**Russian Roulette Project**

19011502 최락원

Digital System(006)

* **Description of the project**

1. Press the reset button.
2. In the shift register, change the bit to 1 as many as you want, except for the last bit.

I recommend changing 3 bits random. (1 means bullet, 0 means empty.)

1. Then press the trigger button until one player's LED is lit.

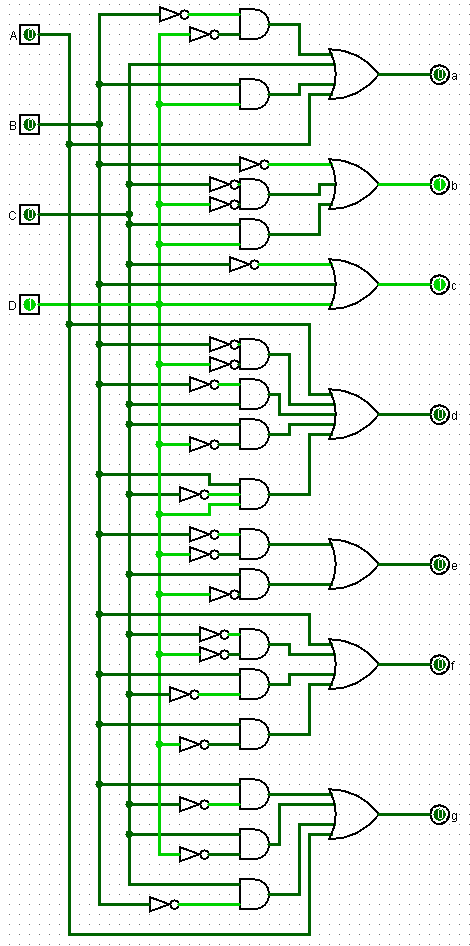
The player with the lights on loses.

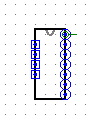
It's not an actual Russian roulette, so I set it as losing if the same person gets caught three times.

* **Description how the project is implemented**

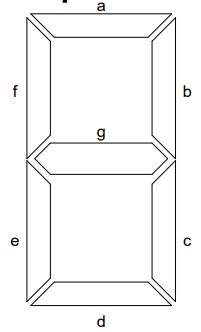
I used one main circuit and two auxiliary circuits.

1. **7segement**

I used it to indicate the number of times each player lost.



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **a** | **b** | **c** | **d** | **e** | **f** | **g** |
| **0** | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| **1** | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| **2** | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| **3** | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| **4** | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| **5** | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| **6** | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| **7** | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| **8** | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **9** | 1 | 1 | 1 | 1 | 0 | 1 | 1 |



a = A + C + BD + B’D’

b = B’ + C’D’ + CD

c = C’ + D + B

d = A + B’D’ + B’C + CD’ + BC’D

e = B’D’ + CD’

f = A + C’D’ + BC’ + BD’

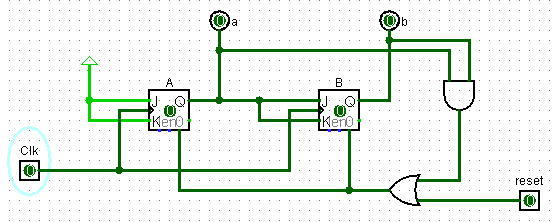
g = A + BC’ + CD’ + B’C

1. **Counter**

테이블이(가) 표시된 사진

자동 생성된 설명I used this to implement the function of counting to 2 and returning to zero.

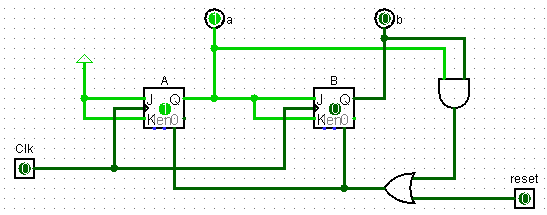
I used two JK flip-flops for this function.



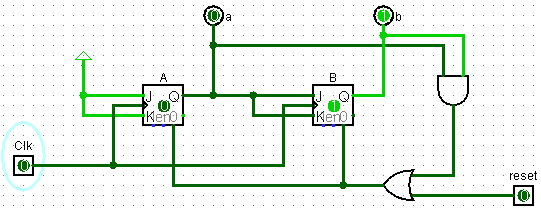


The initial condition is as shown in the picture above.

JA and KA are always 1.



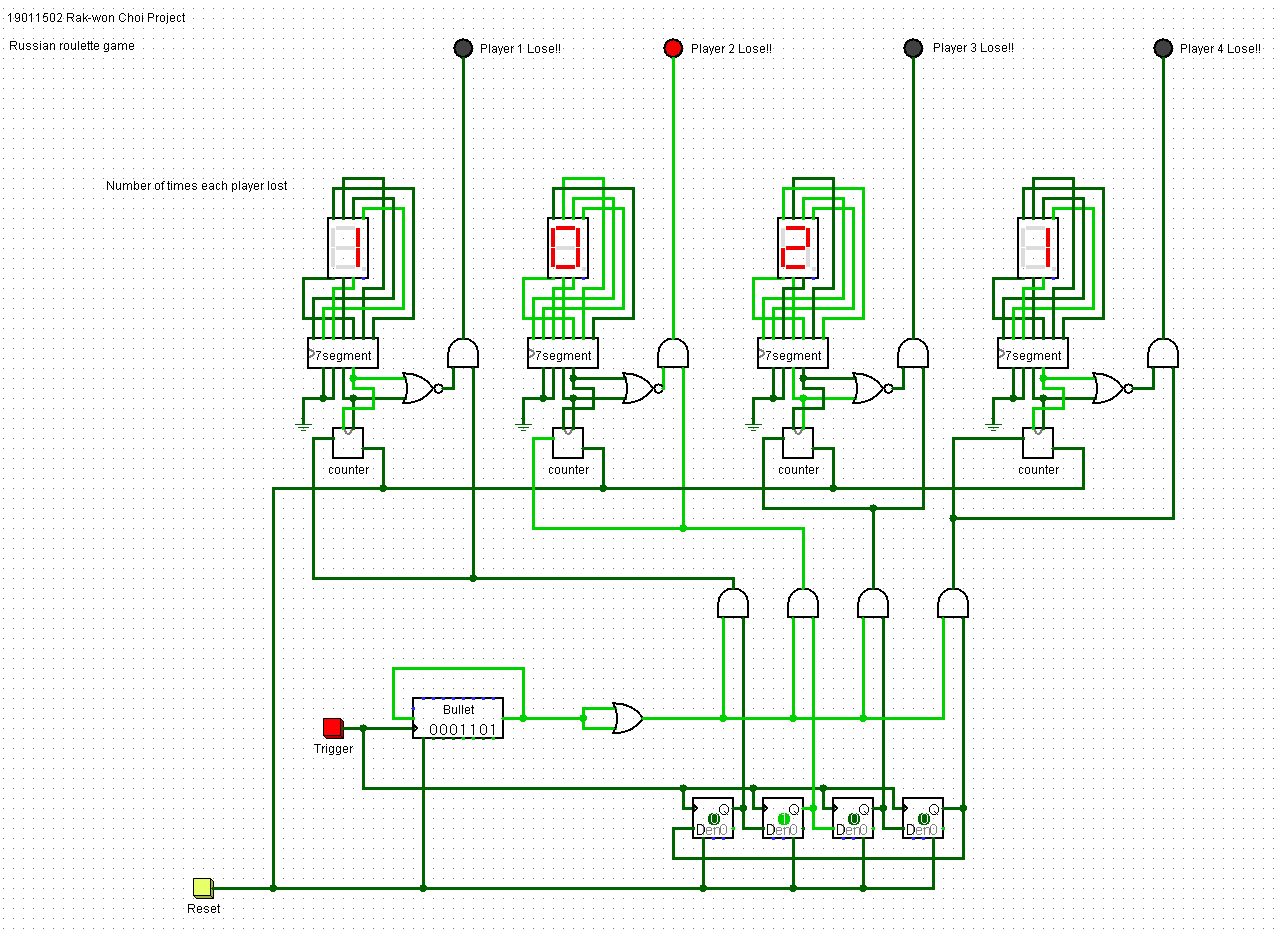
Once the clock changes, it becomes the state of the picture above.



If the clock changes again, it becomes the state of the picture above.

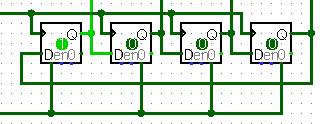
If the clock changes again, both output ‘a’ and ‘b’ become 1, returning to the initial state.

If reset is pressed, the state of all flip flops is initialized to the initial state.

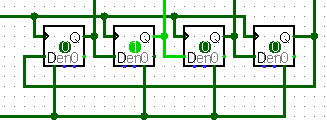
1. **Main**
2. ****

When I press the reset button, the counter and shift register are initialized and the red square box part is initialized as follows.

* **Red Box**



Then, each time the trigger button is pressed, it moves to the side one by one.



A single move of position at each press indicates which of the four players is in turn.

And it repeats because the last output and the first input are connected.

* **Blue Box**

When a bullet is fired(shift register is 1) in the player's turn, the person's count increases by 1.

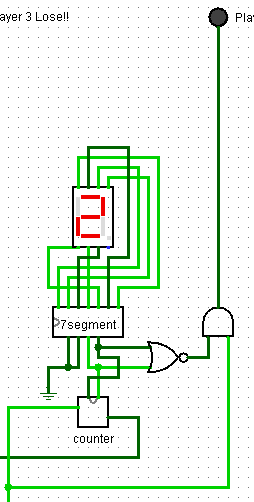
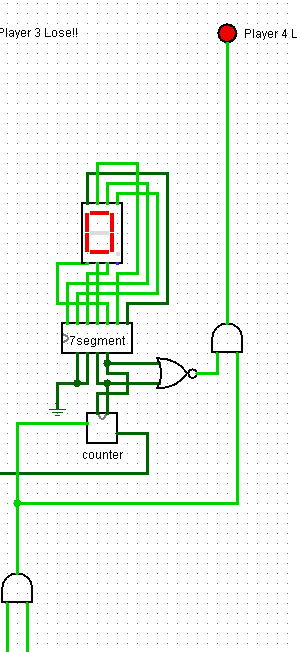
It was implemented as a blue box AND Gate.

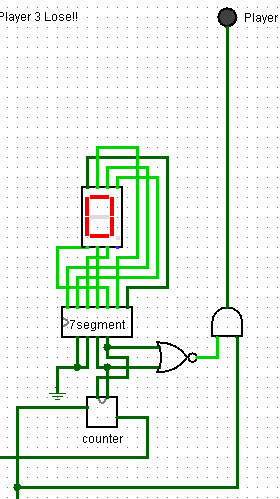
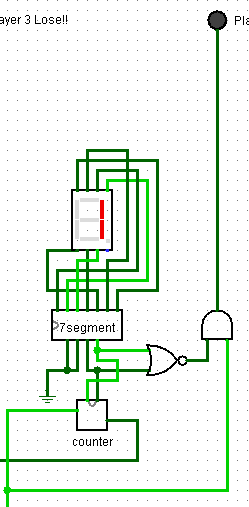
* **Orange Box**

The OR Gate in the orange box was used because of the propagation delay.

In other words, the shift register passes the value quickly, while the value in the red box passes slowly.

If the Or Gate is not used, the counter of the player who completed the current turn and the player on the next turn will increase due to the delay time.

*  **Yellow Box**

In 7segment, we don't use A and B input because we only count to 2. Therefore, A and B were fixed to zero using ground.

The initial state is as shown in the first picture on the left. There is no value delivered to the counter, and only zero is marked.

When a bullet is fired at the player's turn, the value is delivered to the counter. In this case, the value is increased by 1.

When a bullet is fired at 1, the value is changed to 2.

If the value is transmitted in a state of 2, the value returns to zero and the LED indicating the player's defeat is lit.

If the LED comes on when the counter is zero, it also turns on in the initial state. That is, if the value is transferred to the counter, but the counter value is 0, the LED should be turned on. For this purpose, AND Gate was used.