EIE2810 Digital Systems Design Laboratory

Laboratory Report #2

Name: Student ID:
Date: 2024.3.8

The Chinese University of Hong Kong, Shenzhen

- Experiment A: learn to use Multisim simulation
- Experiment B: test logic gates (74HC00, 74HC02, 74HC32, 74HC86, 74HC7266)
- Experiment C: build other basic logic gates by NAND
- Experiment D: combine logic gates to form combined logic

1. Experiment A

1.1 Design

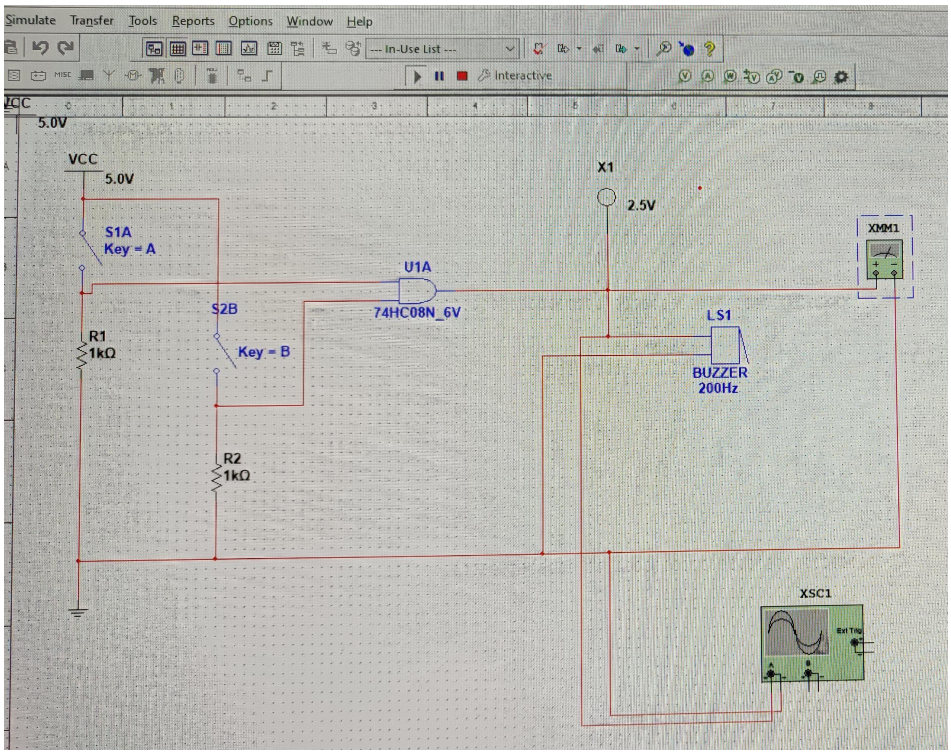


Figure1. circuit designed

1.2 Result

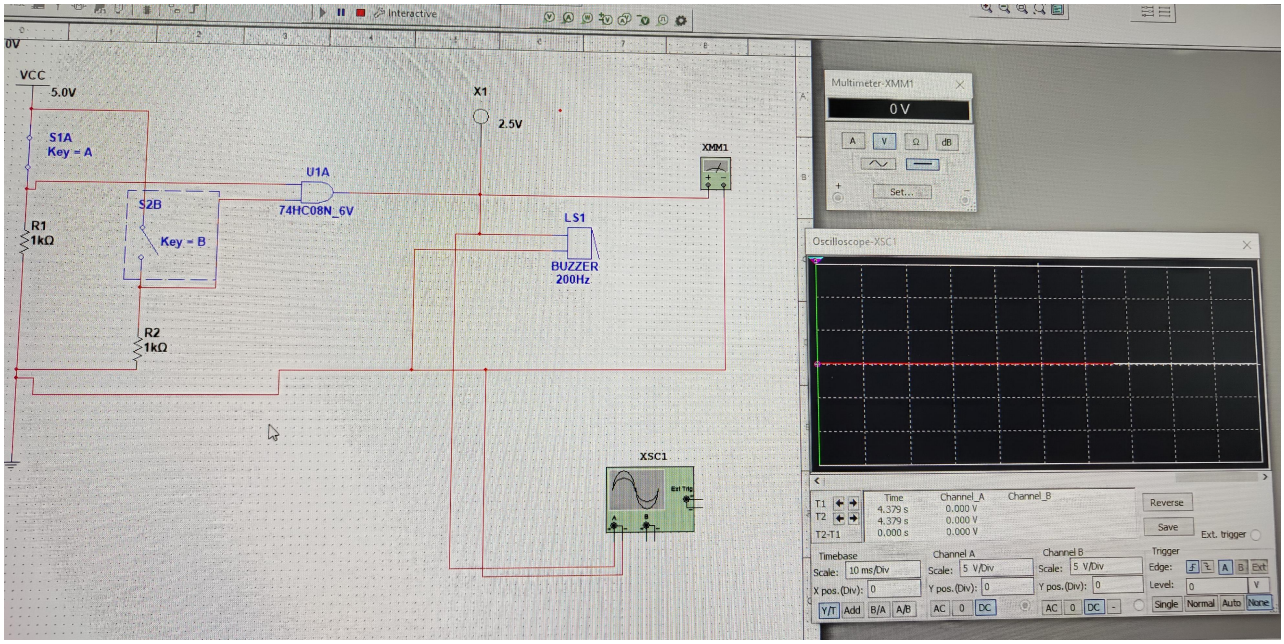


Figure2. test result when input is (1, 0)

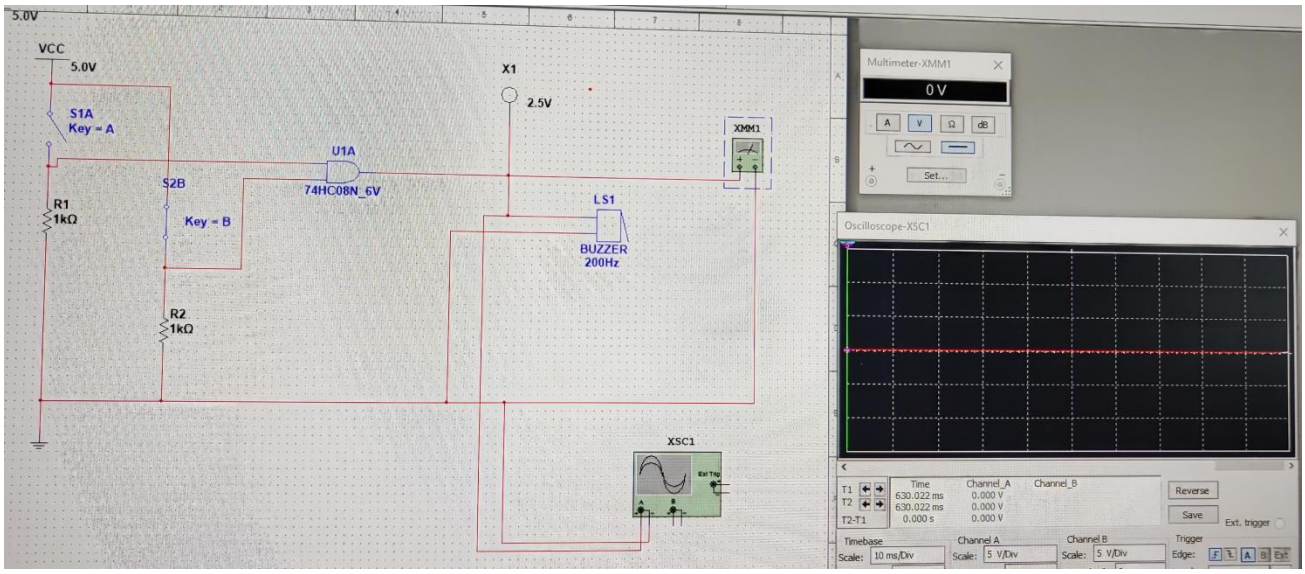


Figure3. test result when input is (0, 1)

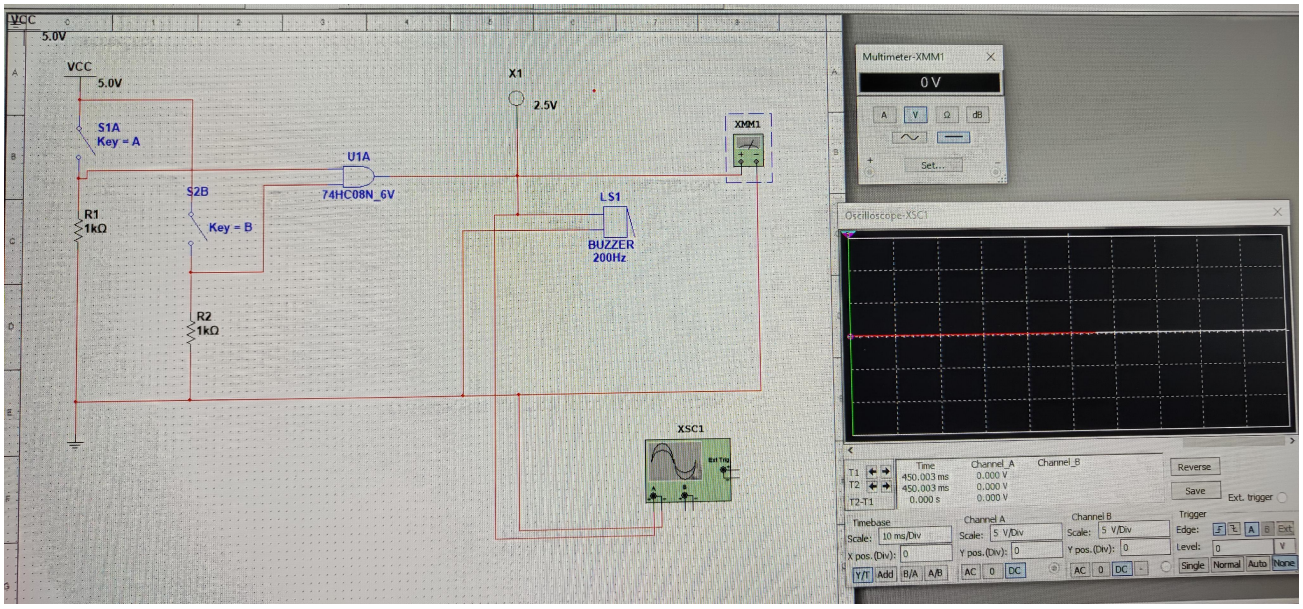


Figure4. test result when input is (0,0)

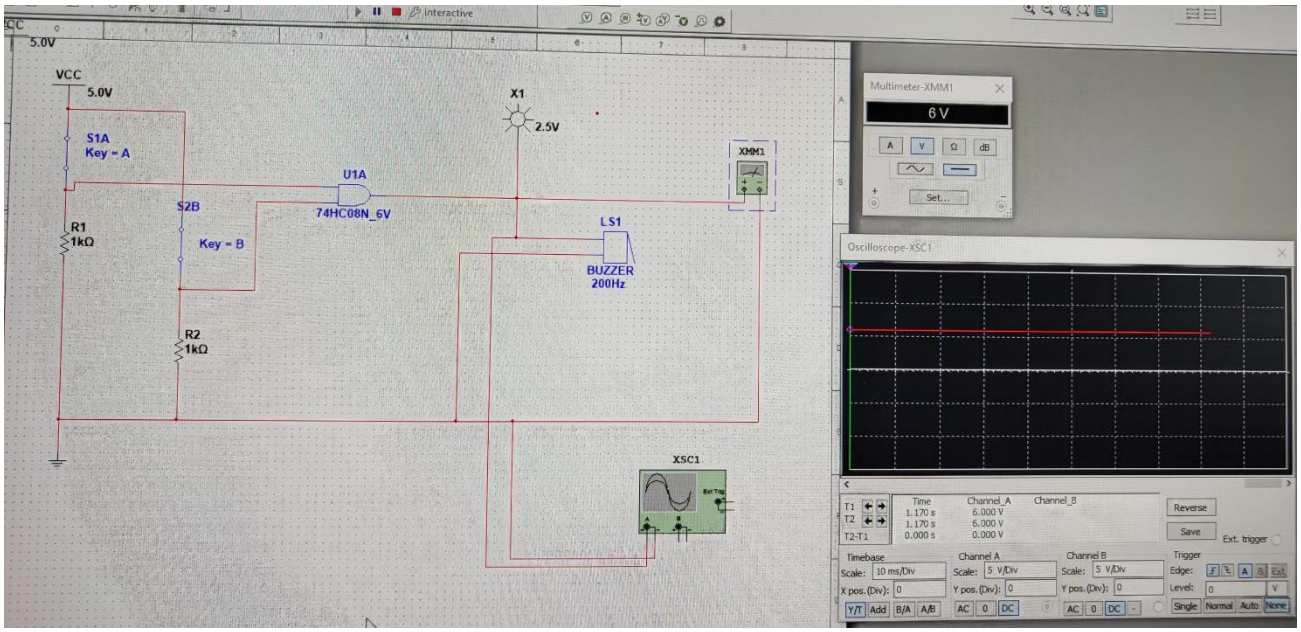


Figure5. test result when input is (1,1)

2. Experiment B

2.1 Result

- 74HC00

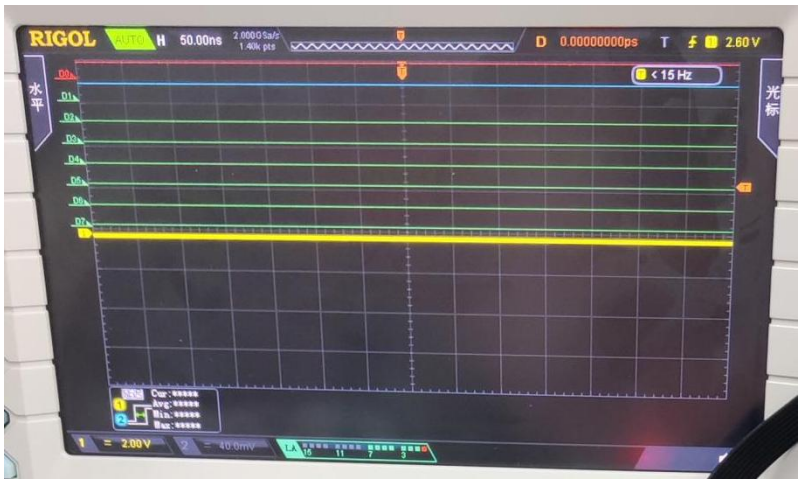


Figure6. wave image of 74HC00 when input is (1, 1)

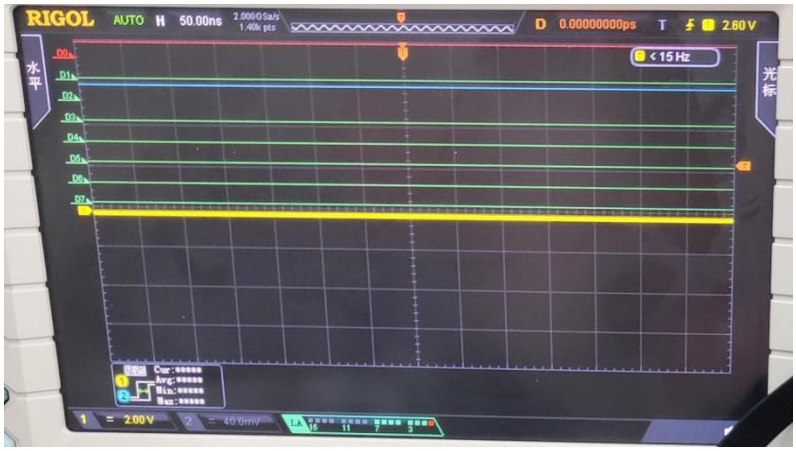


Figure7. wave image of 74HC00 when input is (1, 0)

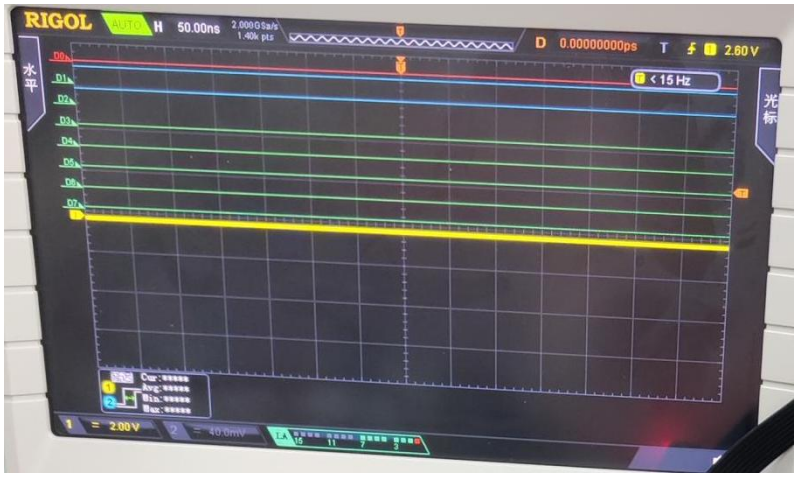


Figure8. wave image of 74HC00 when input is (0, 1)

Figure9. wave image of 74HC00 when input is (0, 0)

Analysis:
When both inputs are HIGH, the output is LOW.
When either input is LOW, the output is HIGH.
When both inputs are LOW, the output is HIGH.
In summary, this circuit implements the NAND logic operation. The output is LOW only when both inputs are HIGH; otherwise, the output is HIGH. This behavior is consistent with the truth table of an NAND gate.

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Table1. truth table for NAND gate

- 74HC02

Figure10. wave image of 74HC02 when input is (1, 1)

Figure11. wave image of 74HC02 when input is (0, 1)

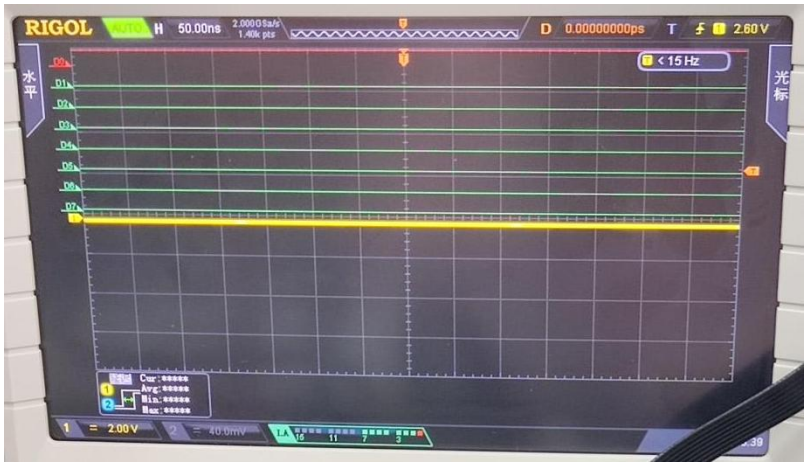


Figure12. wave image of 74HC02 when input is (1, 0)

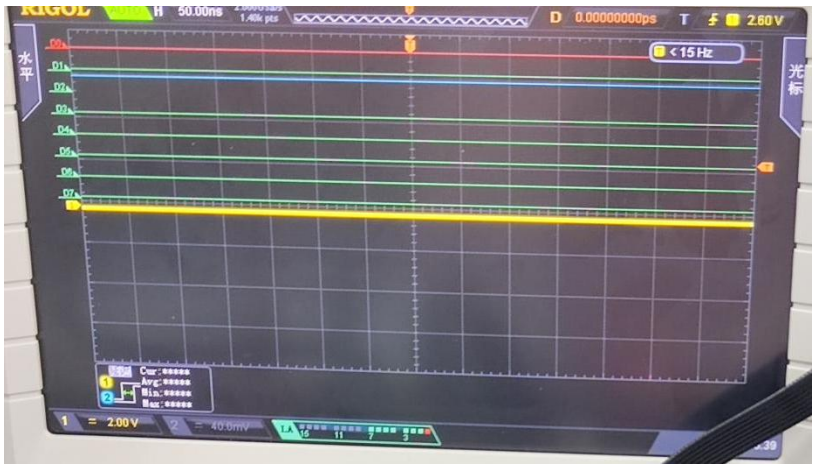


Figure13. wave image of 74HC02 when input is (0, 0)

Analysis:
When both inputs are HIGH, the output is LOW.
When either input is LOW, the output is LOW.
When both inputs are LOW, the output is HIGH.
In summary, this circuit implements the NOR logic operation. The output is HIGH only when both inputs are LOW; otherwise, the output is LOW. This behavior is consistent with the truth table of an NOR gate.

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Table2. truth table for NOR gate

- 74HC32

Figure14. wave image of 74HC32 when input is (1, 1)

Figure15. wave image of 74HC32 when input is (0, 1)

Figure16. wave image of 74HC32 when input is (1, 0)

Figure17. wave image of 74HC32 when input is (0, 0)

Analysis:
When both inputs are HIGH, the output is HIGH.
When either input is LOW, the output is HIGH.
When both inputs are LOW, the output is LOW.
In summary, this circuit implements the OR logic operation. The output is LOW only when both inputs are LOW; otherwise, the output is HIGH. This behavior is consistent with the truth table of an OR gate.

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Table3. truth table for OR gate

- 74HC86

Figure18. wave image of 74HC86 when input is (1, 1)

Figure19. wave image of 74HC86 when input is (0, 1)

Figure20. wave image of 74HC86 when input is (1, 0)

Figure21. wave image of 74HC86 when input is (0, 0)

Analysis:
When both inputs are HIGH, the output is LOW.
When either input is LOW, the output is HIGH.
When both inputs are LOW, the output is LOW.
In summary, this circuit implements the XOR logic operation. The output is HIGH when one of the inputs is LOW; otherwise, the output is LOW. This behavior is consistent with the truth table of an XOR gate.

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Table4. truth table for XOR gate

- 74HC7266

Figure22. image of 74HC7266 when input is (1, 1)

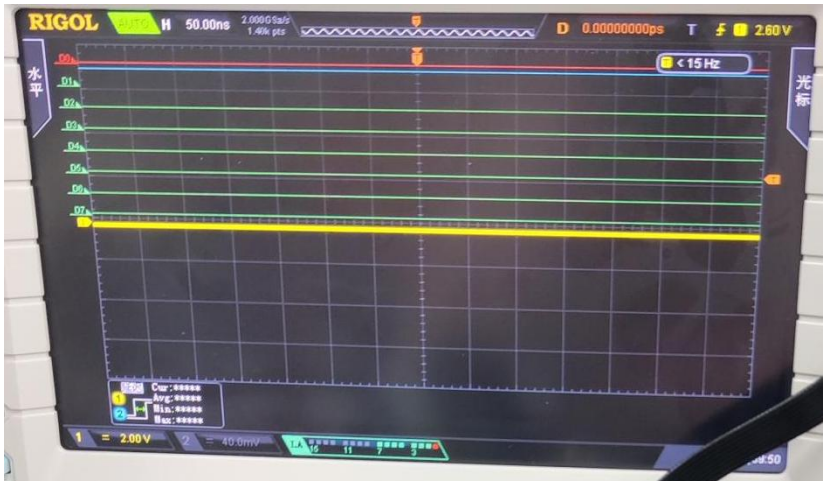


Figure23. image of 74HC7266 when input is (0, 1)

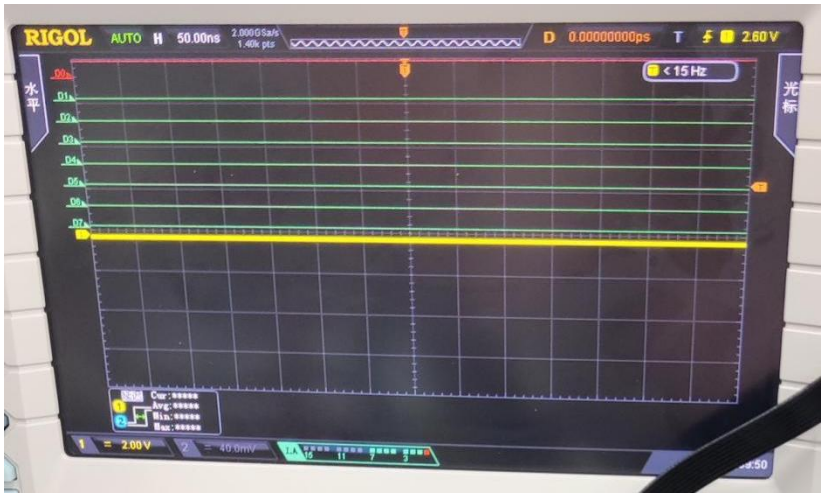


Figure24. image of 74HC7266 when input is (1, 0)

Figure25. image of 74HC7266 when input is (0, 0)

Analysis:
When both inputs are HIGH, the output is HIGH.
When either input is LOW, the output is LOW.
When both inputs are LOW, the output is HIGH.
In summary, this circuit implements the XNOR logic operation. The output is LOW when one of the inputs is LOW; otherwise, the output is HIGH. This behavior is consistent with the truth table of an XNOR gate.

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Table5. truth table for XNOR gate

3. Experiment C

3.1 Design

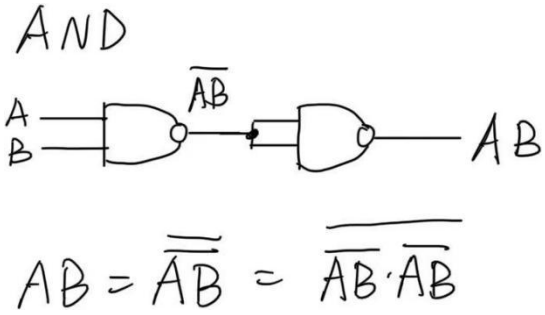


Figure26. circuit diagram and derification process of AND

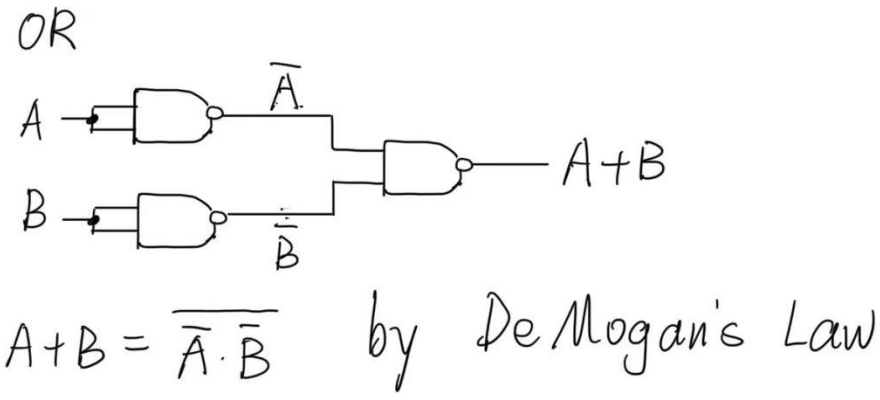
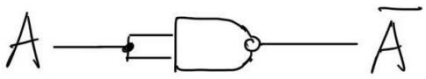


Figure27. circuit diagram and derification process of OR

NOT

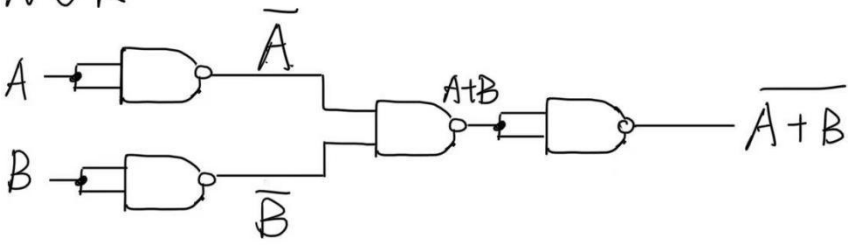


$\because A = A \cdot A$

$\therefore \overline{A} = \overline{A \cdot A}$

Figure28. circuit diagram and derivation process of NOT

NOR



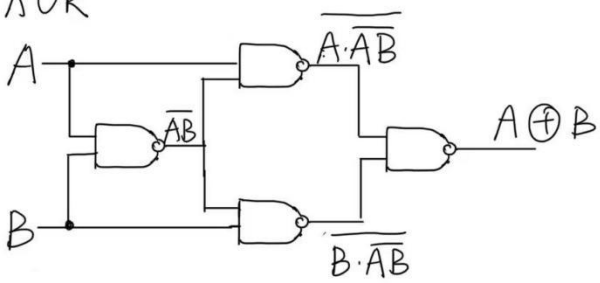
$\because A+B = \overline{\overline{A} \cdot \overline{B}}$

$\therefore \overline{A+B} = \overline{\overline{A} \cdot \overline{B}}$

only need to add 1 NAND gate to the designed OR gate

Figure29. circuit diagram and derivation process of NOR

XOR



$$\overline{\overline{A\overline{B}} \cdot A} \cdot \overline{\overline{A\overline{B}} \cdot B} = \overline{A\overline{B}} \cdot A + \overline{A\overline{B}} \cdot B \quad \text{by De Morgan's Law}$$

$$= \overline{A\overline{B}}(A+B)$$

$$= (\overline{A}+\overline{B})(A+B) \quad \text{De Morgan's Law}$$

$$= \overline{A}B + \overline{B}A \quad \because \overline{A}A = 0.$$

$$\therefore A \oplus B = A\overline{B} + \overline{A}B = \overline{\overline{A\overline{B}} \cdot A} \cdot \overline{\overline{A\overline{B}} \cdot B}$$

Figure30. circuit diagram and derivation process of XOR

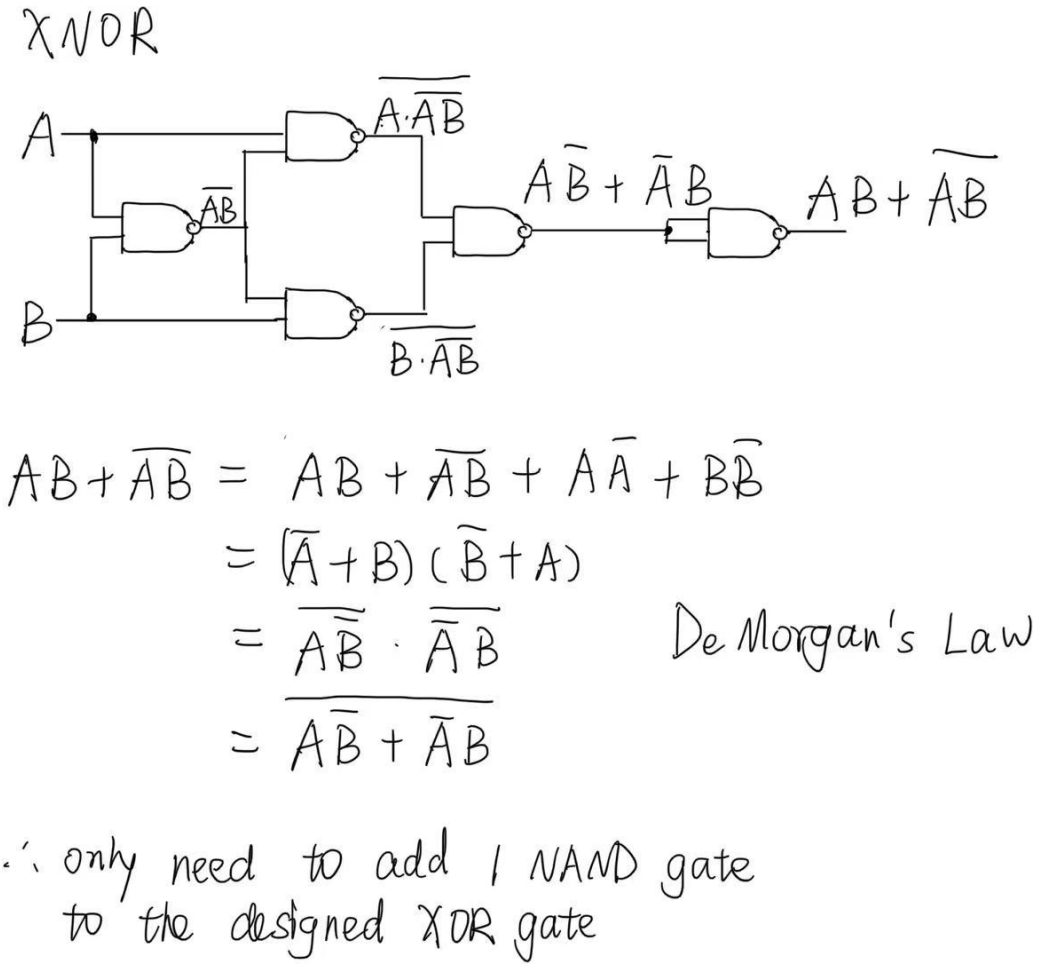


Figure31. circuit diagram and derivation process of XNOR

4. Experiment D

4.1 Design

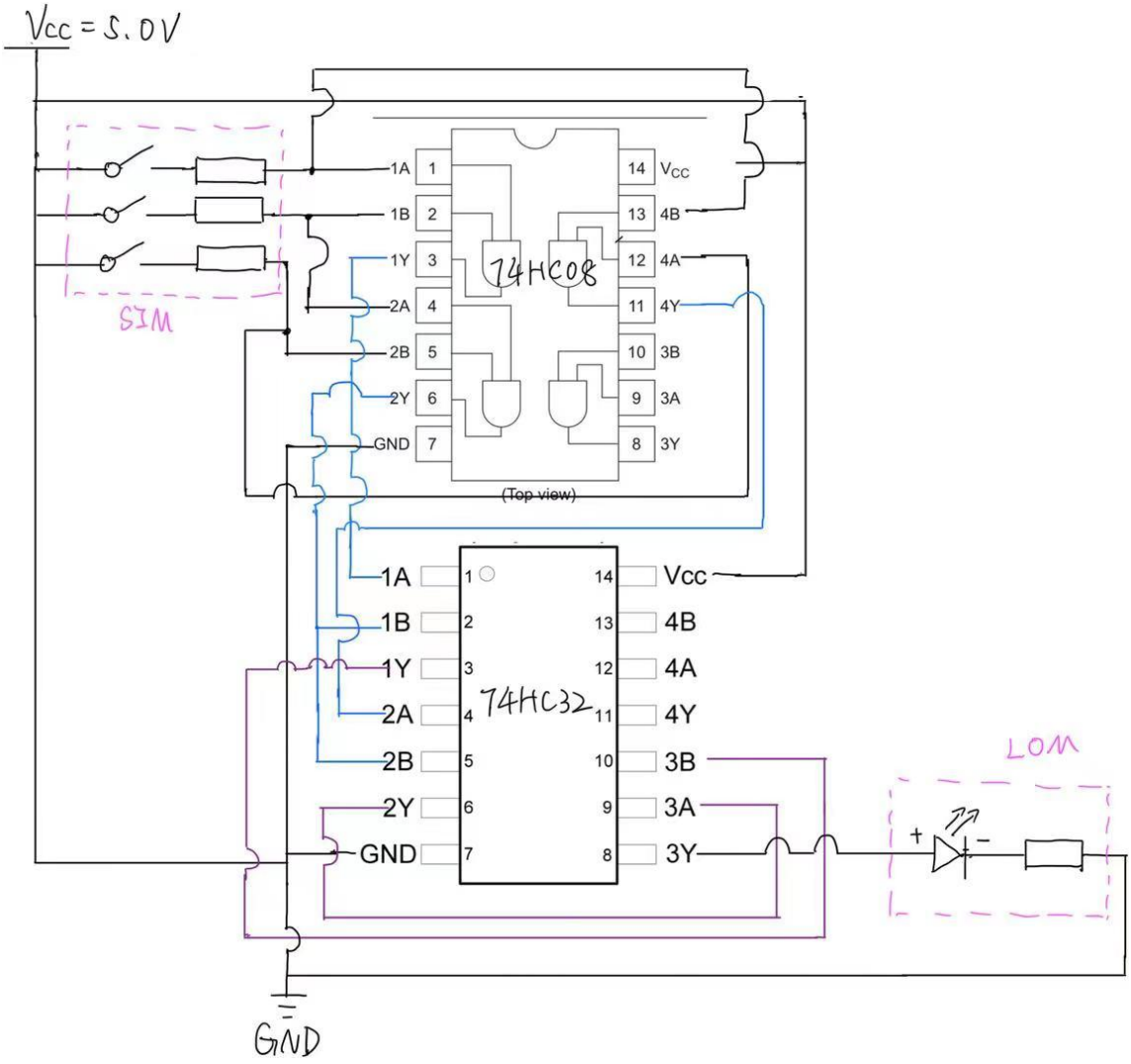


Figure32. circuit diagram

4.2 Result

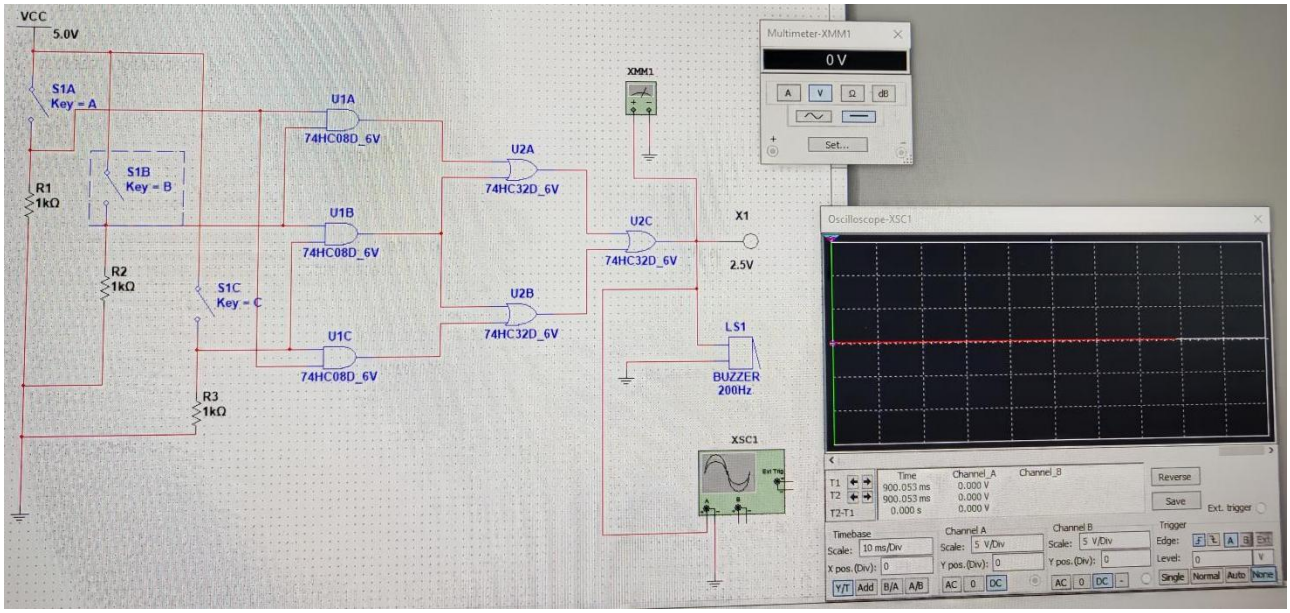


Figure33. diagram built by Multisim with input (0, 0, 0), output 0

Figure34. wave image when input is (0, 0, 0), output is 0

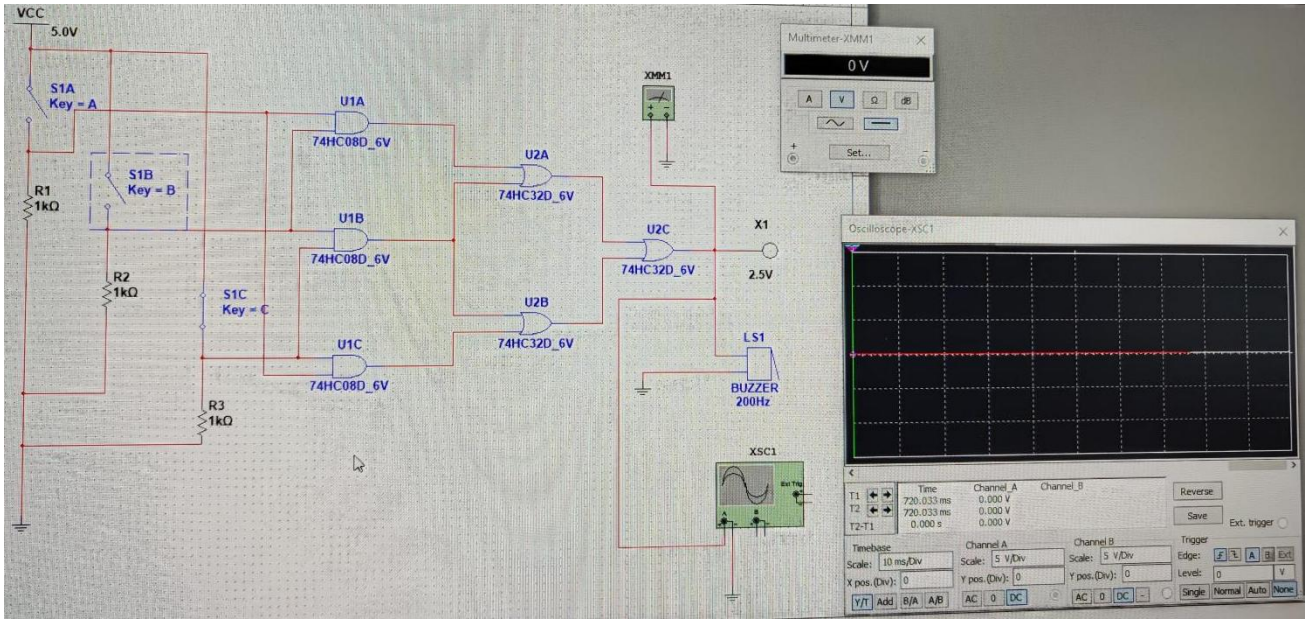


Figure35. diagram built by Multisim with input (0, 0, 1), output 0

Figure36. wave image when input is (0, 0, 1), output is 0

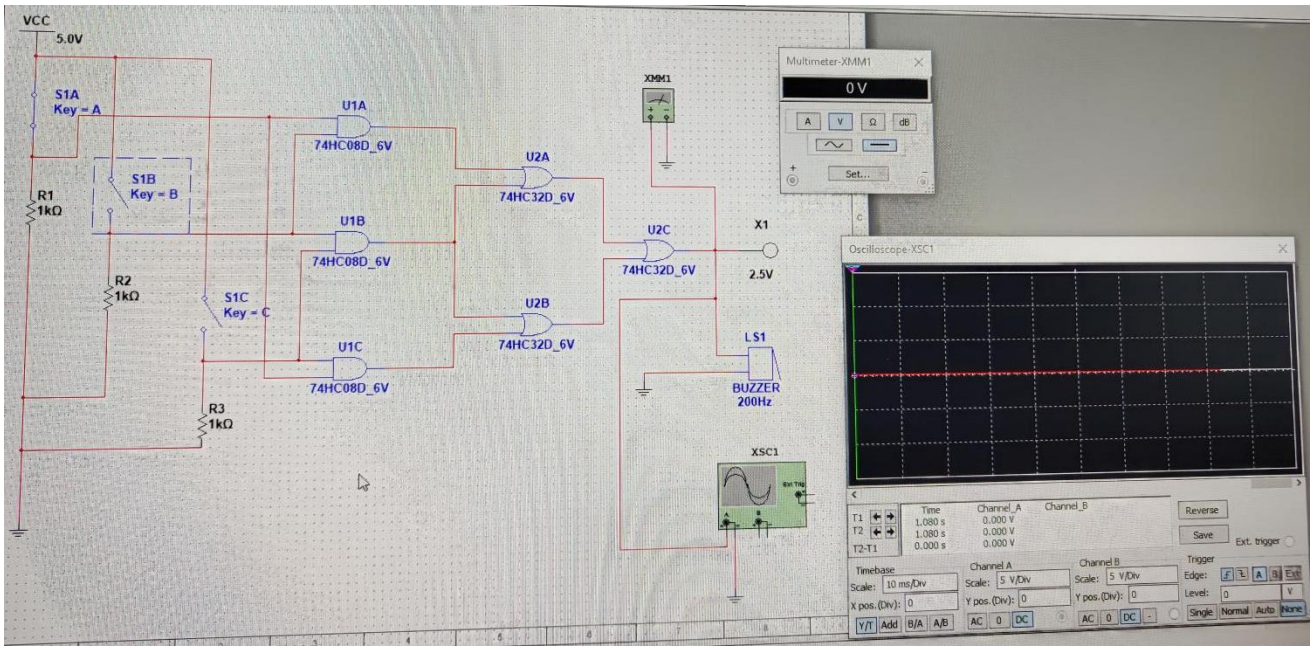


Figure37. diagram built by Multisim with input (1, 0, 0), output 0

Figure38. wave image when input is (1, 0, 0), output is 0

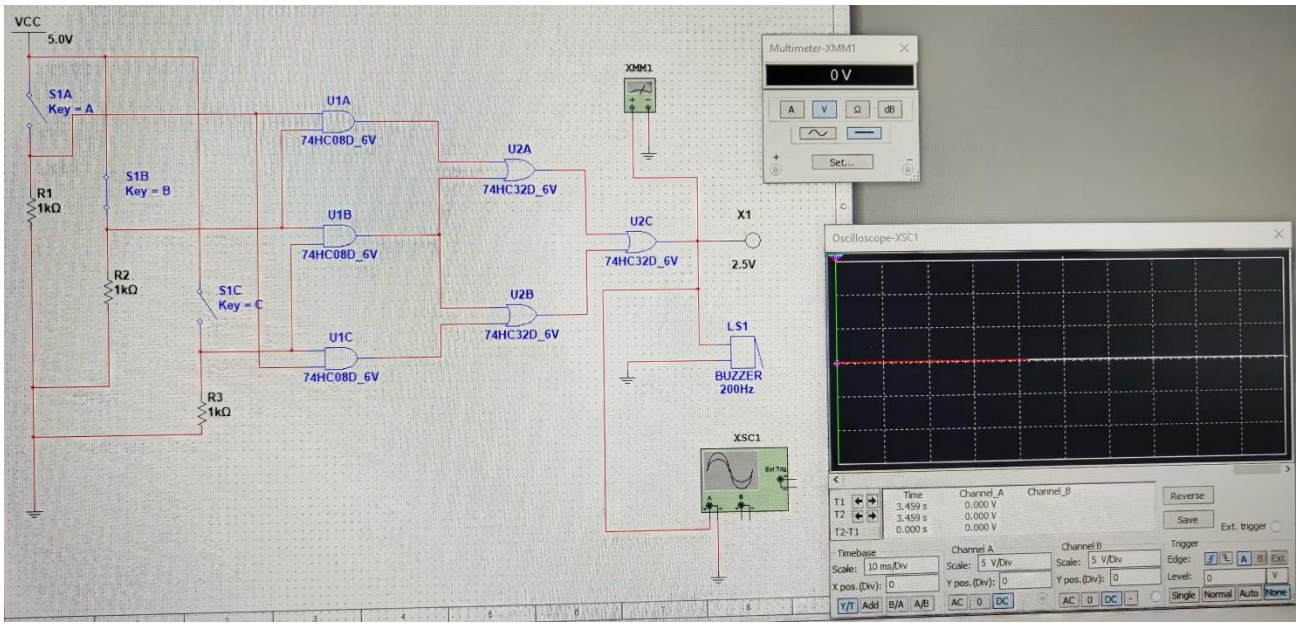


Figure39. diagram built by Multisim with input (0, 1, 0), output 0

Figure40. wave image when input is (0, 1, 0), output is 0

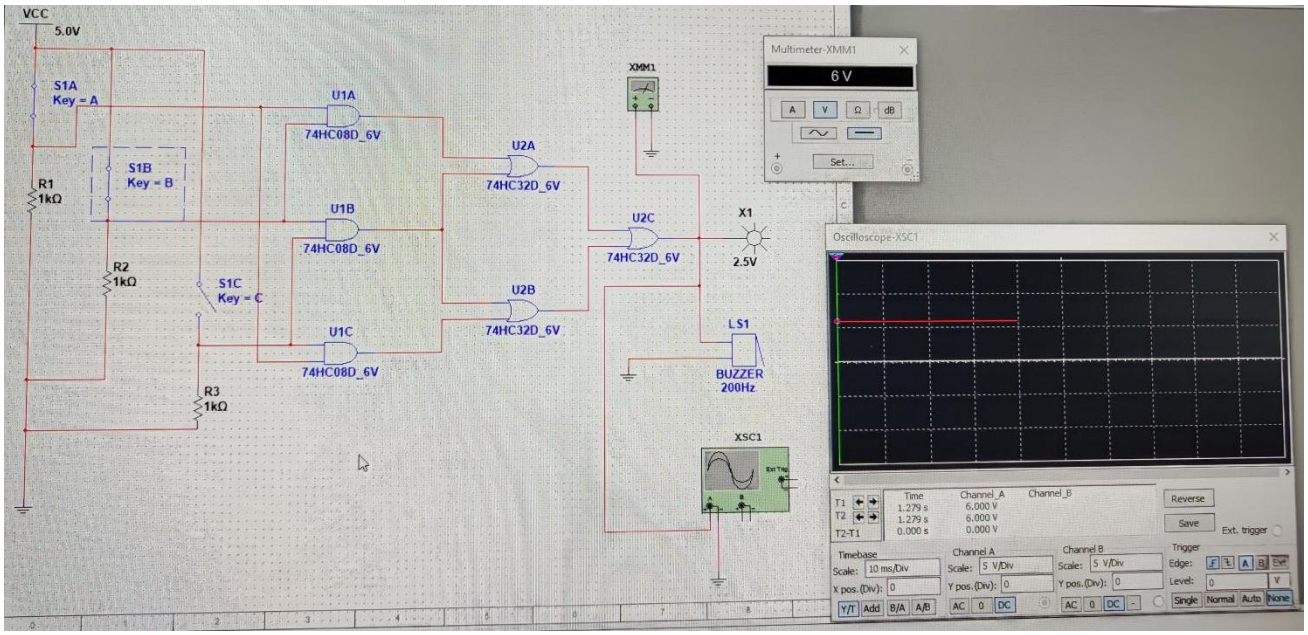


Figure41. diagram built by Multisim with input (1, 1, 0), output 1

Figure42. wave image when input is (1, 1, 0), output is 1

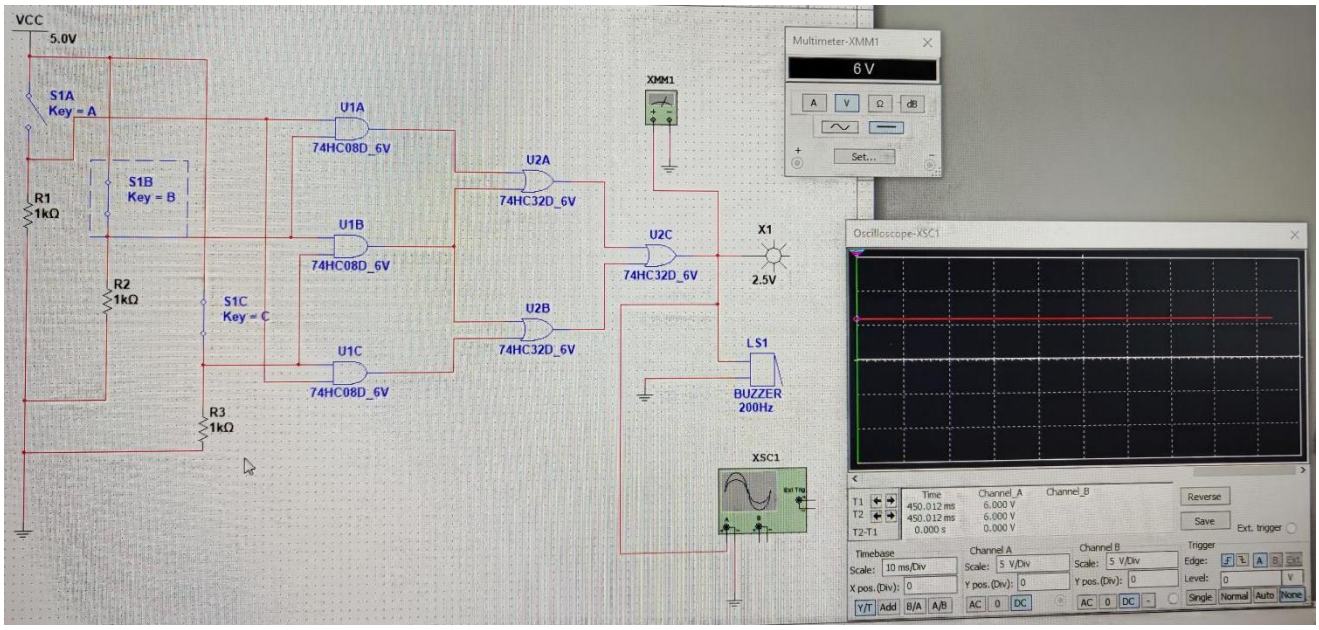


Figure43. diagram built by Multisim with input (0, 1, 1), output 1

Figure44. wave image when input is (0, 1, 1), output is 1

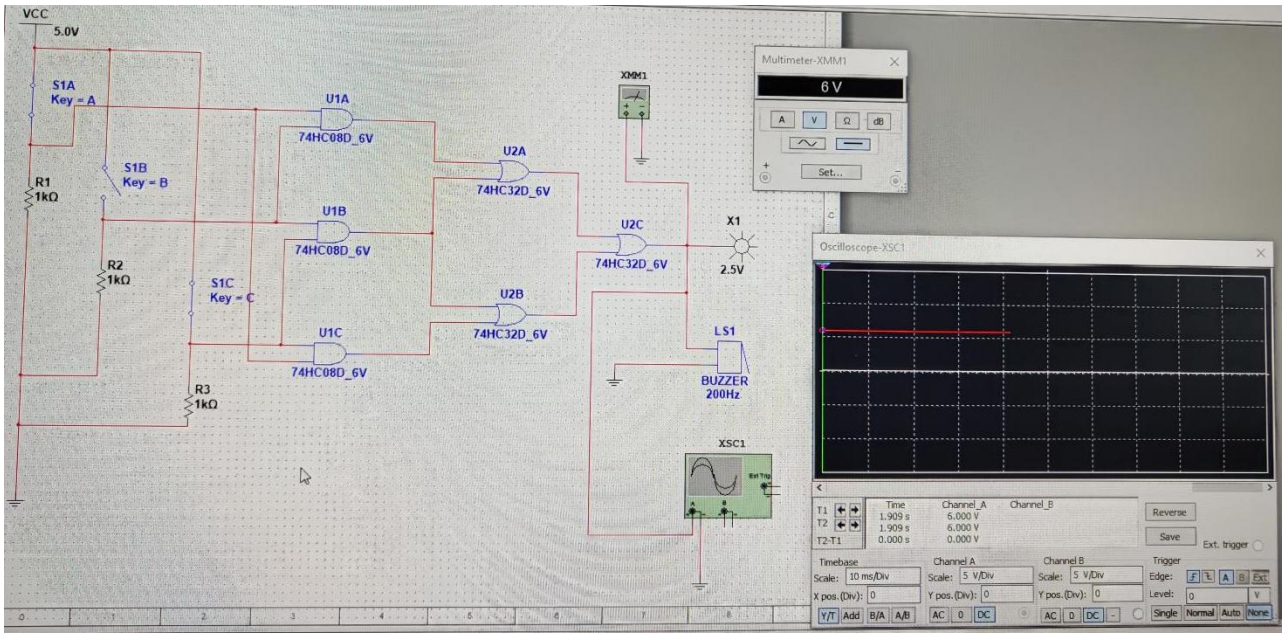


Figure45. diagram built by Multisim with input (1, 0, 1), output 1

Figure46. wave image when input is (1, 0, 1), output is 1

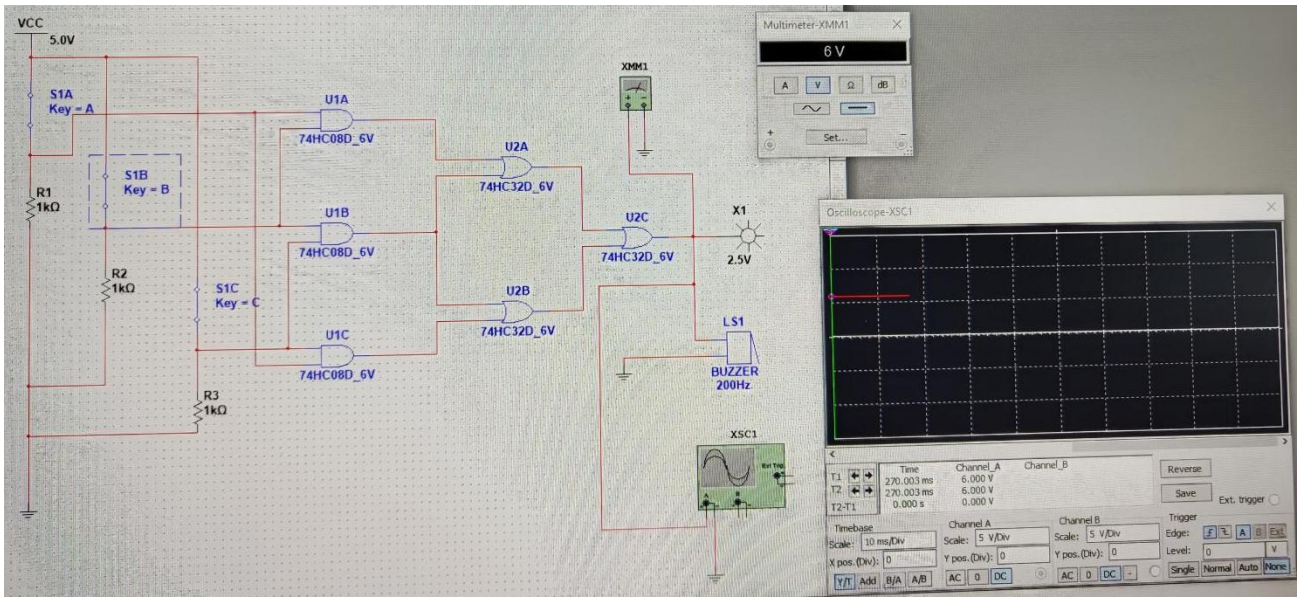


Figure47. diagram built by Multisim with input (1, 1, 1), output 1

Figure48. wave image when input is (1, 1, 1), output is 1

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

0	0	0	0

Table6. corresponding truth table

Analysis:

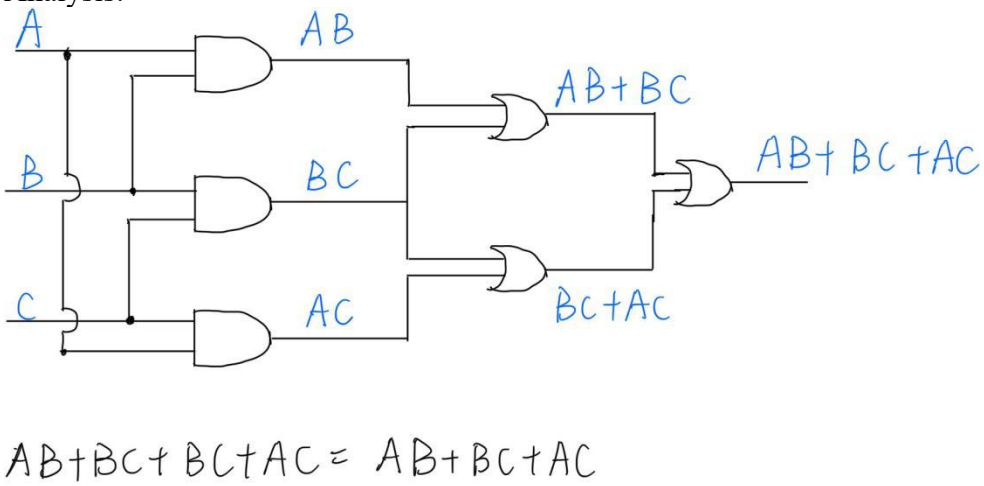


Figure49. corresponding circuit diagram

As shown in the figure above, the output is AB+AC+BC.
Both the logic analyzer waveform and the output of the circuit built with Multisim show that:
When all inputs are HIGH, the output is HIGH.
When only one input is LOW, the output is HIGH.
When only one input is HIGH, the output is LOW.
When all inputs are LOW, the output is LOW.
These results are summarized in the truth table above, which is consistent with the logic expression AB+AC+BC.

4.3 Question

Q.Can you think of any application for this circuit?
A.1. Fault detection: In a system with three components A, B, and C, this circuit could be used to detect if at least two out of the three components are functioning correctly. The output will be HIGH if at least two inputs are HIGH, indicating normal operation. For example, a level sensor in each of three tanks produces a HIGH voltage when the level of water in the tank drops below a specific point. The circuit outputs HIGH when at least two of the tanks drops below the point.
2. Voting system: In a simple voting system (three voters A, B, C) where a decision is made based on the majority, this circuit can be used to determine the majority vote. If at least two out of three voters agree, the output will be HIGH, indicating the majority decision.
3. Security system: In systems where redundancy is used for reliability, this circuit can be used to check if at least two out of three redundant systems are providing the same output. This is often used in safety-critical systems like aircraft control systems.

5. Conclusion

In this lab, I

- Explored basic logic gates (AND, NOT, OR, NAND, NOR, XOR, XNOR)
- Built simple circuits by Multisim
- Understand and test logic gates (74HC00, 74HC02, 74HC32, 74HC86, 74HC7266)
- Design and built other basic logic gates using NAND
- Combined logic gates to form combined logic

I learned:

- How to use Multisim simulation software.
- Use only NAND to design AND, OR, NOT, NOR, XOR, XNOR gates, by theoretically deriving the logic expression. For example, if the two inputs of the NAND gate is same, then it acts just as a NOR gate because of the property:

  A=A*A; Because $A\bar{B} + \overline{A\bar{B}} = A\bar{B} + \bar{A}B$, we can design XNOR gate by simply adding one NAND gate to the designed XOR gate.
- The application of AND-OR logic: check if at least two components outputs “1”.