

# Fundamentals of Computer Systems

## Sequential Logic

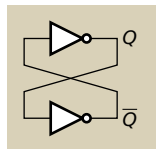
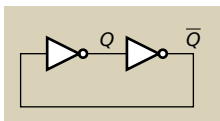
Martha A. Kim

Columbia University

Fall 2013

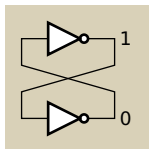
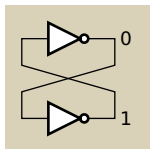


# Bistable Elements

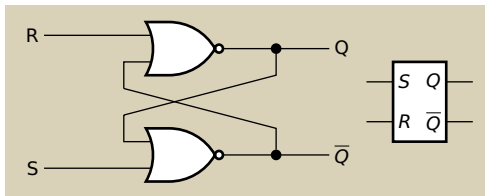


Equivalent circuits; right is more traditional.

Two stable states:

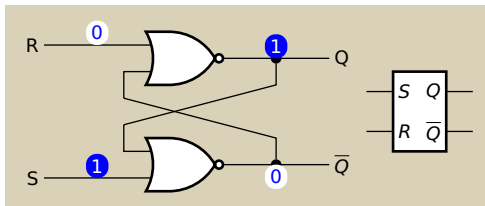


# SR Latch



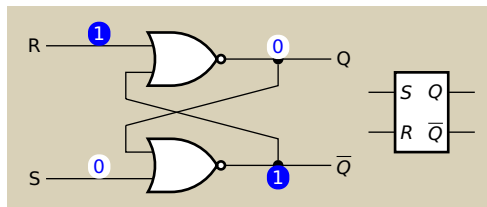
$R$	$S$	$Q$	$\bar{Q}$
0	0		
0	1		
1	0		
1	1		

# SR Latch



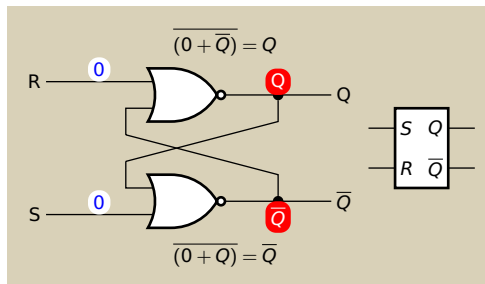
R	S	Q	$\bar{Q}$	
0	0			
0	1	1	0	Set ( $Q = 1$ )
1	0			
1	1			

# SR Latch



R	S	Q	$\bar{Q}$	
0	0			
0	1	1	0	Set ( $Q = 1$ )
1	0	0	1	Reset ( $Q = 0$ )
1	1			

# SR Latch

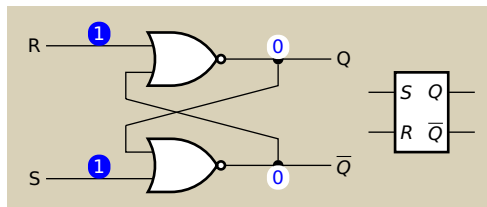


---

R	S	Q	$\bar{Q}$	
0	0	Q	$\bar{Q}$	Hold previous value
0	1	1	0	Set ( $Q = 1$ )
1	0	0	1	Reset ( $Q = 0$ )
1	1			

---

# SR Latch



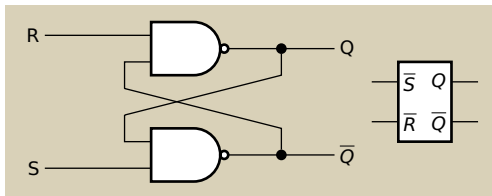
---

R	S	Q	$\bar{Q}$	
0	0	Q	$\bar{Q}$	Hold previous value
0	1	1	0	Set ( $Q = 1$ )
1	0	0	1	Reset ( $Q = 0$ )
1	1	0	0	Bad. Do not use.

---



# $\overline{S}\overline{R}$ Latch

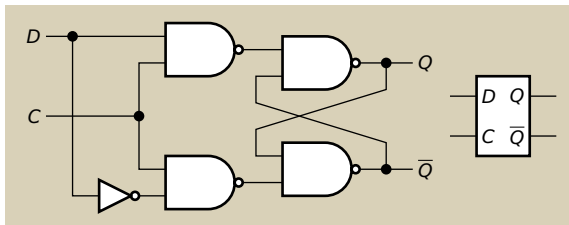


---

$R$	$S$	$Q$	$\overline{Q}$	
0	0	1	1	Bad. Do not use.
0	1	0	1	Reset ( $Q = 0$ )
1	0	1	0	Set ( $Q = 1$ )
1	1	$Q$	$\overline{Q}$	Hold previous value

---

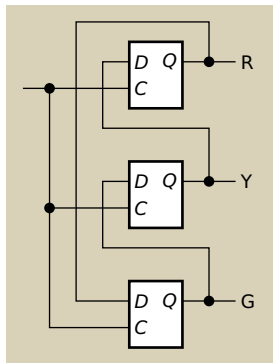
# D Latch



$C$	$D$	$Q$	$\bar{Q}$
0	X	$Q$	$\bar{Q}$
1	0	0	1
1	1	1	0

# A Challenge: Build a traffic light controller

Want the lights to cycle green-yellow-red.

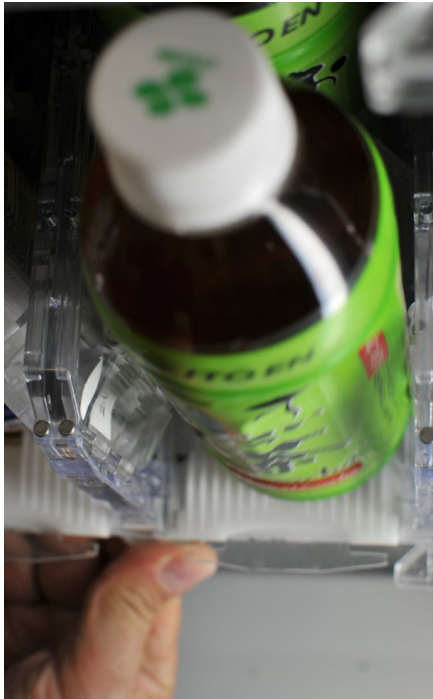


Does this work?







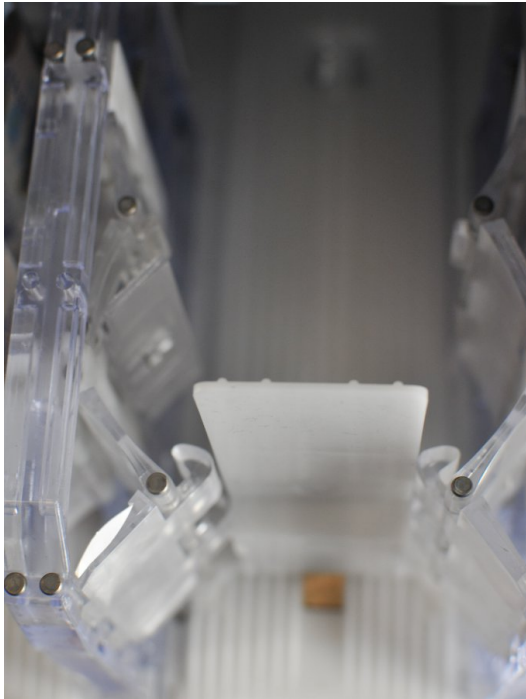


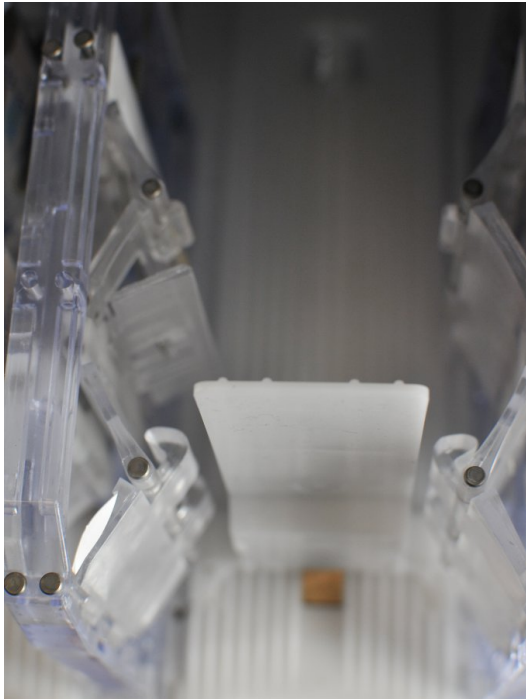


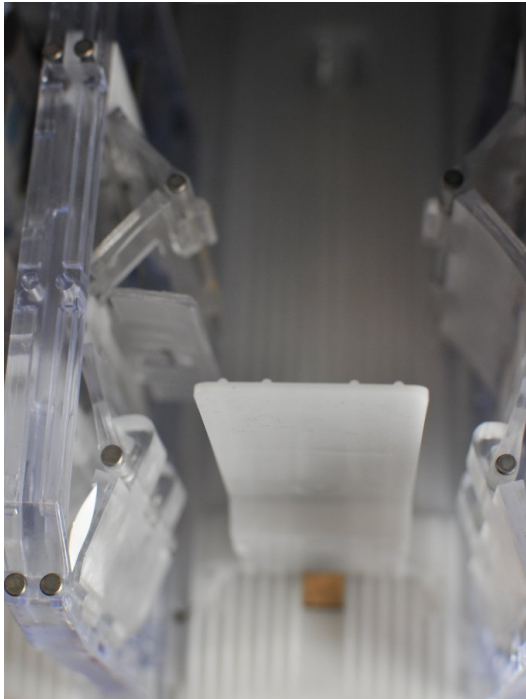


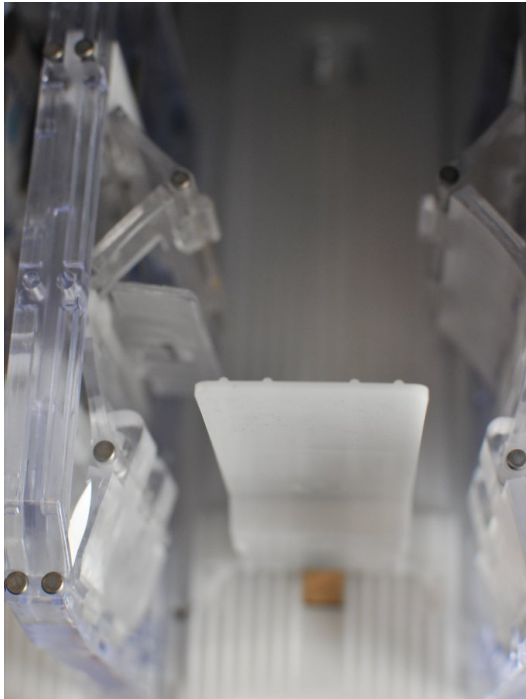


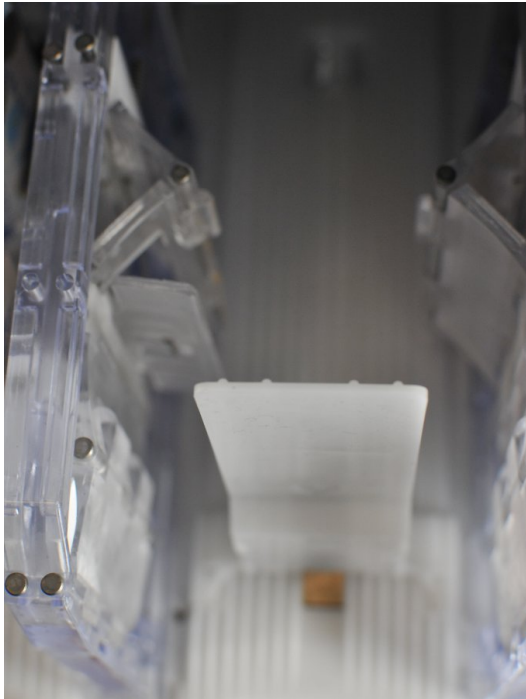




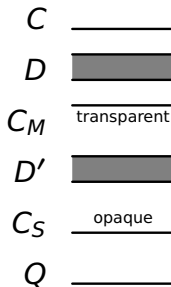
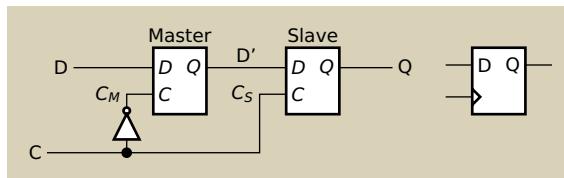






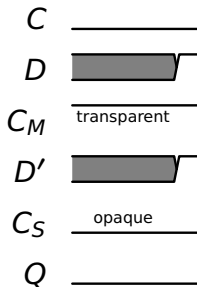
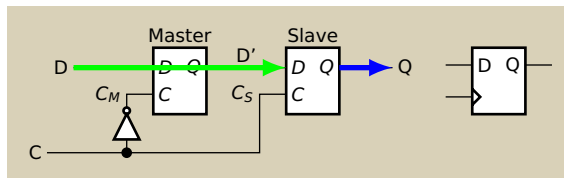


# Positive-Edge-Triggered D Flip-Flop

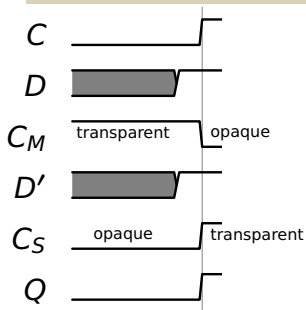
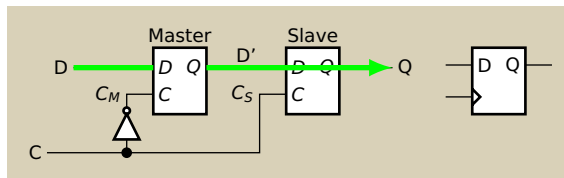




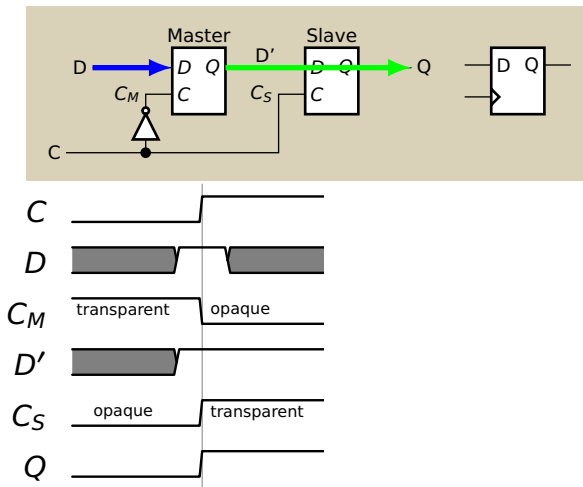
# Positive-Edge-Triggered D Flip-Flop



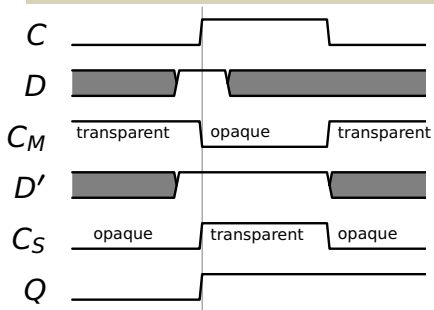
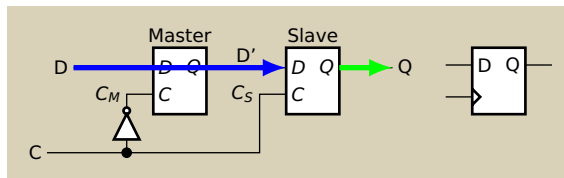
# Positive-Edge-Triggered D Flip-Flop



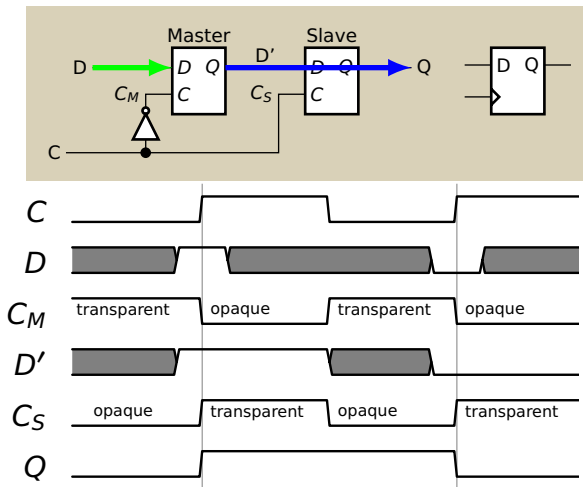
# Positive-Edge-Triggered D Flip-Flop



# Positive-Edge-Triggered D Flip-Flop

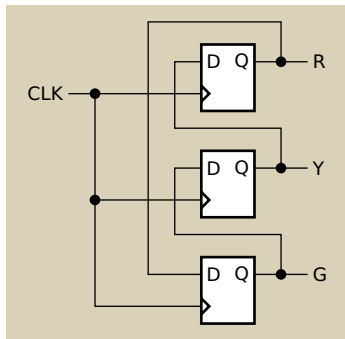


# Positive-Edge-Triggered D Flip-Flop



# The Traffic Light Controller: A second try

Let's try this again with D flip-flops.



CLK \_\_\_

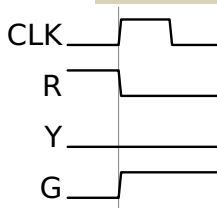
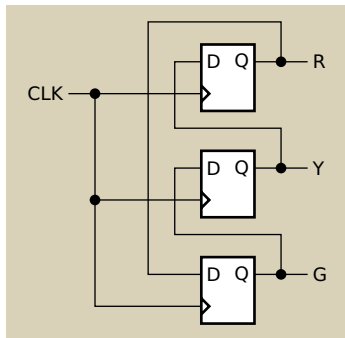
R \_\_\_

Y \_\_\_

G \_\_\_

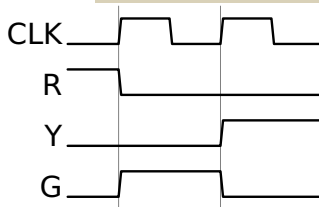
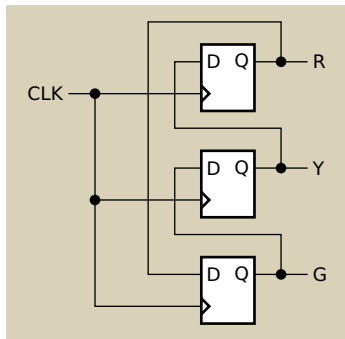
# The Traffic Light Controller: A second try

Let's try this again with D flip-flops.



# The Traffic Light Controller: A second try

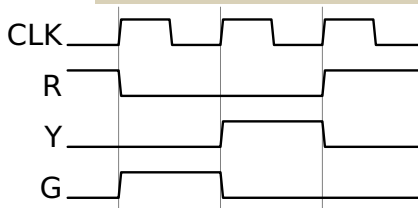
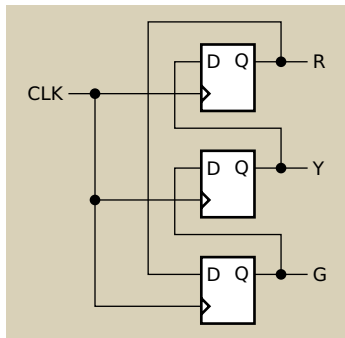
Let's try this again with D flip-flops.





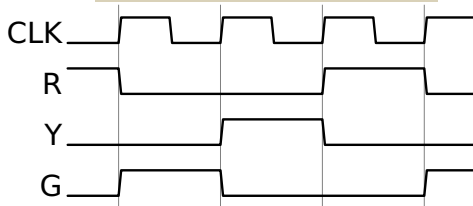
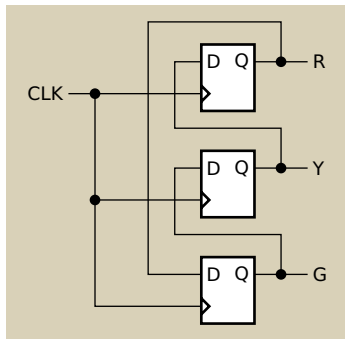
# The Traffic Light Controller: A second try

Let's try this again with D flip-flops.

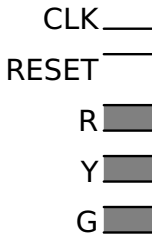
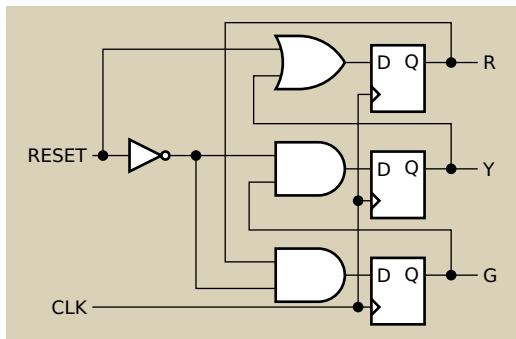


# The Traffic Light Controller: A second try

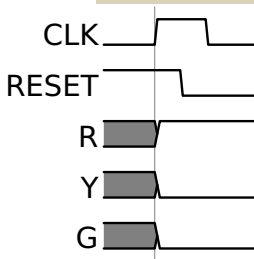
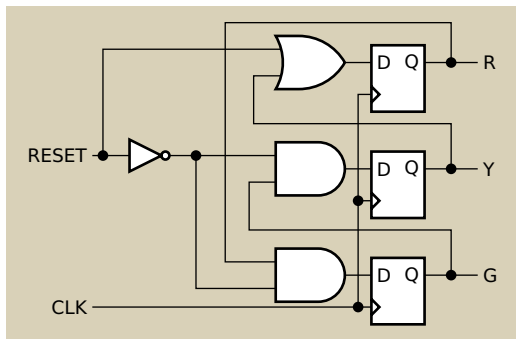
Let's try this again with D flip-flops.



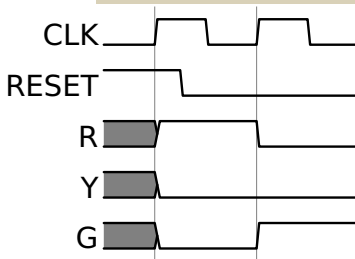
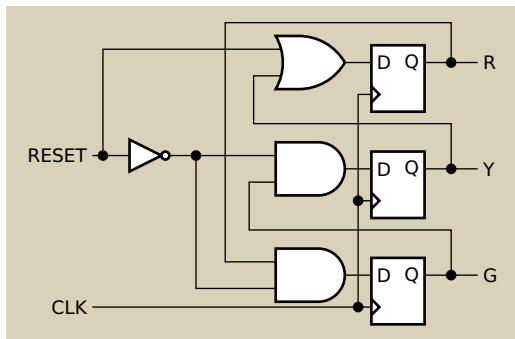
# The Traffic Light Controller with Reset



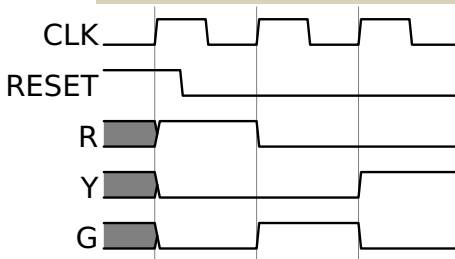
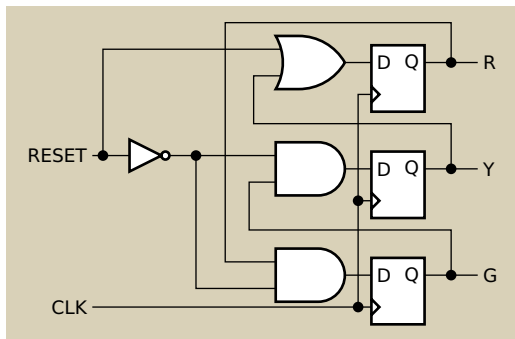
# The Traffic Light Controller with Reset



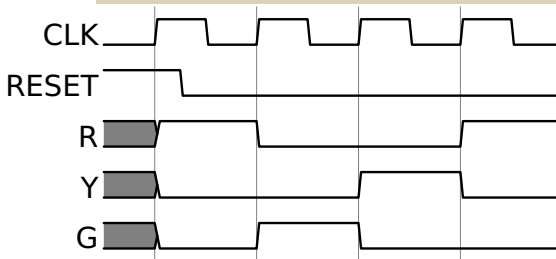
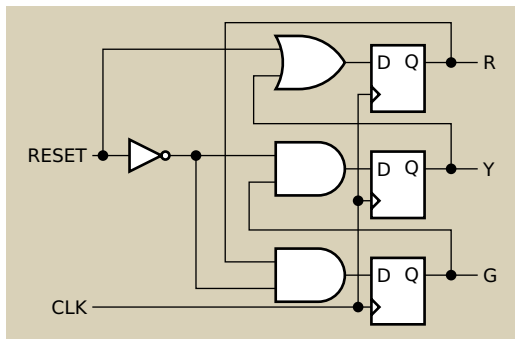
# The Traffic Light Controller with Reset



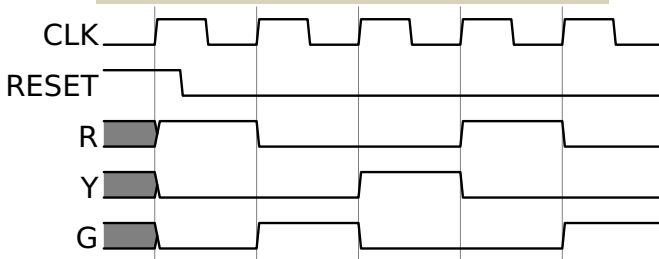
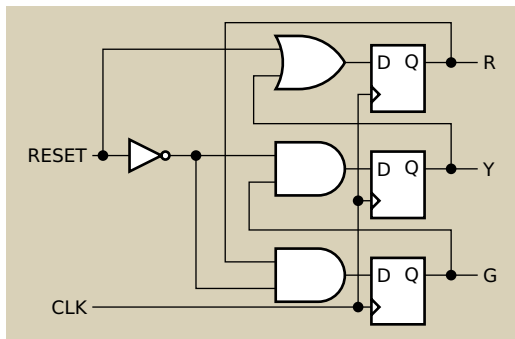
# The Traffic Light Controller with Reset



# The Traffic Light Controller with Reset

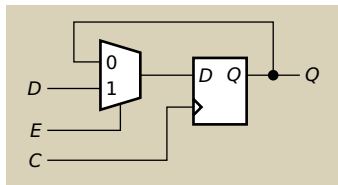


# The Traffic Light Controller with Reset

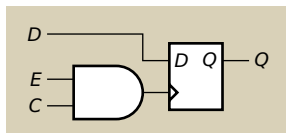




# D Flip-Flop with Enable

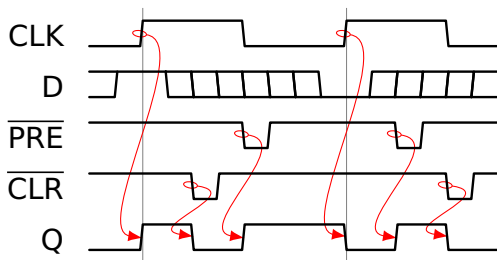
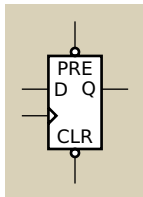


$C$	$E$	$D$	$Q$
$\uparrow$	0	X	$Q$
$\uparrow$	1	0	0
$\uparrow$	1	1	1
0	X	X	$Q$
1	X	X	$Q$

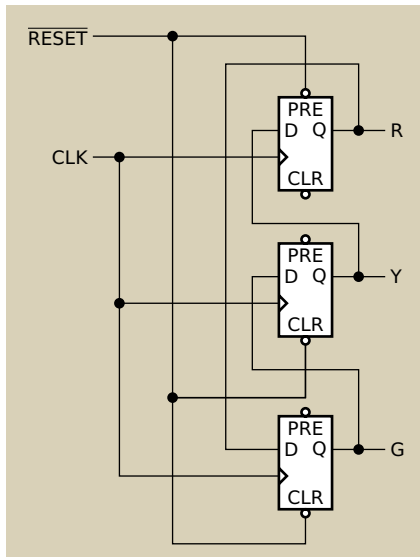


What's wrong with this solution?

# Asynchronous Preset/Clear



# The Traffic Light Controller w/ Async. Reset

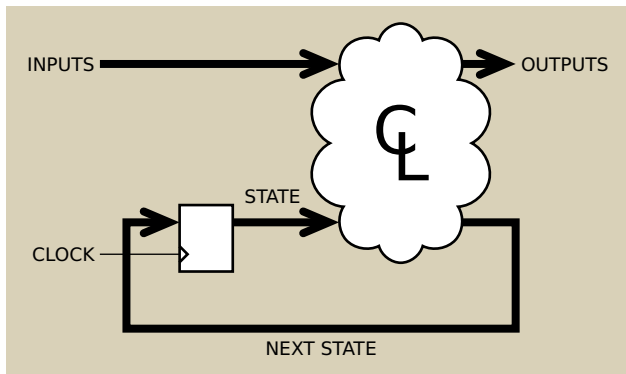


# The Synchronous Digital Logic Paradigm

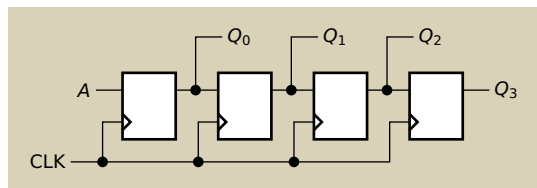
Gates and D  
flip-flops only

Each flip-flop  
driven by the  
same clock

Every cyclic  
path contains  
at least one  
flip-flop

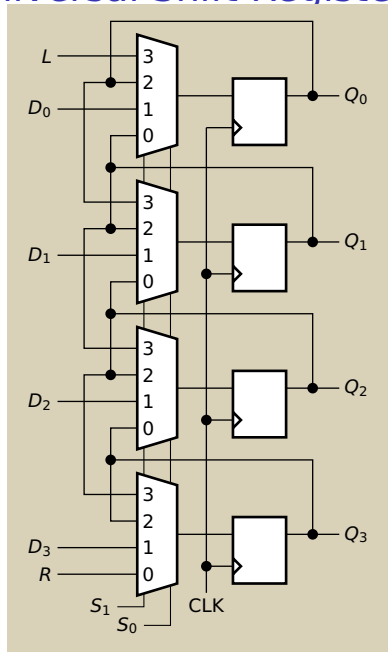


# Cool Sequential Circuits: Shift Registers



A	Q <sub>0</sub>	Q <sub>1</sub>	Q <sub>2</sub>	Q <sub>3</sub>
0	X	X	X	X
1	0	X	X	X
1	1	0	X	X
0	1	1	0	X
1	0	1	1	0
0	1	0	1	1
0	0	1	0	1
0	0	0	1	0
1	0	0	0	1
0	1	0	0	0

# Universal Shift Register

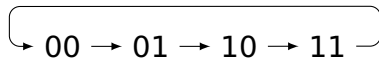


$S_1$	$S_0$	$Q_3$	$Q_2$	$Q_1$	$Q_0$
0	0	$R$	$Q_3$	$Q_2$	$Q_1$
0	1	$D_3$	$D_2$	$D_1$	$D_0$
1	0	$Q_3$	$Q_2$	$Q_1$	$Q_0$
1	1	$Q_2$	$Q_1$	$Q_0$	$L$

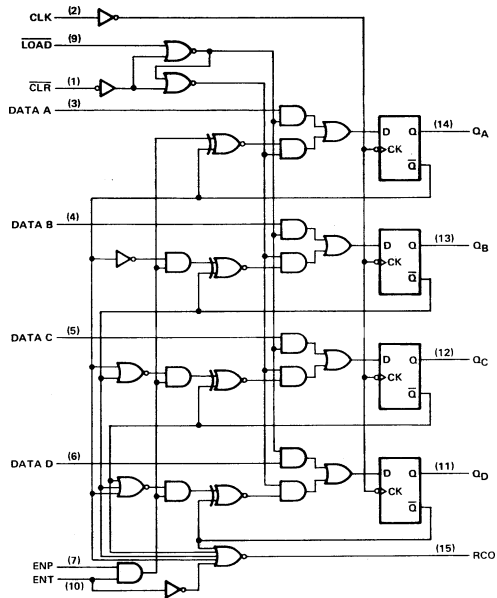
$S_1$	$S_0$	<b>Operation</b>
0	0	Shift right
0	1	Load
1	0	Hold
1	1	Shift left

# Cool Sequential Circuits: Counters

Cycle through sequences of numbers, e.g.,

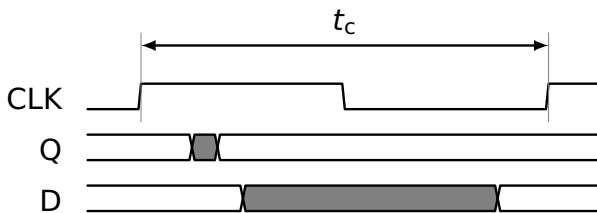
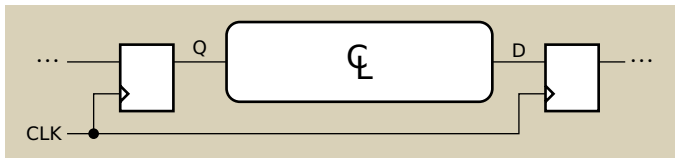


# The 74LS163 Synchronous Binary Counter



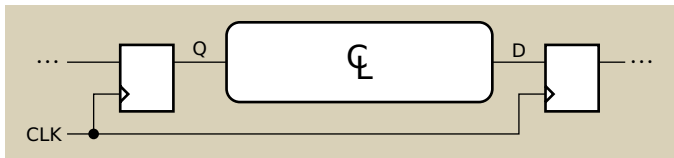


# Timing in Synchronous Circuits

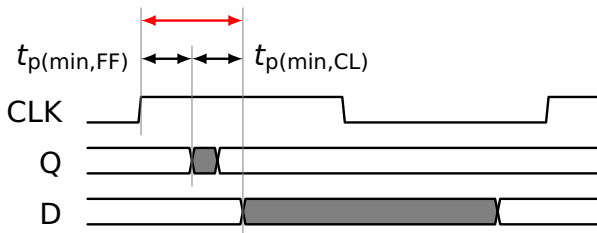


$t_c$ : Clock period. E.g., 10 ns for a 100 MHz clock

# Timing in Synchronous Circuits

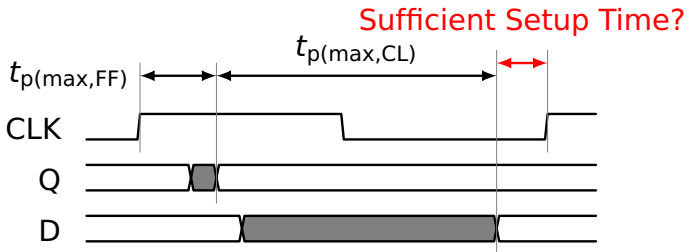
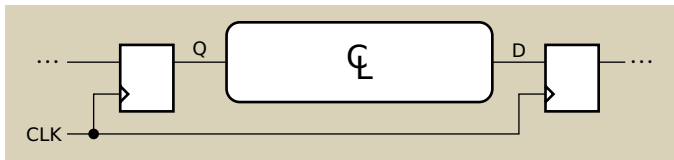


Sufficient Hold Time?



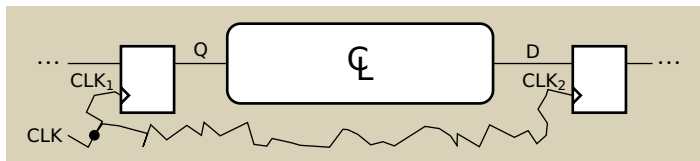
Hold time constraint: how soon after the clock edge can D start changing? Min. FF delay + min. logic delay

# Timing in Synchronous Circuits

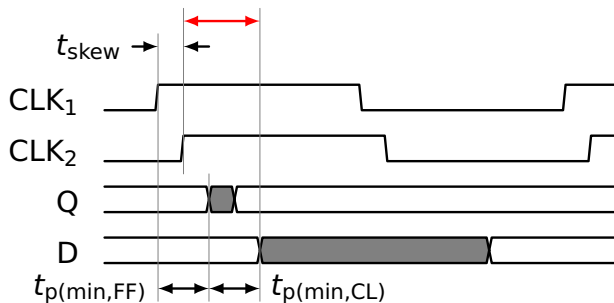


Setup time constraint: when before the clock edge is D guaranteed stable? Max. FF delay + max. logic delay

# Clock Skew: What Really Happens

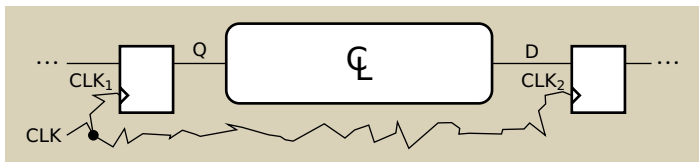


## Sufficient Hold Time?

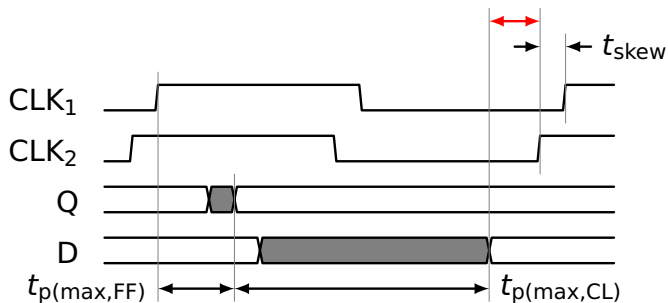


$CLK_2$  arrives late: clock skew reduces hold time

# Clock Skew: What Really Happens



Sufficient Setup Time?



$CLK_1$  arrives early: clock skew reduces setup time