TSN1101 Computer Architecture and Organization

Section A (Digital Logic Design)

Lecture A-06

Sequential Logic Circuits – Flip-flops

TOPIC COVERAGE IN THE LECTURE

- Sequential Logic Circuits
    - Synchronous Vs Asynchronous
- Latches Vs Flip-flops
- Types of Latches
    - S-R Latch / S'-R' Latch
    - Gated S-R Latch
    - Gated D Latch
- Types of Flip-flops
    - Clocked/Edge Triggered
        - S-R Flip-flop
        - J-K Flip-flop
        - D Flip-flop
        - T Flip-flop
- Flip-flops with Asynchronous Inputs
- Finite State Machines
    - Mealy Vs Moore Models

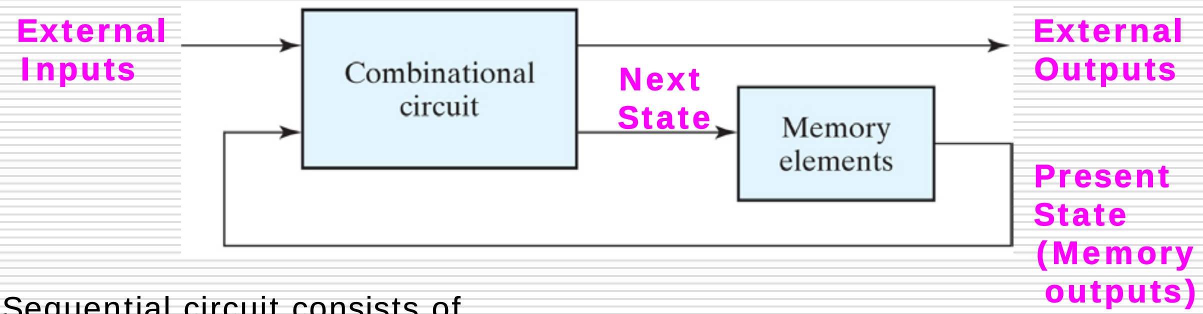
Sequential Logic Circuits

* Introduction
* Asynchronous Sequential Circuits
* Synchronous Sequential Circuits

Sequential Logic Circuits - Introduction

- A Sequential logic circuit consists of combinational logic gates and storage elements.
- Outputs at any time are determined from the external inputs and current state of storage elements.

Block Diagram of a Sequential Logic Circuit



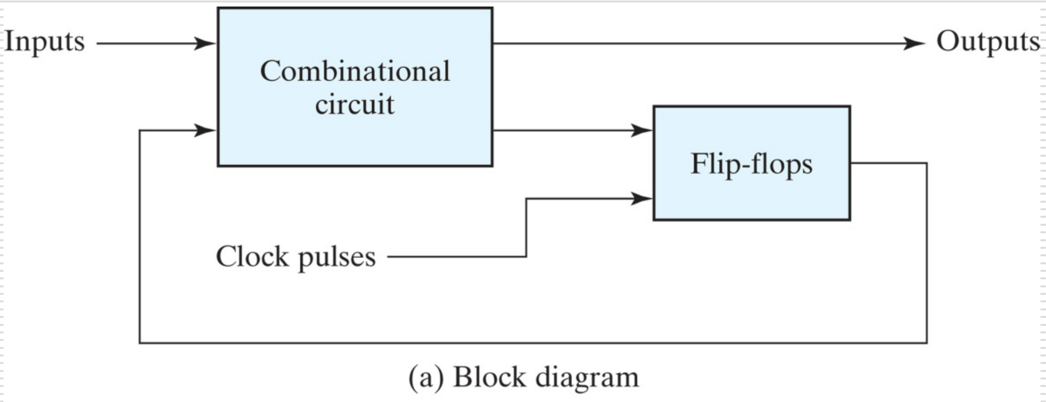
- A Sequential circuit consists of
  - Memory elements: Latches or Flip-Flops
  - Combinational Logic Circuit:
    - To implement multiple-output logic function
    - External Inputs are signals from the external environment
    - Present State are signals from storage elements
    - External outputs are signals to the outside environment
    - Next State are inputs to storage elements.

Sequential Logic Circuits - Classification

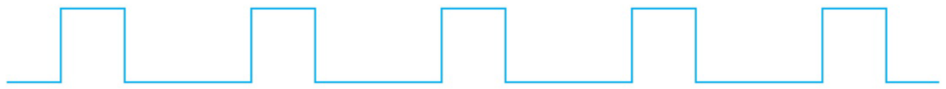
- Classification of sequential circuits depends on the times at which memory elements look into their inputs and change their state.
- Asynchronous Sequential Circuit
  - Behavior depends upon input signals at any instant of time and the order in which inputs change
  - Storage elements used are time-delay devices
  - Can be regarded as combinational circuit with feedback
  - May become unstable at times because of feedback among logic gates
    - can cause difficulties to the designer.
- Synchronous Sequential Circuit/Clocked Sequential Circuit
  - Behavior defined from knowledge of its signals that affect storage elements only at discrete instances of time
  - Synchronization achieved by clock generator which provides clock pulses.
  - Storage elements used are flip-flops
    - A flip-flop is a binary storage device capable of storing one bit of information.
  - No instability problems

Sequential Logic Circuits
- Synchronous sequential circuit

Block Diagram of a synchronous (clocked) Sequential Logic Circuit



(a) Block diagram



(b) Timing diagram of clock pulses

- The outputs can come from
  - Combinational circuit or
  - flip-flops or
  - both
- Flip-flops receive the inputs from
  - Combinational circuit
  - Clock signal with pulses that occur at fixed time intervals
- The state of flip-flops can change only during clock pulse transition

Latches

* Latches Vs Flip-flops

* Basic S-R Latch using

- NAND Gates
- NOR Gates

* Gated S-R Latch using

- NAND Gates
- AND and NOR Gates

* Gated D Latch using

- NAND Gates
- AND, NOR and inverter Gates

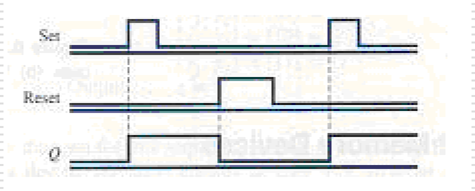
Latches Vs Flip-flop

Similarity:

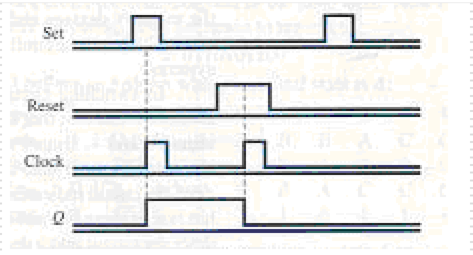
* Both are bistable multivibrators (having two stable states)

Difference:

* Latch does not have clocking source input, while flip-flop has.
  - Latch change the state immediately when the input changes

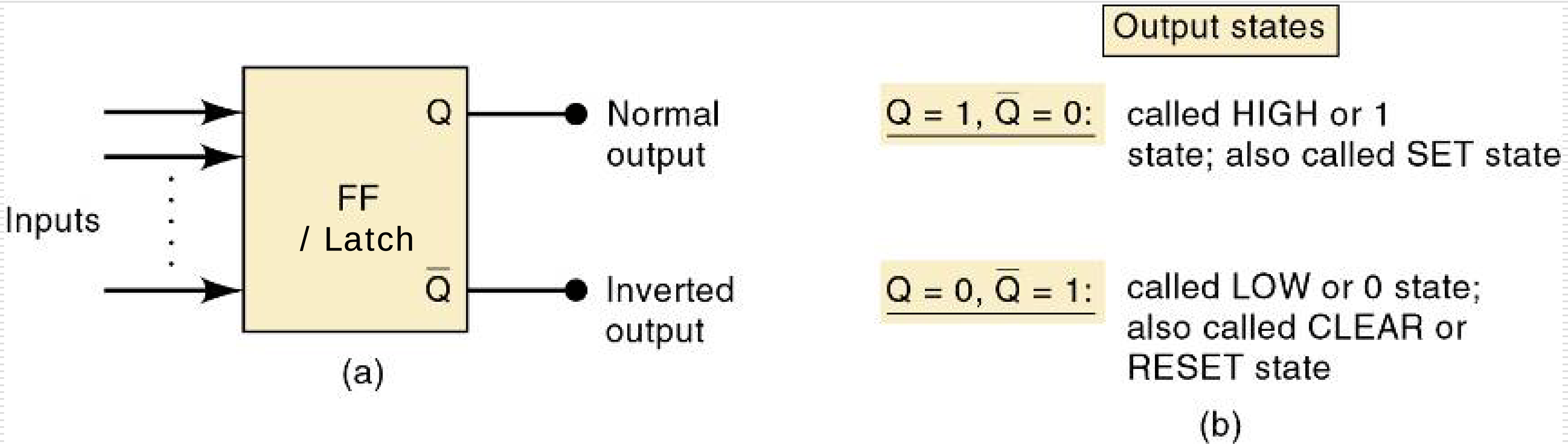


  - Flip-flop waits for the clock signal before changing state.



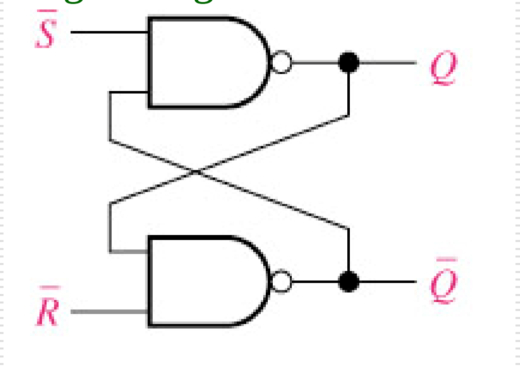
Latches

- A latch is a bistable memory device that can store one bit ('0' or '1') of data.
- It is the simplest possible memory element

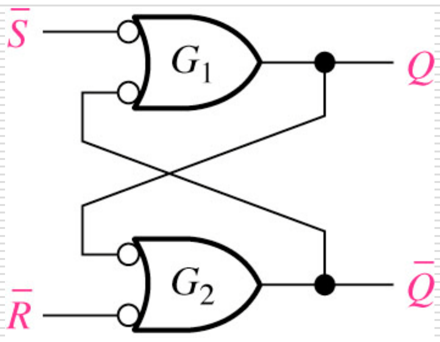

Output states
Inputs
FF / Latch
Q
$\overline{Q}$
Normal output
Inverted output
(a)
$Q = 1, \overline{Q} = 0$: called HIGH or 1 state; also called SET state
$Q = 0, \overline{Q} = 1$: called LOW or 0 state; also called CLEAR or RESET state
(b)

Basic $\overline{S}$ – $\overline{R}$ Latch
- Using NAND Gates

Logic Diagram for active-LOW input S'-R' NAND Latch



≈

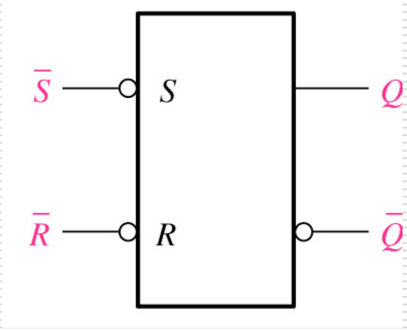


➢Active-LOW input S'-R' latch is formed with two cross-coupled NAND gates.
➢Output of each gate is connected to an input to the opposite gate, producing a feedback.

Truth table for active-LOW input S'-R' Latch

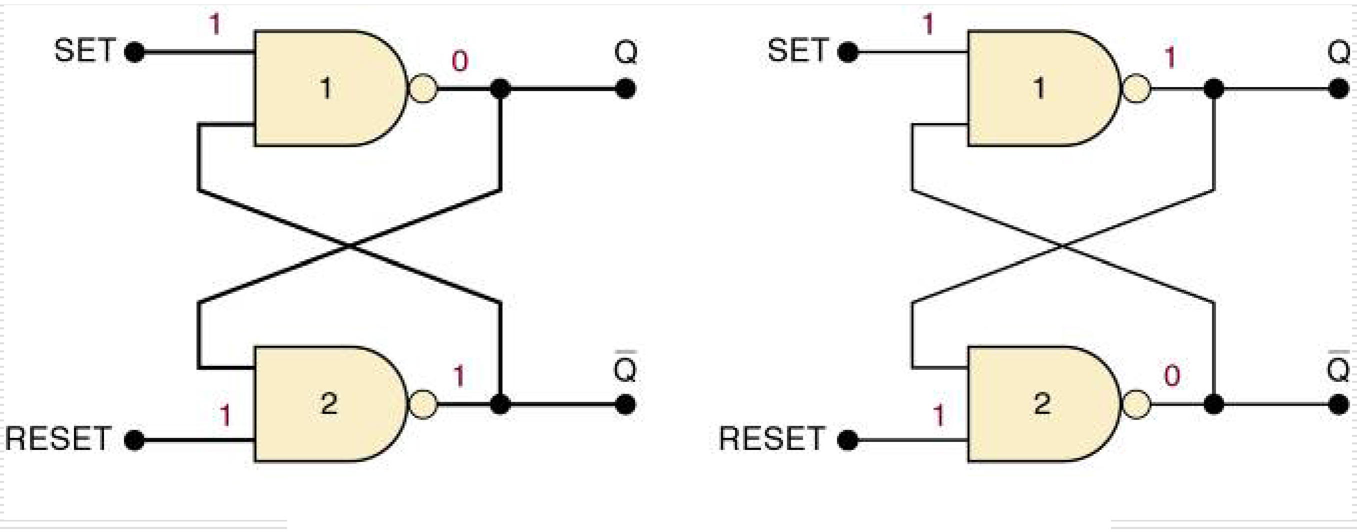
INPUTS		OUTPUTS		Output State / COMMENT
S'	**R'**	**Q**	**Q'**	
1	1	NC	NC	No change (HOLD) Latch remains at previous state.
0	1	1	0	Latch SET
1	0	0	1	Latch RESET
0	0	1	1	Invalid Condition/Forbidden

Logic Symbol



Basic $\overline{S}$ – $\overline{R}$ Latch
- Using NAND Gates (Operation)

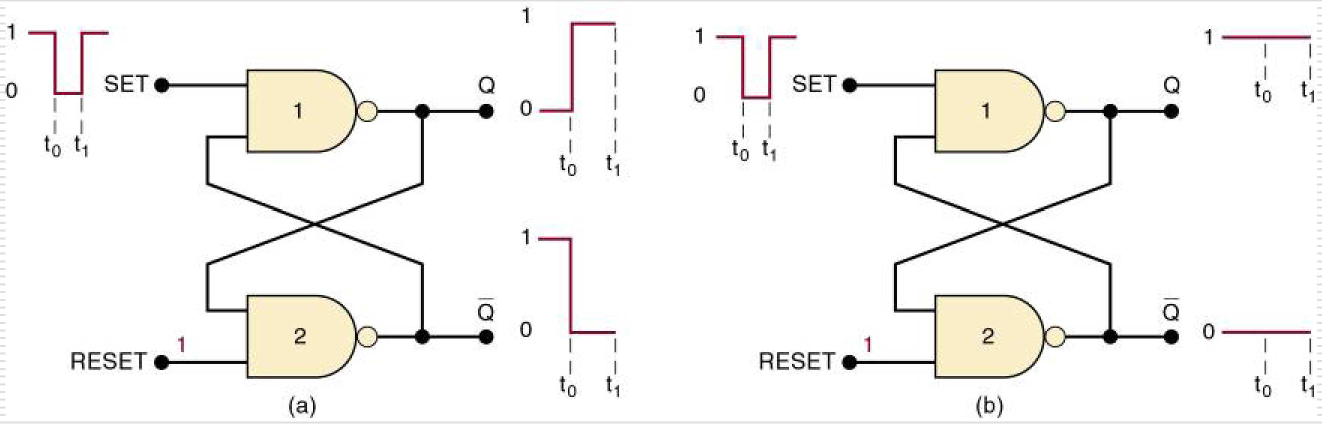
Active-LOW input S'-R' NAND Latch with two possible resting states when **SET=CLEAR=1**



Basic $\overline{S}$ – $\overline{R}$ Latch

- Using NAND Gates (Operation)

Active-LOW input S'-R' NAND Latch when **SET=0 and CLEAR=1**

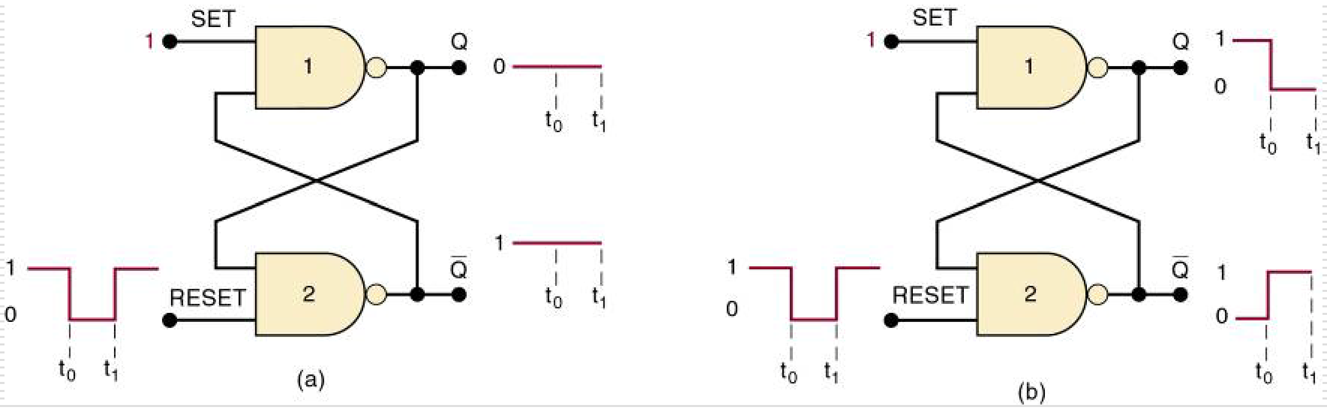


Pulsing the SET input to the 0 state (a) *Q* = 0 prior to SET pulse. (b) *Q* = 1 prior to SET pulse.

In both cases, *Q* ends up HIGH.

Basic $\overline{S}$ – $\overline{R}$ Latch
- Using NAND Gates (Operation)

Active-LOW input S'-R' NAND Latch when **SET=1 and CLEAR=0**



Pulsing the RESET input to the 0 state (a) *Q* = 0 prior to RESET pulse. (b) *Q* = 1 prior to RESET pulse.

In both cases, *Q* ends up LOW.

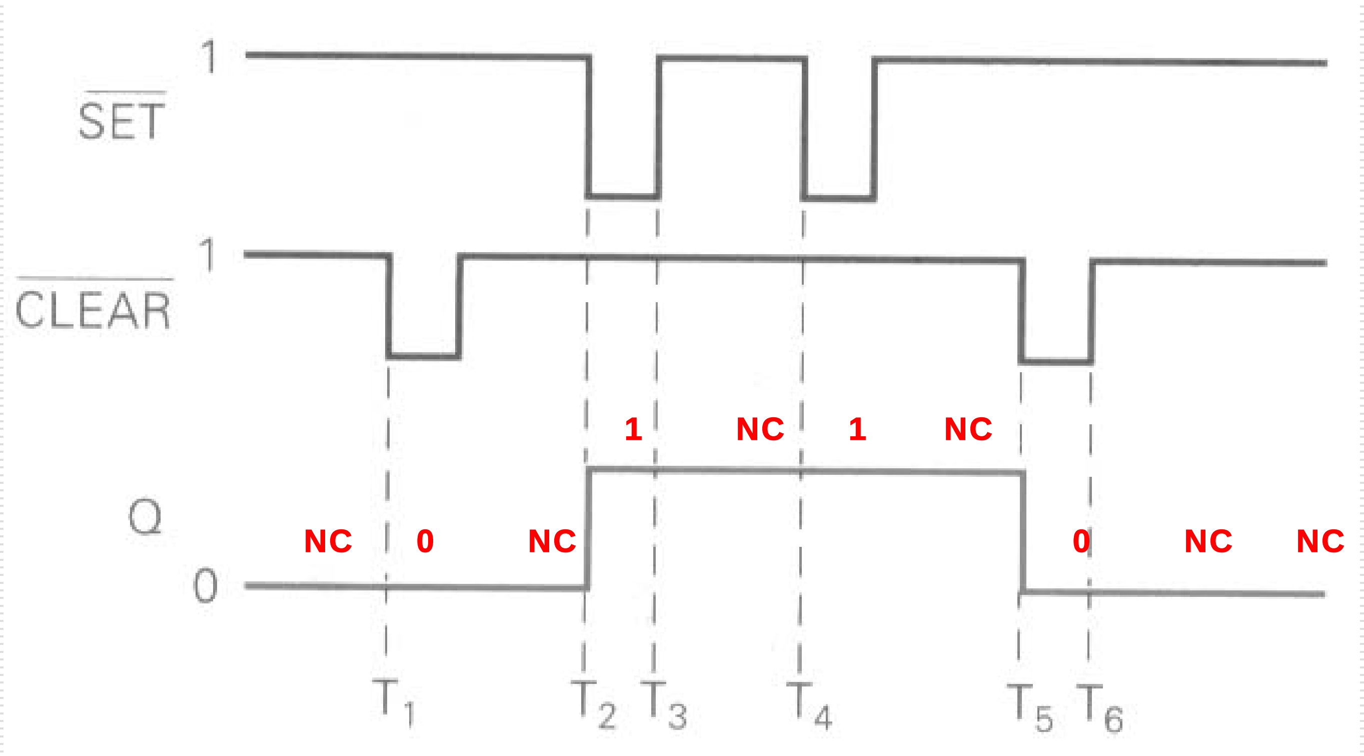
Basic $\overline{S} - \overline{R}$ Latch

- Using NAND Gates (Summary)

- **SET = 1, RESET = 1**—Normal **resting state**, outputs remain in state they were in prior to input.
- **SET = 0, RESET = 1**—Output will go to *Q* = 1 and remains there, even after SET returns HIGH.
  - Called ***setting*** the latch.
- **SET = 1, RESET = 0**—Will produce *Q* = 0 LOW and remains there, even after RESET returns HIGH.
  - Called ***clearing or resetting*** the latch.
- **SET = 0, RESET = 0**—Tries to set and clear the latch at the same time, and produces output that is unpredictable, and this input condition should not be used - **Forbidden**

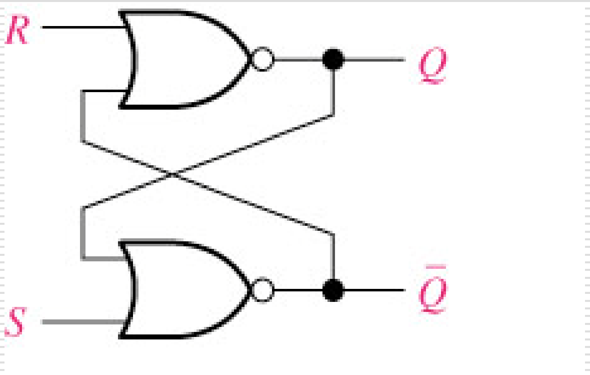
Basic $\overline{S}$ – $\overline{R}$ Latch
- Using NAND Gates (Example)

Assuming Q = 0 initially, and determine the Q waveform for the NAND latch.



Basic S–R Latch
- Using NOR Gates

Logic Diagram for active-HIGH input S-R NOR Latch

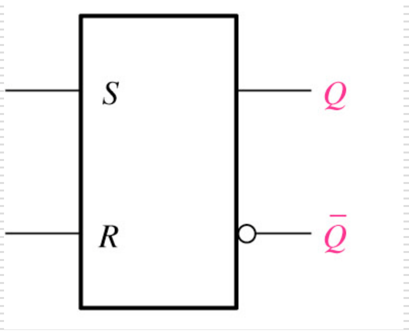


- Active-HIGH input S-R latch is formed with two cross-coupled NOR gates.
- Output of each gate is connected to an input to the opposite gate, producing a feedback.

Truth table for active-HIGH input S-R Latch

INPUTS		OUTPUTS		COMMENT
S	***R***	***Q***	**Q'**	
0	**0**	**NC**	**NC**	**No change (Hold)** **Latch remains at previous state.**
1	**0**	**1**	**0**	**Latch SET**
0	**1**	**0**	**1**	**Latch RESET**
1	**1**	**0**	**0**	**Invalid Condition/ Forbidden**

Logic Symbol

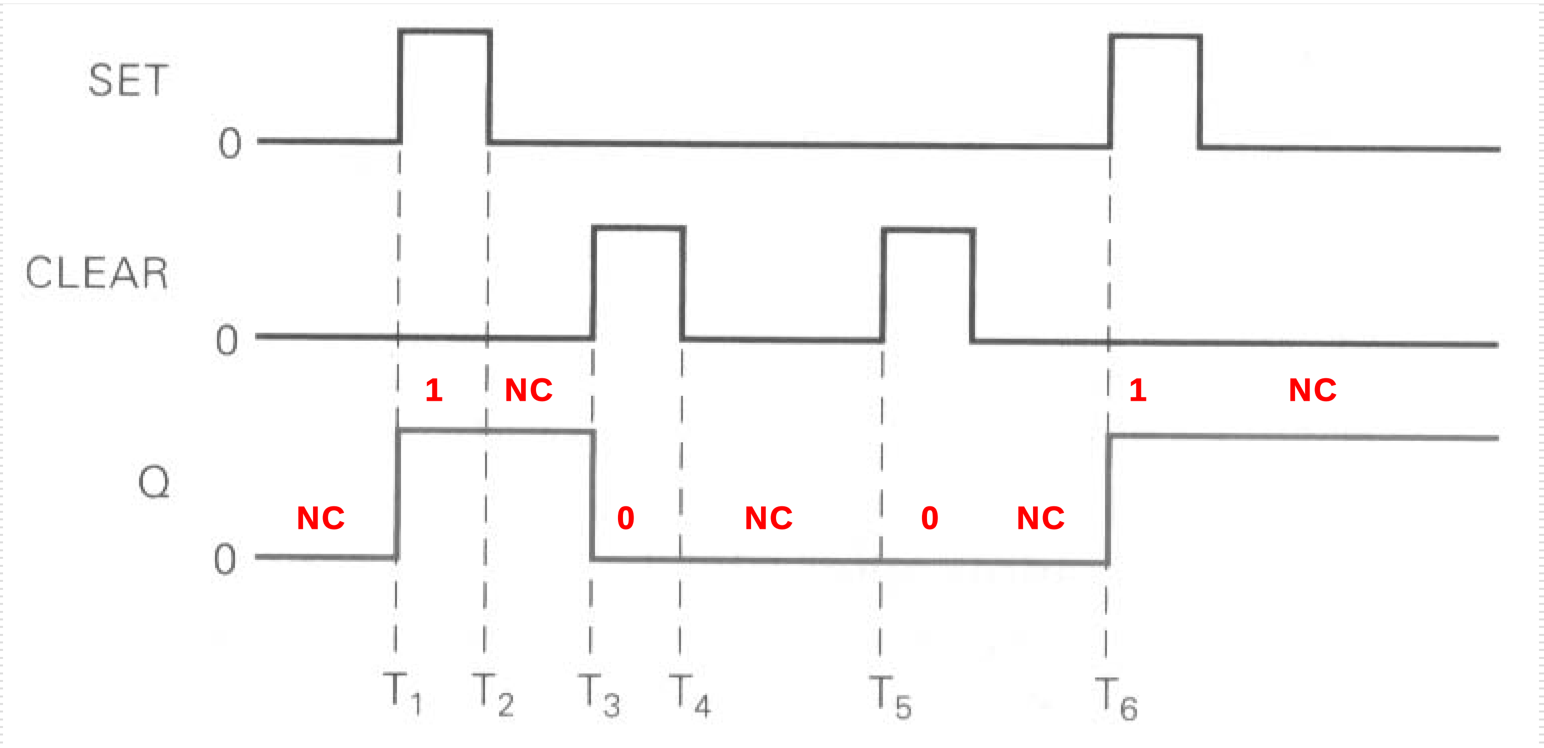


Basic S–R Latch
- Using NOR Gates (Analysis - Summary)

- The analysis of a S-R latch is described as follows:
  - S = 0 and R = 0: Q will remain in prior state (HOLD mode) . It has no effect to the output. It works in HOLD (no change) mode of operation.
  - S = 1 and R = 0: it works in SET mode of operation, Q will set to 1.
  - S = 0 and R = 1: it works in RESET mode of operation. Q will reset to 0.
  - S = 1 and R = 1: this is an unexpected condition, because the two outputs try to set and reset at the same time. It is forbidden mode.

Basic S–R Latch
- Using NOR Gates (Example)

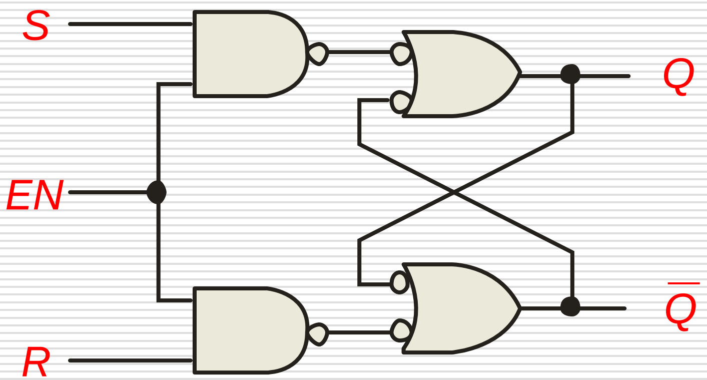
Assuming that Q = 0 initially, determine the Q waveform for the NOR latch.



Gated S–R Latch
- Using NAND Gates

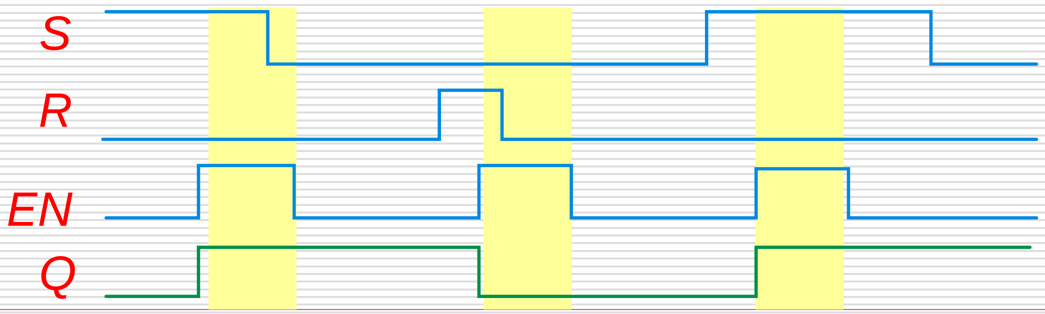
- ➢A gated latch is a variation on the basic latch.
  - ➢The gated latch has an additional input, called enable (*EN*) that must be HIGH in order for the latch to respond to the *S* and *R* inputs.

Logic Diagram for Gated S-R Latch



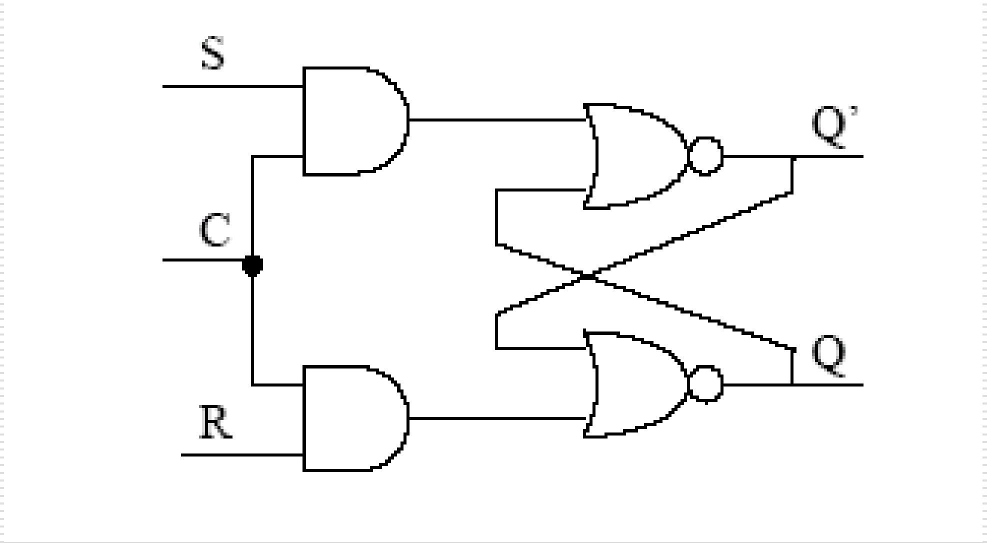
Example: Show the *Q* output with relation to the input signals. Assume *Q* starts LOW.

Solution:



Gated S–R Latch
- Using NOR and AND Gates

Logic Diagram for Gated S-R Latch (with NOR and AND gates)



- When the control input signal (C) is 0, the outputs of the two AND gates are forced to 0, signals applied to the S and R inputs will not affect the cross-coupled NOR latch circuit.
    - Hence the circuit behaves as though S and R were both 0, latching the Q and Q' outputs in their last states.

- When the control input signal (C) is activated (1), signals applied to the S and R inputs can affect the cross-coupled NOR latch circuit.
    - The gated S-R latch will act like a normal S-R latch.

Gated S–R Latch

- Using NOR and AND Gates (Truth Table)

C	S	R	Q (t+1)	Q' (t+1)	Status
0	0	0	Q (t)	Q'(t)	No change
0	1	0	Q (t)	Q'(t)	No change
0	0	1	Q (t)	Q'(t)	No change
0	1	1	Q (t)	Q'(t)	No change
1	0	0	Q (t)	Q'(t)	No change
1	1	0	1	0	SET
1	0	1	0	1	RESET
1	1	1	0	0	Prohibited

No change

Gated SR latch

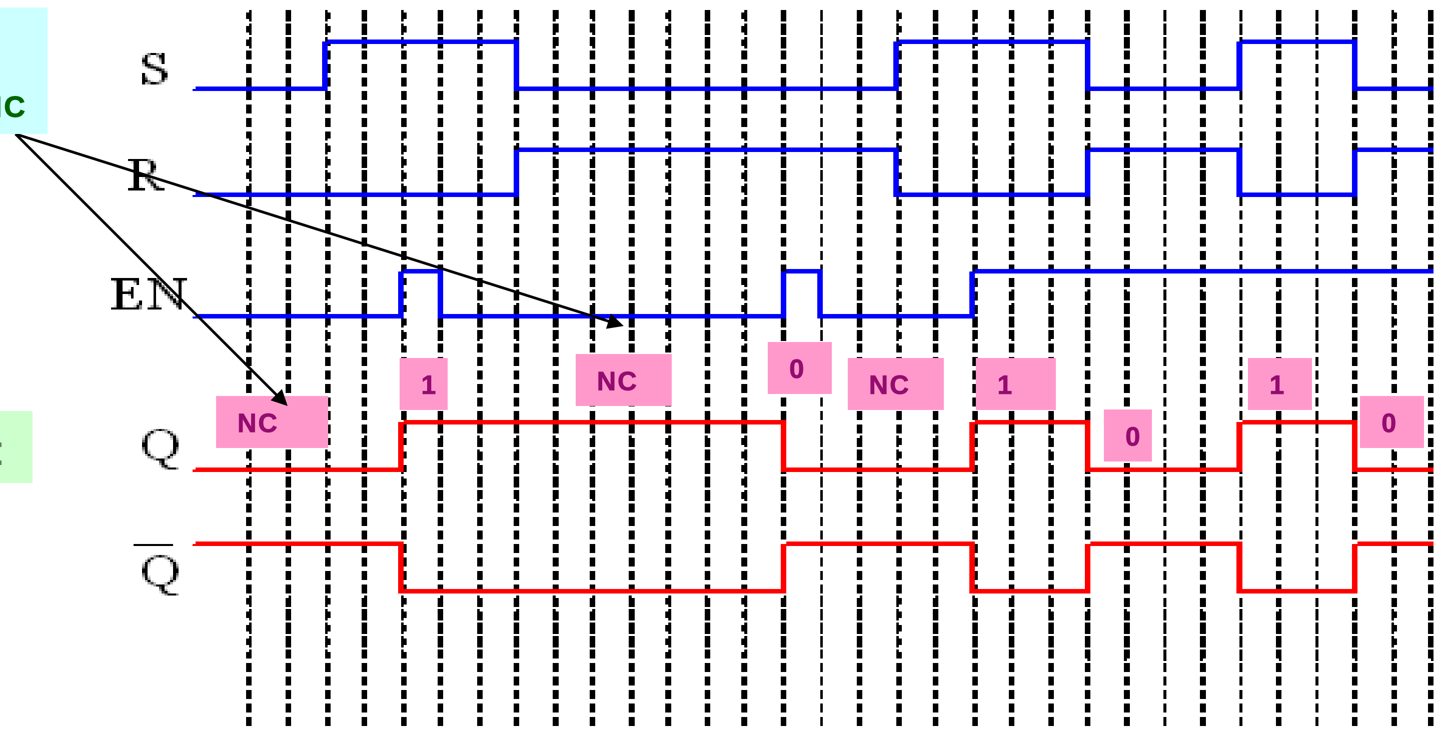
- Active HIGH (NOR)

Gated S–R Latch
- Using NOR and AND Gates (Example)

Determine the Q and $\overline{Q}$ output waveform if the inputs shown below are applied to an active-HIGH gated S-R latch that is initially RESET.

(EN=C) is 0, therefore outputs are NC

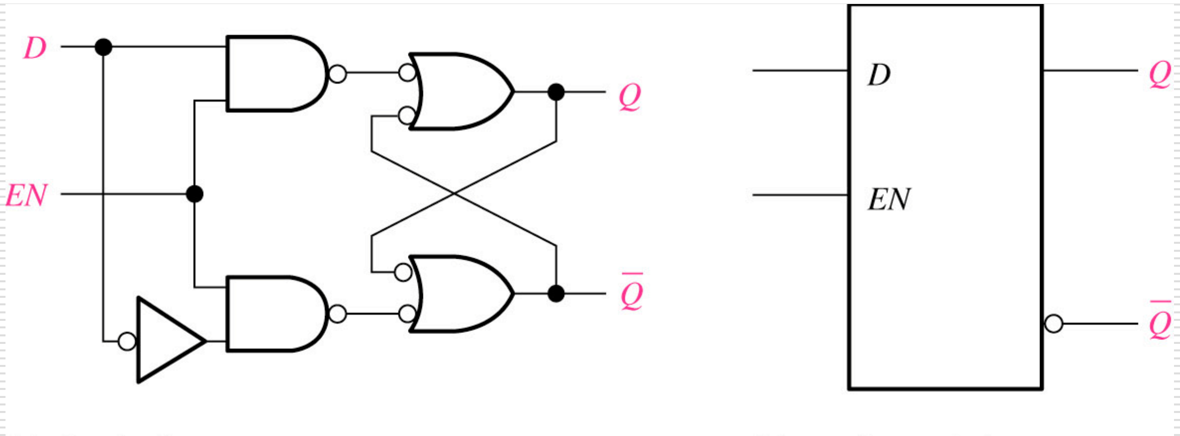
Solution:



Gated D Latch
- Using NAND Gates

The difference between gated D latch and gated S-R latch is that it has only one input.

• The D latch is constructed from a gated S-R latch, with an inverter added to make R input as the complement (inverse) of S input.



(a) Logic diagram (b) Logic symbol

Logical Operation:

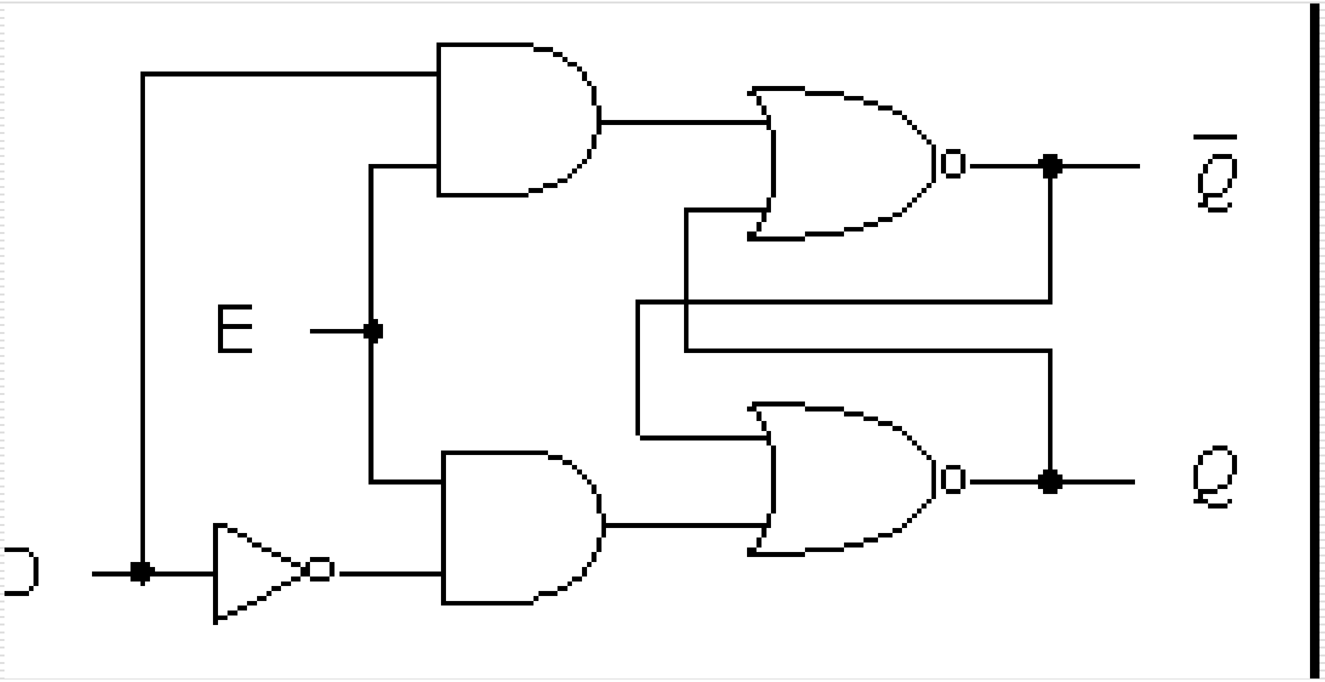
When D input is HIGH and the EN input is HIGH, the latch will SET.

When D input is LOW and the EN input is HIGH, the latch will RESET

In short, Q output follows the D input when EN is HIGH

Gated D Latch

- Using NOR, AND Gates and inverter



Gated D Latch
- Truth Table

Truth table for Gated D Latch

INPUTS		OUTPUTS		COMMENTS
EN	***D***	***Q***	**Q'**	
0	**0**	**NC**	**NC**	**No Change**
0	**1**	**NC**	**NC**	**No Change**
1	**0**	**0**	**1**	**Latch RESET**
1	**1**	**1**	**0**	**Latch SET**

- As with the gated S-R latch, the D latch will not respond to a signal input if the enable input (EN) is LOW (0). it simply stays latched in its last state (NC)
- However when the enable input (EN) is HIGH (1), then the Q output follows the D input as long as the control input remains logic 1.

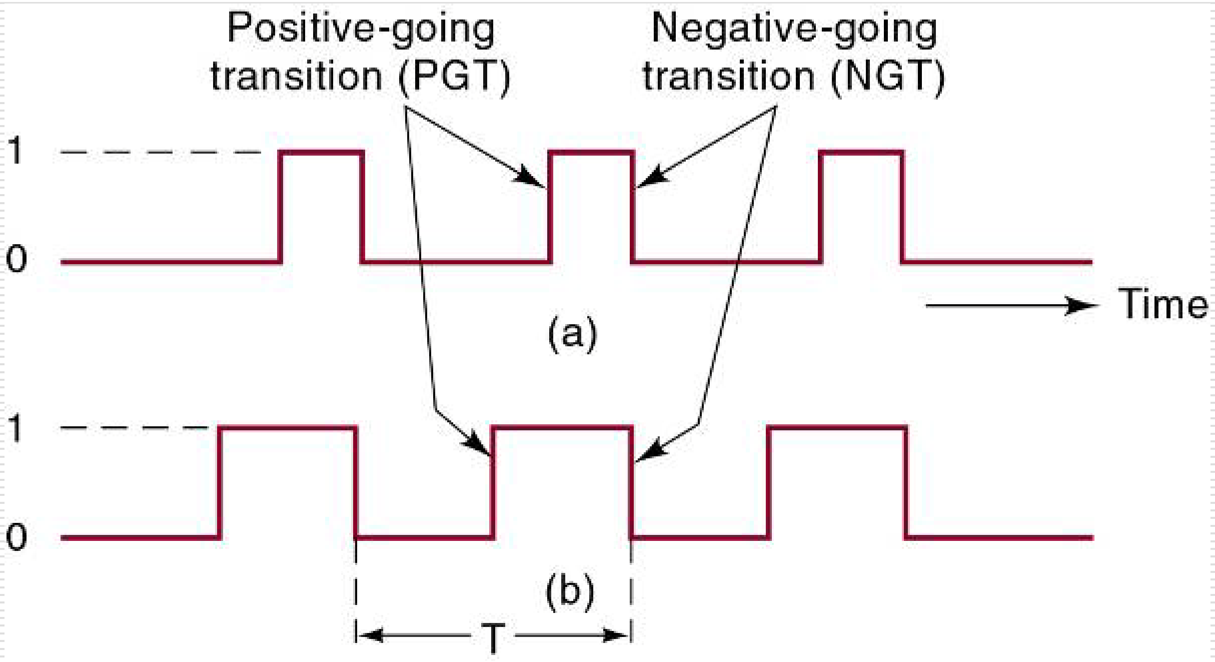
Flip-flops

* Clock Signals and Clocked Flip-flops
  - PGT and NGT
* Clocked S-R Flip-flop
* Clocked J-K Flip-flop
* Clocked D Flip-flop
* Clocked T Flip-flop
* Flip-flops – Summary
  - Characteristic Tables, Characteristic equations and Excitation Tables
* Asynchronous Inputs
* Flip-flop Applications- Examples

Clock Signals and Clocked Flip-Flops - Introduction

- The clock signal is a rectangular pulse train or square wave.
  - Positive going transition (PGT)—clock pulse goes from 0 to 1.
  - Negative going transition (NGT)—clock pulse goes from 1 to 0.

Transitions are also called *edges*.

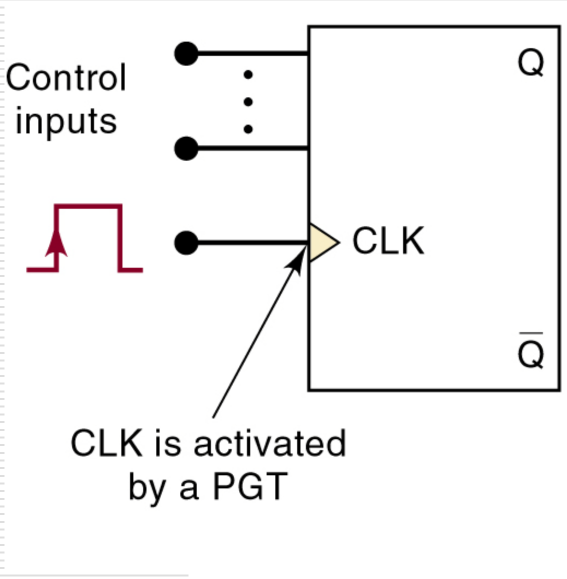

Positive-going transition (PGT)
Negative-going transition (NGT)
1
0
Time
(a)
1
0
(b)
T

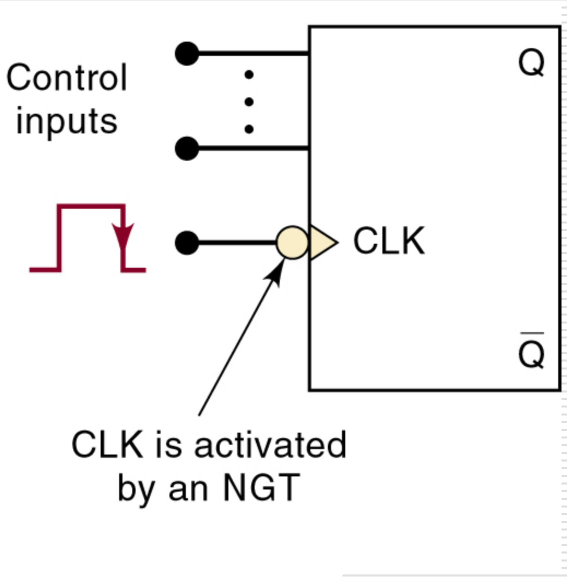
Clock Signals and Clocked Flip-Flops - Introduction

- Clocked FFs change state on one or the other clock transitions.
  - Clock inputs can be labeled as CLK, CK, or CP.

A small triangle at the CLK input indicates that the input is activated with a PGT.

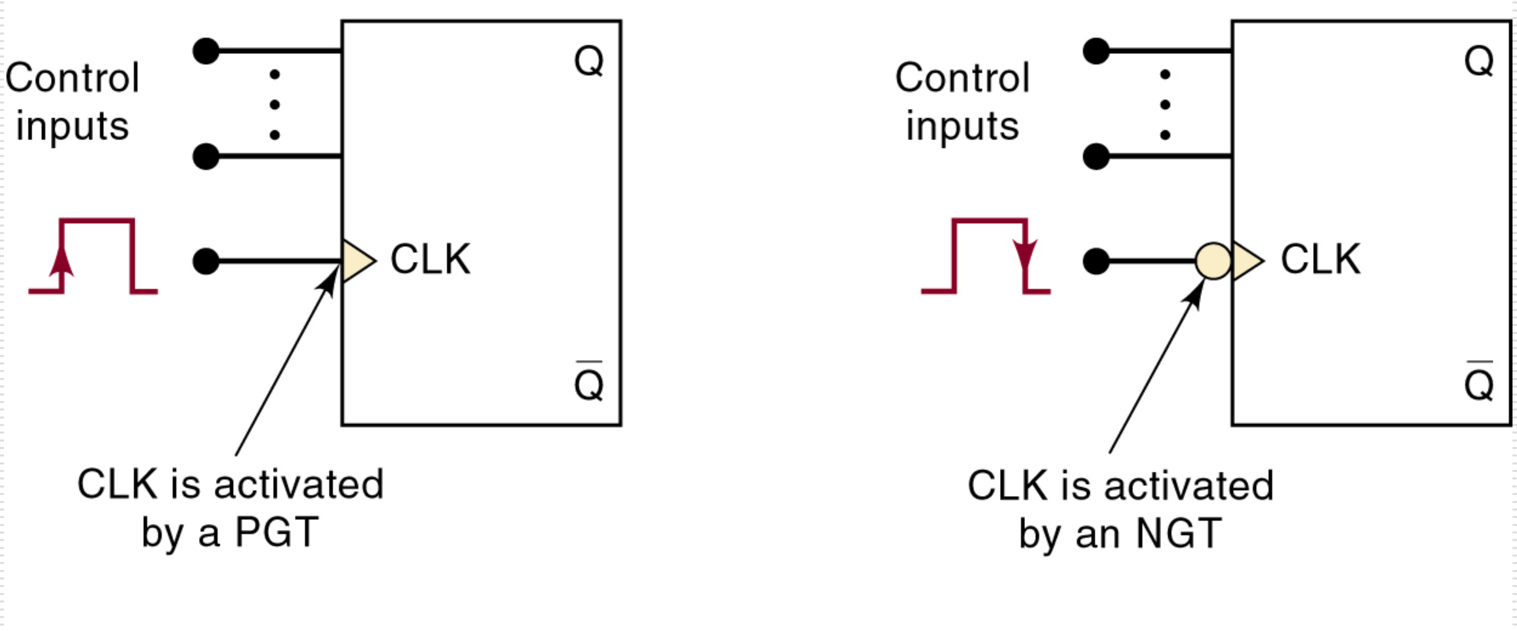
A bubble and a triangle indicates that the CLK input is activated with a NGT.





Clock Signals and Clocked Flip-Flops - Introduction

- Control inputs have an effect on the output only at the active clock transition (NGT or PGT)—also called synchronous control inputs.
  - The control inputs get the outputs ready to change, but the change is not triggered until the CLK edge.

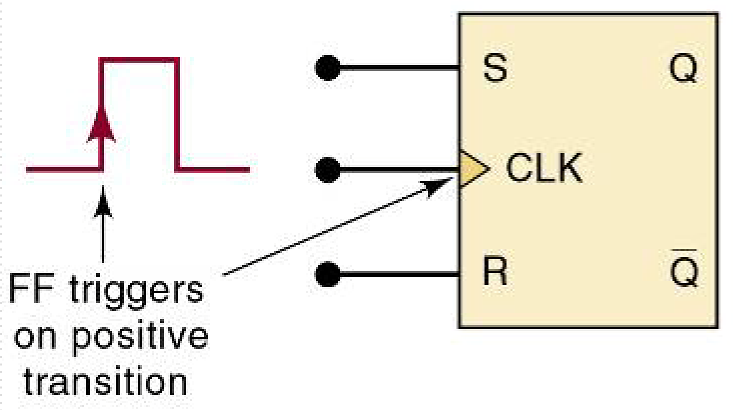


Clocked S-R Flip-Flop

- The *S* and *R* inputs are synchronous *control* inputs, which control the state the FF will go to when the clock pulse occurs.
  - The *CLK* input is the **trigger** input that causes the FF to change states according to the *S* and *R* inputs.
- SET-RESET (or SET-CLEAR) FF will change states at positive- or negative-going clock edges.

Clocked S-R Flip-Flop - Positive Edge Triggering

A clocked S-R flip-flop triggered by the positive-going edge of the clock signal



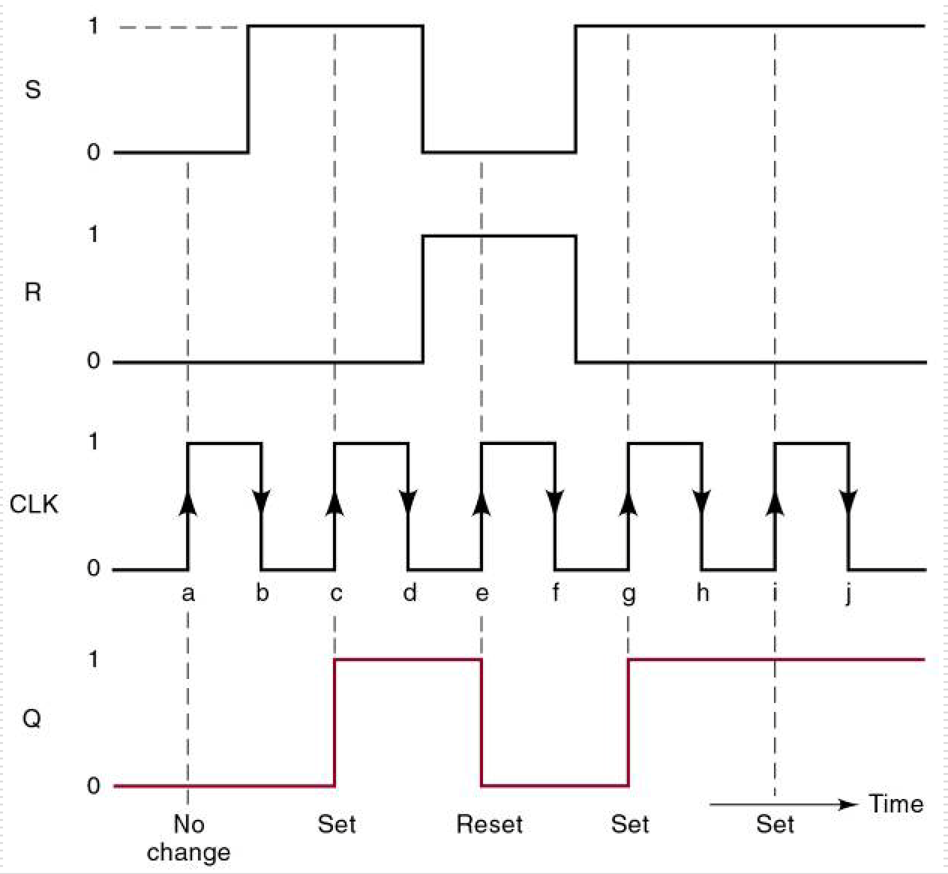
Inputs			Output
S	R	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	Ambiguous

Q_0 is output level prior to ↑ of CLK.
↓ of CLK produces no change in Q.

The *S* and *R* inputs control the state of the FF in the same manner as described earlier for the NOR gate latch, but the FF does *not* respond to these inputs *until* the occurrence of the PGT of the clock signal.

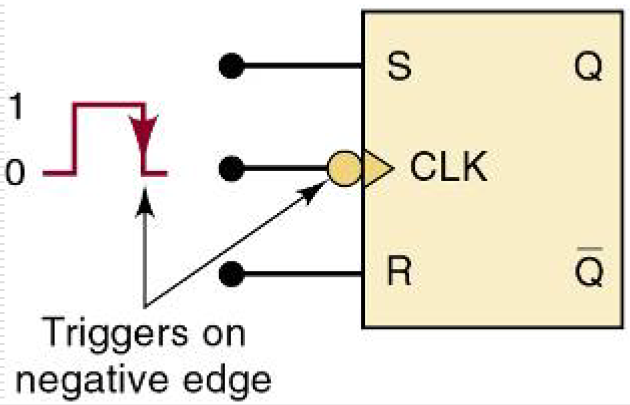
Clocked S-R Flip-Flop
- Positive Edge Triggering (Timing Diagram)

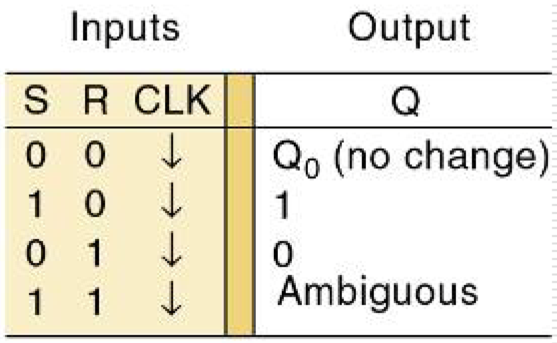
Waveforms of the operation of a clocked S-R flip-flop triggered by the positive-going edge of a clock pulse.



Clocked S-R Flip-Flop
- Negative Edge Triggering

A clocked S-R flip-flop triggered by the negative-going edge of the clock signal.

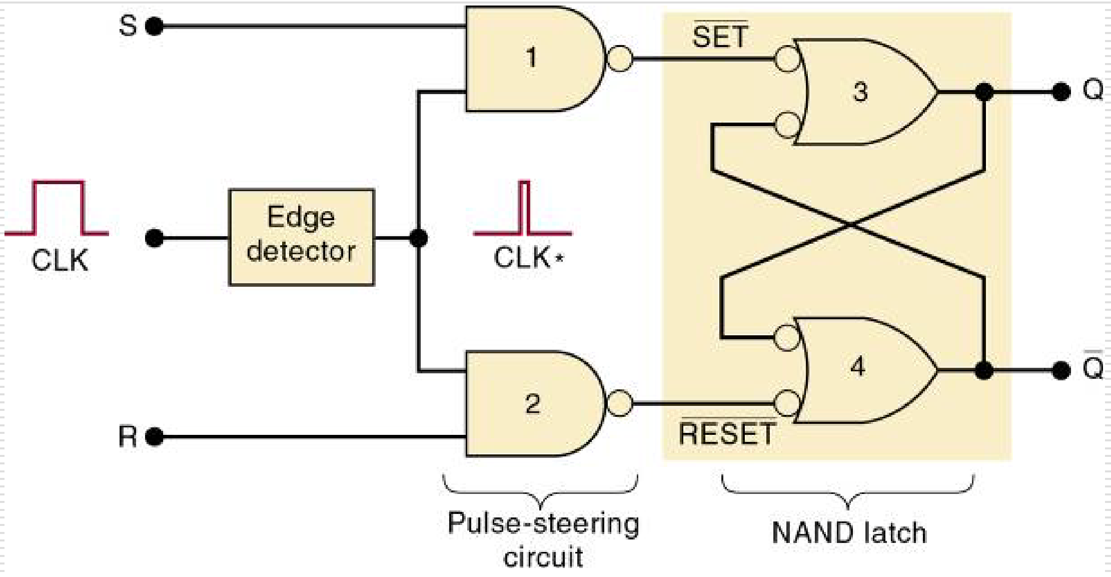




Inputs			Output
S	R	CLK	Q
0	0	↓	Q_0 (no change)
1	0	↓	1
0	1	↓	0
1	1	↓	Ambiguous

Both positive-edge and negative-edge triggering FFs are used in digital systems.

Clocked S-R Flip-Flop
- Logic Diagram

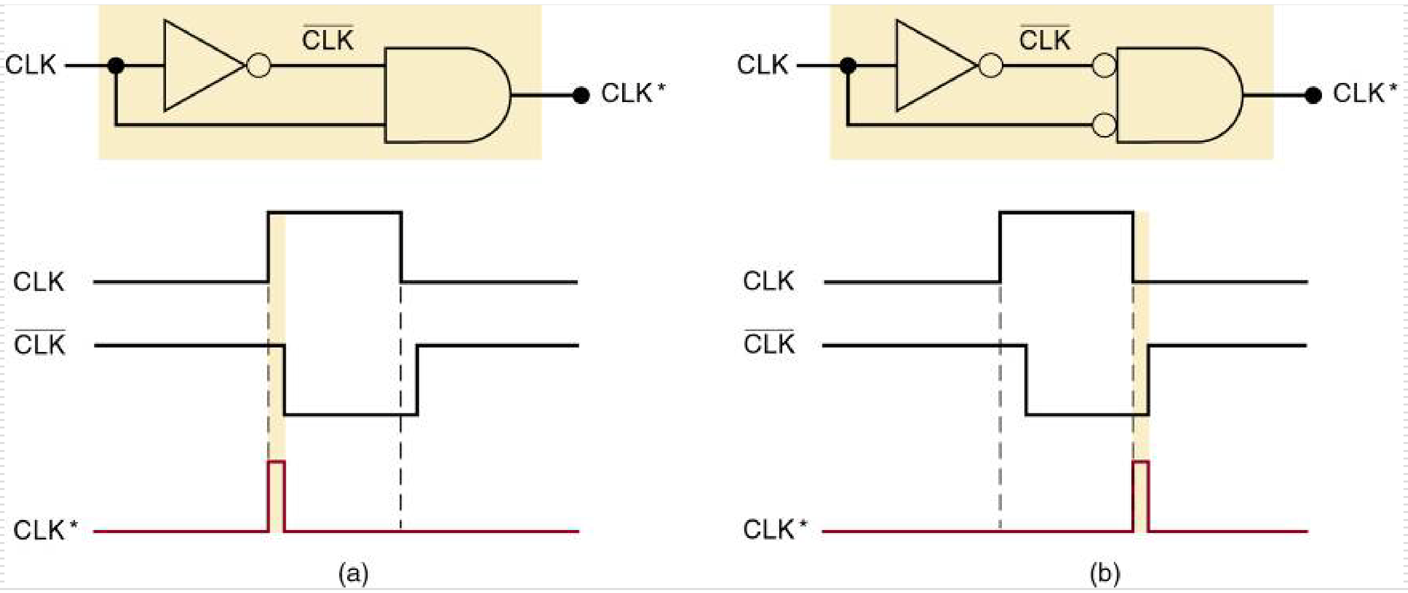


- An edge-triggered S-R flip-flop circuit features:
  - A basic **NAND** gate latch formed by **NAND**-3 and **NAND**-4.
  - A **pulse-steering circuit** formed by **NAND**-1 and **NAND**-2.
  - An **edge-detector circuit.**

Clocked S-R Flip-Flop

- Logic Diagram (Edge Detector Circuit)

- Implementation of edge-detector circuits used in edge-triggered flip-flops: (a) PGT; (b) NGT.



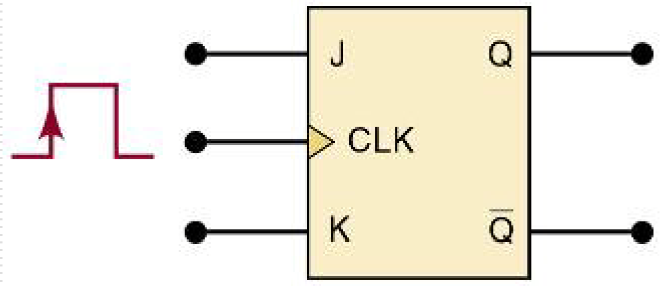
The duration of the *CLK** pulses is typically 2–5 ns.

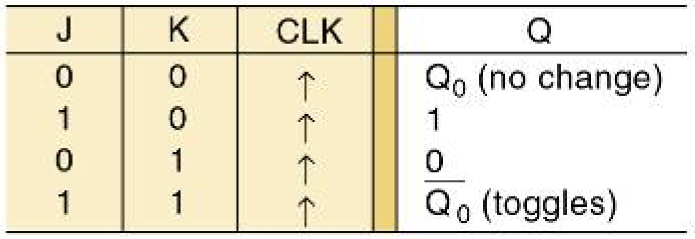
Clocked J-K (Jack-Kilby) Flip-Flop

- Operates like the S-R FF.
  - J is SET, K is CLEAR.
- When J and K are both HIGH, output is toggled to the opposite state.
  - Can use either positive going or negative going clock trigger.
- Much more versatile than the S-R flip-flop, as it has no ambiguous states.
  - Has the ability to do everything the S-R FF does, plus operates in toggle mode.

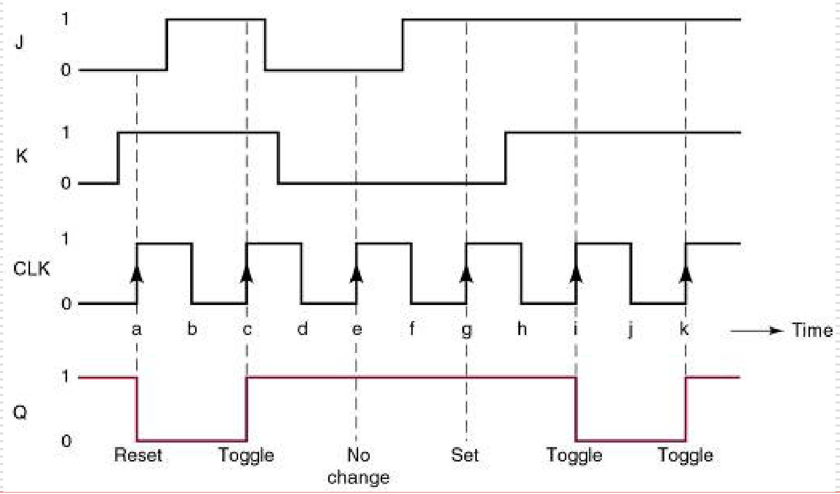
Clocked J-K Flip-Flop
- Positive Edge Triggering

Clocked J-K flip-flop that responds only to the positive edge of the clock.



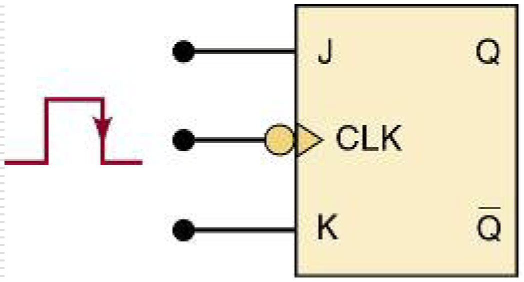


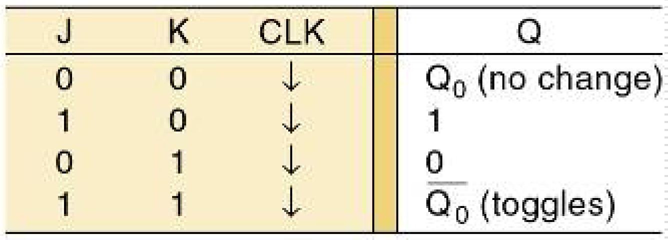
J	K	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q}_0$ (toggles)



Clocked J-K Flip-Flop
- Negative Edge Triggering

Clocked J-K flip-flop that responds only to the negative edge of the clock.

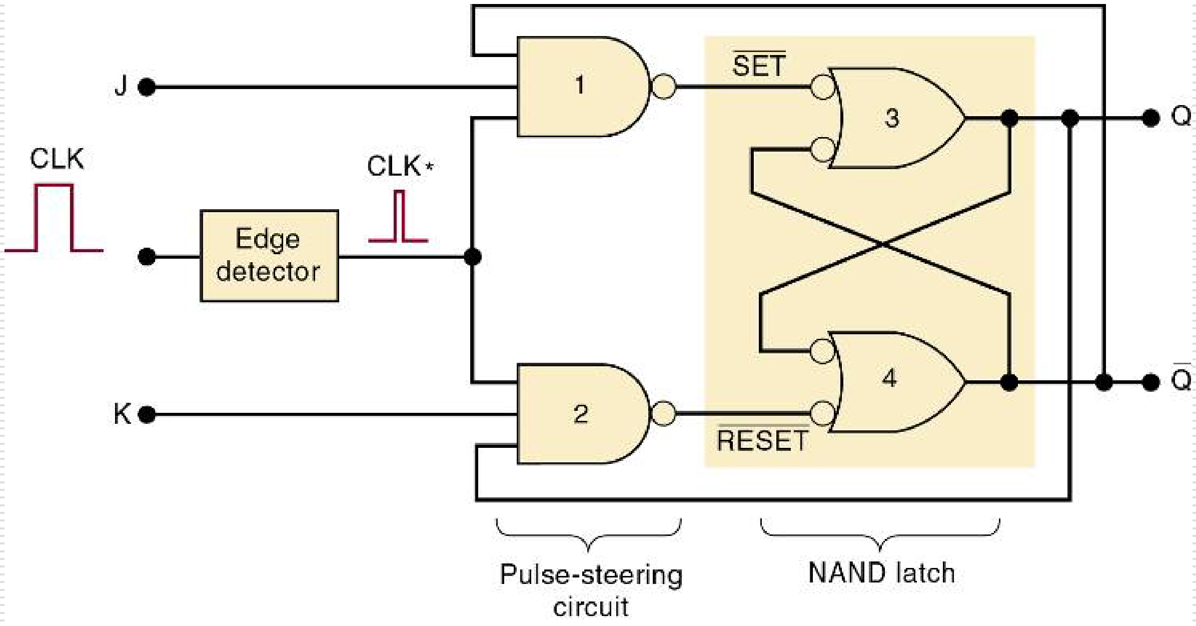




J	K	CLK	Q
0	0	↓	Q_0 (no change)
1	0	↓	1
0	1	↓	0
1	1	↓	$\overline{Q_0}$ (toggles)

Clocked J-K Flip-Flop
- Logic Diagram

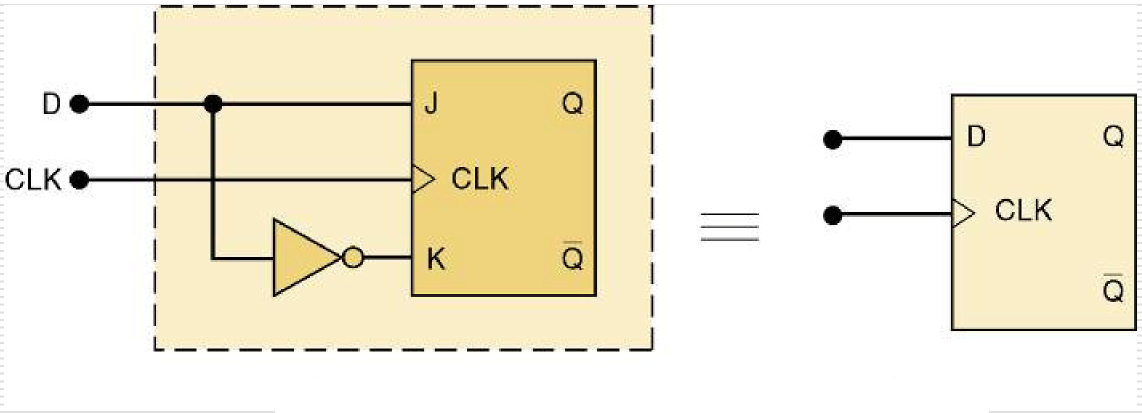
- The internal circuitry of an edge-triggered J-K flip-flop contains the same three sections as the edge-triggered S-R flip-flop.



Clocked D (Data) Flip-Flop

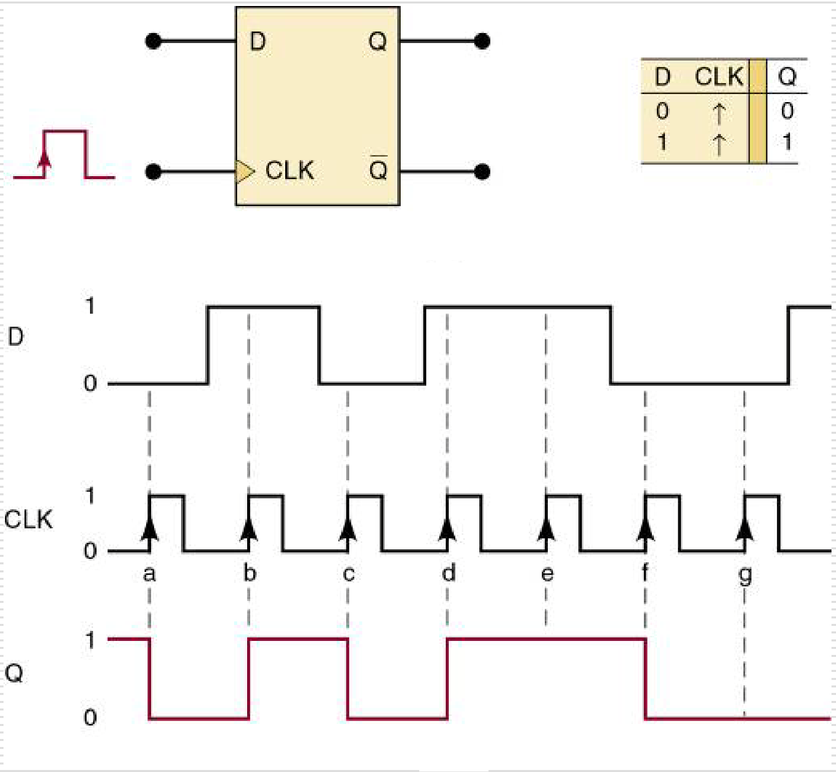
- One data input (D)
- output (Q) changes to the value of the input at either the positive- or negative-going clock trigger.
- can be implemented with a J-K FF by tying the J input to the K input through an inverter.
  - The same can be done to convert the S-R flip-flop to a D flip-flop
- Useful for parallel data transfer.

Positive Edge-triggered D flip-flop implementation from a J-K flip-flop.



Clocked D Flip-Flop
- Positive Edge Triggering

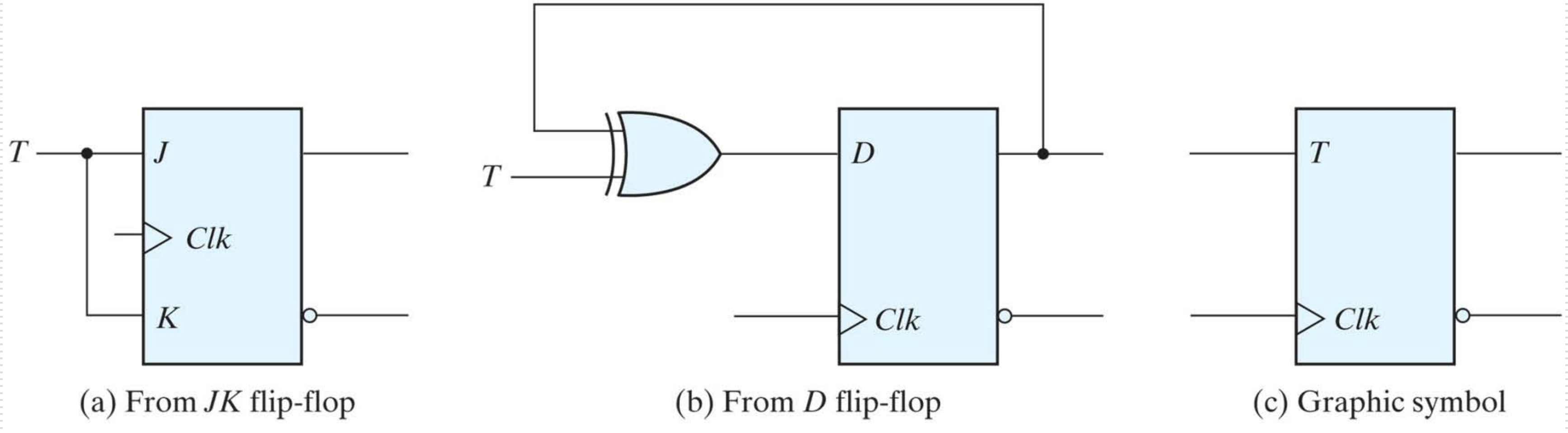
D flip-flop that triggers only on positive-going transitions.



D	CLK	Q
0	↑	0
1	↑	1

Clocked T (Toggle) Flip-Flop - Positive Edge Triggering

An edge-triggered T flip-flop is easily implemented by connecting J and K inputs together in an edge-triggered J-K flip-flop.



(a) From *JK* flip-flop (b) From *D* flip-flop (c) Graphic symbol

When T=0 (J=K=0), a clock edge does not change the output.
When T=1 (J=K=1), a clock edge complements the output.

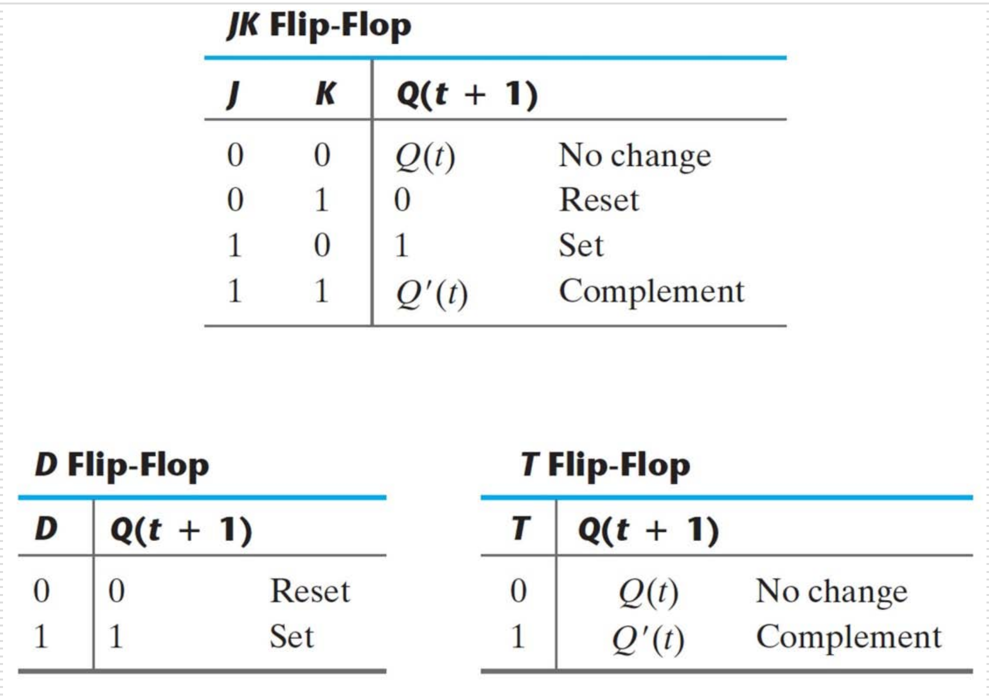
Flip-Flops - Summary

- Characteristic Tables/Equations

Characteristic Tables

SR Flip-flop

S	R	Q (t+1)	
0	0	Q (t)	No change
0	1	0	RESET
1	0	1	SET
1	1	?	Undefined



JK Flip-Flop

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

D Flip-Flop

D	$Q(t+1)$	
0	0	Reset
1	1	Set

T Flip-Flop

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

Characteristic Equations

SR Flip-flop:
$Q(t+1) = S + R'Q(t)$
JK Flip-flop:
$Q(t+1)=JQ'(t) + K'Q(t)$
D Flip-flop:
$Q(t+1)=D$
T Flip-flop:
$Q(t+1) = T \oplus Q(t)$

Flip-Flops - Summary

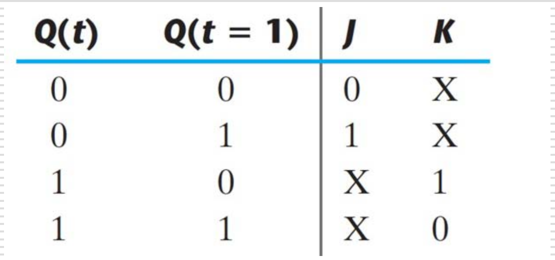
- Excitation Tables

Excitation Tables

SR Flip-flop

Q(t)	Q (t+1)	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

J-K Flip-flop

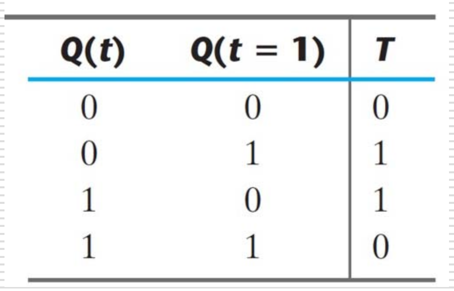


$Q(t)$	$Q(t = 1)$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

D Flip-flop

Q(t)	Q (t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

T Flip-flop



$Q(t)$	$Q(t = 1)$	T
0	0	0
0	1	1
1	0	1
1	1	0

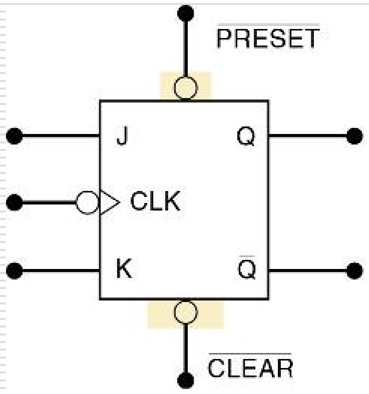
Asynchronous Inputs

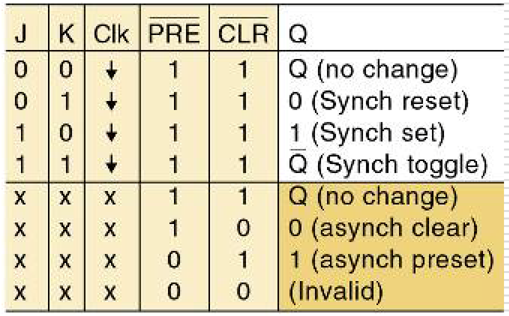
➢Most clocked FFs also have one or more asynchronous inputs which operate independently of the synchronous inputs and clock input.

➢Asynchronous inputs can be used to override all the other inputs in order to place the FF in one state or the other at any time.

➢IC manufacturers do not agree on nomenclature for asynchronous inputs.
- ▪The most common designations are *PRE* (PRESET) and *CLR* (CLEAR).
- ▪Labels such as *S-D* (direct SET) and *R-D* (direct RESET) are also used.

Clocked J-K flip-flop with asynchronous inputs.

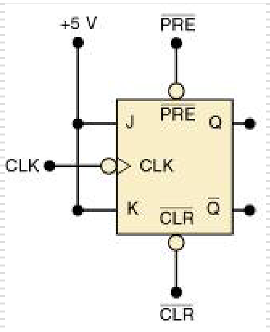


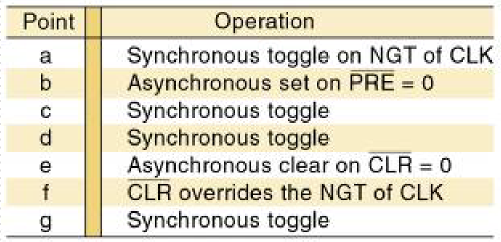


J	K	Clk	$\overline{PRE}$	$\overline{CLR}$	Q
0	0	↓	1	1	Q (no change)
0	1	↓	1	1	0 (Synch reset)
1	0	↓	1	1	1 (Synch set)
1	1	↓	1	1	$\overline{Q}$ (Synch toggle)
x	x	x	1	1	Q (no change)
x	x	x	1	0	0 (asynch clear)
x	x	x	0	1	1 (asynch preset)
x	x	x	0	0	(Invalid)

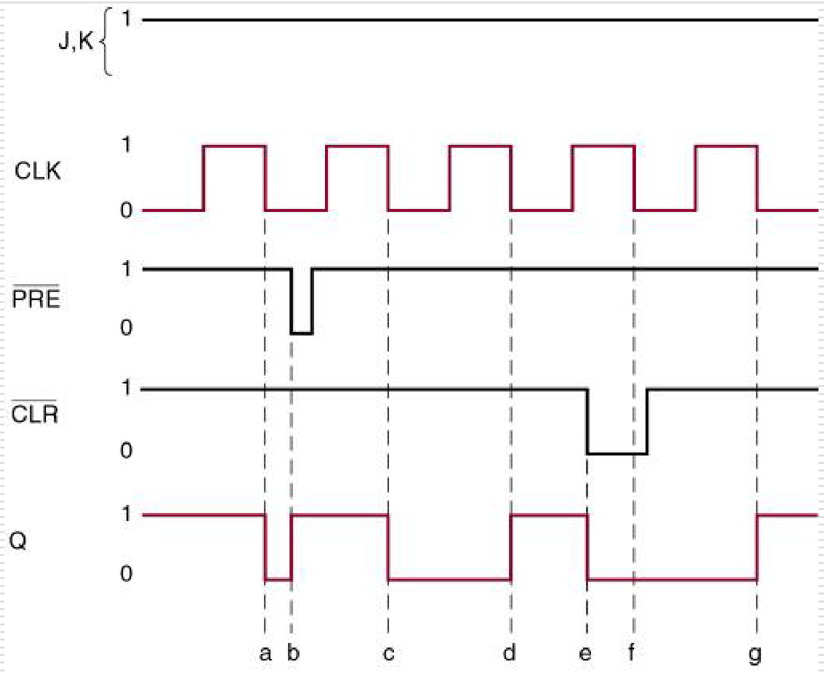
Asynchronous Inputs - Example

A J-K FF that responds to a NGT on its clock input and has active-LOW asynchronous inputs





Point	Operation
a	Synchronous toggle on NGT of CLK
b	Asynchronous set on $\overline{PRE} = 0$
c	Synchronous toggle
d	Synchronous toggle
e	Asynchronous clear on $\overline{CLR} = 0$
f	$\overline{CLR}$ overrides the NGT of CLK
g	Synchronous toggle



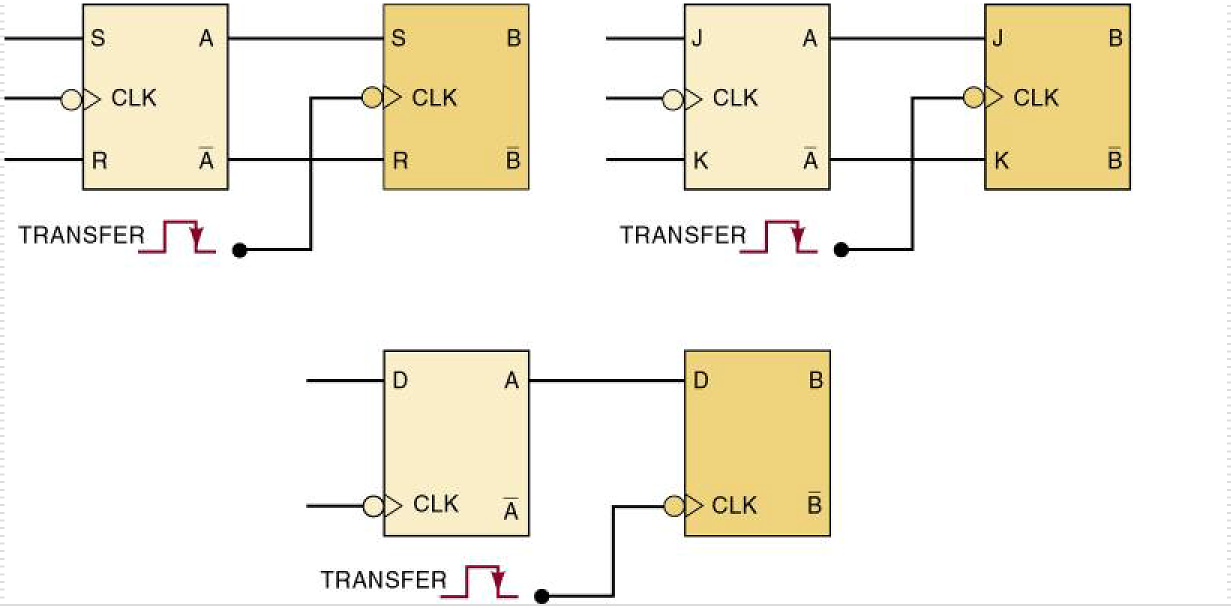
Flip-Flop Applications

- Examples of applications:
  - Storing binary data
  - Transferring binary data between locations
  - Synchronization
  - Frequency Division and Counting

Flip-flop Applications

- An example (Synchronous Data Storage and Transfer)

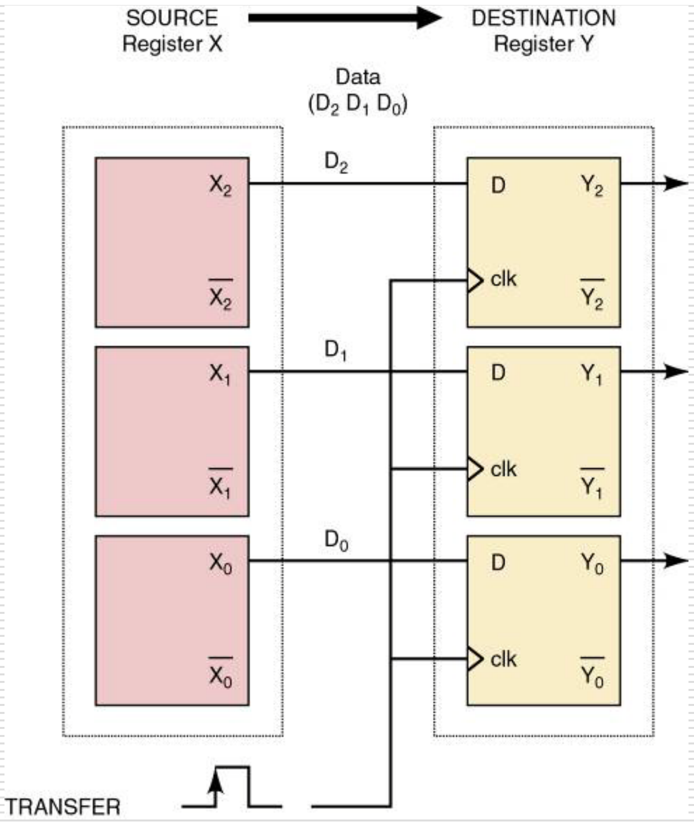
Synchronous data transfer operation by various clocked FFs.



***CLK* inputs are used to perform the transfer.**

Flip-flop Applications
- An example (Parallel Data Storage and Transfer)

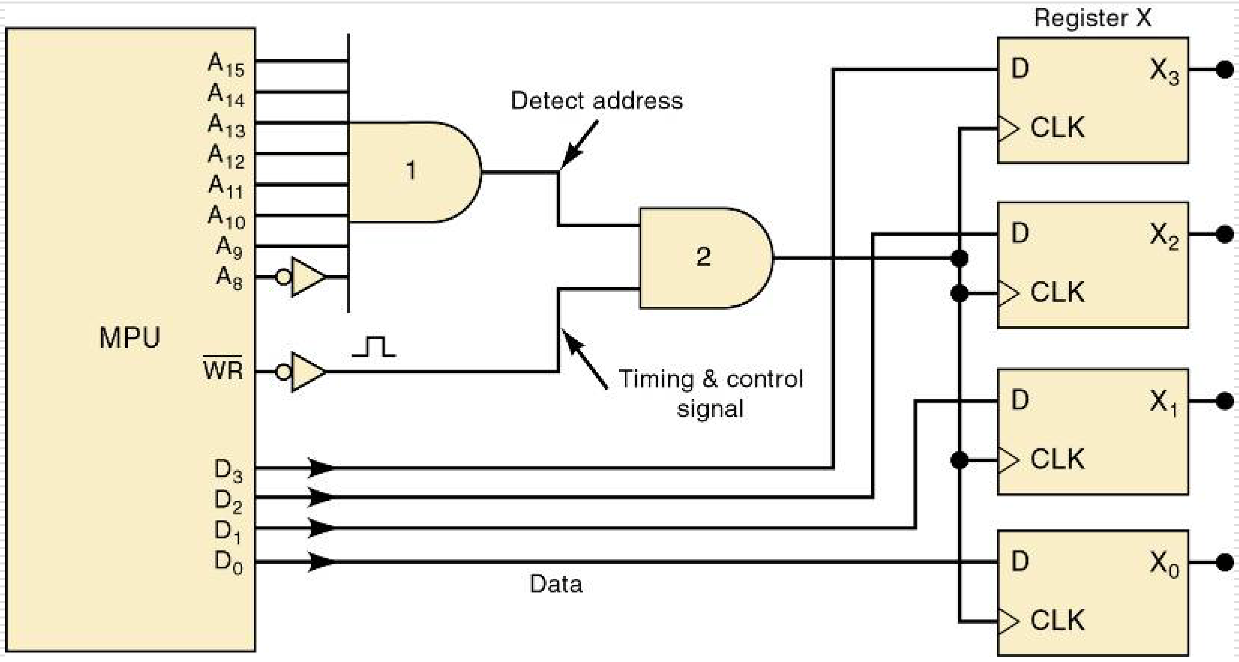
Transferring the bits of a register *simultaneously* is a *parallel* transfer.



Flip-flop Applications

- An example (Microcomputer application)

Microprocessor transferring binary data to an external register



- Microprocessor units (MPUs) perform many functions involving use of registers for data transfer and storage.

Finite State Machines

* Mealy and Moore Models

- Block Diagram
- State Diagram- Example
- State Table - Example

Finite State Machines
- Mealy and Moore Models

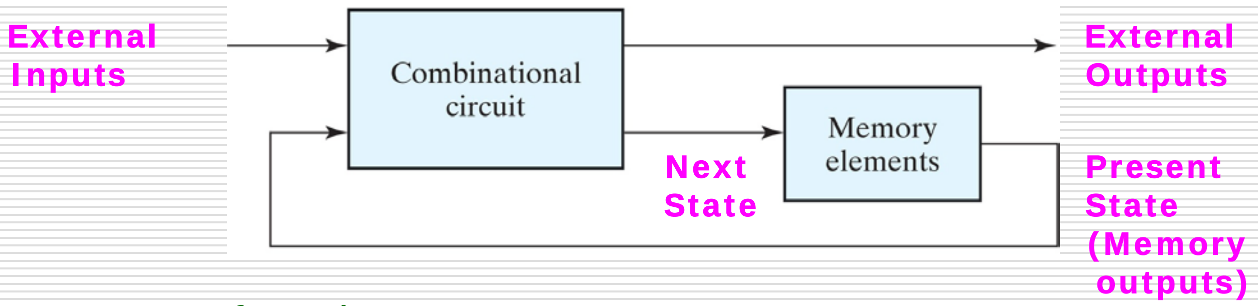
- Sequential Circuits or Sequential Machines are also called *Finite State Machines* (FSMs). Two formal models exist:

- Mealy Model
  - Named after G. Mealy
  - Outputs are a function of inputs and present states
  - Usually specified on the state transition arcs.

- Moore Model
  - Named after E.F. Moore
  - Outputs are a function of ONLY present states
  - Usually specified on the states.

Finite State Machines
- Mealy and Moore Models

- **Block Diagram of a Sequential Logic Circuit**



Next state function
Next State = f(Inputs, Present State)

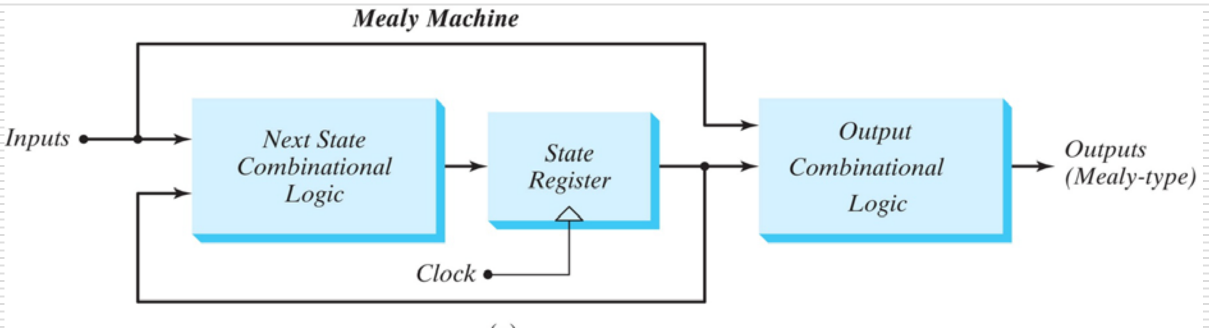
Output function (Mealy)
Outputs = g(Inputs, Present State)

Output function (Moore)
Outputs = h(Present State)

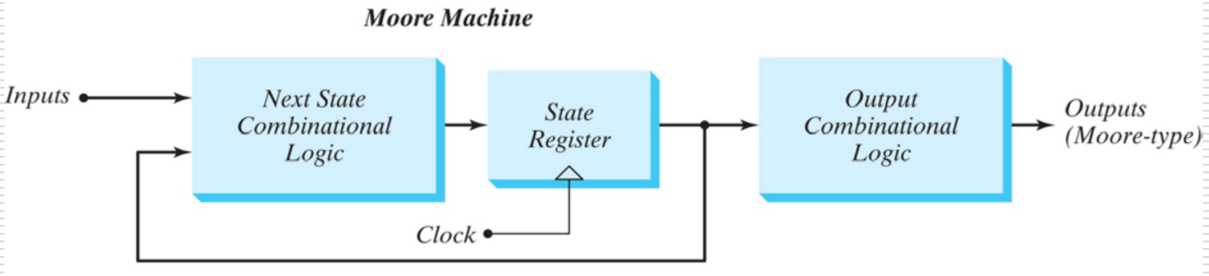
Output function type depends on specification and affects the design significantly

Finite State Machines

- Mealy and Moore Models (Block Diagrams)



(a)

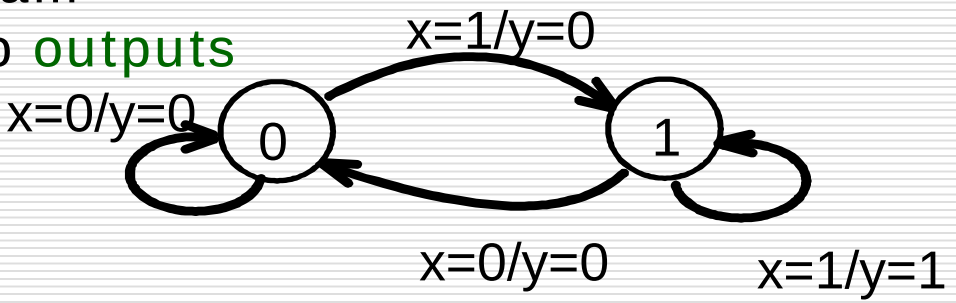


(b)

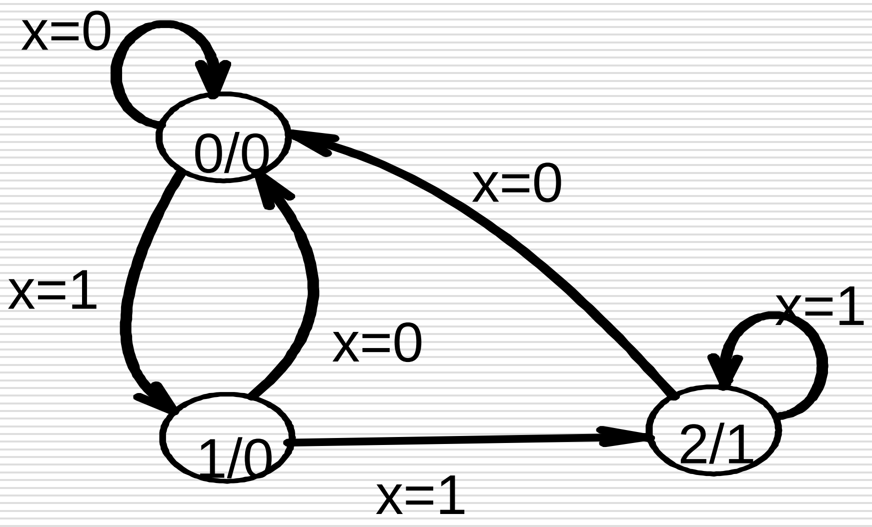
Finite State Machines

- Mealy and Moore Models (Example State Diagrams)

- Mealy Model State Diagram
  maps inputs and state to outputs



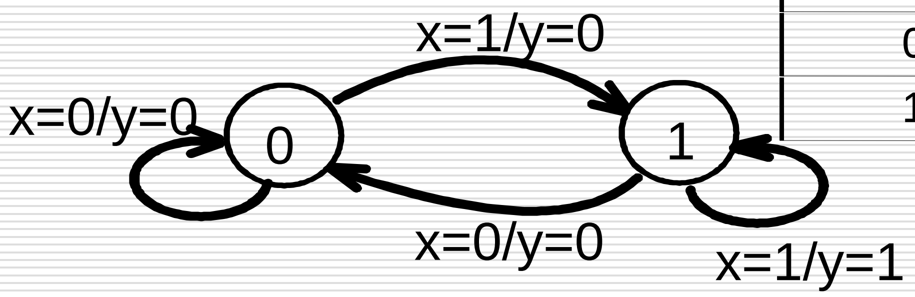
- Moore Model State Diagram
  maps states to outputs



Finite State Machines

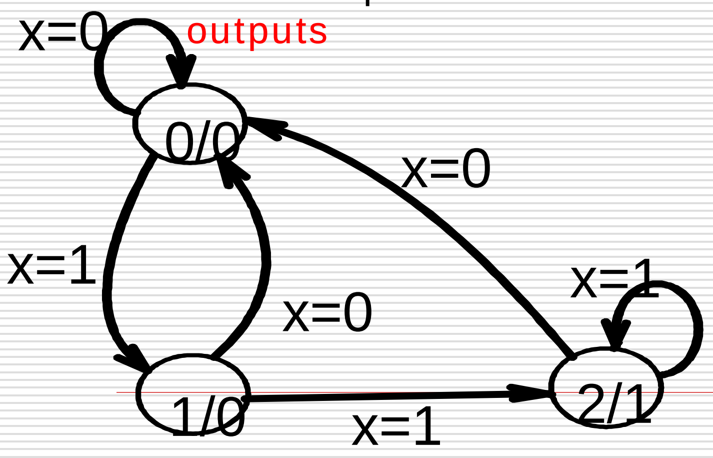
- Mealy and Moore Models (Example State Tables)

- **Mealy Model state table** maps inputs and state to outputs



Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
0	0	1	0	0
1	0	1	0	1

- **Moore Model state table** maps state to outputs



Present State	Next State x=0	Next State x=1	Output
0	0	1	0
1	0	2	0
2	0	2	1

References

Slides adopted from the books

- Ronald J.Tocci, Neal S.Widmer, and Gregory L.Moss, "Digital Systems- Principles and Application"- 10th Edition, Pearson Education International, 2007
- M.Morris Mano and Michael D. Ciletti, " Digital Design," 5th Edition, Pearson Education International, 2012
- M.Morris Mano and Charles R. Kime, " Logic and Computer Design Fundamentals," 4th Edition, Pearson Education International, 2008
- Thomas L.Floyd, "Digital Fundamentals," 10th Edition, Pearson Education International, 2009