CS221: Digital Design

Latches and FF

A. Sahu

Dept of Comp. Sc. & Engg.

Indian Institute of Technology Guwahati

Reference

VahidBook: Frank Vahid, Digital Design (Preview Edition), Wiely India Edition, 2005 *[[FF and Register design]]*

Outline

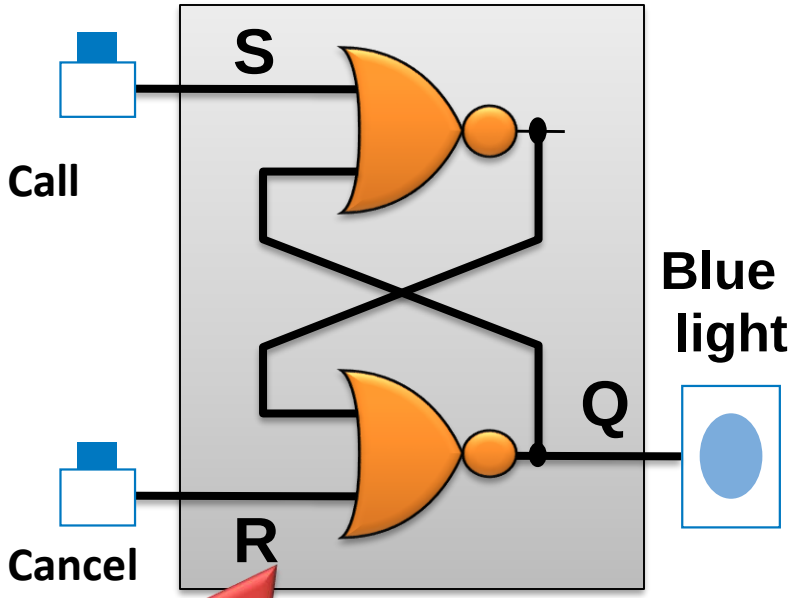
- Combinational Vs Sequential Logic Design
- Design a **Latch**, that stores one bit
  - RS latch, **Race condition**
- Stabilizing RS latch : Level Sensitive

- Clocked Latch : Flip Flop- Edge Sensitive
- JK-Latches (**Race in JK Latch**), JK-FF, T flip flops
- Characterization Table and Equation
  - RS, D, JK and T Flip flop

SR Latch Race Condition

Problem with SR Latch: SR=11

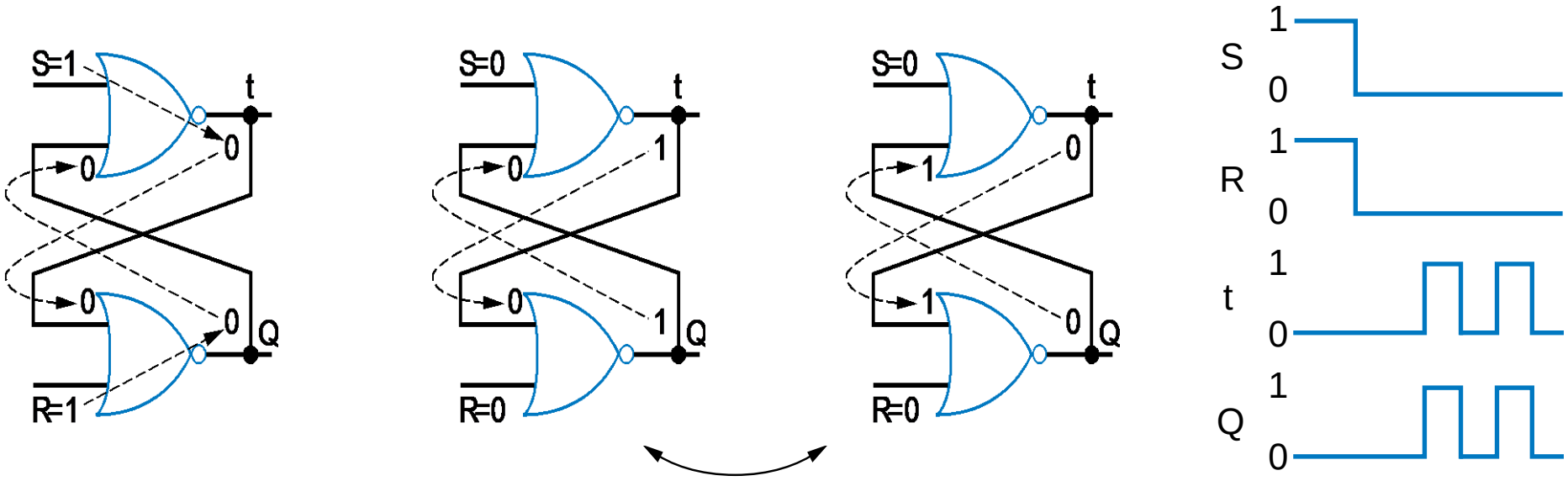
- Problem
  - If S=1 and R=1 simultaneously, we don't know what value Q will take (Unpredictable)



Race Condition :-

Who will win 0 or 1?
Eventually, what will be the value of Q?

Problem with SR Latch

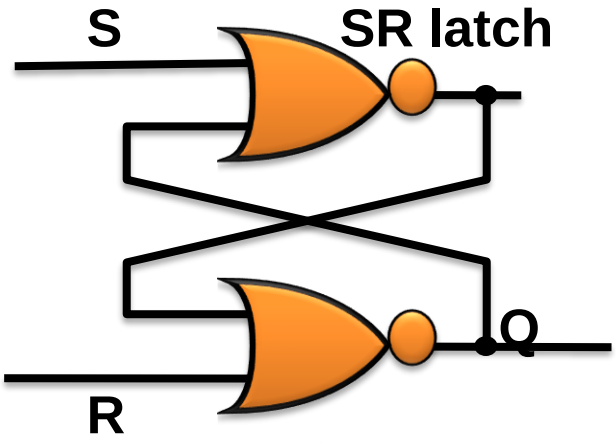


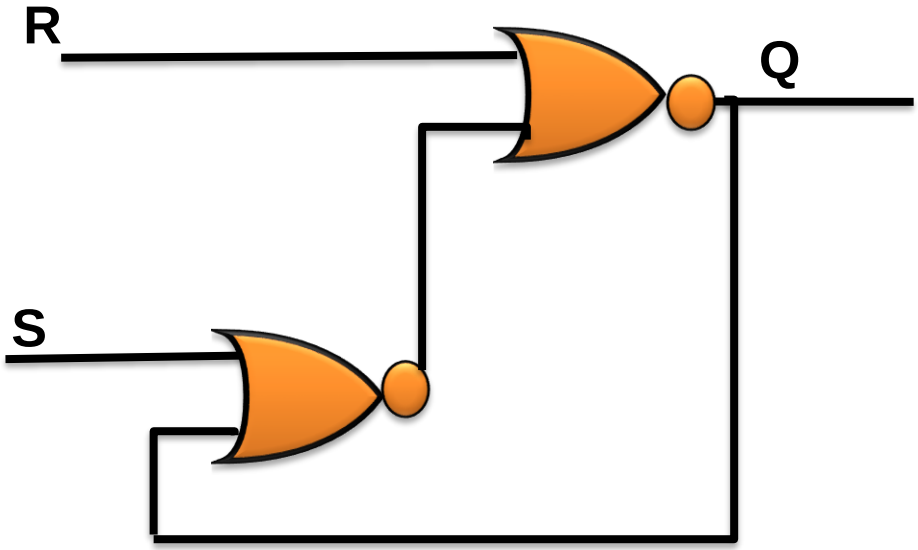
Action 1: S=1, R=1: Q and Q' both settled at 0

Action 2: Then change to S=0, R=0

Problem: After changing to S=0, R=0 ==> Non Predictable behavior : Q should be 0, but it do not

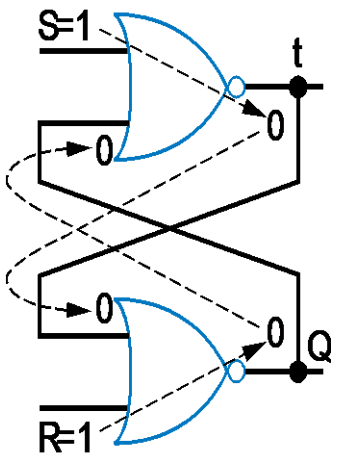
R-S Latch Feedback Different View

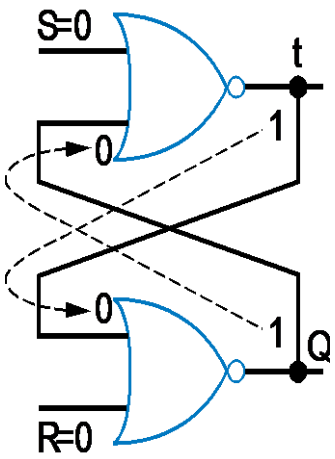


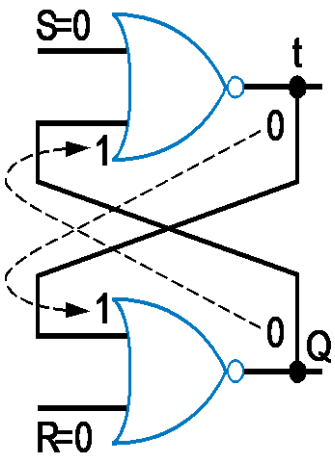


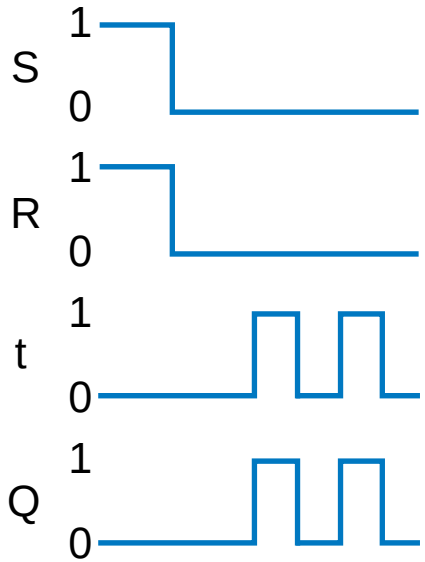
Looks compact, can be drawn in small space, proper cross coupled view

Problem with SR Latch

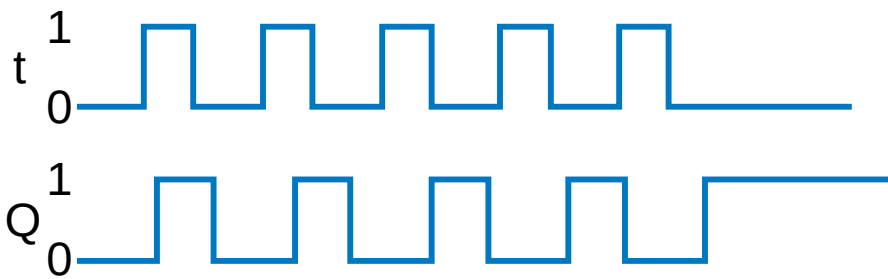






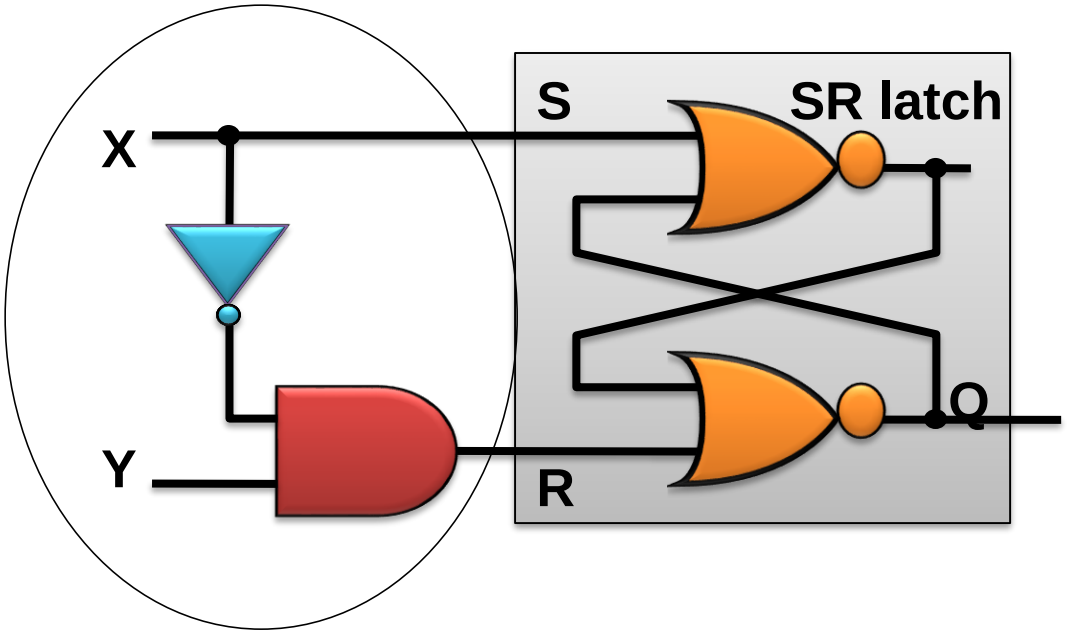


Q may oscillate. Then, because one path will be slightly longer than the other, Q will eventually settle to 1 or 0 – but we don't know which.



Solution to Race Condition

- We try to avoid S=1 and R=1 by the following circuit



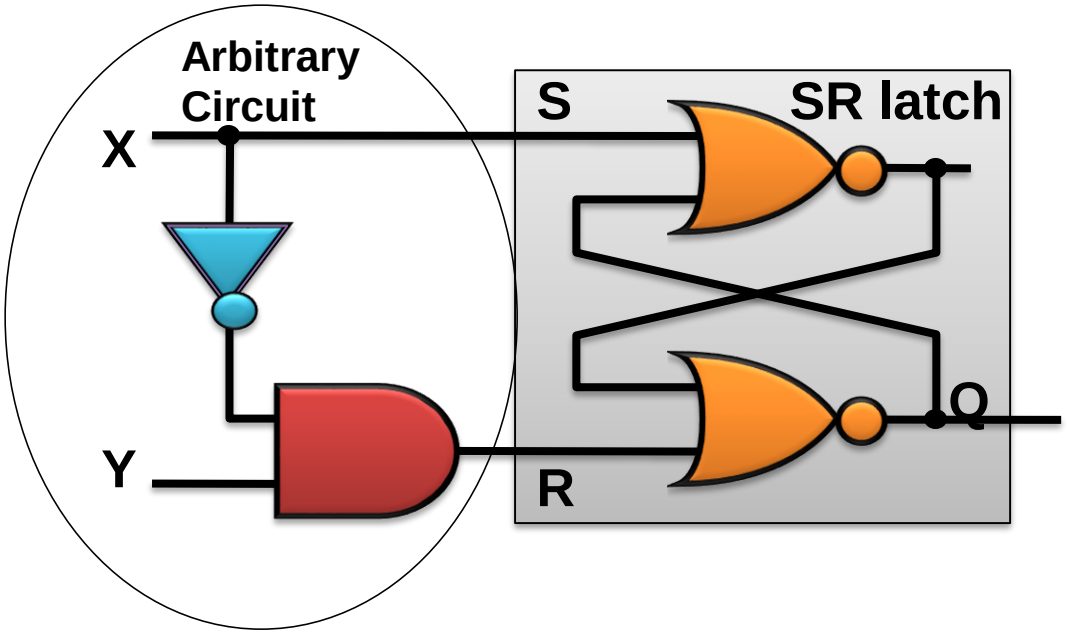
- Can we ensure value S=1 and R=1 will not happen at the same time?

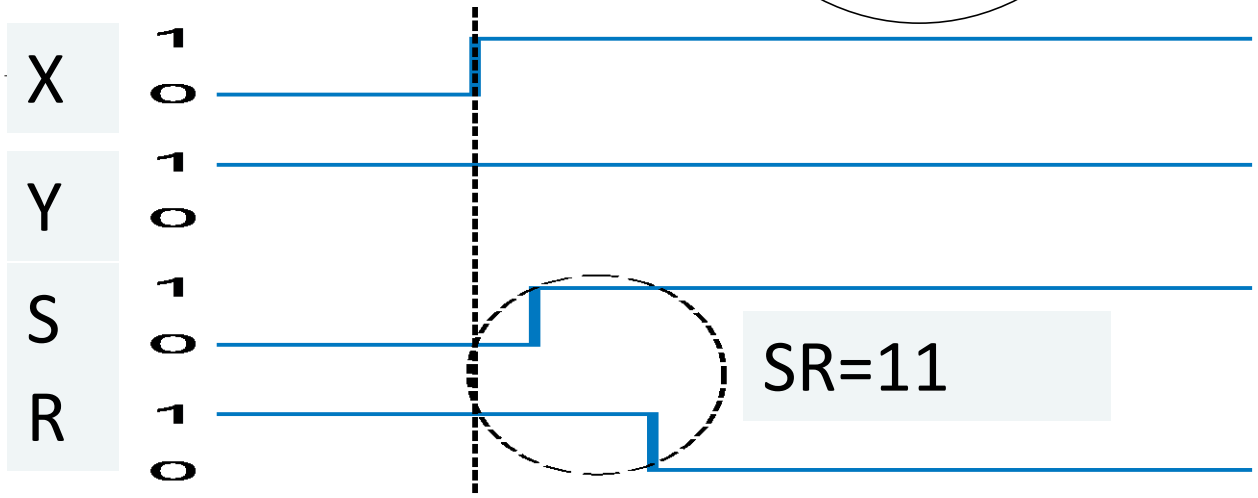
Problem with SR Latch

- Problem not just one of a user pressing two buttons at same time
- Can also occur even if SR inputs come from a circuit that supposedly never sets S=1 and R=1 at same time
  - But does, due to different delays of different paths

Problem with SR Latch

- X=1 and Y=1 set

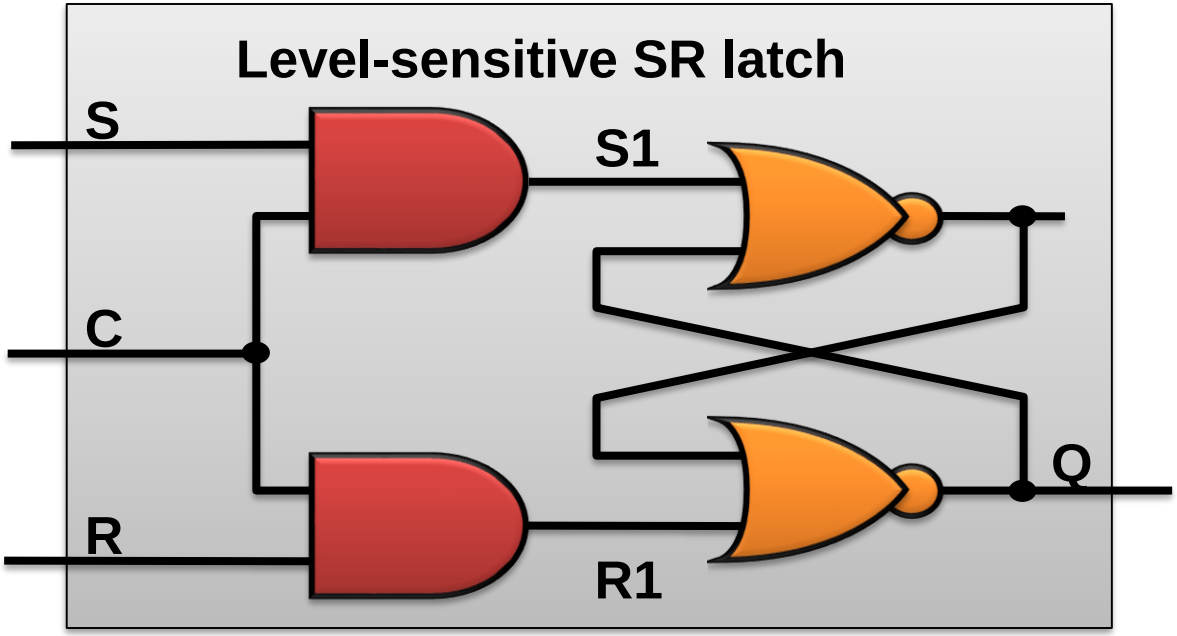


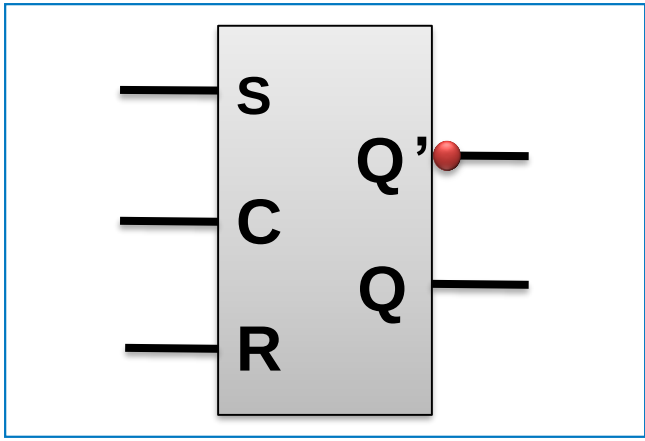


The longer path from X to R than to S causes SR=11 for short time – could be long enough to cause oscillation

Solution: Level-Sensitive SR Latch

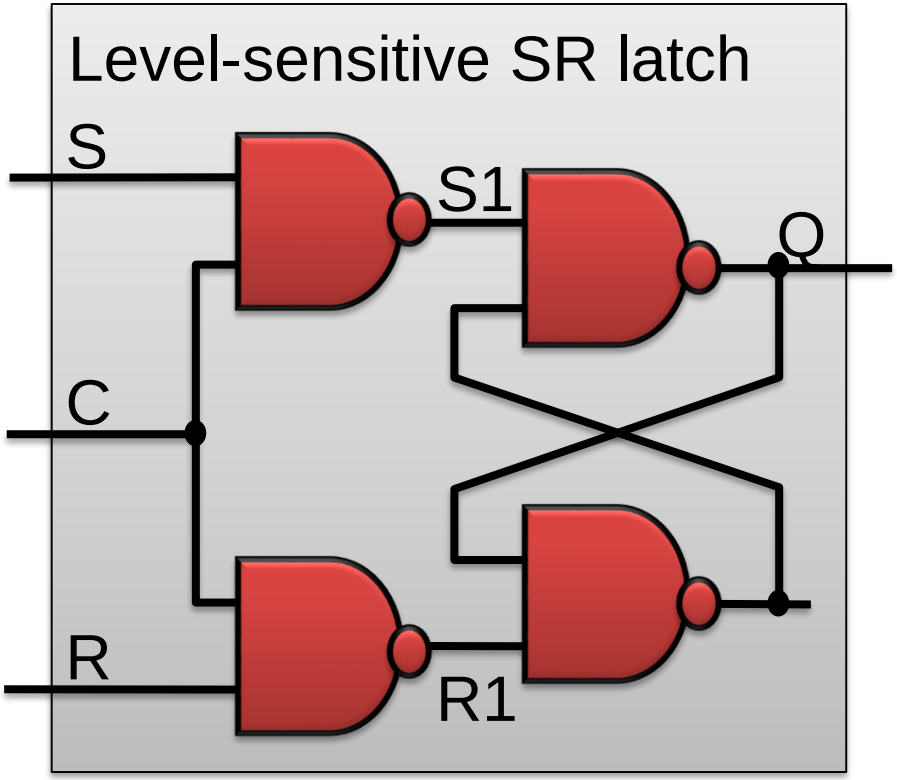
- Add enable input “C or En”
  - Only let S and R change when C =0 : Ensure circuit in front of SR never sets SR=11, except briefly due to path delays
  - Change C to 1 only after sufficient time for S and R to be stable
  - When C becomes 1, the stable S and R value passes through the two AND gates to the SR latch’s S1 R1 inputs.

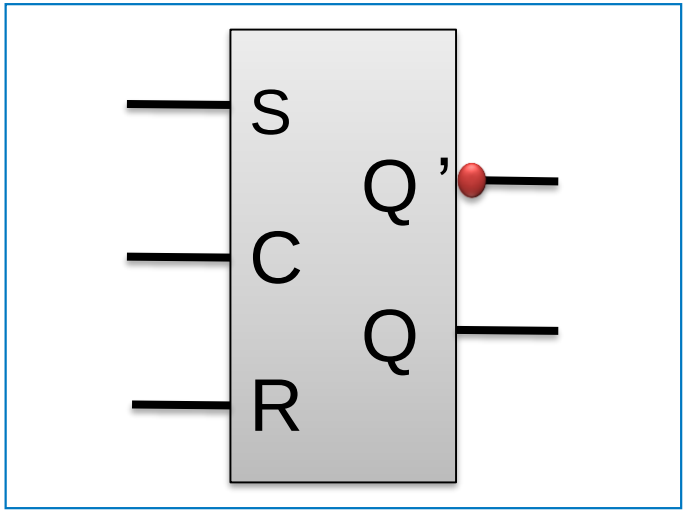




Level-sensitive SR latch symbol

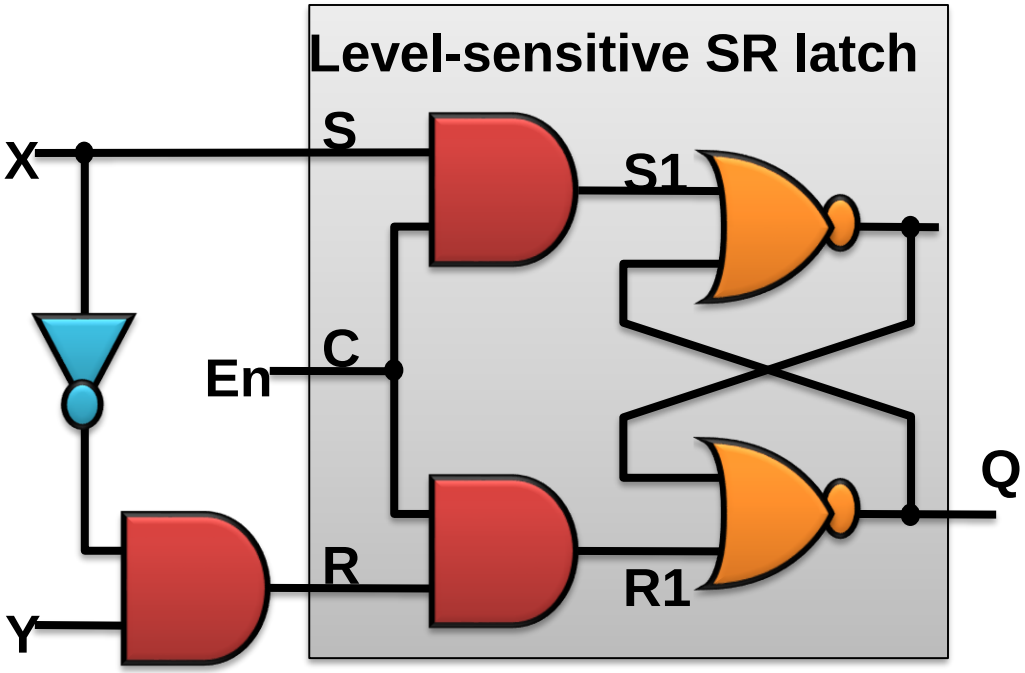
Level Sensitive: SR Latch with NAND Gate

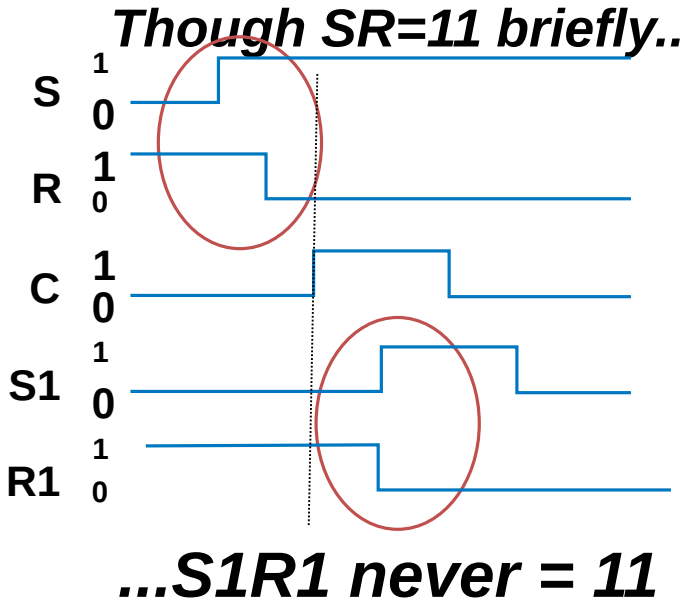




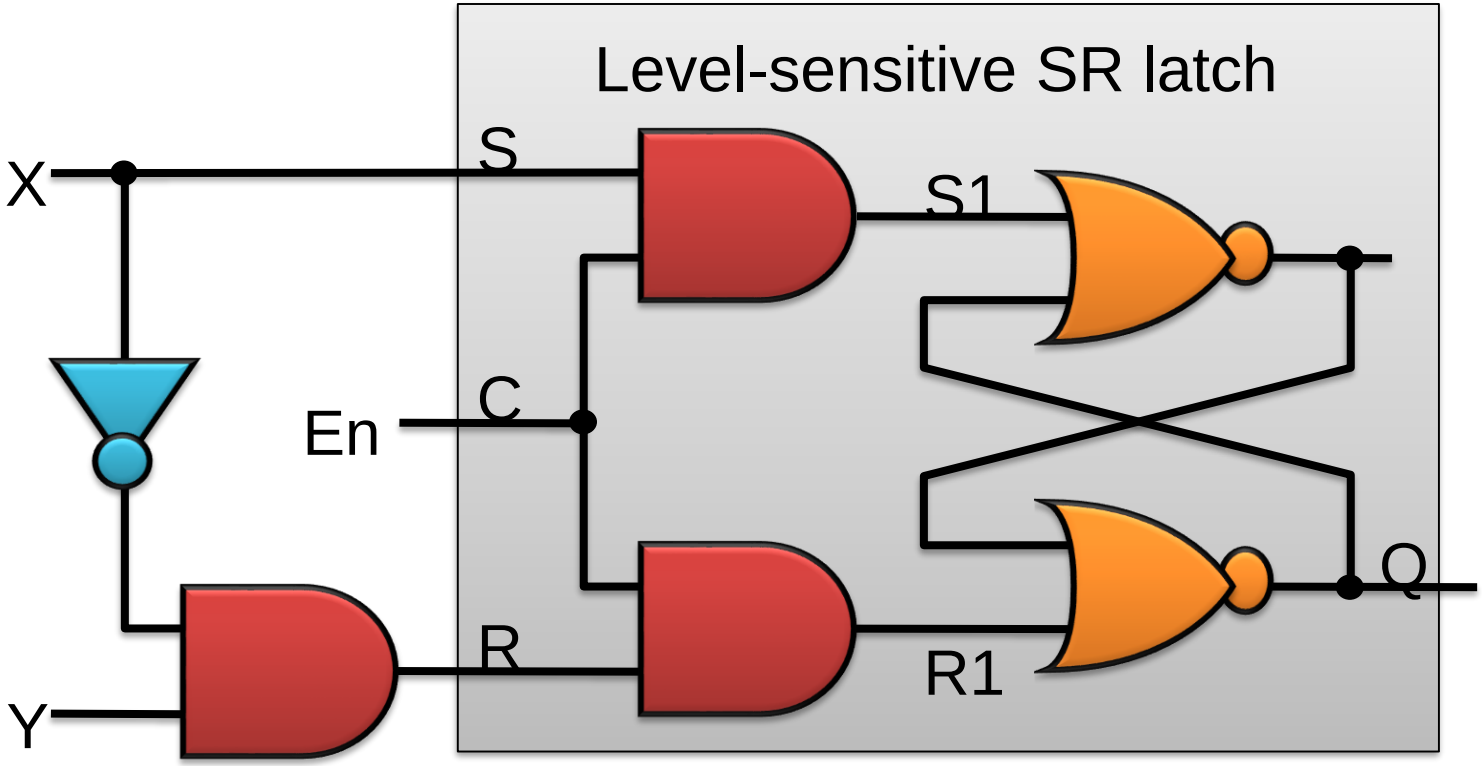
Level-sensitive SR latch symbol

Ensure S=1 and R=1 should not happened to Level-Sensitive SR Latch



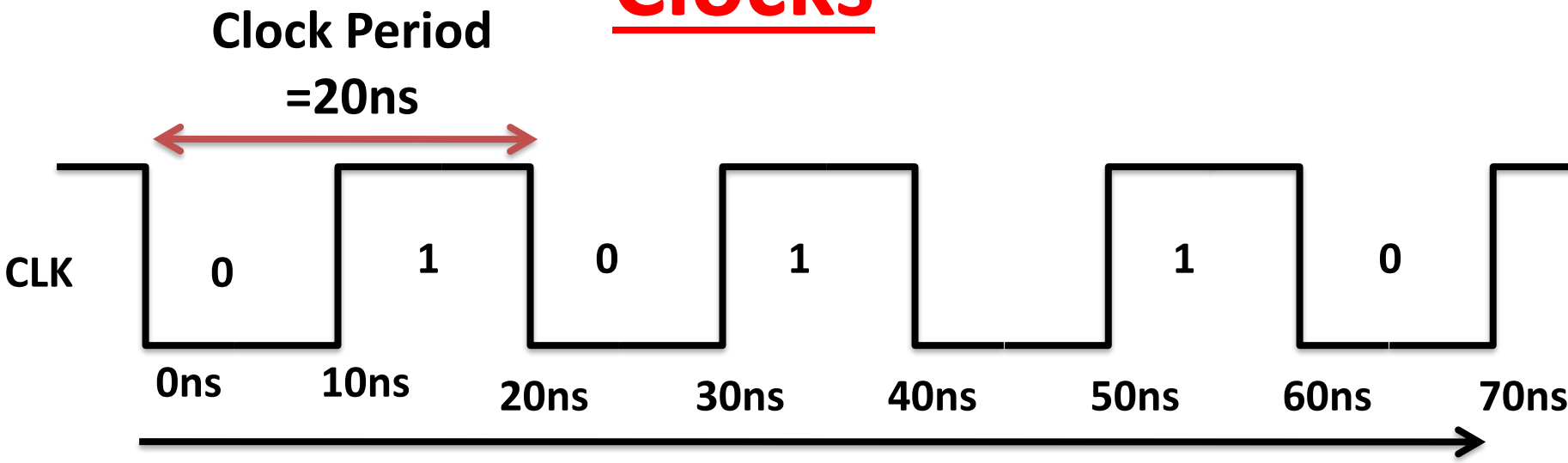


Solution: Ensure, Stabilize, Store



Ensure Stage	Stabilize Stage	Store Stage
Never Happens SR=11	When C=0 Stabilize SR and Use when C=1	Store bit

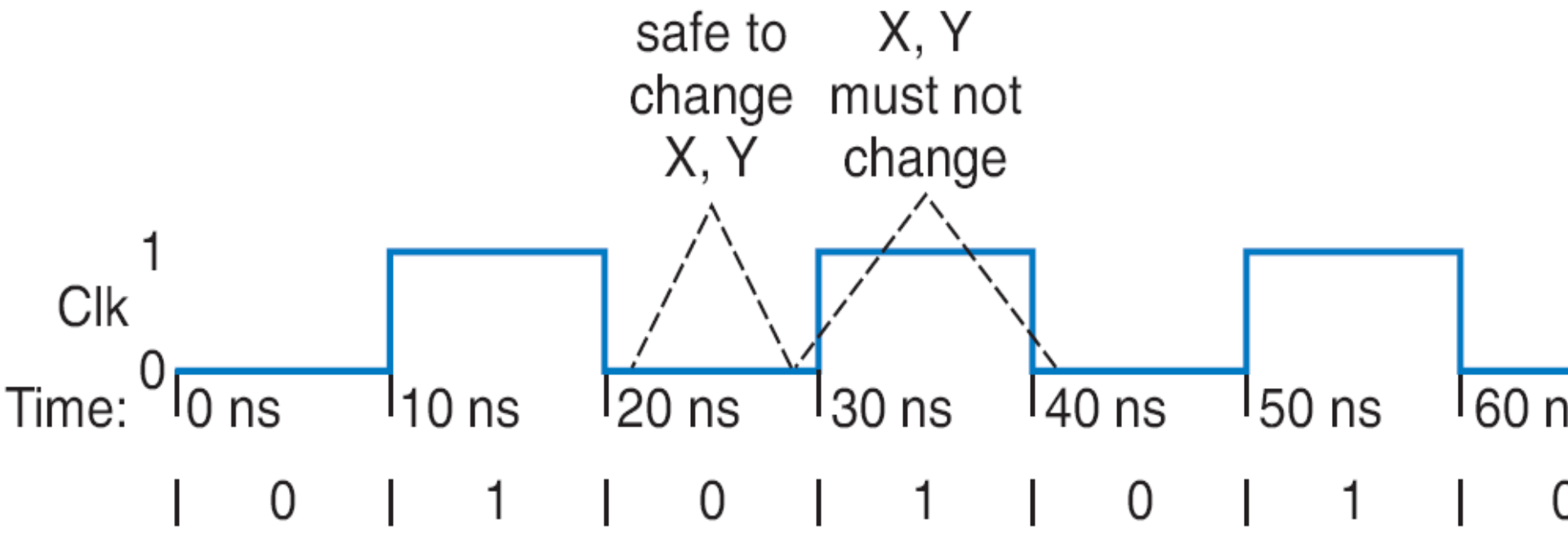
Clocks

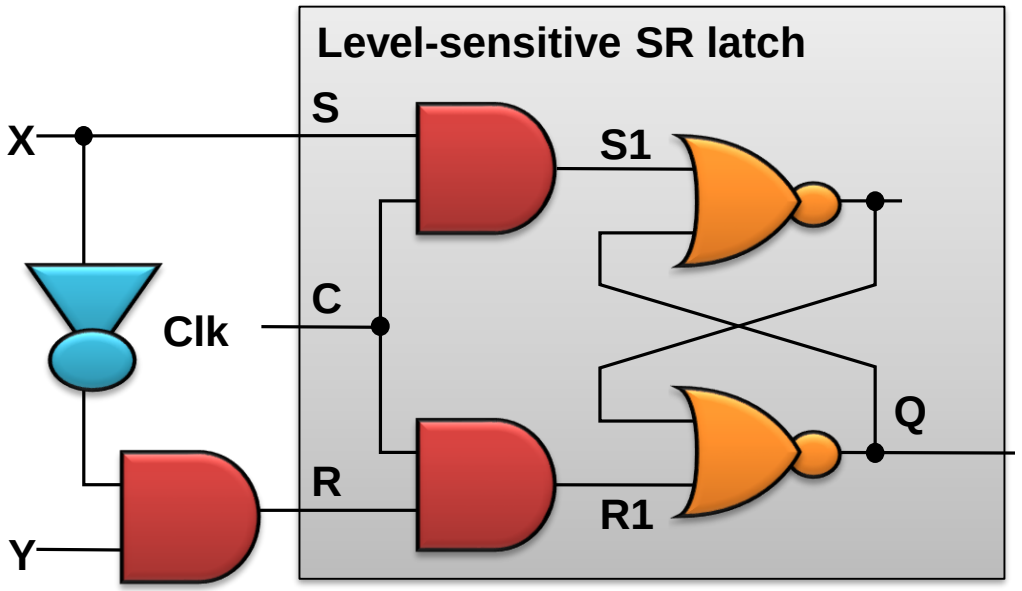


- ***Clock period***: time interval between pulses
- ***Clock cycle***: one such time interval
- ***Clock frequency***: 1/period
  - frequency = 1 / 20 ns = 50 MHz
    - 1 Hz = 1/s

Freq	Period
1 GHz	1 ns
100 MHz	10 ns
10 MHz	100 ns

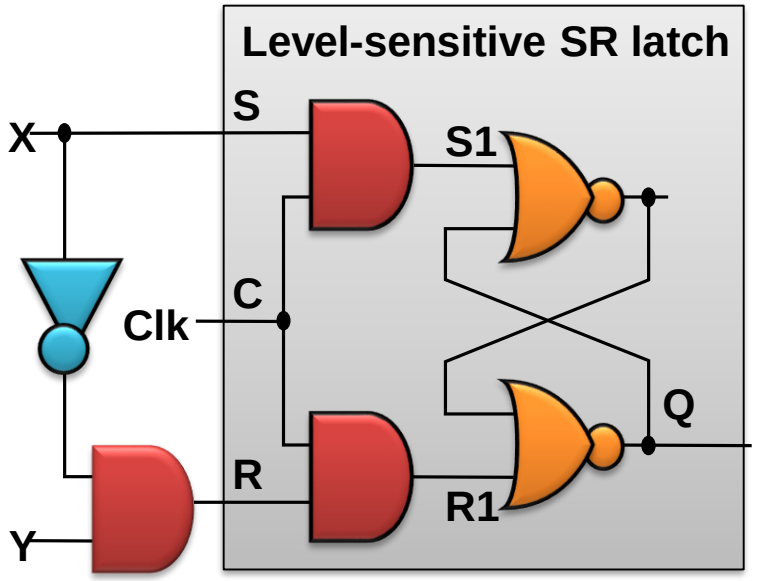
Clock Signal for RS latch





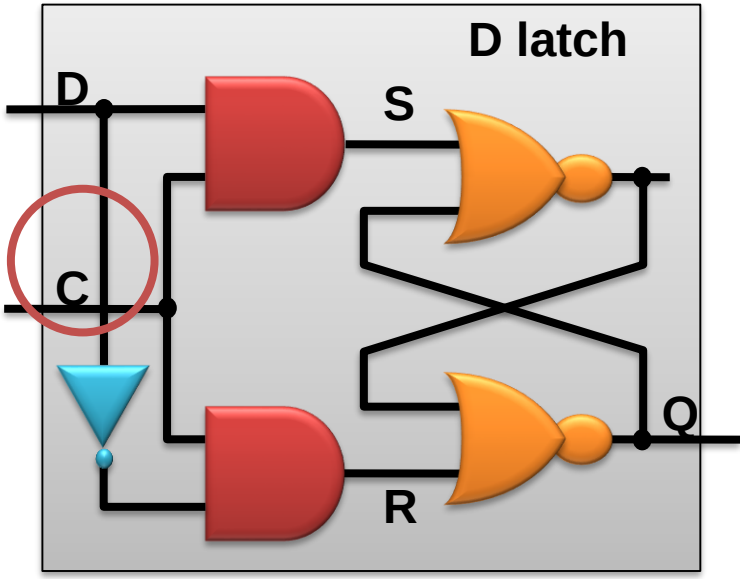
Clock Signals for a RS Latch

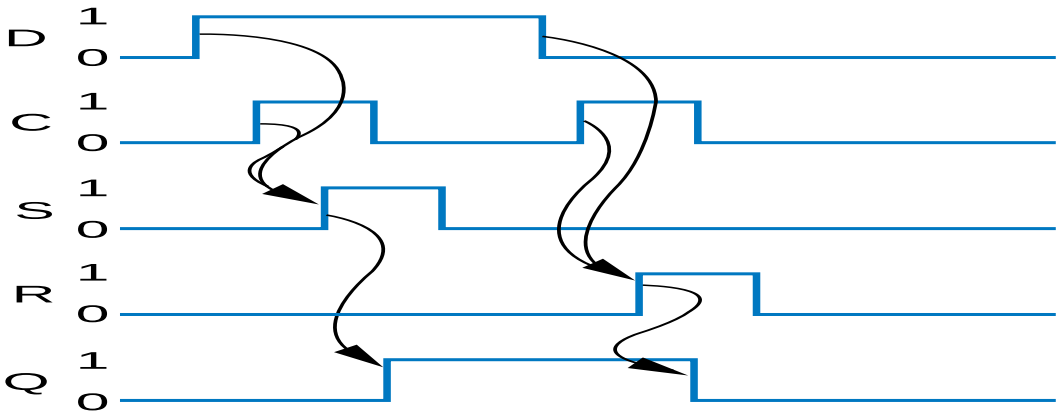
- How do we know when it's safe to set C=1?
  - Most common solution –make C pulse up/down
  - C=0: Safe to change X, Y C=1: Must *not* change X, Y
  - ***Clock*** signal -- Pulsing signal used to enable latches
    - Because it ticks like a clock
  - Sequential circuit whose storage components all use clock signals: ***synchronous*** circuit

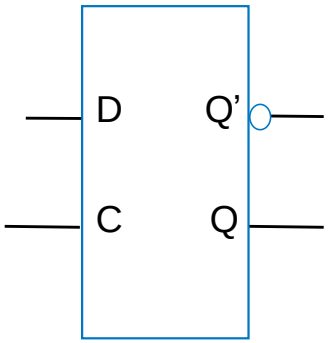


Level-Sensitive D Latch

- SR latch requires careful design to ensure SR=11 never occurs
- D latch relieves designer of that burden
  - Inserted inverter ensures R always opposite of S



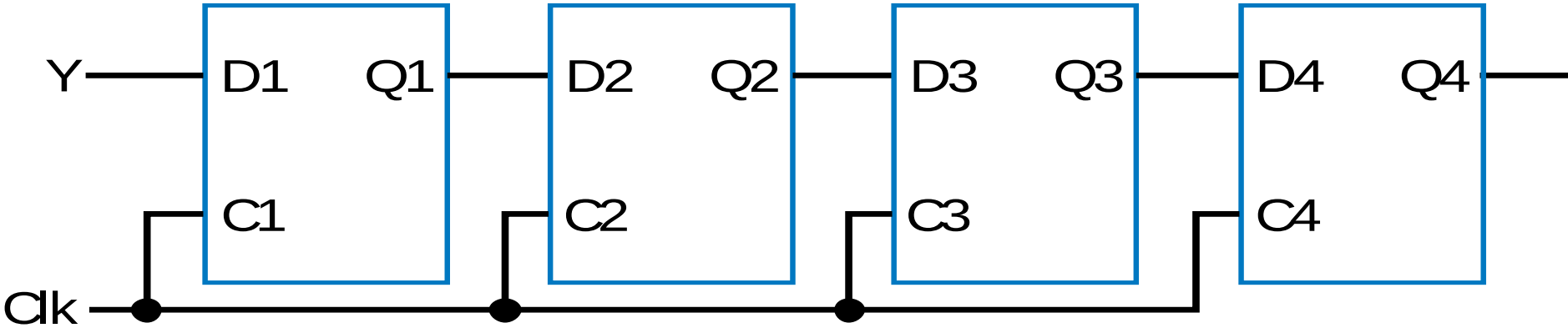




D latch symbol

Level-Sensitive D Latches

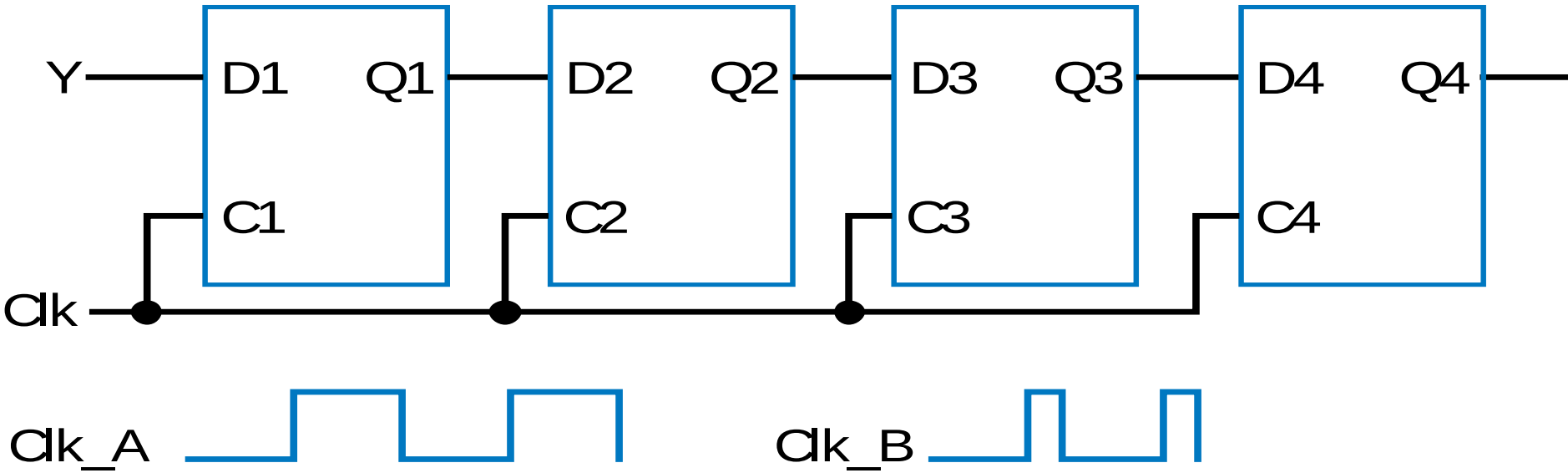
- Suppose D FFs are arrange in linear fashion connected using a single clock signal Clk
- Every clock we want to shift one bit to right
  - Right shift one bit per cycle



Does this circuit (with level sensitive D latch) Shift one bit per cycle ?

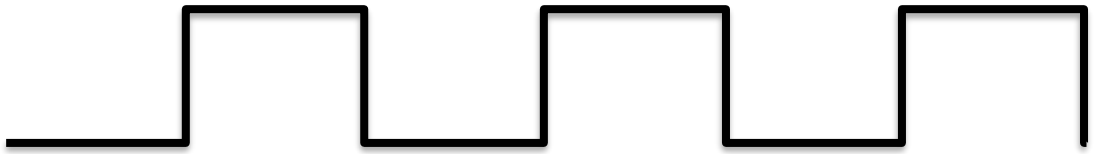
Problem with Level-Sensitive D Latch

- D latch still has problem (as does SR latch)
  - When C=1, through how many latches will a signal travel?
  - Depends on for how long C=1
    - Clk_A -- signal may travel through multiple latches
    - Clk_B -- signal may travel through fewer latches
  - **Hard to pick C that is just the right length**

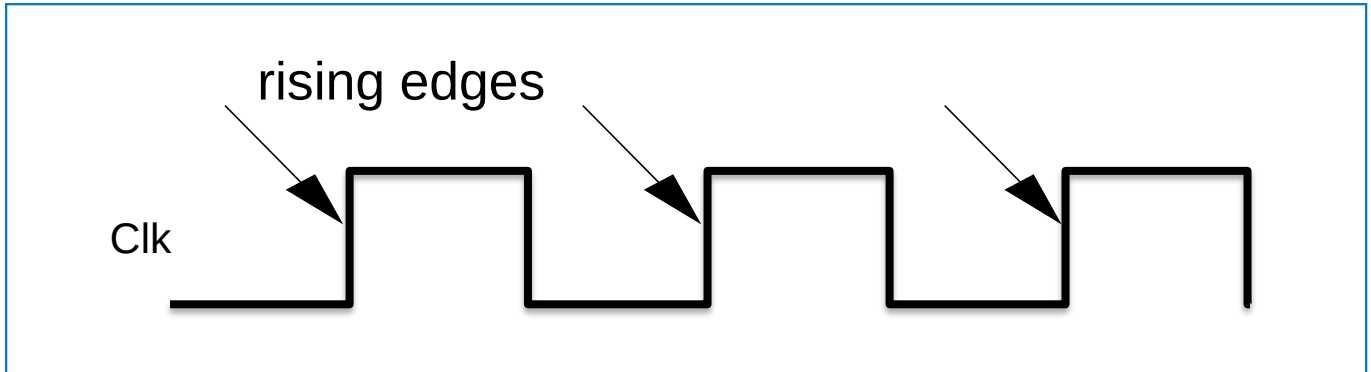


Problem with Level-Sensitive D Latch

- We want do the work: one per clock cycle
  - Independent of length of clock (1 time)



- Can we design bit storage that only stores a value on the rising edge of a clock signal?
  - There is exactly one rising edge per clock cycle
  - There is exactly one falling edge per clock cycle



Make Edge Sensitive Bit Storage

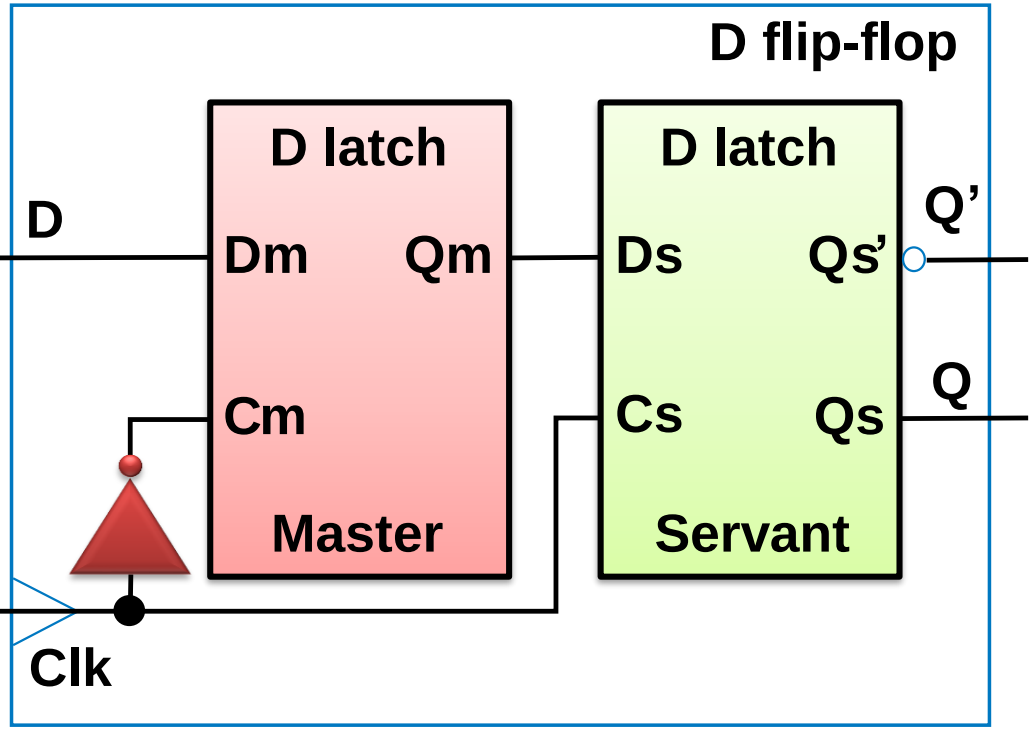
- Latch : Level sensitive storage
- Flip-Flop : Edge sensitive storage
  - Value get changed only at edges of clock

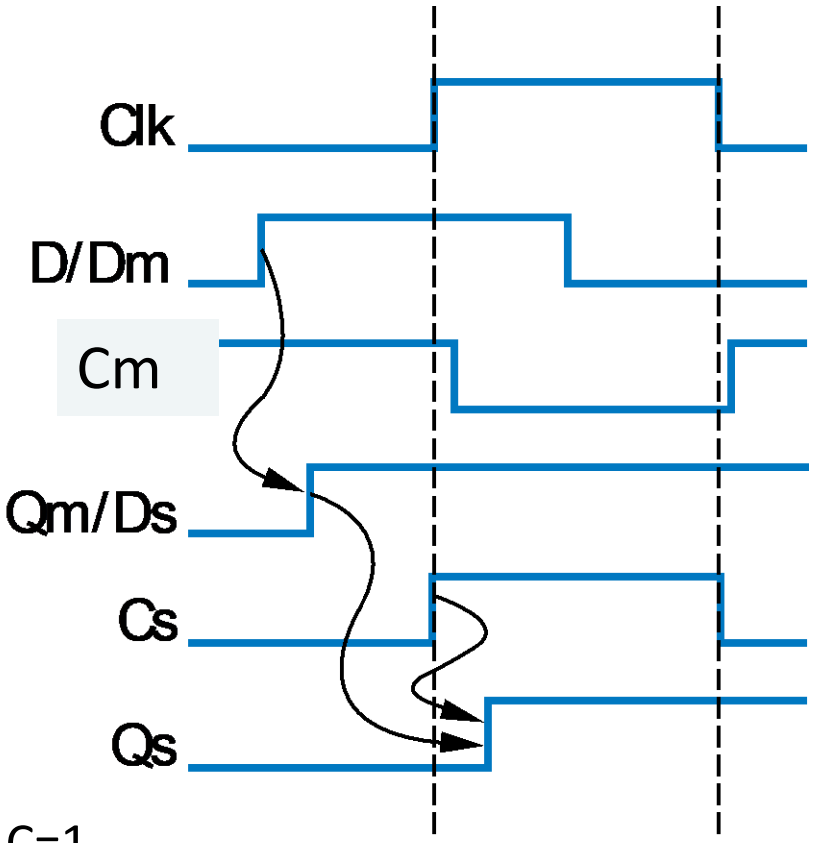
Master -Slave D Flip-Flop

- **Two latches**, output of first goes to input of second, master latch has inverted clock signal
- So master loaded when C=0, then servant when C=1
- When C changes from 0 to 1, master disabled, servant loaded with value that was at D just before C changed
  - i.e., Value at D during rising edge of C

Master -Slave D Flip-Flop

- ***Flip-flop***: stores on clock edge, not level

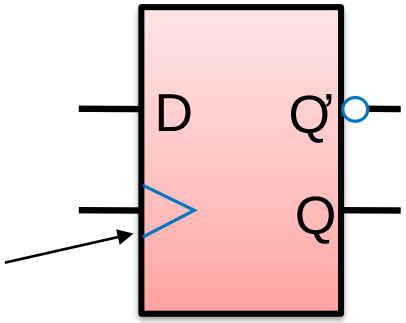




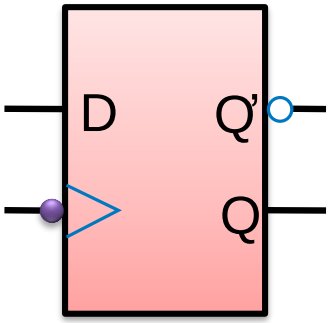
Master loaded when C=0, then servant when C=1
When C changes from 0 to 1, master disabled, servant loaded with value that was at D just before C changed -- i.e., value at D during rising edge of C

D Flip-Flop (Rising & Falling Edges)

The triangle means clock input, edge triggered



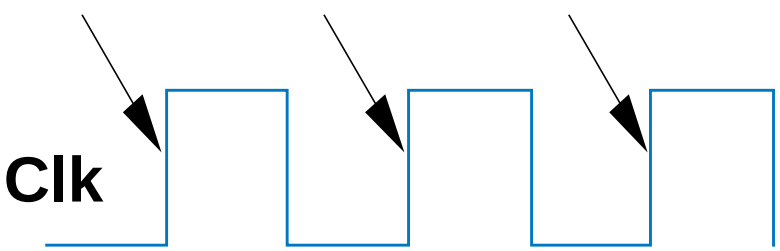
Symbol for rising-edge triggered D flip-flop



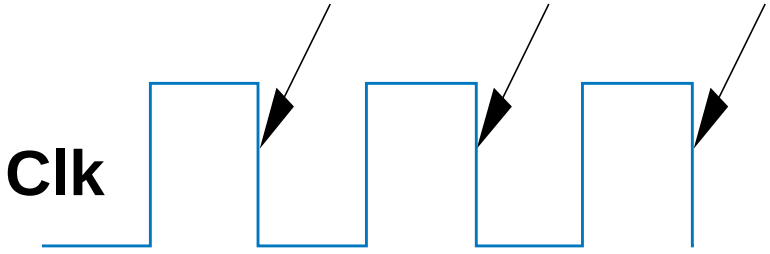
Internal design: Just invert servant clock rather than master

Symbol for falling-edge triggered D flip-flop

Rising edges

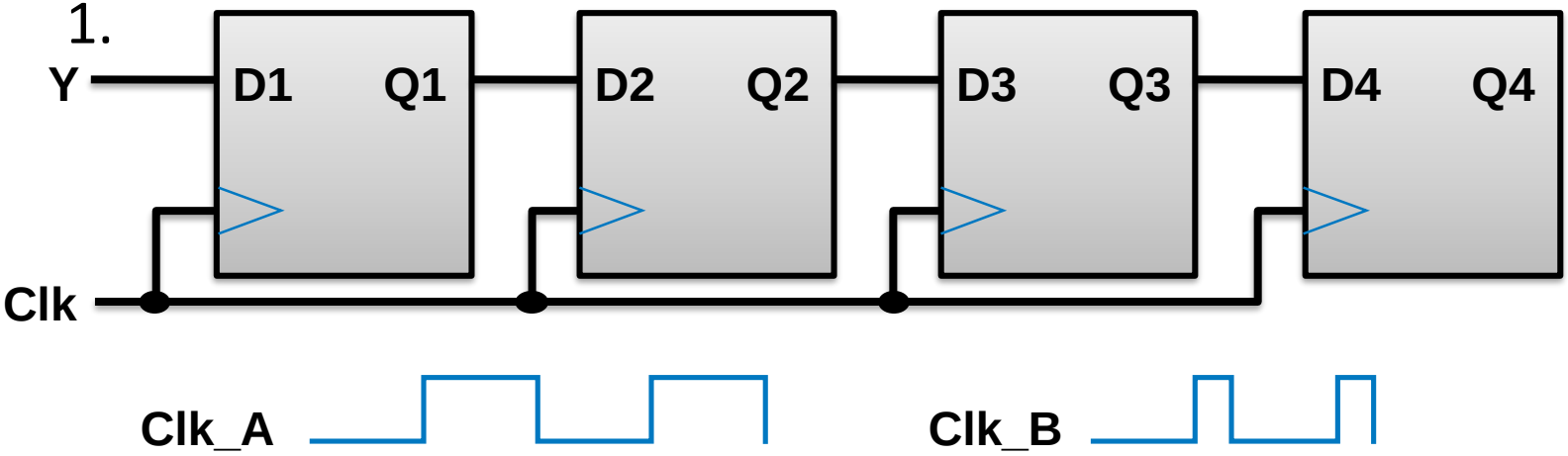


Falling edges



D Flip-Flops

- Solves problem of not knowing through how many latches a signal travels when C=1
  - Signal travels through exactly one FF, for Clk_A or Clk_B. **Why?**
  - Because on rising edge of Clk, all four flip-flops are loaded simultaneously -- then all four no longer pay attention to their input, until the next rising edge. Doesn't matter how long Clk is 1.

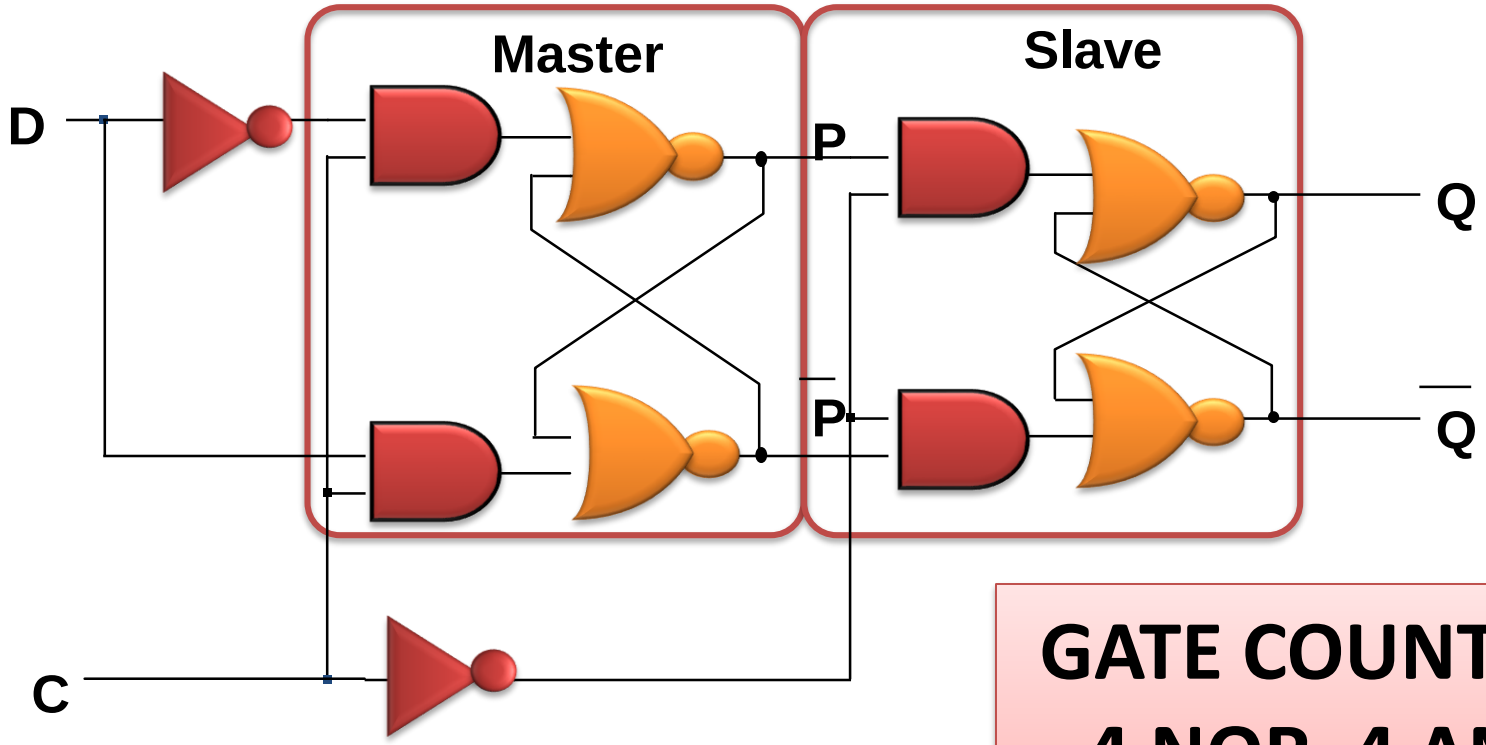


Two latches inside each flip-flop

D Latch vs. D Flip-Flop

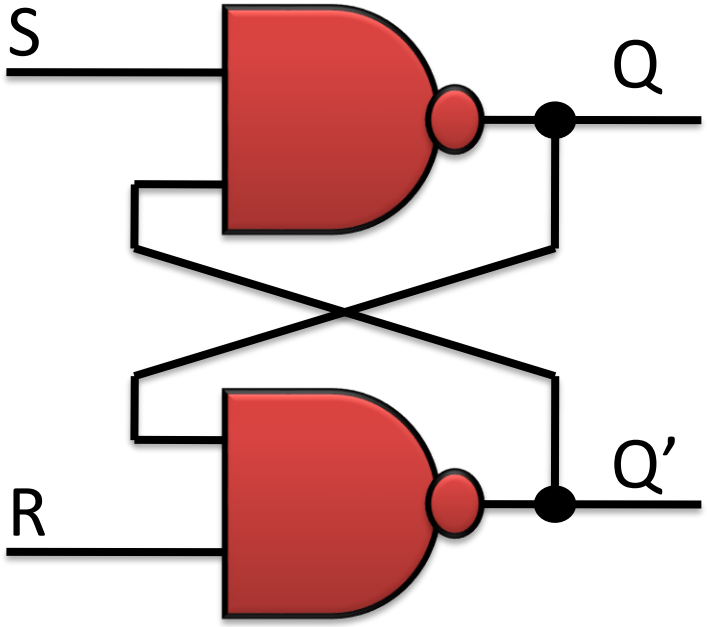
- Latch is level-sensitive: Stores D when C=1
- Flip-flop is edge triggered: Stores D when C changes from 0 to 1
  - Saying “level-sensitive latch,” or “edge-triggered flip-flop,” is redundant
  - Two types of flip-flops -- rising or falling edge triggered.

Positive Edge Triggered D-Flip Flop: Optimization



GATE COUNT: 10
4 NOR, 4 AND and 2 NOT

Remember: SR Latch with NAND Gates



S	R	Q+
0	0	1*0* (Unpredictable)
0	1	1
1	0	0
1	1	Q

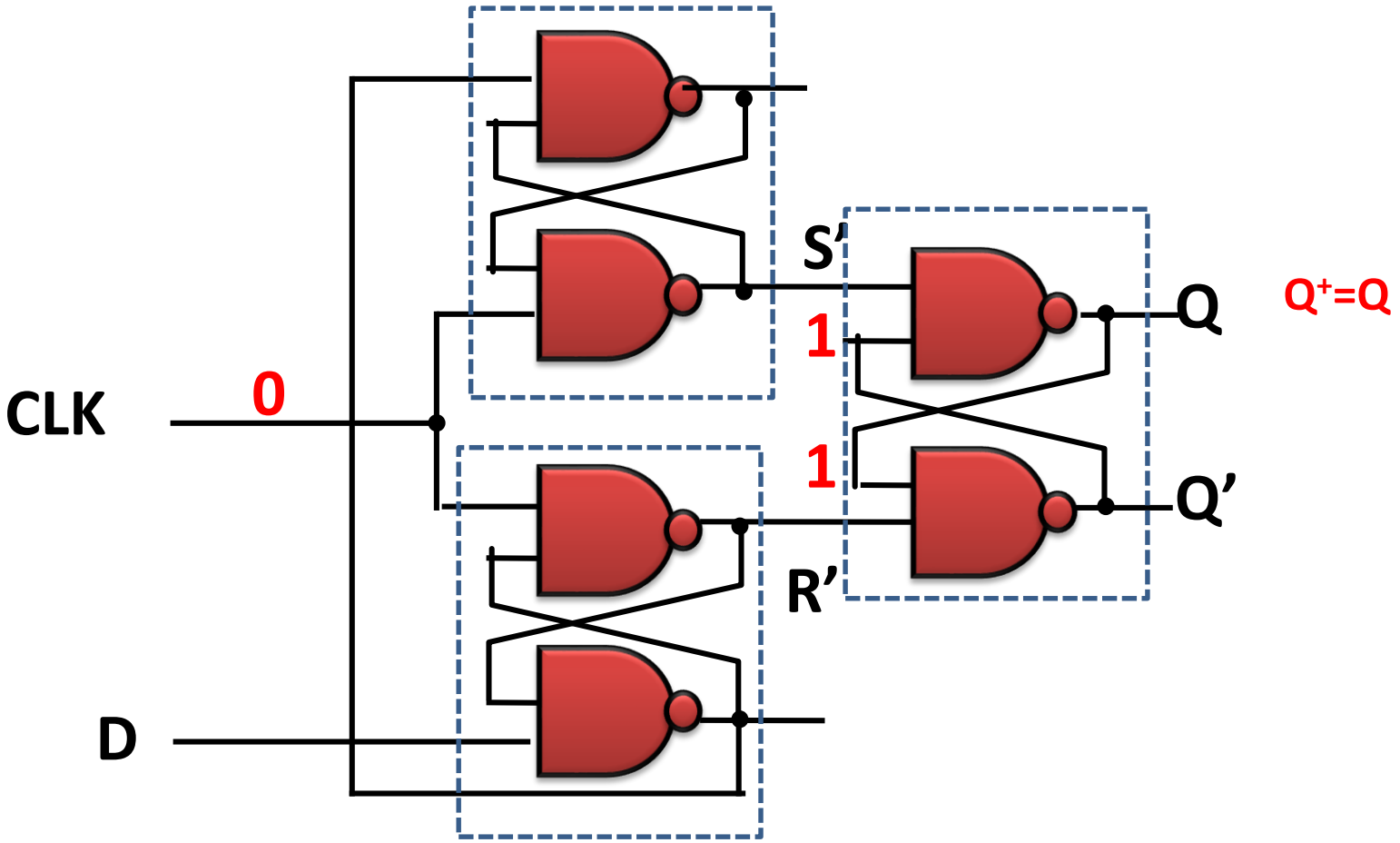
Opposite to SR Latch with NOR Gates

Set will do Q=0 and Reset will Q=1

Ref: ManoBook

Positive-Edge Triggered D-FF: Economical

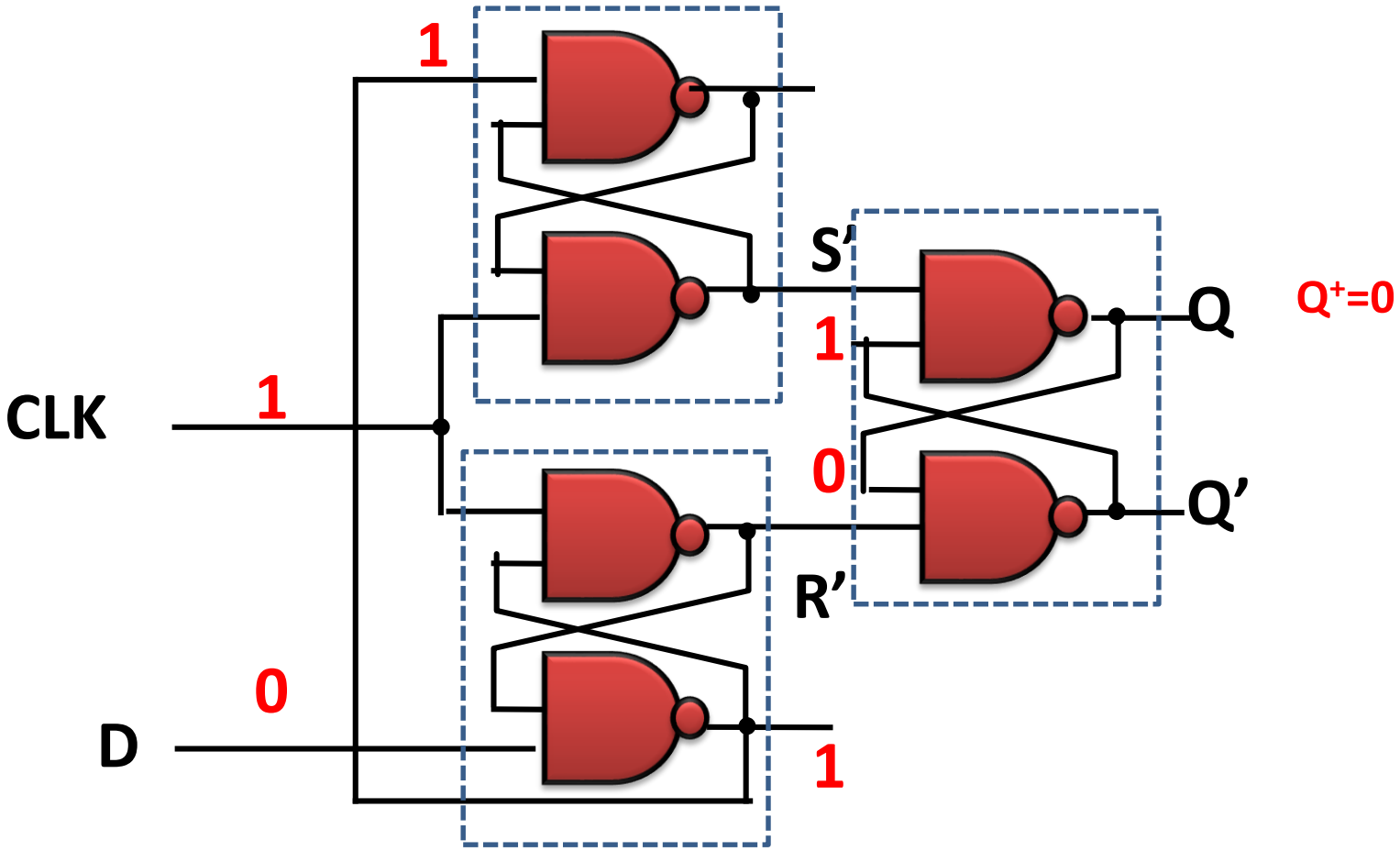
- When CLK=0, S'R'=11, $Q^+=Q$ (Independent of D)



Ref: ManoBook

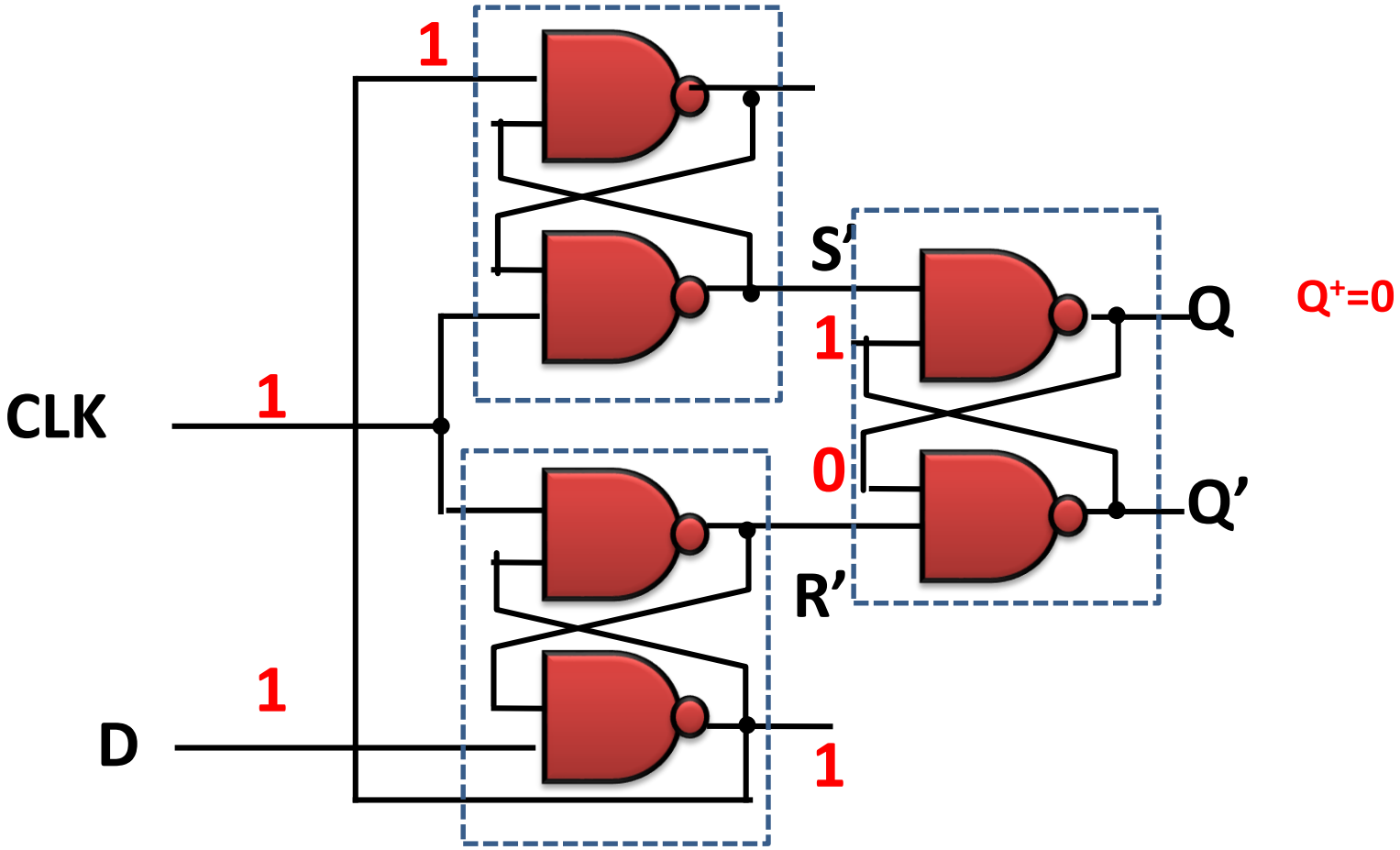
Positive-Edge Triggered D-FF: Economical

- When CLK=1, D=0, Q^+=0

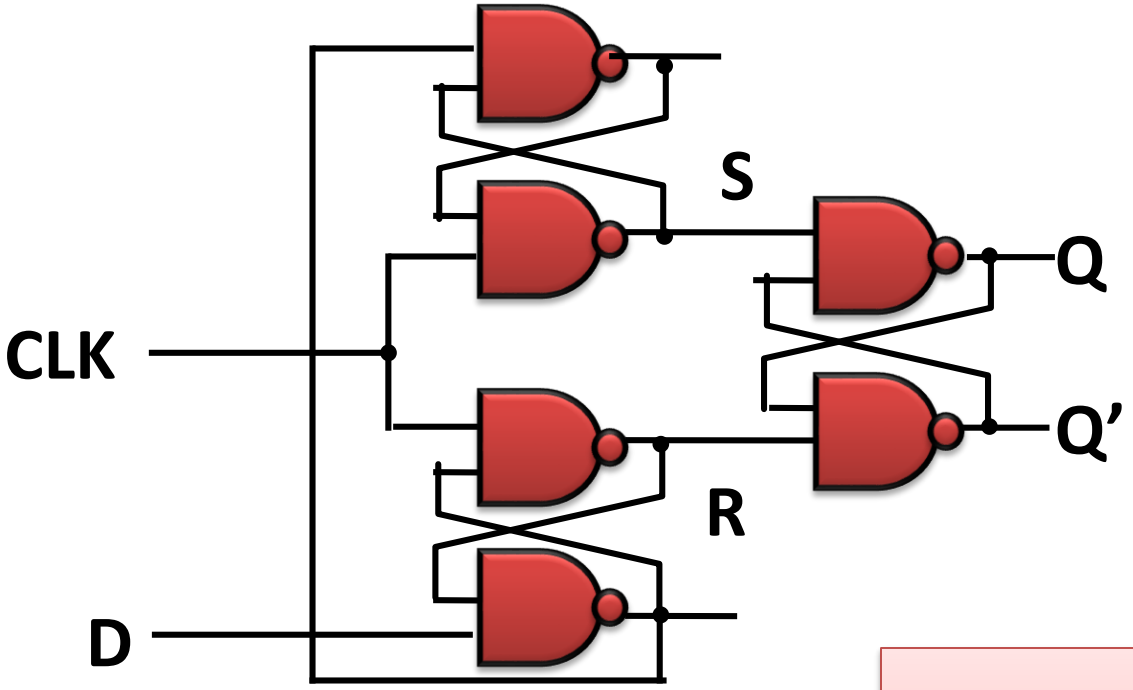


Positive-Edge Triggered D-FF: Economical

- After that When CLK=1, D=1: No changes to R'S': $Q^+=0$ **It locked**



Positive-Egde Trigeered D-FF: Economical



GATE COUNT: 6
6 NAND, Same types

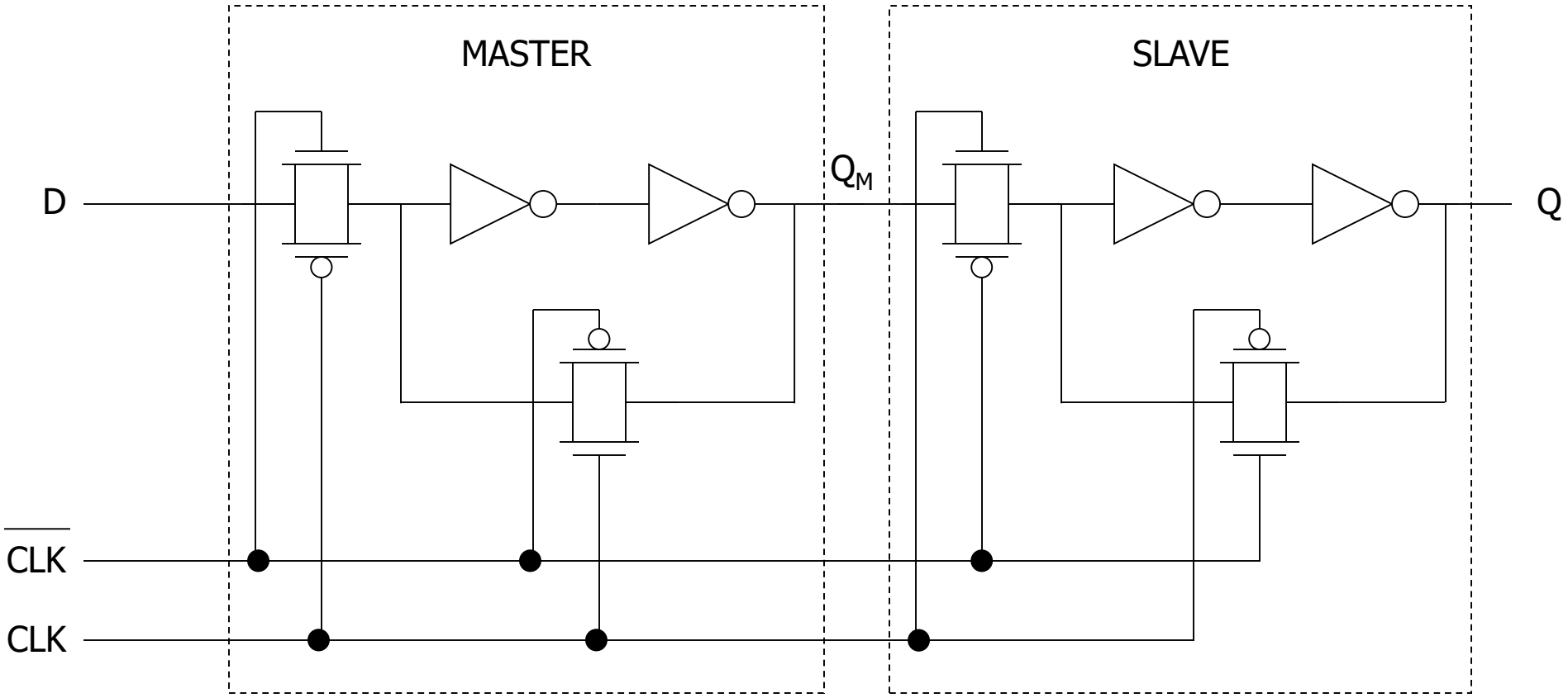
Ref: ManoBook

Transistor level optimization
is
out of syllabus

But showing two slides ☺ ☺ ☺

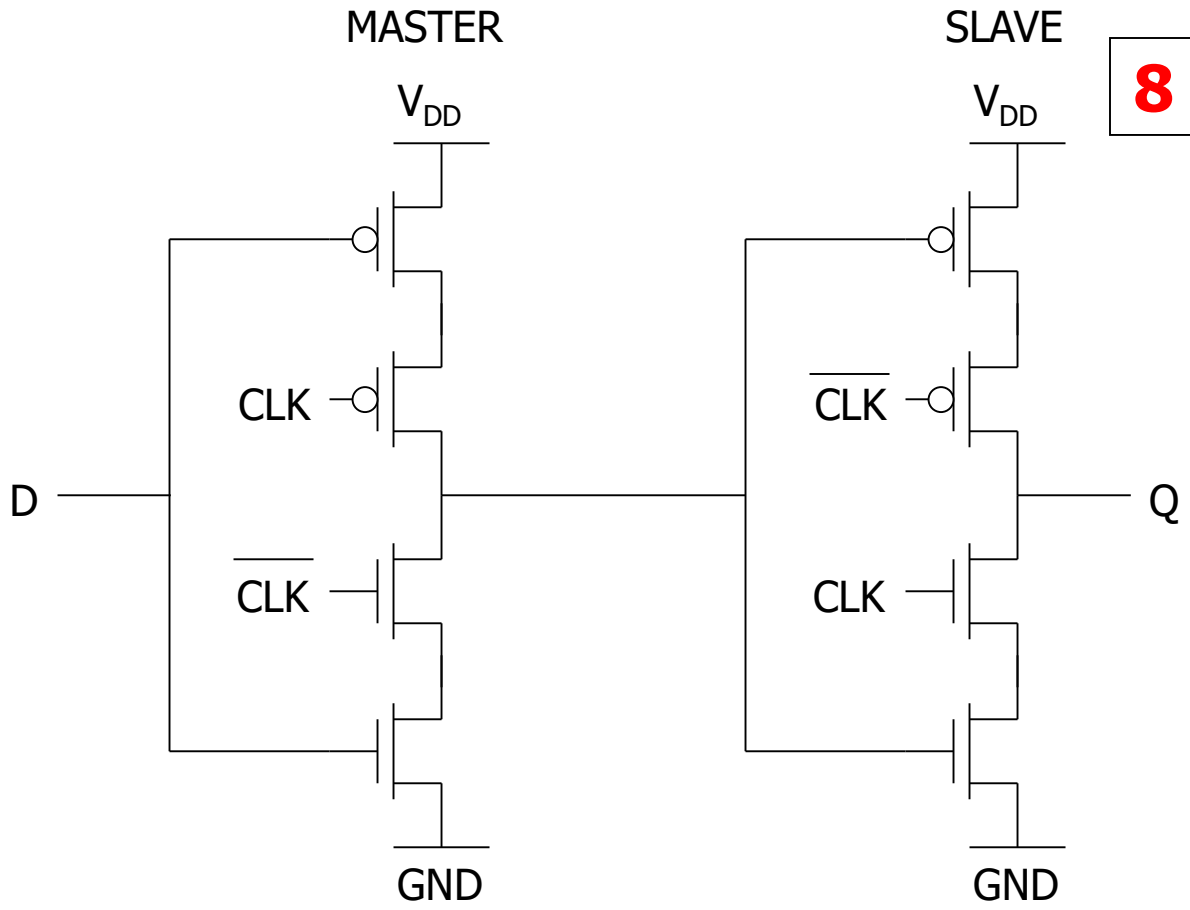
Master-Slave Edge-Triggered Flip-Flop

2 x 8 = 16 Transistors



More Efficient Master-Slave Edge-Triggered Flip-Flop

- Called a C^2MOS (Clocked CMOS) design

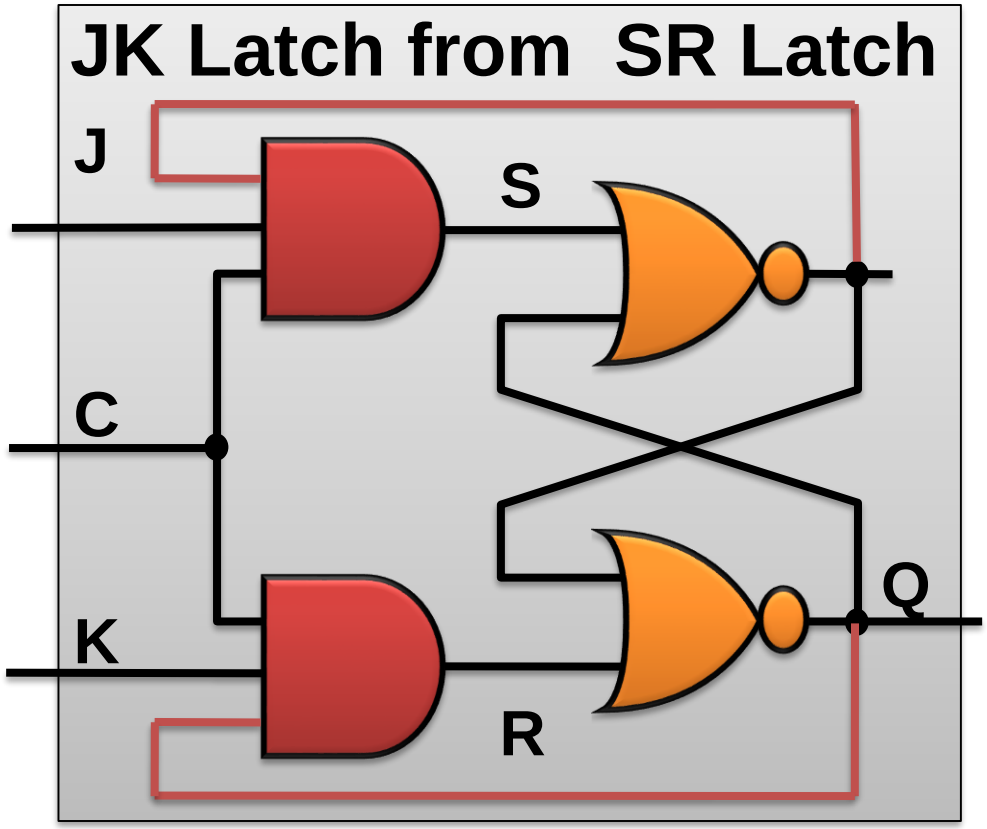


Problem handled in Designing FF

- ✓ OR Gate : worked just like a ringing bell
- ✓ OR gate with Feed back : (Q=1 can never be changed)
- ✓ Two NOR gates with cross coupled out put and input : Solved to store a bit but Race condition
- ✓ Ensure RS=11 will not happed by adding Not and AND gate
- ✓ Stabilize: Enable Signal to put remove : delay of added Ckt
- ✓ RS Latch to D-Latch : Ensure no SR=11
- ✓ Master Slave Latches to make a Flip Flop (or edge triggered Latch)
- ✓ Optimized D-FF (using only NAND Gates)
- ✓ Further optimized lower level using only 8 transitors

J-K Latches

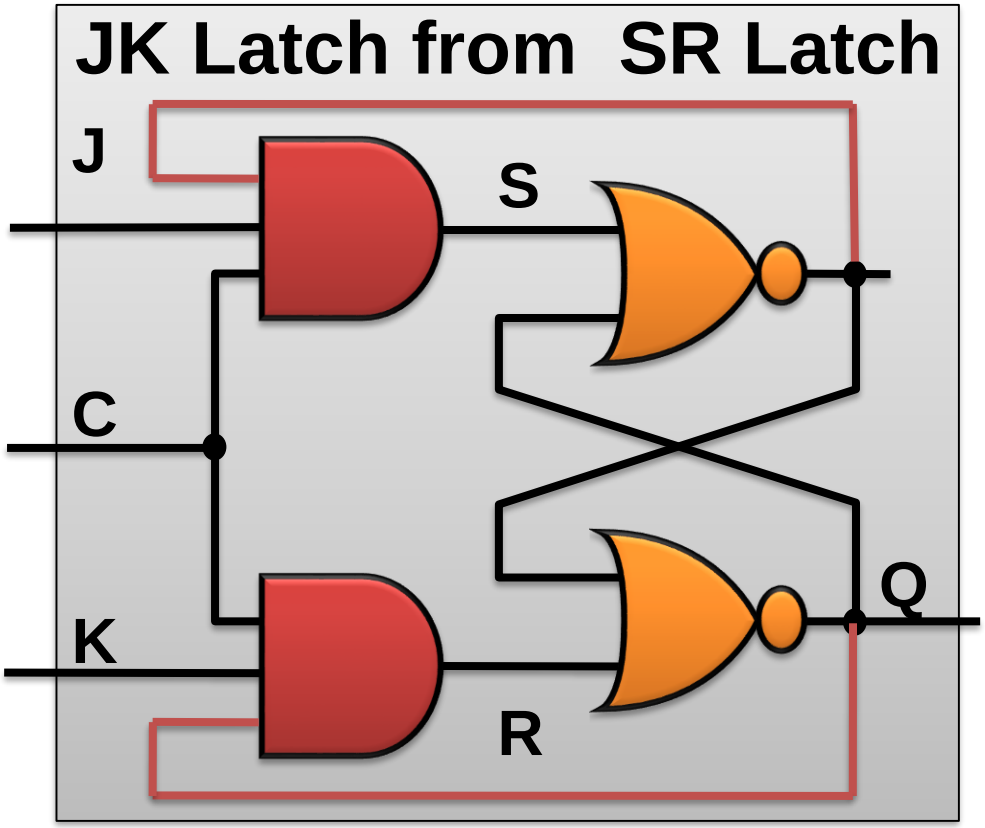
- The JK flip-flop augments the behavior of the SR flip-flop (J=Set, K=Reset) by interpreting the S = R = 1 condition as a "flip" or toggle command.



J	K	Q+
0	**0**	**Qt**
0	1	0
1	0	1
1	**1**	**Qt'**

$Q^{+} = K'Q + JQ'$

J-K Latches: Problem



Q Oscillate between when C=1

C

Q