LAB #6: Registers, Counters, and a Simple Processor

This lab is due at 5pm in the EE locker for CS/EE 3700 on Tuesday, March 27, 2001.
The hardware should be demo’ed to a TA before 5pm on Tuesday, April 3, 2001.
NO LATE HOMEWORK WILL BE ACCEPTED.

1 Laboratory Objectives

This laboratory covers chapter 7 of the book on flip-flops, registers, counters, and a simple processor.
In addition to several book problems, you will wire up a simple processor similar to the one in Figure 7.70 of the book. You will use your adder/subtractor from Lab 4, and add registers and a simple control circuit.

2 Switches and Contact Bounce

In this lab, you will use the “pushbutton switches” for the first time. Each “pushbutton switch” is mounted on a 16 pin wire wrap socket. The pinout for this switch (as mounted in the 16-pin socket) is shown in Figure 1a. Note that the “top” (pins 1 and 16) corresponds to the exposed portion of the wire wrap socket. The switch is shown in its normal position.

![Diagram of pushbutton switch and toggle switch](image)

Figure 1: Switches: (a) pushbutton switch, (b) toggle switch.

The pushbutton switch will be used in your design as the clock signal. You will again wire the switch up in the configuration shown in Figure 2b. In other words, you should wire pins 5 and 12 to GND and pins 7 and 10 to VDD through a resistor. When the button is not pressed, pin 7 will be high and pin 10 will be low. Therefore, pin 7 will be \( \overline{clk} \) and pin 10 will be \( clk \). Therefore, when you push the button, you will cause the clock to go from low to high.

An important problem associated with any mechanical switch is a phenomenon called contact bounce. When a switch is changed, one would hope that the switch makes/breaks contact and remains in that state until it is changed again. Unfortunately, this is not the case. The mechanical contacts make and break the connection several times in rapid succession before settling down. Thus, the signal from the switch “bounces” up and down several times before reaching a stable value. This uncertainty typically lasts a few tens of milliseconds. In computers, this is a very long time, and if no precautions were taken, the computer could mistakenly interpret one press of the switch as several, since it interprets the contact bounce as a rapid sequence of switch changes. If the switch corresponded to a keyboard, each time you typed a key, several letters would be read
and echoed back to the display. Needless to say, this would be a very aggravating problem. You obviously do not want your clock signal to bounce like this.

A "debouncing circuit" is needed to get around this problem. The circuit you should use is a pair of cross coupled NAND gates (see Figure 3). Why does this work? According to Figure 3, there are two input signals S, for Set, and R, for Reset. Note that these signals are active-low. That is, a 0 value on the signal activates its function. To make the device Set, you should put a 0 on the S signal. So, with the switch set to provide a '1' on the S (set) input, and a '0' on the R (reset) input, the output (X) will stay in the '0' state because it is being reset by the active-low reset signal. If we now begin to change the switch, and oscillation is between R=S=1 and R=0, S=1; i.e., we repeatedly tell the circuit to reset or stay in the same state. Since the current state is already 0, nothing changes at the output. Upon pressing the switch further, the switch breaks contact completely (making the inputs R=S=1, i.e., maintaining the current output) before beginning to touch the other switch contact. When contact is first made with the other switch position, the output goes to the 1 state. Oscillations now are between R=S=1 and R=1, S=0, i.e. either set the output to 1 or do nothing. In either case, the latch output remains stable at the set state. A similar sequence of events occurs when we change the switch in the other direction.

![Figure 3: Debouncing circuit.](image)

3 Registers

The chip part 74HC374 is a multi-bit storage element. In reality, it is just an octal (8 in a package) D-type edge triggered flip-flops. Read about the 74HC374 in your data book handout. In particular, determine when the inputs are sampled (On what part of the clock?). Also, check under what conditions the outputs are valid. The '374 has tri-state outputs. When the "output enable" pin of the chip is low, the outputs of the latch behave normally; when it is high, the output pins are essentially disconnected from the internal circuit. This allows many outputs to communicate on the same bus (just as with the open collector outputs). Study the internal circuitry of the tri-state output. Consider under what condition you are allowed to connect several tri-state outputs to the same wire.
Much of the work that a processor performs is not computation, but moving data around. The circuit in Figure 4 illustrates a portion of your design for this lab. When a processor is working with a piece of data, it typically holds that data in a register. In reality a register is nothing more than a collection of latches or flip-flops. To transfer data between registers, one register puts its output onto a bus (in this case, a 3-bit bus). Another register can then take its input from that very same bus. Any number of registers can transfer data over the common bus, as long as only one set of register outputs is writing data onto the bus at any one time.

![Figure 4: Two registers on a tristate bus.](image)

4 Your Simple Processor Design

In this lab, you are to design, simulate, and implement (using your labkit) a simple 3-bit processor similar to the one shown in Figure 7. The design will only have two 3-bit registers (R0 and R1) rather than the four in the book design (You will also need the A and G register for intermediate results). The schematic for these two registers is shown in Figure 4. You can assume that the X and Y register specifiers is one bit each, so you do not need decoders for them. Otherwise, the design is very similar.

You should use the 3-bit adder/subtractor from Lab 4. If you did not get this to work (or have disassembled it), you may use an xor chip and the 4-bit binary adder chip instead (be sure to tie A4 to A3 and B4 to B3 to do a sign extension to 4-bits).

In this design, you are free to use any chips provided in your lab kit (i.e., registers, counters, decoders, etc, and not just simple logic gates). Use a pushbutton switch for your clock and toggle switches for the 3-bit data input, w input, 2-bit function input, and 2 1-bit register specifiers. Display the contents of the bus and the done signal using LEDs.

5 The Chassis and Backplane

You may want to (though are not required to) use multiple boards in this lab. A description of the chassis and backplane was given in lab 4 and is reproduced here for convenience.

The attache case will serve as a chassis for your hardware. The case is equipped with a 5 volt power supply which is controlled by the on/off switch on the bottom of the inside of the case. An AC line cord is also at the bottom of the case near the switch.

Let us first examine the top of the motherboard. Orient the attache case so that it is open in front of you. Remove all four wire wrap boards, exposing the four green edge connectors. In addition to the edge connectors, notice that there are five sets of silver jacks (see Figure 7). The two isolated ones in each set are connected to ground. Of the five jacks sitting side-by-side, the center one is connected to 5 volts. You will not use the others in this course.

Unfortunately, the wire wrap boards are symmetrical, and can be placed into the edge connector with the component side to the right or the left. If you are ever in a position to define a board's geometry, never design it this way, since this leaves the possibility that a clumsy technician will insert the board backwards, often destroying much or all of the circuitry on that board or even other boards. ALWAYS ORIENT THE BOARD SO THAT THE COMPONENT SIDE IS TO
THE RIGHT WHEN THE CASE IS OPEN IN FRONT OF YOU. The edge connector pins on the left side of the connector (see Figure 5) will correspond to numbered lines on the backplane, and those on the right to lettered lines. As you look into the top of the motherboard, pin 1 is the “top left pin” of the edge connector (the one furthest from you as you look into the case). Pin 2 and pin 27 carry ground. Pin 28, the bottom left pin of the connector, carries 5 volts, as does the pin directly across from it (bottom right corner of the connector). These numbers should match those near the wire wrap sockets soldered into the board (on the wiring side) when the board is inserted into the edge connector with the component side to the right.

Now insert a wire wrap board into the edge connector with the component side facing to the right. Notice that the ground inputs from the edge connector, pins 2 and 27, are connected to most of the foil on the wiring side of the board. Notice however, that there are a few islands of foil on the board which are not connected to ground. In particular, locate the two posts just above the rightmost socket (soldered to the edge connector lines on the board), as shown in Figure 6. Look on the component side of the board to see where these posts are connected to. They should be connected to the pin on the bottom right hand corner of the edge connector, which is in turn wired to 5 volts. These two posts will carry the 5 volts from the backplane to the individual chips on the board.

Now let us examine the bottom of the motherboard. Lift up the board and attach it to the magnets on the inside of the top of the attache case. The large bands of silver foil are connected to ground. From this perspective, pin 1 is the bottom (closest to the hinges of the case) left hand pin of each edge connector, and the 5 volt pins are at the top (furthest from the hinges). Note the positions of the ground pins (there are 4 on each edge connector). Don’t worry if some of these pins do not appear to be soldered to the silver ground foil, although at least one pin of each edge connector should be connected to ground. Note that most of the pins on the edge connector are
floating, i.e. are not wired to anything. You will have to wirewrap many of these pins to each other, as will be discussed later.

This completes our discussion of the chassis. You should now be familiar of how the boards and the backplanes are wired, and have some understanding of what additional wiring will be required.

6 Problems

1. Problem 7.7
2. Problem 7.11
3. Problem 7.17
4. Problem 7.35
5. Hardware design
   (a) Draw a schematic in PowerView for your simple processor. Be sure to only use gates available in your labkit. Turn in a printout of your schematic.
   (b) Simulate your design until you are confident that it works correctly. Turn in all your command and log files.
   (c) Wire-up your simple processor.