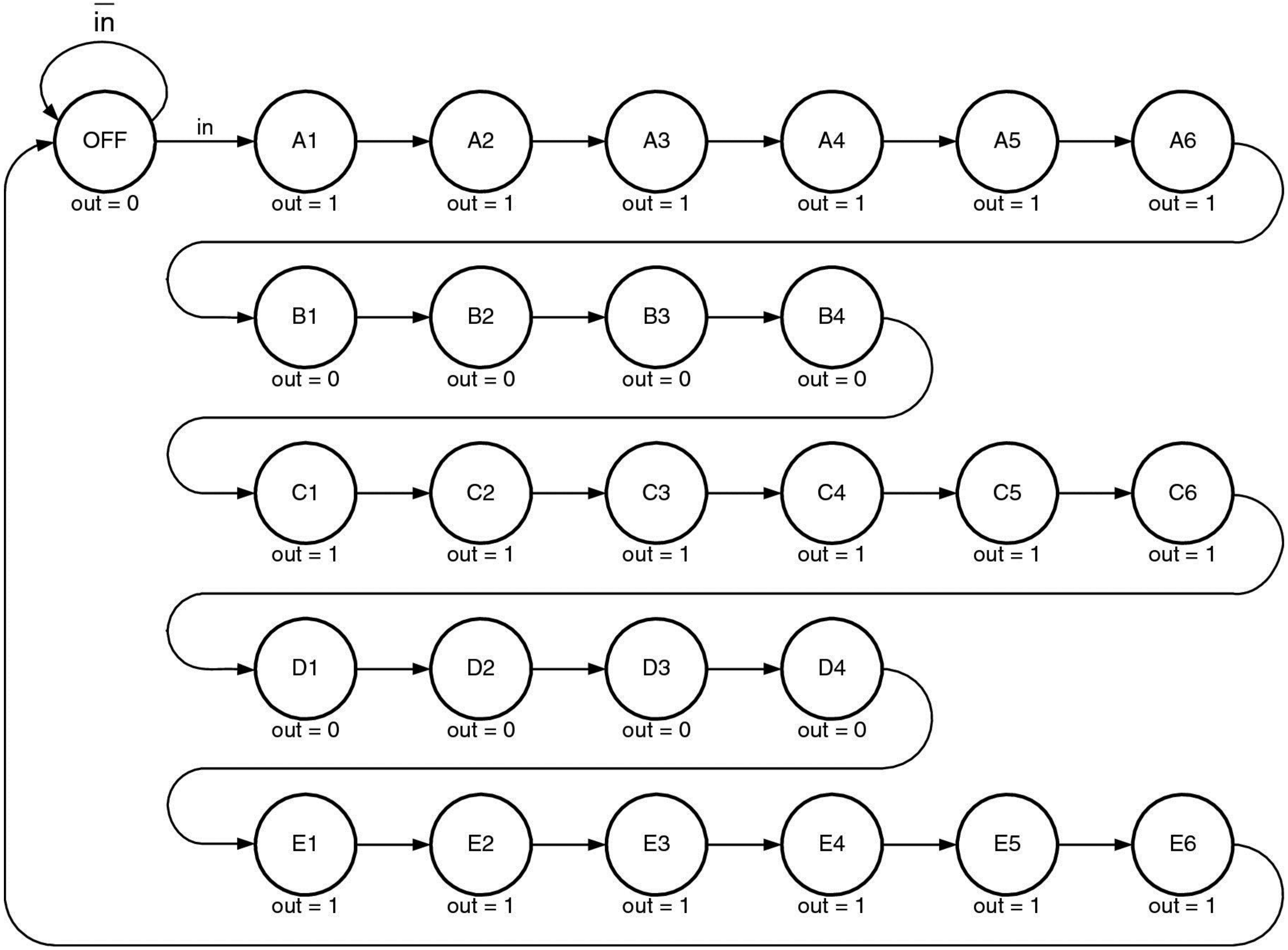
Factoring Finite-State Machines

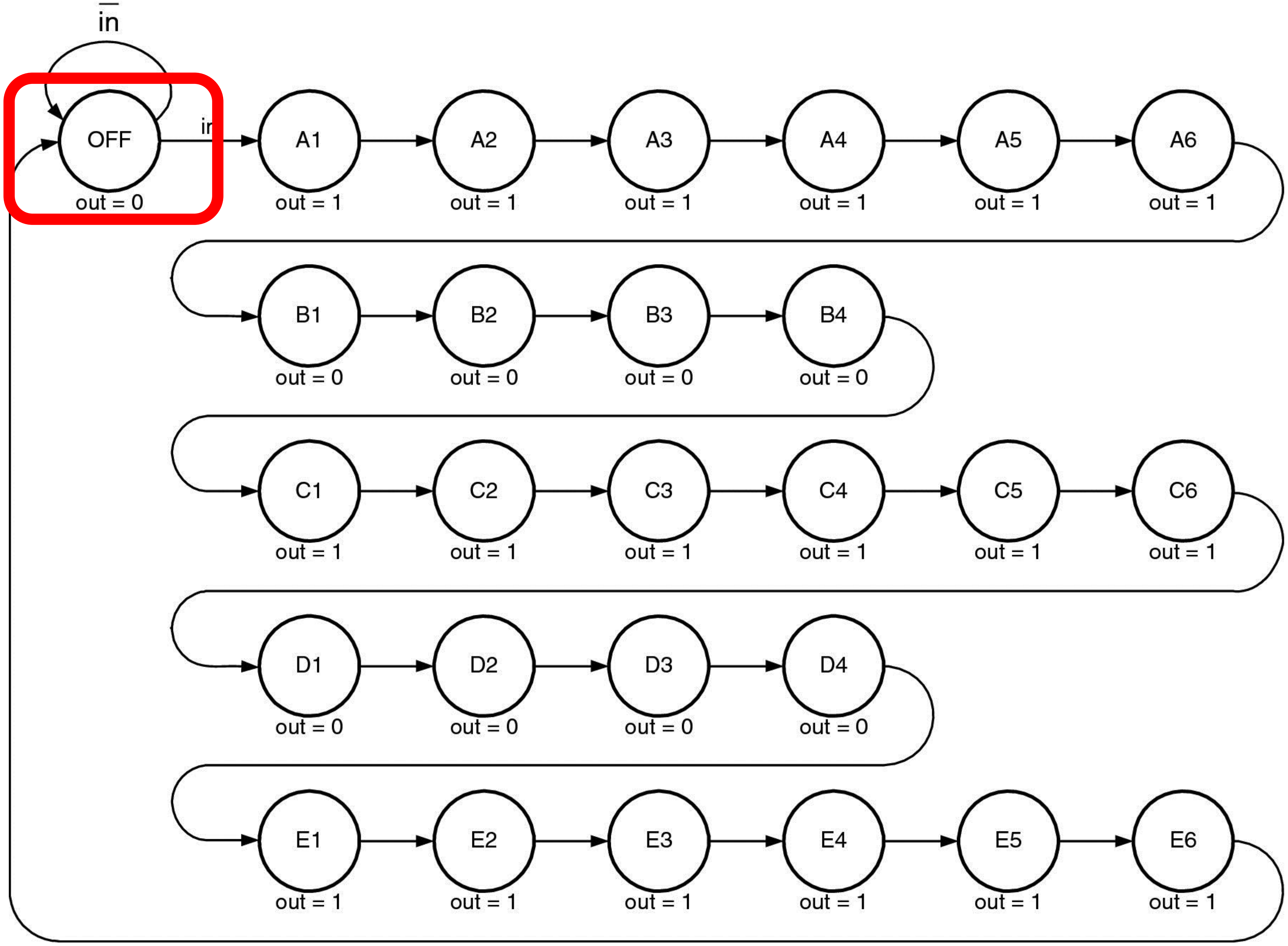
Factoring Finite-State Machines

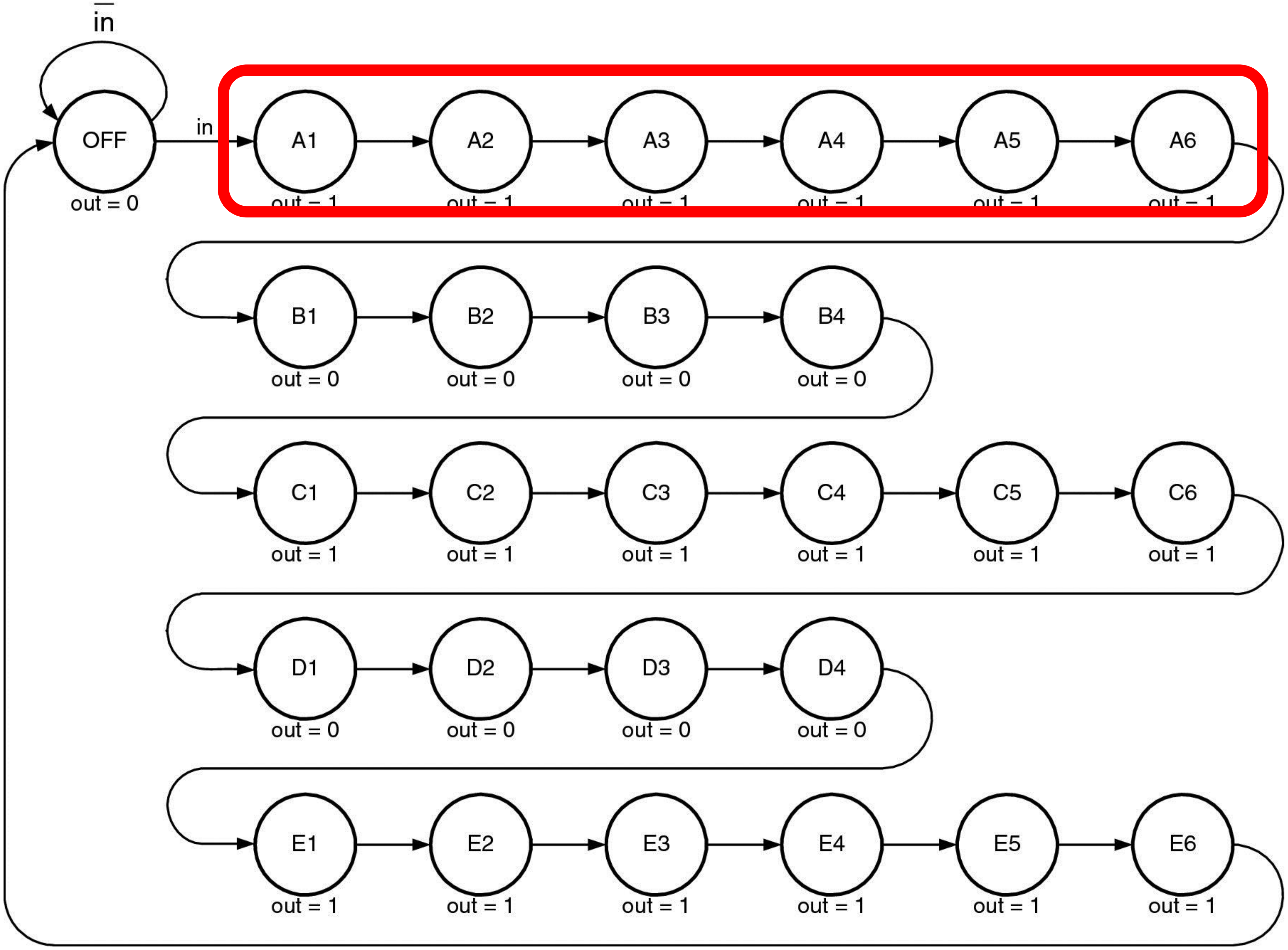
- Splitting a larger, complex FSM
    - Into two or more smaller, less complex FSM
- Each state of the sub-machine represents one dimension or factor of the larger machine
- For example, One portion can be data, the other can be control

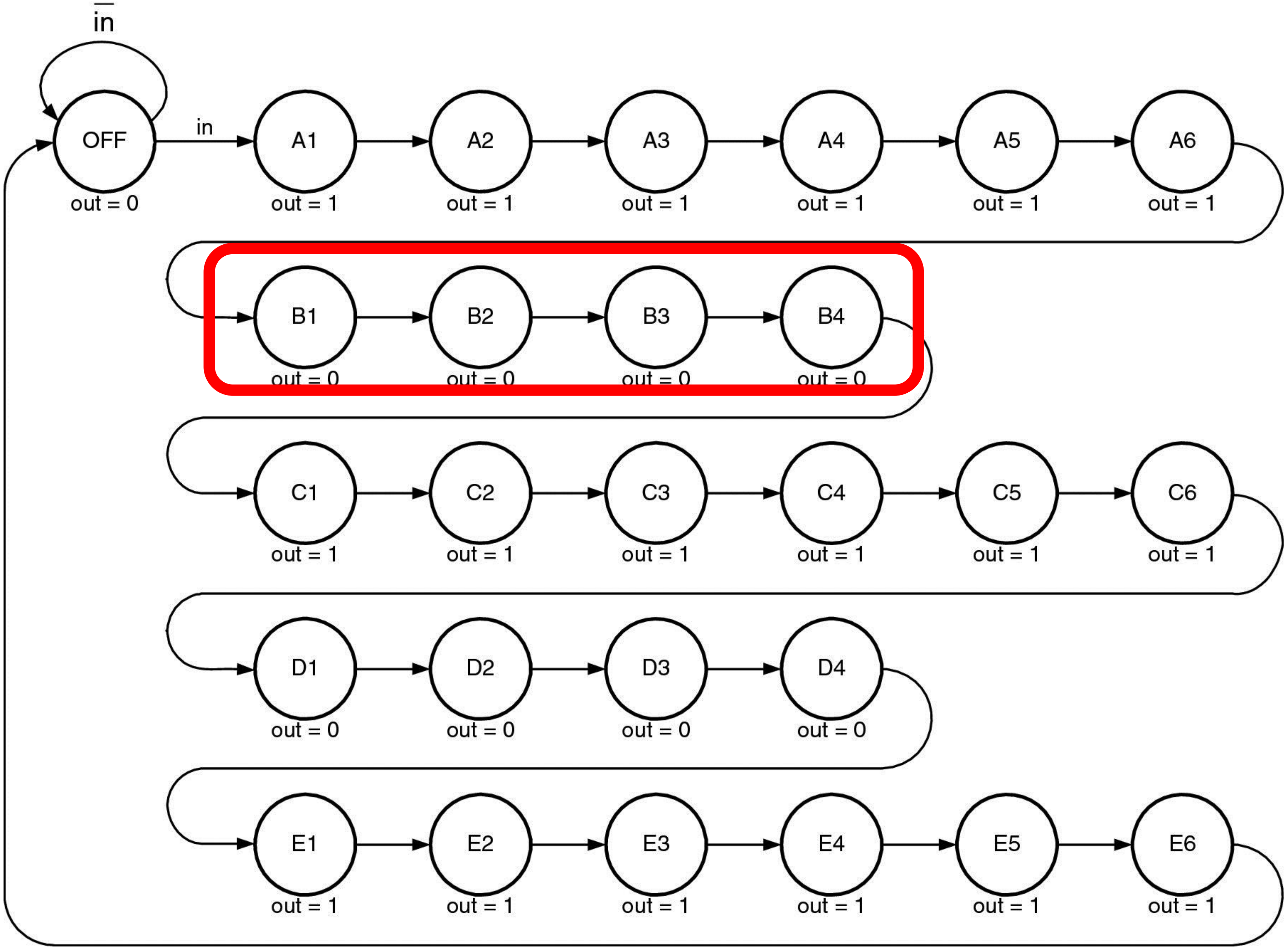
Example 1: Flashing Light

- Single input in
- Single output out (LED)
- Starts at OFF state
- LED hot for Six Cycles
- LED dark for four cycles
- LED hot for six cycles
- LED dark for four cycles
- LED hot for six cycles
- Returns to OFF

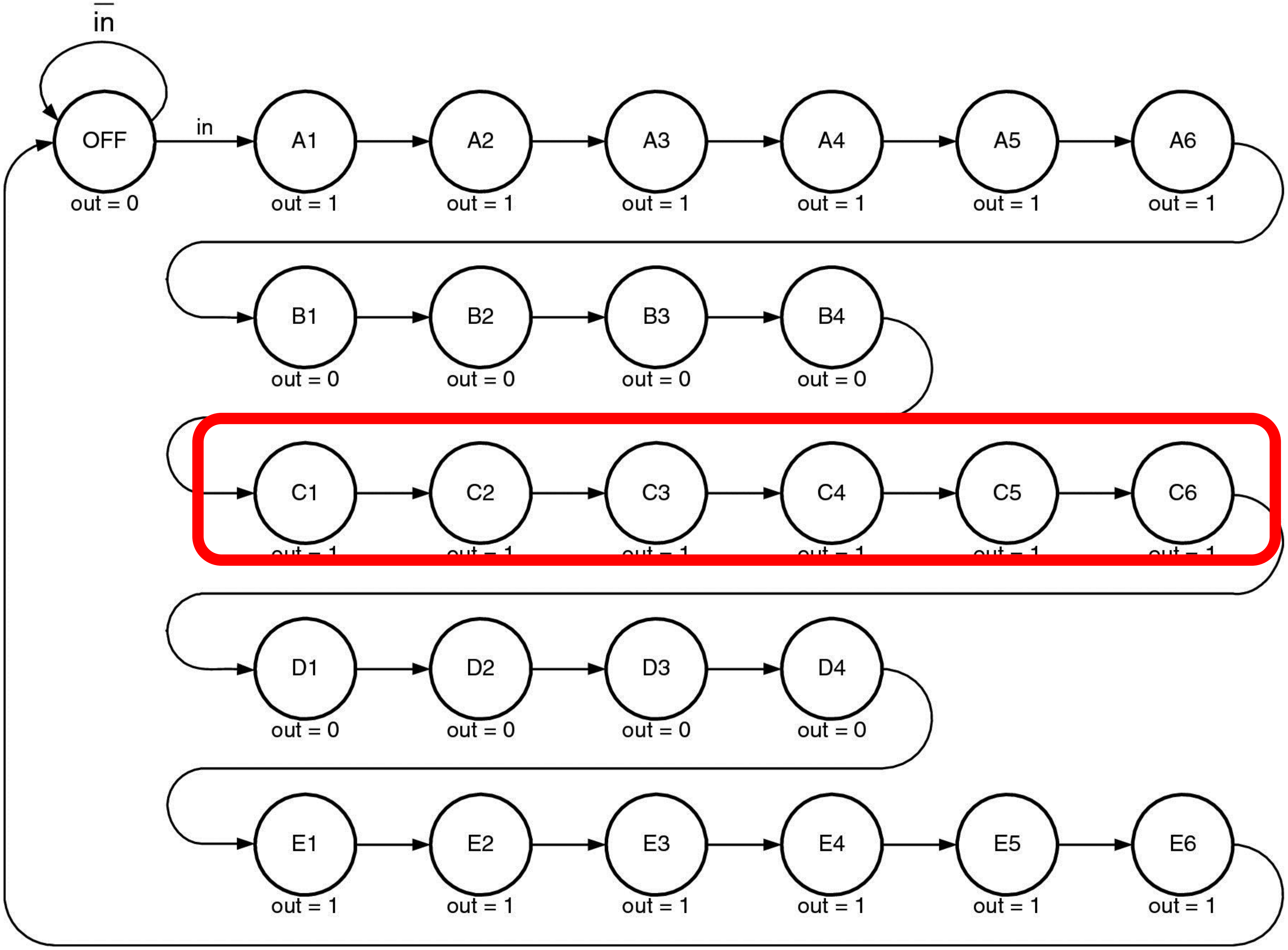

$\overline{\text{in}}$
in
OFF
out = 0
A1
out = 1
A2
out = 1
A3
out = 1
A4
out = 1
A5
out = 1
A6
out = 1
B1
out = 0
B2
out = 0
B3
out = 0
B4
out = 0
C1
out = 1
C2
out = 1
C3
out = 1
C4
out = 1
C5
out = 1
C6
out = 1
D1
out = 0
D2
out = 0
D3
out = 0
D4
out = 0
E1
out = 1
E2
out = 1
E3
out = 1
E4
out = 1
E5
out = 1
E6
out = 1

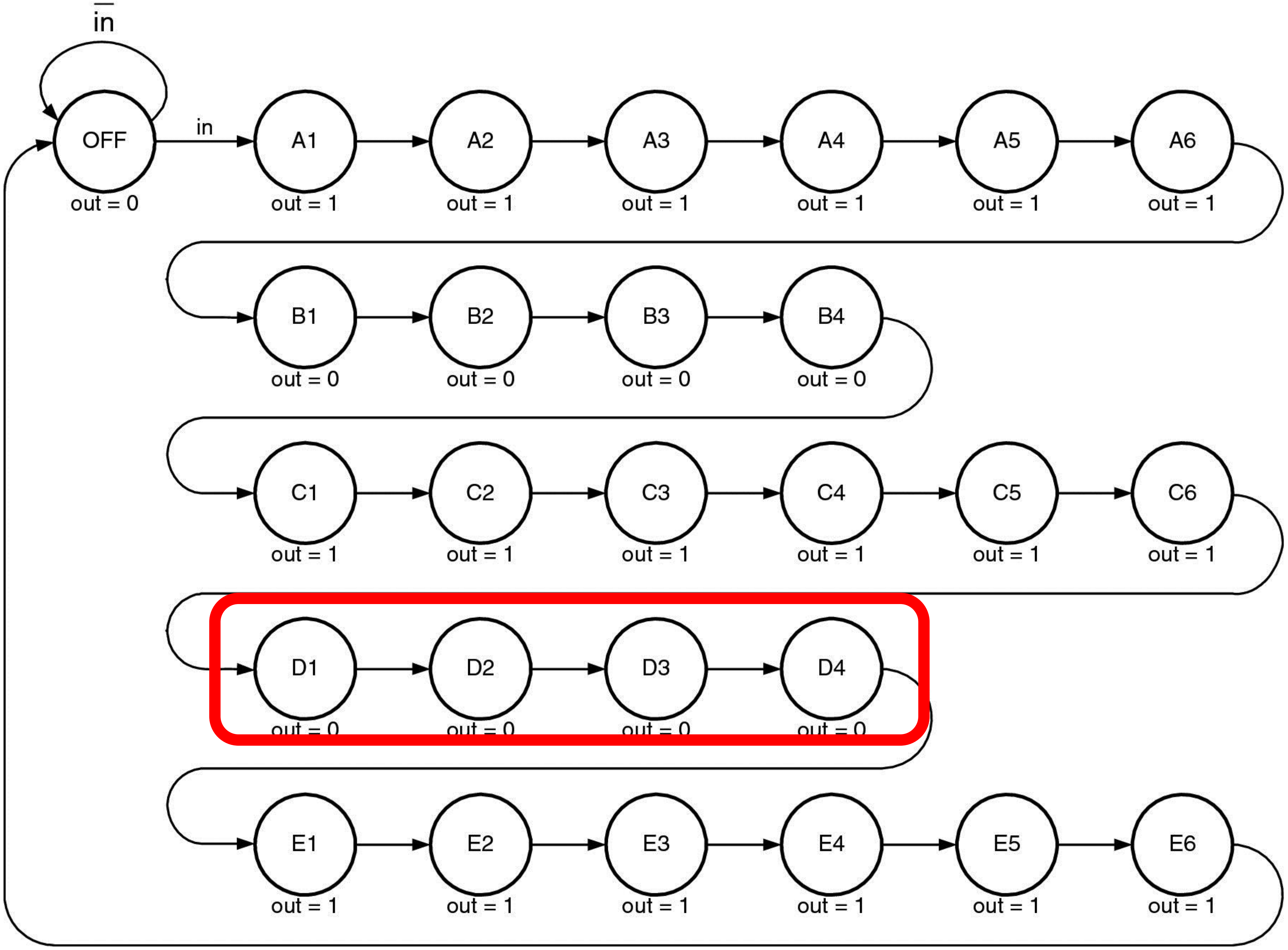

$\overline{\text{in}}$
in
OFF
out = 0
A1
out = 1
A2
out = 1
A3
out = 1
A4
out = 1
A5
out = 1
A6
out = 1
B1
out = 0
B2
out = 0
B3
out = 0
B4
out = 0
C1
out = 1
C2
out = 1
C3
out = 1
C4
out = 1
C5
out = 1
C6
out = 1
D1
out = 0
D2
out = 0
D3
out = 0
D4
out = 0
E1
out = 1
E2
out = 1
E3
out = 1
E4
out = 1
E5
out = 1
E6
out = 1

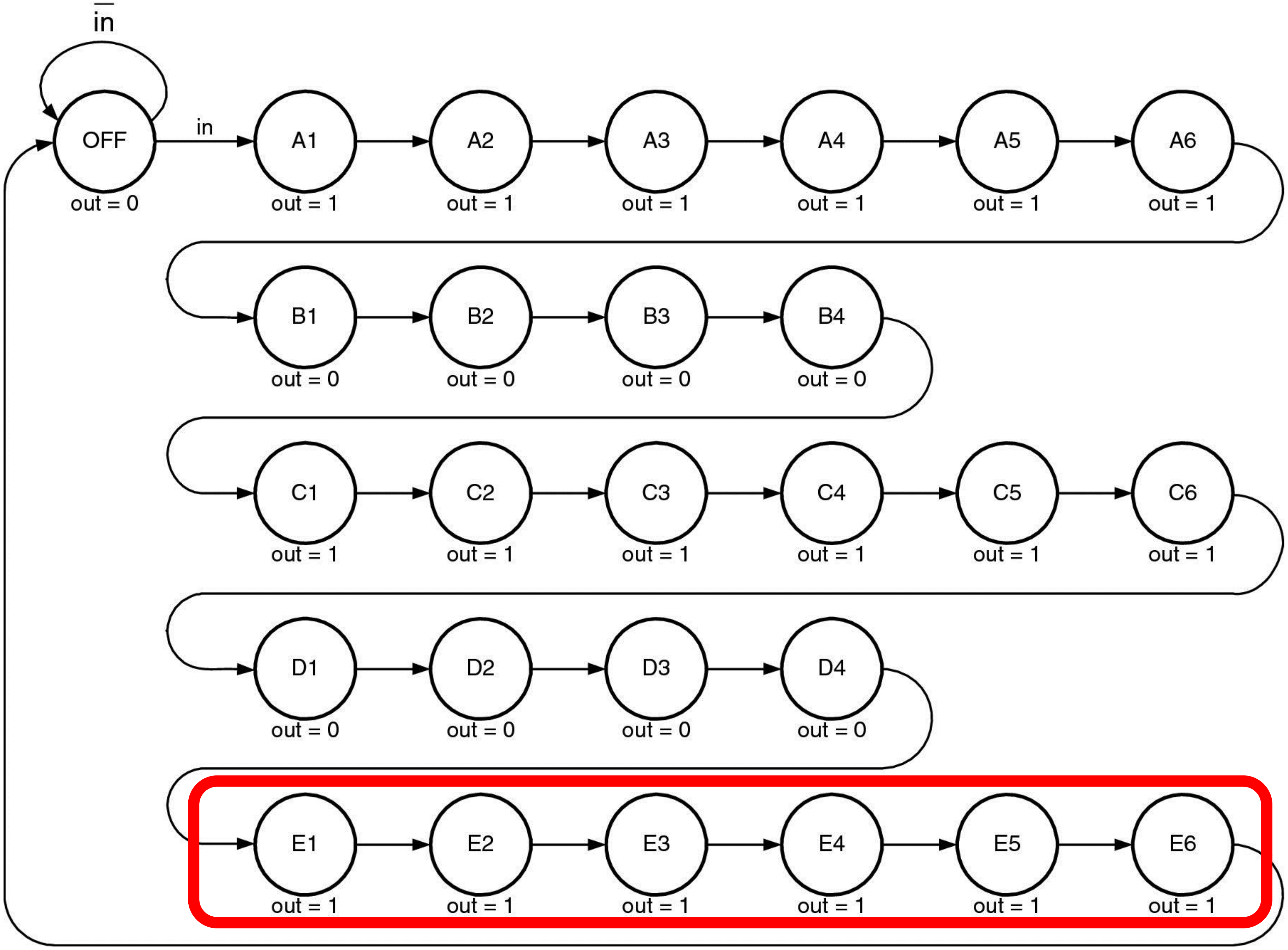

$\overline{in}$
OFF
out = 0
in
A1
out = 1
A2
out = 1
A3
out = 1
A4
out = 1
A5
out = 1
A6
out = 1
B1
out = 0
B2
out = 0
B3
out = 0
B4
out = 0
C1
out = 1
C2
out = 1
C3
out = 1
C4
out = 1
C5
out = 1
C6
out = 1
D1
out = 0
D2
out = 0
D3
out = 0
D4
out = 0
E1
out = 1
E2
out = 1
E3
out = 1
E4
out = 1
E5
out = 1
E6
out = 1



$\overline{\text{in}}$
in
OFF
out = 0
A1
out = 1
A2
out = 1
A3
out = 1
A4
out = 1
A5
out = 1
A6
out = 1
B1
out = 0
B2
out = 0
B3
out = 0
B4
out = 0
C1
out = 1
C2
out = 1
C3
out = 1
C4
out = 1
C5
out = 1
C6
out = 1
D1
out = 0
D2
out = 0
D3
out = 0
D4
out = 0
E1
out = 1
E2
out = 1
E3
out = 1
E4
out = 1
E5
out = 1
E6
out = 1


$\overline{\text{in}}$
in
OFF
out = 0
A1
out = 1
A2
out = 1
A3
out = 1
A4
out = 1
A5
out = 1
A6
out = 1
B1
out = 0
B2
out = 0
B3
out = 0
B4
out = 0
C1
out = 1
C2
out = 1
C3
out = 1
C4
out = 1
C5
out = 1
C6
out = 1
D1
out = 0
D2
out = 0
D3
out = 0
D4
out = 0
E1
out = 1
E2
out = 1
E3
out = 1
E4
out = 1
E5
out = 1
E6
out = 1


$\overline{in}$
in
OFF
out = 0
A1
out = 1
A2
out = 1
A3
out = 1
A4
out = 1
A5
out = 1
A6
out = 1
B1
out = 0
B2
out = 0
B3
out = 0
B4
out = 0
C1
out = 1
C2
out = 1
C3
out = 1
C4
out = 1
C5
out = 1
C6
out = 1
D1
out = 0
D2
out = 0
D3
out = 0
D4
out = 0
E1
out = 1
E2
out = 1
E3
out = 1
E4
out = 1
E5
out = 1
E6
out = 1

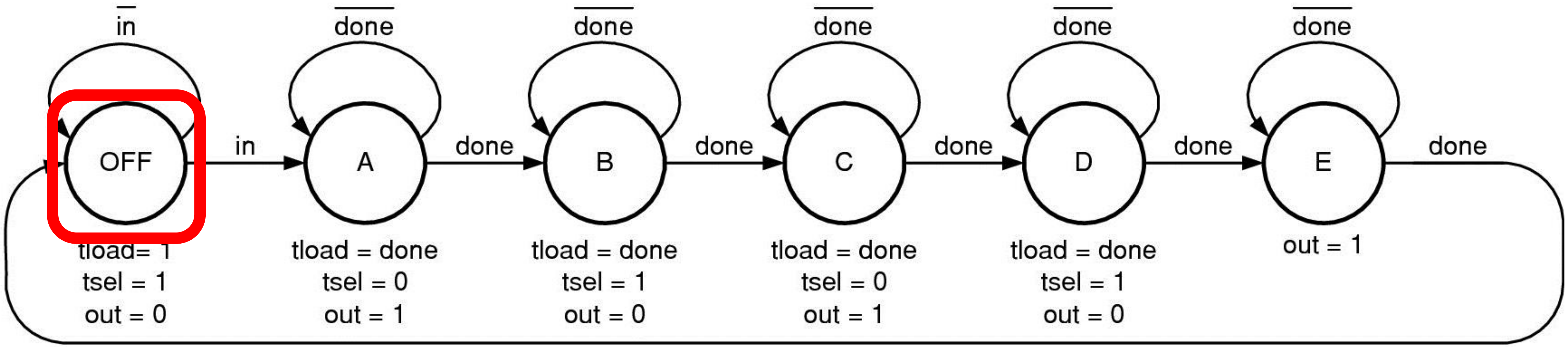

in
in
OFF
out = 0
A1
out = 1
A2
out = 1
A3
out = 1
A4
out = 1
A5
out = 1
A6
out = 1
B1
out = 0
B2
out = 0
B3
out = 0
B4
out = 0
C1
out = 1
C2
out = 1
C3
out = 1
C4
out = 1
C5
out = 1
C6
out = 1
D1
out = 0
D2
out = 0
D3
out = 0
D4
out = 0
E1
out = 1
E2
out = 1
E3
out = 1
E4
out = 1
E5
out = 1
E6
out = 1

Example 1: Flashing Light

- Workable
- But is not Simple, and is rather Stupid
- Ideal state machines should have as few parts as possible
- Ideal state machines should have as few decisions as possible
- The state machine is a single, long chain.
- To change the timing, the entire state machine would have to be rewritten.

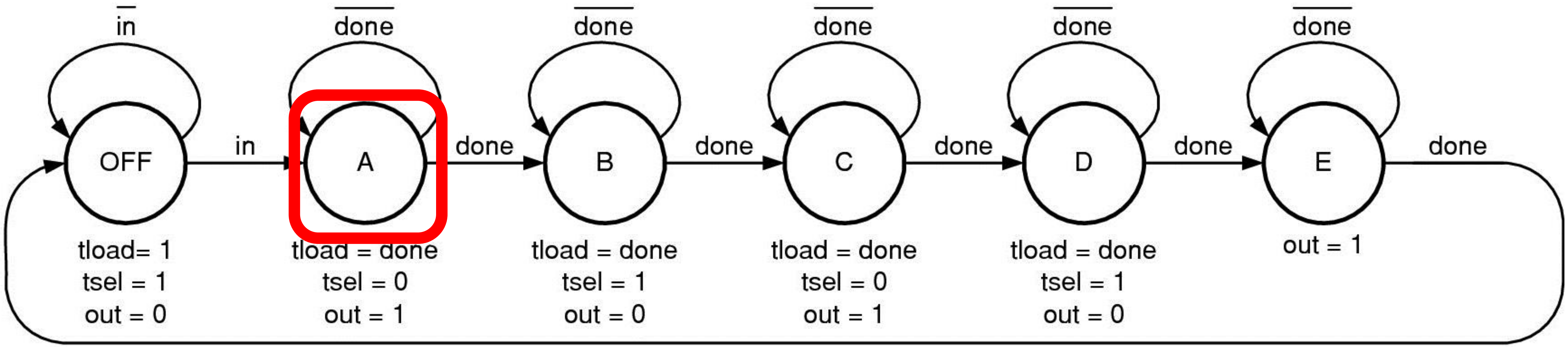
Example 1: Flashing Light

- First, add a Timer with a Load (Chapter 16) to each of the on-off parts of the flash.



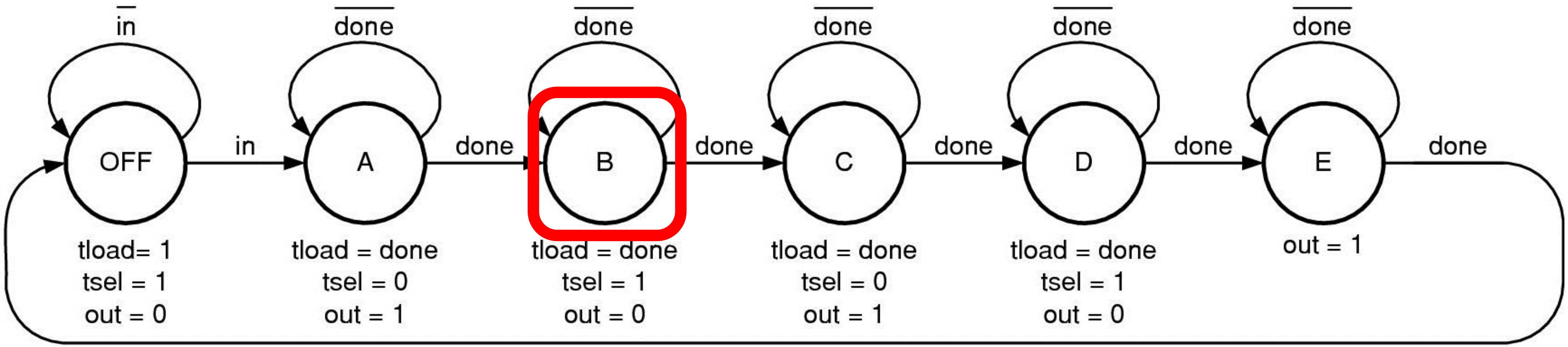
Example 1: Flashing Light

- First, add a Timer with a Load (Chapter 16) to each of the on-off parts of the flash.



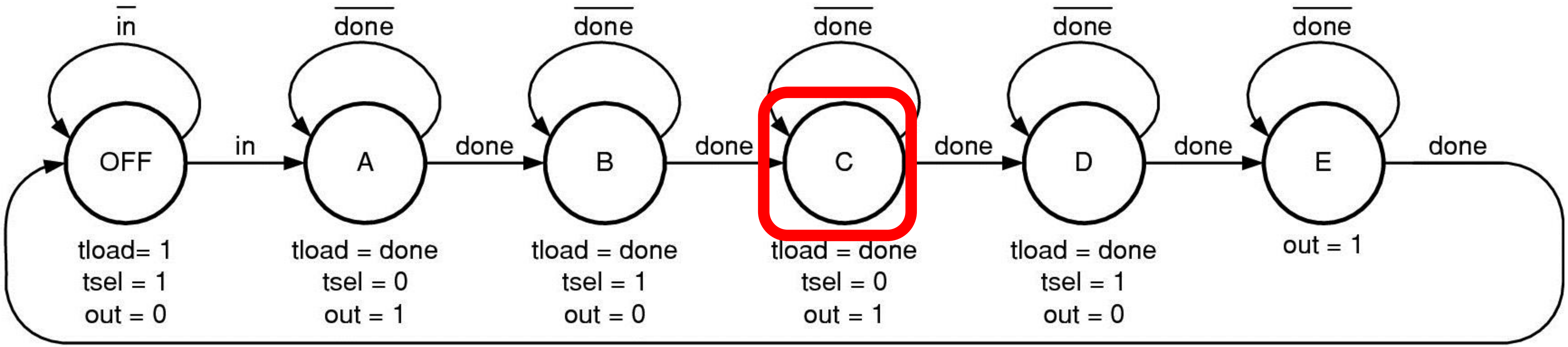
Example 1: Flashing Light

- First, add a Timer with a Load (Chapter 16) to each of the on-off parts of the flash.



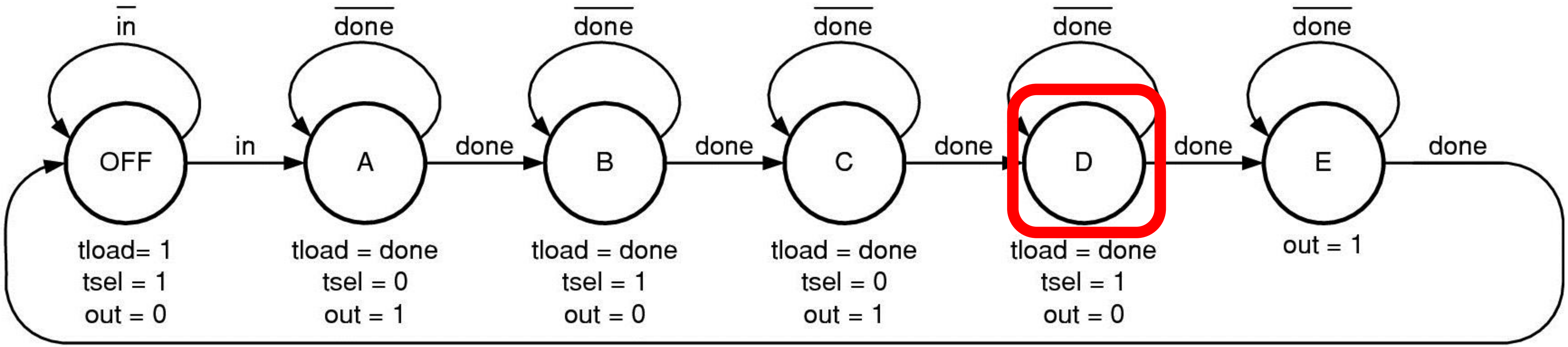
Example 1: Flashing Light

- First, add a Timer with a Load (Chapter 16) to each of the on-off parts of the flash.



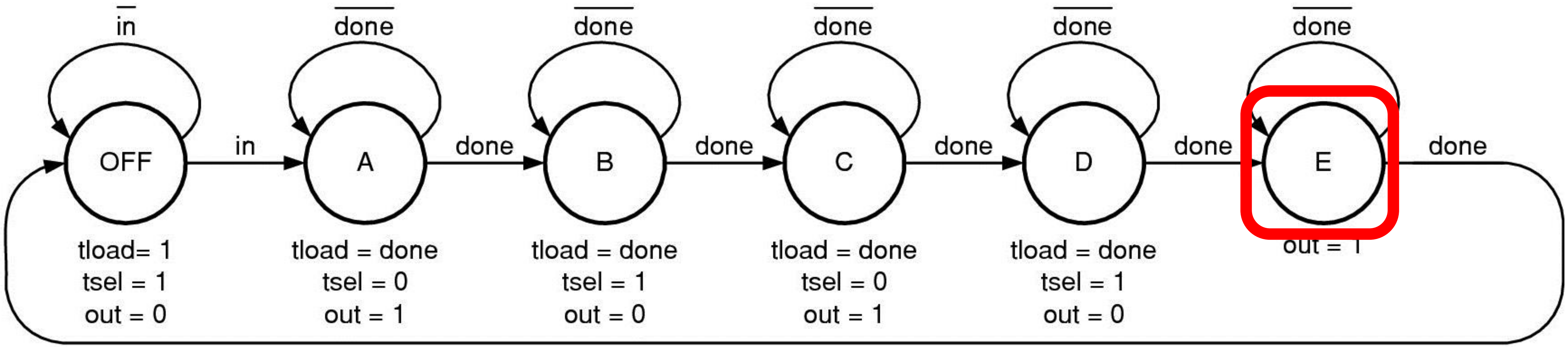
Example 1: Flashing Light

- First, add a Timer with a Load (Chapter 16) to each of the on-off parts of the flash.



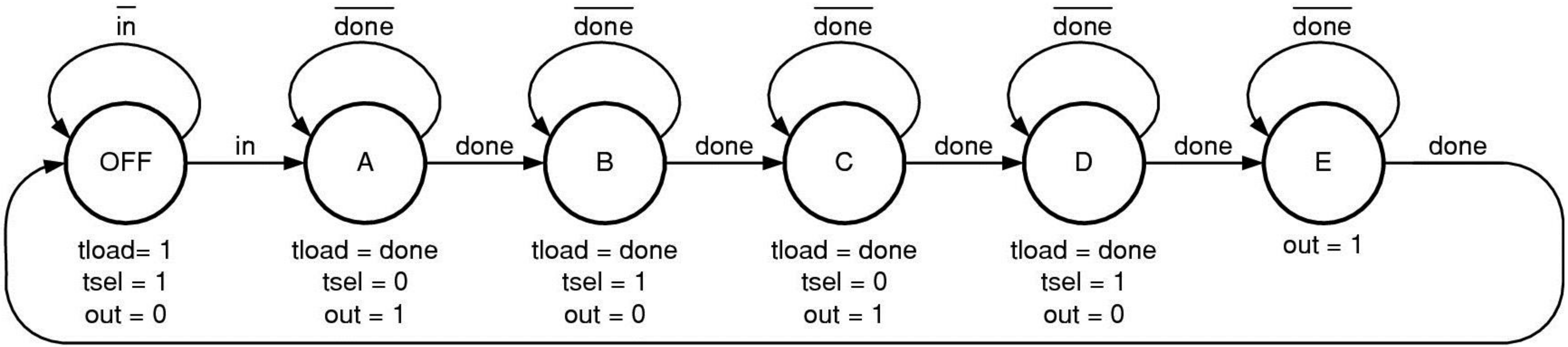
Example 1: Flashing Light

- First, add a Timer with a Load (Chapter 16) to each of the on-off parts of the flash.



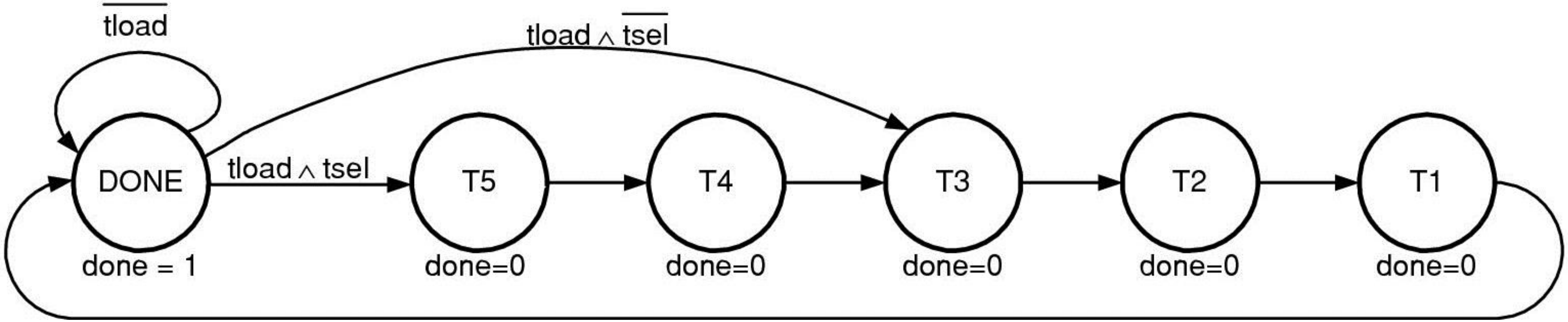
Example 1: Flashing Light

- tLoad to load the timer
- Tsel to say if it is 6 cycles or 4 Cycles
- Out to say LED on or off.



Example 1: Flashing Light

- Replace the sequence with a timer with a load.

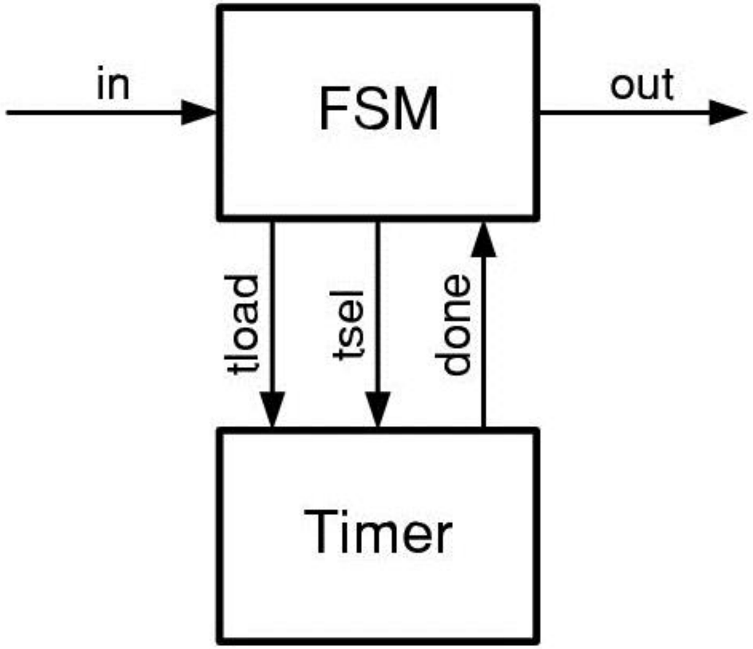


$\overline{\text{tload}}$
$\text{tload} \wedge \overline{\text{tsel}}$
$\text{tload} \wedge \text{tsel}$
DONE
T5
T4
T3
T2
T1
done = 1
done=0
done=0
done=0
done=0
done=0

If the Select is high, the LED is on, And count 5,4,3,2,1,0 for 6 hot
If the Select is low, the LED is off, Jump to T3, and the Count is 3,2,1,0 for 4 dark

Example 1: Flashing Light

- The system has two FSM
- The Control FSM says if the light is on or off.
- The Data FSM says how long the light is on or off, with the data being the countdown.



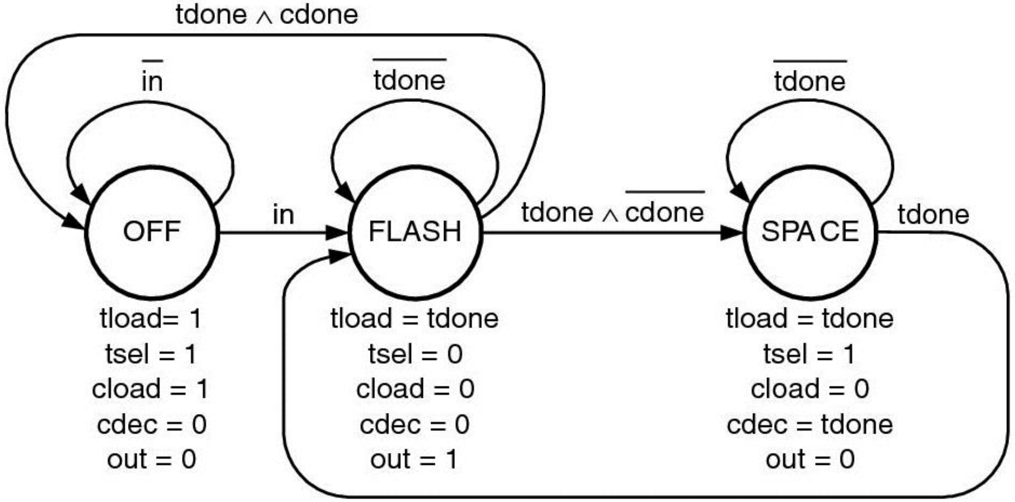
Example 1: Flashing Light

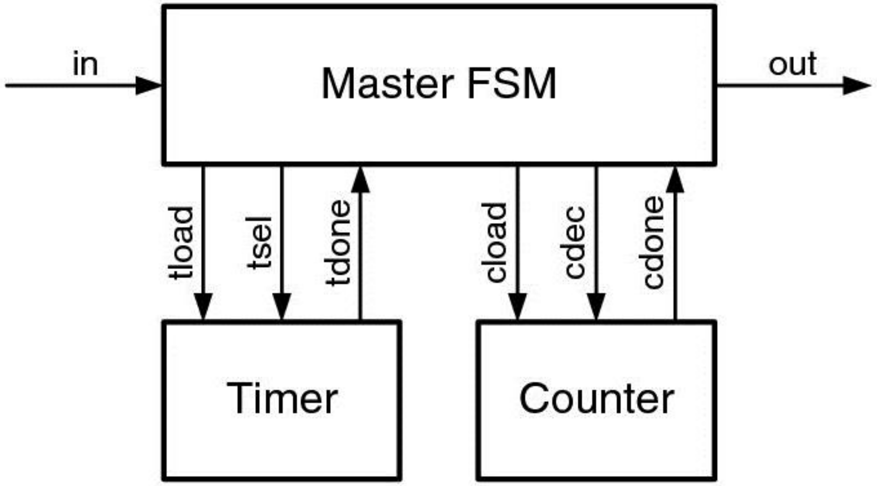
- Can it be reduced further?

Example 1: Flashing Light

- Yes.
- A counter can be added.
- The master state machine has three states: off, flash, space
- The Timer state machine will determine the length of the flash or the length of the space.
- The Counter can determine how many times the flash occurs.

Example 1: Flashing Light

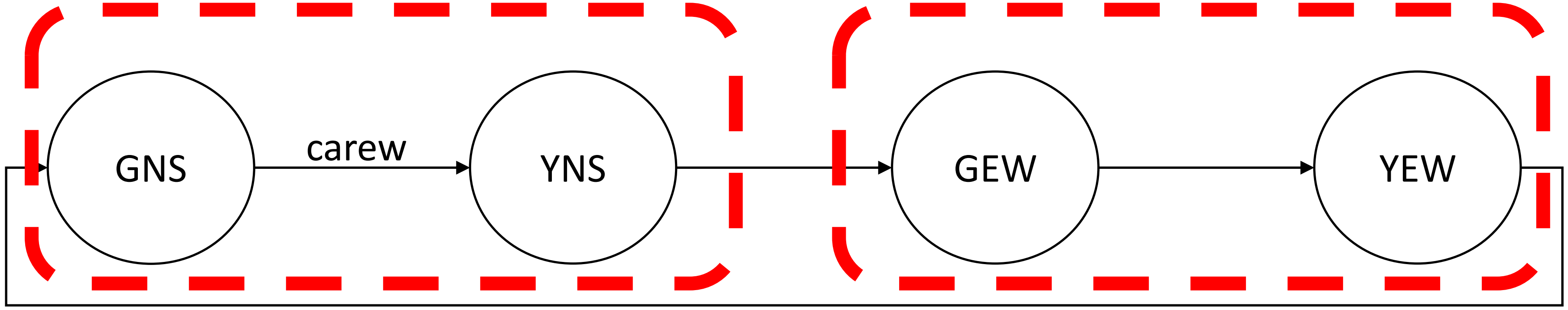




Verilog example available on Blackboard

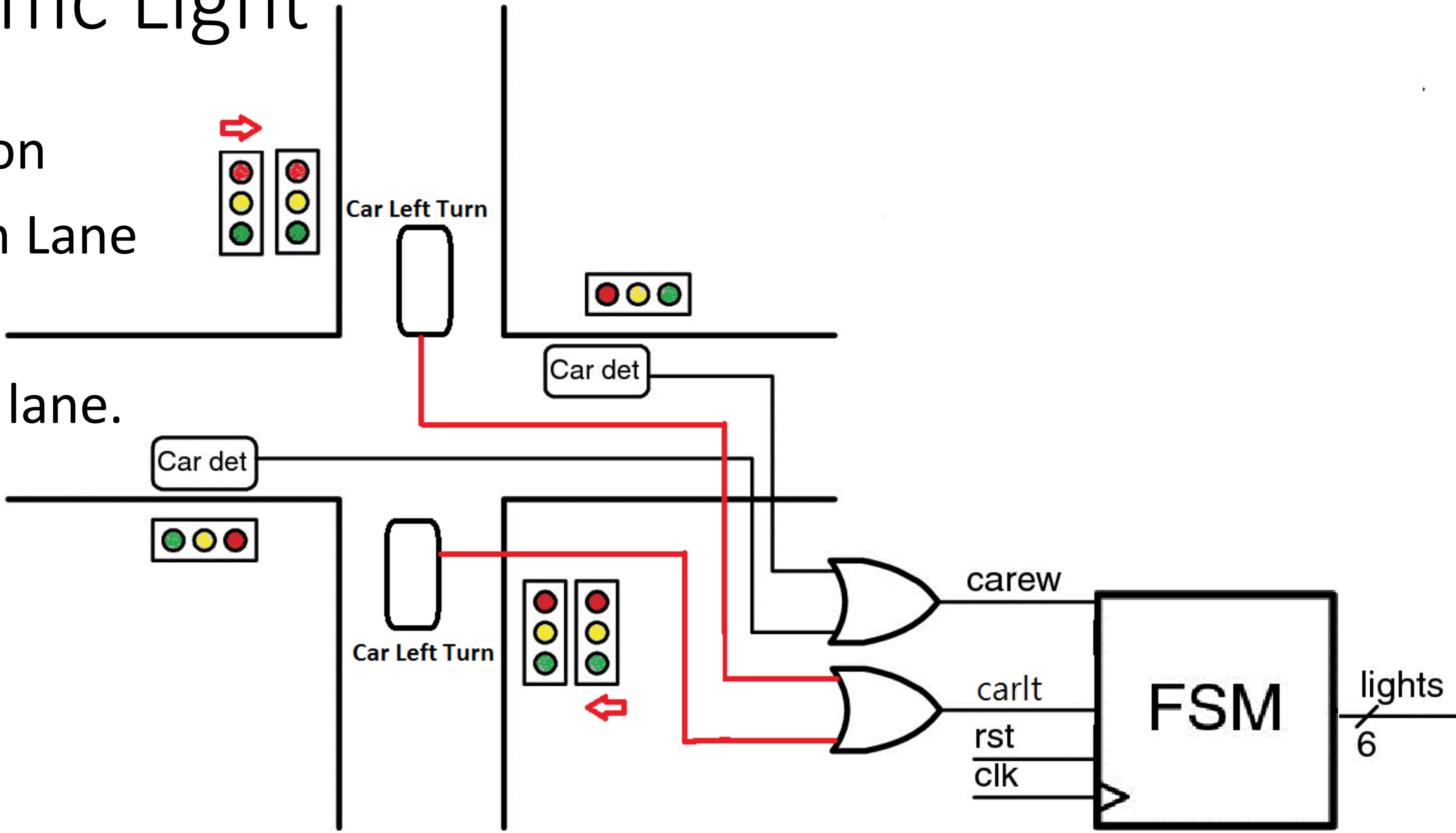
Back to Traffic Light

- Add a new transition to the Traffic Light problem.
- The original traffic light problem is a single timer instance.
- Could be broken up…well, not really. Too few conditions and states. Maybe 2 states?



Back to Traffic Light

- Add a new condition
- Car in the Left Turn Lane
- *carlt*
- in the north-south lane.

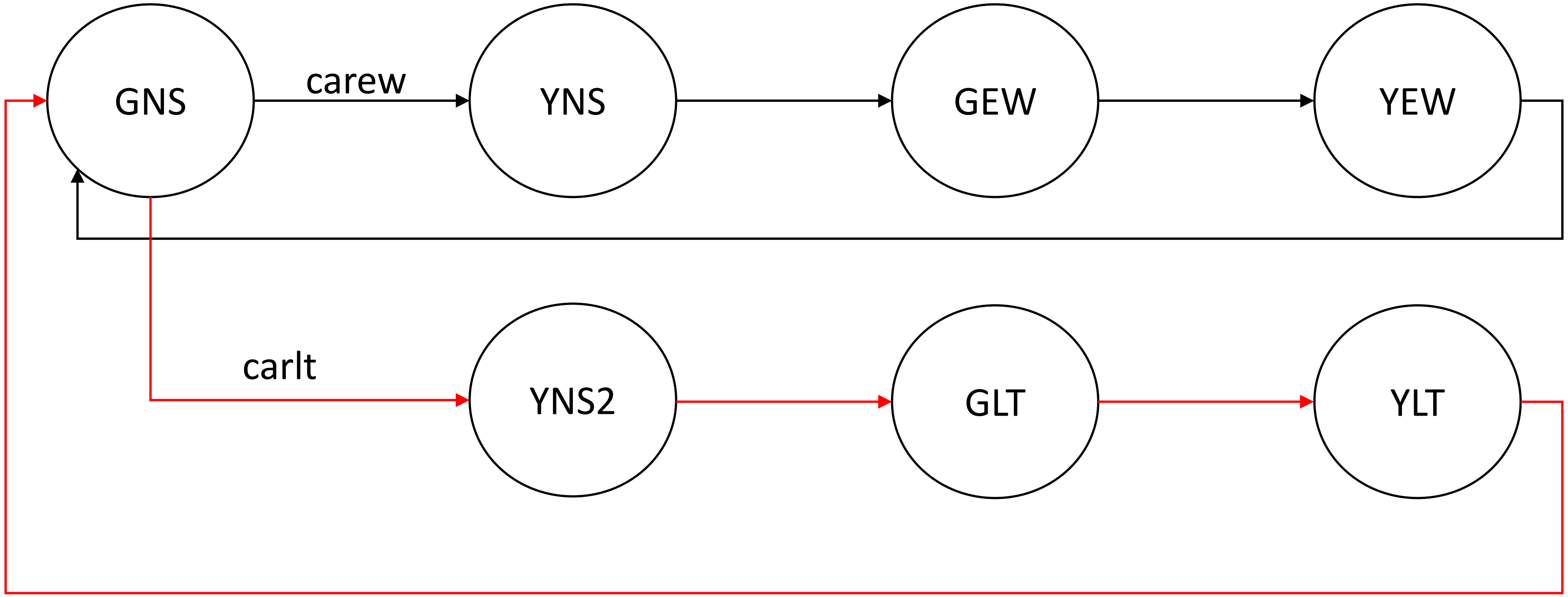


Back to Traffic Light

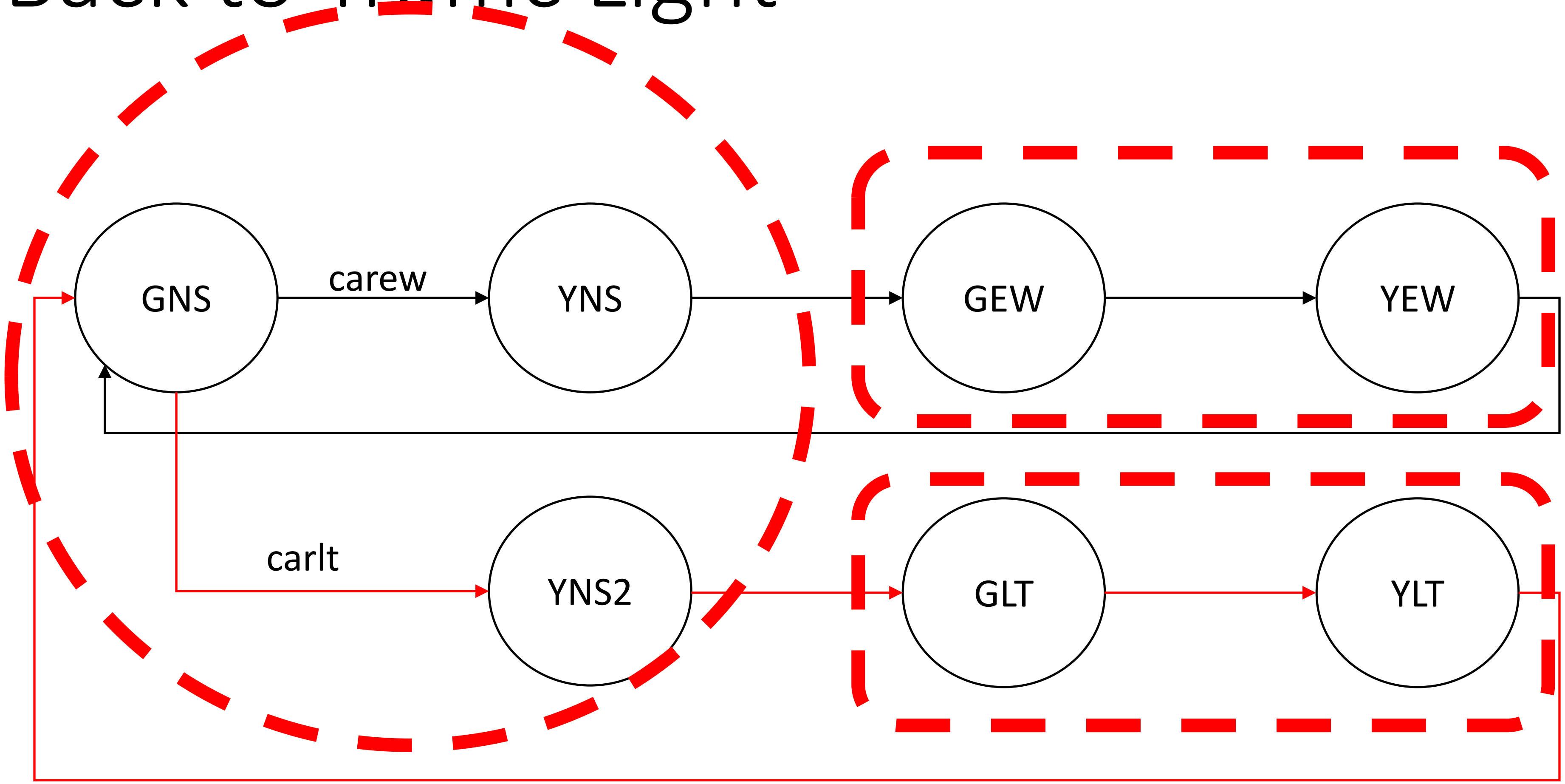
- Now the problem has:
- East-West lights
- North-South Lights
- Left-Turn Lane Lights
- A total of 9 lights

- If a car is detected in the left-turn lane is detected
- Change the north-south light to yellow.
- Change the north-south light to red.
- Turn the left turn lane to green.
- Left Turn has priority.

Back to Traffic Light



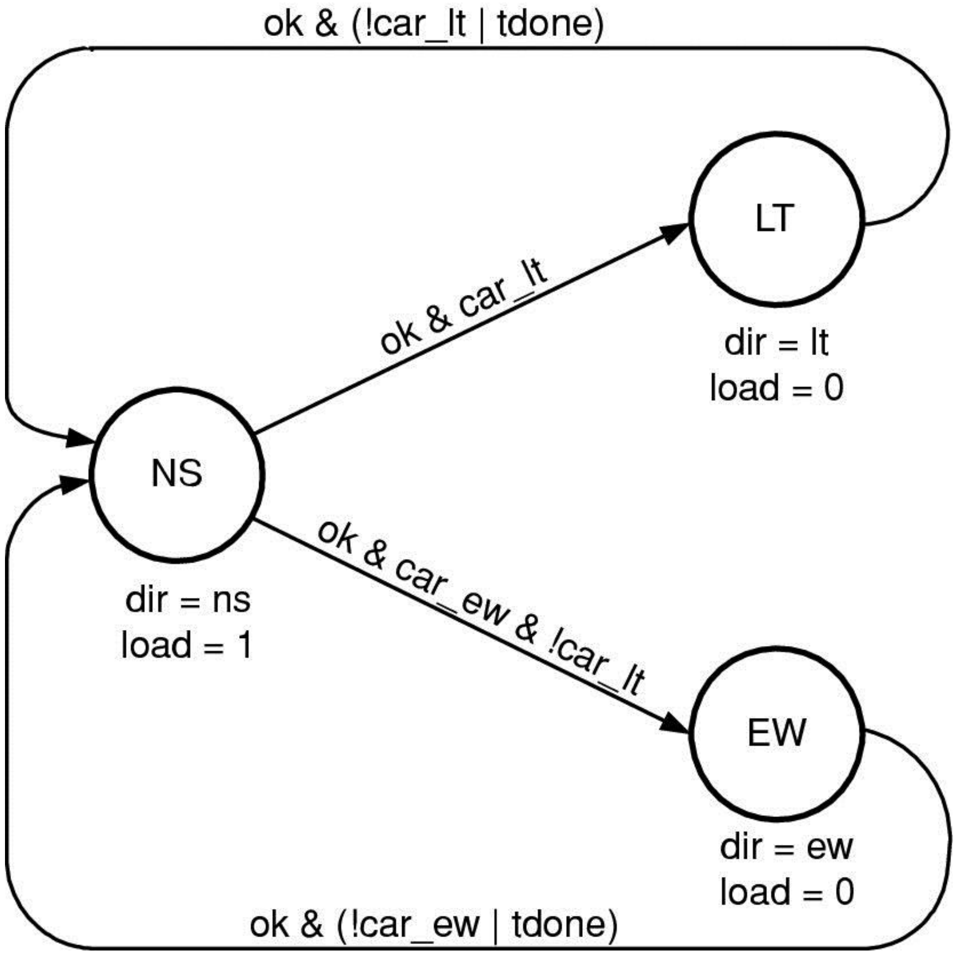
Back to Traffic Light



Now, there are three possibilities:
North-South Traffic: NS
East-West Traffic: EW
Left-Turn Traffic: NS

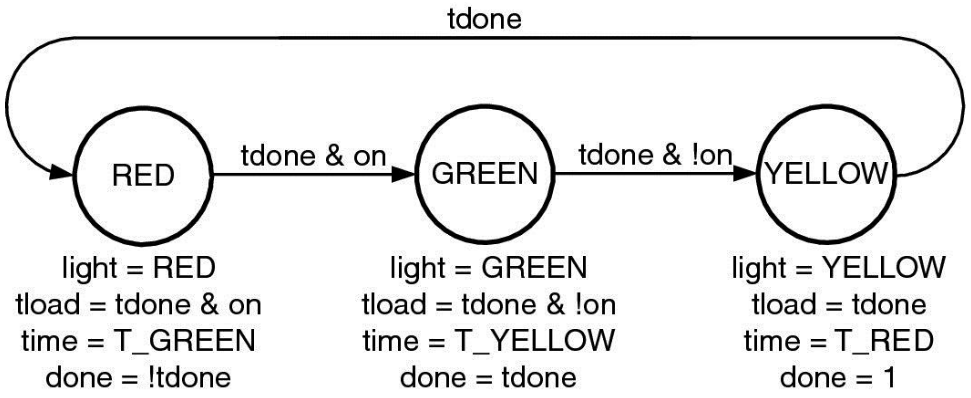
Back to Traffic Light: Controller FSM

- car_ew-car detected east-west
- car_lt-car detected left turn lane
- dir –current direction
- load –load timer command
- ok-a pause, no transition until all is ready
- tdone-timer complete



Back to Traffic Light: Light FSM

- Each timer cycles red, green, yellow
- done=changes on if timer is complete
- light=color
- on-only available if other state is done
- time-next state
- tload-load timer

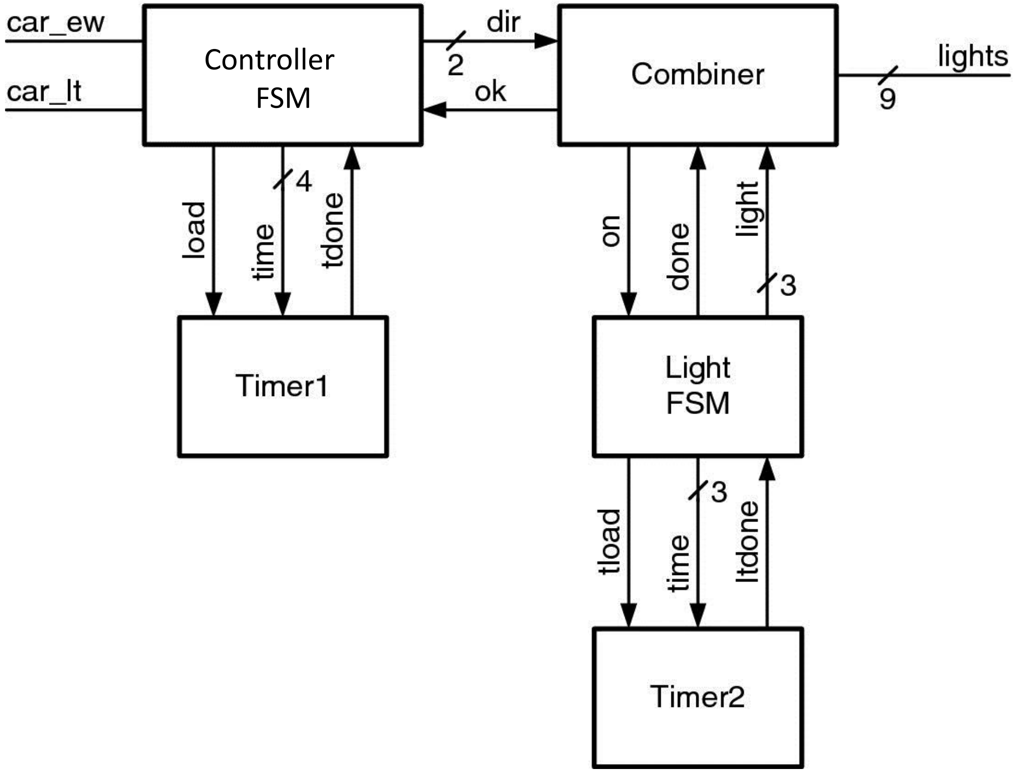


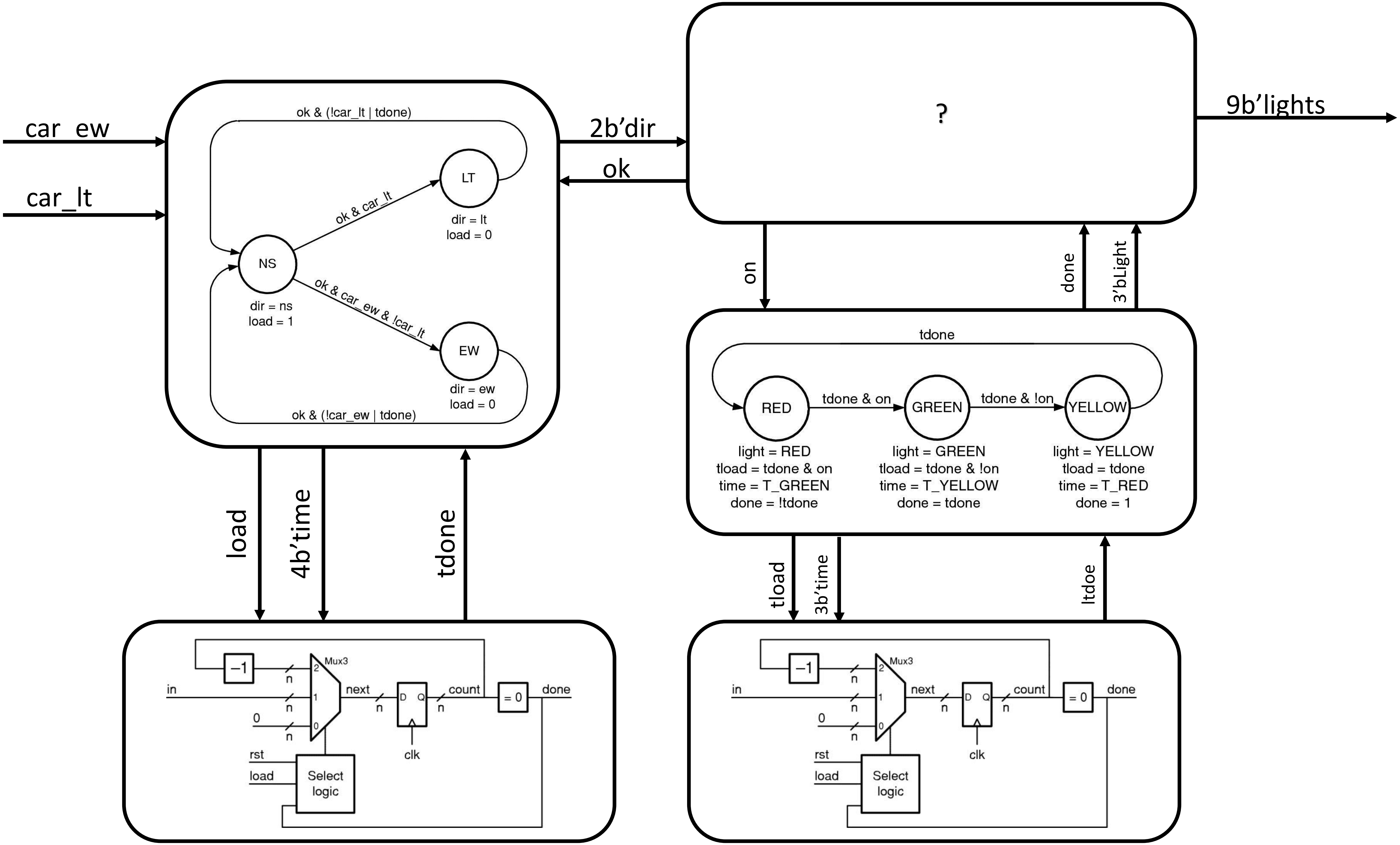
Back to Traffic Light: Pieces…

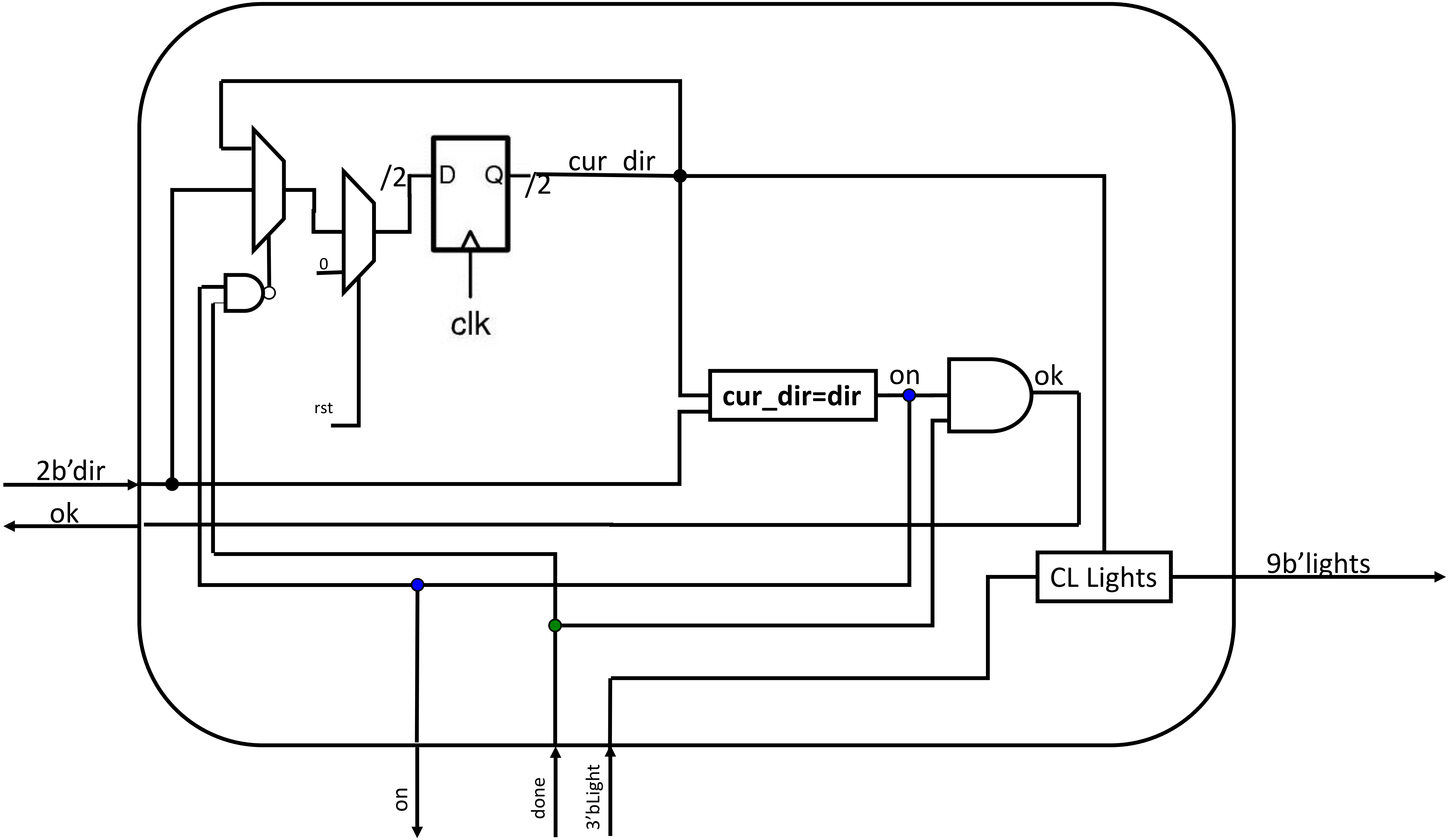
- Components
- Controller FSM
- Light Cycle FSM
- Timer for the Lights
- Timer for the Controller
- Oops…Lights depend on the current state

Back to Traffic Light: Combiner

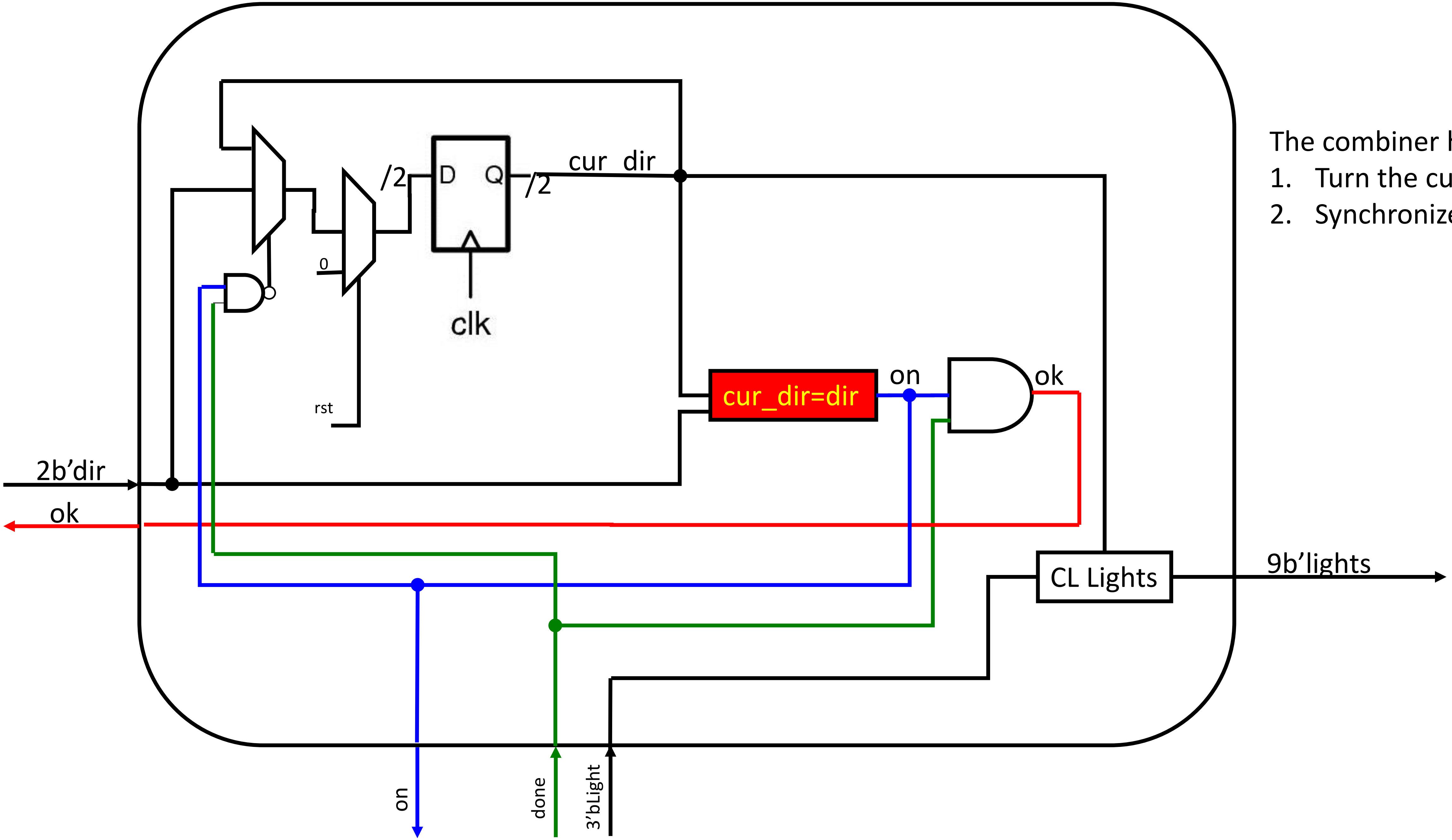
- The NS to EW to NS is Timer1
- The NS to LT to NS is same Timer1
- The state of the Controller FSM is fed to the Combiner
- Combiner remembers the current direction, from Controller
- Red-Green-Yellow is on Timer2.




car_ew
car_lt
ok & (!car_lt | tdone)
LT
dir = lt
load = 0
ok & car_lt
NS
dir = ns
load = 1
ok & car_ew & !car_lt
EW
dir = ew
load = 0
ok & (!car_ew | tdone)
2b'dir
ok
?
9b'lights
on
done
3'bLight
tdone
RED
tdone & on
GREEN
tdone & !on
YELLOW
light = RED
tload = tdone & on
time = T_GREEN
done = !tdone
light = GREEN
tload = tdone & !on
time = T_YELLOW
done = tdone
light = YELLOW
tload = tdone
time = T_RED
done = 1
load
4b'time
tdone
tload
3b'time
ltdoe
−1
Mux3
2
1
0
in
0
n
next
D
Q
count
= 0
done
clk
rst
load
Select logic
−1
Mux3
2
1
0
in
0
n
next
D
Q
count
= 0
done
clk
rst
load
Select logic



cur_dir
/2
D
Q
/2
0
clk
rst
cur_dir=dir
on
ok
2b'dir
ok
CL Lights
9b'lights
on
done
3'bLight



The combiner has two fu

1. Turn the current stat
2. Synchronize the two