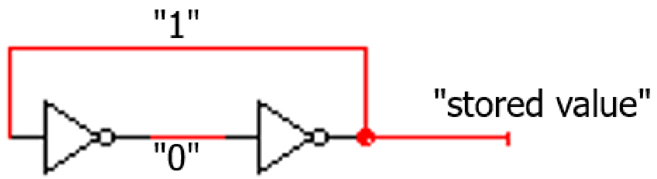
Latches & Flip Flops

Memory In Digital Logic

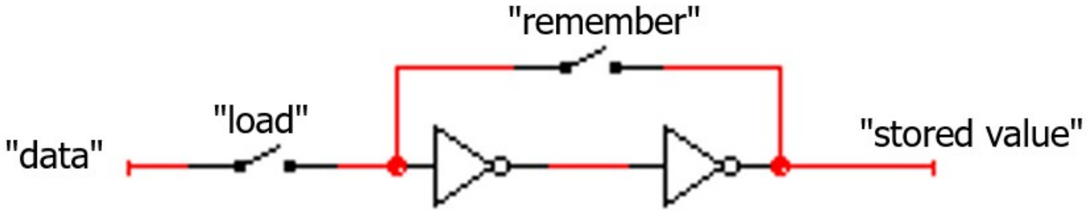
- Advanced logic devices such as latches and flip flops are memory elements
- Allows logic circuits to remember input combinations to generate an output
  - Allows for advanced digital logic designs
- Memory in digital logic is achieved through feedback
- Output of latch or flip flop is referred to as a state

Feedback Concept

- Two inverters example using feed back for memory
  - Output will remain constant



- "load" puts new value in memory
- Closing "remember" keeps value in memory



State

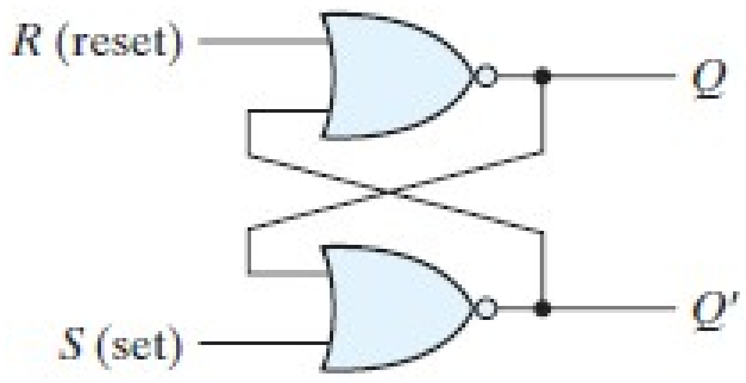
- A state defines what the output of the logic device, or circuit, will be given a set of inputs
- Output may also depend on the previous state (which is held in memory)
- Circuit will transition states as inputs change
  - Output from one state can be input to another state
- Memory elements are used to remember what state a logic circuit is in

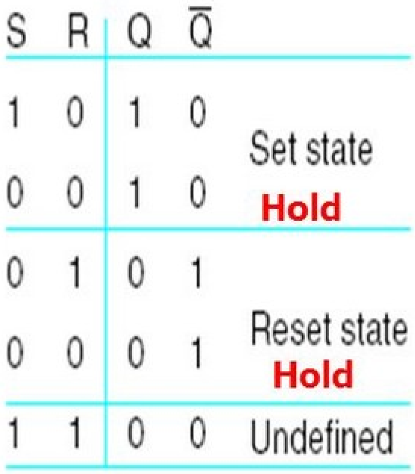
Memory Elements – SR Latch

- Basic storage element that is level triggered
- A latch state is **set** or can be **reset**.
  - If a latch is set, output is 1
  - If a latch is reset, output is 0
- Output (state) is denoted and
- If a latch is set, the state is 1. If a latch is reset, the state is 0.

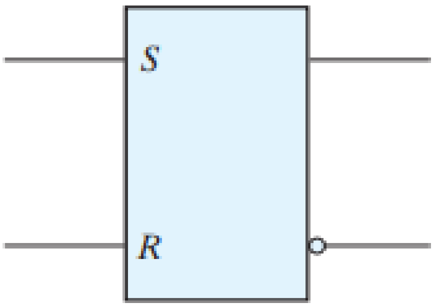
SR Latch

* NOR implementation



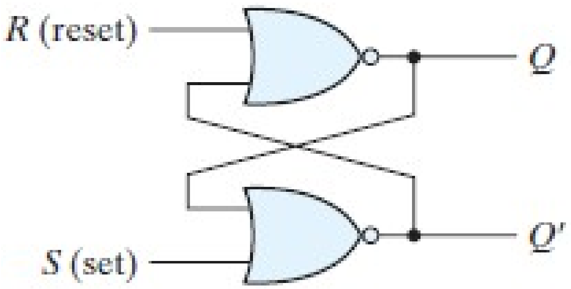


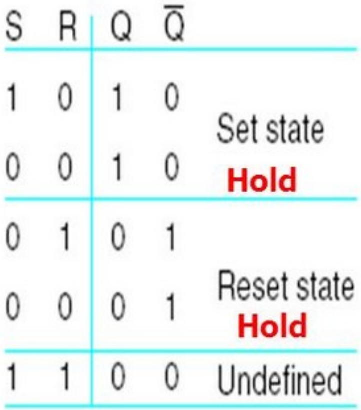
S	R	Q	$\bar{Q}$	
1	0	1	0	Set state
0	0	1	0	**Hold**
0	1	0	1	
0	0	0	1	Reset state **Hold**
1	1	0	0	Undefined



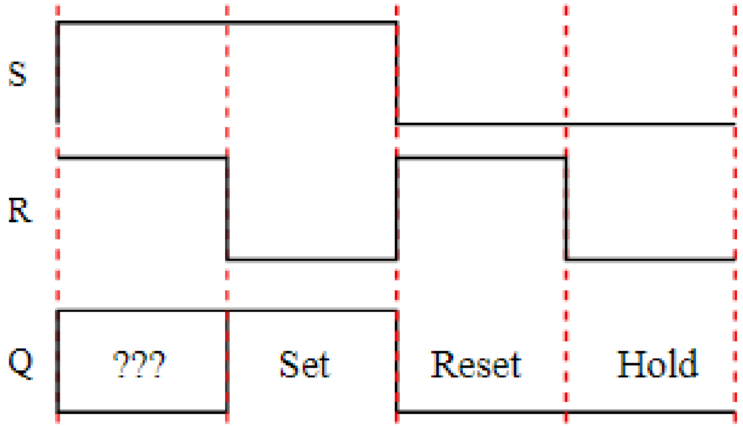
* If both Set and Reset are active, device has “race condition”. Inputs will “race” each other to determine output.

SR Latch Timing Diagram



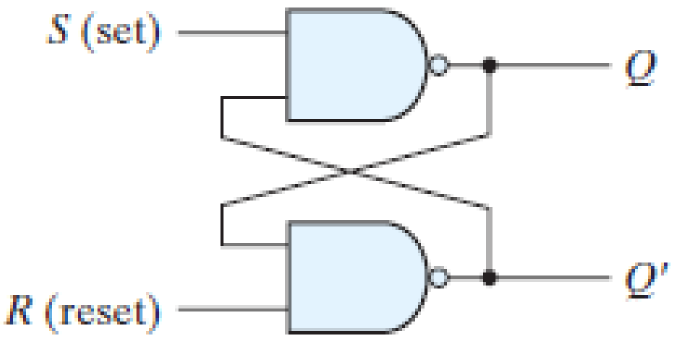


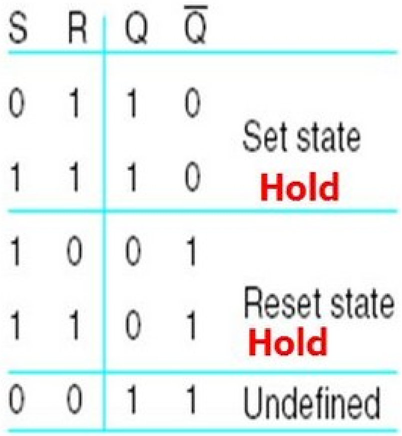
S	R	Q	$\overline{Q}$	
1	0	1	0	Set state
0	0	1	0	**Hold**
0	1	0	1	Reset state
0	0	0	1	**Hold**
1	1	0	0	Undefined



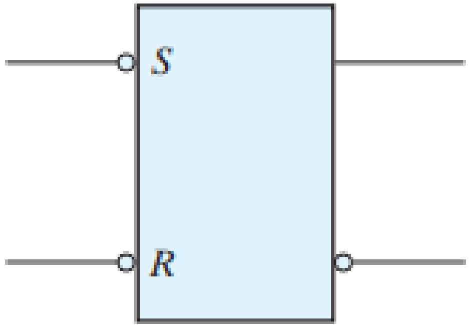
SR Latch

- NAND topology
  - Set and reset are active with 0





S	R	Q	$\overline{Q}$	
0	1	1	0	Set state
1	1	1	0	Hold
1	0	0	1	
1	1	0	1	Reset state Hold
0	0	1	1	Undefined



Latch Analogy

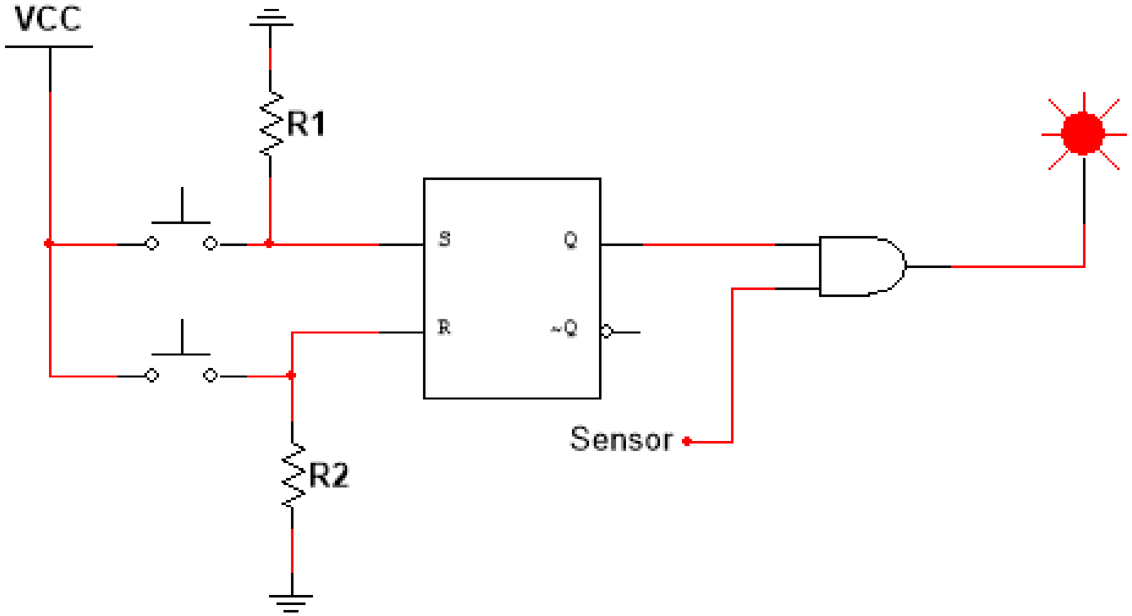
* A door can be locked (set) or unlocked (reset) with a key

* When a key is inserted and turned to locked, it is said to be latched. The key doesn’t need to remain in the lock for the door to be locked.

* When key is inserted and turned to unlock, the door is (reset, unlatched) unlocked. The key does not need to remain in the lock for the door to be unlocked.

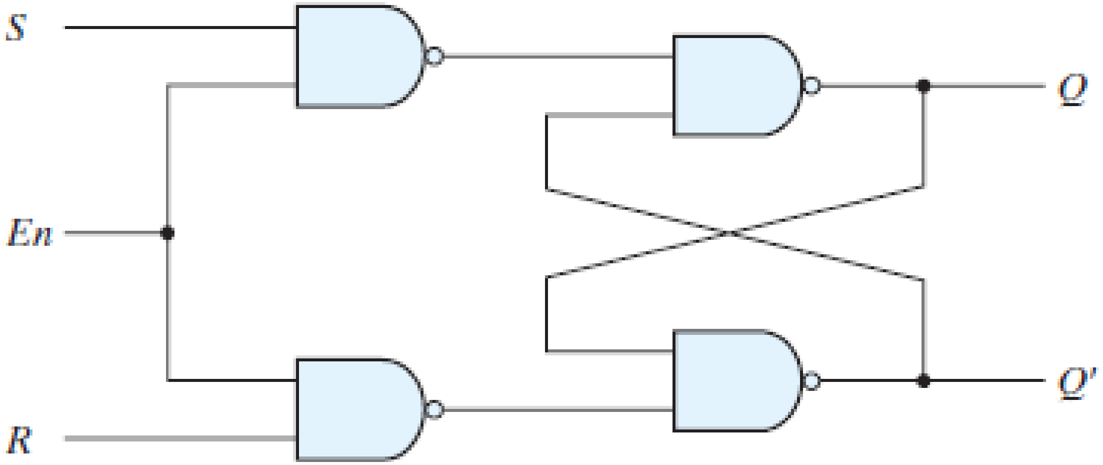
Use of SR Latch

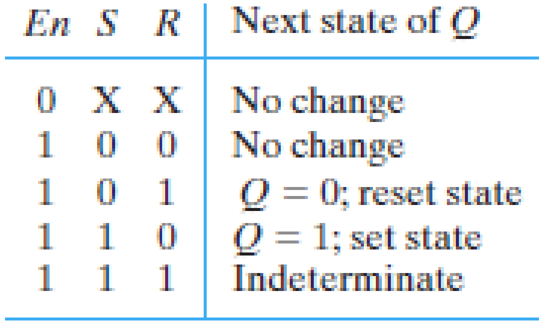
- When anything needs to be "set, reset" in digital logic circuit
  - Digital alarm circuit
  - Keep LED on until reset
  - Switch debouncing



Adding An Enable

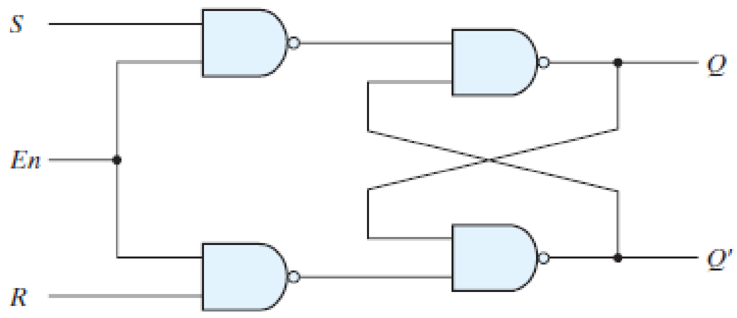
* A 3rd input can act as a control signal, enabling or disabling the latch

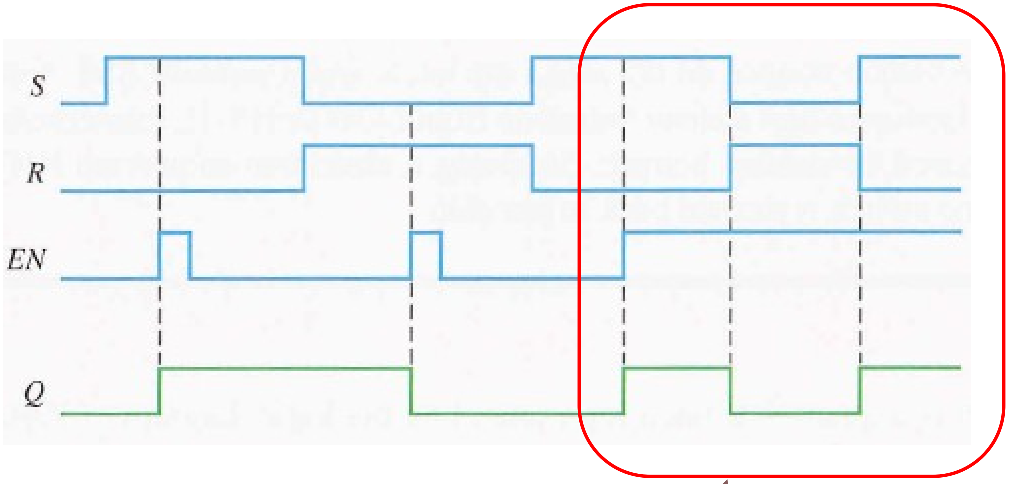




En	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$; reset state
1	1	0	$Q = 1$; set state
1	1	1	Indeterminate

SR Latch with Enable Timing Diagram

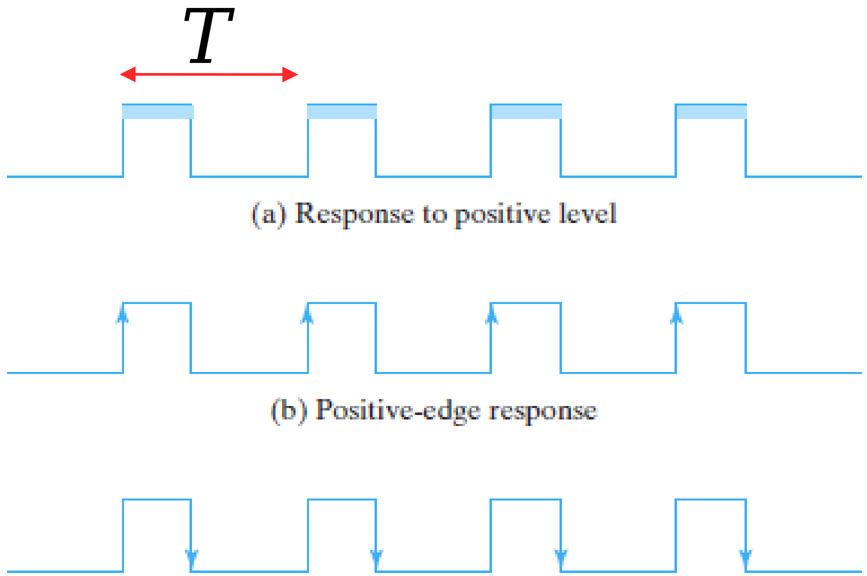




As long as EN is HIGH, the output will respond to the input

Clocks

- Series of HIGH & LOW pulses at a specific frequency
- Logic circuits can operate on the clock edges, or clock level



(a) Response to positive level

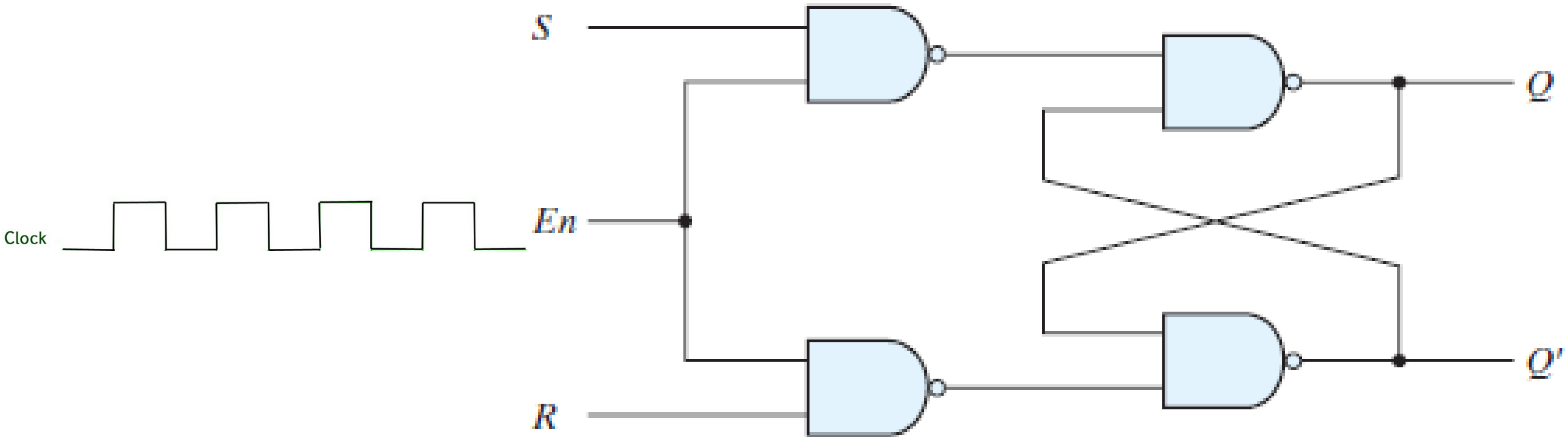
(b) Positive-edge response

(c) Negative-edge response

$$f = \frac{1}{T}$$

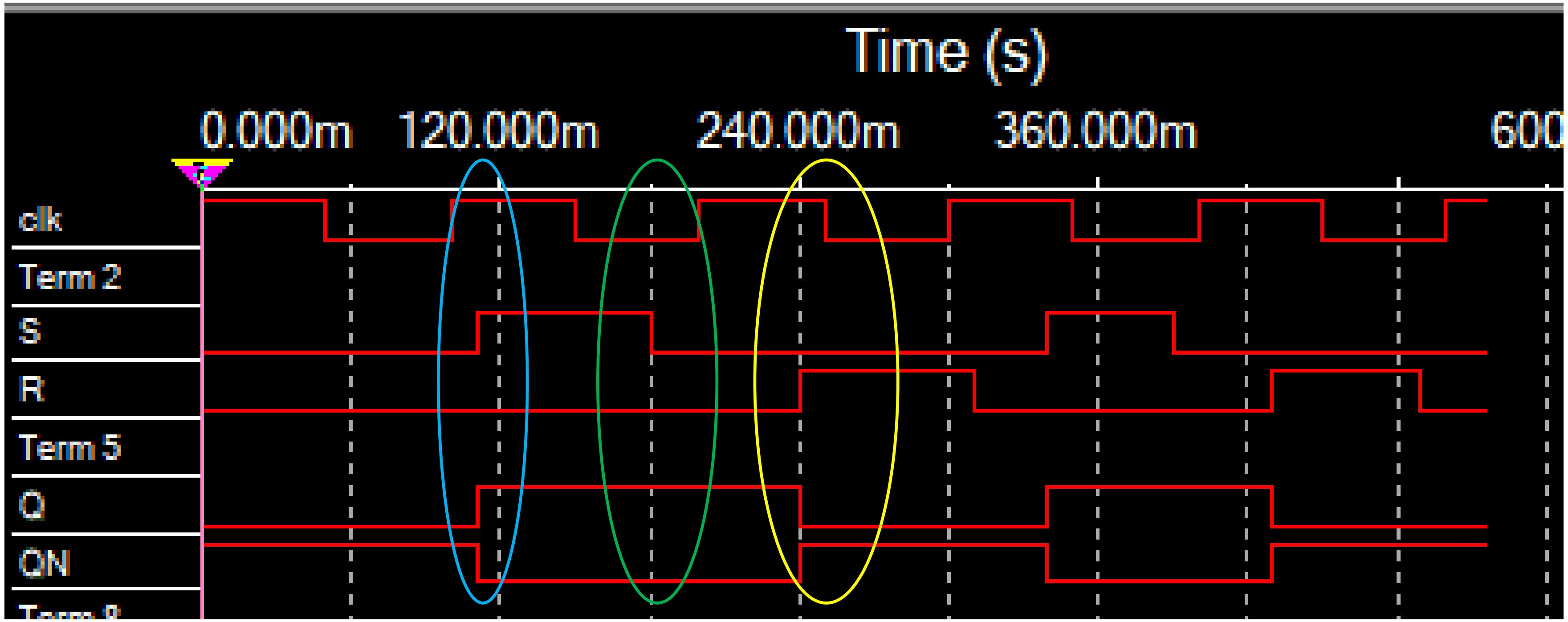
SR Latch & Clock

* If the enable is a clock signal, SR latch can be kept in sync with a clocking source



Timing Diagram

* The output only changes when the enable (clk) is HIGH level



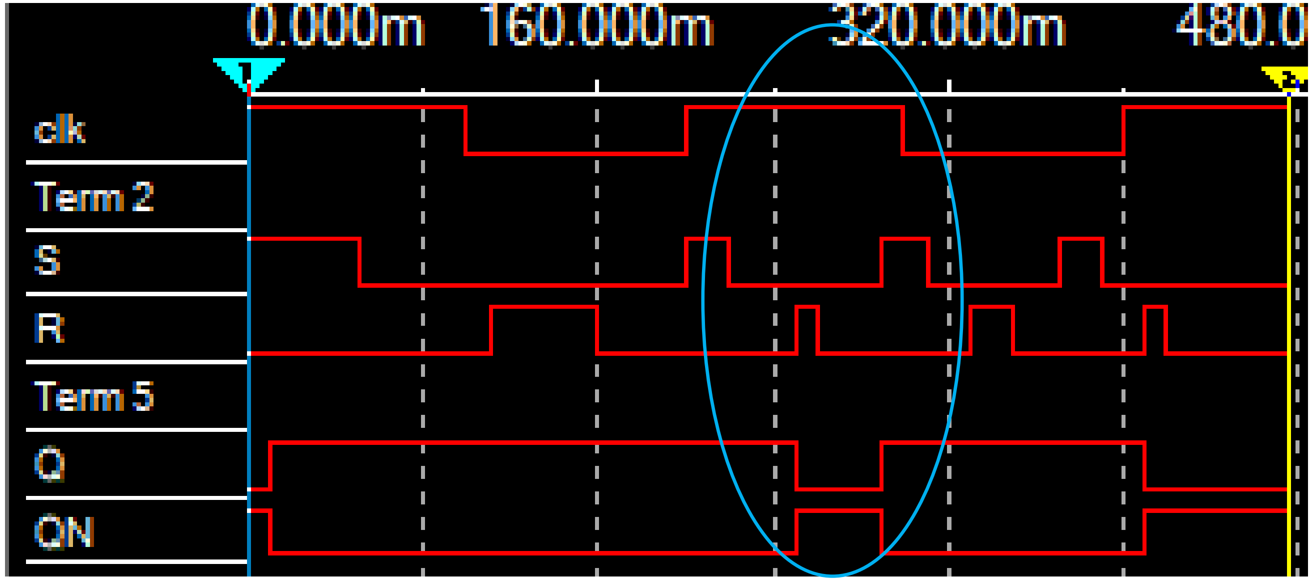
Output set

Output “latched”

Output reset

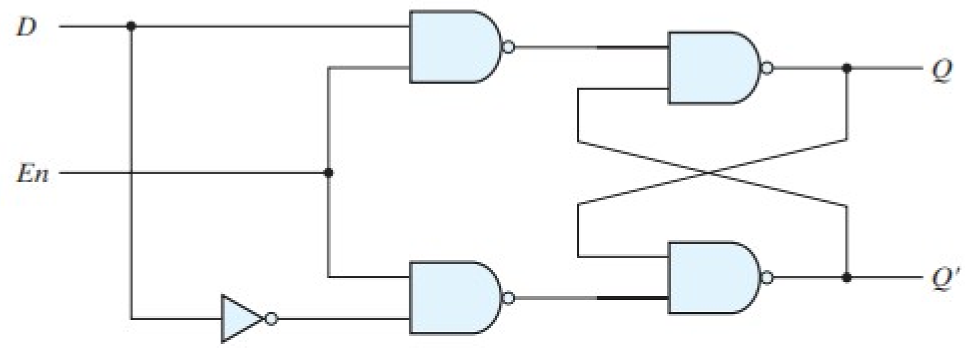
Timing Diagram

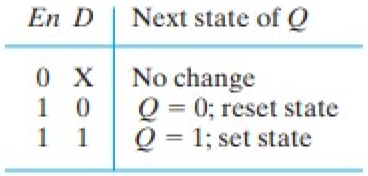
* Possible for multiple transitions in output within single clock pulse



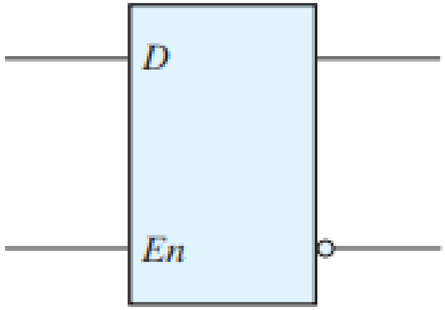
D Latch

* Connecting inputs together of SR to create D Latch
* Single input D (Data). Whenever latch is enabled, Q is the value of D
* Removes unstable state of SR latch



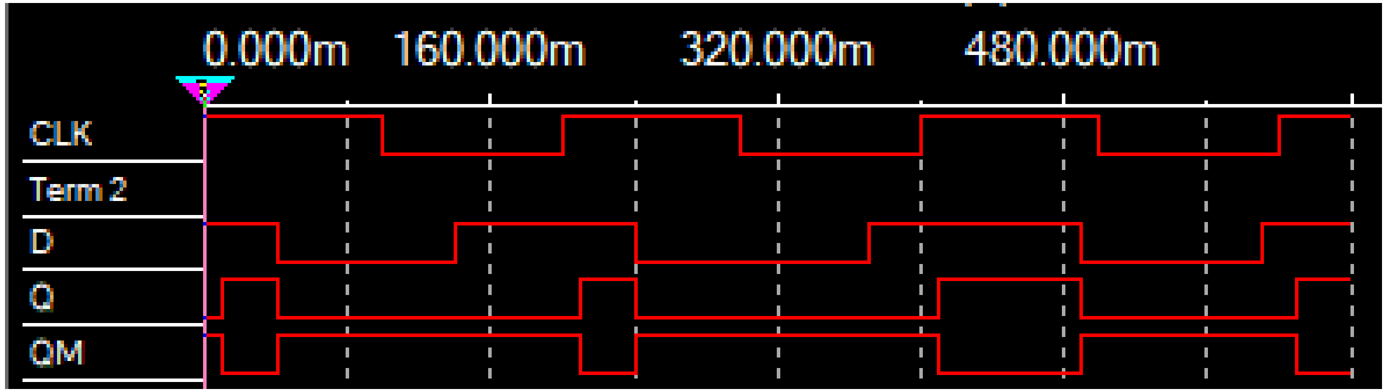


En	*D*	Next state of *Q*
0	X	No change
1	0	*Q* = 0; reset state
1	1	*Q* = 1; set state



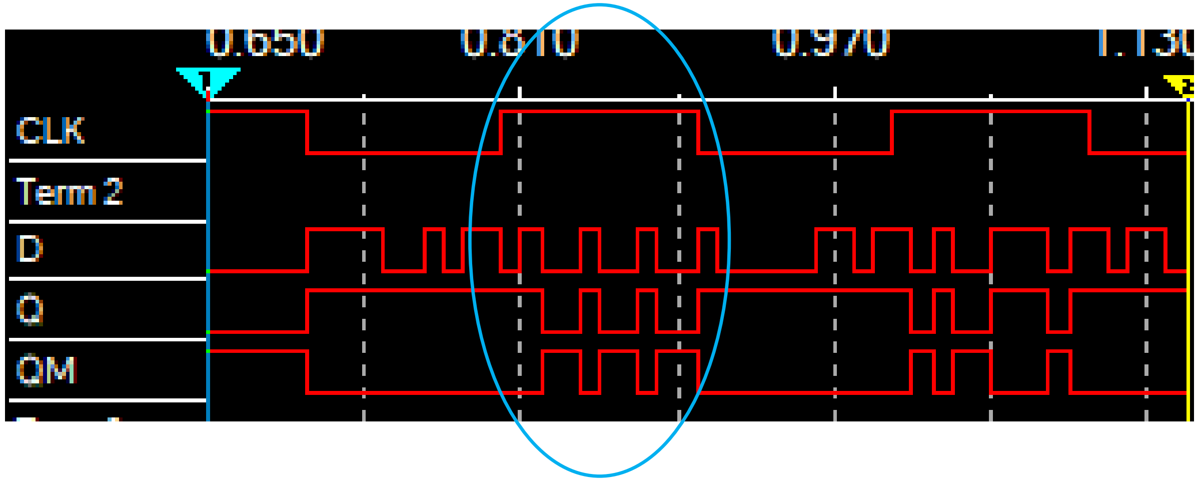
D Latch Timing Diagram

* The output follows the value of D when clock is HIGH



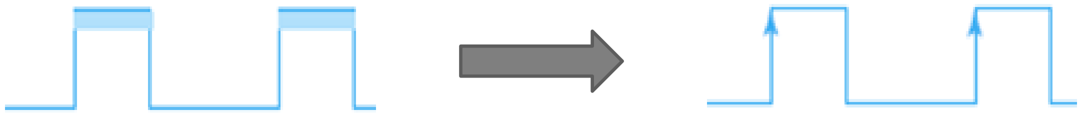
Timing Diagram

- The value of D can change multiple times within one clock pulse



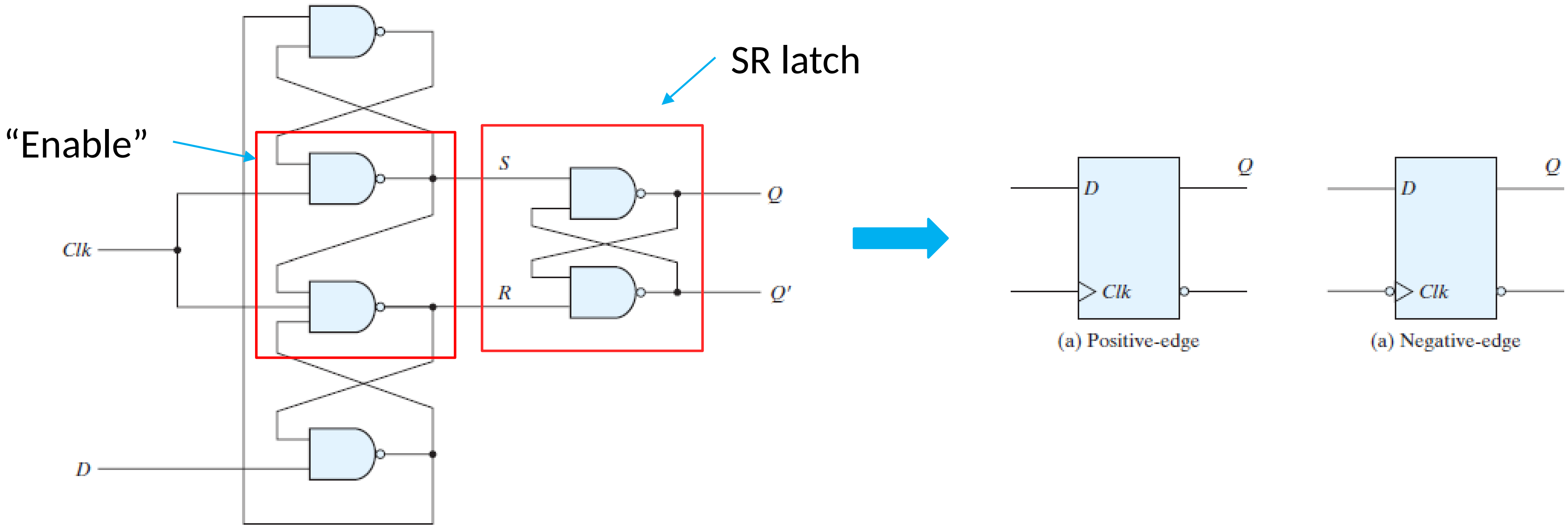
Latch Problem

- While enable or clock is HIGH output will respond to any change in input
  - Latch can exhibit asynchronous behavior
- Need to make a transition enable
  - Enable is only active when transitioning from LOW to HIGH or vice versa
- Only register input on clock’s edges (rising or falling)



D Flip Flop

- Flip flop has D latch behavior, but operates on a clock edge



“Enable”

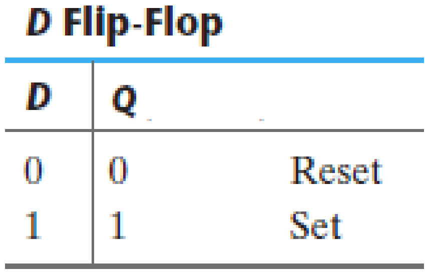
SR latch

(a) Positive-edge

(a) Negative-edge

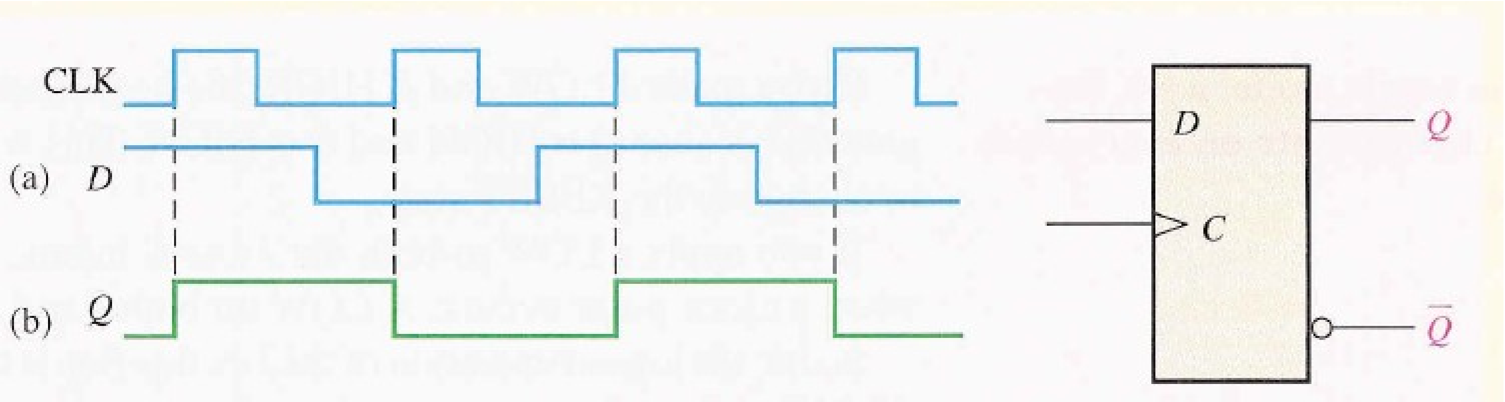
D Flip Flop

* Q only changes on clock edges
  - Will not register any change on D until clock edge



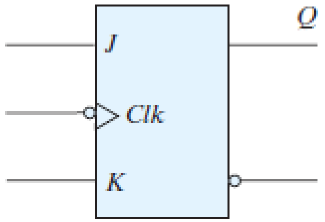
D Flip-Flop

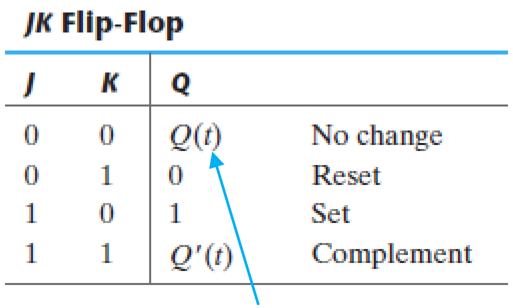
D	Q	
0	0	Reset
1	1	Set



JK Flip Flop

- J is “set” and K is “reset”
- Equivalent to SR Latch, but edge triggered and no unstable input condition

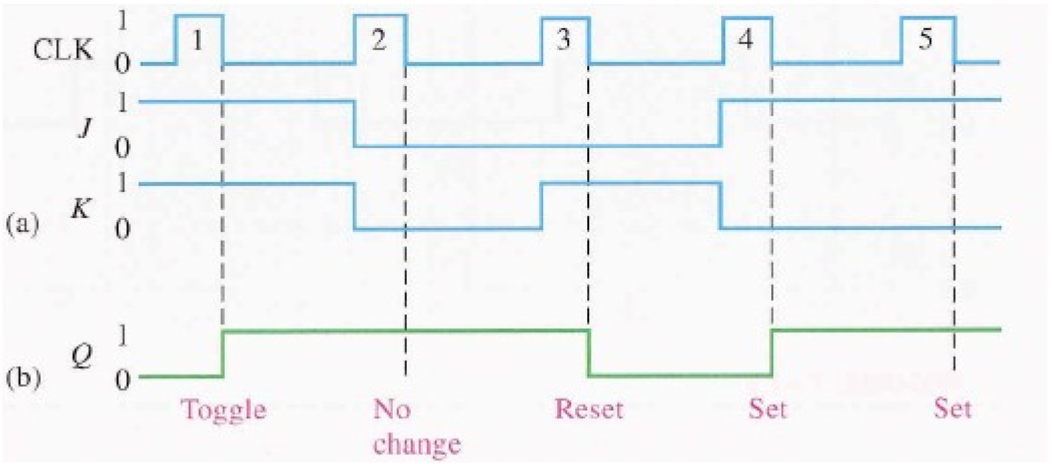




JK Flip-Flop

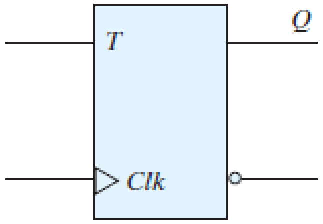
J	K	Q	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

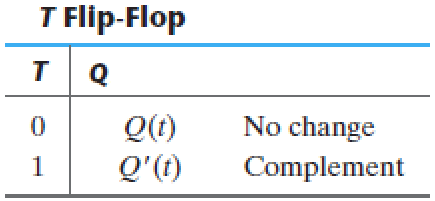
Denotes current state
(or current value of output)



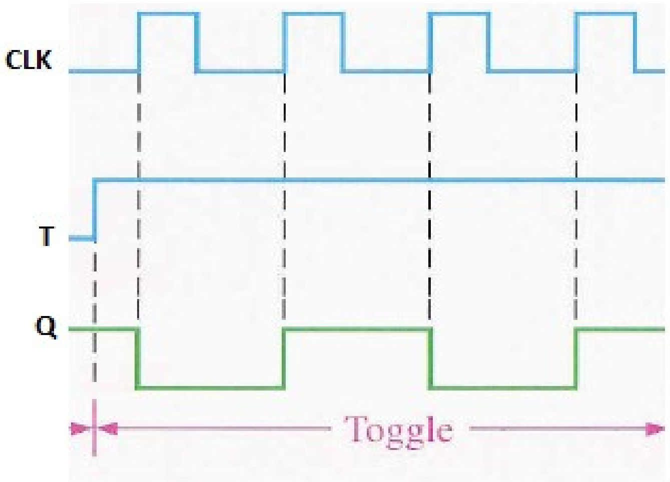
T Flip Flop

- Output toggles between HIGH and LOW as long as T is HIGH



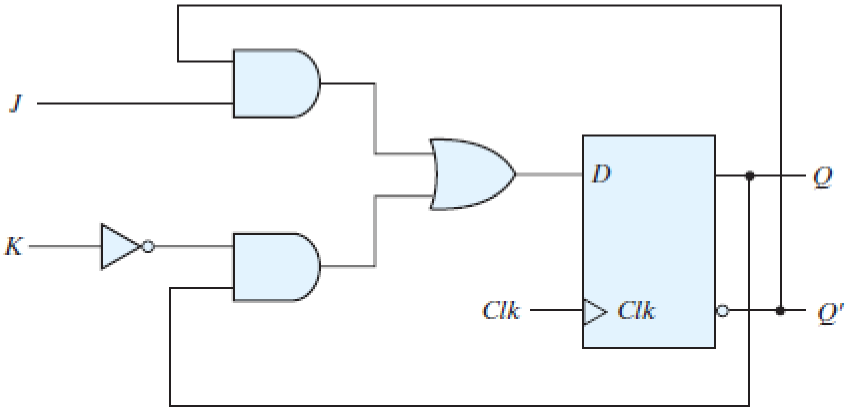

T Flip-Flop

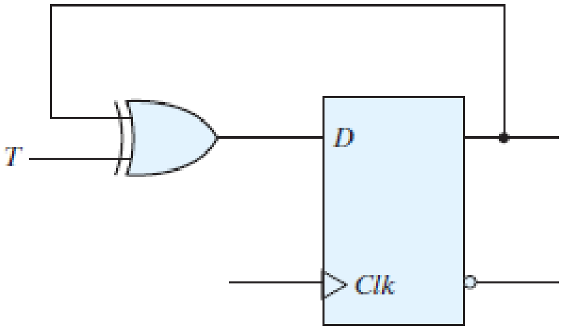
T	Q	
0	$Q(t)$	No change
1	$Q'(t)$	Complement



Making Other Flip Flops From D

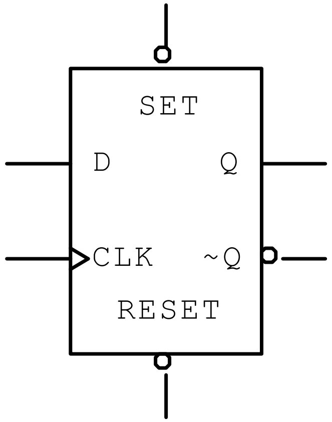
- D flip flop can make JK and T flip flops

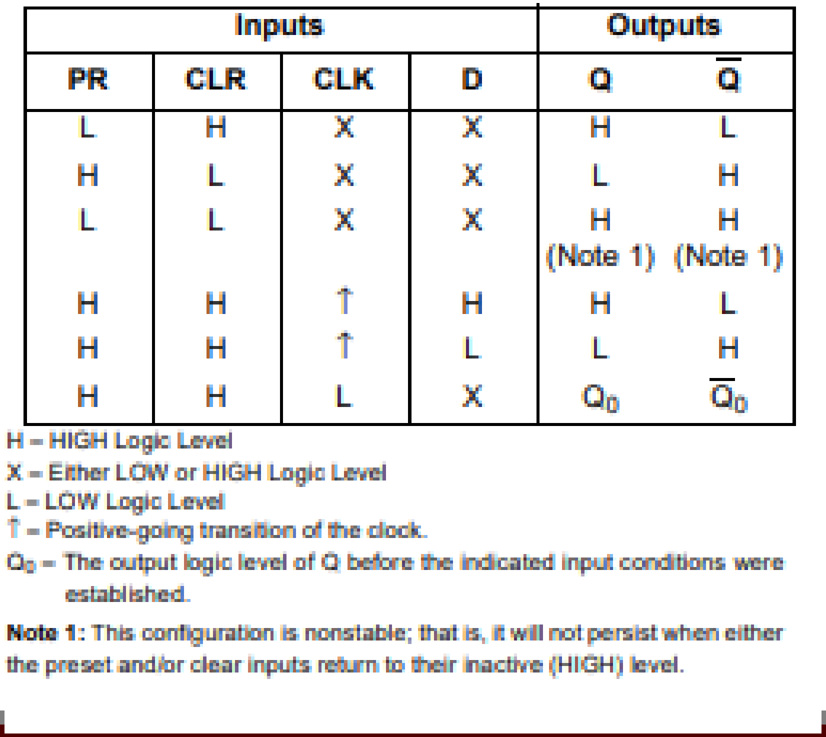




Preset & Clear

- Asynchronous inputs (not dependent on clocks)
- Inputs to help initialize to a given state, or clear to starting state
- Typically active low





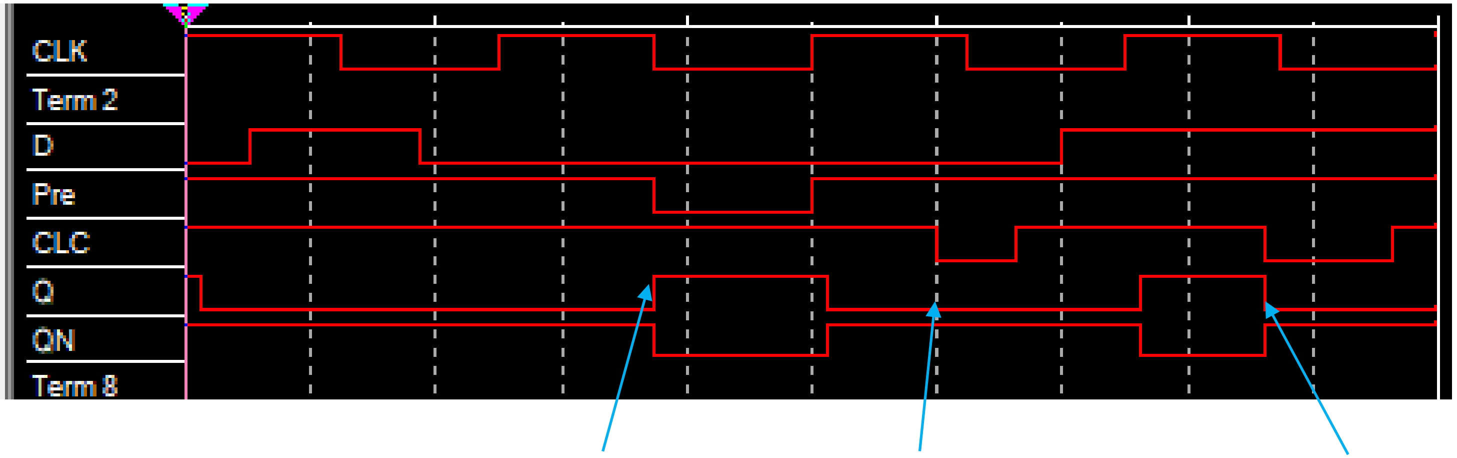
Inputs				Outputs	
PR	CLR	CLK	D	Q	$\overline{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H (Note 1)	H (Note 1)
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\overline{Q}_0$

H = HIGH Logic Level
X = Either LOW or HIGH Logic Level
L = LOW Logic Level
↑ = Positive-going transition of the clock.
Q_0 = The output logic level of Q before the indicated input conditions were established.

Note 1: This configuration is nonstable; that is, it will not persist when either the preset and/or clear inputs return to their inactive (HIGH) level.

Timing Diagram

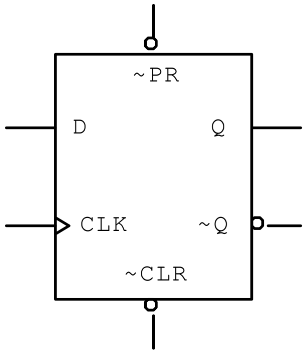
* Clear and preset will change output at anytime regardless of what D is



Preset sets Q

Clear resets Q

Clear resets Q



Flip Flop Uses

* Flip flops are basic storage elements in digital logic

* Building blocks for
  - Counters
  - Timers
  - Memory
  - Frequency dividers
  - Sequential logic circuits
  - Registers