Sequential Circuits

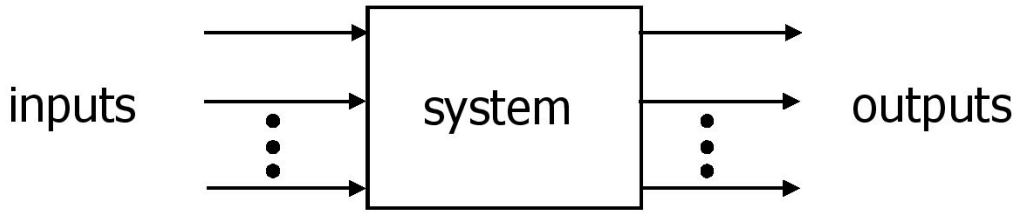
يادآوري

- آموزش تكنيک هاي طراحي و پياده سازي سيستم هاي پيچيده:

- **سيستم:**
  - داراي ورودي ها، خروجي ها و رفتار مشخصي است
    - اين رفتار توسط تابع هايي تعيين مي شود كه ورودي ها را به خروجي ها تبديل (نگاشت) مي كند.
    - مثال: گوشي تلفن:
      - ورودي ها: كليدها
      - خروجي ها: صفحة نمايش و سيگنال هاي ارسالي به مركز تلفن
      - رفتار: شماره گيري و ايجاد ارتباط
    - مثال: خودرو:
      - ورودي ها: پدال ها، سوييچ، فرمان، ...
      - خروجي ها: فرمان پيچش و چرخش چرخ ها، فرمان ترمز، ...
      - رفتار: ....
    - مثال: تلويزيون

Sequential vs. Combinational

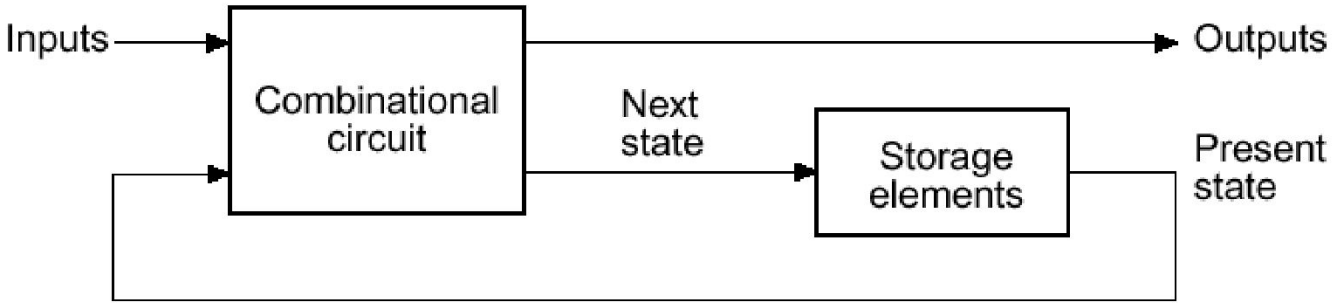
- **Combinational Logic:**
  - Output depends only on current input
    - TV channel selector (0-9)



- **Sequential Logic:**
  - Output depends not only on current input but also on current **state** of the system (which depends on past input values)
    - TV channel selector (up-down)
  - Need some type of memory to remember the current state

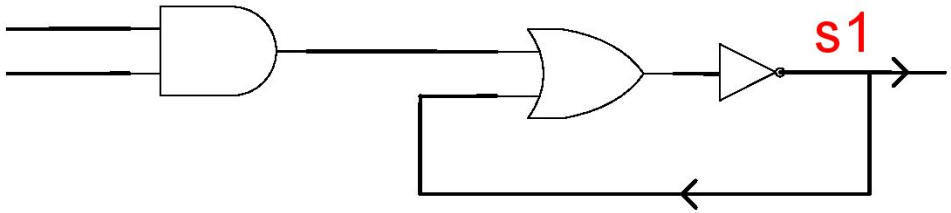
Sequential Logic

- **Sequential logic circuits**
  - Remembers past inputs and past circuit state
  - Outputs from the system can be “fed back” as new inputs
  - The storage elements are circuits that are capable of storing binary information: memory



Feedback Loop

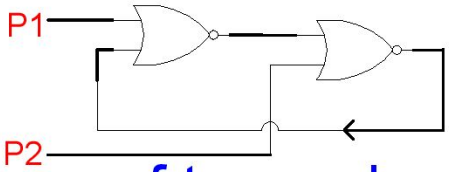
- **Feedback:**
  - A signal s1 depends on another signal whose value depends on s1
    - There can be several intermediate signals



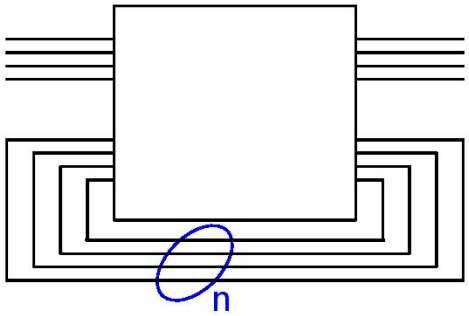
Base of Memory

- Consider the following circuit:

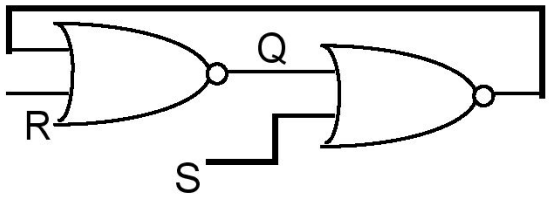
P1 = P2 = 0

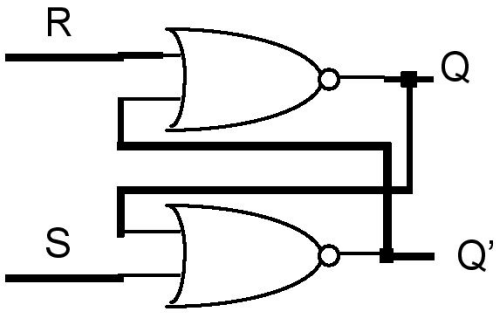


- The feedback line can keep one of two values, 0 or 1.
- Previous circuits (combinational) had no feedback
- A circuit with n feedback lines has 2^n potential states, and that the memory of our circuit depends on the number of its feedback lines:



SR Latch (NOR version)





S	R	Q
0	0	hold
0	1	0
1	0	1
1	1	unstable

Truth Table:
Next State = F(S, R, Current State)

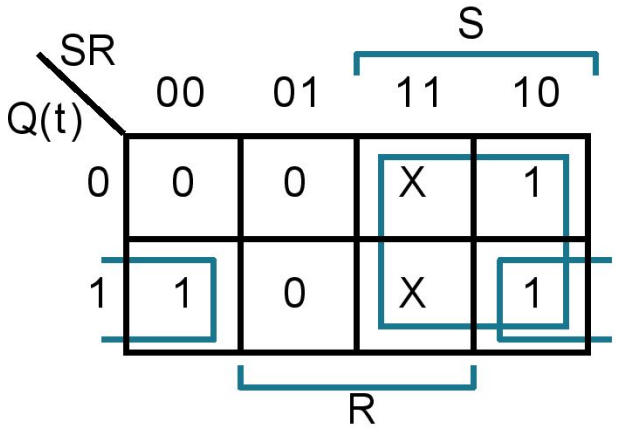
S(t)	R(t)	Q(t)	Q(t+)
0	0	0	0 (hold)
0	0	1	1 (hold)
0	1	0	0 (reset)
0	1	1	0 (reset)
1	0	0	1 (set)
1	0	1	1 (set)
1	1	--	Not allowed

SR Latch

Truth Table:
Next State = F(S, R, Current State)

S(t)	R(t)	Q(t)	Q(t+)
0	0	0	0 (hold)
0	0	1	1 (hold)
0	1	0	0 (reset)
0	1	1	0 (reset)
1	0	0	1 (set)
1	0	1	1 (set)
1	1	0	Not allowed
1	1	1	Not allowed

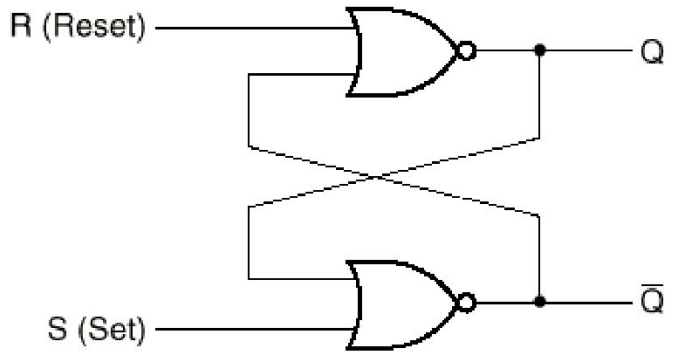
Derived K-Map:



Characteristic Equation:

$$Q+ = S + \overline{R}\, Q_t$$

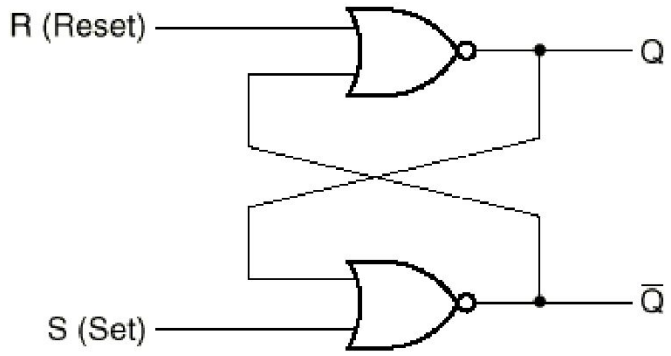
R=S=1 ??



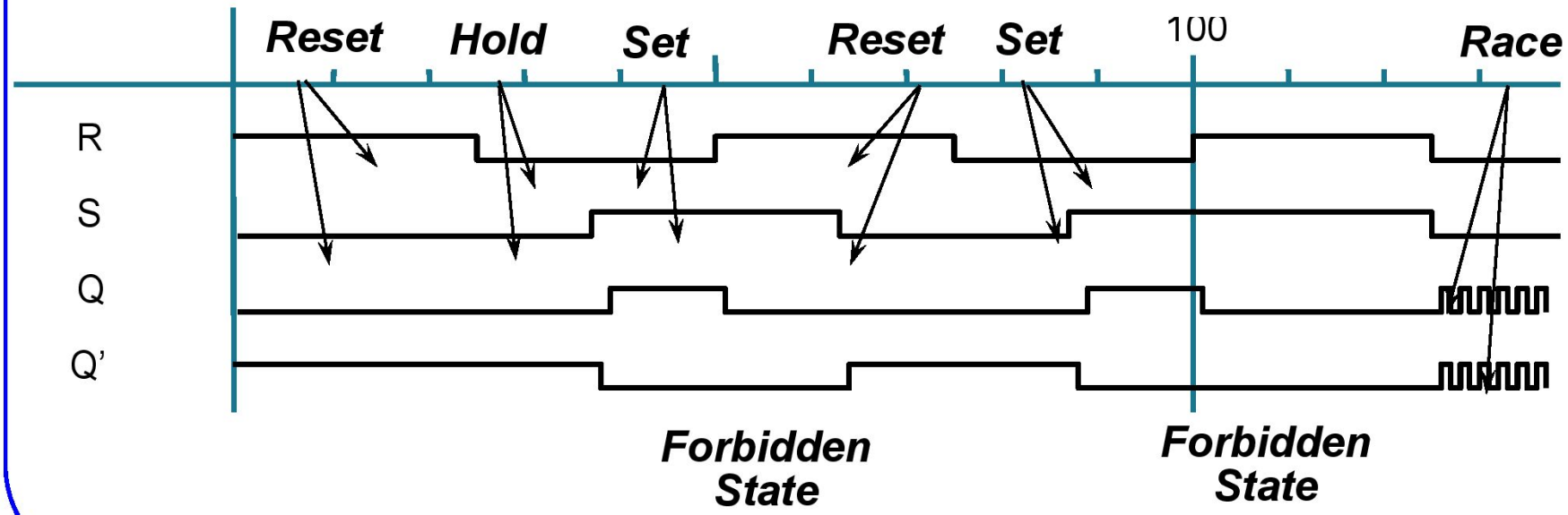
(a) Logic diagram

- **Not allowed, because**
  - When S=R=1, both outputs go to zero
  - If both inputs now go to 0, the state of the SR latch depends on delays
  - Hence, "undefined" state
    - MUST be avoided

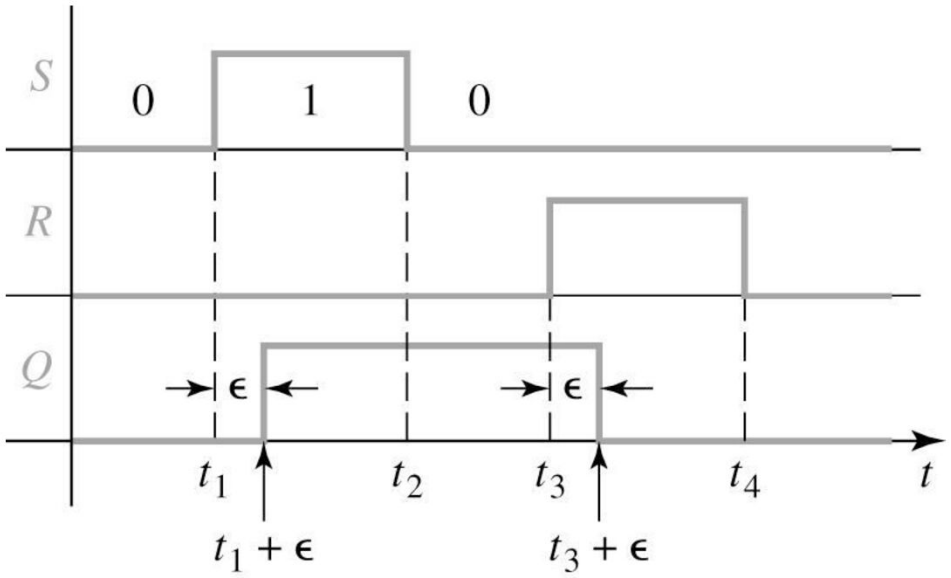
Timing Diagram



(a) Logic diagram



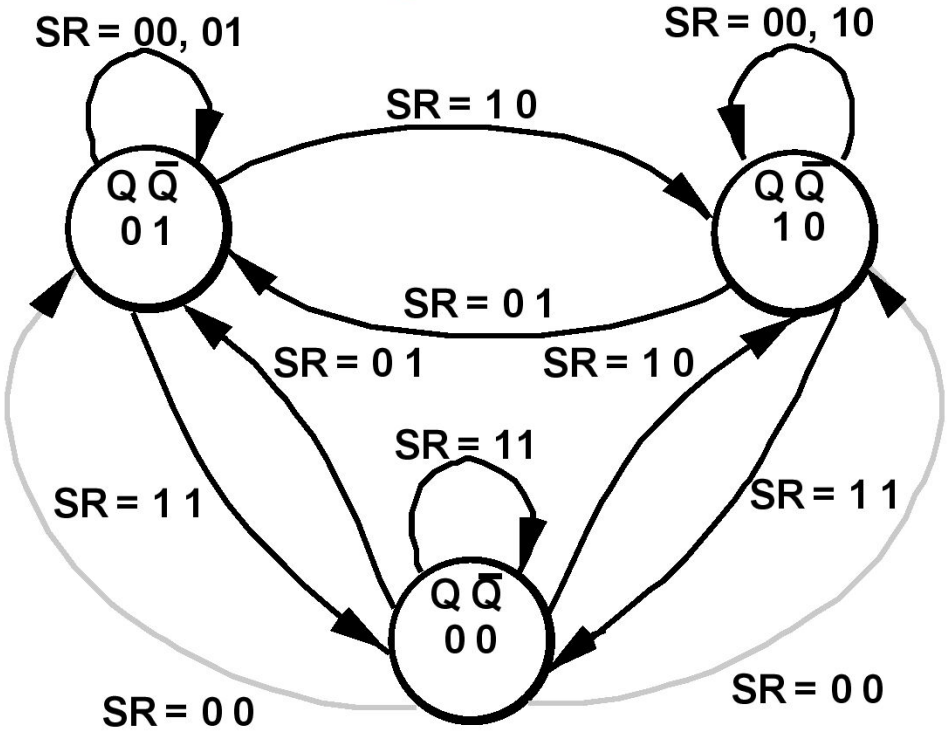
Timing Diagram of SR Latch



- **Note: the delay from S to Q in this timing diagram should be 2ε because from S to Q there are two gate delays**

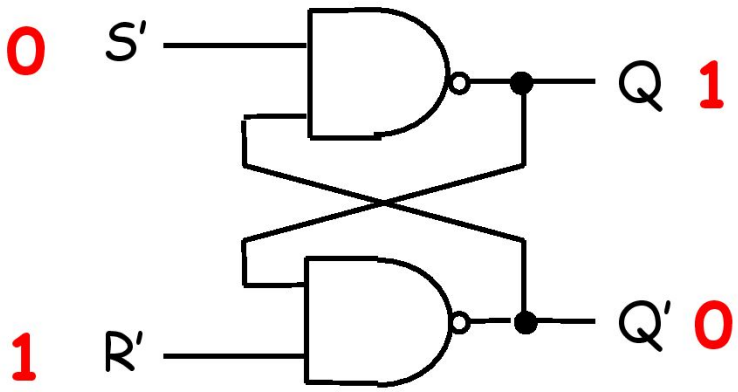
SR Latch State Diagram

- **Observed State Diagram**



- Very difficult to observe SR latch after the 0-0 state
- Ambiguously returns to state 0-1 or 1-0

S'R' Latch (NAND version)

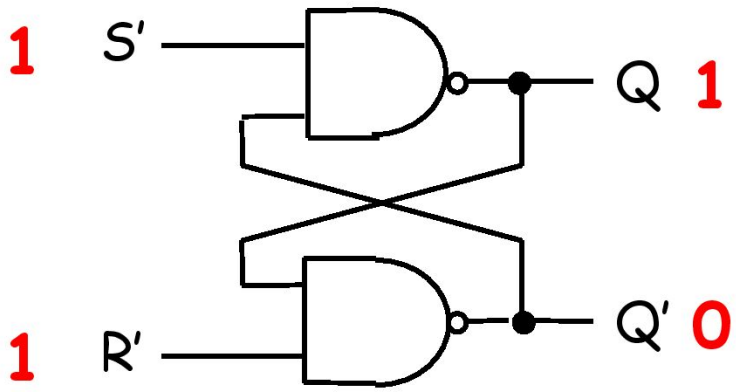


S'	R'	Q	Q'
0	0		
0	1	1	0
1	0		
1	1		

Set

X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

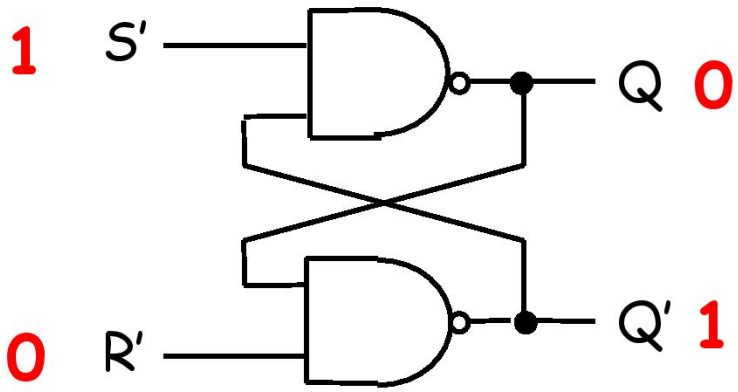
S'R' Latch (NAND version)



S'	R'	Q	Q'	
0	0			
0	1	1	0	Set
1	0			
1	1	1	0	Hold

X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

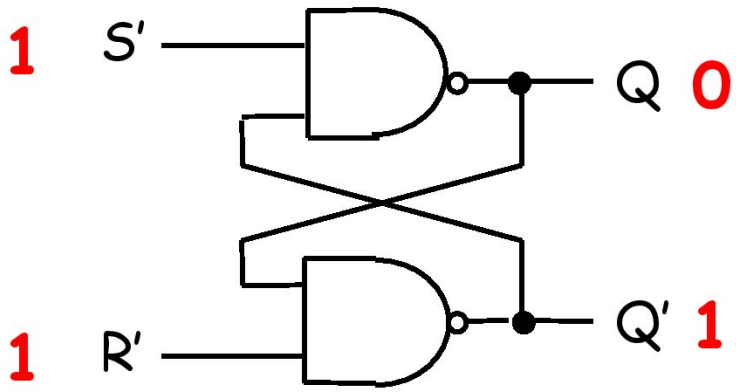
S'R' Latch (NAND version)



S'	R'	Q	Q'	
0	0			
0	1	1	0	Set
1	0	0	1	Reset
1	1	1	0	Hold

X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

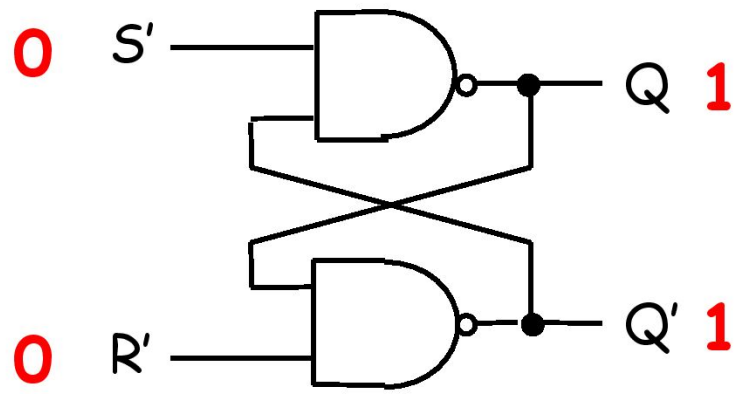
S'R' Latch (NAND version)



S'	R'	Q	Q'	
0	0			
0	1	1	0	Set
1	0	0	1	Reset
1	1	1	0	Hold
		0	1	Hold

X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

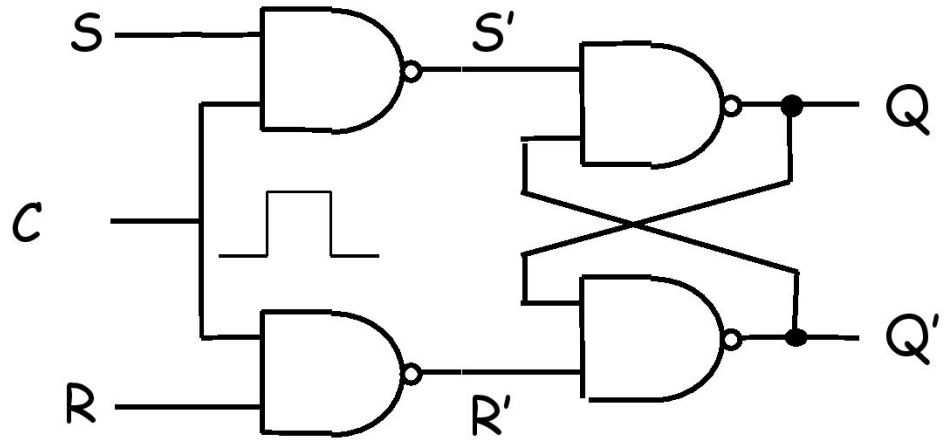
S'R' Latch (NAND version)



S'	R'	Q	Q'	
0	0	1	1	Not allowed
0	1	1	0	Set
1	0	0	1	Reset
1	1	1	0	Hold
		0	1	Hold

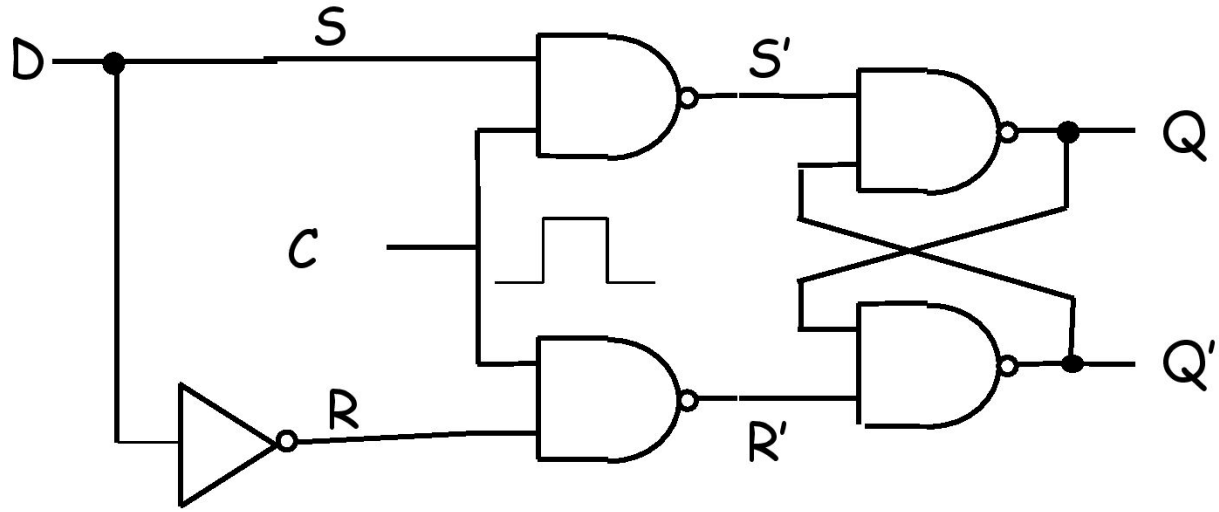
X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

SR Latch with Control (Enable)



S	R	C	S'	R'	Q	Q'	
0	0	1	1	1	Q_0	Q_0'	Hold
0	1	1	1	0	0	1	Reset
1	0	1	0	1	1	0	Set
1	1	1	0	0	1	1	Not allowed
X	X	0	1	1	Q_0	Q_0'	Hold

D Latch (cont.)

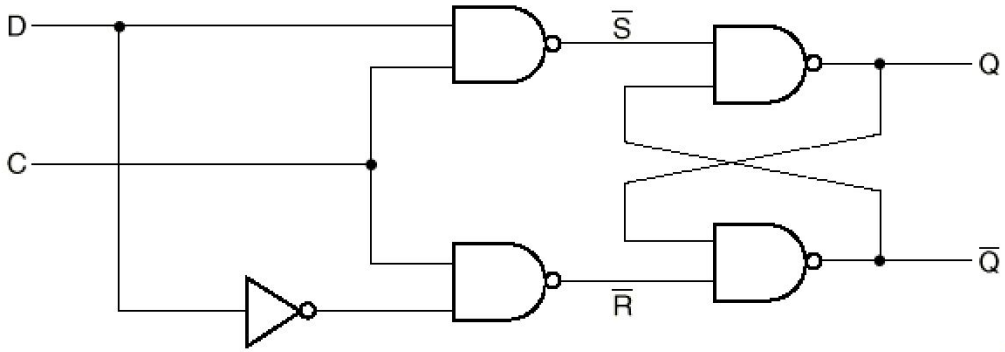


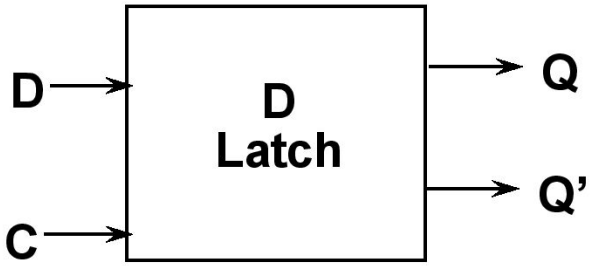
D	C	Q	Q'
0	1	0	1
1	1	1	0
X	0	Q_0	Q_0'

S	R	C	Q	Q'	
~~0~~	~~0~~	~~1~~	Q_0	Q_0'	~~Store~~
0	1	1	0	1	Reset
1	0	1	1	0	Set
~~1~~	~~1~~	~~1~~	1	1	~~Not allowed~~
X	X	0	Q_0	Q_0'	Store

D Latch

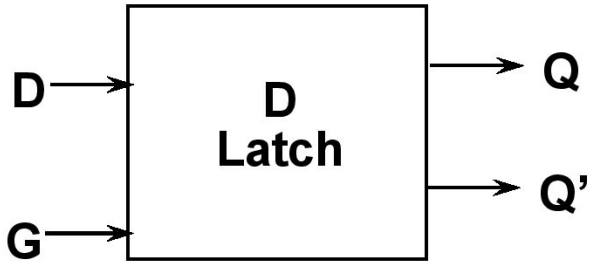
- SR latches are useful in control applications,
  - where we often think in terms of setting a flag in response to some condition, and resetting it when conditions change
- We often need latches simply to store bits presented on a signal line
- → D latch (i.e., Data latch)
- Can eliminate the undesirable indeterminate state in the SR latch:
  - ensures that inputs S and R are never simultaneously 1

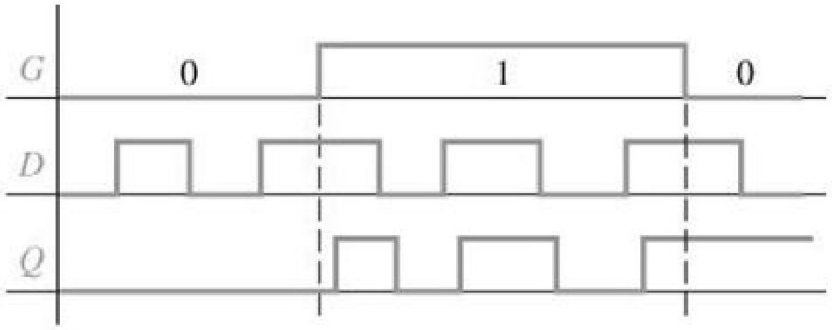




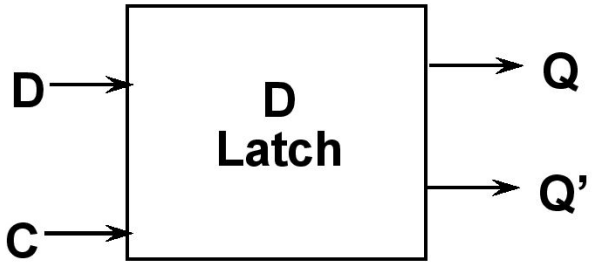
C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

D Latch Timing Diagram

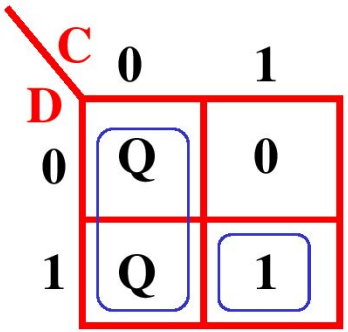


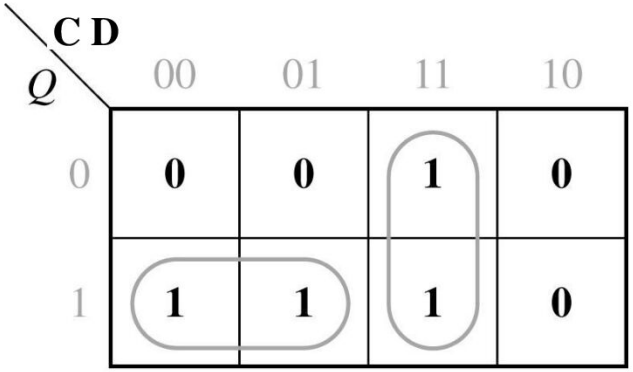


D-Latch Characteristic Equation



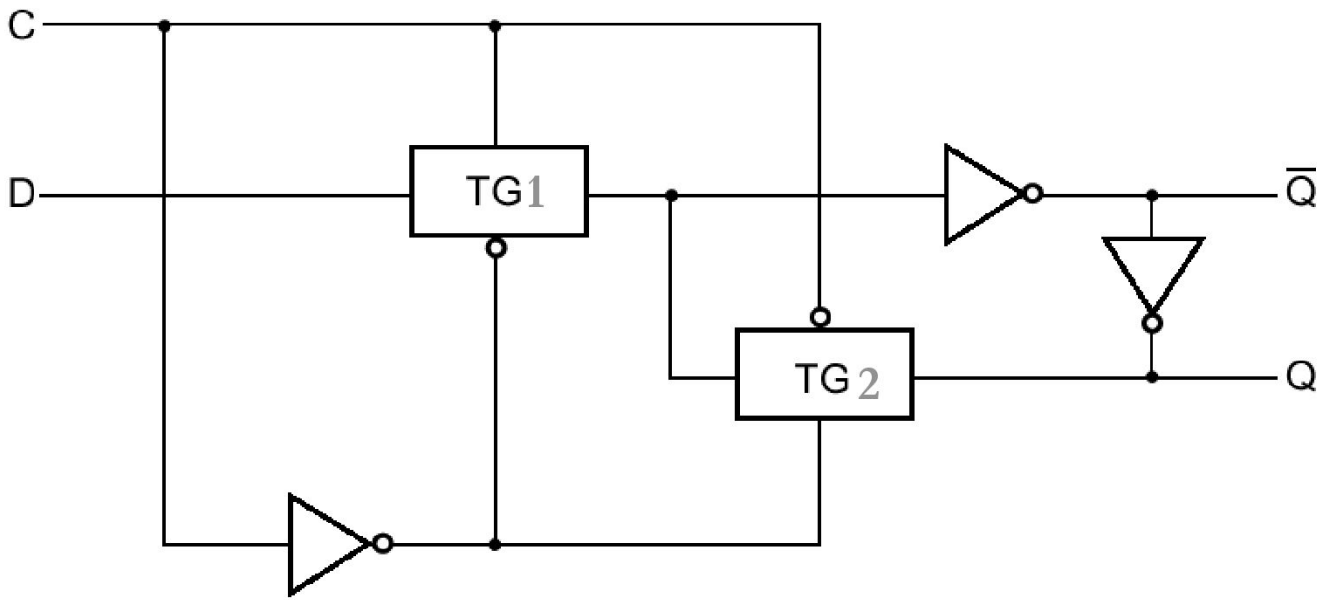
C	D	Q	Q+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1





$$Q+ = C'.Q + C.D$$

D Latch with Transmission Gates

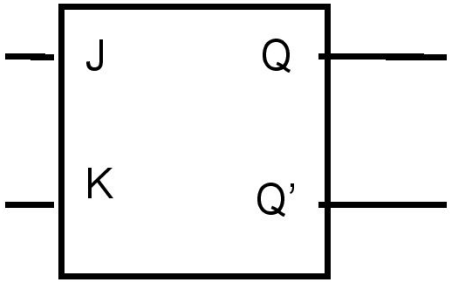


- C=1 → TG1 closes and TG2 opens → Q'=D' and Q=D
- C=0 → TG1 opens and TG2 closes → Hold Q and Q'

JK Latch

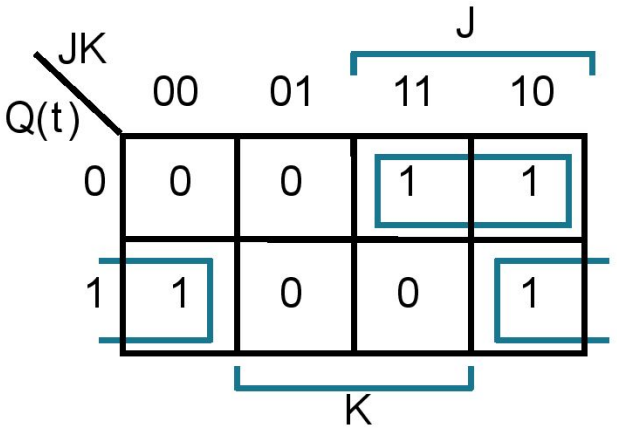
J, K both 1 => toggle

J(t)	K(t)	Q(t)	Q(t+)
0	0	0	0 (hold)
0	0	1	1 (hold)
0	1	0	0 (reset)
0	1	1	0 (reset)
1	0	0	1 (set)
1	0	1	1 (set)
1	1	0	1 (toggle)
1	1	1	0 (toggle)



JK Latch

Derived K-Map:



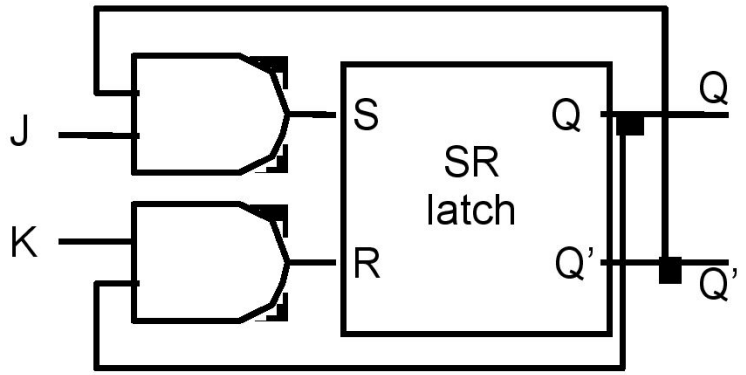
Characteristic Equation:

$$Q+ = Q\,K' + Q'\,J$$

JK Latch Using SR Latch

How to eliminate the forbidden state in SR?

Idea:
-Use output feedback to guarantee that R and S inputs are never both one
AND
-that J, K both one yields a toggle

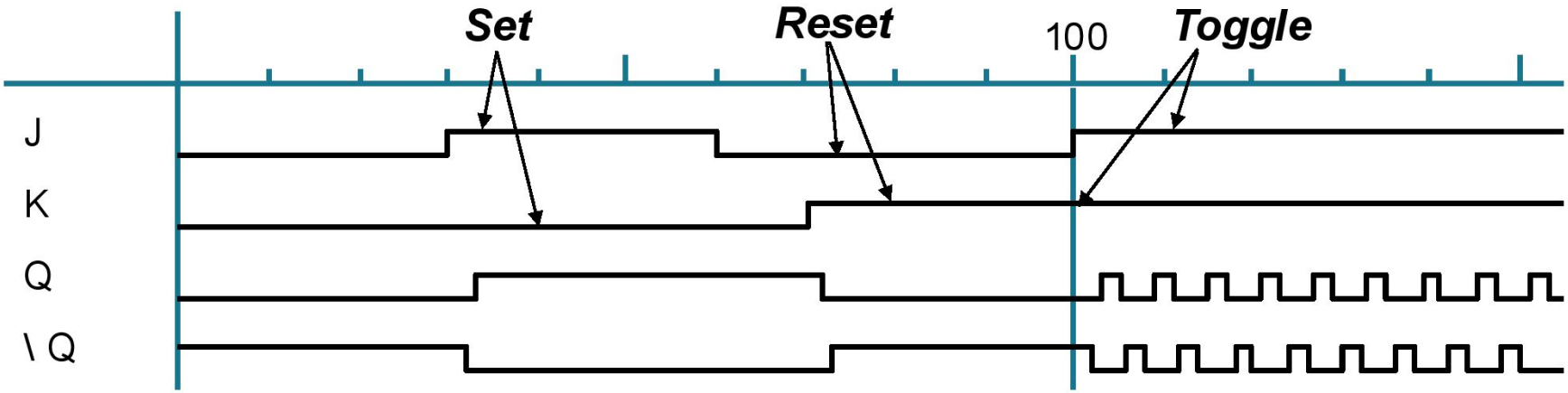


J(t)	K(t)	Q(t)	Q(t+Δ)	
0	0	0	0	HOLD
0	0	1	1	
0	1	0	0	RESET
0	1	1	0	
1	0	0	1	SET
1	0	1	1	
1	1	0	1	TOGGLE
1	1	1	0	

Characteristic Equation:

$$Q+ = Q\,\overline{K} + \overline{Q}\,J$$

JK Latch Race Condition



Toggle correctness condition: single state changes

Solution: Master/Slave flip-flop

Flip-Flops

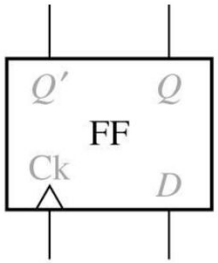
- Latches are “transparent” (= any change on the inputs is seen at the outputs after some delay)
- This may cause synchronization problems!
- Solution: use latches to create flip-flops that can respond (update) ONLY on SPECIFIC times (instead of ANY time)

Alternatives in FF choices

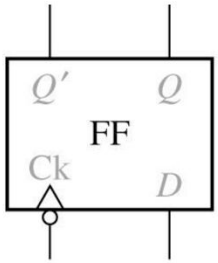
- **Types of FF**
  - RS
  - D
  - JK
  - T

D-FF

- **The most common type of FF**



(a) Rising-edge trigger



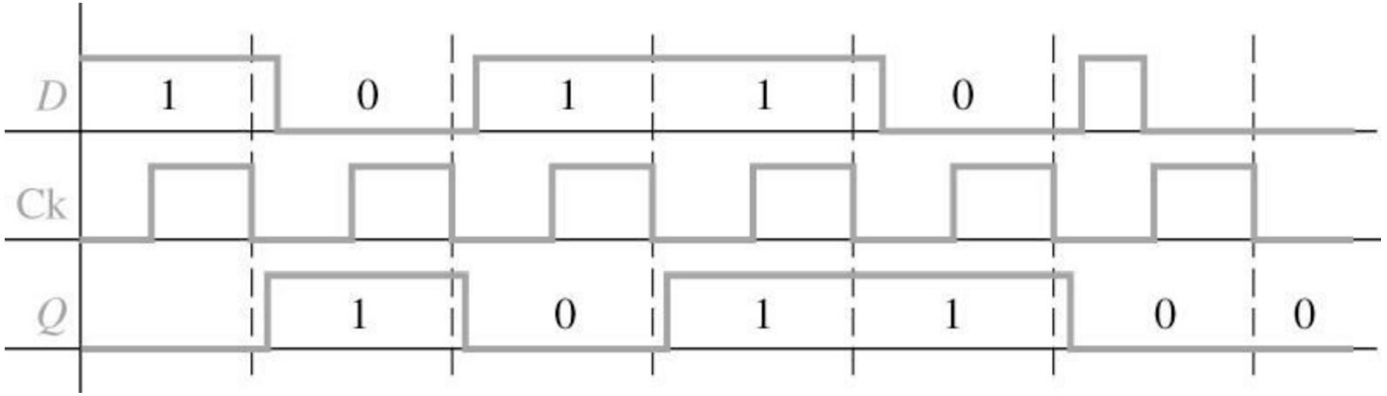
(b) Falling-edge trigger

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

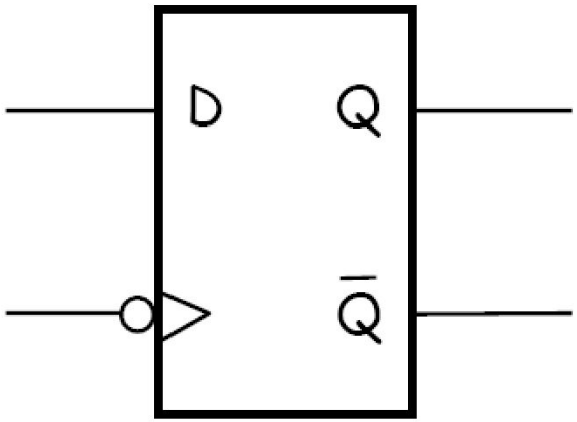
Truth table

$$Q^+ = D$$

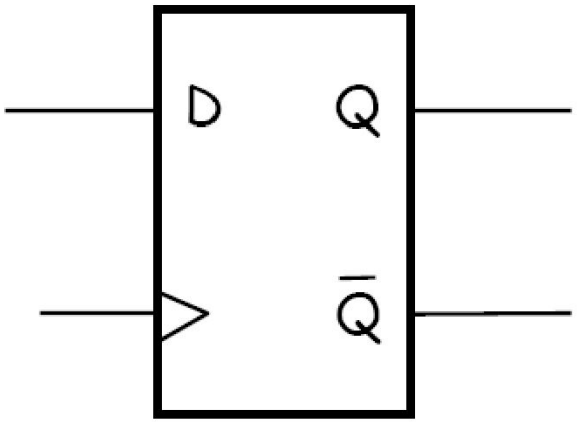
Timing for D Flip-Flop (Falling-Edge Trigger)



Symbols



Negative Edge Triggered

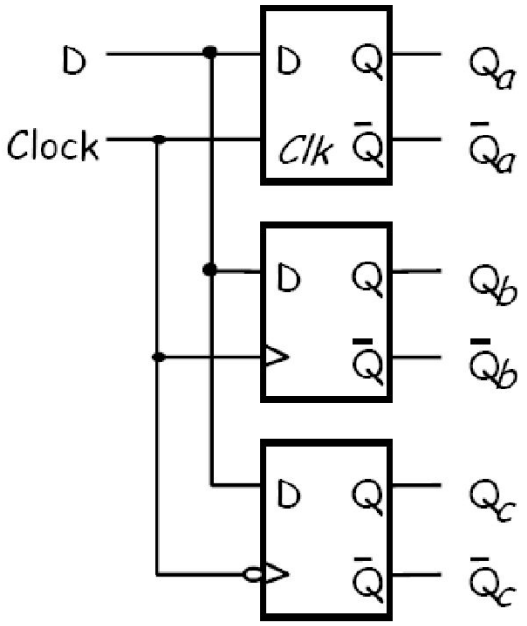


Positive Edge Triggered

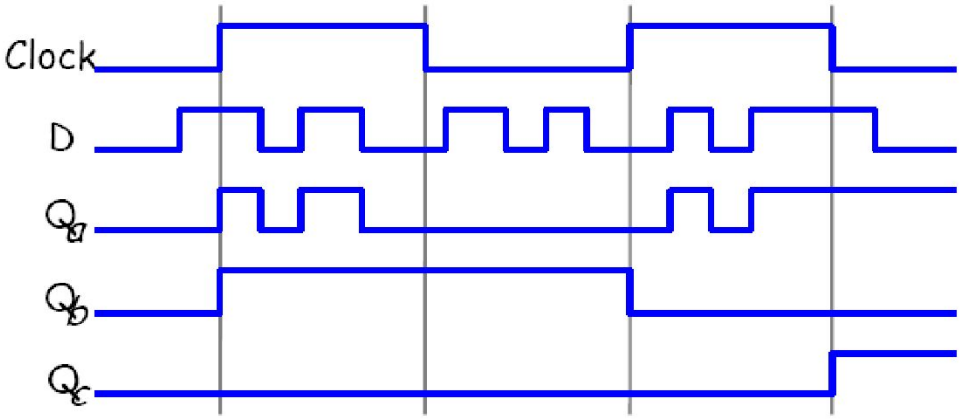
- **Notes:**
  - **The small triangles in the symbol show that it is a FF not a latch**
  - **The small bubbles show the positive or negative edge**

Compare Latch with FF's

- Top device is a latch (how do we know?)
- Bottom two are rising and falling edge flip-flops
- Delays not considered for simplicity

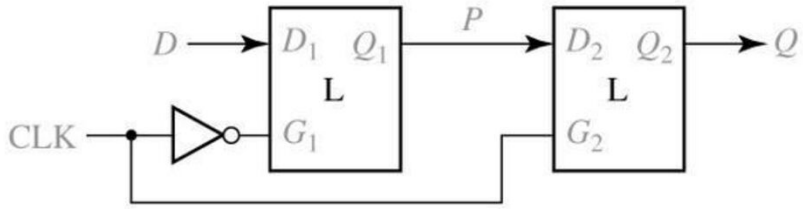


(a) Circuit

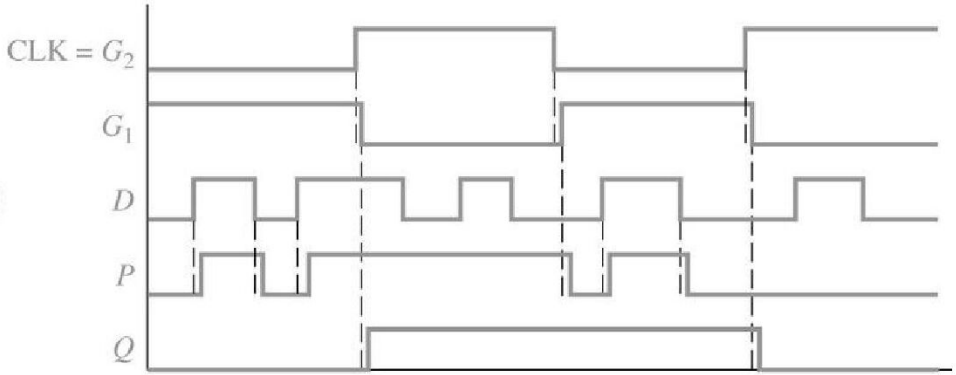


(b) Timing diagram

Rising Edge D-FF: How to make?



(a) Construction from two gated D latches



(b) Time analysis

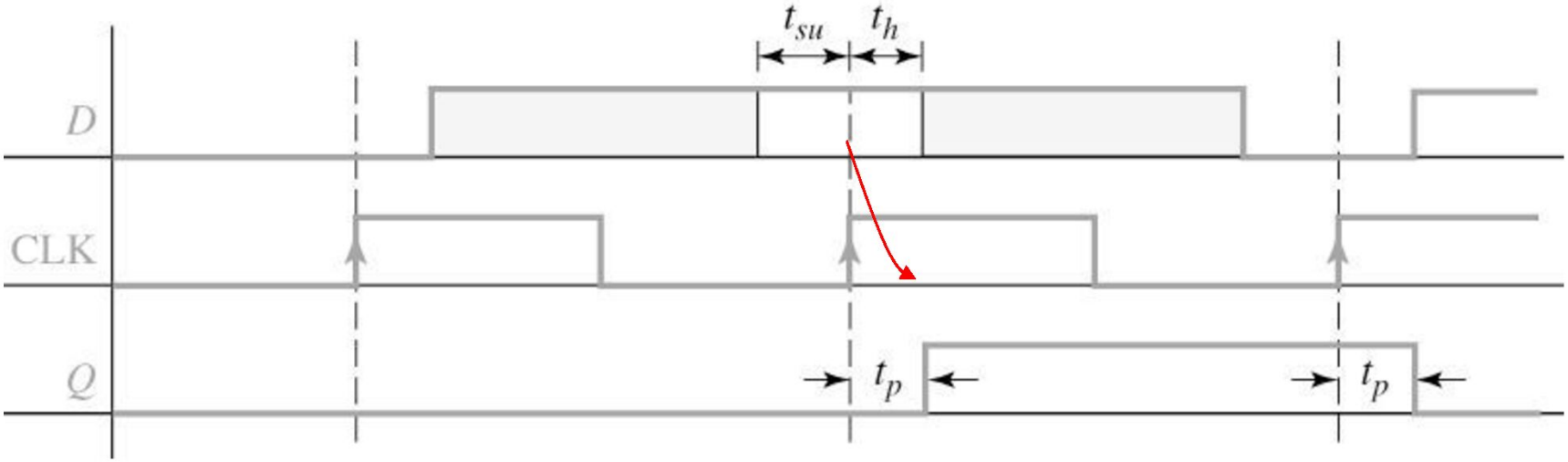
- One way is to use 2 cascaded latches
- What about falling-edge circuit?

Setup Time, Hold Time and Propagation Delay

- **Setup time:**
  - D input must be stable for a certain amount of time BEFORE the active edge of clock
- **Hold time:**
  - D input must be stable for a certain amount of time AFTER the active edge of the clock
- **Propagation Delay (Clock-to-Output):**
  - From the time the clock changes to the time the output changes
- **Propagation Delay (Data-to-Output):**
  - From the time the data changes to the time the output changes
  - Mainly defined for latches

Setup & Hold Time and Propagation Delay

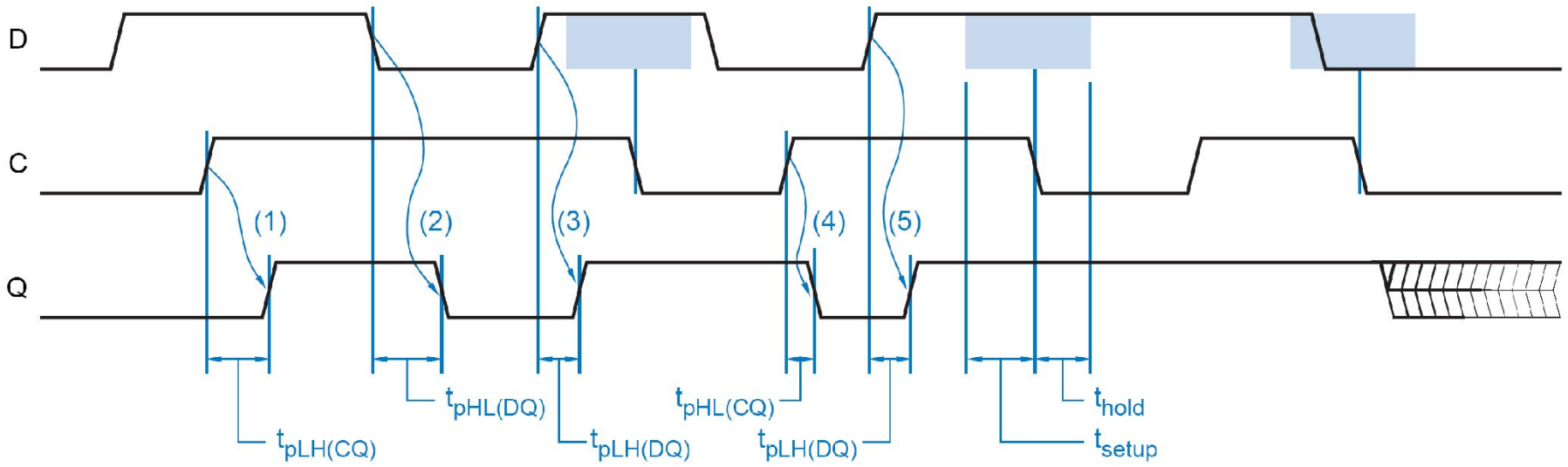
Setup and hold times for a rising edge-triggered D flip-flop



t_{pLH} may be different than t_{pHL}

t_p clock-to-output vs. t_p D-to-output

Timing Parameters of a D-Latch

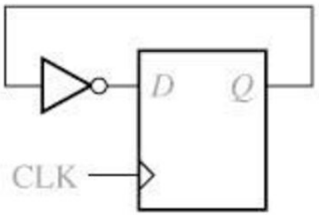


Timing Requirements

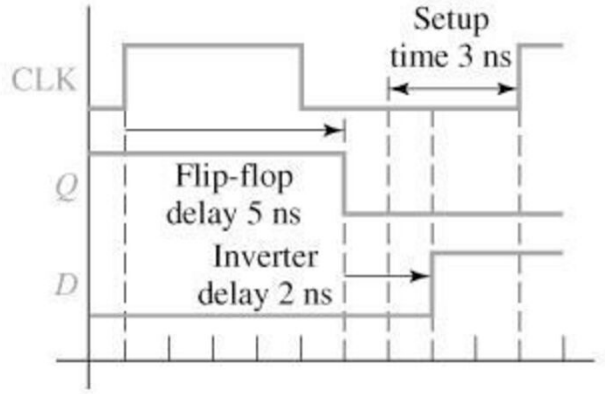
Determination of minimum clock period

Assume:

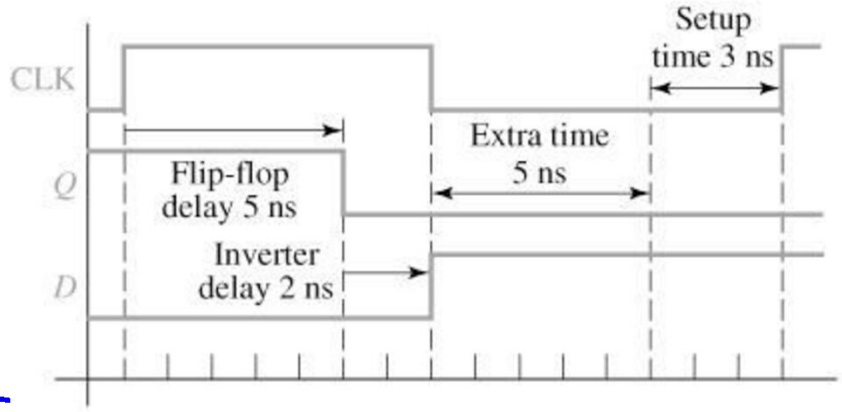
- t_{co} = 5 ns
- $t_{p,inv}$ = 2 ns
- t_{su} = 3 ns



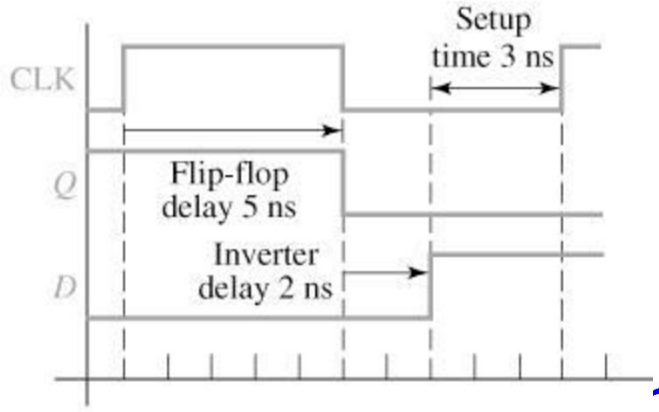
(a) Simple flip-flop circuit



(b) Setup time not satisfied

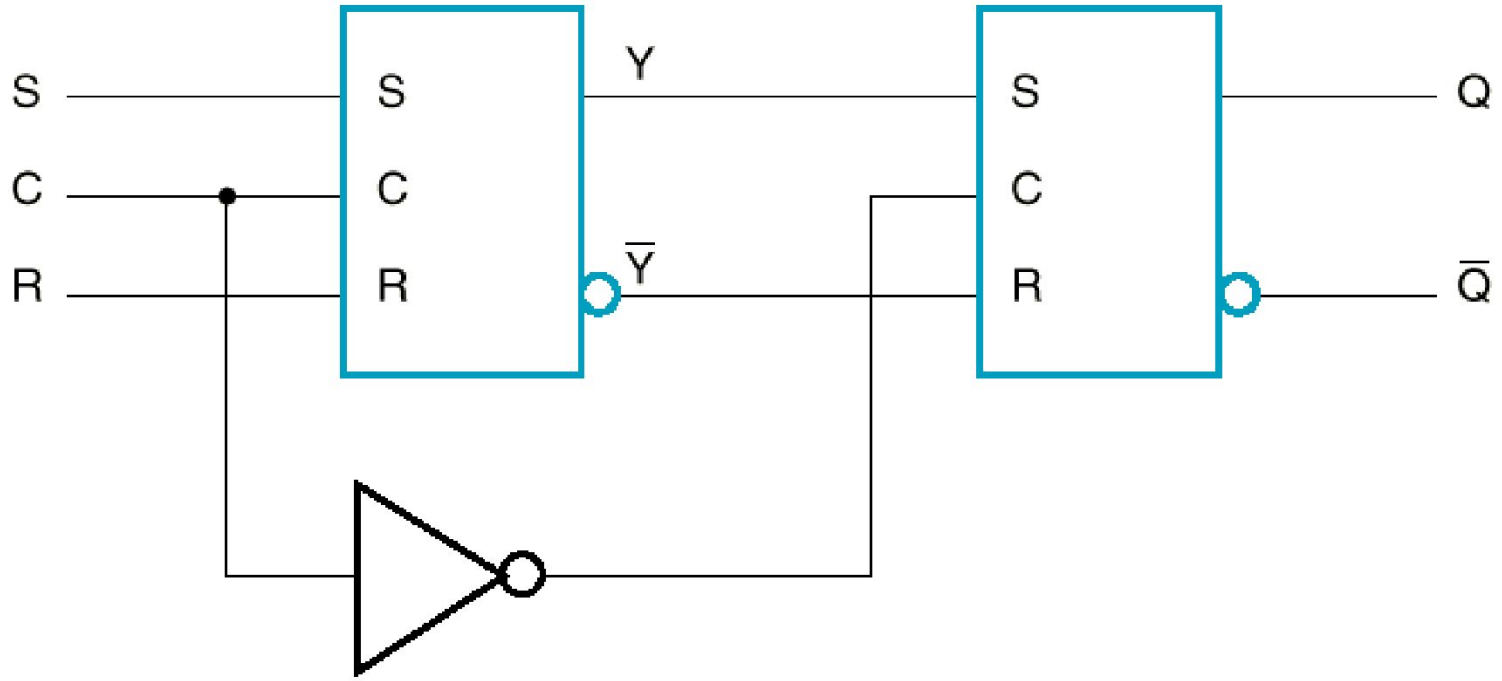


(c) Setup time satisfied



(d) Minimum clock period

Master-Slave FF configuration using SR latches

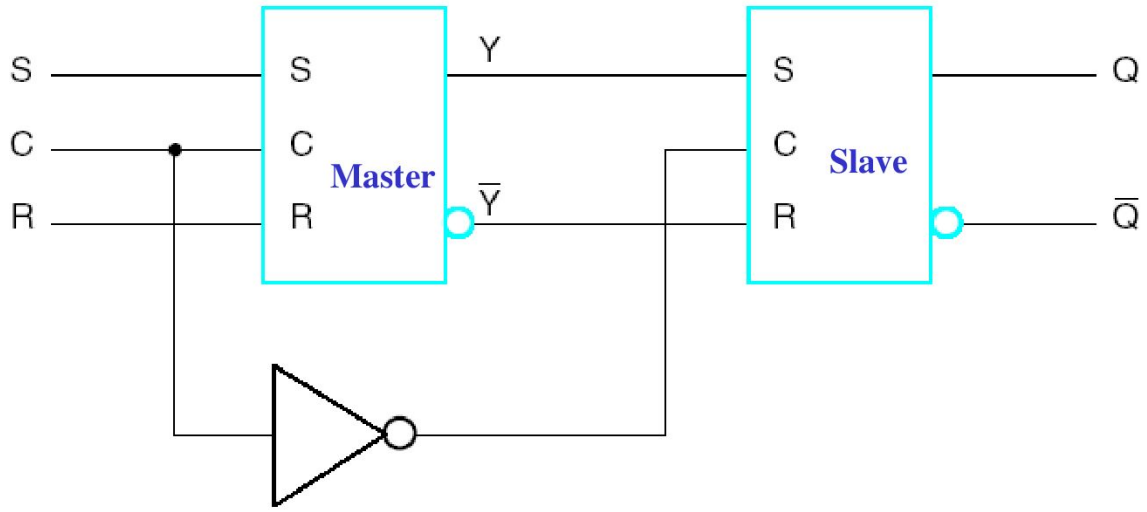


- Enables edge-triggered behavior

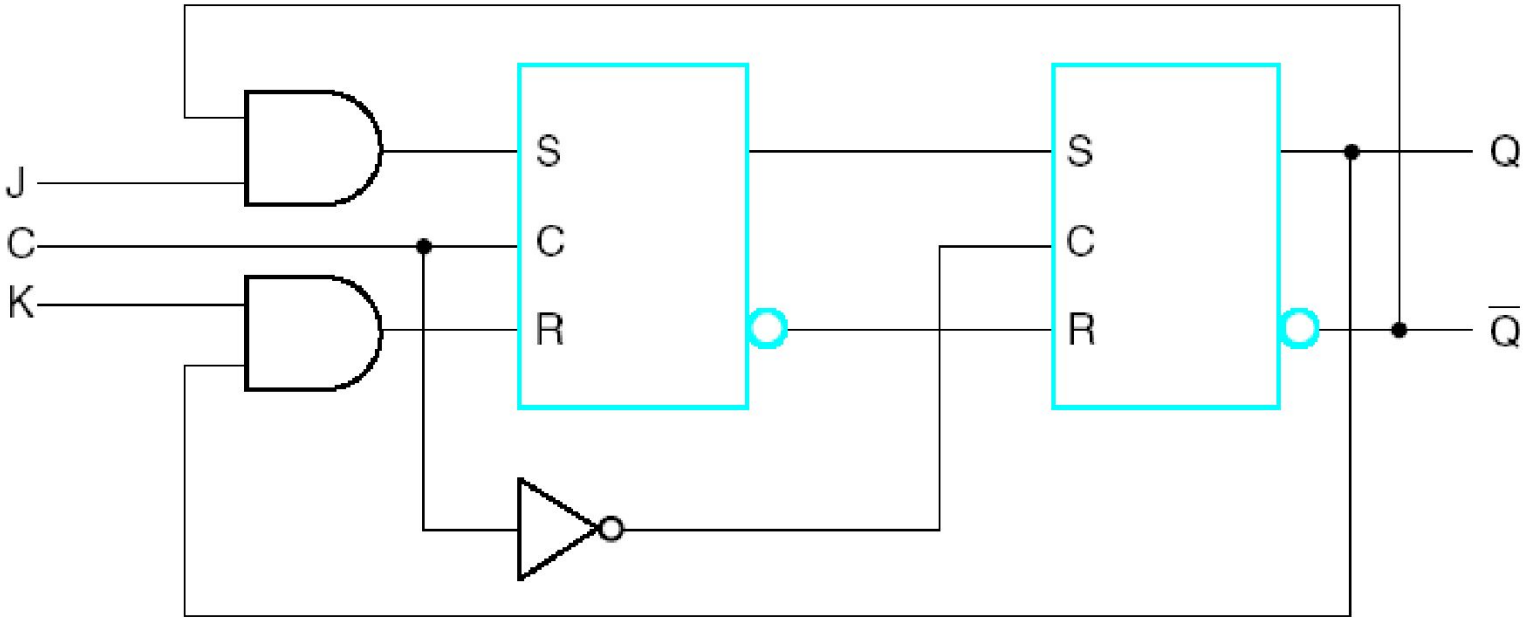
Master-Slave FF configuration using SR latches (cont.)

S	R	C	Q	Q′	
0	0	1	Q_0	Q_0'	Store
0	1	1	0	1	Reset
1	0	1	1	0	Set
1	1	1	1	1	Disallowed
X	X	0	Q_0	Q_0'	Store

- When *C*=1, master is enabled and stores *new* data, slave stores *old* data.
- When *C*=0, master's state passes to enabled slave (Q=Y), master not sensitive to new data (disabled).

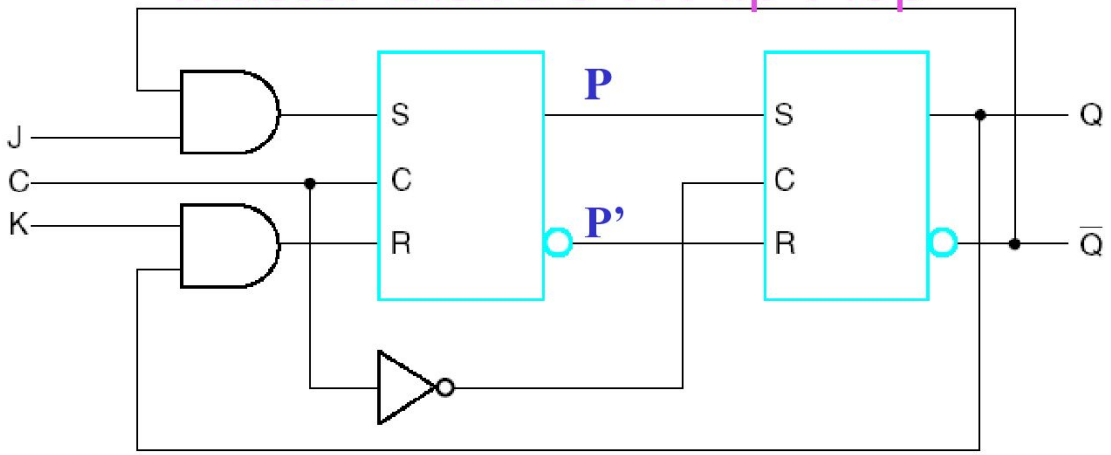


Master-Slave J-K Flip-Flop



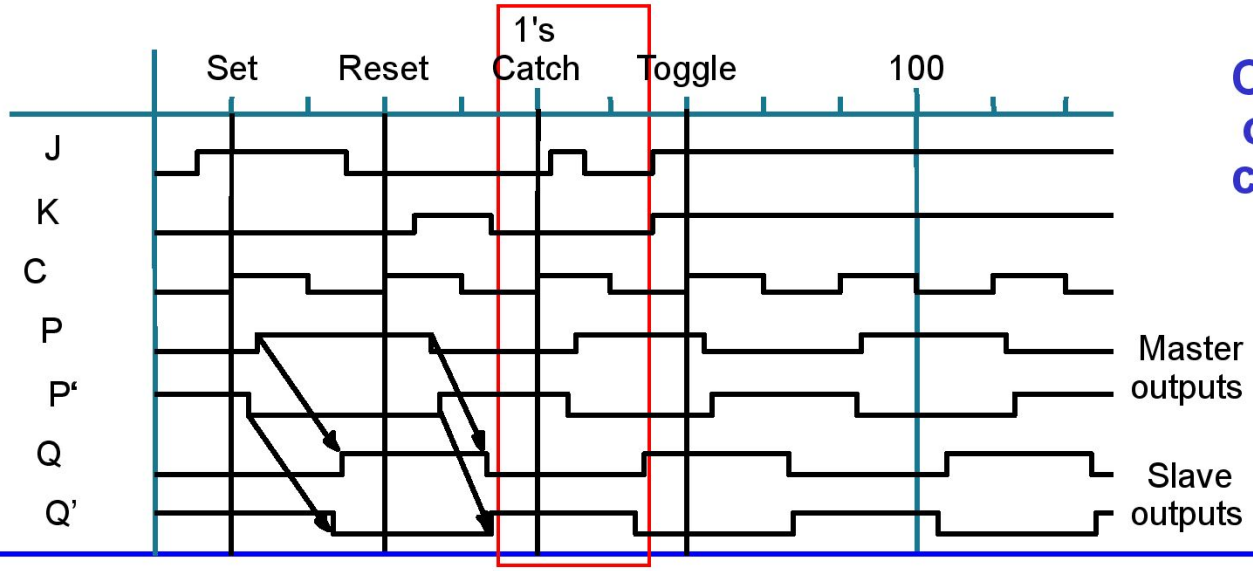
J	K	Next State of Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

Master-Slave J-K Flip-Flop



Master samples inputs while clock high

Slave samples inputs while clock low

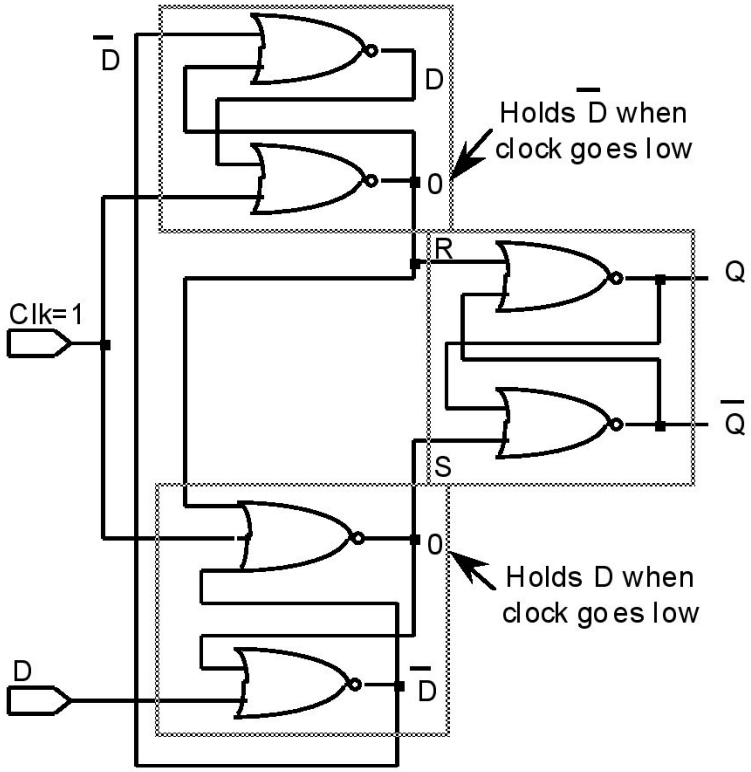


Correct toggle operation but catches J or K glitches!

Edge-Triggered FF

Problem with a Master-Slave FF:
- a 0-1-0 glitch on the J or K inputs leads to a state change!

Solution: **edge-triggered logic**



Negative Edge-Triggered D flipflop

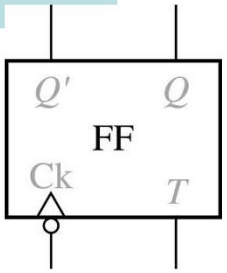
4-5 gate delays

setup, hold times necessary to successfully latch the input

Characteristic Equation:
Q+ = D

T Flip-Flop

T Flip-Flop



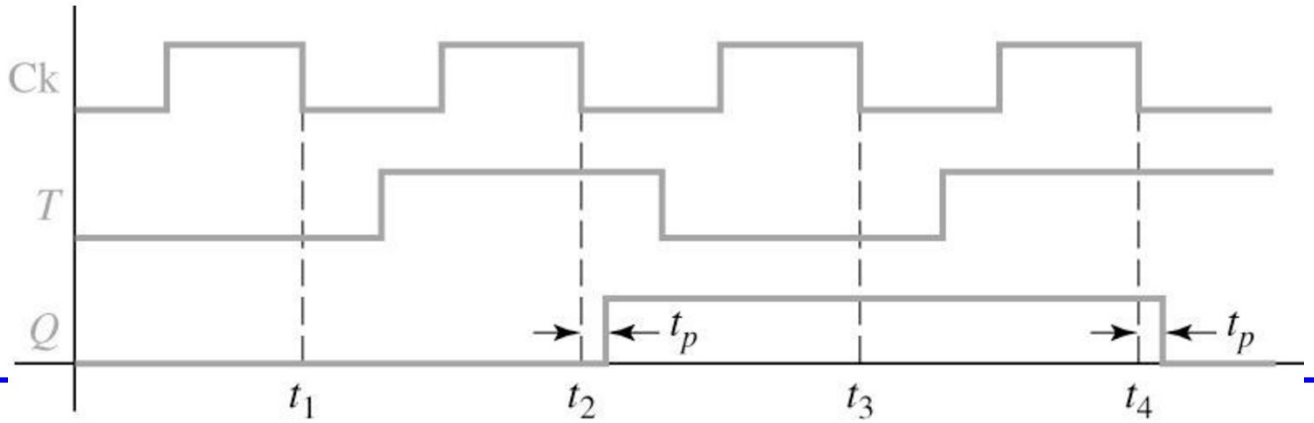
(a)

T	Q	Q^+
0	0	0
0	1	1
1	0	1
1	1	0

(b)

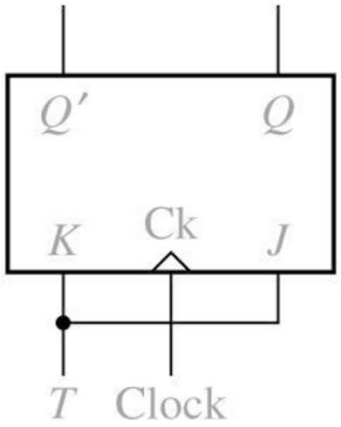
$$Q^+ = T'Q + TQ' = T \oplus Q$$

Timing diagram for a T flip-flop (falling-edge triggered)

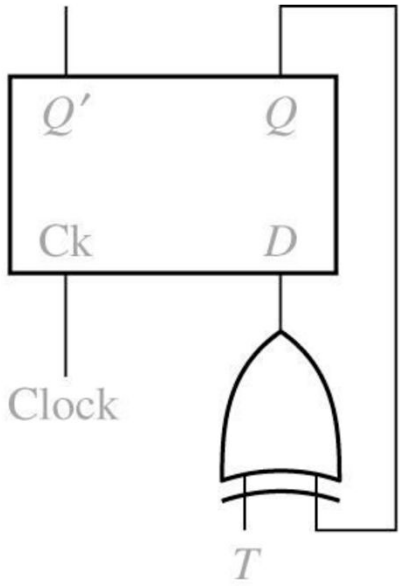


Implementation of T-FF

Implementation of a T flip-flop



(a) Conversion of J-K to T

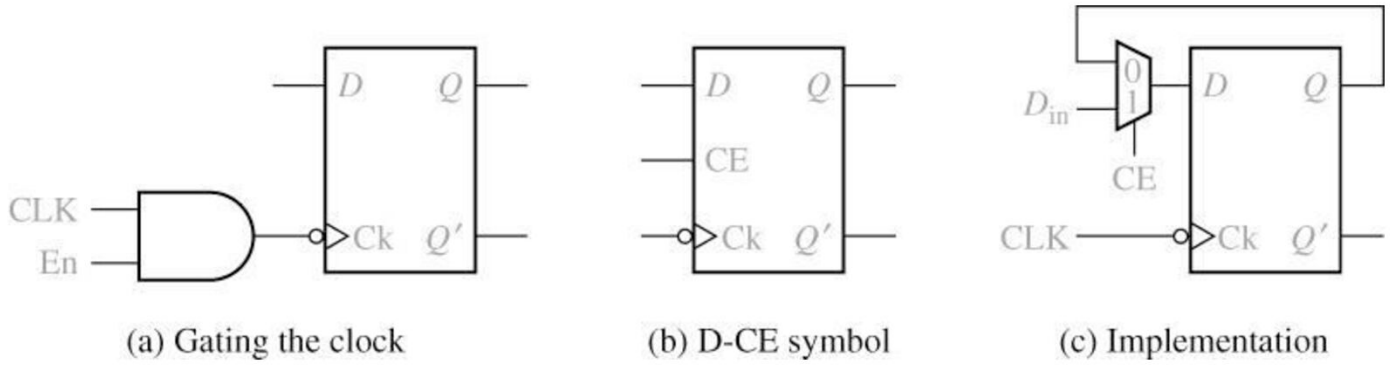


(b) Conversion of D to T

$$Q^+ = JQ' + K'Q = TQ' + T'Q$$

FFs with Additional Inputs

D flip-flop with clock enable



(a) Gating the clock (b) D-CE symbol (c) Implementation

The characteristic equation :

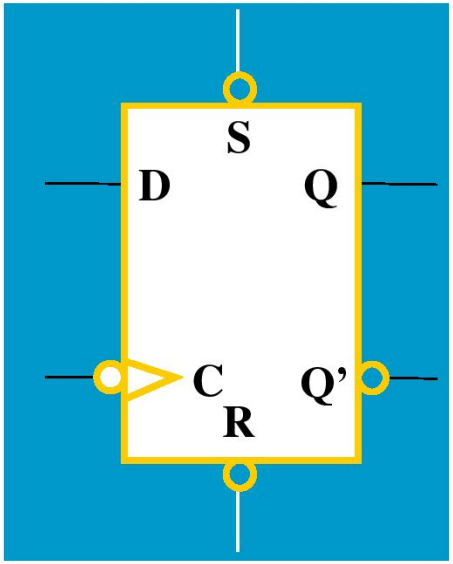
$$Q^+ = Q \cdot CE' + D \cdot CE$$

The MUX output :

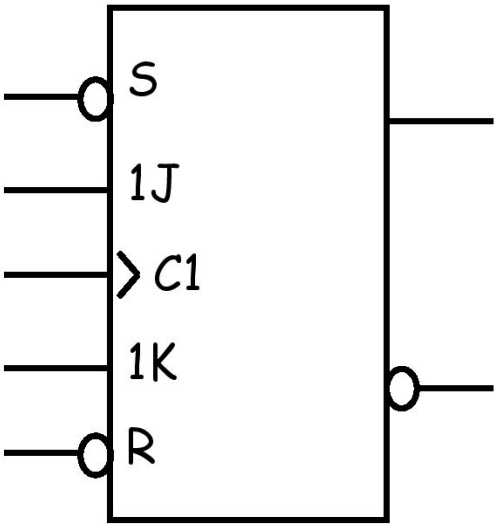
$$Q^+ = D = Q \cdot CE' + D_{in} \cdot CE$$

Asynchronous Preset/Clear

- Many times it is desirable to asynchronously (i.e., independent of the clock) set or reset FFs
- Example: At power-up so that we can start from a known state
- Asynchronous set == direct set == ***Preset***
- Asynchronous reset == direct reset == ***Clear***
- There may be "synchronous" preset and clear



Asynchronous Set/Reset



IEEE standard graphics symbol for JK-FF with direct set & reset

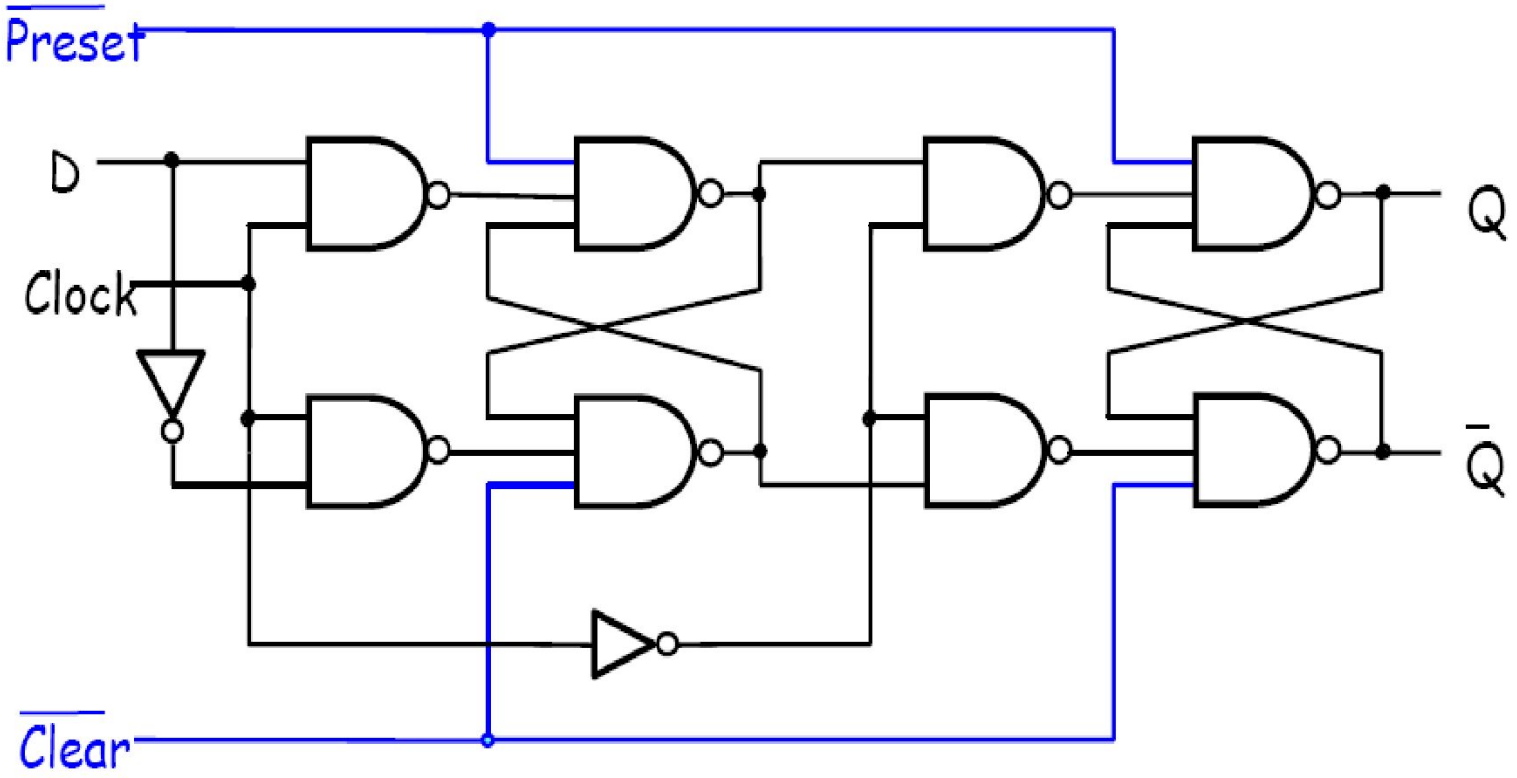
Cn indicates that Cn controls all other inputs whose label starts with n.
In this case, C1 controls 1J and 1K.

Function Table

S	R	C1	1J	1K	Q(t+1)
0	1	X	X	X	1 – *Preset*
1	0	X	X	X	0 – *Clear*
0	0	X	X	X	Undefined
1	1	↑	0	0	Q(t) – Hold
1	1	↑	0	1	0 – Reset
1	1	↑	1	0	1 – Set
1	1	↑	1	1	Q(t)' -- Complement

Asynchronous Inputs

Asynchronous Reset (this one has a preset too)



Synchronous Reset

Synchronous Reset: