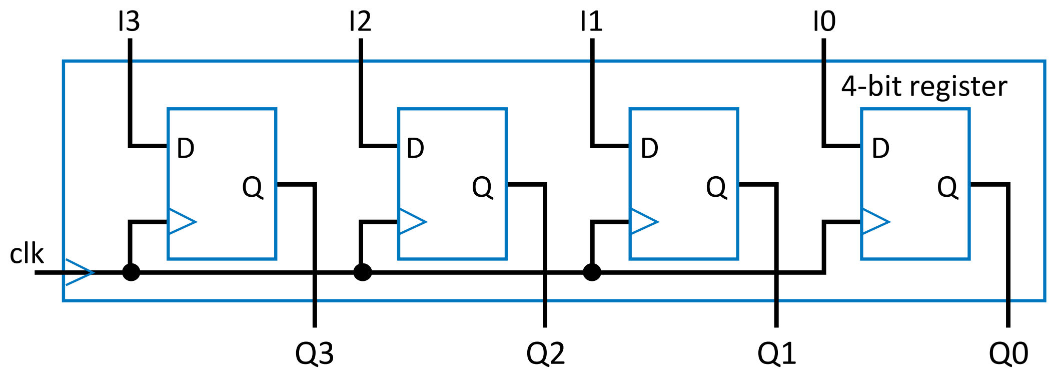
VE270 RC Week 8
Register & Shifter & FSM

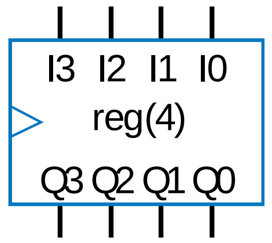
Shi Li

2019.10.9

1. Register
Basic register

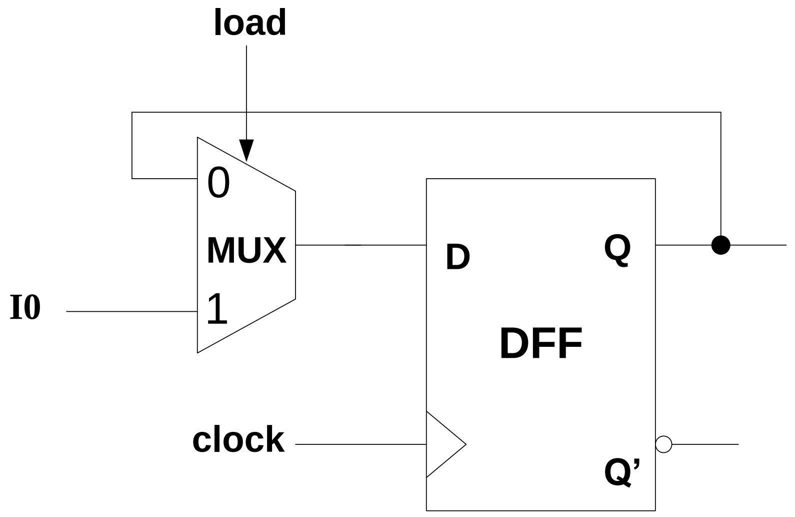
- Used to store data.

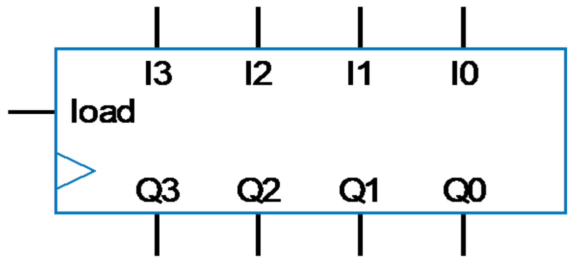




1. Register
Register with load

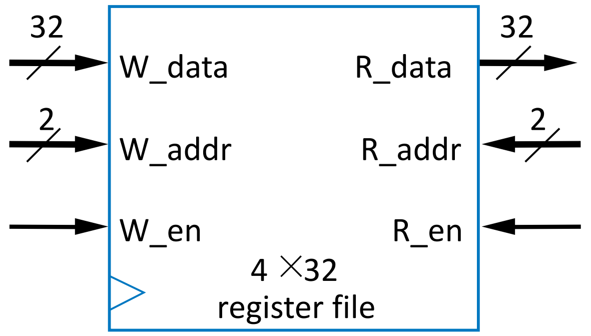
- Add a synchronous load signal.

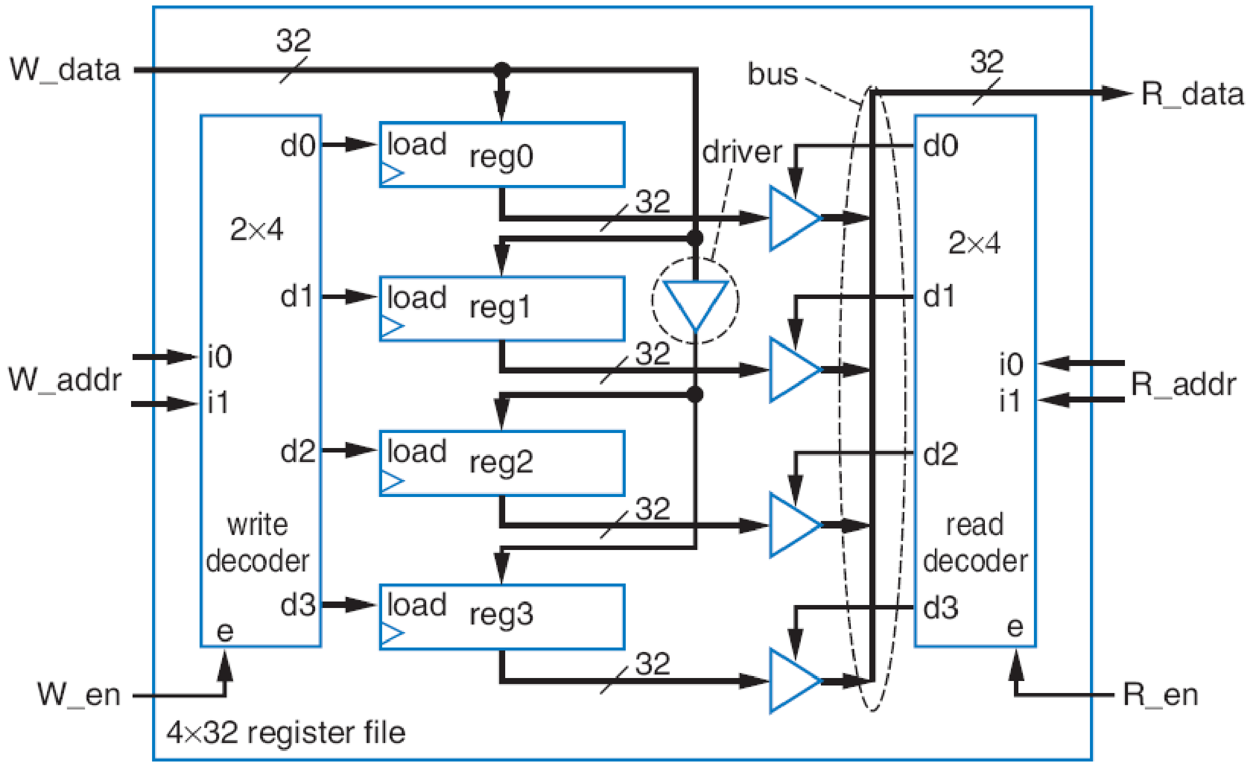




1. Register
Register file

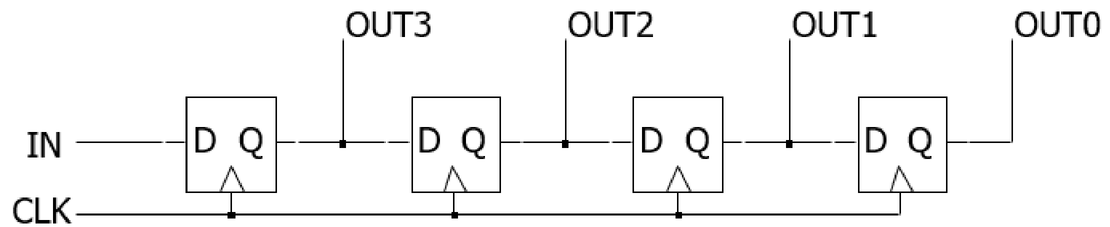
- Used to store a lot of data.



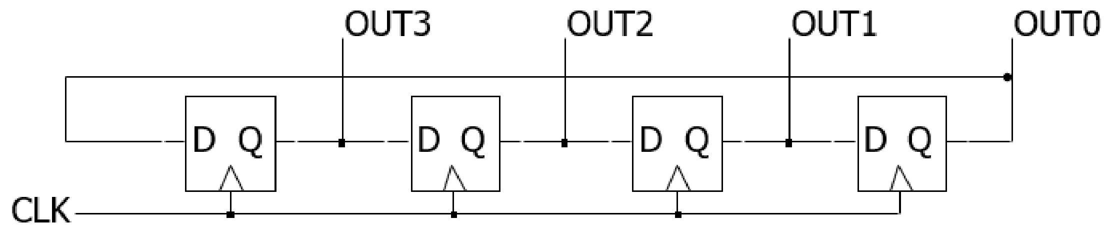


1. Register
Shift register (not shifter) & Rotate register

- Shift or rotate, every clock cycle.



	IN	OUT(3:0)
Initial value:	0	0110
rising edge:	0	0011
rising edge:	0	0001
rising edge:	0	0000
rising edge:	1	1000
rising edge:	0	0100

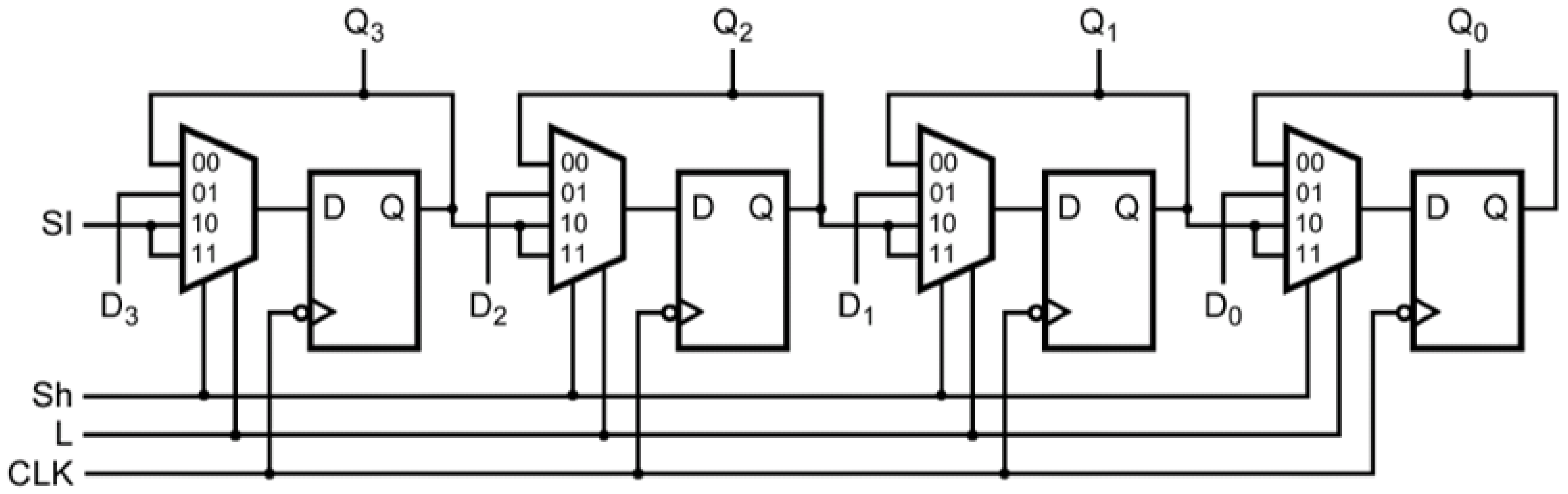


	OUT(3:0)
Initial value:	0110
rising edge:	0011
rising edge:	1001
rising edge:	1100
rising edge:	0110
rising edge:	0011

1. Register
Universal shift register

- Shift or rotate, every clock cycle.
- Exercise: Write Verilog code for it.

Inputs		Action
Sh (Shift)	L (Load)	
0	0	no change
0	1	load
1	X	Shift Right

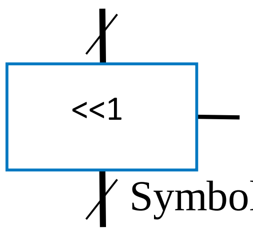


1. Register
Exercise Solution

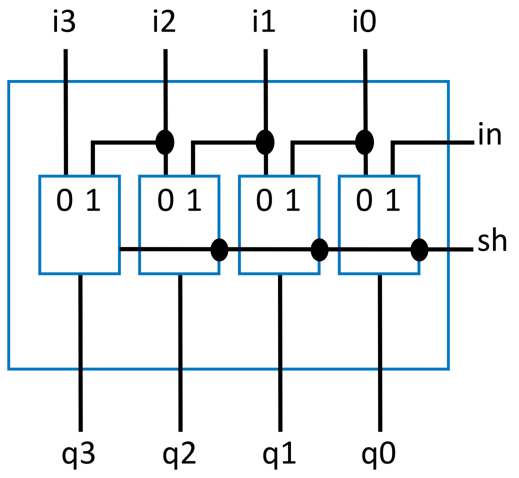
```
module Universal_Shift_Reg (D, Q, SI, Sh, L, Clk);
  input  SI, Sh, L, Clk;
  input  [3:0] D;
  output [3:0] Q;
  reg    [3:0] Q;

  always @ (posedge clock)
  begin
    if (Sh == 1)
    begin
      Q[2:0] <= Q[3:1];
      Q[3]   <= SI;
    end
    else if (L == 1)
      Q <= D;
  end
endmodule
```

2. Shifter (not shift register)
Basic shifter

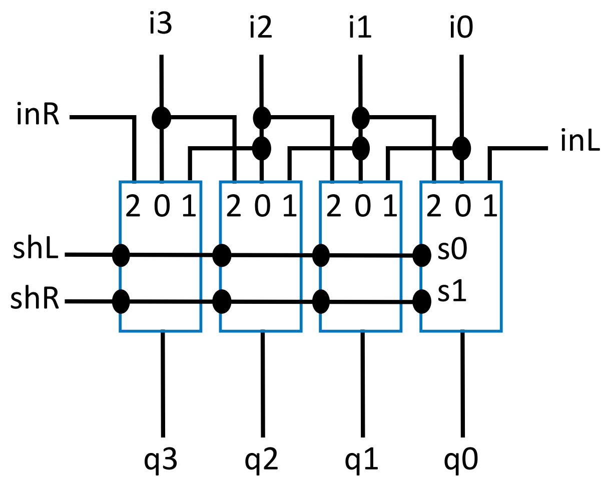
- Combinational component, not sequential! (do not depend on Clk)
- Shift left once = multiply by 2
- Shift right once = divide by 2



Symbol



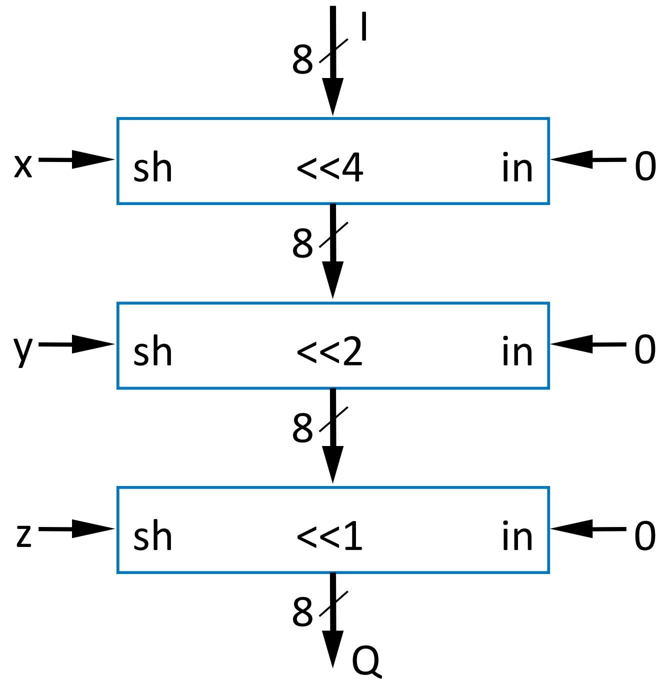
Shifter with left shift or no shift



Shifter with left shift, right shift, and no shift

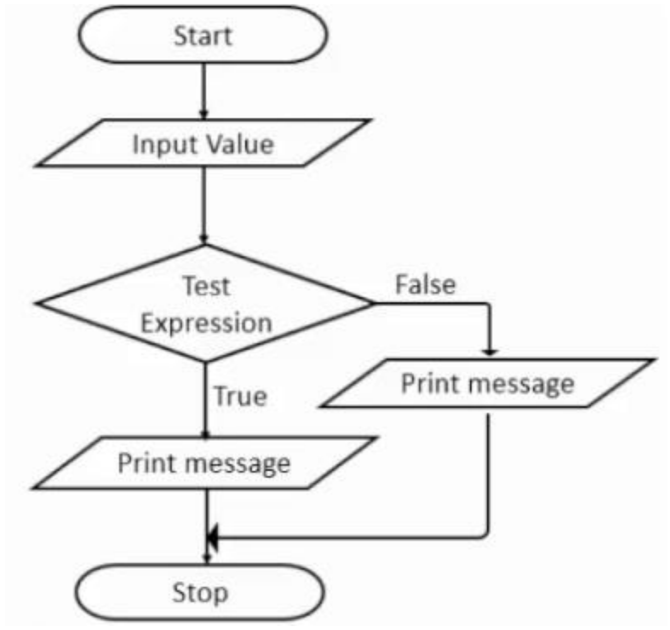
2. Shifter (not shift register) Bigger shifter

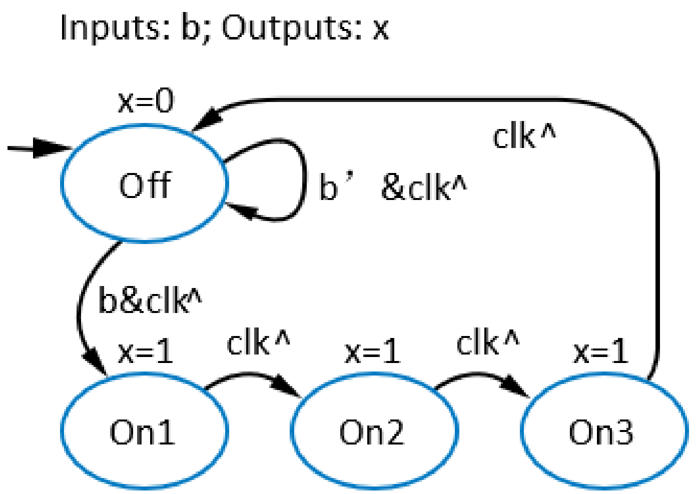
- How to use bigger shifter to shift by any amount?
- Suppose the input is 11111111, how to set xyz so that the output will be 11000000?



3. FSM Concept

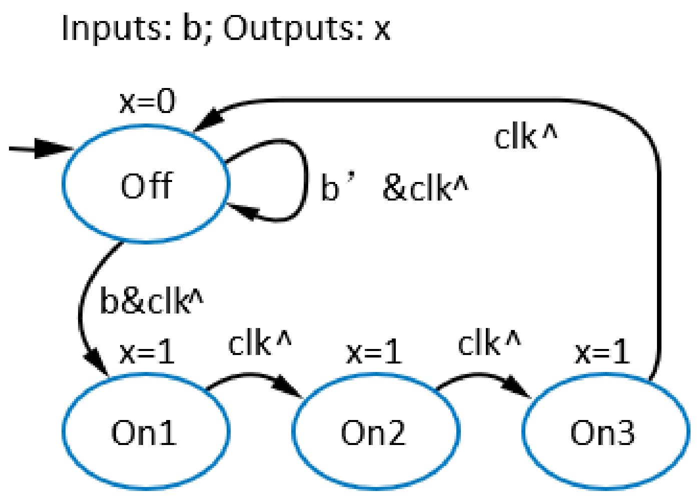
- To some extent, similar to program flowchart.
- One of the most essential concepts in VE270.
- It contains:
  - Set of states
  - Set of inputs & outputs
  - Initial state
  - Set of transitions
  - Set of actions
- What are they in this FSM?





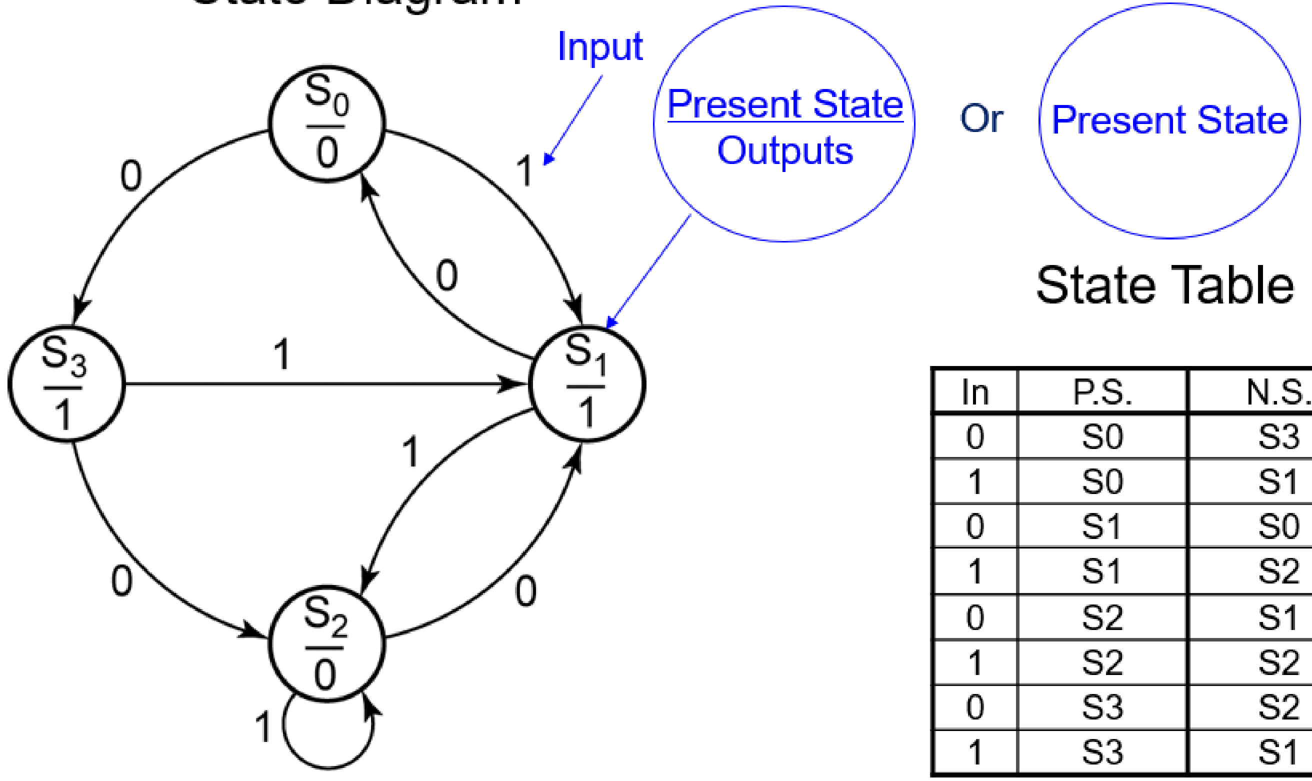
3. FSM Concept

- Solution:
  - Set of states ({Off, On1, On2, On3})
  - Set of inputs & outputs (In: {b}, Out: {x})
  - Initial state (Off)
  - Set of transitions (5 transitions here)
  - Set of actions (x=0 in Off, x=1 in On1, …)



3. FSM
State diagram & State table

State Diagram



State Table

In	P.S.	N.S.	Out
0	S0	S3	0
1	S0	S1	0
0	S1	S0	1
1	S1	S2	1
0	S2	S1	0
1	S2	S2	0
0	S3	S2	1
1	S3	S1	1

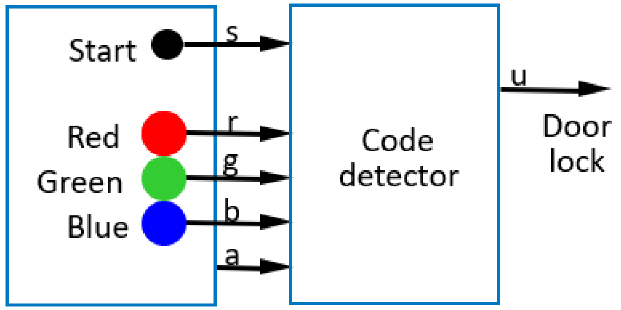
3. FSM
Pay attention to State Transition Property!

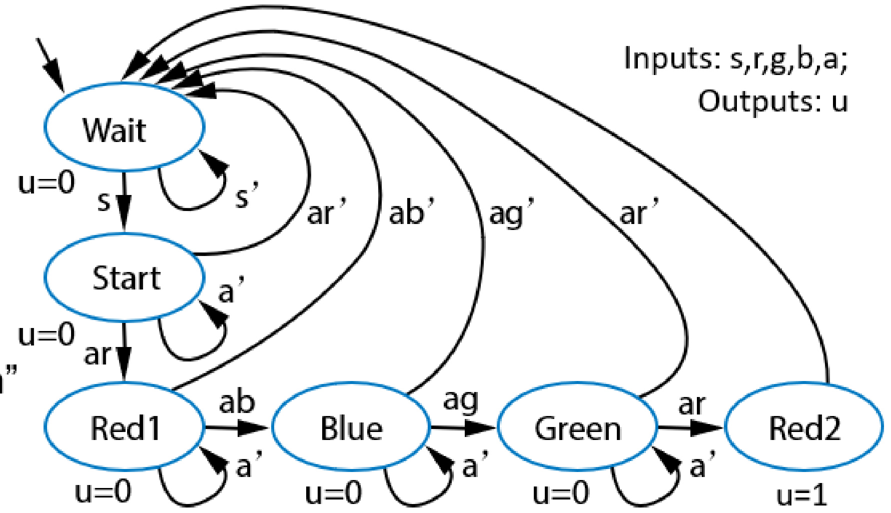
- Only one condition should be true
- One condition must be true
- All conditions must be considered

3. FSM
Example: Digital lock (lecture notes review)

- Unlock door (u=1) only when buttons pressed in sequence:

  - start, then red, blue, green, red
- Input buttons: *s, r, g, b*
- Input *a* indicates that some color button pressed
- FSM
  - Wait for start (s=1) in "Wait"
  - Once started, go to "Start", then
    - If see red, go to "Red1"
    - Then, if see blue, go to "Blue"
    - Then, if see green, go to "Green"
    - Then, if see red, go to "Red2", and u=1
    - Wrong button at any step, return to "Wait", without opening door

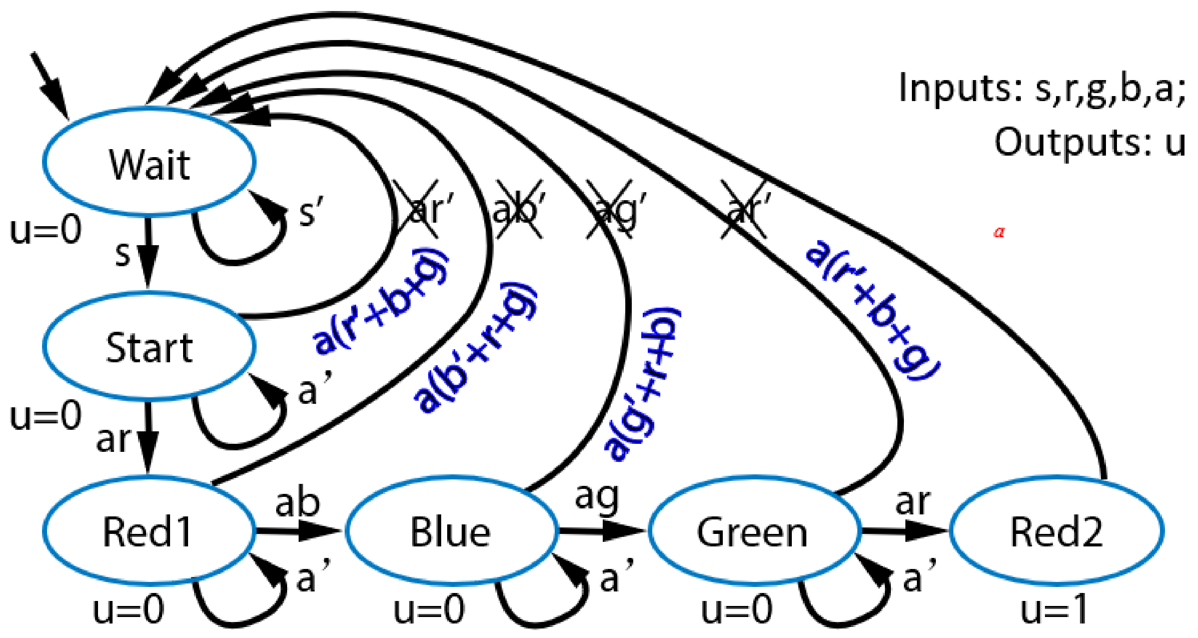




Q: Can you trick this FSM to open the door, without knowing the code?

A: Yes, hold all buttons simultaneously

3. FSM
Example: Digital lock (lecture notes review)



- New transition conditions detect if wrong button pressed, returns to “Wait”

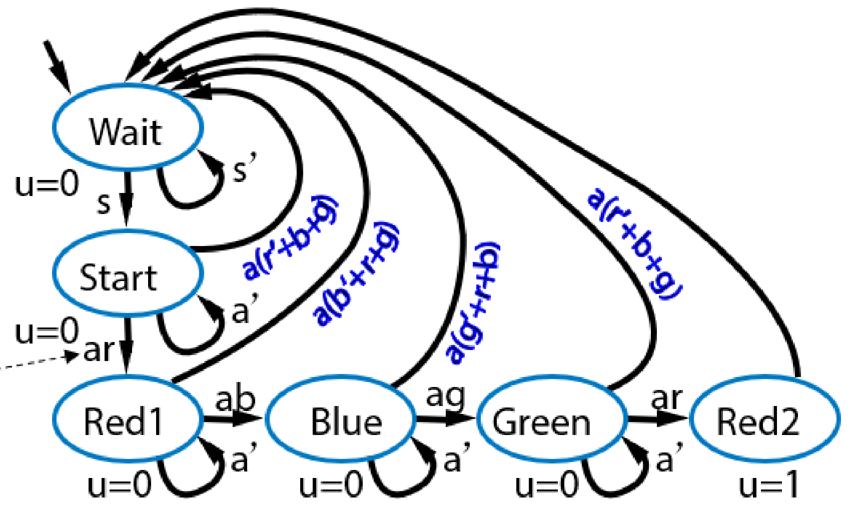
3. FSM
Example: Digital lock (lecture notes review)

- Recall code detector FSM
  - Do the transitions obey the two required transition properties?

    NO!

  - How would it go wrong?

    E.g. arbg = 1111
    How to solve?



Answer: ar should be arb'g'
(likewise for ab, ag, ar)