Lecture 23

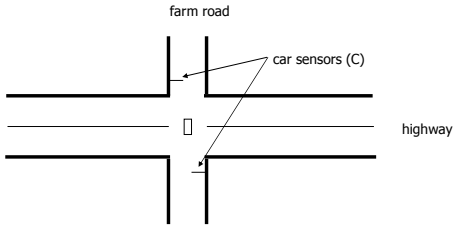
- Design example: Traffic light controller

FSM design process

1. Understand the problem
   – State diagram and state-transition table
2. Determine the machine's states
   – Consider missing transitions: Will the machine start?
   – Minimize the state diagram: Reuse states where possible
3. Encode the states
   – Encode states, outputs with a reasonable encoding choice
   – Consider the implementation target
4. Design the next-state logic
   – Minimize the combinational logic
   – Choices made in steps 2 & 3 affect the logic complexity
5. Implement the FSM

Example: Traffic light controller

- Highway/farm road intersection



Example: Traffic light controller

- A busy highway is intersected by a little used farm road
- Detectors C sense the presence of cars on the farm road
  - With no car on farm road, lights remain Green in highway direction
  - If vehicle on farm road, highway lights go from Green to Yellow to Red, allowing the farm road lights to become Green
  - These stay Green only as long as a farm road car is detected but never longer than a set interval (say, 20 time units)
  - When these are met, farm lights transition from Green to Yellow to Red, allowing highway to return to Green
  - Even if farm road vehicles are waiting, highway gets at least a set interval as Green (say, 20 time units)

What are inputs and outputs?

- Inputs
  - Reset
  - C (farm road car detector)
- Outputs
  - HG (highway green)
  - HY (highway yellow)
  - HR (highway red)
  - FG (farm road green)
  - FY (farm road yellow)
  - FR (farm road red)

What are the states?

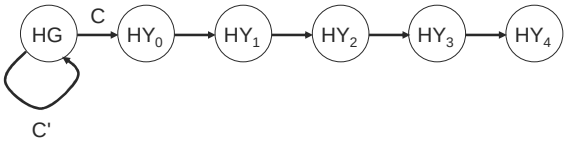
- How about color of the lights?
  - HG (highway green)
  - HY (highway yellow)
  - HR (highway red)
  - FG (farm road green)
  - FY (farm road yellow)
  - FR (farm road red)

Any redundancy?

- If the highway light is green or yellow, the farm road light must be red.
- States
  - HG (highway green)
  - HY (highway yellow)
  - FG (farm road green)
  - FY (farm road yellow)

Partial traffic light controller

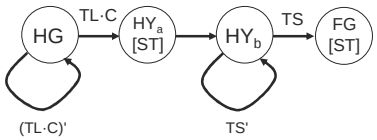
- Useful to walk through typical execution sequence



- What's the problem?

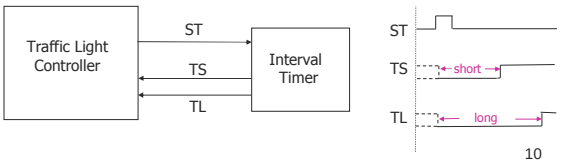
FSM can get very big!

- Utilize external timer like a subroutine



External timer

- ST resets timer and starts counting
- Timer generates a short time pulse (TS) and a long time pulse (TL)
- TS is to be used for timing yellow lights and TL for green lights



Revised FSM specification

- **Inputs**

reset	place FSM in initial state
C	detect vehicle on the farm road
TS	short time interval expired
TL	long time interval expired

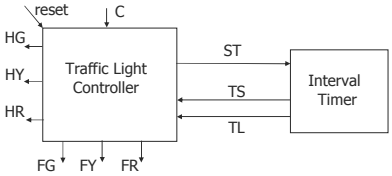
- **Outputs**

HG, HY, HR	assert green/yellow/red highway lights
FG, FY, FR	assert green/yellow/red farm road lights
ST	start timing a short or long interval

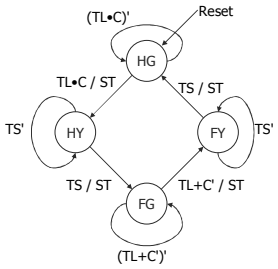
- **State**

HG	highway green (farm road red)
HY	highway yellow (farm road red)
FG	farm road green (highway red)
FY	farm road yellow (highway red)

Traffic light controller



Moore/Mealy hybrid



State encoding

- Let's use a one-hot encoding:
  - HG = 0001
  - HY = 0010
  - FG = 0100
  - FY = 1000

Next-state logic

- $P_3 = (C' \cdot Q_2) + (TL \cdot Q_2) + (TS' \cdot Q_3)$
- $P_2 = (TS \cdot Q_1) + (C \cdot TL' \cdot Q_2)$
- $P_1 = (C \cdot TL \cdot Q_0) + (TS' \cdot Q_1)$
- $P_0 = (C' \cdot Q_0) + (TL' \cdot Q_0) + (TS \cdot Q_3)$

- $ST = (C \cdot TL \cdot Q_0) + (TS \cdot Q_1) + (C' \cdot Q_2) + (TL \cdot Q_2) + (TS \cdot Q_3)$

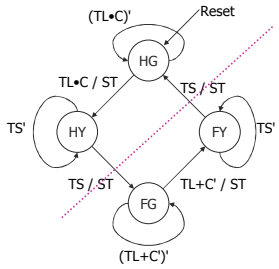
Outputs

- Green = 00
- Yellow = 01
- Red = 10

- Two sets of outputs: H_1H_0 and F_1F_0

- $H_1 = Q_3 + Q_2$
- $H_0 = Q_1$
- $F_1 = Q_1 + Q_0$
- $F_0 = Q_3$

State partitioning



State partitioning