Santa Clara University

Lab #8: Counters

ELEN 21L 51306 Tuesday 2:15-5pm
May 30, 2017

Group 1
Fradet Kelly
Yutong Li

Lab #8: Counters

I. Objectives:

In this lab, we will:

- Use D-type flip-flops to build both a ripple counter and a synchronous counter.
- Use a decoder to full decode a counter's outputs into individual count indicators.

II. Introduction:

In this lab, we will build a 4-bit ripple up counter and a 4-bit synchronous up counter, in which these two counters should function in the same way, which is counting from 0 to 1 to 2 … to F, and to 0 to 1… . Both counters have the same clock input, which is CLOCK_50, except for ripple counter, only the clock input in the first flip flop is from the CLOCK_50, and the other clock inputs are from the output from the previous d flip flop, while for the synchronous counter, all flip flops use CLOCK_50 as their clock input. We also add a led output in the clock of both counters to tell whether the clock is counter. What's more, we also add a decoder on the synchronous up counter, which can decode the 3 least significant bits, and show from a series of led. In the extra credit part, we add a reset button for both counters in different ways, which will be interpreted in the procedure section.

III. Procedure:

We extended our prelab ripple counter from three bits to four bits, and we let Q' be both the D input for the same D flip flop and the clock for the next D flip flop. We implemented this counter with two 7474s, which contain clocked flip flops. We also extended our prelab synchronous up counter. Then we made a truth table with the present states and the next states for each bit. The inputs to the D flip flops would match the next states because Q(t+1) = D for a D flip flop. We wrote K-maps to find the logic functions of each D input value based on the present state. We found the following logic functions: A = a'bcd + ab' + ac' + ad', B = bc' + b'cd + bd', C = c'd + cd', D = d'. We implemented this counter with the 8dff so that all the flip flops would be in one component compared to using two 7474s. We reused the clock counter components from the slot machine inspired lab we did last week in order to activate the timing of both counters. We connected this clock input and the D inputs of all the flip flops to LEDs for observation during testing. We sent both 4-bit outputs to seven-segment displays. The schematic with both the ripple and the synchronous up counters is shown below in Figure 1.

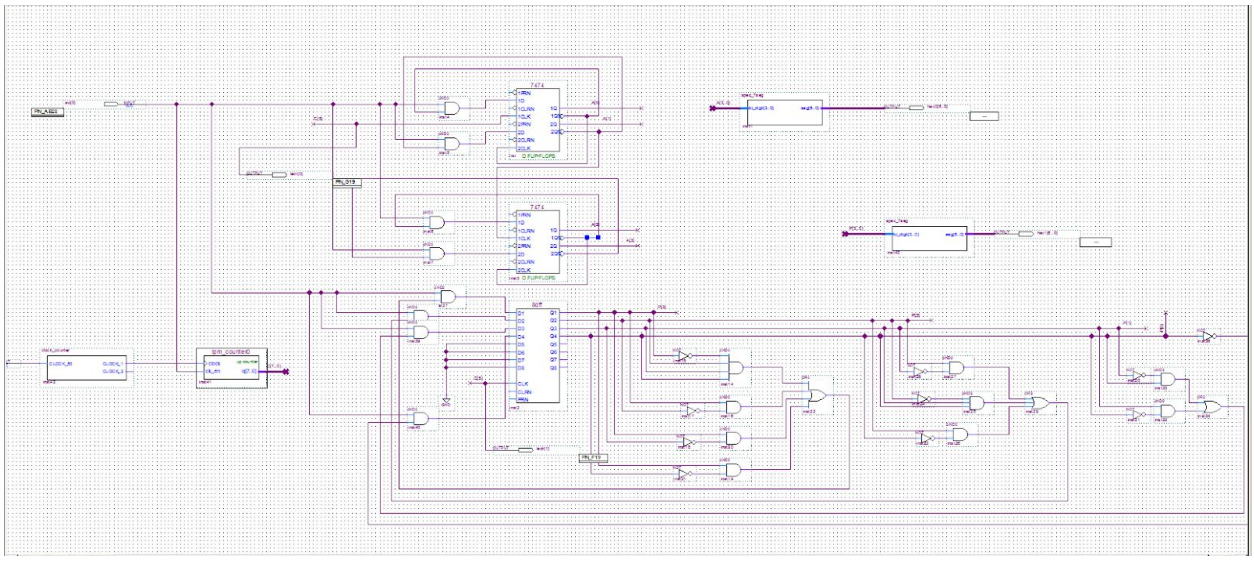

Figure 1. Part 1 Schematic Without Decoder Element

Then we added a 3 to 8 decoder element, shown below in Figure 2, in order to decode the three least significant bits of the synchronous counter output. We used the dec38 decoder. We connected the SENSE input to power to make it active high. INHB is grounded, to enable the decoder. We connected the three least significant bits, P[0], P[1], and P[2] to the select inputs. Then we connected all the Y[7..0] outputs to LEDs.

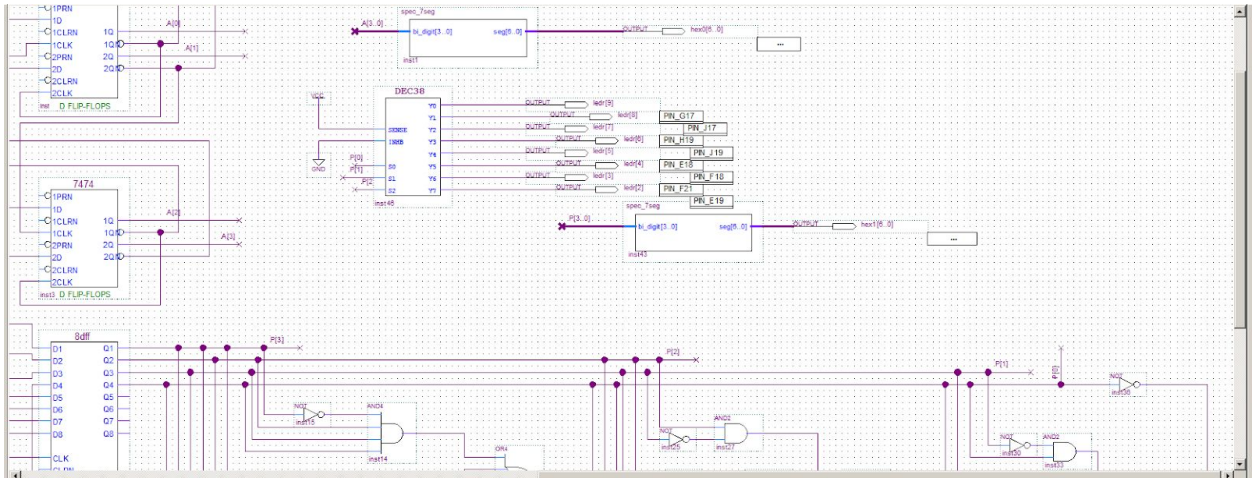

Figure 2. Part 2 Decoder Element

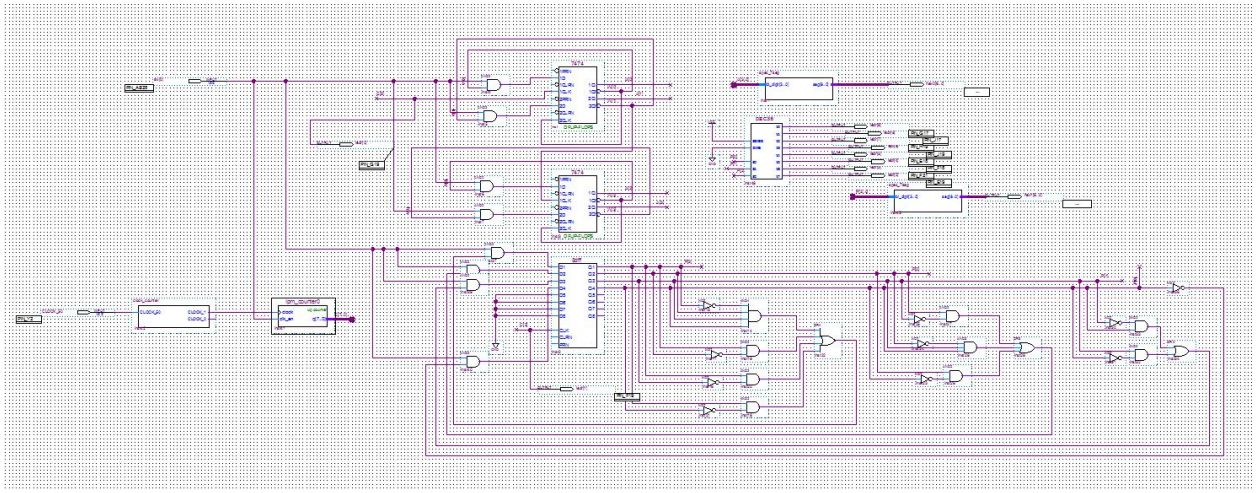

Figure 2b. Schematic with decoder element

In order to implement reset, for synchronous up counter, we add a series of 2to1MUX in front of the input of the dff. If the select switch is low, the original flip flop input is passed through to the MUX to the flip flop output for each bit. If the select switch is activated, the other input to the MUX, which is grounded, is passed through the MUX to the flip flop for each bit the make the value 0000, resetting the counters.

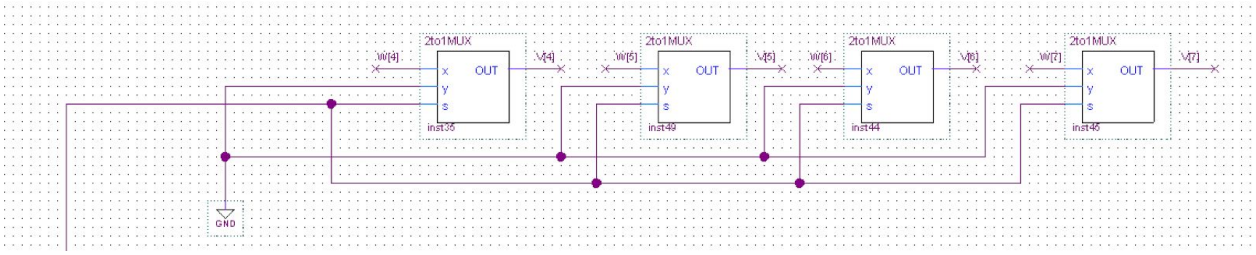

Figure 3. A series of 2to1MUX to implement reset on synchronous counter

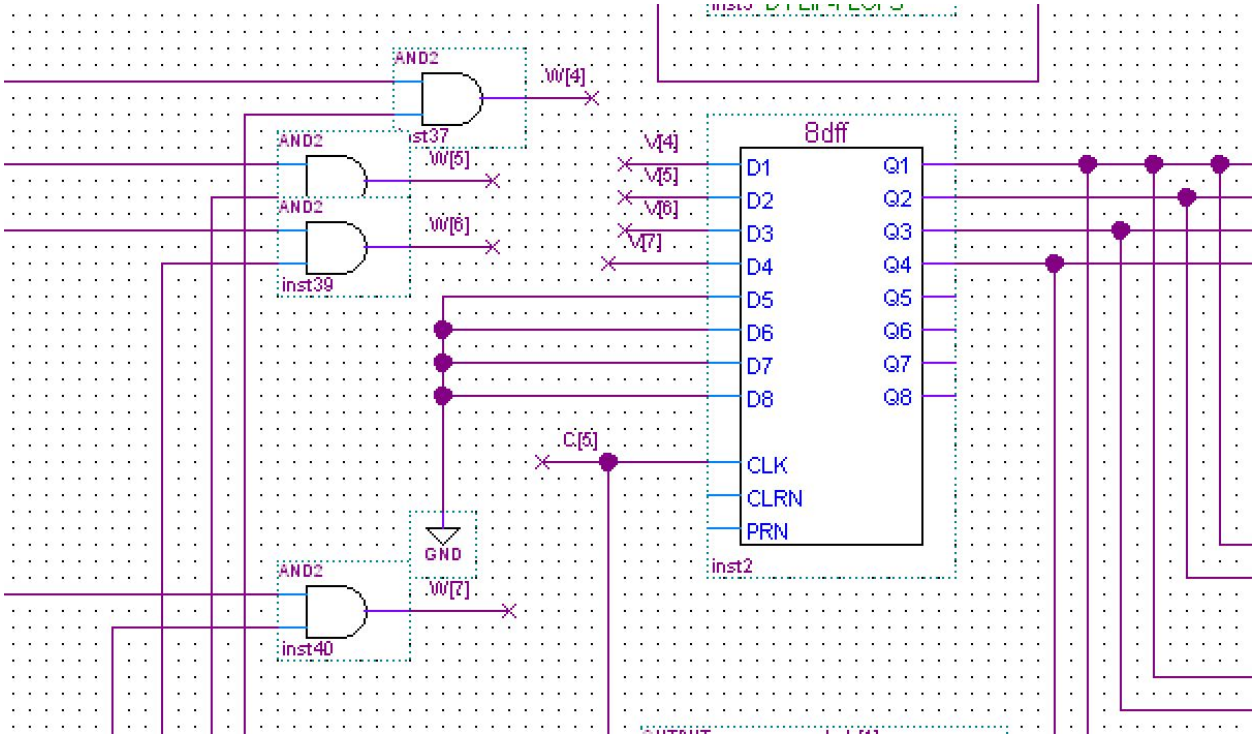

Figure 4. Details of the connection between the 2to1MUX and the 8dff

In order to implement reset, for ripple counter, we can't just add a series of 2to1MUX in front of the input because our ripple counters don't use the same clock, which means the counter is only going to stop when we reset the counter, and starts counting from the bit that it stopped. However, we can see that in the 7474 symbol, there is an input called CLRN, which has provided us with a reset input, so all we need to do is to connect these CLRN with the same switch that we have for synchronous up counter above.

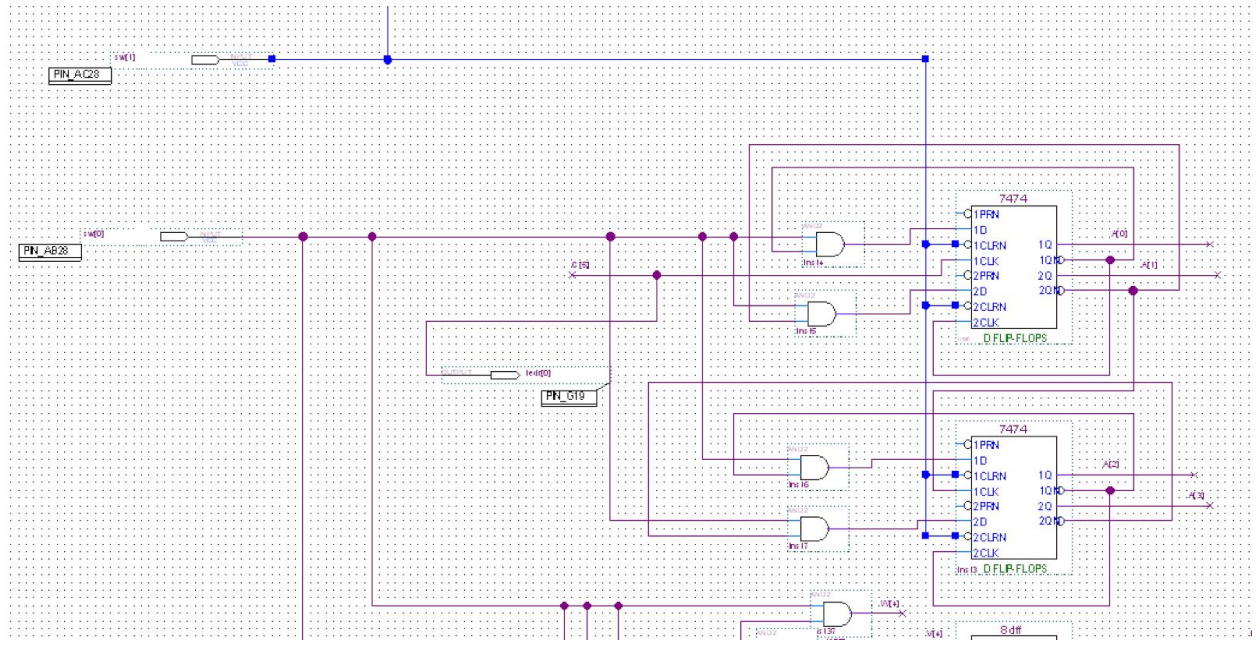

Figure 5. The connection between the reset switch and the ripple counter

IV. Results:

Part 1. Ripple counter and synchronous up counter

After downloading the circuit to the board, we can see that both counter work in the same way, counting from 0 to f again and again when the count_enable is high, and stop counting when the count_enable is low. Also, since we add a led output on the clock. When the count_enable is at high, the LEDs indicating that the counters are counting are blinking, and are off when the count_enable is at low.

Part 2. Adding a decoder

By adding a 3to8 decoder, using the three least significant bits from the synchronous up counter as switches, and assigning 7 adjacent leds to the outputs in which the top output is the leftmost led, and the bottom output is the rightmost led. After downloading the modified circuit to the board, we see that when the counters are counting, the leds are blinking:

- One at a time
- Sequentially
- Start from the leftmost led
- Start over from the leftmost led after the rightmost led is high
- Stop blinking and stay at the same led when the count_enable is at low

Part 3. Reset

After using a switch to implement the reset, whenever the count_enable is at high and the reset switch is at high,

- The bits in both of the 7 segment displays will become 0
- The led indicating the clock is still blinking
- The only led that is on among the series of led that we added in the decoder part will be the leftmost one, because the bit in the synchronous up counter becomes 0

When the count_enable is at high and the reset switch is at low,

- The bits in both of the 7 segment displays will count from 0-1-2-3-.....

V. Final Questions

1.

For ripple counter, if we extend it to 8-bit counter, generally we will need:

1. Four 7474
2. Two 7 segment displays

If we have D_0, D_1, D_2, and D_3 as the inputs for the 4 least significant bits,

$D_4 = (Q_4')$ (count_enable)

$D_5 = (Q_5')$ (count_enable)

$D_6 = (Q_6')$ (count_enable)

$D_7 = (Q_7')$ (count_enable)

For synchronous up counter, if we extend it to 8-bit counter, generally we will need:

1. To use all 8 inputs and 8 outputs in 8df
2. Two 7 segment displays

Let A, B, C and D refer to the input of the first four D flip flops, and let a, b, c, d, e, f, g, h be output of the D flip flops, in which a is that of the most significant bit.

A = a'bcdefgh + ab' + ac' + ad' + ae' + af' + ag' + ah'

B = b'cdefgh + bc' + bd' + be' + bf' + bg' + bh'

C = c'defgh + cd' + ce' + cf' + cg' + ch'
D = d'efgh + de' + df' + dg' + dh'

2.

In order to let the counter be able to count both up and down, we decide to use a 2to1MUX before the D input, and with a switch. If the switch is at high, the counter will count up and the circuit will be the same as the one that we already have. If the switch is at low, the counter will count down, and the inputs will be shown in the logic equations below.

For the ripple down counter, we just let Q be the input for the dff, and let Q' be the clock fro the next dff. As you can see in Figure 4, we have:

- Q AND count_enable as the input of the D flip flop,
  - $D_0 = Q_0$(count_enable)
  - $D_1 = Q_1$(count_enable)
  - $D_2 = Q_2$(count_enable)
  - $D_3 = Q_3$(count_enable)
- and Q' as the clock for next bit.

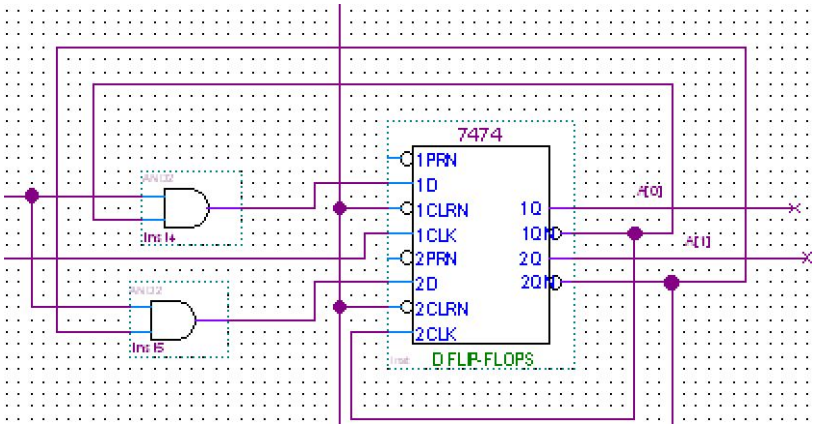

Figure 4. Part of the ripple up counter

If we want a ripple DOWN counter, all we need to do is have:

- Q' AND count_enable as the input of the D flip flop,
  - $D_0 = Q_0'$(count_enable)
  - $D_1 = Q_1'$(count_enable)
  - $D_2 = Q_2'$(count_enable)
  - $D_3 = Q_3'$(count_enable)
- and Q as the clock for next bit.

For the synchronous DOWN counter, where a, b, c, d are the bits store in the dff, and D_0, D_1, D_2, D_3 are the value of the input of the dff.

$D_0 = a'b'c'd' + ab' + ac + ad$
$D_1 = b'c'd' + bc + bd$
$D_2 = c'd' + cd$
$D_3 = d'$

VI. Conclusion:

Overall, the lab went well and we had some extra time to work on the extra credit in part 3. However, we still encountered a few obstacles. First, we initially had the output bits for the synchronous counter reversed, so it was not counting in order. Once we corrected that mistake, the counter worked. Then for the ripple counter, it was initially counting in order but down instead of up. We had the clock inputs of the flip flops coming from the Q of the previous flip flop. Once we switched the clock inputs to be Q' it counted up, but out of order. We analyzed the schematic and found that the last flip flop's input was not connected correctly. The ripple counter worked after fixing that mistake. A mistake we made when adding the decoder element was trying to skip an LED. Once we assigned the LEDs that were next to each other, the decoder LEDs made the correct pattern. The extra credit took quite a few tries to implement, but we eventually figured out how to design the reset capability using 2 to 1 Muxes. In this lab, we worked with 7474 counters and the dec38 decoder for the first time, and we also got more practice with the 8dff component. We also practiced original design to implement the extra credit.

VII. Reference lists:

- Dr. Sally Wood, Dr. Samiha Mourad, Dr. Radhika Grover. *Laboratory #8: Counters.* Spring 2017. Print
- Bin_7seg_temp.txt. Spring 2017. Print
- Dr. Sally Wood, Dr. Shoba Krishnan. *7-Segment Display Tutorial.* Spring 2017. Print
- Altera Corporation - University Program. *Quartus II Introduction Using Schematic Designs.* Spring 2017. Print
- Altera Corporation - University Program. *Quartus II Introduction to Simulation of Verilog Designs.* Spring 2017. Print
- Terasic Technologies Inc. *DE2-115 User Manual.* Spring 2017. Print.