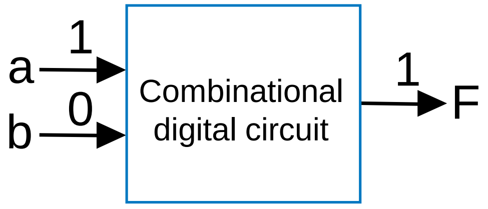
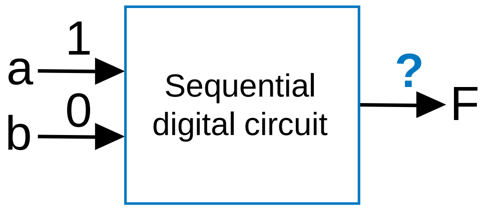
B38DB: Digital Design and Programming
Sequential Logic Design – Flip Fops

Mustafa Suphi Erden

Heriot-Watt University
School of Engineering & Physical Sciences
Electrical, Electronic and Computer Engineering
Room: EM 2.01
Phone: 0131-4514159
E-mail: m.s.erden@hw.ac.uk

Introduction



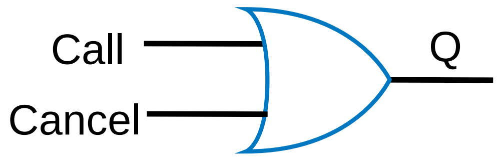


Must know the sequence of past inputs

- Sequential circuit
  - Output depends not only on present inputs, but also on the circuit's present ***state***, which is all the bits currently ***stored*** in the circuit and **depends on the past sequence of inputs**.
- We will:
  - Design a new building block, a **flip-flop**, that stores one bit
  - Combine that block to build multi-bit storage **register**
  - Describe the sequential behavior using a **finite state machine (FSM)**
  - Convert a FSM to a **controller** – a sequential circuit having a register and combinational logic

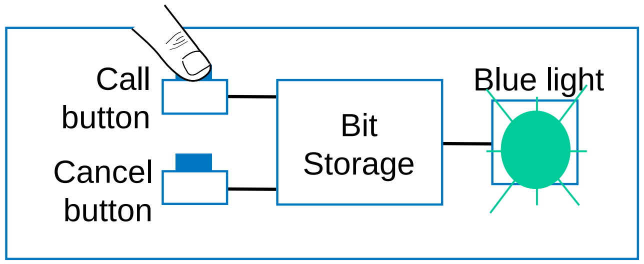
Example: Bit Storage

- Flight attendant call button
  - Press call: light turns on
    - ***Stays on*** after button released
  - Press cancel: light turns off
  - Logic gate circuit to implement this?

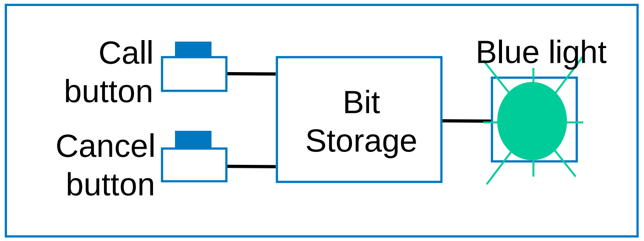


Doesn't work. Q=1 when Call=1, but doesn't stay 1 when Call returns to 0

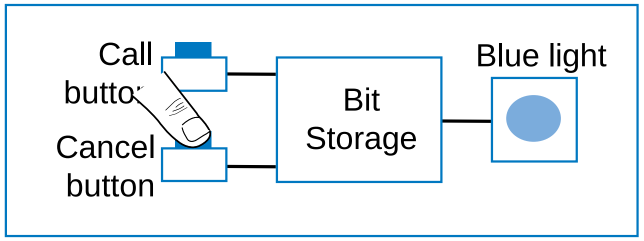
We need some form of "feedback" in the circuit



1. Call button pressed – light turns on



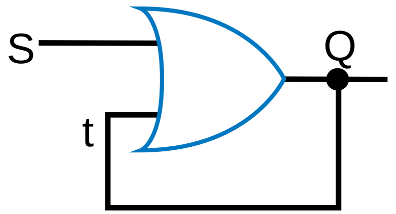
*2. Call button released – light **stays on***

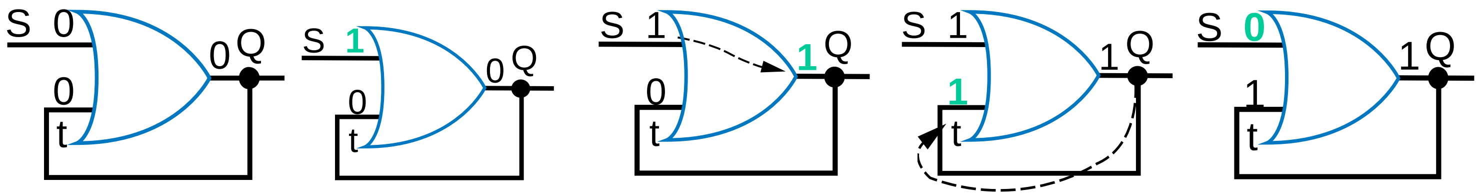


3. Cancel button pressed – light turns off

First Attempt at Bit Storage

- We need some sort of feedback
  - Does the circuit on the right do what we want?

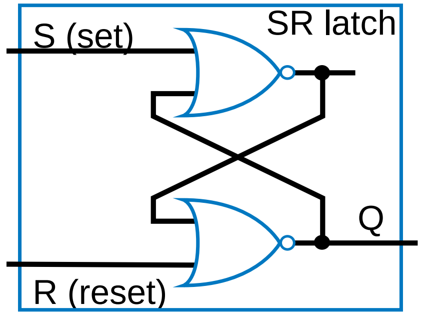


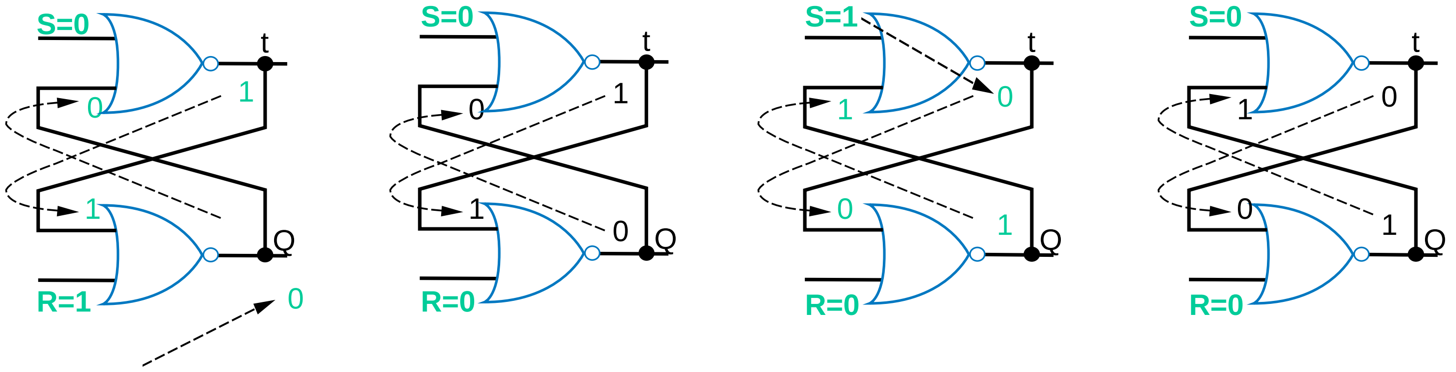


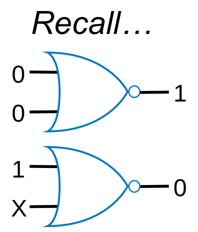
– No: Once Q becomes 1 (when S=1), Q stays 1 forever – no value of S can bring Q back to 0

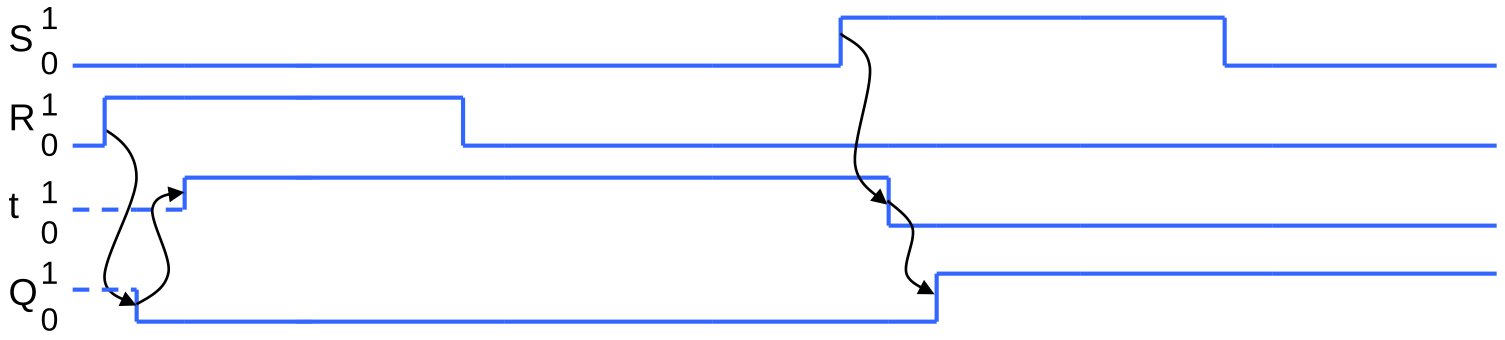
Bit Storage Using an SR Latch

- Does the circuit on the right, with cross-coupled NOR gates, do what we want?
  - Yes!



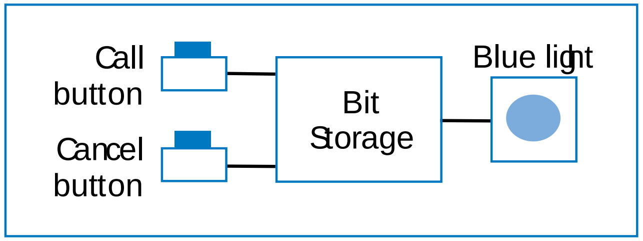




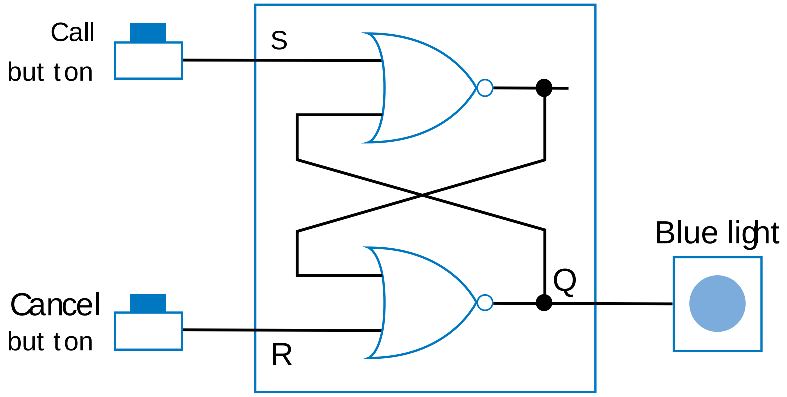


Example: Using SR Latch for Bit Storage

- SR latch can serve as bit storage as in the previous example of flight-attendant call button
  - Call=1 : sets Q to 1
    - Q stays 1 even after Call=0
  - Cancel=1 : resets Q to 0

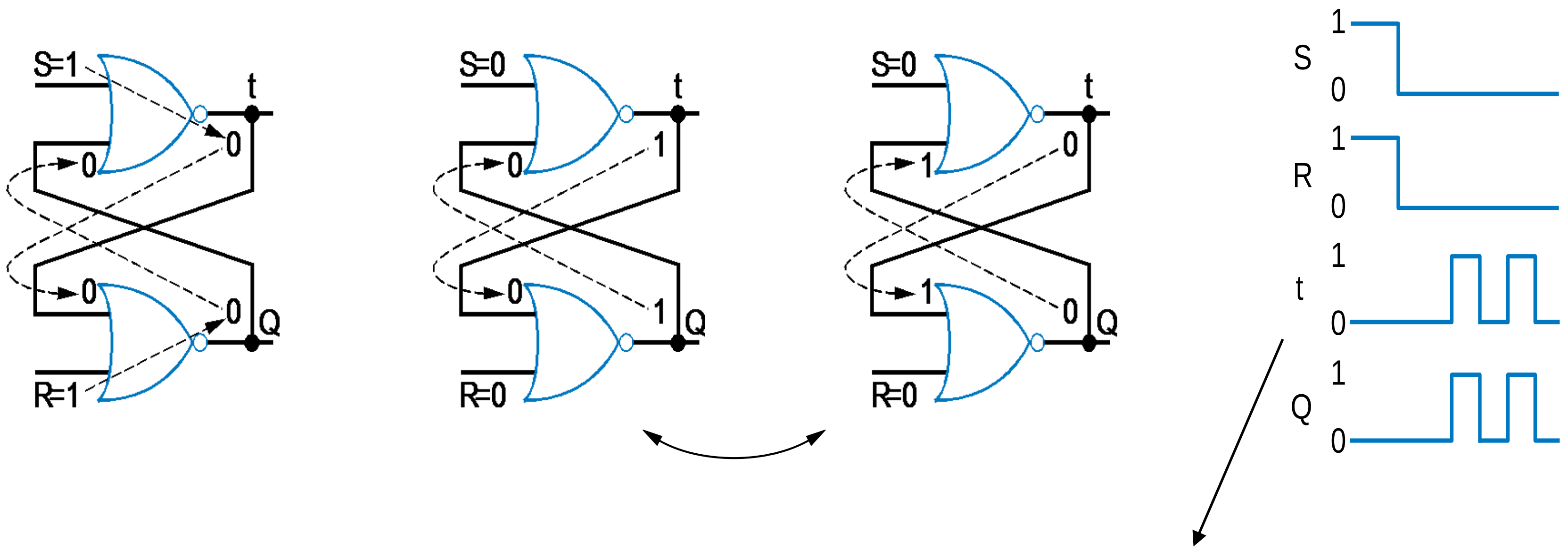


- But, there's a problem...

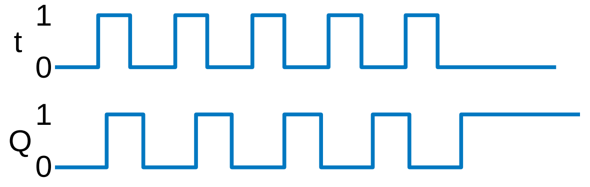


Problem with SR Latch

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

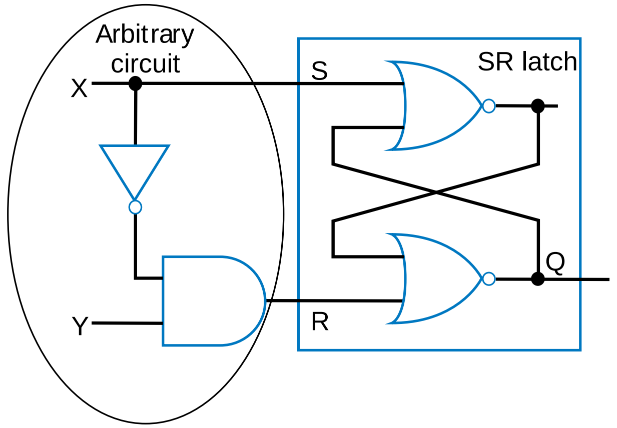


Q may oscillate. Then, because one path will be slightly longer than the other, Q will eventually settle to 1 or 0 – but we don't know which.

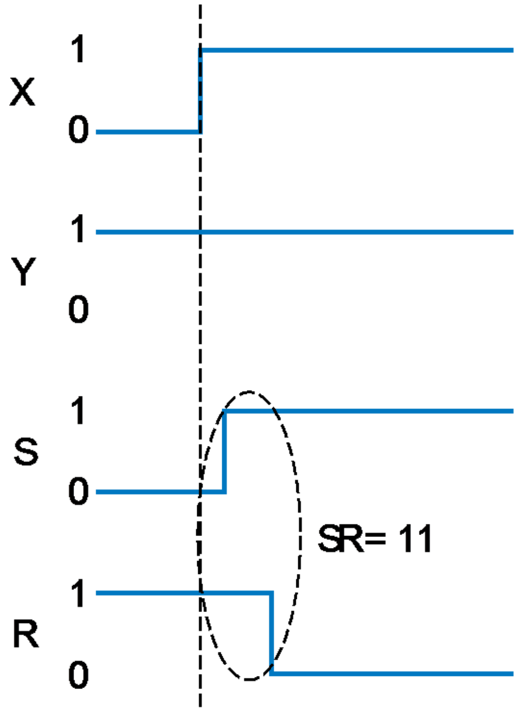


Problem with SR Latch

- This problem can occur even if SR inputs come from a circuit that supposedly never sets S=1 and R=1 at the same time
- It happens due to different delays of different paths

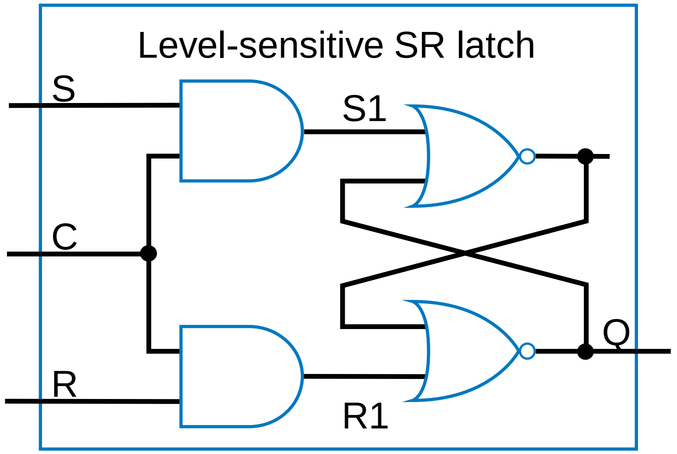


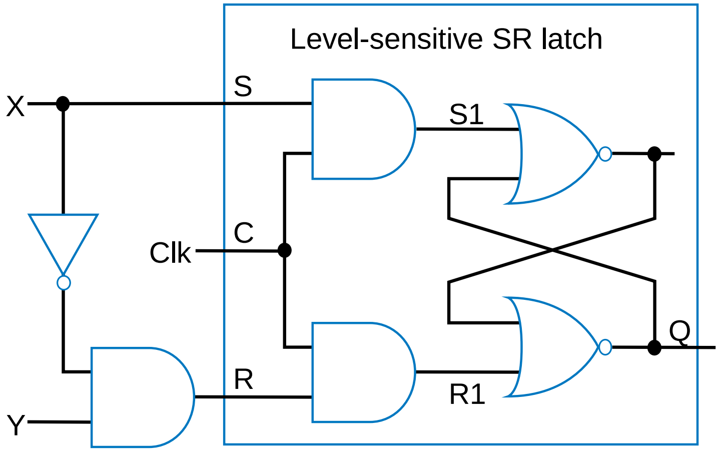
The longer path from X to R than to S might result SR=11 for short time – could be long enough to cause oscillations



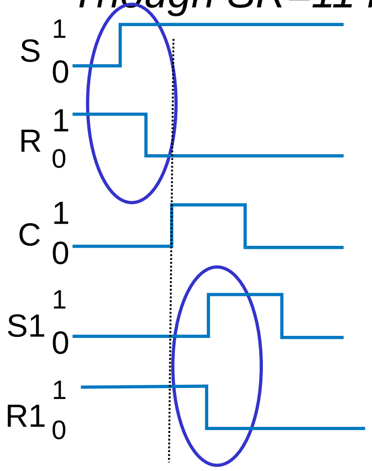
Solution: Level-Sensitive SR Latch

- Add enable input "C" as shown
  - Only let S and R change when C=0
  - Change C to 1 only after sufficient time for S and R to be stable

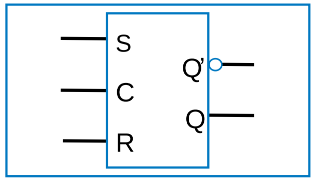




Though SR=11 briefly...



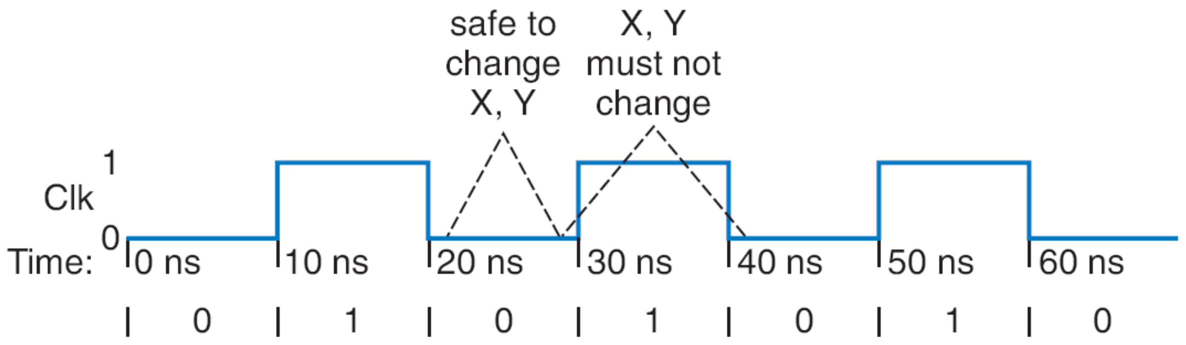
...S1R1 never = 11

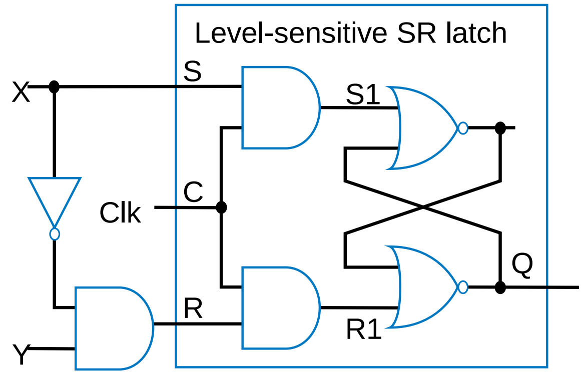


Level-sensitive SR latch symbol

Clock Signals for a Latch

- How to choose the timing to set C=1?
  - Most common solution → make C a pulse signal going up and down
    - C=0: Safe to change X, Y
    - C=1: Must *not* change X, Y
  - ***Clock*** signal – A pulsing signal used to enable latches
  - Sequential circuit storage components all use clock signals:
    - ***synchronous*** circuit → Most common type sequential circuit
    - asynchronous circuits – important topic, but left for advanced courses

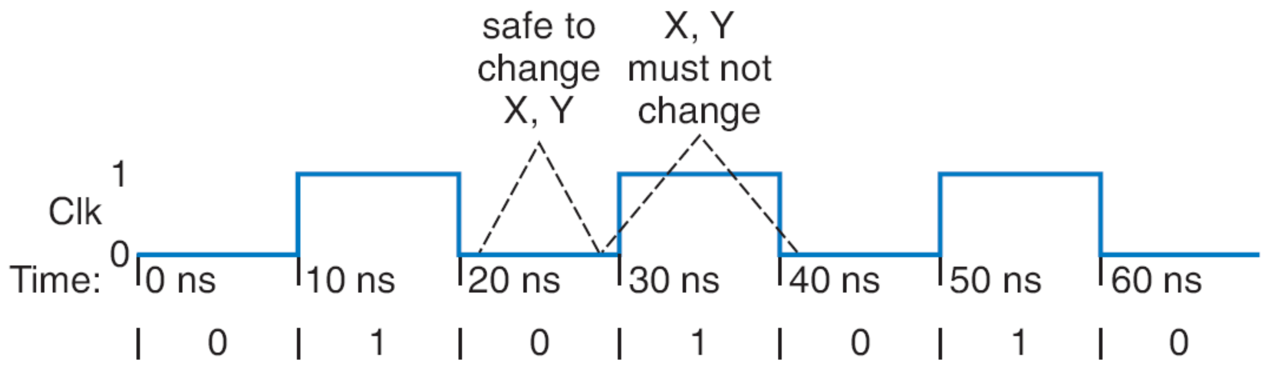




Clocks

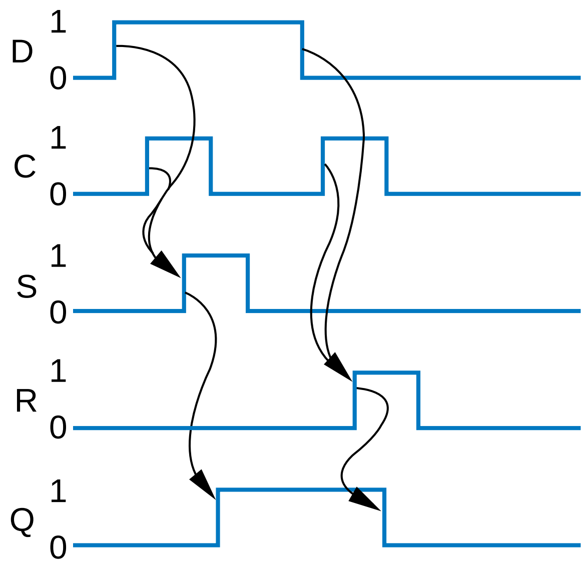
- ***Clock period***: time interval between pulses
  - The signal below: period = 20 ns
- ***Clock cycle***: one such time interval
  - The signal below shows 3.5 clock cycles
- ***Clock frequency***: 1/period
  - The signal below: frequency = 1 / 20 ns = 50 MHz

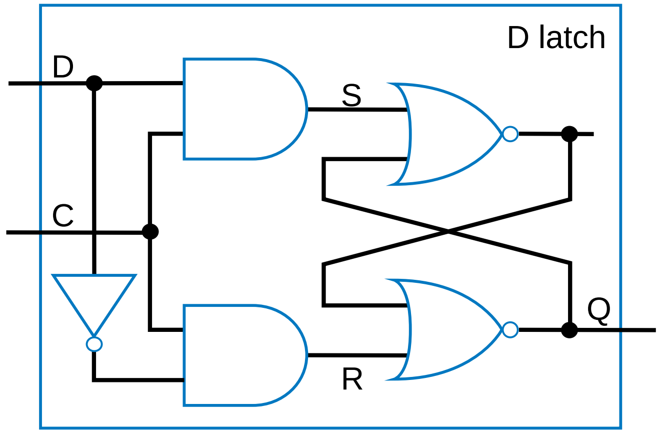
Freq	Period
100 GHz	0.01 ns
10 GHz	0.1 ns
1 GHz	1 ns
100 MHz	10 ns
10 MHz	100 ns

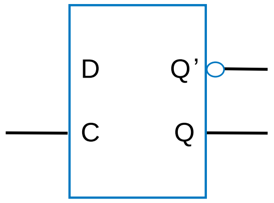


Level-Sensitive D Latch

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



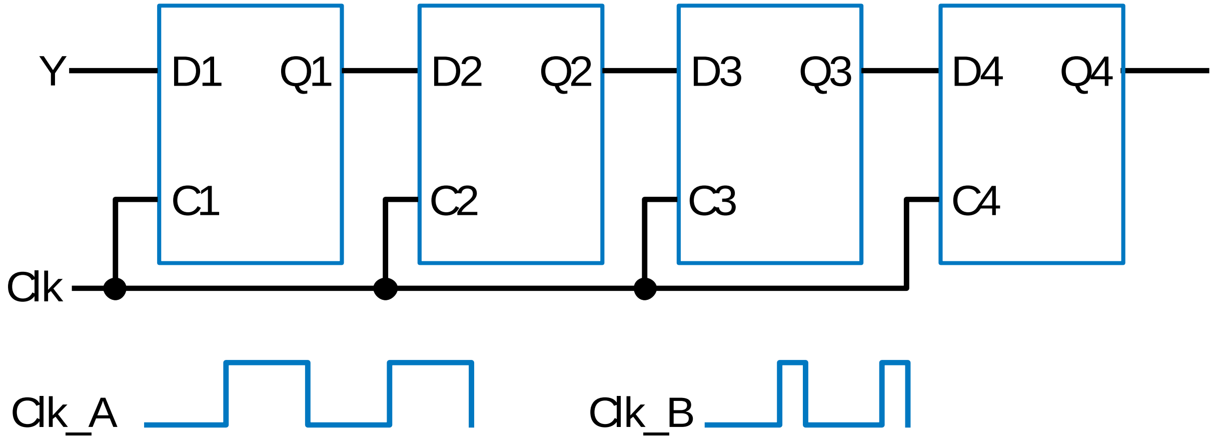


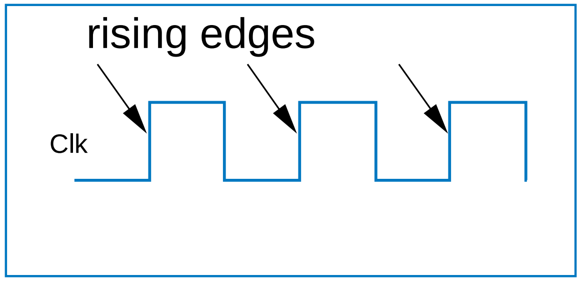


D latch symbol

Problem with Level-Sensitive D Latch

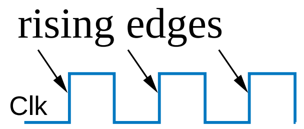
- D latch still has problem (as does SR latch)
  - When C=1, through how many latches will a signal travel?
  - Depends on for how long C=1 holds
    - Clk_A - signal may travel through multiple latches
    - Clk_B - signal may travel through fewer latches
  - Can we design a bit storage that stores a value only on the **rising edge** of a clock signal?



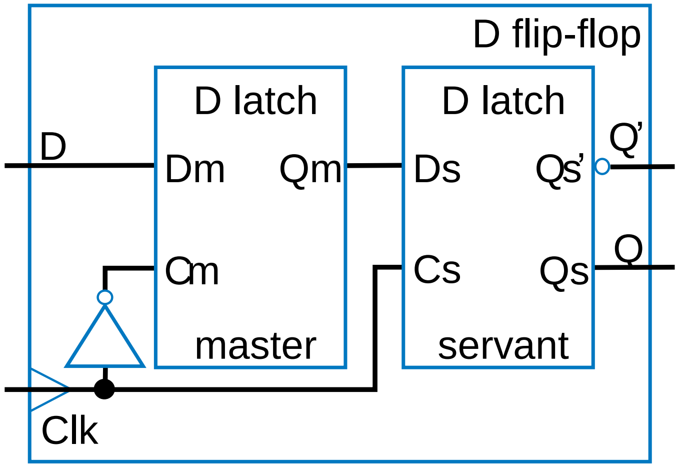


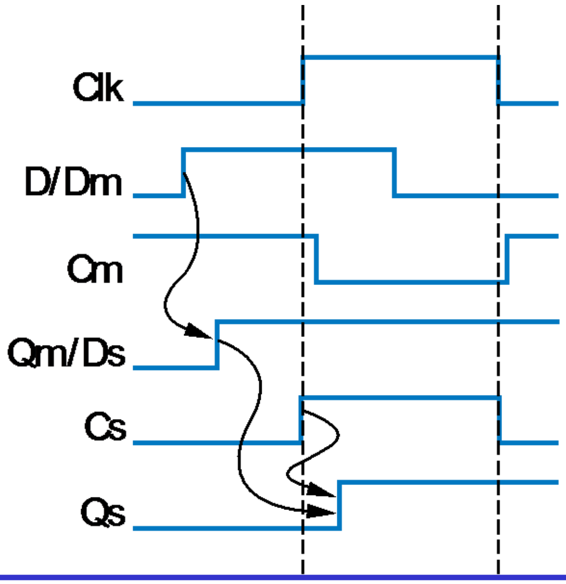
D Flip-Flop (1/3)

- ***Flip-flop***: Bit storage that stores on clock edge, not level
- "Master-servant" design
  - Two latches → output of the first goes to the input of the second, with the first latch having inverted clock signal
  - "Master latch" loaded when C=0, servant loaded when C=1
  - When C changes from 0 to 1, master is disabled, servant is loaded with the value that was at D just before C changed – i.e., with the value at D during the rising of C



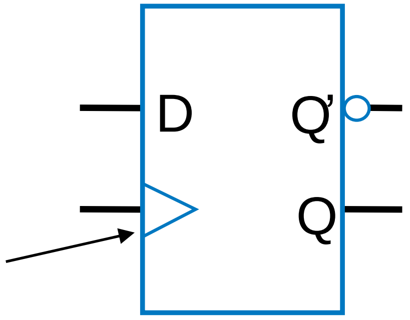
Note: Hundreds of different flip-flop designs exist



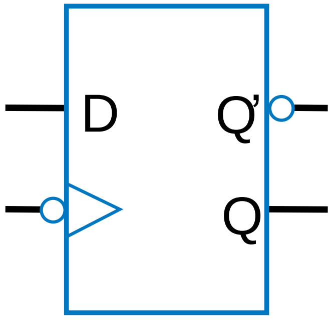


D Flip-Flop (2/3)

The triangle means clock input, edge triggered

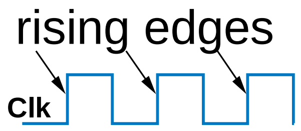


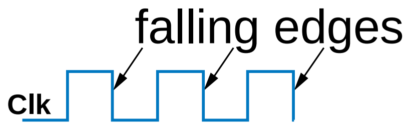
Symbol for rising-edge triggered D flip-flop



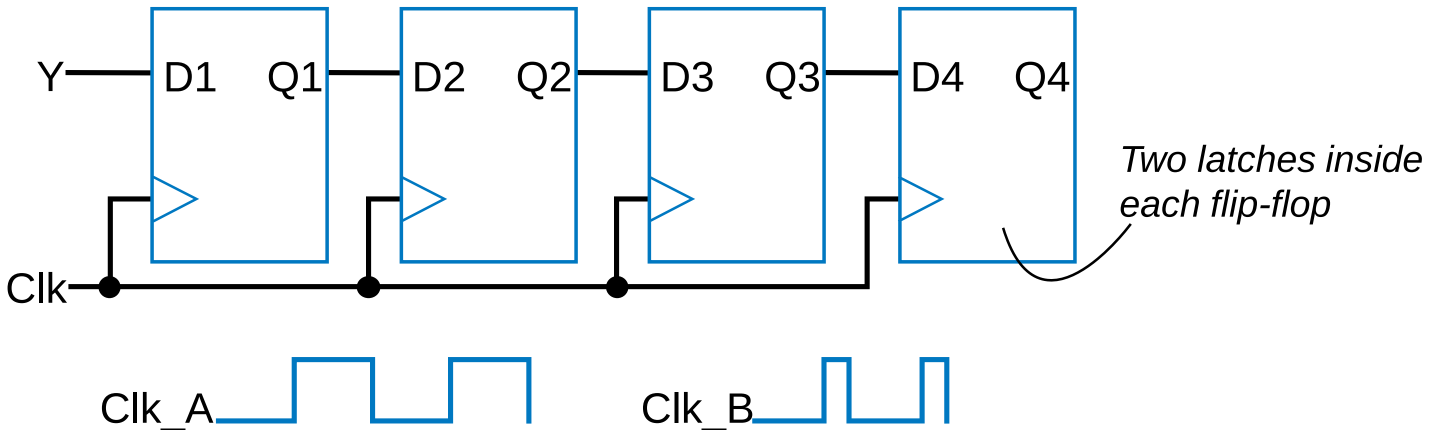
Symbol for falling-edge triggered D flip-flop

Internal design: Just invert servant clock rather than master



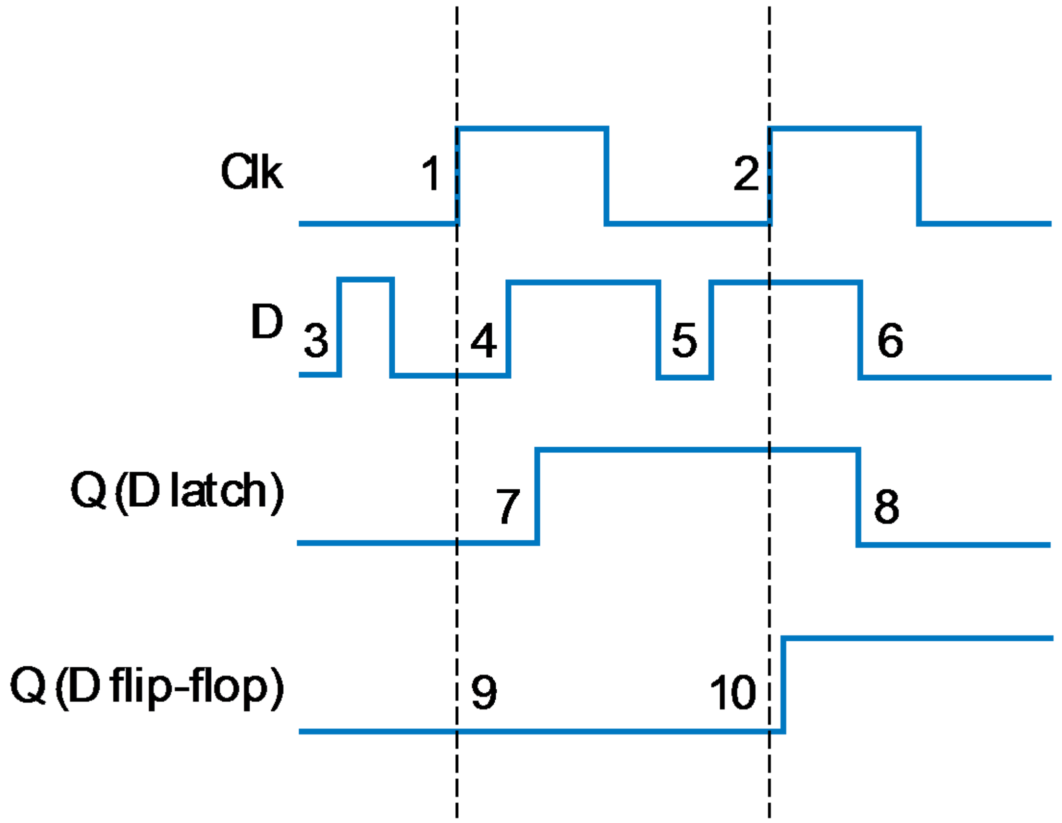


D Flip-Flop (3/3)



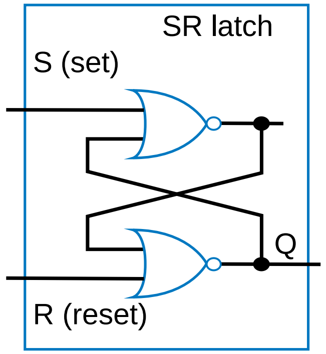
- The signal travels **through exactly one flip-flop**, for Clk_A or Clk_B
- On rising edge of Clk, all four flip-flops are loaded simultaneously and then they do not pay attention to their input, until the next rising edge.
- Doesn’t matter how long Clk signal stays at level 1.

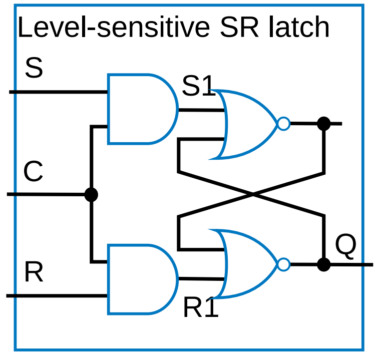
D Latch vs. D Flip-Flop

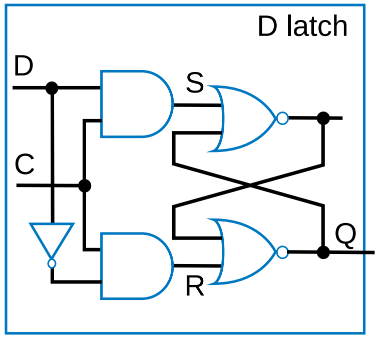


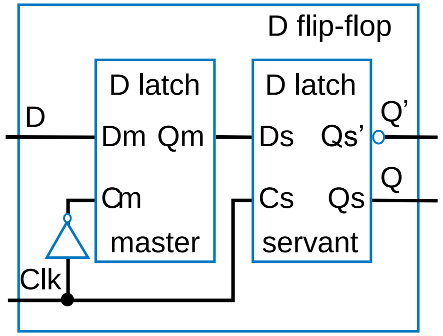
- **Latch** is level-sensitive: Stores D when C=1
- **Flip-flop** is edge triggered: Stores D when C changes from 0 to 1

Bit Storage Summary









Feature: S=1 sets Q to 1, R=1 resets Q to 0. Problem: SR=11 yield undefined Q.

Feature: S and R only have effect when C=1. We can design outside circuit so SR=11 never happens when C=1. Problem: avoiding SR=11 can be a burden.

Feature: SR can't be 11 if D is stable before and while C=1, and will be 11 for only a brief glitch even if D changes while C=1. Problem: C=1 too long propagates new values through too many latches: too short may not enable a store.

Feature: Only loads D value present at rising clock edge, so values can't propagate to other flip-flops during same clock cycle. Tradeoff: uses more gates internally than D latch, and requires more external gates than SR – but gate count is less of an issue today.

- We considered increasingly better bit storage until we arrived at the robust D flip-flop bit storage