Demultiplexers

DEMULTIPLEXERS

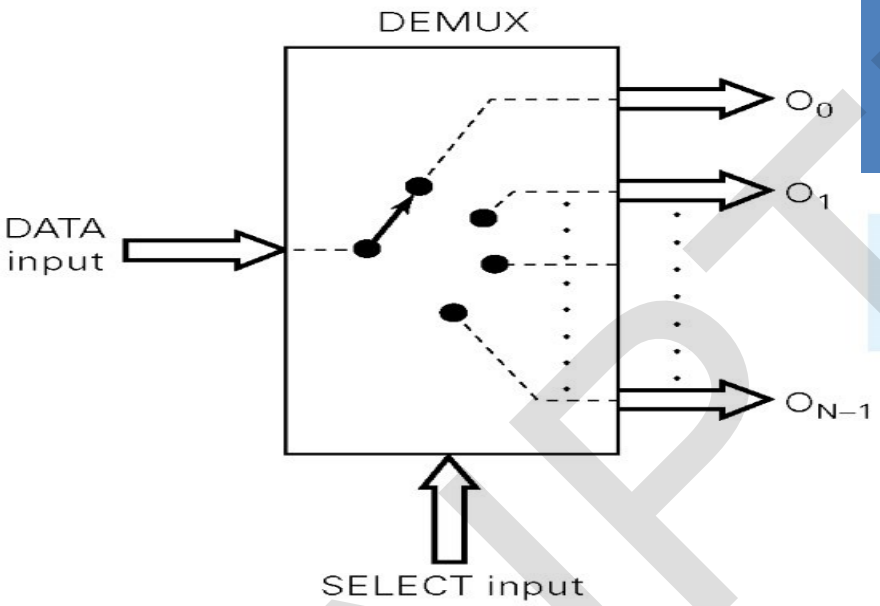
A DEMULTIPLEXER (DEMUX) basically reverses the multiplexing function. It takes data from one line and distributes them to a given number of output lines. For this reason, the demultiplexers is also known as a data distributor.

A multiplexer takes several inputs and transmits one of them to the output.

A demultiplexer (DEMUX) performs the reverse operation ; it takes a single input and distributes it over several outputs.

General demultiplexer

Functional diagram:-

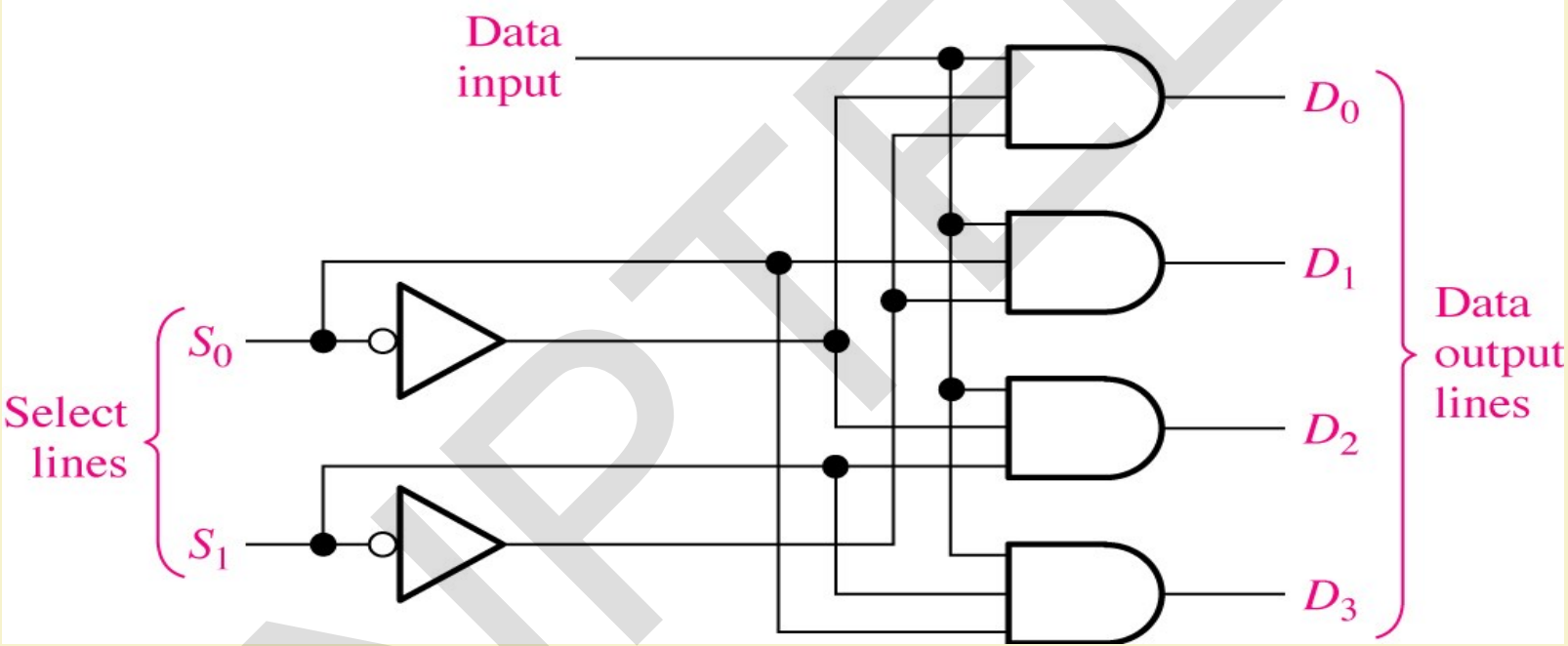


The large arrow indicates one or more lines. The select i/p code determines to which output the DATA input will be transmitted

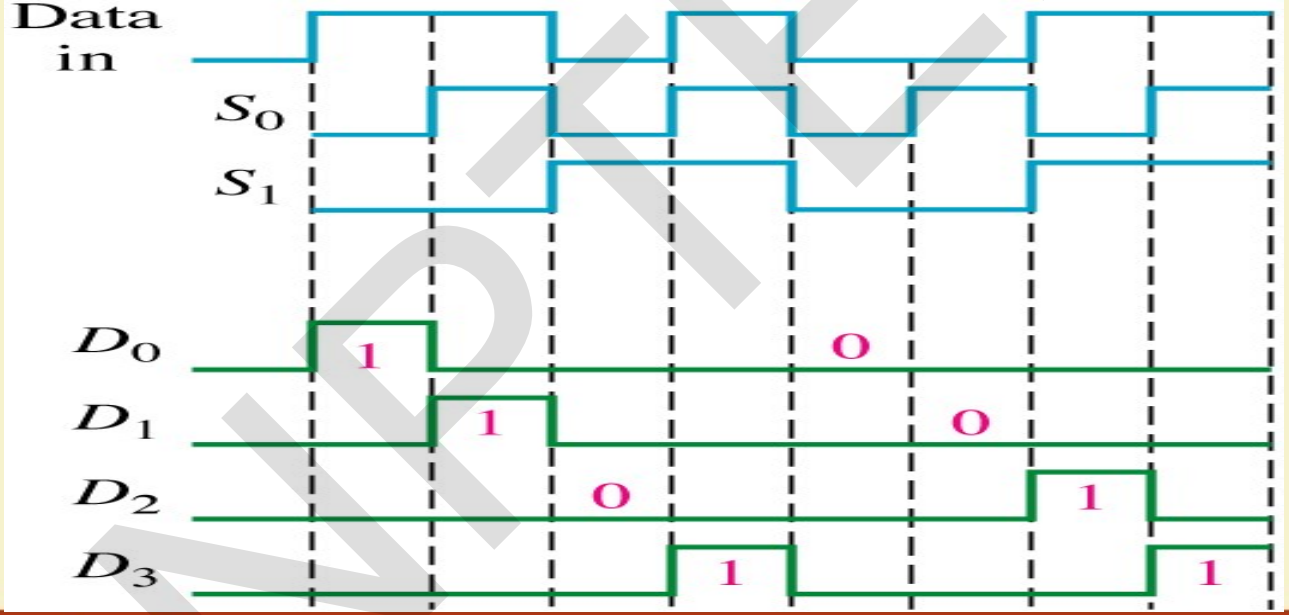
DATA input is transmitted to only one of the outputs as determined by select input code

In other words, the demultiplexer takes one data input source and selectively distributes it to 1 of N output channels just like multiposition switch.

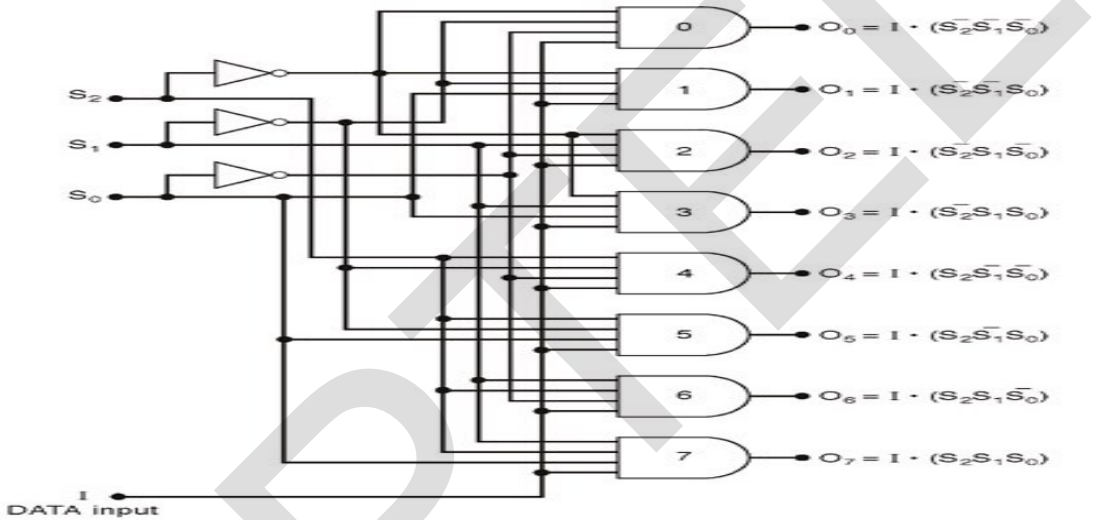
A 1-line-to-4-line demultiplexer.



The serial data input waveform (Data in) and data select inputs (S_0 and S_1) and the corresponding data output waveforms (D_0 through D_3) are shown below



1-line- to-8 line demultiplexer

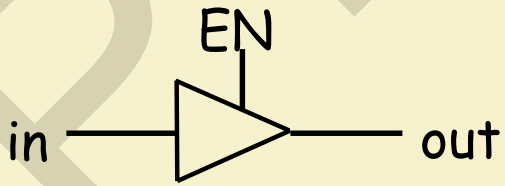


SELECT code			OUTPUTS							
S_2	S_1	S_0	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Note: I is the data input

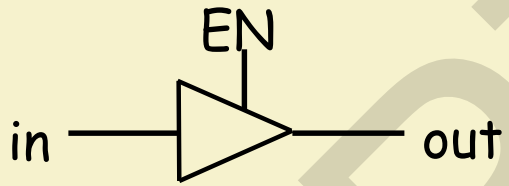
Buffers

- Buffer:
  - Doesn’t change the input.
  - Only amplifies.



Three-State Buffers

- Buffer output has 3 states: 0, 1, Z
- Z stands for High-Impedance ≅ Open circuit

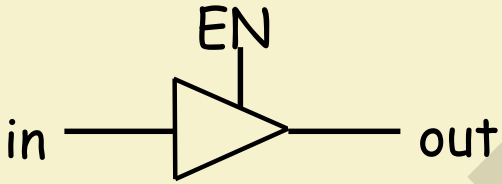


EN	in	out
0	X	Z
1	0	0
1	1	1

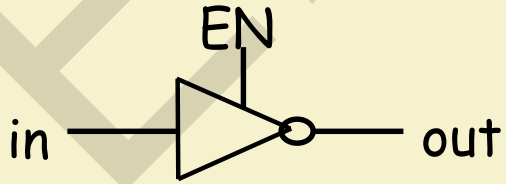
EN = 0 ➔ out = Z (open circuit)

EN = 1 ➔ out = in (regular buffer)

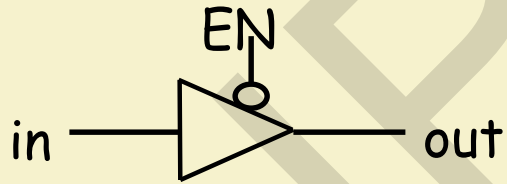
Three-state buffer(BUF)/inverter(INV) symbols



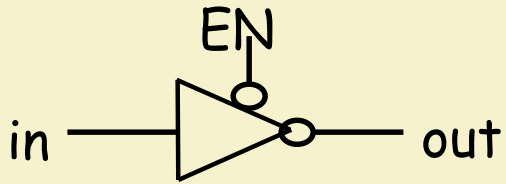
3-state BUF, EN high



3-state INV, EN high



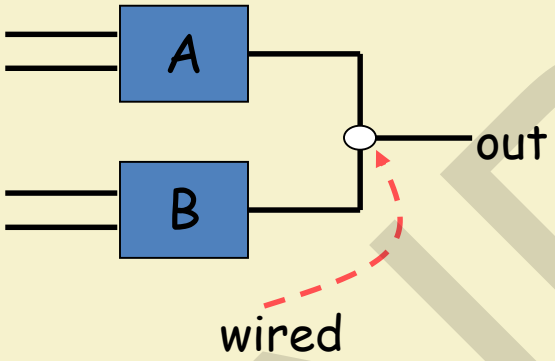
3-state BUF, EN low



3-state INV, EN low

Open Collector/Drain Gates

– Outputs of two (or more) gates must not be wired together:



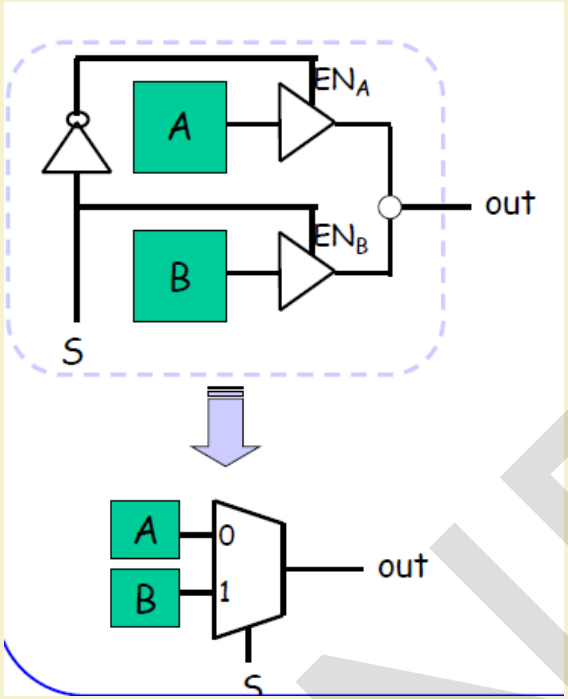
If $A = B$

→ out = $A = B$

If $A \neq B$

→ a large enough current can be created, that causes excessive heating and could damage the circuit.

Multiplexed output lines using three-state buffers



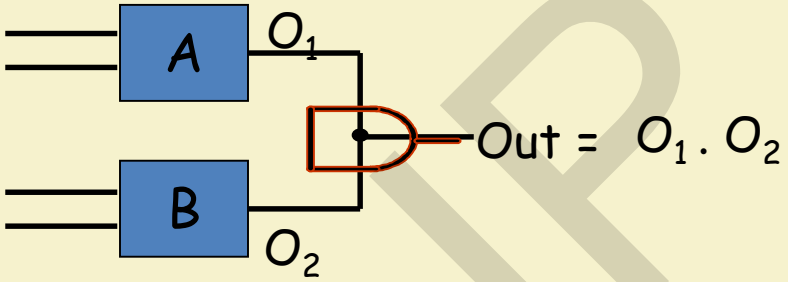
S	A	B	EN_A	EN_B	out
0	0	0	1	0	0
0	0	1	1	0	0
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	1	0	1	1
1	1	0	0	1	0
1	1	1	0	1	1

A

B

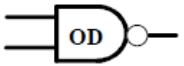
Open Collector/Drain Gates

- Outputs of **some** gates can be wired:
  - The result: O_1 AND O_2
    - Open Collector Gates in TTL Technology
    - Open Drain Gates in CMOS Technology



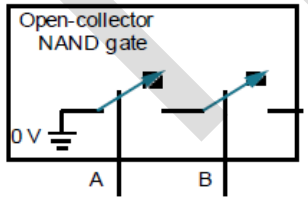
Open Collector/Drain Gates

– Open Collector (Open Drain) NAND Truth Table:



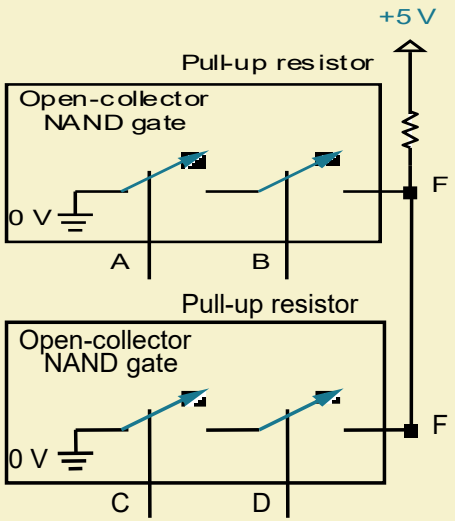
A	B	F=(A·B)'
0	0	Z
0	1	Z
1	0	Z
1	1	0

- **Z (Hi-Z) (Hi-Impedance):**
  - As if it is unconnected



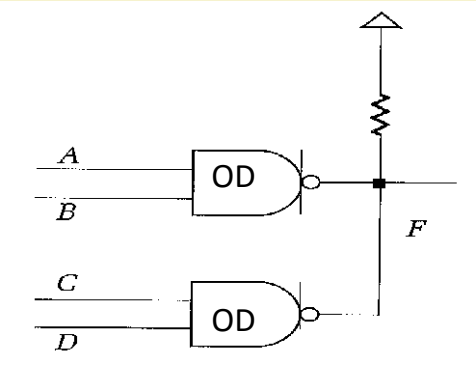
  - The gate cannot pull up the output
    - → needs a resistor to pull it up if an input is '0'

Open Collector/Drain Gates



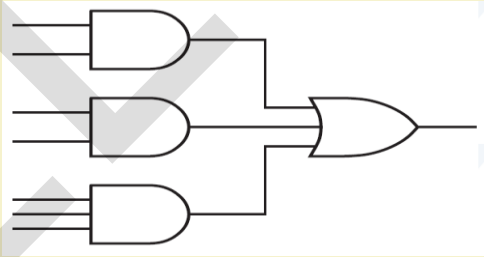
- Wired AND:

If A and B are "1", output is actively pulled low
if C and D are "1", output is actively pulled low
if one gate is low, the other high, then low wins
if both gates are "1", the output floats, pulled high by resistor
Hence, the two NAND functions are AND'd (wired) together!

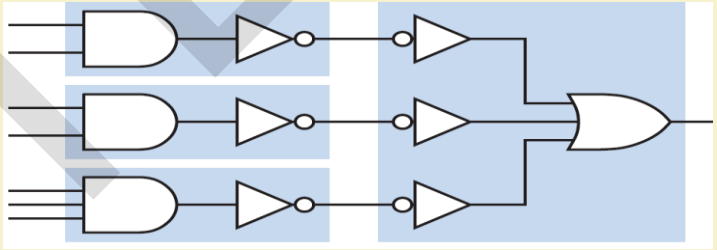


NAND-Only Implementation

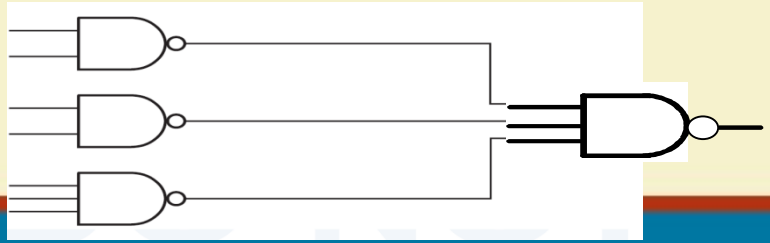
– Find Sum-of-Product form.



➢ Inventers can be added

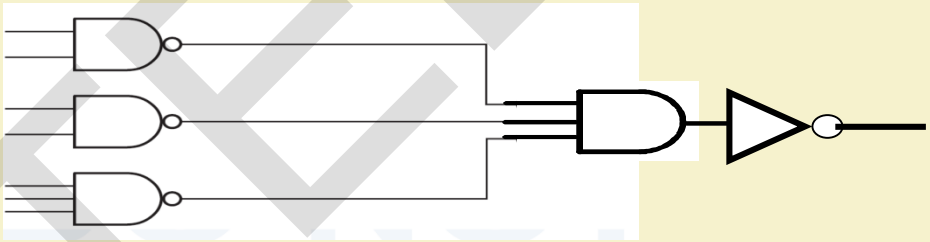


➢ Equivalent NAND-only

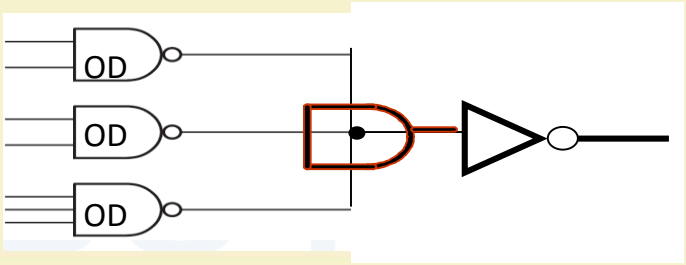


NAND-Only Implementation

- NAND = AND + NOT

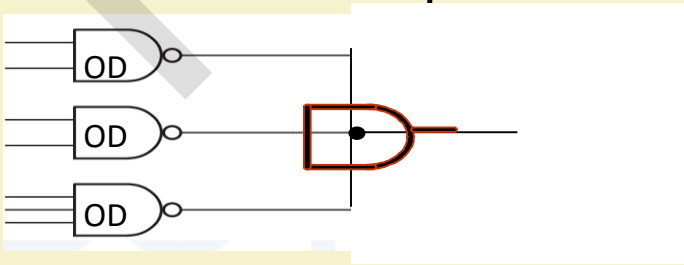


- Use Open Collector/Drain NAND Gates:



Another Method

- Instead of finding the circuit for F, find the circuit for F' in the first stage
  - Then there will be no Inverter at the output



Wired OR

- Open drain/collector gate with active low inputs.

Programmable Logic Array (PLA)

PLA Logic Implementation

Key to Success: Shared Product Terms

Equations

Example:

$F0 = A + \bar{B}\,\bar{C}$
$F1 = A\,\bar{C} + A\,B$
$F2 = \bar{B}\,\bar{C} + A\,B$
$F3 = \bar{B}\,C + A$

Personality Matrix

Product term	Inputs A	B	C	Outputs F_0	F_1	F_2	F_3
A B	1	1	-	0	(1)	(1)	0
$\bar{B}$ C	-	0	1	0	0	0	1
A $\bar{C}$	1	-	0	0	1	0	0
$\bar{B}$ $\bar{C}$	-	0	0	(1)	0	(1)	0
A	1	-	-	(1)	0	0	(1)

Reuse of terms

Input Side:

1 = asserted in term
0 = negated in term
- = does not participate

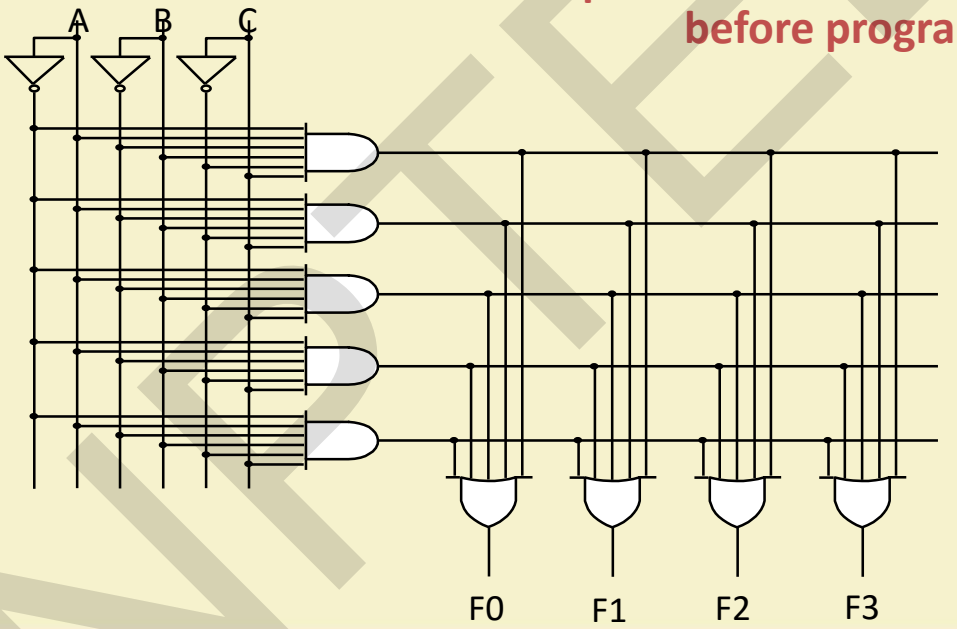
Output Side:

1 = term connected to output
0 = no connection to output

PLA Logic Implementation

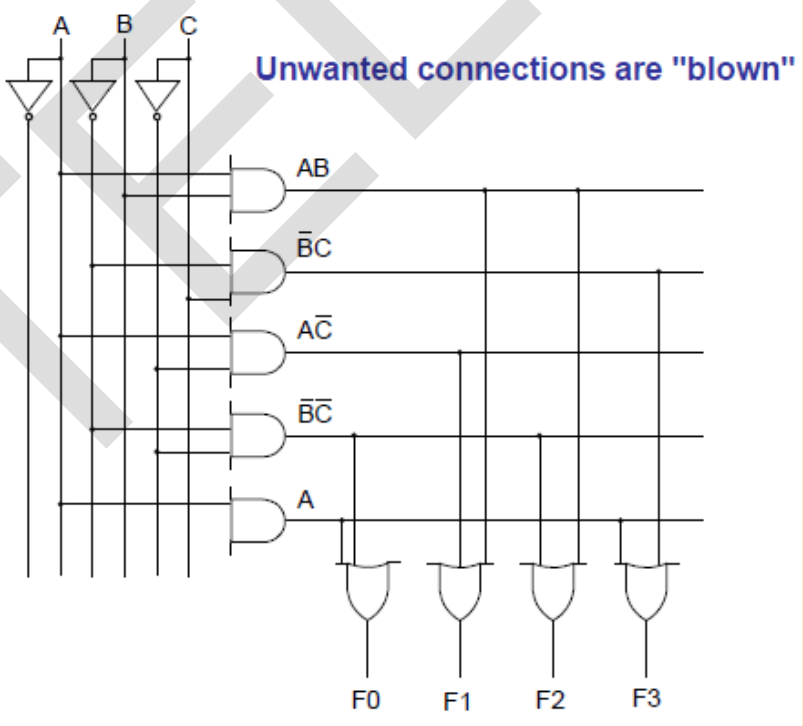
Example Continued - Unprogrammed device

All possible connections are available before programming



PLA Logic Implementation

Example Continued - Programmed part



Note: some array structures work by making connections rather than breaking them

PLA Logic Implementation

Alternative representation

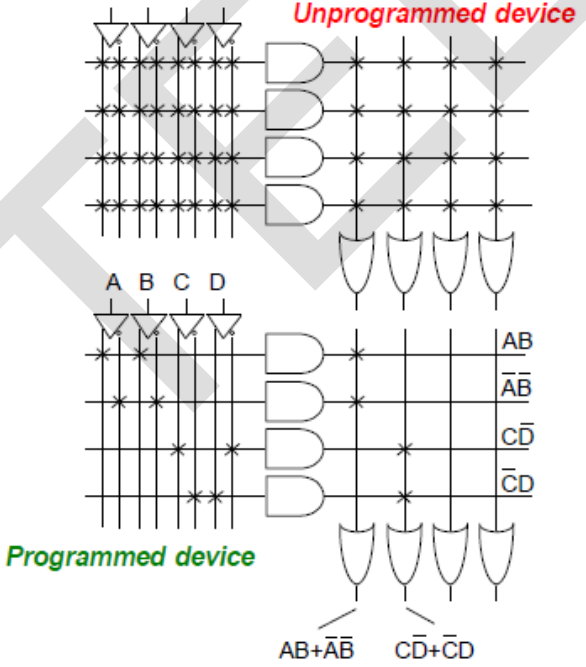
Short-hand notation so we don't have to draw all the wires!

X at junction indicates a connection

Notation for implementing

$F0 = A\,B + \bar{A}\,\bar{B}$

$F1 = C\,\bar{D} + \bar{C}\,D$



PLA Logic Implementation

Design Example

Multiple functions of A, B, C

$F1 = A\,B\,C$

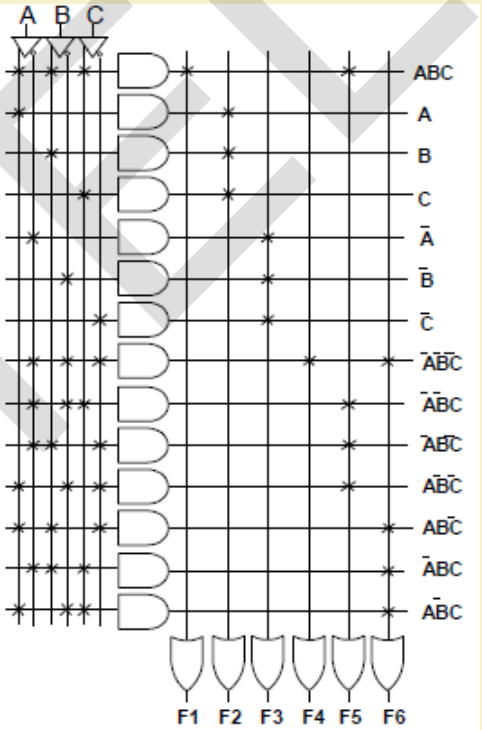
$F2 = A + B + C$

$F3 = \overline{A\,B\,C}$

$F4 = \overline{A + B + C}$

$F5 = A \oplus B \oplus C$

$F6 = \overline{A \oplus B \oplus C}$



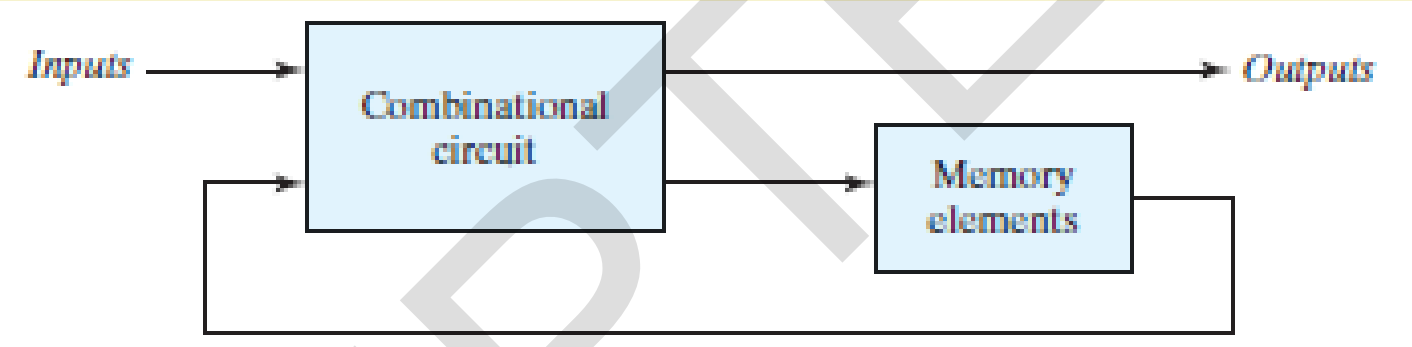
Sequential Circuits

Santanu Chattopadhyay
Electronics and Electrical Communication Engineering

Sequential Digital Circuits

- Sequential circuits are digital circuits in which the outputs depend not only on the current inputs, but also on the previous state of the output.
- The basic sequential circuit elements can be divided in two categories:
- Level-sensitive (Latches)
  - High-level sensitive
  - Low-level sensitive
- Edge-triggered (Flip-flops)
  - Rising (positive) edge triggered
  - Falling (negative) edge triggered
  - Dual-edge triggered

Basic Sequential Circuit

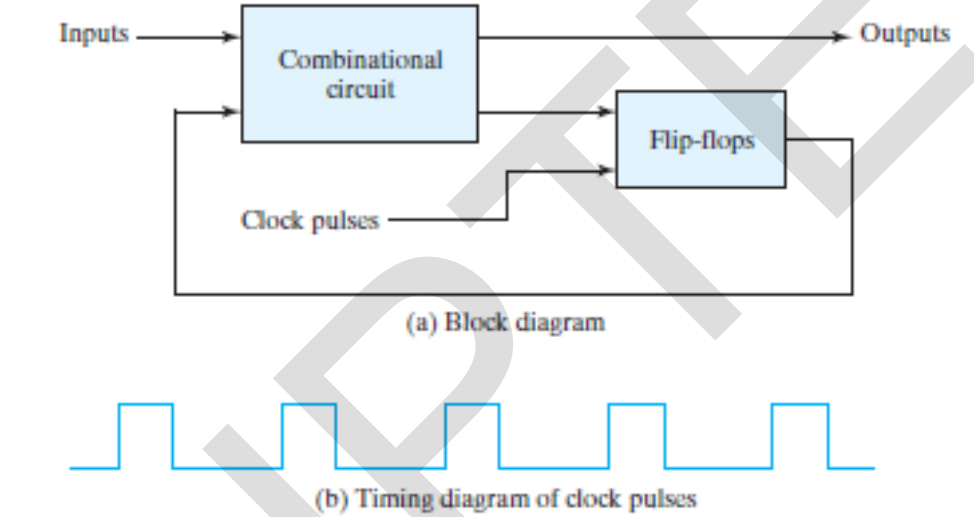


Synchronous *vs*. Asynchronous

There are two types of sequential circuits:

- **Synchronous** sequential circuit: circuit output changes only at some discrete instants of time. This type of circuits achieves synchronization by using a timing signal called the *clock*.
- **Asynchronous** sequential circuit: circuit output can change at **any** time (clockless).

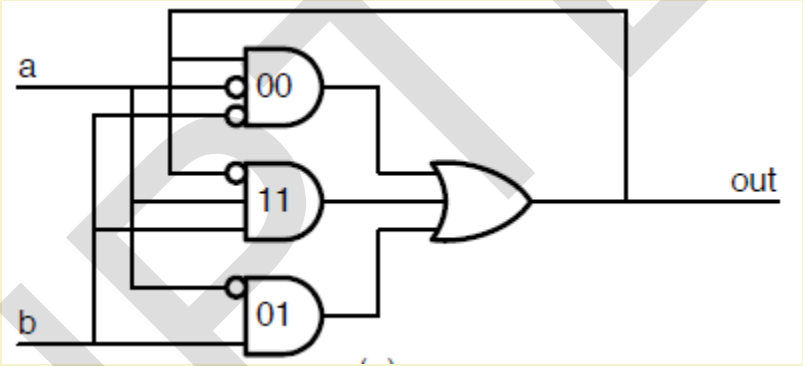
Synchronous Sequential Circuit



(a) Block diagram

(b) Timing diagram of clock pulses

Asynchronous Sequential Circuit



Applications - Synchronous

- Predominates asynchronous circuits
- Typically used to perform activities that need to happen at precise times

Applications – Asynchronous

- Signal processing
- Fast arithmetic unit
- Simple microprocessors
- Memory(static,RAM,FIFOs)
- ILLIAC

Advantages –Synchronous

- Simplicity
- Widely taught and understood
- Available components
- Simple way to deal with noise and hazard

Disadvantages –Synchronous

- Sensitive to variations in physical parameters
- Not modular
- Power consumption
- Clock distribution is difficult due to clock skew
- The maximum possible clock rate is determined by the slowest logic path in the circuit, otherwise known as the critical path

Advantages –Asynchronous

- High performance
- Low power dissipation
- Low noise and EM emission
- Good match with heterogeneous system timing

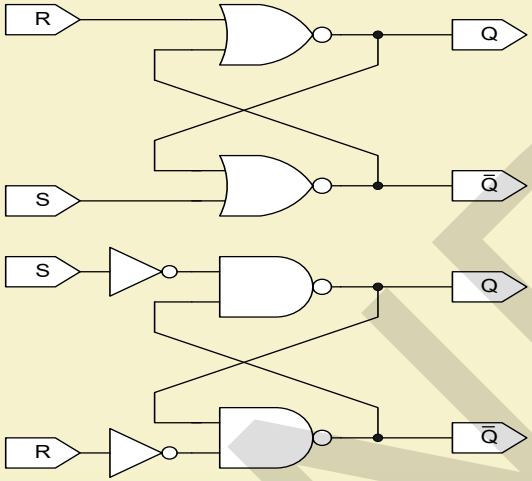
Disadvantages – Asynchronous

- Substantial circuit level overhead
- Lack of CAD tools
- Delay

The Set/Reset (SR) Latch

The Set/Reset latch is the most basic unit of sequential digital circuits. It has two inputs (S and R) and two outputs outputs Q and Q'. The two outputs must always be complementary, i.e if Q is 0 then Q' must be 1, and vice-versa. The S input sets the Q output to a logic 1. The R input resets the Q output to a logic 0.

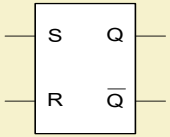
Circuit Diagram



Truth Table

S	R	Q+	Q´+	Function
0	0	Q	Q´	Latch
0	1	0	1	Reset
1	0	1	0	Set
1	1	0	0	Illegal

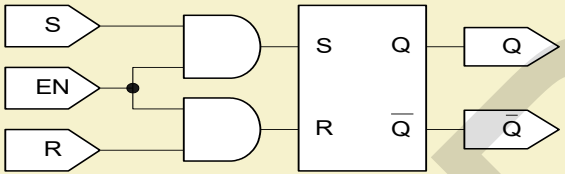
Logic Symbol



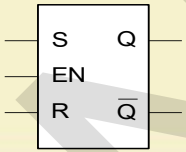
The Gated Set/Reset (SR) Latch

To be able to control when the S and R inputs of the SR latch can be applied to the latch and thus change the outputs, an extra input is used. This input is called the Enable. If the Enable is 0 then the S and R inputs have no effect on the outputs of the SR latch. If the Enable is 1 then the Gated SR latch behaves as a normal SR latch.

Circuit Diagram



Logic Symbol



Truth Table

EN	S	R	Q+
0	0	0	Q
0	0	1	Q
0	1	0	Q
0	1	1	Q
1	0	0	Q
1	0	1	0
1	1	0	1
1	1	1	U

Truth Table

EN	S	R	Q+	Function
0	X	X		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

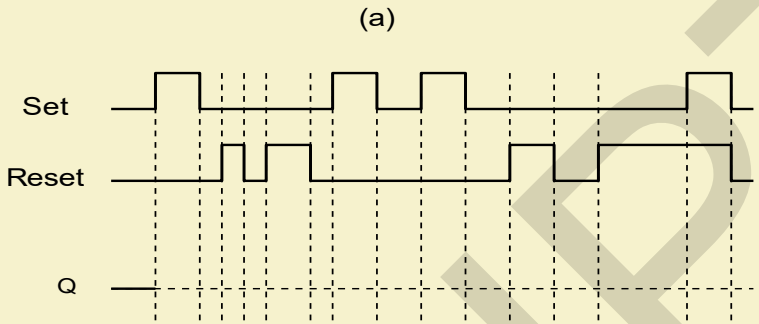
SR Latch :- Example

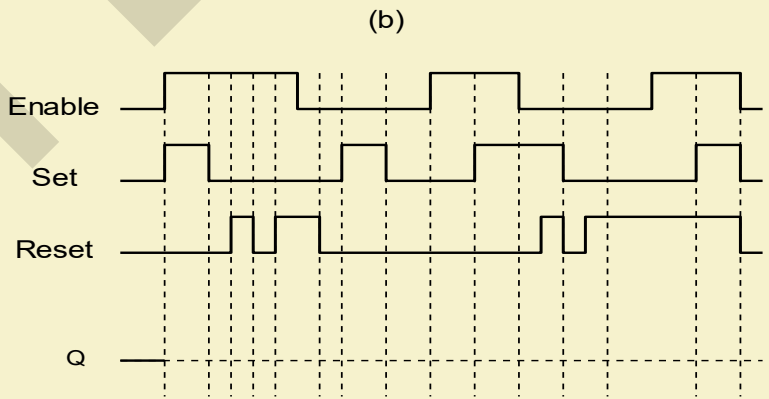
Complete the timing diagrams for :

(a) Simple SR Latch

(b) SR Latch with Enable input.

Assume that for both cases the Q output is initially at logic zero.

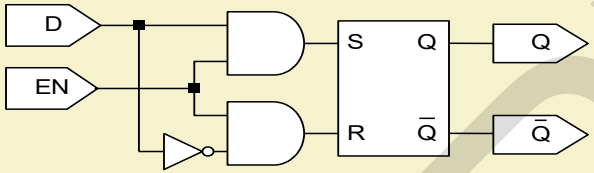




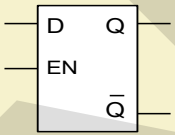
The Data (D) Latch

A problem with the SR latch is that the S and R inputs can not be at logic 1 at the same time. To ensure that this can not happen, the S and R inputs can by connected through an inverter. In this case the Q output is always the same as the input, and the latch is called the Data or D latch. The D latch is used in Registers and memory devices.

Circuit Diagram



Logic Symbol



Truth Table

EN	D	Q	Q+
0	0	0	Q
0	0	1	Q
0	1	0	Q
0	1	1	Q
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

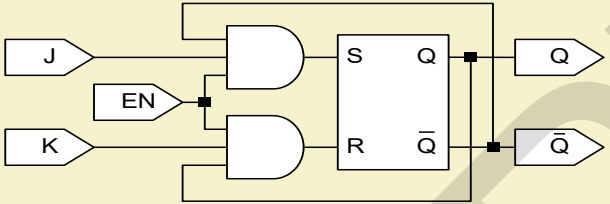
Truth Table

EN	D	Q+	Function
0	0		
0	1		
1	0		
1	1		

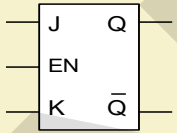
The JK Latch

Another way to ensure that the S and R inputs can not be at logic 1 simultaneously, is to cross connect the Q and Q' outputs with the S and R inputs through AND gates. The latch obtained is called the JK latch. In the J and K inputs are both 1 then the Q output will change state (Toggle) for as long as the Enable 1, thus the output will be unstable. This problem is avoided by ensuring that the Enable is at logic 1 only for a very short time, using edge detection circuits.

Circuit Diagram



Logic Symbol



Truth Table

EN	J	K	Q	Q+
0	X	X	X	Q
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Truth Table

EN	J	K	Q+	Function
0	X	X		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

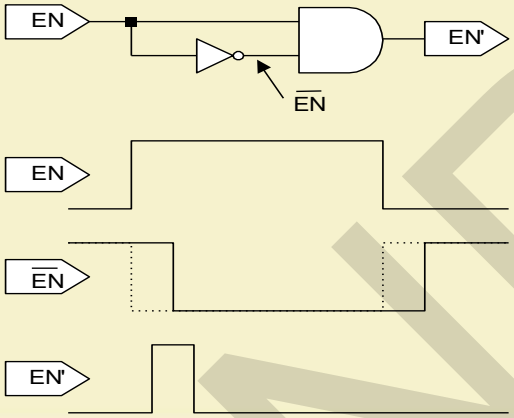
Latches and Flip-Flops

- **Latches** are also called **transparent** or **level triggered** flip flops, because the change on the outputs will follow the changes of the inputs as long as the Enable input is set.
- **Edge triggered** flip flops are the flip flops that change there outputs only at the transition of the Enable input. The enable is called the Clock input.

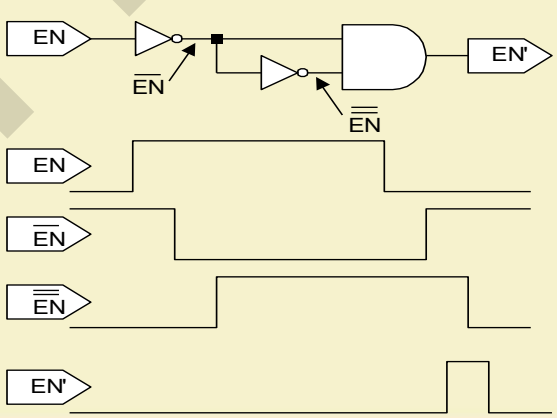
Edge Detection Circuits

Edge detection circuits are used to detect the transition of the Enable from logic 0 to logic 1 (positive edge) or from logic 1 to logic 0 (negative edge). The operation of the edge detection circuits shown below is based on the fact that there is a time delay between the change of the input of a gate and the change at the output. This delay is in the order of a few nanoseconds. The Enable in this case is called the Clock (CLK)

Positive Edge Detection



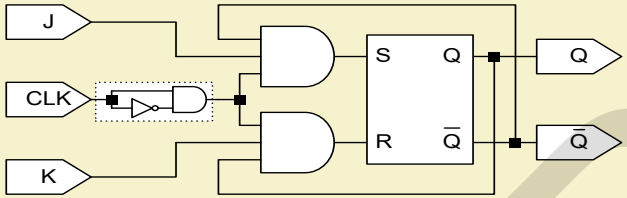
Negative Edge Detection



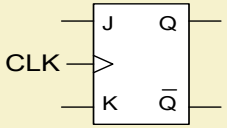
The JK Edge Triggered Flip Flop

The JK edge triggered flip flop can be obtained by inserting an edge detection circuit at the Enable (CLK) input of a JK latch. This ensures that the outputs of the flip flop will change only when the CLK changes (0 to 1 for +ve edge or 1 to 0 for –ve edge)

Positive Edge JK Flip Flop

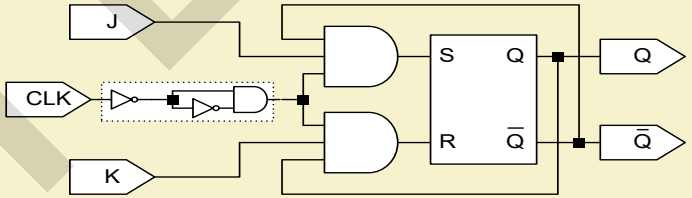


Logic Symbol

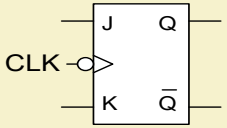


CLK	J	K	Q_{N+1}	Function
⌐	X	X	Q	
↑	0	0	Q	
↑	0	1	0	
↑	1	0	1	
↑	1	1	Q'	

Negative Edge JK Flip Flop



Logic Symbol

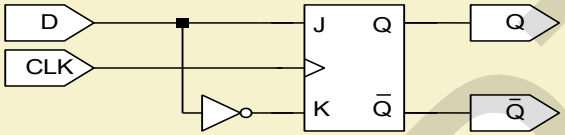


CLK	J	K	Q_{N+1}	Function
⌐	X	X		
↓	0	0		
↓	0	1		
↓	1	0		
↓	1	1		

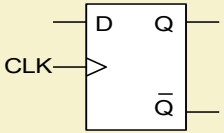
The D Edge Triggered Flip Flop

The D edge triggered flip flop can be obtained by connecting the J with the K inputs of a JK flip through an inverter as shown below. The D edge trigger can also be obtained by connecting the S with the R inputs of a SR edge triggered flip flop through an inverter.

Positive Edge D Flip Flop

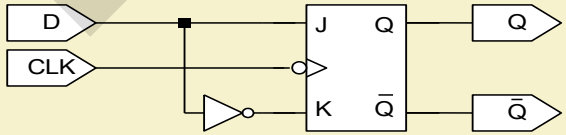


Logic Symbol

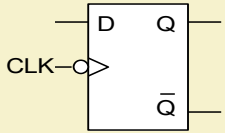


CLK	D	Q_{N+1}	Function
↓	X	Q	
↑	0	0	
↑	1	1	

Negative Edge D Flip Flop



Logic Symbol

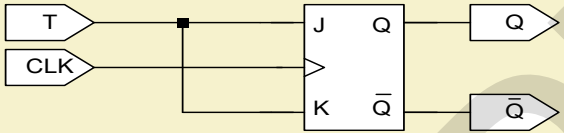


CLK	D	Q_{N+1}	Function
↑	X	Q	
↓	0	0	
↓	1	1	

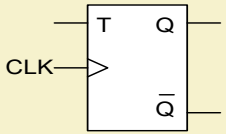
The Toggle (T) Edge Triggered Flip Flop

The T edge triggered flip flop can be obtained by connecting the J with the K inputs of a JK flip directly. When T is zero then both J and K are zero and the Q output does not change. When T is one then both J and K are one and the Q output will change to the opposite state, or toggle.

Positive Edge T Flip Flop

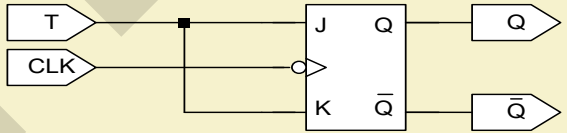


Logic Symbol

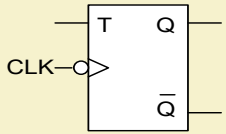


CLK	T	Q_{N+1}	Function
↓	X	Q	
↑	0	Q	
↑	1	Q´	

Negative Edge T Flip Flop



Logic Symbol

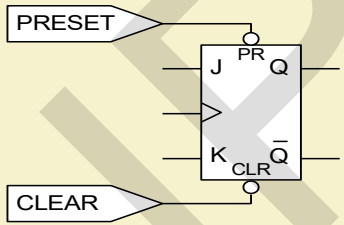


CLK	T	Q_{N+1}	Function
↑	X	Q	
↓	0	Q	
↓	1	Q´	

Flip Flops with asynchronous inputs (Preset and Clear)

Two extra inputs are often found on flip flops, that either clear or preset the output. These inputs are effective at any time, thus are called asynchronous. If the Clear is at logic 0 then the output is forced to 0, irrespective of the other normal inputs. If the Preset is at logic 0 then the output is forced to 1, irrespective of the other normal inputs. The preset and the clear inputs can not be 0 simultaneously. If the Preset and Clear are both 1 then the flip flop behaves according to its normal truth table.

Positive Edge JK Flip Flop with Preset and Clear



CLK	PR	CLR	J	K	Q_{N+1}	Function
↓	0	0	X	X		
↑	0	1	X	X	1	
↑	1	0	X	X	0	
↑	1	1	0	0	Q	
↑	1	1	0	1	0	
↑	1	1	1	0	1	
↑	1	1	1	1	Q'	

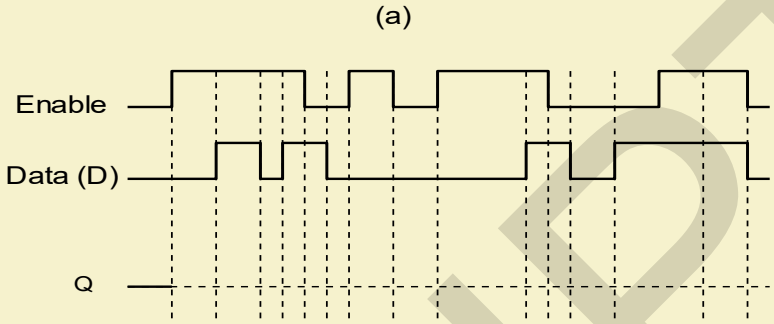
Data (D) Latch :- Example

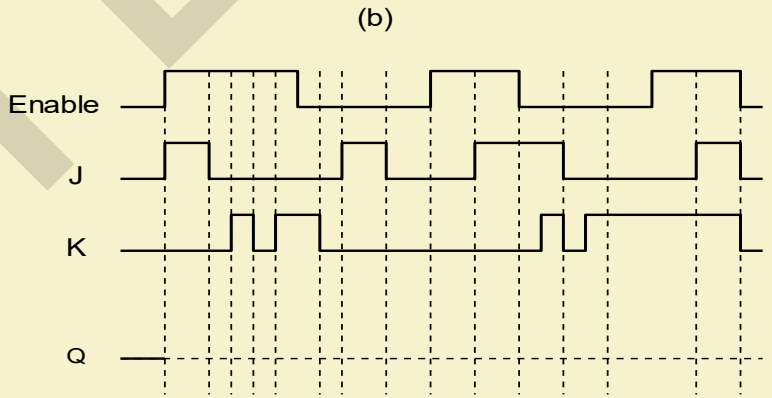
Complete the timing diagrams for :

(a) D Latch

(b) JK Latch

Assume that for both cases the Q output is initially at logic zero.





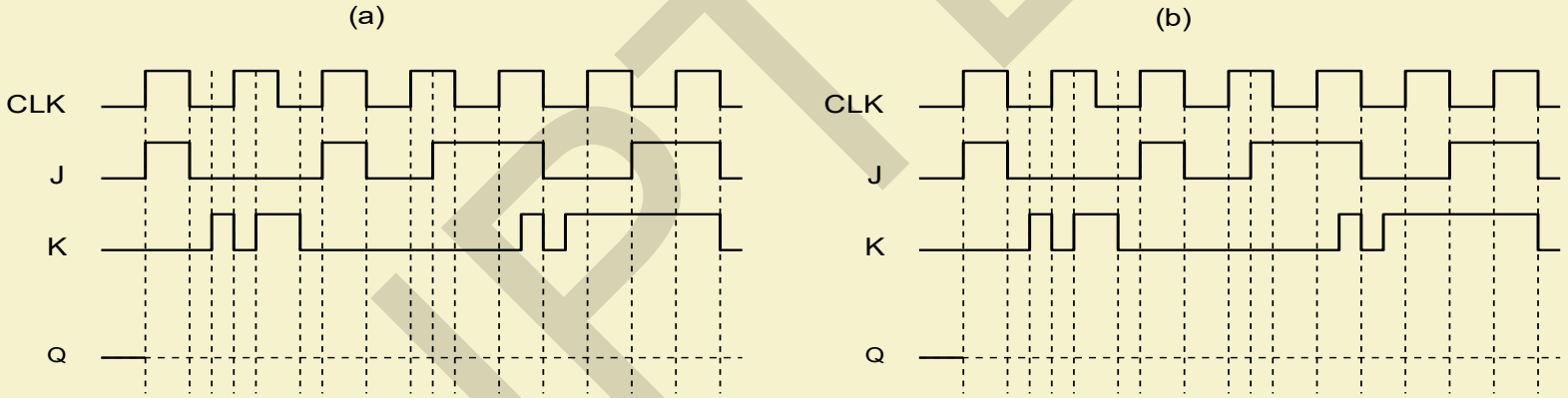
JK Edge Triggered Flip Flop :- Example

Complete the timing diagrams for :

(a) Positive Edge Triggered JK Flip Flop

(b) Negative Edge Triggered JK Flip Flop

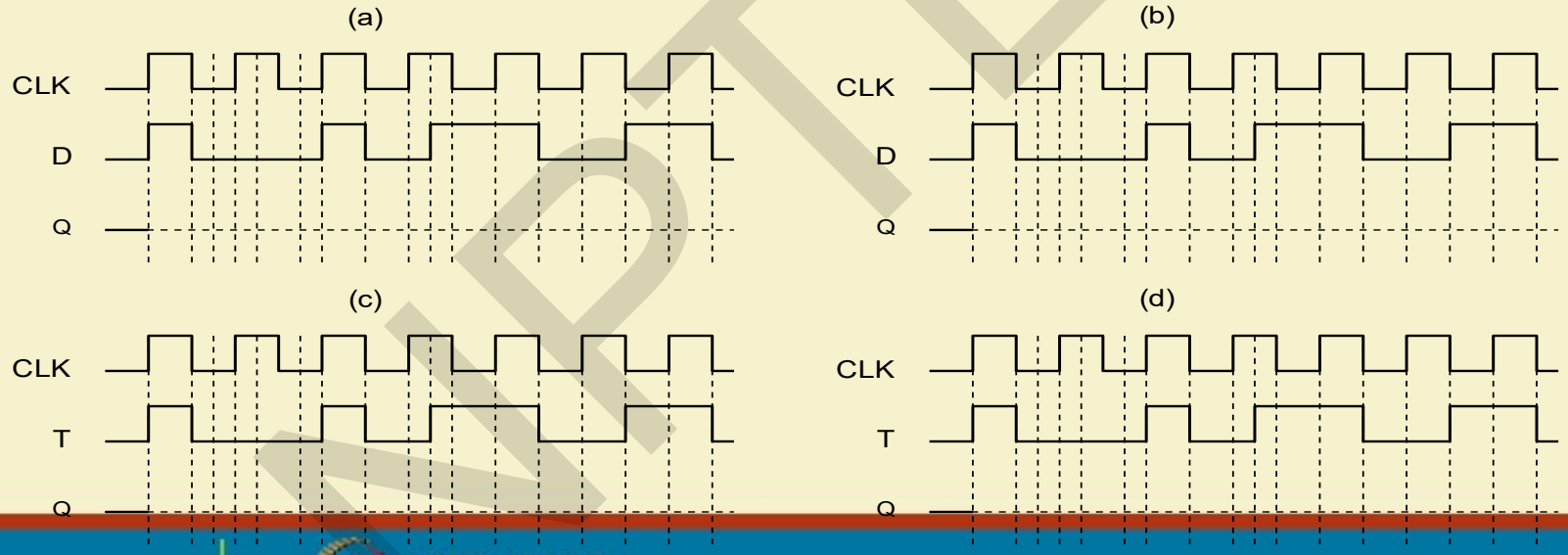
Assume that for both cases the Q output is initially at logic zero.



D and T Edge Triggered Flip Flops :- Example

Complete the timing diagrams for :

(a) Positive Edge Triggered D Flip Flop
(b) Positive Edge Triggered T Flip Flop
(c) Negative Edge Triggered T Flip Flop
(d) Negative Edge Triggered D Flip Flop



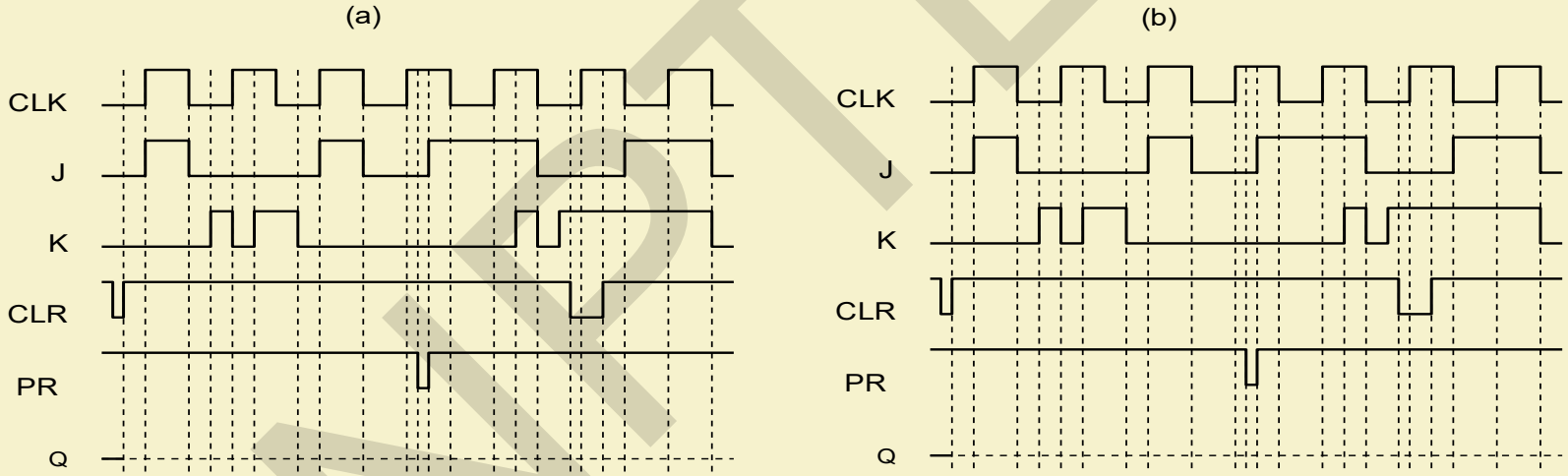
JK Flip Flop With Preset and Clear:- Example

Complete the timing diagrams for :

(a) Positive Edge Triggered JK Flip Flop

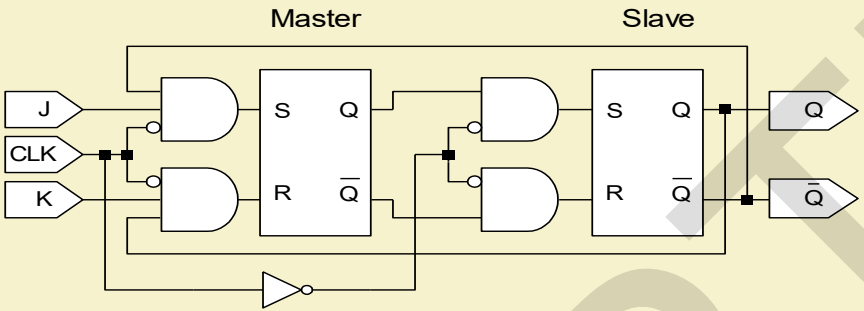
(b) Negative Edge Triggered JK Flip Flop.

Assume that for both cases the Q output is initially at logic zero.

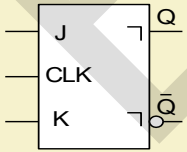


Level Triggered Master Slave JK Flip Flop

A Master Slave flip flop is obtained by connecting two SR latches as shown below. This flip flop reads the inputs when the clock is 1 and changes the output when the clock is at logic zero.



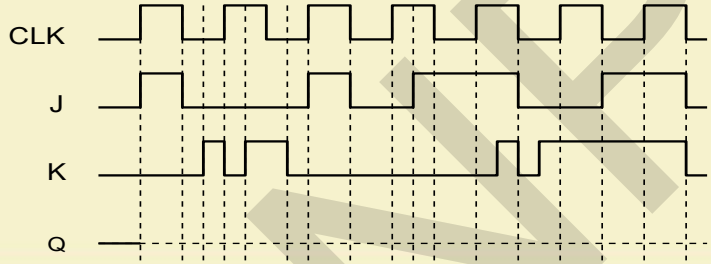
Logic Symbol



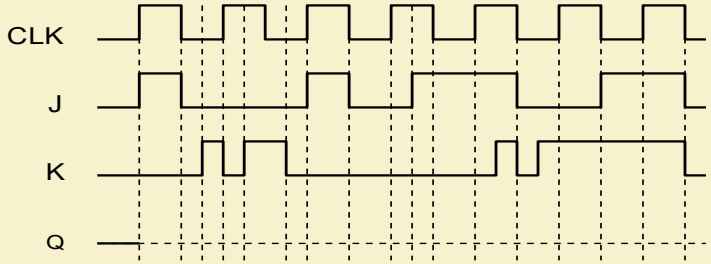
Truth Table

CLK	J	K	Q	Function
⎍	0	0		
⎍	0	1		
⎍	1	0		
⎍	1	1		

(a) Positive Master Slave JK Flip Flop

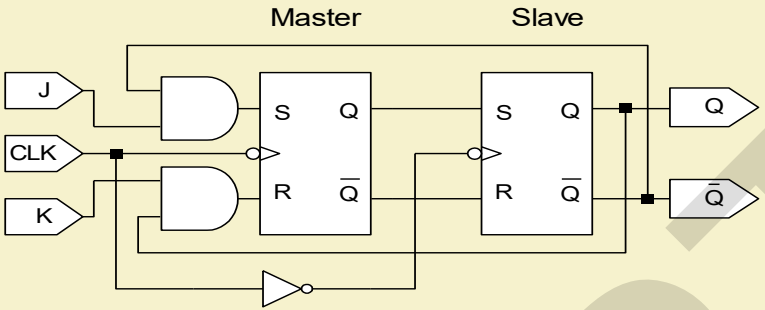


(b) Negative Master Slave JK Flip Flop

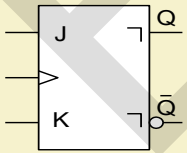


Edge Triggered Master Slave JK Flip Flop

A Master Slave flip flop is obtained by connecting two SR latches as shown below. This flip flop reads the inputs when the clock is 1 and changes the output when the clock is at logic zero.



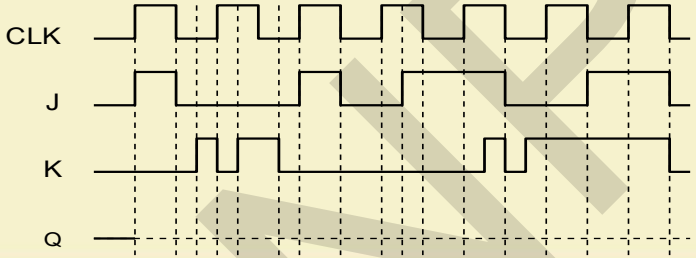
Logic Symbol



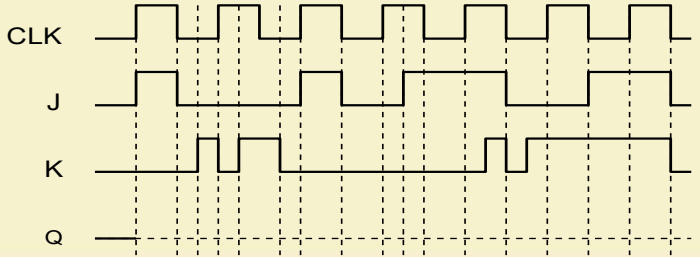
Truth Table

CLK	J	K	Q	Function
↓	0	0		
↓	0	1		
↓	1	0		
↓	1	1		

(a) Positive Master Slave JK Flip Flop

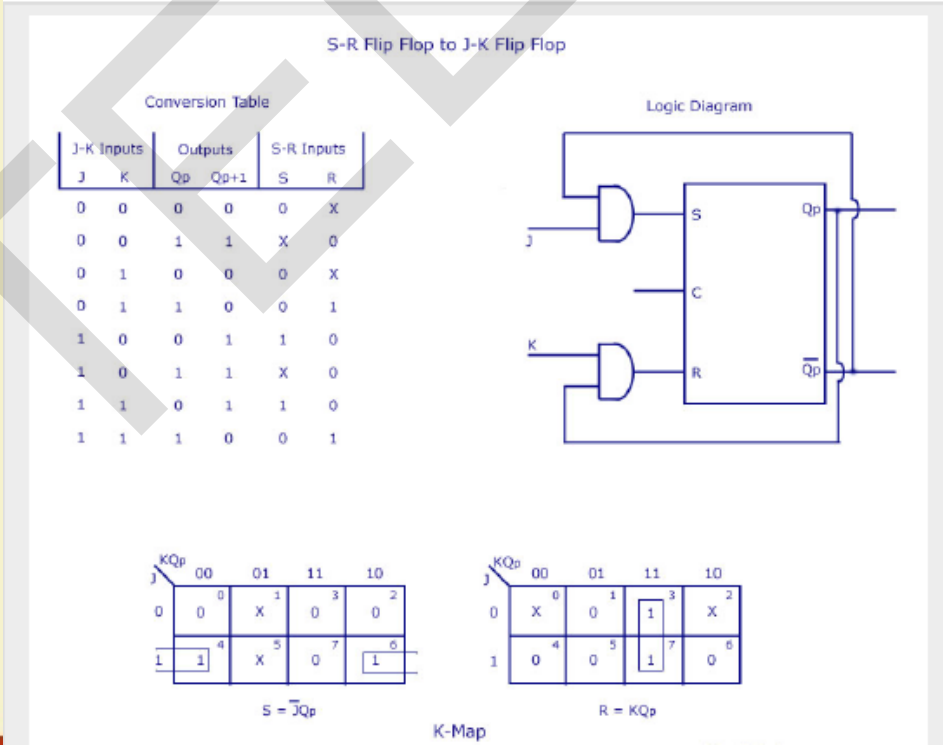


(b) Negative Master Slave JK Flip Flop



Conversion between Flipflops

SR to JK

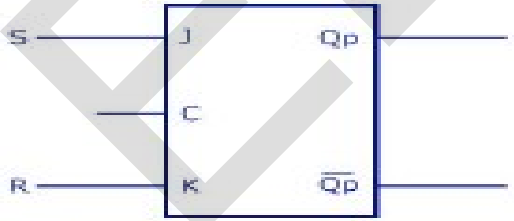


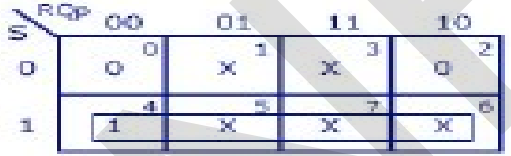
JK to SR

Conversion Table

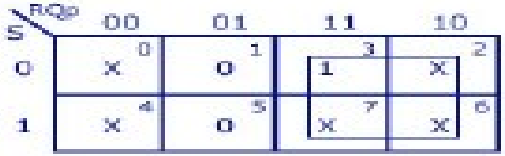
S-R Inputs		Outputs		J-K Inputs	
S	R	Qp	Qp+1	J	K
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	X	1
1	0	0	1	1	X
1	0	1	1	X	0
1	1	Invalid		Dont care	
1	1	Invalid		Dont care	

Logic Diagram





J=S



K=R

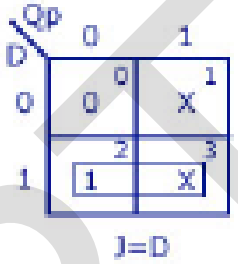
K-maps

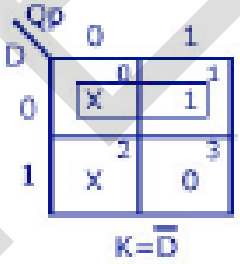
JK to D

Conversion Table

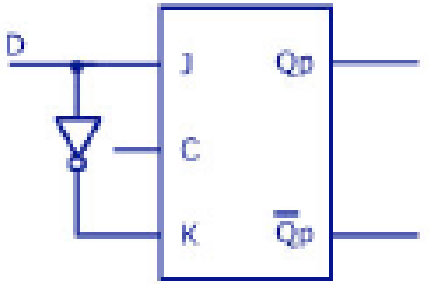
D Input	Outputs		J-K Inputs	
	Qp	Qp+1	J	K
0	0	0	0	X
0	1	0	X	1
1	0	1	1	X
1	1	0	X	0

K-maps





Logic Diagram

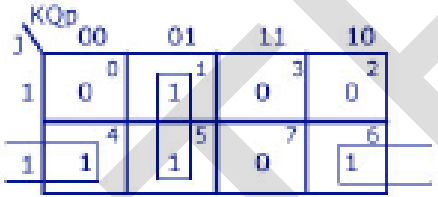


D to JK

Conversion Table

J-K Input		Outputs		D Input
J	K	Qp	Qp+1	
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

K-map



$D = J\overline{Q}p + \overline{K}Qp$

Logic Diagram