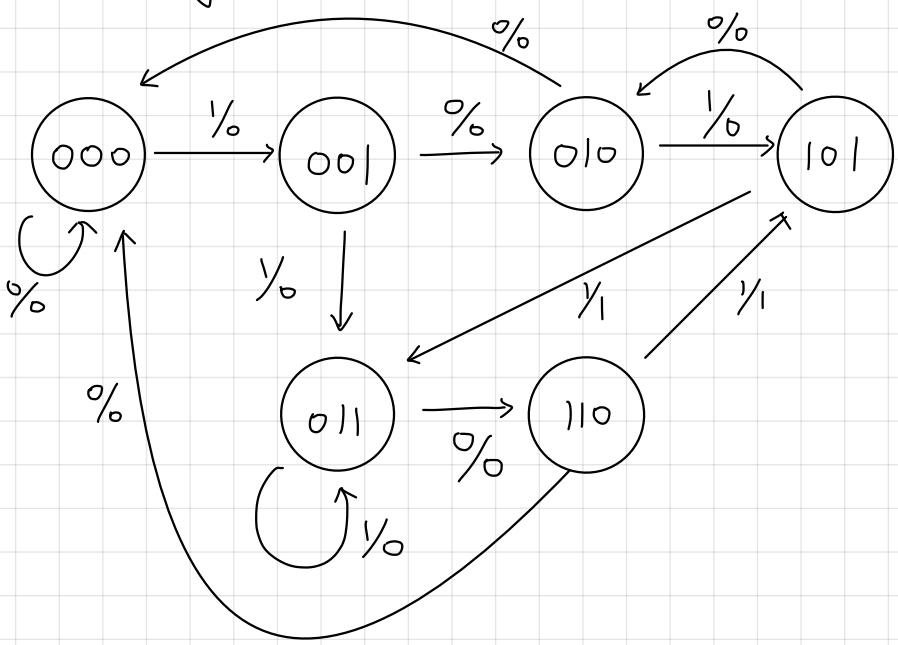
- Patterns Matching

1. State diagram



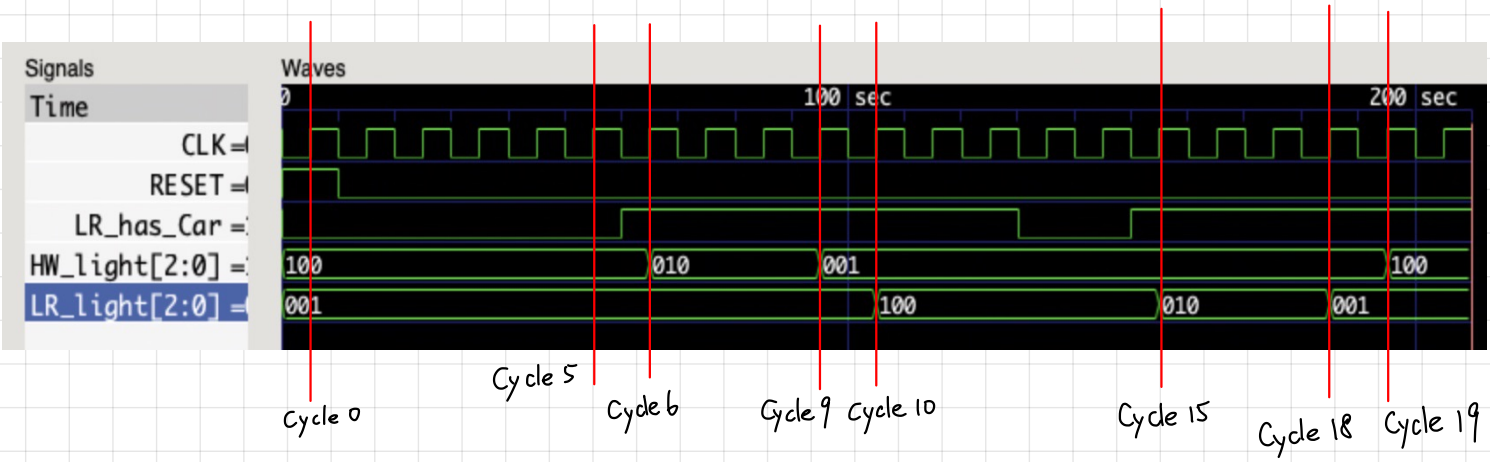
2. Execution result:

cycle	reset	data	flag
0	1	1	0
1	0	0	0
2	0	1	0
3	0	1	0
4	0	1	0
5	0	0	0
6	0	1	1
7	0	0	0
8	0	1	0
9	0	1	1
10	1	0	0
11	0	1	0

- Traffic light controller

Execution result:

cycle	LR_has_Car	HW_light	LR_light
0	0	100	001
1	0	100	001
2	0	100	001
3	0	100	001
4	0	100	001
5	0	100	001
6	1	010	001
7	1	010	001
8	1	010	001
9	1	001	001
10	1	001	100
11	1	001	100
12	1	001	100
13	0	001	100
14	0	001	100
15	1	001	010
16	1	001	010
17	1	001	010
18	1	001	001
19	1	100	001
20	1	100	001

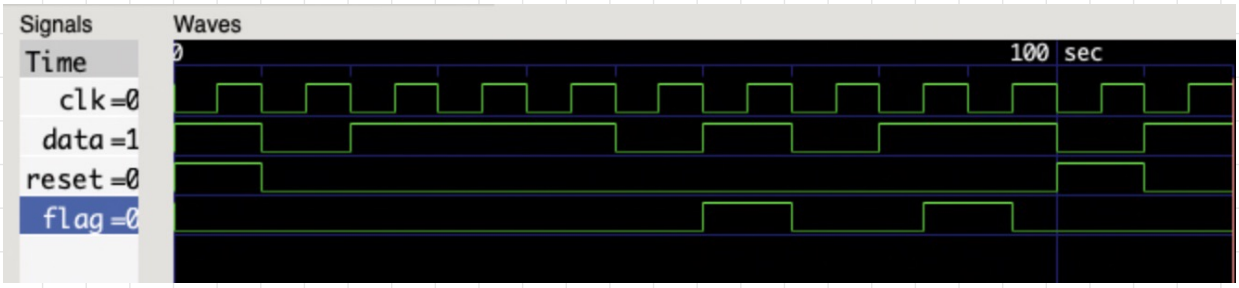


Problem faced:

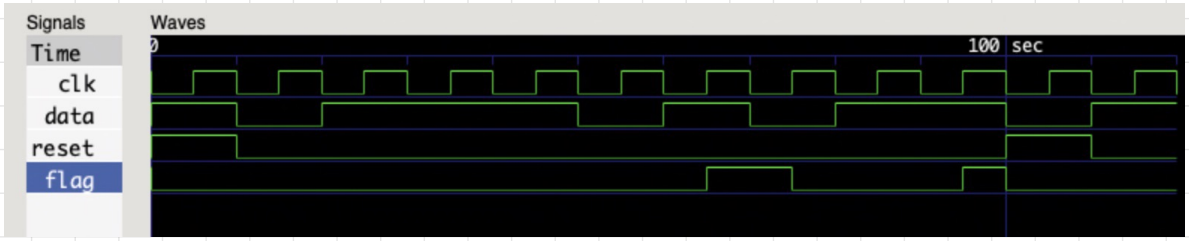
The most significant problem I faced is the testbench's coding. The flip-flop way to design the clock took me sometime to think of how to implement. Second, the gate delay or "setup time" also took me a while, which I thought is the regular codes problem until I used GTK wave to draw the execution result.

Notes For Problem 1:

If we implement the module by making the flag change every time, the result will be the following chart:



However, a mealy machine should only update values when reset == 1 or posclk edge, so I implement in another way, The result will be:



And there will be a gate delay when we output the flag value at posclk edge without a small delay!