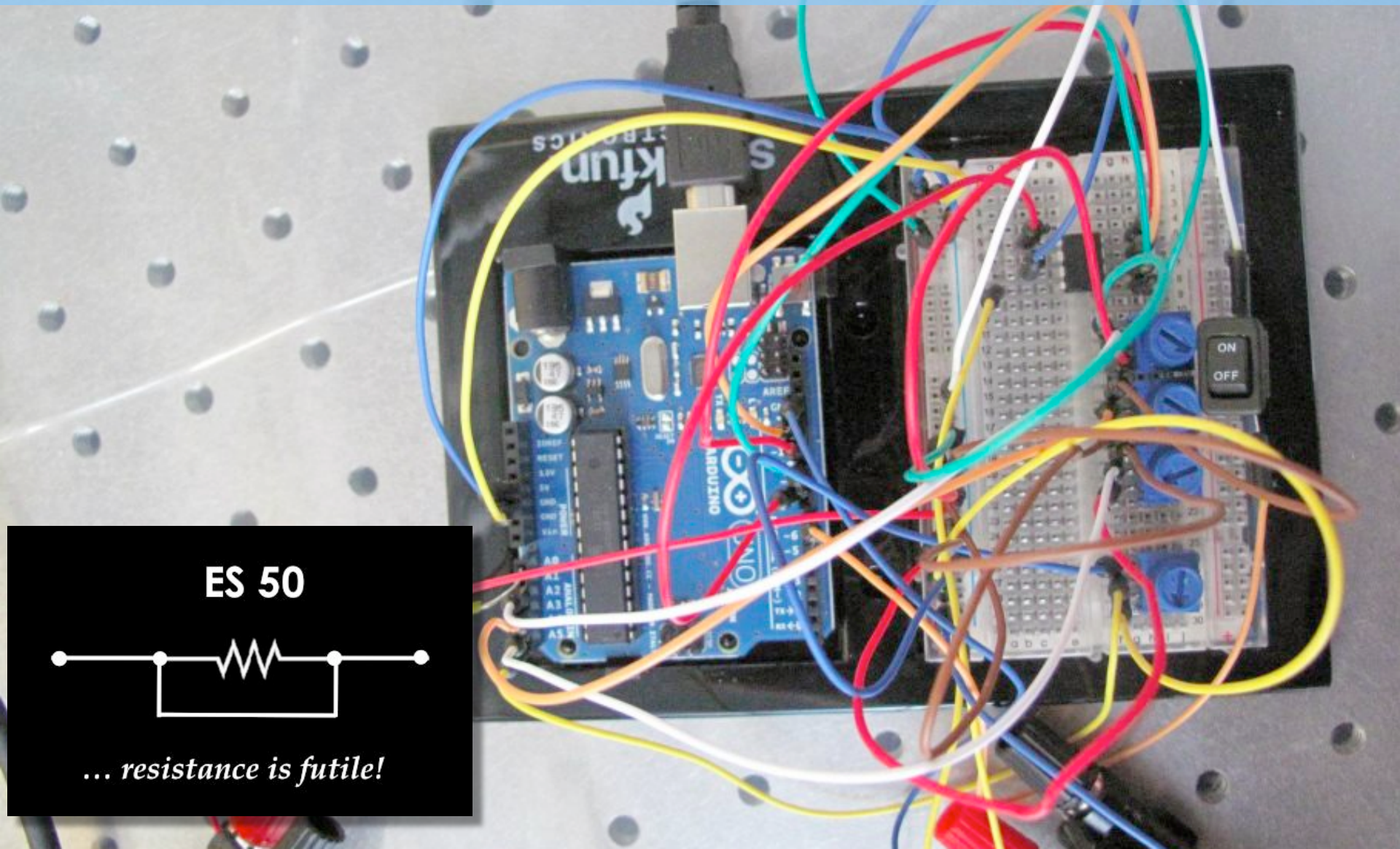
ES 50: Introduction to Electrical Eng.

Explore EE, satisfy a Gen Ed requirement & have fun



ES 50

… resistance is futile!

Lecture 12: Sequential Logic

October 26, 2015

HARVARD
School of Engineering and Applied Sciences

Upcoming Events

Lab 7 - Project Proposal

- Not a typical lab, please choose your final project team (3-4 people)
- Fill out this google sheet with project ideas: http://goo.gl/forms/UGPHnBxsOZ

Class Wed Oct 28

- Guest Speaker, Katie Cagen '14, Microsoft Hardware
- Former EE student and ES 50 TF

Sqeuential Logic

Doesn't only depend on current input values

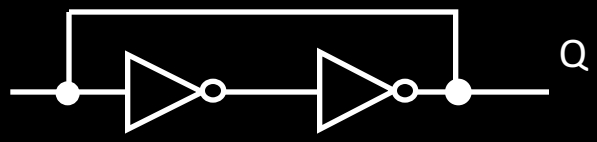
- Depends on the past input values which are used to determine the output value
- Memory & Counting elements of a computer

What do we want from a memory element?

- Ability to retain/remember a single bit with possible values (state)
  - 0 or 1: This allows us to read a previous stored value
- Ability to change the value (state). Since there's only a single bit, there are only two choices:
  - Set the bit to 1
  - Reset, or clear, the bit to 0

We rely on concept of feedback, output is looped back to the input

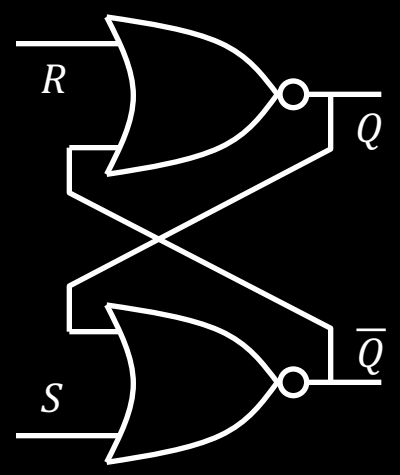
- Example: In circuit below, if Q is 1 it will always be 1, if it is 0 it will always be 0; But, how do we change the state?



SR Latch

Made out of two cross-coupled NOR gates

- The "SR" stands for set/reset
  - S & R inputs are used to change the "state" of the circuit
- To figure out how Q & $\overline{Q}$ change, we have to look at
  - The inputs S & R
  - The current values of Q & $\overline{Q}$



SR Latch - basic memory element

Logic Equations

- $Q_{out} = (R + \overline{Q_{in}})'$ & $\overline{Q_{out}} = (S + Q_{in})'$

If $R = 0$ & $S = 0$

- $Q_{out} = Q_{in}$ & $\overline{Q_{out}} = \overline{Q_{in}}$
- So the latch does not change the state

If $R = 0$ & $S = 1$

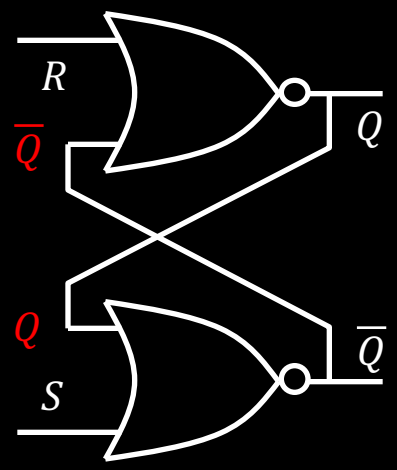
- $\overline{Q_{out}} = 0$ so $Q_{out} = 1$, we are SETTING $Q = 1$

If $R = 1$ & $S = 0$

- $Q_{out} = 0$ so $\overline{Q_{out}} = 1$, we are RESETTING $Q = 0$

If $R = 1$ & $S = 1$

- $Q_{out} = 0$ but $\overline{Q_{out}} = 0????$
- This is logically inconsistent!



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

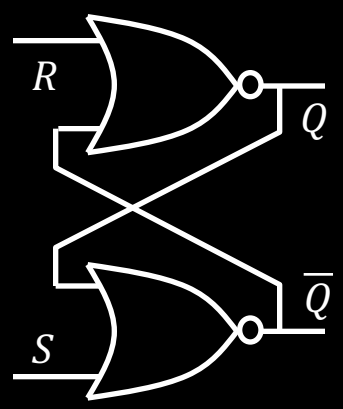
S	Q	Q'
0	0	1
0	1	0
1	0	0
1	1	0

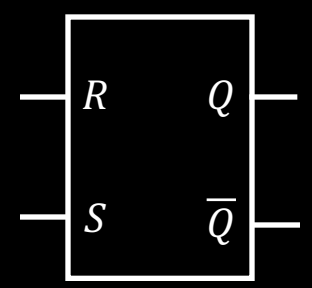
SR Latch - Truth Table

S	R	Present State Q(t)	Next State Q(t+1)
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	X (undefined)
1	1	1	X (undefined)

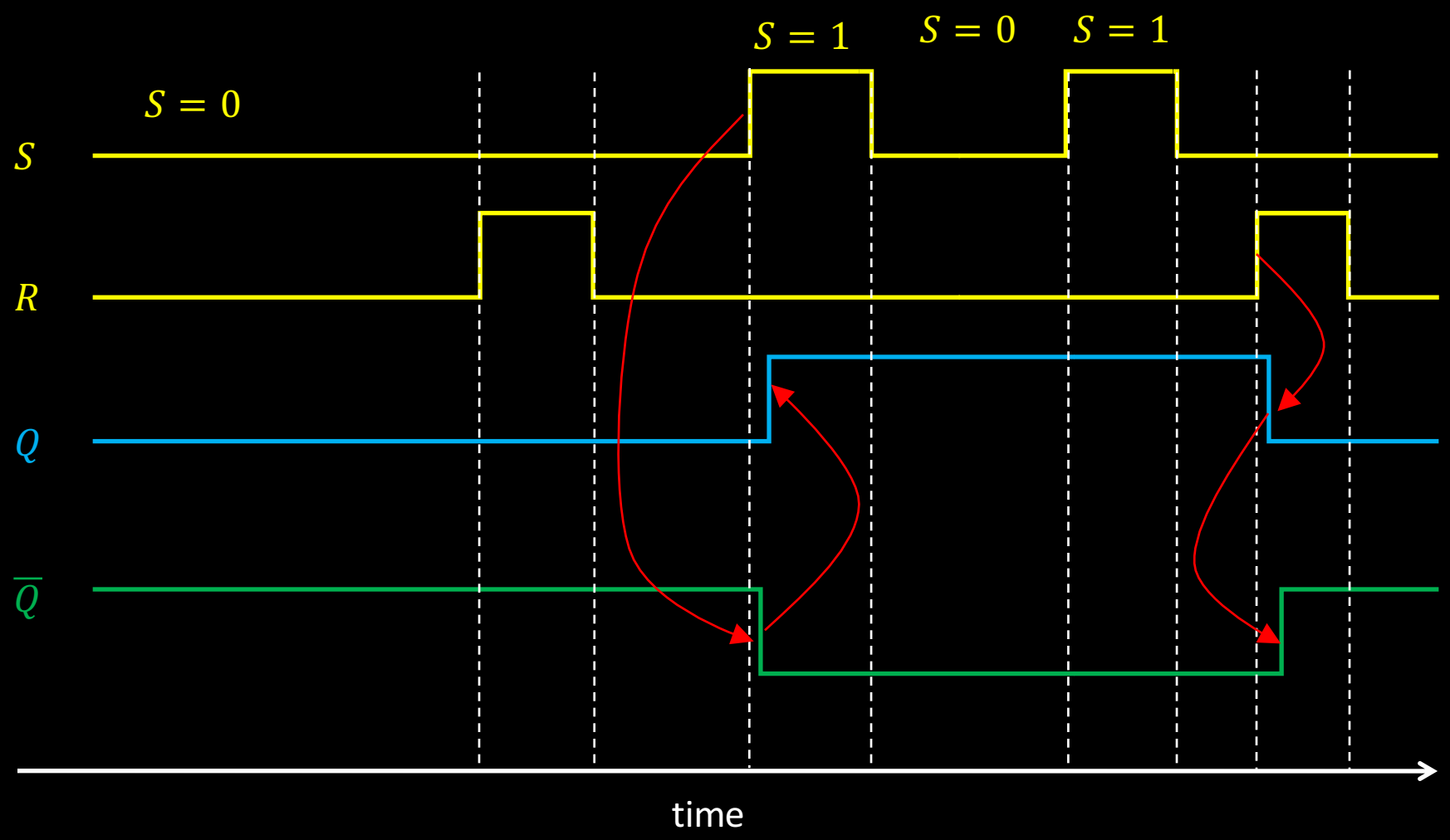
S	R	Q(t+1)
0	0	Q(t) no change
0	1	0
1	0	1
1	1	undefined

$Q(t)$ is current (present) state, & $Q(t+1)$ is next state

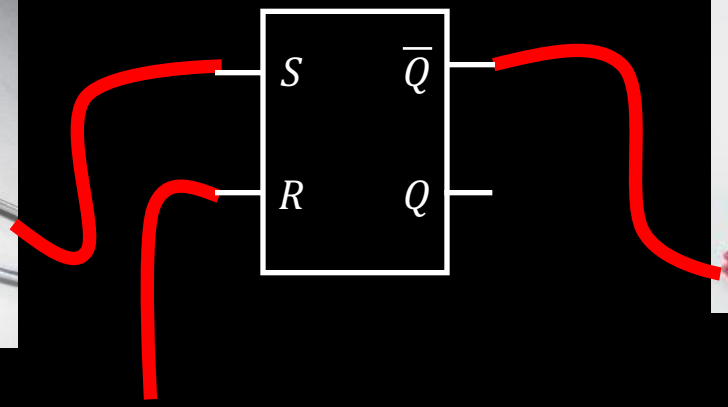


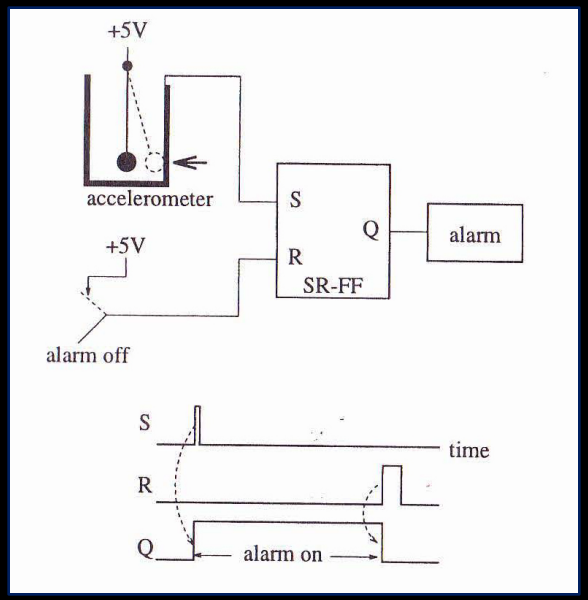


Timing Diagram - Example

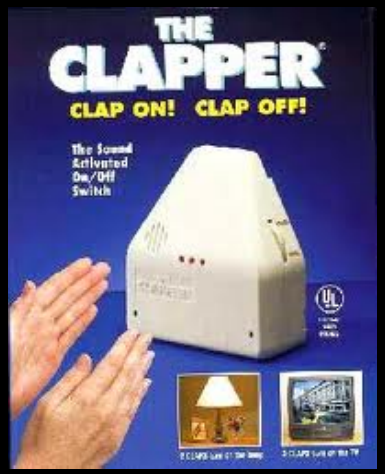


Application: Alarm, Buzzer, “clapper”etc

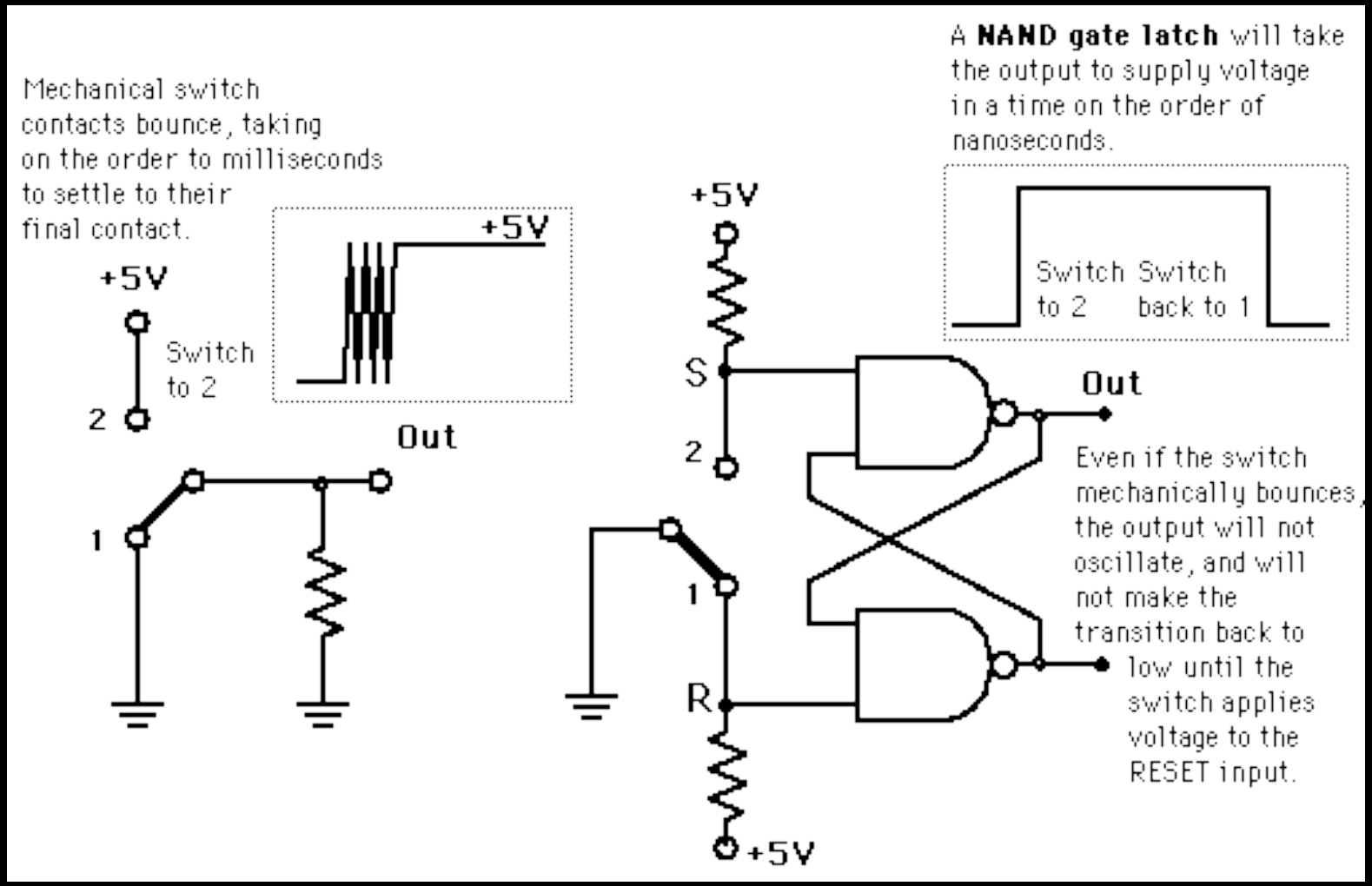


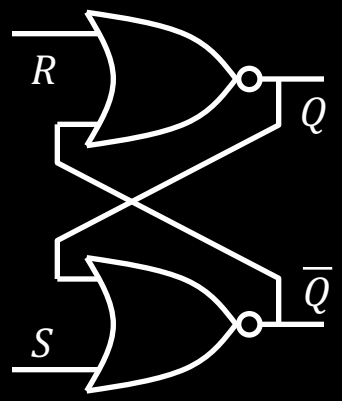


SR latches can be used to register & remember transient signals & keep them memorized even after they are gone



Application: Switch Debouncing



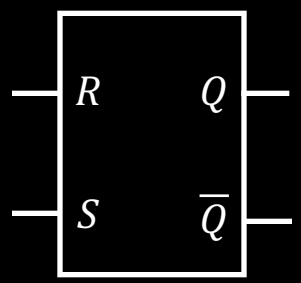


Gating an SR Latch

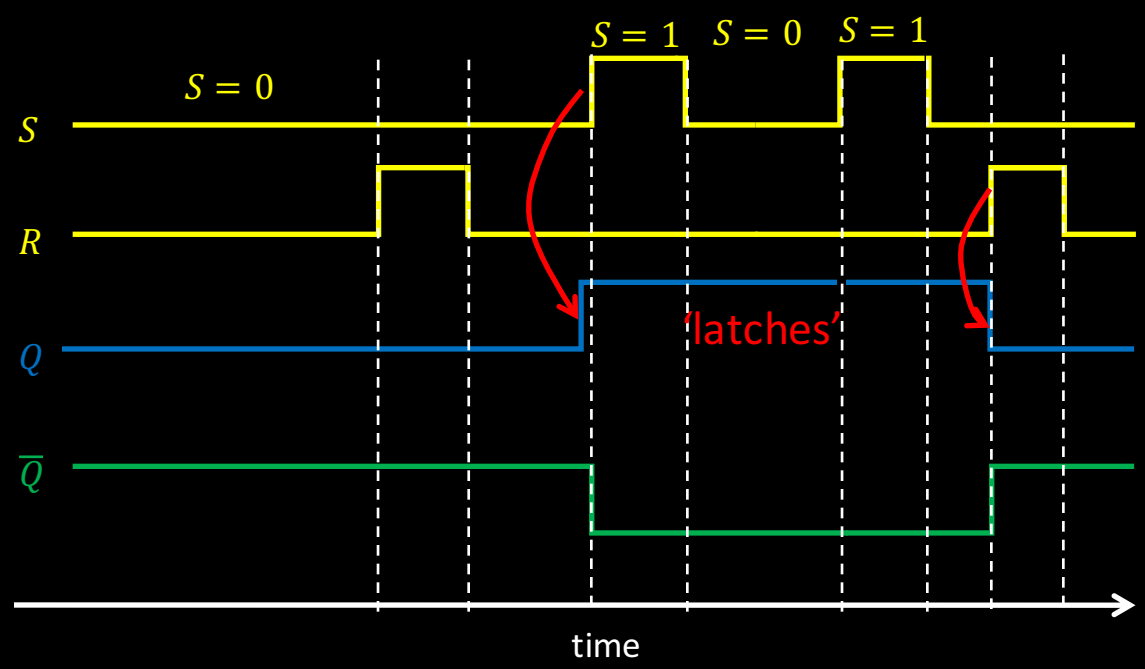
Changes in state determined by inputs, S & R. What if we could 'gate' or 'clock' the transitions?

SYNCHRONOUS

ASYNCHRONOUS



TIMING DIAGRAM



Asynchronous vs. Synchronous Sequential Circuits

Asynchronous Circuits

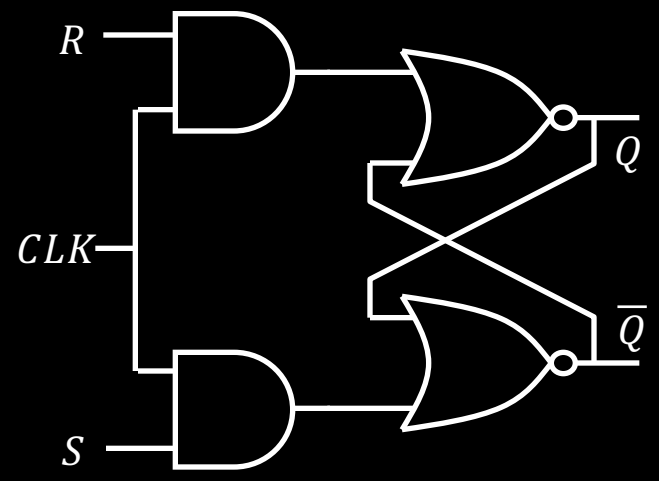
- Respond immediately to changes at the input
- Example: SR Latch that we studied previously

Synchronous Circuits

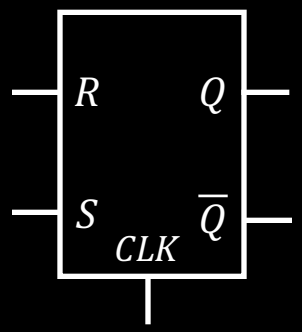
- State changes are controlled by a "Clock"
  - A clock allows ordering of events & keeps the whole machine in sync
- Changes can happen only when clock is "active"
- Example: computers are made largely of synchronous sequential circuit components

A "clock" is a special circuit that sends synchronizing electrical pulses

- Clock produces periodic electrical signal
- Also known as clock, CLK, etc.

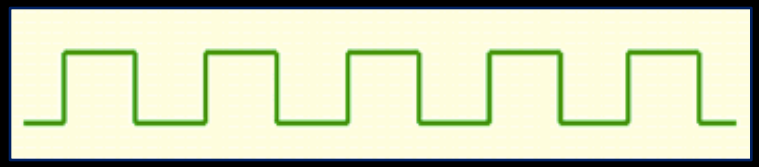


Gated SR Circuit



high, 1, 5V

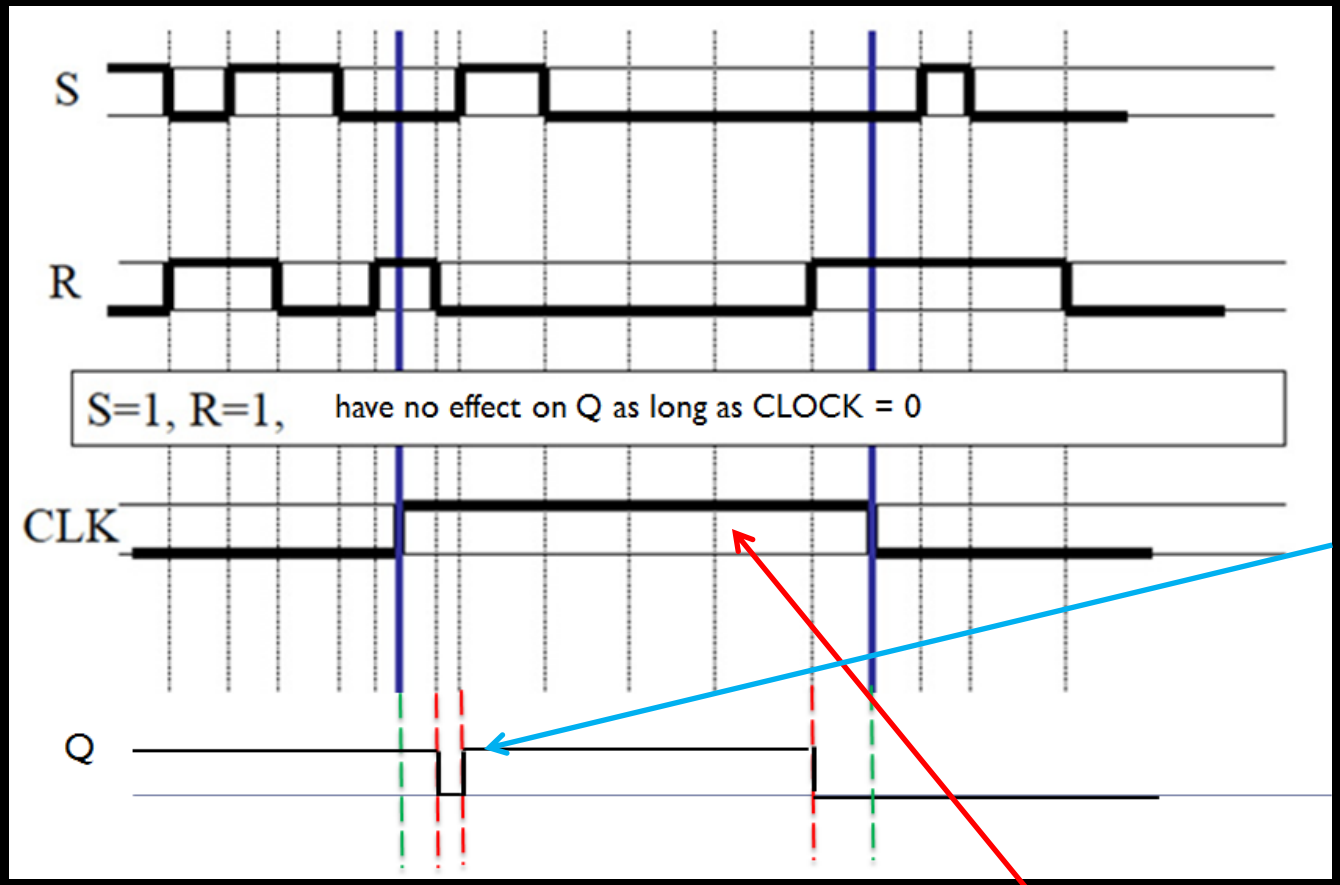
low, 0, 0V



Clock pulse

time

Gated SR Latch: timing sequence

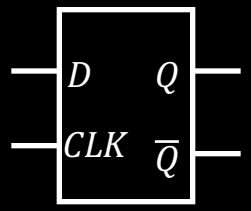


May actually be a slight delay in the response of the circuit to the clock (not shown)

This SR latch is sensitive to the LEVEL of CLK signal. The CLOCK defines the time within which the Set and Reset 'work'

- A SR latch can also be sensitive to only the EDGE (rising or falling) of CLK. This is called a Flip-Flop. For many applications, edge-sensitive 'flip-flops' are preferred

D Latch (Data Latch)

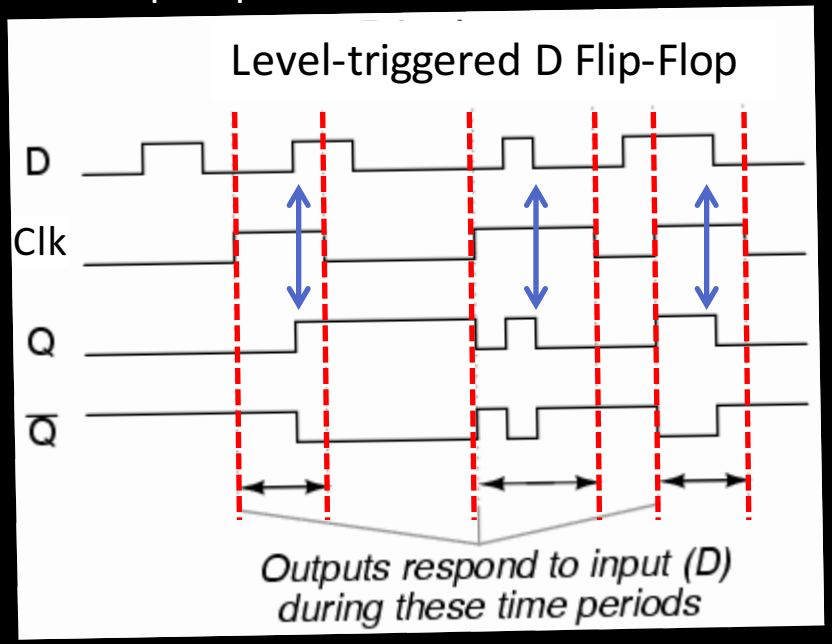


CLK	D	Q	Q’
0	0	latch	latch
0	1	latch	latch
1	0	0	1
1	1	1	0

D	Q
0	0
1	1

- The CLK allows ‘latching’ or memory storage
- When $CLK = 1$, the input at D determines the output
  - This is called D LATCH
  - Can be used as 1-bit memory circuit, storing either a "high" or a "low" state when disabled, and "reading" new data from the D input when gated
  - SRAM (Static Random Access Memory) in computers is implemented as D flip-flops!

Clock level ‘high’

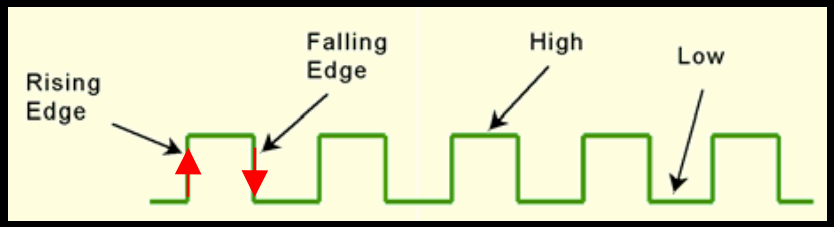


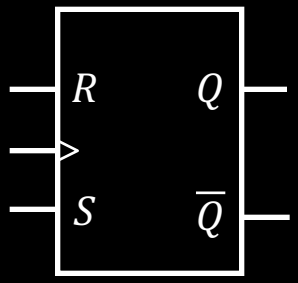
Level Triggered vs Edge Triggered

Level-triggered circuits change state when the clock voltage is at its highest (or lowest) level. Example: Gated SR Latch shown in the previous slide.

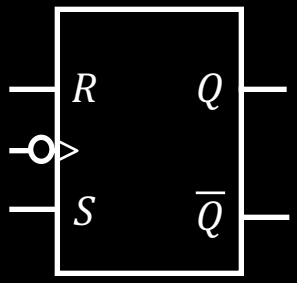
- The problem with *Latch is that state keeps changing* according to the values of the input during the period when the clock is active (i.e. $CLK = 1$)
- To allow complete synchronization in the system, we can restrict the change in state only at the edges (rising or falling) of the clock.

Circuits that change state on the rising edge, or falling edge of the clock pulse are called edge-triggered latches, or simply flip-flops! The CLOCK serves as a 'trigger' to the change of state of the circuit



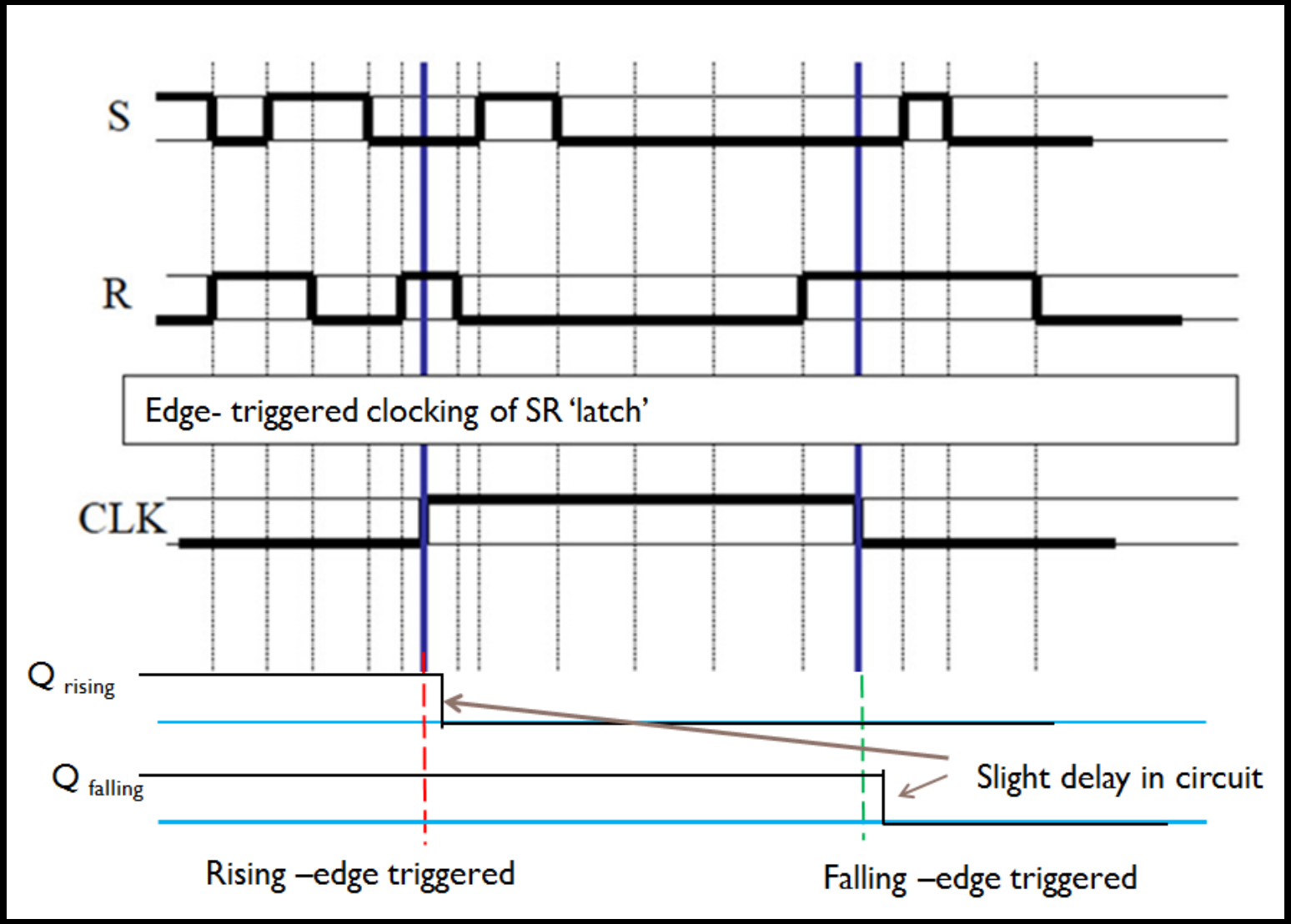


Rising (positive) edge trigger



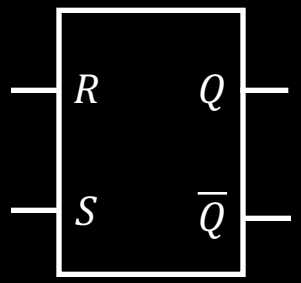
Falling (negative) edge trigger

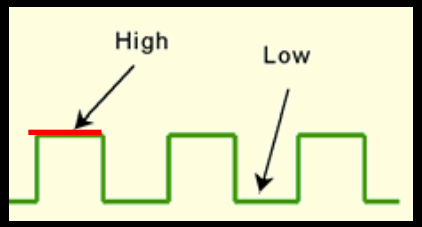
SR Flip-Flop: timing sequence, edge-triggered

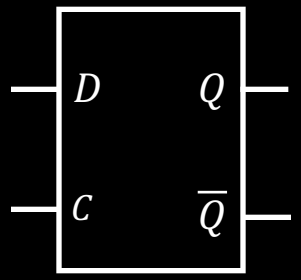


Level Triggered vs Edge Triggered

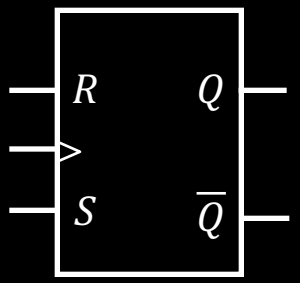
Level sensitive

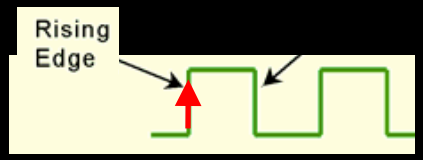


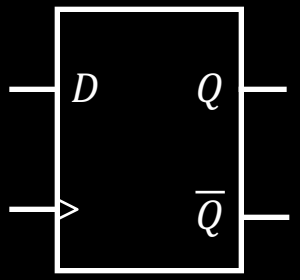




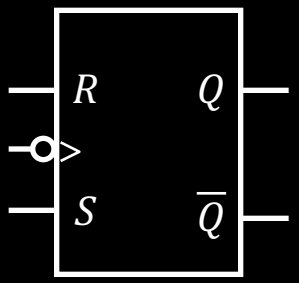
Rising edge triggered

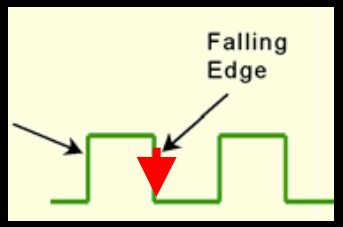


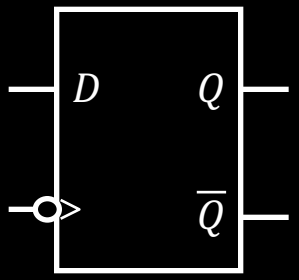




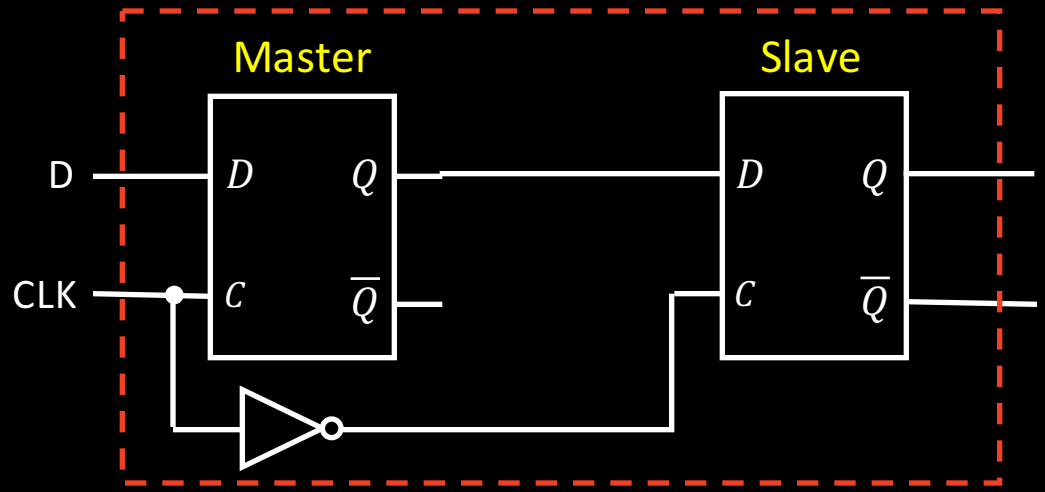
Falling edge triggered

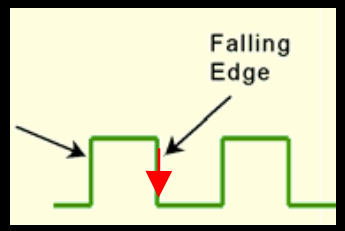


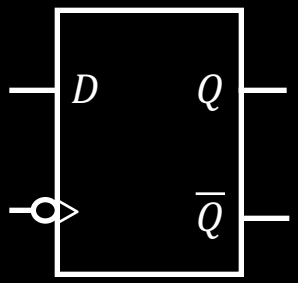




How do we make edge-triggered flip-flops?



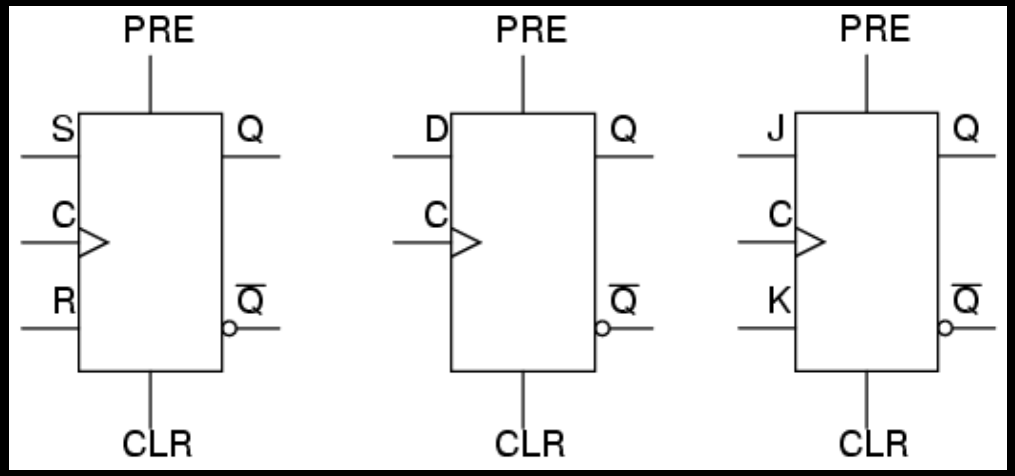




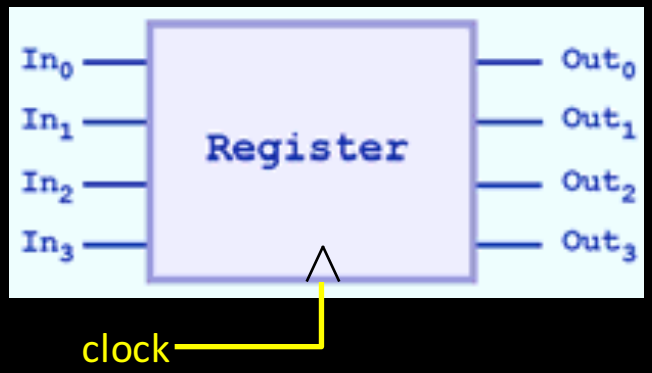
- When $C = 1$, master is enabled & stores new data, slave does nothing
- When $C = 0$, master's state passes the date to the slave. Slave is now enabled as $C = 0$, and it stores the new value.
  - Master is not sensitive to any changes at the input D at this point!
- In this way, the data that is at the input of the master immediately before CLK goes from high to low (falling edge) is memorized!
  - Hence, this is falling-edge triggered D flip-flop!
- How would you make rising-edge triggered D flip-flop?
  - Add a NOT gate to the clock!

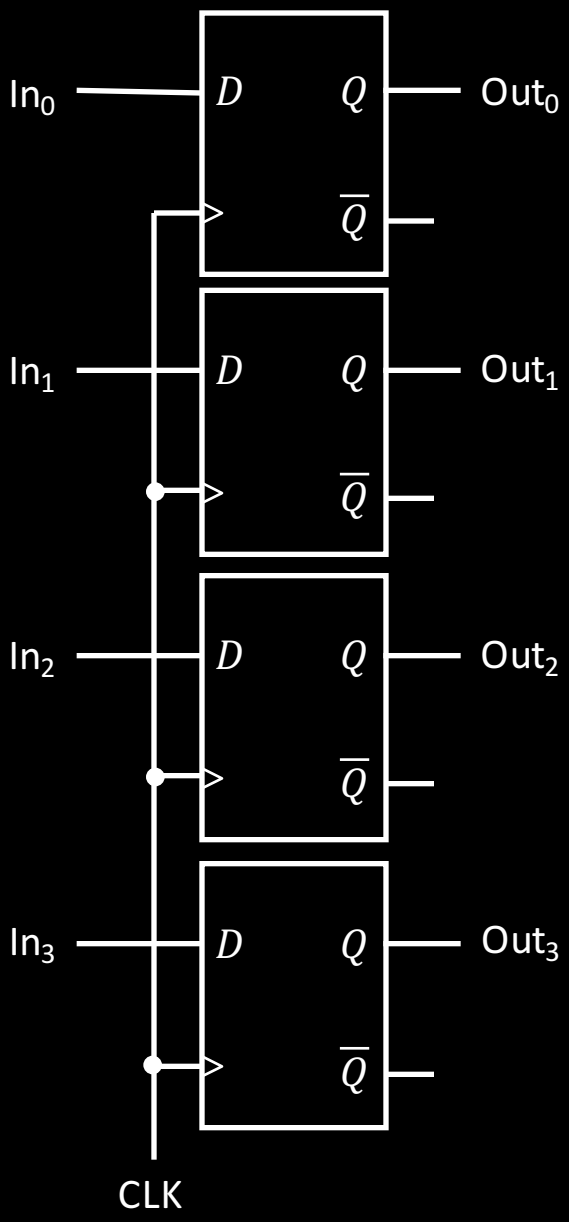
Preset and Clear Signals (asynchronous)

- We've been studying ***synchronous*** flip flops that are gated by a clock.
- Most flip-flops also have ***asynchronous*** inputs that affect state of the flip-flop independent of the clock
  - D Flip-Flop with preset (PRE) & clear (CLR) inputs, If the
    - PRE = 1 & CLR = 0, then flop is set to 1
    - PRE = 0 & CLR= 1, then flop is cleared/reset to 0
- Asynchronous signals PRE and CLR "do not care" about the clock – that is once they are supplied to the flip flop, flip-flop responds to them regardless of clock.
  - SR, JK & all other flip-flops can also have asynchronous inputs

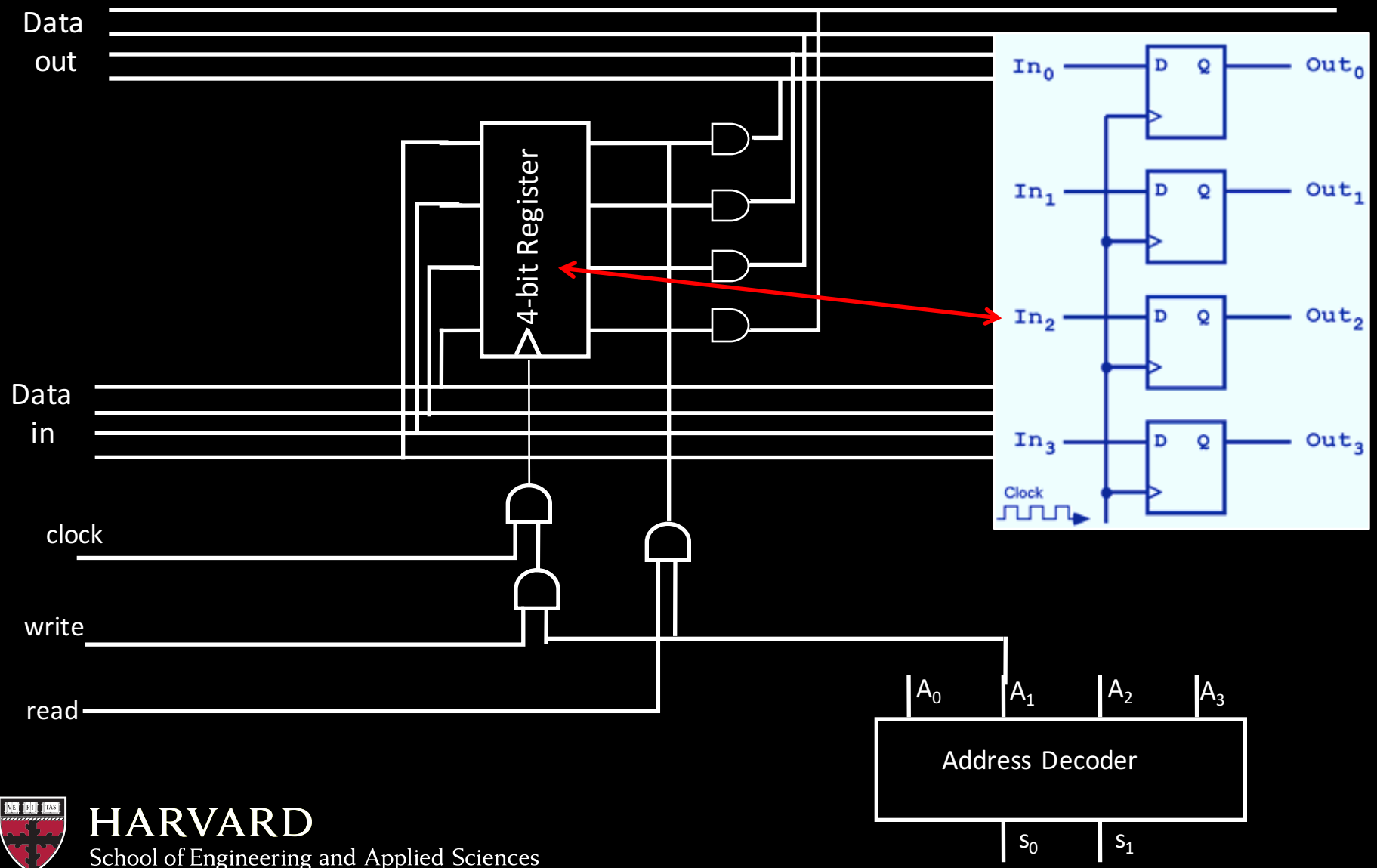


Memory, 4-Bit Register





4-bit Computer Memory

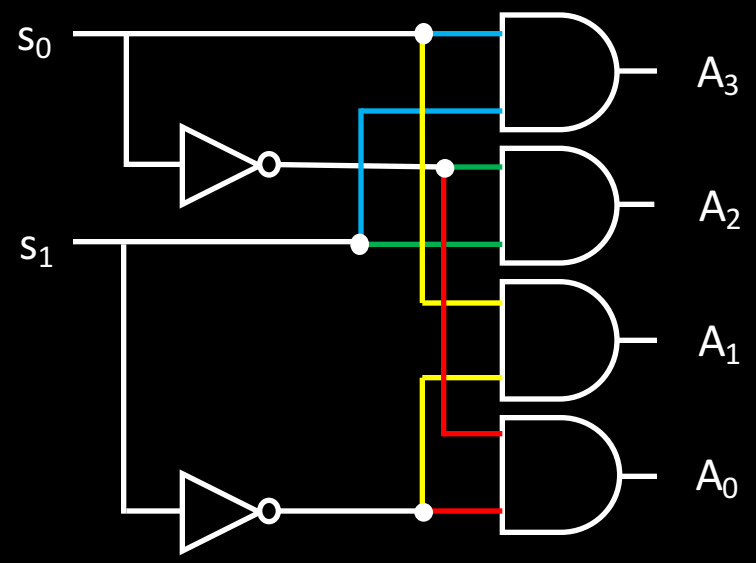


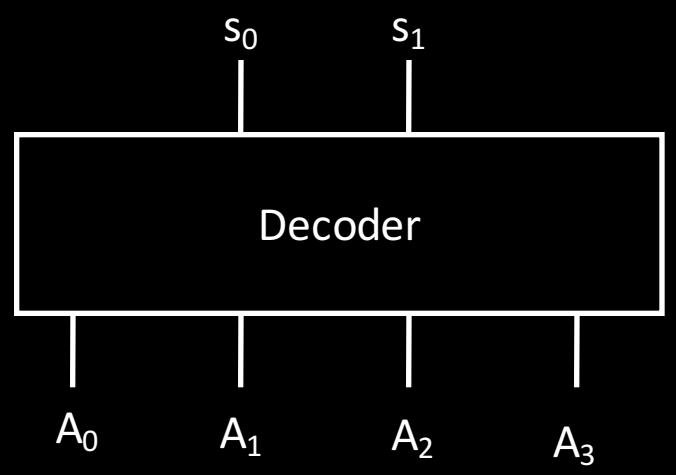
How does address decoder work?

To represent 4 locations we need only 2-bits for address

- Address decoder performs the address decoding - selects a particular *row or word in memory*
- To store information we use D Flip-Flop for each bit

s_1	s_0	Out
0	0	A_0
0	1	A_1
1	0	A_2
1	1	A_3

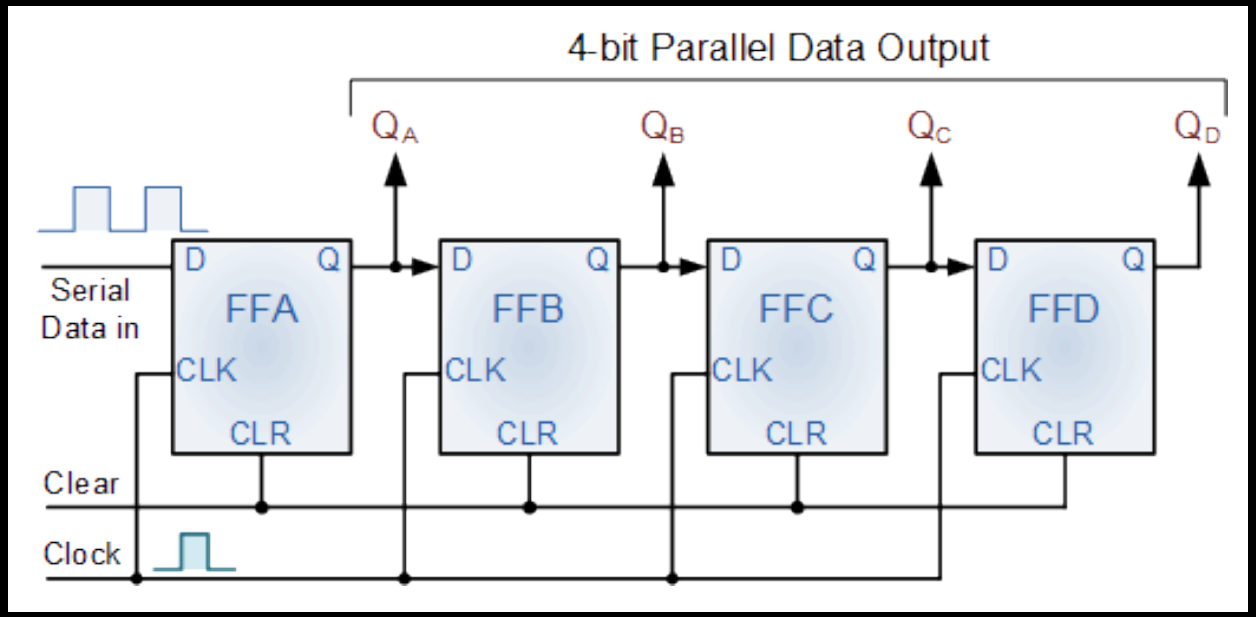


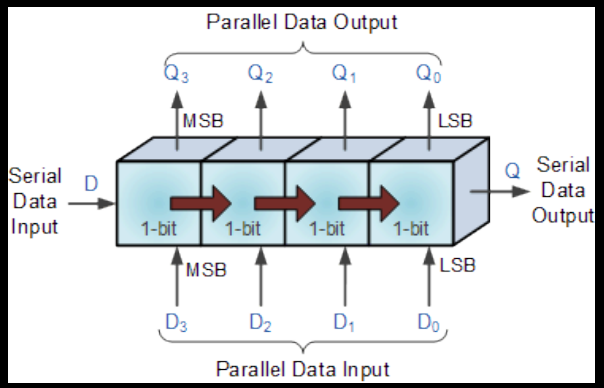


Application: Shift Register

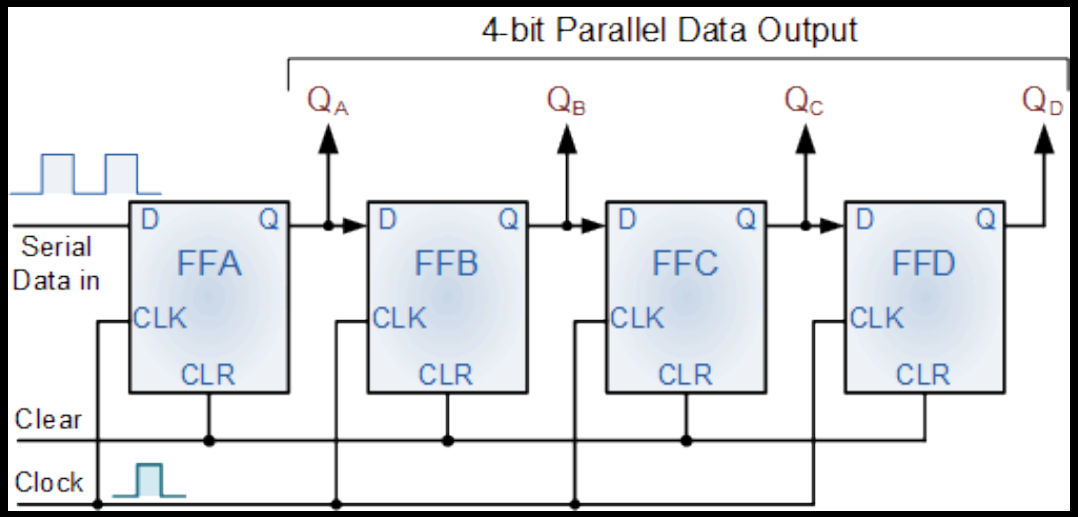
Shiift Registers are a way of moving data into and out of a device

- A way of transferring a serial input of bits to a *parallel output* or visa versa

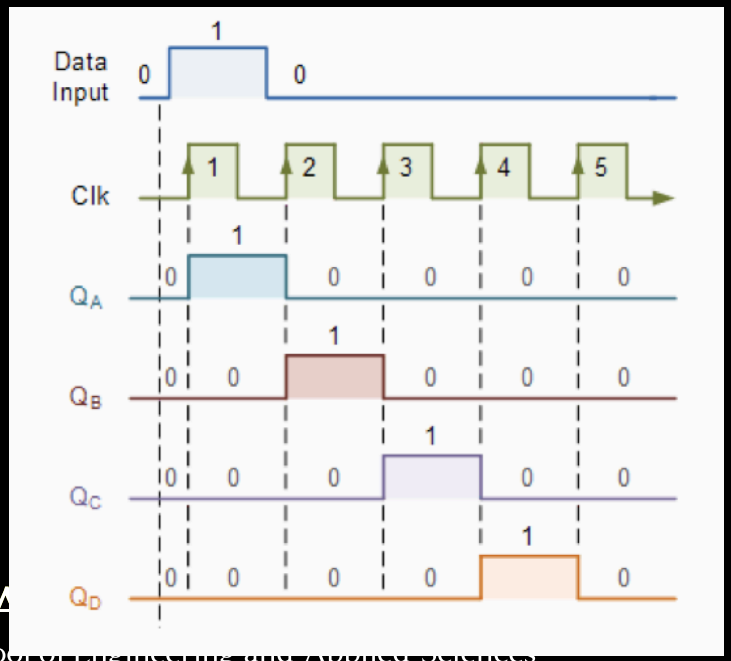




Application: Shift Register



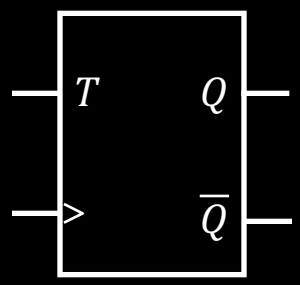
Why are these things useful? It seems they slow things down!



Clock Pulse	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	0	0	0	0

T (toggle) Flip-Flop

If $T = 1$, Q toggles (goes from 0 to 1, that is from 1 to 0) on the active edge of the clock

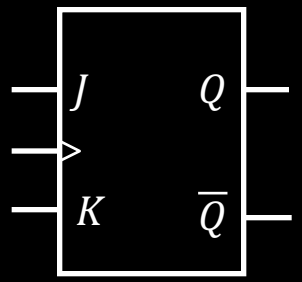


T	Q(t)	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

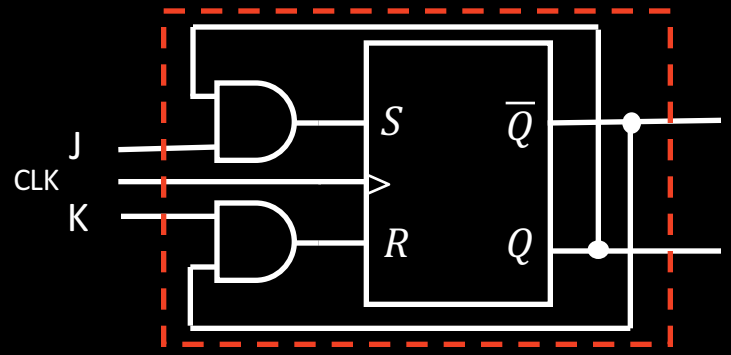
This can also be written in more compact notation:

T	Q(t+1)
0	Q(t)
1	Q(t)'

JK Flip-Flop



To make perfectly sure that we never have an unstable circuit (i.e. $SR = 11$), we can modify the SR Flip-Flop circuit as follows



J	K	Q(t+1)
0	0	Q(t) no change
0	1	0 (reset to 0)
1	0	1 (set to 1)
1	1	Q'(t)

If the circuit is SET

- $Q = 1, \overline{Q} = 0$
- K is 'inhibited' (does not influence outcome)

If circuit is RESET

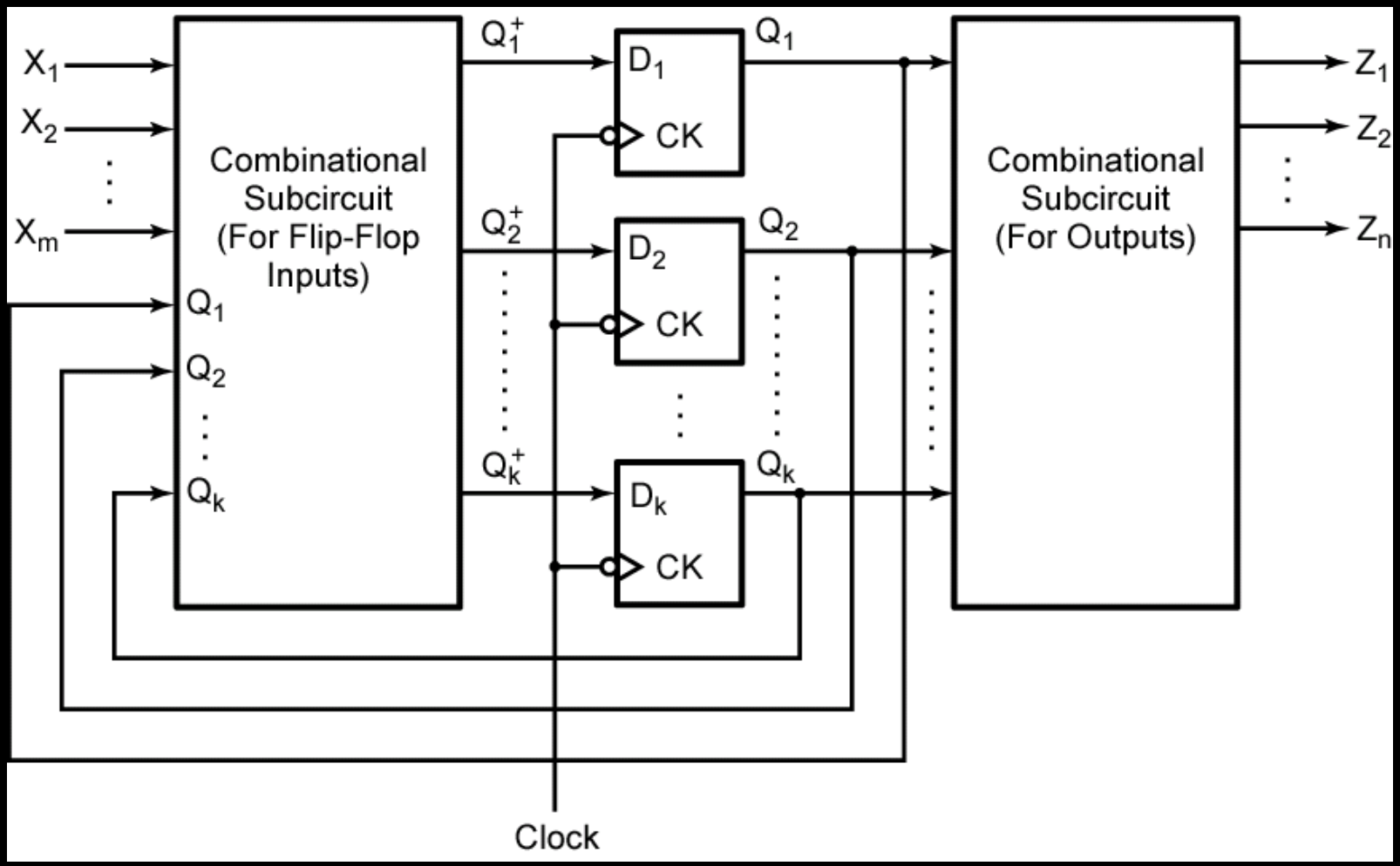
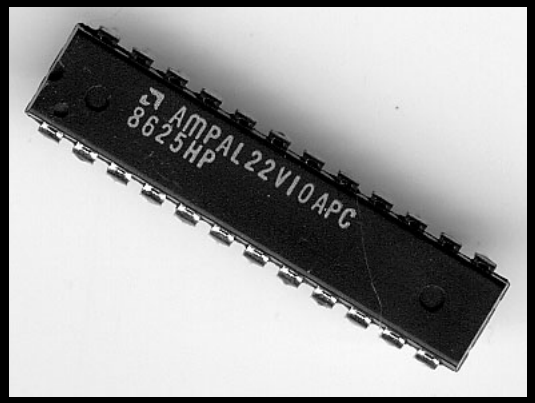
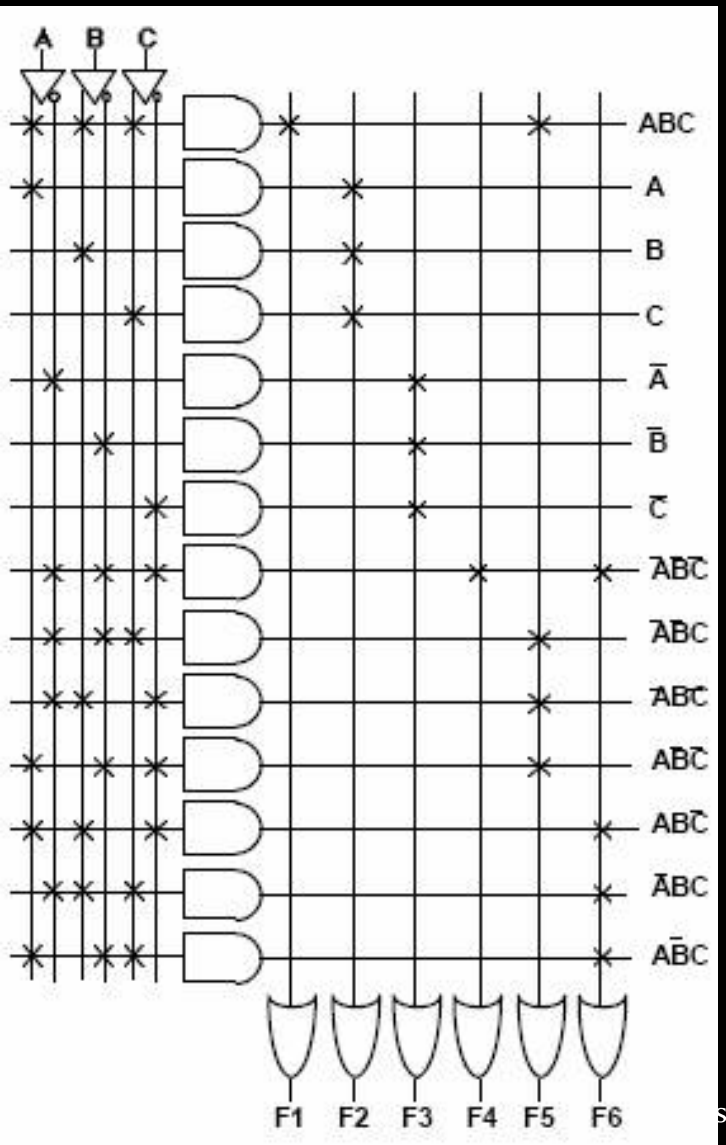
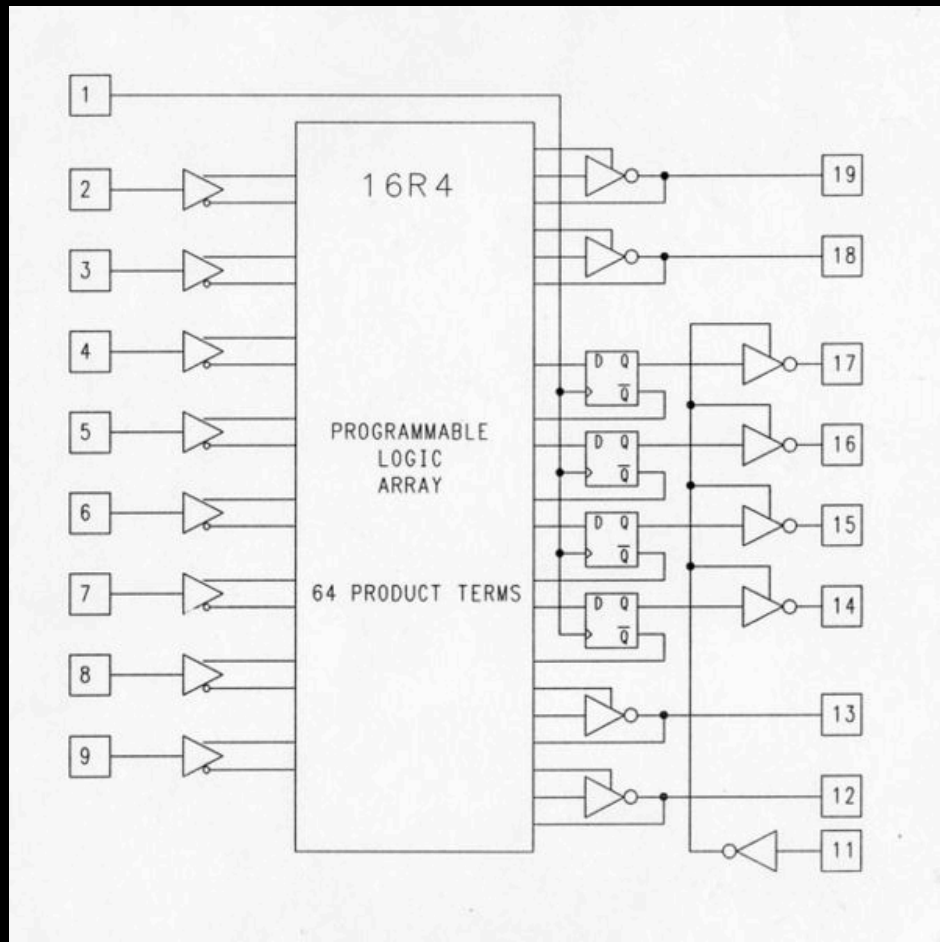
- $Q = 0, \overline{Q} = 1$
- J is inhibited

K & J are 'interlocked' & cannot both be activated

- Circuit can cycle through $K = 1, J = 1$ at every clock cycle

Proposed by Jack Kilby, Inventor of Integrated Circuit, shared 2000 Nobel Prize in Physics

You can make many useful things using following approach:



Integrated version: Programmable Logic Arrays