CS221: Digital Design

FSM- Correctness and Completeness

A. Sahu

Dept of Comp. Sc. & Engg.

Indian Institute of Technology Guwahati

Outline

- Finite State Machine
- FSM Completeness
- FSM Correctness

FSM Completeness

- Finite State Machine Transition Function

  $\delta : S \times I \rightarrow S$

- δ : For all the states of FSM and For all the type of input
- FSM Completeness
  - If for all the states, all the input combination specified in transition function δ

FSM Correctness

- Finite State Machine Transition Function

  $\delta : S \times I \rightarrow S$

- FSM is correct if For all the state of FSM
  - AND operation of every pair of out going edges of a state =0
    - *Meaning: for some condition FSM should not transition to more than one state*
  - OR operation all out going edges of a state =1
    - *Meaning: there should be at least one transition for every condition*

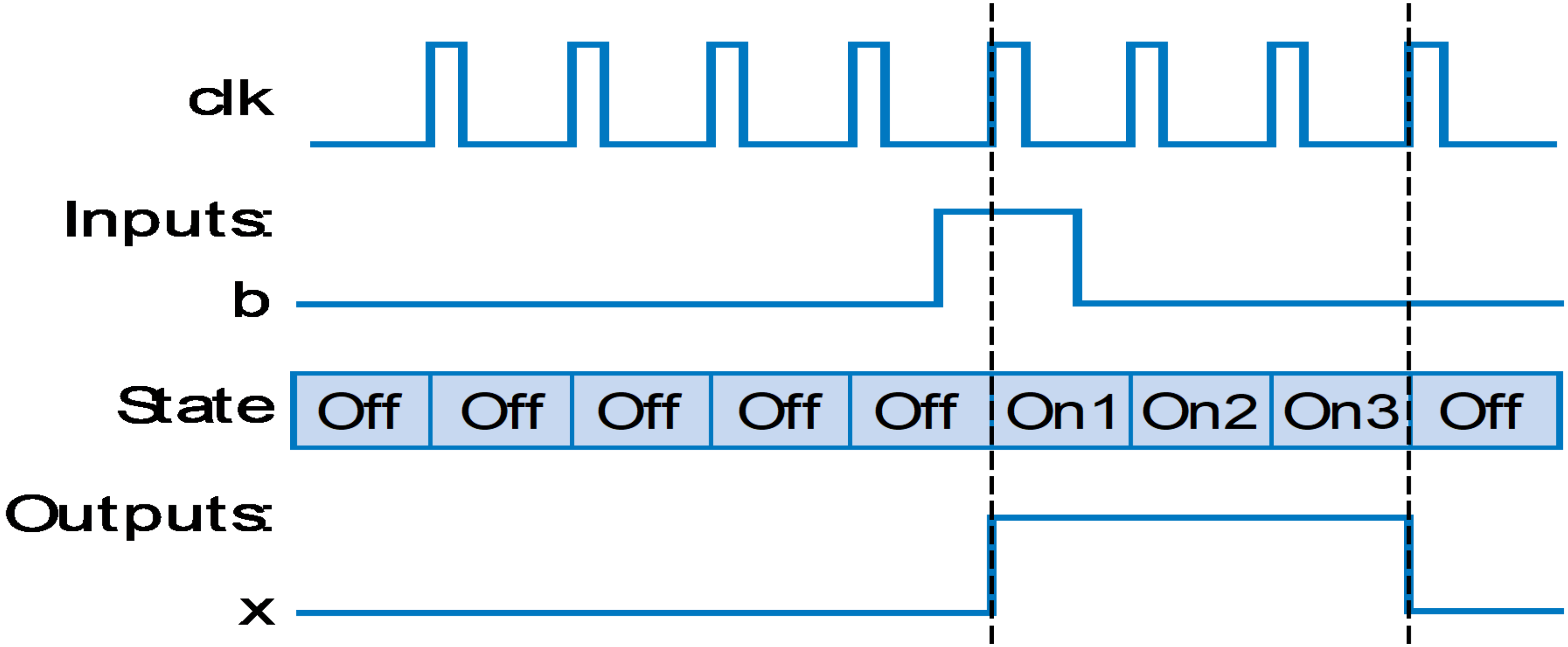
FSM Completeness

Example Three-Cycles High Laser Timer

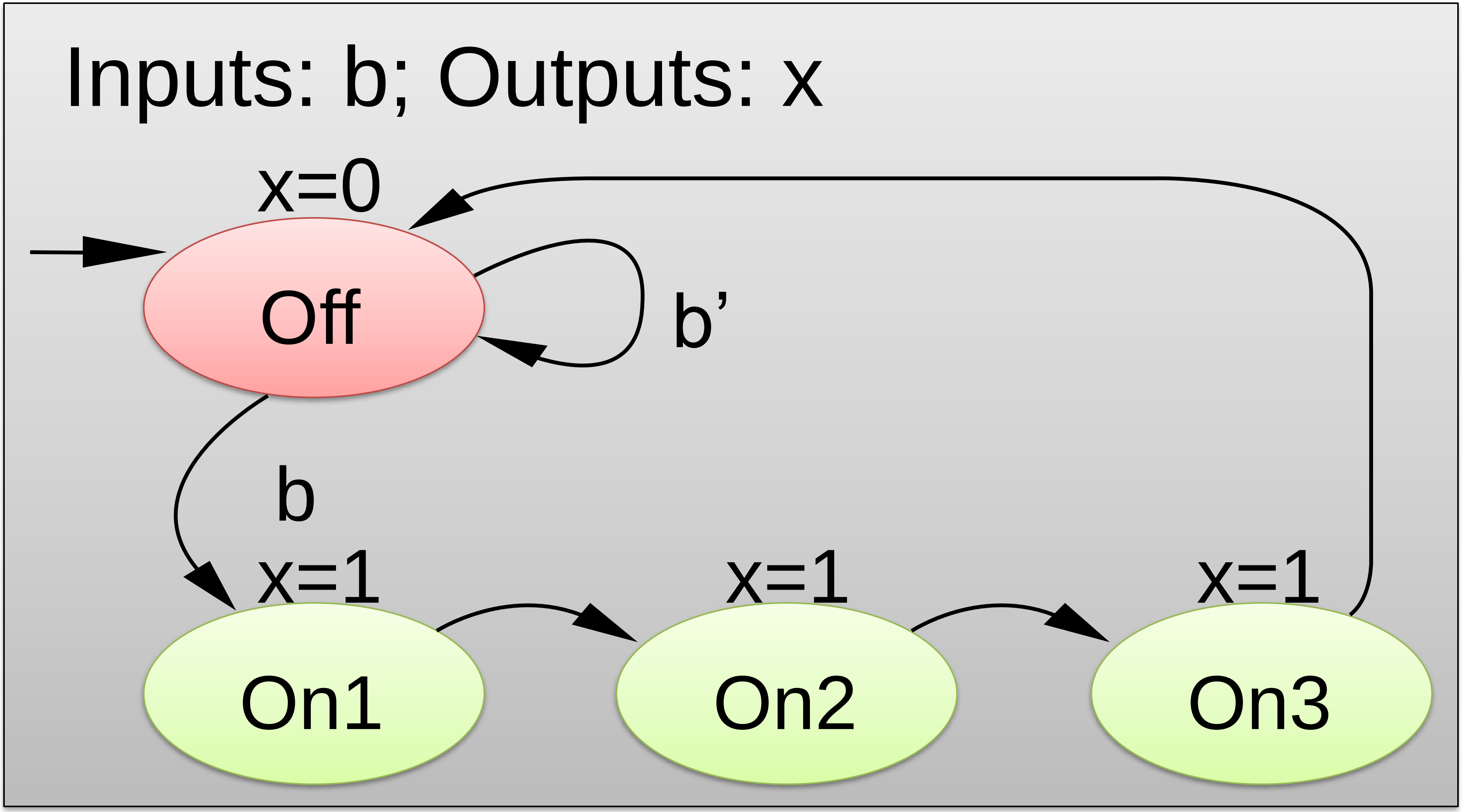
Extend FSM to Three-Cycles High Laser Timer

- Four states: Wait in “Off” state while b is 0 (b’)
- When b=1 (& rising clock edge), transition to On1
  - Sets X=1
  - On next two clock edges, transition to On2, then On3, which also set x=1
- So x=1 for three cycles after button pressed

Extend FSM to Three-Cycles High Laser Timer

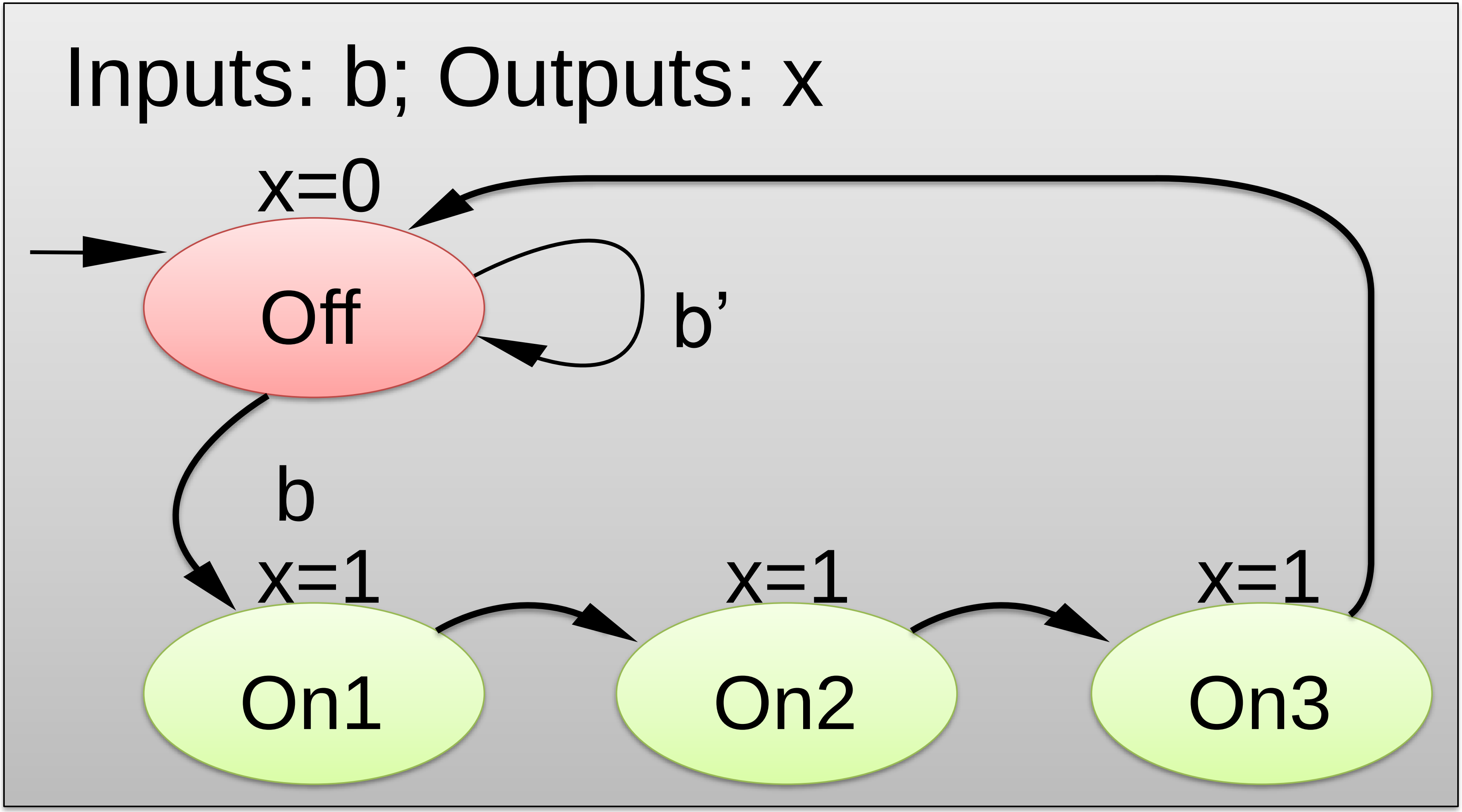


FSM of Three-Cycles High Laser Timer



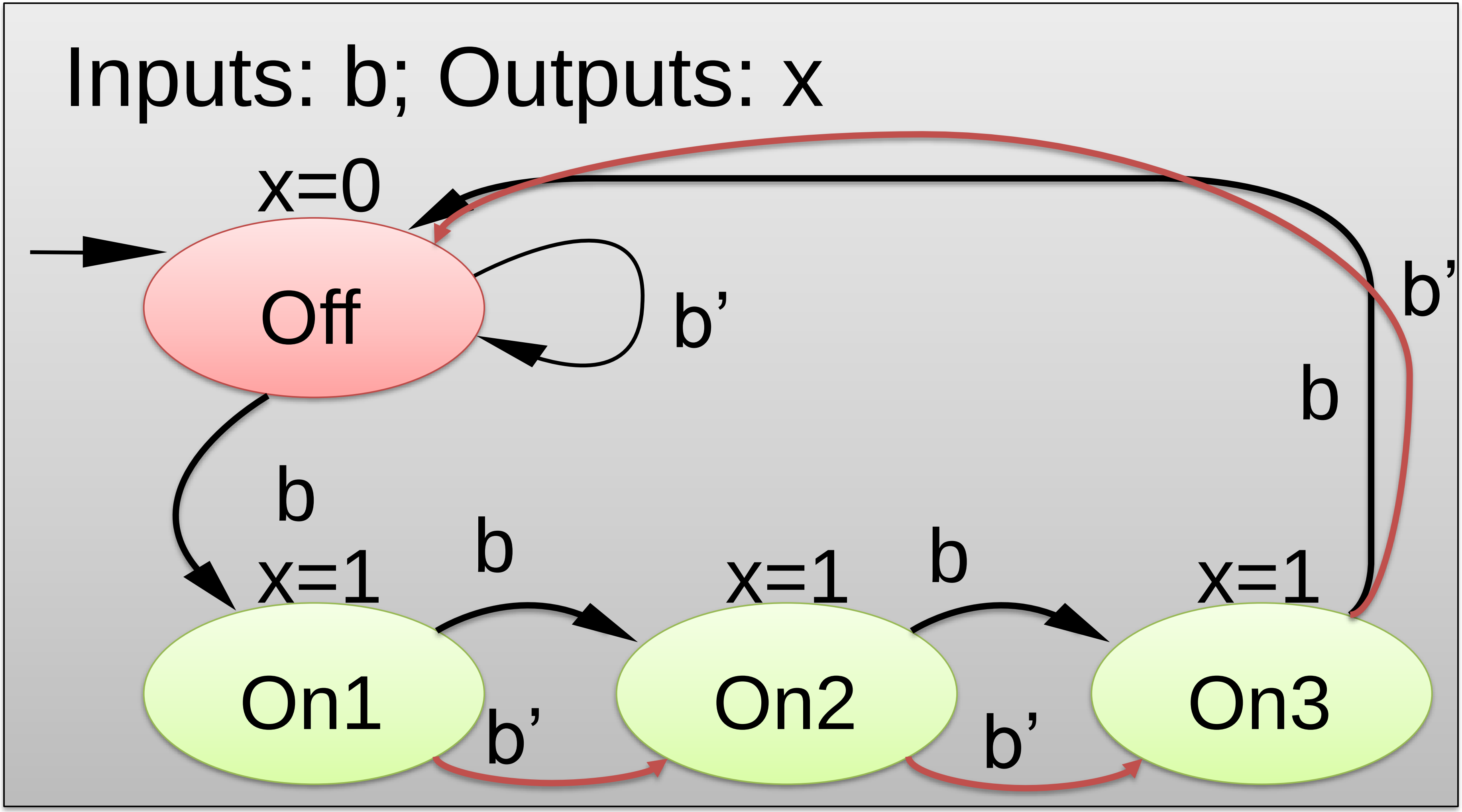
Note: Transition with no associated condition thus transistions to next state on next clock cycle

FSM Completeness



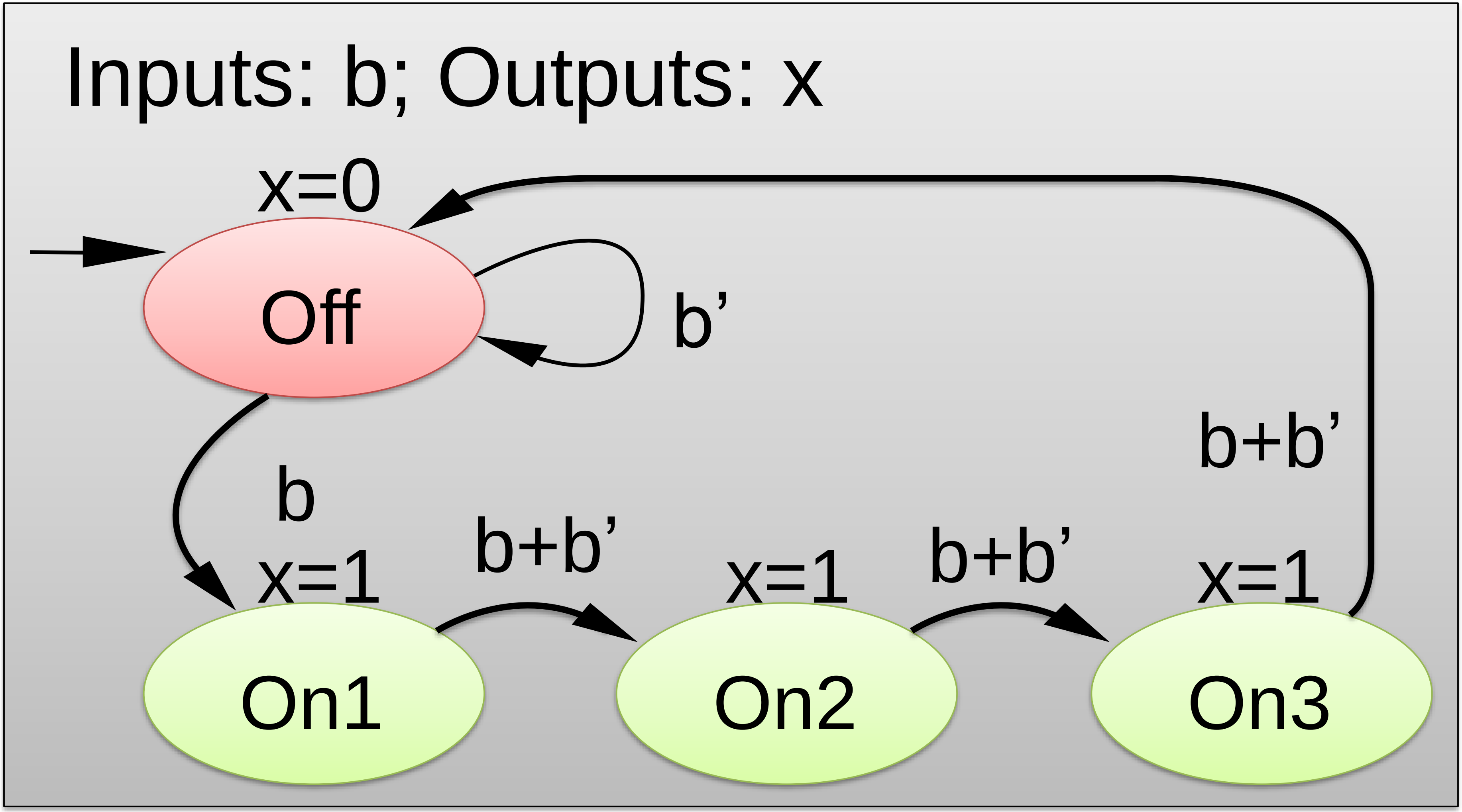
Value of b=1:   0111..repeat,   ***Is this FSM complete?***

FSM Compltetenesss



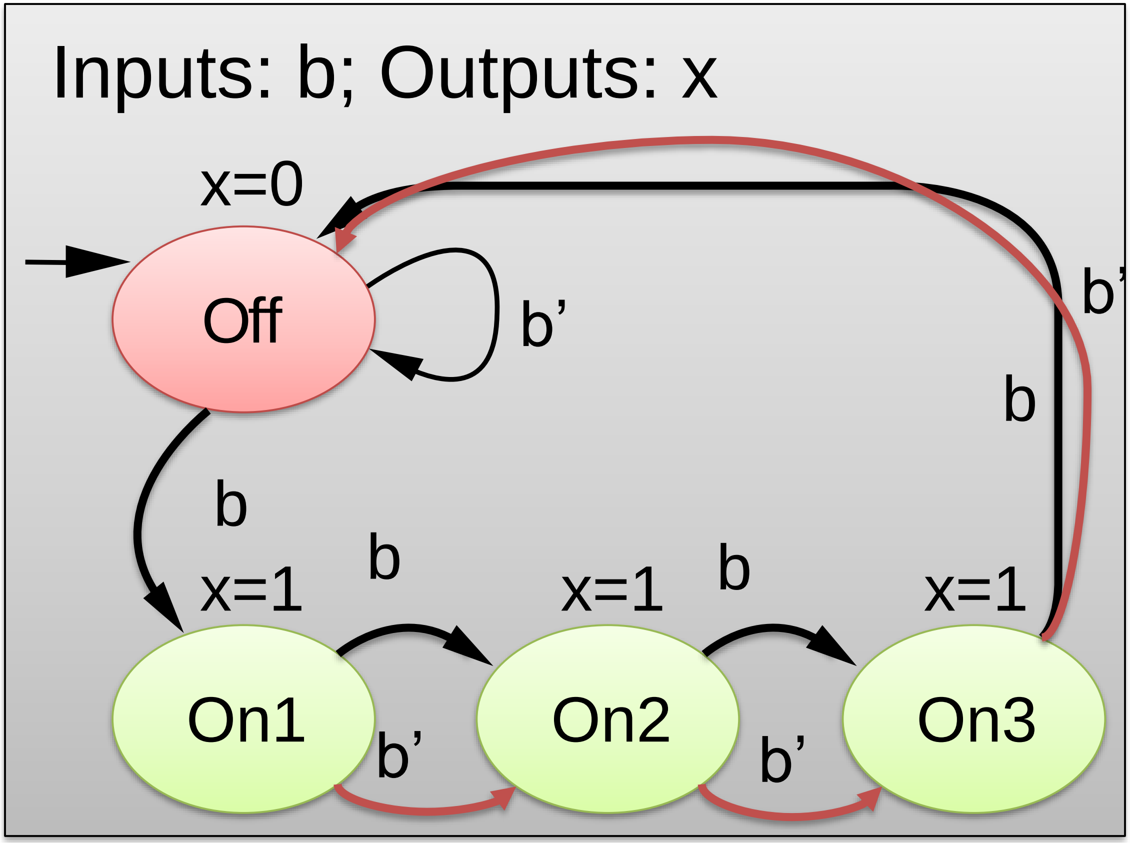
Value of b=1:   0111..repeat,   ***Is this FSM complete?***

FSM Completeness



Value of b=1: 0111..repeat, ***Is this FSM complete?***

FSM Correctness



AND operation every pair of out going edges of a state =0

$$b.b'=0$$

OR operation all out going edges of a state =1

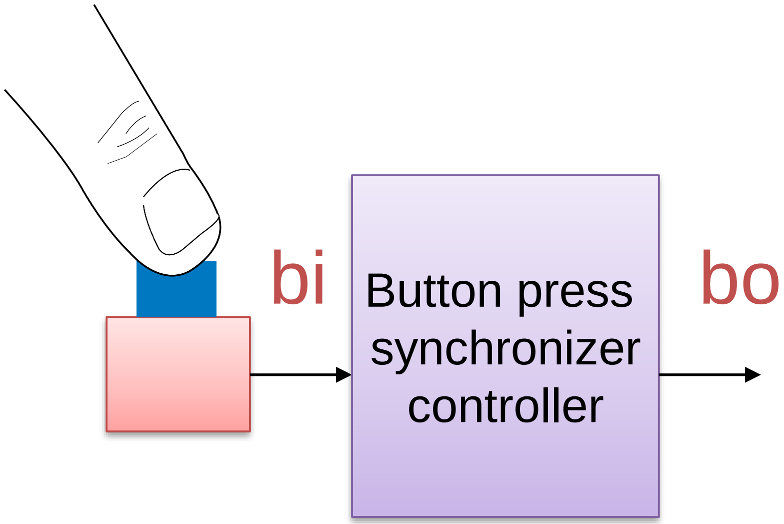
$$b+b'=1$$

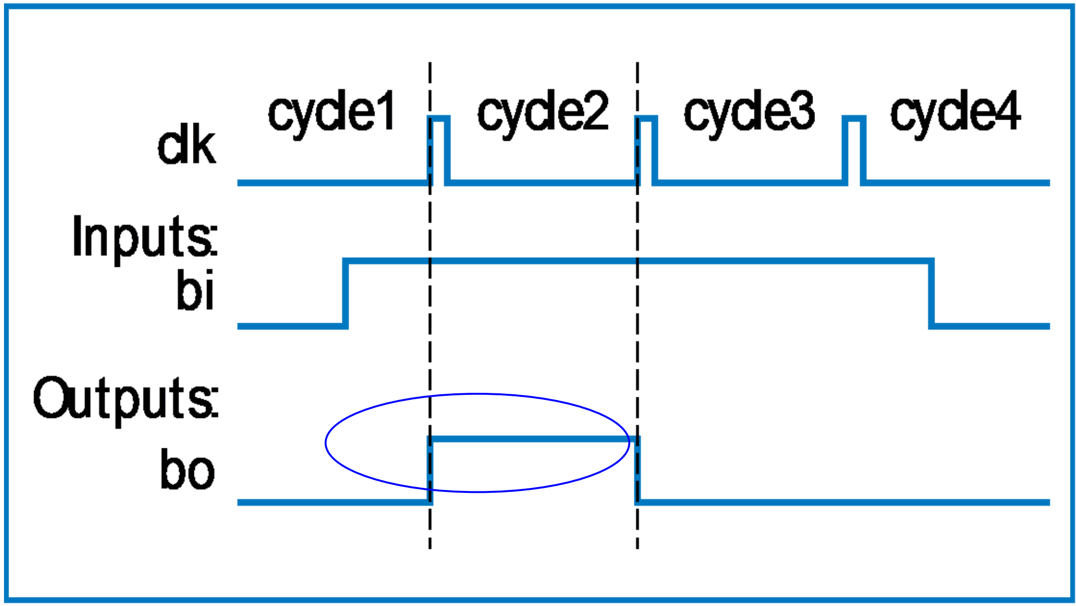
Is this FSM is correctly specified: based on rule

FSM Example 9:
Button Press Synchronizer

Example 9 : Button Press Synchronizer

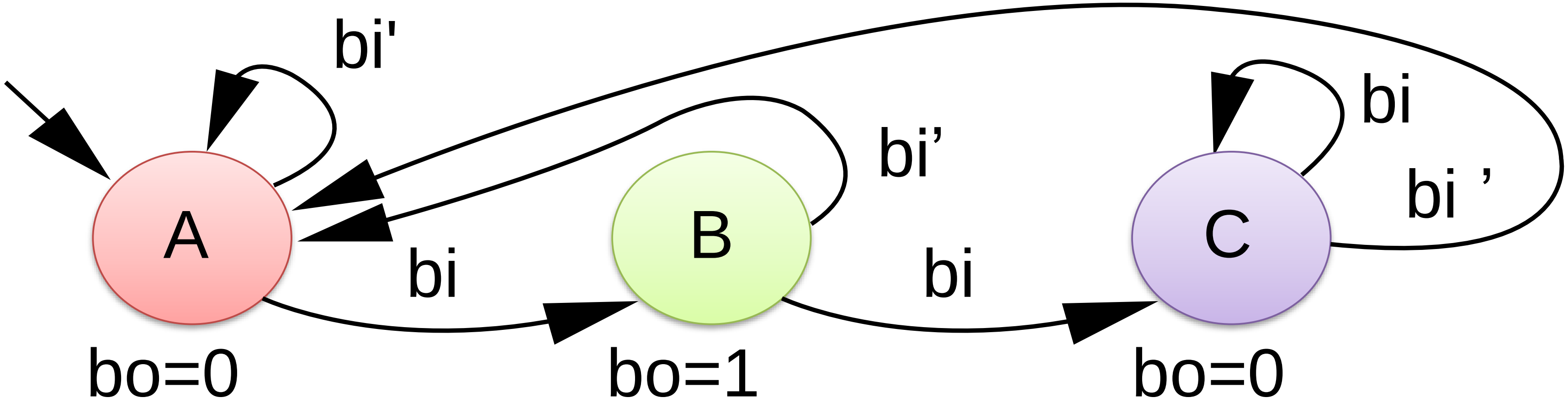
- **English Language Specification**
- **All most all the keyboards use this method**
- We want simple sequential circuit
  - Converts button press to single cycle duration
  - Regardless of length of time that button actually pressed





FSM Example 9 : Button Press Synchronizer

FSM inputs: bi; FSM outputs: bo



I am Off
When B=0

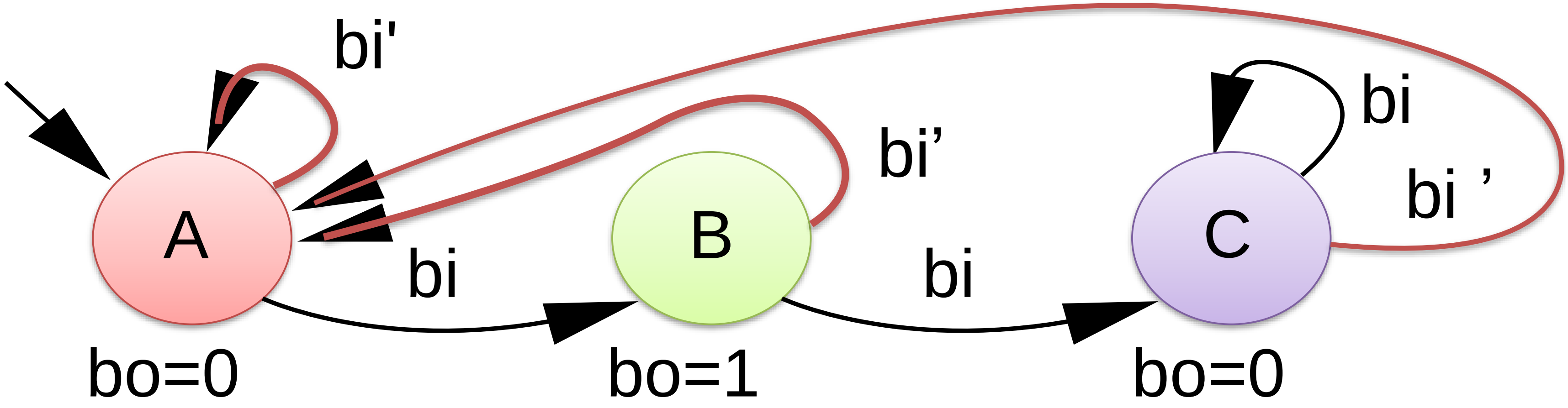
I am On in First
CLK and
When B=1

I am Off
Even if B=1

Step 1: Design FSM

Button Press Synchronizer: Completeness & correctness

FSM inputs: bi; FSM outputs: bo

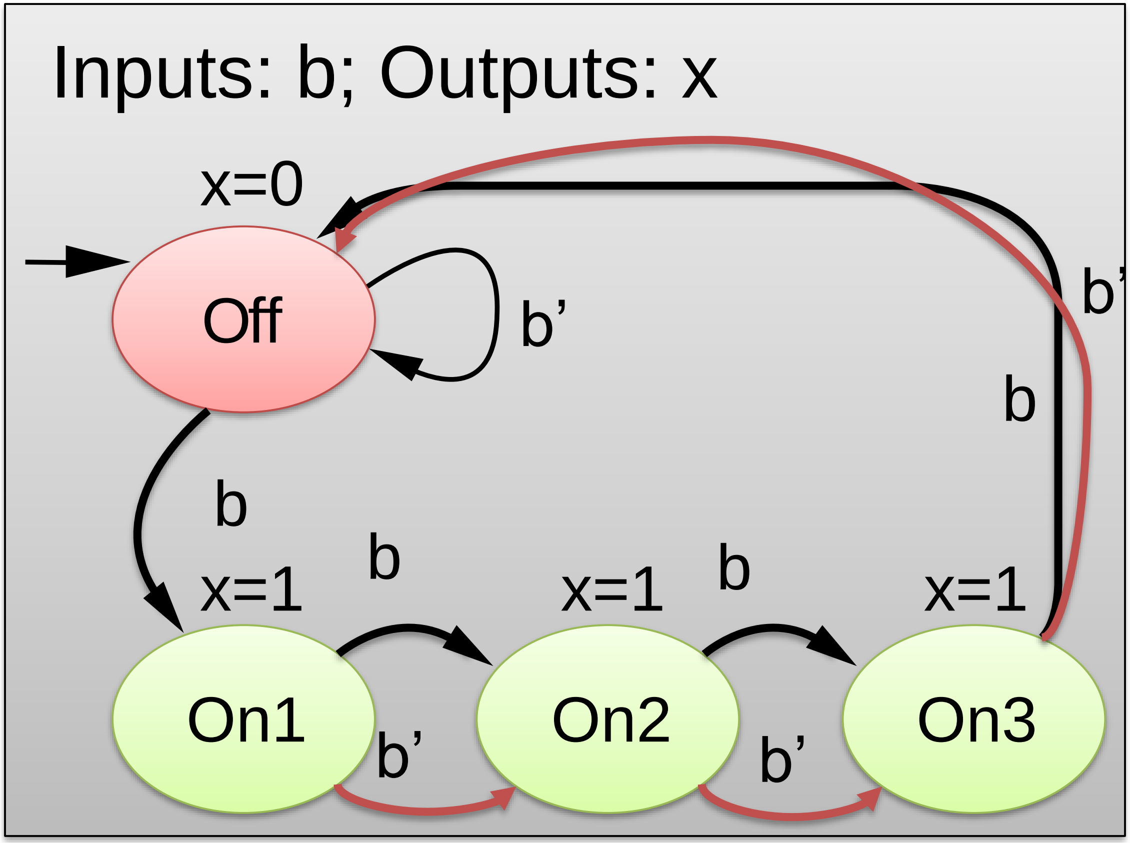


For all the state exactly two out going transitions, one for b and other for b’

b.b’=0 b+b’=1

FSM With Transition specified in Boolean expression or Compressed form

FSM Correctness



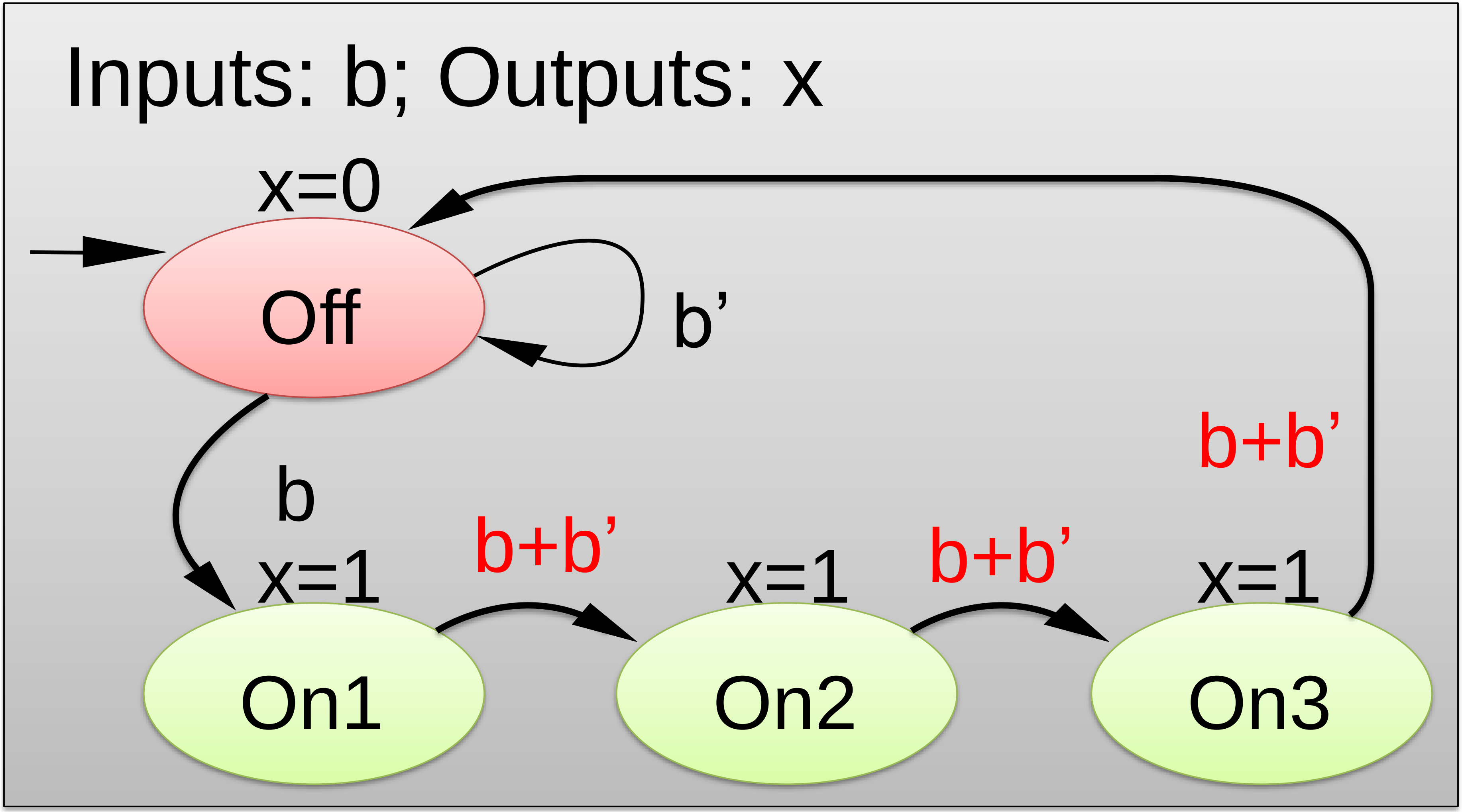
AND operation every pair of out going edges of a state =0

$$b.b'=0$$

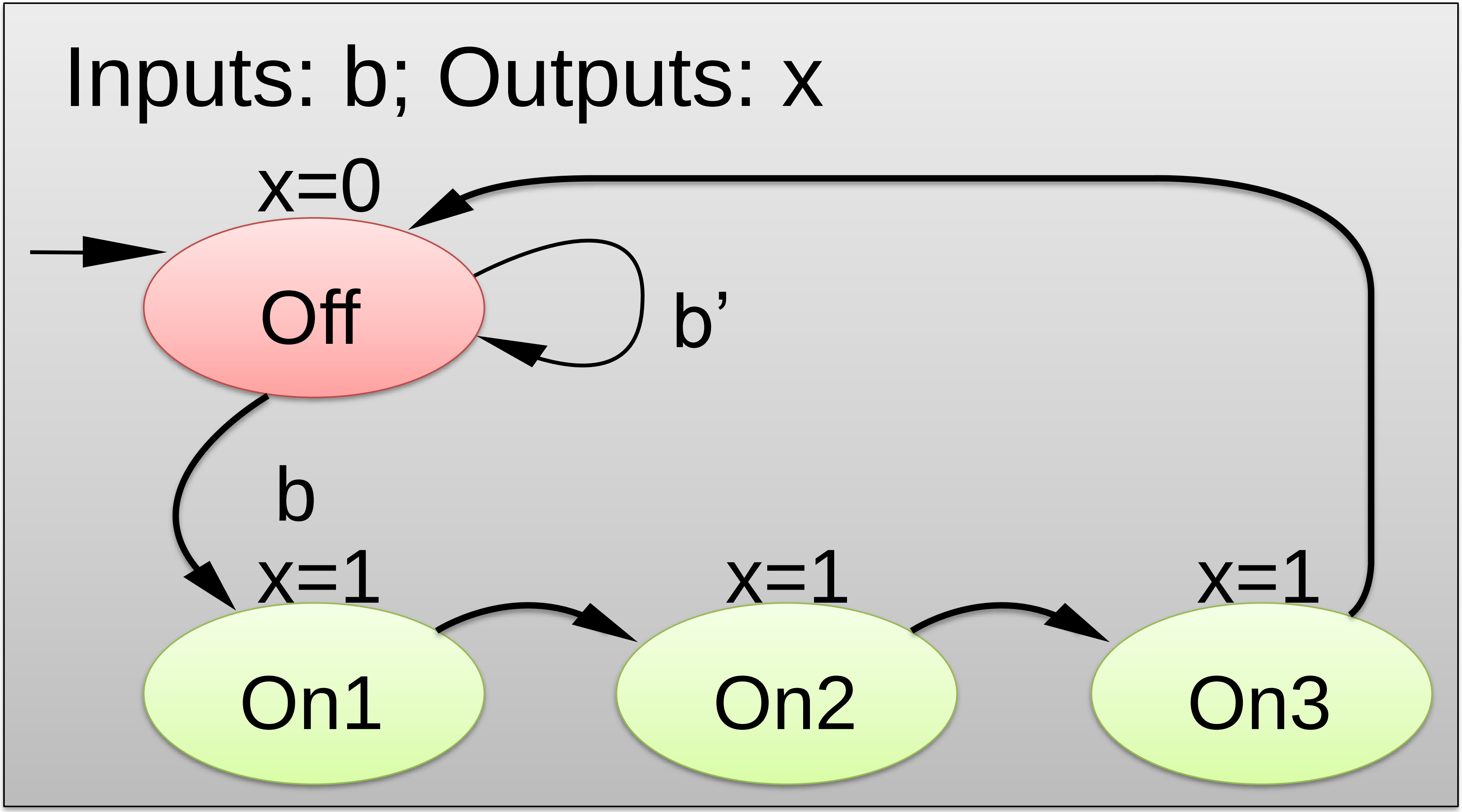
OR operation all out going edges of a state =1

$$b+b'=1$$

FSM Completeness



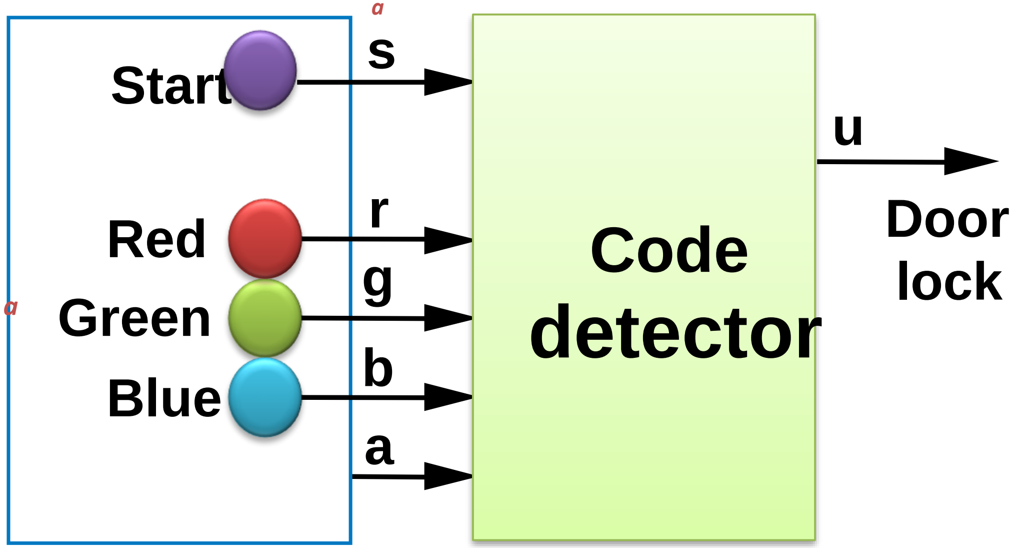
FSM Completeness



FSM Example 11: Code Detector

FSM Example : Code Detector

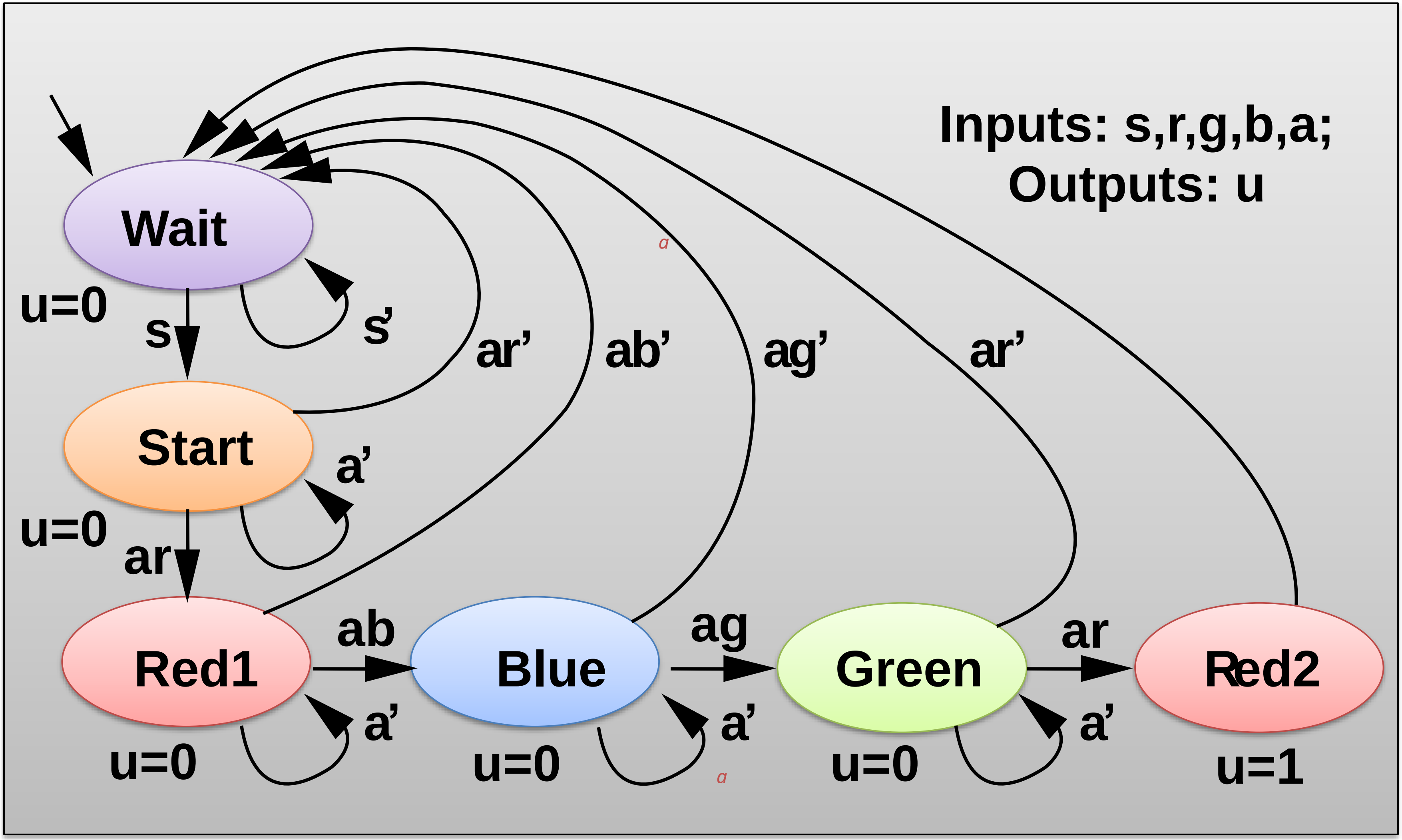
- Unlock door (u=1) only when buttons pressed in sequence:
  - **start, then red, blue, green, red**
- Input from each button: *s, r, g, b*
  - Also, output *a* indicates that some colored button pressed



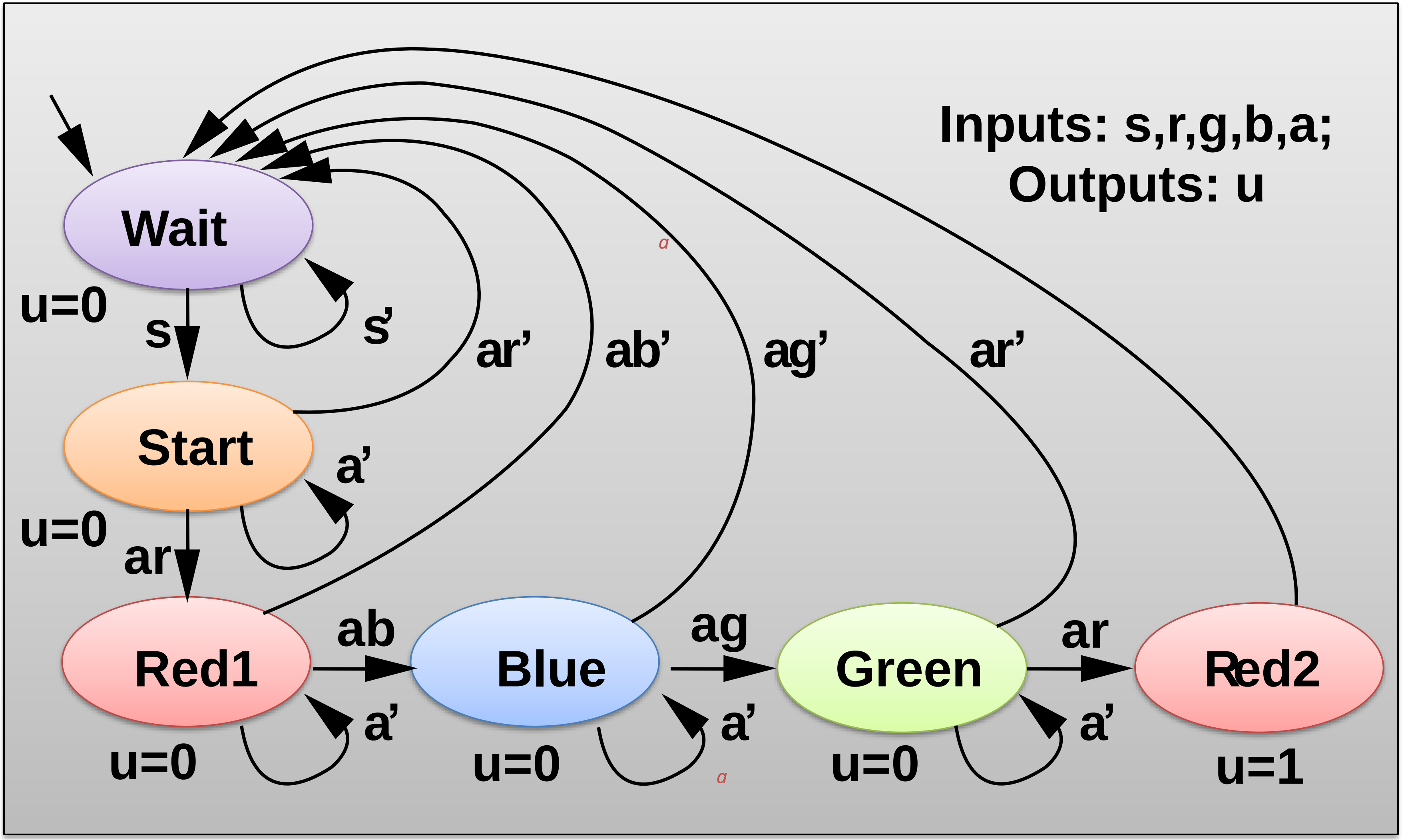
FSM Example 11: Code Detector

- Wait for start (s=1) in “Wait”,
- **Once started (“Start”)**
  - 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”
  - In that state, open the door (u=1)
  - Wrong button at any step, return to “Wait”

FSM Example 11 : Code Detector



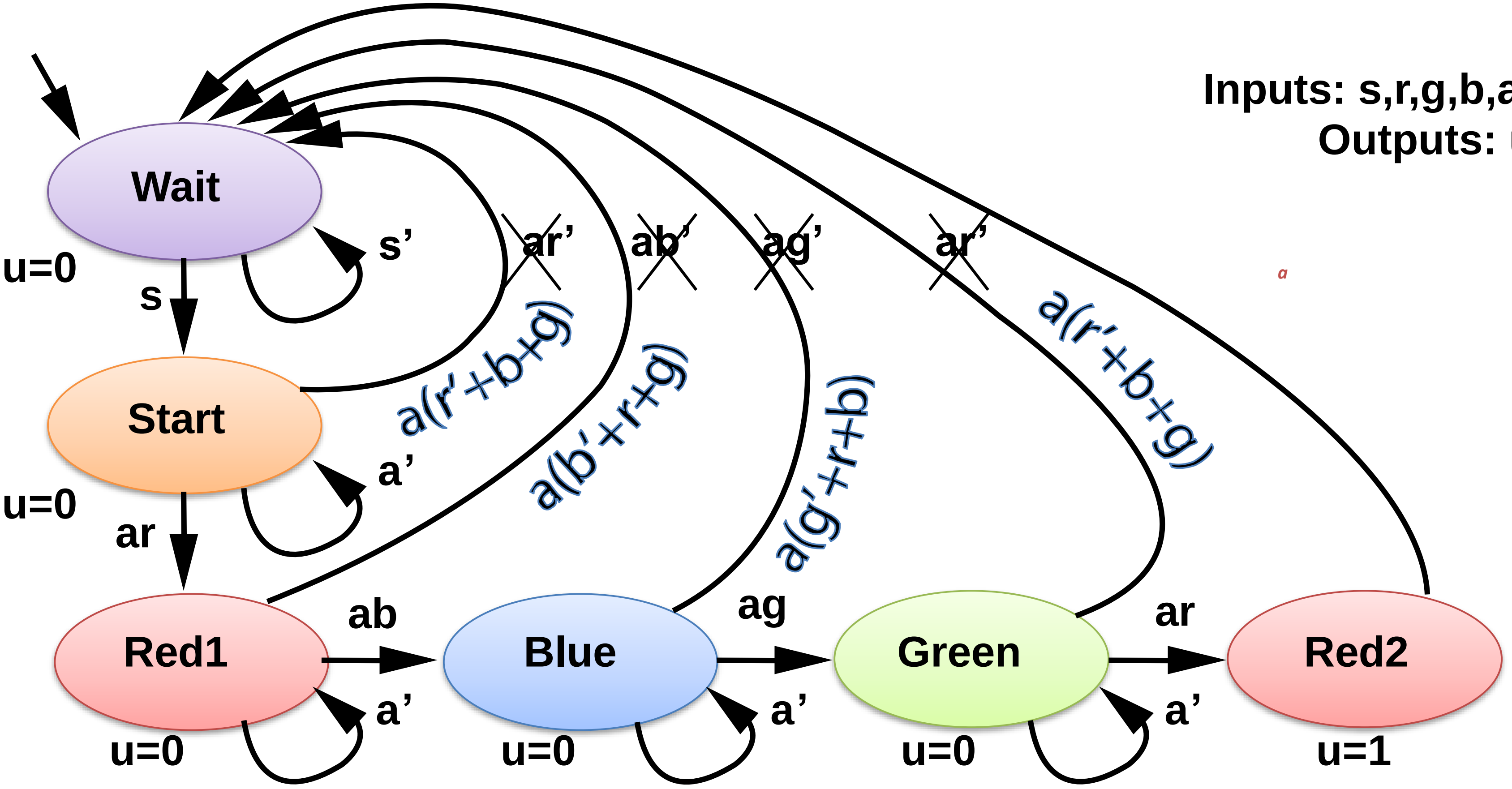
FSM Example 11 : Code Detector



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

A: Yes, hold all buttons simultaneously

Improve FSM for Code Detector

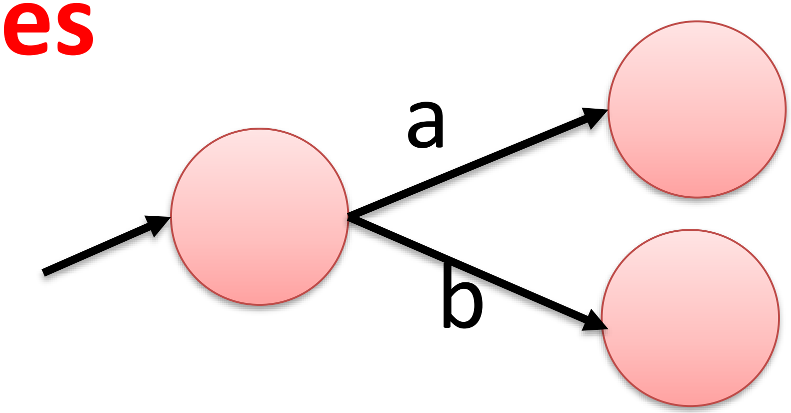


Inputs: s,r,g,b,a;
Outputs: u

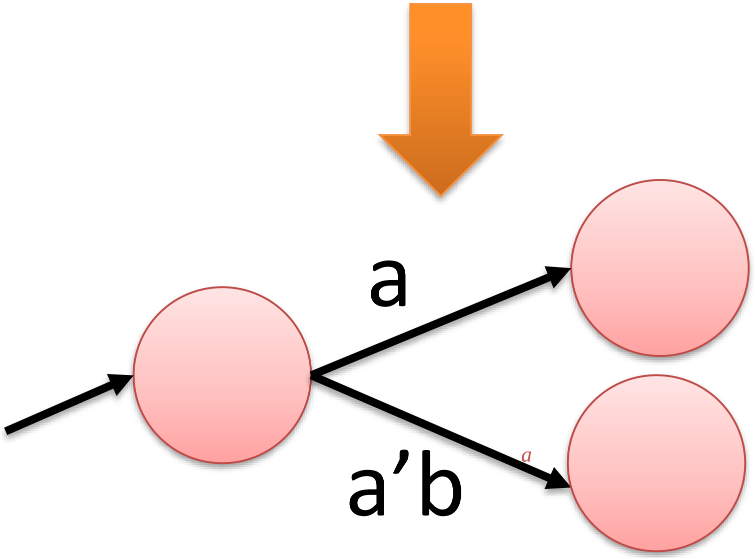
- New transition conditions detect if wrong button pressed, returns to "Wait"
- FSM provides formal, concrete means to accurately define desired behavior

Common Pitfalls Regarding Transition Properties

- *Only* one condition should be true
  - For all transitions leaving a state
  - Else, which one?

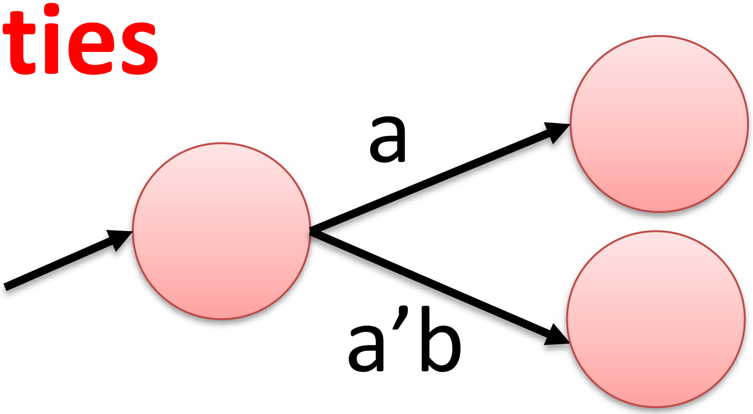


ab=11
Next state?

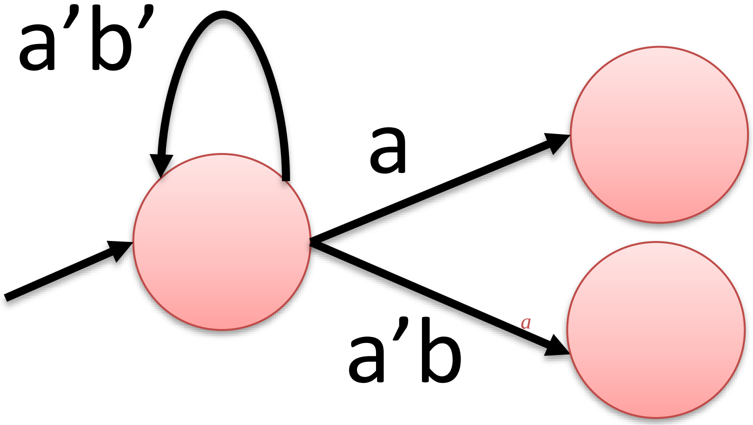


Common Pitfalls Regarding Transition Properties

- *One* condition must be true
  - For all transitions leaving a state
  - Else, where go?

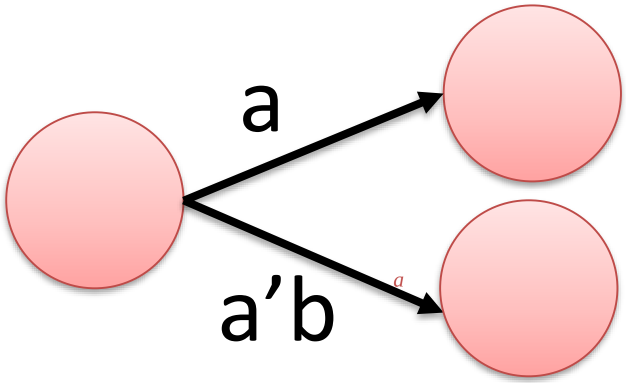


What if ab=00?



Verifying Correct Transition Properties

- Can verify using Boolean algebra
  - Only one condition true: AND of each condition pair (for transitions leaving a state) should equal 0
  - → proves pair can never simultaneously be true
  - Example



Answer:

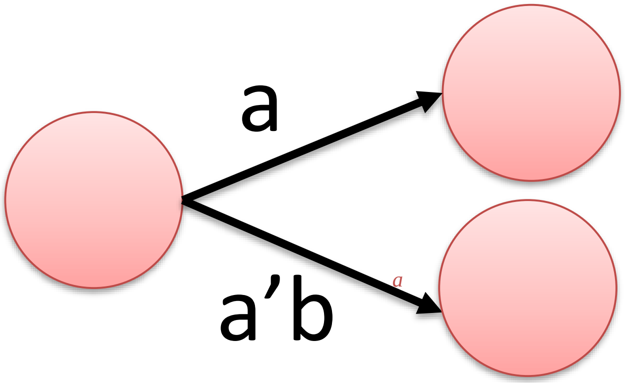
a * a’b
= (a * a’) * b
= 0 * b
= 0
OK!

Verifying Correct Transition Properties

- Can verify using Boolean algebra
  - One condition true: OR of all conditions of transitions leaving a state) should equal 1

a

  - → proves at least one condition must be true
  - Example



a + a’b
= a*(1+b) + a’b
= a + ab + a’b
= a + (a+a’)b
= a + b

Fails! Might not be 1 (i.e., a=0, b=0)

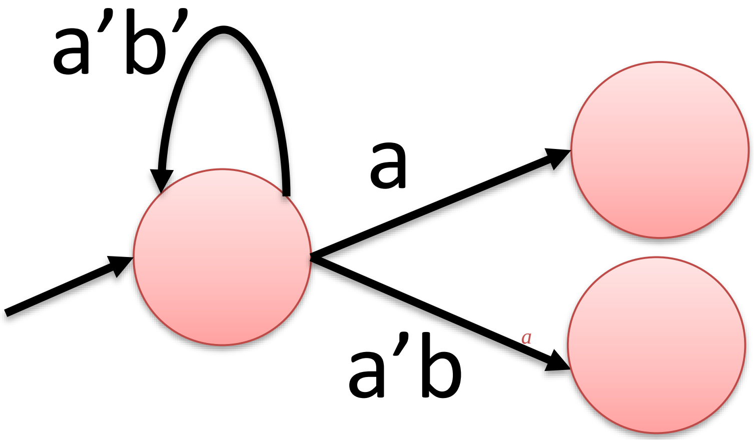
Verifying Correct Transition Properties

- Can verify using Boolean algebra
  - Only one condition true : among all pairs of transition from a state
  - One condition true : All the transitions from a state

Q: For shown transitions, prove whether:

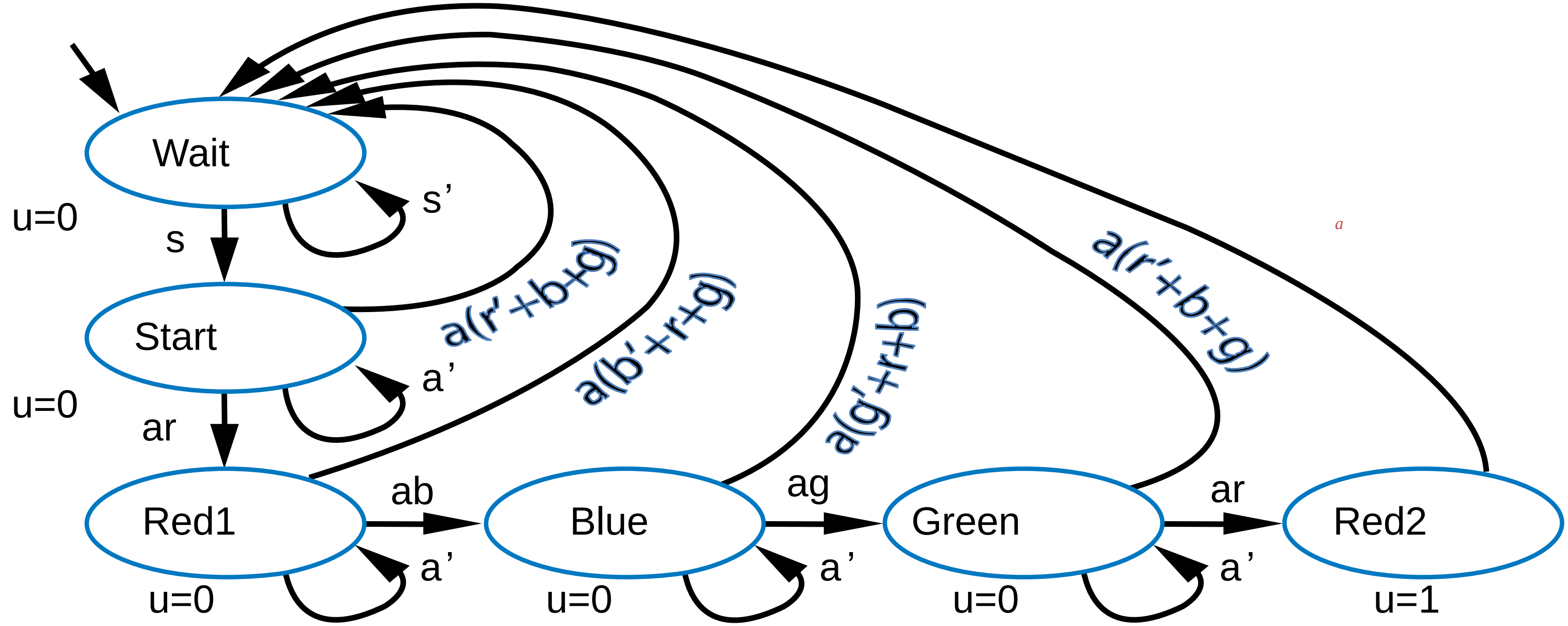
* Only one condition true (AND of each pair is always 0)

* One condition true (OR of all transitions is always 1)



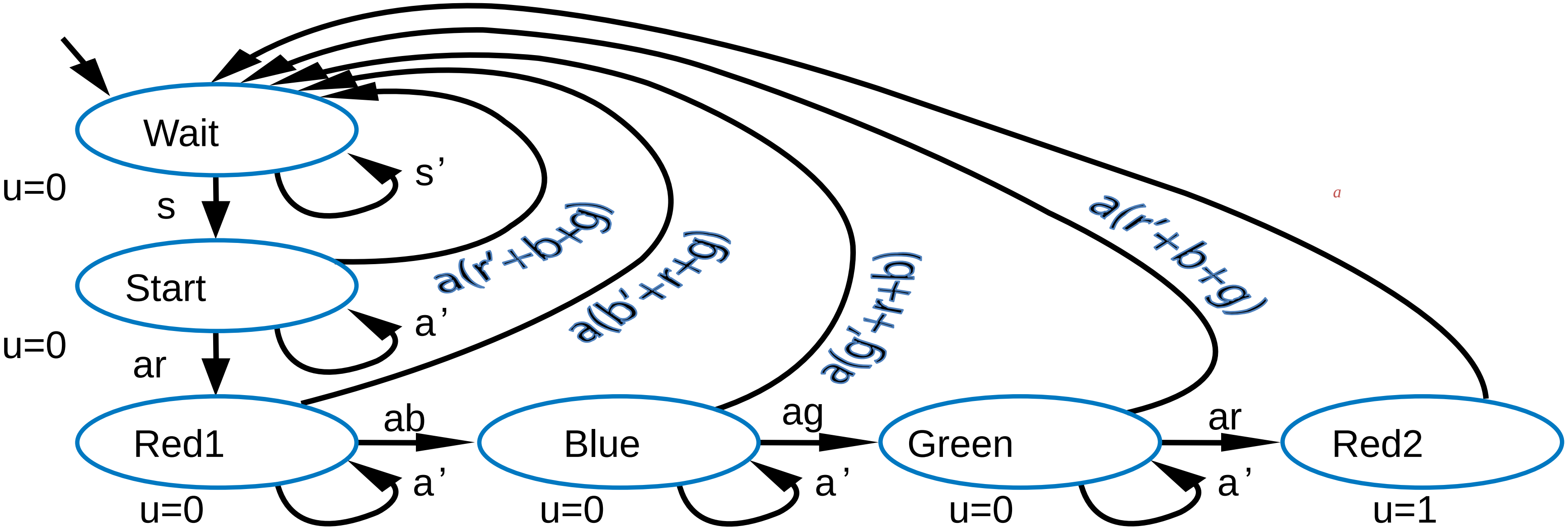
a + a’b +a’b’ = 1

Evidence that Pitfall is Common



- Recall code detector FSM
  - We “fixed” a problem with the transition conditions
  - Do the transitions obey the two required transition properties?
    - Consider transitions of state *Start*, and the “only one true” property

Evidence that Pitfall is Common



Consider transitions of state *Start*, and "only one true" property

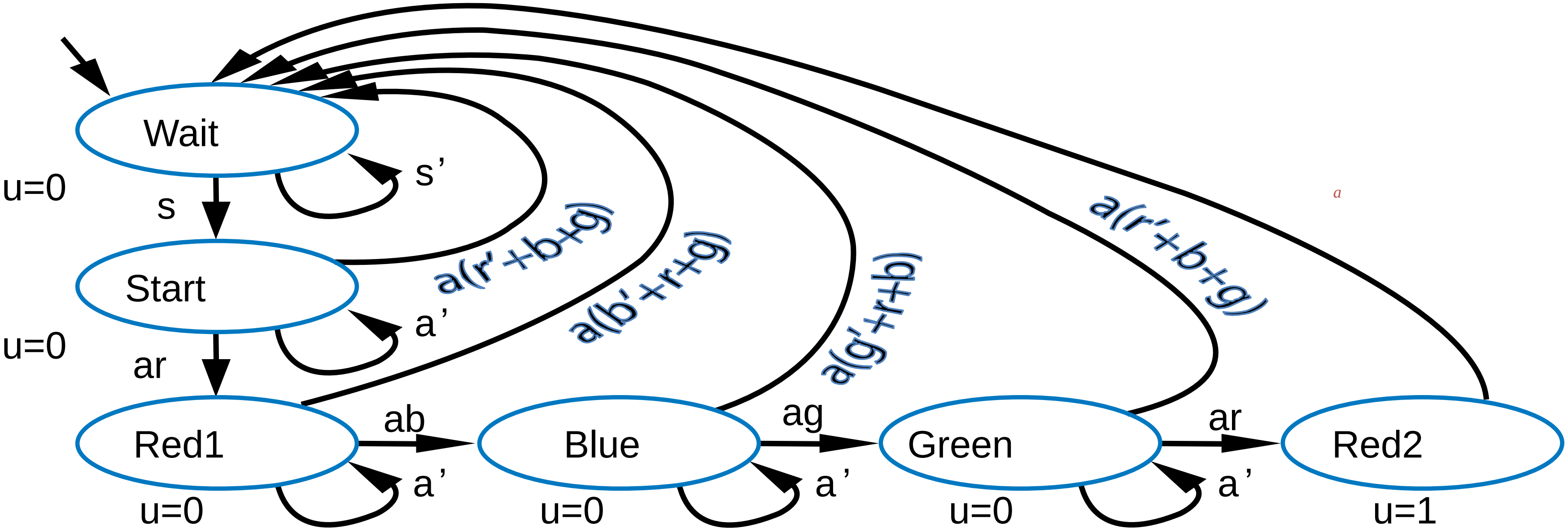
ar * a'
= (a*a')r
= 0

a' * a(r'+b+g)
= 0*r
= 0

ar * a(r'+b+g)
= (a'*a)*(r'+b+g) = 0*(r'+b+g)
= (a*a)*r*(r'+b+g) = *r*(r'+b+g)
= arr'+arb+arg
= 0 + arb+arg
= arb + arg
= ar(b+g) **// not ZERO**

Fails! Means that two of Start's transitions could be true

Evidence that Pitfall is Common



Consider transitions of state *Start*, and “only one true” property

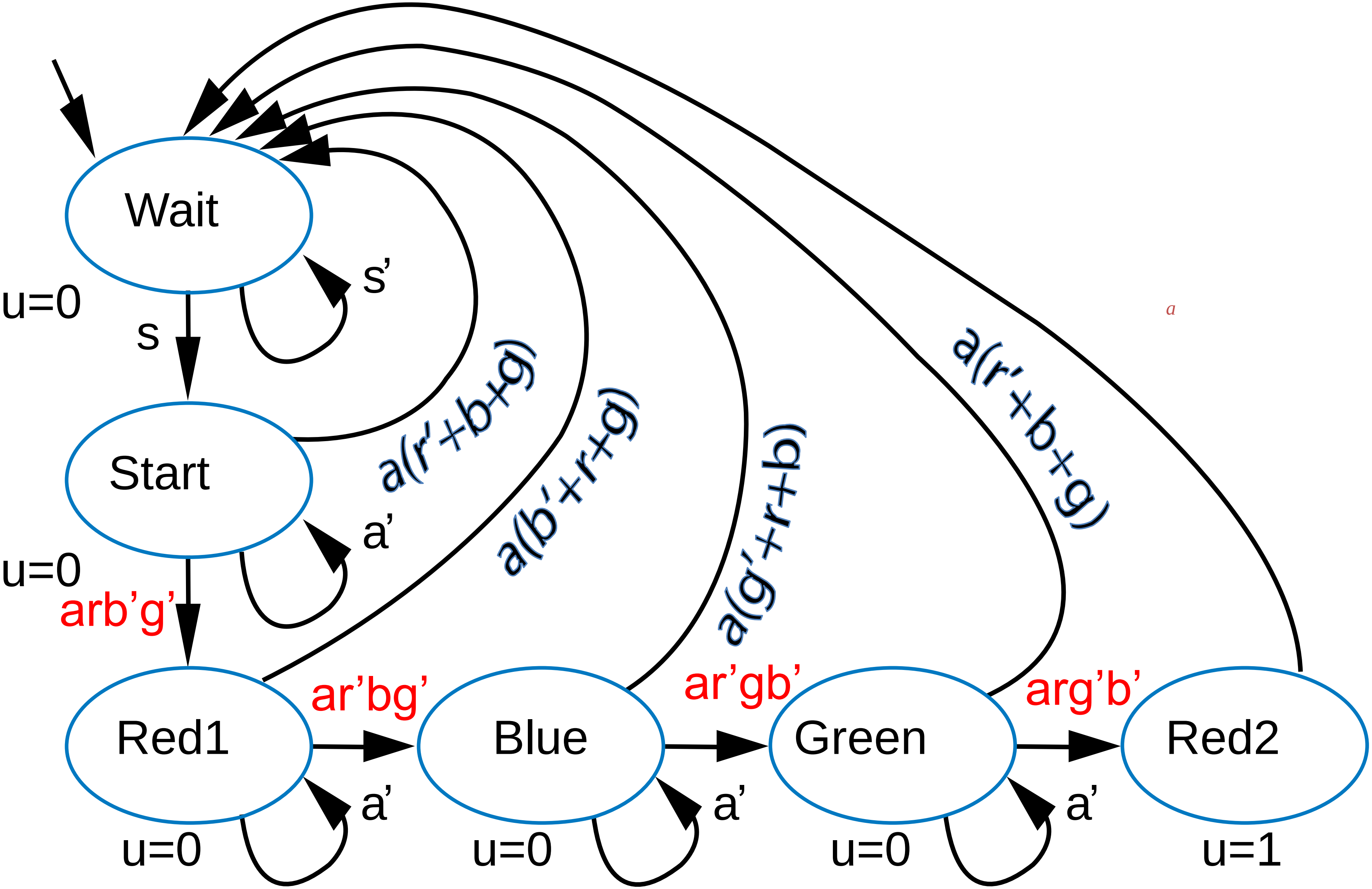
Intuitively: press red and blue buttons at same time: conditions ar, and a(r’+b+g) will both be true. Which one should be taken?

Q: How to solve?

A: ar should be arb’g’ (likewise for ab, ag, ar)

arb’g’ * a(r’+b+g) = 0

Evidence that Pitfall is Common


Wait
u=0
s'
s
a(r'+b+g)
a(b'+r+g)
a(g'+r+b)
a(r'+b+g)
a
Start
a'
u=0
arb'g'
Red1
ar'bg'
Blue
ar'gb'
Green
arg'b'
Red2
a'
a'
a'
u=0
u=0
u=0
u=1