ECE 385

Spring 2023

Lab 1

Introductory TTL Experiment

Gustavo Fonseca
TA: Hongshuo Zhang

1. **Introduction**

   The purpose of the lab was to demonstrate the importance of considering static hazard glitches when designing logic circuits. Static hazard glitches can occur when aspects of logic circuits have distortional gate delays. In this lab, I was given a simple circuit that had this glitch. My goal in this lab was to discover a way to avoid this glitch by adjusting the circuit. This lab is useful for processor architecture as it explores various glitches that could occur in circuit design.

2. **Description of Circuit**

   In this lab the original circuit was given to me. The original circuit followed the following Karnaugh Map:

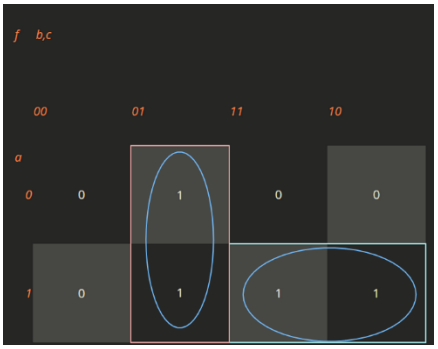

Figure 1: Original K-Map

   This results in the equation: B'C + AB

   Since I want to minimize the amount of IC chips in use, I transformed this equation to one with only NAND gates using De Morgan's Law. This results in: ((AB)'(B'C)')'. However, this design causes the aforementioned static hazard glitch. The difference in gate delays between B' and the rest of the data path causes the glitch. This glitch occurs when switching B for 0 to 1 or vice versa. In order to fix this, we need to add redundancy to the circuit. If I update the K-Map to reflect this change:

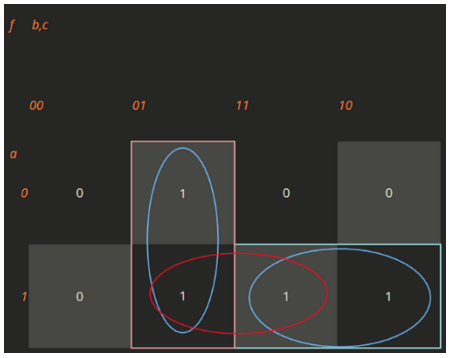

Figure 2: New K-Map with redundancy

   We can see that the solution to this glitch is to cover all adjacent min-terms in the K-map. This then results in: ((AB)'(B'C)')' + AC

   Using De Morgan's Law to make this equation with NANDs we get:

(Z’(AC)’)’, where Z = ((AB)’(B’C)’)’

With this new design, there is no longer the issue of the static hazard glitch.

3. **Component Layout**
    a. **Naïve Design**

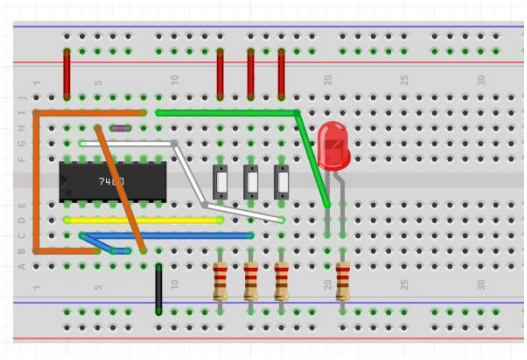

Figure 3: Breadboard Layout for Naive Design

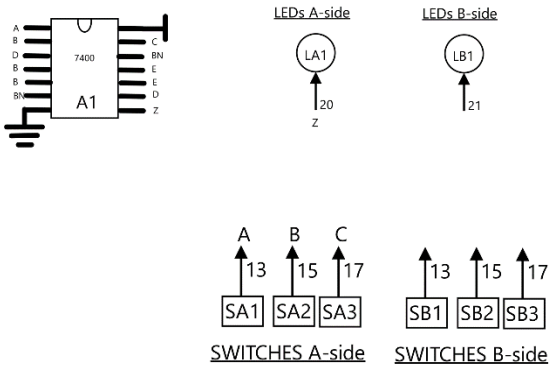

Figure 4: Component Layout Naive Design

b. Redundant Design

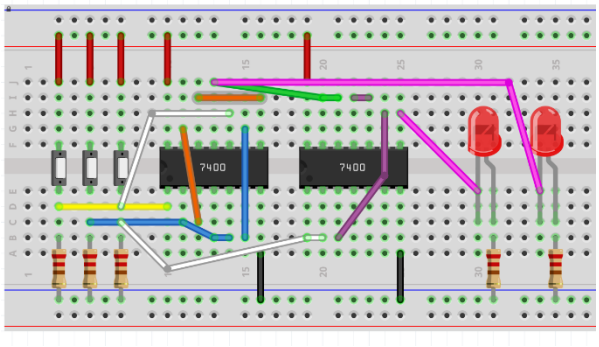

Figure 5: Breadboard for Redundant Design

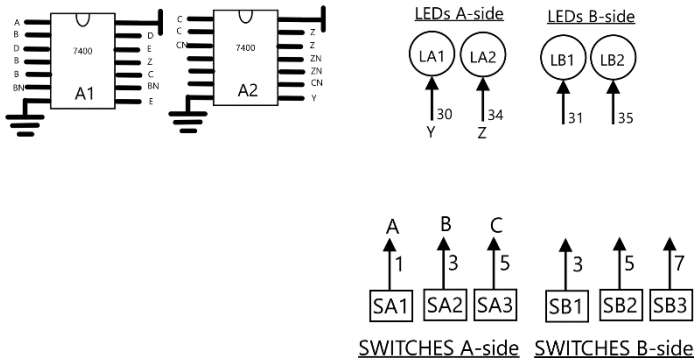

Figure 6: Component Layout for Redundant Design

4. Circuit Diagrams

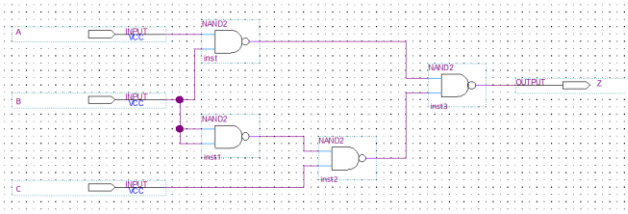

Figure 7: Circuit Diagram for Naive Design, Z = ((AB)'(B'C)')'

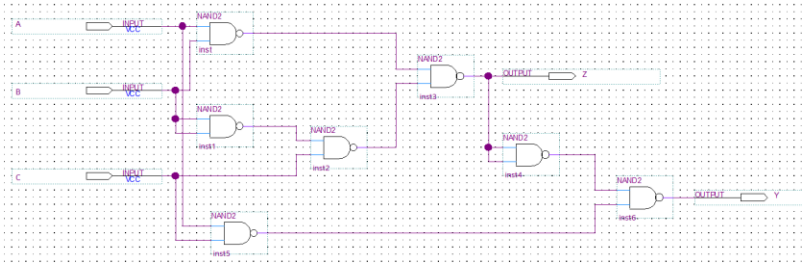

Figure 8: Circuit Diagram for Redundant Design, Y = (Z'(AC)')'

5. **Documentation**
    a. **Truth Tables**

The truth table for the original circuit and the redundant design are identical:

Truth Table

	A	B	C	Y
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Figure 9: Truth Table for both designs

b. **Oscilloscope Printouts**

Here, we can see the Static-1 Hazard glitch as the output goes to a logical low momentarily before going to back to the correct logical one.

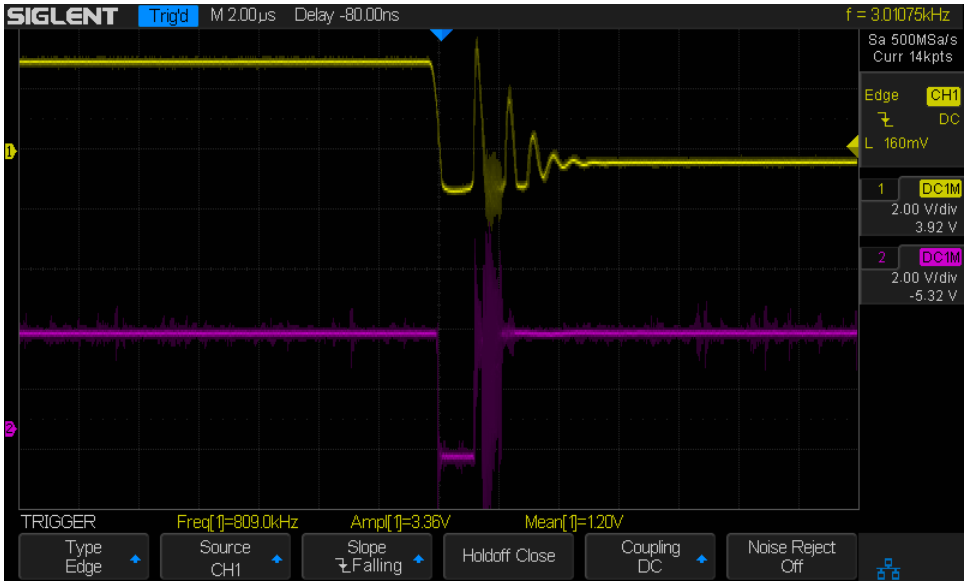

Figure 10: Original Reading

In Figure 11, we can see that, with the redundancy added to the system, the static-1 hazard glitch no longer occurs as the output stays at a logical one through the switch of the input.

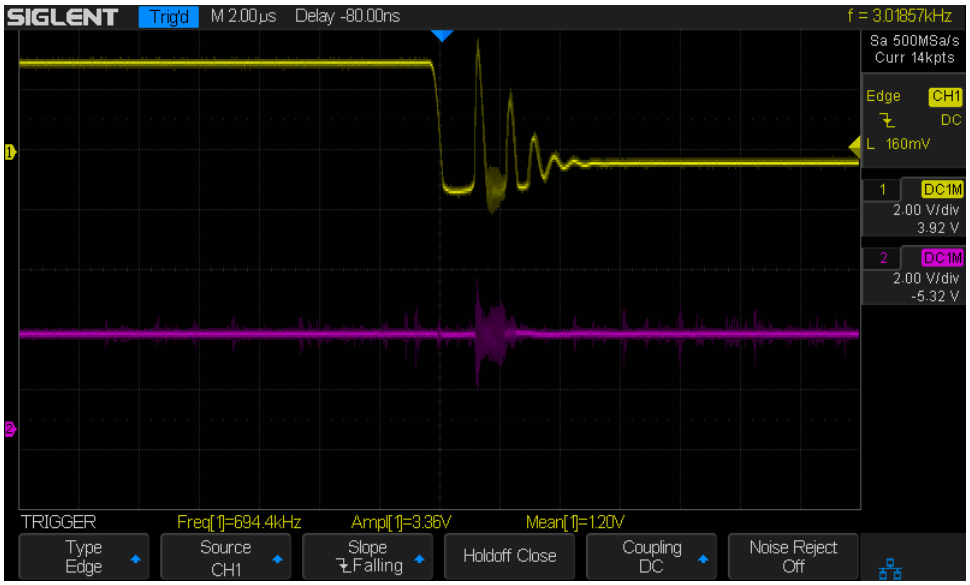

Figure 11: New Reading with Redundancy

6. **Pre-Lab Questions**
    a. Not all groups may observe static hazards (why?)
    The reason that not all groups will get the glitch is because each NAND IC has different gate delays, thus the glitch may just appear as noise when the IC has a small gate delay.
    b. Why does the hazard appear when you add more inverters?
    The reason that static hazard may only appear when you add more inverters, or a capacitor is because this increases the delay of the signal. With more delay of a signal change, the larger the glitch will last. With only one inverter, the delay is very small; thus, the glitch could just appear as noise within oscilloscope reading.
7. **Lab Questions**
    a. Complete a truth table of the output. Does it respond like the circuit of part A?
    Yes, the new redundant circuit acts the same as the circuit of part A in terms of logic without taking gate delay into account.
    b. Consider the following question and explain: for the circuit of part A of the pre-lab, at which edge (rising/falling) of the input B are we more likely to observe a glitch at the output?
    We would be more likely to observe the glitch at the falling edge of input B. When B goes from low to high, the output Z has the same value regardless of which input comes first. Therefore, the different gate delays become irrelevant in this case.
8. **Post-Lab Questions**

a. **Timing Diagram**
Considering the maximum gate delay of 20ns, the timing diagram becomes:

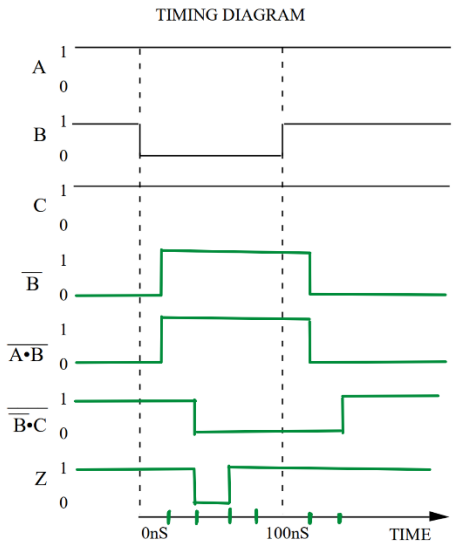

Figure 12: Timing Diagram for original design

b. How long does it take the output Z to stabilize on the falling edge of B (in ns)?
It takes a maximum of 60 nanoseconds for Z to stabilize on the falling edge of B as there are 3 gate delays of 20 nanoseconds between B and Z.

c. How long does it take on the rising edge (in ns)?
It takes 0 nanoseconds for Z to stabilize on the rising edge of B as Z does not change on B's rising edge.

d. Are there any potential glitches in the output, Z?
Yes, there is a static-1 glitch in this circuit. This occurs when B goes from high to low, as there is an uneven gate delay between 2 inputs into Z: (AB)' with 1 gate delay, and (B'C)' with 2 gate delays. This causes the output Z, which is the NAND of these 2 signals, to become 0 before the value of (B'C)' to be updated, going back up to 1. The 0 result is not supposed to happen. If there was no gate delay, then Z would always stay at 1.

e. Explain how and why the debouncer circuit given in General Guide Figure 17 (GG.32) works. Specifically, what makes it behave like a switch and how the ill effect of mechanical contact bounces is eliminated?
The debouncer circuit works because, when the SPDT switch is not connected to anything both A and B are connected to logical high. This causes the output Q and Q' to hold their value. When the switch connects to A, A is connected to ground, making Q a logic high. When the switch connects to B, Q goes to a logic low. This method prevents the effect of mechanical contact bounces as, instead of a normal switch that has 2 states, high or low, this has 3 states, high, low, or hold. Thus, when the switch bounces when connecting to A or B, instead of bouncing

from high to low, the switch bounces from high to hold or low to hold. As a result, the output does not change due to the result of the bounce.

9. **General Guide Questions**
    a. What is the advantage of a larger noise immunity?
    The advantage of a larger noise immunity is that we do not have to worry about having noise within system. If the noise immunity is too small than a small amount of noise from elements within the system may result in a signal being incorrectly flipped by noise.
    b. Why is the last inverter observed rather than simply the first?
    The reason we check the last inverter is because noise is added after going through each inverter. Checking after the first inverter would only allow us to see the noise applied by one inverter, while checking the last would allow us to see the total noise of the system.
    c. Given a graph of output voltage (VOUT) vs. input voltage (VIN) for an inverter, how would you calculate the noise immunity for the inverter?
    In order to calculate the noise immunity for an inverter we have to look at nominal values and ranges for the inverter. To find the noise immunity for a logical low we start at the nominal value for logic low of the input, 0.35V, then we increase the input voltage until the inverter's output voltage leaves the nominal range for logic high, in this case 1.15V. We can then subtract the nominal value for logic low from this value to get the noise immunity for a logic low to be 0.8V. Using the same method for calculating the noise immunity for logic high, we arrive at the result of 2.15V.
    d. If we have two or more LEDs to monitor several signals, why is it bad practice to share resistors?
    It is bad practice to share resistors for multiple LEDs because different LEDs have different voltage and current ratings from one another. Thus, the LEDs may not function properly, either being dimmer or not turning on at all.
10. **Conclusion**

In this lab, I have learned the importance of considering glitches when deciding on how to design digital circuits. Static hazard glitches are very important to consider when implementing a circuit that has multiple gate delays.

I also learned how to deal with mechanical contact bounce. This is important to consider because several circuits have mechanical switches. These lessons will be very important moving forward in this course and in my career.

Regarding what worked and what did not work, there is not much to consider. This lab is mainly used to test the importance of using redundancies to ensure that static hazard glitches do not occur.

Regarding how to improve my circuit, we could rewrite the logic equation to make the circuit have less gate delay. Going back to the redundant K-map, we could use the expression: B'C + AB + AC. If we used De Morgan's Law on this equation we get:

((B’C)’(AB)’(AC)’)’. This would eliminate some gate delays, but we would still have to use 2 IC chips as we now need a triple NAND gate chip.