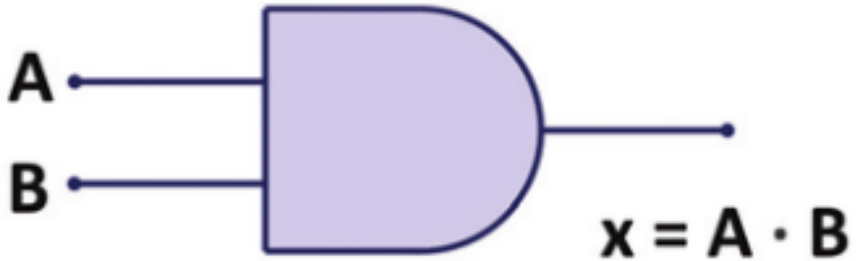
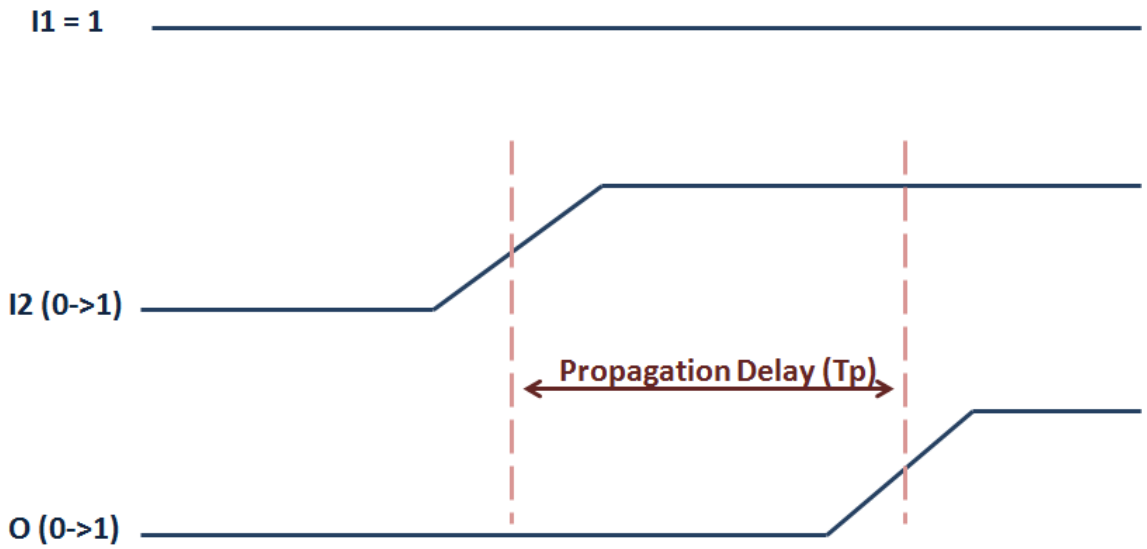
Lecture 16 – Sequential circuits

Chapter 5

Gate delays

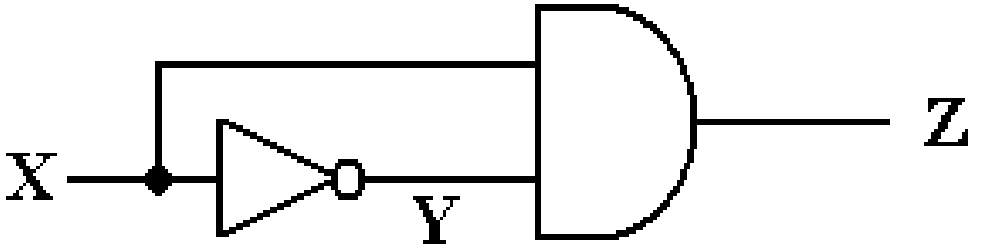
- When we apply an input to a gate, the output does not change immediately – there is a delay between the application of the input and appearance of the correct output
- This delay is generally of the order of nanoseconds, but can add up as signal goes through multiple gates
- *While making a digital IC design, gate delay (and by extension timing) is the single biggest consideration of all time!*
- Other things to worry about are silicon real estate, power consumption, reliability, testability, etc.

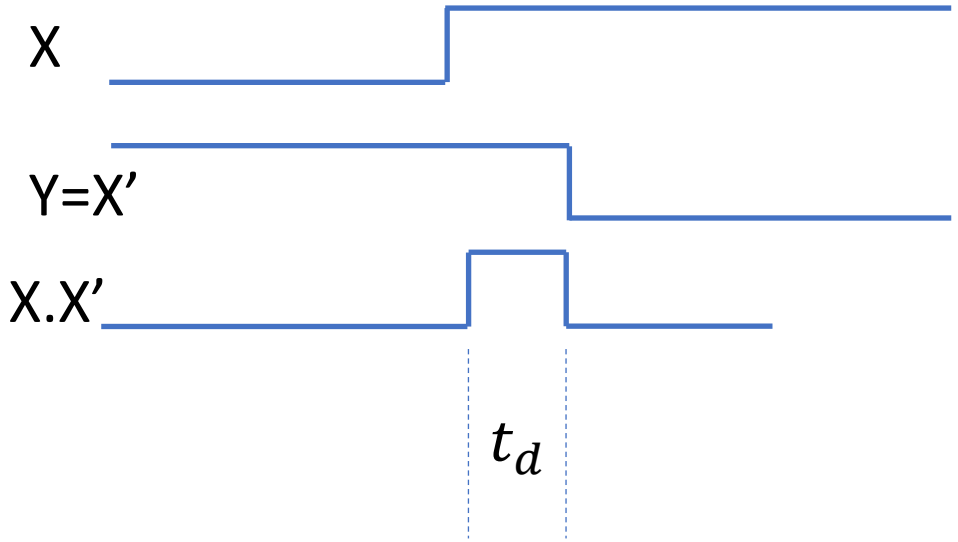




Glitches

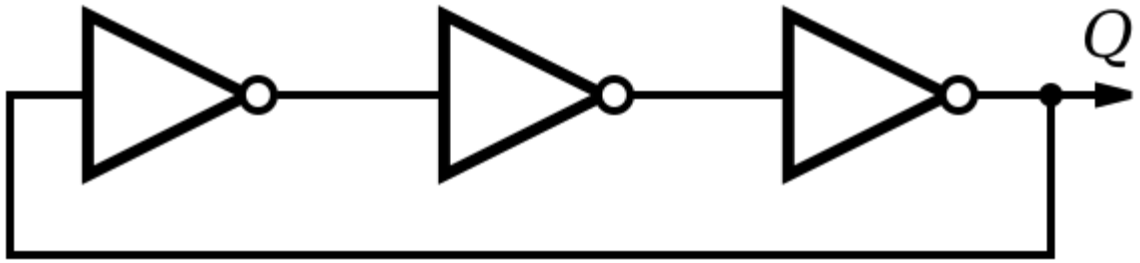
- *Improperly handled timing can lead to glitches and cause unexpected results*
- This is particularly true for multi-level logic implementation wherein different literals are bypassing some levels
- Consider an AND gate connected to X and X’
- Such a connection will ALWAYS produce a glitch for X (0 -> 1)
- You can be sad about this glitch, or you can call this circuit a “pulse generator”! ☺

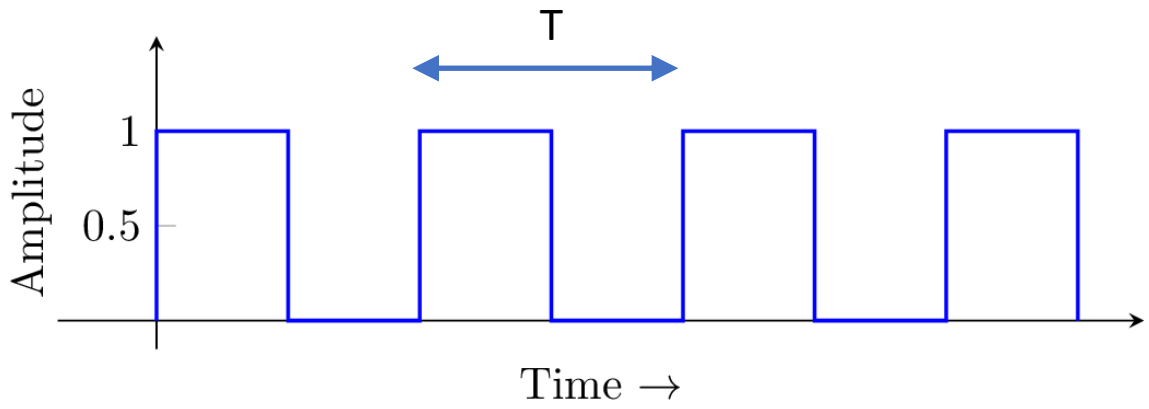




Ring oscillators

- Another clever use of gate delays in is in the construction of ring oscillators
- It is a series of odd number of NOT gates with the output connected back to the input gate of the first gate
- With this system, we can generate a continuous square wave at Q
- What is the frequency of the wave?
- In this case, it is $f = \frac{1}{T} = \frac{1}{3 \times 2t_d}$



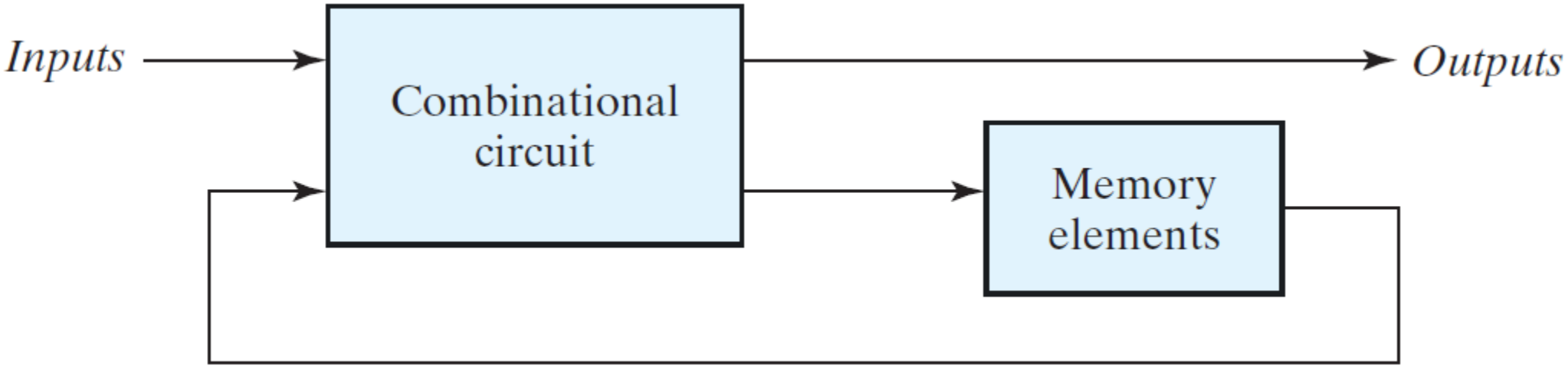


Sequential circuits

- The technology enabling and supporting modern digital devices is critically dependent on electronic components that can store information, i.e., have memory
- We will examine the operation and control of these devices and their use in circuits and enables you to better understand what is happening in these devices when you interact with them
- The digital circuits considered thus far have been combinational—their output depends only and immediately on their inputs—they have no memory, i.e., dependence on past values of their inputs
- Sequential circuits, however, act as storage elements and have memory, i.e., their output depends on past inputs as well

Sequential circuits

- The main way sequential circuits remember things is through feedback paths and memory elements
- We know how combinational circuits are created
- The simplest storage elements (memory) used in sequential circuits are called *latches*



Latches

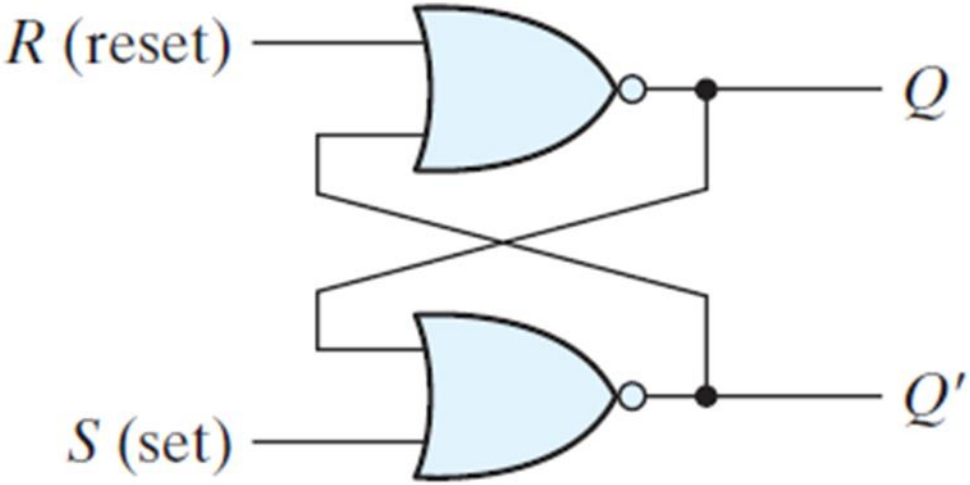
- A storage element in a digital circuit can maintain a binary state indefinitely (as long as power is delivered to the circuit), until directed by an input signal to switch states

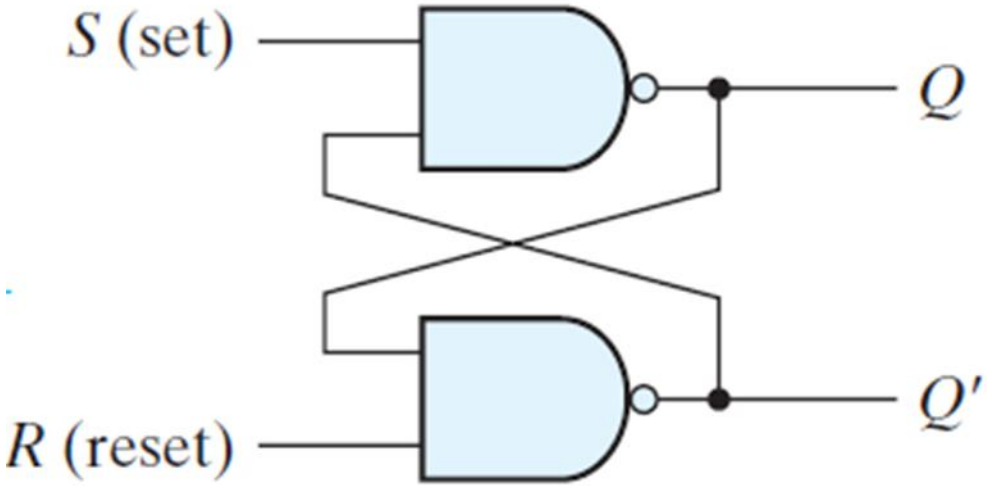
- Such a storage element is called a *latch*

- The major differences among various types of storage elements are in the number of inputs they possess and in the manner in which the inputs affect the binary state

SR Latch

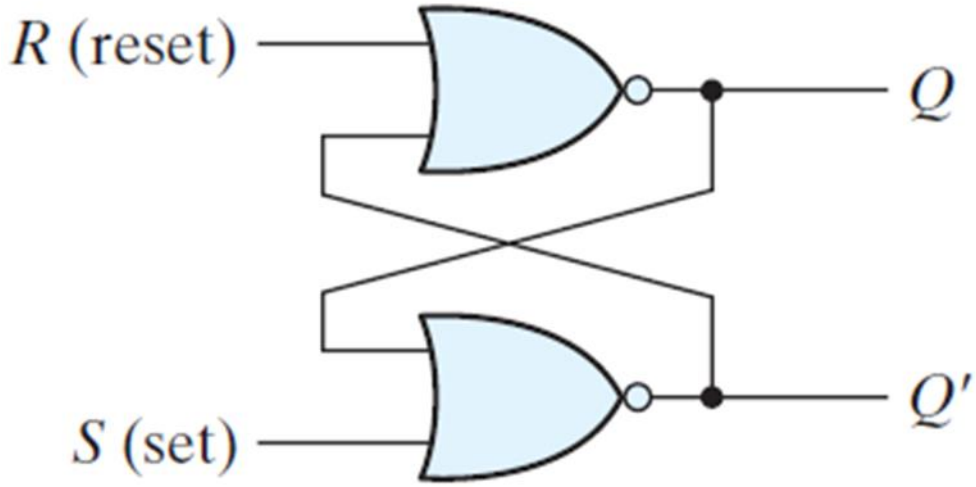
- The (Set – Reset) *SR* latch is a circuit with two cross-coupled NOR gates or two cross-coupled NAND gates, and two inputs labeled *S* and *R*





SR Latch

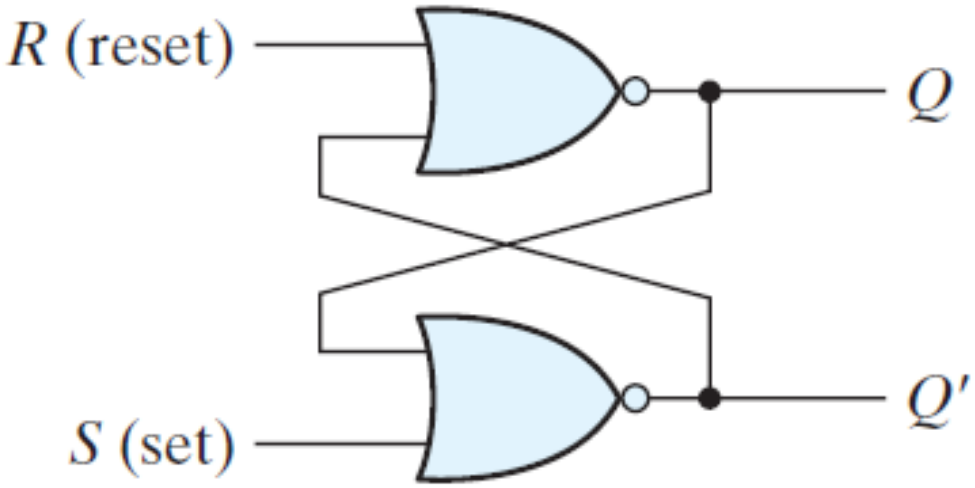
- We realize that the outputs Q and Q' are normally the complement of each other



- The latch has two stable states
- When output $Q = 1$ and $Q' = 0$, the latch is said to be in the *set state*
- When $Q = 0$ and $Q' = 1$, it is in the *reset state*

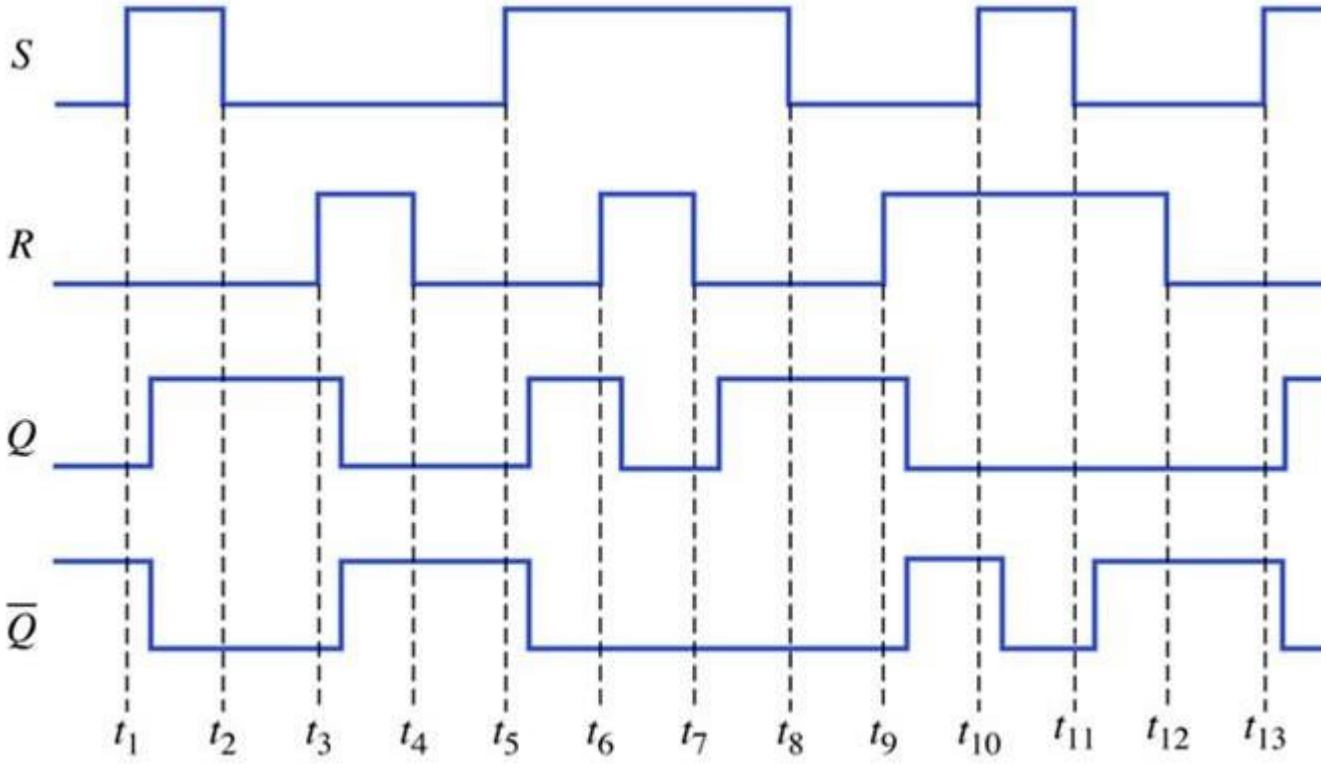
SR Latch

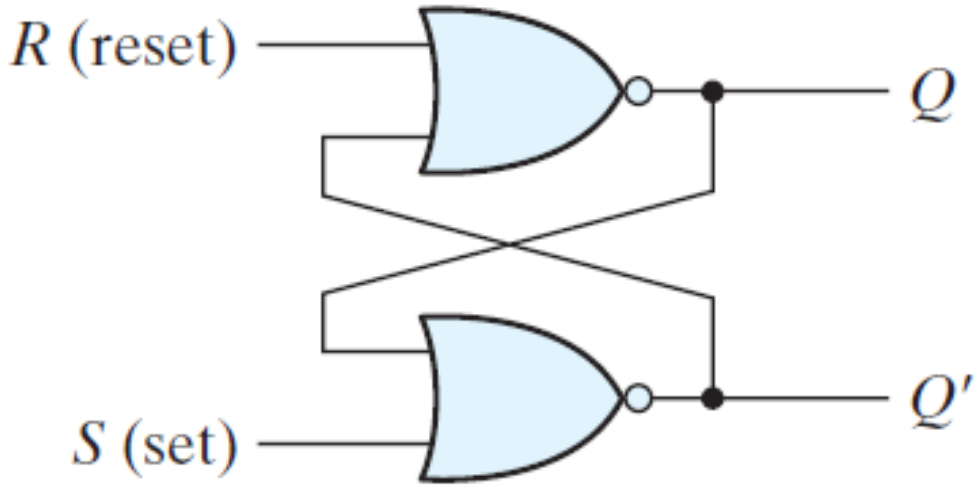
- Setting and resetting can be done through the two inputs (S,R)
- After setting or resetting, applying 00 at the input retains the previous state
- However, when both inputs are equal to 1 at the same time, a condition in which both outputs are equal to 0 (rather than be mutually complementary) occurs
- If both inputs are then switched to 0 simultaneously, the device will enter an unpredictable or undefined state or a metastable state



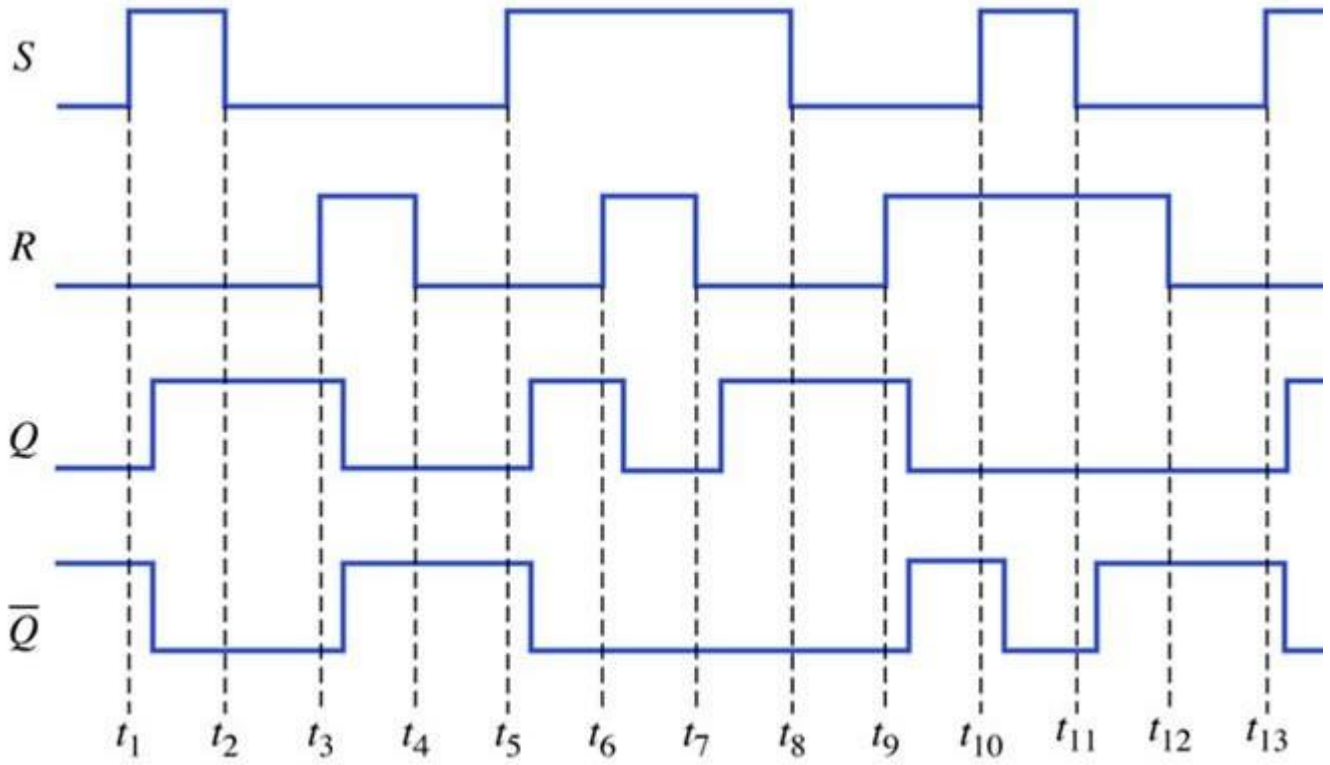
S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	(forbidden)

SR Latch



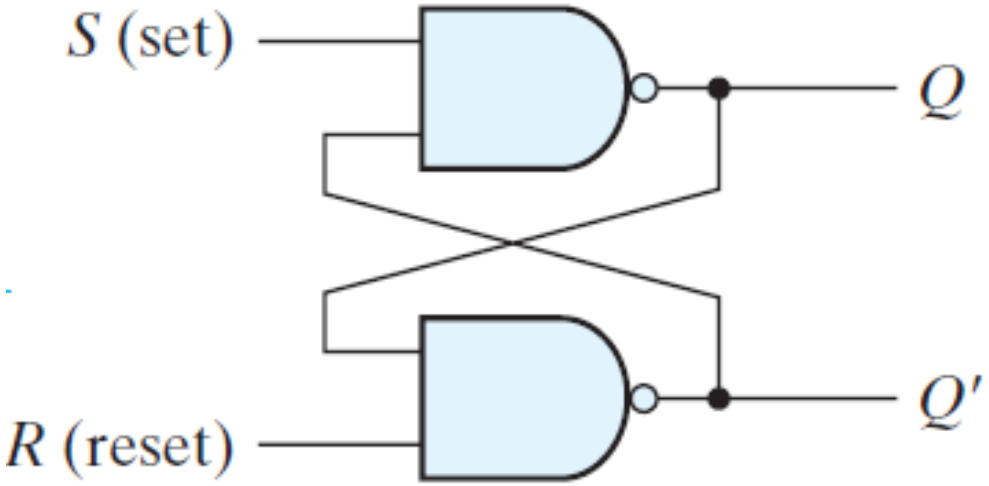


S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	(forbidden)


S
R
Q
$\overline{Q}$
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13}

SR Latch – NAND implementation

- The *SR* latch with two cross-coupled NAND gates behaves in a similar way to NOR implementation
- It operates with both inputs normally at 1, unless the state of the latch has to be changed
- The application of 0 to the *S* input causes output *Q* to go to 1, putting the latch in the set state
- When the *S* input goes back to 1, the circuit remains in the set state
- The condition that is forbidden for the NAND latch is both inputs being equal to 0 at the same time, an input combination that should be avoided



S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(after $S = 1, R = 0$)
0	1	1	0	
1	1	1	0	(after $S = 0, R = 1$)
0	0	1	1	(forbidden)