CPE201
Digital Design

By Benjamin Haas

Class 25: Counter Design

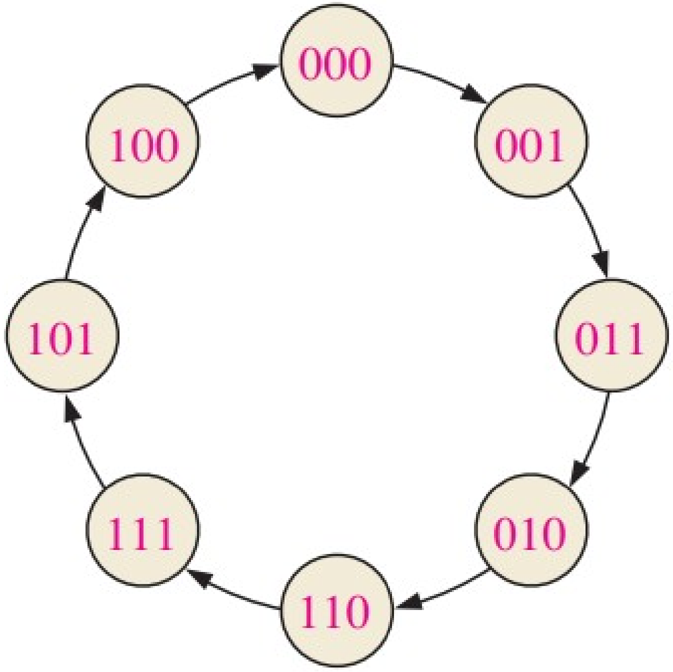
University of Nevada, Reno

Outline

- Designing Counters
  - Examples
- System Design Example

Design

- Step 1 - Create state machine
  - For counters, the state is the output
- Ex: 3-bit gray code counter

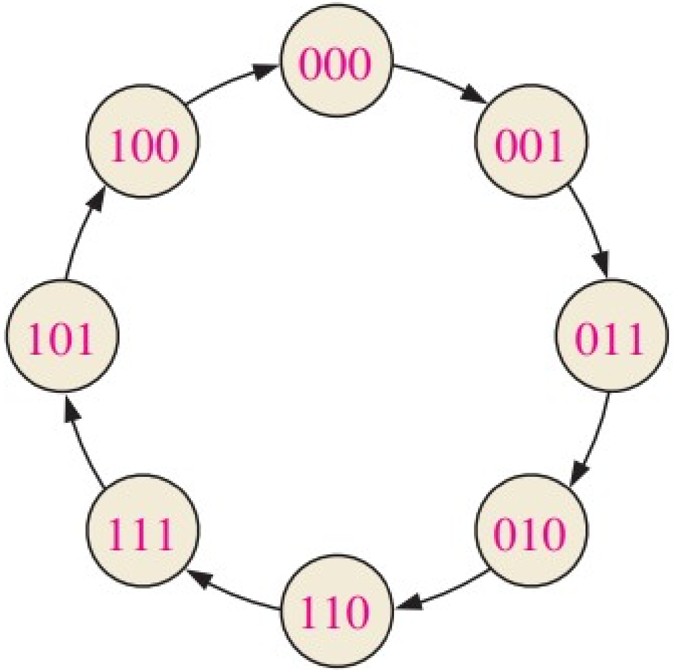


Design

- Step 2 – Next state table

Next-state table for 3-bit Gray code counter.

Present State			Next State		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	1	0	1	0
0	1	0	1	1	0
1	1	0	1	1	1
1	1	1	1	0	1
1	0	1	1	0	0
1	0	0	0	0	0



Design

- Step 3 – Map state transitions to circuit inputs

his case

Next-state table for 3-bit Gray code counter.

Present State			Next State		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	1	0	1	0
0	1	0	1	1	0
1	1	0	1	1	1
1	1	1	1	0	1
1	0	1	1	0	0
1	0	0	0	0	0

Transition table for a J-K flip-flop.

Output Transitions			Flip-Flop Inputs	
Q_N		Q_{N+1}	J	K
0	$\longrightarrow$	0	0	X
0	$\longrightarrow$	1	1	X
1	$\longrightarrow$	0	X	1
1	$\longrightarrow$	1	X	0

Q_N: present state
Q_{N+1}: next state
X: "don't care"

Design

- Step 3b – Table of inputs to create next

J_0	K_0	Q_2	Q_1	Q_0	Q_0 next
1	X	0	0	0	1
X	0	0	0	1	1
X	1	0	1	1	0
0	X	0	1	0	0
1	X	1	1	0	1
X	0	1	1	1	1
X	1	1	0	1	0

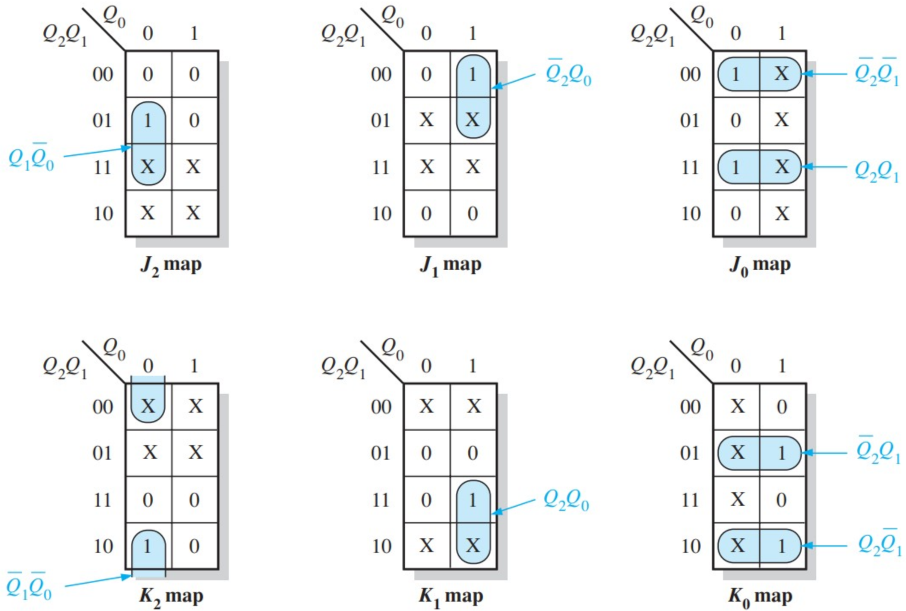
Transition table for a J-K flip-flop.

Output Transitions			Flip-Flop Inputs	
Q_N		Q_{N+1}	J	K
0	$\longrightarrow$	0	0	X
0	$\longrightarrow$	1	1	X
1	$\longrightarrow$	0	X	1
1	$\longrightarrow$	1	X	0

Q_N: present state
Q_{N+1}: next state
X: "don't care"

Design

- Step 4 – Karnaugh Maps
  - JK FFs are ACTIVE HIGH, so S

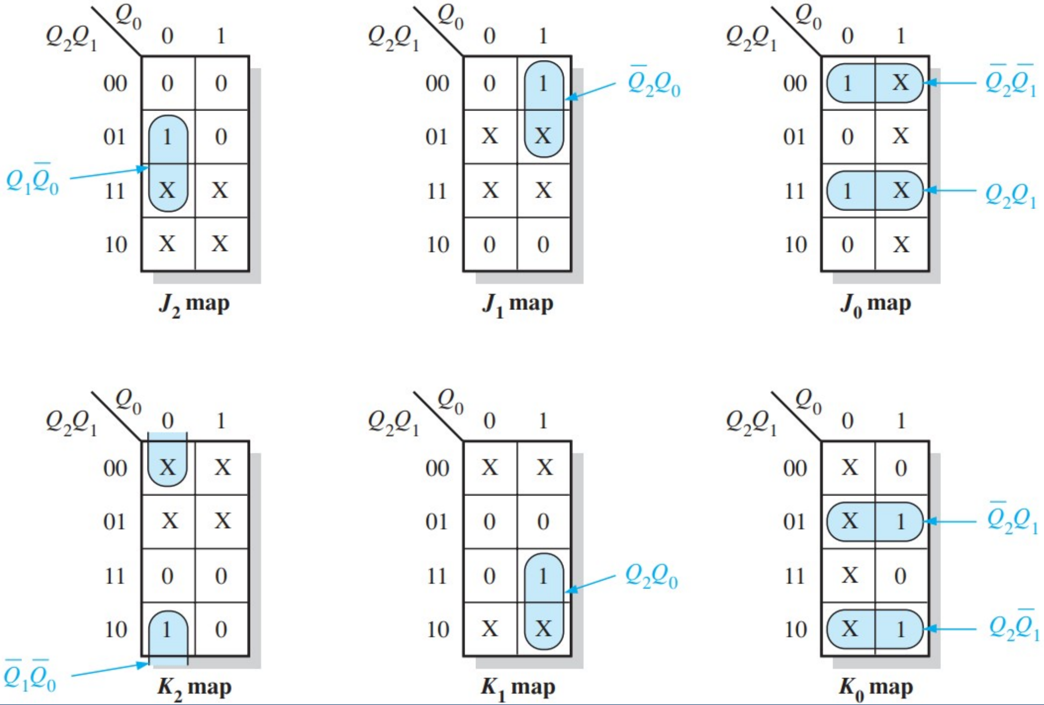


J_0	K_0	Q_2	Q_1	Q_0	Q_0 next
1	X	0	0	0	1
X	0	0	0	1	1
X	1	0	1	1	0
0	X	0	1	0	0
1	X	1	1	0	1
X	0	1	1	1	1
X	1	1	0	1	0

Design

- Step 5 – Logical Expressions

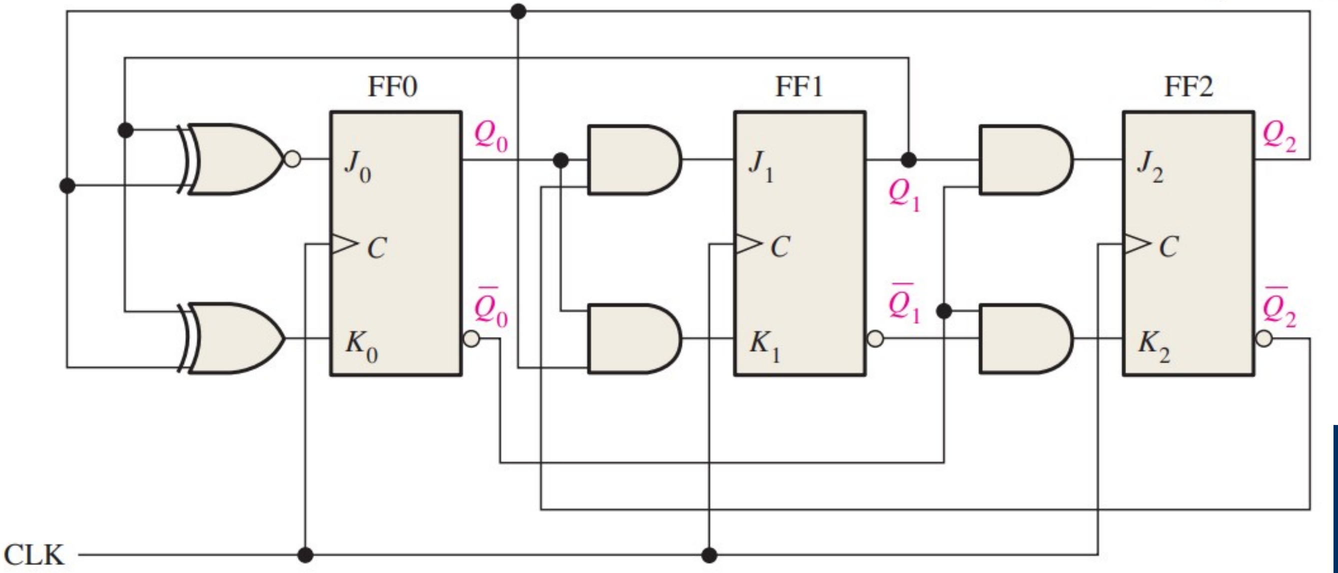
$$J_0 = Q_2 Q_1 + \overline{Q}_2 \overline{Q}_1 = \overline{Q_2 \oplus Q_1}$$
$$K_0 = Q_2 \overline{Q}_1 + \overline{Q}_2 Q_1 = Q_2 \oplus Q_1$$
$$J_1 = \overline{Q}_2 Q_0$$
$$K_1 = Q_2 Q_0$$
$$J_2 = Q_1 \overline{Q}_0$$
$$K_2 = \overline{Q}_1 \overline{Q}_0$$



Design

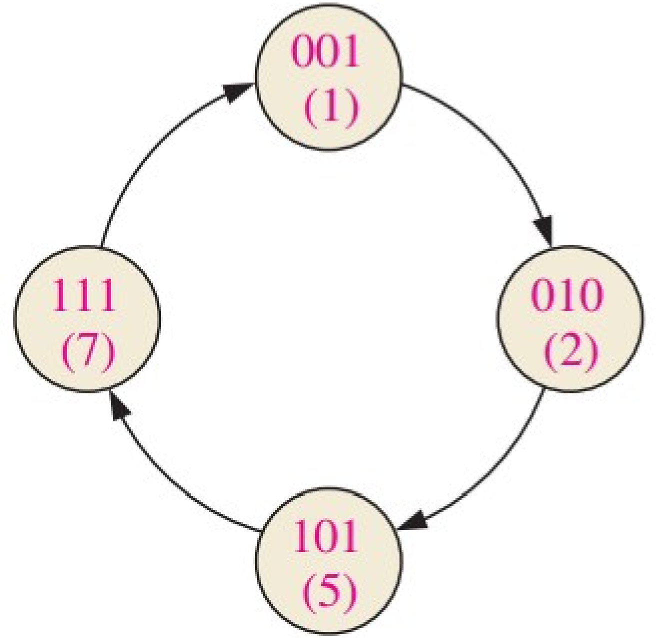
$$J_0 = Q_2Q_1 + \overline{Q}_2\overline{Q}_1 = \overline{Q_2 \oplus Q_1}$$
$$K_0 = Q_2\overline{Q}_1 + \overline{Q}_2Q_1 = Q_2 \oplus Q_1$$
$$J_1 = \overline{Q}_2Q_0$$
$$K_1 = Q_2Q_0$$
$$J_2 = Q_1\overline{Q}_0$$
$$K_2 = \overline{Q}_1\overline{Q}_0$$

- Step 6 - Implementation



Example

- 3-bit irregular counter
  - State machine given (step
  - Implement with D FFs



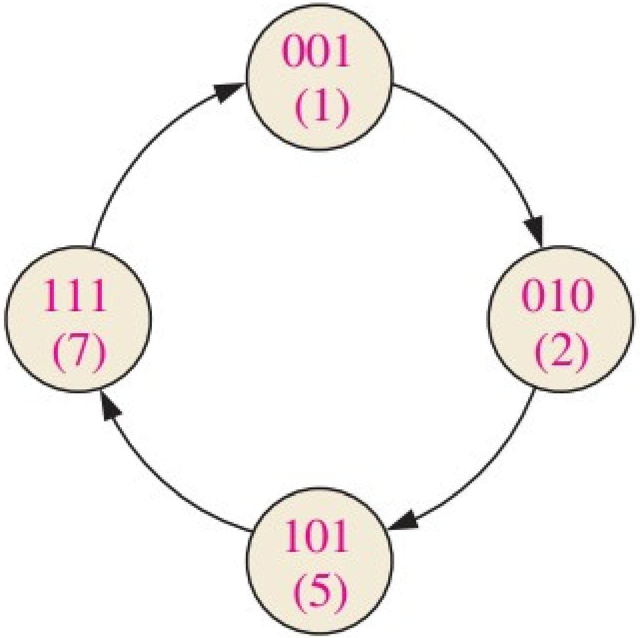
University of Nevada, Reno

Example

- Step 2 – next state table

Next-state table.

Present State			Next State		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	1	0	1	0
0	1	0	1	0	1
1	0	1	1	1	1
1	1	1	0	0	1



Example

- Step 3b - Table of inputs to create next

Transition table for a D flip-flop.

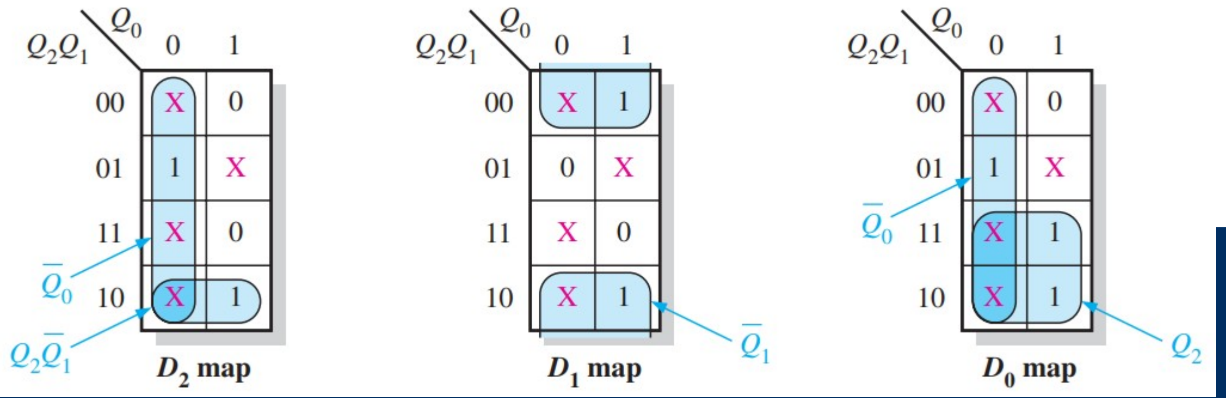
Output Transitions			Flip-Flop Input
Q_N		Q_{N+1}	D
0	$\longrightarrow$	0	0
0	$\longrightarrow$	1	1
1	$\longrightarrow$	0	0
1	$\longrightarrow$	1	1

D_0	Q_2	Q_1	Q_0	Q_0 next
0	0	0	1	0
1	0	1	0	1
1	1	1	1	1
1	0	0	1	1

Example

- Step 4 Karnaugh Maps
  - Pink X's are Don't Care because the state is not in the state machine
  - D FFs are ACTIVE HIGH, so SOP

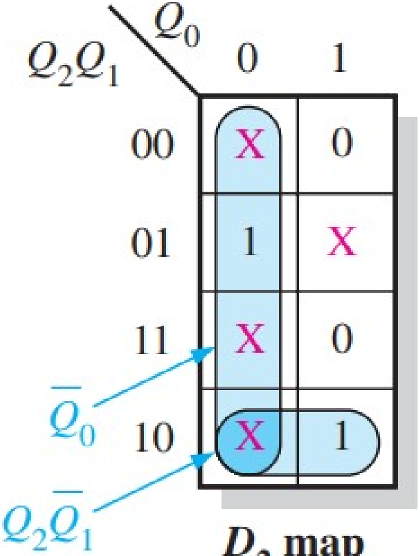
D_0	Q_2	Q_1	Q_0	Q_0 next
0	0	0	1	0
1	0	1	0	1
1	1	1	1	1



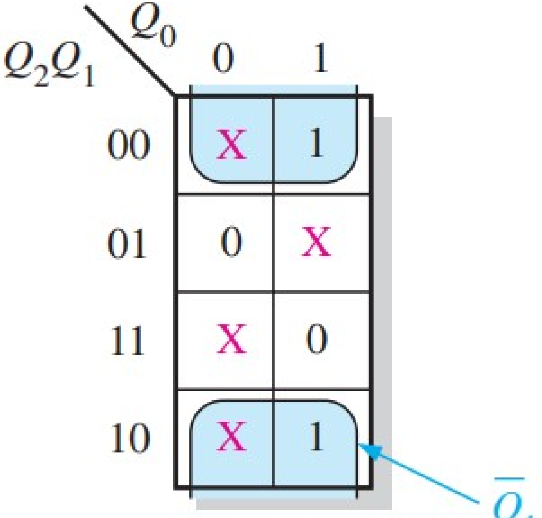
Example

- Step 5 – Logical Expressions

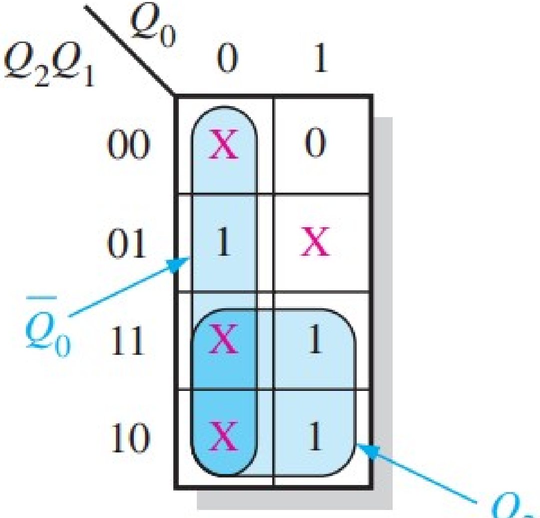
$$D_0 = \overline{Q}_0 + Q_2$$
$$D_1 = \overline{Q}_1$$
$$D_2 = \overline{Q}_0 + Q_2\overline{Q}_1$$



D_2 map



D_1 map

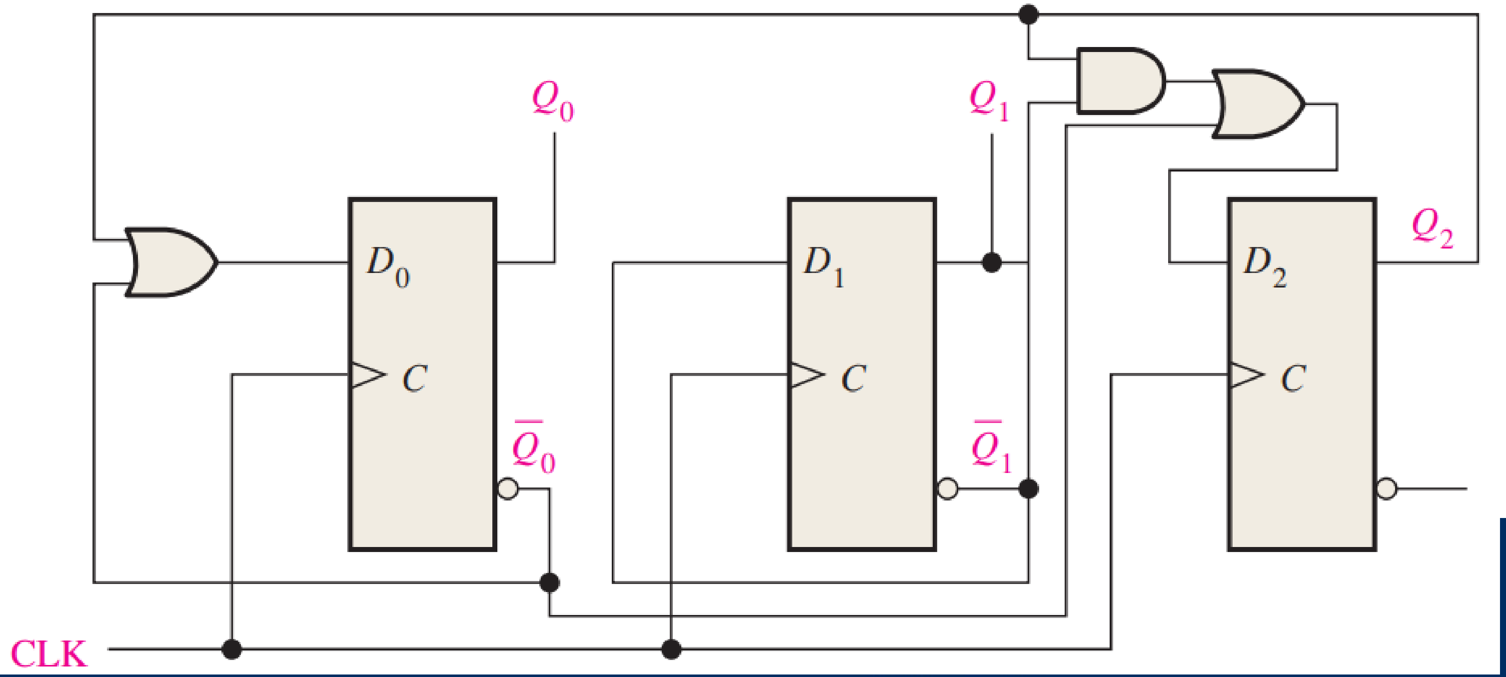


D_0 map

Example

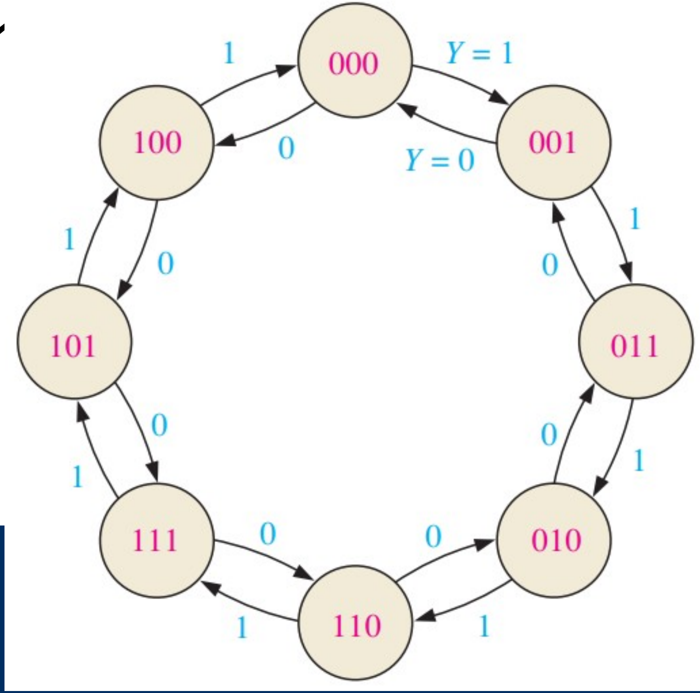
$$D_0 = \overline{Q}_0 + Q_2$$
$$D_1 = \overline{Q}_1$$
$$D_2 = \overline{Q}_0 + Q_2\overline{Q}_1$$

- Step 6 - Implementation



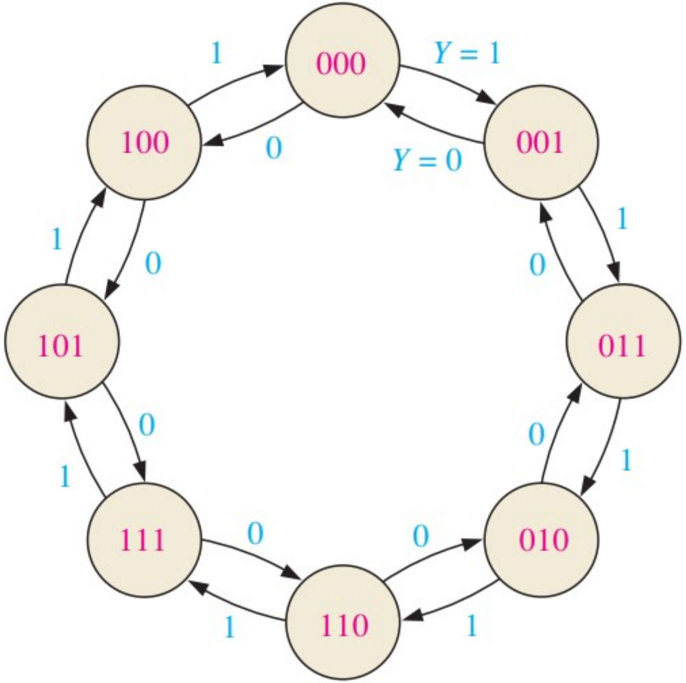
Example 2

- 3-bit Gray Code up/down counter
  - State machine given (step
  - Implement with JK FFs



Example 2

- Step 2 – Next state table



Next-state table for 3-bit up/down Gray code counter.

Present State			Next State					
			$Y = 0$ (DOWN)			$Y = 1$ (UP)		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	1
0	1	1	0	0	1	0	1	0
0	1	0	0	1	1	1	1	0
1	1	0	0	1	0	1	1	1
1	1	1	1	1	0	1	0	1
1	0	1	1	1	1	1	0	0
1	0	0	1	0	1	0	0	0

Y = UP/$\overline{\text{DOWN}}$ control input.

Example 2

- Consider the up/down' bit to be part of the state

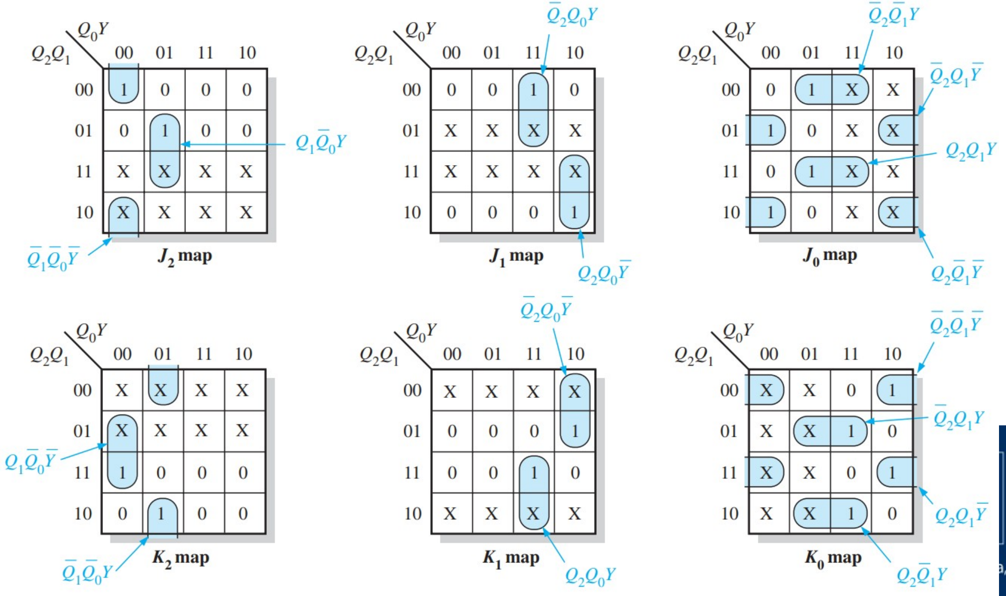
$Y = 0$ (DOWN)

$Y = 1$ (UP)

Present State				Next State		
Y	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	0	1	0	0
0	0	0	1	0	0	0
0	0	1	1	0	0	1
0	0	1	0	0	1	1
0	1	1	0	0	1	0
0	1	1	1	1	1	0
0	1	0	1	1	1	1
0	1	0	0	1	0	1
1	0	0	0	0	0	1
1	0	0	1	0	1	1
1	0	1	1	0	1	0
1	0	1	0	1	1	0
1	1	1	0	1	1	1
1	1	1	1	1	0	1
1	1	0	1	1	0	0
1	1	0	0	0	0	0

Example 2

- Step 3b – Same process as before
- Step 4 – Karnaugh Maps for 4 inputs



Step 5 and Step 6 are the same process as before

Traffic Signal Design

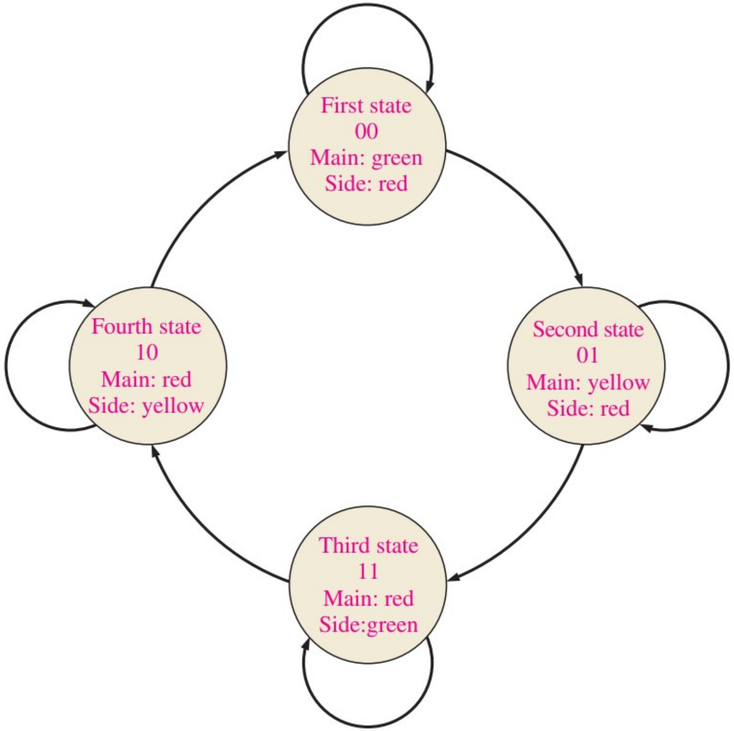
- Lights at a busy road perpendicular to a side street
- Requirements:
  - Green light on main for at least 25s before a change
  - Green light on side street as long as there are vehicles, up to 25s max
  - Yellow for 4s on both streets

Traffic Signal Design

- Inputs:
  - Vehicle sensor input (1 = vehicle present)
  - System clock (1Hz)
- Outputs:
  - Red, yellow, green signals for both streets (1 = on)

Traffic Signal Design

- Basic state machine
  - You know how traffic lights work
  - Also added gray code to represent each state as a number



Traffic Signal Design

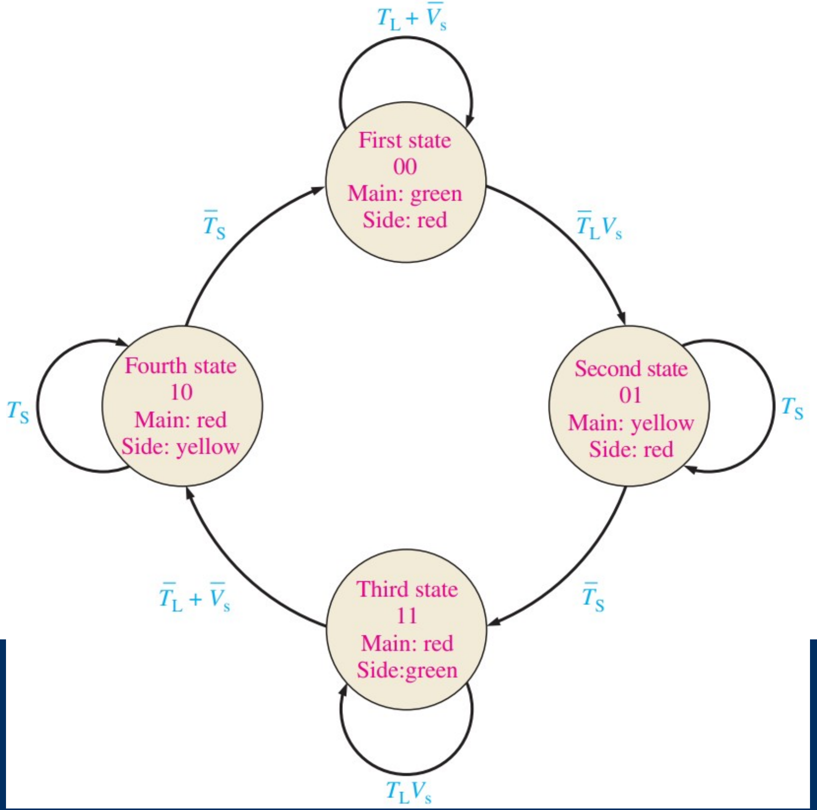
- Let's describe the transitions
  - First state is 'normal', takes a vehicle on side street and 25s to have passed to change to next state
  - Second state, wait for 4s on yellow
  - Third state, stay here as long as a vehicle is present and it is less than 25s in this state
  - Fourth state, same as second state

Traffic Signal Design

- Now make some inputs/variables for the state machine
  - Make from state transitions, make vars binary
  - V_S = vehicle present on side street
  - T_L = long timer (25s) is on
  - T_S = short timer (4s) is on

Traffic Signal Design

- Note that each transition condition is literally the NOT of the looping condition



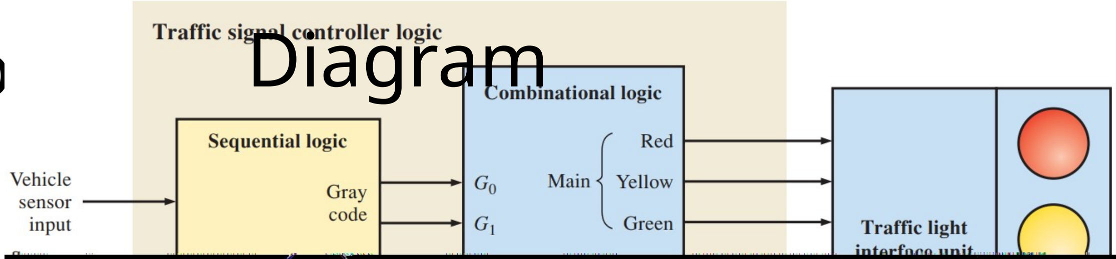
Traffic Signal Design

- The controller can be broken into several parts
  - Sequential logic - takes in all inputs on state machine and outputs state gray code
  - Combinational Logic – decodes state, triggers timers, and creates light outputs
  - Timers – input triggers and clock, outputs for if each timer is running (creates T_L and T_S)

Traffic Signal Design

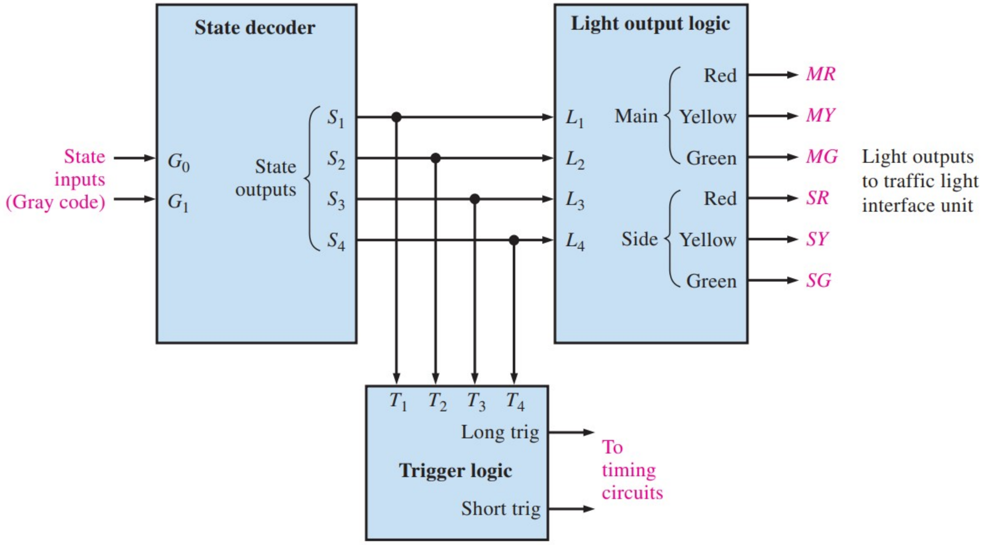
- Break each block down until it does one thing
  - Sequential is co [redacted]
  - Timers
  - Combination st [redacted]
    does too much [redacted]

Block Diagram



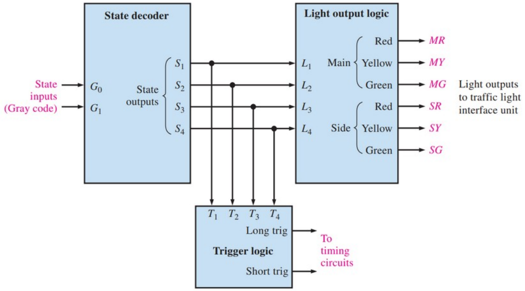
Traffic Signal Design

- Do your thinking in blocks to simplify problems
  - State decoder – current state is HIGH
  - State directly ties to light colors and starts a timer



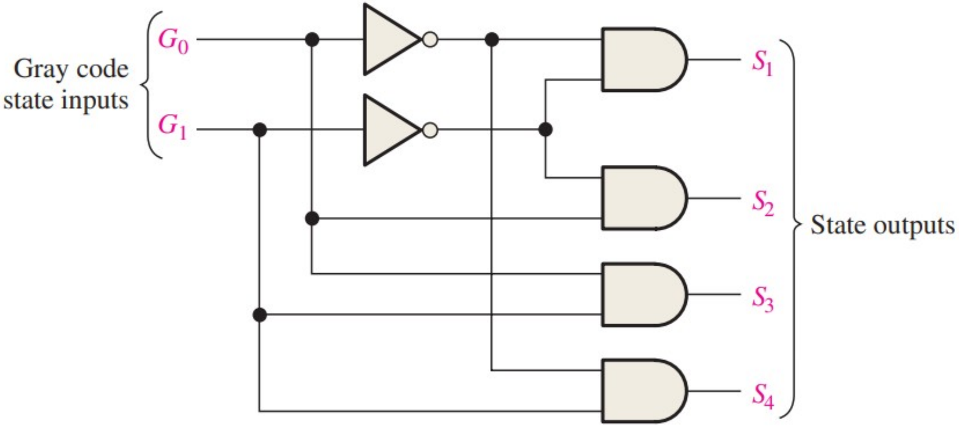
Traffic Signal Design

- State decoder from truth table



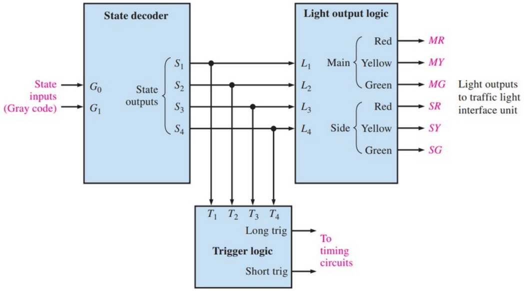
Truth table for the state decoder.

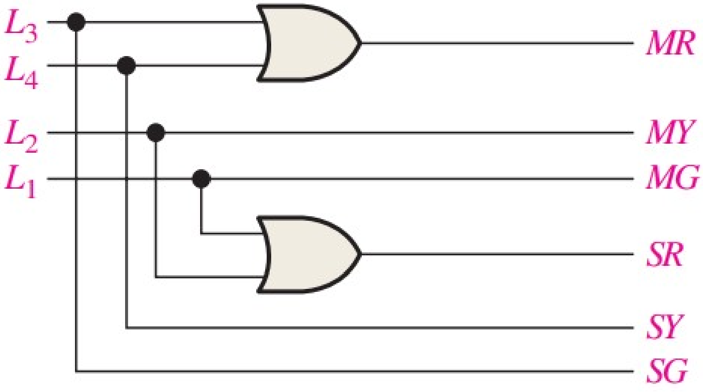
State Inputs (Gray Code)		State Outputs			
G_1	G_0	S_1	S_2	S_3	S_4
0	0	1	0	0	0
0	1	0	1	0	0
1	1	0	0	1	0
1	0	0	0	0	1

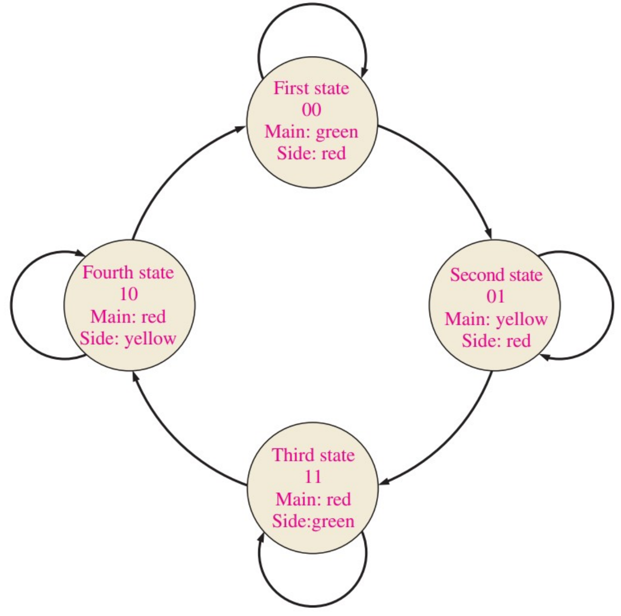


Traffic Signal Design

- Inputs/Outputs are not usually named the same
- MR = L3 + L4
- MY = L2
- MG = L1
- SR = L1 + L2
- SY = L4
- SG = L3

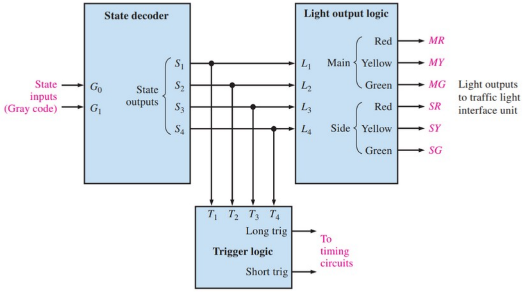


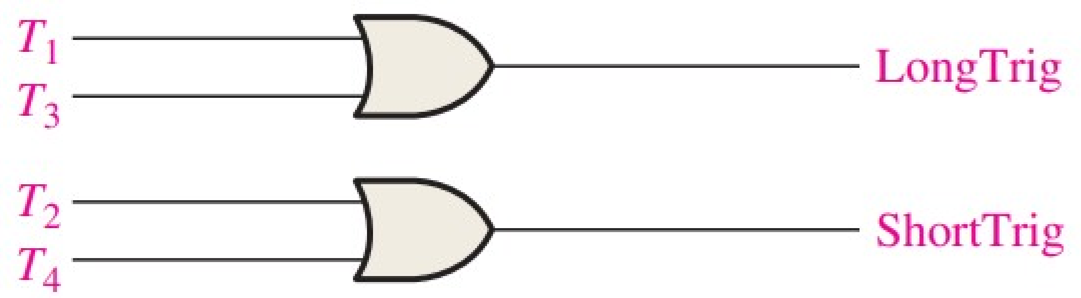


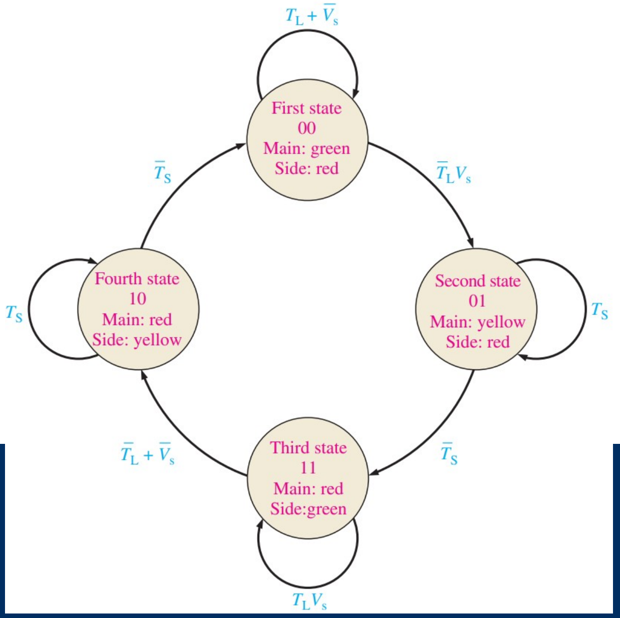


Traffic Signal Design

- S1 and S3 start long timer
- S2 and S4 start short timer

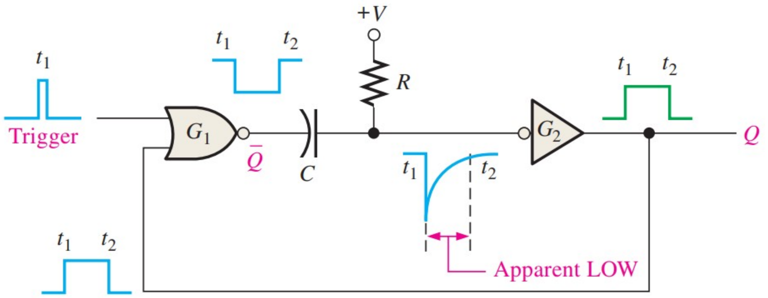


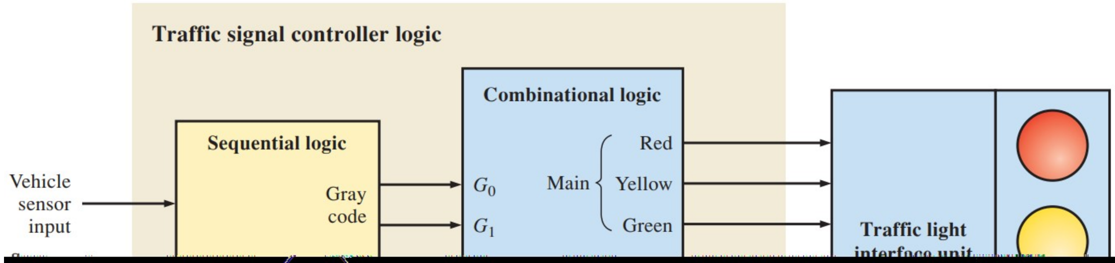




Traffic Signal Design

- TS and TL are high when timer is running, just like a one shot





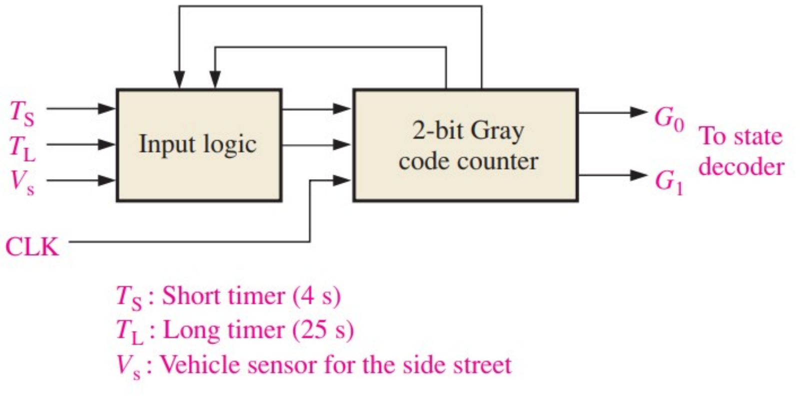
[redacted]

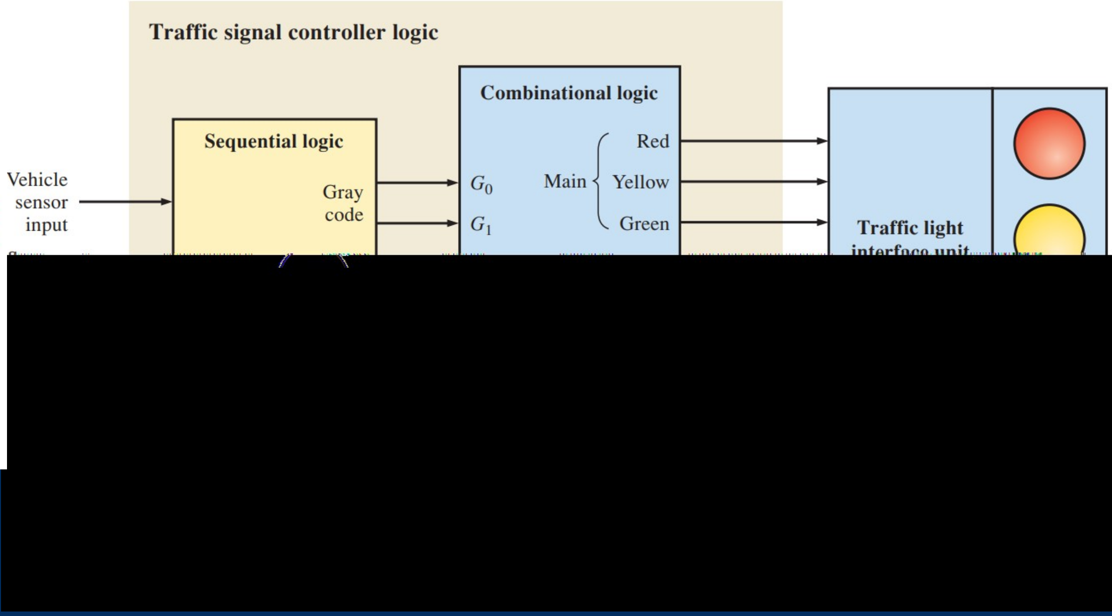
Traffic Signal Design

- $t_W = 0.7RC$
- $4s = 0.7R_{4s}C_{4s}$
- If $C_{4s} = 1000\mu F$, then $R_{4s} = 5{,}714\Omega$
- $25s = 0.7R_{25s}C_{25s}$
- If $C_{25s} = 1000\mu F$, then $R_{25s} = 35{,}714\Omega$

Traffic Signal Design

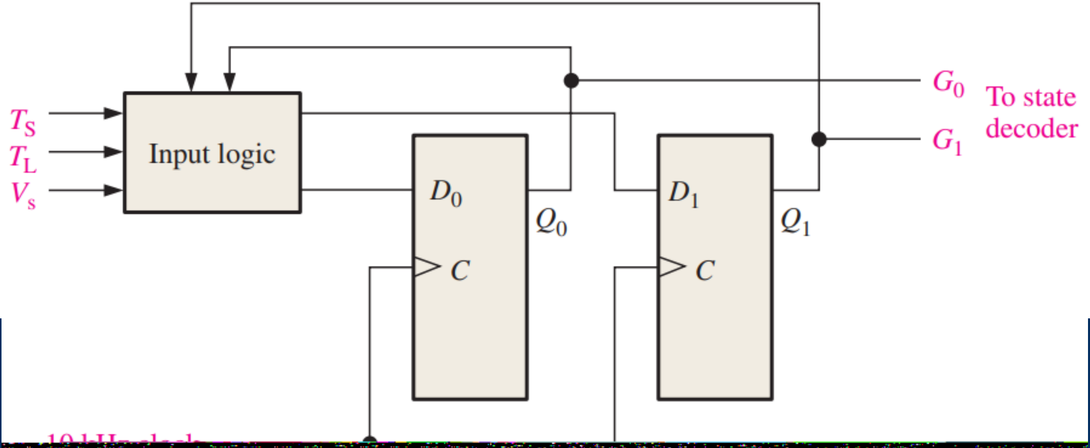
- One chunk left, it's a 2-bit counter

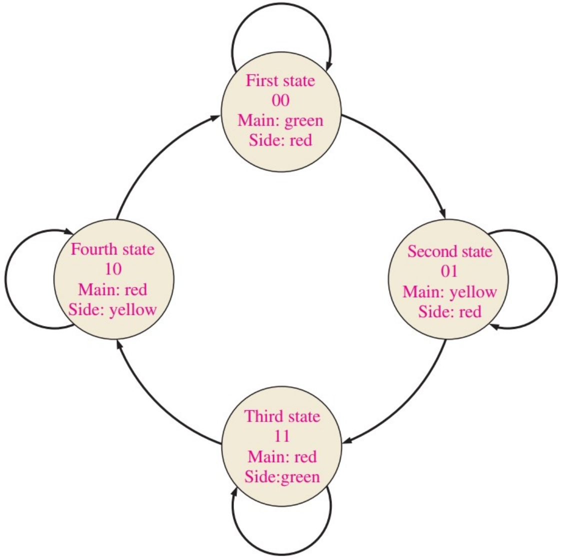




Traffic Signal Design

- We’ll use D FFs
  - Arbitrarily pick system clock at 10kHz
  - Step 1 - State machine is giver





Traffic Signal Design

- Step 2- Next state table

[redacted]

Traffic Signal Design

- Karnaugh maps would be very complicated with 5 variables (G_1, G_0, V_S, T_L, T_S)
  - Also a sparse table, so you can opt to not simplify the circuit to be quick
  - Write out the terms that make $D_0 = 1$

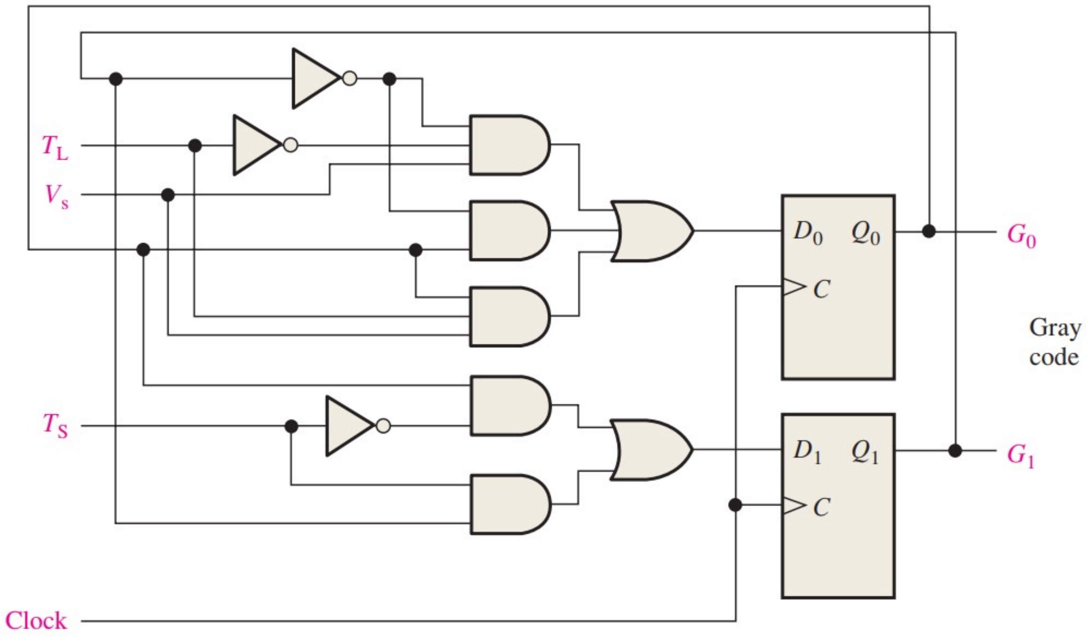
Traffic Signal Design

- $D_0 = G_1G_0'T_L'V_S + G_1G_0'T_S + G_1G_0T_S' + G_1G_0T_LV_S$
- $D_1 = G_1'G_0T_S' + G_1$ [redacted]

  $G_1G_0'T_S$

Traffic Signal Design

- That's all the pieces!



Reading

- This lecture
  - 9.5, Ch6 and Ch7 Applied Logic
- Next lecture
  - Sections 12.1-12.3