Project 2: Counter With Display

CS 200 • 20 Points Total
Due Friday, February 10, 2017

Objectives

- Design a modulo-10 (BCD) counter using JK flip flops.
- Design a BCD to 7-segment LED converter.
- Implement your designs using the Logisim software.
- Practice designing combinational and sequential circuits.

Overview

At every clock pulse a JK flip flop outputs one of the following four values based on its control inputs: a one, a zero, the same thing it output during the last clock cycle, or the opposite of what it output during the last clock cycle.

This makes it an ideal building block for various types of counters because we have something that can be reset when necessary and subsequently toggled at will. Each bit of a counter can be represented by one JK flip flop, and we can have each bit be set up to toggle if all the bits to the right are set to 1 AND a clock pulse happens.

A standard 4-bit counter could be called a modulo-16 counter, since it counts 0-15 (0000-1111) and then resets to zero. For part of this project we want to design a modulo-10 counter that counts 0-9 (0000-1001) and then resets to zero on the next count. This could also be called a Binary-Coded Decimal (BCD) counter since its value can always be represented by a single base-10 digit.

To display the output from a modulo-10 counter we can use a seven-segment display. The trick with these is that every segment is controlled via a separate output from a combinational logic circuit. So in the same way that sum and carry are separate functions for an adder, so too are top-right segment, top segment, center segment, etc. for a 7-seg display. You'll build a 4-bit BCD value to 7-seg converter as part of this project.

Requirements

BCD To 7-Seg Converter

Write a truth table that lists all combinations of a 4-bit input and has 7 columns of output - one for each of the segments on a 7-segment display. In each row of input, identify whether each segment should be on or off. Or, you could make 7 truth tables, each with 4 inputs and a single output, if combining them into one big table is confusing.
Write functions that describe the behavior of each segment. You can assume a decoder is available. For example, 4 bits of input would give you 16 lines of decoded output E0 through E15. The top-right segment (SegTR) on a 7-seg display is on for every number 0-9 except for numbers 5 and 6, so for the function SegTR you might write \((E_5 + E_6)'\). The 7-seg display we'll be working with also has an eighth decimal point segment; you don't need to use this so just keep it off: \(\text{Seg8}(x_3,x_2,x_1,x_0) = 0\).

You may have noticed that we didn’t account for numbers greater than 9 because we know our counter will work perfectly and never give anything to the display circuit other than 0-9. But just in case, you might want to account for the possibility that something messes up, so you can assume that 10-15 will not turn on the display. So in the example above, the function for SegTR would be \((E_5 + E_6 + E_{10} + E_{11} + E_{12} + E_{13} + E_{14} + E_{15})'\). You can also, for elegance’ sake, cause the middle segment to light up for 10-15, which will show a ‘dash’ when you are testing your counter and it gives you something other than the outputs for 0-9.

Create a "BCD To 7-Seg Display" converter circuit in Logisim. Have 1 4-bit input pin (which will represent a BCD number 0..9) and 2 4-bit output pins (these will hooked to the top 4 and bottom 4 input bits on a 7-seg display). Implement your individual segment functions in between. Recommendation: send the input bits to a decoder, have 8 separate functions that use the decoder outputs as inputs, and then bundling the groups of 4 output bits. Feel free to use any basic gates you want including NAND gates and NOR gates.

Create a "7-seg Test" circuit to test out all 10 display combinations that your converter should be able to handle. Create a 4-bit input pin, feed it into the converter (stamped as an abstract logic block in the figure below), then hook the outputs to a 7-segment display. Copy and paste your circuit 10 times and change the inputs to be each of the 10 different values. Here's an example of a display test for the input value "4":

In the figure above, the input into the display circuit (shown as an abstract logic block) is a 4-bit wire bundle. There are two outputs, each also consisting of a 4-bit wire bundle. The outputs are unbundled at the display using a splitter with wires to each of the display inputs. Notice that the bundles are black but the split wires are black if 0 or green if 1, and correspond to the lit or dark segments of the display.

**Modulo-10 Counter**

In Logisim, design a modulo-10 counter using JK flip-flops. The book shows a schematic of a modulo-16 counter on page 142; this can serve as the basis for your design.
Notice that the JK flip-flops in Logisim have two blue input pins on the bottom. These are "set" on the left and "clear" on the right. These are both asynchronous - for instance, when "clear" becomes active, the flip-flop will change its output to zero no matter what the J, K, or clock inputs are.

There are several approaches you can use to making the counter modulo-10. The easiest would be to trigger the asynchronous clear on all the flip-flops if the count is at 9 AND the clock is high.

Bundle the 4 output wires of your counter together and hook them to a 7-seg converter and display. Hit CTRL+T to step through the different values of your counter!

**Project Report**

The final step of this assignment is to create a report consisting of a cover page, an overview of the project, sample output, and the source code. See Assignment Policies on either the class website or Bb Learn.