to supply power: header pins (marked Vin and GND) or the DC transformer input jack. The input supply should be at least 5 volts; for example, an unregulated 9 V DC transformer suffices. It is regulated down to 3.3 V, 2.5 V, and 1.2 V to serve various components on the board. The 3.3 V regulated voltage is also available at another header pin to serve other components you might want to connect to your board.

*Do not* simultaneously supply power through the Vin header and the DC transformer; the two supply inputs are connected so you will create a short circuit.

In the lab, the easiest way to supply power is from a benchtop power supply to the Vin header pin. If you do not have a power supply available, a DC transformer is an inexpensive alternative. For unteathered applications such as robotics, you can also use a battery pack providing at least 5 V.

The PIC and FPGA I/O's will operate at 3.3 V, which we will call $V_{CC}$. The FPGA contains tiny logic transistors that would burn out at 3.3 V, so the 1.2 and 2.5 V supplies are needed to operate the digital logic and the analog circuitry within the FPGA.

**Header Pins**

The board provides 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 $V_{CC}$. 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 also sent to the PIC and the FPGA.

The utility board provides a fast clock from a crystal oscillator and a single-step clock using a debounced toggle switch. A jumper placed across one of three pairs of pins controls the frequency. The board contains a 40 MHz clock oscillator. A divide-by-two circuit constructed using one D flip-flop provides the 20 MHz output. (Some students have found that the 20 MHz clock was more reliable than the 40 MHz clock, but you're welcome to try to use either.) 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 from the toggle switch. 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 J1: 40 MHz, 20 MHz, or MAN (for manual clock generation via the toggle switch marked SW7).

**DIP Switches**

The DIP switches provide a convenient source of inputs to circuits in your FPGA. When the switch is closed, it delivers a high output. When the switch is open, a pull-down resistor produces a low output.

**Reset Pushbuttons**

There are two reset pushbutton switches on the board. The pushbutton near the PIC resets the PIC, and the one near the FPGA resets the FPGA. Upon reset, the PIC will simply restart its program and begin as if it has just powered up. The FPGA, however, will have its memory cleared. If the PROM is programmed, the FPGA will immediately reprogram itself from the PROM. You will notice that these pushbuttons actually have 4 pins. However, only two pins are used, while the other two are left unconnected (and simply provide mechanical stability).

The FPGA is configured by reading a pattern of bits into internal SRAM specifying the logic and interconnections inside the chip. This configuration is loaded onto the PROM through the JTAG cable.

**Debugger/Programmer Jacks**

There are two programming connections at the top of the board. The double row of header pins (marked JTAG) are for programming the FPGA. The 6 pin jack (which resembles a telephone jack) is for programming the PIC microcontroller via the In Circuit Debugger (ICD 2) dongle. You will program and debug your PIC in Lab 4.

**LED array**

There is a 10 segment red light-emitting diode (LED) array on the board. The ‘ON’ LED (far left LED) simply tells that the board is powered ON and receiving 3.3 V. The LED immediately to the right of the ON LED is unused. The other 8 red LEDs can be driven by the FPGA or PIC. 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. If a voltage

is applied at one of the LED pins, the corresponding light is off for ground (GND) and on for voltage 3.3V ($V_{cc}$).

There are a few concerns that must be addressed whenever using LEDs. One is that LEDs glow nicely when given about 5 mA of current but may burn out 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 $\sim \frac{3.3V}{330\Omega} =$ 10 mA of current to flow.

**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 the Programmable Read Only Memory (PROM) on board. On powerup, the FPGA will serially configure itself from the PROM if the PROM has been programmed. It will also reprogram itself from the PROM when you push the reset button.

## 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 first 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 Vin 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.

The FPGA and PROM should already be soldered to your board. These are fine-pitch SMT components that require practice to attach without creating solder bridges between pins.

CMOS components such as the FPGA are sensitive to static electricity. Before you touch your board or solder anything, discharge your body by touching a large metal object. This is good practice when handling electronic components in any lab.

When you begin soldering, moisten the sponge. When the iron first heats up, tin the tip by applying a generous amount of solder all over the tip, then wiping off the excess on the sponge. Periodically repeat as you work to keep the tip looking silvery rather than black and blistered. 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 component pin rather than balling up on the component. 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 your connections are intermittent or unreliable. Some of the components will be soldered close to vias (the small holes through the board used to connect between wiring layers on the printed circuit board). 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 across the room. 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 good practice 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 – R10 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 5 330 Ω resistors and are installed at R12 and R11. 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 Voltage Regulators and Oscillator**

The next step is to insert the voltage regulators and oscillator. These parts are surface mount (SMT) parts. Soldering a surface mount device would be easiest if you had three hands: one to hold the part with pliers, a second to hold the soldering iron, and a third to apply the solder. One solution is to get a friend to hold the part, taking care not to apply the soldering iron to the friend. Another is to apply a blob of solder to one of the pads on the printed circuit board before the part is placed. Then place the part with the pliers and use the soldering iron to melt the blob onto the pin to hold the part in place. Solder the remaining pins to their pads. Then apply more solder to the first pin to be sure a good joint is formed (a gray grainy appearance indicates an unreliable cold solder joint, while a silvery appearance covering the pad and pin usually means a good connection).

The oscillator, located at U5, is a 4-pin chip, and its location is next to the PIC socket. This location is marked with U4. Be sure to align the notch on the board with the notch on the silkscreen. Solder the surface mount parts from the top of the board. Be sure not to fill neighboring vias.

Next add the voltage regulators, labeled U8, U9, and U10. Be sure to put each voltage regulator in the correct location. If you do not, your board will not work and you may burn out the FPGA. Refer to the schematic for the correct parts. The large pad on one end of the voltage regulator requires a substantial amount of heat to warm up, so you will need to apply the soldering iron for longer than usual. Be sure that you form a good solder joint between the large pad and the tab on the regulator; it is a common place for poor connections.

**☐ 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.

The advantage of the socket is that you can easily replace your PIC should you accidentally burn it out.

**☐ Insert the ICs**

Install the 74AC74 flip-flop chip (U4) 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 shows a notch and an indication of pin 1. It may be helpful to bend the rows of 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.

### ☐ Install the Pushbutton Switches

There are two small pushbutton switches that come with your kit, which will be installed in SW5 and SW6. 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 the board because the sudden popping into position could stab your finger. 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 SW1-4. Align the writing on the DIP switches with the silk screen writing on the board.

Finally, insert the LED Array at U6. Align the notch on the part with the notch on the board for correct alignment.

### ☐ Insert the Bypass Capacitors

The voltage regulators supply current sufficient to meet the average demand of the circuits on the board. However, they cannot respond quickly enough to deliver spikes of current at high frequency, such spikes would cause the voltage to briefly droop. Instead, this current is drawn from bypass capacitors on the board, keeping the voltage at the desired level. A typical digital system uses several sizes of bypass capacitors. Larger capacitors store more charge and can deliver more current, but react more slowly. Smaller capacitors can rapidly deliver charge to handle short spikes of demand. Overall, the bypass capacitors reduce the noise on the voltage supplies. Note that the chips and board have some inherent capacitance as well, so the voltage might be stable enough even without bypass capacitors. However, noise-related problems are so difficult to reproduce and fix that it is always better practice to put a generous number of inexpensive bypass capacitors on the board than to skimp on bypassing and hope the board works anyway.

There are three 0.01 µF capacitors, four 0.1 µF capacitors, and three 10 µF capacitors to install. Install the 0.01 µF caps in C1, C4, and C7 (polarity does not matter). Install the 0.1 µF's in C2, C5, and C8, and C10 (again polarity does not matter). Now install the 10µF's in C3, C6, and C9 (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 positive terminal (+) 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 negative (-) lead. If you install an electrolytic with the incorrect polarity, it can leak or explode. Small capacitors are often labeled with an three-digit code similar to resistors. For example, 223 indicates $22 \cdot 10^3$ 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 the bottom side. A 7x2 row of header pins should be installed in the slot marked JTAG. This will allow you to program the FPGA with the Xilinx Parallel IV 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 J1. This will be for selecting the clock speed. Also put a jumper horizontally on two of the pins marked 20 MHz. Finally, install a 10x1 female header at U11. This allows access to some more pins on the FPGA.

### ☐ The FPGA and PROM

The FPGA and PROM have already been installed for you. If your board malfunctions and you suspect a solder bridge, it is a good idea to take a look at these under the microscope to check the soldering.

### ☐ Install the ICD connector and the DC Jack

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

### ☐ 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 above the bar of LEDs and is marked SW7. 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 or to vias (solder bridges) and cold solder joints (gray, dull joints).

After you've checked your soldering and verified with a multimeter that there is no power to ground short on the board, 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.

### ☐ Test the Power Supply

After you've placed the utility board, connect a red wire from the red Vin banana plug connector to the upper power row. Then, from this row, connect a wire to the row of pins next to the Vin pin on the utility board. Then connect a black wire from the GND banana plug connector of the breadboard to the second power row (below the red wire). Connect this row to the Gnd pin of the utility board. Make sure these wires do not touch each other.

Turn on a benchtop power supply and check the 0-6 V scale indicates 5 V. Adjust it if necessary. Please keep the supply set to 5 V. However, it is good practice to check the supply before connecting it to your board because the knobs occasionally get bumped and you don't want to connect an incorrect voltage that damages your board or causes erratic operation.

When the power supply is set correctly, turn it off. Get a red and a black pair of banana plugs and connect the power supply to your breadboard (never connect power circuits while the supply is turned on). Turn the supply back on. The power LED should turn on. Feel your FPGA and make sure it is not getting warm. Also check your other components, especially the voltage regulators. The current from the supply should be close to 0, certainly well under 100 mA. Check the 3.3 V output pin and confirm that the voltage regulator is producing the correct value. Also check the 1.2 V and 2.5 V outputs.

### ☐ Test the LEDs and switches

You can test your LEDs and switches by connecting a wire in the breadboard between one of the switch outputs and one of the LED inputs. Toggle the switch and check that the LED toggles. Repeat to check all the LEDs and switches. Remove the wire when you are done so that it doesn't interfere with subsequent testing of the FPGA. If anything

is amiss, turn off the power supply immediately before damaging any parts. Look for solder bridges on your board.

☐ **Clean Up**

Clean up your lab station. Discard the refuse you accumulated while soldering. Tin the iron and turn it off. Please try to keep the lab clean and neat as you work because you share it with many others.

## FPGA Design

Your first goal is to write some Verilog modules to test the hardware on your board and operate a 7-segment display. By writing a module to control the LEDs based on the switches and clock, you will show that these components work and also that the FPGA, PROM, and power supply are functional. The system should have the following inputs and outputs:

| | | |
|---|---|---|
| clk | input | the clock input |
| s[3:0] | input | the four DIP switches |
| led[7:0] | output | the 8 lights on the LED bar |
| seg[6:0] | output | the segments of a common-anode 7-segment display |

The following tables define the relationship of the LEDs to the switches and clock:

| S0 | LED0 | LED1 |
|---|---|---|
| 0 | OFF | ON |
| 1 | ON | OFF |

| S1 | LED2 | LED3 |
|---|---|---|
| 0 | OFF | ON |
| 1 | ON | OFF |

| S2 | LED4 | LED5 |
|---|---|---|
| 0 | OFF | ON |
| 1 | ON | OFF |

| S3 | S2 | LED6 |
|---|---|---|
| 0 | 0 | OFF |
| 0 | 1 | OFF |
| 1 | 0 | OFF |
| 1 | 1 | ON |

| CLK | LED7 |
|---|---|
| 0 | OFF |
| 1 | ON |

**Table 1 – I/O requirements**

The 7-segment display should display a single hexadecimal digit specified by s[3:0]. Remember that you will be using a common anode display. The anode (positive terminal) of all of the LEDs is tied to 3.3 V through a single (“common”) pin. Each segment’s cathode (negative terminal) is connected to a pin. Therefore, you will need 7 separate control signals. Remember that a logic 0 applied to the cathode will turn on the segment. The segments are defined as shown below. Let seg[0] be A and seg[6] be G.

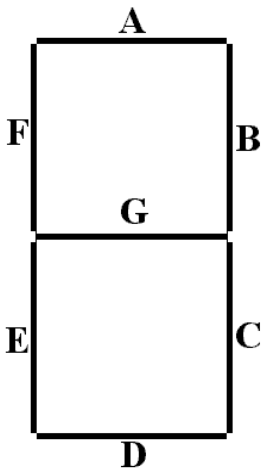


Figure 2 – Seven Segment Display layout

Launch the Xilinx Project Navigator. Create a new HDL project named lab1_xx, where xx are your initials. A good place to store the project is on your Charlie directory. It should be mapped to a letter drive and there should be no spaces in the path name (this is a good practice with any CAD tool.)

Set the product family to Spartan3, device to XC3S400, and package to TQ144. Set the synthesis tool to Synplify Pro (VHDL/Verilog) and simulator to Modelsim-SE Mixed. These are top-of-the-line 3rd party tools for FPGA design used by many companies today.

Create modules to perform these functions. Use a reasonable amount of hierarchy. Name the top level module lab1_xx.v. You will reuse the 7-segment display code in future labs, so it should be a module of its own. Every module should begin with your name and email address, the date of creation, and a brief summary of its purpose, so that somebody else can understand what the module does and get ahold of you if they need help. Comment the modules as appropriate.

## FPGA Simulation

When you are done, simulate the system in Modelsim and prove that it does what you intended. Create a new Test Bench Waveform source named lab1_tbw and associate it with your top-level module. The clock should come up with a default waveform alternating every 100 ns. Click on the s[3:0] waveform around 100 ns and use the Pattern Wizard to produce a Count Up pattern for 16 cycles. Set the simulation duration to 4000 ns. Save and close the wizard.

Configure Xilinx to find the path for the ModelSim simulator. Go to Edit • Preferences. Under the ISE General • Integrated Tools tab, set the Model Tech Simulator to C:\modeltech_6.4\win32\modelsim.exe (or wherever you installed it, if you are working on your own computer).

In the "Sources for" panel, choose "Behavioral Simulation." Click on lab1_tbw and choose Simulate Behavioral Model. Xilinx should launch the ModelSim simulator.

Unfortunately, it seems to default to simulating for only 1000 ns. In the transcript pane, type run 3000000 to run for another 3 million picoseconds (3000 ns) to complete the simulation. Choose Wave * Zoom * Zoom Full or click on the Zoom Full icon to see all the waveforms. Make sure the outputs match your expectations.

Get into the habit of watching the console pane at the bottom of the Project Navigator window. Learn what messages are normal and look for abnormal messages; this will save a good deal of time and frustration when something acts up.

## Pin Assignment

Next, assign pins to relate the signal names in your Verilog code to physical pin numbers on the FPGA.. Be sure the top-level source file is selected. In the User Constraints process, select Floorplan IO – Pre-Synthesis. You may get some spurious error messages, but the Xilinx PACE Constraints Editor should open.

The FPGA pinouts are shown in the Utility Board Schematic. Most of the user input/output (I/O pins are tapped out to the headers and labeled on the board silk screen. Some have special functions; for example, FPGA pin P97 is connected to LED0 and also to the PIC port RD0. The clock is connected to pin P124 and not to the board because a 40 MHz clock would be degraded by the parasitic capacitance and inductance of the breadboard.

In the Location column (labeled Loc), assign each signal an appropriate pin. For example, clk should be P124 and led[0] should be P97. 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 1, 2, 4, 5, 6, 7, 8, since they are contiguous on the PC Board. Now, under the I/O Std. field, select the voltage levels to use. Use the Low-voltage CMOS 3.3V levels – i.e. LVCMOS33. You can choose this from the select menu under "I/O Std". You can set all the pins at once by selecting them all, then right-clicking to bring up a menu and choosing Create Constraint… -> I/O Std -> LVCMOS33. Save your assignments and close the Constraints Editor.

## Generating the FPGA Configuration Files

Now you will synthesize your HDL and program the circuit you created into the FPGA. To do so, you will need to "burn" the PROM with your circuit, which will then load the FPGA on powerup.

In the Sources pane, change to Sources for Implementation. Be sure your top-level module, lab1_xx.v, is selected. Double click on "Implement Design" in the Processes pane. Once this is complete, you should see a green check mark beside the Synthesize and Implement Design processes. Check for messages in the console pane.

Choose Synthesize • Launch Tools • View RTL Schematic. Look at the top level module and then drop into each submodule. Make sure the hardware matches your expectations. Then close Synplify.

Look at the map and pad reports and make sure they match your expectations. Do the number of LUTs and I/O blocks match what you want? Are the pins assigned correctly and set to LVCMOS33 logic levels?

## Programming the PROM

With lab2_xx.v selected in the source window, double-click the Generate Programming File process. You should then see a green check mark beside the Generate Programming File option. This step produces a "lab1_dh.bit" bitstream file containing all the configuration information for the FPGA. The bit file must be converted to MCS format, then downloaded to the PROM over the Xilinx Parallel Cable IV using the JTAG[1] interface.

Click on "Generate PROM/ACE File" under "Configure Target Device" in the Processes for Source window. This will (slowly) open up a new window. Click on "Prepare a PROM File" in the first window. Choose Xilinx PROM, MCS format, and a Memory Fill Value of FF. Type in the PROM File Name, for example "lab1_xx". This will create a file called lab1_xx.mcs in your current project directory. Click on Next.

In the next window, select "I am using a Xilinx PROM in Serial Mode." Click on "Next," then "Auto Select PROM," then "Next," then "Finish." When prompted to add a device, choose the file with the ".bit" extension, such as "lab1_xx.bit". If there isn't a ".bit" file, you need to re-implement your design using "Implement Design" in the "Processes for Source" window. Then click No when asked if you wish to add another device file to the data stream (because you only have one FPGA on the board). Choose Operations -> Generate File… You should see a message indicating that "PROM File Generation Succeeded" and see an icon of a xcf02s PROM at 81% capacity.

## Testing Your Utility Board

Plug in the Xilinx Parallel Cable IV to the JTAG header pins. Line up the notch with the notch on the cable. Turn on the power supply.

Now program your FPGA. In the iMPACT window, double click on Boundary Scan. Then in the main pane, right click and Add Xilinx Device… Choose lab1_xx.mcs, your PROM file. Select the PROM part name xcf02s. Right-click on the PROM icon and choose Program… In the programming properties dialog, check the "Load FPGA" box so that the FPGA will automatically load itself after programming. Then click OK. You

[1] JTAG stands for the *Joint Test Access Group.* The JTAG interface is a serial interface used to communicate with chips on a board using a few pins during testing and configuration. JTAG is also called Boundary Scan.

should see the MCS file being downloaded to the PROM followed by a "Programming Succeeded" message.

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 PROM and program itself. If anything goes haywire, press the Xilinx Reset button to make the FPGA reload itself from the PROM.

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. Choose the manual mode for the clock by moving the jumper to Man. Toggle the clk manually using SW7, the large toggle switch.

Use a voltmeter 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.

If you have problems, the following checklist may help:

1. Check the voltage outputs on each of the voltage regulators.
2. Check that the power supply is hooked up correctly. Measure with the volt meter that you have 5 volts between the $V_{in}$ and GND pins on your FPGA board. Check that the current out of the power supply is less than 100 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.
3. If the Xilinx programmer software is unable to recognize the Parallel Cable IV and the current consumption jumps to about 500 mA when the cable is attached to the board, the cable is probably damaged. Label the cable as bad with a post-it and let the instructor know.
4. If the LEDs flicker on and off when you tap on the board or toggle switches, you probably have a loose connection or marginal solder joint somewhere. Two common problems in the past are inadequate solder joints on the voltage regulators and poor banana plugs that are too small for the recepticals.
5. If the LEDs do something but do 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.

## 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 as to the on-board bank of LEDs. You can limit the current into each segment of the display the same way you did for the

LEDs on board, 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.

Consult the data sheet in your lab manual for the pinout of the common anode dual seven segment display. All seven segments share the same anode, which should be connected to VCC (3.3 V). Each of the segments has its own cathode, which can be pulled to 0 to turn on the segments.

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 3.3 V. Then connect the input pins of the same side of the display to the header pins you chose. Remember to add suitable resistor between each of the inputs to the display and the header pins. These LEDs are common anode LEDs. That is, all the anodes from the LEDs are connected to a single $V_{DD}$ ($V_{DD}1$ or $V_{DD}2$). You are driving the cathode of each LED. Given this information, you might need to modify your Verilog file. Do so in the simplest way possible.

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 4.

## What to Turn In

Turn in your lab writeup including the following information:

- Verilog code
- Schematics of your breadboarded circuits sufficient for another engineer to reconstruct the circuit on the breadboard.
- How many hours did you spend on the lab? This will not count toward your grade, but will help calibrate the lab for the future.

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:

This lab manual and previous utility boards have been developed by Prof. Sarah Harris, Prof. David Money Harris, Elizabeth Reynolds '02, Marty Weiner '02, Fernando Mattos

’00, and Eliot Gerstner ’99. Nate Pinckney, Raj Roy, Kevin Lloyd, Dan Rinzler and Dan Chan helped test the current version of the board during Spring 2005.

## Appendix: FPGA Initialization Sequence

If your FPGA board is assembled incorrectly or your PROM 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 32 through 40 of the Spartan 3 Data Book.

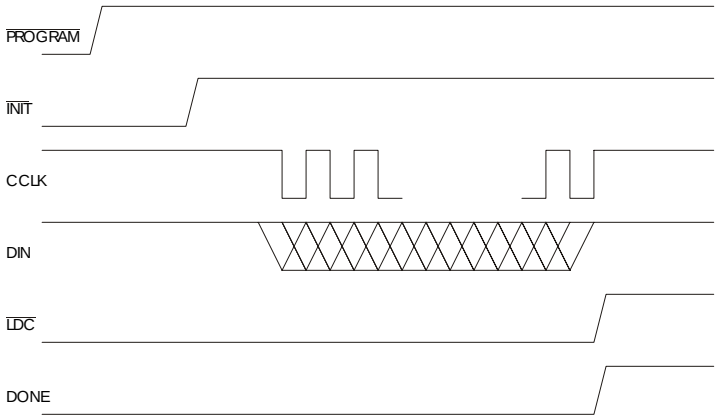
The configuration is set to Master Serial Mode because pins M0, M1, and M2 (pins 37, 38, and 39) on the FPGA are grounded. The FPGA generates appropriate control signals to drive the PROM, which serially transmits the configuration information into the FPGA. Other configuration modes could be used if you were trying to configure multiple FPGAs at once.

When you press the Xilinx reset switch (SW6), the PROGRAMb[2] 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, and DONE low. It also drives CCK (Configuration Clock) high. The FPGA drives INITb high and the serial PROM in turn sends data to the FPGA on DIN.

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 6 MHz. The PROM responds by sending appropriate data signals. The XC3S400 FPGA has 1,699,136 bits of internal state, so at 6 MHz the PROM will require approximately a fraction of a second to configure the FPGA.

When configuration is complete, CCLK stops toggling and becomes high. The DONE signal goes high to indicate successful configuration. If this signal is not both high, the FPGA was not configured.

The configuration waveforms are shown below:




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 PROM may be misconfigured (try somebody else’s and see if it works) or your board may have a bug.

[2] 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).