Lab 6
Simon Carballo
1618309
11/29/2020
Section D

Lab6 Write Up

- **Description:** In this lab we were tasked to design a sequential circuit that detects when a turkey goes left or right between two IR sensors and adds/subtracts a count depending on the direction the turkey goes. In this case we are using buttons as a replacement to the IR sensors. This time the challenge of the lab is to use our past knowledge to design this system.

- **Design:** For this design we have the Top Level that calls for multiple subsections which includes: Clock(Provided), StateMachine, 8-bit Counter, Two's Complement Converter, 2to1x8MUX, Ring Counter, Selector, and the Segment_Display Converter. As we go over each subsection we will look more into depth of how each section is designed.
    - Clock:
      This code is provided by the lab manual. This module intakes the basys3 clkin value and outputs the global clock and reset for all of the counters in the Top_Level. It takes in the inputs of clkin from the constraint file, and manipulates it to become the net clk for the sequential design. We also use this to get Dig_sel for the ring counter (Explained in Ring Counter Module). In this lab we also use the qsec, which outputs the clk signal which is high for one clk cycle every ¼ of a second making it possible to be used as a different form of clk.

    - State Machine;
      The entire lab is operated using a state machine that functions with D Flip-Flops and control logic. In this lab I built to control logic using 7 different states: IDLE, L, L_B, LR, R, R_B, RL. The states were controlled with two inputs: Left, and Right. These states were used to determine the location of the turkey and whether or not to count up, down, or not at all. My control logic was built using the following boolean algebra using one-hot encoding:

        - IDLE:

PS	L(Left)	R(Right)
Q0	0	0
Q1	0	0
Q3	0	0

Q4	0	0
Q6	0	0

D0 = ~L*~R*(Q0+Q3+Q4+Q6)

- L:

PS	L(Left)	R(Right)
Q0	1	0
Q1	1	0
Q2	1	0

D1 = L*~R*(Q0+Q1+Q2)

- L_B:

PS	L(Left)	R(Right)
Q1	1	1
Q2	1	1
Q3	1	1

D2 = L*R*(Q1+Q2+Q3)

- LR:

PS	L(Left)	R(Right)
Q2	0	1
Q3	0	1

D3 = ~L*R*(Q2+Q3)

- R

PS	L(Left)	R(Right)
Q0	0	1
Q4	0	1
Q5	0	1

D4 = ~L*R*(Q0+Q4+Q5)

- R_B:

PS	L(Left)	R(Right)
Q4	1	1

Q5	1	1
Q6	1	1

D5 = L*R*(Q4+Q5+Q6)

- RL:

PS	L(Left)	R(Right)
Q5	1	0
Q6	1	0

D6 = L*~R*(Q5+Q6)

Outputs:

- Reset = IDLE
- CountUp = LR*~Right*~Left
- CountDw = RL*~Left*~Right
- L_LED = ~Left
- R_LED = ~Right
- On = Left+Right

- In my design I mixed Mealy and Moore, because I added more outputs after designing my model.
- With all of these boolean expressions we can combine them into one module to create a State Machine that acts as the brain of the game.

- 8-bit Counter(Converted from 16-bit counter from previous labs):
  In our previous lab write-up we examined how the U/D 16-bit counter was made and how it works. Since we only require 8 bits maximum in this lab I added a reset to the counter and set it to reset when the most significant bit is high(This function is only used for the timer counter). Finally for the Up/Dw Counter I added an output that detects when the values have gone negative or not.

- Two Complement Converter:
  Since we will be using negative numbers in this design we need to account for two's complement to prevent the counter from going to FF when one is subtracted from 00. This can be done by taking the NOT values of the counter and adding one to the value. I created this module by just creating an 8-bit adder(From lab2) and adding one to it.

- 2to1x8 MUX:

In order to determine whether I should display positive values or negative values I used a two to one 8bit mux that will select the right values depending on whether the values are negative or not.

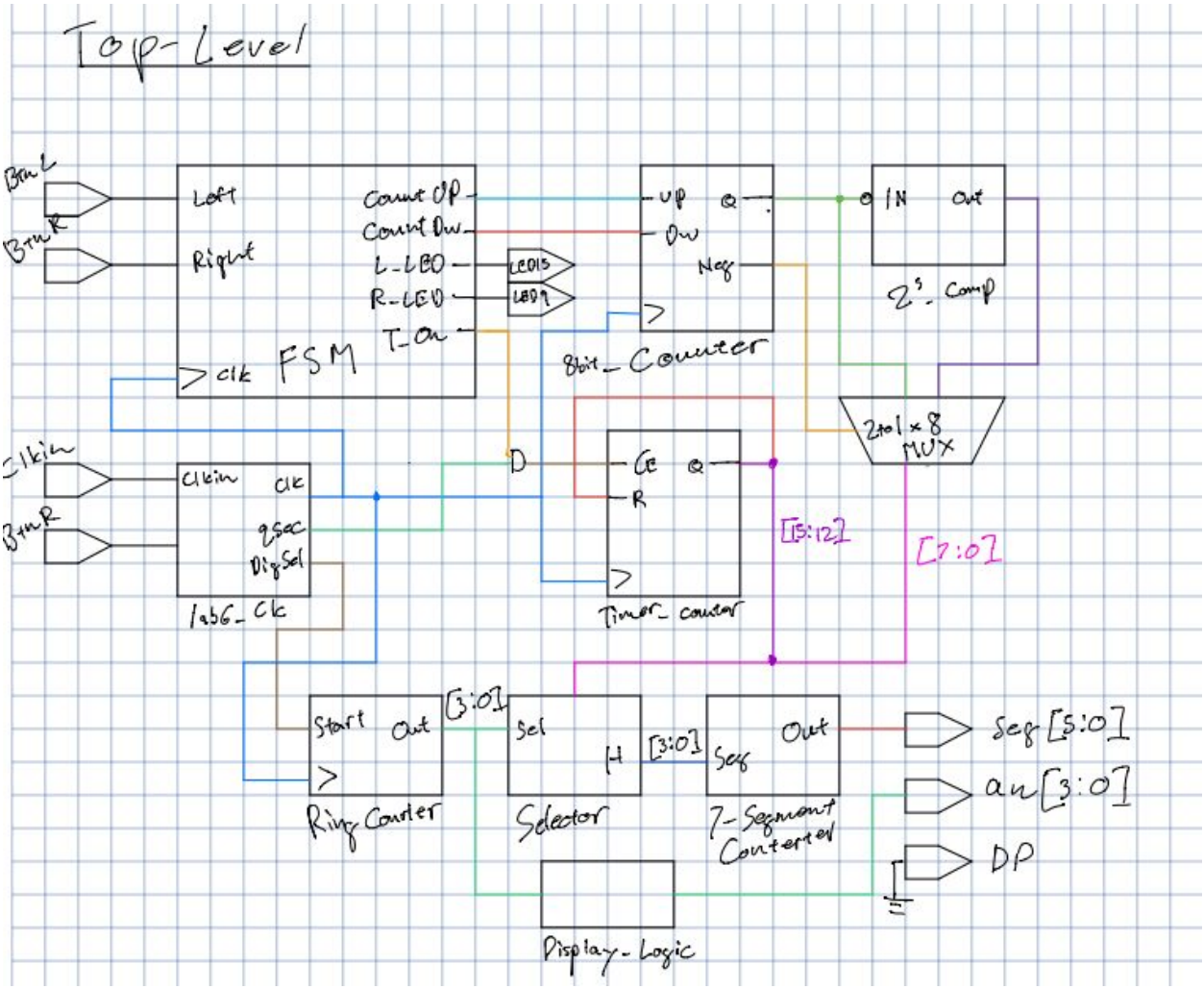
- Ring Counter:
  The ring counter is used as encode to create a one-hot/single active in 4-bit bus. This is used to set values for the selector. The module is created using a start(CE) and the clock(clk) which iterates through 4 flip flops as one state. In order to do this we connect the 4 flip flops in a complete loop and initialize one of the flip flops. This way we would get the outputs 0001, 0010, 0100, 1000, repeat.

- Selector:
  The selector is used to select a segment of it's inputs and output it with it's respective weight. This module is made with just boolean expressions with the inputs being the ring counter and the 16-bit counter and the output being the input to a seven_segment converter. As we review above the ring counter will feed a set of 4-bits with one active bit therefore giving it a state value for a certain part of the 16-bit you want to pass through. In this case, we want to pass through every 4-bits of the counter starting from 0. In pseudo code, it looks like this:
  - H is N[15:12] when sel=(1000)
  - H is N[11:8] when sel=(0100)
  - H is N[7:4] when sel=(0010)
  - H is N[3:0] when sel=(0001)

  When we put it in verilog it looks like this:
  - H = (N[15:12] & {4{(sel[3] & ~sel[2] & ~sel[1] & ~sel[0])}})|
    (N[11:8] & {4{(~sel[3] & sel[2] & ~sel[1] & ~sel[0])}})|
    (N[7:4] & {4{(~sel[3] & ~sel[2] & sel[1] & ~sel[0])}})|
    (N[3:0] & {4{(~sel[3] & ~sel[2] & ~sel[1] & sel[0])}});

- Seven Segment Converter:
  Finally we have the Seven Segment Converter. This module has been reviewed in previous lab reports and it works exactly the same. It takes in 4 inputs and uses boolean expressions to output the proper values to its respective segments. Unlike previous labs, we also needed to account for a negative sign. In order to do this, I made a case for when the ring counter for the specific display and negative are TRUE, every segment except for G(6) will be off/1.

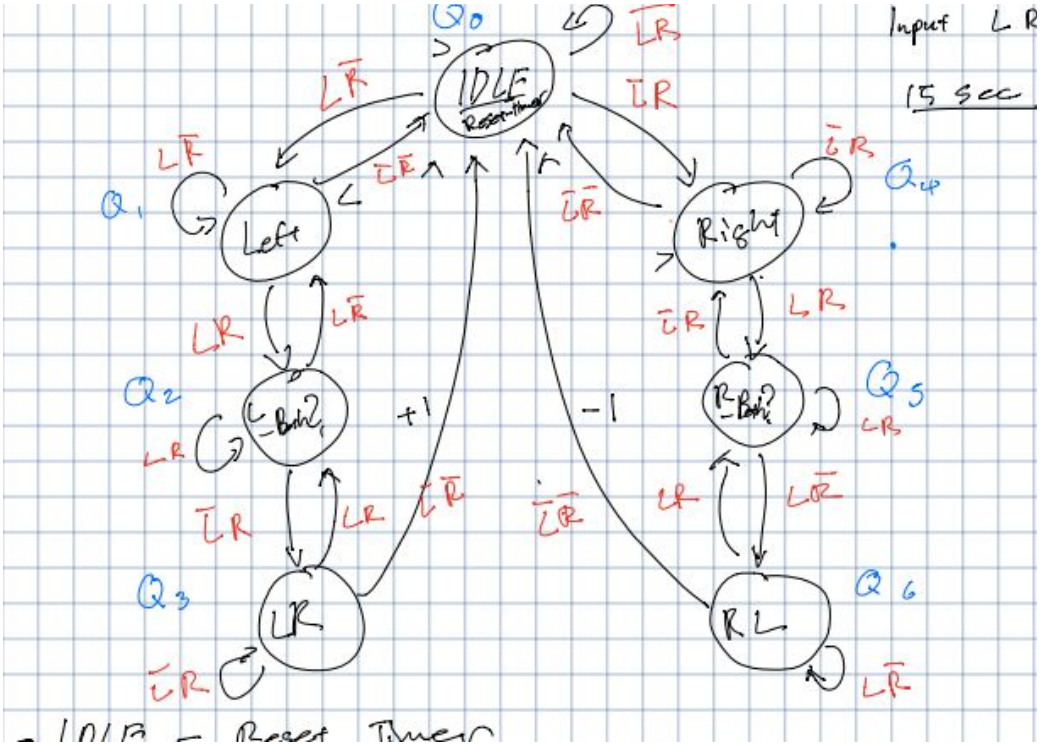
- Top Level:



- In this diagram we see all the modules that were used in this lab. I have compacted many of the logic for the wires to fit everything into the page.

- **Testing & Simulation:**
  In this lab the logic was mostly straightforward and simple. Unfortunately due to the many modules that were involved in this lab unlike the others, there was a higher chance for errors, therefore leading to more time searching for the cause of these errors. I bumped into a timing loop early on in my design and could not run any tests until I located the error. This process took a very long time, but after I found the issue I was able to visually test the sequential circuit using a visual inspection of the basys3 board.

- **Results:**
  1. State Diagram (Equations Explained in Module)



- IDLE: Is a state when no inputs are triggered
- Left: Is a state when the left trigger is triggered
- Right: Is a state when the right trigger is triggered
- L_Both: Is a state when both triggers are triggered after Left was initially triggered
- R_Both: Is a state when both triggers are triggered after Right was initially triggered
- LR: Is a state when Left is no longer triggered after state L_Both.
- RL: Is a state when Right is no longer triggered after state R_Both.

- Note: Since I did the opposite counts for the direction Turkey is going when designing the circuit the L and R are swapped on the Top_Level.

- **Conclusion:**
  This lab I got to experience designing my sequential circuit from scratch. Since I took CE12 my freshman year, I had forgotten how to do two's complement, but after getting help I was smoothly sailing through this lab. I'm glad my design is working after making a few bug fixes. If I had more time I would definitely go over the state diagram and simplify it, as there were many states that had similar functionality.