Computer Architecture 2015 Exam

Griffin Tschurwald

November 2015

1 Specification Document

This specification document is for a bike light controller that has 4 modes, and is controlled by a button. This controller powers a single LED. Here we list the inputs, outputs, a finite state machine for the system, and a timing diagram showing the different output states.

1.1 Inputs and Outputs

Input: Button press signal. This will be high when the button is depressed, and low when the button is released. This input should be conditioned to watch for the hardware "bouncing".

Output: LED signal. This will be high to turn the LED on, and low to turn the LED off. Also, a voltage of half of high will be output to activate the Dim light mode.

1.2 Finite State Machine and Modes

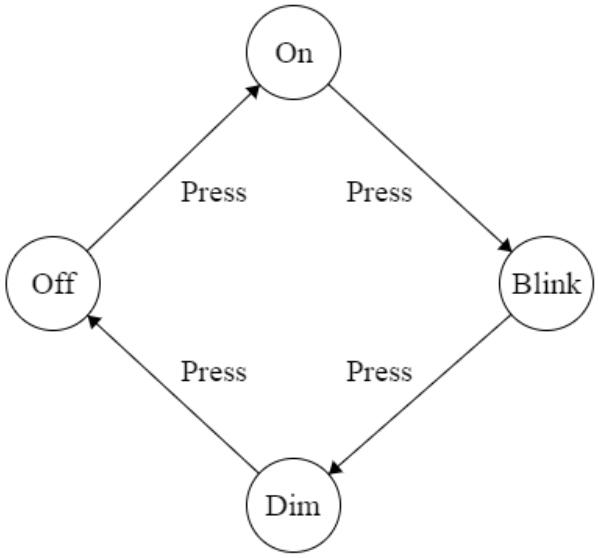
Below is a diagram showing the finite state machine for the system. Here we will also list the bike light modes corresponding to the states of the FSM. Each transition is activated by a single button press. Below we will list the states and what they mean for the LED otuput.

Off: In this state, there should be no output to the LED. The LED output signal will always be low.

On: In this state, the LED should be constantly on. This means the output signal should stay high.

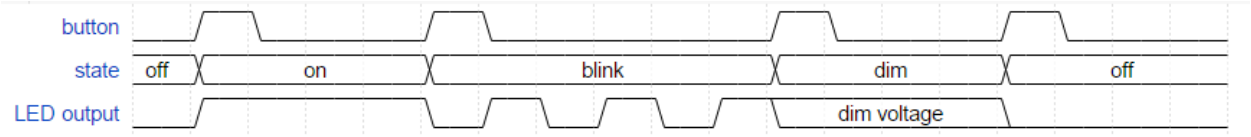
Blink: In this state, the LED output should alternate between low and high.

Dim: In this state, the LED output will always be on, but will be at a voltage half that of high signal.



1.3 Timing Diagram

This timing diagram will show how the LED output will change based on the button input, based on what state the FSM is in. In this diagram, the controller starts at the off position on the FSM. In the diagram, the output changes instantaneously with the input, although a small delay will likely occur in the finished product.

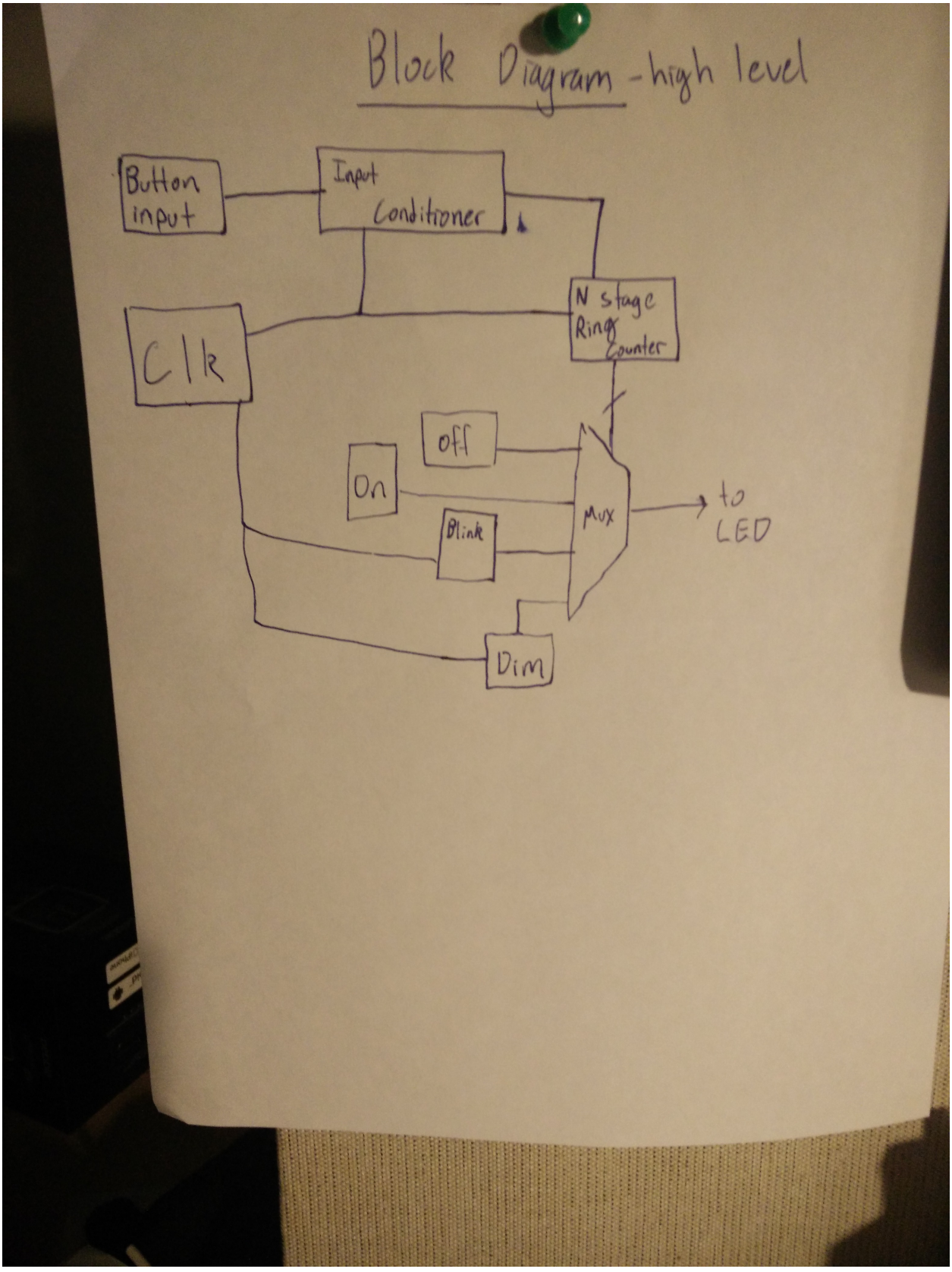


1.4 Measurement Specifications

The bike light must adhere to specific rates of output based on the mode. The dim mode voltage should be half that of high voltage. If PWM is used to generate this signal, then use a 50% duty cycle. The blink mode should blink on and off at a rate of roughly 4hz.

2 Block Diagram

The block diagram shows a high level overview of what the controller will look like. Below is a list with an overview of what each component will do, and whether I will draw a detailed schematic for it.



Button Input: This is just the button signal. It is high when the button is depressed, and low when the

button is not pressed. It outputs its signal on a single wire.

Clk: This is the clock we were given which has a frequency of 32,768 hz. It outputs its signal on a single wire.

Input Conditioner: This component needs to condition the input to prevent debouncing from registering as many button presses. It will also sync the conditioned input signal to the internal clock. It takes the clock signal and button as input, and outputs a one bit signal that is the conditioned output.

N stage Ring counter: This is the component that will represent the states of our FSM. It should be incremented only once when the conditioned signal goes high, so I put a D flip flop with some logic so that it will go high only on the rising edge of the conditioned input. This takes a single bit signal as input, and outputs a 4 bit wire with a one hot signal.

Off signal: This is just a wire set to LOW. It ouputs low on a single bit wire at all times.

On signal: This is just a wire set to High. It outputs high on a single bit wire at all times.

Output Mux: This is the multiplexer that chooses which signal to output. It does this based on the signal it gets from the n stage ring counter. It take that 4 bit signal as input, and outputs the final signal to the LED.

Blink signal: This is generated by a series of frequency dividers. I need a hz right around 4, and the speed of the clock divided by 2^13 is 4hz. Each frequency divider doubles the period, so I needed 13. It takes the clock signal as input, and outputs a blink signal to the mux.

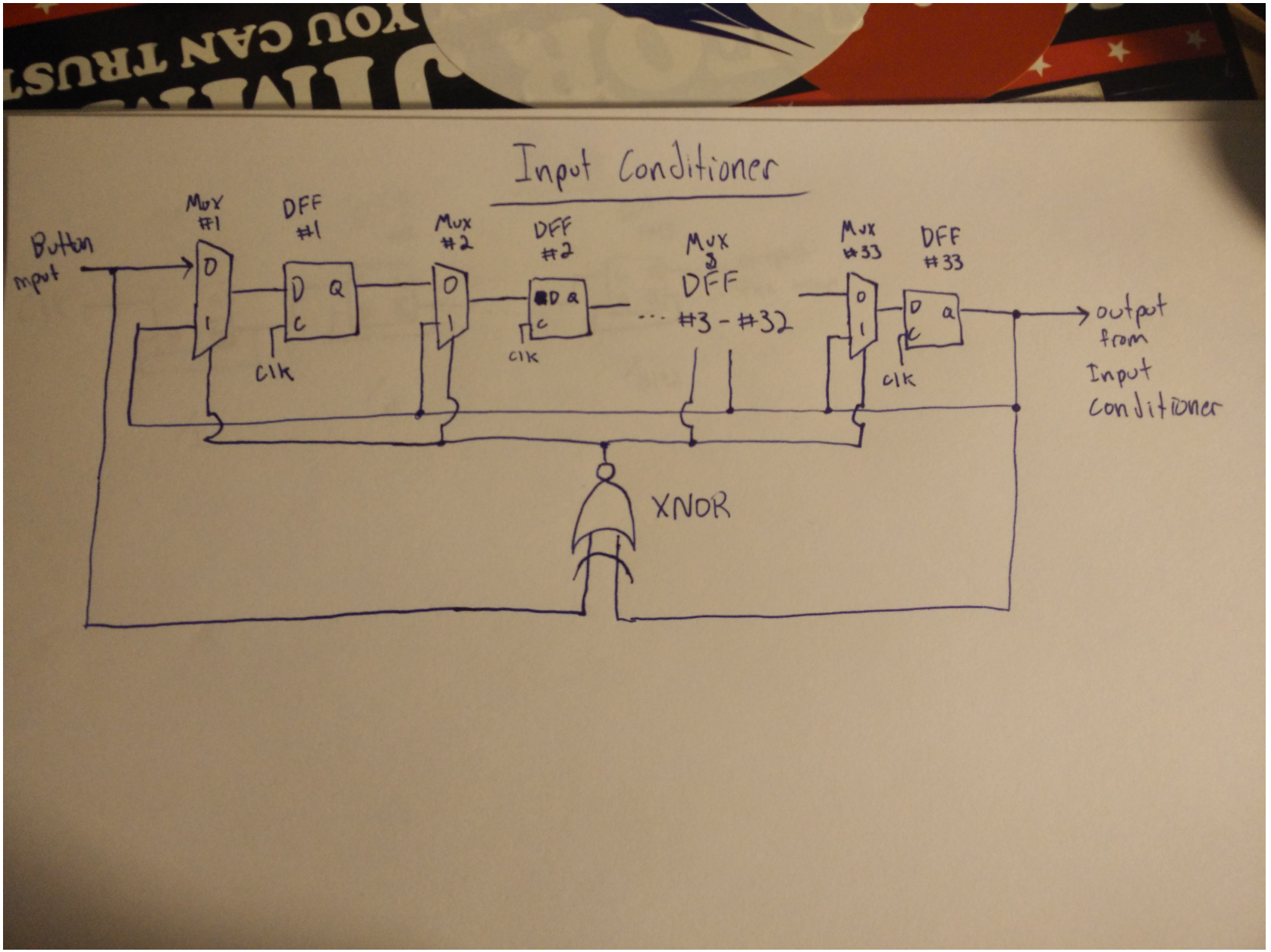
Dim signal: This is just a pwm signal generated by the clock. Its input and output are the clock signal.

3 Schematic

Below I will show diagrams of all schematics, as well as a short description.

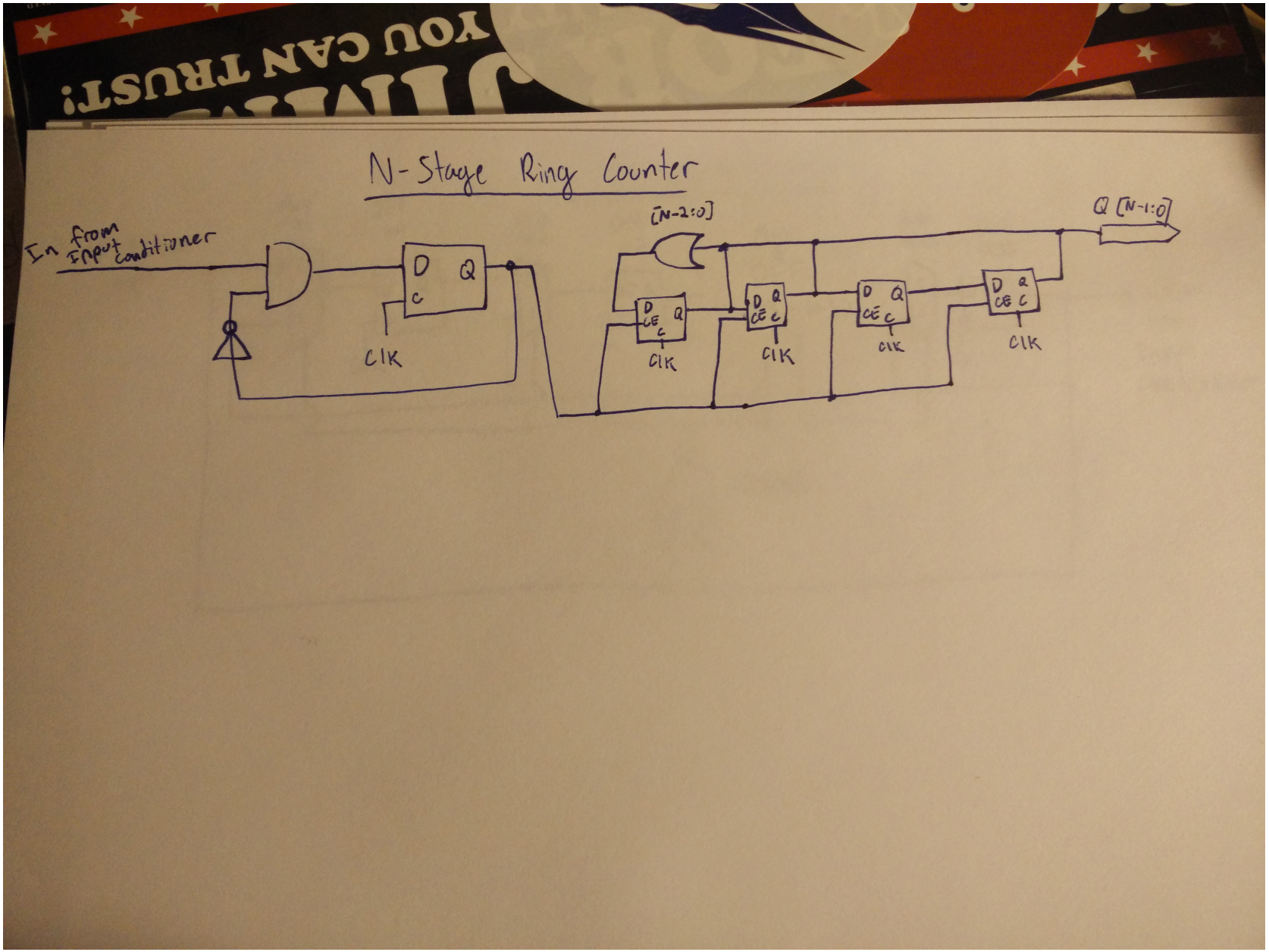
3.1 Input Conditioner

Given that the button noise decays within one millisecond, I decided to wait for one millisecond of steady signal from the button before propagating the conditioned signal. I calculated that one clock cycle corresponded to .031 ms, so I needed at least 33 clock cycles of waiting time before changing my output conditioned signal. Additionally, I needed to reset my counter whenever the button signal matched the output for at least one clock rising edge. This meant that tiny aberrations in the middle of a clock cycle would not cause the counter to reset. To build this counter, I linked 33 D flip-flops. They all start low, and then any signal takes 1 clock cycle to move one flip flop down the line. This meant I had the delay I wanted, but I still needed to have a reset whenever the signal changed. To reset every flip flop, I put a mux in front of all of them. When the input to this mux is low, the signal from the previous flip flop is the input to the flip flop. However, in the case when the button input matches the output from the last flip flop, the counter needs to be reset. I used a XNOR gate to compare these 2 signals, then have that feed into every mux. When this signal goes high and a reset is desired, all flip flops are reset the value of the output. In my diagram I didn't draw many of the flip flops in the middle, as they're the same as the ones on the edges and I felt it was redundant to draw all of them.


Input Conditioner
Button input
Mux #1
DFF #1
Mux #2
DFF #2
Mux #3 - DFF #3 - #32
Mux #33
DFF #33
D
Q
C
clk
0
1
XNOR
output from Input Conditioner

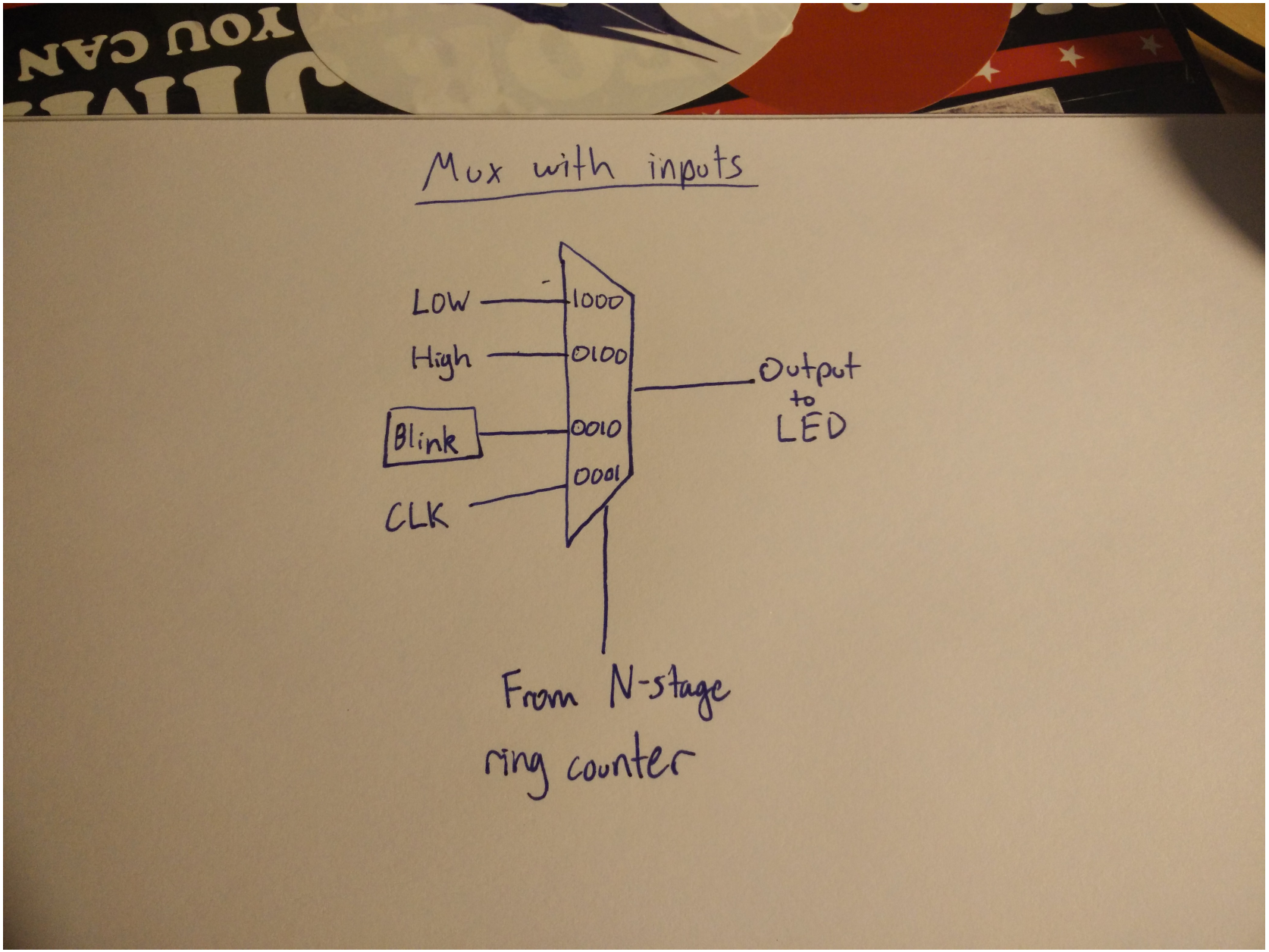
3.2 N stage Ring Counter

This is the component that tracks the finite state machine states. When enable is active, on the first clock cycle the "hot" bit moves one down the line of D flip flops, starting at the first one. Because the conditioned input will stay high as long as the button is pressed, I added some components in front that will only go high for one clock cycle on the rising edge of the conditioned input to solve this problem.


N-Stage Ring Counter
In from Input Conditioner
[N-2:0]
Q [N-1:0]
D
Q
C
CE
CLK

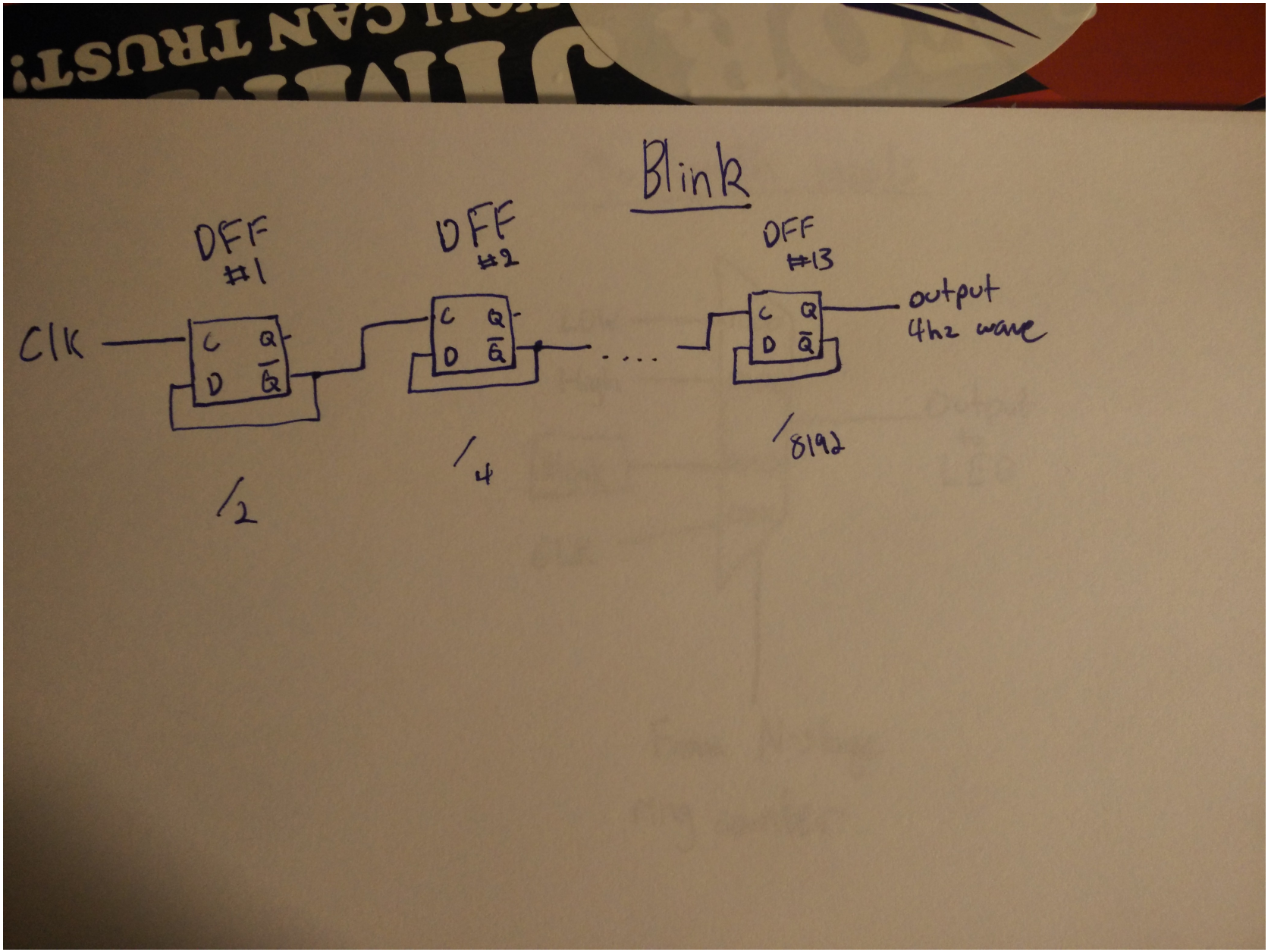
3.3 Multiplexer

This multiplexer will choose the correct input signal based on the one-hot value passed to it by the ring counter. This will require less gates than a standard 4 input mux becuase the input is a one hot signal, so the bits of that can be passed directly to and gates with the 4 inputs.


Mux with inputs
LOW
High
Blink
CLK
1000
0100
0010
0001
Output
to
LED
From N-stage
ring counter

3.4 Blink generator

This is generated by a series of frequency dividers. I need a hz right around 4, and the speed of the clock divided by 2^{13} is 4hz. Each frequency divider doubles the period, so I needed 13. I utilized D flip flops as my frequency dividers.


Blink
DFF #1
DFF #2
DFF #13
Clk
C Q
D Q̄
output 4hz wave
1/2
1/4
1/8192

4 Cost Estimation

Below is a list of components with their cost.

Clk: 2

Input Conditioner: 33 DFF * 13 + 33 mux * 7 + 1 XNOR * 8 = 668

N stage Ring counter: 21*4 - 1 = 83

Off signal: 0

On signal: 0

Output Mux: 9

Blink signal: 13*13 = 169

Dim signal: 0

Total: 931