1) ALL RED 5s (no traffic allowed)

2) NS green, peds white, EW RED

↓

3) 24s, * end 15s early if EW sensor is active

4) NS peds flashing, car lights green, 6s

5) NS car lights yellow, peds flashing 4s

6) NS peds red, NS car lights yellow, 1s

7) ALL RED, 2s

8) Repeat steps 2-8 in orthogonal direction

NOT responsible for resetting sensors after allowing traffic

* TLC connected to 2 sensors: NS & EW
  - ↳ represented by a single bit (Y/N waiting cars/peds) for total 2 bits
  - → If current direction yellow/flashing, then sensors have no effect

* Car traffic only in 1 direction (accidents!), pedestrian traffic may be allowed in same direction (???) (24s + 6s max

* Car traffic should not be green in a direction for more than 30 seconds at a time (fairness!)

* Green cycle → shortened if cars/pedestrians waiting, must last at least 15 seconds

* After green cycle → lights yellow 5s before turning red

* Whenever traffic finishes one cycle, all lights red before new cycle

* Peds cross in direction when car traffic is green

* Peds CAN'T cross on last second of car yellow cycle

* Ped lights blink 10s before turning red

Bit	States	Meaning
0	S_all_red_start	startup all red (5s)
1	S_NS_green	NS cars green, NS peds white (15s ≤ t ≤ 24s)
2	S_NS_peds_flash	NS peds flash 6s (cars still green)
3	S_NS_yellow	NS yellow 4s (peds still flash)
4	S_NS_PED_RED	NS peds red 1s (cars still yellow)
5	S_all_red_NS_2_EW	all red gap NS-EW (2s)
6	S_EW_green	EW cars green, EW peds white (15s ≤ t ≤ 24s)
7	S_EW_PED_flash	EW peds flash 6s (cars still green)
8	S_EW_Yellow	EW yellow 4s (peds still flash)
9	S_EW_PED_RED	EW peds red 1s (cars still yellow)
10	S_all_red_EW2NS	all red gap EW-NS (2s)
11		

Bit	States	One-hot value (binary)
0	S_all_red_start	00000 000001
1	S_NS_green	00000 000010
2	S_NS_peds_flash	00000 000100
3	S_NS_yellow	00000 001000
4	S_NS_PED_RED	00000 010000
5	S_all_red_NS_2_EW	00000 100000
6	S_EW_green	00001 000001
7	S_EW_PED_flash	00010 000001
8	S_EW_yellow	00100 000001
9	S_EW_PED_RED	01000 000001
10	S_all_red_EW2NS	10000 000001
11		

* Default "hold" rule: if no condition below is true, stay in currstate

			What we need for next state		
S_all_red_start	00000 000001	→	t0		else hold
S_NS_green	00000 000010	→	t0	OR cutNS	else hold
S_NS_peds_flash	00000 000100	→	t0		else hold
S_NS_yellow	00000 001000	→	t0		else hold
S_NS_PED_RED	00000 010000	→	t0		else hold
S_all_red_NS_2_EW	00000 100000	→	t0		else hold
S_EW_green	00001 000001	→	t0	OR cutEW	else hold
S_EW_PED_flash	00010 000001	→	t0		else hold
S_EW_yellow	00100 000001	→	t0		else hold
S_EW_PED_RED	01000 000001	→	t0		else hold
S_all_red_EW2NS	10000 000001	→	t0		else hold

ANY State w/ rst_any=1 goes to s_all_red_start

S[i] = current bit

N[i] = next bit

Last page cont.

N[S_NS_green] = (S[S_all_red_start] & to) | (S[S_all_red_ew2ns] & to)

N[S_NS_ped_flash] = S[S_NS_green] & (to | cutNS)

N[S_ns_yellow] = S[S_ns_ped_flash] & to

N[S_ns_ped_red] = S[S_ns_yellow] & to

N[S_all_red_ns2ew] = S[S_ns_ped_red] & to

N[S_ew_green] = S[S_all_red_ns2ew] & to

N[S_ew_ped_flash] = S[S_ew_green] & (to | cut EW)

N[S_EW_yellow] = S[S_ew_ped_flash] & to

N[S_ew_ped_red] = S[S_ew_yellow] & to

N[S_all_red_ew2ns] = S[S_ew_ped_red] & to

All other N[...] = 0, plus reset override

TLC → HAS to manage 4 outputs
- NS car traffic lights
- EW car traffic lights
- NS ped traffic lights
- EW ped traffic lights

↓

These outputs are controlled by 3 different outputs (from tlc.sv)

↳ Car traffic lights → [1:0] light_ns & [1:0] light_ew

2-bit buses that can each be assigned 1 of 3 macros →
- LIGHT_GREEN = 2'b01
- LIGHT_YELLOW = 2'b10
- LIGHT_RED = 2'b00

* Macro → outputs depend only on the current state

↳ 2 ped lights → a single 4-bit variable [3:0] ped_sigs

red, flashing green, green

PED_NS_[Amber, Orange, White]_EW_[Amber, Orange, White]

* really only need NS_W_EW_A, NS_O_EW_A, NS_A_EW_A, NS_A_EW_W, NS_A_EW_O

can be assigned 1 of 9 macros to describe both ped lights

Transition Logic:

t0 → timer_out = 5'd0

cut NS → car_ew && (timer_out ≤ 5'd9) // NS green may end early

cut EW → car_ns && (timer_out ≤ 5'd9) // EW green may end early

rst_any → rst || click_rst_pos // synchronous reset (click edge detected)

Fsm - State Assignments:

	Car lights (NS, EW)	Ped Signals	
		NS_A	EW_A
S_all_red_start	(R, R)		
S_NS_green	(G, R)	W	A
S_NS_peds_flash	(G, R)	O	A
S_NS_yellow	(Y, R)	O	A
S_NS_PED_RED	(Y, R)	A	A
S_all_red_NS_2_E	(R, R)	A	A
S_EW_green	(R, G)	A	W
S_EW_PED_flash	(R, G)	A	O
S_EW_yellow	(R, Y)	A	O
S_EW_PED_RED	(R, Y)	A	A
S_all_red_EW2NS	(R, R)	A	A

timer_en → waiting state

INPUTS

clk (100MHz) → raw clock from FPGA board
↳ used to generate 1Hz clock slow

clk_slow / clk_en (1Hz) → 1-pulse-per-second signal from clk
* TLC ffs update on this clock

rst → system reset → when high, FSM forced into start state

click_rst → manual (edge-detected) reset click
↳ rst_any = rst || click_rst_pos
* edge detected to create click_rst_pos

car_ns → North-South Direction Sensor
↳ goes high if there's a car or pedestrian waiting on NS side (used to end EW green phase early in early cut logic)
* cut EW = (car_ns && timer_out <= 5'd9));

car_ew → East-West Direction Sensor
↳ goes high if there's a car or pedestrian waiting on EW side (used to end NS green phase early in early cut logic)
* cut NS = (car_ew && timer_out <= 5'd9));

timer_out [4:0] → count down timer value
↳ comes from Timer module that counts down once per second
→ 5-bit number (ranges from 00000 (0) to 11111 (31))
* t0 = (timer_out = 5'd0); // time will be up

* when rst = 1
→ clock divider, timer, TLC FSM all reset
→ FSM goes to S_ALL_RED_START

click_rst
→ user button (raw signal)
* can be noisy / held down for multiple clock cycles (can stay high for multiple seconds)

click_rst_pos
* AKA positive edge detected
→ edge-detected version
/ while pressed * debounced moment the button was pressed, not how long it was held

rst_any
→ FSM's final reset line
/ for / * high if system tick after powering up or user clicked the button
click

OUTPUTS

`light_ns[1:0]` → NS car light signal
↳ 2-bit bus (wires) that chooses NS traffic light colors

`light_ew[1:0]` → Same as above for EW
* "Opposite" of NS (one is red when the other is green)

`ped_sigs[3:0]` → Pedestrian signal 4-bit bus that represents both pedestrian lights

`timer_init[4:0]` → How long each phase lasts
→ FSM tells timer how many seconds to count down for the current state

`timer_load` → 1-bit signal that goes high for one clock tick whenever FSM enters new state
↳ loads `timer_init` value & starts counting down

`timer_en` → enables timer
* equals 1 in any working state
↳ Keeps timer running while waiting in any state

* `timer_load` = 1 when state changes

`timer_rst` → resets the timer counter
↳ resets the timer to its last loaded value

* MACROS

LIGHT_RED
LIGHT_YELLOW
LIGHT_GREEN

* MACROS

PED_NS_WHITE_EW_AMBER
↳ NS walk, EW don't walk

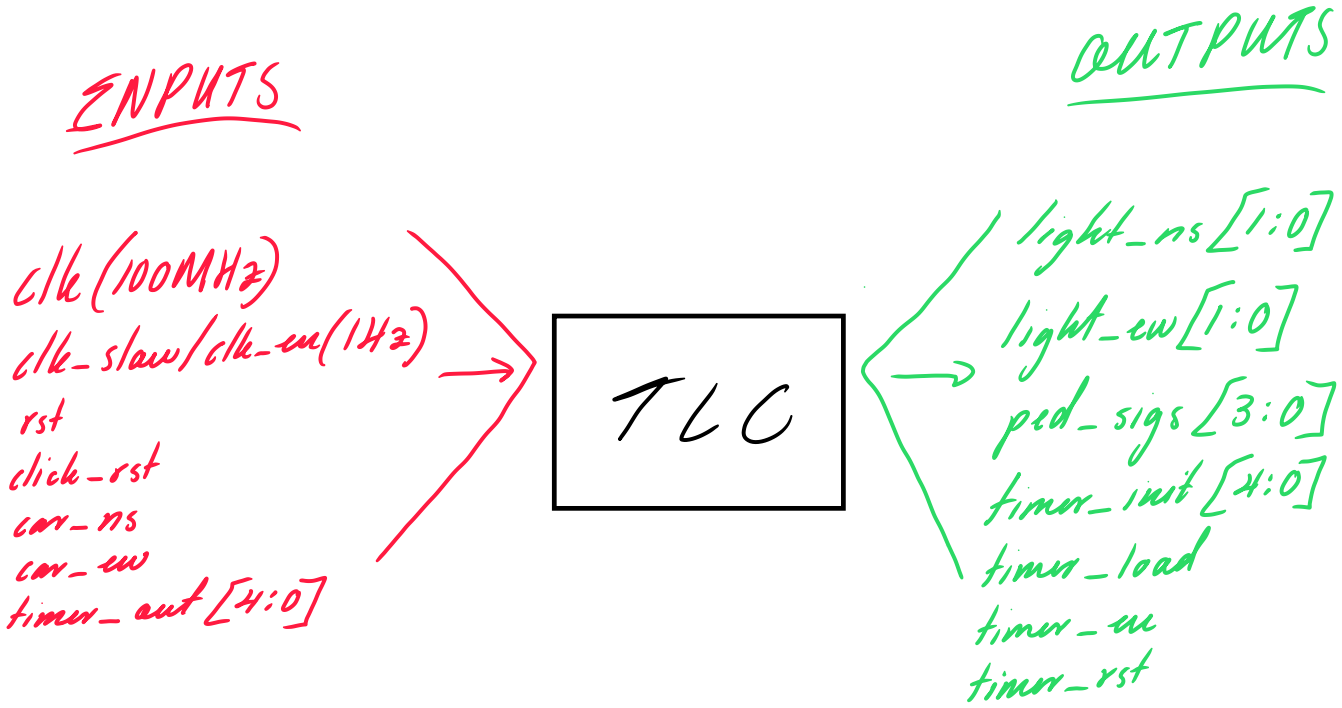
PED_NS_ORANGE_EW_AMBER
↳ NS finish (flashing) crossing, EW don't walk

PED_NS_AMBER_EW_AMBER
↳ NS don't walk, EW don't walk

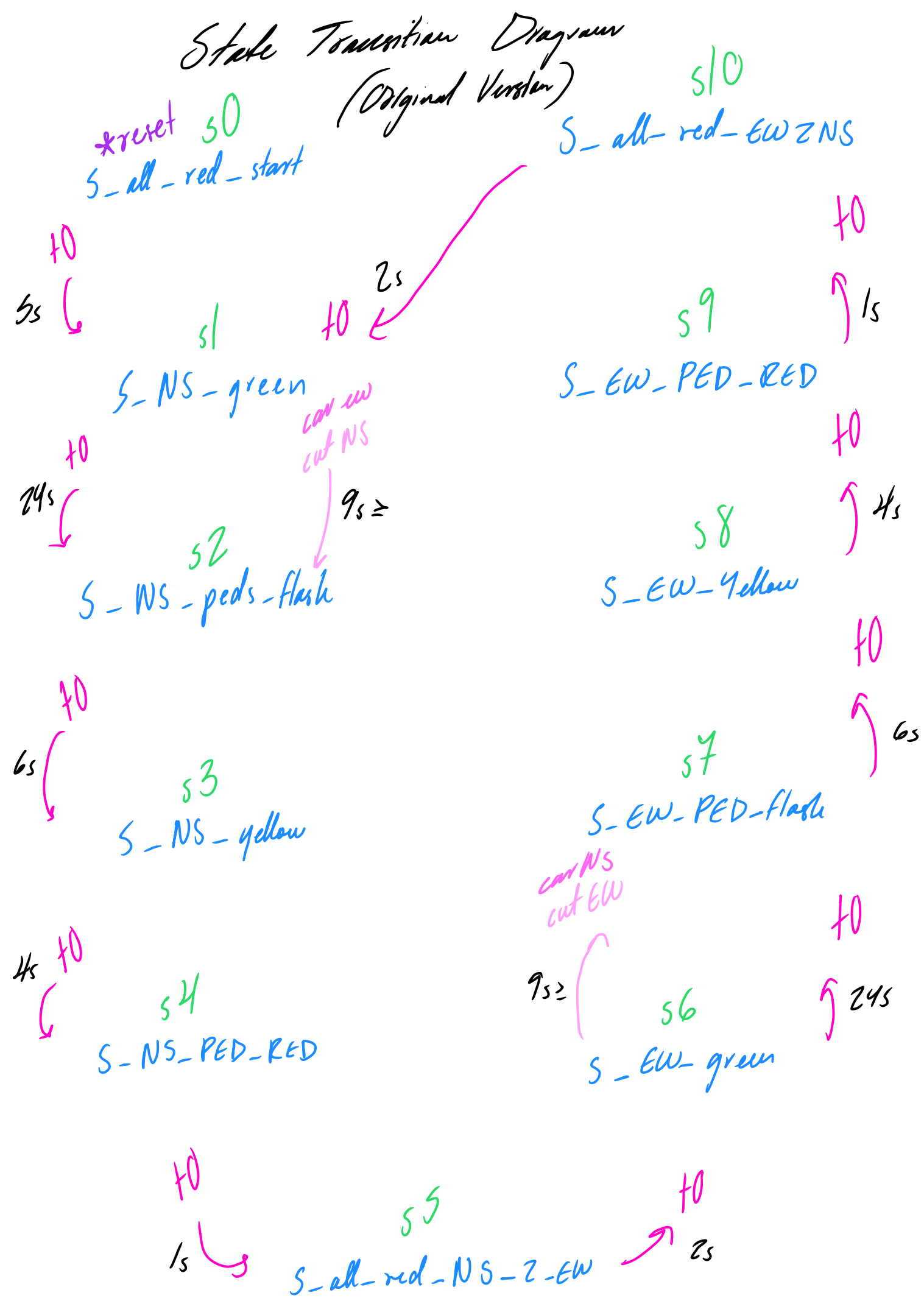
PED_NS_AMBER_EW_WHITE
↳ NS don't walk, EW walk

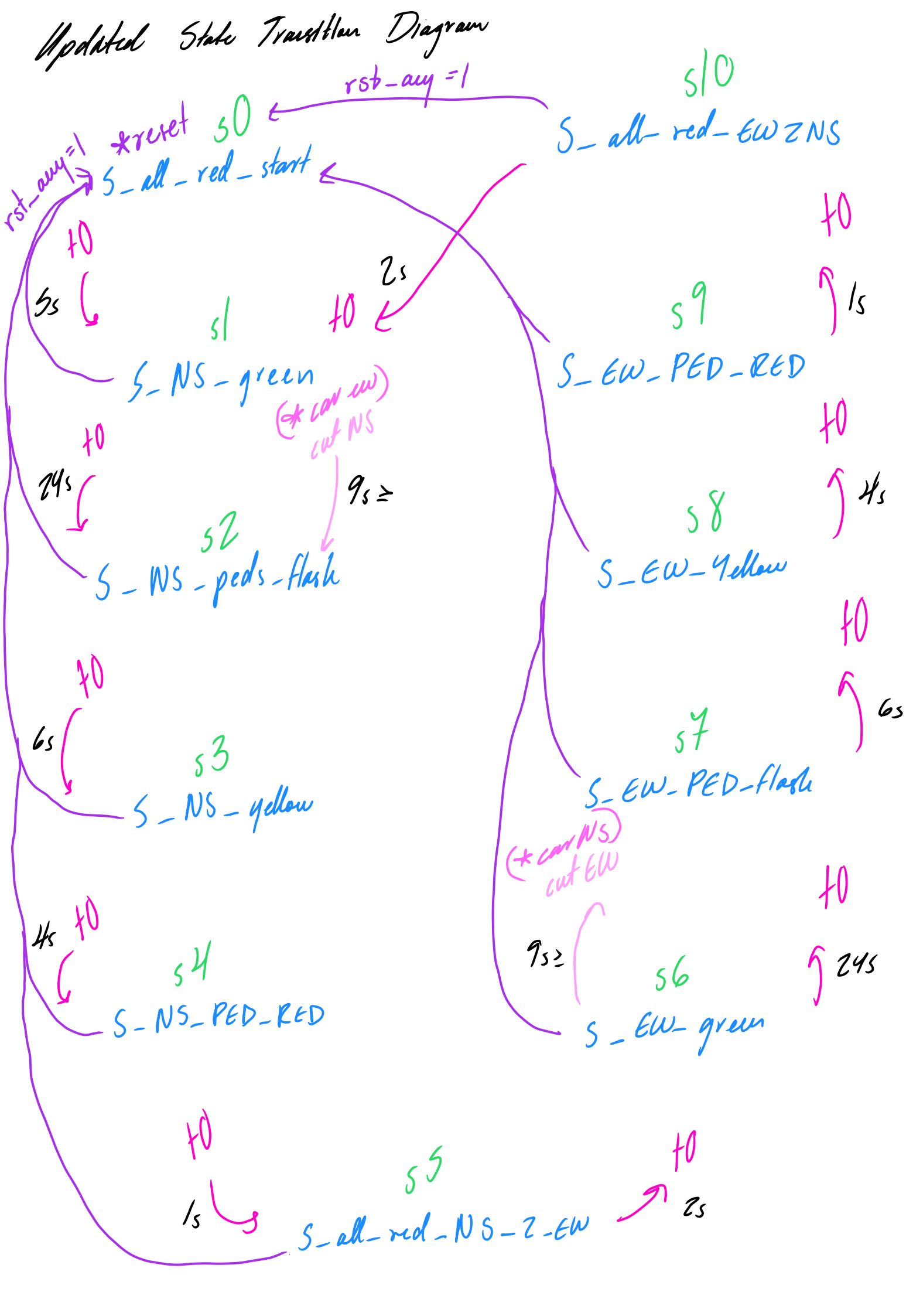
PED_NS_AMBER_EW_ORANGE
↳ NS don't walk, EW finish (flashing) crossing

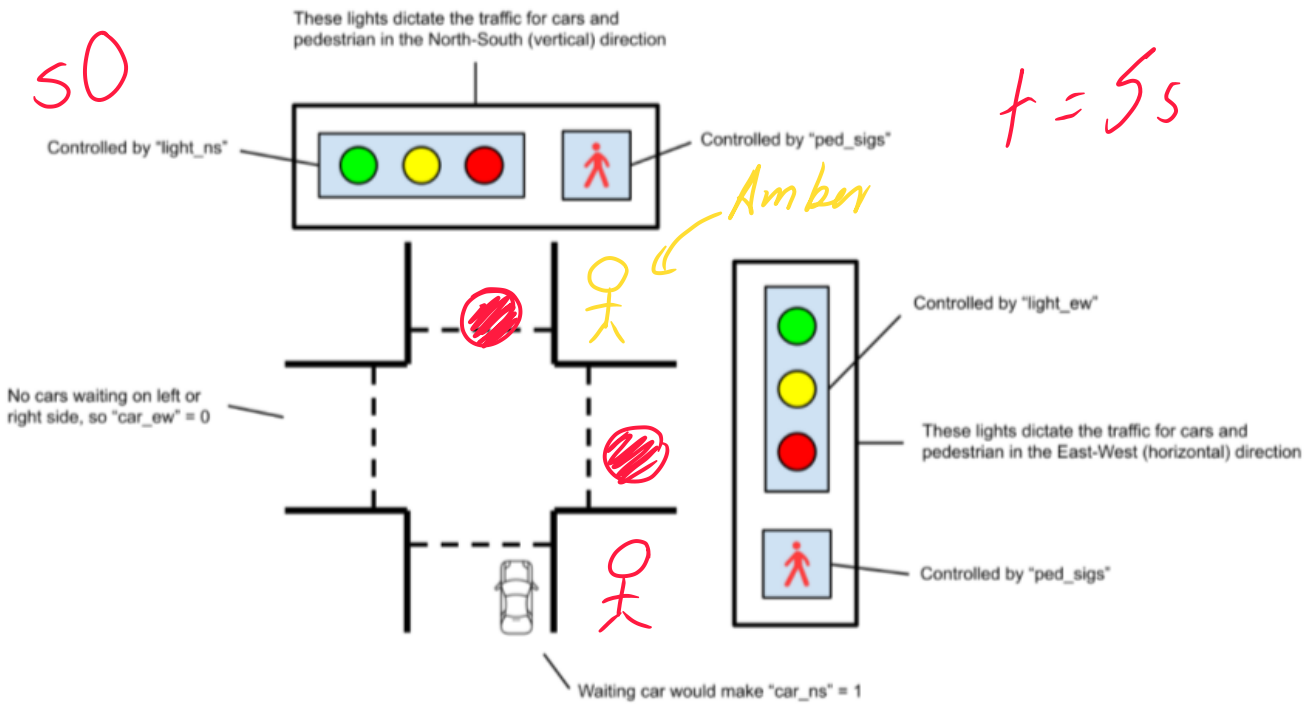
Black Box View of FSM:

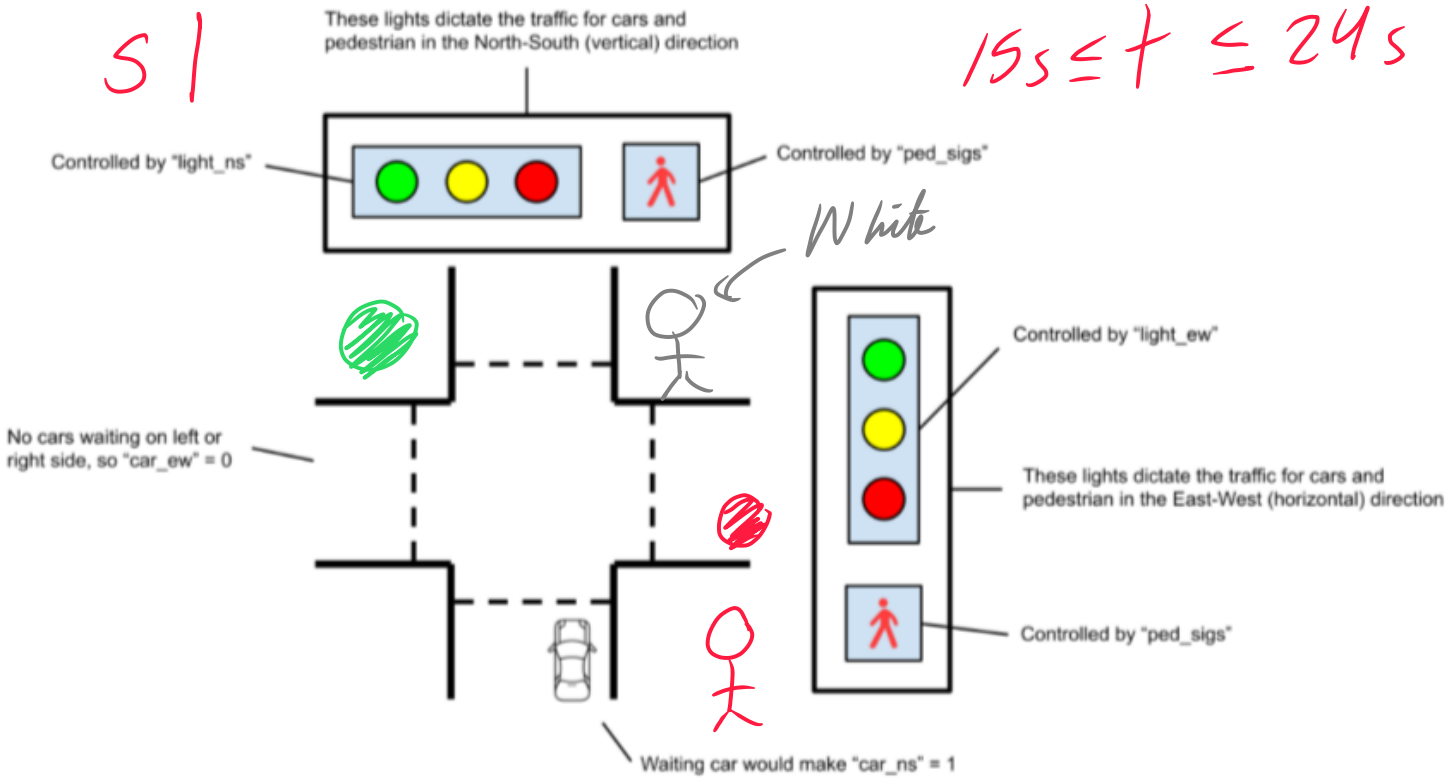


State Transition Diagram (Original Version)



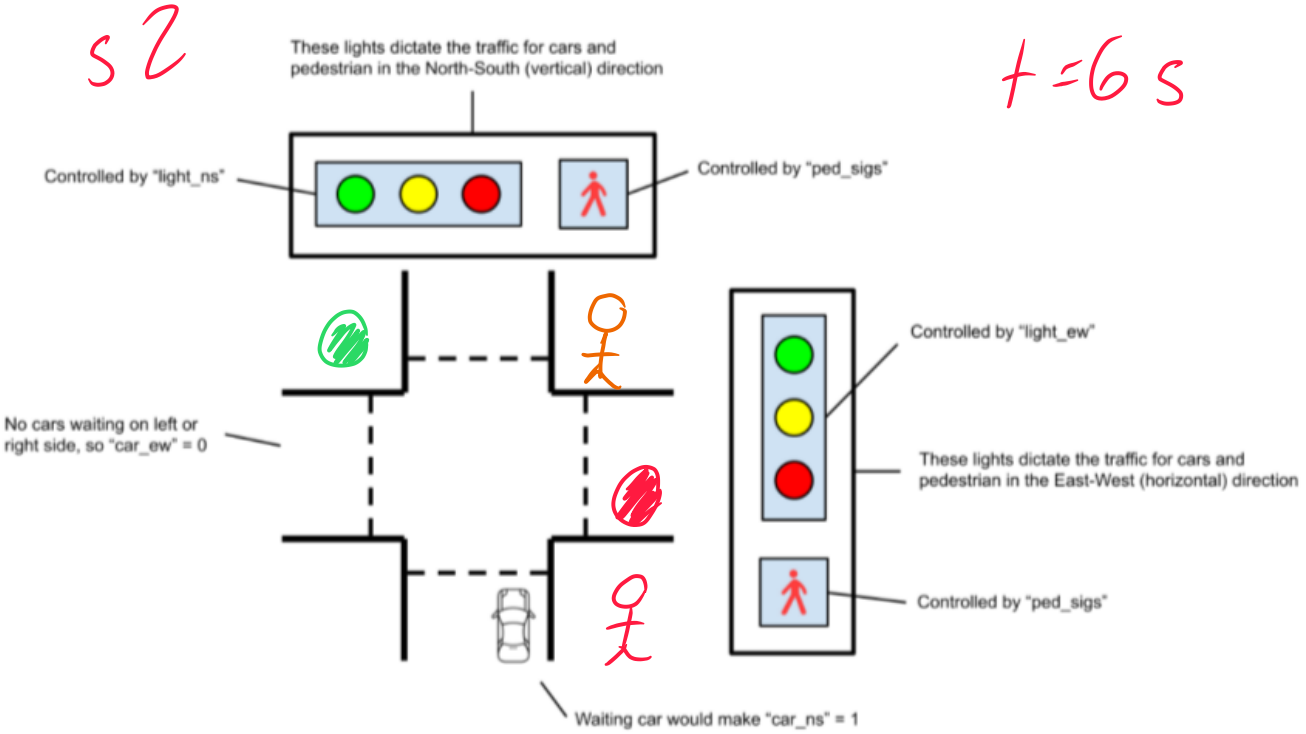

Updated State Transition Diagram
rst_any=1
s10
S_all_red_EWZNS
rst_any=1
*reset
s0
S_all_red_start
t0
2s
t0
5s
s1
S_NS_green
t0
1s
s9
S_EW_PED_RED
(*car EW)
cut NS
t0
24s
9s≥
s2
S_NS_peds_flash
t0
4s
s8
S_EW_Yellow
t0
6s
s3
S_NS_yellow
t0
6s
s7
S_EW_PED_flash
(*car NS)
cut EW
t0
4s
t0
9s≥
s4
S_NS_PED_RED
s6
S_EW_green
t0
24s
t0
1s
s5
S_all_red_NS_2_EW
t0
2s


s0
t = 5s
These lights dictate the traffic for cars and pedestrian in the North-South (vertical) direction
Controlled by "light_ns"
Controlled by "ped_sigs"
Amber
Controlled by "light_ew"
No cars waiting on left or right side, so "car_ew" = 0
These lights dictate the traffic for cars and pedestrian in the East-West (horizontal) direction
Controlled by "ped_sigs"
Waiting car would make "car_ns" = 1


s1
15s ≤ t ≤ 24s
These lights dictate the traffic for cars and pedestrian in the North-South (vertical) direction
Controlled by "light_ns"
Controlled by "ped_sigs"
White
Controlled by "light_ew"
No cars waiting on left or right side, so "car_ew" = 0
These lights dictate the traffic for cars and pedestrian in the East-West (horizontal) direction
Controlled by "ped_sigs"
Waiting car would make "car_ns" = 1

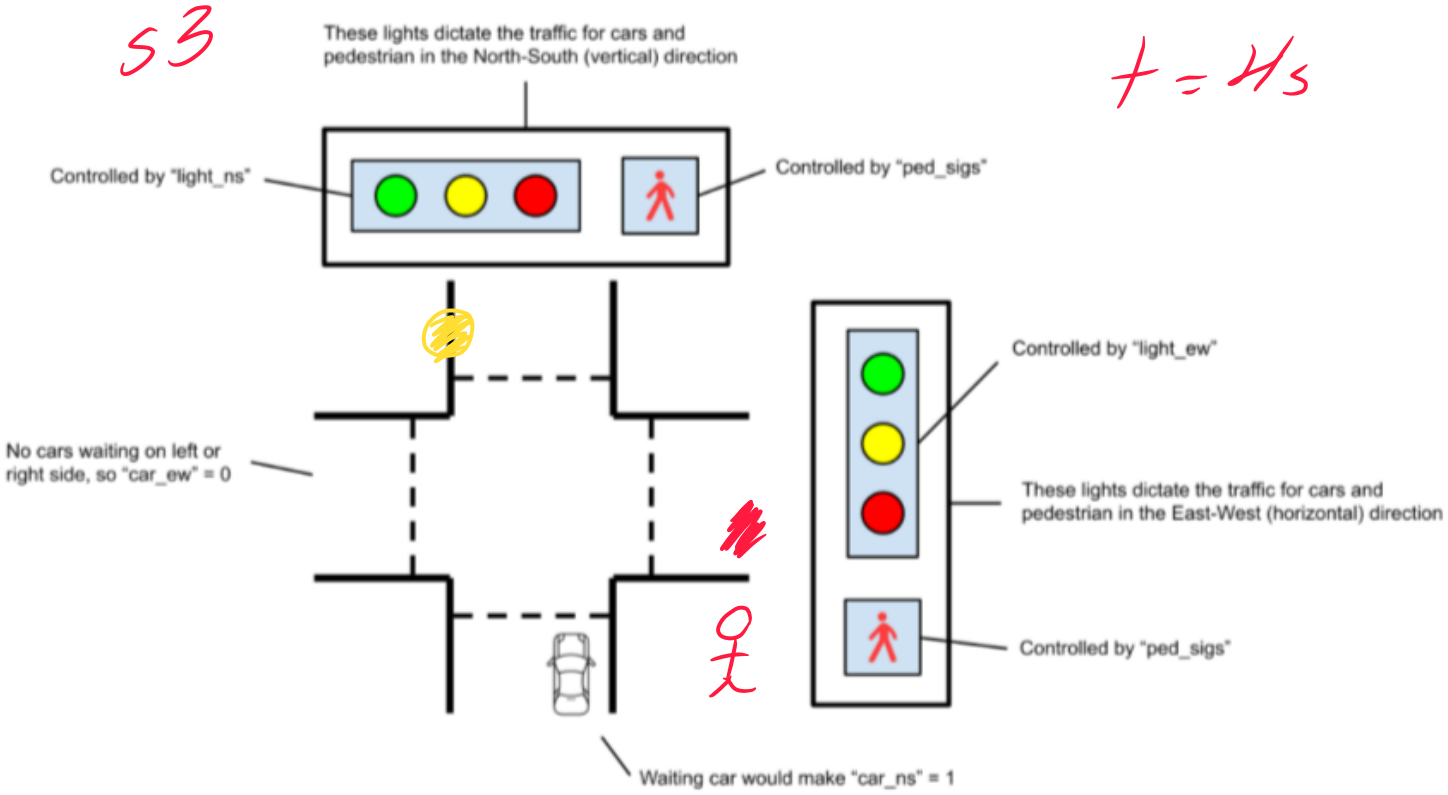
s2

t=6 s


These lights dictate the traffic for cars and pedestrian in the North-South (vertical) direction
Controlled by "light_ns"
Controlled by "ped_sigs"
Controlled by "light_ew"
No cars waiting on left or right side, so "car_ew" = 0
These lights dictate the traffic for cars and pedestrian in the East-West (horizontal) direction
Controlled by "ped_sigs"
Waiting car would make "car_ns" = 1

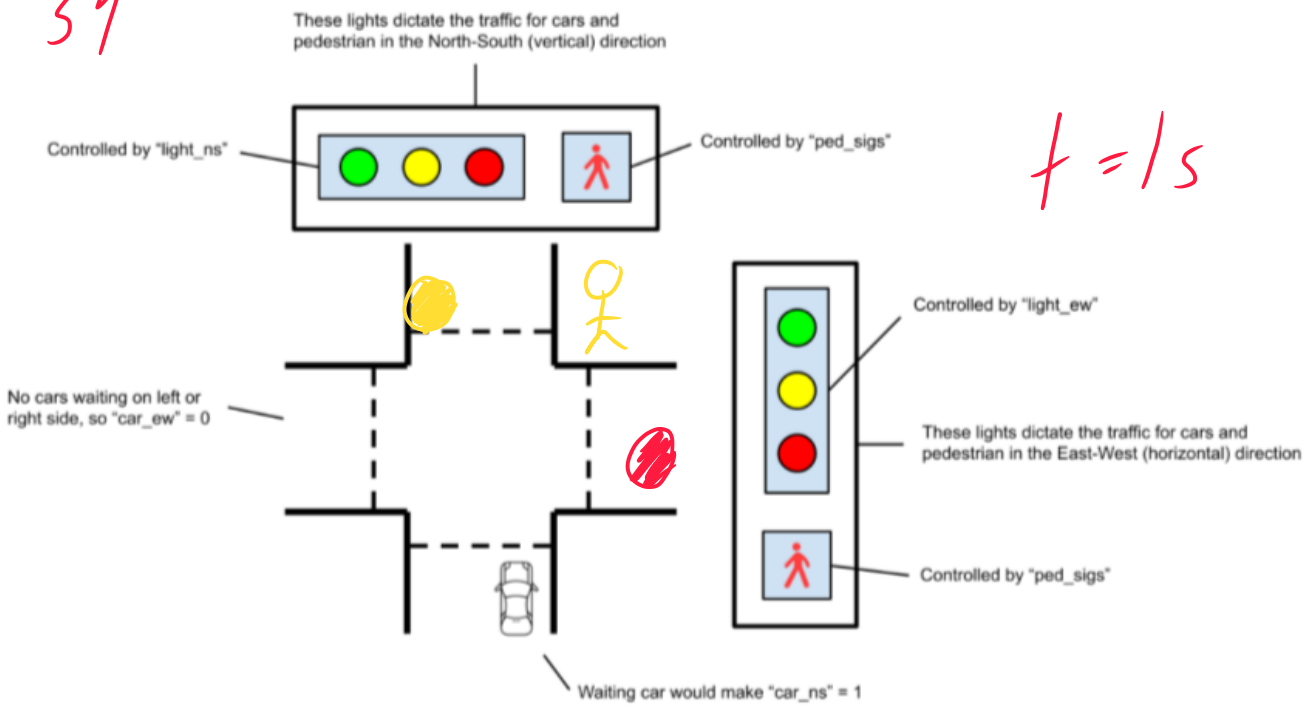
s3

t=4 s


These lights dictate the traffic for cars and pedestrian in the North-South (vertical) direction
Controlled by "light_ns"
Controlled by "ped_sigs"
Controlled by "light_ew"
No cars waiting on left or right side, so "car_ew" = 0
These lights dictate the traffic for cars and pedestrian in the East-West (horizontal) direction
Controlled by "ped_sigs"
Waiting car would make "car_ns" = 1

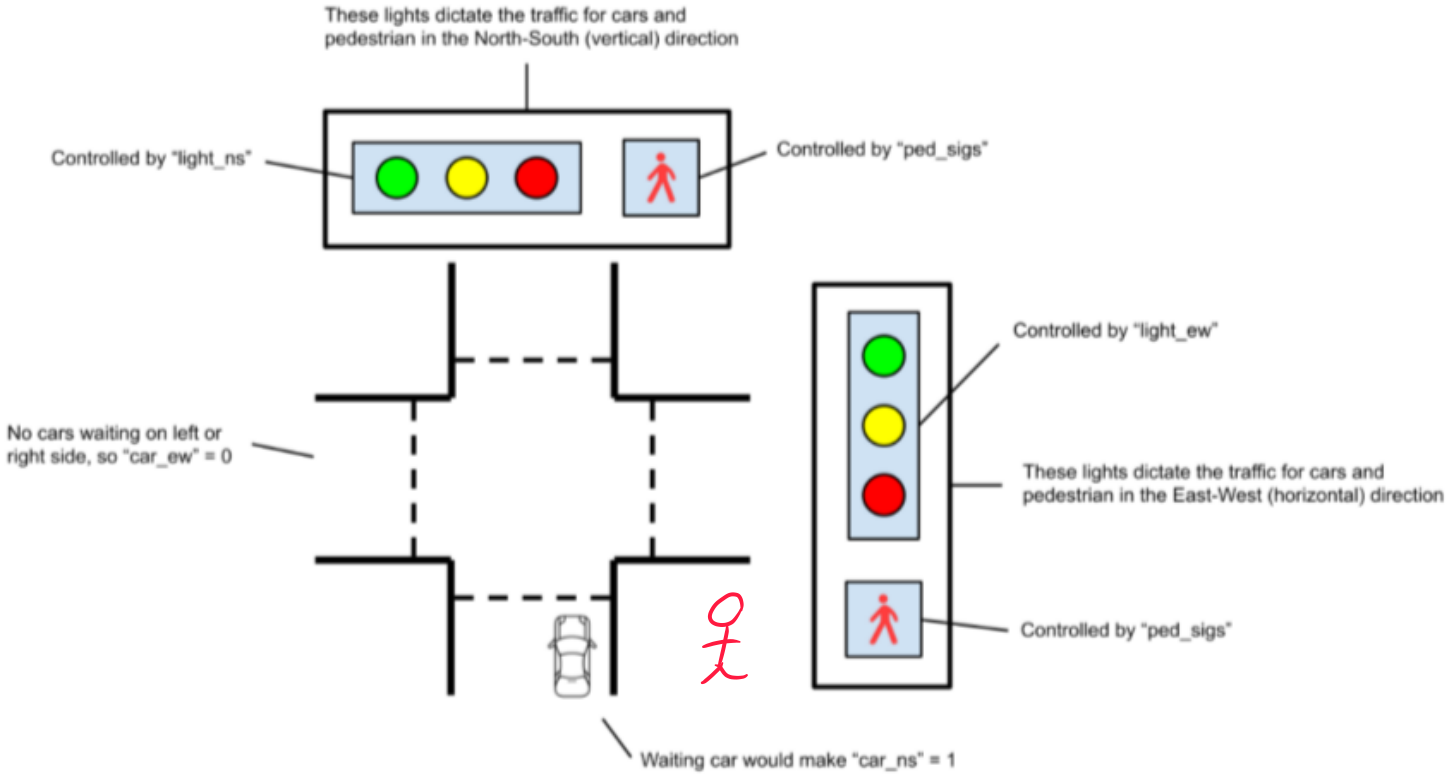
s4

t = 1s

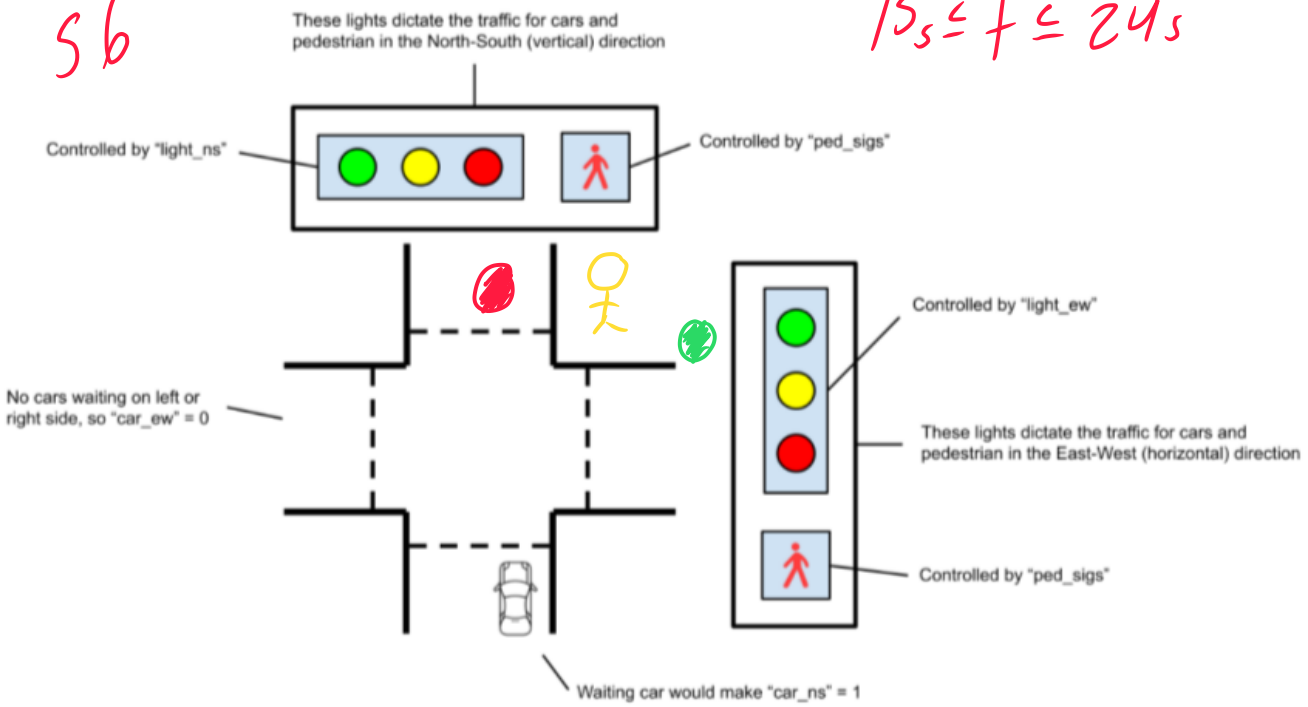

These lights dictate the traffic for cars and pedestrian in the North-South (vertical) direction
Controlled by "light_ns"
Controlled by "ped_sigs"
Controlled by "light_ew"
These lights dictate the traffic for cars and pedestrian in the East-West (horizontal) direction
Controlled by "ped_sigs"
No cars waiting on left or right side, so "car_ew" = 0
Waiting car would make "car_ns" = 1

s5

t = 2s

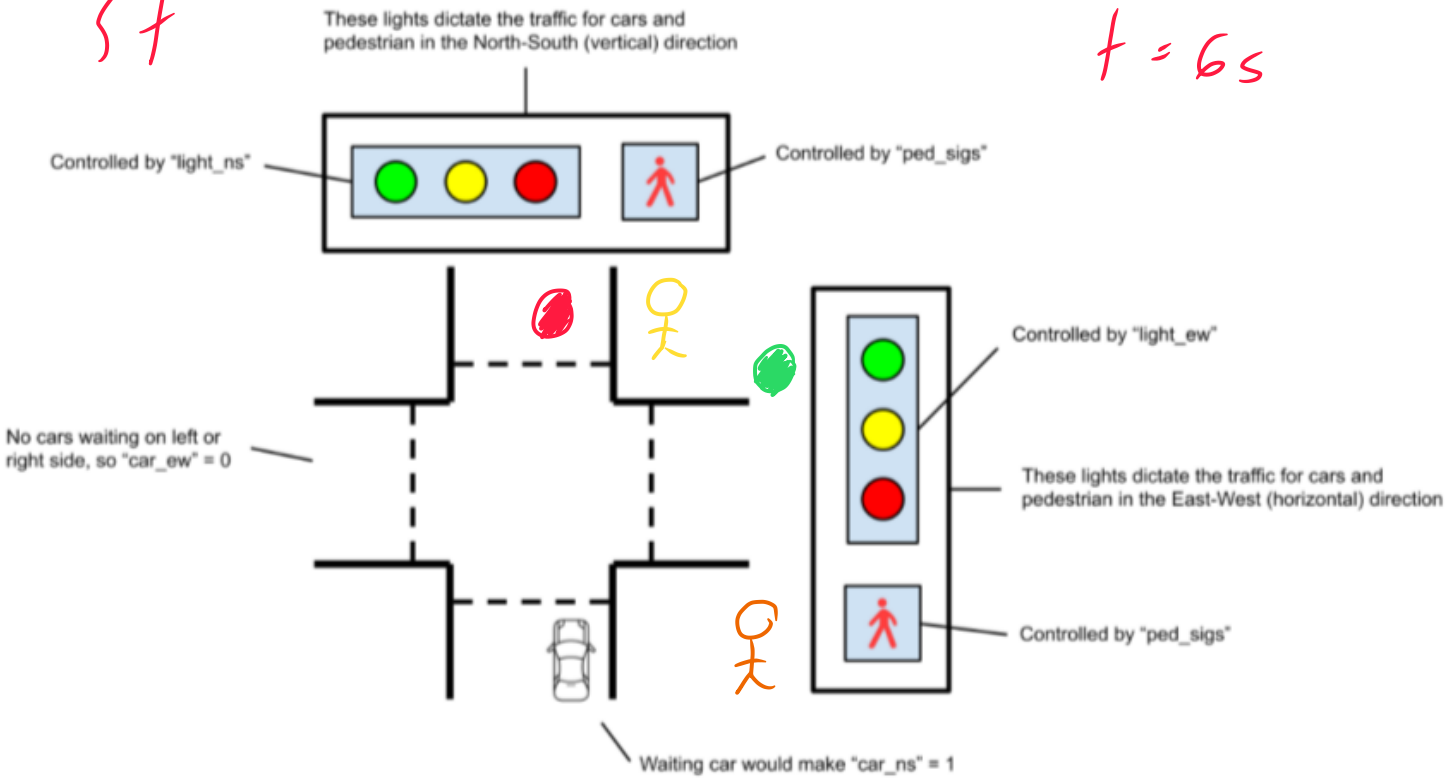

These lights dictate the traffic for cars and pedestrian in the North-South (vertical) direction
Controlled by "light_ns"
Controlled by "ped_sigs"
Controlled by "light_ew"
These lights dictate the traffic for cars and pedestrian in the East-West (horizontal) direction
Controlled by "ped_sigs"
No cars waiting on left or right side, so "car_ew" = 0
Waiting car would make "car_ns" = 1

s6



$15s \leq t \leq 24s$

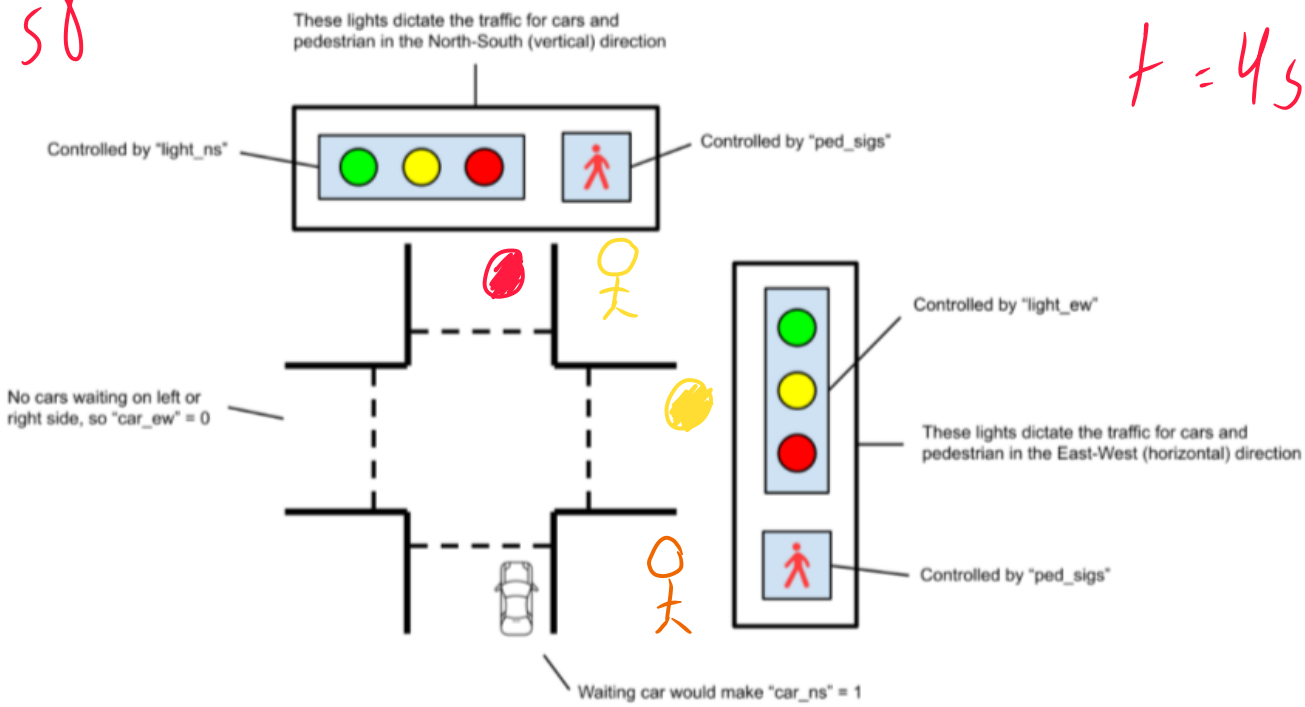
s7



$t = 6s$

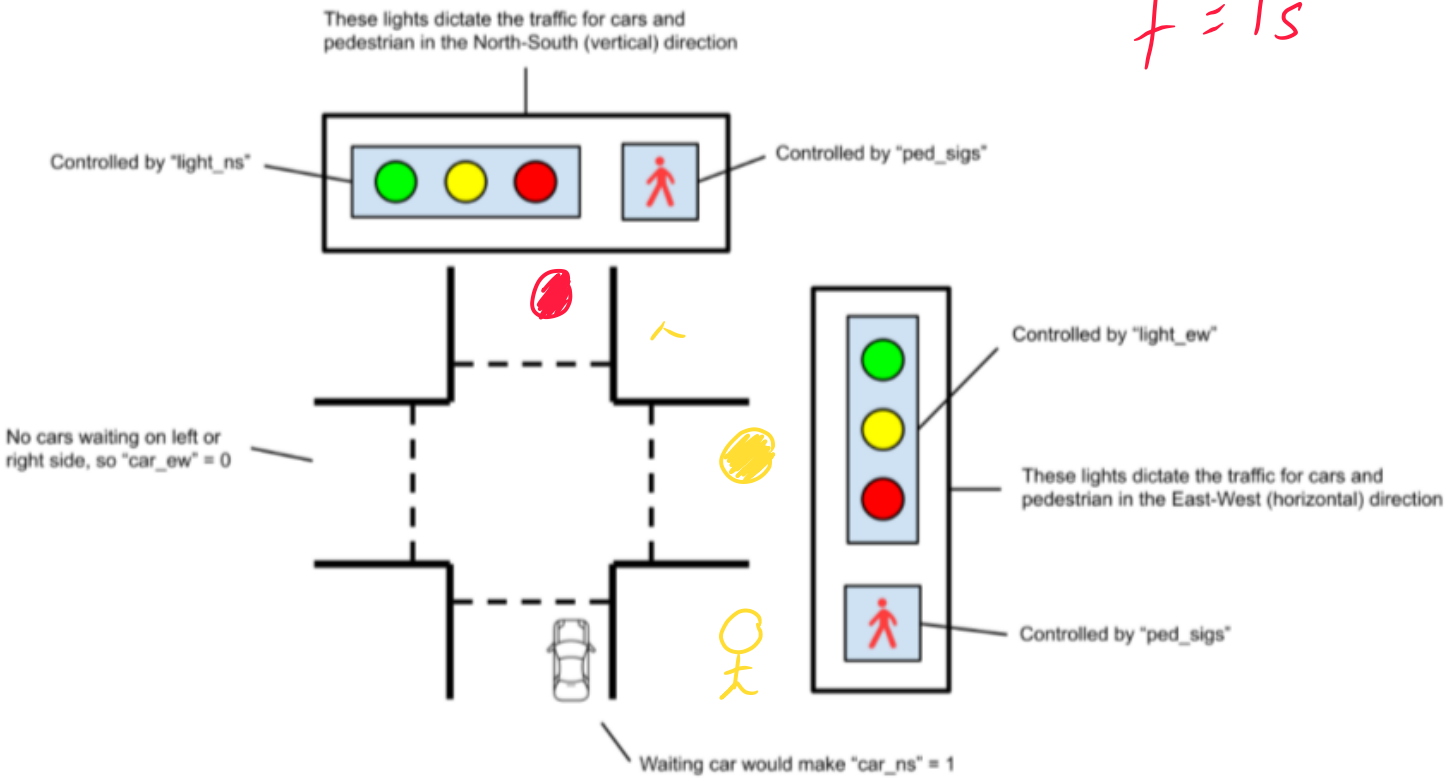
s8

t = 4s


These lights dictate the traffic for cars and pedestrian in the North-South (vertical) direction
Controlled by "light_ns"
Controlled by "ped_sigs"
Controlled by "light_ew"
No cars waiting on left or right side, so "car_ew" = 0
These lights dictate the traffic for cars and pedestrian in the East-West (horizontal) direction
Controlled by "ped_sigs"
Waiting car would make "car_ns" = 1

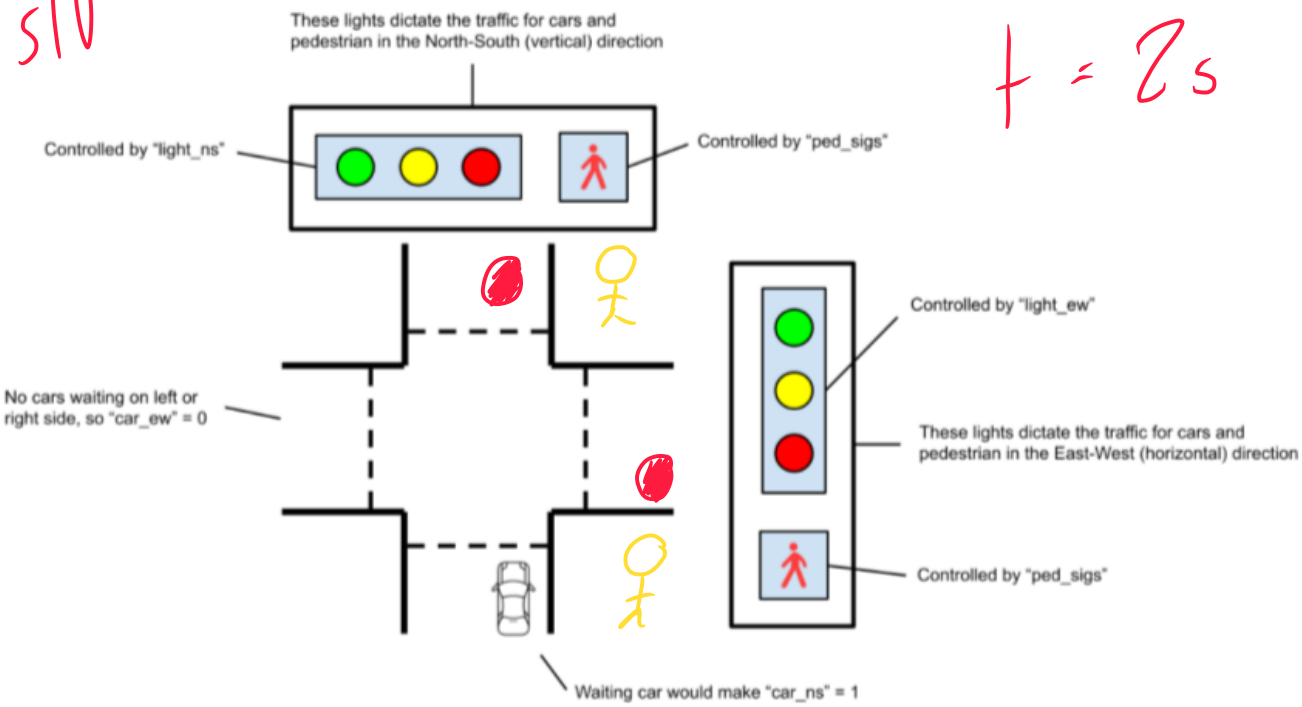
s9

t = 1s


These lights dictate the traffic for cars and pedestrian in the North-South (vertical) direction
Controlled by "light_ns"
Controlled by "ped_sigs"
Controlled by "light_ew"
No cars waiting on left or right side, so "car_ew" = 0
These lights dictate the traffic for cars and pedestrian in the East-West (horizontal) direction
Controlled by "ped_sigs"
Waiting car would make "car_ns" = 1

s10

t = 2s


These lights dictate the traffic for cars and pedestrian in the North-South (vertical) direction
Controlled by "light_ns"
Controlled by "ped_sigs"
No cars waiting on left or right side, so "car_ew" = 0
Controlled by "light_ew"
These lights dictate the traffic for cars and pedestrian in the East-West (horizontal) direction
Controlled by "ped_sigs"
Waiting car would make "car_ns" = 1

Clock-Timer Interactions

How The FSM works (Moore)

FSM Diagram: