ECE 385

Spring 2020

Experiment 1

Introductory TTL Experiment

Arkajit Dutta

Section ABD – Tuesday 6 pm

Nicholas Cebry, Wenjie Pan

Purpose of Circuit

The first lab was intended to introduce us to TTL logic and how we can build a 2-1 multiplexer using AND, NOR, OR gates. We later had to convert the entire logic in terms of NAND gates as we had to use 7400 gates for the implementation of the lab. This lab also introduced the topic of Static-1 hazards, which arises when the output is 0 temporarily when the logic suggests the output should be one. In this lab, we overcame this problem by combining adjacent SOP terms in the Karnaugh Maps to not allow for a delay when we switch between different SOP terms.

A multiplexer has inputs and select bits which help determine the output from the multiplexer. In this lab, we worked with a 2-1 multiplexer with Inputs – A,B, one select bit – S and output – C. The select bit is used to select the input logic. A multiplexer is extremely relevant in processor architecture and is the primary component in many design aspects.

Description of Circuit

In this circuit, we had 2 inputs – A and B, one select bit – S and output – C. The output C is dependent on the select bit. Therefore, if bit S is 1 it will select B and therefore, output C will be 1 when B is 1. When bit S is 0 it will select A and output C will be 1 when A is 1. The functioning of a MUX is described in the figure below.

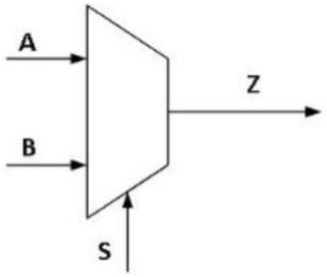

Fig 2.1:2-1 MUX

Truth Table

A	B	C	F(ABC)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Karnaugh Map

C	AB: 00	01	11	10
0	0	0	1	1
1	0	1	1	0

Fig 2.2: Truth Table and K-map

Using the K-Map and Truth Table we can deduce the logic of the circuit – **F(ABS) = AS' + BS.** The logic is implemented below using AND, NOT, and OR gates, as seen in figure 2.4. We had to convert the entire logic into NAND gates as we are using 7400 chips for the lab, therefore the updated logic circuit using only NAND gates is seen in Figure 2.5

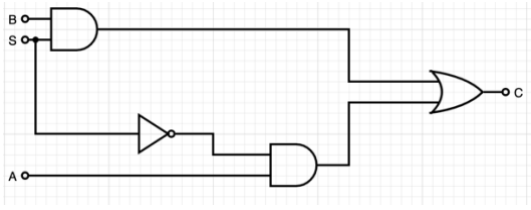

Fig 2.4: Logic Circuit

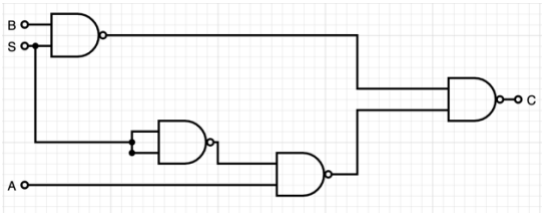

Fig 2.5: Logic Circuit using NAND gates

As seen in both the logic diagrams, the process takes a while longer for the AS' term as there is an extra gate (NOT in Figure 2.4 and an additional NAND gate in Figure 2.5). This leads to a delay in the clock cycle and therefore, momentarily, the output C is not what the logic suggests it should be. This is a Static-1 hazard which is caused by the output C being 0 when it should be 1. The delay in the clock cycle is demonstrated by the timing diagram in Figure 2.6

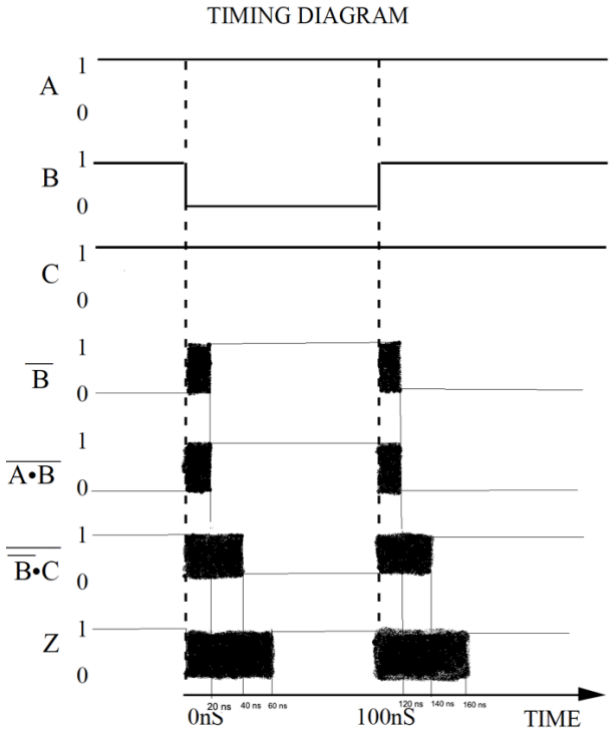

Fig 2.6: Timing Diagram

The glitch that occurs is the Static-1 Hazard as the output is delayed by anywhere between 0-60 ns, resulting in inaccurate results momentarily. To overcome this hazard, we combined the adjacent SOP terms on the K-Map so that there is virtually no time spent transitioning from one SOP term to another. This would negate the delay and overcome the static-1 hazard. If we combine the adjacent terms on the K-Map we get a redundant term –AB– which helps remove the glitch.

The new logic for the circuit would therefore be **F(ABS) - AS' + BS + AB.**

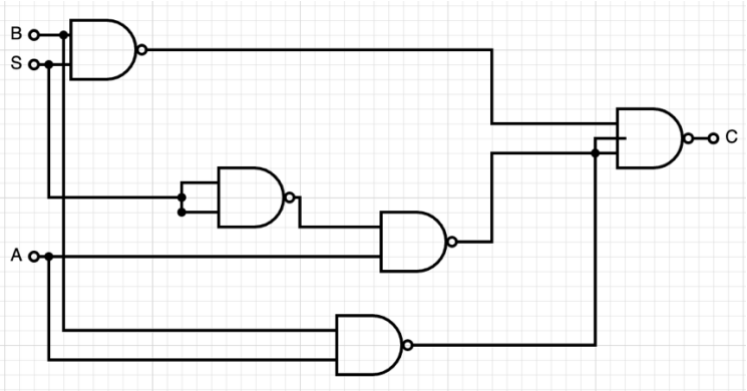

Fig 2.7: NAND glitch-free circuit

Key – Output C (green)
Select Bit S (yellow)

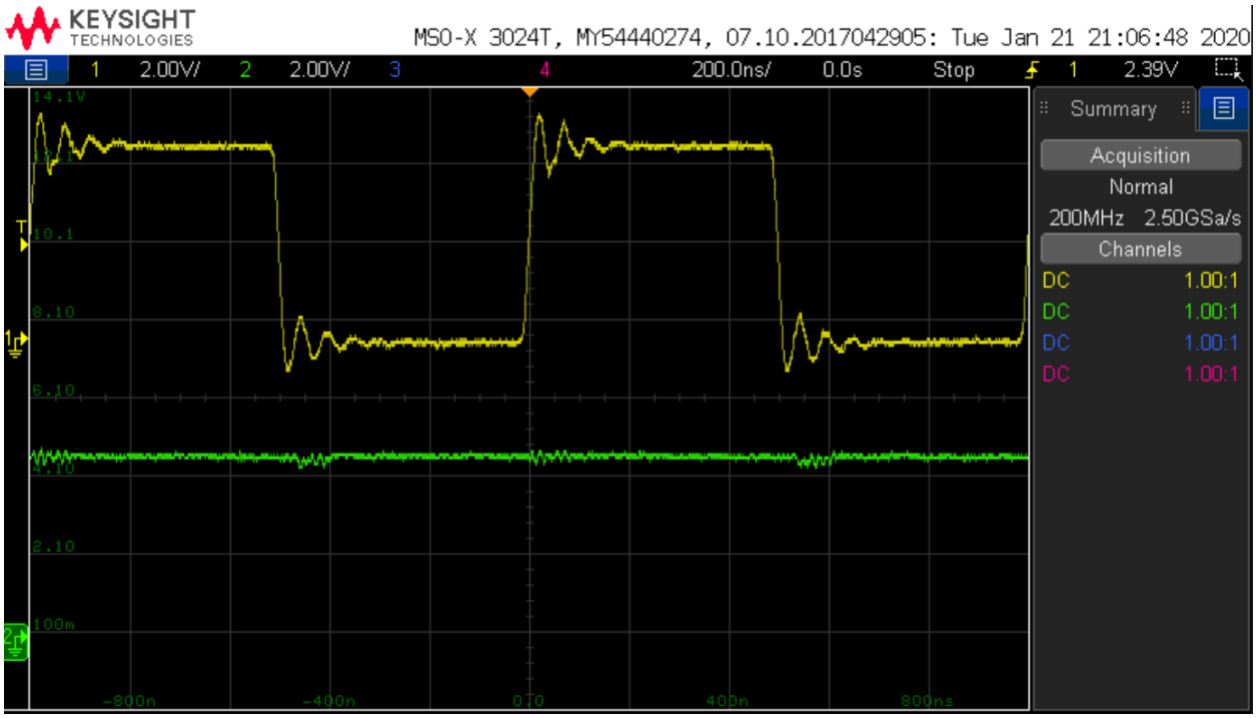

Fig 2.8 : Oscilloscope trace with no delay

The inputs B and A were both turned to 1, therefore the output C should also be at 1 at all times, irrespective of the value of Select Bit S, which is reflected in the diagram above. The circuit behaves exactly as it should during its steady state and therefore we have successfully removed the Static-1 Hazard by adding the additional term **AB.**

Key – Output C (green)
Select Bit S (yellow)

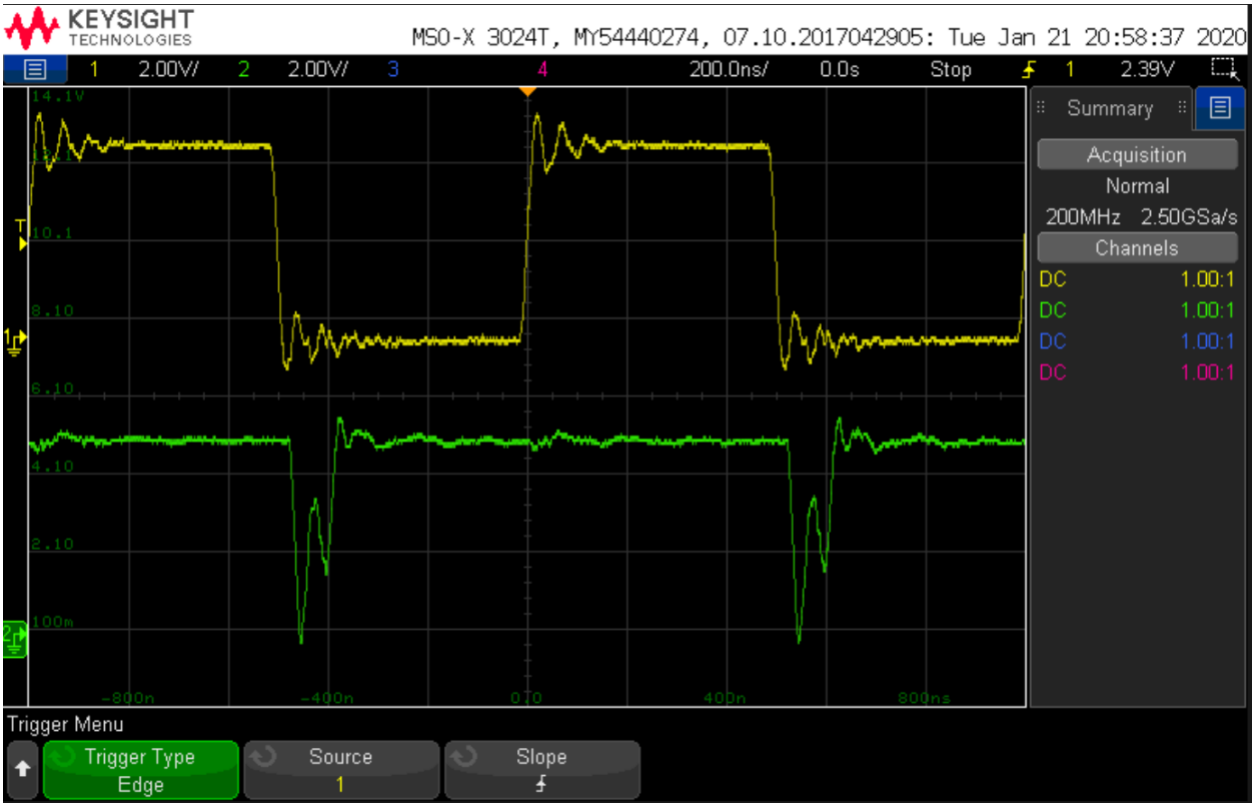

Fig 2.9: Oscilloscope trace showing a delay

In this diagram the glitch is visible during the falling edge of S, as the output C falls to 0 momentarily before reaching the steady state of 1. The rising edge of S does not have an effect because the logic is always 1 when S is 1. The glitch is seen again on the second falling edge of S.

Circuit and Component Layout

In this circuit, we used 3 7400 chips (2 NAND input). The first 7400 chip performed the NAND functions a described in the logic above. The second NAND gate was primarily used to invert the Select Bit (S) by passing it through 4 NAND gates. The third NAND gate had the output of A'B' and the final output from the first 7400 chip. These two were inputs for the second NAND gate in the third 7400 chip, which gave the output C.

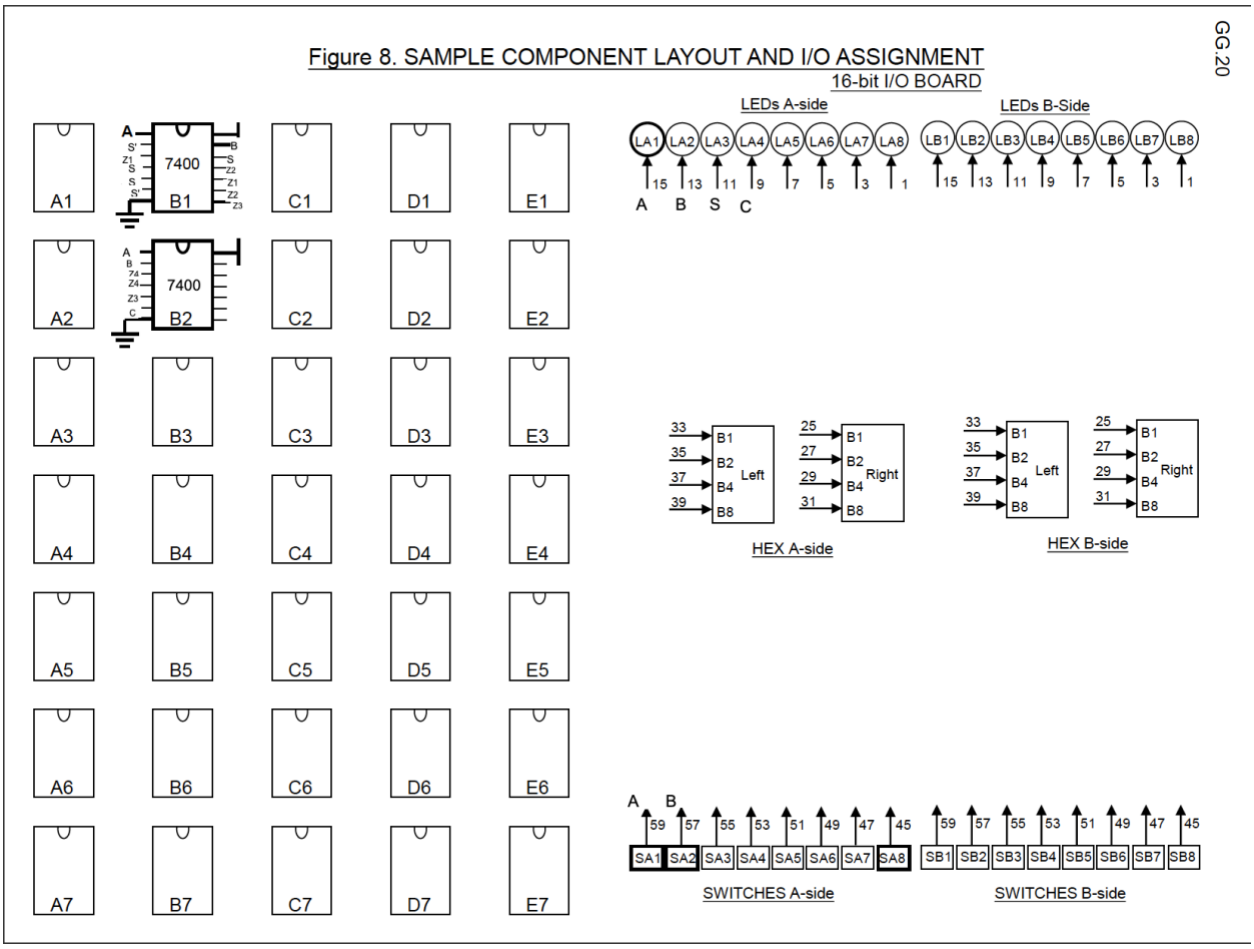

Fig 3.1: Component Layout

Post-Lab Questions

1) As seen in Figure 2.6, there were delays of a maximum of 60ns for Z on both the rising and falling edge of B. These delays were caused by the sum of 20ns for B to stabilize in a cycle and 40ns for $\overline{B}$.C.

2) Mechanical switches are spring loaded, therefore, during switching (turning on and off) the terminals hit each other and bounce back. This effect can be nullified by using a debouncing switch, as shown in Figure 3.2. A debounce switch holds the values of the inputs A and B, because the switch acts like an SR latch. Thus, it removes the ill-effects of bouncing as experienced in a mechanical switch.

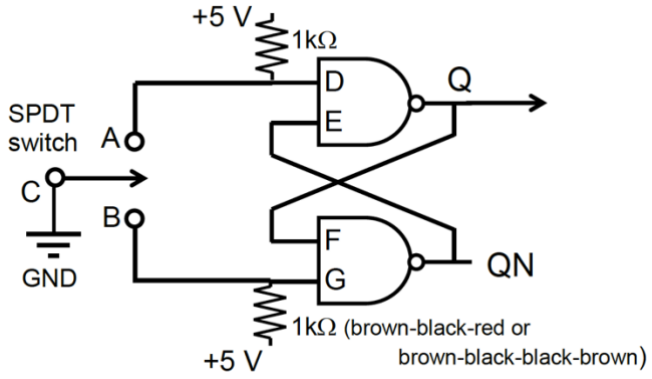

Fig 3.2: Debouncing switch

GG.6: Having noise immunity increases the reliability of our results and outputs, as we can be confident that no background noises are affecting our results. The last inverter is observed instead of the first inverter because after the first inverter multiple gates are added, which will significantly increase the time delay. Noise immunity can be calculated by looking at the graph of V_{out} vs V_{in} and comparing the input voltage to the output voltage. The upper boundary of the range – lower boundary of the range would give us the noise immunity.

GG.29: It is bad practice to share resistors when multiple LED's are used because every LED has a current limit in theory, but in real life these characteristics may not necessarily apply. Therefore, one LED might use more current than the other LED's in connection therefore affecting functionality of the whole circuit.

Conclusion

Being an introductory lab, this lab was relatively easy to implement, test and debug. We did encounter a slight problem when our V_{cc} on the protoboard was not displaying 5V on the multimeter. After the TA assisted us with that problem, there were no real issues in the functionality of the circuit. The concept of Static – 1 hazards was new for me and it was really interesting to learn about how it affects circuits and can glitch the output. The idea of sometimes not fixing a glitch is also intriguing because while designing circuits, engineers need to weigh in the complexity of the circuit and it might not be feasible to always remove glitches. This lab was extremely informative and forms the foundation of other TTL labs to follow.