Unit 04 Fundamental of Digital Circuits

Half Adder and Full Adder

ICT A/L 2020

SAMANTHA RAJAPAKSHA

M.Sc. in IT (SLIIT) B.Sc.(Engineering) University of Moratuwa

Contact: 0710826165

www.64bits.lk

Half Adder

- A half adder is used to add two binary digits together, A and B.
- It produces S, the sum of A and B, and the corresponding carry out C.
- Although by itself, a half adder is not extremely useful, it can be used as a building block for larger adding circuits (Full Adder).

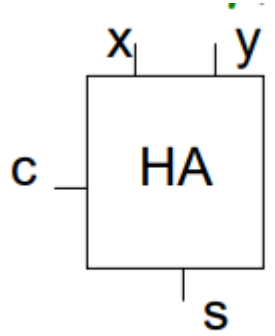
X	0	0	1	1
+ Y	+ 0	+ 1	+ 0	+ 1
C S	0 0	0 1	0 1	1 0

X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Half Adder

x	y	c
0	0	0
0	1	0
1	0	0
1	1	1

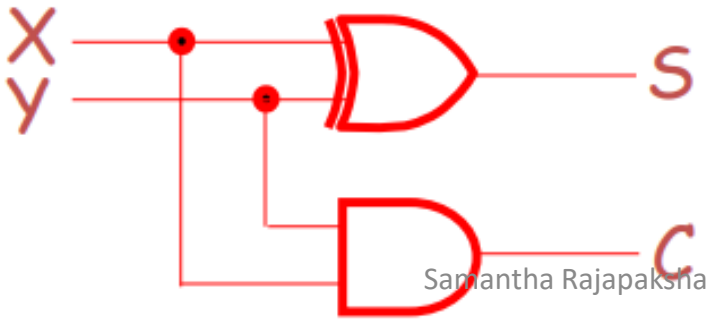
x	y	s
0	0	0
0	1	1
1	0	1
1	1	0



half-adder symbol

$$C = X \cdot Y$$

$$S = X \cdot \overline{Y} + \overline{X} \cdot Y = X \oplus Y$$



Full Adder

- A full adder is similar to a half adder, but includes a carry-in bit from lower stages.
- Like the half adder, it computes a sum bit, S and a carry bit, C.

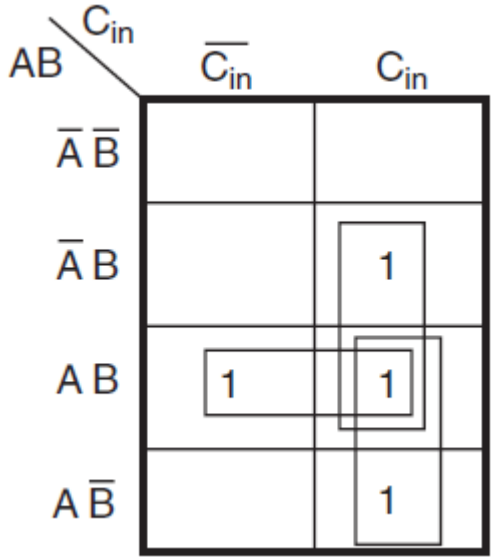
X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Z	0	0	0	0
X	0	0	1	1
+ Y	+ 0	+ 1	+ 0	+ 1
C S	0 0	0 1	0 1	1 0

Z	1	1	1	1
X	0	0	1	1
+ Y	+ 0	+ 1	+ 0	+ 1
C S	0 1	1 0	1 0	1 1

Full Adder

$$C_{out} = AB + A\,C_{in} + B\,C_{in}$$



Full Adder Truth table

INPUTS			OUTPUTS	
A	B	C_{in}	SUM	CARRY OUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$C_{out} = \overline{A}.B.C_{in} + A.\overline{B}.C_{in} + A.B.\overline{C}_{in} + A.B.C_{in}$$

$$A \oplus B = A \cdot \overline{B} + \overline{A} \cdot B$$

Full Adder

Full Adder Truth table

INPUTS			OUTPUTS	
A	B	C_{in}	SUM	CARRY OUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

XOR (Exclusive-OR)

$x \oplus y$

$\overline{x}\,y + x\,\overline{y}$

x	y	XOR
0	0	0
0	1	1
1	0	1
1	1	0

XNOR (Exclusive-NOR)
(Equivalence)

$\overline{x \oplus y}$

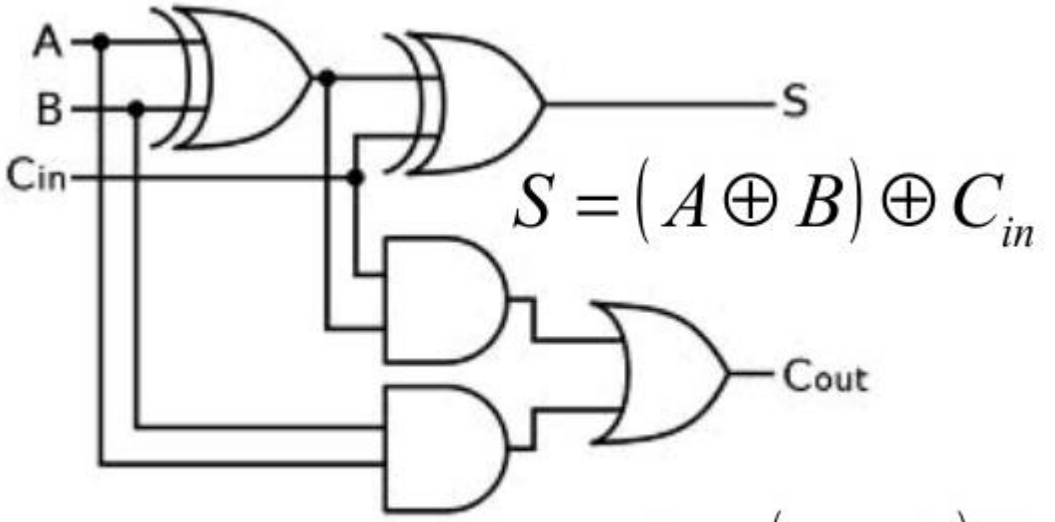
$x \odot y$

$\overline{x}\,\overline{y} + x\,y$

x	y	XNOR
0	0	1
0	1	0
1	0	0
1	1	1

$$\begin{aligned}
\text{Sum} &= \overline{A}\ \overline{B}\ C_{in} + \overline{A}\ B\ \overline{C}_{in} + A\ \overline{B}\ \overline{C}_{in} + A\ B\ C_{in} \\
&= C_{in}\ (\overline{A}\ \overline{B} + AB) + \overline{C}_{in}\ (\overline{A}\ B + A\ \overline{B}) \\
&= C_{in}\ (A \odot B) + \overline{C}_{in}\ (A \oplus B) \\
&= C_{in}\ (\overline{A \oplus B}) + \overline{C}_{in}\ (A \oplus B) \\
&= C_{in} \oplus (A \oplus B)
\end{aligned}$$

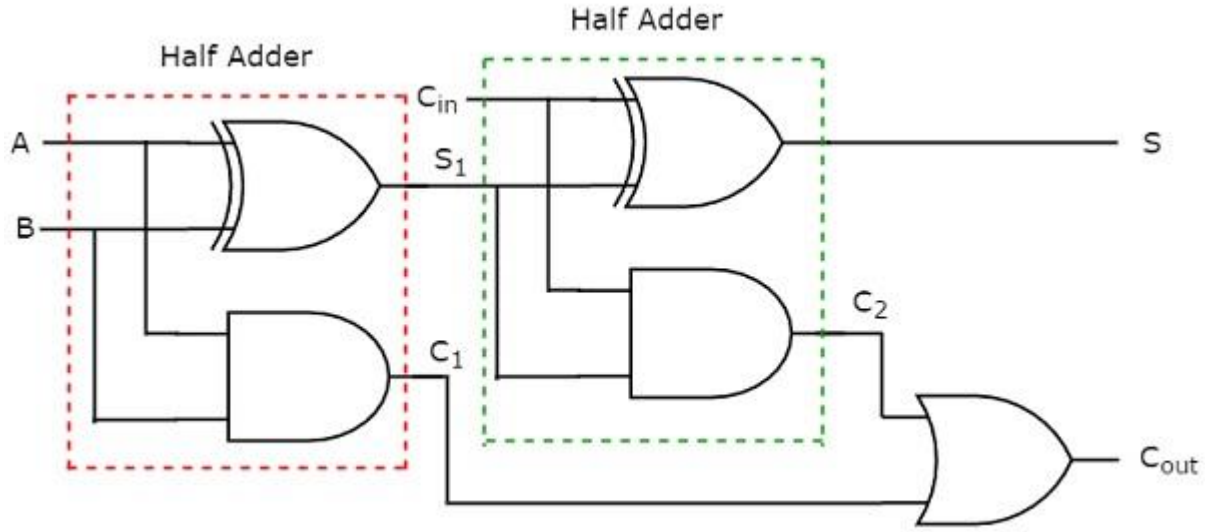
Full Adder using two half Adder



$$S = (A \oplus B) \oplus C_{in}$$

$$C_{out} = AB + (A \oplus B)C_{in}$$

$$C_o = \bar{A}BC_i + A\bar{B}C_i + AB\bar{C}_i + ABC_i$$
$$= C_i(\bar{A}B + A\bar{B}) + AB(\bar{C}_i + C_i)$$
$$= C_i(A \oplus B) + AB$$



Flip Flop

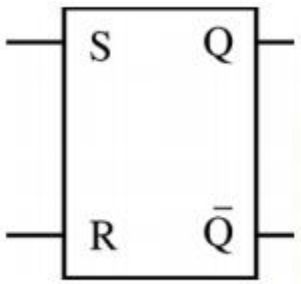
- **Combinational circuits**: Output depends only on the input of that time.
- **Sequential Circuit**: Output depends not only on the present inputs but also on the previous inputs and outputs.
- This type of circuit is required to perform sequence of actions without getting any further inputs. Use for memory storage (SRAM).
- Flip flops can also be considered as the most basic idea of a Random-Access Memory. When a certain input value is given to them, they will be remembered.

SR Latch

Flip-flops is called a bi-stable device, since it has only two possible stable output states: 1 (high) and 0 (low).

It has the capability of remaining in a particular state until input signal cause it to change.

- S is the SET input, R is RESET input
- Q andQ are the complementary outputs



1. As long as the inputs *S* and *R* are both 0, the outputs of the flip-flop remain unchanged.
2. When *S* is 1 and *R* is 0, the flip-flop is *set* to $Q = 1$ and $\bar{Q} = 0$.
3. When *S* is 0 and *R* is 1, the flip-flop is *reset* to $Q = 0$ and $\bar{Q} = 1$.
4. It is "not allowed" (NA) to place a 1 on *S* and *R* simultaneously since the output will be unpredictable.

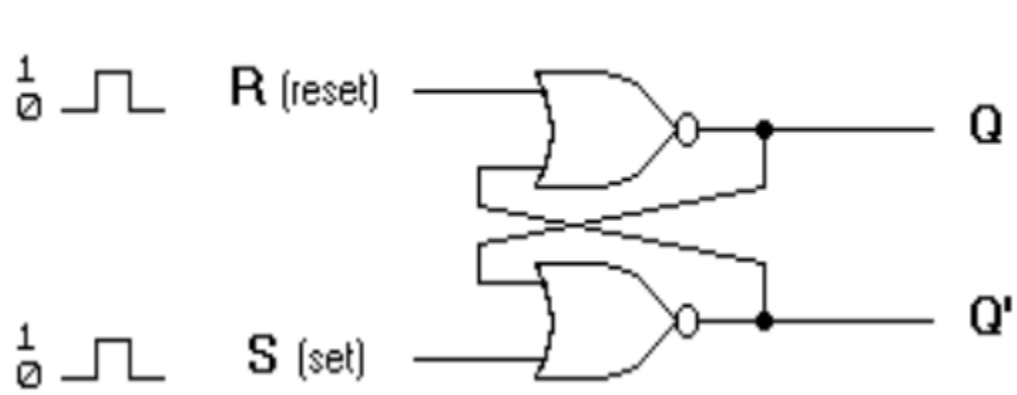
SR Latch

A flip-flop circuit can be constructed from two NAND gates or two NOR gates.

Flip-flop has two outputs, Q and , and two inputs, set and reset.

This type of flip-flop is referred to as an SR flip-flop or SR latch.

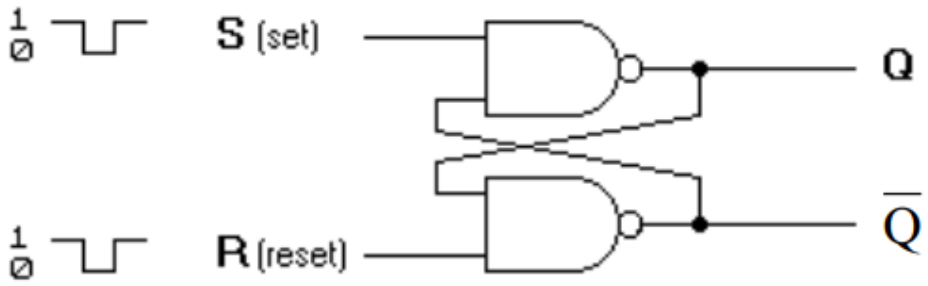
Basic flip-flop circuit with NOR gates



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	↖ NA

SR Latch

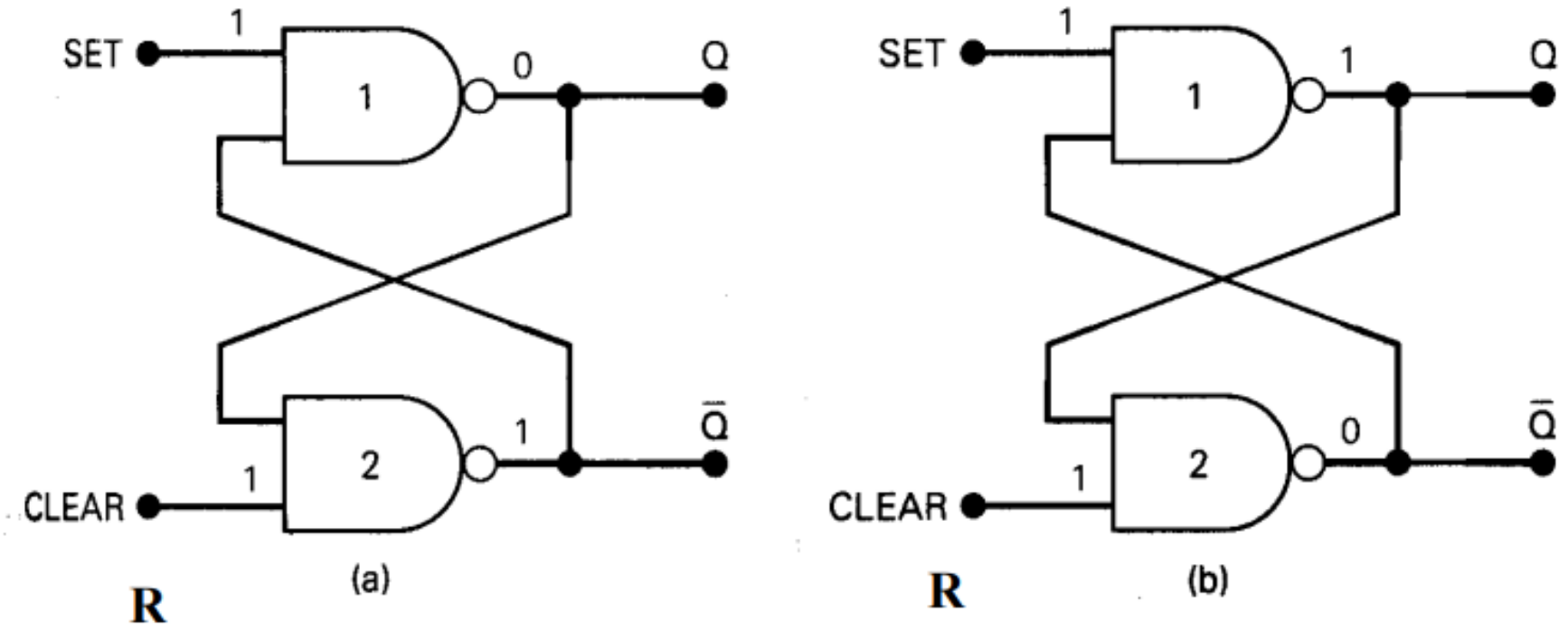
Basic flip-flop circuit with NAND gates (NAND latch)



S	R	Q	$\overline{Q}$	
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	↖ NA

SR Latch Using NAND gates

NAND latch has two possible resting states when SET = CLEAR = 1.



After S=1, R=0

After S=0, R=1

Figure Pulsing the SET input to the 0 state will always produce the $Q = 1$, $\overline{Q} = 0$: output state (a) $Q = 0$ prior to SET pulse; (b) $Q = 1$ prior to SET pulse.

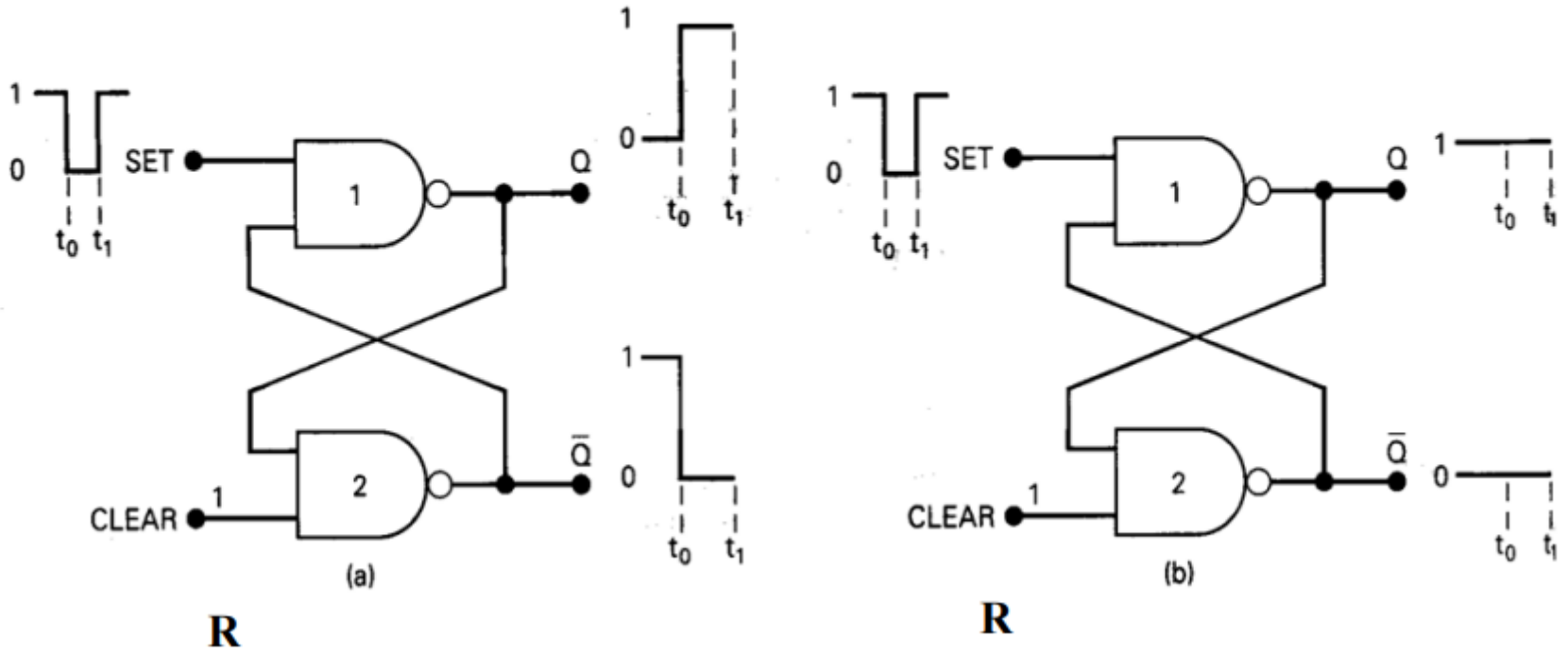
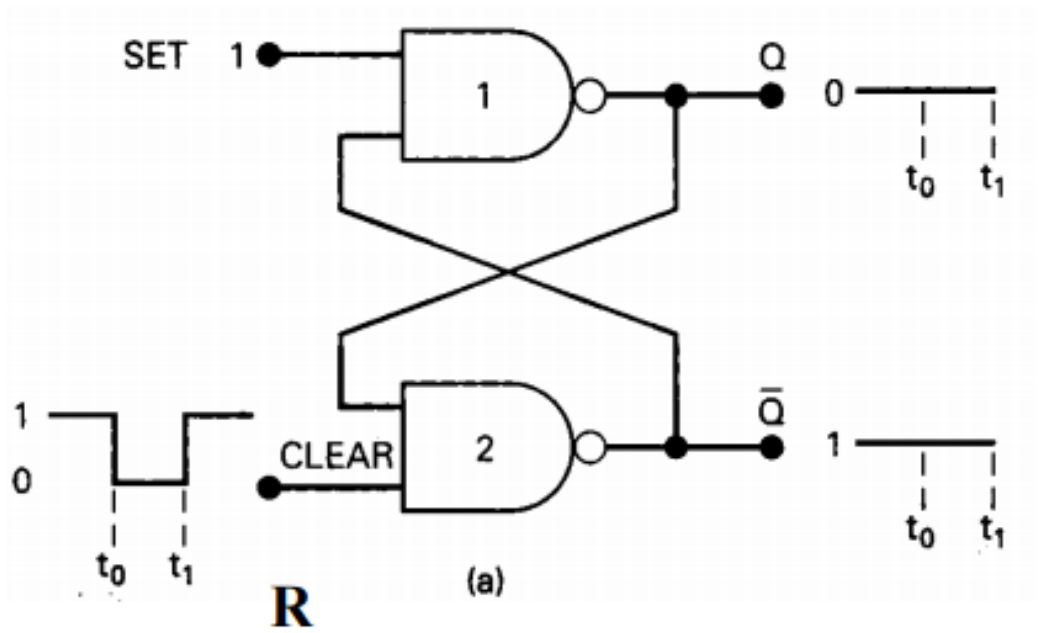
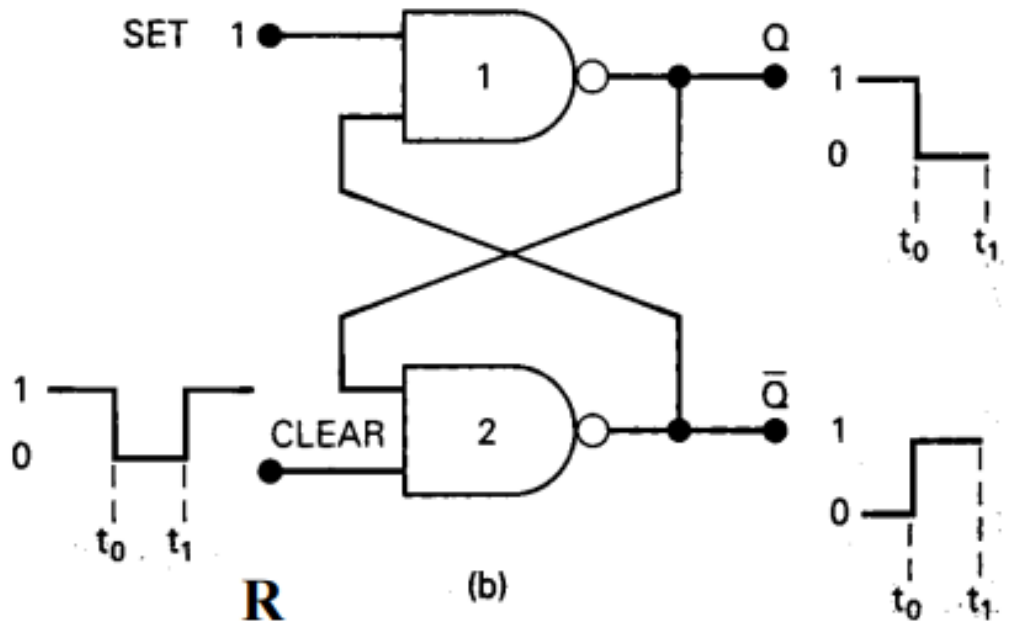
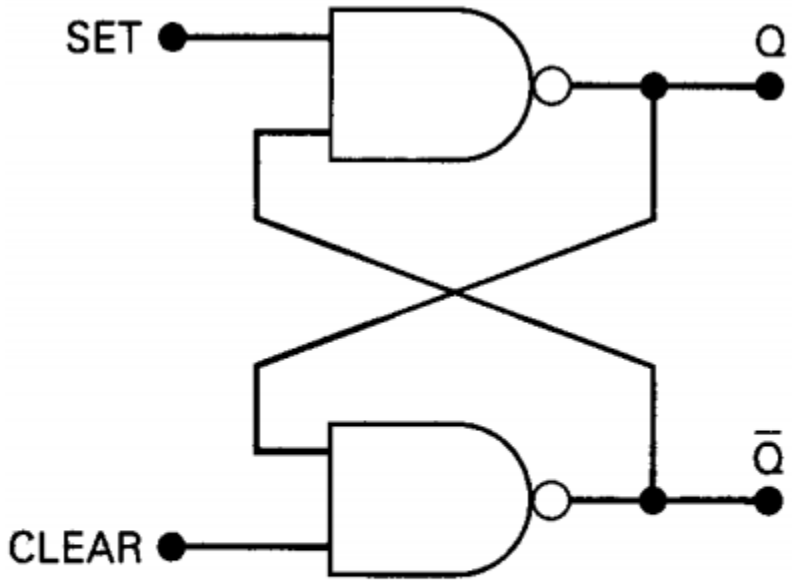


Figure Pulsing the CLEAR input to the LOW state will always produce $Q = 0$, $\overline{Q} = 1$:
(a) $Q = 0$ prior to CLEAR pulse; (b) $Q = 1$ prior to CLEAR pulse.







(a)

Set	Clear	Output
1	1	No change
0	1	Q = 1
1	0	Q = 0
0	0	Invalid*

*produces $Q = \overline{Q} = 1$

(b)

(a) NAND latch; (b) truth table.