ECE 385
Spring 2023
Experiment #1

Introductory Experiment

Cain Gonzalez
Section AL1 / 4:00pm-5:00pm
Dohun Jeong

Purpose of Circuit

The purpose of this lab is to become familiar with static hazards and noise and how to design circuits in such a way that we can avoid them. To do this we make a simple 2-1 MUX in both a naive and redundant way and examine the output's response to each.

Written Description of Circuit

To begin, we need to create a truth table according to how a 2-1 MUX should function. In this case, we will say C is our 0th input to the MUX, A is our 1st input, and B is our selector bit. Thus, when B is 0 the output should be whatever C is and when B is 1 the output should be whatever A is. The truth table and associated K-Map are as follows with minterms highlighted:

A	**B**	**C**	**F**
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

A BC->	**00**	**01**	**11**	**10**
0	0	1	0	0
1	0	1	1	1

It is expected that when both A and C are 1, the output should be one regardless of the value of B. However, when we actually build this circuit and change the value of B, we notice a very small moment when the output of the MUX is actually 0 as seen in Figure 1.

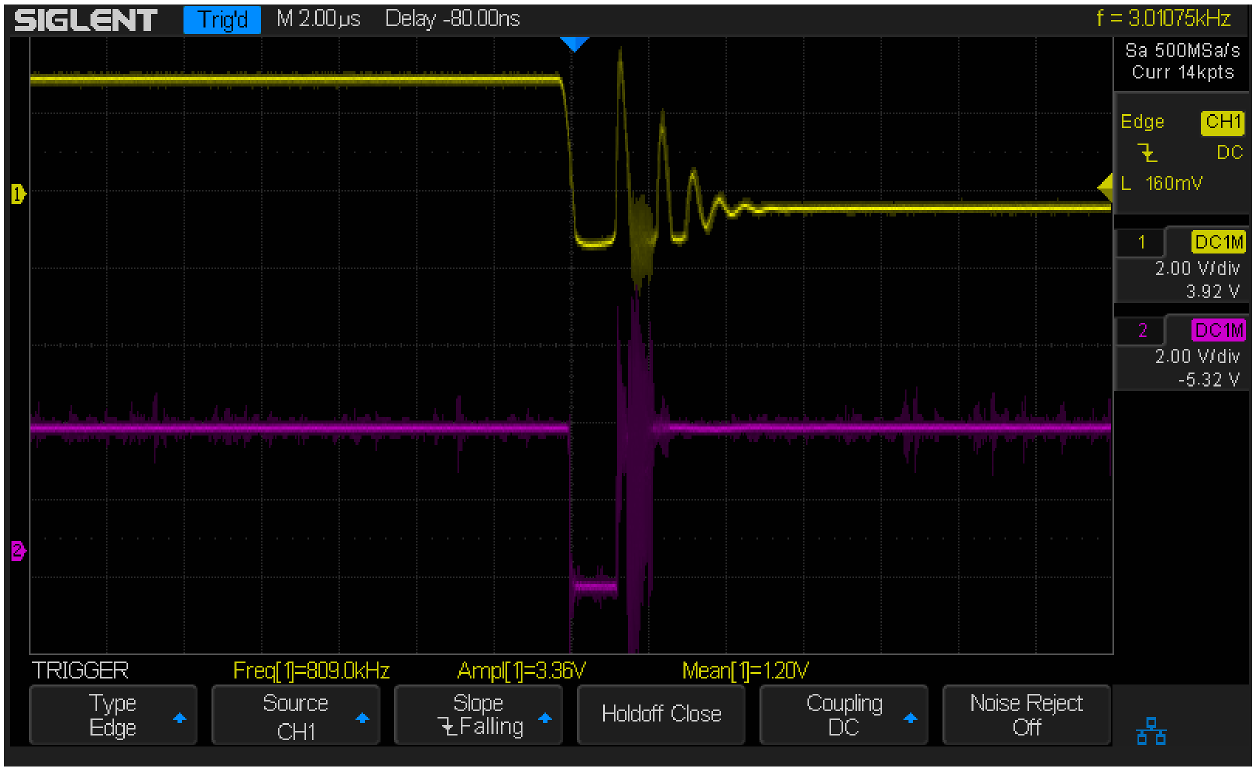


Figure 1. The yellow signal is B and the purple signal is the output

This is called a Type 1 Static Hazard, which will be explained in further detail later in the report. To account for this, we update our K-Map to include redundancy, meaning that we circle extra terms to make input transitions smoother and negate the effects of gate delays on our output.

A BC->	**00**	**01**	**11**	**10**
0	0	1	0	0
1	0	1	1	1

Here, we have the second column and last two elements of the second row as terms, but the difference is that we add an additional term which covers the second two terms of the second row, adding a layer of redundancy. After building the circuit with the additional term and changing the value of B, we see that the error is no longer present, shown in Figure 2.

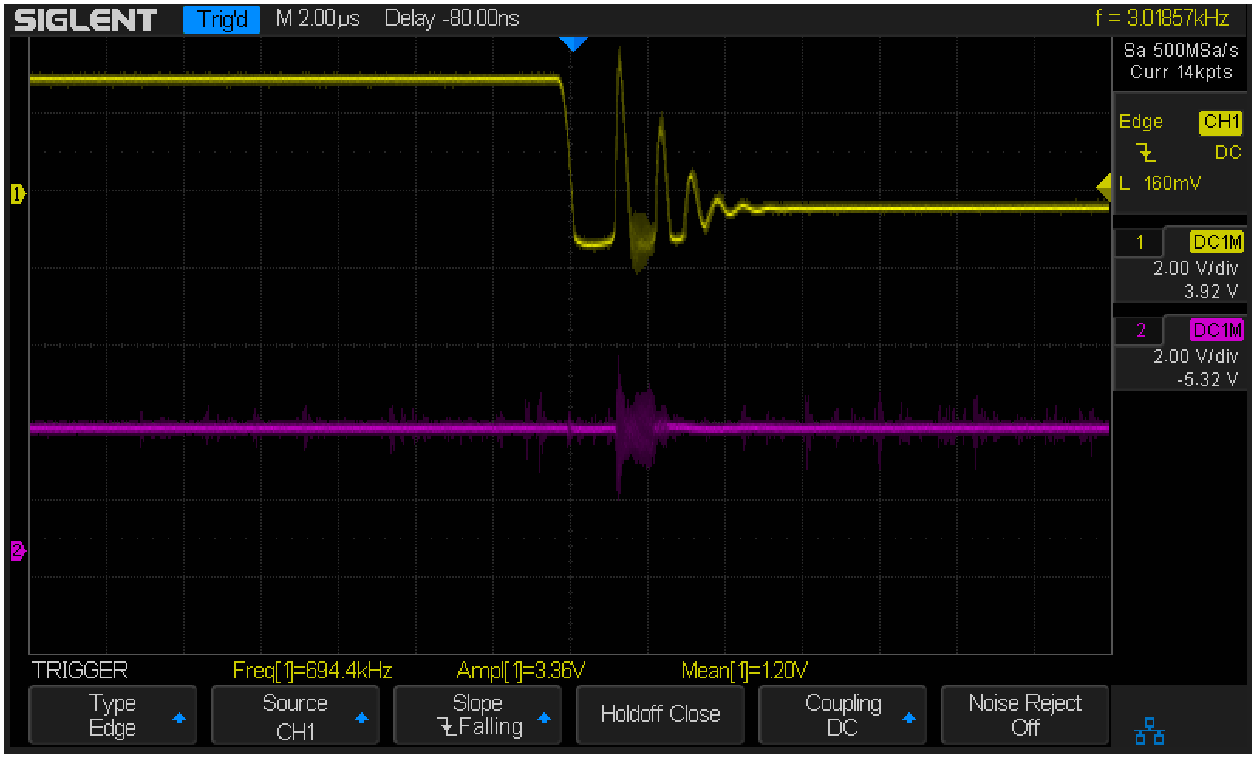


Figure 2. After adding redundancy to the circuit, the Type 1 Static hazard is no longer present

After making the switch, you can see that the Static Hazard is no longer present.

An additional note to be made about the circuit is that placed near the two 7400 ICs are two 0.1uF capacitors wired between Vdd and GND. These are placed there to act as decoupling capacitors and help reduce the noise created by switches within the CMOS transistors. The reason they are required is that when CMOS transistors switch states, there is a short period of time in which there is a short circuit between Vdd and GND, which generates a lot of noise. Placing the decoupling capacitors between power and ground acts as a buffer when this occurs and helps reduce the noise.

Logic Diagrams

Figure 3 shows the naive 2-1 MUX implementation, without redundancy. Here, if both A and C are 1, it is expected that F would always be one regardless of B. But say B is 1 and switches to 0. Because of the gate delay of a NAND gate, the topmost NAND (A NAND B) will flip to 0 at the same time the leftmost NAND (B NAND B) flips to 1. However, this means both inputs to the rightmost NAND gate ((A NAND B) NAND (B' NAND C)) will be 1 for a very short amount of time, driving F to 0. A moment later B' NAND C will update and go to 0, driving F back to 1. This temporary incorrect output is called a Type 1 Static Hazard.

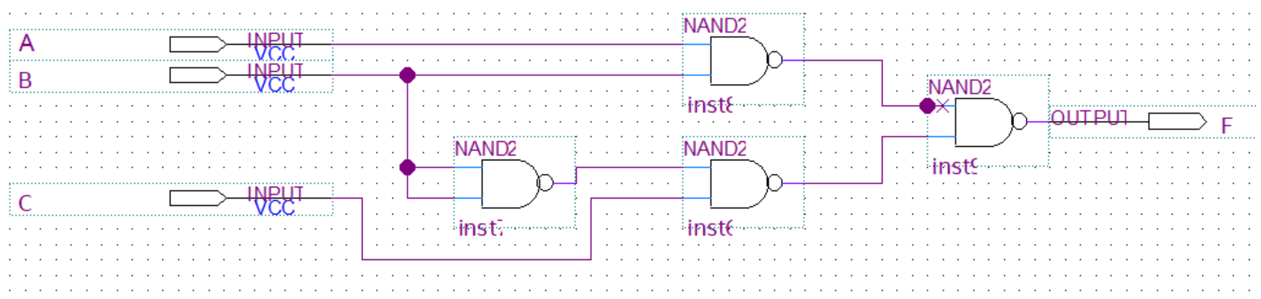


Figure 3. Logic diagram with Type 1 Static Hazard

To fix this, we add a bit of redundancy to our circuit in the form of an additional term in the K-Map, overlapping AC. This ends up becoming 3 additional NAND gates that you can see in Figure 4, but as was seen in Figure 2, it eliminates the static hazard and keeps the output consistent while A and C are 1. Here, F is equivalent to Z it just has a different name.

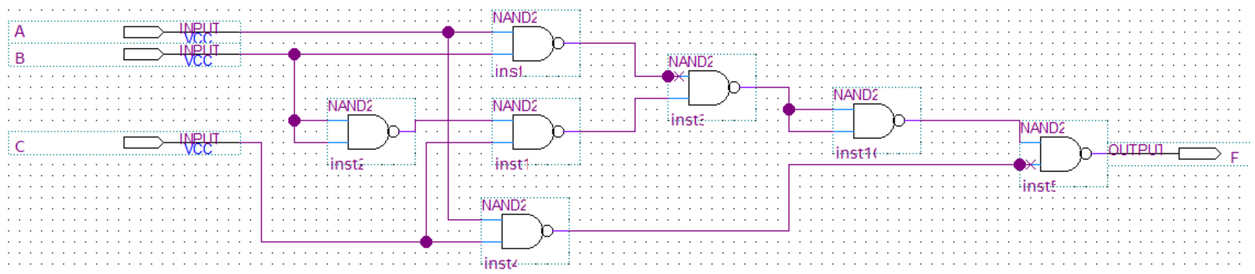


Figure 4. Logic diagram with no Static Hazard

Component Layout

In our lab we use two 7400 quad 2-1 NAND chips, the first of which we use to compute A NAND B, B’, and ((A NAND B) NAND (C NAND B’)). The second one is used to compute A NAND C, ((A NAND B) NAND (C NAND B’))’, and Z. The chip pin labels and fritzing are shown in Figure 5.

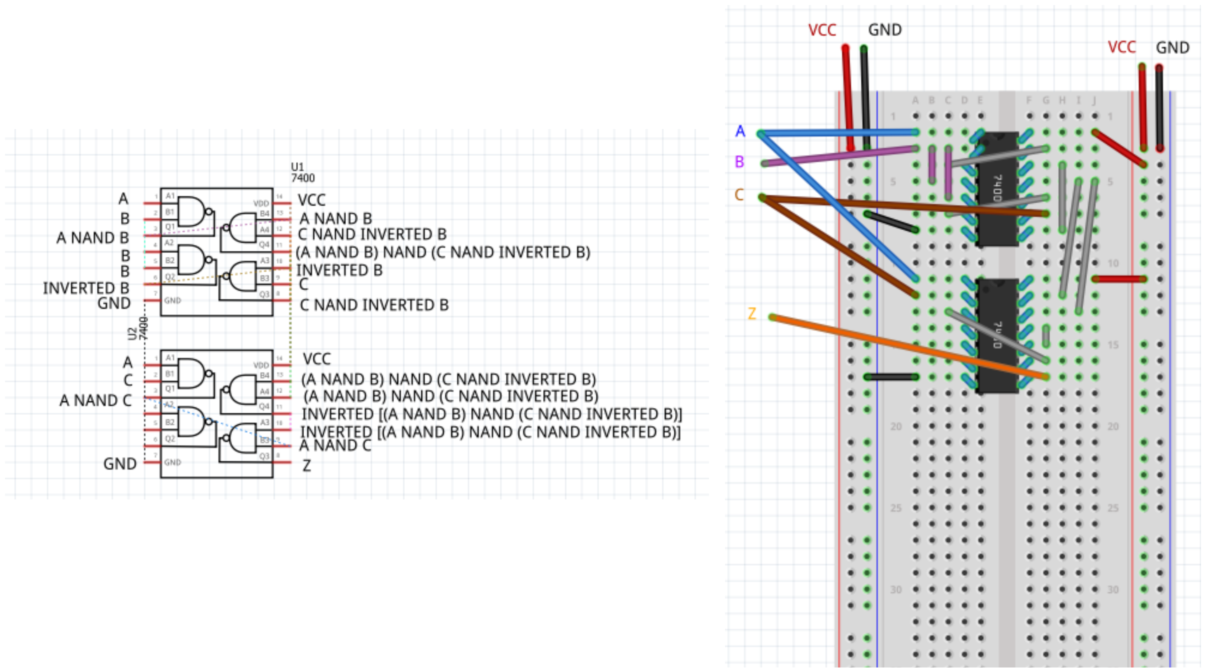

U1
7400
A
B
A NAND B
B
B
INVERTED B
GND
VCC
A NAND B
C NAND INVERTED B
(A NAND B) NAND (C NAND INVERTED B)
INVERTED B
C
C NAND INVERTED B
U2
7400
A
C
A NAND C
GND
VCC
(A NAND B) NAND (C NAND INVERTED B)
(A NAND B) NAND (C NAND INVERTED B)
INVERTED [(A NAND B) NAND (C NAND INVERTED B)]
INVERTED [(A NAND B) NAND (C NAND INVERTED B)]
A NAND C
Z
VCC
GND
VCC
GND
A
B
C
Z
7400
7400

Figure 5. The chip pinout and fritzing (In this case, F from the other schematics is Z)

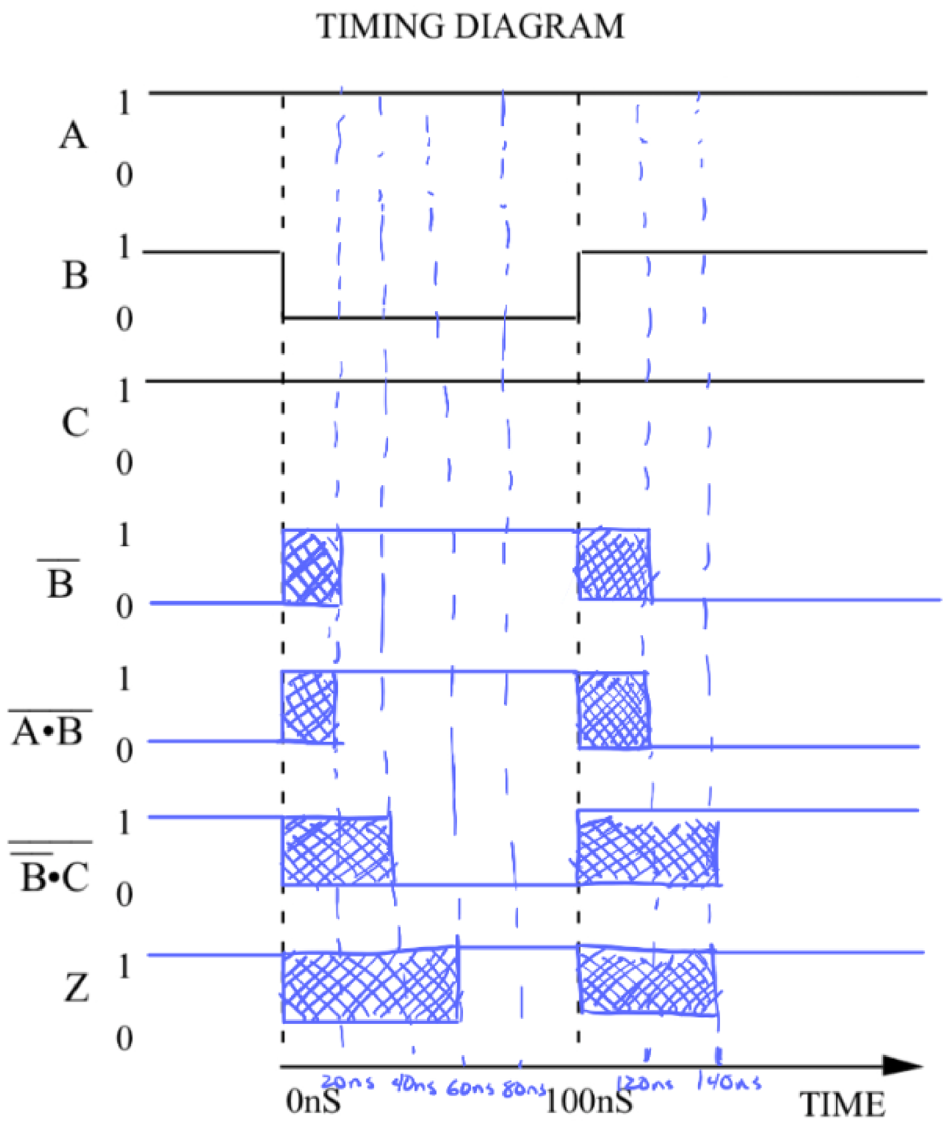
Answers to Pre-Lab Questions

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?

For the circuit of part A, we are more likely to observe a glitch on the falling edge because Z is undetermined for a greater period of time (60ns) than it is on a rising edge (40ns). The reason the maximum undetermined period for Z is greater on the falling edge than the rising edge is because on the rising edge, (AB)’ is 0 after 20ns, which guarantees that Z is 0, whereas on the rising edge, (AB)’ is 1, which cannot by itself determine Z. Thus, Z is undetermined for an additional 20ns as it waits for (B’C)’ to be determined.

Answers to Post-Lab Questions

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 Z 40ns to stabilize on the rising edge, where it takes Z 60ns to stabilize on the falling edge. There are potential glitches in Z, both on the rising and falling edge. On the rising edge a glitch could occur because after B switches to 1, it must pass through the inverter gate and the (B’C)’ gate to reach the Z output gate, whereas it must only pass through (AB)’ to reach the Z output gate through the other path, meaning that there is a period of time where it is possible for both inputs to be 1 and for Z to be incorrect. However, since (AB)’ becomes 0 after 40ns, Z is guaranteed to be 1 from that point on. Thus, it only takes Z a maximum of 40ns to stabilize on the rising edge. On the falling edge the same phenomenon occurs, but (AB)’ becomes 1 which cannot by itself determine Z, so we must wait for another 20ns for Z to stabilize.

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?**
The debouncer circuit in the general guide works like a latch. When the switch is changed, the value changes, and since the NAND gates feed into each other, they provide each other support in the case that any bouncing occurs. Additionally, it behaves as a switch because when C is switched from either one spot to the other, that spot becomes a 1 and the other becomes 0, thus making the output to its gate 1. The other gate then becomes 0. When you flip the switch again, the process reverses, and thus it acts like a switch.

General Guide Questions

What is the advantage of a larger noise immunity? Why is the last inverter observed rather than simply the first? Given a graph of output voltage (VOUT) vs. input voltage (VIN) for an inverter, how would you calculate the noise immunity for the inverter? See the following figure
A larger noise immunity is advantageous because it allows for there to be more error when receiving an input and thus is more reliable to not switch unexpectedly. The reason the last inverter is observed as opposed to the first one is because the first only creates one “pulse” of noise, while the last one receives the culmination of all the previous “pulses” that come before it. Finally, to calculate the noise immunity of the inverter, you take the range of valid inputs for each logic level (1 and 0) and label each of those as their own noise immunities. Then, you take the smaller of the two and that ends up being the overall noise immunity for the inverter.

Irrespective of which polarity you choose for your LEDs, it is important that each LED has its own resistor. 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 because we want each LED to represent a unique signal and when we share resistors we share current and voltage, which may lead to one or more of them displaying incorrect values. To be safe, it is just best to have a resistor for each LED.

Conclusions

After watching the video and completing the lab write up, I have learned a lot about static hazards, IC chips, and switches, and how to avoid the common issues that come along with using them. When building circuits in the future, I will be sure to practice these good habits to make debugging easier. In order to make circuits better in the future, I will make sure to place 0.1uF capacitors close to the ICs to reduce noise, use debounced switches, and implement redundancy in my logic diagrams to prevent type 1 static hazards.