CPE 462
VHDL: Simulation and Synthesis

Topic #07 - d) Finite State Machines with delay

Nuno Alves (nalves@wne.edu), College of Engineering

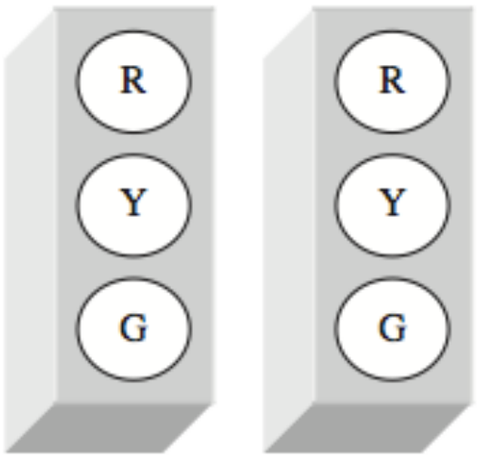
Finite State Machines (FSM) with delay

- On previous classes we learned about FSM
- FSM are machines that alternate between different states
- Today we are going to learn about FSM which change their states after some time.

Traffic Light Controller (TLC)

- We are going to implement the control for a traffic light in a intersection.
- There are only two traffic lights (**R**ed, **Y**ellow and **G**reen).
- There are three modes of operation: Regular, Test and Standby.
- In the regular mode, the FSM start when light#1=Red and light#2=Green. The machine remains in this state for 30 seconds

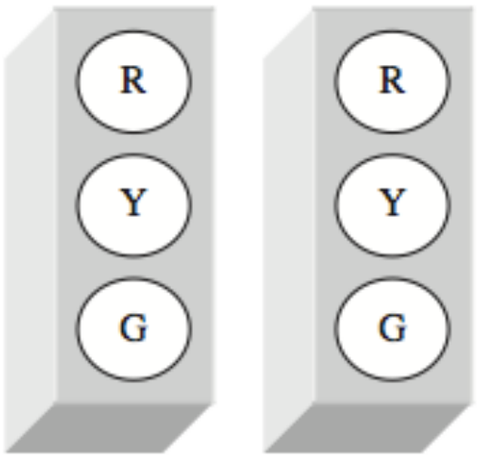
State	Operation Mode		
	REGULAR	TEST	STANDBY
	Time	Time	Time
RG	timeRG (30s)	timeTEST (1s)	---
RY	timeRY (5s)	timeTEST (1s)	---
GR	timeGR (45s)	timeTEST (1s)	---
YR	timeYR (5s)	timeTEST (1s)	---
YY	---	---	Indefinite



Traffic Light Controller (TLC)

State	Operation Mode		
	REGULAR	TEST	STANDBY
	Time	Time	Time
RG	timeRG (30s)	timeTEST (1s)	---
RY	timeRY (5s)	timeTEST (1s)	---
GR	timeGR (45s)	timeTEST (1s)	---
YR	timeYR (5s)	timeTEST (1s)	---
YY	---	---	Indefinite

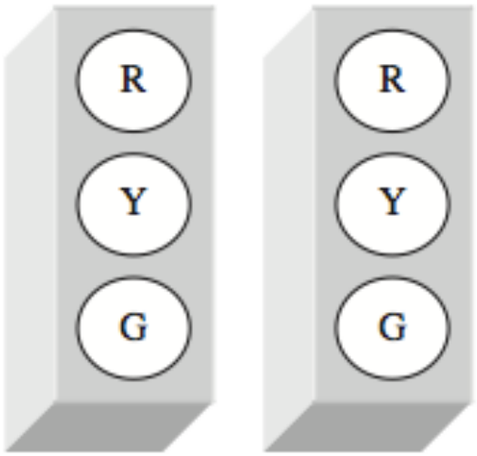
- Then the state machine will change to light#1=Red and light#2=Yellow and remain there for 5 seconds.



Traffic Light Controller (TLC)

State	Operation Mode		
	REGULAR	TEST	STANDBY
	Time	Time	Time
RG	timeRG (30s)	timeTEST (1s)	---
RY	timeRY (5s)	timeTEST (1s)	---
GR	timeGR (45s)	timeTEST (1s)	---
YR	timeYR (5s)	timeTEST (1s)	---
YY	---	---	Indefinite

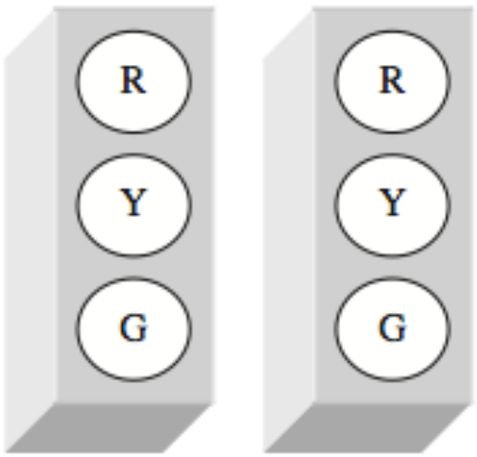
- Then the state machine will change to light#1=Green and light#2=Red and stay there for 45 seconds
- ... and so on



Traffic Light Controller (TLC)

State	Operation Mode		
	REGULAR	TEST	STANDBY
	Time	Time	Time
RG	timeRG (30s)	timeTEST (1s)	---
RY	timeRY (5s)	timeTEST (1s)	---
GR	timeGR (45s)	timeTEST (1s)	---
YR	timeYR (5s)	timeTEST (1s)	---
YY	---	---	Indefinite

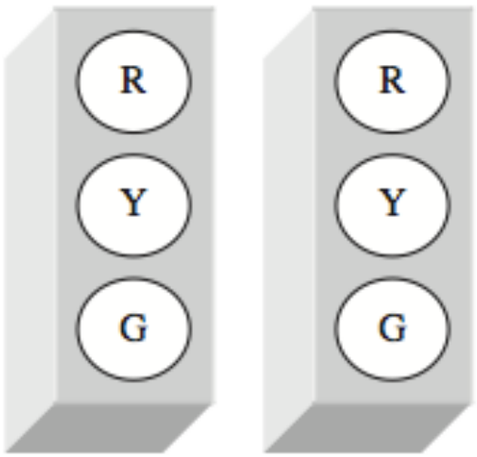
- If the machine is in **TEST** mode...
  - All pre-programmed times are going to be overwritten (by a manual switch) with a small value, such that the system can be easily tested during maintenance (1 second per state).



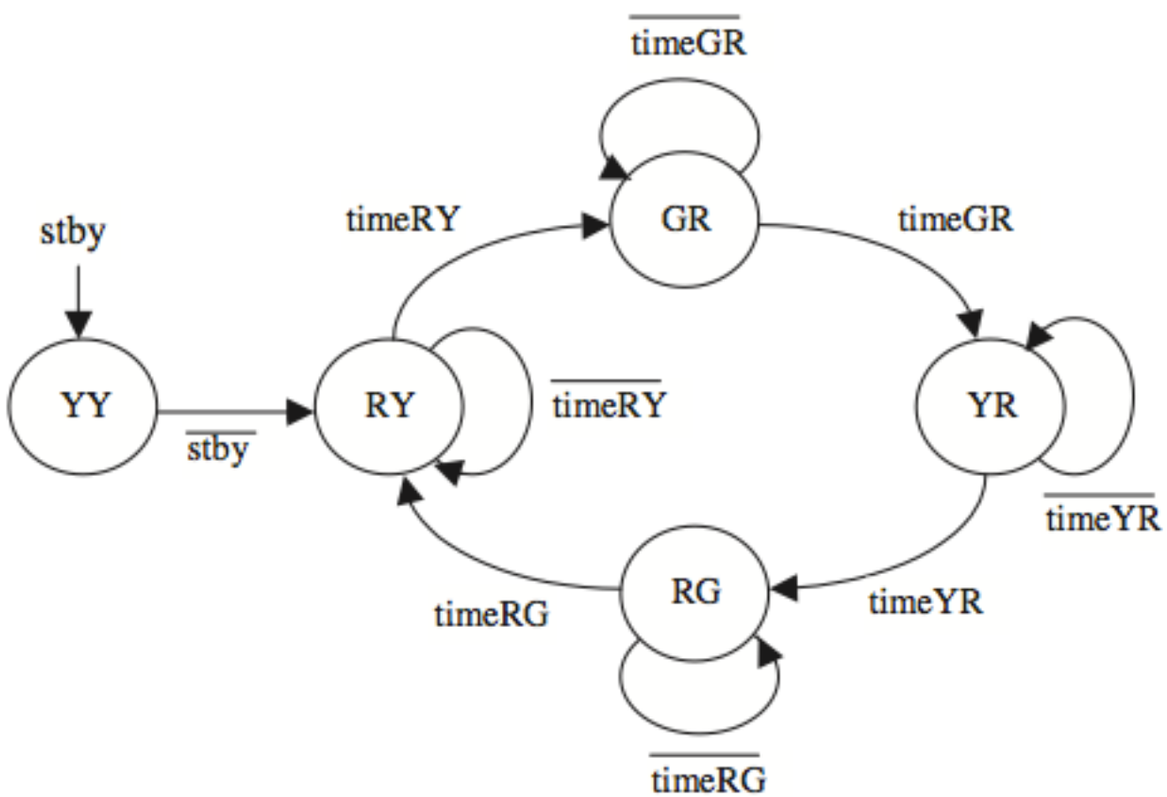
Traffic Light Controller (TLC)

State	Operation Mode		
	REGULAR	TEST	STANDBY
	Time	Time	Time
RG	timeRG (30s)	timeTEST (1s)	---
RY	timeRY (5s)	timeTEST (1s)	---
GR	timeGR (45s)	timeTEST (1s)	---
YR	timeYR (5s)	timeTEST (1s)	---
YY	---	---	Indefinite

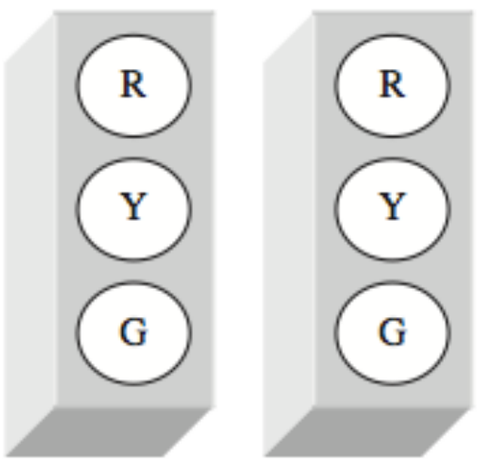
- If the machine is in **STANDBY** mode...
- Both traffic lights will be **Y**ellow, until the machine gets out of standby mode.
- This will happen in a sensor detects some problem in the circuitry.



Traffic Light Controller (TLC)



State	Operation Mode		
	REGULAR	TEST	STANDBY
	Time	Time	Time
RG	timeRG (30s)	timeTEST (1s)	---
RY	timeRY (5s)	timeTEST (1s)	---
GR	timeGR (45s)	timeTEST (1s)	---
YR	timeYR (5s)	timeTEST (1s)	---
YY	---	---	Indefinite



VHDL implementation

- Next slide will show the full VHDL implementation of this FSM, which is based on the FSM template #1.

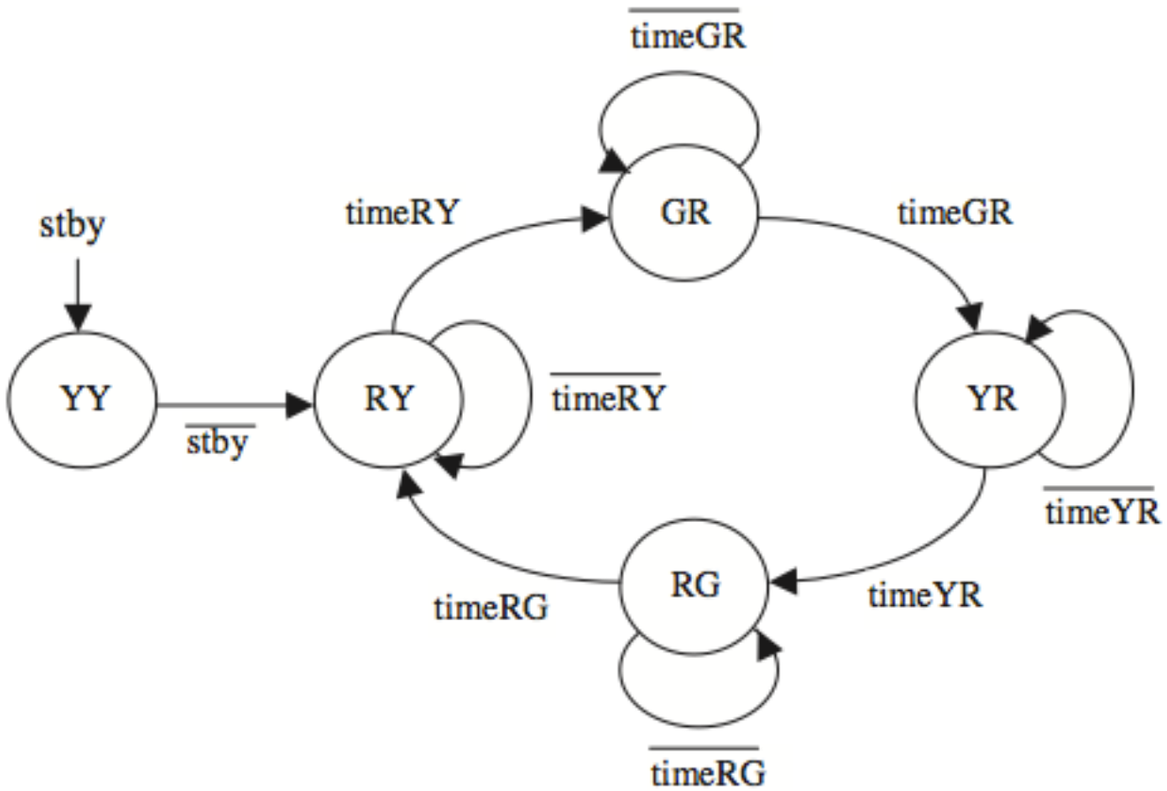
- Here is the entity part of the code

```
-----------------------------------------------
ENTITY tlc IS
   PORT ( clk, stby, test: IN STD_LOGIC;
          r1, r2, y1, y2, g1, g2: OUT STD_LOGIC);
END tlc;
-----------------------------------------------
```

- The signal clk is a 60Hz clock (60 beats per second)

- ... So there will be 60 * 5 beats in 5 seconds

State	Operation Mode		
	REGULAR	TEST	STANDBY
	Time	Time	Time
RG	timeRG (30s)	timeTEST (1s)	---
RY	timeRY (5s)	timeTEST (1s)	---
GR	timeGR (45s)	timeTEST (1s)	---
YR	timeYR (5s)	timeTEST (1s)	---
YY	---	---	Indefinite



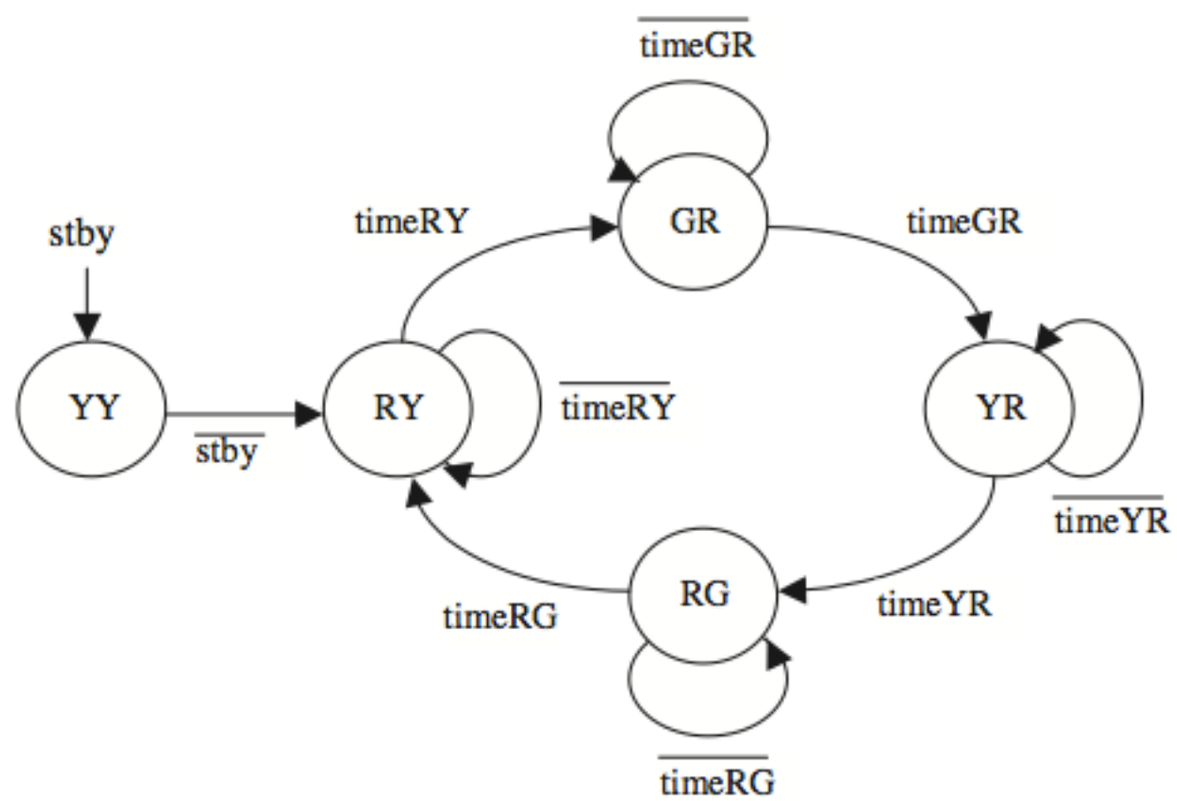
```
1  ------------------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------------------
5  ENTITY tlc IS
6     PORT ( clk, stby, test: IN STD_LOGIC;
7            r1, r2, y1, y2, g1, g2: OUT STD_LOGIC);
8  END tlc;
9  ------------------------------------------------
10 ARCHITECTURE behavior OF tlc IS
11    CONSTANT timeMAX : INTEGER := 2700;
12    CONSTANT timeRG : INTEGER := 1800;
13    CONSTANT timeRY : INTEGER := 300;
14    CONSTANT timeGR : INTEGER := 2700;
15    CONSTANT timeYR : INTEGER := 300;
16    CONSTANT timeTEST : INTEGER := 60;
17    TYPE state IS (RG, RY, GR, YR, YY);
18    SIGNAL pr_state, nx_state: state;
19    SIGNAL time : INTEGER RANGE 0 TO timeMAX;
20 BEGIN
21    -------- Lower section of state machine: ----
22    PROCESS (clk, stby)
23       VARIABLE count : INTEGER RANGE 0 TO timeMAX;
24    BEGIN
25       IF (stby='1') THEN
26          pr_state <= YY;
27          count := 0;
28       ELSIF (clk'EVENT AND clk='1') THEN
29          count := count + 1;
30          IF (count = time) THEN
31             pr_state <= nx_state;
32             count := 0;
33          END IF;
34       END IF;
35    END PROCESS;
36    -------- Upper section of state machine: ----
37    PROCESS (pr_state, test)
38    BEGIN
39       CASE pr_state IS
40          WHEN RG =>
41             r1<='1'; r2<='0'; y1<='0'; y2<='0'; g1<='0'; g2<='1';
42             nx_state <= RY;
43             IF (test='0') THEN time <= timeRG;
44             ELSE time <= timeTEST;
45             END IF;
46          WHEN RY =>
47             r1<='1'; r2<='0'; y1<='0'; y2<='1'; g1<='0'; g2<='0';
48             nx_state <= GR;
49             IF (test='0') THEN time <= timeRY;
50             ELSE time <= timeTEST;
51             END IF;
52          WHEN GR =>
53             r1<='0'; r2<='1'; y1<='0'; y2<='0'; g1<='1'; g2<='0';
54             nx_state <= YR;
55             IF (test='0') THEN time <= timeGR;
56             ELSE time <= timeTEST;
57             END IF;
58          WHEN YR =>
59             r1<='0'; r2<='1'; y1<='1'; y2<='0'; g1<='0'; g2<='0';
60             nx_state <= RG;
61             IF (test='0') THEN time <= timeYR;
62             ELSE time <= timeTEST;
63             END IF;
64          WHEN YY =>
65             r1<='0'; r2<='0'; y1<='1'; y2<='1'; g1<='0'; g2<='0';
66             nx_state <= RY;
67       END CASE;
68    END PROCESS;
69 END behavior;
70 ------------------------------------------------
```

```
1  ------------------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------------------
5  ENTITY tlc IS
6     PORT ( clk, stby, test: IN STD_LOGIC;
7            r1, r2, y1, y2, g1, g2: OUT STD_LOGIC);
8  END tlc;
9  ------------------------------------------------
10 ARCHITECTURE behavior OF tlc IS
11    CONSTANT timeMAX : INTEGER := 2700;
12    CONSTANT timeRG : INTEGER := 1800;
13    CONSTANT timeRY : INTEGER := 300;
14    CONSTANT timeGR : INTEGER := 2700;
15    CONSTANT timeYR : INTEGER := 300;
16    CONSTANT timeTEST : INTEGER := 60;
17    TYPE state IS (RG, RY, GR, YR, YY);
18    SIGNAL pr_state, nx_state: state;
19    SIGNAL time : INTEGER RANGE 0 TO timeMAX;
20 BEGIN
21    -------- Lower section of state machine: ----
22    PROCESS (clk, stby)
23       VARIABLE count : INTEGER RANGE 0 TO timeMAX;
24    BEGIN
25       IF (stby='1') THEN
26          pr_state <= YY;
27          count := 0;
28       ELSIF (clk'EVENT AND clk='1') THEN
29          count := count + 1;
30          IF (count = time) THEN
31             pr_state <= nx_state;
32             count := 0;
33          END IF;
34       END IF;
35    END PROCESS;
36    -------- Upper section of state machine: ----
37    PROCESS (pr_state, test)
38    BEGIN
39       CASE pr_state IS
40
41                                                  '0'; g2<='1';
42
43
44
45
46
47                                                  '0'; g2<='0';
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
```

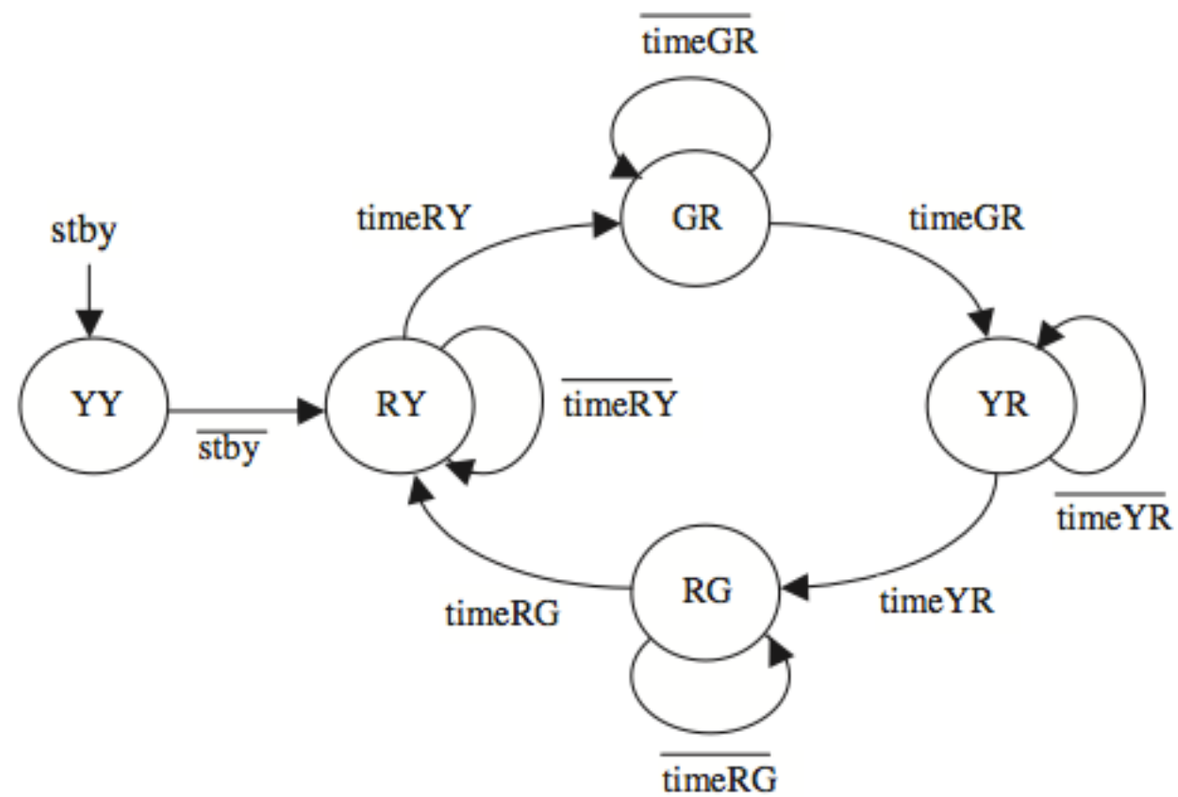
State	Operation Mode		
	REGULAR	TEST	STANDBY
	Time	Time	Time
RG	timeRG (30s)	timeTEST (1s)	---
RY	timeRY (5s)	timeTEST (1s)	---
GR	timeGR (45s)	timeTEST (1s)	---
YR	timeYR (5s)	timeTEST (1s)	---
YY	---	---	Indefinite

Defining different delays for each transition

Declaring all 5 states



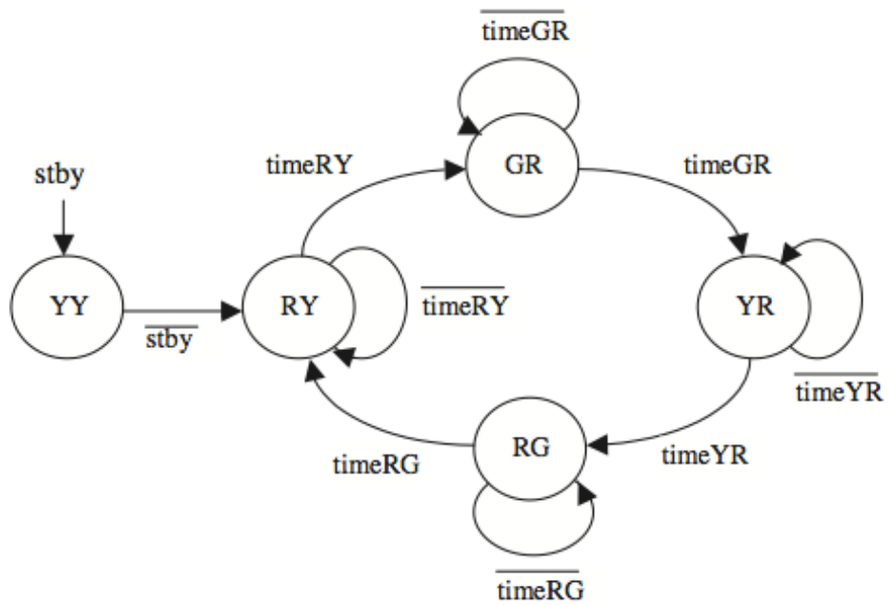
```
1  ------------------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------------------
5  ENTITY tlc IS
6     PORT ( clk, stby, test: IN STD_LOGIC;
7            r1, r2, y1, y2, g1, g2: OUT STD_LOGIC);
8  END tlc;
9  ------------------------------------------------
10 ARCHITECTURE behavior OF tlc IS
11    CONSTANT timeMAX : INTEGER := 2700;
12    CONSTANT timeRG : INTEGER := 1800;
13    CONSTANT timeRY : INTEGER := 300;
14    CONSTANT timeGR : INTEGER := 2700;
15    CONSTANT timeYR : INTEGER := 300;
16    CONSTANT timeTEST : INTEGER := 60;
17    TYPE state IS (RG, RY, GR, YR, YY);
18    SIGNAL pr_state, nx_state: state;
19    SIGNAL time : INTEGER RANGE 0 TO timeMAX;
20 BEGIN
21    -------- Lower section of state machine: ----
22    PROCESS (clk, stby)
23       VARIABLE count : INTEGER RANGE 0 TO timeMAX;
24    BEGIN
25       IF (stby='1') THEN
26          pr_state <= YY;
27          count := 0;
28       ELSIF (clk'EVENT AND clk='1') THEN
29          count := count + 1;
30          IF (count = time) THEN
31             pr_state <= nx_state;
32             count := 0;
33          END IF;
34       END IF;
35    END PROCESS;
```




```
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52          WHEN GR =>
53             r1<='0'; r2<='1'; y1<='0'; y2<='0'; g1<='1'; g2<='0';
54             nx_state <= YR;
55             IF (test='0') THEN time <= timeGR;
56             ELSE time <= timeTEST;
57             END IF;
58          WHEN YR =>
59             r1<='0'; r2<='1'; y1<='1'; y2<='0'; g1<='0'; g2<='0';
60             nx_state <= YY;
61             IF (test='0') THEN time <= timeYR;
62             ELSE time <= timeTEST;
63             END IF;
64          WHEN YY =>
65             r1<='0'; r2<='0'; y1<='1'; y2<='1'; g1<='0'; g2<='0';
66             nx_state <= RY;
67             END IF;
68       END PROCESS;
69    END behavior;
70 ------------------------------------------------
```

If standby sensor is active, will immediately go to YY state

Otherwise, increment the counter. If the appropriate time has elapsed, then we proceed to next state.



If we are in the RG state:

1. We set the appropriate outputs (r1 becomes 1, r2 becomes 0 and so on...)

2. We specify what is going to be the next state (RY)

3. If we are in the **TEST** mode of operation, then we will not wait for the time that is defined in the **REGULAR** mode of operation.

```
-------- Upper section of state machine: ----
PROCESS (pr_state, test)
BEGIN
   CASE pr_state IS
      WHEN RG =>
         r1<='1'; r2<='0'; y1<='0'; y2<='0'; g1<='0'; g2<='1';
         nx_state <= RY;
         IF (test='0') THEN time <= timeRG;
         ELSE time <= timeTEST;
         END IF;
      WHEN RY =>
         r1<='1'; r2<='0'; y1<='0'; y2<='1'; g1<='0'; g2<='0';
         nx_state <= GR;
         IF (test='0') THEN time <= timeRY;
         ELSE time <= timeTEST;
         END IF;
      WHEN GR =>
         r1<='0'; r2<='1'; y1<='0'; y2<='0'; g1<='1'; g2<='0';
         nx_state <= YR;
         IF (test='0') THEN time <= timeGR;
         ELSE time <= timeTEST;
         END IF;
      WHEN YR =>
         r1<='0'; r2<='1'; y1<='1'; y2<='0'; g1<='0'; g2<='0';
         nx_state <= RG;
         IF (test='0') THEN time <= timeYR;
         ELSE time <= timeTEST;
         END IF;
      WHEN YY =>
         r1<='0'; r2<='0'; y1<='1'; y2<='1'; g1<='0'; g2<='0';
         nx_state <= RY;
   END CASE;
END PROCESS;
END behavior;
-----------------------------------------------------
```