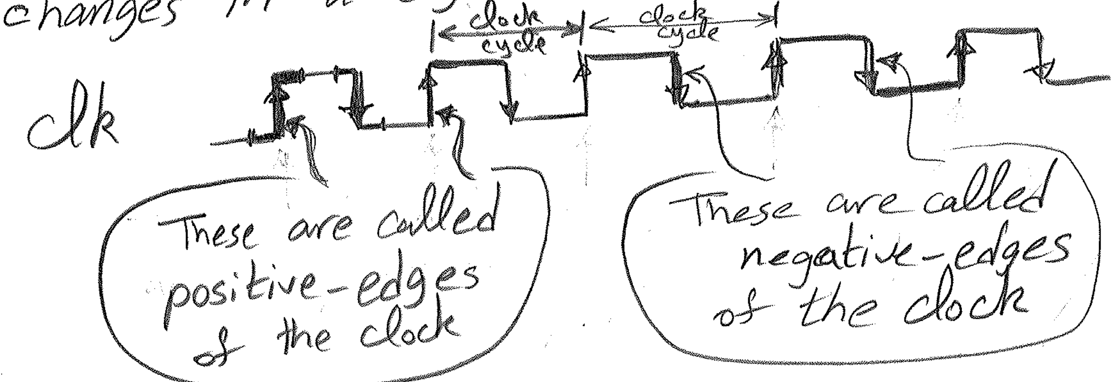
2 Flip-Flops:

* Flip-flops are synchronous bistable devices as opposed to latches which are asynchronous bistable devices.

- "bistable" means that the device can reside in either of two states, one of which $Q=1$ and the other $Q=0$.
- "Asynchronous" means that the device may change its state at any point in time with any change in inputs.
- "Synchronous" means that the device can only change its state at specific points in time. These points in time are identified (or marked) by changes in a signal named clock (clk)


clk
clock cycle
clock cycle
These are called positive-edges of the clock
These are called negative-edges of the clock

- Flip-flops can change its output at either the positive-edges of the clock (called positive-edge (triggered) flip-flops, or at the negative edges of the clock (called negative-edge (triggered) flip-flops but not both!

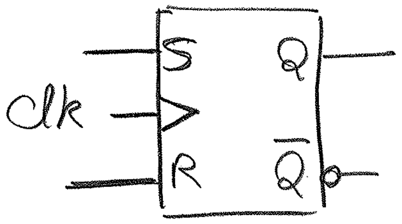
* We study here three types of (edge-triggered) flip-flops:

① S-R flip-flops ② D flip-flops ③ J-K flip-flops

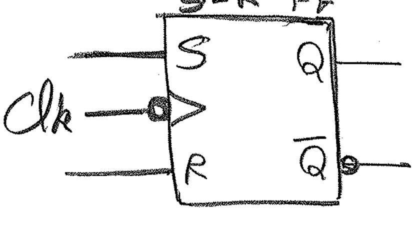
① S-R Flip-Flop:

- Logic symbol:

+ve edge S-R FF



-ve edge S-R FF



- Truth Table:

clk	S	R	Q	$\bar{Q}$	
↑	0	0	Q_{prev}	$\bar{Q}_{prev}$	No change
↑	0	1	0	1	Reset
↑	1	0	1	0	Set
↑	1	1	?	?	Invalid
$\not\uparrow$	×	×	Q_{prev}	$\bar{Q}_{prev}$	

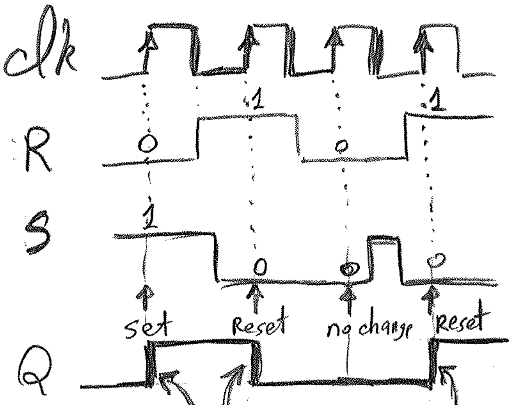
+ve edge of the clock (i.e. clock transition from low → high)

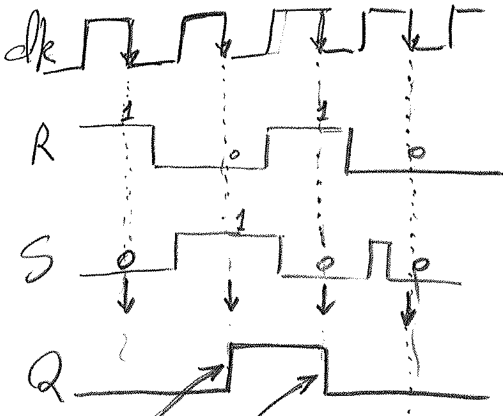
No +ve edge

clk	S	R	Q	$\bar{Q}$
↓	0	0	Q_{prev}	$\bar{Q}_{prev}$
↓	0	1	0	1
↓	1	0	1	0
↓	1	1	?	?
$\not\downarrow$	×	×	Q_{prev}	$\bar{Q}_{prev}$

No edge ⇒ No change

- Timing Diagram:





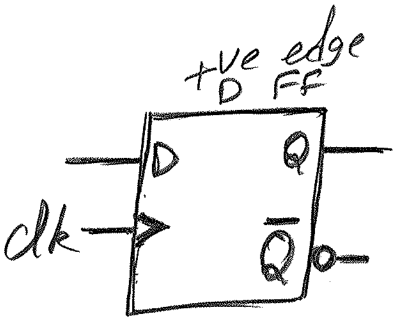
changes in output can only happen at the edge of the clock

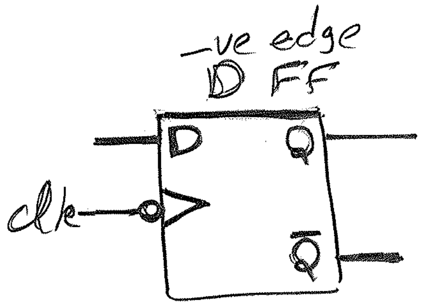
⇓

Synchronous device

② D-flip flop:

- Logic symbol:





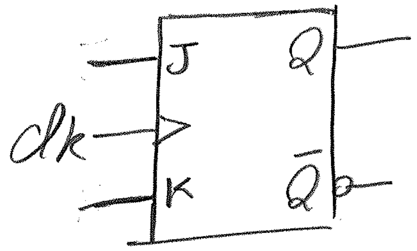
- Truth Table:

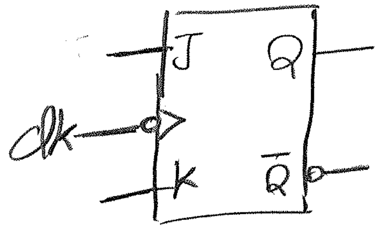
	clk	D	Q	$\overline{Q}$
Load ↝	$\uparrow$	X	D	$\overline{D}$
save ↝	$\not\uparrow$	X	Q_{prev}	$\overline{Q}_{prev}$

clk	D	Q	$\overline{Q}$
$\downarrow$	X	D	$\overline{D}$
$\not\downarrow$	X	Q_{prev}	$\overline{Q}_{prev}$

③ J-K flip flop:

- Logic symbol:





- Truth Table:

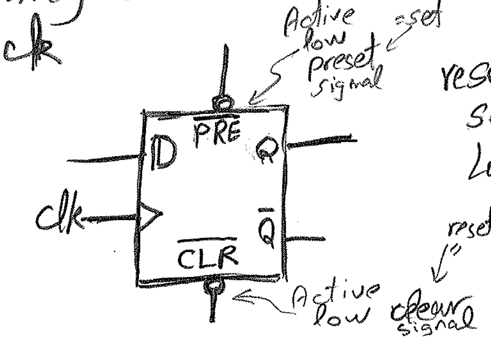
	clk	J	K	Q	$\overline{Q}$
No change →	$\uparrow$	0	0	Q_{prev}	$\overline{Q}_{prev}$
reset →	$\uparrow$	0	1	0	1
set →	$\uparrow$	1	0	1	0
Toggle →	$\uparrow$	1	1	$\overline{Q}_{prev}$	Q_{prev}
No change →	$\not\uparrow$	X	X	Q_{prev}	$\overline{Q}_{prev}$

clk	J	K	Q	$\overline{Q}$
$\downarrow$	0	0	Q_{prev}	$\overline{Q}_{prev}$
$\downarrow$	0	1	0	1
$\downarrow$	1	0	1	0
$\downarrow$	1	1	$\overline{Q}_{prev}$	Q_{prev}
$\not\downarrow$	X	X	Q_{prev}	$\overline{Q}_{prev}$

NOTE: Flip-flops might have additional i/p's for resetting and/or setting the output asynchronously, i.e., regardless of the clock

Example:

D-flip flop with clear & preset signals



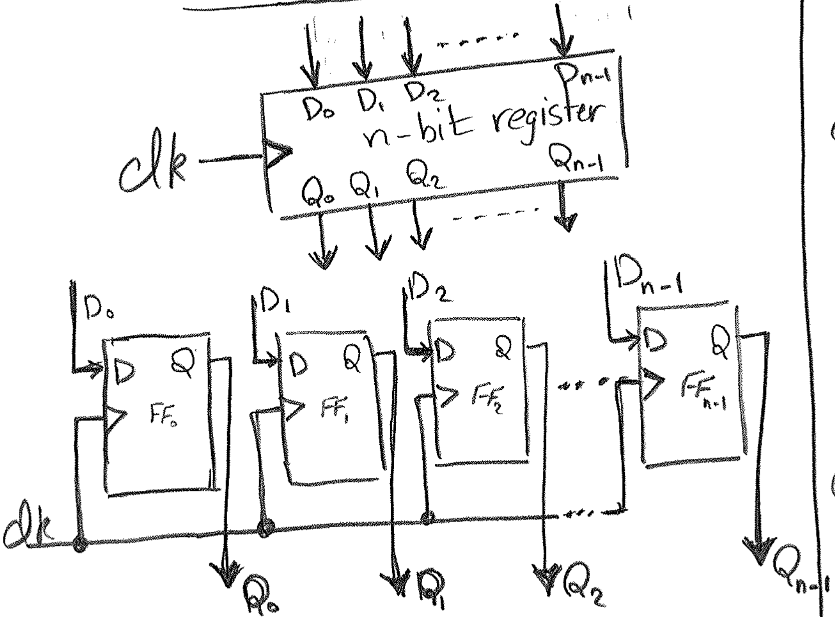
	CLR	PRE	clk	D	Q	$\overline{Q}$
reset →	$\boxed{0}$	X	X	X	0	1
set →	1	$\boxed{0}$	X	X	1	0
Load →	1	1	$\uparrow$	X	D	$\overline{D}$
save →	1	1	$\not\uparrow$	X	Q_{prev}	$\overline{Q}_{prev}$

Applications of flip-flops:

(11) Registers:

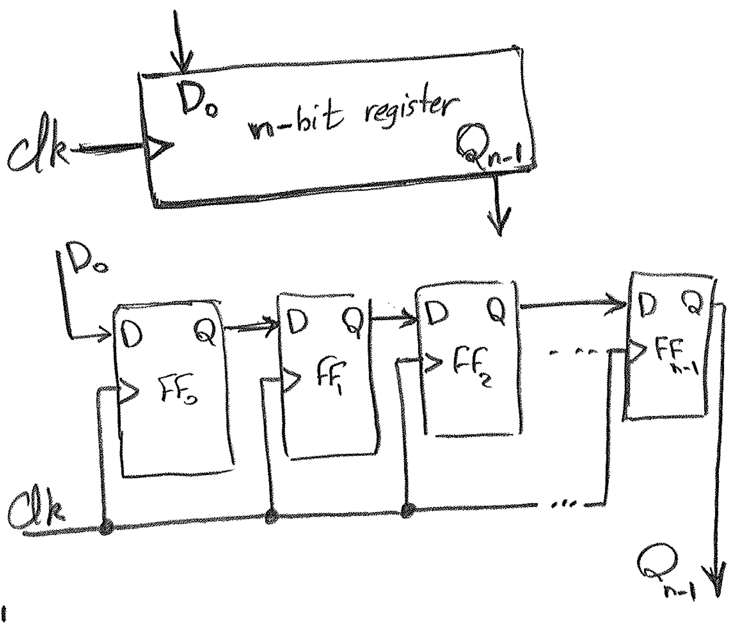
- Registers are digital circuits with two basic functions:
  ① Data storage ② Data movement
- Registers are typically built using D-flip flops
- A register built using $\boxed{n \text{ flip-flops}}$ can store $\boxed{n \text{ bits}}$.
- Types:

① Parallel-in / Parallel-out (PIPO)



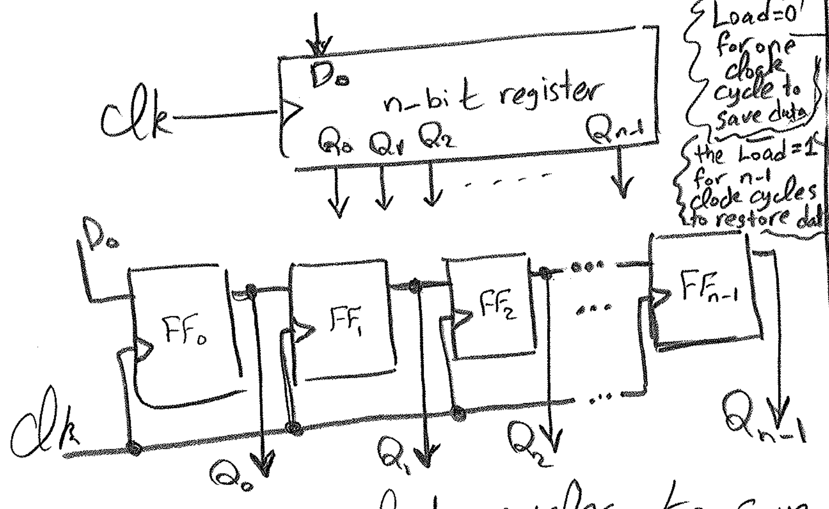
- Need 1 clock cycle to save and restore data!

② Serial-In/Serial-Out (SISO)



- Need 2n-1 clock cycles to save and restore data!

③ Serial-in/Parallel-out (SIPO)

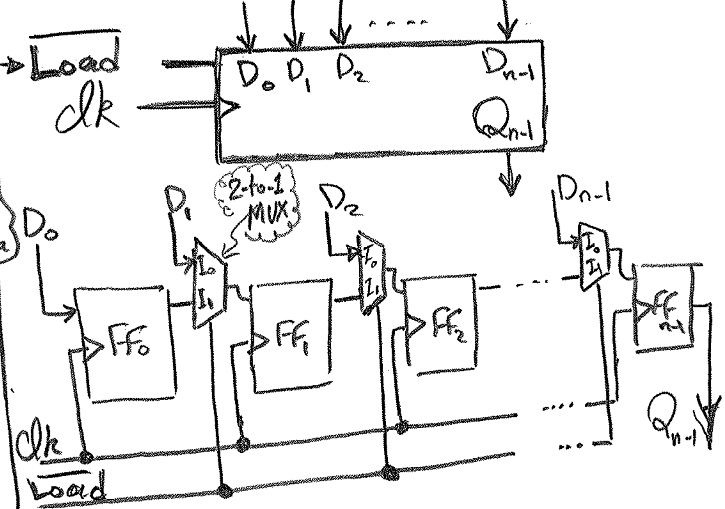


Load=0 for one clock cycle to save data

the Load=1 for n-1 clock cycles to restore data

- Need n clock cycles to save & restore data!

④ Parallel-In/Serial-Out (PISO)

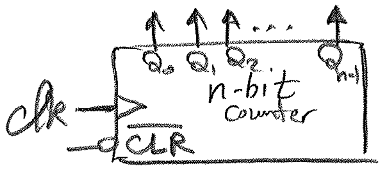


- Need n clock cycles to save & restore!

② Counters :

- Counters are digital circuits that produce a repeated sequence of binary values on outputs

- Just like registers:
  * counters are built using flip-flops (plus additional combinational logic elements).
  * To build an n-bit counter, we need n flip-flops.



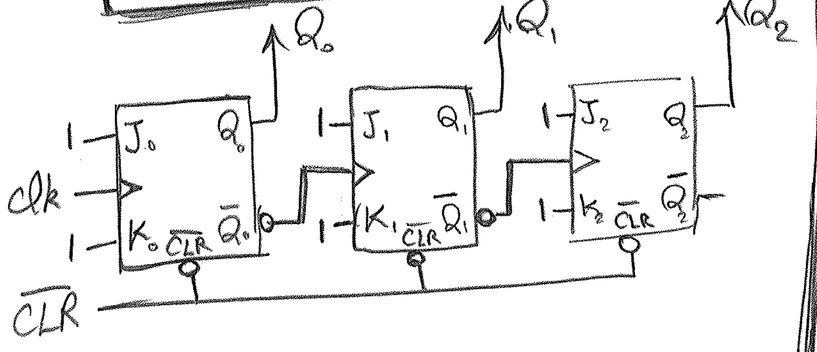
- Types: ① Asynchronous Counters
  ② Synchronous Counters

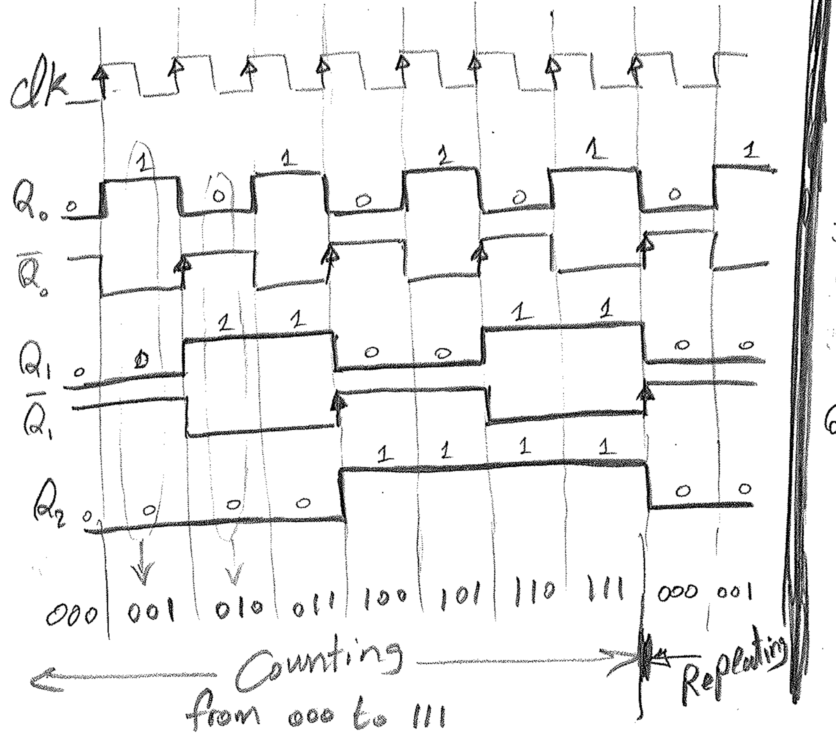
{Flip flops have different clock signals}

{flip-flops share the same clock signal}

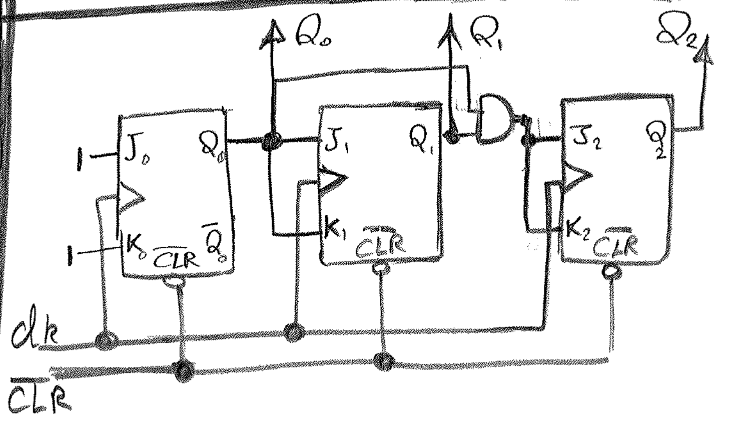
- Examples:

3-bit Asynchronous Counter

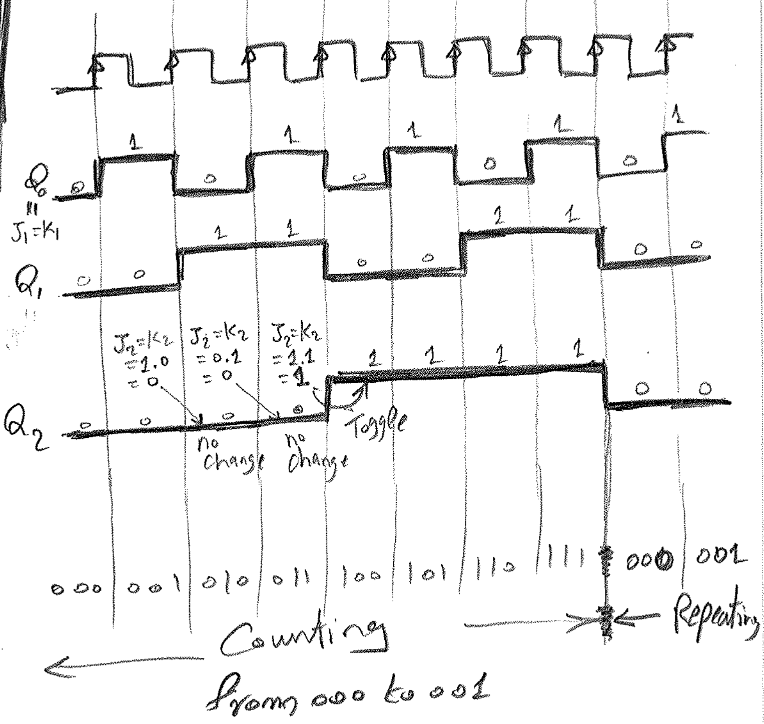




3-bit Synchronous Counter



Q_0
Q_1
Q_2
1
J_0
Q_0
K_0
$\overline{CLR}$
$\overline{Q_0}$
J_1
Q_1
K_1
J_2
Q_2
K_2
clk
$\overline{CLR}$


$Q_0 = J_1 = K_1$
Q_1
Q_2
$J_2=K_2 = 1.0 = 0$
$J_2=K_2 = 0.1 = 0$
$J_2=K_2 = 1.1 = 1$
no change
no change
Toggle
Counting from 000 to 001
Repeating