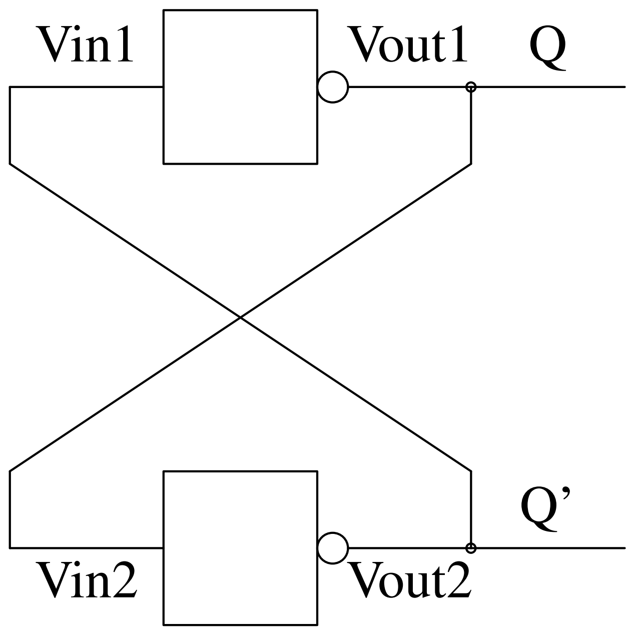
ECE 27000 S22 P 11

# Chapter 10. Sequential Logic Elements

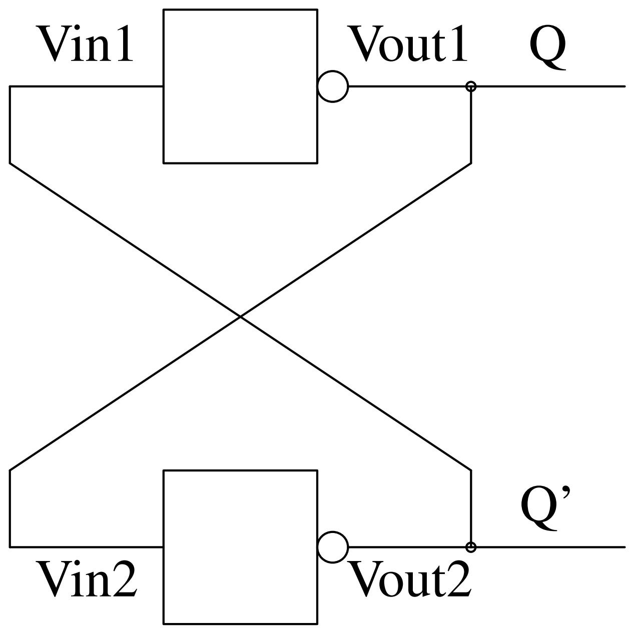
## 10.1. Bistable Elements

Memory in a digital logic circuit is created by feedback.
The simplest way to create memory is with two inverters connected in a loop.

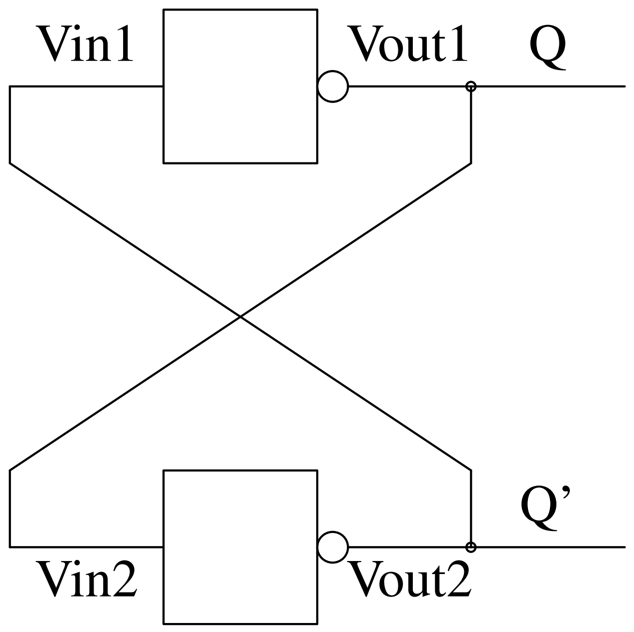


This element does not have an input.
We have no way of controlling its state.
Later we will see memory elements with inputs.

The circuit is called *bistable* since it has two stable states.
Let us use the value of $Q$ as the circuit state.



If *Q* is HIGH:
Inverter 2 has HIGH on its input and LOW on its output.
Inverter 1 has LOW on its input and HIGH on its output.
This gives one stable assignment.

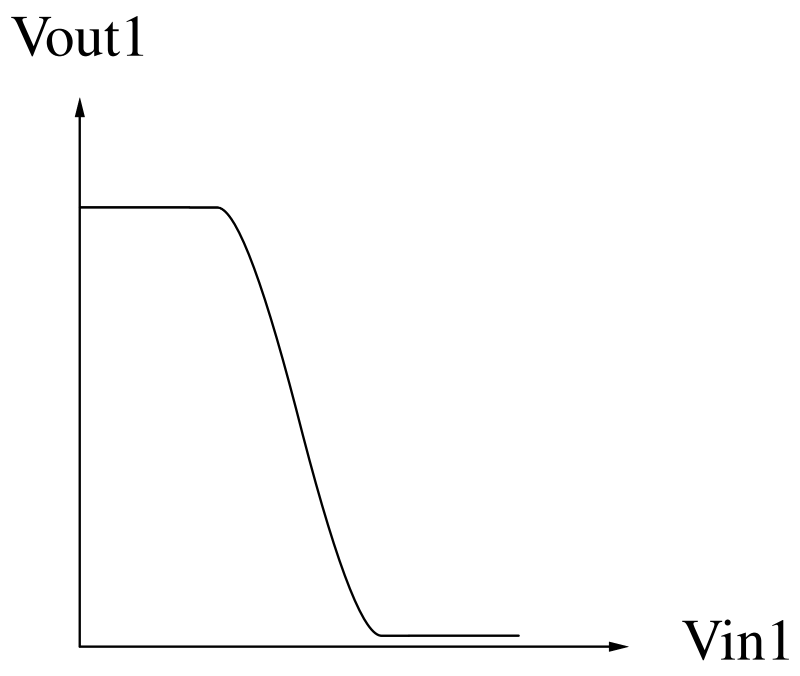


If *Q* is LOW:
Inverter 2 has LOW on its input and HIGH on its output.
Inverter 1 has HIGH on its input and LOW on its output.
This gives a second stable assignment.

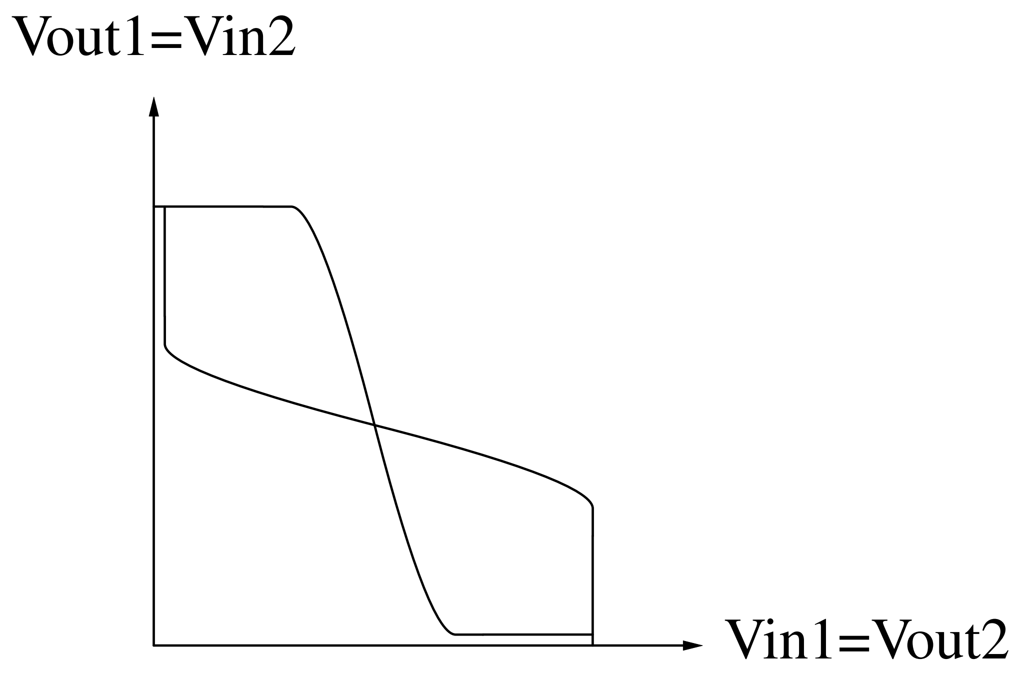
The two possible states are represented by $Q = 0$ and $Q = 1$.

Analyzing the steady-state behavior of the bistable element across the range of possible voltages indicates that there is another possible state.

If we plot $V_{out1}$ as a function of $V_{in1}$ (or $V_{out2}$ as a function of $V_{in2}$) we obtain the following transfer function:



In steady-state we have
$V_{in2} = V_{out1}$ and $V_{in1} = V_{out2}$.
Plotting both transfer functions on the same graph using these relations we obtain:



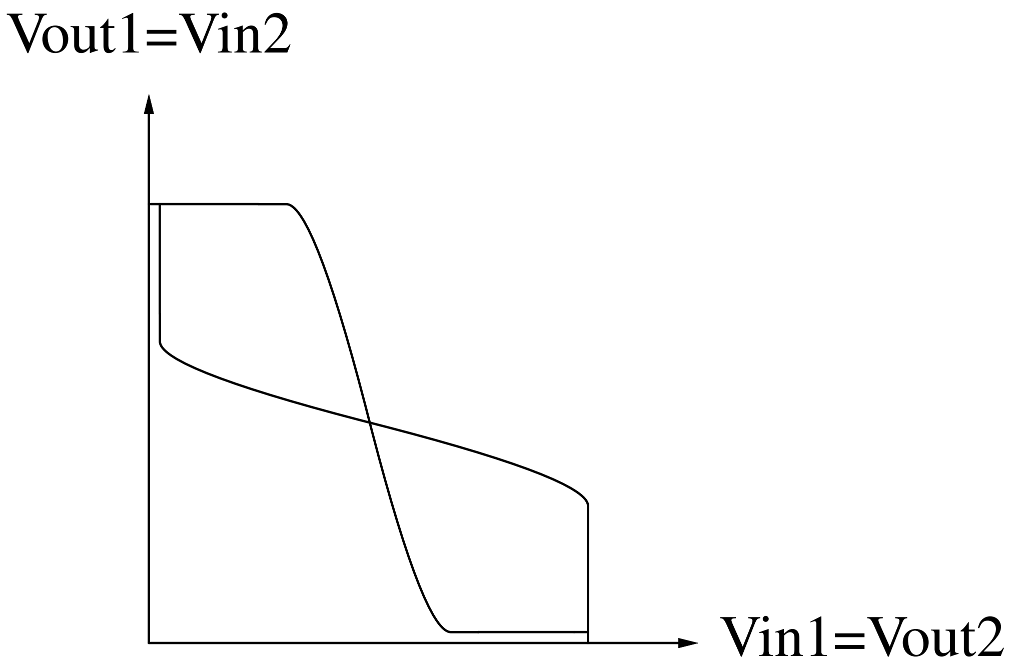
There are three points on both transfer functions where $V_{in2} = V_{out1}$, $V_{in1} = V_{out2}$.
These are the points where the bistable element will be at equilibrium.

Two of the points correspond to the states $Q = 0$ and $Q = 1$ we found earlier.

The third point is called a *metastable state* and it occurs in the middle of the voltage range.
This point does not have valid logic values corresponding to it.

A circuit can stay at the states $Q = 0$ and $Q = 1$ indefinitely.
A small amount of noise on $V_{in1}$ when $V_{in1}$ is LOW will keep $V_{out1}$ at HIGH and the circuit will not change state.

A circuit will not stay at the metastable point indefinitely.
It takes a small amount of noise to drive it out of this point.



Suppose that $V_{in1}$ reduces by a small amount due to noise.
$V_{out1}$ will increase as a result.
$V_{in2} = V_{out1}$ will increase by the same amount.
$V_{out2}$ will decrease.
This implies that $V_{in1}$ will decrease further.
This is repeated until the circuit reaches the stable state where $V_{in1}$ and $V_{out2}$ are LOW, and $V_{in2}$ and $V_{out1}$ are HIGH.

A bistable element (and a synchronous sequential circuit in general) will enter the metasable state when it is given inputs that are not "weak" enough to keep it in its current state, and not "strong" enough to drive it to the other state.

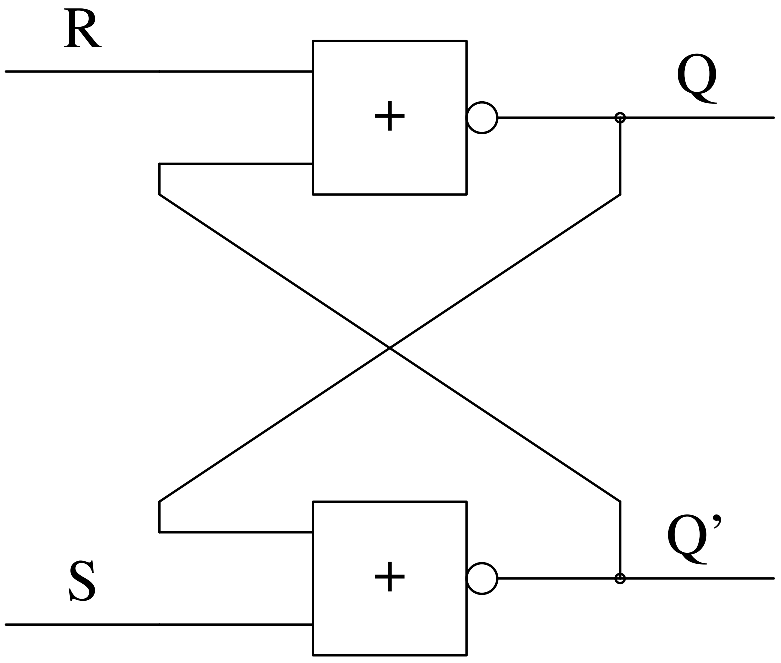
## 10.2. Latches and Flip-Flops

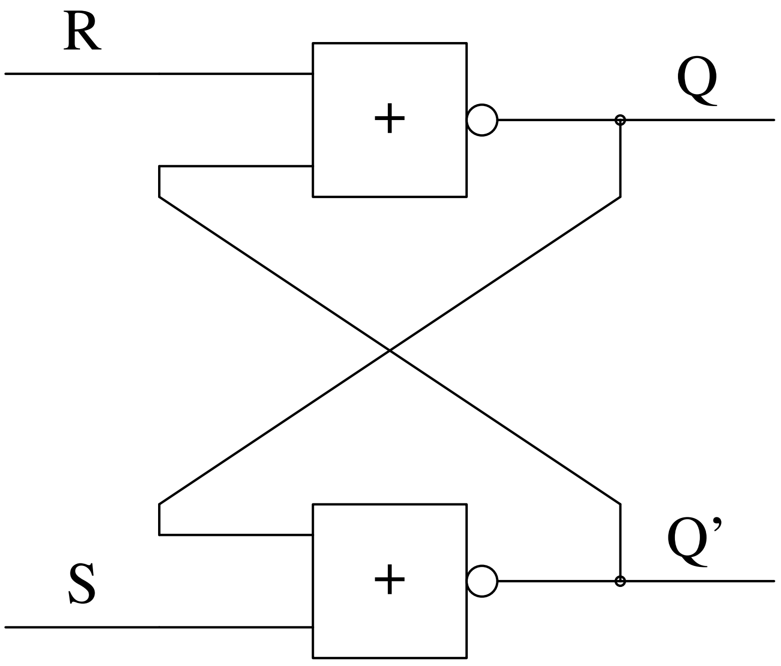
A *flip – flop* is a sequential device that samples its inputs and changes its output only when a clock signal is asserted.

A *latch* is a sequential device that responds to its inputs continuously and can change its outputs at any time.

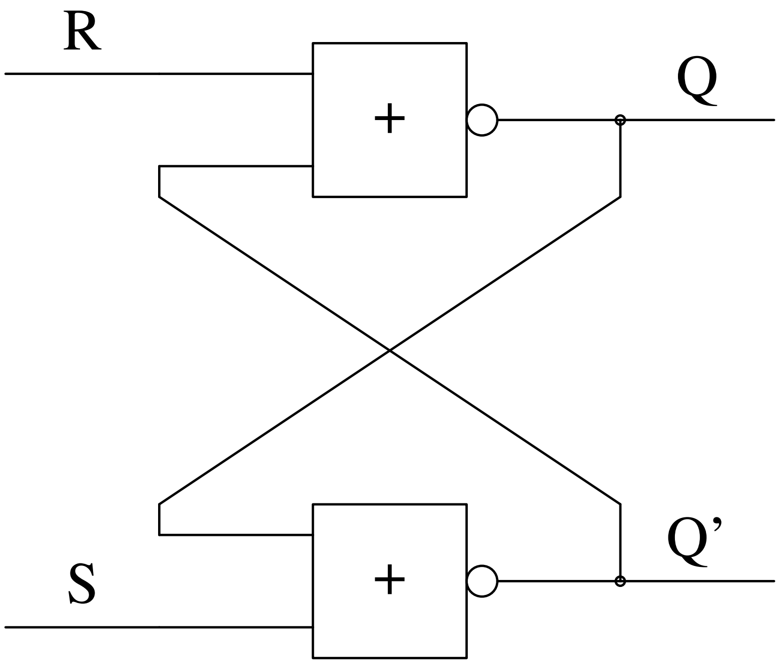
## S-R Latch

An $S-R$ $(set-reset)$ latch based on NOR gates:

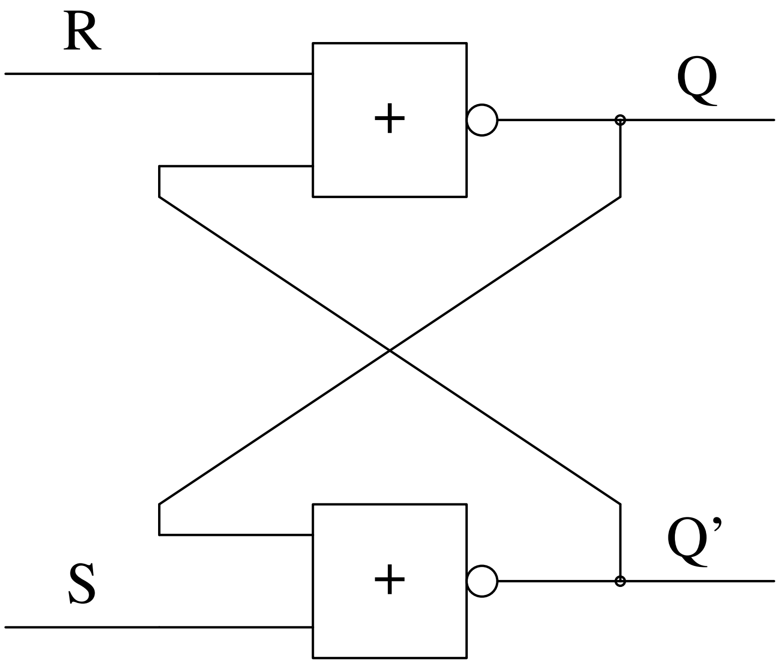




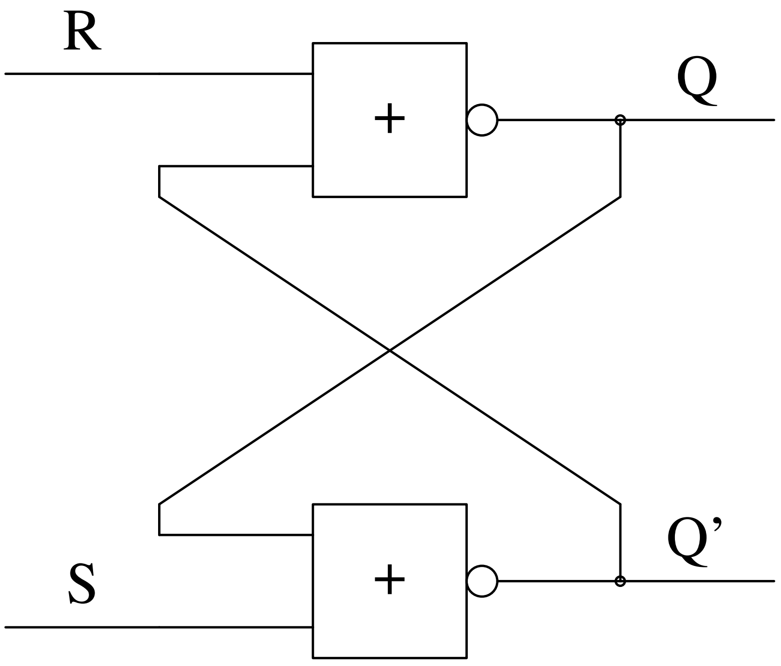
If S=R=0:
The feedback loop retains one of the two logic states, Q=0 or Q=1.



If S=1 and R=0:
S=1 implies Q’=0.
R=0 and Q’=0 imply Q=1.
Q=1 and S=1 imply Q’=0.
The state becomes Q=1.
If S is changed to 0 the state will be retained.
S is also called the *set* input.



If S=0 and R=1:
R=1 implies Q=0.
S=0 and Q=0 imply Q'=1.
Q'=1 and R=1 imply Q=0.
The state becomes Q=0.
If R is changed to 0 the state will be retained.
R is also called the *reset* input.

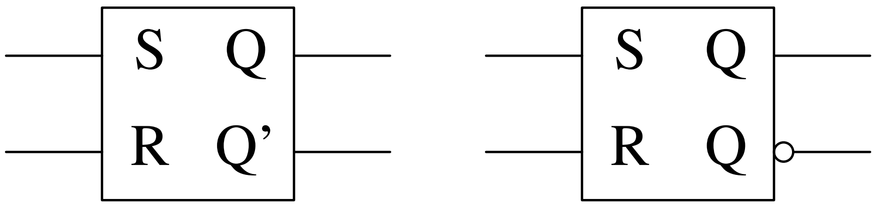


If S=1 and R=1, logic analysis shows that Q=Q'=0.
Functionally this is not a desired situation.

In addition, if S and R change from 1 to 0 simultaneously, the latch may go into the metastable state.

We will avoid S=R=1 when designing circuits with S-R latches (and, later, S-R flip-flops).

Logic symbols for an S-R latch:



Timing parameters for an S-R latch:
The propagation delay is the time it takes for a transition on an input signal to produce a transition on an output signal.

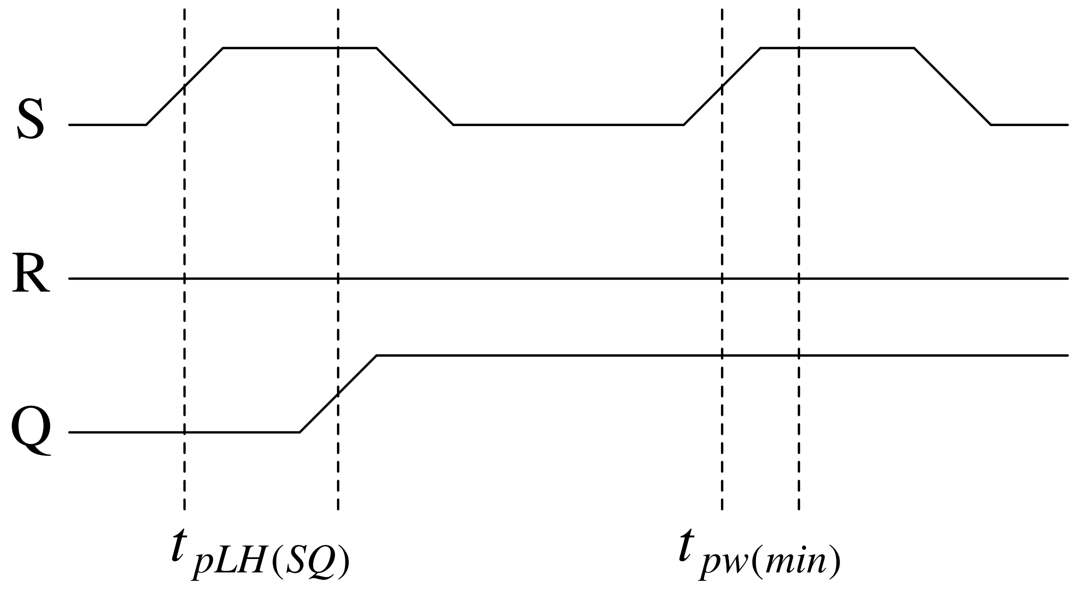
This may be different for different inputs, outputs, and transitions. We have:

$t_{pLH(SQ)}$

$t_{pHL(RQ)}$

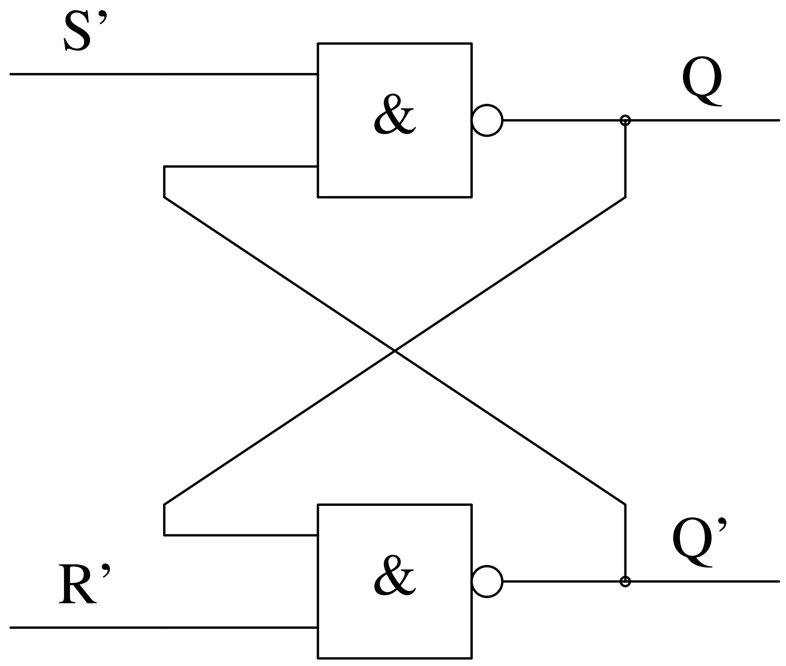
$t_{pHL(SQ')}$

$t_{pLH(RQ')}$



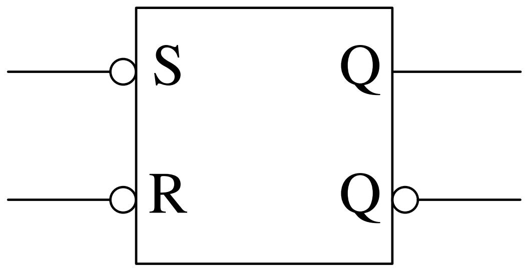
A $minimum - pulse - width$ for S and R indicates the minimum amount of time S and R should be HIGH to ensure that the latch does not go into the metastable state.

An S’-R’ latch is similar to an S-R latch except that it uses NAND gates and has active-low S and R inputs.

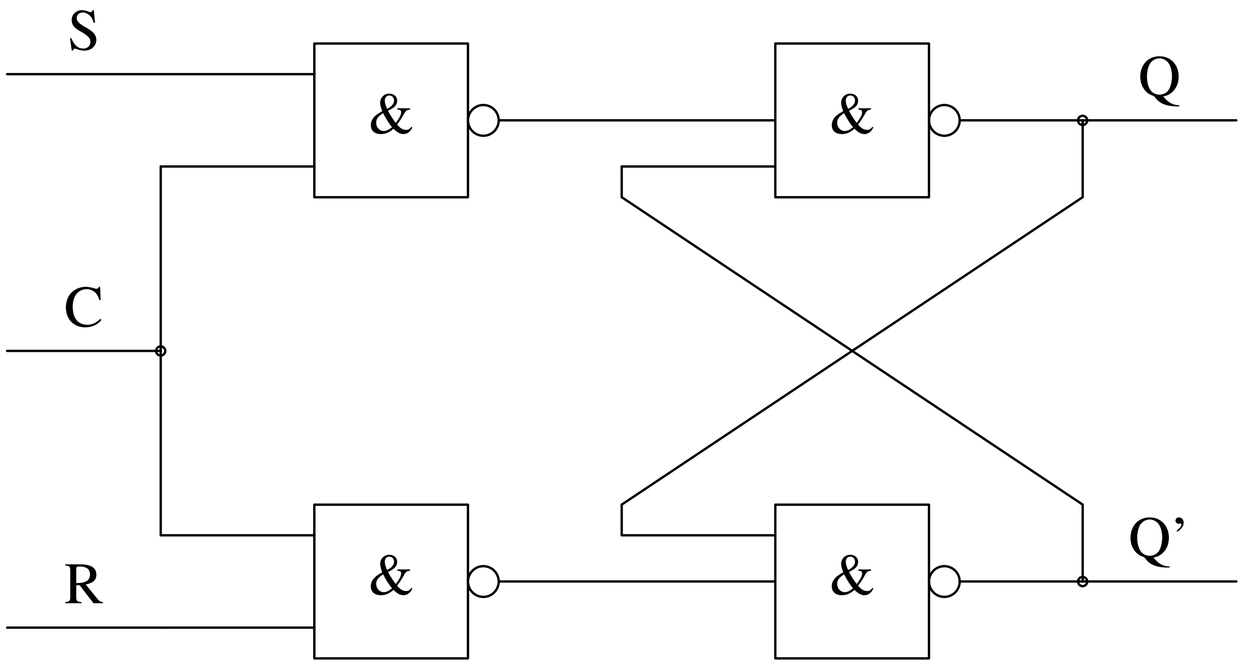


| S | R | S’ | R’ | Q | Q’ | |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | same | same | |
| 0 | 1 | 1 | 0 | 0 | 1 | |
| 1 | 0 | 0 | 1 | 1 | 0 | |
| 1 | 1 | 0 | 0 | 1 | 1 | unused |

Logic symbol:



An S-R latch may have an enable input.
It operates as an S-R latch only when the enable input is asserted.



| C | S | R | Q | Q' | |
|---|---|---|---|---|---|
| 0 | x | x | same | same | |
| 1 | 0 | 0 | same | same | |
| 1 | 0 | 1 | 0 | 1 | |
| 1 | 1 | 0 | 1 | 0 | |
| 1 | 1 | 1 | 1 | 1 | unused |

An S-R latch is used when it is convenient to think of states as being set or reset.
A D latch is used when it is convenient to think of states as storing information.

A D latch is similar to an S-R latch except that S=D and R=S’ permanently.
As a result, D=1 results in Q=1, and D=0 results in Q=0.
A D latch is also called a *transparent latch* since Q follows D.

| C | S | R | Q | Q’ | |
|---|---|---|---|---|---|
| 0 | x | x | same | same | |
| 1 | 0 | 0 | same | same | |
| 1 | 0 | 1 | 0 | 1 | |
| 1 | 1 | 0 | 1 | 0 | |
| 1 | 1 | 1 | 1 | 1 | unused |

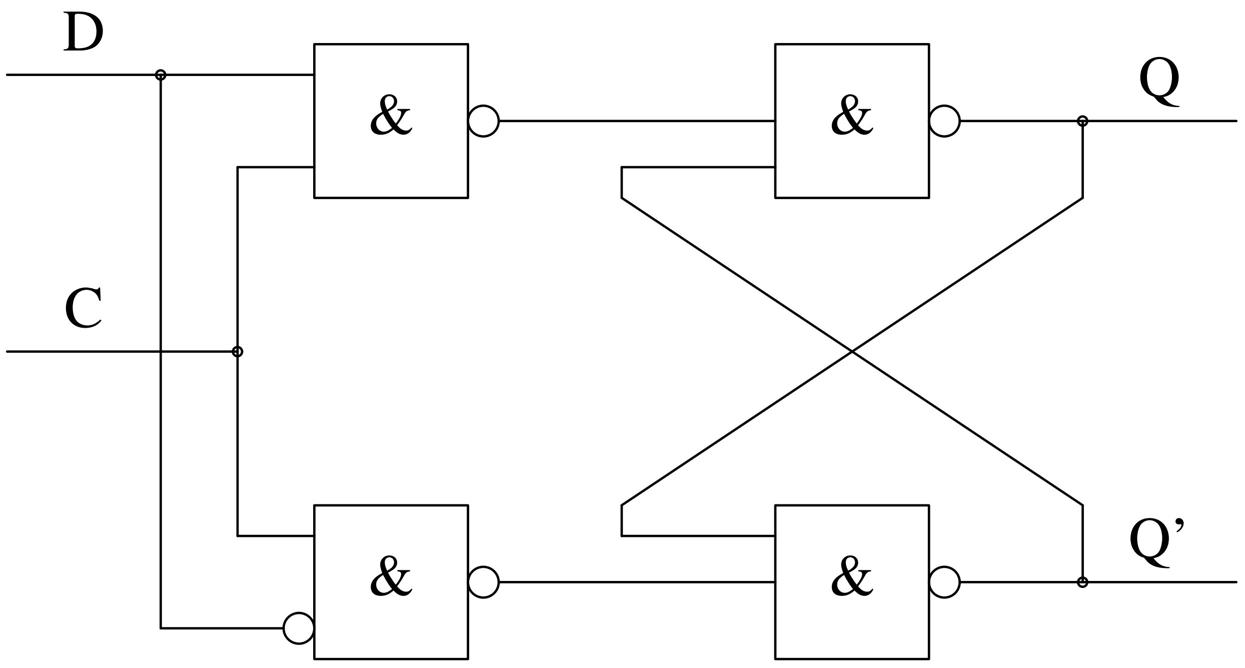
R=S’:

| C | S | R | Q | Q’ |
|---|---|---|---|---|
| 0 | x | x | same | same |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |

S=D:

| C | D | Q | Q’ |
|---|---|---|---|
| 0 | x | same | same |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

A D latch with an enable:



Propagation delays for the D latch are

$t_{pLH(DQ)}$

$t_{pHL(DQ)}$

$t_{pLH(CQ)}$

$t_{pHL(CQ)}$

$t_{pLH(DQ')}$

$t_{pHL(DQ')}$

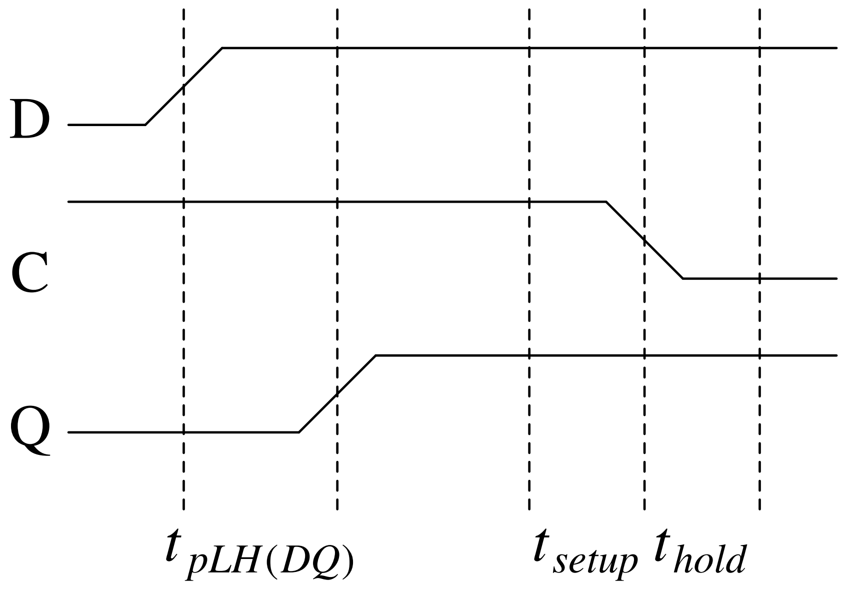
$t_{pLH(CQ')}$

$t_{pHL(CQ')}$

The latch captures a state and remains there after C goes to 0.
To avoid metastability, there is a window of time around the falling edge of C when D must not change.

The window of time begins $t_{setup}$ time units before the falling edge of C. This is called the *setup time*.

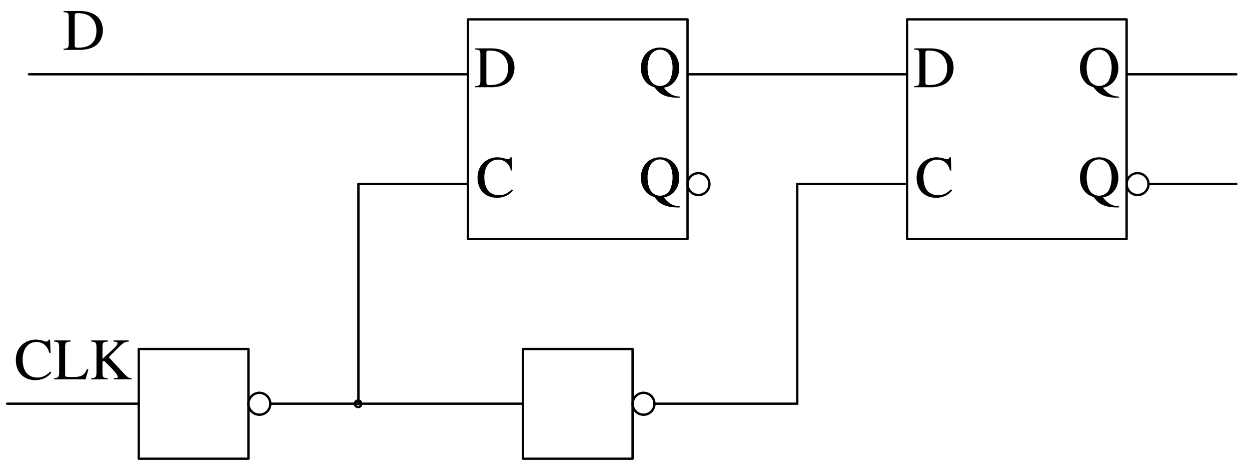
The window of time ends $t_{hold}$ time units after the falling edge of C. This is called the *hold time*.



If D changes during the setup or hold times, the state that the latch will end up is unpredictable, and it may be the metastable state.

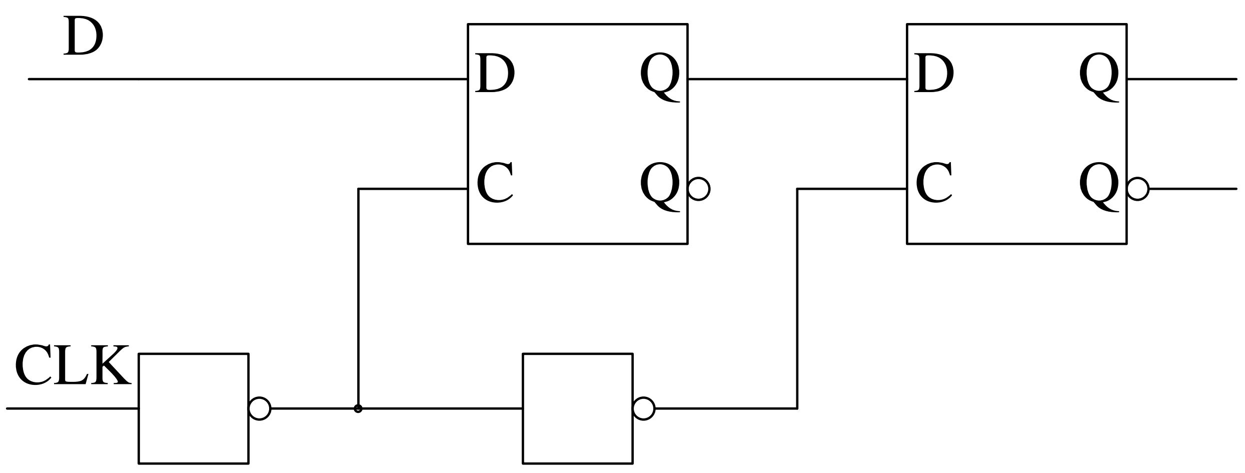
It is possible to connect a clock to the enable signal of a latch to obtain a synchronous element.
However, this element will respond to multiple changes on its inputs as long as the clock signal is high.

A *positive-edge-triggered D flip-flop* uses two D latches to create a circuit that samples its D input and changes its Q and Q’ outputs only at the rising edge of a clock signal CLK.



Latch 1 responds to the D input when CLK is 0.
When CLK is 1 it retains its state.

Latch 2 responds to the output of latch 1 when CLK goes to 1.
When CLK is 0 it retains its state.

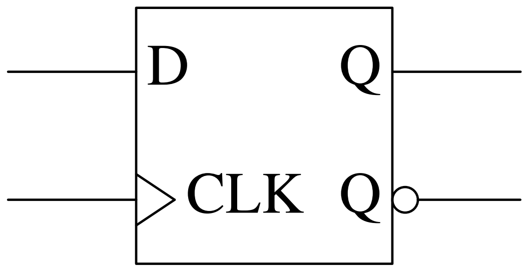


CLK=0:
Latch 1 changes its state.
Latch 2 ensures that the state (the rightmost Q) does not change.

CLK=1:
Latch 1 holds its state.
Latch 2 responds to the change of state captured in latch 1.
Since its input does not change again, its output becomes stable.

The state (rightmost Q) changes according to the last D input before the rising edge of the clock. While the clock is high, latch 1 prevents additional changes from occurring.

Logic symbol for a positive-edge-triggered D flip-flop:



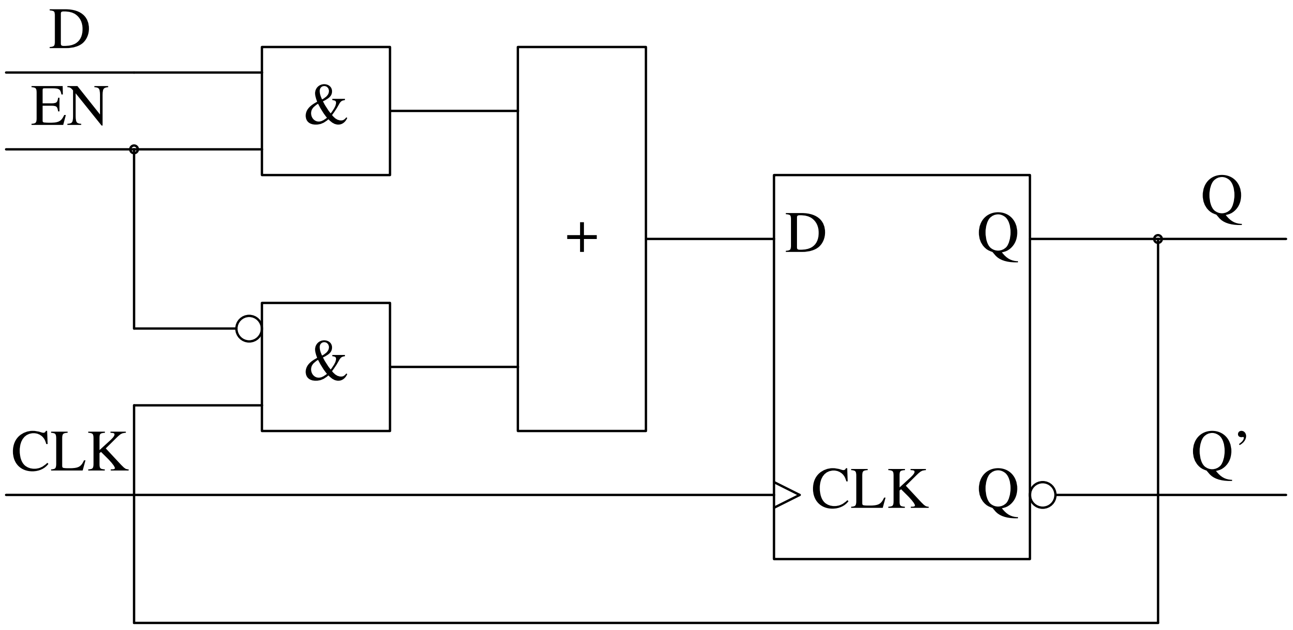
Propagation delays and setup and hold times are measured with respect to the rising edge of the clock.

A negative-edge-triggered D flip-flop is obtained by inverting the clock input.

Some flip-flops have set and reset functions that are not synchronized with the clock.

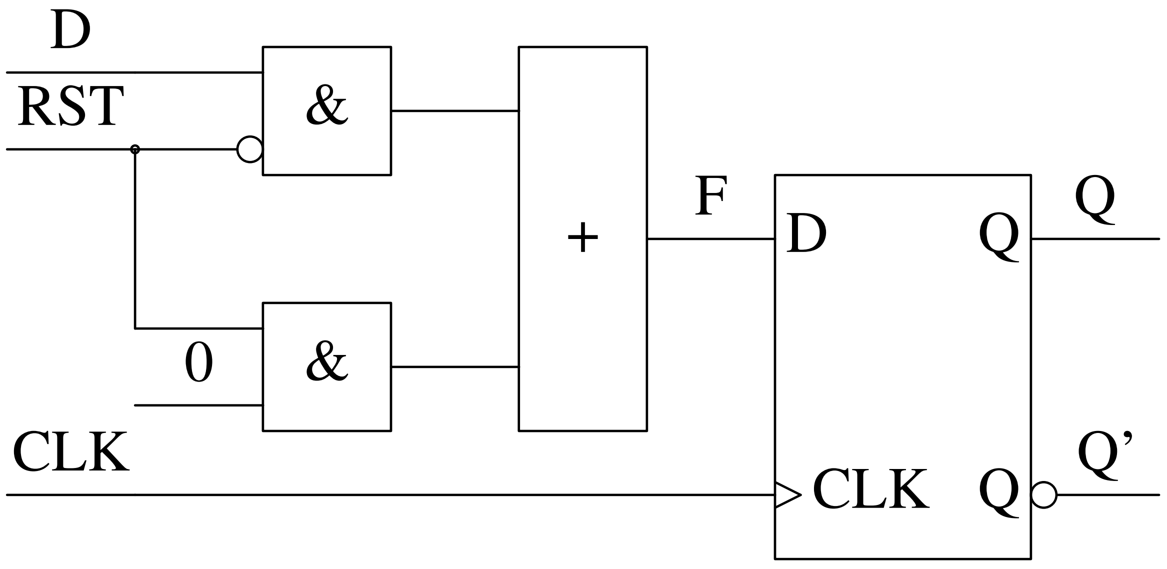
A D flip-flop with an enable input.
At the rising edge of the clock:
If the enable input is asserted, the external D input determines the flip-flop state.
If the enable input is negated, the flip-flop holds its previous state.

This is implemented by a multiplexer that selects between the external D input and Q.



A D flip-flop with a reset input.
At the rising edge of the clock:
If the reset input is asserted, the flip-flop state is reset to 0.
If the reset input is negated, the external D input determines the flip-flop state.

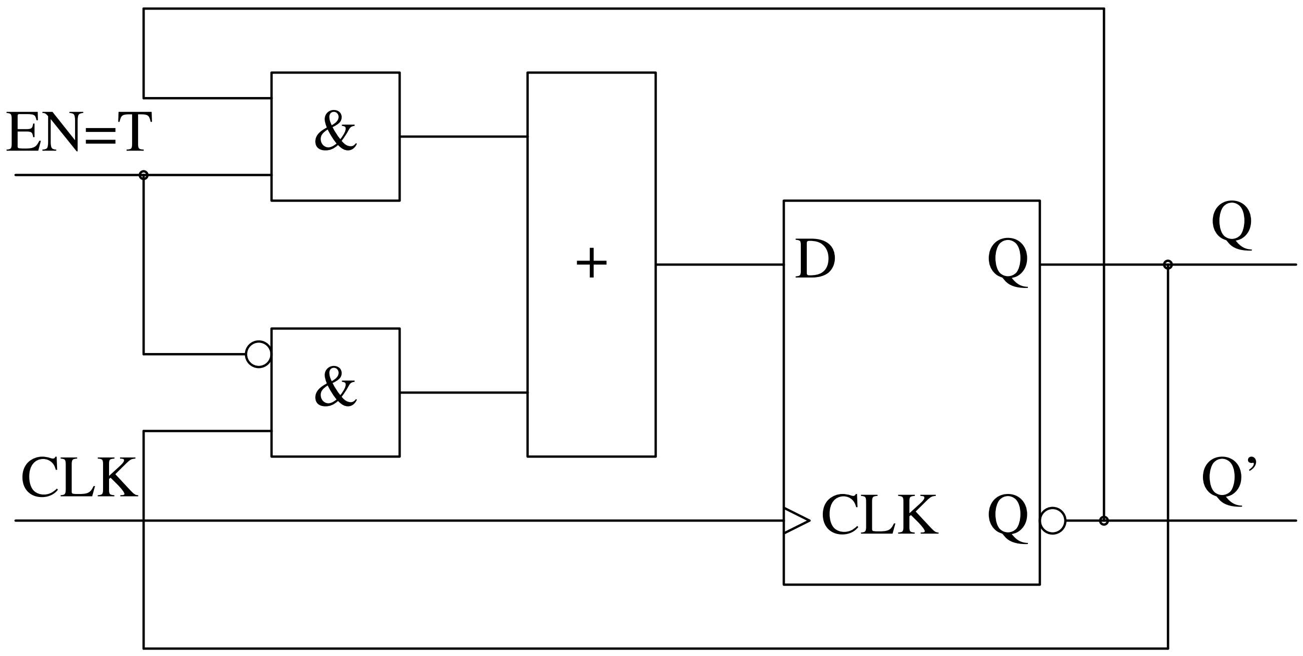
This is implemented by a multiplexer that selects between the external D input and 0.



The MUX can be simplified:
$F = RST' \cdot D + RST \cdot 0 = RST' \cdot D$

A $T$ ($toggle$) $flip-flop$ changes state on every rising edge of the clock.

A T flip-flop with enable changes state on a rising edge of the clock if the enable input is asserted.
Possible implementation:



Characteristic equation:
$Q* = TQ' + T'Q = T \oplus Q$
$T = Q \oplus Q*$

## Example: An up-counter

| Q2 | Q1 | Q0 | Q2* | Q1* | Q0* | T2 | T1 | T0 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |

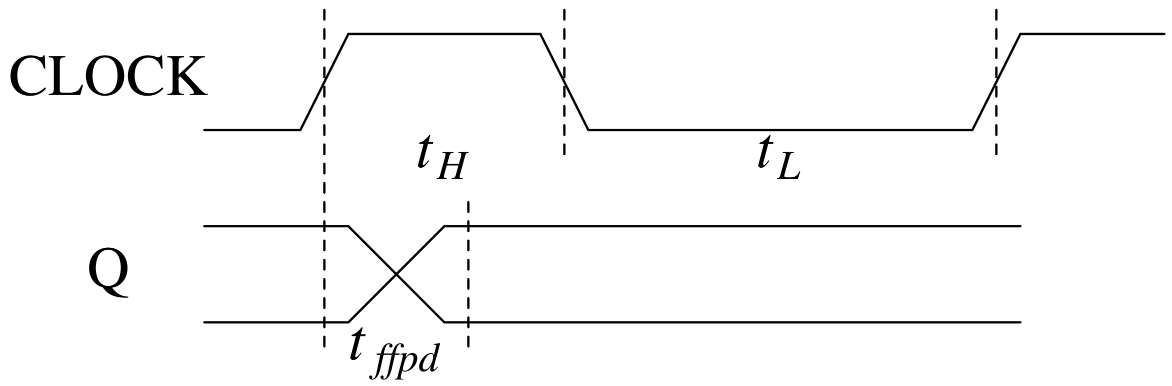
$$T_0 = 1$$

$$T_1 = Q_0$$

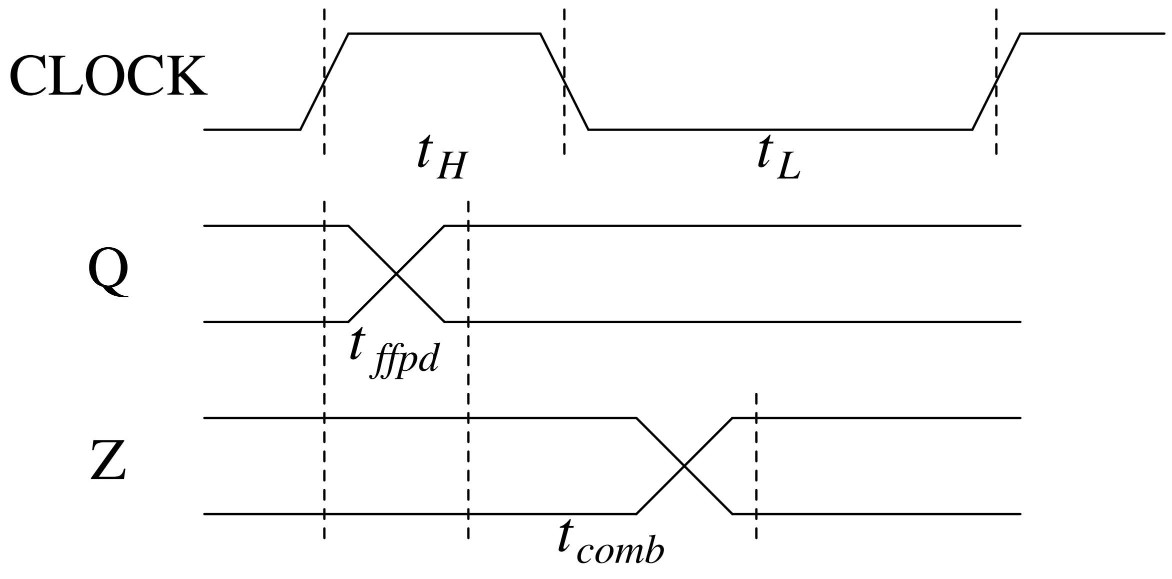
$$T_2 = Q_1 \cdot Q_0$$

## Sequential Circuit Clocking Considerations

A typical timing diagram:



*Q* represents the output of a flip-flop.
$t_{ffpd}$ is the time from the rising edge of the clock, until the output of the flip-flop is stable.



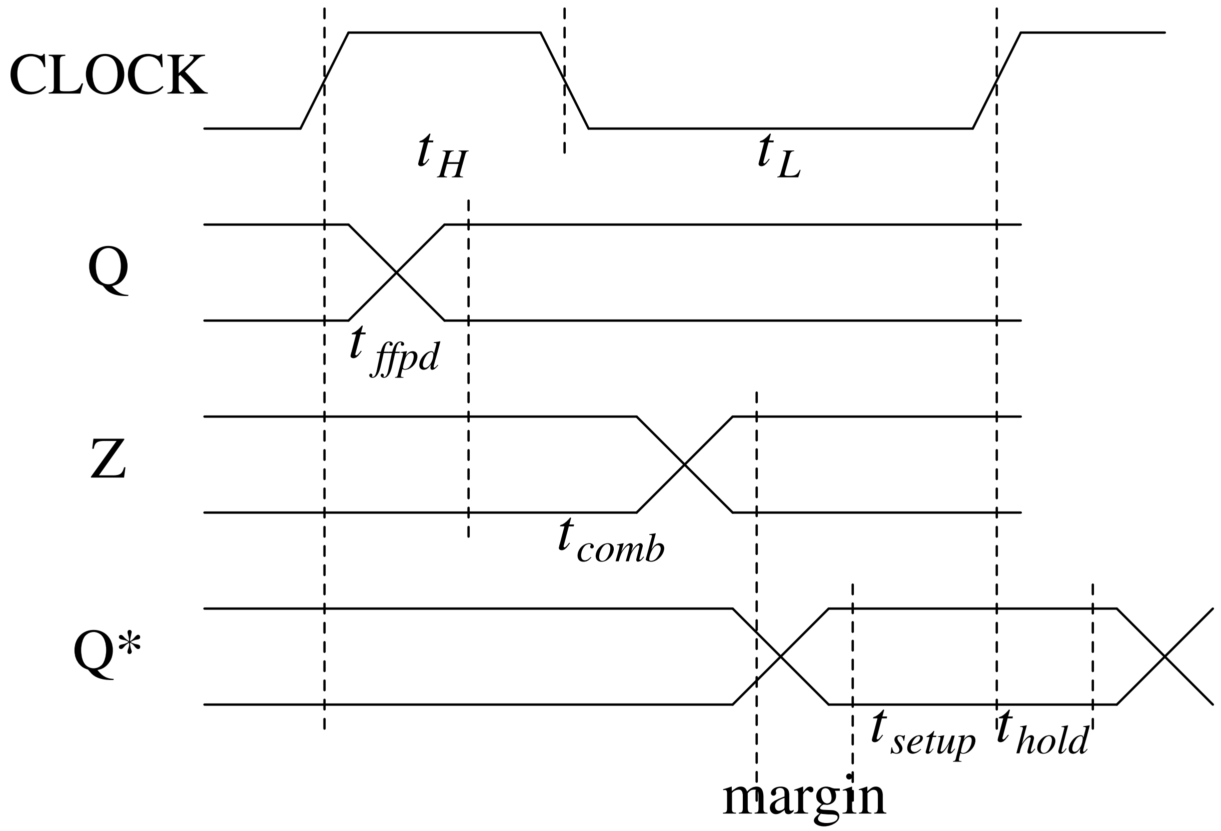
*Z* represents the outputs of the combinational logic.

$t_{comb}$ is the time it takes for the changes on the inputs of the combinational logic (including the outputs of the flip-flops) to propagate to the outputs of the combinational logic.

The outputs of the combinational logic include the inputs to the flip-flops.

They need to be stable $t_{setup}$ time units before the next rising edge of the clock.

They also need to be held for $t_{hold}$ time units after the next rising edge of the clock.



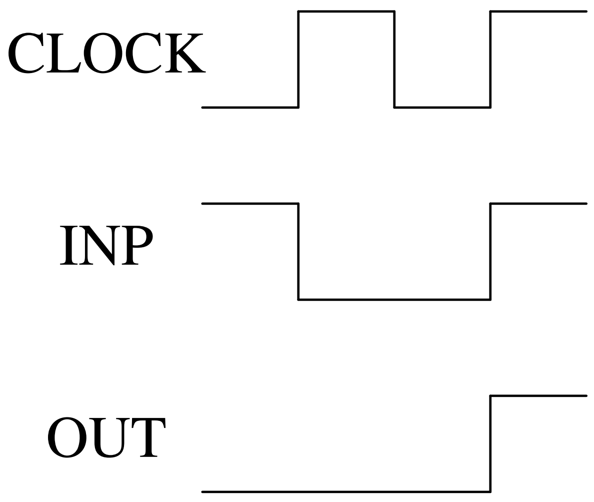
For proper operation we must have
$t_{ffpd} + t_{comb} + t_{setup} < t_{clk}$.

The value $t_{clk} - t_{ffpd(max)} - t_{comb(max)} - t_{setup}$ is called the *setup-time margin*.
It is computed using worst-case (maximum) propagation delays.
For $t_{comb}$ this implies finding the worst-case path through the combinational logic.

The sum $t_{ffpd(min)} + t_{comb(min)}$ must be larger than the hold time to ensure that the flip-flop inputs do not start changing too early.
The value $t_{ffpd(min)} + t_{comb(min)} - t_{hold}$ is the *hold-time margin*.

A timing diagram meant to specify functional behavior may align changes with the clock edge. This implies that the changes occur sometime after the clock edge.

Example:



## 10.3. Latches and Flip-Flops in Verilog

A basic D latch:

```
module VrDlatch(D,G,Q);
    input D,G;
    output reg Q;
    always @(D or G)
    begin
        if(G==1) Q<=D;
    end
endmodule
```

Since the condition G=0 is left out (the if has no else), the synthesis tool recovnizes that a latch is needed for Q to hold its state.

A basic D flip-flop:

```
module VrDff(CLK,D,Q);
    input CLK,D;
    output reg Q;
    always @(posedge CLK)
        Q<=D;
endmodule
```

The always block is exceuted at the positive edge of CLK.
In this case, Q is assigned the value of D.
Q is held at the same value at any other time.

Read more in Section 10.3.

## 10.4. Multibit Registers and Latches

A collection of two or more D flip-flops with a common clock input is called a *register*.

Example: An 8-bit register in Verilog.

```
module Vr8bReg(CLK,EN_L,D,Q);
    input CLK,EN_L;
    input [1:8] D;
    output reg [1:8] Q;
    always @(posedge CLK)
        if(En_L==0) Q<=D;
endmodule
```

The always block is exceuted at the positive edge of CLK.
In this case, the 8-bit Q is assigned the 8-bit value of D.
Q is held at the same value at any other time.

Read more in Section 10.4.

yyy