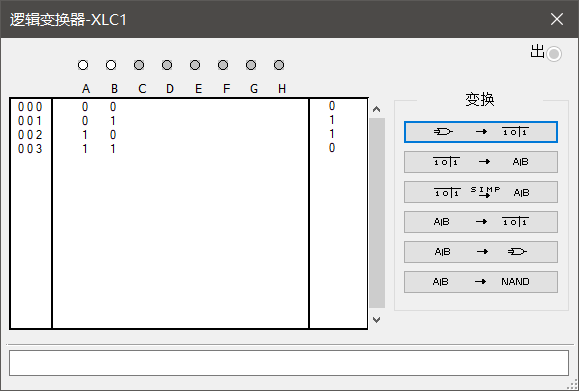
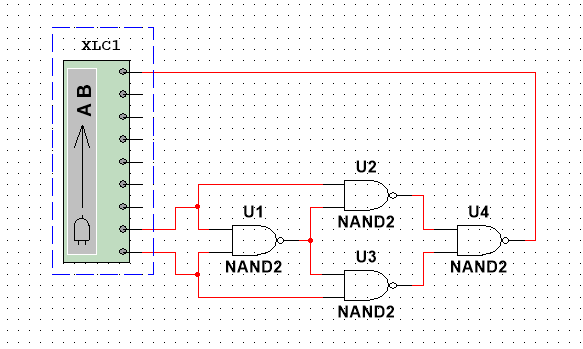
**1. CMOS logic gate**

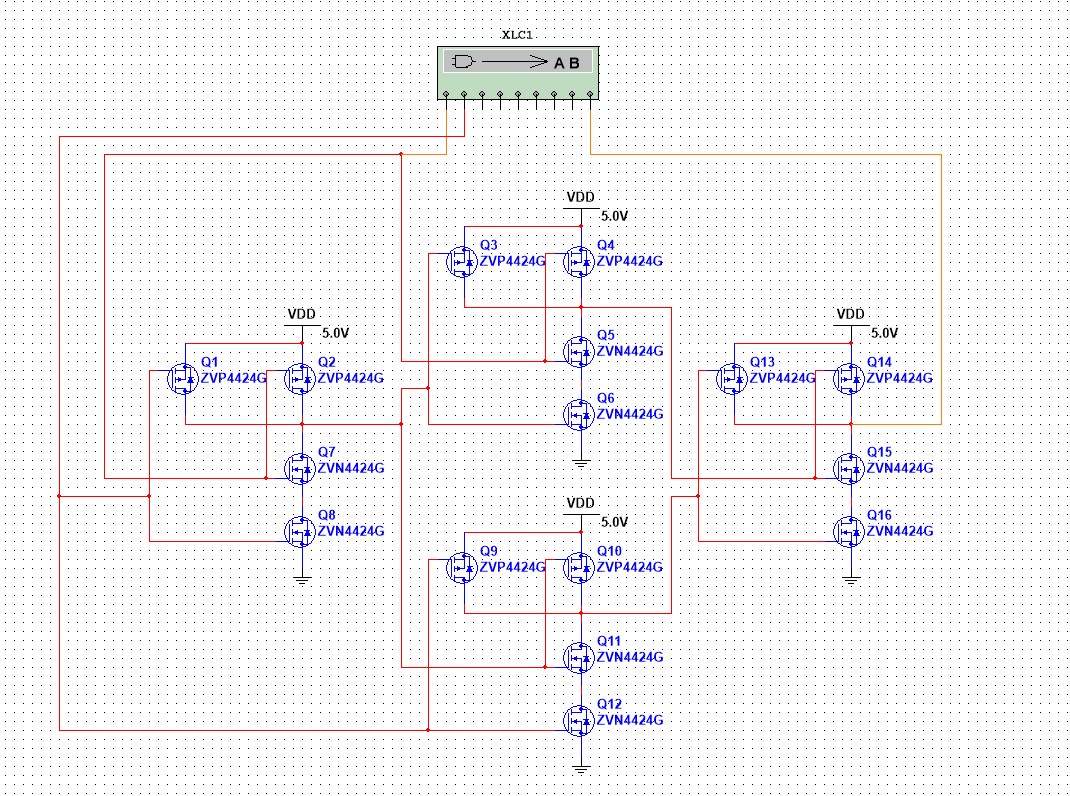
**An XOR gate can be built using NAND gate. The circuit diagram is shown below. Try building a XOR in multisim with NAND gate.**

**- Draw the truth table. Implement the circuit in multisim, use logic converter to generate its truth table, compare the simulation results with your answer**

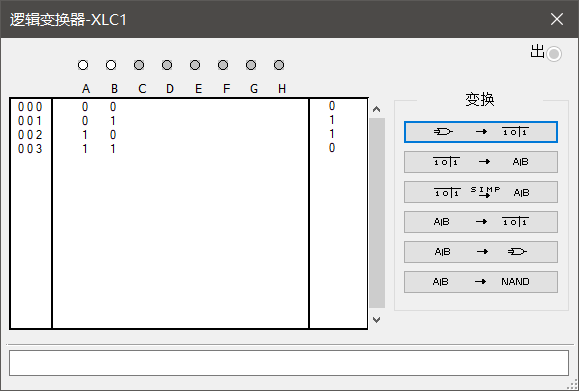
|  |  |  |
| --- | --- | --- |
| A | B | Output |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

****

**- Implement your design in Multisim, it should be CMOS-level implementation. If you don’t know which MOSFET to use, you can use ZVN4424 and ZVP4424**



**- Use logic converter to generate the truth table of your design.**



**2. Combinational logic circuit exercises**

**- For each one of the truth tables, write down the Boolean equations in canonical sum of products form.**

（a）+A+AB

（b）+ABC

（c）+B+A+AC+ABC

（d）+D+C+CD+A+AC+ABC

（e）+CD+BD+BC+AD+AC+AB+ABCD

**- Simplify the logic using the K-map**

（a）+A+AB

|  |  |  |  |
| --- | --- | --- | --- |
|  | B | 1 | 0 |
| A |  | B |  |
| 1 | A | 1 | 1 |
| 0 |  | 0 | 1 |

+A

（b）+ABC

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | BC | 00 | 01 | 11 | 10 |
| A |  |  | C | BC | B |
| 0 |  | 1 | 0 | 0 | 0 |
| 1 | A | 0 | 0 | 1 | 0 |

+ABC

（c）+B+A+AC+ABC

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | BC | 00 | 01 | 11 | 10 |
| A |  |  | C | BC | B |
| 0 |  | 1 | 0 | 0 | 1 |
| 1 | A | 1 | 1 | 1 | 0 |

+AC+B

（d）+D+C+CD+A+AC+ABC

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | CD | 00 | 01 | 11 | 10 |
| AB |  |  | D | CD | C |
| 00 |  | 1 | 1 | 1 | 1 |
| 01 | B | 0 | 0 | 0 | 0 |
| 11 | AB | 0 | 0 | 0 | 1 |
| 10 | A | 1 | 0 | 0 | 1 |

+ A+ABC

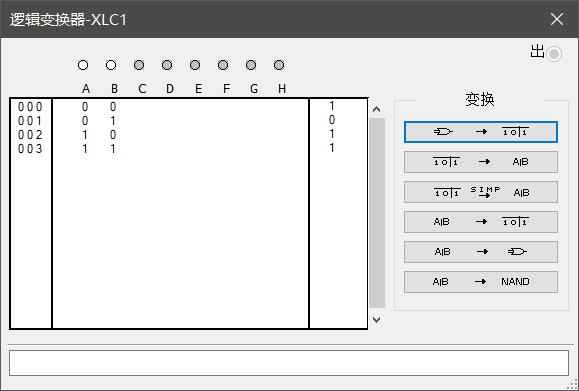
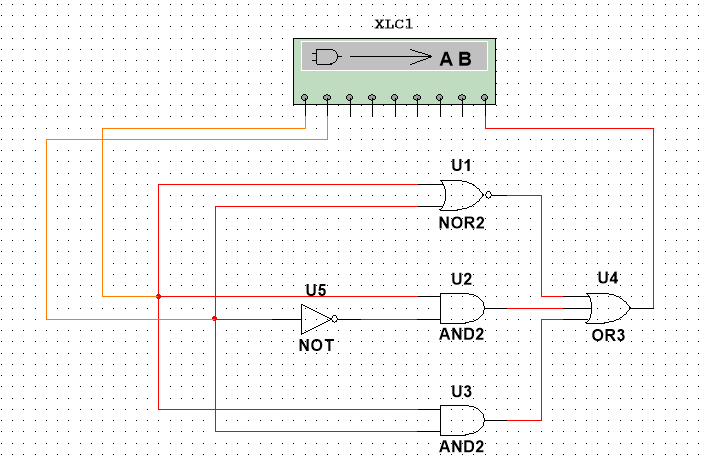
（e）+CD+BD+BC+AD+AC+AB+ABCD

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | CD | 00 | 01 | 11 | 10 |
| AB |  |  | D | CD | C |
| 00 |  | 1 | 0 | 1 | 0 |
| 01 | B | 0 | 1 | 0 | 1 |
| 11 | AB | 1 | 0 | 1 | 0 |
| 10 | A | 0 | 1 | 0 | 1 |

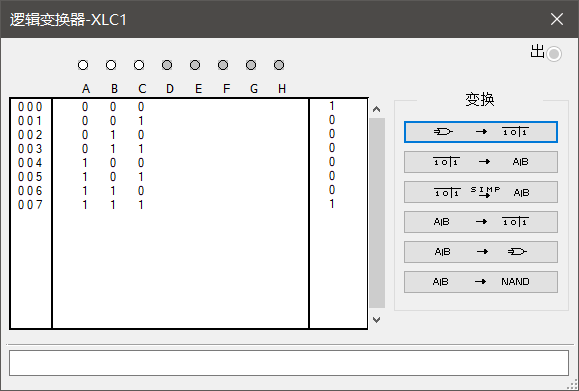
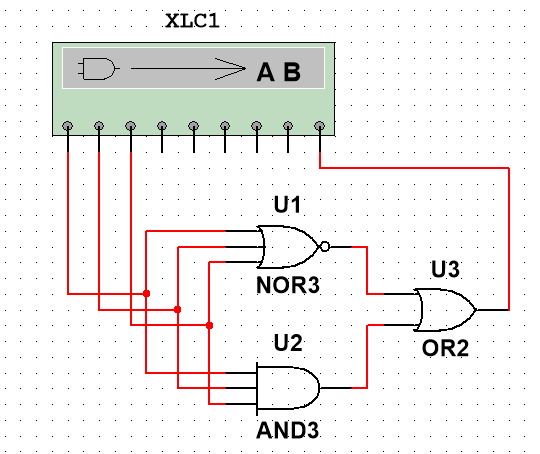
+CD+BD+BC+AD+AC+AB+ABCD

**- Implement the circuits in Multisim using logic gates. Use logic converter to generate their truth tables.**

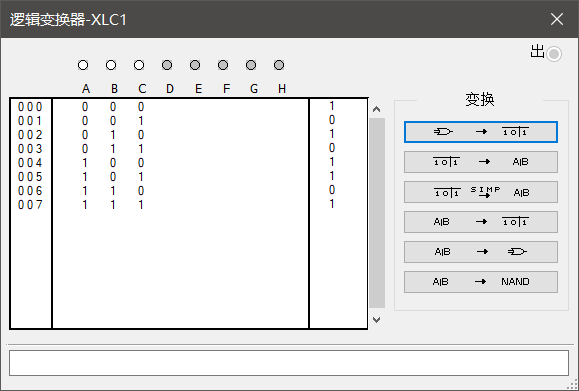
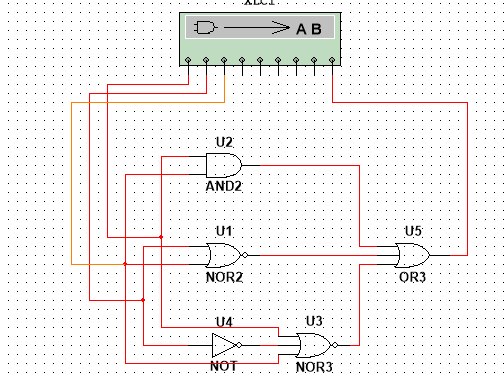
（a）



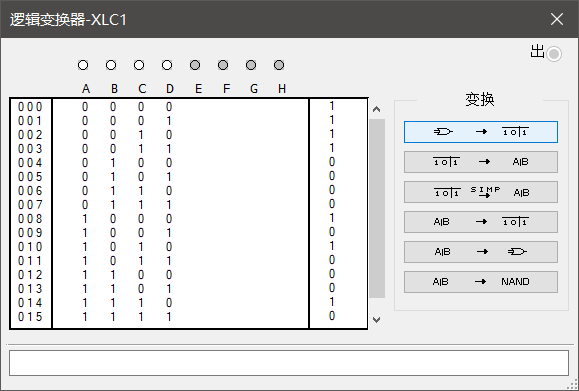
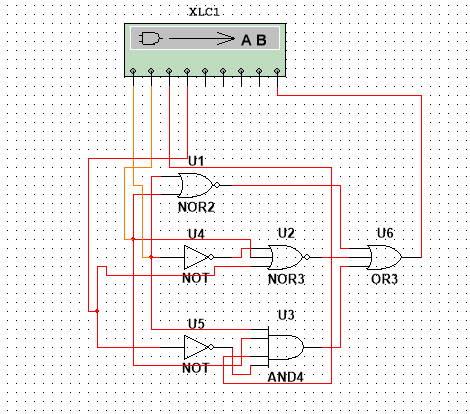
（b）



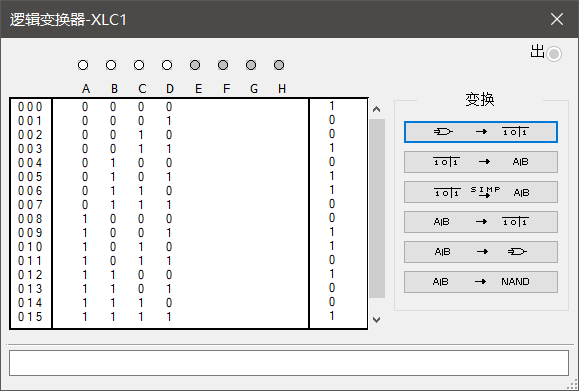
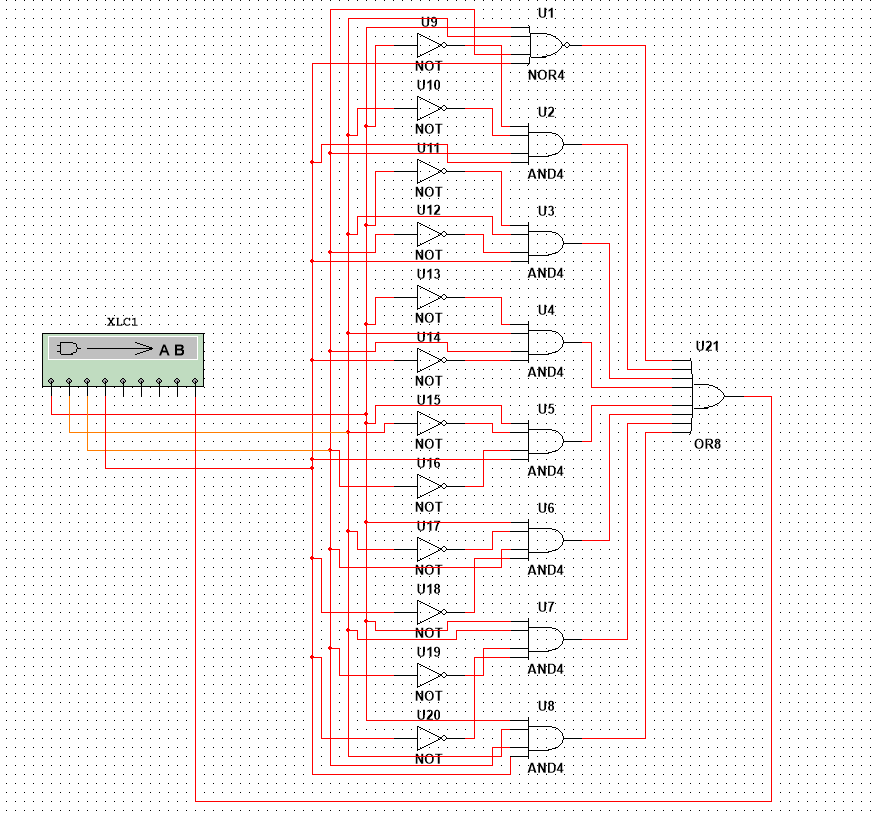
（c）



（d）



（e）



**3. Majority voting and 14-segment display**

**Referring to the majority voting system introduced in the lecture, design a logic circuit with Multisim and show the voting result with a 14-segment display as follows (T for Trump and b for Biden)**

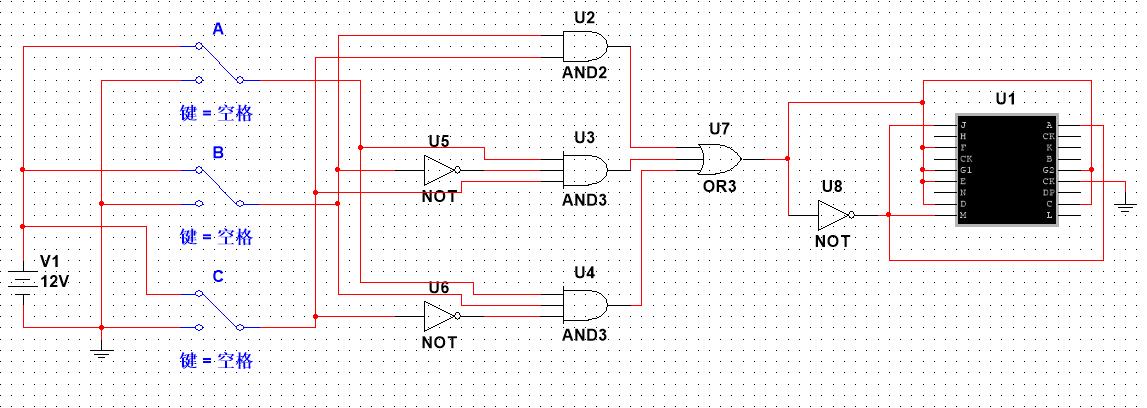
**- Show the design procedures(Truth table, Boolean equations, etc) and final circuit schematic.**

1 for Biden

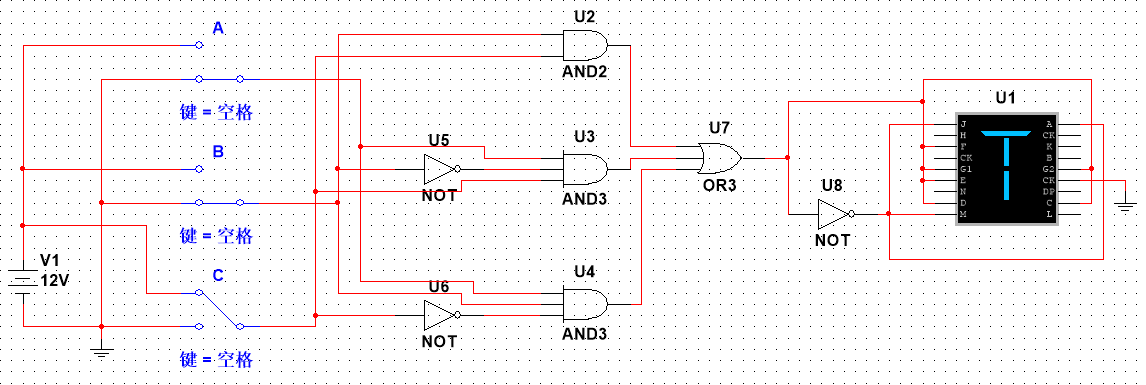
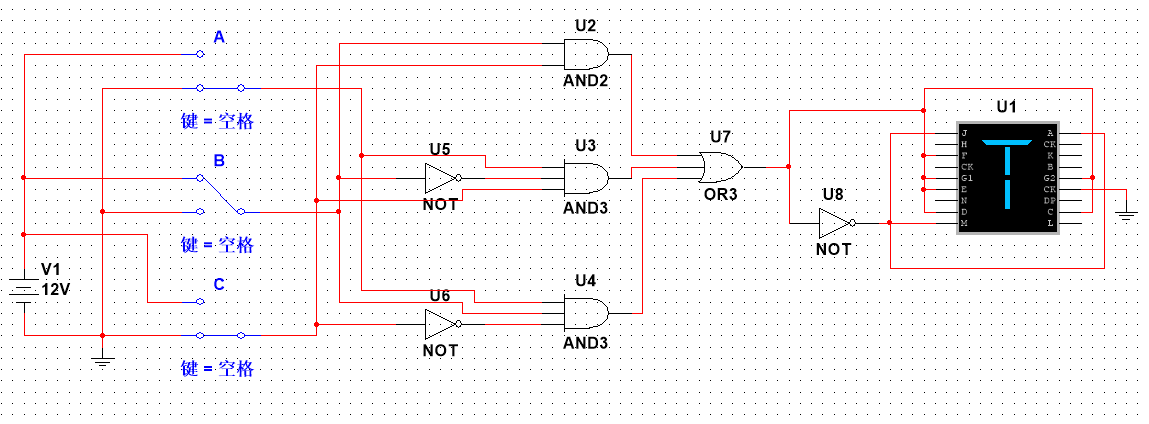
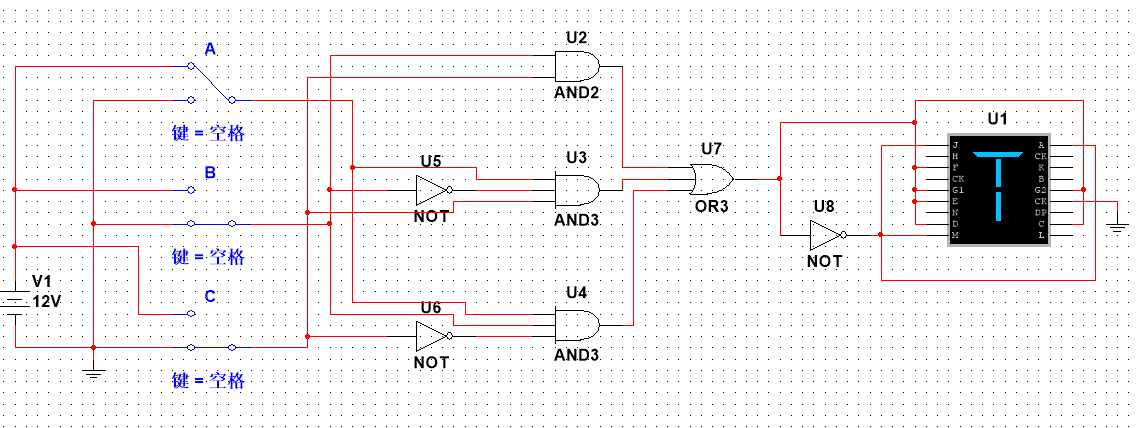
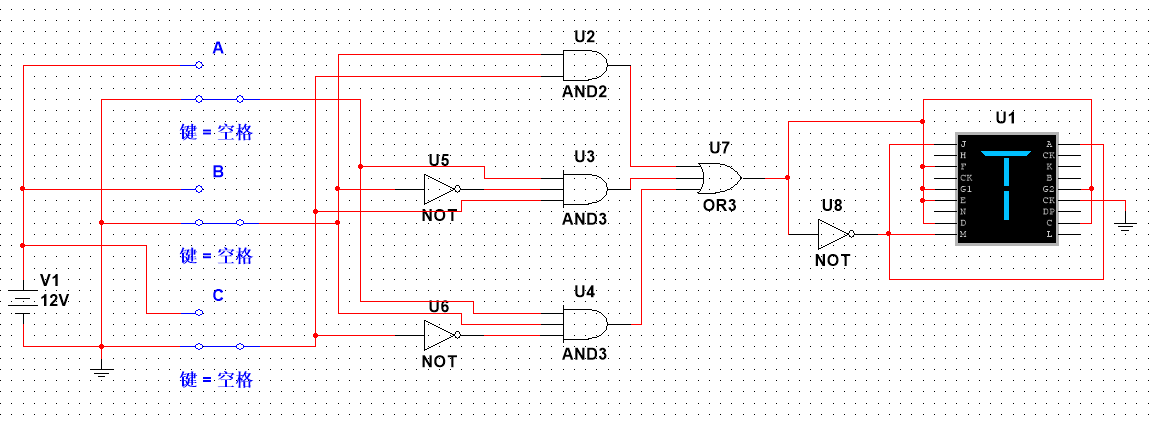
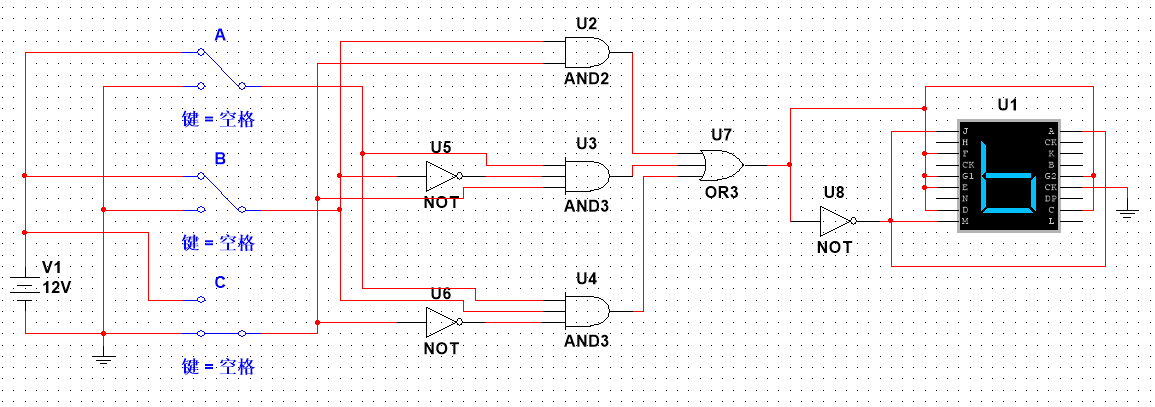
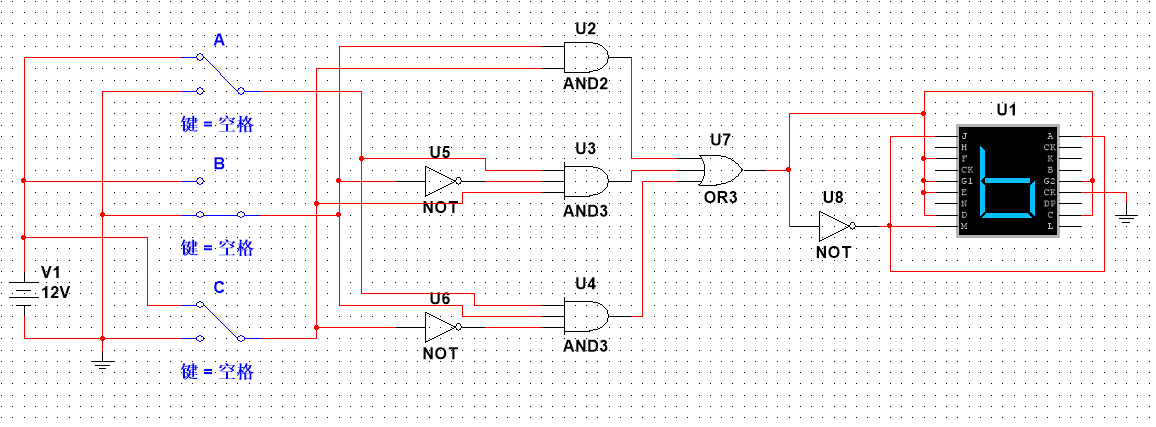
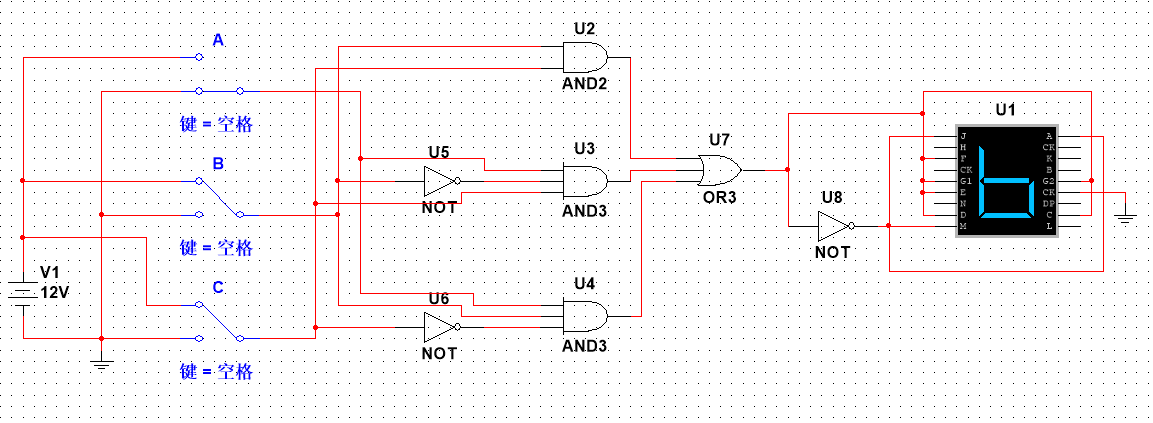
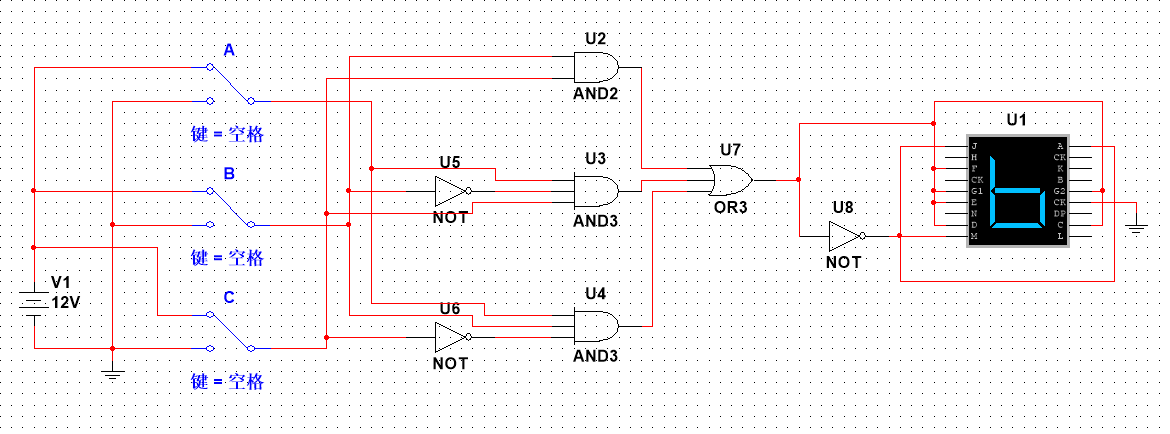
0 for Trump

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | BC | 00 | 01 | 11 | 10 |
| A |  |  | C | BC | B |
| 0 |  | 0 | 0 | 1 | 0 |
| 1 | A | 0 | 1 | 1 | 1 |

BC+AC+AB



**- Show the result pictures of eight input conditions.**

****

**4. MCU development**

**- Make your esp32’s onboard LED blink with 25Hz frequency for one second, then 5Hz for one second. Repeat this cycle (25Hz 5Hz 25Hz 5Hz 25Hz……) for 100 times, then turn off the LED**

|  |
| --- |
| import machine  import time  pin2 = machine.Pin(2 , machine.Pin.OUT)  for i in range(100):  for i in range(25):  pin2.value(1)  time.sleep(0.02)  pin2.value(0)  time.sleep(0.02)  for i in range(10):  pin2.value(1)  time.sleep(0.1)  pin2.value(0)  time.sleep(0.02)  pin2.value(0) |