ECE 385 Lab Report 1

Spring 2023
Eshaan Tibrewala
Lab Section: SL Friday 4:00PM
TA: Shitao Liu

Purpose of Circuit

The purpose of this circuit is to eliminate potential static hazards at the output of 2-to-1 multiplexers (MUX). Static hazards are an example of propagation delays, and are glitches in Transistor-Transistor Logic (TTL) gates. These glitches are very temporary, as the signal returns to its steady state, however they produce momentarily produce incorrect output values between the time the inputs are changing. In this lab, we are looking at static-1 hazards in 2-to-1 MUX specifically, where the output is originally at high (1), but changes to low (0) temporarily before returning to high. A 2-to-1 MUX uses a selection bit to toggle between two potential input bits. This simple function is very important to many complex circuits and systems, so even a minute glitch can cause significant issues and destroy functionality. Since this is also the first lab of the class, this circuit will help to better understand the equipment, processes, and software that we will use throughout the class.

Description of Circuit and Lab Process

The circuit is supposed to represent a functioning 2-to-1 MUX. A 2-to-1 MUX has three 1-bit signal inputs, and one 1-bit output. Of the three inputs, one of them serves as the selector bit and toggles the output between the other two signals. In *FIGURE 1* below, ***B*** serves as the selector bit, so when ***B*** is low the output ***Z*** reflects input signal ***C*** and when ***B*** is high output ***Z*** reflects input signal ***A***.

A	B	C	Z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

FIGURE 1: 2-1 MUX Truth Table

Using the truth table, we can make the following K-MAP in *FIGURE 2*.

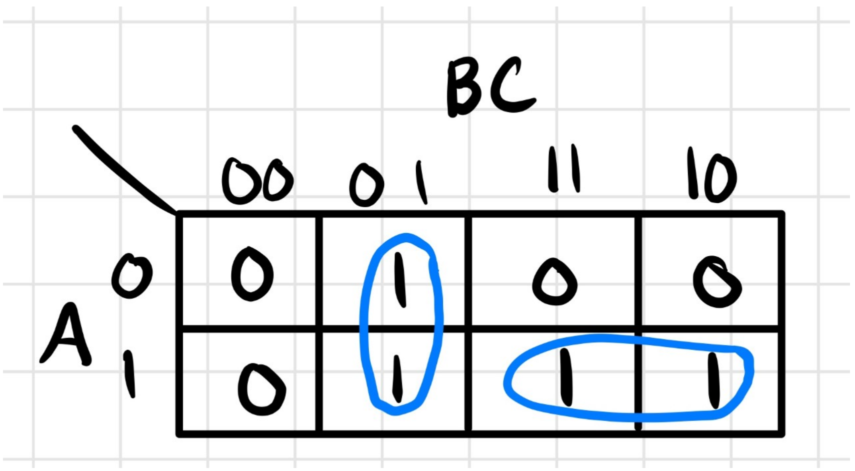


FIGURE 2: 2-to-1 MUX K-MAP

The following sum of products (SOP) expression can be derived from grouping all the min-terms shown in the K-MAP (circled in blue) from *FIGURE 2*. SOP expressions are in the form of AND-OR:

$$(1)\ Z = (\neg B\ \wedge\ C)\ \vee\ (B\ \wedge\ A)$$

$$(2)\ Z = \overline{\overline{(\neg B \wedge C)}\ \wedge\ \overline{(B\ \wedge\ A)}}$$

Equation (1) shows the minimum SOP expression for the K-MAP in *FIGURE 2*, however it cannot be implemented using only a quad-NAND gate unless De Morgan's Law is applied, as is the case with this lab. Equation (2) therefore shows the correct logic expression so that the circuit can be expressed using only NAND gates. *FIGURE 4* shows a simplified logic diagram for the circuit and was included for the purpose of understanding the functionality of a 2-to-1 MUX and to identify where the static-1 glitch occurs. The actual logic diagrams with the 7400 chips used to build the circuit can be found in the **Logic Diagrams** section.

While the SOP expression is technically correct, the oscilloscope clearly shows that there is a static-1 hazard somewhere in the circuit. The bottom pink/purple line momentarily drops to 0, before returning to 1, showing the exact time when the glitch occurred. Another thing to note here is that static-1 hazards are difficult to catch, so the output of the invertor was driven through a $1\ \mu F$ capacitor on the actual circuit so that the oscilloscope could pick up the glitch. The capacitor accomplishes this by extending the delay caused by the invertor. The oscilloscope result can be seen in *FIGURE 3.*

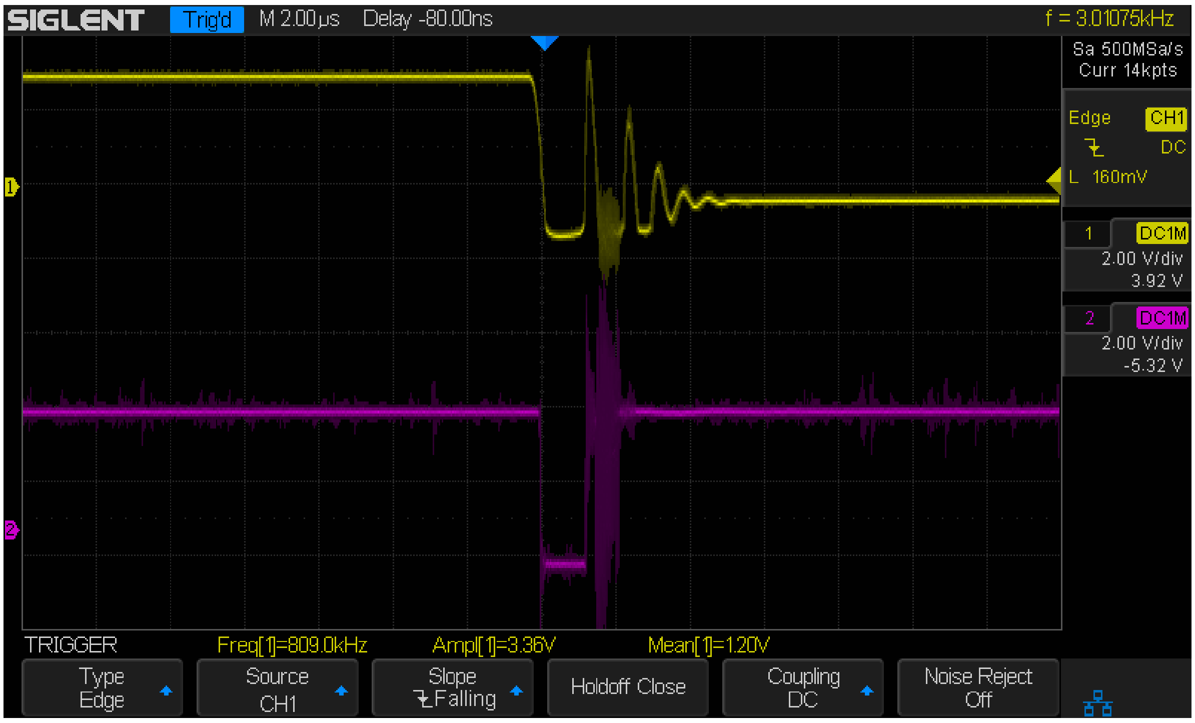


FIGURE 3: Oscilloscope Result for Initial SOP Expression

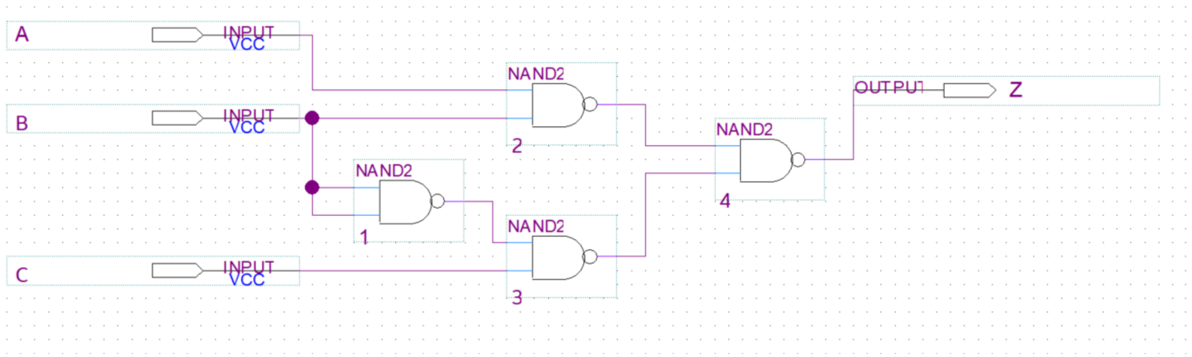


FIGURE 4: Logic Diagram for 2-to-1 MUX with NAND Gates

From a deeper analysis of the oscilloscope data, the static-1 glitch occurs exactly at the moment that the selector signal ***B*** changes its value from 1 to 0, changing the output to reflect signal ***C***. The logic diagram in *FIGURE 4* gives a more detailed analysis on the static-1 hazard and exactly where and when it occurs. In the current scenario, the input signals ***A*** and ***C*** were both set to 1. The selector bit ***B***, was also set to 1, resulting in the 2-to-1 MUX outputting the signal ***A.*** However, when ***B*** changed positions (the yellow line changes from high to low), the purple line momentarily drops to low, outputs 0, before returning to its steady state where it outputs signal ***C***, which in this case is also 1. This error in the circuit occurs because when ***B*** changes its signal, the 0 signal is driven through NAND gate 2 where it is then forwarded to the top input of NAND gate 4. However, in the bottom of the diagram, the 0 signal from the original switch has to go through NAND gate 1, before it is driven through NAND gate 3 which outputs a 1, and then finally catches up to the other signal at NAND gate 4. So as seen from *FIGURE 4*, there is a slight moment where the circuit outputs the incorrect signal because the switch from ***B*** was not propagated throughout the circuit at the same rate.

However fixing this static-1 hazard is fairly simple and can be accomplished by modifying the SOP expression so that it includes a redundant such that all adjacent min-terms on the K-MAP are included, as shown in *FIGURE 5*. The redundant term to be added is shown by the red grouping, which was not included in the original expression.

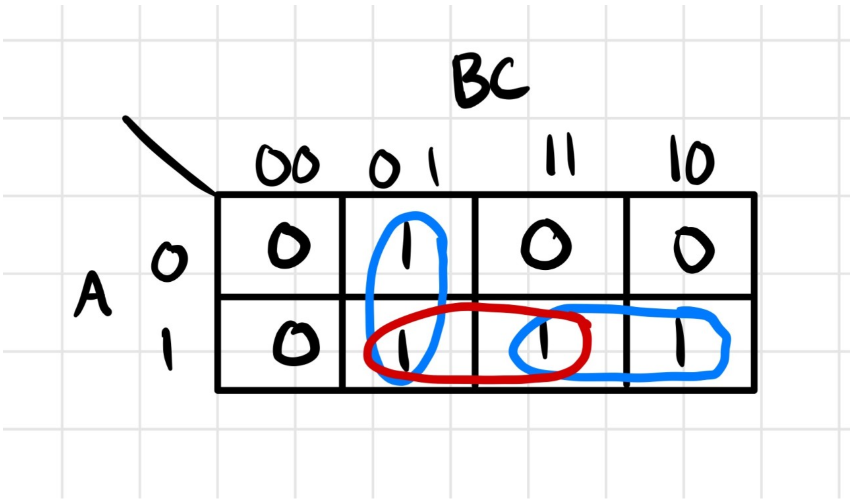


FIGURE 5: K-MAP with Redundant Min-Terms

The updated SOP expression derived from the K-MAP in *FIGURE 5* is:

$$(1)\ Z = BC + B'A + AC$$
$$(2)\ Z = (B\ \wedge\ C)\ \vee\ (\neg B\ \wedge\ A)\ \vee\ (A\ \wedge\ C)$$

After simulating the new expression on the oscilloscope, *FIGURE 6* clearly shows that now when ***B*** changes its position, there is no static-1 hazard in the circuit since the delay throughout the circuit is now balanced, so when ***B*** switches, the signal change will propagate throughout the circuit without major delay, ensuring that the output remains accurate at all times. While the static-1 glitch has been resolved, it is important to note that these is still a slight moment where the output is not consistent, but that is due to the chip components having slight delays by default and therefore any change in the selector input cannot be guaranteed to be instantaneous and flawless.

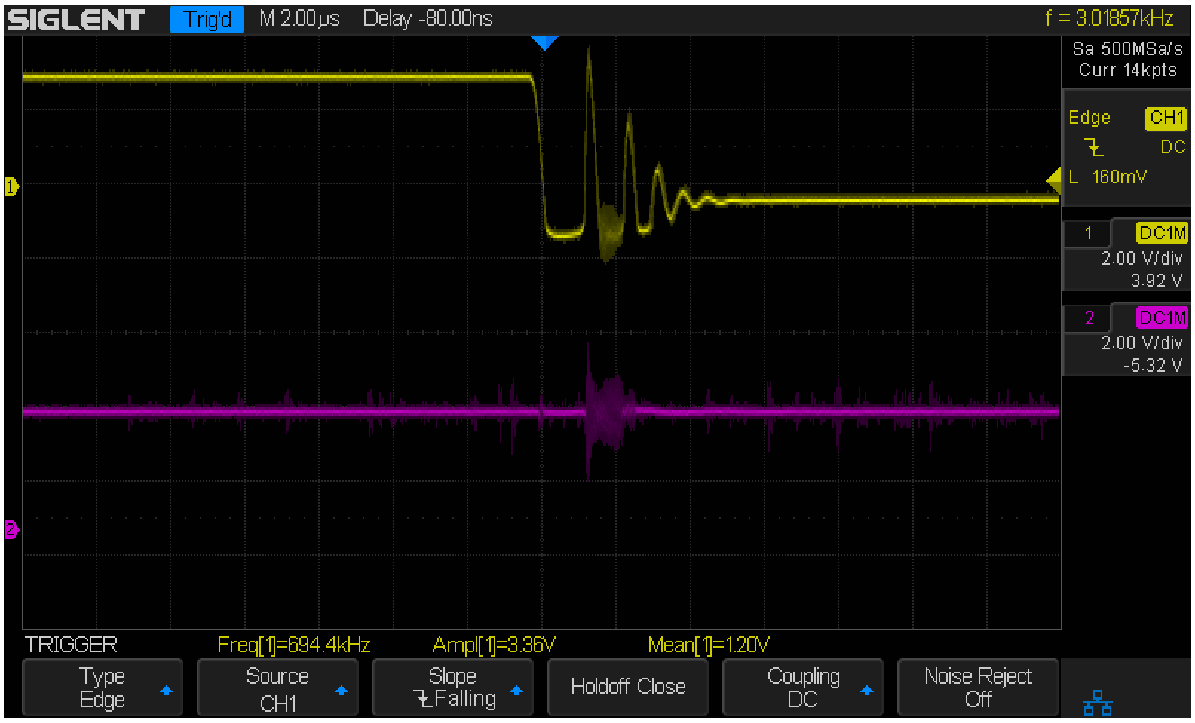


FIGURE 6: Oscilloscope Data for SOP Expression with Redundant AC

Logic Diagrams

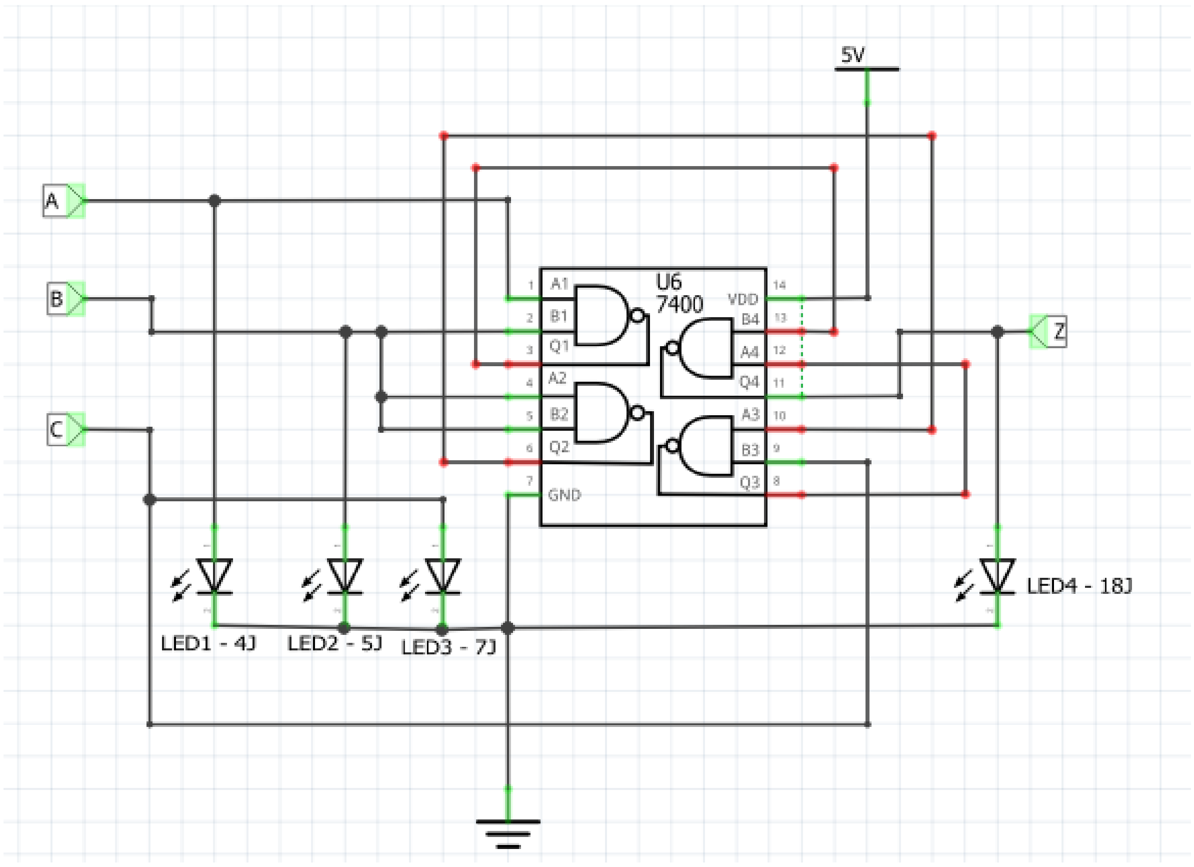


FIGURE 7: Logic Diagram for Lab Circuit with Static-1

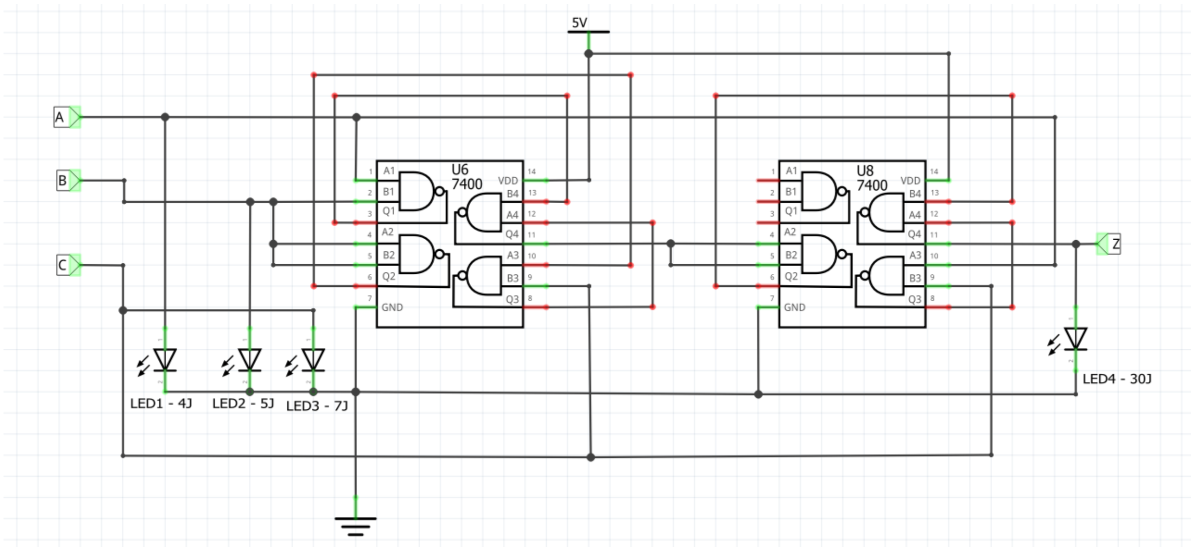


FIGURE 8: Logic Diagram for Lab Circuit with Solved Static-1

Component Layout

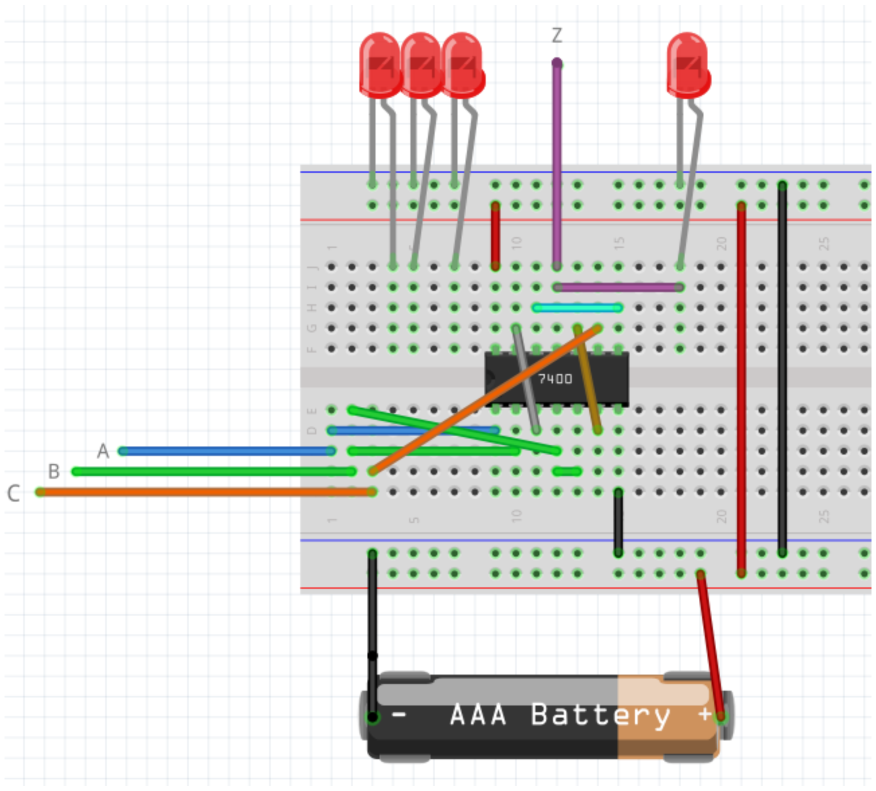


FIGURE 9: Component Layout for Lab Circuit with Static-1

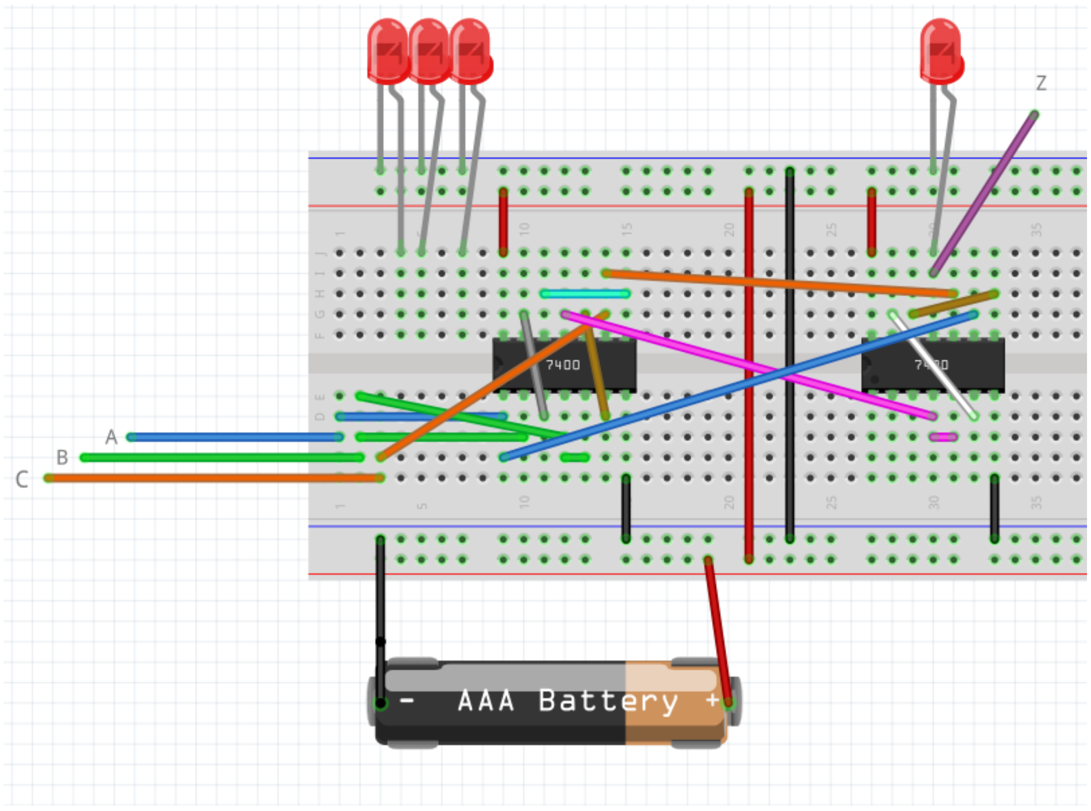


FIGURE 10: Component Layout for Lab Circuit with Solved Static-1

Pre-Lab Questions

1. **Not all groups may observe static hazards. Why?**

This could occur because while each chip essentially functions the same, they can all have slightly different delays or other characteristics since they are not exactly the same. This can cause irregularities in performance across different chips, leading to slightly different outputs, with one possibility being a lack of static hazard. More specifically, since static-1 glitches occur at the delay at the invertor in a 2-to-1 MUX, one chip could be manufactured such that the gate delays for the other gates are so minimal that the incorrect signal never propagates to the output.

2. **If you do not observe a static hazard, chain an odd number of inverters together in place of the single inverter from Figure 16 or add a small 1.2 capacitor to the output of the inverter until you observe a glitch. Why does the hazard appear when you do this?**

   If a hazard was not observed originally, it is observable either after adding more inverters because the delay from each gate is added up as the signal propagates through the gate, and if enough inverters are added the summed up delay would be significant enough to be read and analyzed by an oscilloscope. Adding a small capacitor also makes the static-1 glitch more noticeable because similar to the gate delays, it essentially slows down the signal as it is traveling through the circuit because capacitors require additional time to completely discharge to zero, allowing the signal to read by an oscilloscope.

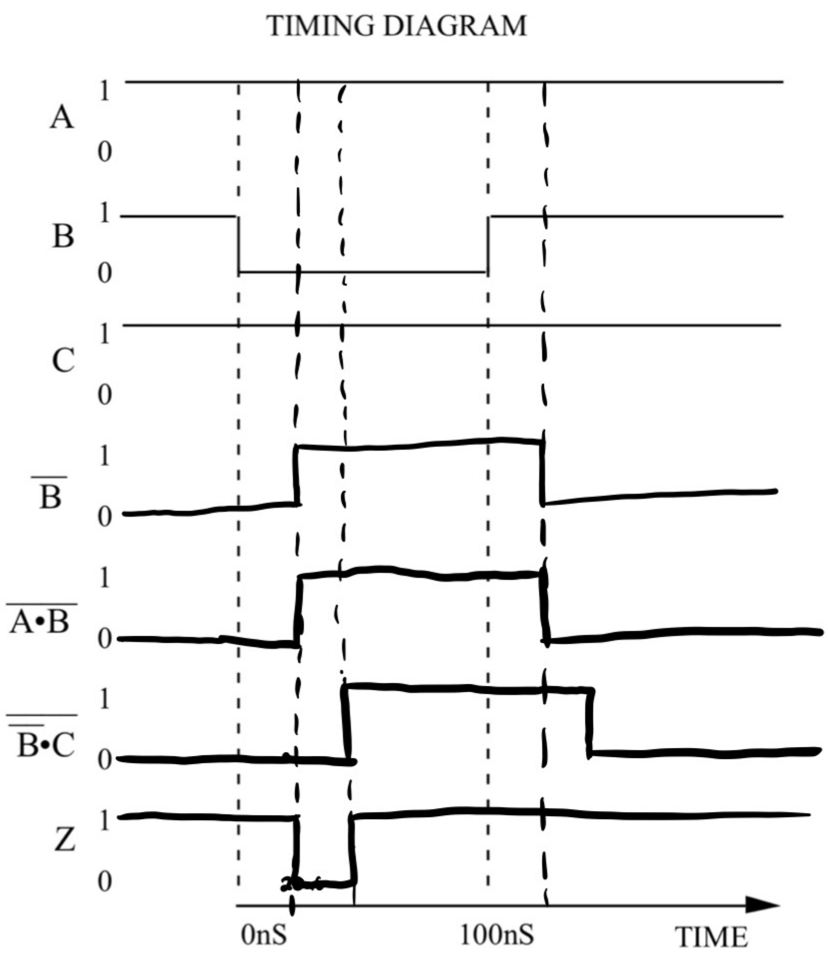
Lab Questions

1. **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 are most likely to observe a static-1 glitch at the falling edge of input ***B*** because a static-1 glitch occurs when the output is momentarily incorrect at the time that ***B*** is switched. When ***B*** is high, the output simply relies on the value of signal ***C***, and maintains a steady output of 1 as long as the signal ***C*** is carrying binary one. However the moment ***B*** is switched to zero, the output instantly also changes to zero since the other signal does not travel through the inverter and the AND gate fast enough to maintain the steady one binary output, resulting in a static hazard.

Post-Lab Questions

Timing Diagram



1. **How long does it take the output Z to stabilize on the falling edge of B (in ns)? How long does it take on the rising edge (in ns)? Are there any potential glitches in the output, Z? If so, explain what makes these glitches occur.**

   It takes the output ***Z*** 40ns to stabilize on the falling edge of ***B***. On the rising edge however, it only takes 0ns to stabilize because Z is already high at the rising edge. There is a potential for a static-1 hazard when ***B*** is switched because ***Z*** drops to 0 for a moment before stabilizing even though it is supposed to stay constant at high.

2. **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?**

   A debouncer circuit is necessary because when a mechanical switch is pressed, it "bounces" multiple times in a fraction of a second, sending conflicting electrical signals through the circuit. The debouncer circuit eliminates this issue by ensuring that only one signal is passed through when the switch is either pressed or released, allowing the switch to function accurately. The switch in Figure 17 on the General Guide works because it essentially behaves like a RS latch with pull-up resistors. In order to eliminate contact bounces, all inputs must be held at some logic level, and the pull-up resistors make this possible. When the switch is securely in contact with position ***A***, the input ***D*** is connected to ground and carries a signal of 0 while input ***G*** carries a signal of 1 since it is connected to power through the resistor. The signals of ***D*** and ***G*** are switched when the switch is securely attached to position ***B***. So in the instant when the switch is not securely attached to position ***A*** or ***B*** in the debouncer circuit, both inputs ***D*** and ***G*** carry a signal 1 because they remain attached to power through the resistors. This ensures that each input is held at a constant logic level, allowing the switch to function correctly while it is in the process of "bouncing".

General Guide Questions

GG.6

1. **What is the advantage of a larger noise immunity?**

   A larger noise community essentially ensures that the output of a gate is less susceptible to erratic changes in the pulse and voltage. Noise immunity is the maximum amplitude of a positive or negative pulse that can be added to either logic 1 or 0 without changing the output of the gate. Therefore, a larger immunity ensures that the output of the gate is more reliable and consistent with the expected logic value at the gate.

2. **Why is the last inverter observed rather than simply the first?**

   The last inverter is always observed because using the first inverter as reference would not give you an accurate measure of the accumulated noise. Observing the last inverter would give a much better reference for the size of the noise, and result in better success when troubleshooting the gate and trying to eliminate unwanted influence from the pulses.

3. **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 the inverter, you would look at the graph and determine the minimum output voltage and maximum output voltage. Then you would find the last point where maximum voltage (logic 1) is outputted and the first point where minimum output voltage (logic 0) is outputted. We then use these values to determine the two noise immunities for the inverter, one for a low signal input and one for a high signal input. Noise immunity for logic zero can be calculated by subtracting the last point where maximum voltage is outputted by the minimum output voltage. Noise immunity for logic one can be calculated by subtracting the maximum output voltage by the first point where minimum voltage is outputted. The final noise immunity is then the smaller value of the two calculated noise immunities.

GG. 31

1. **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 across multiple LEDs because each LED is slightly different, and therefore requires varying amounts of current since they could operate at different voltages. Therefore, sharing resistors would limit the amount of current each LED receives, driving a majority of the current to the LED with the lowest voltage. As a result, the other LEDs would not be able to function properly due to not draining adequate amount of current. Theoretically, if all the LEDs operated the same way and at the same voltage, then sharing a single resistor is feasible because then all LEDs would drain equal amounts of current and operate properly. However, this is very difficult to do, and therefore standard practice is to use a single resistor for each LED because it ensures that the LED is receiving proper current without risking the functionality of that LED or the other LEDs around it.

Conclusions

This experiment introduced us to static-1 hazards, and how to account for them when building circuits. Previously, we focused on deriving minimal SOP equations and focusing on K-MAPs and other logic relations, but now we are learning their significance and tradeoffs when designing circuits. For example, we saw that the static-1 hazard was resolved with a simple redundant SOP term by grouping all adjacent min-terms. While this adds more components, it also resolves the issue of the extended delay at the inverter of a 2-1 MUX. We also saw that every chip is manufactured and therefore functions slightly differently, so there is also the possibility of not experiencing a static-1 hazard in the first place. It was also interesting to see simple debugging tricks to further our understanding of the material. For example, I learned that adding a small capacitor increases the delay, allowing the oscilloscope to show us the static-1 error in the first place. Without, the glitch is so minimal that even the oscilloscope might not pick it up. This also exposed us to the concept of noise immunity, and how it is important to ensure that logic gates can ignore the affects of unwanted pulses and signals. While this interruptions are very short, they cause the whole circuit and system to function inaccurately. A debouncer switch is another example of this. Connecting them to pull-up resistors ensures that all inputs always have a logic bit assigned, ensuring that the output will remain consistent while the switch is bouncing. This lab was also my first time writing a lab report, so it taught me how to approach a lab and maintain records of all the detail and work that went into designing and building the circuit.