Lecture Outline

Reminders to self:

- ❑ Turn on lecture recording to Cloud
- ❑ Turn on Zoom microphone

- Last Lecture
  - Finished ROM
  - Programmable Logic Arrays
  - Started Latches & Flip-Flops (groundwork for S-R Latch)
- Today's Lecture
  - Latches
  - Gated Latches
  - Flip-Flops

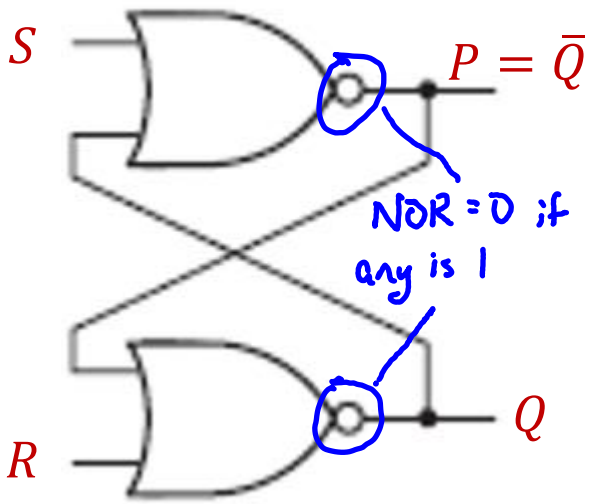
Handouts and Announcements

- Announcements
  - Homework Problem 9-4
    - Posted on Carmen yesterday (2/21)
    - Due: 11:25am Monday 2/27
  - Homework Reminder:
    - HW 9-1 past due
    - HW 9-2 & 9-3 due: 11:59pm Thursday 2/23
  - Read for Friday: pages 352-358, 370, 375-380

Set-Reset Latch

Set-Reset Latch (SR Latch)

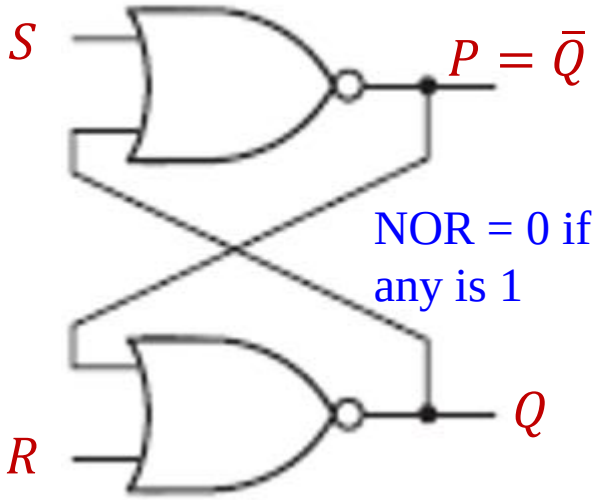
- Can be made from two NOR gates with feedback in cross-coupled form
- Term present state $[Q(t)]$ used to denote state of Q output of the latch or flip-flop at time any input signal changes
- Term next state $[Q(t+\varepsilon)]$ denotes state of Q output after latch or flip-flop has reacted to input change and stabilized
- Normally we will write next state with + superscript, without explicitly showing $t+\varepsilon$



Present State				Next State		
S	R	Q	P	Q^+	P^+	
0	0	0	1	0	1	Hold state
0	0	1	0	1	0	
0	1	0	1	0	1	Reset state
0	1	1	0	0	1	
1	0	0	1	1	0	Set state
1	0	1	0	1	0	

Set-Reset Latch (SR Latch)

$S = 1$
$R = 1$ } BAD!

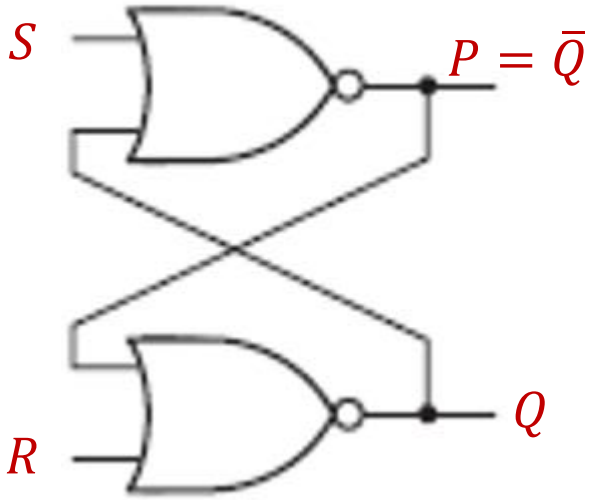


- Makes both $P = Q = 0$, but then $P \neq \bar{Q}$
- If S and R are simultaneously changed to 0, both P and Q may both change to 1
- The 1s propagate through the NOR gates changing P and Q to 0
- Latch may continue to oscillate if the gate delays are equal
- Unstable : ***S = R = 1 must be avoided***

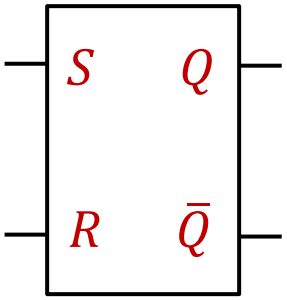
The designer of the circuit using the SR Latch is responsible for avoiding it!

ME!

Set-Reset Latch (SR Latch)



SR Latch Component



- $S = 0, R = 0$ "Hold" $Q^+ = Q$
- $S = 0, R = 1$ "Reset" $Q^+ = 0$
- $S = 1, R = 0$ "Set" $Q^+ = 1$
- $S = 1, R = 1$ Do Not Allow

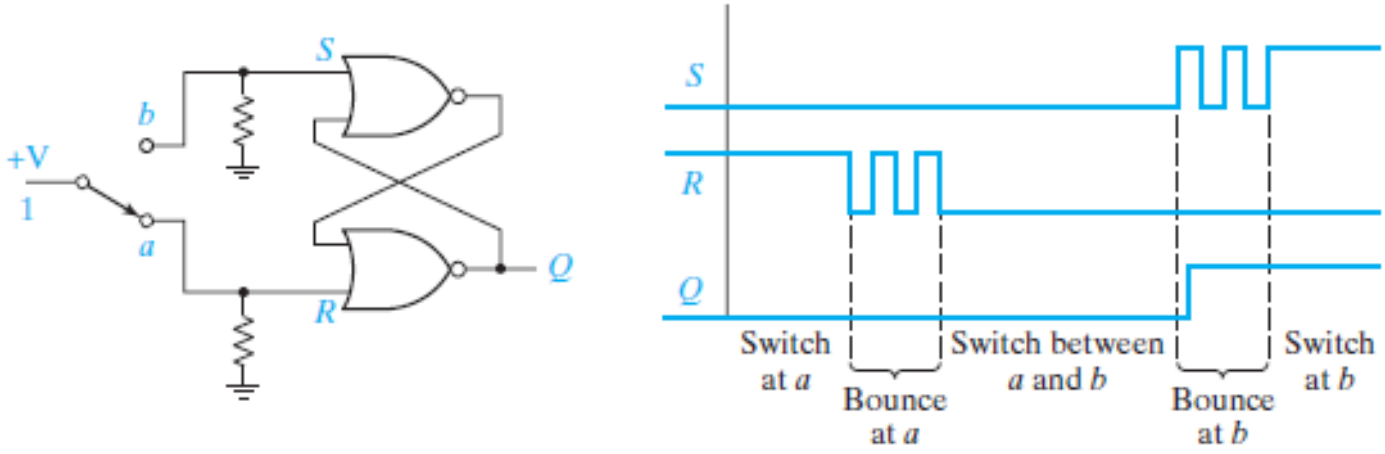
Very Bad

- Note: No clock input in component figure
- We will later use SR flip-flops, which will have a clock input
- That is how you can tell them apart

Set-Reset Latch (SR Latch)

- A useful application of the SR Latch is switch debouncing
- Switch contacts tend to vibrate open and closed several times before settling to their final position
- Produces an electronically noisy transition
- This noise can interfere with proper operation of a logic circuit

FIGURE 11-9
Switch Debouncing with an S-R Latch

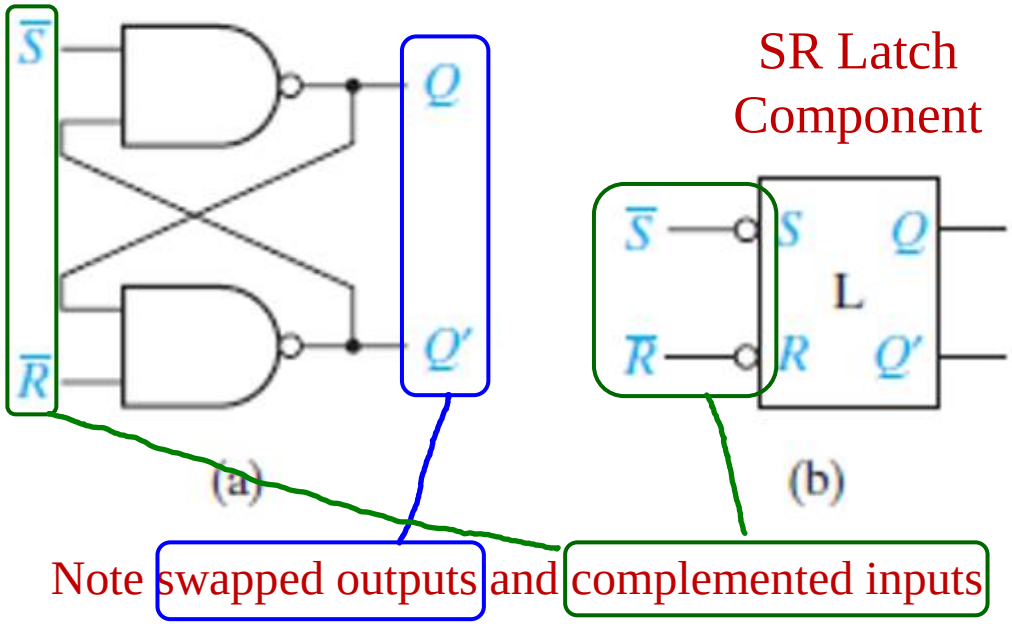


- This debouncing circuit works for double-throw switches (position *a* or position *b*)
- Debouncing a single-throw switch requires a different circuit

$\bar{S}$-$\bar{R}$ Latch

A NAND version is also possible

$\bar{S}-\bar{R}$ Latch Circuit



Present State		Next State	
$\bar{S}$	$\bar{R}$	Q	Q^+
1	1	0	0
1	1	1	1
1	0	0	0
1	0	1	0
0	1	0	1
0	1	1	1
0	0	0	– } Inputs not allowed
0	0	1	– }

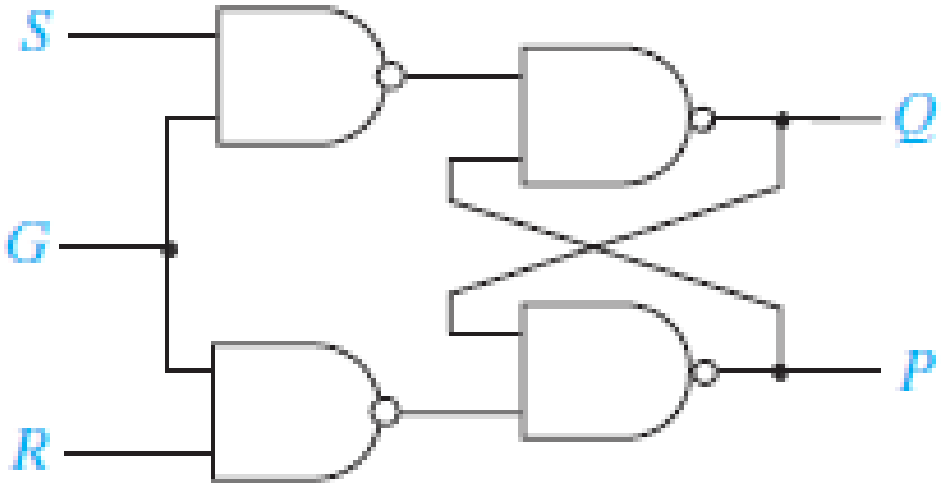
- $\bar{S} = 1, \bar{R} = 1$ Hold
- $\bar{S} = 0, \bar{R} = 1$ Set
- $\bar{S} = 1, \bar{R} = 0$ Reset
- $\bar{S} = 0, \bar{R} = 0$ Unstable: Do Not Allow

Gated Latches

- Gated latches have an additional input called the gate or enable input
- When the gate input is inactive
  - Which may be the high or low value
  - State of the latch cannot change
- When the gate input is active
  - Active-high or active-low
  - Latch is controlled by the other inputs
  - Operates as previously described

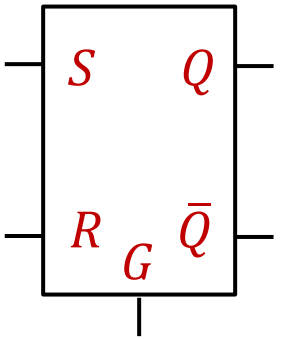
Gated $\bar{S}$-$\bar{R}$ Latch

FIGURE 11-11
NAND-Gate Gated S-R Latch



$G = 0 \implies X = Y = 1$ Hold

$G = 1 \implies \begin{cases} X = \bar{S} \\ Y = \bar{R} \end{cases}$ Regular operation



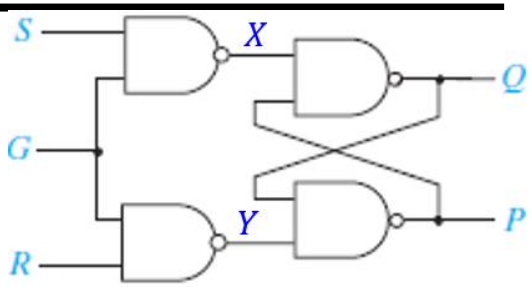
Avoid $X = Y = 0$: $G = 1, S = 1, R = 1$

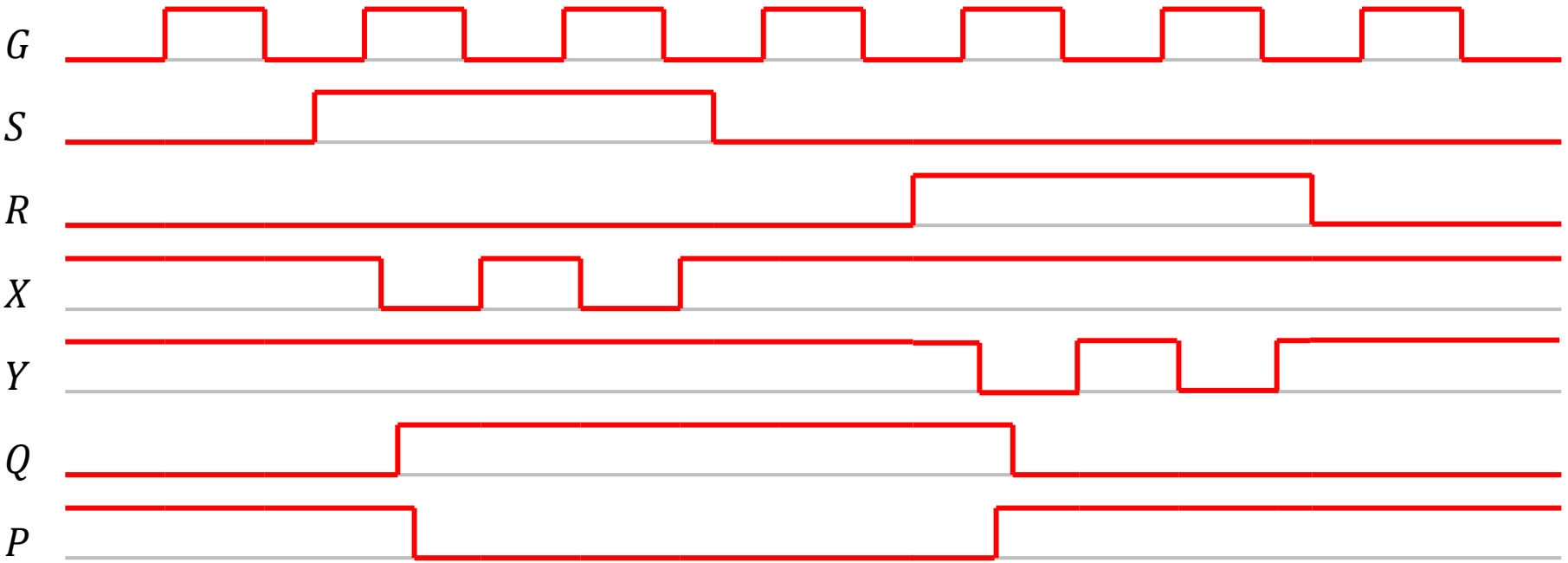
Avoid changing S or R when $G = 1$

- Let S and R stabilize before changing G from 0 to 1
- Else glitches from static or dynamic hazards in circuits that generate S or R can change latch to undesired state if they arrive after $G = 1$

Gated S-R Latch

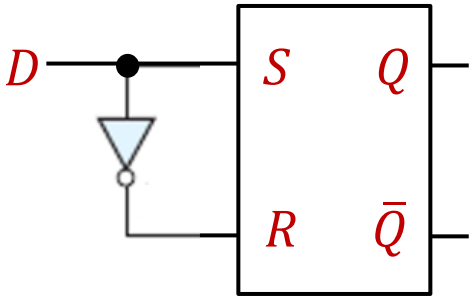
Timing diagram with G serving as a "clock"





D Latch

D-Latch: Data Latch



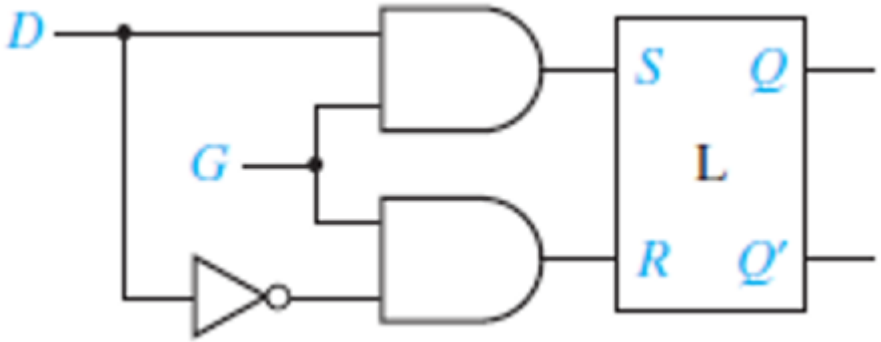
$D = 1 \implies S = 1, R = 0, \quad Q^+ = 1$ set

$D = 0 \implies S = 0, R = 1, \quad Q^+ = 0$ Reset

Inverter results in: No Hold State (00); No Bad State (11)

But without Hold state it does not really have memory ⇨ Make it gated

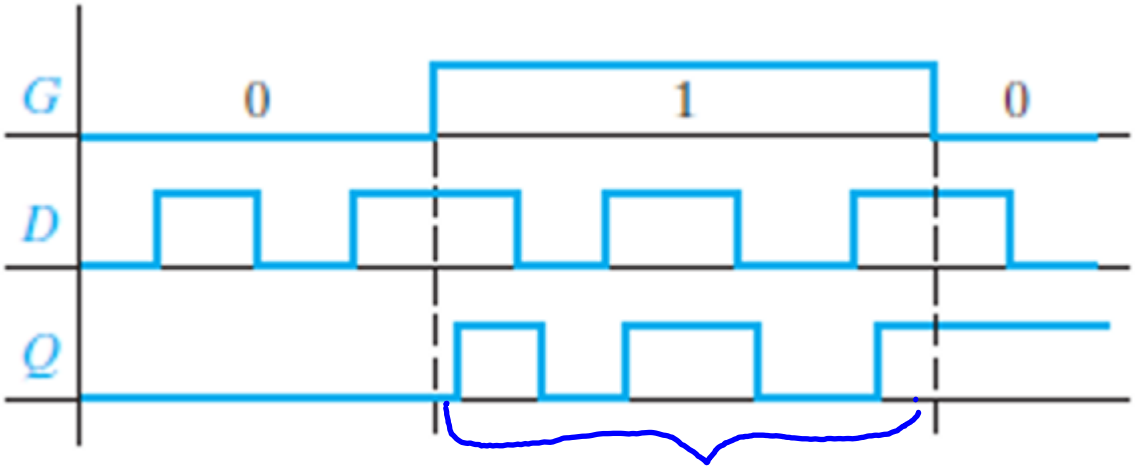
Gated D Latch



$G = 1 \implies S = D, R = \overline{D}$ Set or Reset based on value of D

$G = 0 \implies \begin{cases} S = 0 \\ R = 0 \end{cases}$ Hold

Change due to $G = 1$ while D is already stable



Changes due to D changing while $G = 1$

A.k.a. "transparent latch" since $Q = D$ when G is active

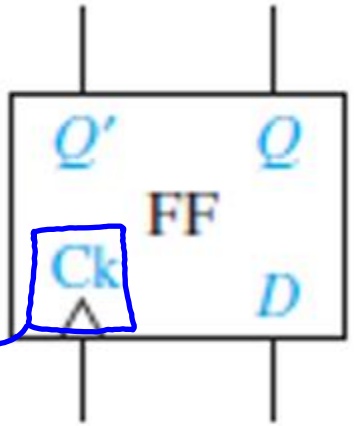
Gated Latches vs Edge-Triggered Flip-Flops

- With Gated Latches (Gated Memory)
  - Wait until input (S and R, or D) is stable before setting $G = 1$ (or 0 if active-low) to write to Q
    - This is " Level-Sensitive " memory
    - Q updated based on G level of 0 or 1
    - Must avoid changing input(s) while G active
  - Using G as a clock for synchronizing writing to memory might be doable for one or a few latches
  - But ensuring all that timing for stable inputs is problematic for large numbers of latches
- Edge-Triggered Flip-Flops
  - Better if we could set up latches that would update at a specific moment in time
  - Use "Edge-triggered" Flip-Flops, designed to change states based on either
    - Rising edge of clock only, or
    - Falling edge of clock only.
    - Rather than whenever clock is at 1 (or at 0)

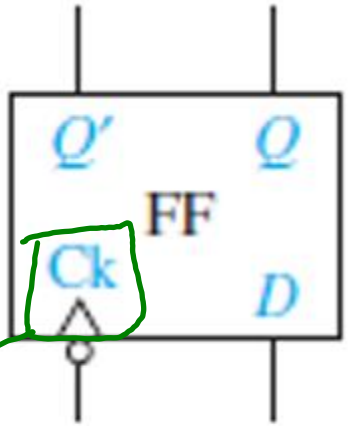
Edge-Triggered and Leader-Follower Flip-Flops

- Edge-Triggered
  - If inputs to flip-flop only need to be stable for a short period of time around the clock edge,
  - Then we refer to the flip-flop as edge-triggered
- Leader-Follower Edge-Triggered
  - Refers to a particular implementation
  - Uses two gated latches in such a way that the flip-flop outputs only change on a clock edge
- Alert!
  - Not all leader-follower flip-flops are edge-triggered
  - Leader-follower can be used to make an edge-triggered flip-flop
  - But is not sufficient to guarantee it is edge-triggered (details matter)

Edge-Triggered D Flip-Flop



(a) Rising-edge trigger



(b) Falling-edge trigger

D	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1

(c) Truth table

Arrowhead identifies clock input (even if not labeled "Ck")

Output can change in response to a 0→1 transition on the clock (positive sloped edge)

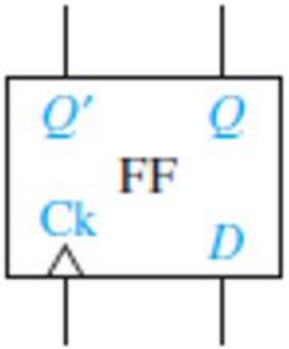
Bubble on clock input flips action of clock

Output can change in response to a 1→0 transition on the clock (negative sloped edge)

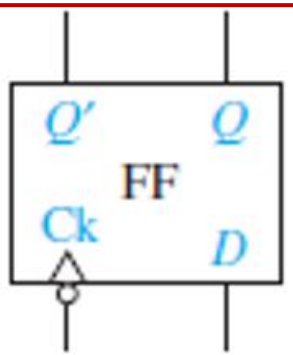
Since flip-flop is edge-triggered:

- Refers to output state **after** active edge of clock hits
- Output **does not** change on change of *D* by itself

Edge-Triggered D Flip-Flop



(a) Rising-edge trigger

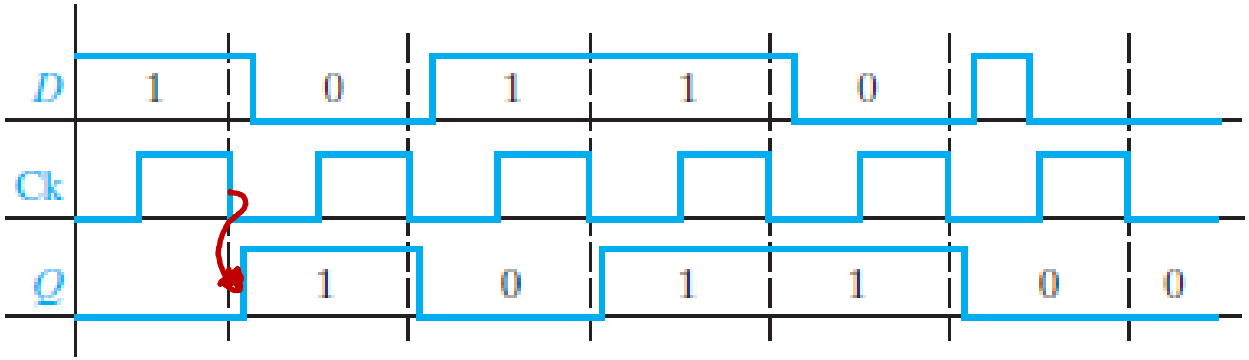


(b) Falling-edge trigger

D	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1

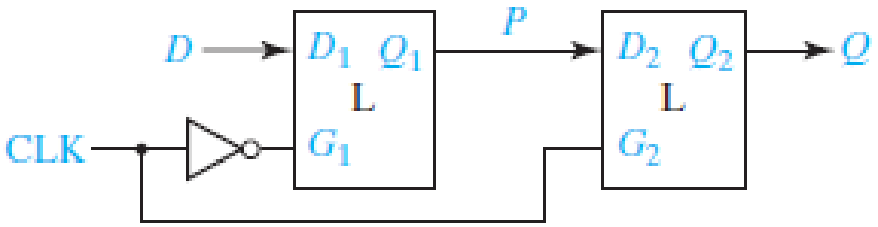
(c) Truth table

FIGURE 11-18
Timing for D Flip-Flop (Falling-Edge Trigger)

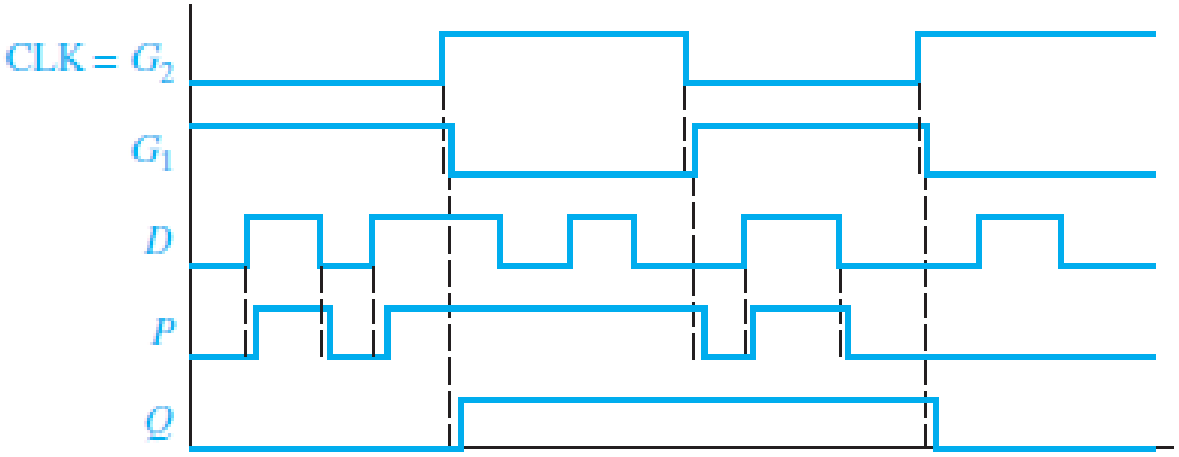


Edge-Triggered D Flip-Flop

FIGURE 11-19
D Flip-Flop (Rising-Edge Trigger)



(a) Construction from two gated D latches



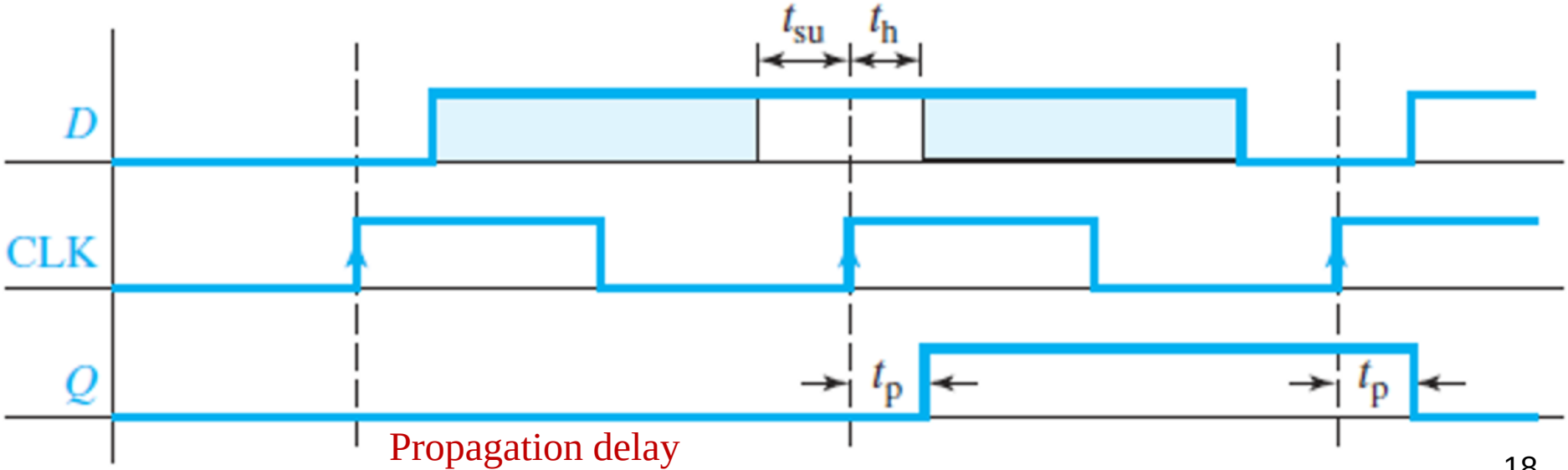
(b) Timing analysis

The two *D* latches will not be transparent at the same time

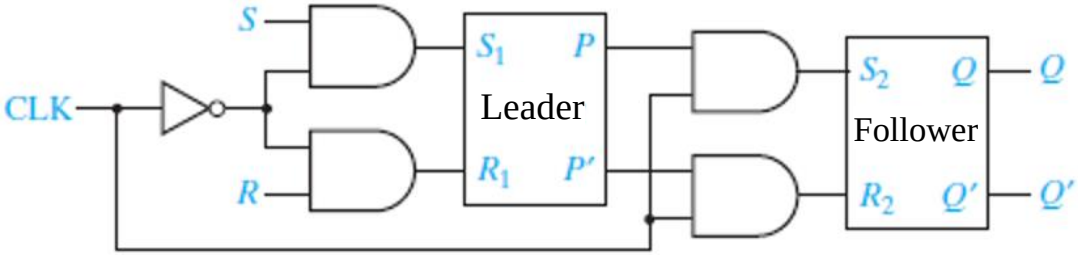
- There is a small interval due to inverter gate delay when it looks like both might be transparent
- But there is also a delay through the latches during that interval
- Careful attention to internal timing needed

Edge-Triggered D Flip-Flop

- To function properly, D input must be stable for a period of time before and after the active edge of clock
  - Set up time $\equiv t_{su}$ is time D must be stable before active edge of clock
    - Allows D to propagate through first gated latch
  - Hold time $\equiv t_h$ is time D must be stable after active edge of clock
    - Needed so that current D is stored in first latch, before any change
- Shown here for rising-edge triggered case
- Usually found on data sheets



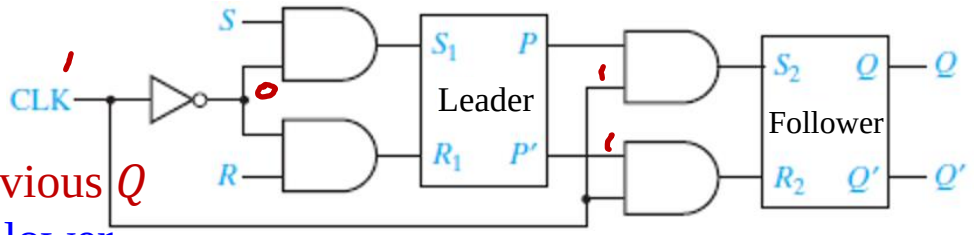
S-R Flip-Flop



Operation Details

- Leader-Follower configuration
- $CLK = 0$ — S and R set or reset outputs P and P' of Leader gated latch
- CLK changes to 1 — Leader gated latch holds values of P and P' and
- $CLK = 1$ — Leader gated latch continues to hold values of P and P'
- CLK changes to 0 — Values of Q and Q' latched into the Follower gated latch

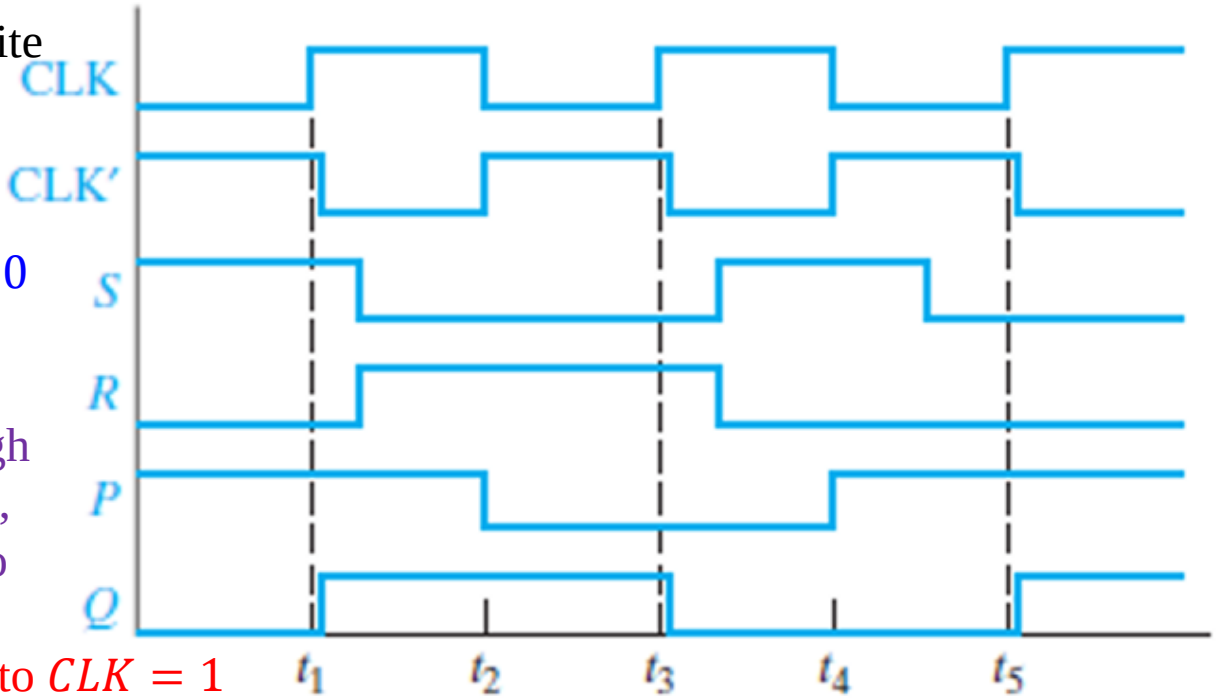
S-R Flip-Flop



Operation Details

- $CLK = 0$ S and $R \rightarrow P$, Follower holds previous Q
- $CLK \rightarrow 1$ Leader holds P, transfers it to Follower
- $CLK = 1$ Leader continues to hold P, Q does not change even if S or R change
- $CLK \rightarrow 0$ Q latched into Follower, Leader processes new inputs

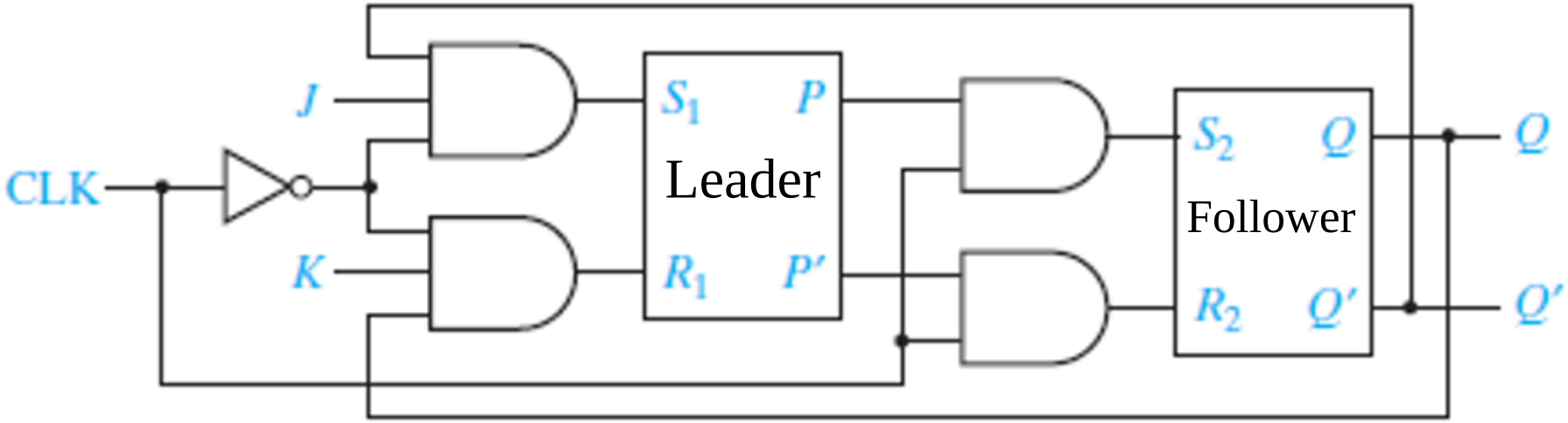
- S-R leader-follower not quite identical to edge-triggered
- Look at t_4
- P changes as leader processes new $S = 1, R = 0$
- R and S both at 0 before rising clock edge at t_5
- Since P already latched high in leader, Q goes high at t_5, even though should be zero for $R = S = 0$
- Fix: Restrict R, S changes to $CLK = 1$



J-K Flip-Flop

- Uses present values of Q and Q' to make $[S = 1, R = 1]$ into a "Toggle" condition
- One way to implement with rising-edge CLK:

Gate for follower enabled by CLK



Gate for leader, enabled by $\overline{CLK}$, with of Q and Q' also fed back in cross-connected manner

Note: While $J = K = 1$ is OK, feeding present Q and Q' through ANDs ensures S_1 and R_1 not $= 1$ at same time

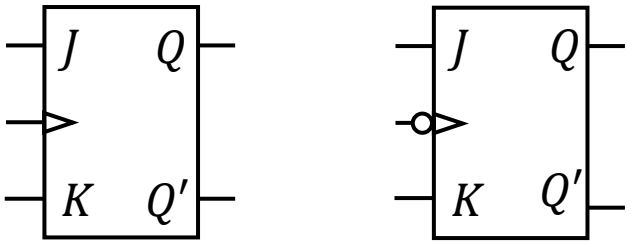
- $CLK = 0, J = 1, K = 1$ $S_1 = Q'$ and $R_1 = Q$

  If $Q = 1 \rightarrow S_1 = 0, R_1 = 1$ ⇨

  If $Q = 0 \rightarrow S_1 = 1, R_1 = 0$ ⇨

- Toggle mode:

J-K Flip-Flop



J-K Flip-Flop ⇨ General purpose do everything flip-flop

- $J = K = 0$ Hold
- $J = 0, K = 1$ Reset
- $J = 1, K = 0$ Set
- $J = K = 1$ Toggle

Present state

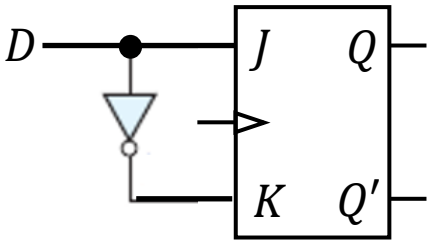
After active-edge of clock

J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

J-K Flip-Flop

J-K Flip-Flop ⇨ General purpose, do everything flip-flop

- For example, D Flip-Flop from J-K

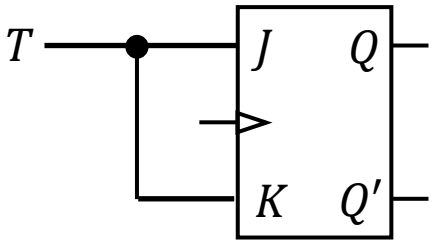


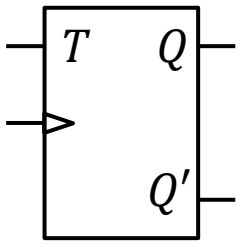
D	Q	Q^+
0	X	0
1	X	1

T Flip-Flop

J-K Flip-Flop ⇨ General purpose, do everything flip-flop

- For example, T Flip-Flop from J-K





T	Q	Q^+
0	X	Q
1	X	$\bar{Q}$