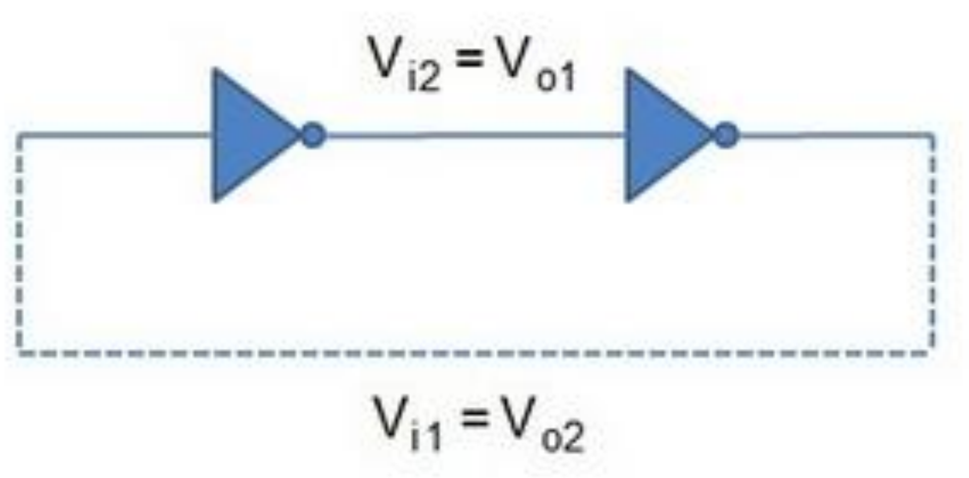
Latches and Flip Flops

Basic Stable Memory Element



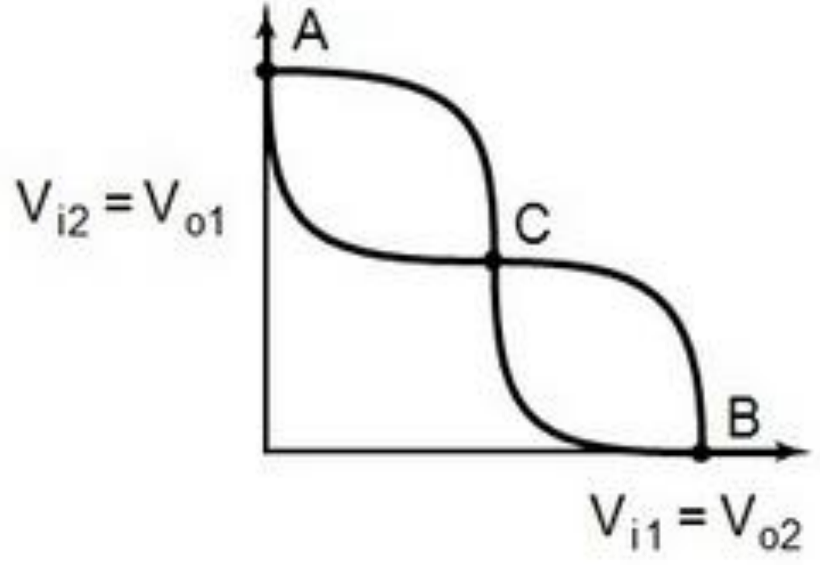
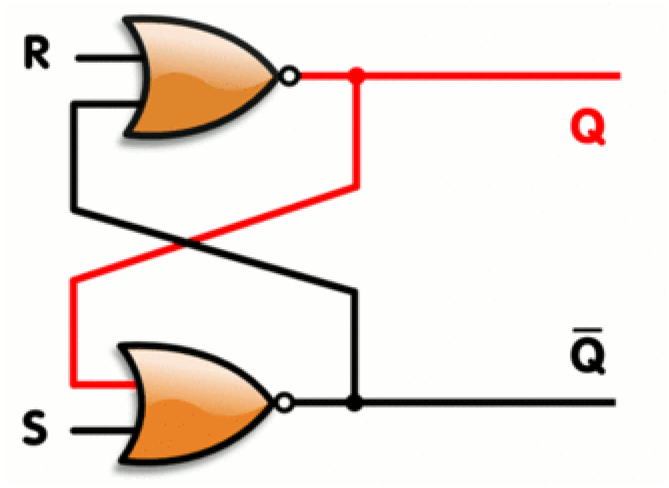


Fig.1: Two cascaded inverters along with their superimposed VTCs.

Bistable Multivibrators



Latch

- Bistable multivibrator

- State change triggers
  - Level triggered
  - Clock edge triggered (Flip Flops)

Memory Elements and Their Excitation Functions

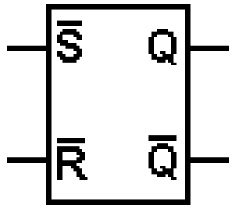
To generate the *Y*'s: memory devices must be supplied with appropriate input values

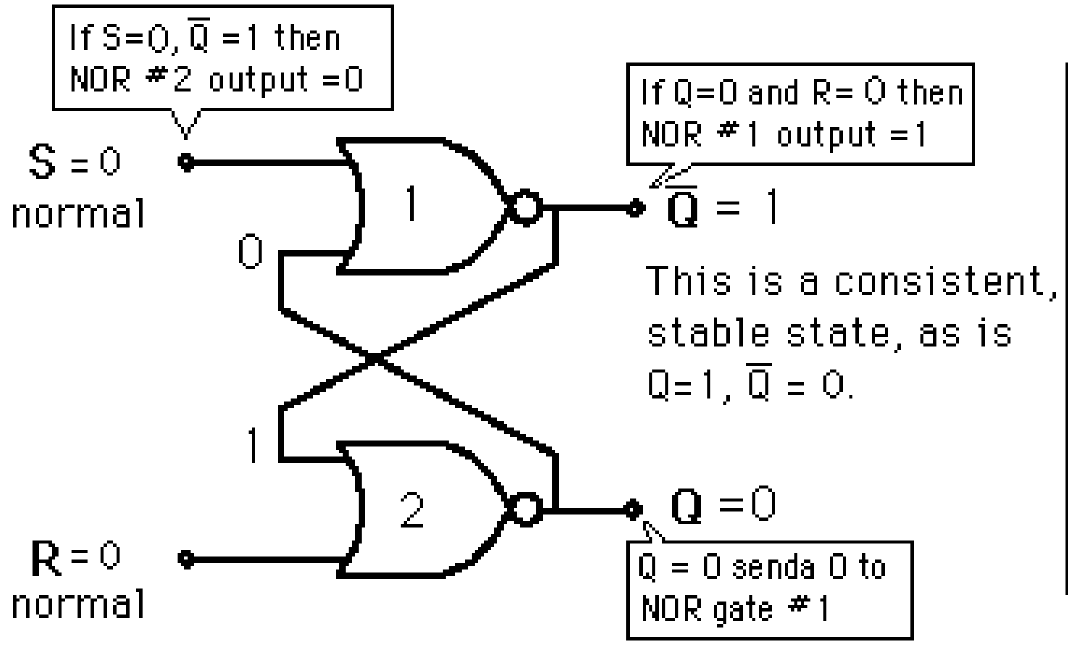
- Excitation functions: switching functions that describe the impact of x_i's and y_j's on the memory-element input
- Excitation table: its entries are the values of the memory-element inputs

Most widely used memory elements: flip-flops, which are made of latches

- Latch: remains in one state indefinitely until an input signals directs it to do otherwise

Set-Reset (SR) Latch (NOR Gate)




If S=0, $\overline{Q}$ =1 then NOR #2 output =0
S =0 normal
1
0
1
2
R =0 normal
If Q=0 and R= 0 then NOR #1 output =1
$\overline{Q}$ = 1
Q =0
Q = 0 sends 0 to NOR gate #1

This is a consistent, stable state, as is Q=1, $\overline{Q}$ = 0.

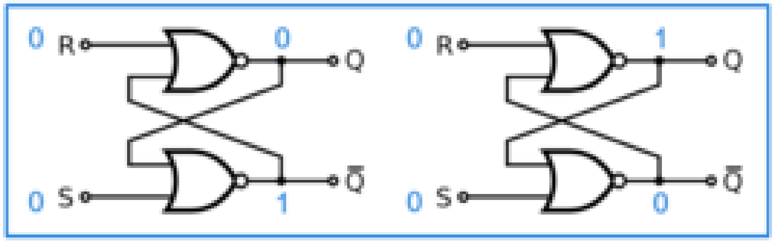
Truth Table

Set	Reset	Output
0	0	No change*
1	0	Q=1
0	1	Q=0
1	1	Invalid state

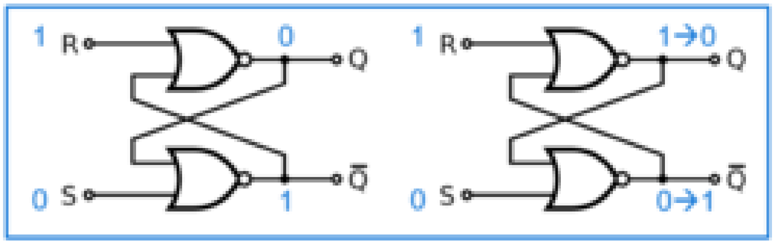
*can be used for data storage

Working

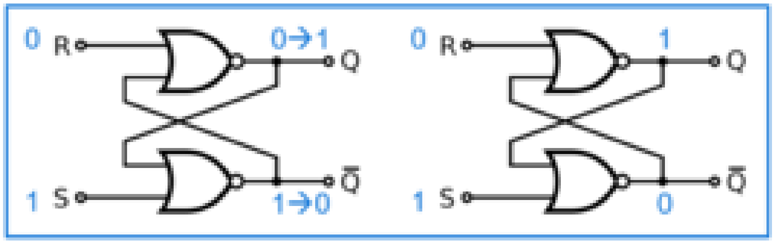
$(R, S) = (0, 0)$→ Hold the previous state as $(Q_{next}, \overline{Q_{next}}) = (Q, \overline{Q})$



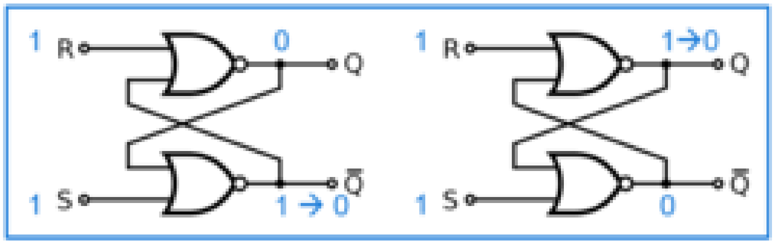
$(R, S) = (1, 0)$→ Reset as $(Q_{next}, \overline{Q_{next}}) = (0, 1)$



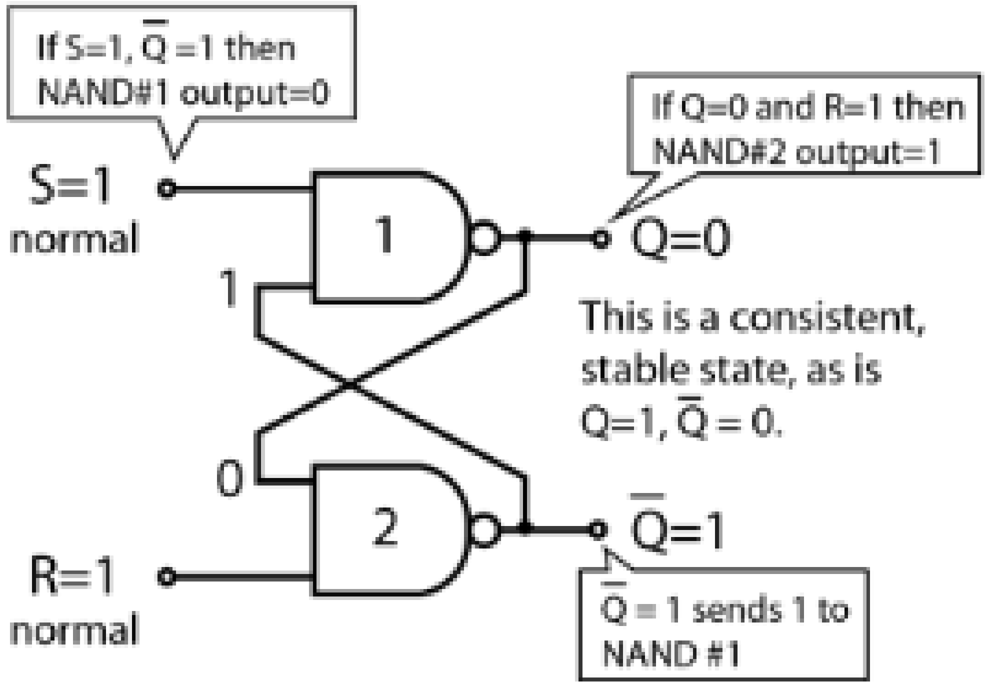
$(R, S) = (0, 1)$→ Set as $(Q_{next}, \overline{Q_{next}}) = (1, 0)$



$(R, S) = (0, 1)$→ Not allowed as $(Q_{next}, \overline{Q_{next}}) = (0, 0)$ that is not logical

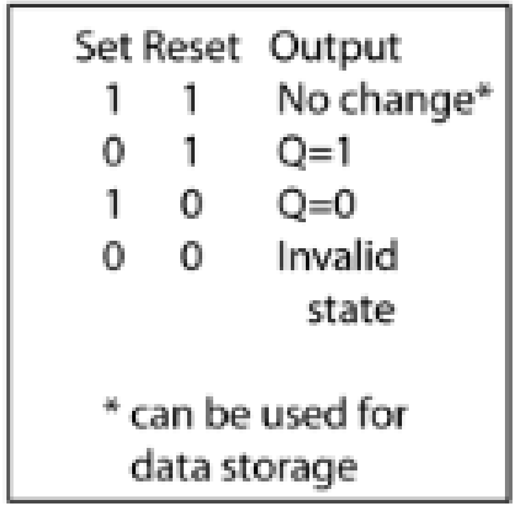


Set-Reset (SR) Latch (NAND Gate)


If S=1, $\overline{Q}$ =1 then NAND#1 output=0
S=1
normal
1
1
If Q=0 and R=1 then NAND#2 output=1
Q=0
0
2
$\overline{Q}$=1
R=1
normal
$\overline{Q}$ = 1 sends 1 to NAND #1

This is a consistent, stable state, as is Q=1, $\overline{Q}$ = 0.

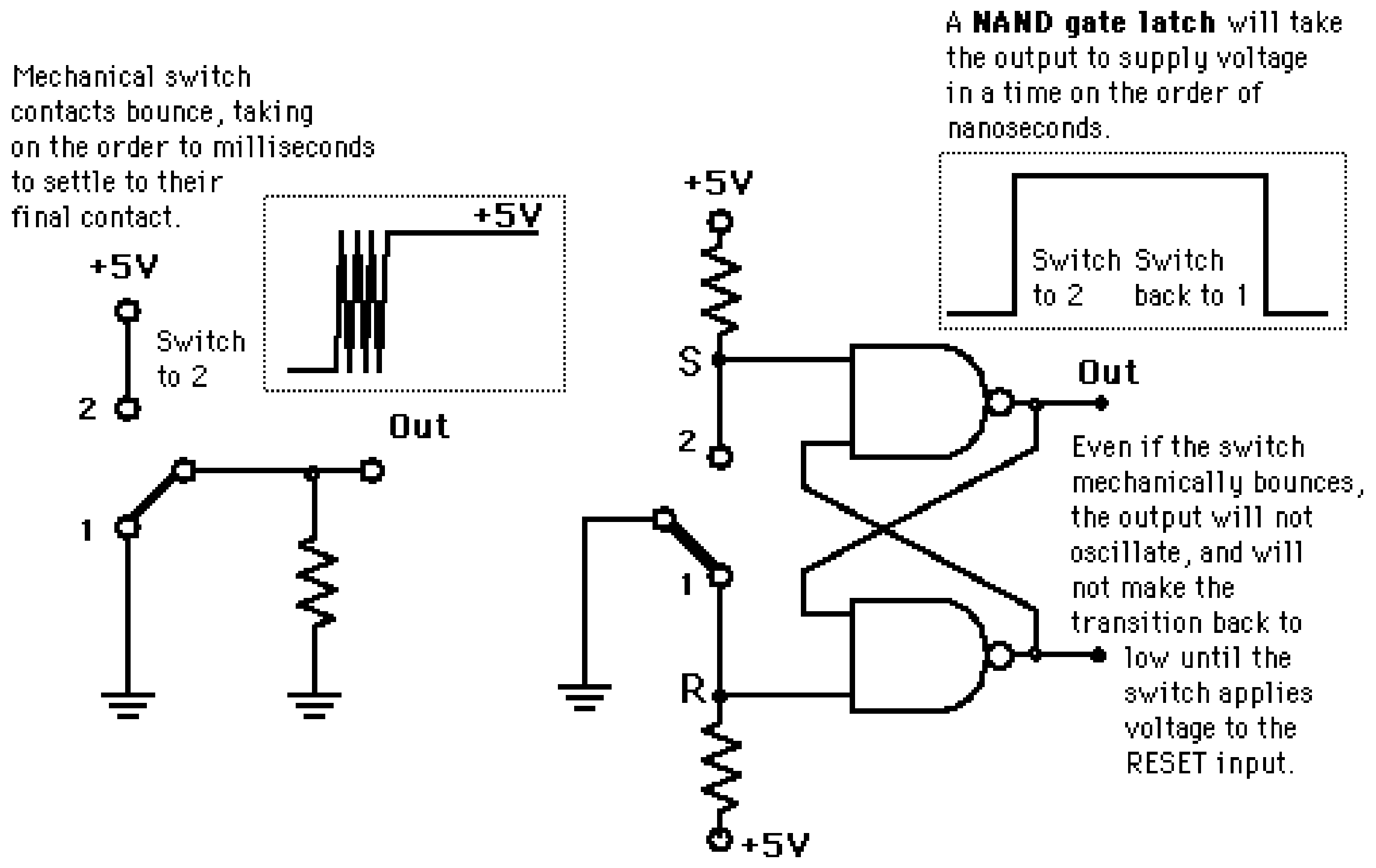
Truth Table



Set	Reset	Output
1	1	No change*
0	1	Q=1
1	0	Q=0
0	0	Invalid state

* can be used for data storage

Chatterless Switch

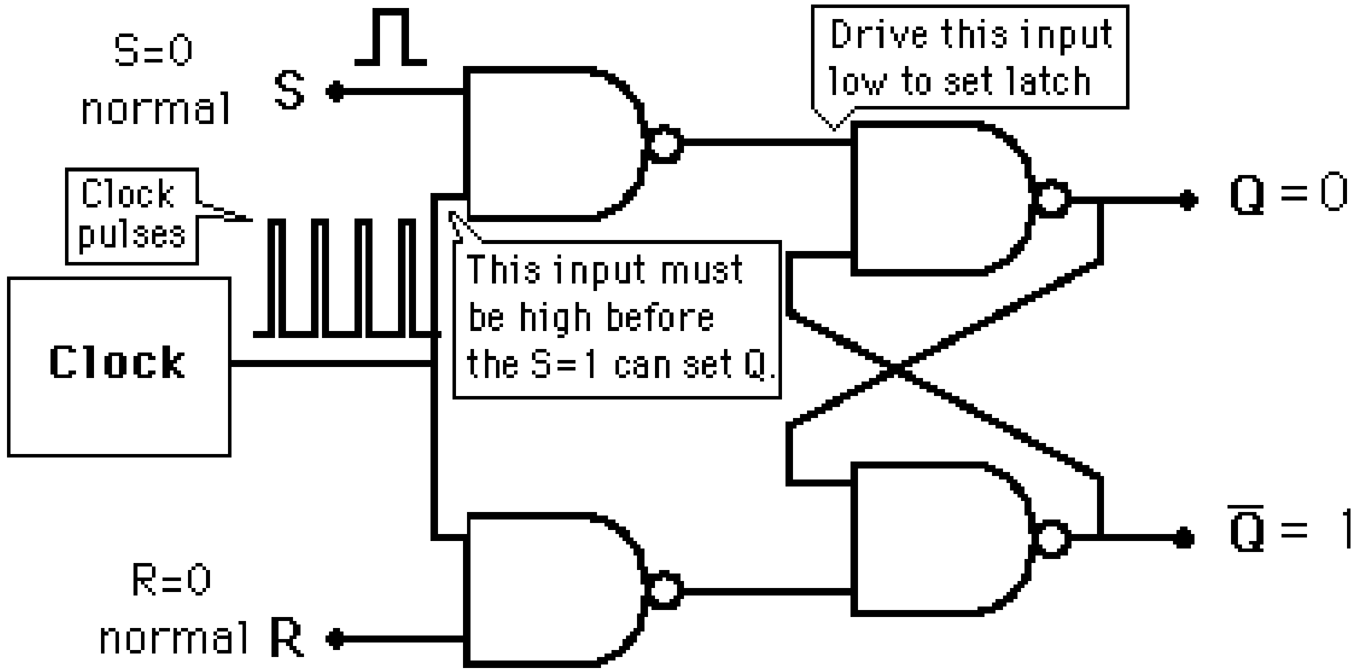


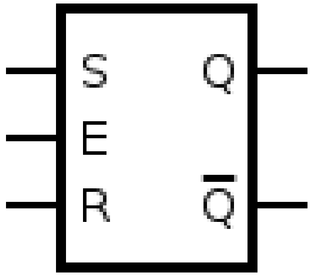
Mechanical switch contacts bounce, taking on the order to milliseconds to settle to their final contact.

A **NAND gate latch** will take the output to supply voltage in a time on the order of nanoseconds.

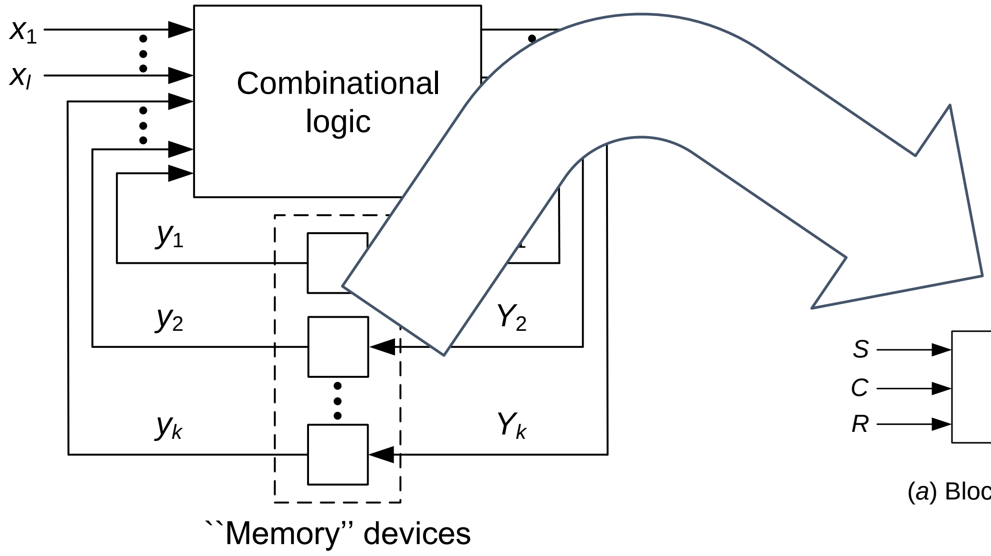
Even if the switch mechanically bounces, the output will not oscillate, and will not make the transition back to low until the switch applies voltage to the RESET input.

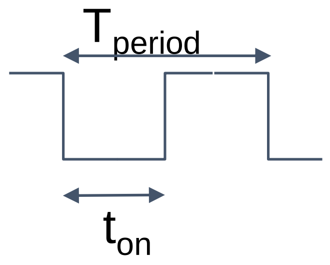
Clocked/Gated SR Latch

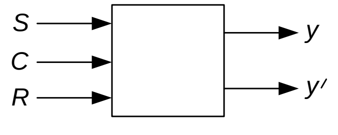




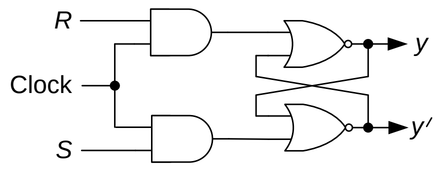
Timing the SR latch







(*a*) Block diagram.

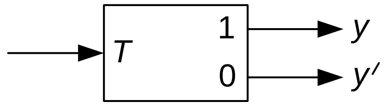


(*b*) Logic diagram.

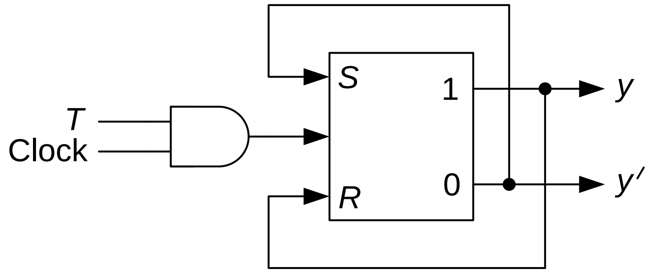
- A clock is a periodic signal that is used to keep time in sequential circuits.
- Duty Cycle is the ration of t_{on}/T_{period}
- We want to keep t_{on} small so that in the same clock pulse only a single computation is performed.
- We want to keep T_{period} sufficient so that there is enough time for the next input to be computed.

Trigger or *T* Latch

Value 1 applied to its input triggers the latch to change state



(*a*) Block diagram.



(*b*) Deriving the T latch from the clocked SR latch.

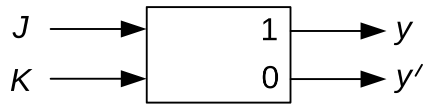
Excitations requirements:

Circuit change From:	To:	*Required* value *T*
0	0	0
0	1	1
1	0	1
1	1	0

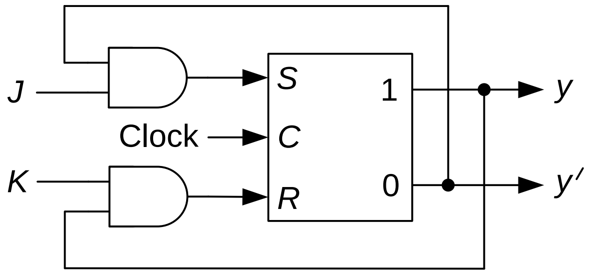
$$y(t+1) = Ty'(t) + T'y(t)$$
$$= T \oplus y(t)$$

The *JK* Latch

Unlike the *SR* latch, *J* = *K* = 1 is permitted: when it occurs, the latch acts like a trigger and switches to the complement state



(*a*) Block diagram.



(*b*) Constructing the JK latch from the clocked SR latch.

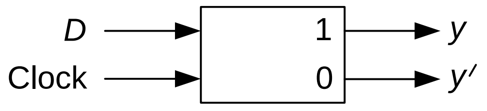
Excitation requirements:

Circuit change		*Required value*	
From:	*To:*	*J*	*K*
0	0	0	–
0	1	1	–
1	0	–	1
1	1	–	0

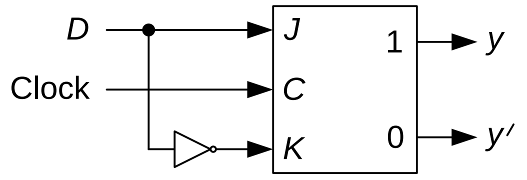
The *D* Latch

The next state of the *D* latch is equal to its present excitation:

$$y(t+1) = D(t)$$



(*a*) Block diagram.



(*b*) Transforming the JK latch to the D latch.

Clock Timing

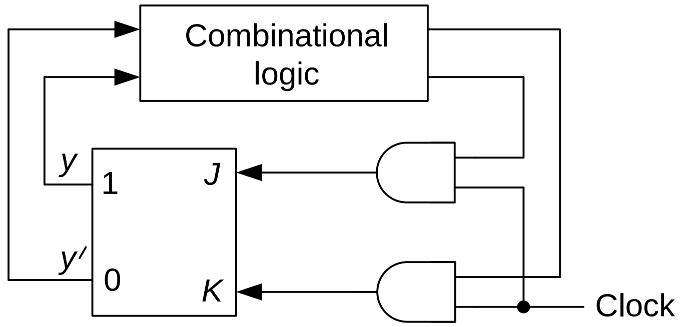
Clocked latch: changes state only in synchronization with the clock pulse and no more than once during each occurrence of the clock pulse

Duration of clock pulse: determined by circuit delays and signal propagation time through the latches

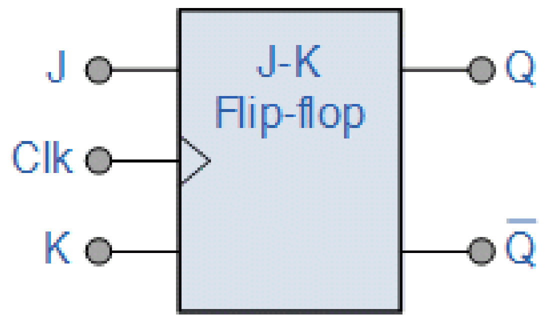
- Must be long enough to allow latch to change state, and
- Short enough so that the latch will not change state twice due to the same excitation

Excitation of a *JK* latch within a sequential circuit:

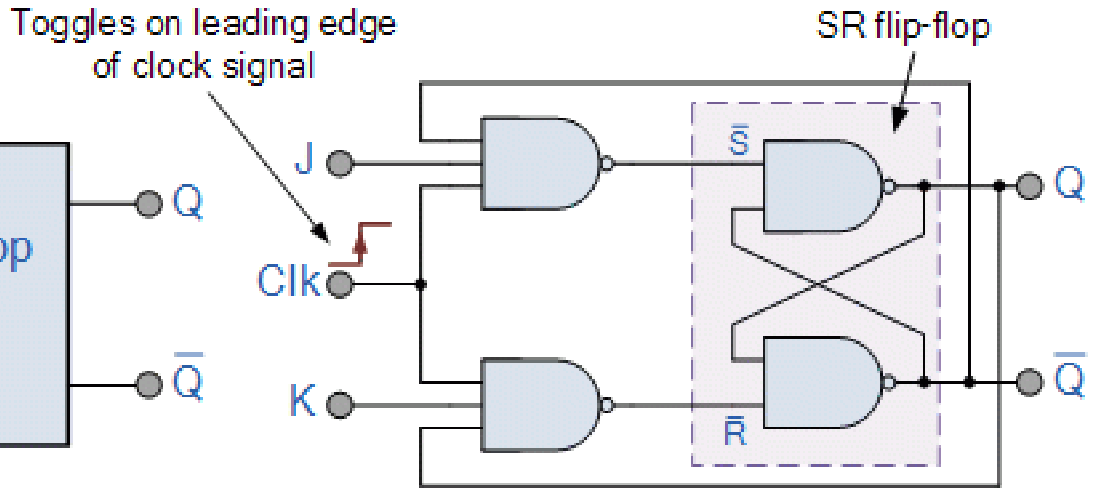
- Length of the clock pulse must allow the latch to generate the *y*'s
- But should not be present when the values of the *y*'s have propagated through the combinational circuit



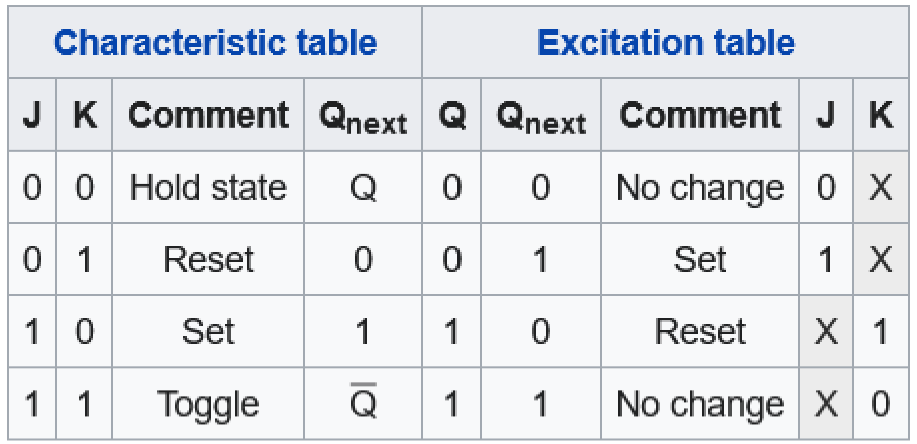
JK Flip Flop



Symbol

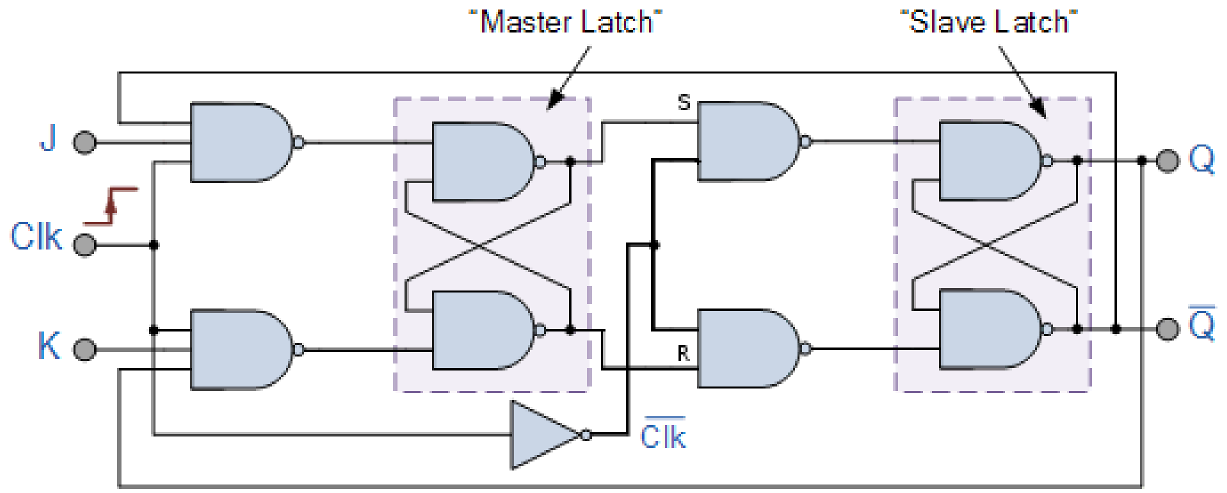


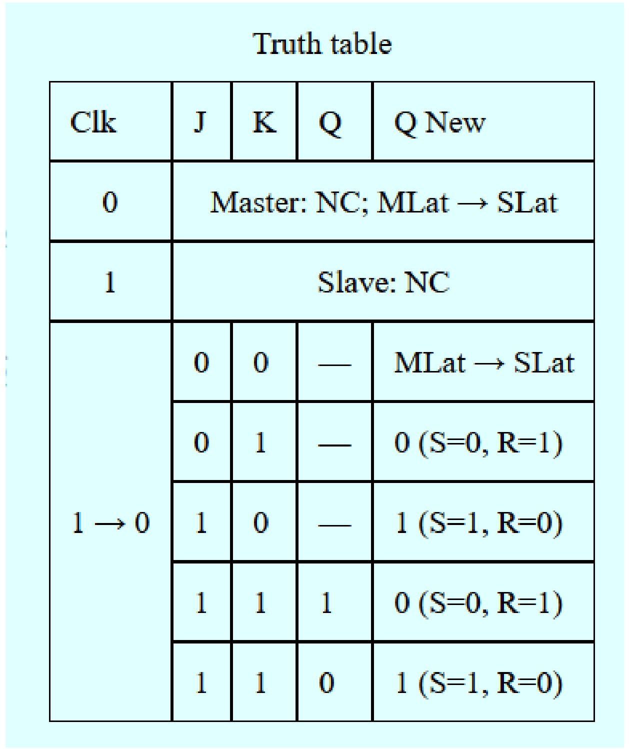
Circuit



Characteristic table				Excitation table				
J	K	Comment	Q_{next}	Q	Q_{next}	Comment	J	K
0	0	Hold state	Q	0	0	No change	0	X
0	1	Reset	0	0	1	Set	1	X
1	0	Set	1	1	0	Reset	X	1
1	1	Toggle	$\overline{Q}$	1	1	No change	X	0

Master Slave JK Flip Flop

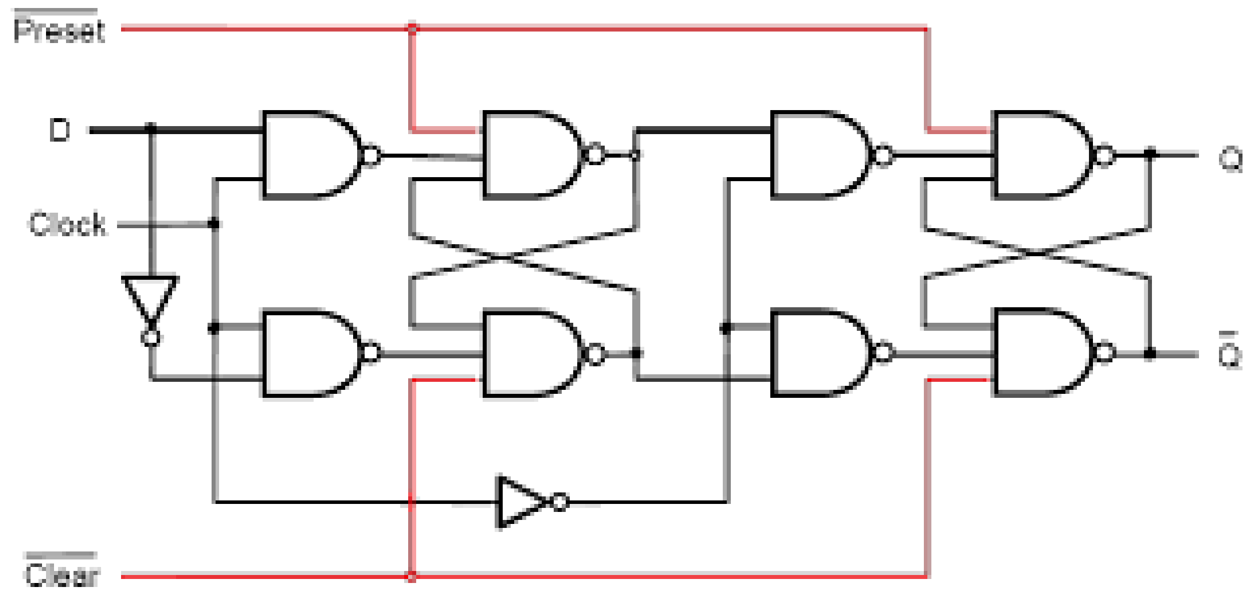




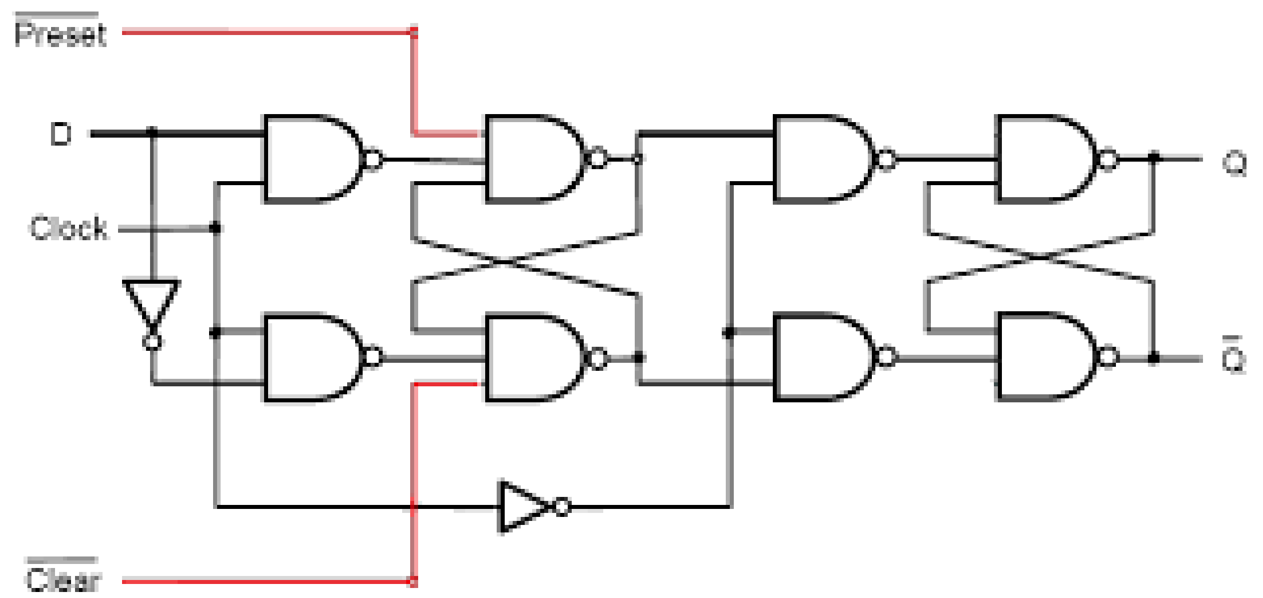
Truth table

Clk	J	K	Q	Q New
0	Master: NC; MLat → SLat			
1	Slave: NC			
1 → 0	0	0	—	MLat → SLat
	0	1	—	0 (S=0, R=1)
	1	0	—	1 (S=1, R=0)
	1	1	1	0 (S=0, R=1)
	1	1	0	1 (S=1, R=0)

D Flip Flop with Asynchronous Preset Clear



D Flip Flop with Synchronous Preset Clear



Residual problem with above DFF

- Let the gate delay be Δ
- Let D=0, Clk=1
- Let D=1 and just before Δ time, Clk=0
- Master latch has the invalid input combination of 00
- Before valid inputs could appear at the steering gates of the master latch, Clk=0
- Now, invalid input combination of 11 is presented to the slave steering gates, with Clock'=1
- Slave latch has the invalid input combination of 00
- Output of DFF is indeterminate

- Problem may be avoided if D remains steady when Clk=1, allowed to change only when Clk=0

Setup and hold times

- Setup time: It is defined as the minimum amount of time *before* the active clock edge by which the data must be stable for it to be latched correctly; any violation in this required time causes incorrect data to be captured and is known as a setup violation

Time available for data at D2 to reach D1 after active clock edge

$T_{clk} + T_{skew} - T_{setup}$

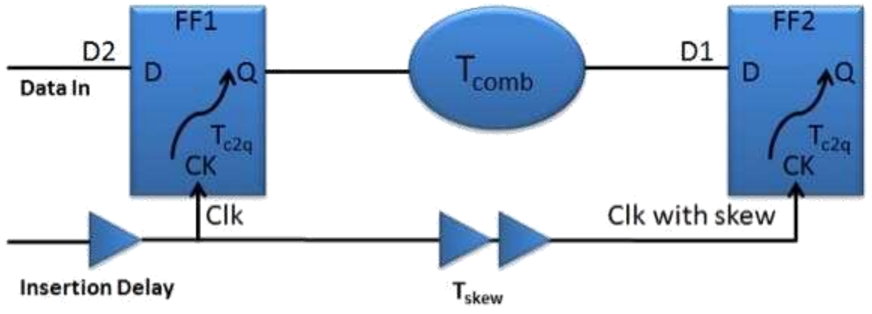
Time needed for date to reach D1 from D2 after active clock edge

$T_{c2q} + T_{comb}$

Resulting constraint

$T_{c2q} + T_{comb} \leq T_{clk} + T_{skew} - T_{setup}$

$T_{c2q} + T_{comb} + T_{setup} \leq T_{clk} + T_{skew}$



Hold time

It is defined as the minimum amount of time *after* the active clock edge by which the data must be stable for it to be latched correctly; any violation in this required time causes incorrect data to be captured and is known as a hold violation

Minimum time for data at D2 to reach D1 after active clock edge

$T_{c2q} + T_{comb}$

Time for data to remain steady at D1 after active clock edge

$T_{skeq} + T_{hold}$

Resulting constraint

$T_{c2q} + T_{comb} \geq T_{skeq} + T_{hold}$

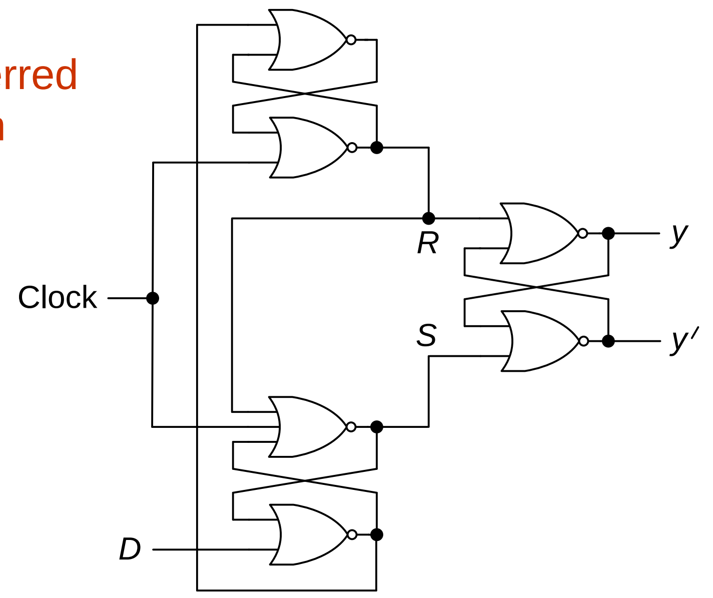
Edge-triggered Flip-flop

Positive (negative) edge-triggered *D* flip-flip: stores the value at the *D* input when the clock makes a 0 -> 1 (1 -> 0) transition

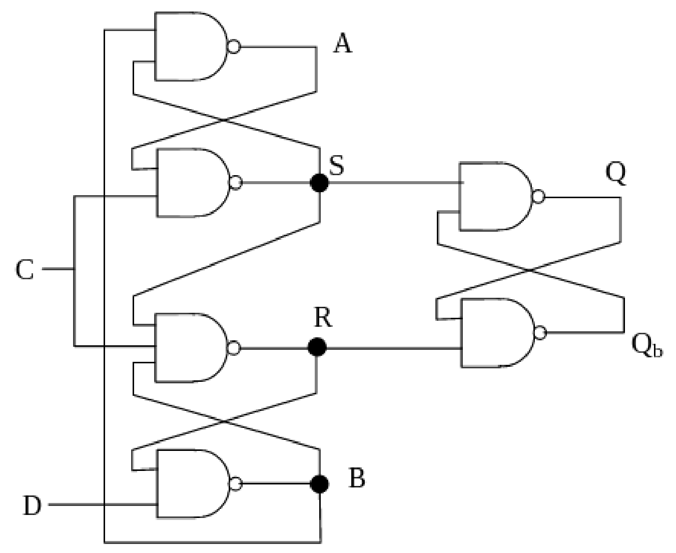
- Any change at the *D* input after the clock has made a transition does not have any effect on the value stored in the flip-flop

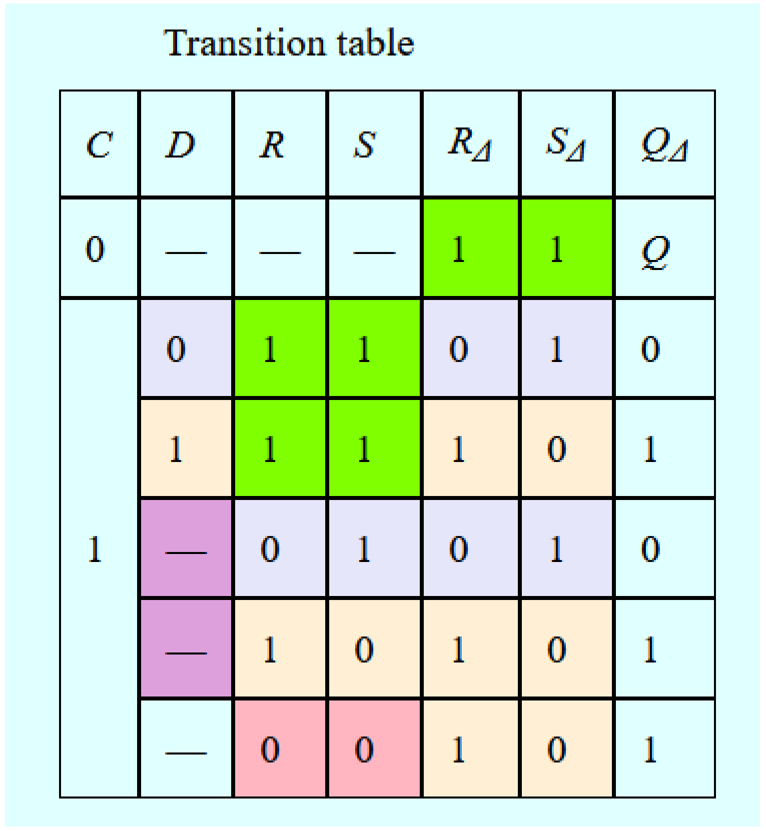
A negative edge-triggered *D* flip-flop:

- When the clock is high, the output of the bottommost (topmost) NOR gate is at *D'* (*D*), whereas the *S*-*R* inputs of the output latch are at 0, causing it to hold previous value
- When the clock goes low, the value from the bottommost (topmost) NOR gate gets transferred as *D* (*D'*) to the *S* (*R*) input of the output latch
  - Thus, output latch stores the value of *D*
- If there is a change in the value of the *D* input after the clock has made its transition, the bottommost NOR gate attains value 0
  - However, this cannot change the *SR* inputs of the output latch



Positive Edge Triggered DFF



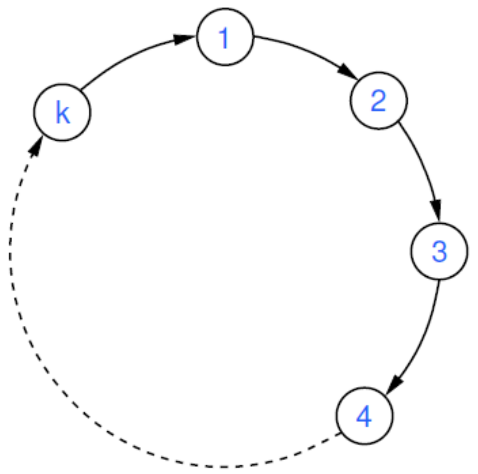


Transition table

C	D	R	S	R_Δ	S_Δ	Q_Δ
0	—	—	—	1	1	Q
1	0	1	1	0	1	0
	1	1	1	1	0	1
	—	0	1	0	1	0
	—	1	0	1	0	1
	—	0	0	1	0	1

- $S_\Delta = (C \cdot A)' = C' + A' = C' + B \cdot S = C' + (D' + R') \cdot S$
- $R_\Delta = (S \cdot C \cdot B)' = S' + C' + B' = S' + C' + R \cdot D$
- As long as $C = 0$, $R = S = 1$ and the output latch retains its older value
- When C changes from 0 to 1, R and S change as follows:
  - If $\boldsymbol{D}$**=0**: $R_\Delta = 0$, $S_\Delta = 1$ (from $R = S = 1$) and $\boldsymbol{Q}$**=0**
  - If $\boldsymbol{D}$**=1**: $R_\Delta = 1$, $S_\Delta = 0$ (from $R = S = 1$) and $\boldsymbol{Q}$**=1**
  - Thereafter, R and S remain stable, while C=1, irrespective of changes in D
- The invalid state of $R = S = 0$ is never reached
- If some how $R = S = 0$, state immediately changes to $R = 1$ and $S = 0$

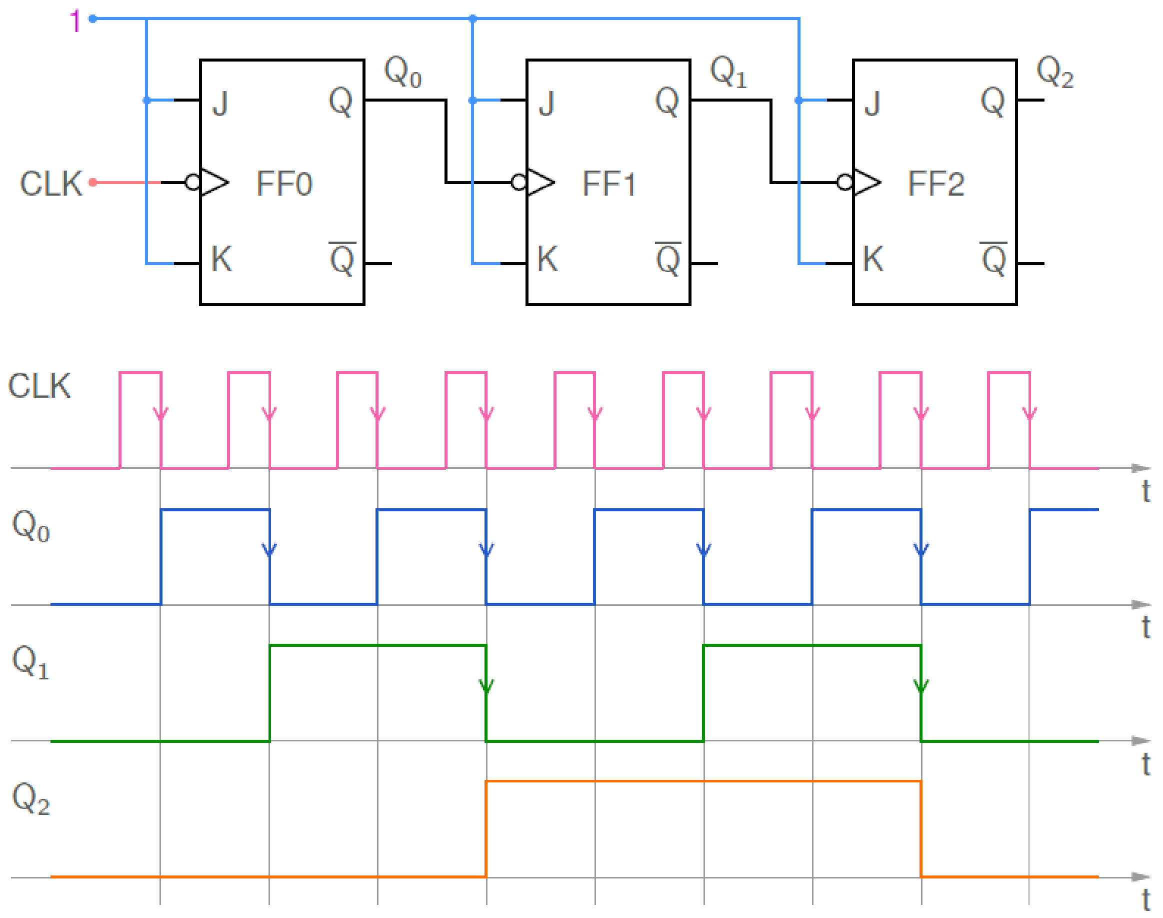
Counters



State transition diagram

- A counter with k states is called a modulo-k (mod-k) counter.
- A counter can be made with flip-flops, each flip-flop serving as a memory element with two states (0 or 1).
- If there are N flip-flops in a counter, there are 2^N possible states (since each flip-flop can have $Q = 0$ or $Q = 1$). It is possible to exclude some of these states.
  $\rightarrow$ N flip-flops can be used to make a mod-k counter with $k \leq 2^N$.
- Typically, a reset facility is also provided, which can be used to force a certain state to initialize the counter.

Binary Ripple Counter (Asynchronous)

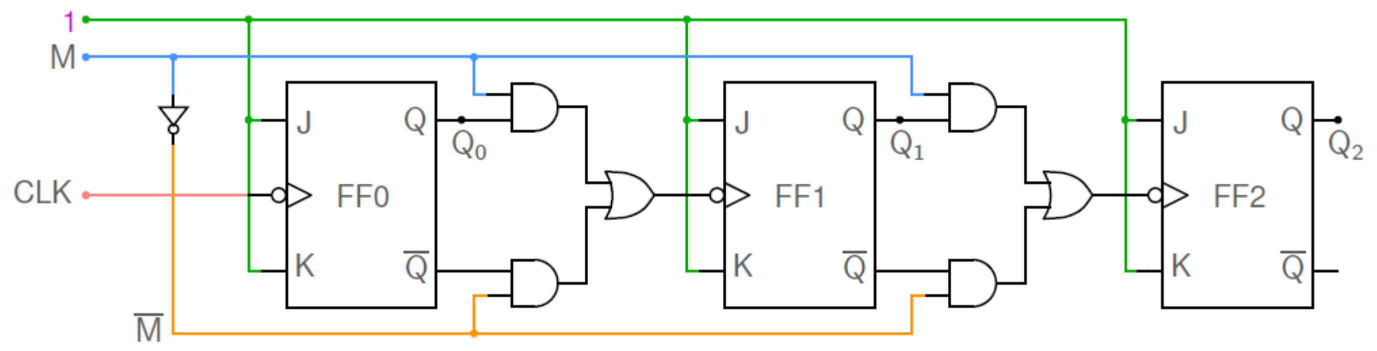


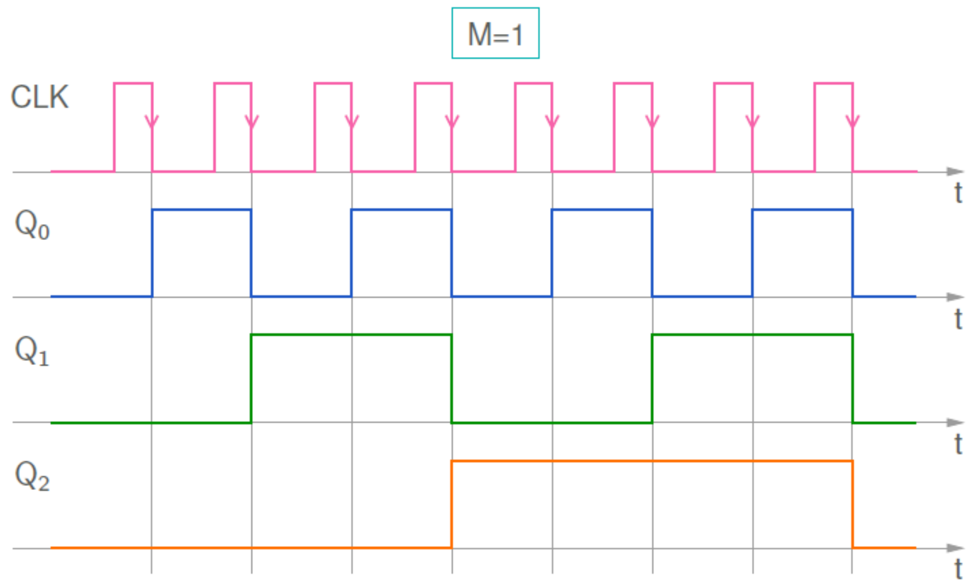
Q_2	Q_1	Q_0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1
0	0	0

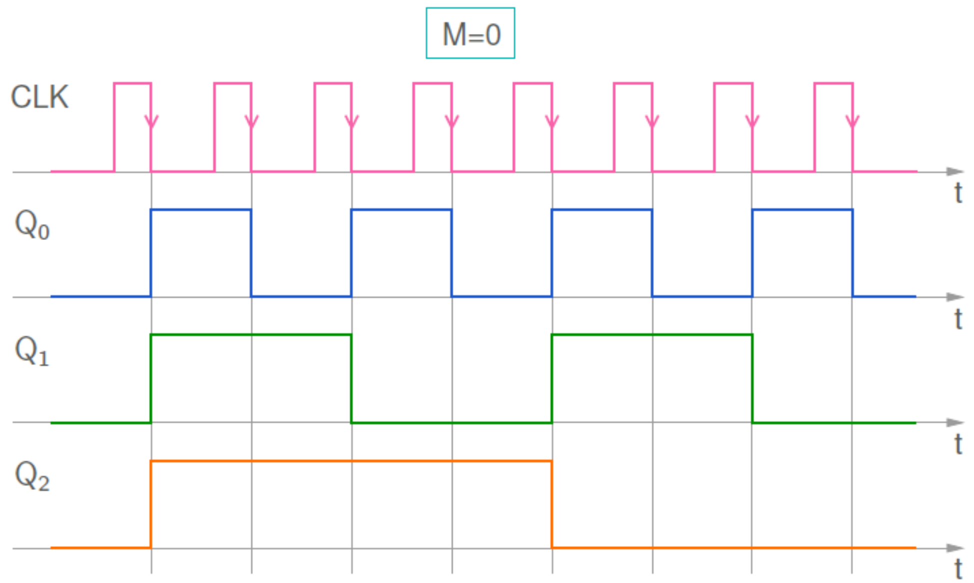
repeats

* If positive edge-triggered flip-flops are used, we get a binary *down* counter (counting down from 111 to 000).

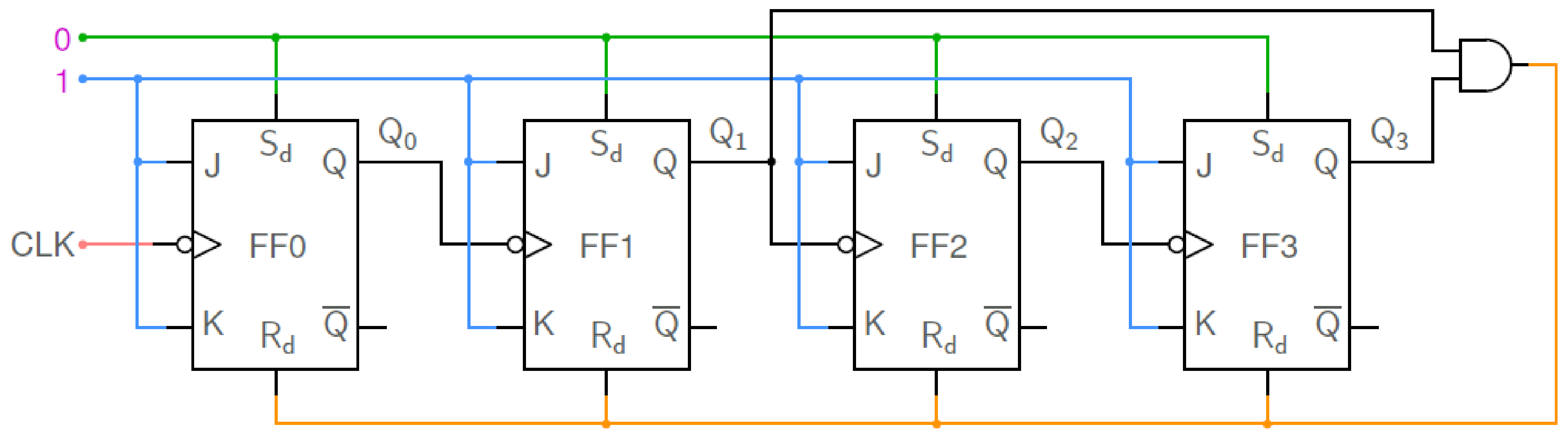
Up/Down Binary Ripple Counter



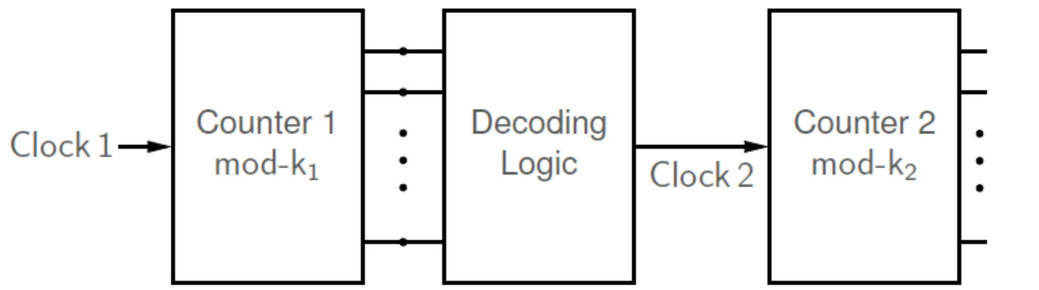


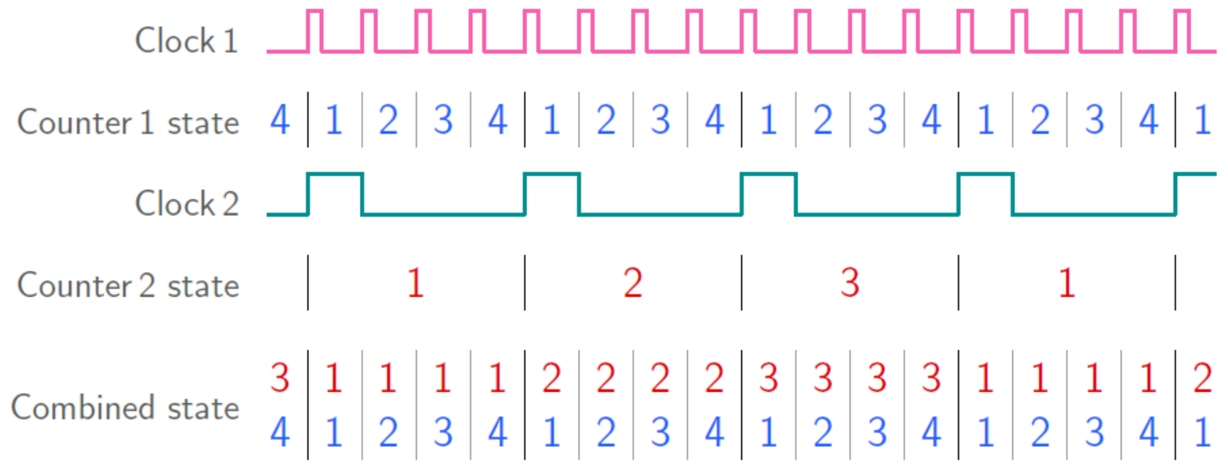


Decade Counter



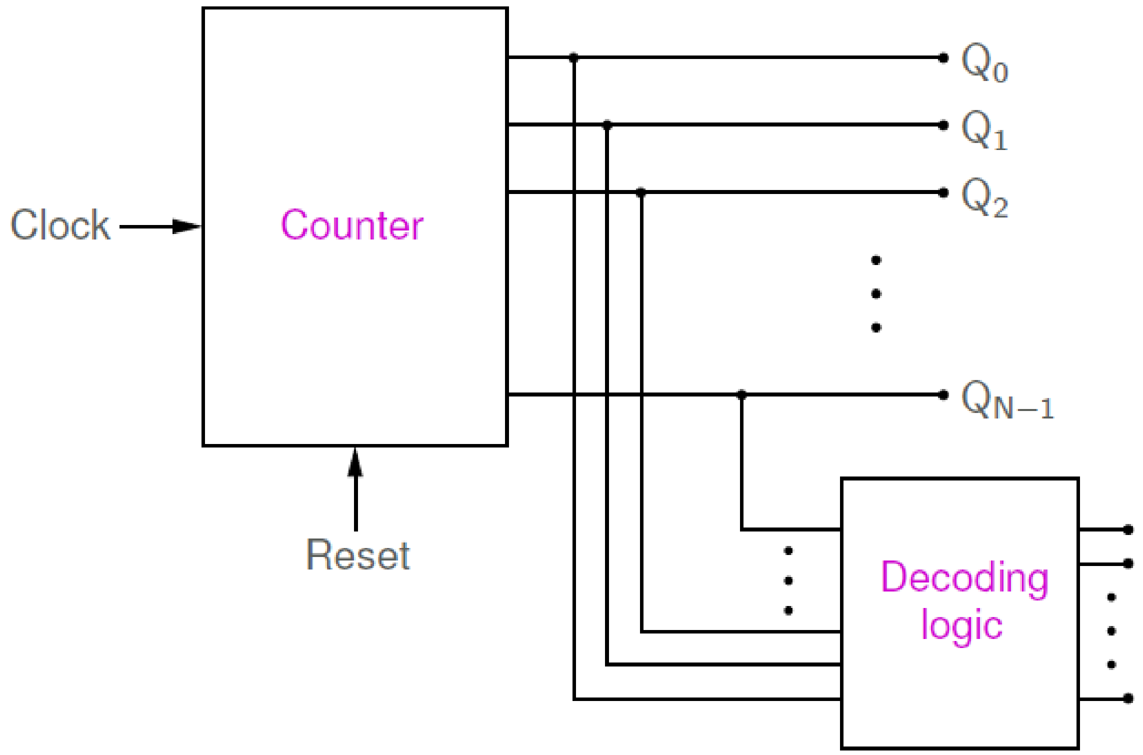
Combination of Counters



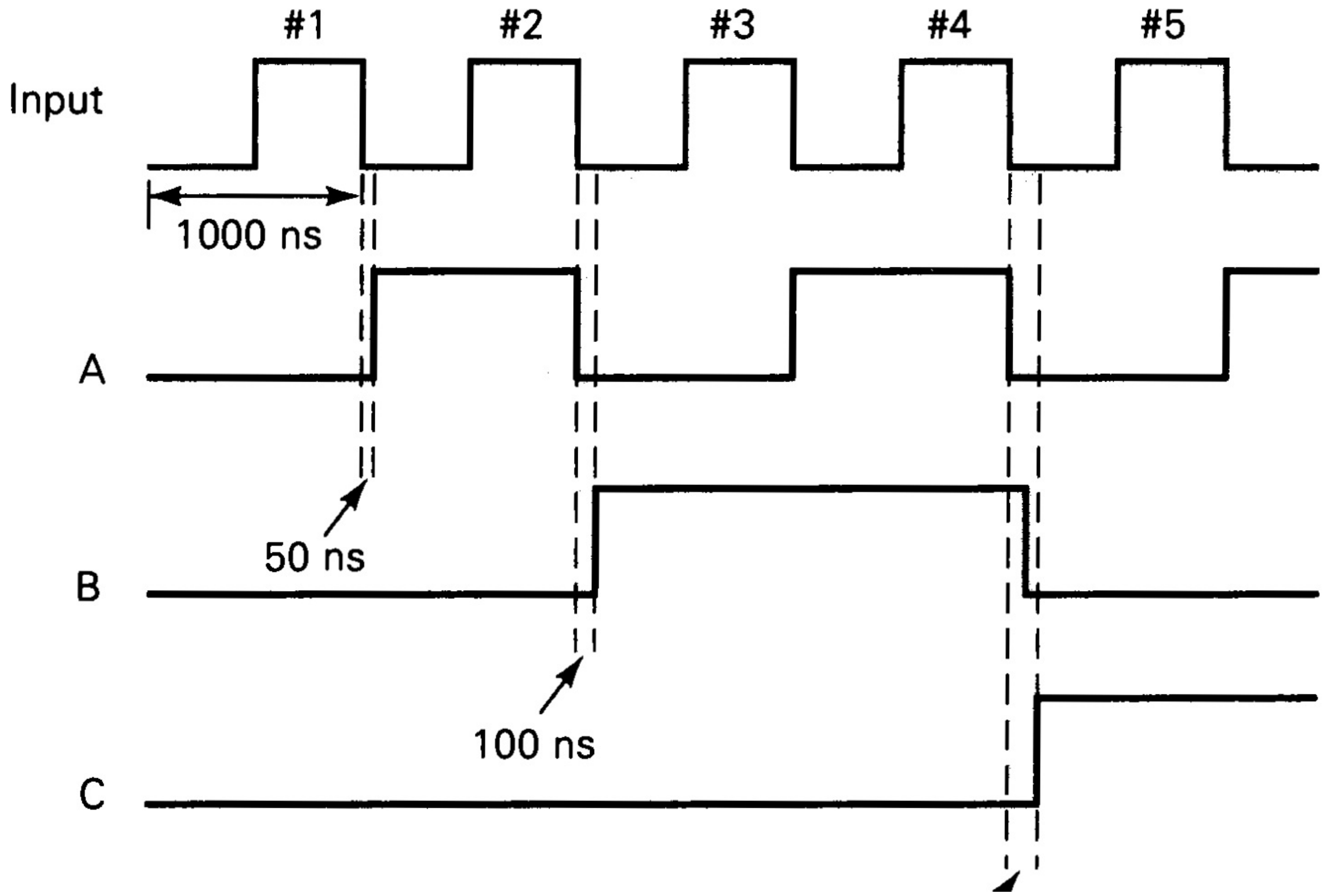


$\rightarrow$ the combined counter is a mod-k_1k_2 counter.

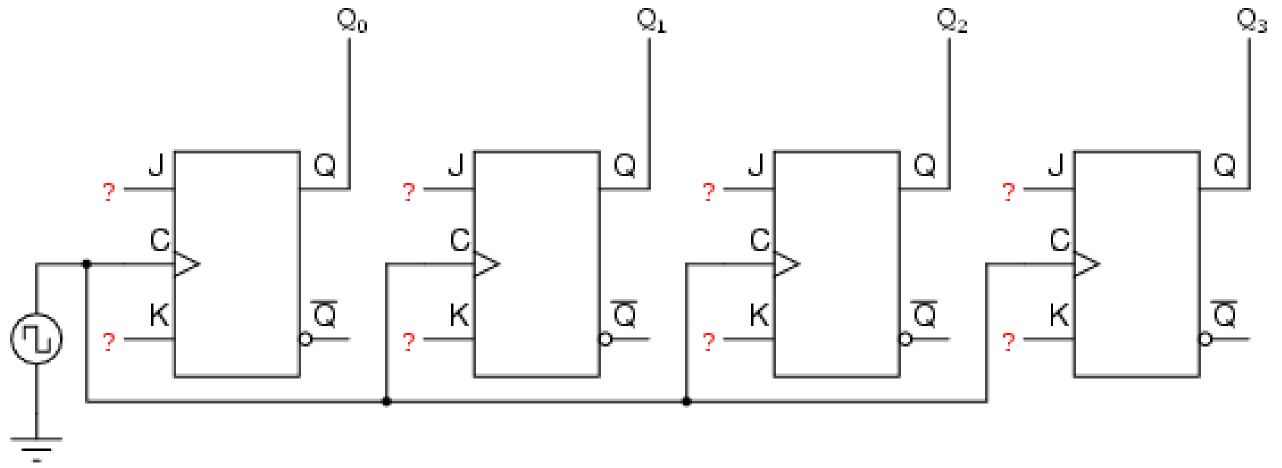
Counter Applications

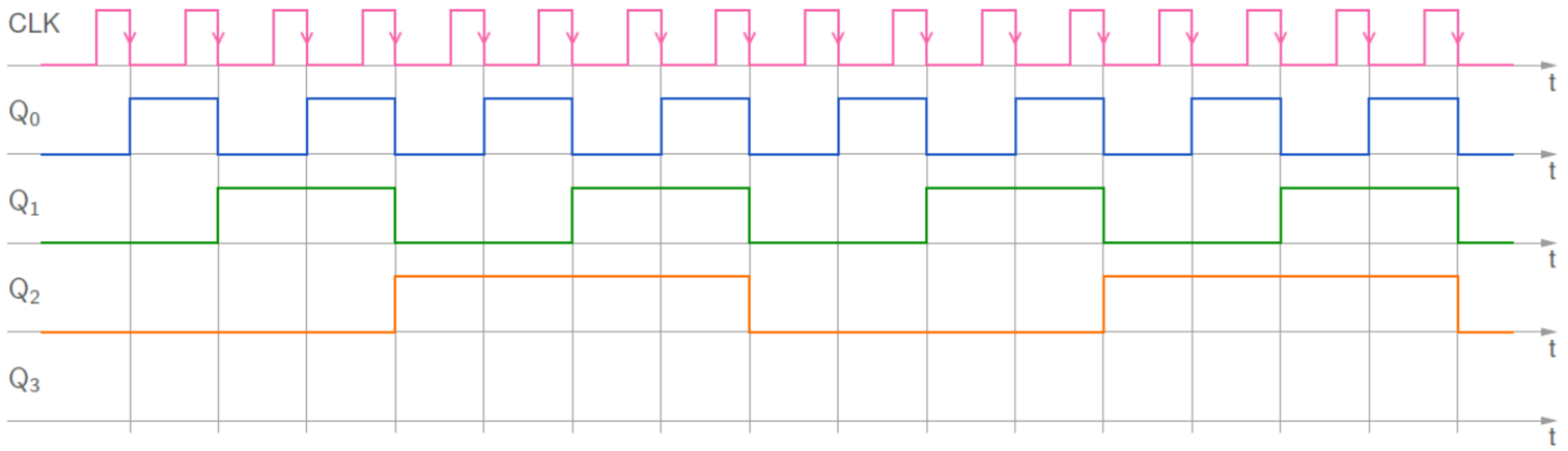


Glitch (Temporary Undesired States)



Synchronous Counters

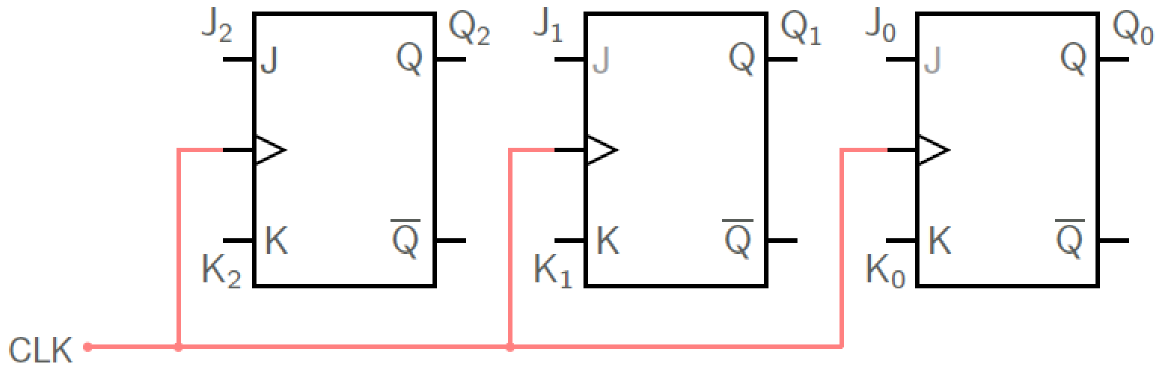




Synchronous Counter using JK Flip Flop

state	Q_2	Q_1	Q_0
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
1	0	0	0

repeats



CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X
↑	0	1	1	X
↑	1	0	X	1
↑	1	1	X	0

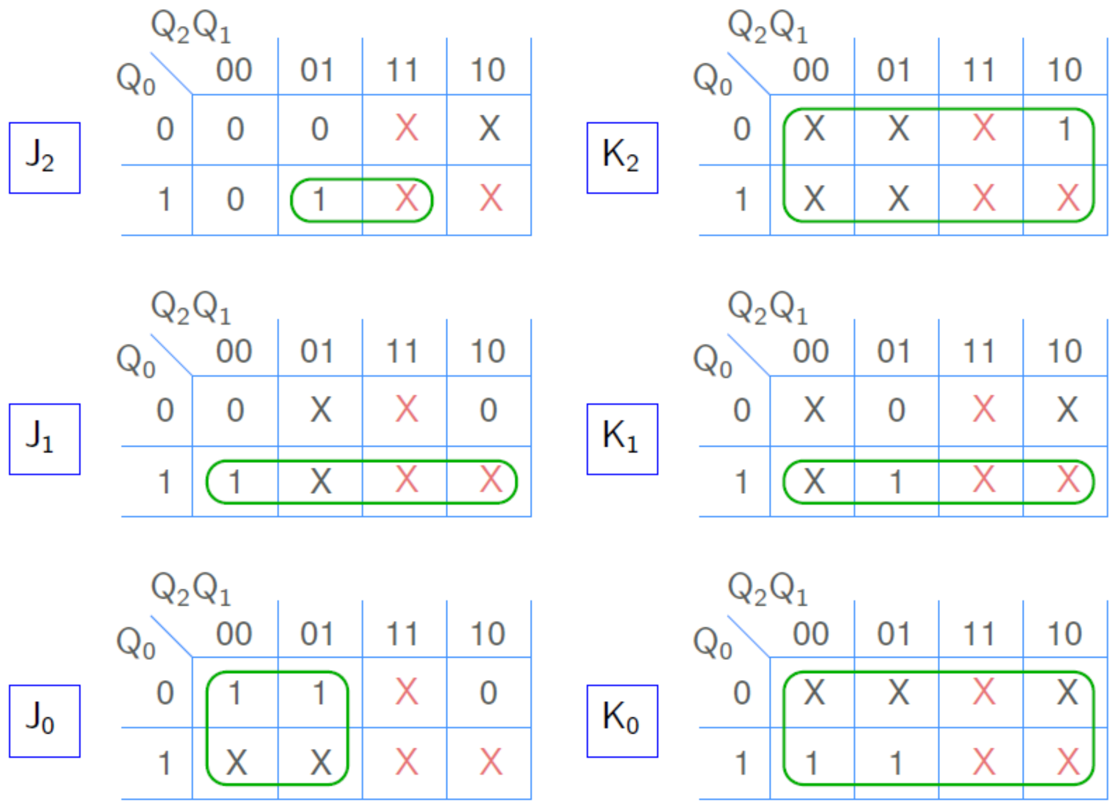
Design a synchronous mod-5 counter with the given state transition table.

Outline of method:

* State 1 $\rightarrow$ State 2 means
  Q_2: $0 \rightarrow 0$,
  Q_1: $0 \rightarrow 0$,
  Q_0: $0 \rightarrow 1$.
* Refer to the right table. For Q_2: $0 \rightarrow 0$, we must have $J_2 = 0$, $K_2 = X$, and so on.

Synchronous Counter

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X	X	1
3	0	1	0	0	X	X	0	1	X
4	0	1	1	1	X	X	1	X	1
5	1	0	0	X	1	0	X	0	X
1	0	0	0						



J_2

Q_0 \ Q_2Q_1	00	01	11	10
0	0	0	X	X
1	0	1	X	X

K_2

Q_0 \ Q_2Q_1	00	01	11	10
0	X	X	X	1
1	X	X	X	X

J_1

Q_0 \ Q_2Q_1	00	01	11	10
0	0	X	X	0
1	1	X	X	X

K_1

Q_0 \ Q_2Q_1	00	01	11	10
0	X	0	X	X
1	X	1	X	X

J_0

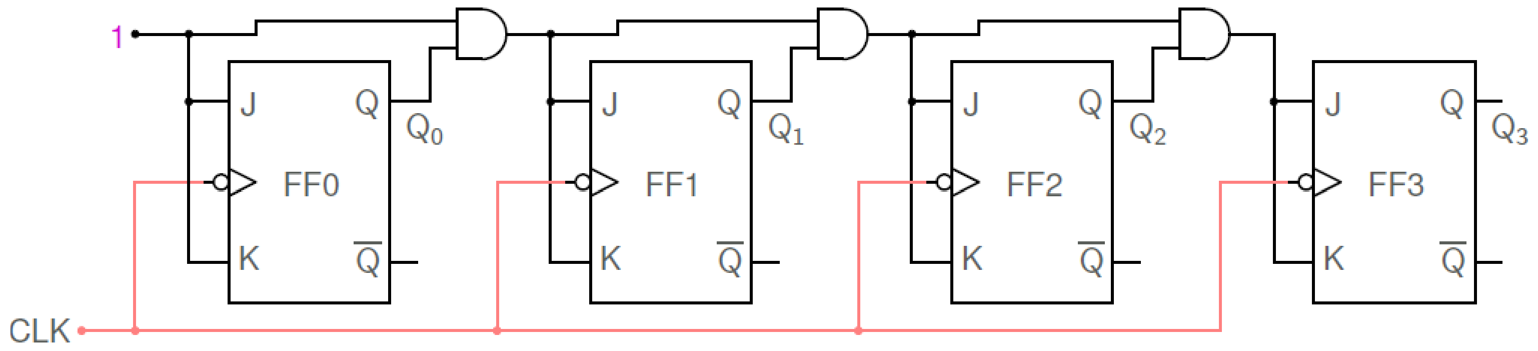
Q_0 \ Q_2Q_1	00	01	11	10
0	1	1	X	0
1	X	X	X	X

K_0

Q_0 \ Q_2Q_1	00	01	11	10
0	X	X	X	X
1	1	1	X	X

$$J_2 = Q_1 Q_0,\ K_2 = 1,\ J_1 = Q_0,\ K_1 = Q_0,\ J_0 = \overline{Q_2},\ K_0 = 1$$

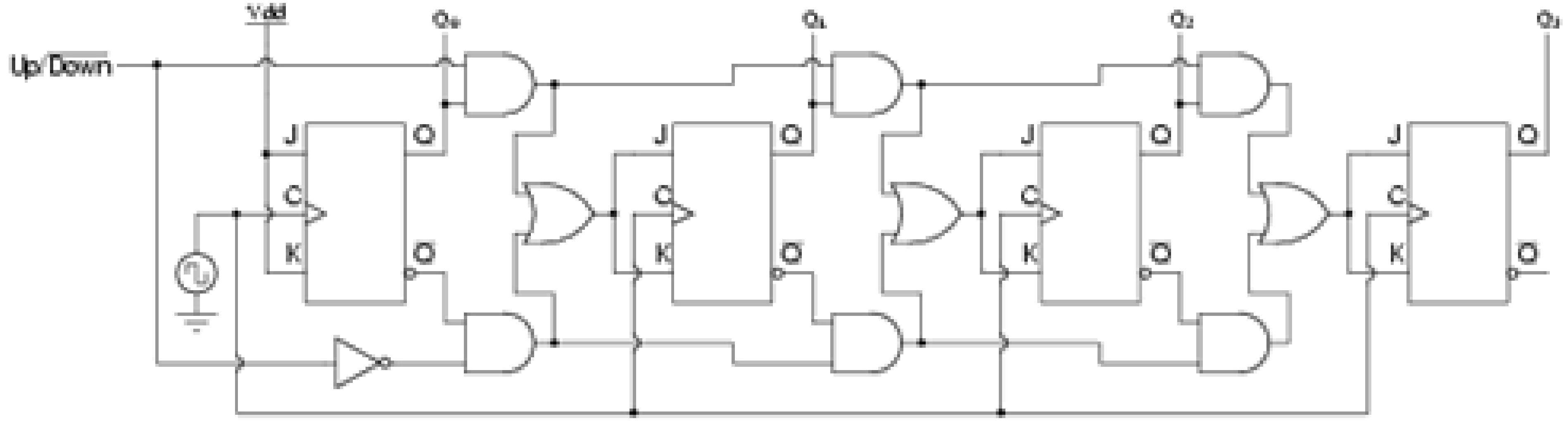
Synchronous Counter



* Since all flip-flops are driven by the same clock, the counter is called a "synchronous" counter.
* $J_0 = K_0 = 1$, $J_1 = K_1 = Q_0$, $J_2 = K_2 = Q_1 Q_0$, $J_3 = K_3 = Q_2 Q_1 Q_0$.
* FF0 toggles after every active edge.
  FF1 toggles if $Q_0 = 1$ (just before the active clock edge); else, it retains its previous state. (Similarly, for FF2 and FF3)

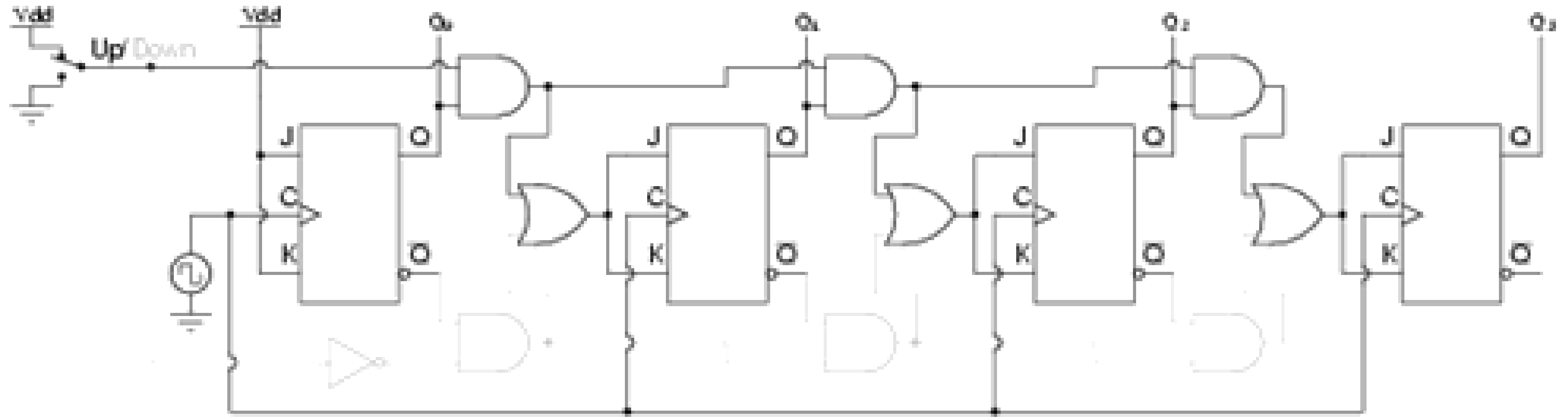
Up/Down Counter

A four-bit synchronous "up/down" counter



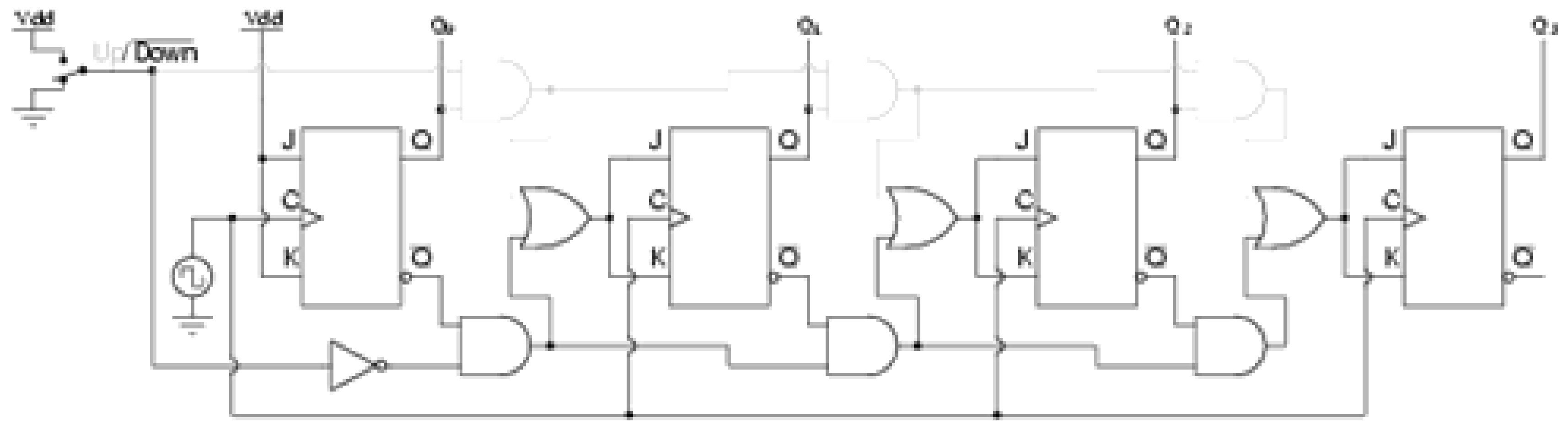
UP Counting

Counter in "up" counting mode

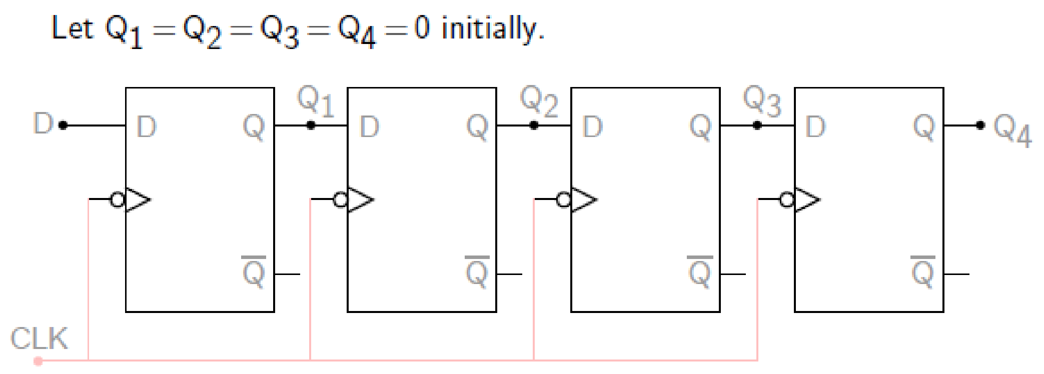


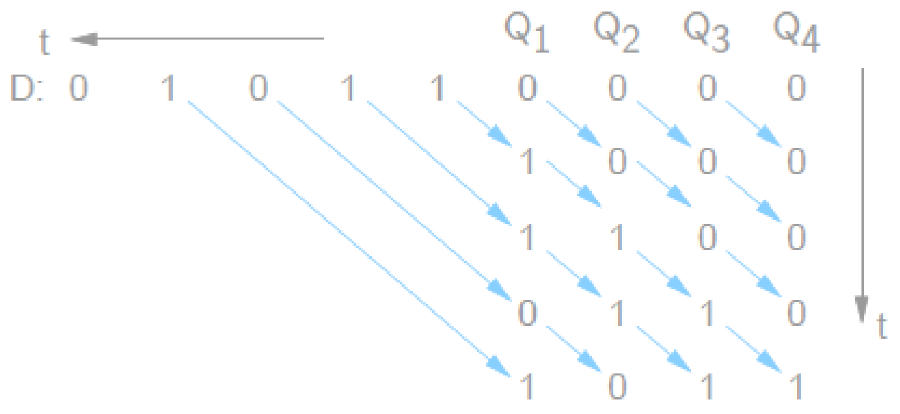
Down Counting

Counter in "down" counting mode



Shift Registers



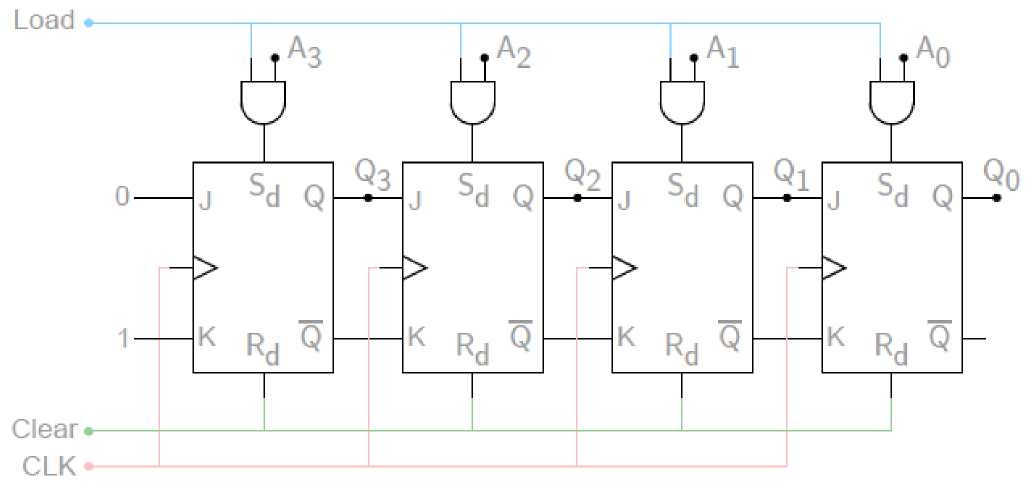


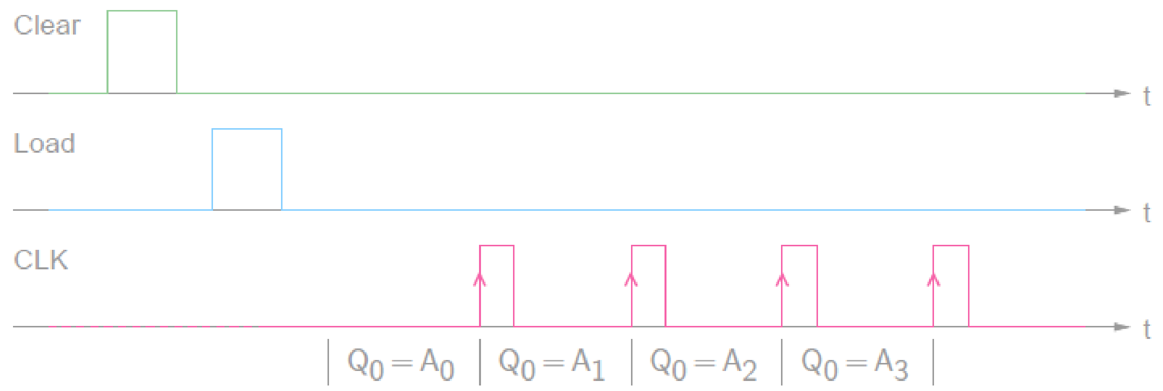
Q_1	Q_2	Q_3	Q_4
0	0	0	0
1	0	0	0
1	1	0	0
0	1	1	0
1	0	1	1

Serial In Parallel Out

Used in Computer Memory as Bit Arrays
Converts Serial Input to Parallel Output

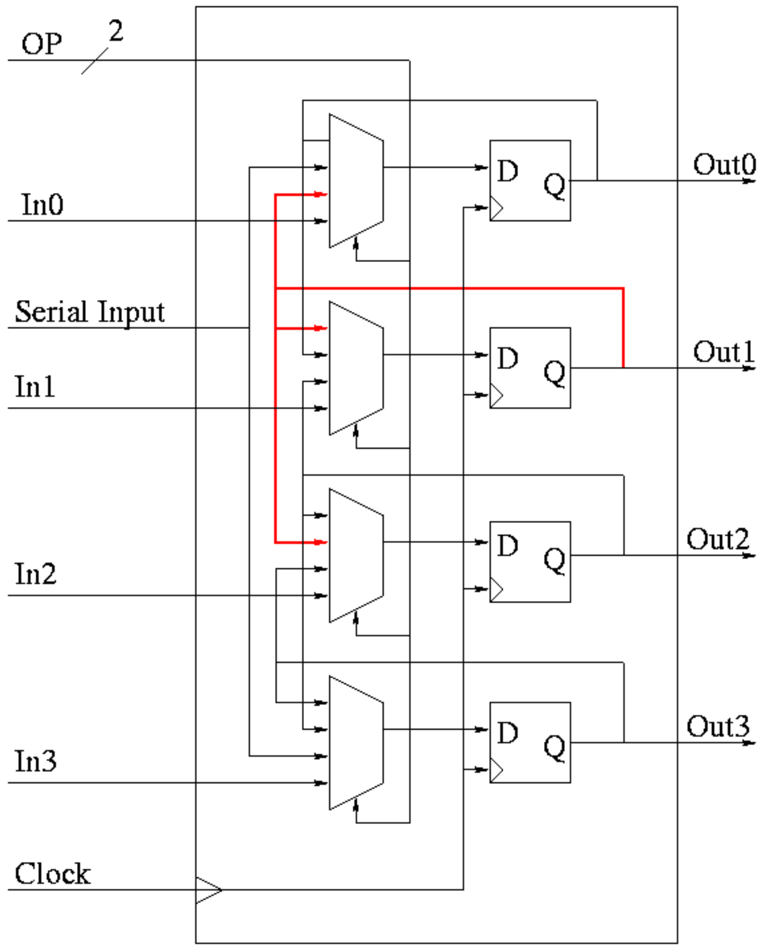
Parallel In Serial Out





- All flip-flops are cleared in the beginning (with $R_d = \text{Clear} = 1,\ S_d = 0$).
- When Load $= 1$, $S_d = A_i$, $R_d = 0 \rightarrow A_i$ gets loaded into the i^{th} flip-flop. (We will assume that CLK has been made 0 in this initial phase.)
- Subsequently, with every clock pulse, the data shifts right

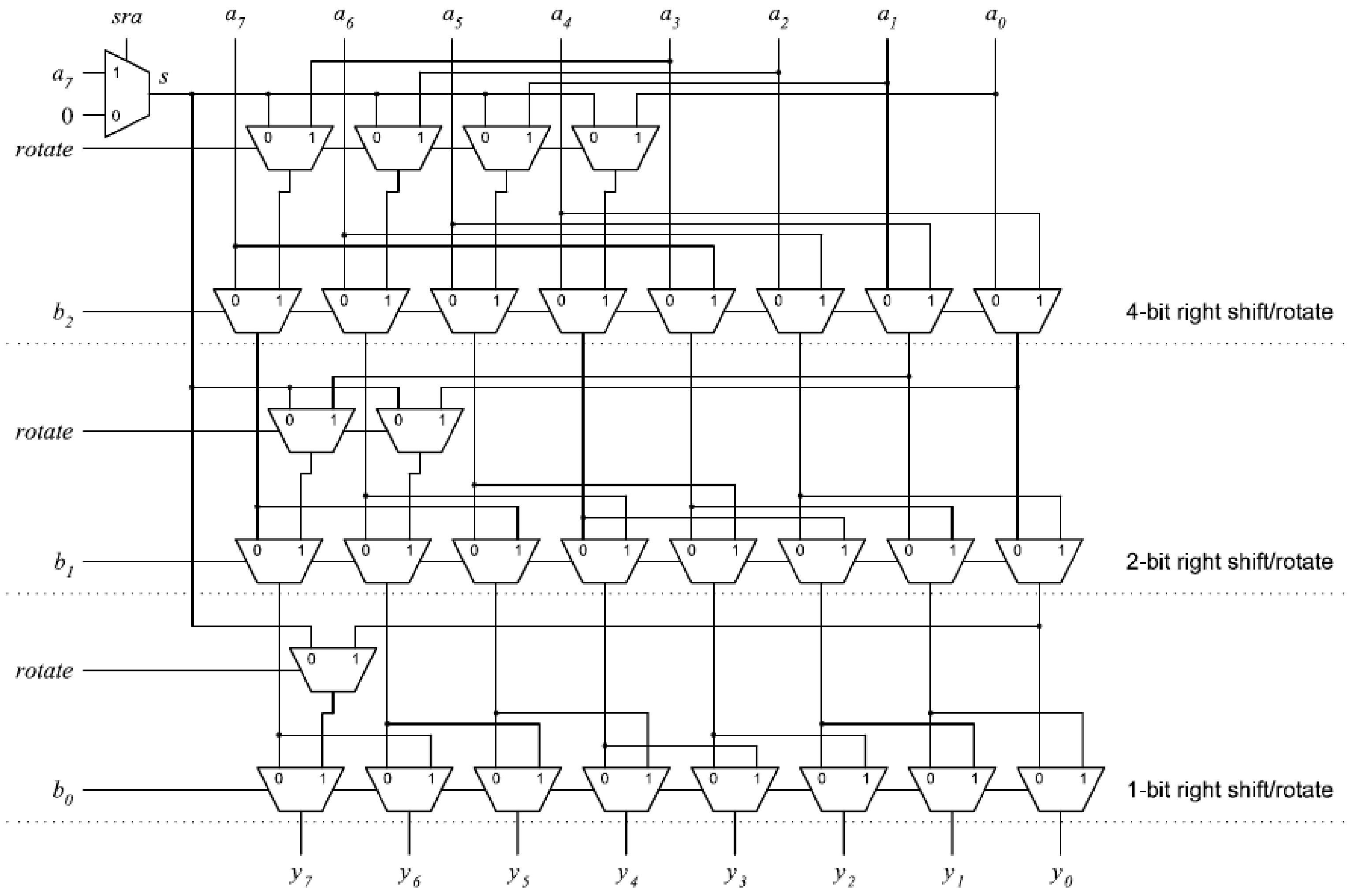
Bidirectional Shift Register with Parallel IO



4–bit bidirectional shift register with parallel I/O

OP=00: nop   OP=01: left–shift   OP=10: right–shift   OP=11: load

Barrel Shift/Rotate


sra
a_7
a_6
a_5
a_4
a_3
a_2
a_1
a_0
a_7
0
s
rotate
b_2
4-bit right shift/rotate
rotate
b_1
2-bit right shift/rotate
rotate
b_0
1-bit right shift/rotate
y_7
y_6
y_5
y_4
y_3
y_2
y_1
y_0