LAB 5: A Debounced Switch and Counters

OBJECTIVES
To understand the design, function and operation of a debounced switch and simple counter circuits.

MATERIALS
- Your entire lab kit
- Suggested Quartus Components
- In “others | maxplus2” library
  - 7474: Dual D-flip flops
- In “primitives | storage” library
  - dff
- In “primitives | logic” library
  - not, and2, or2, bor2, etc.
- In “primitives | pin” library
  - input, output
- In “primitives | other” library
  - vcc, gnd
- Bring your printouts from lab 4 to this lab:
  - UF-3701 Programming WARNING!
  - 3064 Pin Definitions
  - MAX 3000 specification sheet

INTRODUCTION

NOTES ON UF-3701 BOARD AND QUARTUS
The ground connection is the most important one for the UF-3701 board. Connect the ground pin (labeled G on your UF-3701 board) to the ground on your breadboard. Similarly, connect the power pin (labeled 5V on your UF-3701 board) to the power on your breadboard. Be very careful that you do NOT reverse these pins; if you do, the chip will be destroyed and you will need to buy a new one. (See the website for information on purchasing a new CPLD.)

Pin 43 on your UF-3701 board should generally be selected for your clock input. This is a global clock pin. Pin 2 may also be used.

DEBOUNCED SWITCHES
The switches that you have been using this semester are known as single-pole-single-throw (SPST) switches. When you move the SPST switch in a switch circuit from ON to OFF or from OFF to ON, the resulting output bounces around between low and high voltages for a short time. If the switch circuit output is used as a synchronizing signal (such as a clock) in a digital machine, weird things will happen. If the machine is a counter, the count may seem to jump wildly. This is obviously undesirable, so a debouncing circuit must be built. We discussed several debouncing circuits in class. You will design and build one of these debounced switch circuits using a single-pole-double-throw (SPDT) switch. You will use this debounced switch circuit in this lab and rest of the labs this semester. The SPDT switch has three aligned pins. The center pin is connected to one or the other of the outside pins, depending on the position of the switch. You can use your multimeter on the resistance (Ω) setting to verify the operation of this switch. There are two pins on the switch in addition to the three aligned pins. These two pins have no useful electrical purpose and are used for mounting only. These pins are electrically connected to the center pin of the three aligned pins, and thus must be removed, i.e., carefully bent or broken off of the device. (If these two pins are not removed, when the switch is plugged into a breadboard, the switch will have no effect.)

Your debounced switch circuit should use the two axial resistors (also known as radial resistors) in your lab kit. An axial resistor is shown here.

COUNTERS
A synchronous counter is a device that progresses through a known sequence with every clock input signal. The counter advances to the next state/number at a rising (or falling) edge of each clock pulse. The counter sequence is arbitrary, i.e., it may count up, down, or in some strange sequence. The counter you will design in this lab will have a custom count sequence with some special additional inputs.

PRE-LAB REQUIREMENTS

1. Make a debounced switch circuit for your clock input (as discussed in class) using your SPDT switch and other circuitry. You may use a NAND chip, a NOR chip, or a 74’74 chip in your design. I strongly suggest either a NAND or NOR chip, since they are generally cheaper than a 74’74 chip. You could even use the CPLD, although I don’t recommend it. Since you will use this debounced switch circuit for this and the remaining labs this semester, build your debouncer on a corner of your breadboard.

There is no easy way to test your debounced circuit. You can NOT design and simulate the debounce circuit that was taught in class in Quartus. There are no resistor components available. A voltmeter will not help, since the bounce rate is in the order of milliseconds. You could test it with a digital or storage oscilloscope if one was available. Instead, you will test your debounced circuit with a counter designed in the next part of the pre-lab.

2. Design a counter (shown in Figure 1) to count through the sequence 00, 11, 01, 10, 00, …

   ![Figure 1: Simple counter block diagram.](image)

   Note that there is only a single input (CLK) and two active-high outputs (Q1 and Q0).

   a. Make a next-state truth table. The “inputs” for this table are Q1 and Q0; the “outputs” are Q1+ and Q0+.
b. Using D-flip-flops, determine the next state equations for $D_1 = Q_1' = f(Q_1, Q_0)$. Use K-maps, if necessary, for each $D_i$ to get MSOP or MPOS equations. Note: There will be two 2-input K-Maps.

c. Design the required counter circuit in Quartus. (I suggest that you do it first on paper, but this is not required and will not be submitted.) I suggest that you use one of the below two possible D-flip-flops available in Quartus.
   i. Use “others | maxplus2 | 7474” for the left item in Figure 2.
   ii. Use “primitives | storage | dff” for the right item in Figure 2.

d. Simulate the circuit and, as always, annotate this simulation. Verify that your design counts as required with each rising CLK edge.

![Figure 2: Two D-FF available in Quartus.](image)

3. The above items, including circuit schematic (with CPLD pin numbers) and annotated Quartus simulation results should be part of the emailed lab document. (As usual, all pre-lab material must be emailed prior to the start of your lab. Be sure to also email your archive files.)

4. Test your 2-bit counter design, using your debounced switch for the clock input. Use appropriate LED circuits for the count (Q) outputs.
   a. Download your counter design to your UF-3701 board (also known as the 3064 CPLD breakout board) with **nothing else connected to the board** except power and ground. (See the first two paragraphs of Part 3 in lab 4 for more information.) Remove power from your breadboard.
   b. Move your UF-3701 board onto a pre-wired area with the given CLK input and count outputs. Reconnect power to your breadboard.
   c. Toggle the debounced CLK input switch to verify that your counter counts as expected. If your counter output does not exactly match the required count sequence as you toggle the switch input, then your debounced switch circuit is **not designed and/or built correctly**. Verify your debounced switch circuit design and construction. (The only way to easily test your debounced switch circuit is with a counter.)

5. Replace the debounced CLK input circuit to your counter and replace it with a normal (un-debounced) SPST switch input circuit for the CLK. Write down the outputs with 10 successive clocks. Compare each successive count to what you should get. How does counting with an un-debounced clock input compare to counting with a debounced clock input? Put this info in your emailed lab document.

6. Design a counter that will count forward with the following sequence:

   **000, 011, 010, 100, 111, 101, 000, …**

   This counter will also count backward in the reverse order:

   **000, 101, 111, 100, 010, 011, 000, …**

   A block diagram for the counter is shown if Figure 3.

![Figure 3: Forward/Back counter block diagram.](image)

Your counter can also pause the counting. These three modes (forward, backward, pause) will be controlled with 2-inputs: F(H) and B(L). When neither forward nor backward is true, the counter will ignore the CLK input and hold its count value.

F and B should **never** be simultaneously true, so your counter should deal with this case in the most cost-effective fashion, i.e., if you assume that a user will never make both inputs true, design the most inexpensive circuit that you can that accomplishes the required goals, i.e., use “don’t cares.” Note that the counter does not include $Q_2, Q_1, Q_0 = %001$ and %110, where % is a prefix for binary. These two counts should also contribute “don’t cares” in your next-state truth tables and K-maps.

Your counter should have a means to **asynchronously** set and clear each bit. SET(L) and CLR(L) are the inputs to asynchronously set and clear a particular counter bit. These SET and CLR inputs will allow you to start the counter at any desired count. (If you initialize your counter at the
count \(Q_3Q_2Q_1Q_0=\%001\) or \(\%110\), the next count is not specified in the problem description. The next count will be determined by the values selected for the “don’t cares” associated with these counts.)

As you may recall, when we first started discussing circuits with feedback, I stated that with these types of circuits it is often easier to deal with voltages rather than with logic. Let me suggest that you design this (and all) counter(s) with active-high state-bits (to generate the next-state circuits) and then generate the appropriate output circuits with the required activation levels. In this case, use active-high \(Q_2, Q_1,\) and \(Q_0\) in your design of the counter next state circuits. But when creating the final circuit, the outputs will be \(Q_3, Q_2, Q_1,\) and \(Q_0\).

Finally, the counter should have an additional output indicating the count is at a “special number.” Use “101” for this number. Output \(Sp(H)\) should be true when the count is \(Q_3Q_2Q_1Q_0=\%101\) (not when the next count is \(Q_3Q_2Q_1Q_0=\%101\)).

a. Make a next-state truth table with the inputs: \(F, B, Q_2, Q_1, Q_0\) and outputs \(Q_2^*, Q_1^*, Q_0^*,\) and \(Sp\). (Ignore the SET and CLR for now.)
b. Using D-flip-flops, determine the next state equations for \(D_i = Q_i.f(F,B,Q_2,Q_1,Q_0)\). Use a K-map for each output to determine MSOP or MPOS equations. Note: There will be three 5-input K-Maps.
c. Determine the equation for the output \(Sp\). Use a K-map to get an MSOP or an MPOS equation. Hint: From the counter description, does \(Sp\) depend on the inputs \(F\) and \(B\)?
d. Design the required counter circuit in Quartus. (I suggest that you do it first on paper, but this is not required and will not be submitted.) Don’t forget to include the SET and CLR inputs in your circuit. (In Quartus, choose D-FF’s with asynchronous Set and Clear inputs. Both of the D-FF’s shown in Figure 2 have asynchronous Set and Clear inputs.)
e. Simulate the circuit and add annotations.
   i. Verify that your design counts forward, counts backwards and holds the count with the appropriate input combinations.
   ii. Verify that each bit can be set and cleared by using the SET and CLR, \((i=0,1,2)\) inputs.
   iii. What is the next count for each value of \(F\) and \(B\) for \(Q_2Q_1Q_0=\%001\) and \(\%110\)?

7. As always, put your design and simulation in your lab document and email this file along with your design archives.

Note: The circuits for this lab must be built at home as part of the pre lab requirements. Leave a spot to program the CPLD on a corner of your board and spots for your

CPLD for both of your counter designs. (You could use the same location for both of our counter designs if you select the pins appropriately.)

**IN-LAB**

1. As always, bring a printout of your Summary document to lab.
2. Demonstrate your simple counter design from Pre-Lab part 4c to your TA. Show that it counts as required.
3. Demonstrate your simple counter design from Pre-Lab part 5 to your TA. Show that it does not always count as required.
4. Download your forward/back counter design to your UF-3701 board with nothing else connected to the board except power and ground. The ground connection is the most important one for the UF-3701 board. Connect the ground pin (labeled G on your UF-3701 board) to the ground on your breadboard. Similarly, connect the power pin (labeled 5V on your UF-3701 board) to the power on your breadboard.
5. Test your design with your debounced switch for the clock input. Use appropriate switch circuits for the other inputs and LED circuits for the count (\(Q_2, Q_1, Q_0\)) and \(Sp\) outputs.
   a. Verify that your design counts forward, counts backward and holds the count with the appropriate input combinations.
   b. Verify that each bit can be set and cleared by using the SET and CLR, \((i=0,1,2)\) inputs.