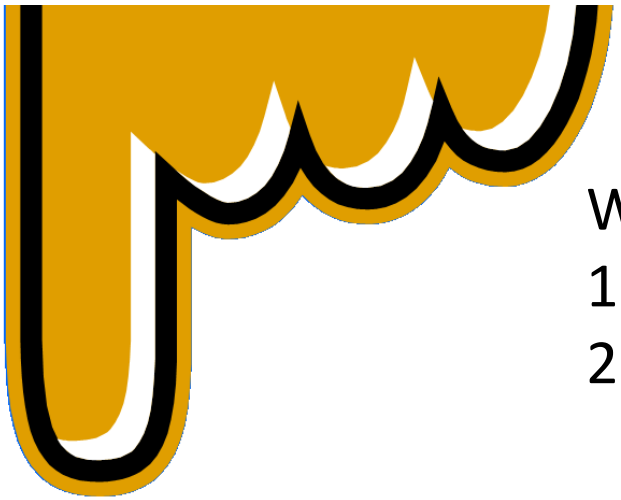
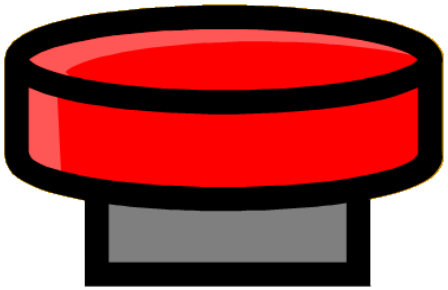
SEQUENTIAL CIRCUITS - I

Design a circuit to do this >>



When the button is pushed :

1) Turn On the light *if* it is Off
2) Turn Off the light *if* it is On



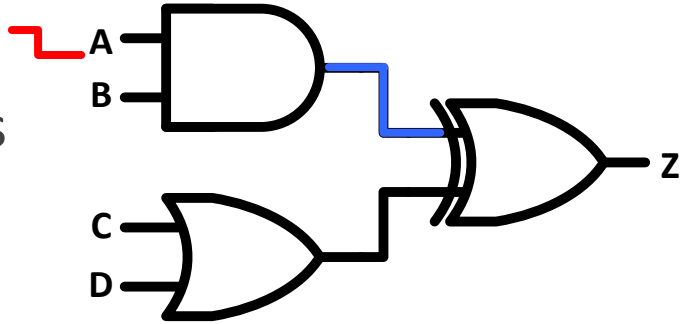
What is missing?

1) Remembering the previous state of the bulb ➔ *MEMORY*
2) Responding to an input *EVENT* (cf. input value)

Sequential Logic Circuits?

Combinational Logic Circuits:

- Outputs depend on current inputs

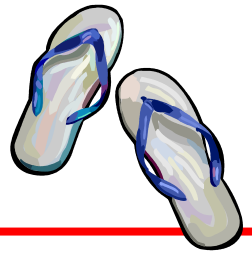


Sequential Logic Circuits:

- Outputs depend on ***current and previous*** inputs ➔ Memory!
- Requires separation of previous, current, future : **states**
- 2 Types of sequential circuits:

Synchronous	Asynchronous
Clocked: need a clock input	Unclocked
Responds to inputs at discrete time instants governed by a clock input	Responds whenever input signals change

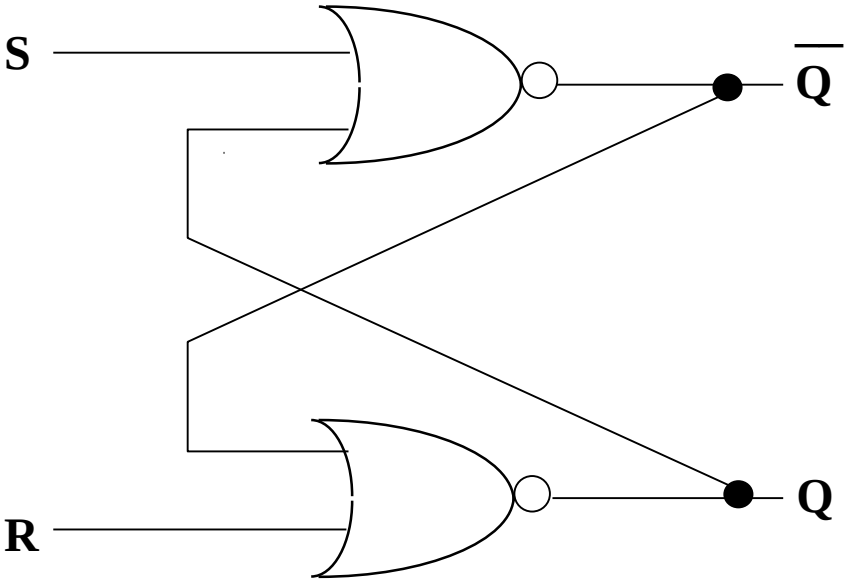
SR Flip-flop (FF)



The simplest memory element has two stable states :

Flip-Flop **(FF)** ➔ it can store 1 bit of information

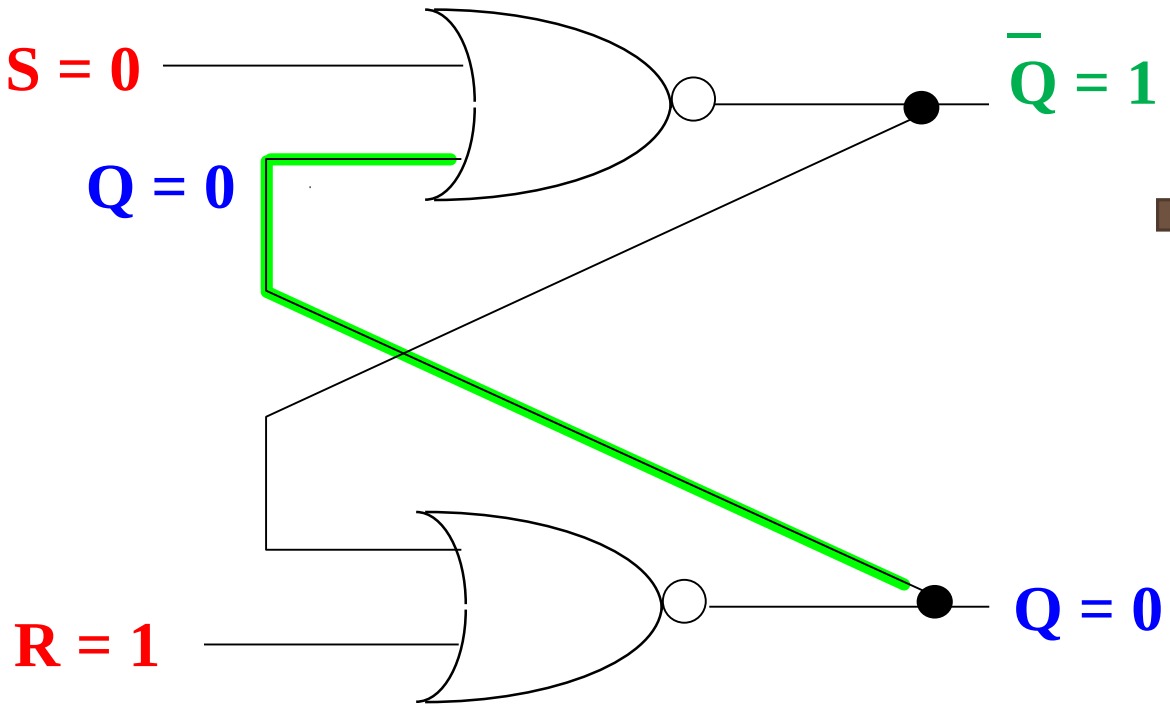
Most basic FF : Set-Reset (SR) Flip-flop / Latch



Implemented with NOR / NAND gates

S	R	Output	Q+
0	0		
0	1		
1	0		
1	1		
0 0 is the **rest state**			

SR Flip-flop (FF)

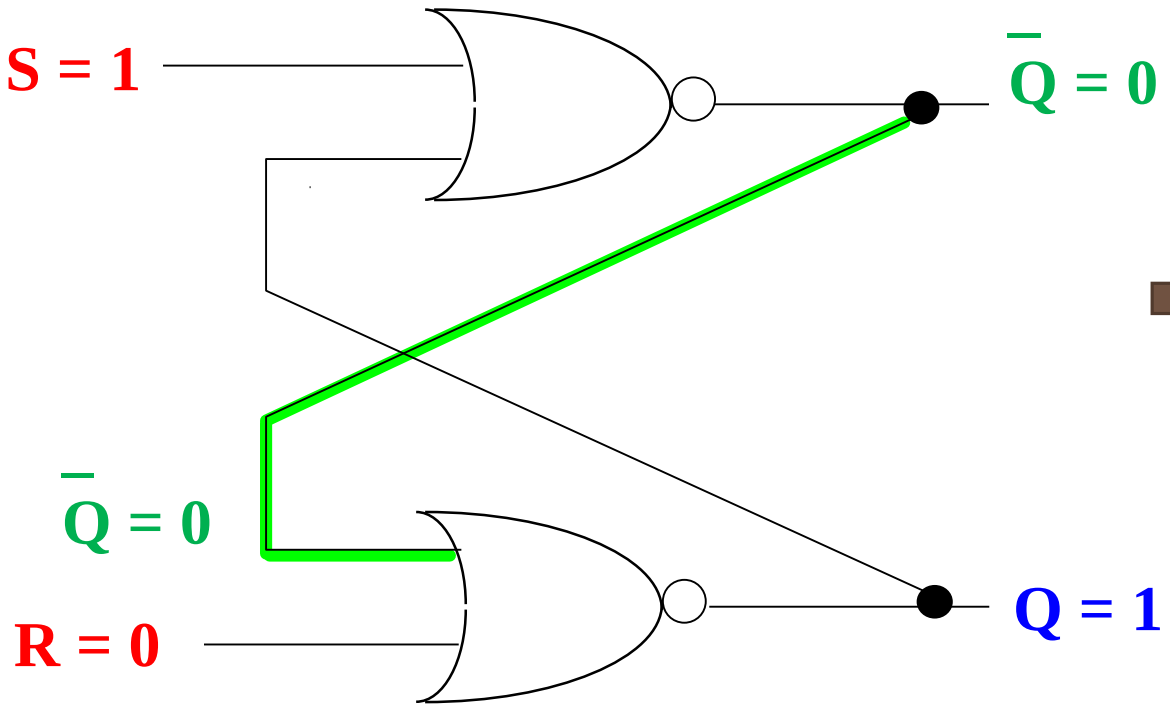


NOR Implementation

S	R	Output Q
0	0	
0	1	Q = 0
1	0	
1	1	

A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

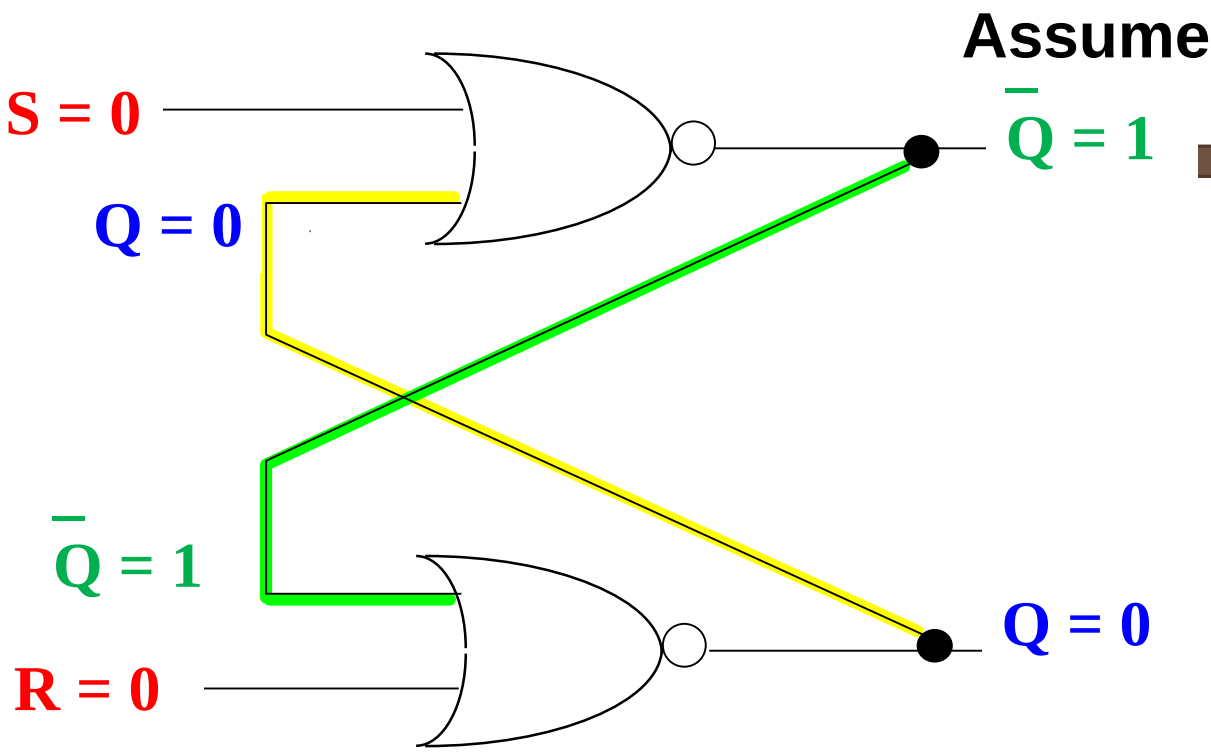
SR Flip-flop (FF)



S	R	Output Q
0	0	
0	1	Q = 0
1	0	Q = 1
1	1	

A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

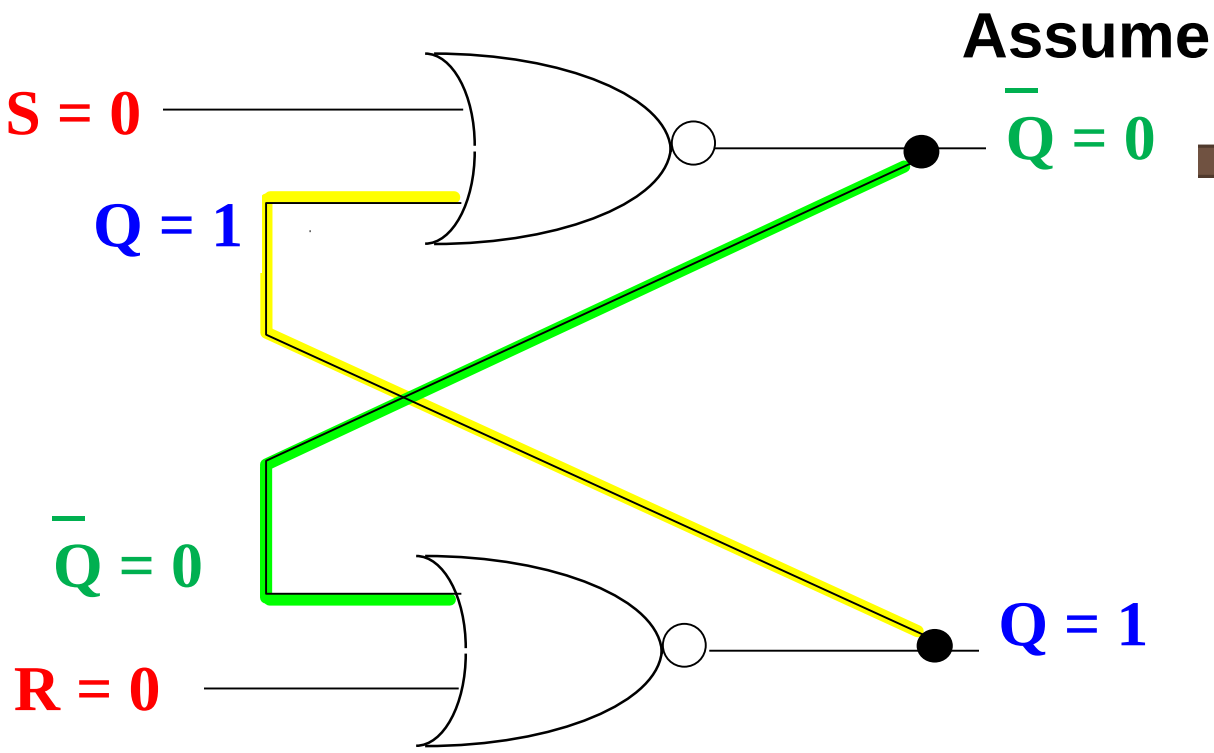
SR Flip-flop (FF)



S	R	Output Q
0	0	No Change
0	1	Q = 0
1	0	Q = 1
1	1	

A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

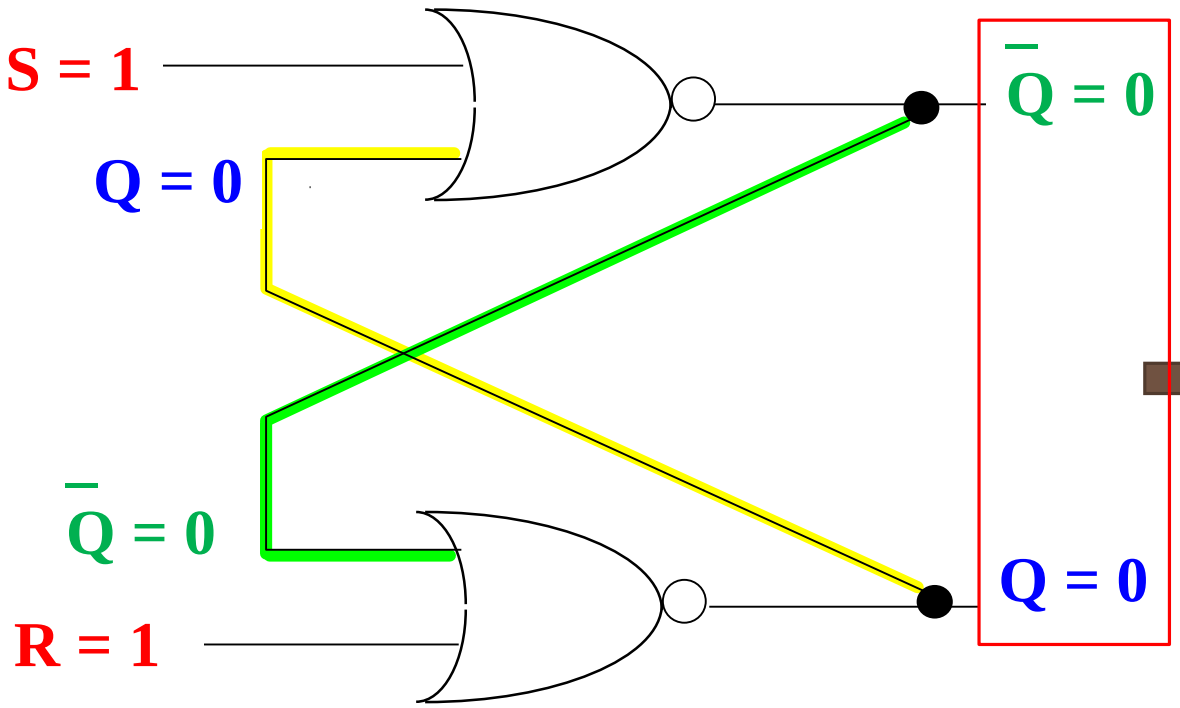
SR Flip-flop (FF)



S	R	Output Q
0	0	No Change
0	1	Q = 0
1	0	Q = 1
1	1	

A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

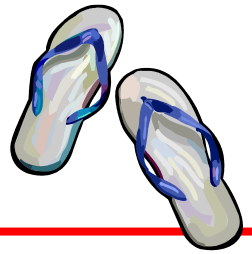
SR Flip-flop (FF)



S	R	Output Q
0	0	Hold
0	1	Q = 0
1	0	Q = 1
1	1	Invalid

A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

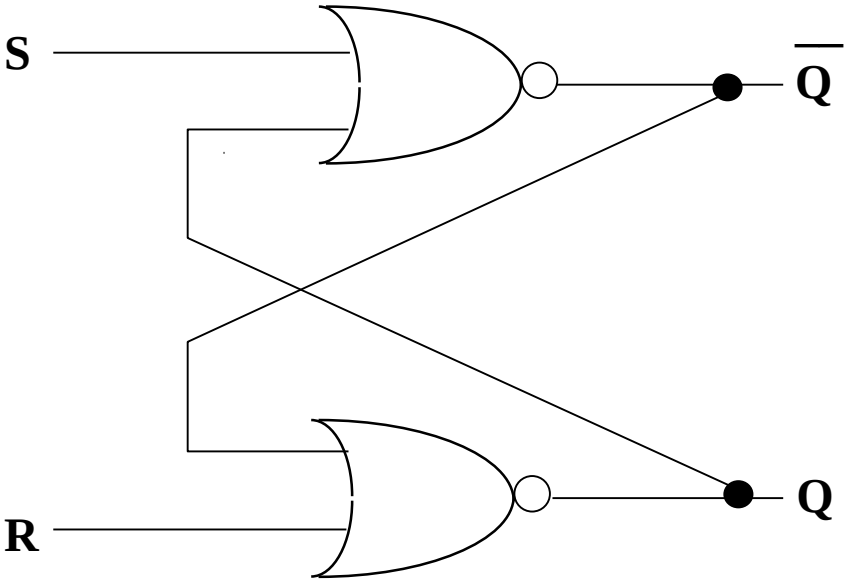
SR Flip-flop (FF)



The simplest memory element has two stable states :

Flip-Flop **(FF)** ➔ it can store 1 bit of information

Most basic FF : Set-Reset (SR) Flip-flop / Latch

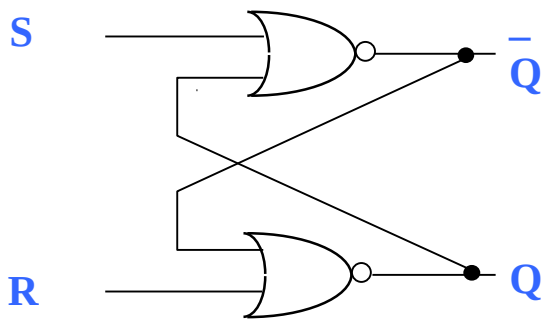


Implemented with NOR / NAND gates

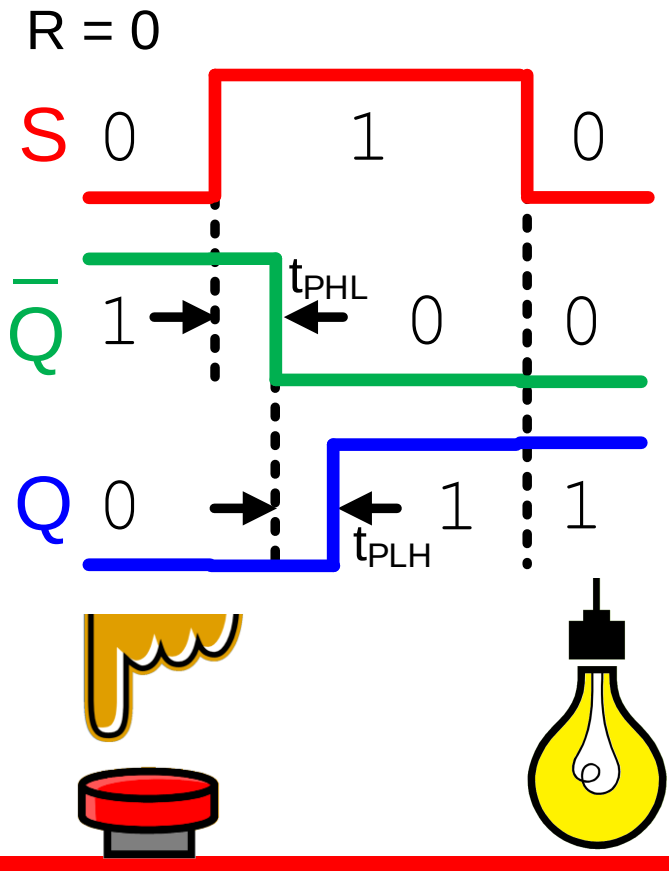
S	R	Output	Q+
0	0	Hold	Q
0	1	Q = 0	0
1	0	Q = 1	1
1	1	Invalid	Invalid
0 0 is the **rest state**			

SR Flip-flop (FF)

- FF can record and store transient events. ⎍
- Switching is not instantaneous ➔ propagation delays



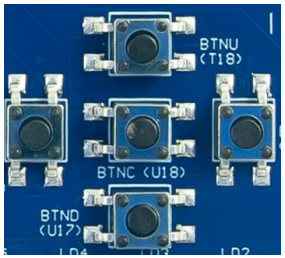
S	R	Output
0	0	Hold
0	1	Q = 0
1	0	Q = 1
1	1	Invalid
0 0 is the **rest state**		

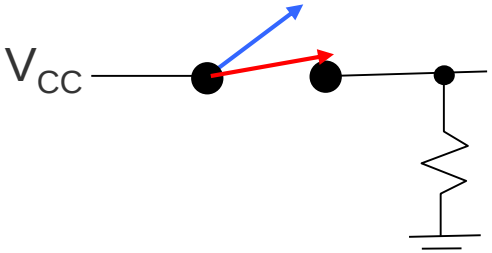


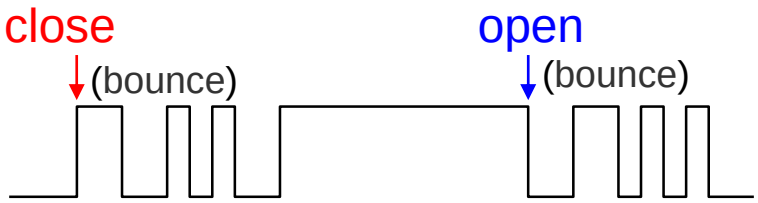
1) Assume that the *rest state* is:
S = R = 0; let Q = 0, $\overline{Q}$ = 1

2) If S→⎍ while R = 0 ⇒ Q = 1, $\overline{Q}$ = 0, i.e., the event (S going high) is recorded and stored as Q = 1.

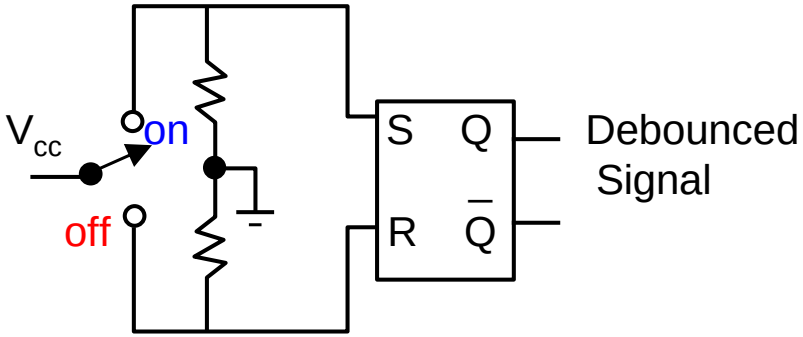
A Simple Application…

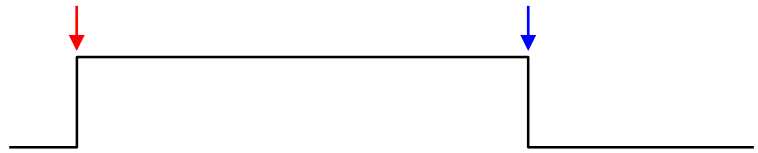






(a) Output of typical mechanical switch



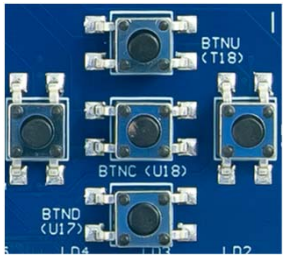


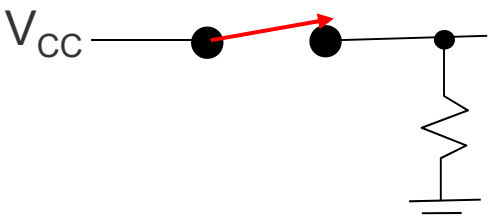
(b) Debounced switch output signal

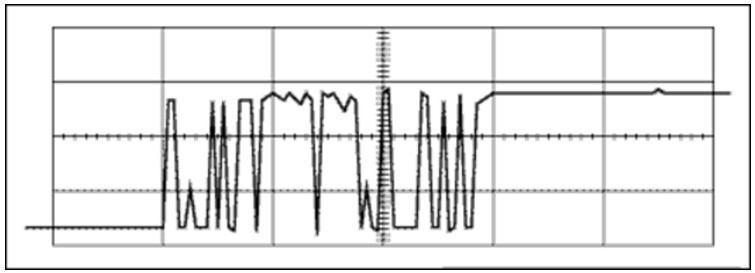
- Mechanical switches bounce before settling down which may cause problems as inputs.
- Switch debouncing is a common use of S-R FFs.

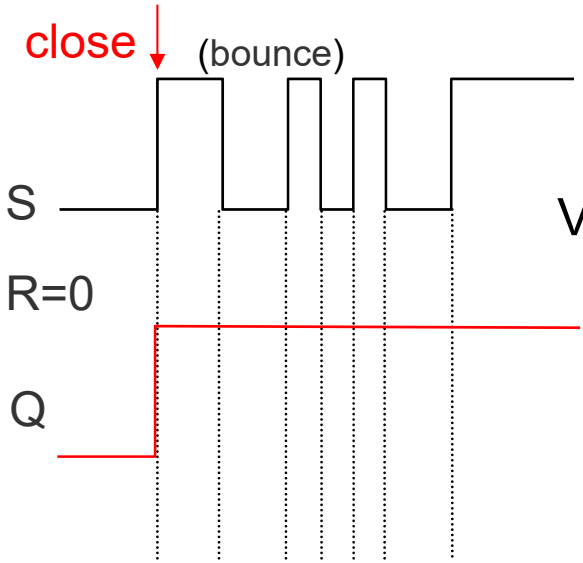
S	R	Output
0	0	Hold
0	1	Q = 0
1	0	Q = 1
1	1	Invalid
0 0 is the **rest state**		

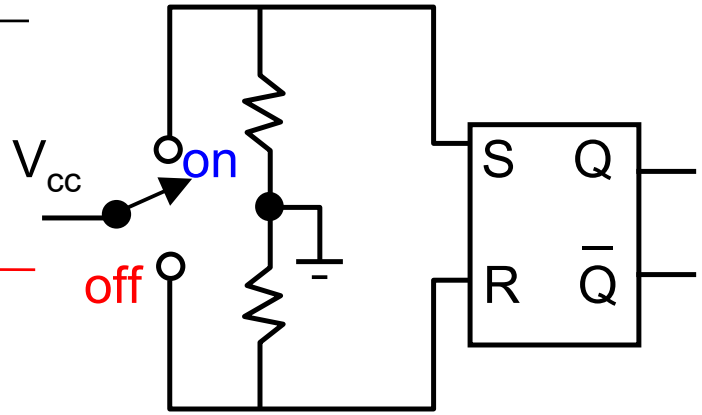
A Simple Application…









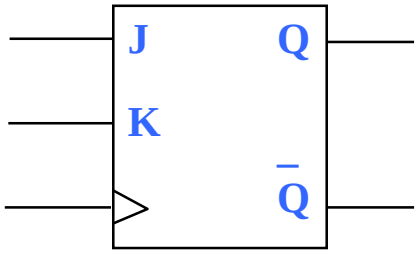


Debounced Signal

S	R	Output
0	0	Hold
0	1	Q = 0
1	0	Q = 1
1	1	Invalid
0 0 is the **rest state**		

JK FF

The JK FF is based on SR with 2 improvements : CLK & TOGGLE



CLK	J	K	Q+	
↑	0	0	Q	Hold
↑	0	1	0	Clear
↑	1	0	1	Set
↑	1	1	$\overline{Q}$	Toggle

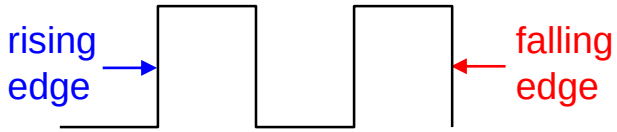
condensed characteristic table

CLK	J	K	Q	Q^+
↑	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	1
↑	1	1	1	0

characteristic table

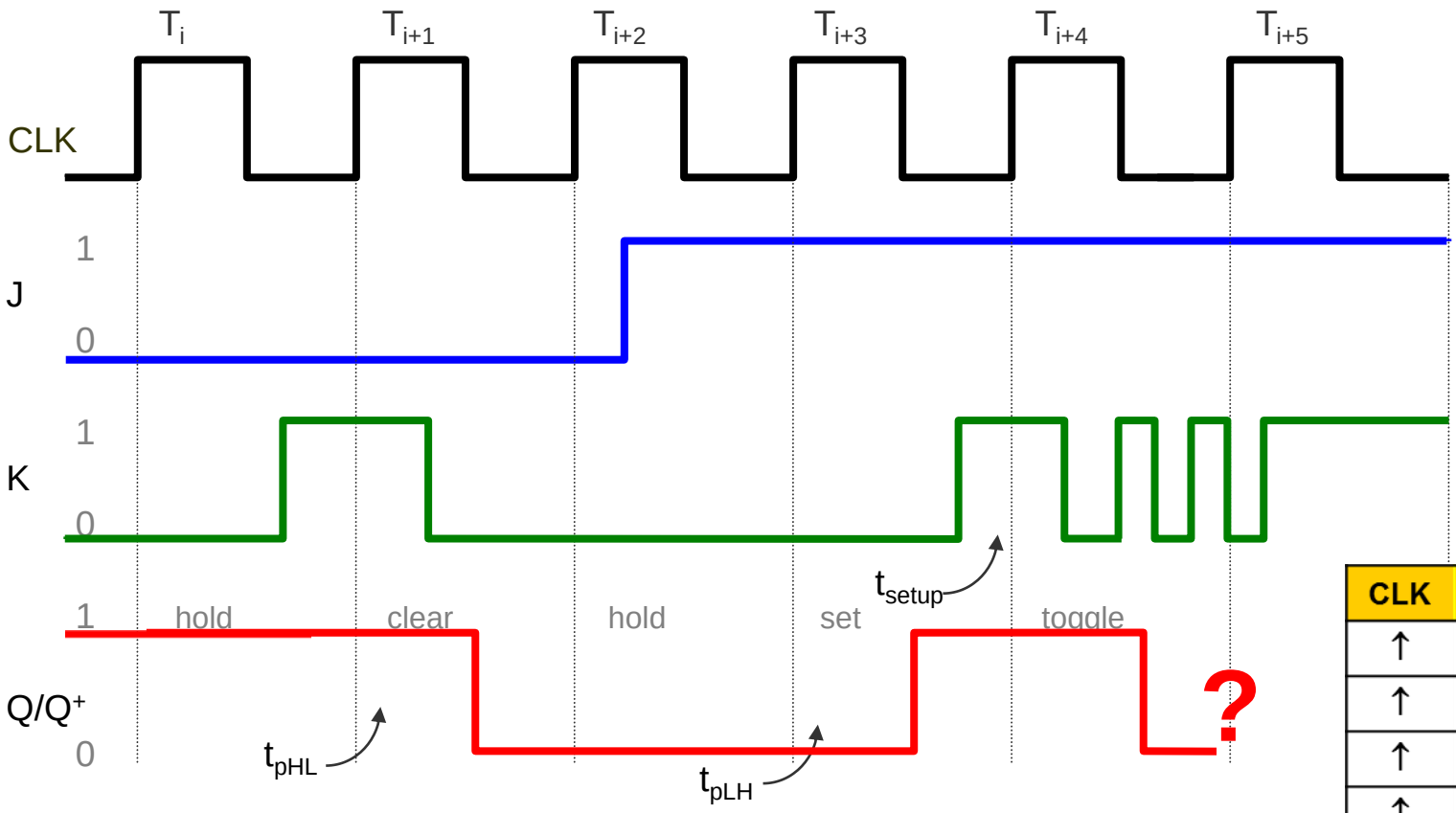
The JK FF is a **synchronous** circuit:

- *Clock input* is a controlling input.
  It specifies when circuit read inputs / change outputs.
- *Synchronous circuits* respond only at the active clock edges
  i.e., LOW → HIGH, HIGH → LOW transitions



- At any other time, changing inputs have no effect on the output.

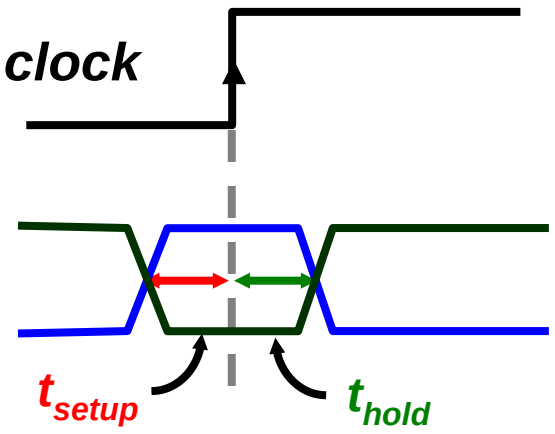
Respond @ Active Clock Edges



CLK	J	K	Q+
↑	0	0	Q
↑	0	1	0
↑	1	0	1
↑	1	1	$\overline{Q}$

- When inputs don't change ➔ FF outputs don't change.
- If inputs change ➔ FF output changes state only at active clock edge.

FF Timing Parameters



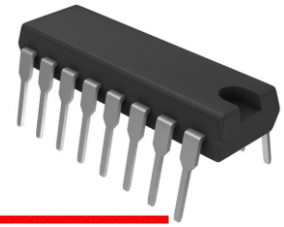
t_{setup} :	minimum time before the *active* clock edge by which FF inputs must be stable.
t_{hold}:	minimum time inputs must be stable after *active* clock edge
t_{pHL} :	time taken for FF output to change state from High to Low.
t_{pLH} :	time taken for FF output to change state from Low to High.

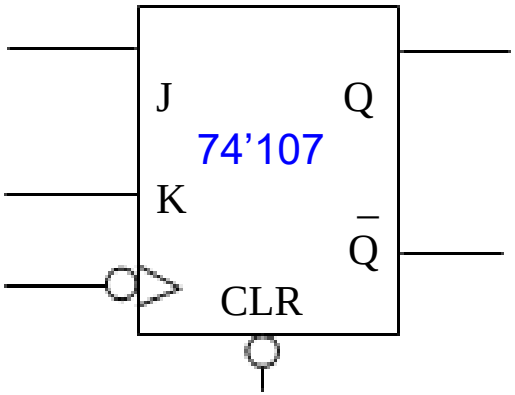
What happens if inputs change state right at the active clock transition?
Answer: output is ***unpredictable***

Thus, input changes must meet required setup & hold times of device == Operating Speed of device

http://www.ti.com/lit/ds/symlink/sn74107.pdf

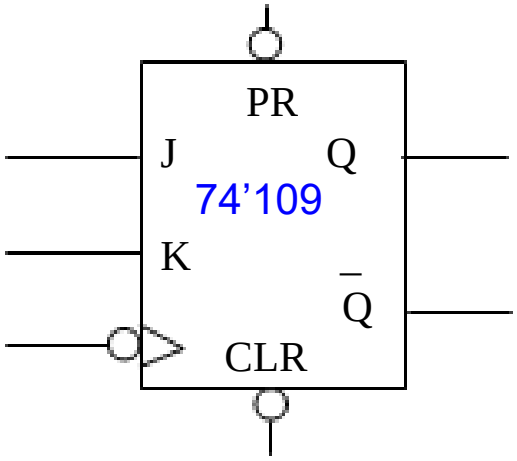
Commercially Available JK FFs





74'107 with asynchronous clear

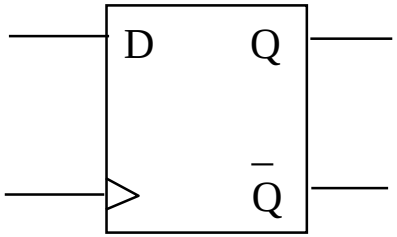
CLK	CLR	J	K	Q^+
X	L	X	X	L
↓	H	L	L	Q
↓	H	L	H	L
↓	H	H	L	H
↓	H	H	H	$\overline{Q}$



74'109 with direct set & direct clear

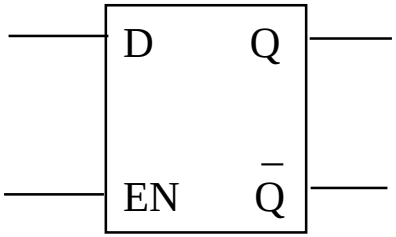
CLK	PR	CLR	J	K	Q^+
X	L	H	X	X	H
X	H	L	X	X	L
X	L	L	X	X	not allowed
↓	H	H	L	L	Q
↓	H	H	L	H	L
↓	H	H	H	L	H
↓	H	H	H	H	$\overline{Q}$

Other Flip-Flops…



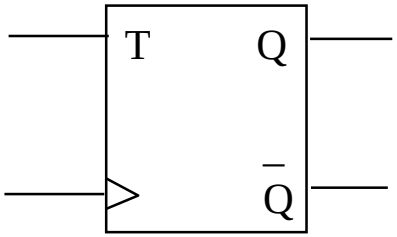
D Flip-Flop

CLK	D	Q^+
↑	0	0
↑	1	1



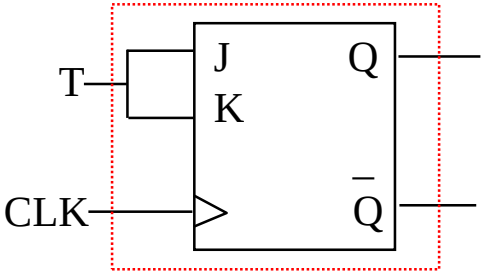
D Latch

EN	D	Q^+
0	X	No change
1	0	0
1	1	1



T Flip-Flop

CLK	T	Q^+
↑	0	Q
↑	1	$\bar{Q}$

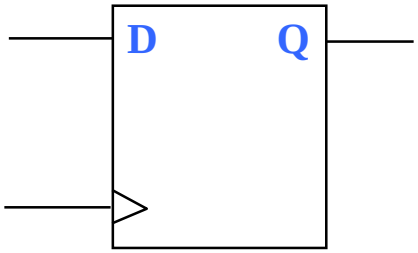


A T Flip-flop made from a J-K Flip-flop

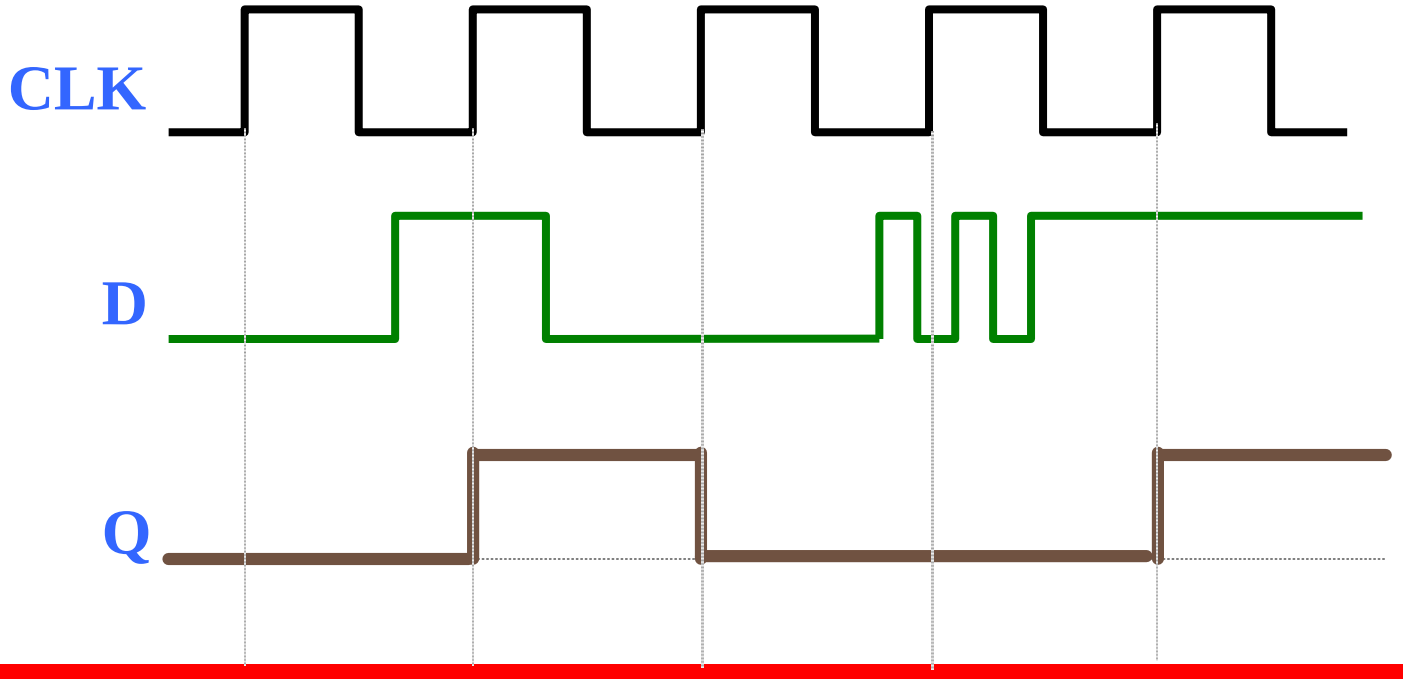
CLK	J	K	Q+
↑	0	0	Q
↑	0	1	0
↑	1	0	1
↑	1	1	$\bar{Q}$

Since **T Flip-flops** are easy to construct from other FFs, they are not available commercially.

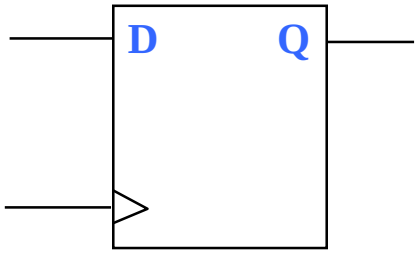
Verilog Time! – D-FF



CLK	D	Q+
↑	0	0
↑	1	1



Verilog Time! – D-FF



CLK	D	Q+
↑	0	0
↑	1	1

↑ `always @ (posedge __)`

↓ `always @ (negedge __)`

```
module dff ( input   d, clk,
             output  reg q       );



always @ (posedge clock)

begin

    q <= d;


end



endmodule
```

Some notes on: `always & reg`

Registers

- Anything assigned in an **`always`** block *must* be type **`reg`**
- In Verilog, the term register (**`reg`**) simply means a variable that can hold a value
- Values of registers can be changed instantaneously. This is different from the **`wire`** type!

Always Block

- Conceptually, the **`always`** block runs o*nce* when a signal in *sensitivity list* changes value.
- Statements within **`always`** block are executed ***sequentially***.
- **`begin`** and **`end`** behave like parentheses/brackets
- No ~~**`assign`**~~!

Summary

- SR Flip Flop & Applications
- JK Flip Flop
- FF Timing Parameters
- Commercial JK Flip Flops
- Verilog description of D Flip Flop

Practice Question

Given the circuit diagram below, complete the timing diagram below by filling in Q and $\overline{Q}$. Assume that the initial value of Q is '0' and include all propagation delays.