We wish to synchronize the actions of our computer with a clock pulse. We prefer to use either the rising edge of the clock signal or the falling edge. We find that circuits are too volatile when we use asynchronous circuits – when we wait for signals to come to their final values. Therefore, we look for a way to synchronize state changes in flip-flops.

- When the clock is high, input values $\boldsymbol{S}$ and $\boldsymbol{R}$ control the next value retained by the flip-flop.
- The current state is referred to as $\boldsymbol{Q}$. $\boldsymbol{Q}$ has only two values: 0 and 1.
- Because the value of $\boldsymbol{Q}$ changes over time, we denote the current and next value of $\boldsymbol{Q}$ as $\boldsymbol{Q(t)}$ and $\boldsymbol{Q(t+1)}$ respectively indicating that one clock pulse has occurred between $\boldsymbol{Q(t)}$ and $\boldsymbol{Q(t+1)}$.

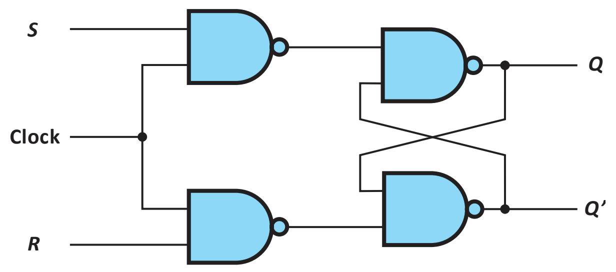


FIGURE 3.31 (f) Clocked SR Flip-Flop implemented using cross coupled NAND gates

- We observe that the SR Flip-Flop has exactly the same inputs and outputs whether it is implemented using cross-coupled NOR gates or cross-coupled NAND gates.
- We prefer the SR Flip-Flop implementation using cross-coupled NAND gates because it is implemented using only NAND gates whereas the clocked SR Flip-Flop implemented with NOR gates requires two AND gates as well as two NOR gates.

S	R	$Q(t)$	$Q(t+1)$	Action
0	0	**0**	0	No change
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	?	Undefined
1	1	1	?	

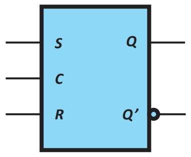
FIGURE 3.31 (c)
SR **Characteristic** Table

$Q(t)$	$\rightarrow$ $Q(t+1)$	S	R	Action
0	0	0	0	No Change
0	0	0	1	Reset
0	1	1	0	Set
1	0	0	1	Reset
1	1	0	0	No Change
1	1	1	0	Set

Expanded SR Excitation Table

$Q(t)$	$\rightarrow$ $Q(t+1)$	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

SR Excitation Table



Clocked SR Flip-Flop Schematic Symbol