EIE2810 Digital Systems Design Laboratory

Laboratory Report #3

Name: Tengfei Ma

Student ID: 121090406

Date: 2024/3/15

The Chinese University of Hong Kong, Shenzhen

This laboratory consists of 3 experiments, they are:

- Experiment A: Realizing the Combinational Logic of $Y = AB + \overline{B}C\overline{D} + \overline{A}CD$ With Both Hardware and Software
- Experiment B: Finding and Eliminating the Timing Hazard in the Finished Circuit for the Combinational Logic of $Y = AB + \overline{B}C\overline{D} + \overline{A}CD$ With Both Hardware and Software
- Experiment C: Design and Build a Multiply Logic Operator

1. Experiment A

1.1 Design

The aim logic is $Y = AB + \overline{B}C\overline{D} + \overline{A}CD$, to realize this logic, a circuit is designed. The designed circuit diagram is shown in Figure 1.

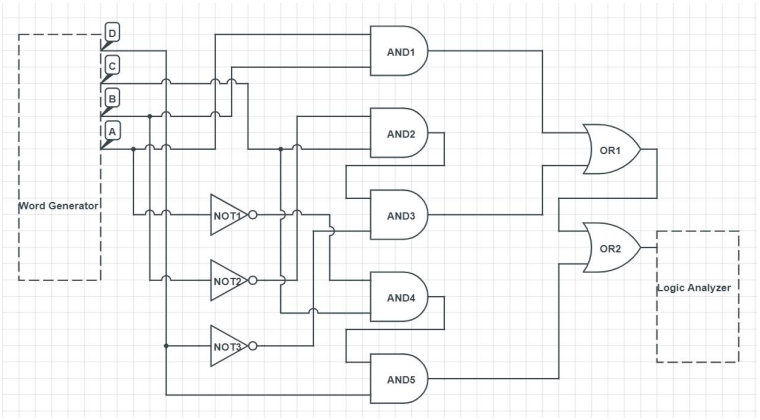

Figure 1 The Designed Gate Level Circuit for the Logic $Y = AB + \overline{B}C\overline{D} + \overline{A}CD$

As shown in Figure 1, 3 NOT Gates are used to produce $\overline{A}$, $\overline{B}$, and $\overline{D}$, 1 AND Gate is used to produce AB, 2 AND Gates are used to produce $\overline{B}C\overline{D}$, 2 AND Gates are used to produce $\overline{A}CD$, and 2 OR Gates are used to combine AB, $\overline{B}C\overline{D}$, and $\overline{A}CD$. A word generator is used to generate the signal while a logic analyzer is used to show the output signal.

1.2 Results

1.2.1 The Results of the Simulation

For the logic $Y = AB + \overline{B}C\overline{D} + \overline{A}CD$, its truth table is shown in Table 1.

Table 1

A	B	C	D	Y
0	0	0	0	0
1	0	0	0	0
0	1	0	0	0
0	0	1	0	1
0	0	0	1	0
1	1	0	0	1
1	0	1	0	1
1	0	0	1	0
0	1	1	0	0
0	1	0	1	0
0	0	1	1	1
1	1	1	0	1
1	1	0	1	1
1	0	1	1	0
0	1	1	1	1
1	1	1	1	1

The IC level circuit used for simulation in Multisim is shown in Figure 2, and the simulation results are shown in Figure 3.

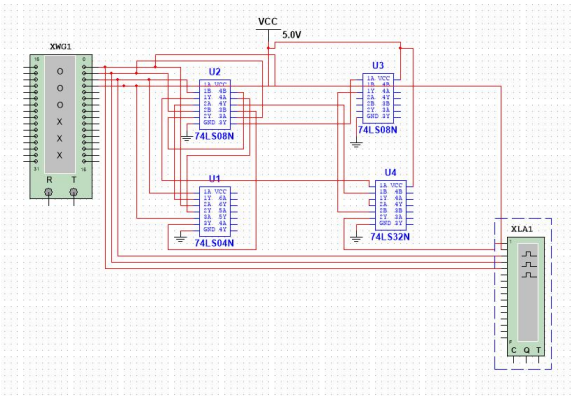

Figure 2 The Chip Level Circuit Diagram for Simulation

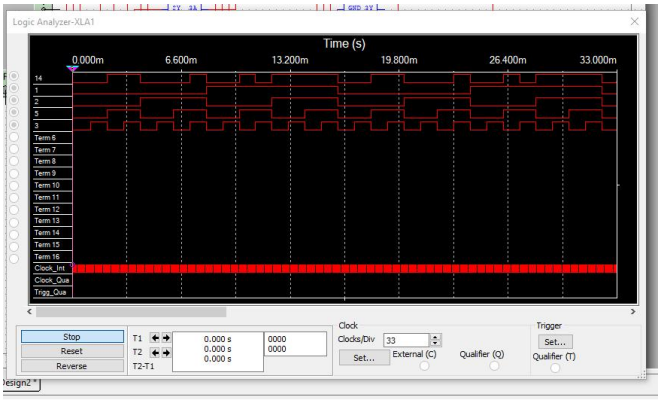

Figure 3 The Simulation Results

In Figure 3, the first line represents signal Y, the second line represents signal A, the third line represents signal B, the fourth line represents signal C, and the fifth line represents signal D. As shown in Figure 3, when signal A and signal B are both at a high level, no matter what states signal C and signal D are, signal Y is at a high level; when B is at a low level, C is at a high level, and D is at a low level, no matter what state signal A is, Y is at a high level; when signal A is at a low level, both signal C and signal D are at a high level, no matter what state signal B is, signal Y is at a high level; and signal Y is at a low level otherwise, which all goes well with the truth table. This means our designed circuit successfully realized the aiming logic.

1.2.2 The Results of Hardware Verification

The finished circuit is shown in Figure 4, and the luminous conditions for different input logic are shown in Figure 5 and Figure 6.

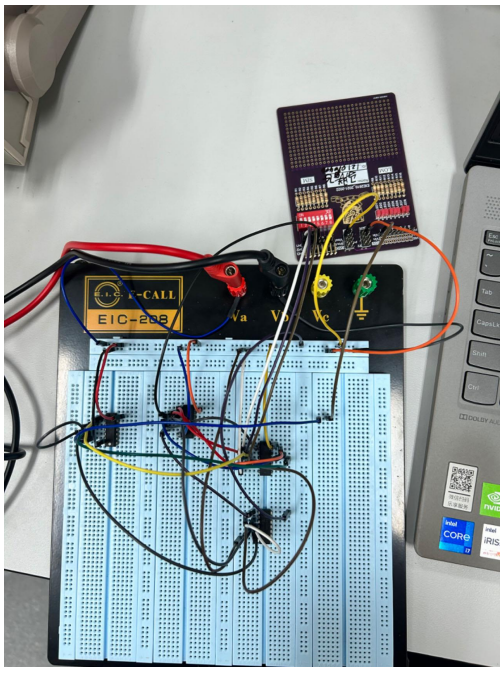

Figure 4 The Finished Circuit for Verification

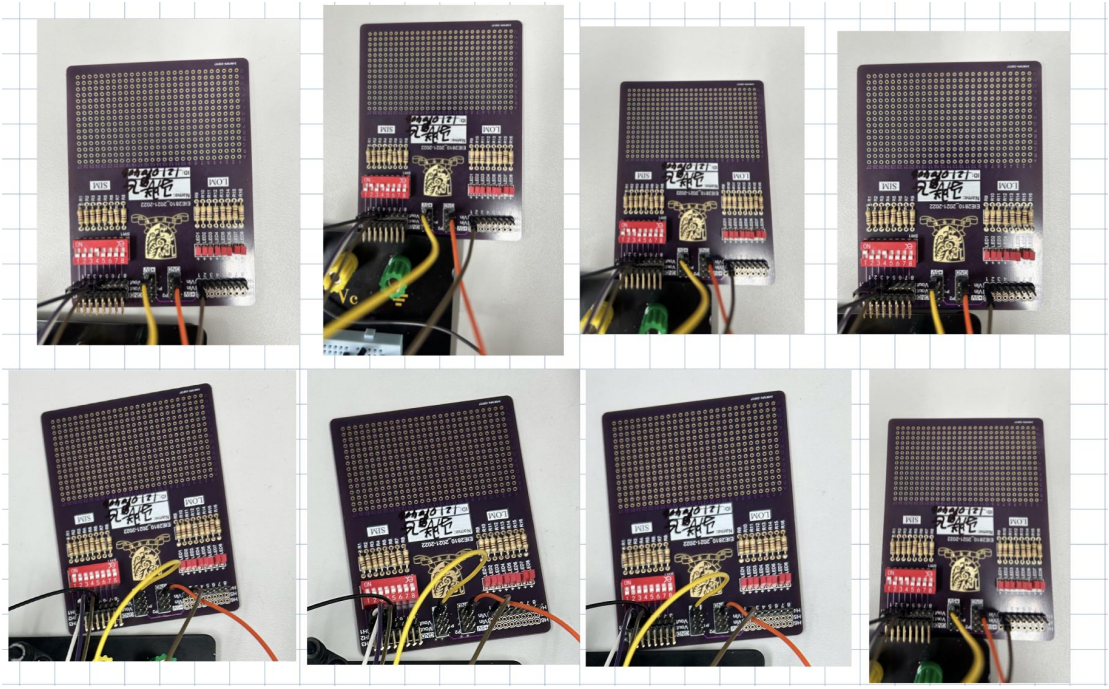

Figure 5 The Conditions That the LED is not Lightened

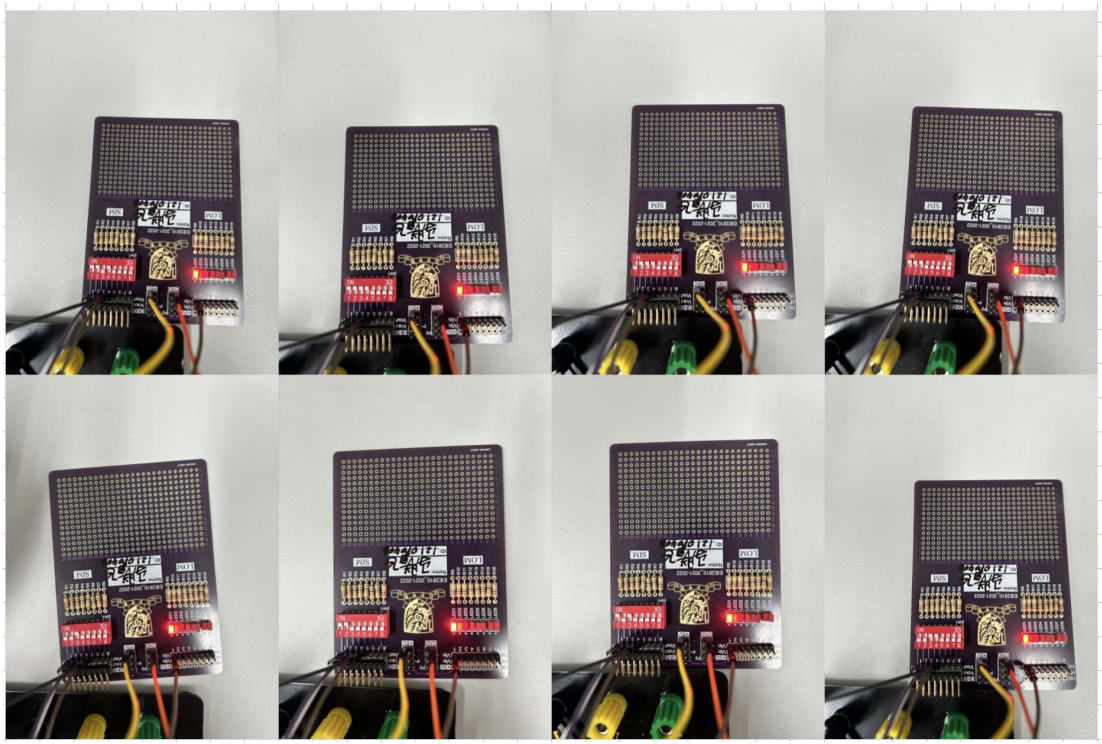

Figure 6 The Conditions That the LED is Lightened

As shown in Figure 5 and Figure 6, when signal A and signal B are both at a high level, no matter what states signal C and signal D are, LED is lightened (at high-level voltage); when B is at a low level, C is at a high level, and D is at a low level, no matter what state signal A is, LED is lightened (at high-level voltage); when signal A is at a low level, both signal C and signal D are at a high level, no matter what state

signal B is, LED is lightened (at high-level voltage); and the LED is not lightened (at low-level voltage) otherwise, which all goes well with the truth table. This means we successfully used hardware to verify the aiming logic.

2. Experiment B

2.1 Design

To eliminate the glitch, a BCD signal is added to the original circuit, the gate-level circuit diagram for this design is shown in Figure 7. In Figure 7, there are some wires with yellow color connecting the signal generator and the circuit, they are hard to figure out but they do exist.

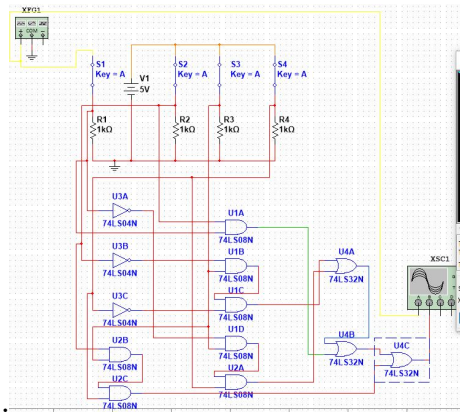

Figure 7 The Gate Level Circuit Diagram Designed to Eliminate the Glitches

2.2 Results

2.2.1 The Results of Software Simulation

A circuit shown in Figure 7 is built with Multisim for simulation. Before we add the BCD signal, the signal given by the signal generator and the signal got by the oscilloscope is shown in Figure 8.

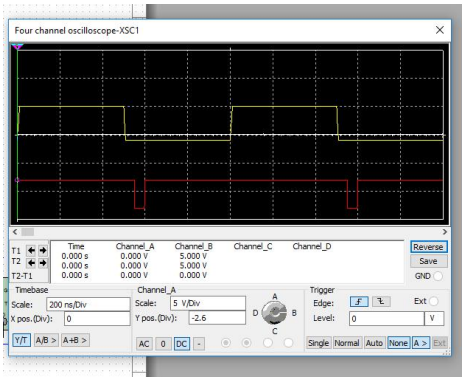

Figure 8 The Signal Got by Oscilloscope With Glitches

In Figure 8, the yellow signal represents the signal given by the signal generator while the red one represents the signal got by the oscilloscope. As shown in Figure 8, there exist some glitches when the input signal changes from high level to low level. After adding a BCD signal to the output signal, we can have the signal given by the signal generator and the signal got by the oscilloscope just like the signals shown in Figure 9.

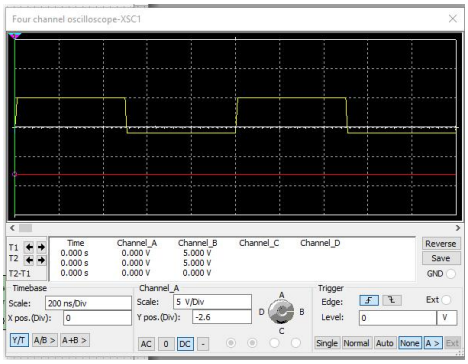

Figure 9 The Signal Got by Oscilloscope Without Glitches

In Figure 9, the yellow signal represents the signal given by the signal generator while the red one represents the signal got by the oscilloscope. As shown in Figure 9, the glitches in Figure 8 disappear in Figure 9 with the presence of a BCD signal. By comparing Figure 8 and Figure 9, we can see that we successfully eliminate the glitches by adding a BCD signal to the output signal in the software.

2.2.2 The Results of Hardware Verification

A verification circuit is built according to Figure 7. The figure shown in the oscilloscope before eliminating the glitch is shown in Figure 10 while the figure shown on the oscilloscope after eliminating the glitch is shown in Figure 11.

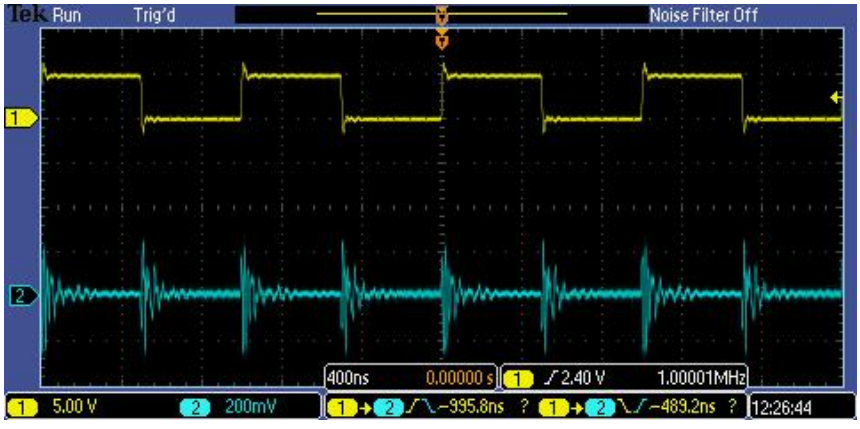

Figure 10 The Received Signal Before Eliminating the Glitch

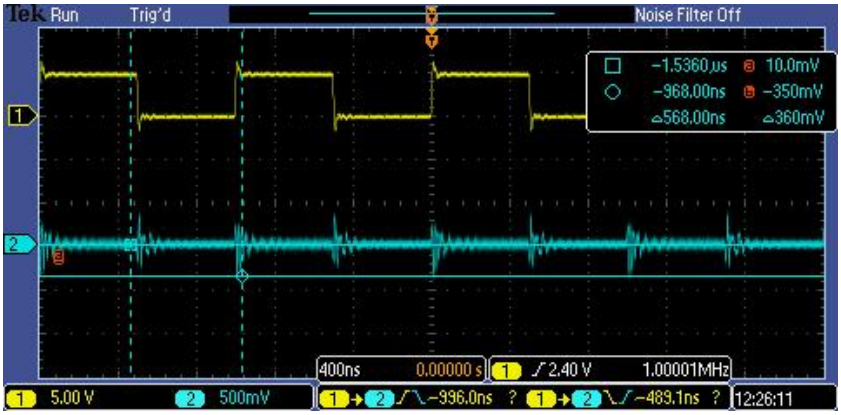

Figure 11 The Received Signal After Eliminating the Glitch

By comparing Figure 10 and Figure 11, we can see that the glitch signals in Figure 11 are much smaller than the glitch signals in Figure 10, which means we successfully eliminated the glitches to some extent. The possible reasons for the appearance of the glitches that cannot be eliminated are:

1) Not Ideal Elements

Although the tools and circuit elements we used were pretty accurate, they were not ideal. This means that it is hard to eliminate the glitches completely.

2) The Gibbs Phenomenon

As shown in Figure 10 and Figure 11, the edges of the input signal have the Gibbs phenomena, which will also be shown in the output signal. So, the "glitches" in the output signal are actually the represents of the Gibbs phenomena in the output voltage, which cannot be eliminated by adding a BCD signal.

2.3 Questions

2.3.1 Two Other Time Hazards

By constructing the K-map (Figure 12), we can know that the two other time hazards exist between 1011, 1111 (i.e. B and $\overline{B}$) and 1110, 1111 (i.e. D and $\overline{D}$).

	CD	00	01	11	10
AB					
00				X	X
01				X	
11		X	X	X	X
10					X

Figure 12 The K-map For the Given Logic

As shown in Figure 12, we can know that to eliminate the time hazard of B and $\overline{B}$, we need to add an $AC\overline{D}$ to the given logic. To eliminate the time hazard of D and $\overline{D}$, we need to add $\overline{A}\,\overline{B}\,C$ to the given logic.

2.3.2 The Reason That Y=A$\overline{A}$ Has Time Hazards

The glitch is caused by a race condition between the clock signal and the A and $\overline{A}$ signals at the inputs of the AND gates. The propagation delays between clock and A and $\overline{A}$ create a short-duration coincidence of high levels (or low levels) at the leading edges of alternate clock pulses, then a glitch comes out. So, as long as we use A, $\overline{A}$, and AND Gate in a circuit, the glitches would always occur.

3. Experiment C

3.1 Design

The truth table for the multiplier is shown in Table 2.

Table 2

A1	A2	B1	B2		Y3	Y2	Y1	Y0
0	0	0	0		0	0	0	0
0	0	0	1		0	0	0	0
0	0	1	0		0	0	0	0
0	1	0	0		0	0	0	0
1	0	0	0		0	0	0	0
0	0	1	1		0	0	0	0
0	1	0	1		0	0	0	1
1	0	0	1		0	0	1	0
0	1	1	0		0	0	1	0
1	0	1	0		0	1	0	0
1	1	0	0		0	0	0	0
0	1	1	1		0	0	1	1
1	0	1	1		0	1	1	0
1	1	0	1		0	0	1	1
1	1	1	0		0	1	1	0
1	1	1	1		1	0	0	1

According to Table 2, we can know the logic for Y3, Y2, Y1, and Y0 are:

$Y3 = A1A2B1B2$

$$Y2 = A1\overline{A2}B1\overline{B2} + A1\overline{A2}B1B2 + A1A2B1\overline{B2}$$

$$Y1 = A1\overline{A2}\,\overline{B1}B2 + \overline{A1}A2B1\overline{B2} + \overline{A1}A2B1B2 + A1\overline{A2}B1B2 + A1A2\,\overline{B1}B2 + A1A2B1\overline{B2}$$

$$Y0 = \overline{A1}A2\overline{B1}B2 + \overline{A1}A2B1B2 + A1A2\,\overline{B1}B2 + A1A2B1B2$$

The circuit is designed according to the logic above, the designed circuit is shown in Figure 13

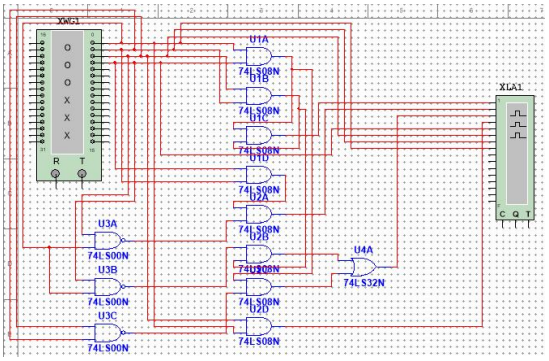

Figure 13 The Designed Circuit for the Multiplier

3.2 Results

3.2.1 The Results of the Simulation

The simulation results are shown in Figure 14.

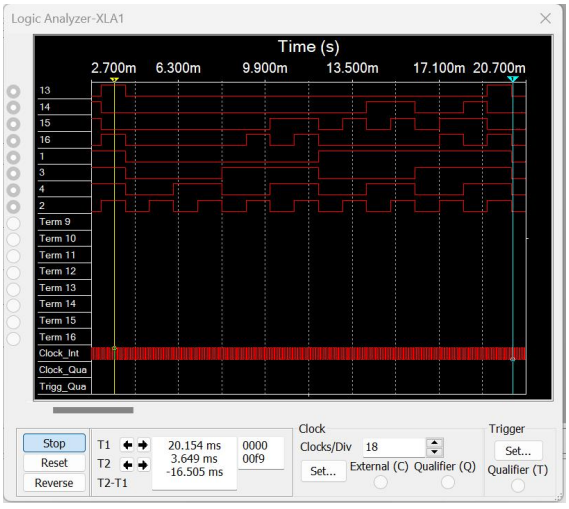

Figure 14 The Simulation Results

In Figure 14, the first four lines represent the output signal, the highest one is the most significant bit (MSB) of the output signal and the fourth line is the least significant bit (LSB) of the outputs signal, the next four lines represent the input signal, the fifth line represents the MSB of the input signal and the eighth line represents the LSB of the

input signal. As shown in Figure 14, the output is 0001_2 (1_{10}) when the input is 01_2 and 01_2 (1_{10}), the output is 0010_2 (2_{10}) when one input is 01_2 (1_{10}) and another is 10_2 (2_{10}), the output is 0100_2 (4_{10}) when both inputs are 10_2 (2_{10}), the output is 0011_2 (3_{10}) when one input is 11_2 and another is 01_2 (1_{10}), the output is 0110_2 (6_{10}) when one input is 11_2 (3_{10}) and another is 10_2 (2_{10}), the output is 1001_2 (9_{10}) when both two inputs are 11_2 (3_2). and the output is 0000_2 (0_{10}) otherwise. All of the results go well with the truth table, which means we successfully simulated the multiplier using Multisim.

3.2.2 The Results of the Hardware

The finished circuit assembled according to Figure 13 is shown in Figure 15. One 74HC00 (NAND Gate), two 74HC08 (AND Gate), and one 74HC32(OR Gate) are used to construct the multiplier.

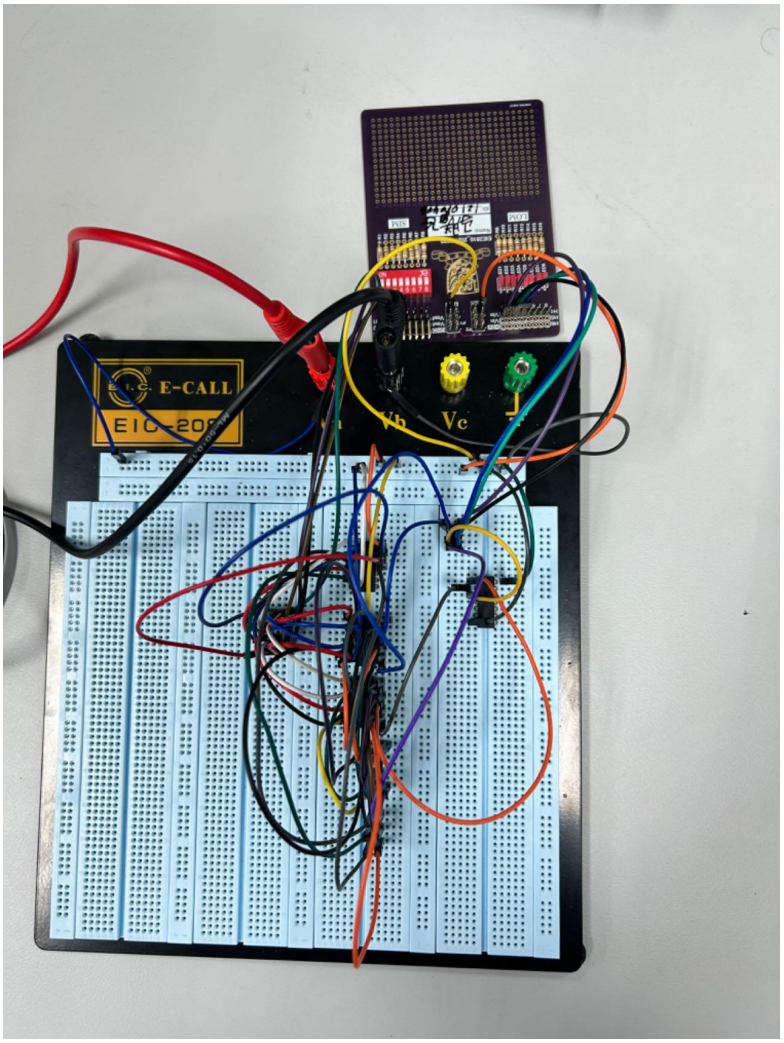

Figure 15 The Finished Circuit For the Multiplier

The luminous conditions for different input logic are shown in Figure 16 and Figure 17.

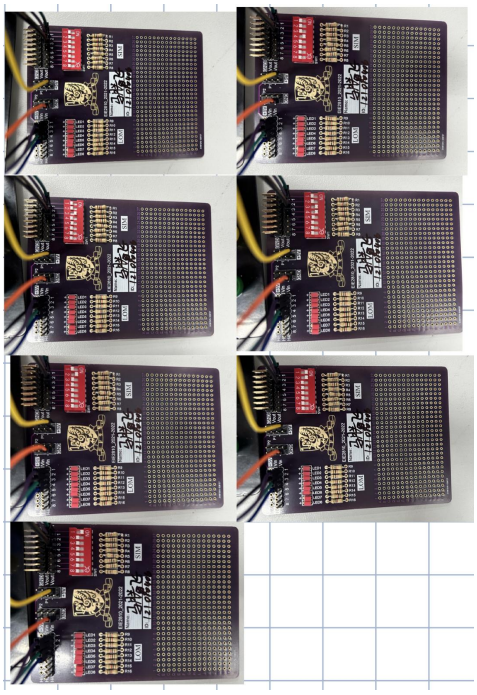

Figure 16 The Conditions That the LEDs are not Lightened

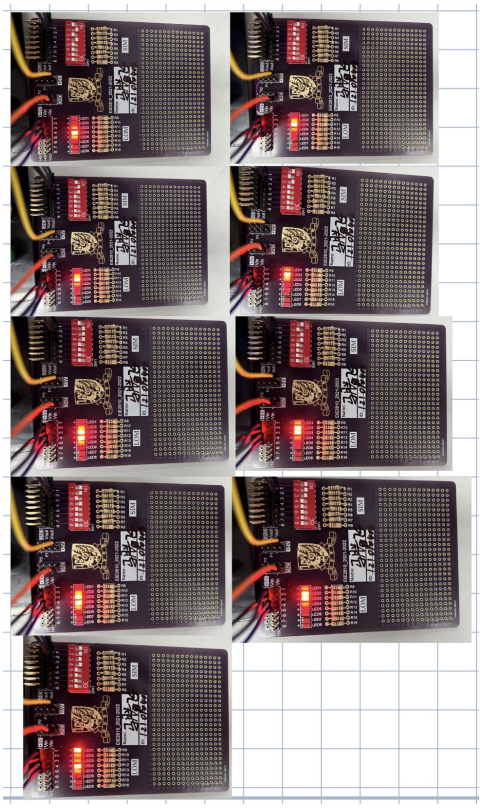

Figure 17 The Conditions That the LEDs are Lightened

As shown in Figure 16 and Figure 17, the output is 0001_2 (1_{10}) when the input is 01_2 and 01_2 (1_{10}), the output is 0010_2 (2_{10}) when one input is 01_2 (1_{10}) and another is 10_2 (2_{10}), the output is 0100_2 (4_{10}) when both inputs are 10_2 (2_{10}), the output is 0011_2 (3_{10})

when one input is 11_2 and another is 01_2 (1_{10}), the output is 0110_2 (6_{10}) when one input is 11_2 (3_{10}) and another is 10_2 (2_{10}), the output is 1001_2 (9_{10}) when both two inputs are 11_2 (3_2). and the output is 0000_2 (0_{10}) otherwise. All of the results go well with the truth table, which means we successfully constructed the multiplier with hardware.

4. Conclusion

In this lab, we learned how to construct a combinational logic circuit, observed timing hazard and eliminated it, and designed and constructed a multiplier. From the lab, we know:

1) The way to construct the combinational logic circuit.

2) The phenomenon of timing hazards and. The principle of timing hazard and the way to eliminate the timing hazard.

3) To construct a required circuit, we need to first find out its logic, simplify the logic and get the truth table, then design and construct the circuit according to the truth table.