5. How to prevent accidentally inheriting latches?

4. Note use of synthesizable directives:
- `always`
- `initial`
- `asynchronous` usually results in less logic (reset is actually synchronized when it

3. Is this reset `asynchronous` or `asynchronous`?

2. Separate `sequential` and `combinational` logic

1. Code each FSM by itself in one module.

6. FSM Verilog Notes

Why can we state `// synopsys Log simulate` for `FSM`?

- `current-state` tell `sylnopsys` what the state vector is.
- `next-state` tell `sylnopsys` what the state vector is.
- `synopsys enum` states and `// synopsys state-vector`
Creating hierarchy

traffic_light_controller

traffic_fsm
(contains only FSM-specific portion)

created by group....
Test

Monday March 6
4:05 pm
DAN 429
(Both sections unless you make alternative arrangements with me)

Next set of notes to appear Thursday
FSM Synthesis: Script Extract...
always @ (current_state or car) or reset begin
red = 0; yellow = 0; green = 0;
if (!reset) red = 1; else
Case (current_state)

50: begin
    if (car) begin
        next_state = 51;
        yellow = 1;
    end
    else begin
        next_state = 50;
        red = 1;
    end
end

51: begin
    green = 1;
    next_state = 52
end

52: begin
    red = 1;
    next_state = 50;
end

end case
end
clock
car
current_state: 00
next_state:

triggers:
1: 0 → 2
2: 0 → 2
3: 0 → 2

triggers 2
Exercise
Recode as Nealy machine.