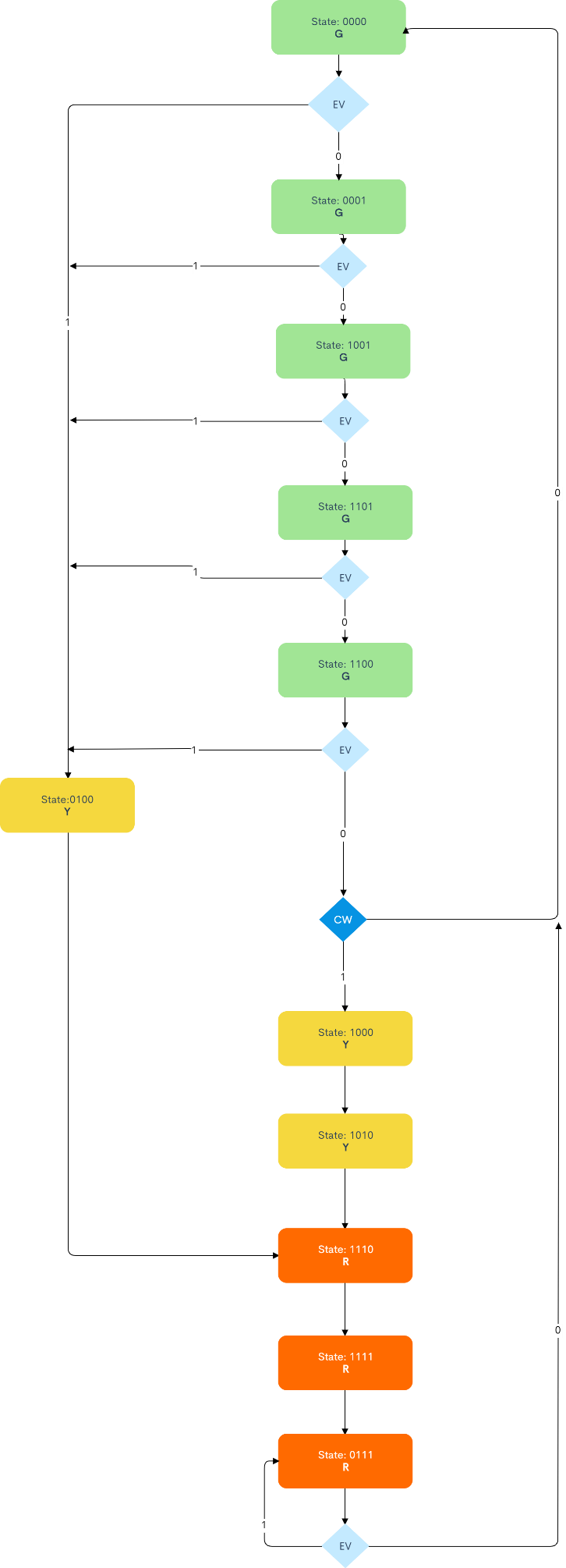
Compared to other concepts in this course state machines are particularly applicable because they have widespread use in practice. Many real world systems including traffic signals, scheduling systems, and embedded systems function as state machines. As such the ability to create a state machine with adequate planning and get the result one wants is invaluable. Being able to create state machines allows one to create controllable behavior in otherwise static circuitry so that we can create a wide array of consumer products and industry machinery which is operable by someone without a background in engineering by providing simple control systems which influence the state of the machine.



Behavior is as follows:

Start at State 0 and turn green for 5 cycles, if at any point an Emergency Vehicle approaches the light, turn yellow on the next cycle (State 4). If EV is never true, and no car is waiting – go back to State 0. If CW is true, turn yellow for 2 cycles and then turn red for 3 cycles. If on the third red cycle and EV is true, remain red. Otherwise go to State 0.

**NSTT:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S3** | **S2** | **S1** | **S0** | **CW** | **EV** | **S3+** | **S2+** | **S1+** | **S0+** | **G** | **Y** | **R** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |

**State-Bit Equations:**

S3+ (SOP) = (/S3/S2/S1S0/EV) + (S3/S2/S1S0/EV) + (S3S2/S1S0/EV) + (S3S2/S1/S0CW/EV) + (/S3S2/S1/S0) + (S3/S2/S1/S0) + (S3/S2S1/S0) + (S3S2S1/S0)

S2+ (POS) = (S3+S2+S1+S0+EV) \* (S3+S2+S1+/S0+EV) \* (/S3+/S2+S1+S0+EV) \* (/S3+S2+S1+S0) \* (S3+/S2+/S1+/S0+EV)

S1+ (POS) = (S3+S2+S1+S0) \* (S3+S2+S1+/S0) \* (/S3+S2+S1+/S0) \* (/S3+/S2+S1+/S0) \* (/S3+/S2+S1+S0) \* (S3+/S2+/S1+/S0+EV)

S0+ (SOP) = (/S3/S2/S1/S0/EV) + (/S3/S2/S1S0/EV) + (S3/S2/S1S0/EV) + (S3S2S1/S0) + (S3S2S1S0) + (/S3S2S1S0EV)

**GYR Truth Table (for clarity):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S3** | **S2** | **S1** | **S0** | **G** | **Y** | **R** |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | \* | \* | \* |
| 0 | 0 | 1 | 1 | \* | \* | \* |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | \* | \* | \* |
| 0 | 1 | 1 | 0 | \* | \* | \* |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | \* | \* | \* |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 |

**GYR Equations (State-Based):**

G = (/S3/S2/S1/S0) + (/S3/S2/S­1S0) + (S3/S2/S1S0) + (S3S2/S1/S0) + (S3S2/S1S0)

Y = (/S3S2/S1/S0) + (S3/S2/S1/S0) + (S3/S2S1/S0)

R = (/S3S2S1S0) + (S3S2S1/S0) + (S3S2S1S0)

**VHDL (included in archive):**

entity Lab5\_DFF\_Traf\_Cont is port (

State: in bit\_vector(3 downto 0);

CW, EV: in bit;

NewState: out bit\_vector(3 downto 0);

G,Y,R: out bit

);

end Lab5\_DFF\_Traf\_Cont;

architecture behavior of Lab5\_DFF\_Traf\_Cont is

signal Sbits: bit\_vector(3 downto 0); --Store state for GYR Calculations

begin

--State Bits Calculation

Sbits(3) <= ((not State(3)) and (not State(2)) and (not State(1)) and State(0) and (not EV))

or (State(3) and (not State(2)) and (not State(1)) and State(0) and (not EV))

or (State(3) and State(2) and (not State(1)) and State(0) and (not EV))

or (State(3) and State(2) and (not State(1)) and (not State(0)) and CW and (not EV))

or ((not State(3)) and State(2) and (not State(1)) and (not State(0)))

or (State(3) and (not State(2)) and (not State(1)) and (not State(0)))

or (State(3) and (not State(2)) and State(1) and (not State(0)))

or (State(3) and State(2) and State(1) and (not State(0)));

Sbits(2) <= (State(3) or State(2) or State(1) or State(0) or EV)

and (State(3) or State(2) or State(1) or (not State(0)) or EV)

and ((not State(3)) or (not State(2)) or State(1) or State(0) or EV)

and ((not State(3)) or State(2) or State(1) or State(0))

and (State(3) or (not State(2)) or (not State(1)) or (not State(0)) or EV);

Sbits(1) <= (State(3) or State(2) or State(1) or State(0))

and (State(3) or State(2) or State(1) or (not State(0)))

and ((not State(3)) or State(2) or State(1) or (not State(0)))

and ((not State(3)) or (not State(2)) or State(1) or (not State(0)))

and ((not State(3)) or (not State(2)) or State(1) or State(0))

and (State(3) or (not State(2)) or (not State(1)) or (not State(0)) or EV);

Sbits(0) <= ((not State(3)) and (not State(2)) and (not State(1)) and (not State(0)) and (not EV))

or ((not State(3)) and (not State(2)) and (not State(1)) and State(0) and (not EV))

or (State(3) and (not State(2)) and (not State(1)) and State(0) and (not EV))

or (State(3) and State(2) and State(1) and (not State(0)))

or (State(3) and State(2) and State(1) and State(0))

or ((not State(3)) and State(2) and State(1) and State(0) and EV);

--GYR Calculation

G <= ((not Sbits(3)) and (not Sbits(2)) and (not Sbits(1)) and (not Sbits(0)))

or ((not Sbits(3)) and (not Sbits(2)) and (not Sbits(1)) and Sbits(0))

or (Sbits(3) and (not Sbits(2)) and (not Sbits(1)) and Sbits(0))

or (Sbits(3) and Sbits(2) and (not Sbits(1)) and (not Sbits(0)))

or (Sbits(3) and Sbits(2) and (not Sbits(1)) and Sbits(0));

Y <= ((not Sbits(3)) and Sbits(2) and (not Sbits(1)) and (not Sbits(0)))

or (Sbits(3) and (not Sbits(2)) and (not Sbits(1)) and (not Sbits(0)))

or (Sbits(3) and (not Sbits(2)) and Sbits(1) and (not Sbits(0)));

R <= ((not Sbits(3)) and Sbits(2) and Sbits(1) and Sbits(0))

or (Sbits(3) and Sbits(2) and Sbits(1) and (not Sbits(0)))

or (Sbits(3) and Sbits(2) and Sbits(1) and Sbits(0));

--Set State+

NewState <= Sbits;

end behavior;