# DIGITAL ELECTRONICS: ECE 213


**Topic: Adders, Subtractors, Decoders and Multiplexer**

**UNIT III: Introduction to Combinational Logic Circuits and Logic Families**

**Lecture No.: 15**

**Prepared By: Irfan Ahmad Pindoo**

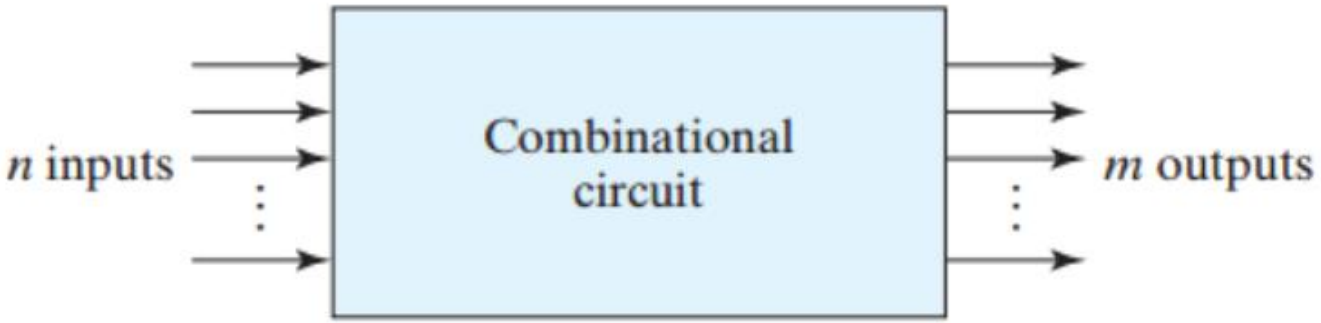
**Assistant Professor**

**VLSI Design, ECE**

**School of Computer Science and Engineering**

# Introduction to Combinational Circuits

- Combinational logic is a type of digital logic which is implemented by Boolean circuits, where the output is a pure function of the present inputs only.



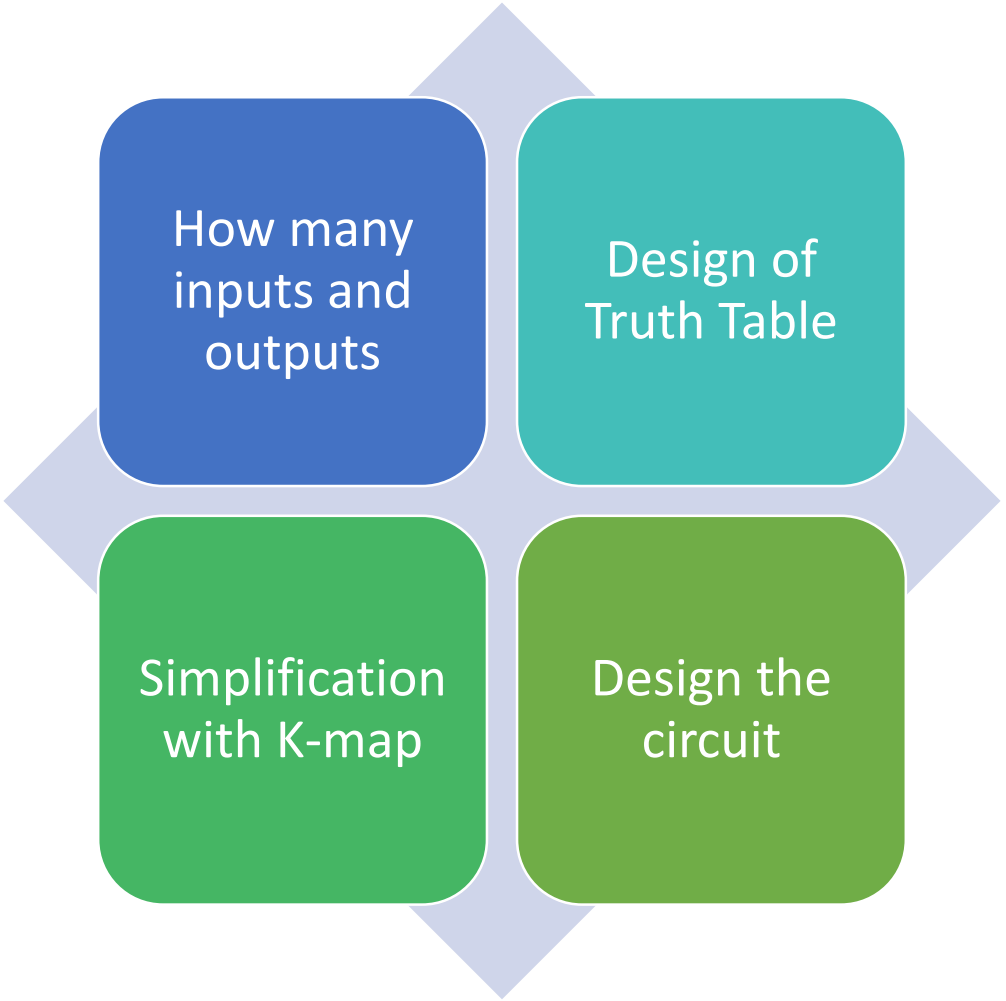

- The combinational circuit do not use any memory.
- The previous state of input does not have any effect on the present state of the circuit.

# Design Procedure

1. From the specifications of the circuit, determine the required number of inputs and outputs and assign a symbol to each.
2. Derive the truth table that defines the required relationship between inputs and outputs.

3. Obtain the simplified Boolean functions for each output as a function of the input variables.
4. Draw the logic diagram and verify the correctness of the design (manually or by simulation).

# Design Procedure:
# BCD to Excess-3 Code Conversion



# Design Procedure:
# BCD to Excess-3 Code Conversion

STEP: 1 and 2

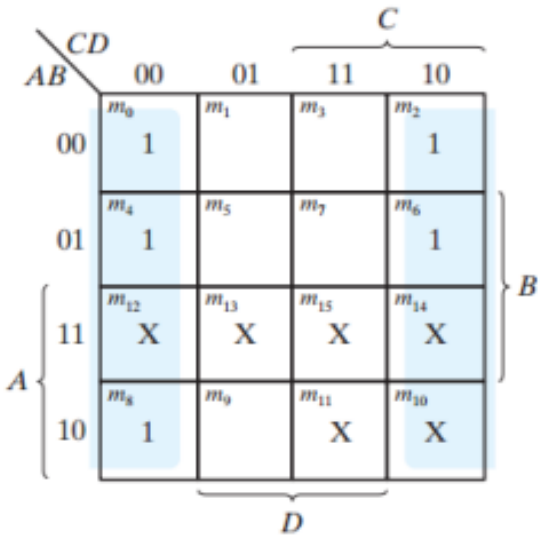
*Truth Table for Code Conversion Example*

| Input BCD | | | | Output Excess-3 Code | | | |
|---|---|---|---|---|---|---|---|
| *A* | *B* | *C* | *D* | *w* | *x* | *y* | *z* |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

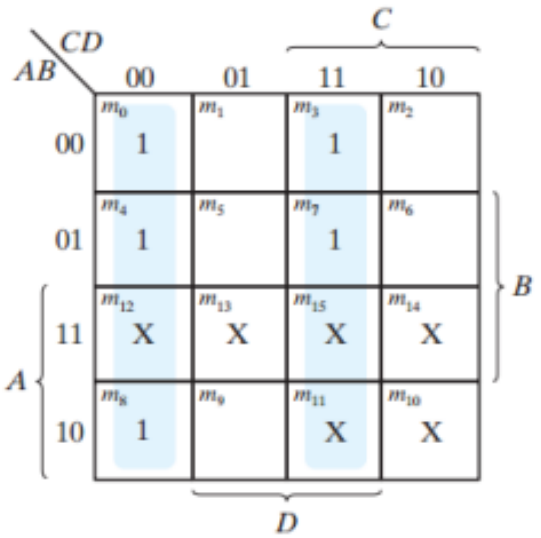
# Design Procedure:
# BCD to Excess-3 Code Conversion

# Design Procedure:
# BCD to Excess-3 Code Conversion

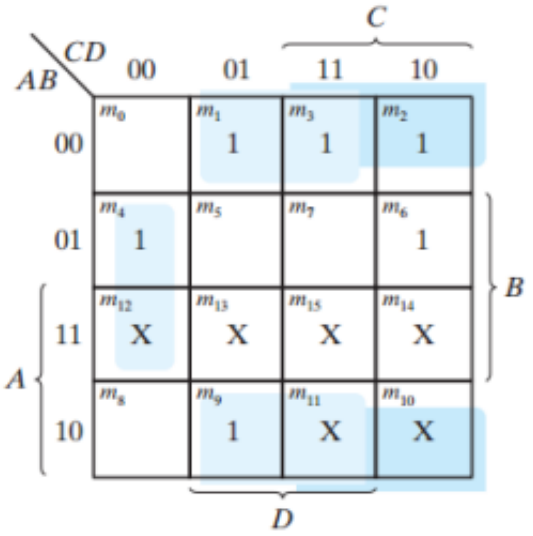
STEP: 3



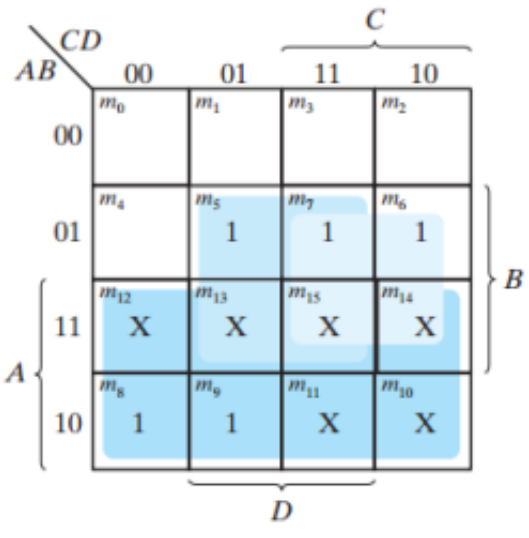

$z = D'$




$y = CD + C'D'$




$x = B'C + B'D + BC'D'$




$w = A + BC + BD$

# Design Procedure:
# BCD to Excess-3 Code Conversion

STEP: 3

$$z = D'$$

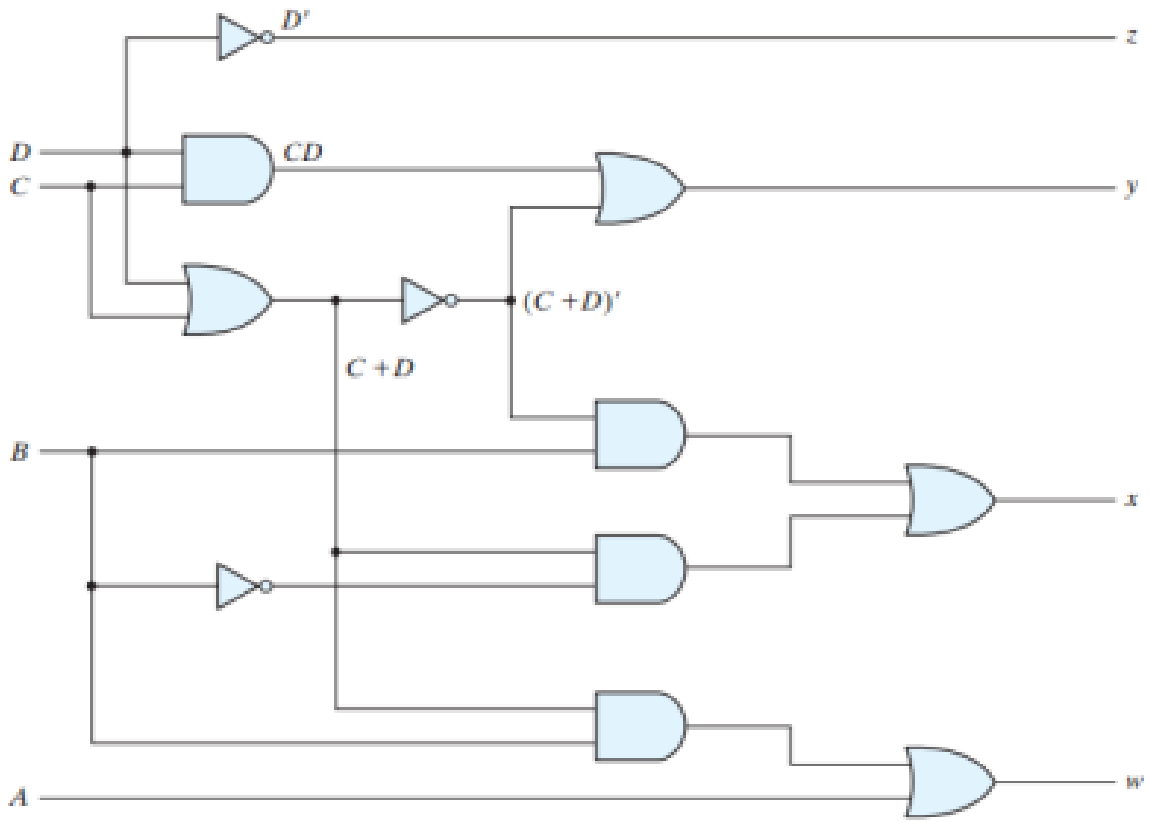
$$y = CD + C'D' = CD + (C + D)'$$

$$x = B'C + B'D + BC'D' = B'(C + D) + BC'D'$$
$$= B'(C + D) + B(C + D)'$$

$$w = A + BC + BD = A + B(C + D)$$

# Design Procedure:
# BCD to Excess-3 Code Conversion

STEP: 4




$$z = D'$$
$$y = CD + C'D' = CD + (C + D)'$$
$$x = B'C + B'D + BC'D' = B'(C + D) + BC'D'$$
$$= B'(C + D) + B(C + D)'$$
$$w = A + BC + BD = A + B(C + D)$$

# Half Adder

- A combinational logic circuit with two inputs and two outputs.
- The half adder circuit add two single bits without any carry.This circuit has two outputs carry and sum.

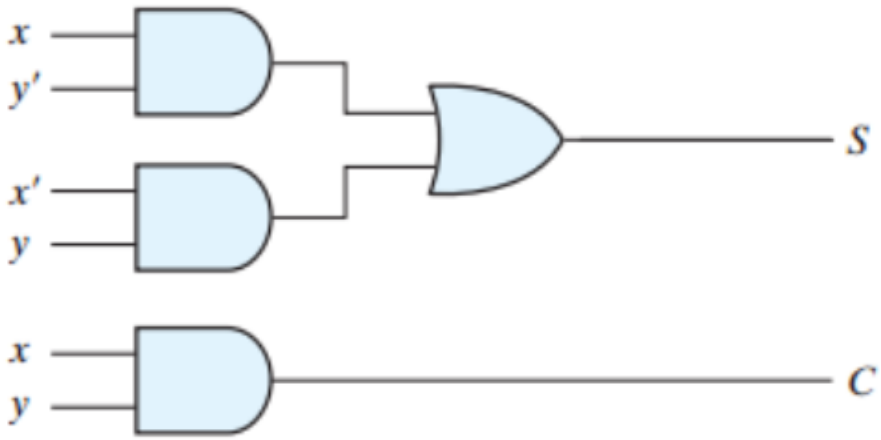

*Half Adder*

| x | y | C | S |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

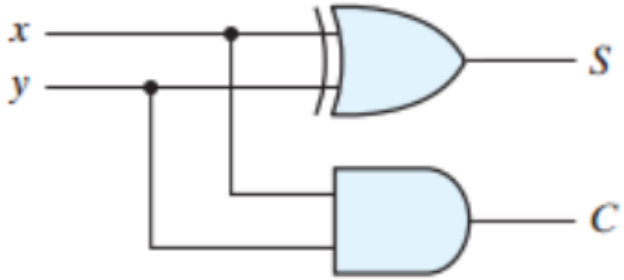
# Half Adder

*Half Adder*

| x | y | C | S |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |




(a) $S = xy' + x'y$
$C = xy$




(b) $S = x \oplus y$
$C = xy$

# QUICK QUIZ (POLL)

If A and B are the inputs of a half adder, the sum is given by _____

a) A AND B

b) A OR B

c) A XOR B

d) A EX-NOR B

# QUICK QUIZ (POLL)

Total number of inputs in a half adder is _____

a) 2

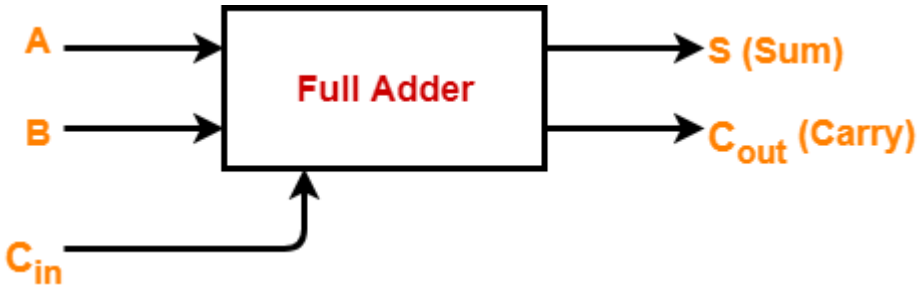
b) 3

c) 4

d) 1

# Full Adder

- A combinational logic circuit with 3 inputs and 2 outputs.
- The Full adder circuit adds 3 bits including carry.




| A | B | Cin | Sum | Cout |
|---|---|---|---|---|
| 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 |

$$\overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$$

$$\overline{A}(\overline{B}C + B\overline{C}) + A(\overline{B}\overline{C} + BC)$$

$$\overline{A}(B \oplus C) + A(B \Theta C)$$

$$Let\ B \oplus C = D$$

$$\overline{A}D + A\overline{D}$$

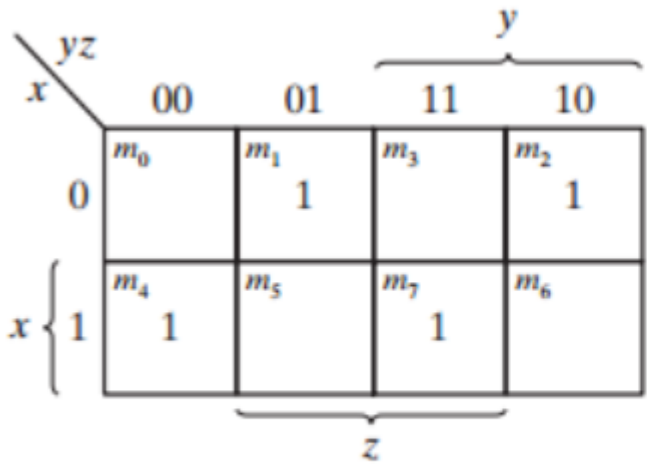
$$A \oplus D$$

$$A \oplus B \oplus C$$

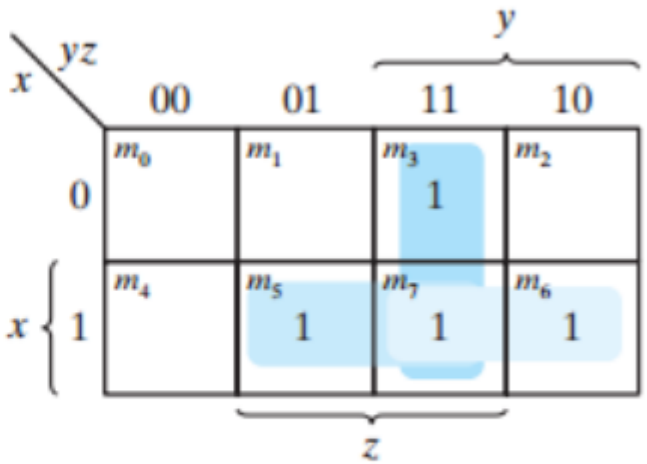
# Full Adder

## Full Adder

| 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 |

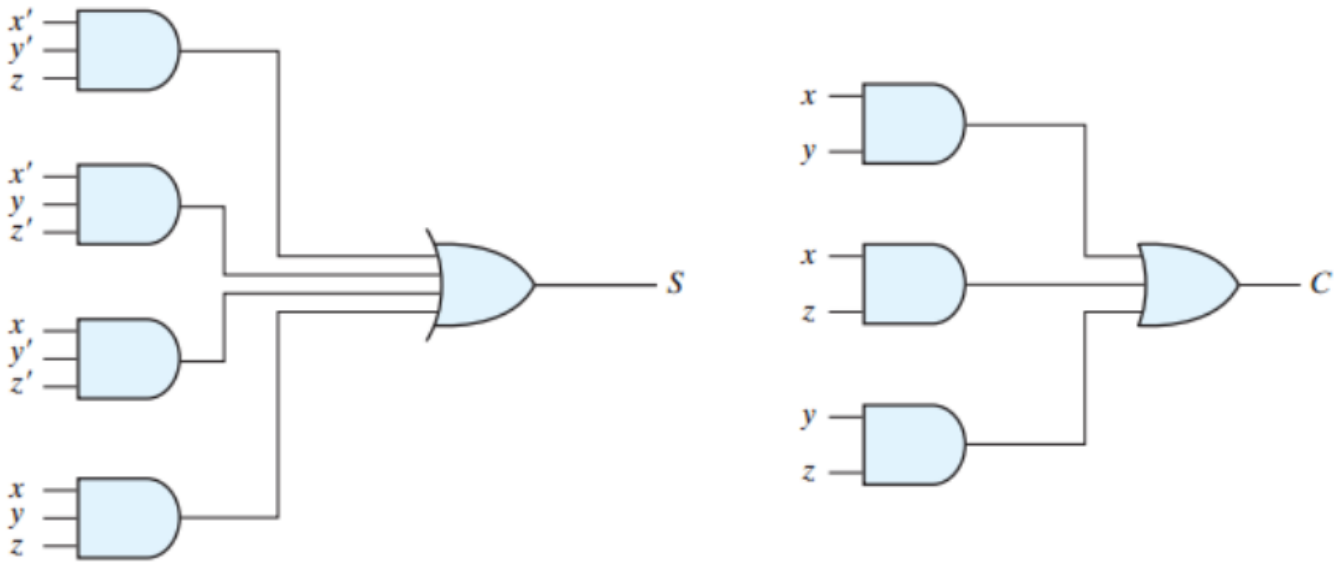



(a) $S = x'y'z + x'yz' + xy'z' + xyz$




(b) $C = xy + xz + yz$

# Full Adder



# QUICK QUIZ (POLL)

If A, B and C are the inputs of a full adder then the carry is given by _____

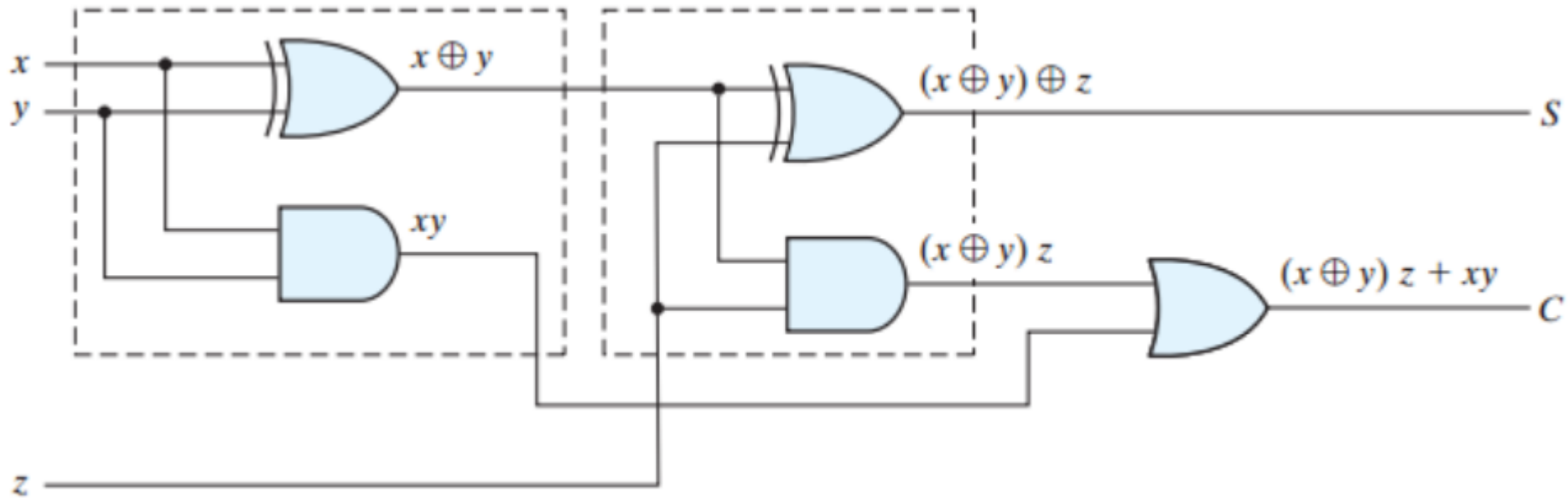
a) A AND B OR (A OR B) AND C

b) A OR B OR (A AND B) C

c) (A AND B) OR (A AND B)C

d) A XOR B XOR (A XOR B) AND C

# Implementation of Full Adder using Half Adders



# DIGITAL ELECTRONICS: ECE 213


**Topic: Adders, Subtractors, Decoders and Multiplexer**

**UNIT III: Introduction to Combinational Logic Circuits and Logic Families**

**Lecture No.: 16**

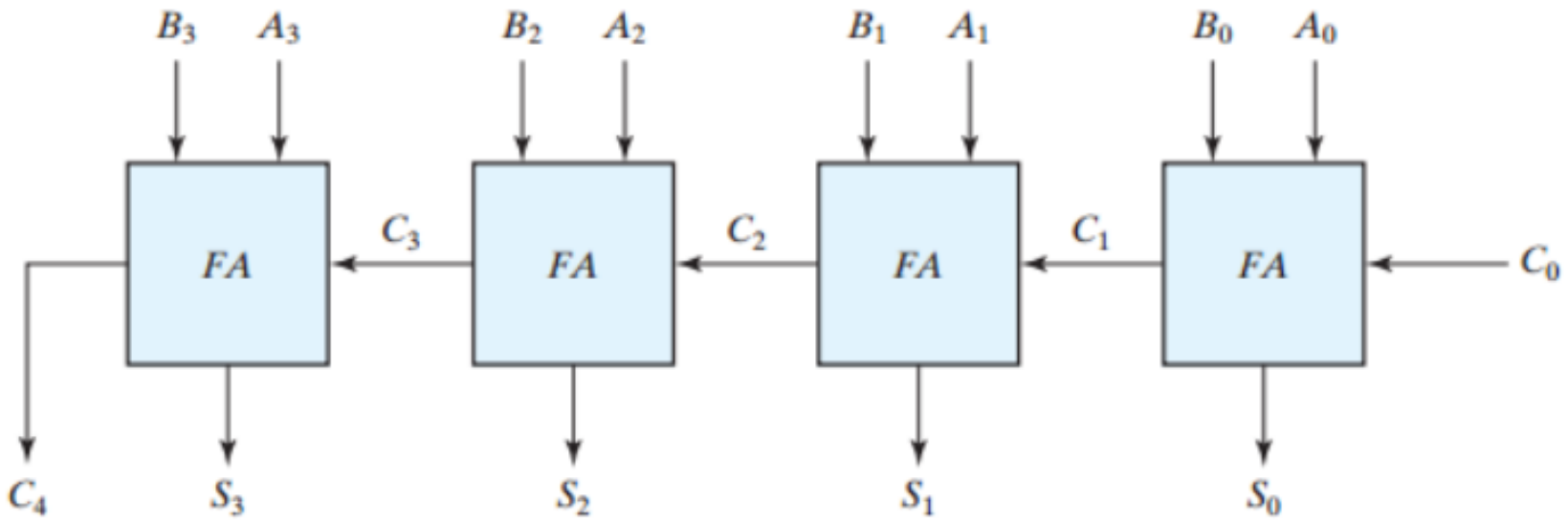
**Prepared By: Irfan Ahmad Pindoo**

**Assistant Professor**

**VLSI Design, ECE**

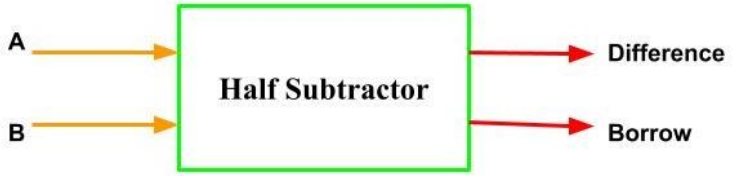
**School of Computer Science and Engineering**

# Implementation of Ripple Carry Adder (RCA):4-bits



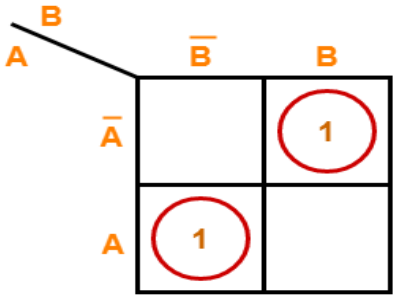
# Half Subtractor

- Combinational circuit perform binary subtraction
- accepts 2 inputs and provides two outputs: Difference and Borrow



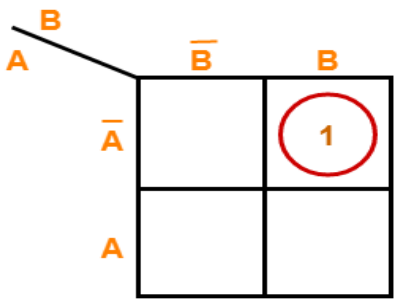

| Inputs | | Outputs | |
|---|---|---|---|
| A | B | D (Difference) | b (Borrow) |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

For D:



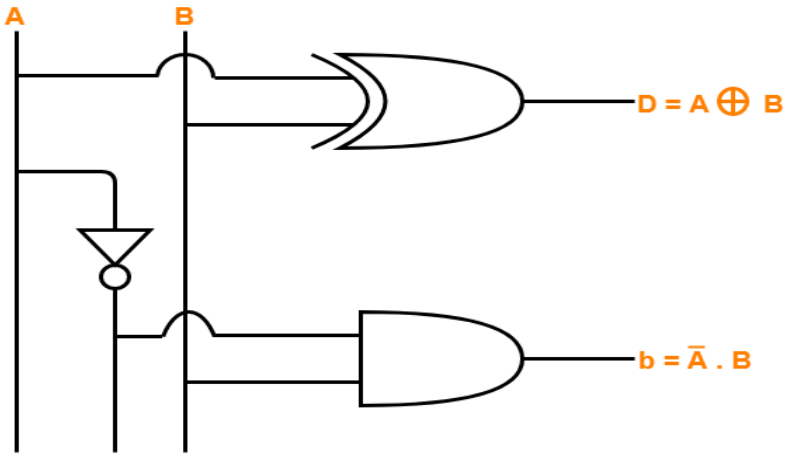

$D = A \oplus B$

For b:




$b = \bar{A}.B$

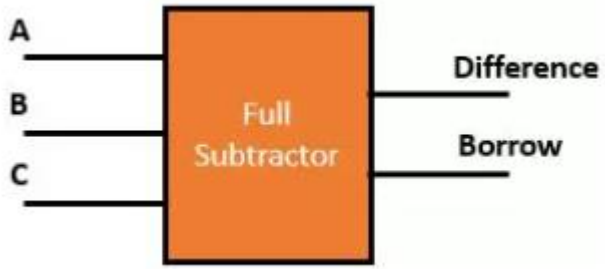
K Maps




Half Subtractor Logic Diagram

# Full Subtractor

- Performs subtraction of 3 bits
- This circuit has three inputs and two outputs.
- The three inputs A, B and C, denote the minuend, subtrahend, and previous borrow, respectively.
- The two outputs are: D and Bout



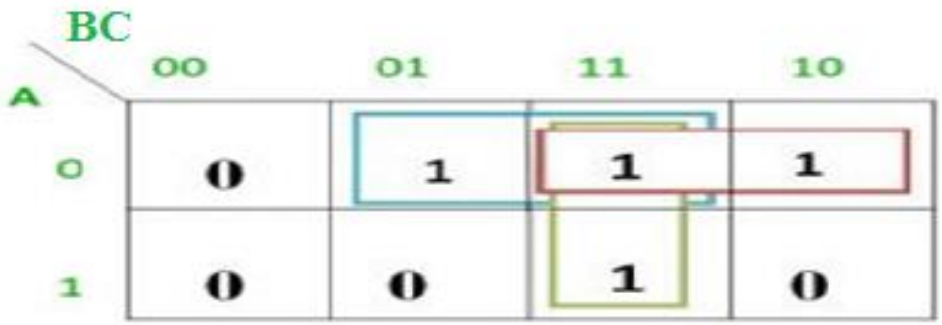

| Input | | | Output | |
|---|---|---|---|---|
| A | B | C | Difference | Borrow |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

$$
\begin{aligned}
Difference &= \overline{A}\ \overline{B}\ C + \overline{A}\ B\ \overline{C} + A\ \overline{B}\ \overline{C} + ABC \\
&= C\ (\overline{A}\ \overline{B} + AB) + \overline{C}\ (\overline{A}\ B + A\ \overline{B}) \\
&= C\ (A \odot B) + \overline{C}\ (A \oplus B) \\
&= C\ (A \overline{\oplus} B) + \overline{C}\ (A \oplus B) \\
&= C \oplus (A \oplus B)
\end{aligned}
$$

Sum(A, B,C) = ∑m (1, 2, 4, 7)

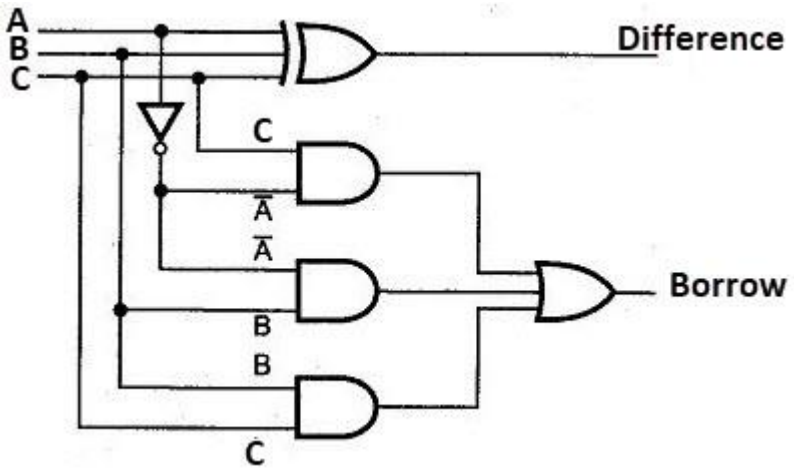
Bout(A, B,C) = ∑m (1,2,3,7)

# Full Subtractor

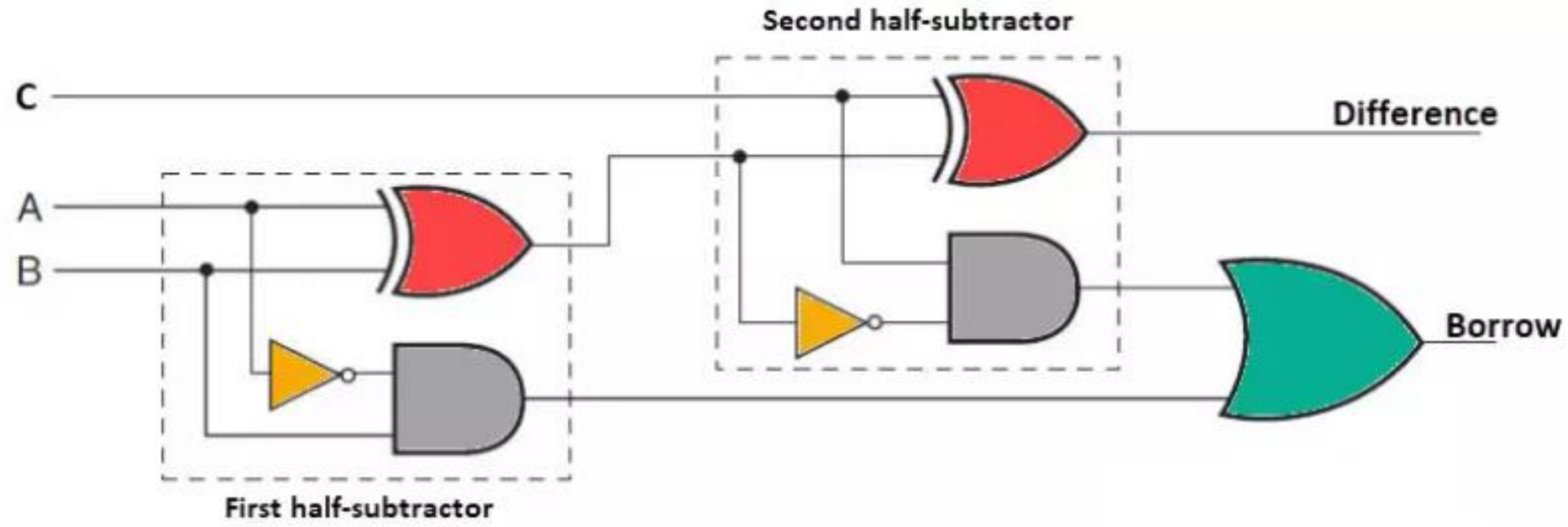



$$Borrow = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + \overline{A}\,BC + ABC$$
$$= \overline{A}\,B + \overline{A}\,C + BC$$

$$\begin{aligned} Bout &= A'B'C + A'BC' + A'BC + ABC \\ &= C(AB + A'B') + A'B(C + C') \\ &= C(\ A \text{ XNOR } B) + A'B \\ &= C\ (A \text{ XOR } B)' + A'B \end{aligned}$$



# Implementation of Full Subtractor using Half Subtractor



# QUICK QUIZ (POLL)

The difference between half adder and full adder is _____

a) Half adder has two inputs while full adder has four inputs

b) Half adder has one output while full adder has two outputs

c) Half adder has two inputs while full adder has three inputs

d) All of the Mentioned

# QUICK QUIZ (POLL)

How many AND, OR and EXOR gates are required for the configuration of full adder?

a) 1, 2, 2

b) 2, 1, 2

c) 3, 1, 2

d) 4, 0, 1

# Try Yourself

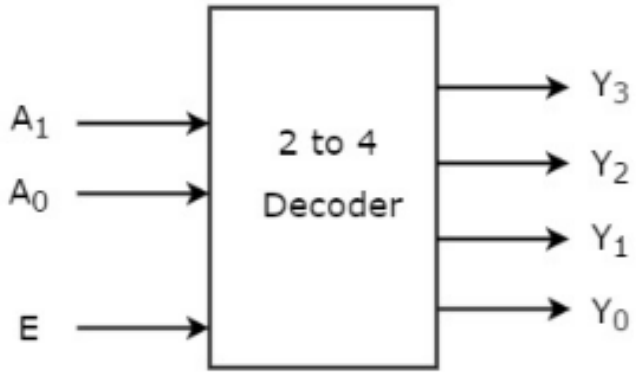
**Design a 3 input digital circuit which produces high output for ODD decimal equivalent?**

# Decoder

## What is a Decoder?

- A combinational circuit that has ‘n’ input lines and maximum of $2^n$ output lines.
- One of these outputs will be active High based on the combination of inputs present, when the decoder is enabled.
- The outputs of the decoder are min terms of ‘n’ input variables lines when it is enabled

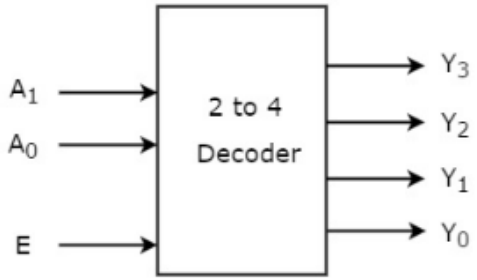
# 2x4 Decoder




| Enable | Inputs | | Outputs | | | |
|---|---|---|---|---|---|---|
| E | $A_1$ | $A_0$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
| 0 | x | x | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |

# 2x4 Decoder

| Enable | Inputs | | Outputs | | | |
|---|---|---|---|---|---|---|
| E | $A_1$ | $A_0$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
| 0 | x | x | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |



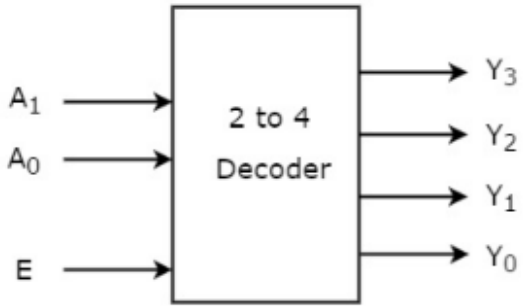

$$Y_3 = E.A_1.A_0$$

$$Y_2 = E.A_1.A_0'$$

$$Y_1 = E.A_1'.A_0$$

$$Y_0 = E.A_1'.A_0'$$

# 2x4 Decoder

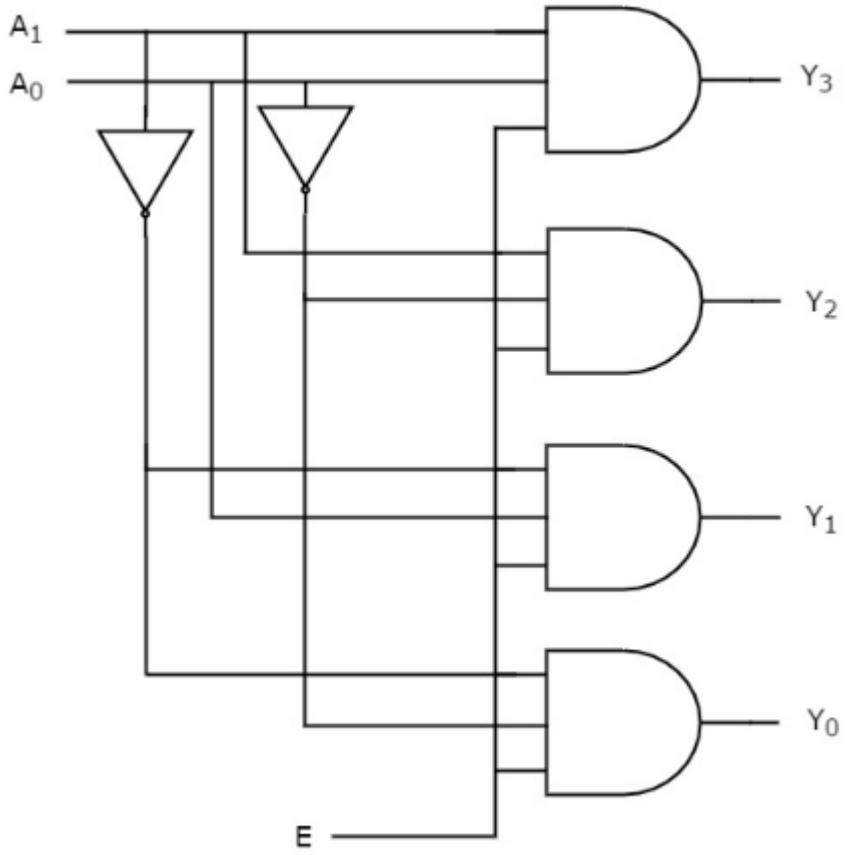



$$Y_3 = E.A_1.A_0$$

$$Y_2 = E.A_1.A_0'$$

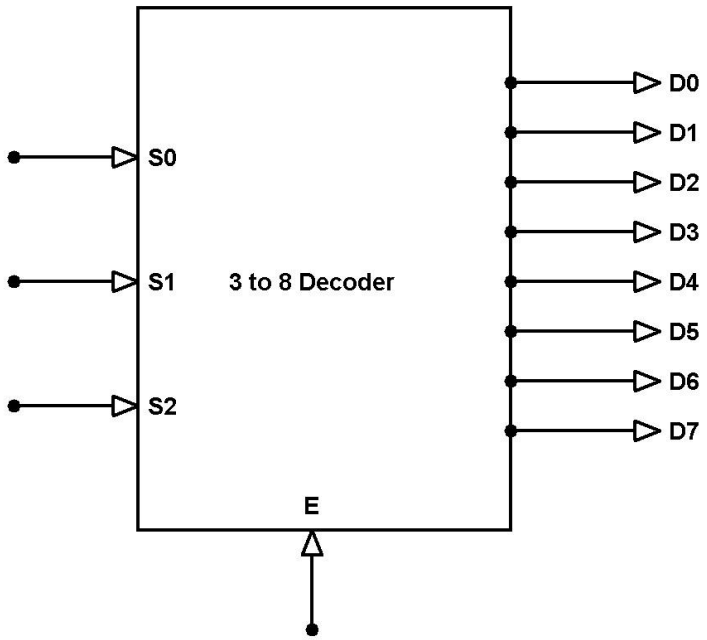
$$Y_1 = E.A_1'.A_0$$

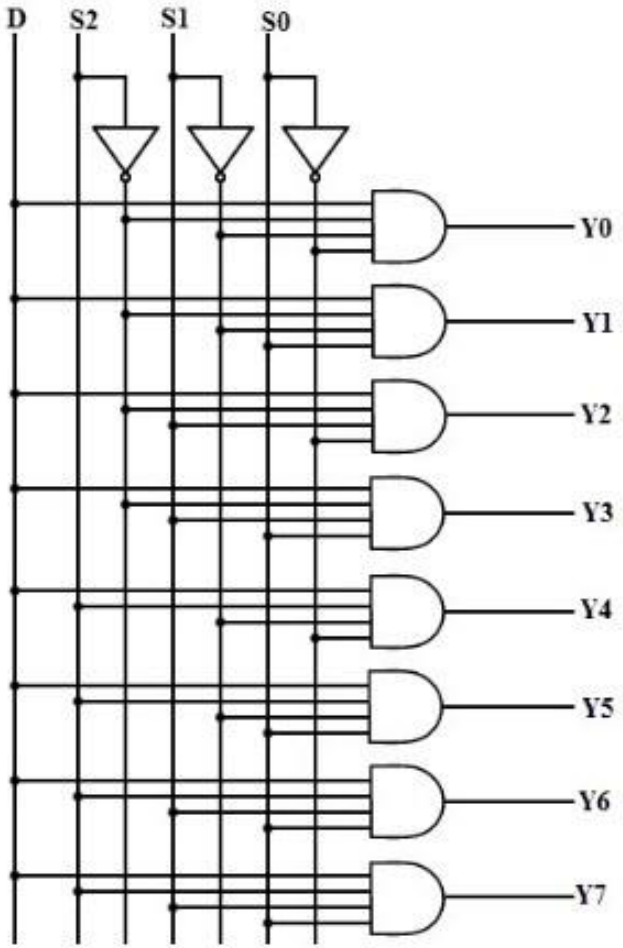
$$Y_0 = E.A_1'.A_0'$$




Therefore, the outputs of 2 to 4 decoder are nothing but the **min terms** of two input variables $A_1$ & $A_0$, when enable, E is equal to one. If enable, E is zero, then all the outputs of decoder will be equal to zero.

# 3x8 Decoder

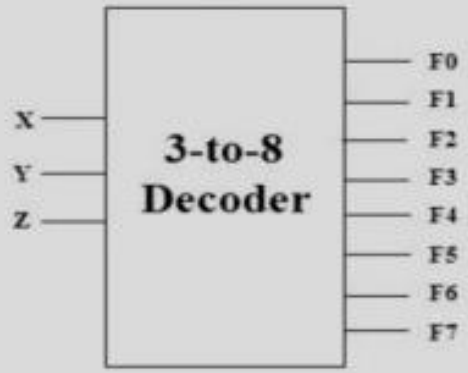

S0
S1
S2
3 to 8 Decoder
E
D0
D1
D2
D3
D4
D5
D6
D7

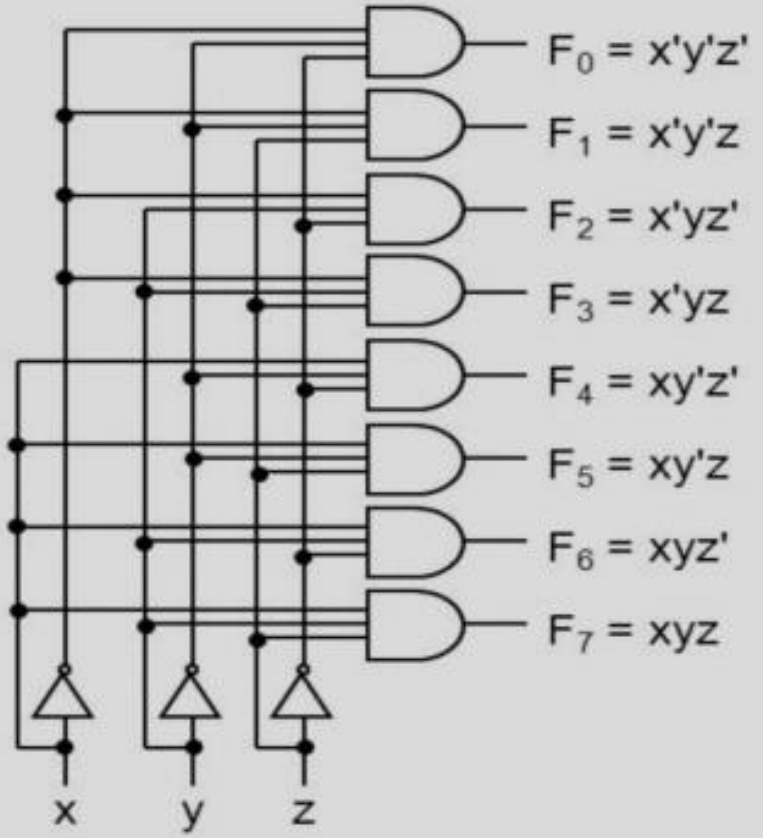

D
S2
S1
S0
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7

# 3x8 Decoder: Active High

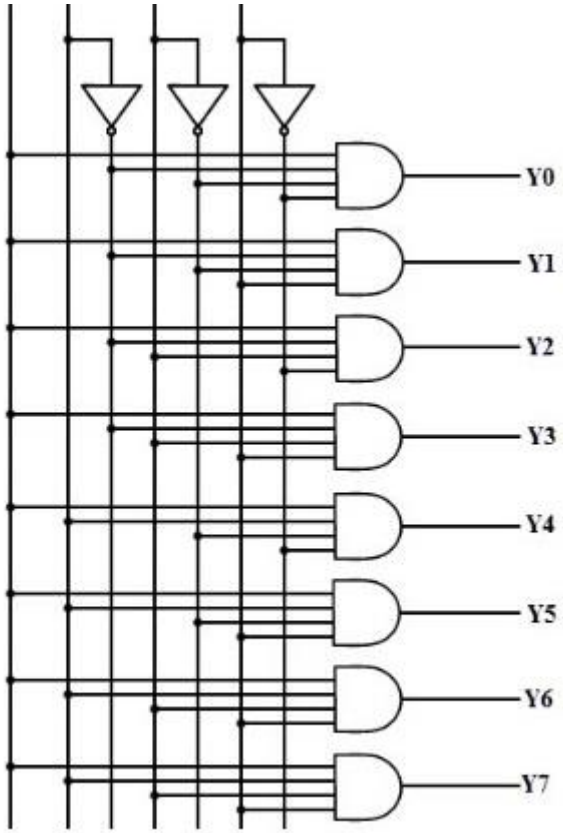
**Truth Table:**

| x | y | z | $F_0$ | $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ | $F_7$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |



# 3x8 Decoder: Active Low




| Inputs | | | Outputs | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| A | B | C | $Y_0$ | $Y_1$ | $Y_2$ | $Y_3$ | $Y_4$ | $Y_5$ | $Y_6$ | $Y_7$ |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

# Full Adder Design using 3:8 Decoder

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 |

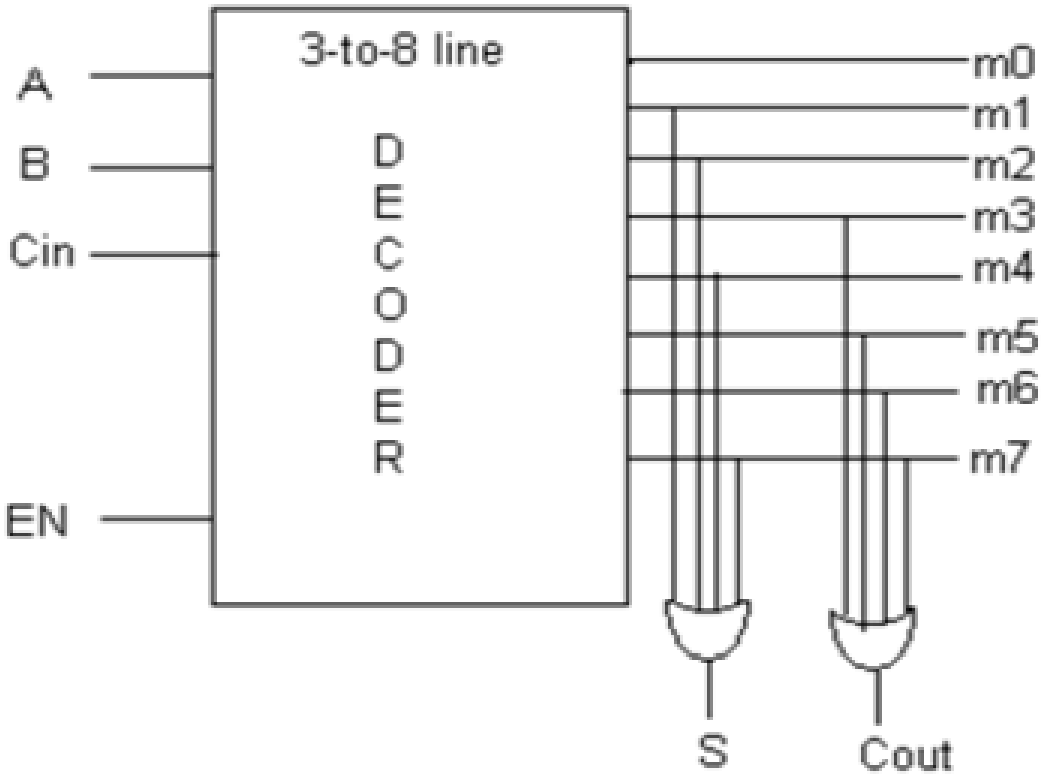
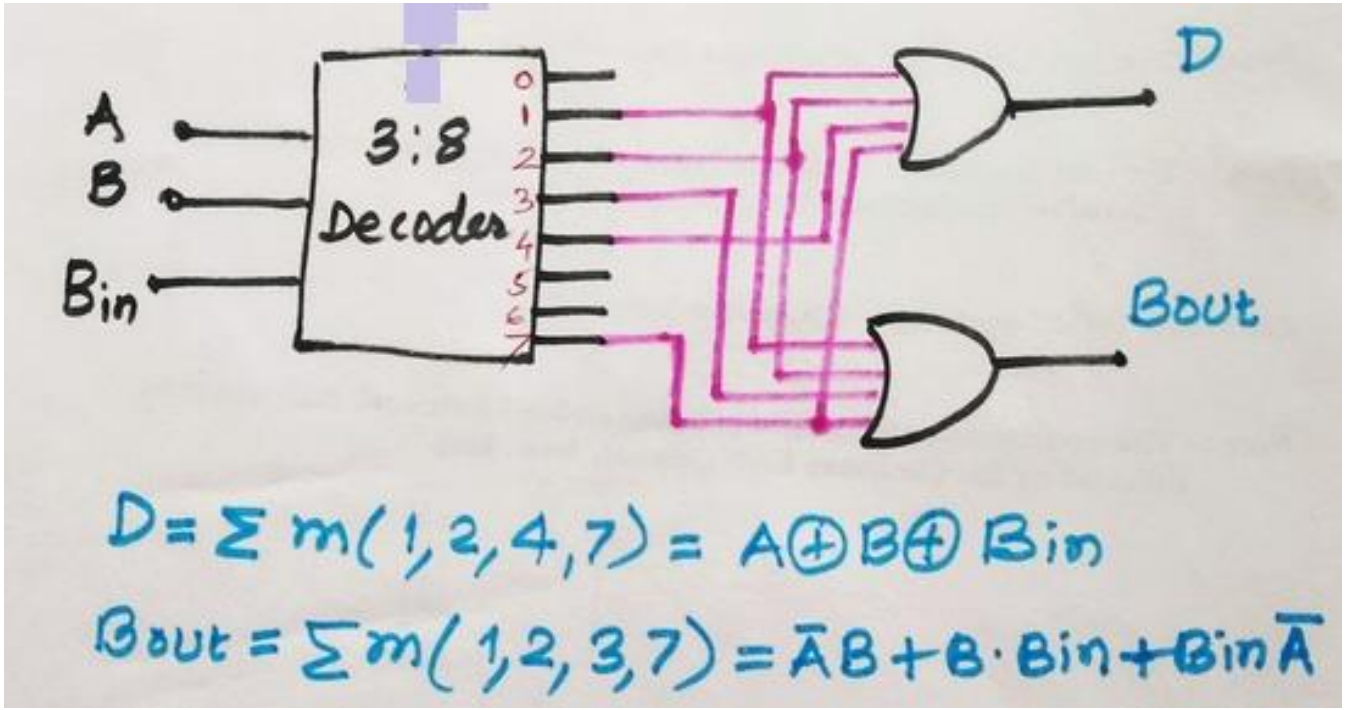



Fig1: Full Adder Implementation using 3:8 decoder

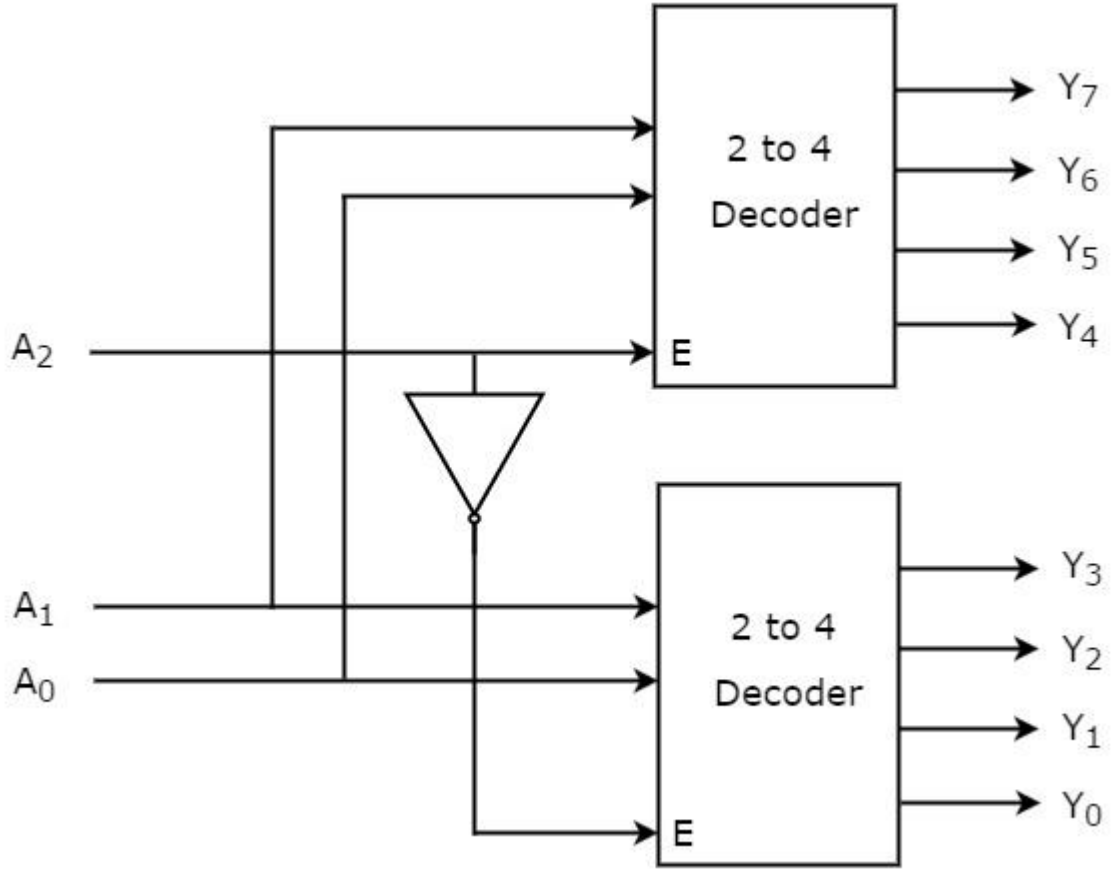
# Full Subtractor Design using 3:8 Decoder

| Inputs | | | Outputs | |
|---|---|---|---|---|
| A | B | $B_{in}$ | D | $B_{out}$ |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |



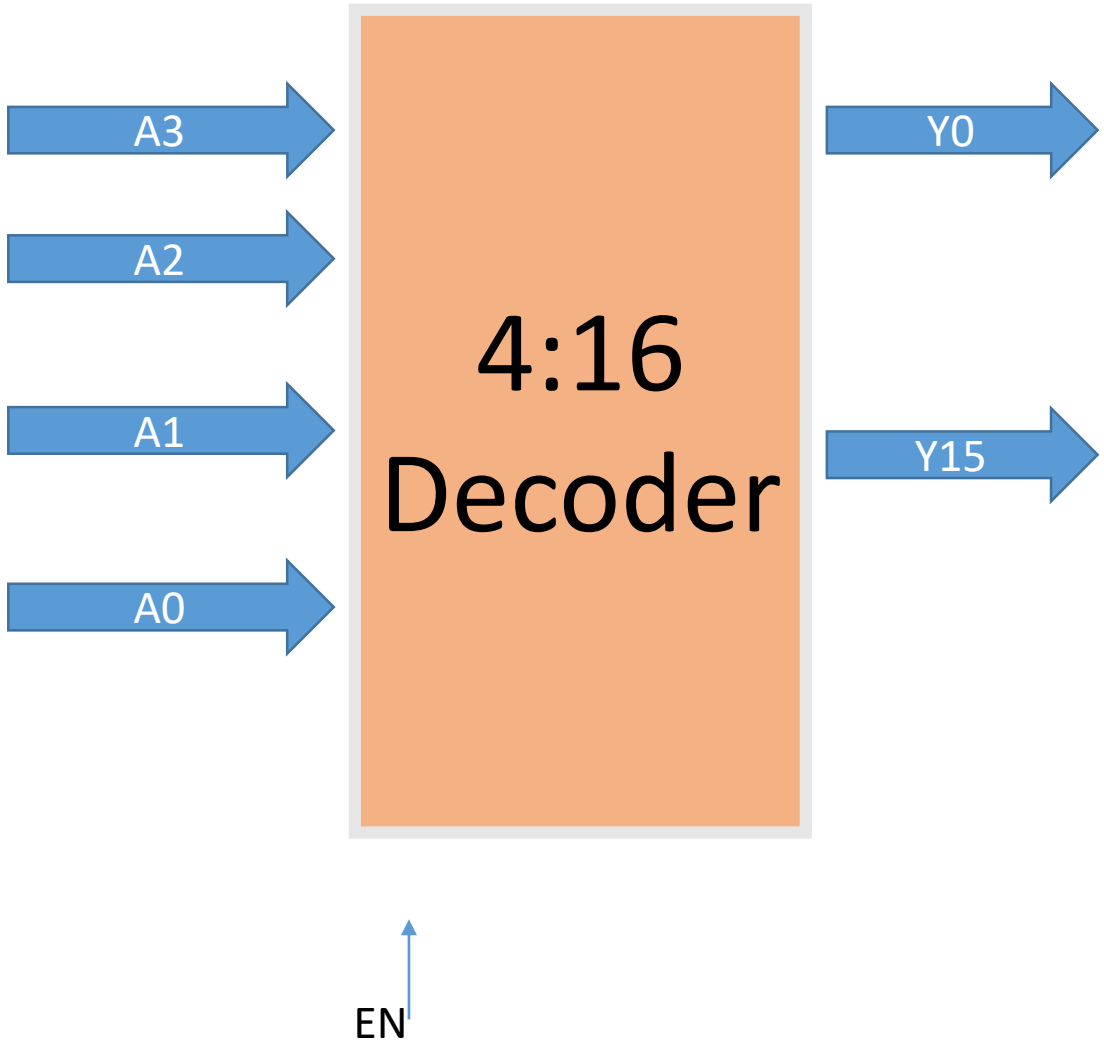
# 3:8 Decoder using 2:4 Decoder

| A2 | A1 | A0 | HIGH OUTPUT |
|---|---|---|---|
| 0 | 0 | 0 | Y0 |
| 0 | 0 | 1 | Y1 |
| 0 | 1 | 0 | Y2 |
| 0 | 1 | 1 | Y3 |
| 1 | 0 | 0 | Y4 |
| 1 | 0 | 1 | Y5 |
| 1 | 1 | 0 | Y6 |
| 1 | 1 | 1 | Y7 |



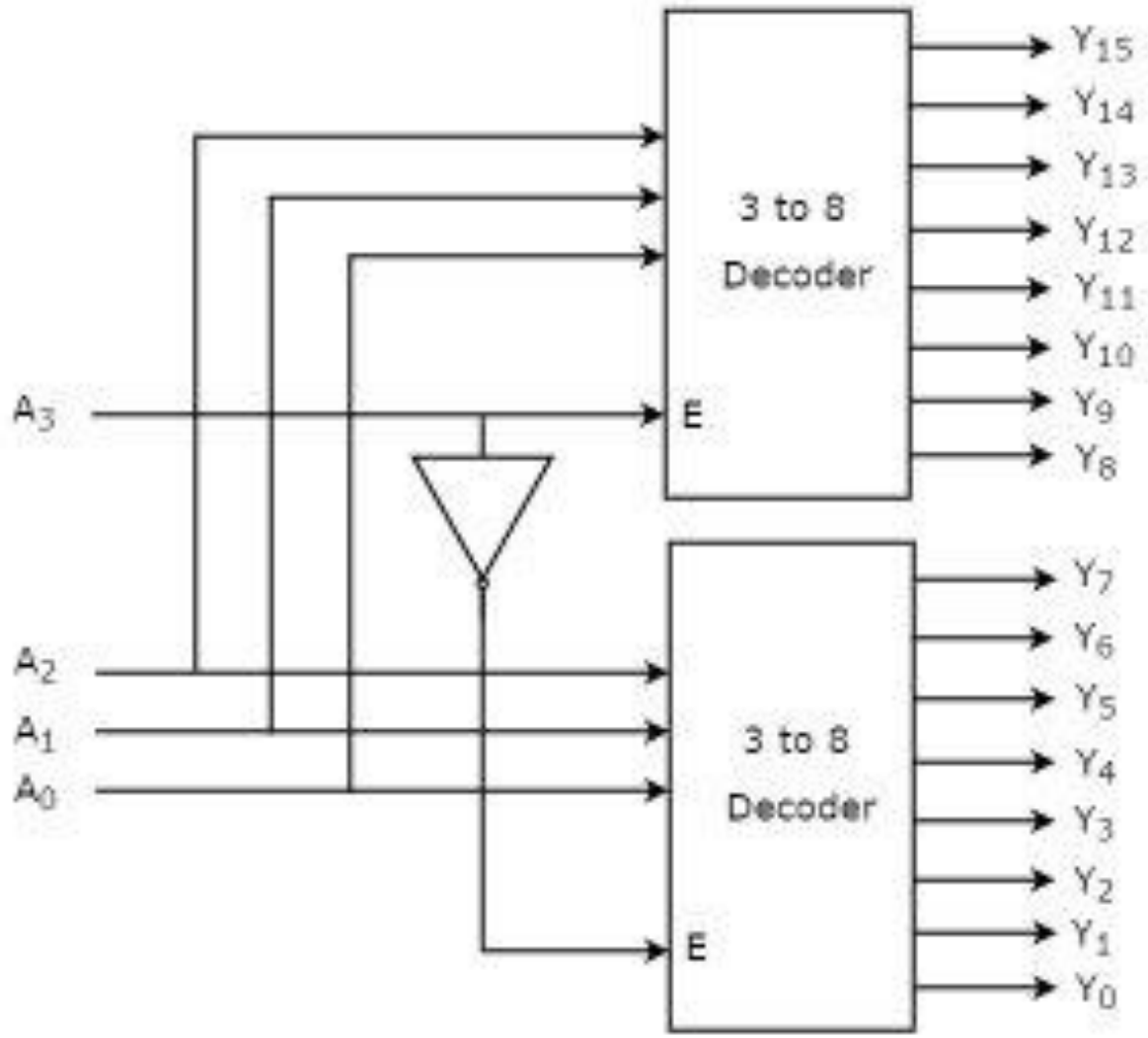
# 4:16 Decoder

| A3 | A2 | A1 | A0 | HIGH OUTPUT |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Y0 |
| 0 | 0 | 0 | 1 | Y1 |
| 0 | 0 | 1 | 0 | Y2 |
| 0 | 0 | 1 | 1 | Y3 |
| 0 | 1 | 0 | 0 | Y4 |
| 0 | 1 | 0 | 1 | Y5 |
| 0 | 1 | 1 | 0 | Y6 |
| 0 | 1 | 1 | 1 | Y7 |
| 1 | 0 | 0 | 0 | Y8 |
| 1 | 0 | 0 | 1 | Y9 |
| 1 | 0 | 1 | 0 | Y10 |
| 1 | 0 | 1 | 1 | Y11 |
| 1 | 1 | 0 | 0 | Y12 |
| 1 | 1 | 0 | 1 | Y13 |
| 1 | 1 | 1 | 0 | Y14 |
| 1 | 1 | 1 | 1 | Y15 |



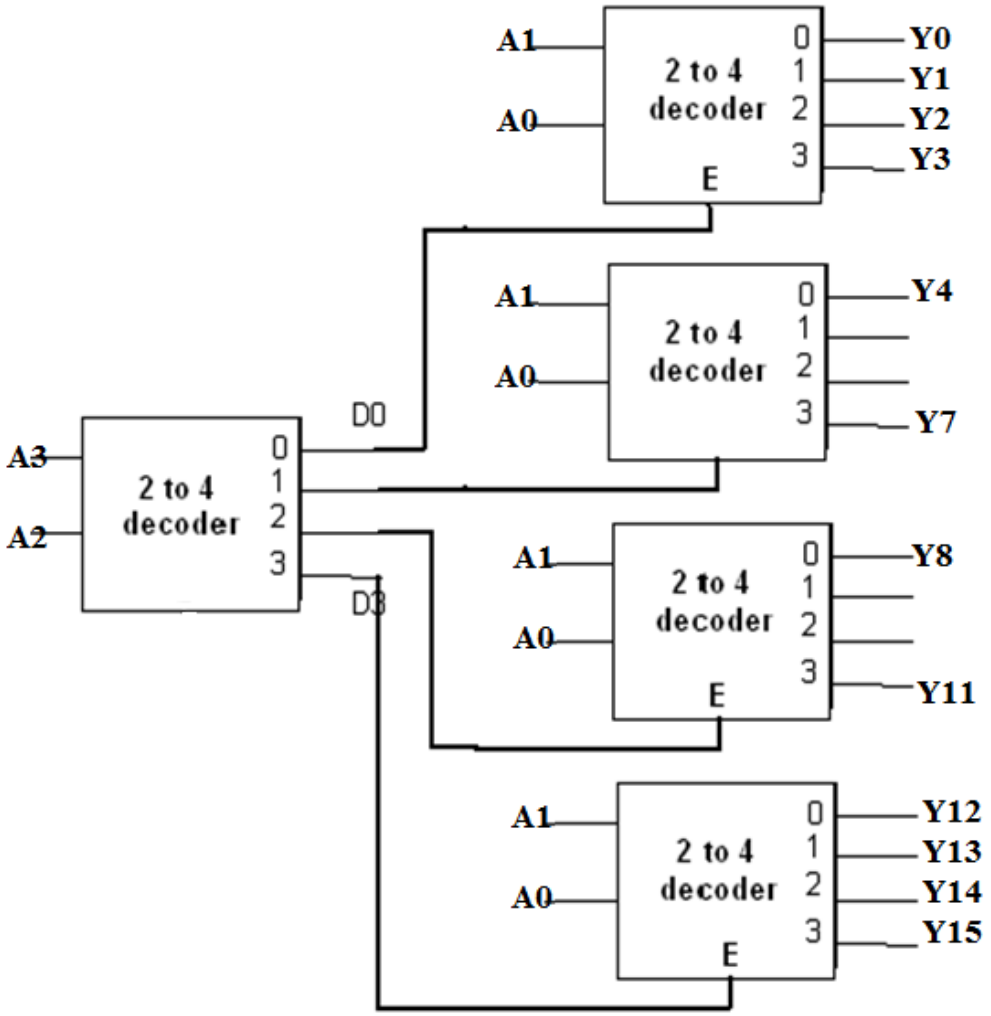
# 4:16 Decoder using 3:8 Decoder

| A3 | A2 | A1 | A0 | HIGH OUTPUT |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Y0 |
| 0 | 0 | 0 | 1 | Y1 |
| 0 | 0 | 1 | 0 | Y2 |
| 0 | 0 | 1 | 1 | Y3 |
| 0 | 1 | 0 | 0 | Y4 |
| 0 | 1 | 0 | 1 | Y5 |
| 0 | 1 | 1 | 0 | Y6 |
| 0 | 1 | 1 | 1 | Y7 |
| 1 | 0 | 0 | 0 | Y8 |
| 1 | 0 | 0 | 1 | Y9 |
| 1 | 0 | 1 | 0 | Y10 |
| 1 | 0 | 1 | 1 | Y11 |
| 1 | 1 | 0 | 0 | Y12 |
| 1 | 1 | 0 | 1 | Y13 |
| 1 | 1 | 1 | 0 | Y14 |
| 1 | 1 | 1 | 1 | Y15 |



# 4:16 Decoder using 2:4 Decoder

| A3 | A2 | A1 | A0 | HIGH OUTPUT |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Y0 |
| 0 | 0 | 0 | 1 | Y1 |
| 0 | 0 | 1 | 0 | Y2 |
| 0 | 0 | 1 | 1 | Y3 |
| 0 | 1 | 0 | 0 | Y4 |
| 0 | 1 | 0 | 1 | Y5 |
| 0 | 1 | 1 | 0 | Y6 |
| 0 | 1 | 1 | 1 | Y7 |
| 1 | 0 | 0 | 0 | Y8 |
| 1 | 0 | 0 | 1 | Y9 |
| 1 | 0 | 1 | 0 | Y10 |
| 1 | 0 | 1 | 1 | Y11 |
| 1 | 1 | 0 | 0 | Y12 |
| 1 | 1 | 0 | 1 | Y13 |
| 1 | 1 | 1 | 0 | Y14 |
| 1 | 1 | 1 | 1 | Y15 |



# DIGITAL ELECTRONICS: ECE 213


**Topic: Adders, Subtractors, Decoders and Multiplexer**

**UNIT III: Introduction to Combinational Logic Circuits and Logic Families**

**Lecture No.: 17**

**Prepared By: Irfan Ahmad Pindoo**

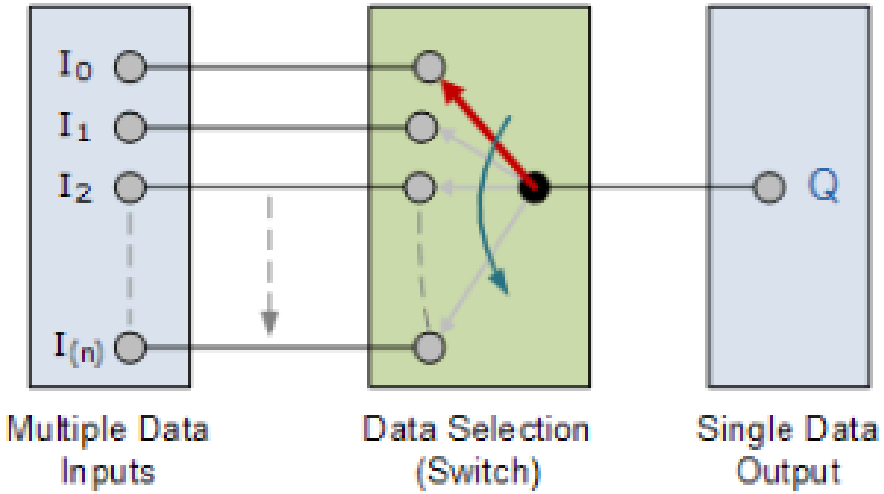
**Assistant Professor**

**VLSI Design, ECE**

**School of Computer Science and Engineering**

# Multiplexer

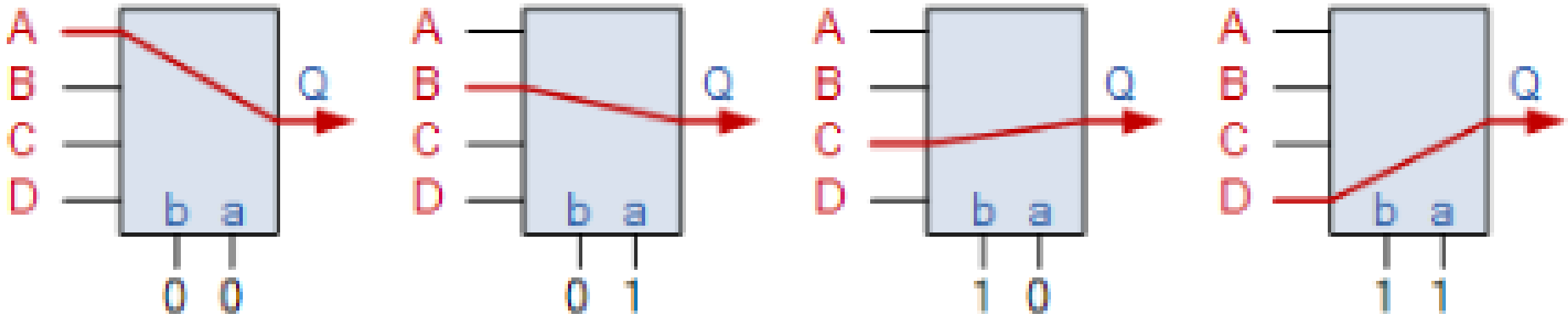
- The *multiplexer*, shortened to "MUX" or is a combinational logic circuit designed to switch one of several input lines through to a single common output line by the application of a control signal.
- Multiplexers operate like very fast acting multiple position rotary switches connecting or controlling multiple input lines called "channels" one at a time to the output.



# Multiplexer

- **Multiplexer** is a combinational circuit that has maximum of $2^n$ data inputs, ‘n’ selection lines and single output line. One of these data inputs will be connected to the output based on the values of selection lines.

- Since there are ‘n’ selection lines, there will be $2^n$ possible combinations of zeros and ones. So, each combination will select only one data input. Multiplexer is also called as **Mux**.



# QUICK QUIZ (POLL)

Which combinational circuit is renowned for selecting a single input from multiple inputs & directing the binary information to output line?

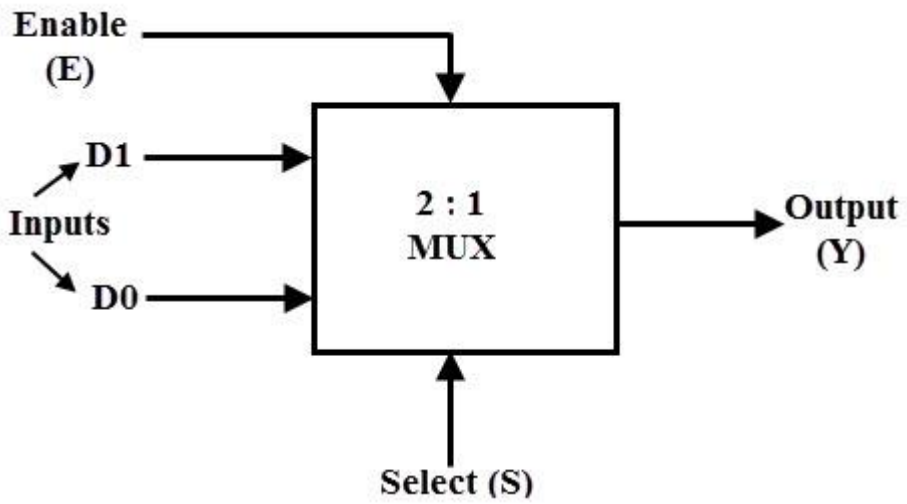
a) Data Selector

b) Data distributor

c) Both data selector and data distributor

d) DeMultiplexer

# 2:1 Multiplexer




$$Y = D_0 \bar{S} + D_1 S$$

| Select | Inputs | | Output |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

# QUICK QUIZ (POLL)

What is the function of an enable input on a multiplexer chip?

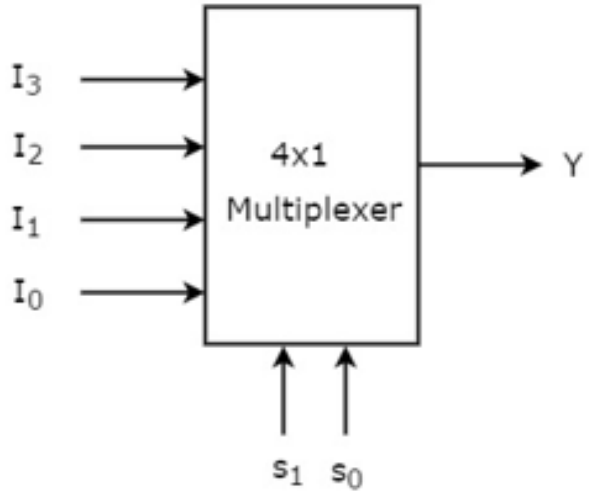
a) To apply Vcc

b) To connect ground

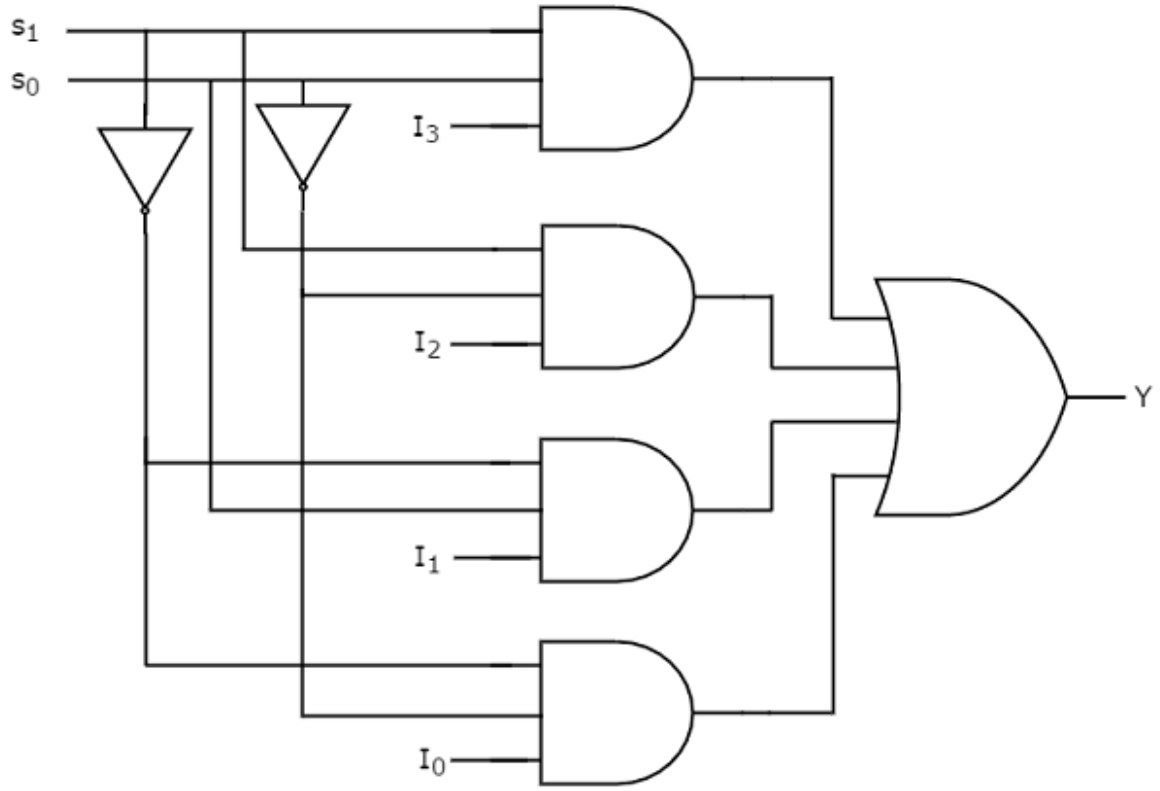
c) To active the entire chip

d) To active one half of the chip

# 4:1 Multiplexer




$$Y = S_1'S_0'I_0 + S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3$$

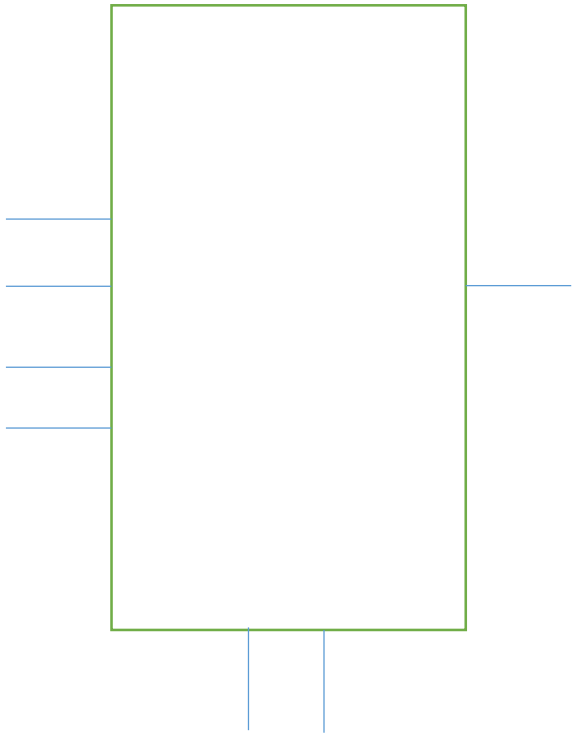



| Selection Lines | | Output |
|---|---|---|
| **S1** | **S0** | **Y** |
| 0 | 0 | I0 |
| 0 | 1 | I1 |
| 1 | 0 | I2 |
| 1 | 1 | I3 |

# Boolean Function Implementation using Multiplexer

**Implement f =$\sum(1, 3, 5, 6)$ using 4:1 MUX?**

| A | B | C | f |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |



# Boolean Function Implementation using Multiplexer

**TRY YOURSELF**

**Implement f =$\sum(0, 2, 4, 5, 6)$ using 4:1 MUX?**

# Full Adder Implementation using Multiplexer

**TRY YOURSELF**

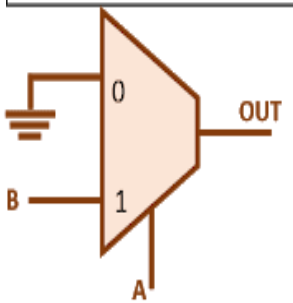
# Logic Gates Implementation using Multiplexer

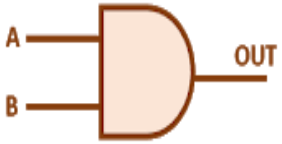
## AND GATE

| A | B | OUT |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

OUT = 0 when A = 0

OUT = B when A = 1



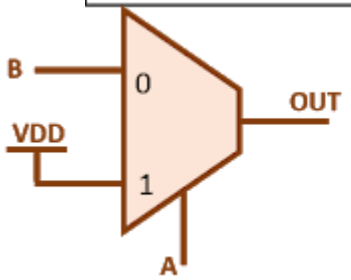



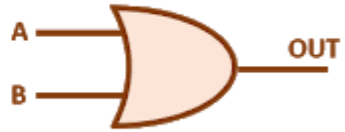

## OR GATE

| A | B | OUT |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

OUT = B when A = 0

OUT = 1 when A = 1



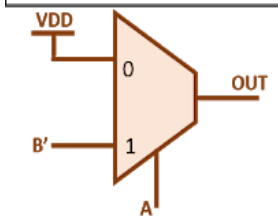



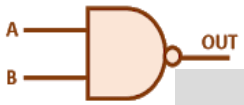

## NAND GATE

| A | B | OUT |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

OUT = 1 when A = 0

OUT = B' when A = 1



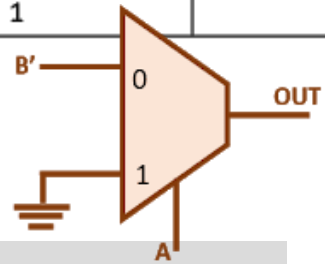



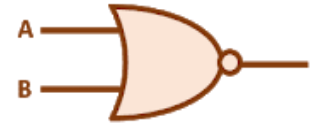

## NOR GATE

| A | B | OUT |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

OUT = B' when A = 0

OUT = 0 when A = 1



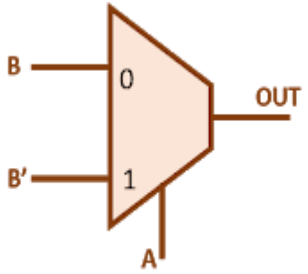
# Logic Gates Implementation using Multiplexer

## XOR GATE

| A | B | OUT |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

OUT = B when A = 0

OUT = B' when A = 1



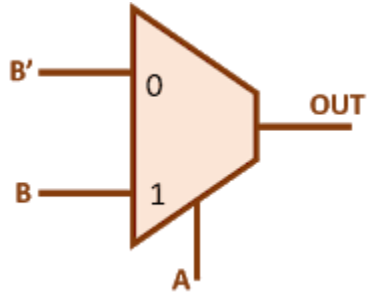


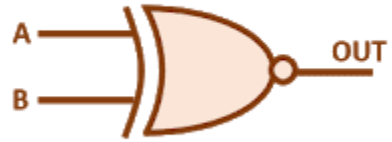
## XNOR GATE

| A | B | OUT |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

OUT = B' when A = 0

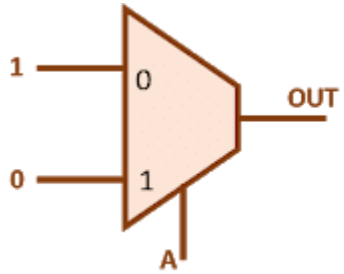
OUT = B when A = 1

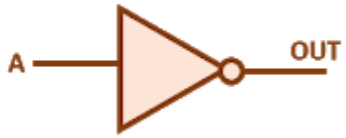



## NOT GATE

| A | OUT |
|---|---|
| 0 | 1 |
| 1 | 0 |



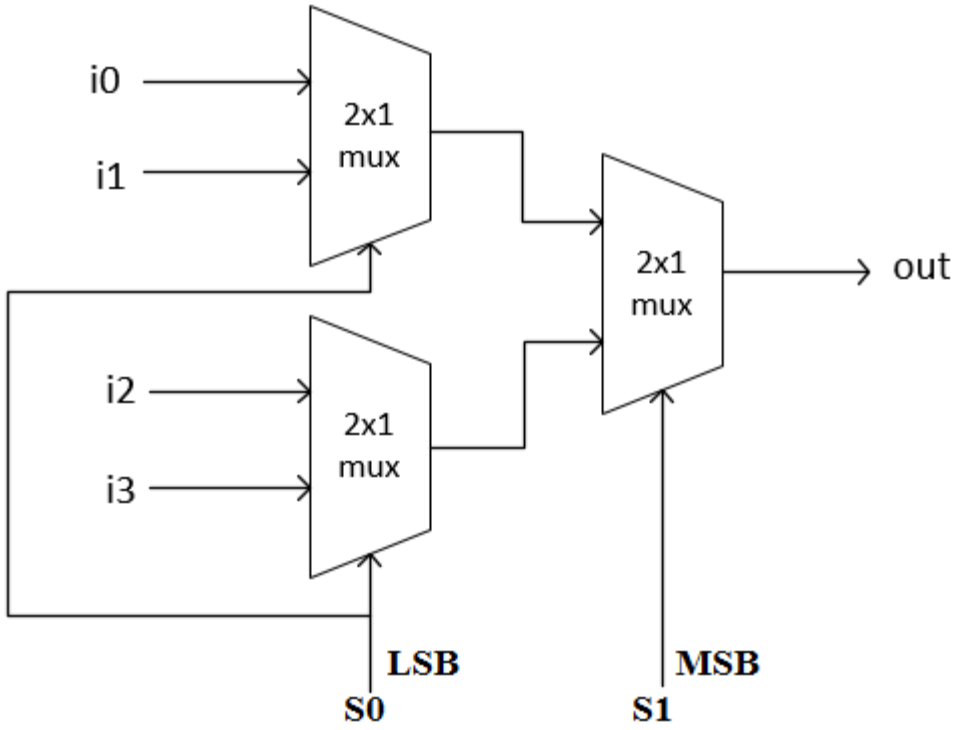



# Multiplexer Trees

Multilplexers with higher number of inputs can be implemented by cascading two or more multiplexers with less number of inputs.

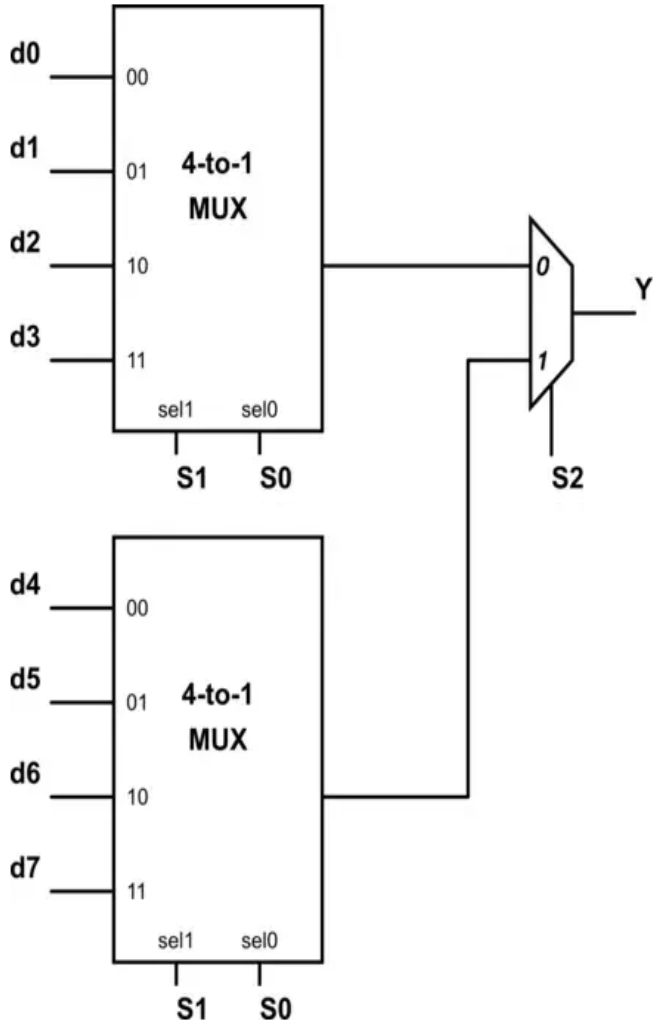
# 4:1 MUX using 2:1 MUX

| S1 (MSB) | S0(LSB) | Out |
|---|---|---|
| 0 | 0 | I0 |
| 0 | 1 | I1 |
| 1 | 0 | 12 |
| 1 | 1 | I3 |



# 8:1 MUX using 4:1 MUX

| Select Data Inputs | | | Output |
|---|---|---|---|
| $S_2$ | $S_1$ | $S_0$ | Y |
| 0 | 0 | 0 | $D_0$ |
| 0 | 0 | 1 | $D_1$ |
| 0 | 1 | 0 | $D_2$ |
| 0 | 1 | 1 | $D_3$ |
| 1 | 0 | 0 | $D_4$ |
| 1 | 0 | 1 | $D_5$ |
| 1 | 1 | 0 | $D_6$ |
| 1 | 1 | 1 | $D_7$ |



# 8:1 MUX: Example

**Question: Implement the Boolean function $f(A, B, C, D) = \sum(1, 3, 4, 11, 12, 13, 14, 15)$ using Multiplexer?**

# 8:1 MUX: Example

**Question: Implement the Boolean function $f(A, B, C, D) = \sum(1, 3, 4, 11, 12, 13, 14, 15)$ using Multiplexer?**

# DIGITAL ELECTRONICS: ECE 213


**Topic:** Demultiplexers and Encoders

**UNIT III:** Introduction to Combinational Logic Circuits and Logic Families

**Lecture No.: 18**

Prepared By: Irfan Ahmad Pindoo

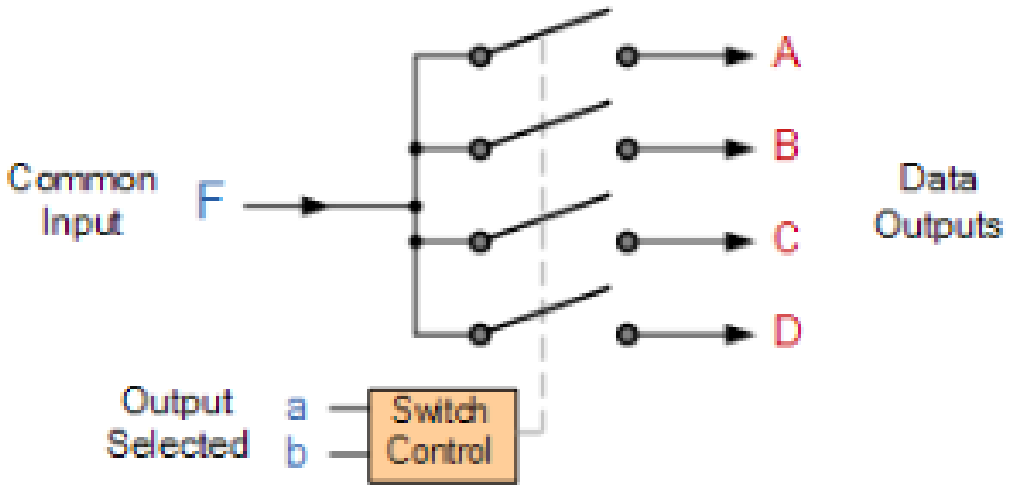
Assistant Professor

VLSI Design, ECE

School of Computer Science and Engineering

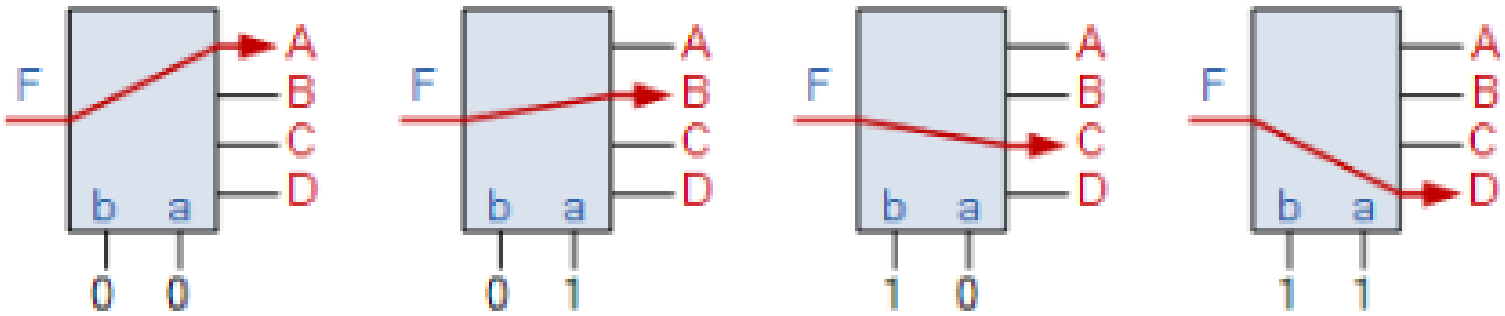
# De-Multiplexer

- De-Multiplexer is a combinational circuit that performs the reverse operation of Multiplexer.
- It has single input, ‘n’ selection lines and maximum of $2^n$ outputs.
- The input will be connected to one of these outputs based on the values of selection lines.

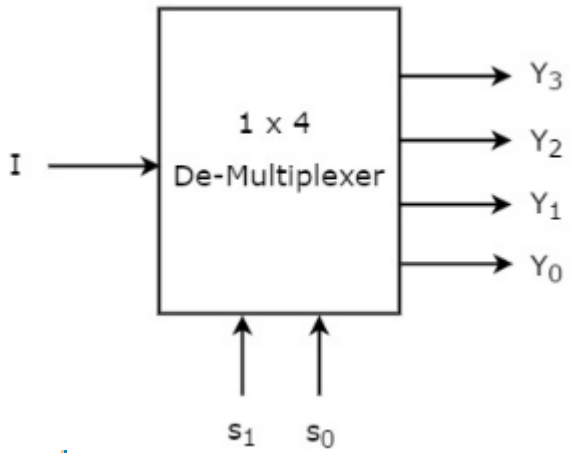


# De-Multiplexer

- The **demultiplexer** converts a serial data signal at the input to a parallel data at its output lines as shown below.



# 1:4 De-Multiplexer




| Selection Inputs | | Outputs | | | |
|---|---|---|---|---|---|
| $S_1$ | $S_0$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
| 0 | 0 | 0 | 0 | 0 | I |
| 0 | 1 | 0 | 0 | I | 0 |
| 1 | 0 | 0 | I | 0 | 0 |
| 1 | 1 | I | 0 | 0 | 0 |

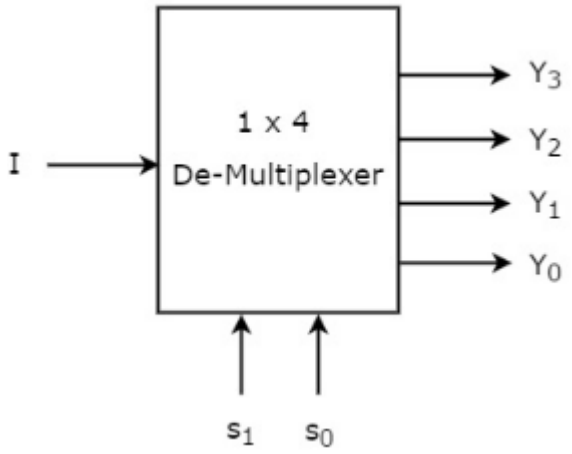
$$Y_3 = s_1 s_0 I$$

$$Y_2 = s_1 s_0' I$$

$$Y_1 = s_1' s_0 I$$

$$Y_0 = s_1' s_0' I$$

# 1:4 De-Multiplexer

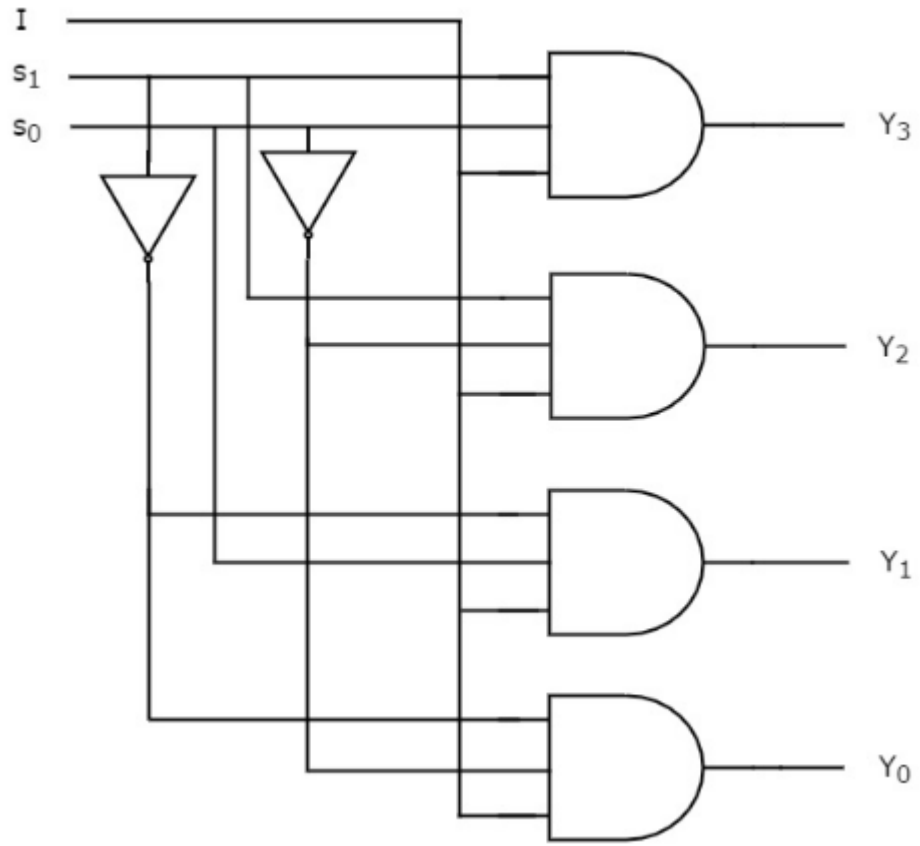



$$Y_3 = s_1 s_0 I$$

$$Y_2 = s_1 s_0' I$$

$$Y_1 = s_1' s_0 I$$

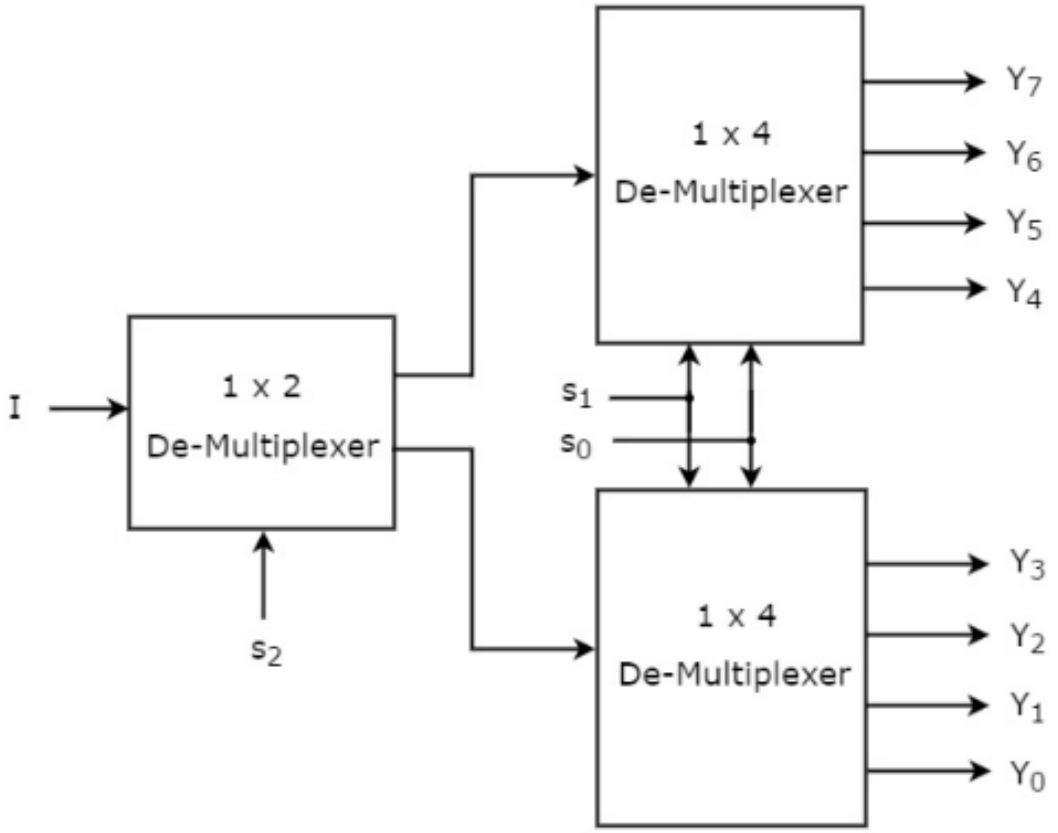
$$Y_0 = s_1' s_0' I$$




| Selection Inputs | | Outputs | | | |
|---|---|---|---|---|---|
| $S_1$ | $S_0$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
| 0 | 0 | 0 | 0 | 0 | I |
| 0 | 1 | 0 | 0 | I | 0 |
| 1 | 0 | 0 | I | 0 | 0 |
| 1 | 1 | I | 0 | 0 | 0 |

# 1:8 De-MUX using 1:4 De-MUX

| Selection Inputs | | | Outputs | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $S_2$ | $S_1$ | $S_0$ | $Y_7$ | $Y_6$ | $Y_5$ | $Y_4$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | I |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | I | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | I | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | I | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | I | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | I | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | I | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | I | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



# QUICK QUIZ (POLL)

To implement 1:8 de-mux how many 1:2 de-mux would be required?

a) 2
b) 4
c) 5
d) 6

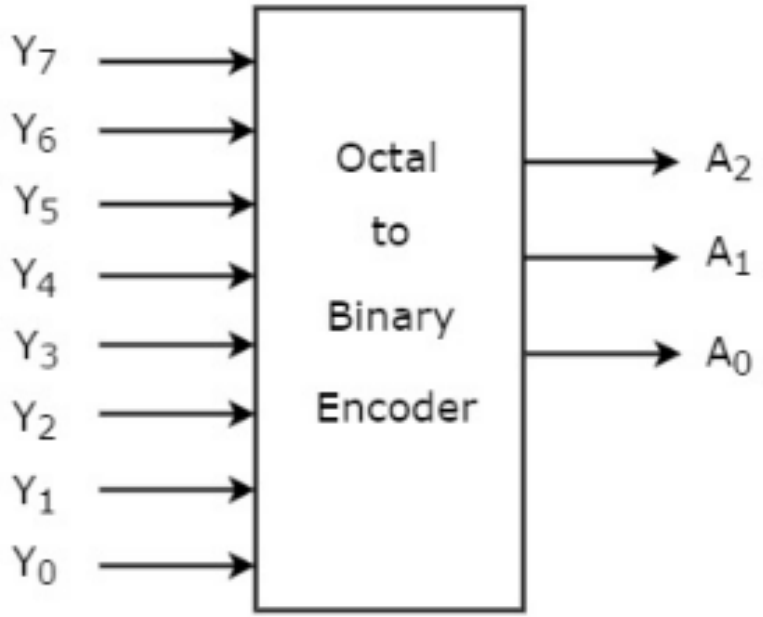
# Encoder

- An encoder is a combinational circuit that performs the inverse operation of a decoder.
- An encoder has $2^n$(or fewer) input lines and *n* output lines.
- The output lines, as an aggregate, generate the binary code corresponding to the input value.

# Encoder: Example

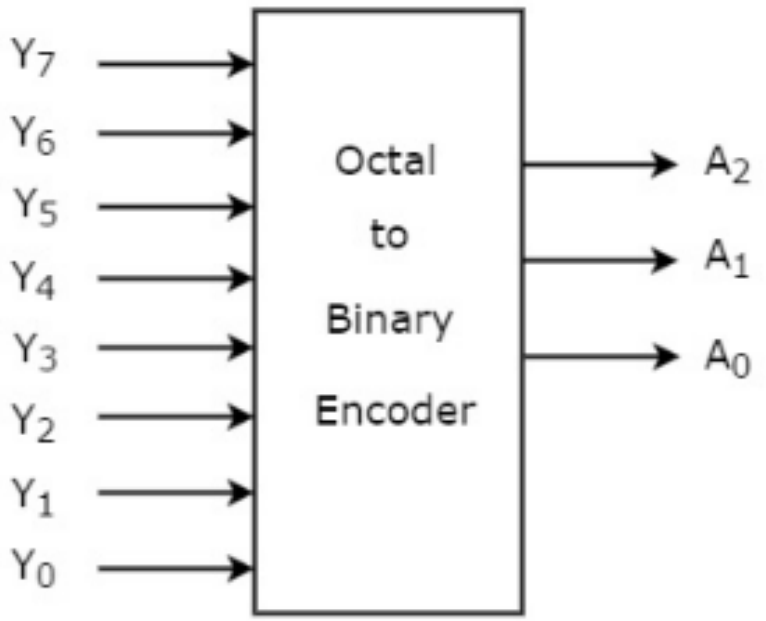
## Qsn: Design an octal to binary encoder?

Octal to binary Encoder has eight inputs, $Y_7$ to $Y_0$ and three outputs $A_2$, $A_1$ & $A_0$. Octal to binary encoder is nothing but 8 to 3 encoder. The **block diagram** of octal to binary Encoder is shown in the following figure.



# Encoder: Example

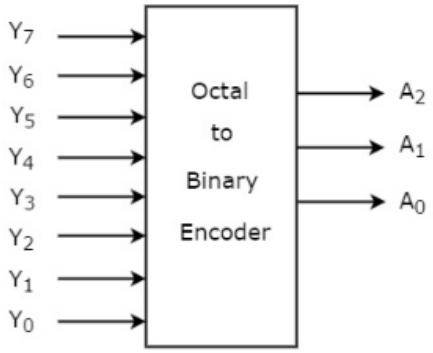
Qsn: Design an octal to binary encoder?



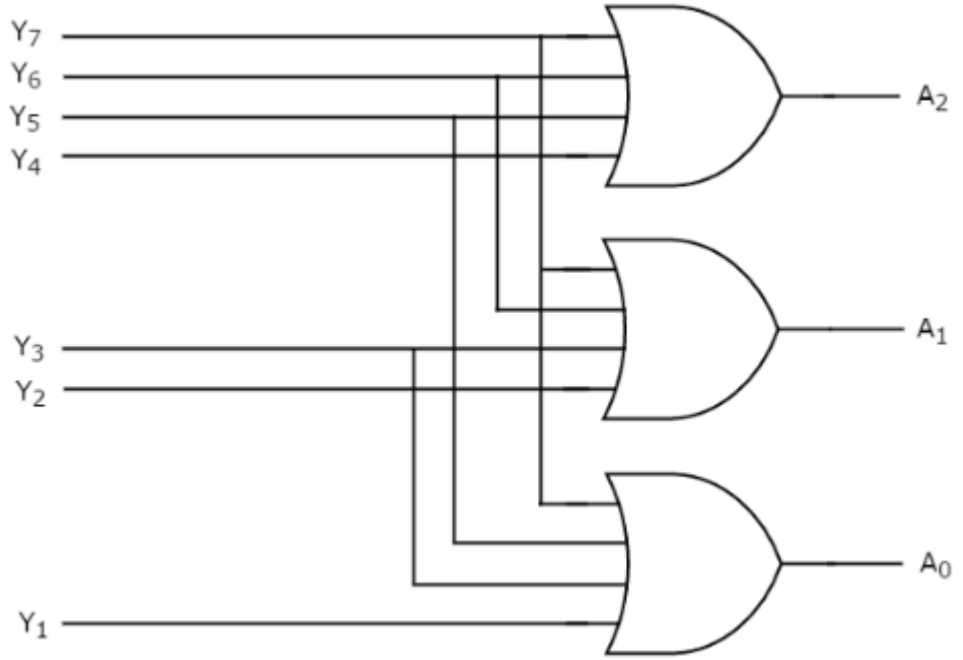

| Inputs | | | | | | | | Outputs | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $Y_7$ | $Y_6$ | $Y_5$ | $Y_4$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ | $A_2$ | $A_1$ | $A_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

# Encoder: Example

## Qsn: Design an octal to binary encoder?




| Inputs | | | | | | | | Outputs | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $Y_7$ | $Y_6$ | $Y_5$ | $Y_4$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ | $A_2$ | $A_1$ | $A_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |




$$A_2 = Y_7 + Y_6 + Y_5 + Y_4$$

$$A_1 = Y_7 + Y_6 + Y_3 + Y_2$$

$$A_0 = Y_7 + Y_5 + Y_3 + Y_1$$

# QUICK QUIZ (POLL)

How is an encoder different from a decoder?

a) The output of an encoder is a binary code for 1-of-N input

b) The output of a decoder is a binary code for 1-of-N input

c) The output of an encoder is a binary code for N-of-1 output

d) The output of a decoder is a binary code for N-of-1 output

# QUICK QUIZ (POLL)

How many outputs will a decimal-to-BCD encoder have?

a) 4

b) 8

c) 12

d) 16

# DIGITAL ELECTRONICS: ECE 213


**Topic: Priority Encoders, Comparator and Parity Generator and Checker**

**UNIT III: Introduction to Combinational Logic Circuits and Logic Families**

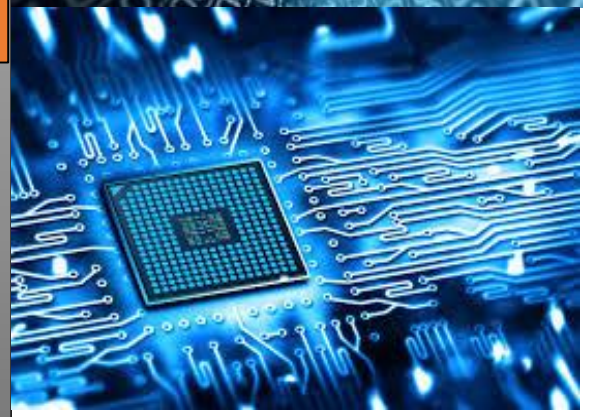
**Lecture No.: 19**

**Prepared By: Irfan Ahmad Pindoo**

**Assistant Professor**

**VLSI Design, ECE**

**School of Computer Science and Engineering**



# Limitations of Encoder

- There is an ambiguity, when all outputs of encoder are equal to zero. Because, it could be the code corresponding to the inputs, when only least significant input is one or when all inputs are zero.

- If more than one input is active High, then the encoder produces an output, which may not be the correct code. For example, if both Y3 and Y6 are '1', then the encoder produces 111 at the output. This is neither equivalent code corresponding to Y3, when it is '1' nor the equivalent code corresponding to Y6, when it is '1'.

- So, to overcome these difficulties, we should assign priorities to each input of encoder. Then, the output of encoder will be the binary code corresponding to the active High inputs, which has higher priority. This encoder is called as **priority encoder**.

# Priority Encoder

- A priority encoder is an encoder circuit that includes the priority function. The operation of the priority encoder is such that if two or more inputs are equal to 1 at the same time, the input having the **highest priority will take precedence.**

- In addition to the two outputs x and y, the circuit has a third output designated by V; this is a **valid bit indicator** that is set to 1 when one or more inputs are equal to 1. If all inputs are 0, there is no valid input and V is equal to 0.
- The other two outputs are not inspected when V equals 0 and are specified as don't-care conditions.

# Priority Encoder

## Designing of 4X2 Priority Encoder

*Truth Table of a Priority Encoder*

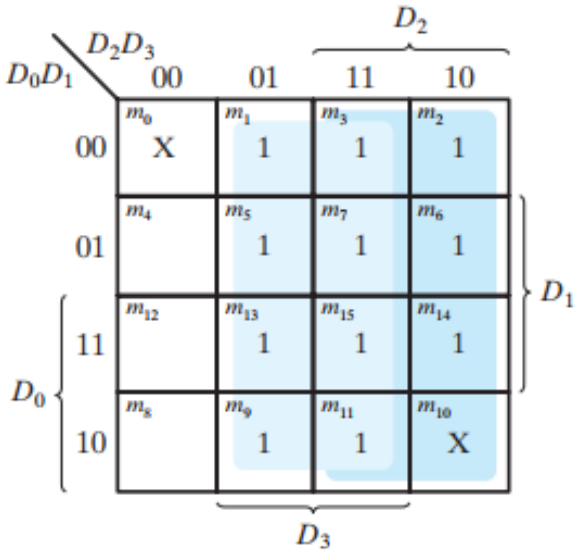
| Inputs | | | | Outputs | | |
|---|---|---|---|---|---|---|
| $D_0$ | $D_1$ | $D_2$ | $D_3$ | $x$ | $y$ | $V$ |
| 0 | 0 | 0 | 0 | X | X | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| X | 1 | 0 | 0 | 0 | 1 | 1 |
| X | X | 1 | 0 | 1 | 0 | 1 |
| X | X | X | 1 | 1 | 1 | 1 |

# Priority Encoder

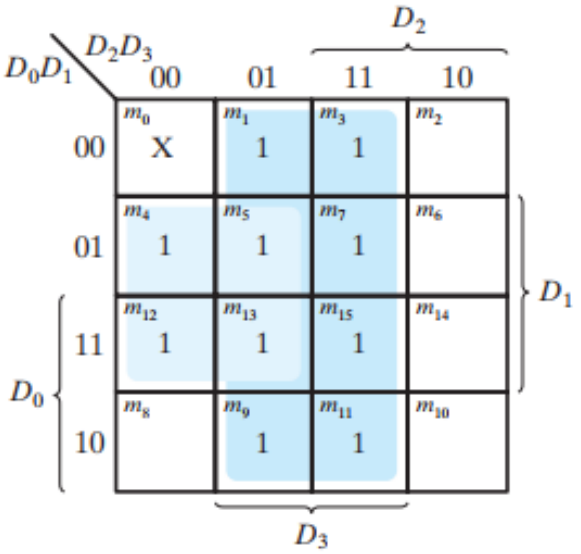
## Designing of 4X2 Priority Encoder

*Truth Table of a Priority Encoder*

| Inputs | | | | Outputs | | |
|---|---|---|---|---|---|---|
| $D_0$ | $D_1$ | $D_2$ | $D_3$ | $x$ | $y$ | $V$ |
| 0 | 0 | 0 | 0 | X | X | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| X | 1 | 0 | 0 | 0 | 1 | 1 |
| X | X | 1 | 0 | 1 | 0 | 1 |
| X | X | X | 1 | 1 | 1 | 1 |




$x = D_2 + D_3$




$y = D_3 + D_1D'_2$

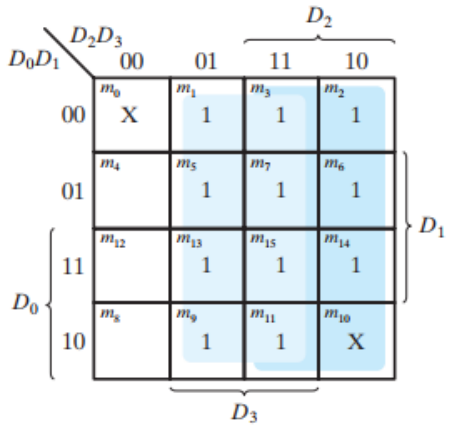
# Priority Encoder

## Designing of 4X2 Priority Encoder

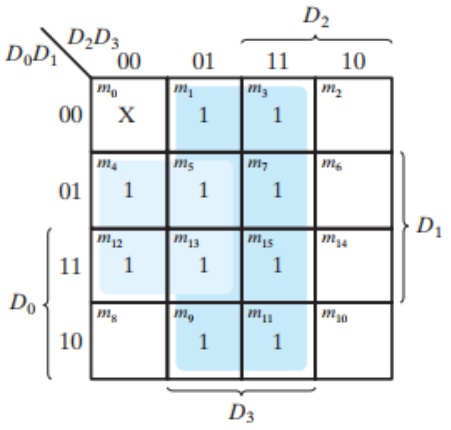
*Truth Table of a Priority Encoder*

| Inputs | | | | Outputs | | |
|---|---|---|---|---|---|---|
| $D_0$ | $D_1$ | $D_2$ | $D_3$ | $x$ | $y$ | $V$ |
| 0 | 0 | 0 | 0 | X | X | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| X | 1 | 0 | 0 | 0 | 1 | 1 |
| X | X | 1 | 0 | 1 | 0 | 1 |
| X | X | X | 1 | 1 | 1 | 1 |

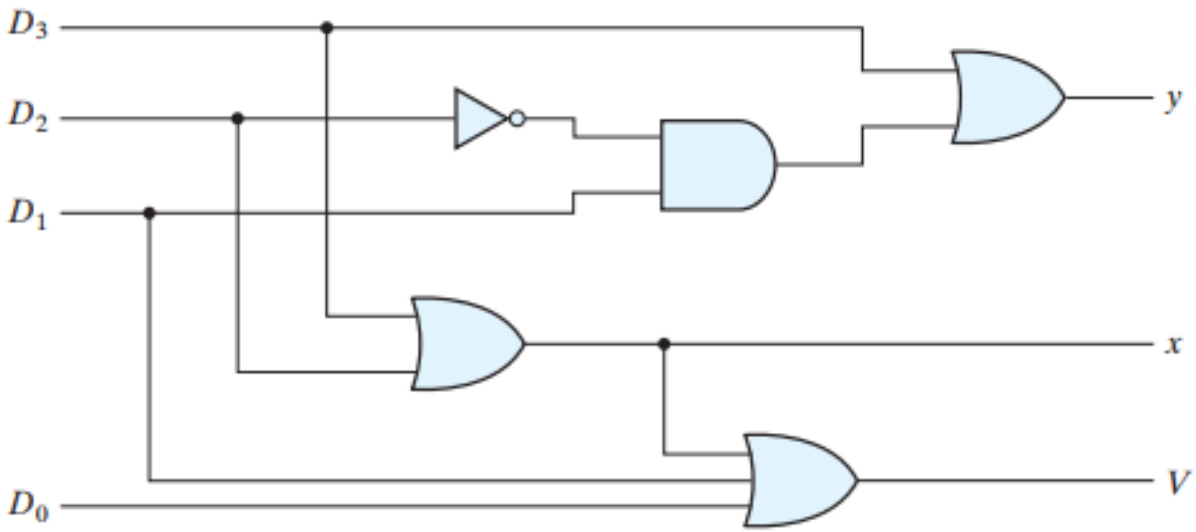
$$x = D_2 + D_3$$
$$y = D_3 + D_1 D_2'$$
$$V = D_0 + D_1 + D_2 + D_3$$




$x = D_2 + D_3$




$y = D_3 + D_1 D'_2$



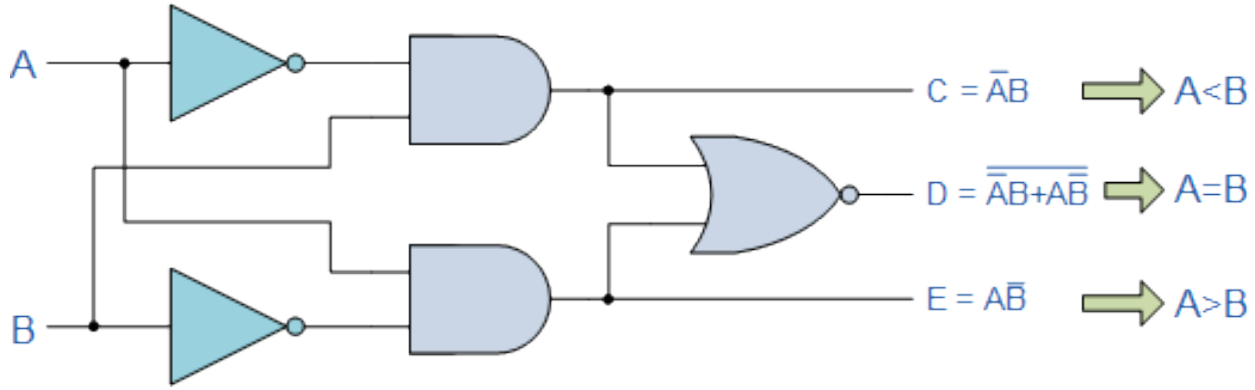
# Magnitude Comparator

- Another common and very useful combinational logic circuit is that of the Digital Comparator circuit.
- Digital or Binary Comparators are made up from standard AND, NOR and NOT gates that compare the digital signals present at their input terminals and produce an output depending upon the condition of those inputs.

# 1-Bit Magnitude Comparator

- Another common and very useful combinational logic circuit is that of the Digital Comparator circuit.
- Digital or Binary Comparators are made up from standard AND, NOR and NOT gates that compare the digital signals present at their input terminals and produce an output depending upon the condition of those inputs.

| A | B | A<B | A=B | A>B |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |



# 2-Bit Magnitude Comparator

**Step 1:**

**Determine number of inputs and outputs.**

**Step 2 and 3:**

| INPUT | | | | OUTPUT | | |
|---|---|---|---|---|---|---|
| A1 | A0 | B1 | B0 | A<B | A=B | A>B |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 |

# 2-Bit Magnitude Comparator

**Step 2 and 3:**

| INPUT | | | | OUTPUT | | |
|---|---|---|---|---|---|---|
| A1 | A0 | B1 | B0 | A<B | A=B | A>B |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 |

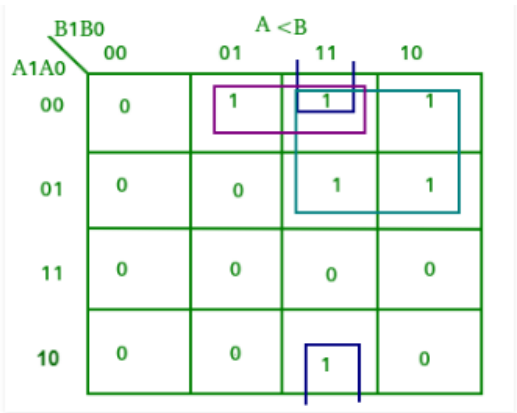
**Step 4: K Maps**

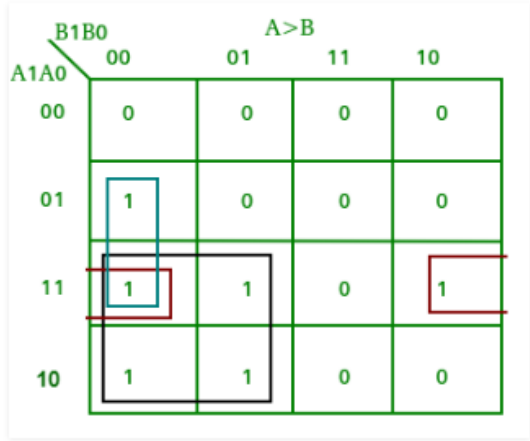
# 2-Bit Magnitude Comparator

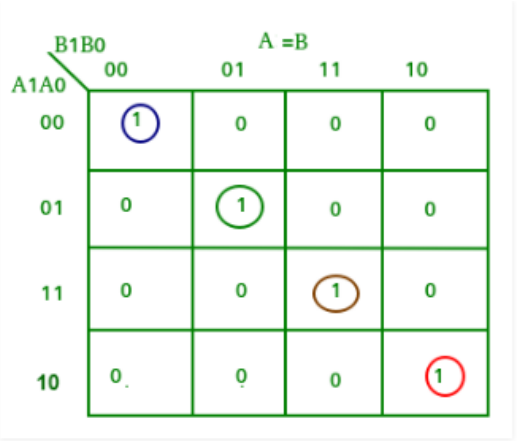
**Step 2 and 3:**

| INPUT | | | | OUTPUT | | |
|---|---|---|---|---|---|---|
| A1 | A0 | B1 | B0 | A<B | A=B | A>B |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 |

**Step 4: K Maps**

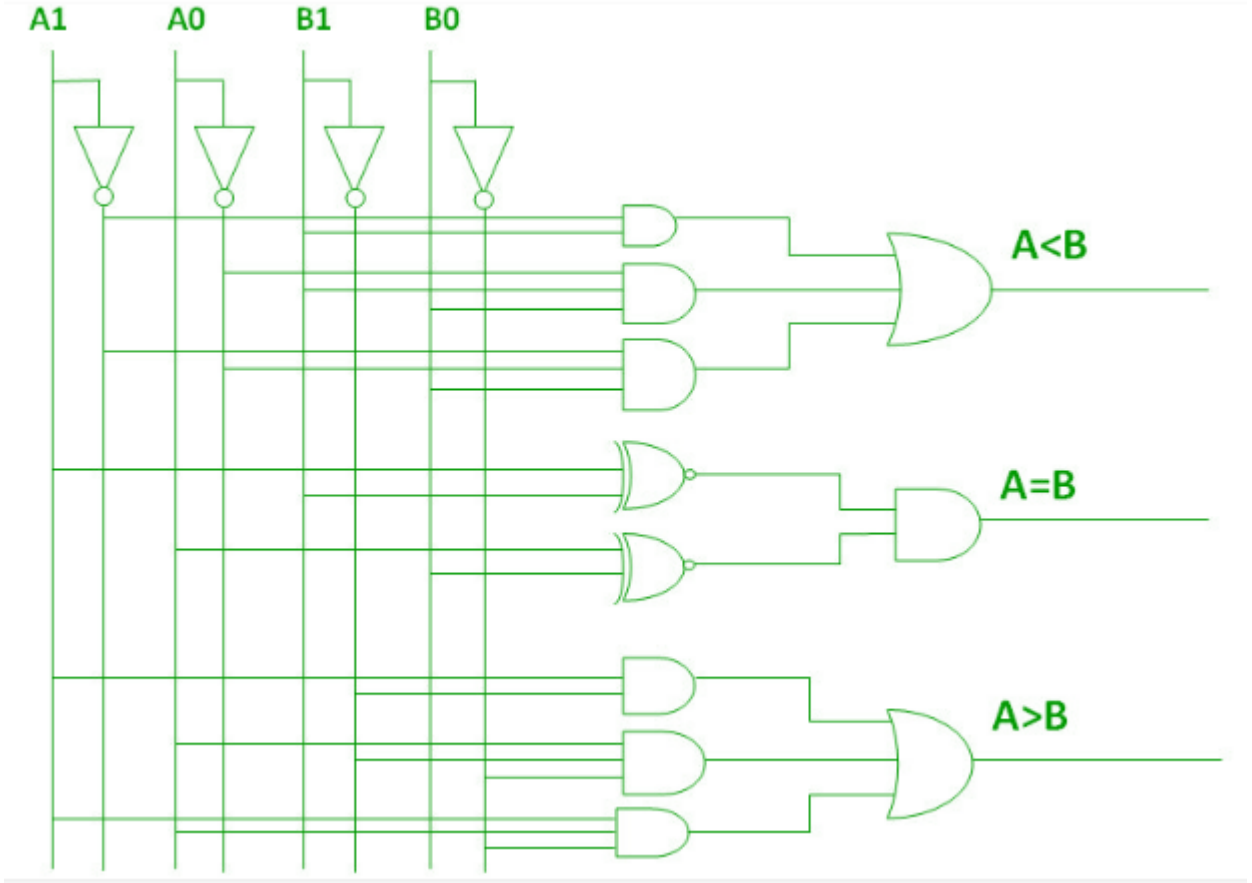


# 2-Bit Magnitude Comparator

**Step 5: Logic Circuit**



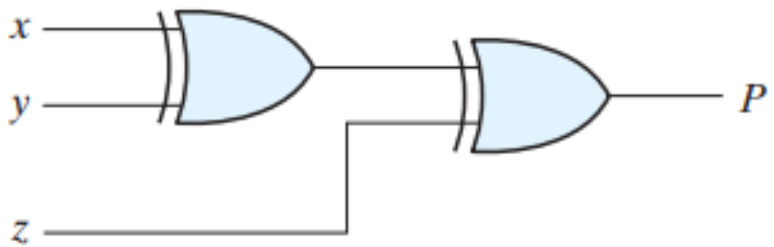
# Parity Generation and Checking

- Exclusive-OR functions are very useful in systems requiring error detection and correction codes.
- A parity bit is an extra bit included with a binary message to make the number of 1's either odd or even.
- The message, including the parity bit, is transmitted and then checked at the receiving end for errors.
- The circuit that generates the parity bit in the transmitter is called a parity generator. The circuit that checks the parity in the receiver is called a parity checker.

# Parity Generation and Checking

*Even-Parity-Generator Truth Table*

| Three-Bit Message | | | Parity Bit |
|---|---|---|---|
| **x** | **y** | **z** | **P** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |



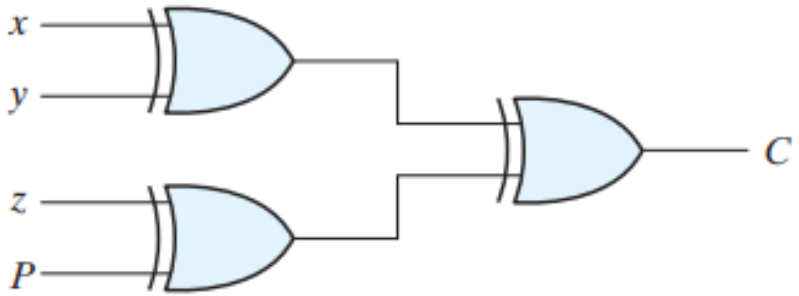

(a) 3-bit even parity generator

$$P = x \oplus y \oplus z$$

# Parity Generation and Checking

*Even-Parity-Checker Truth Table*

| Four Bits Received | | | | Parity Error Check |
|---|---|---|---|---|
| *x* | *y* | *z* | *P* | *C* |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |




(b) 4-bit even parity checker

$$C = x \oplus y \oplus z \oplus P$$

# DIGITAL ELECTRONICS: ECE 213


## Topic: Digital IC Logic Families

## UNIT III: Introduction to Combinational Logic Circuits and Logic Families

## Lecture No.: 20

Prepared By: Irfan Ahmad Pindoo

Assistant Professor

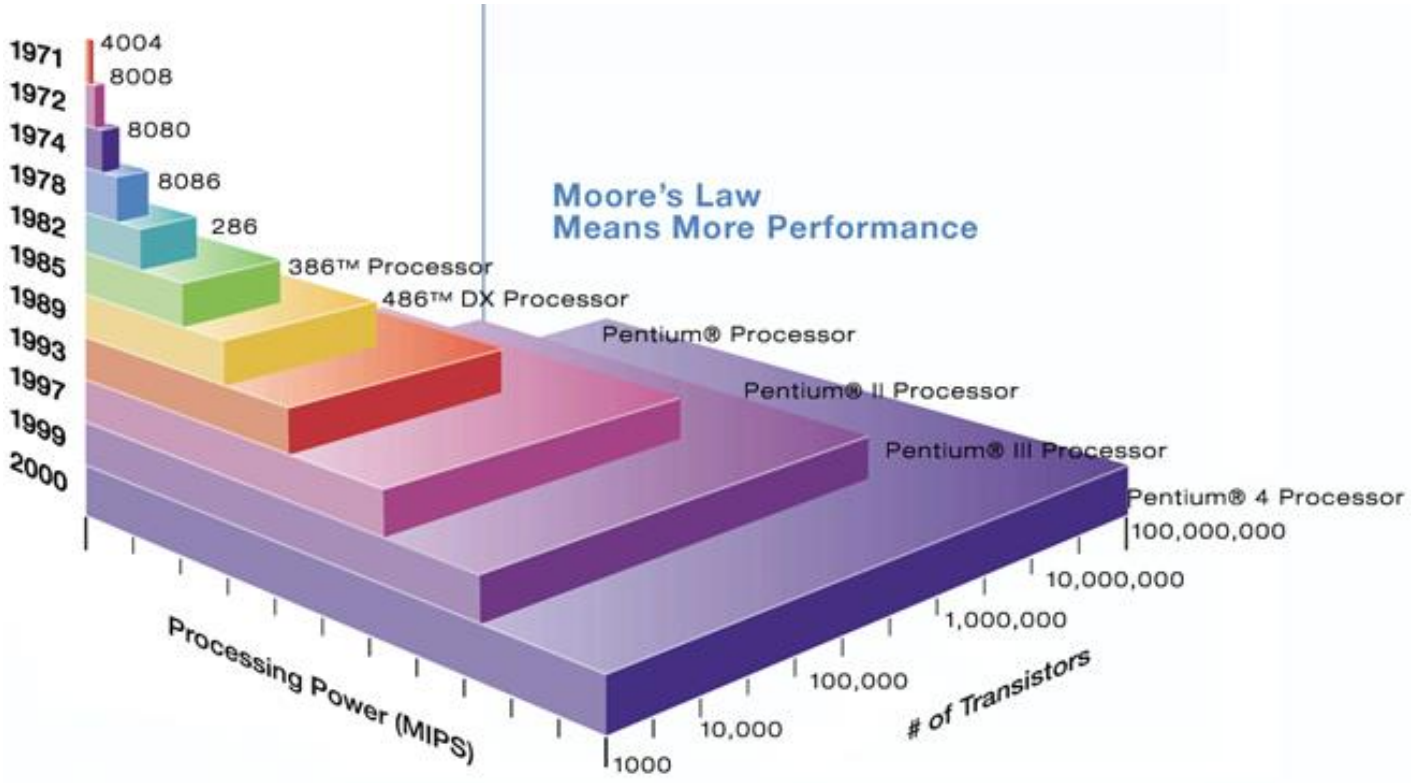
VLSI Design, ECE

School of Computer Science and Engineering

# DIGITAL IC LOGIC FAMILIES:

# Moore's Law

- A prediction made by Moore (a co-founder of Intel) in 1965: "… a number of transistors to double every 2 years."



# Logic Families

**OBSOLETE ONES:**

1. Diode Logic.
2. Diode Transistor Logic (DTL).
3. Resistor Transistor Logic (RTL).

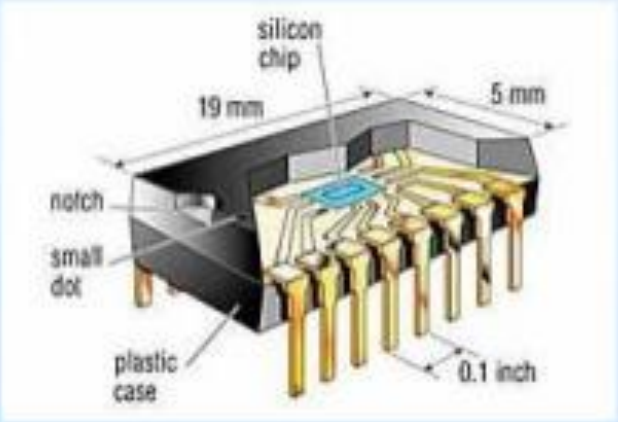
**CURRENT ONES:**

1. TTL (Transistor Transistor Logic)
2. ECL (Emitter Coupled Logic)
3. CMOS (Complementary Metal Oxide Semiconductor)

# Characteristics of an Ideal Logic Family

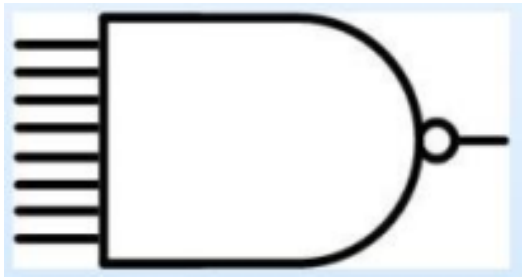
**The most important parameters for evaluating and comparing logic families include :**

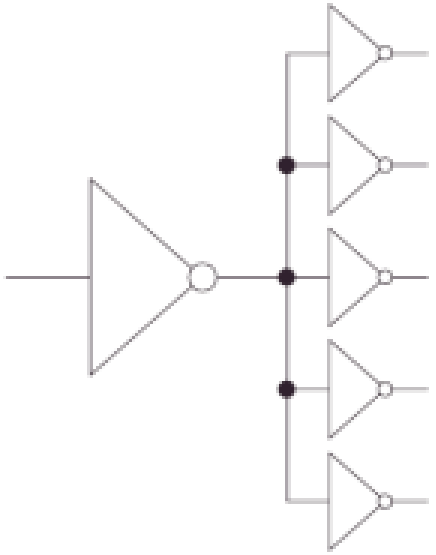
- Logic Levels
- Power Dissipation
- Propagation delay
- Noise margin
- Fan-out ( loading )



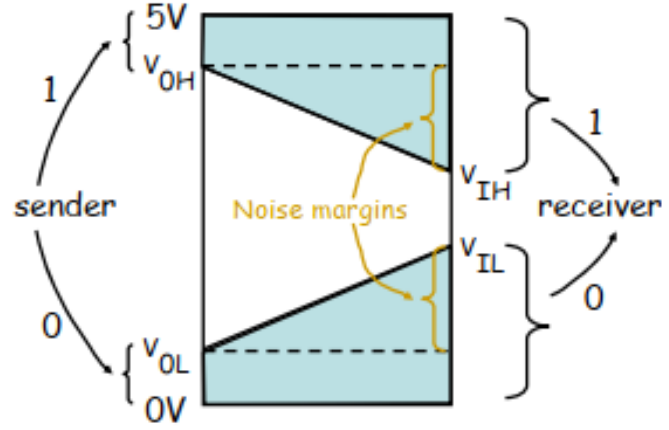
# Characteristics of Logic Family

- Fan in : The number of inputs that the gate can handle properly with out disturbing the output level.
- Fan out : The number of inputs that can driven simultaneously by the output with out disturbing the output level.
- Noise immunity : Noise immunity is the ability of the logic circuit to tolerate the noise voltage.





Hold the sender to tougher standards!




"1" noise margin: $V_{IH} - V_{OH}$

"0" noise margin: $V_{IL} - V_{OL}$

As illustrated in Figure the *noise margin for a logical 0* is given by

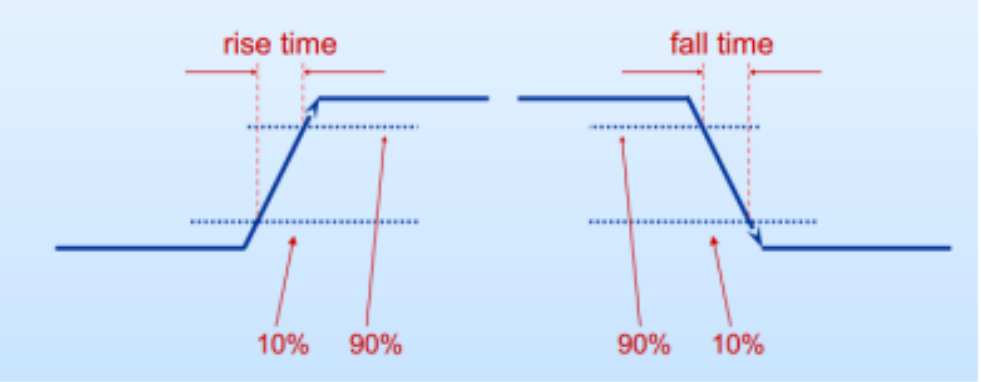
$$NM_0 = V_{IL} - V_{OL}$$

and the *noise margin for a logical 1* is given by

$$NM_1 = V_{OH} - V_{IH}.$$

# Characteristics of Logic Family

- Noise Margin : The quantative measure of noise immunity is called noise margin.
- Propagation Delay : The propagation delay of gate is the average transition delay time for the signal to propagate from input to output
- Threshold Voltage : The voltage at which the circuit changes from one state to another state
- Operating Speed : The speed of operation of the logic gate is the time that elapses between giving input and getting output.
- Power Dissipation : The power dissipation is defined as power needed by the logic circuit.



# QUICK QUIZ (POLL)

If a logic circuit has a fan out of 4 then the circuit

A. 4 input

B. has 4 outputs

C. can drive maximum of 4 inputs

D. gives output 4 times the input

# QUICK QUIZ (POLL)

The rise time ($t_r$) is the time it takes for a pulse to rise from its _____ point up to its _____ point. The fall time ($t_f$) is the length of time it takes to fall from the _____ to the _____ point.

**A.** 10%, 90%, 90%, 10%

**B.** 90%, 10%, 10%, 90%

**C.** 20%, 80%, 80%, 20%

**D.** 10%, 70.7%, 70.7%, 10%

# QUICK QUIZ (POLL)

A RTL consists of

A. Resistor ,transistor and inductors

B. Resistors,diodes and bipolar junction transistor

C. Resistors,capacitors and diodes

D. Resistors and transistors

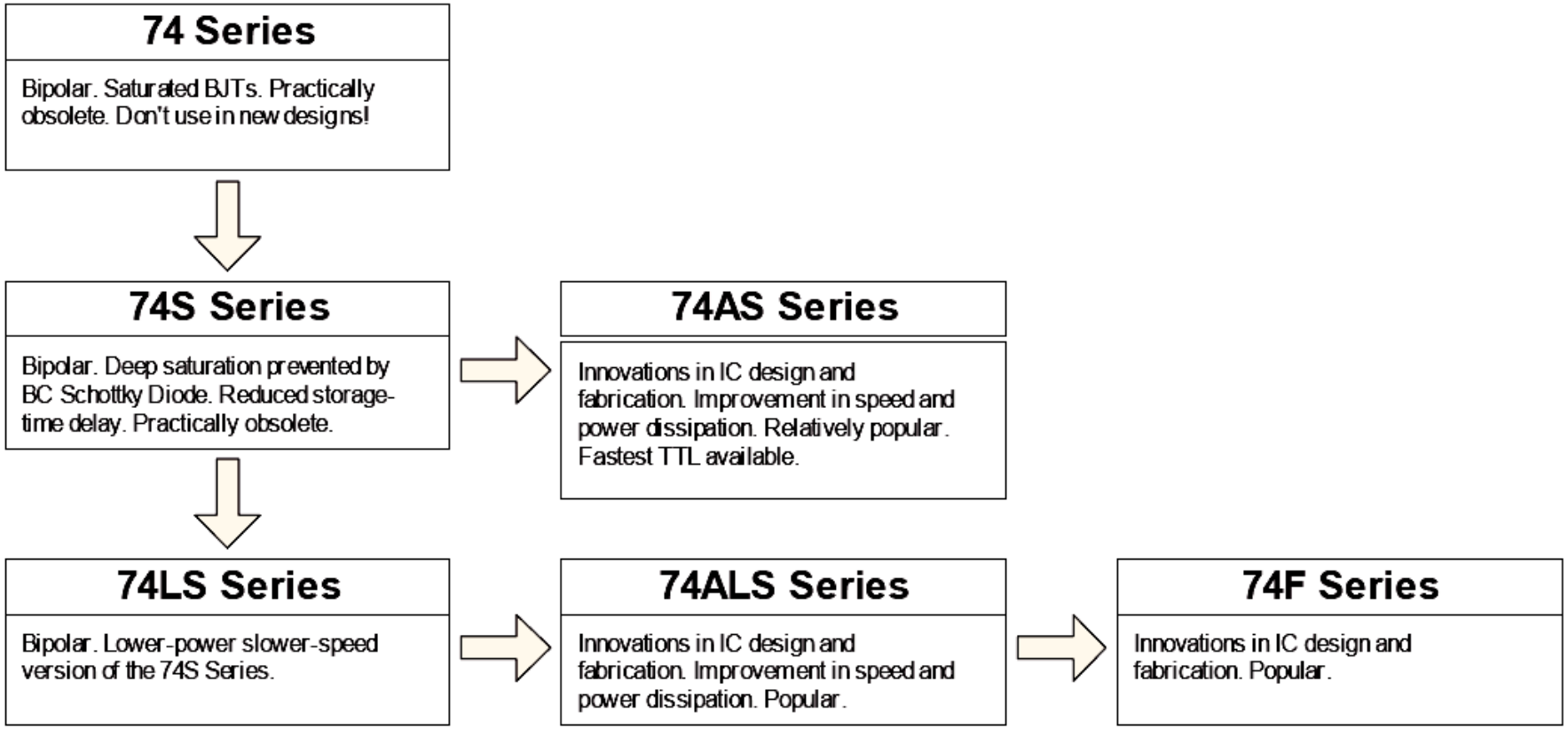
# TTL IC Family

- In Transistor-Transistor logic or just TTL, logic gates are built only around transistors
- TTL was developed in 1965
- Through the years basic TTL has been improved to meet performance requirements. There are many versions or families of TTL. For example
  - Standard TTL
  - High Speed TTL (twice as fast, twice as much power)
  - Low Power TTL (1/10 the speed, 1/10 the power of "standard" TTL)
  - Schhottky TTL etc. (for high-frequency uses )

# TTL IC Family

- Transistor-Transistor Logic Families:
  - 74L — Low power
  - 74H — High speed
  - 74S — Schottky
  - 74LS — Low power Schottky
  - 74AS — Advanced Schottky
  - 74ALS — Advance Low power Schottky

# TTL IC Family Evolution




Legacy: don't use in new designs

Widely used today

# ECL IC Family

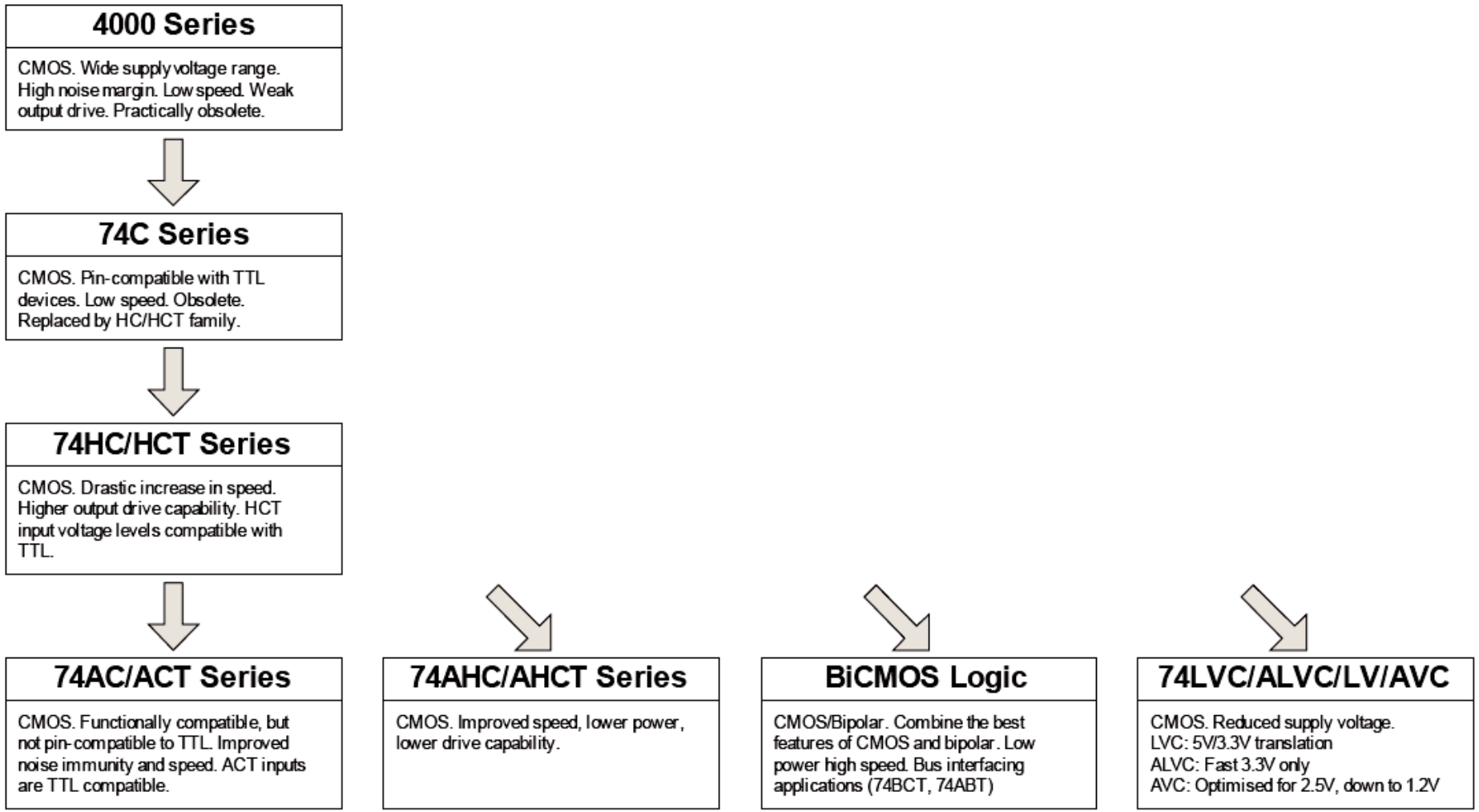
- PROS: Fastest logic family available (~1ns)
- CONS: low noise margin and high power dissipation
- Operated in emitter coupled geometry (recall differential amplifier or emitter-follower), transistors are biased and operate near their Q-point (never near saturation!)
- Logic levels. "0": –1.7V. "1": –0.8V
- Such strange logic levels require extra effort when interfacing to TTL/CMOS logic families

# CMOS IC Family

## Complementary MOS (CMOS)

- Other variants: NMOS, PMOS (obsolete)
- Very low static power consumption
- Scaling capabilities (large integration all MOS)
- Full swing: rail-to-rail output

# CMOS Family Evolution



# QUICK QUIZ (POLL)

**The standard TTL gates are marketed as _____ series.**

a) 80

b) 82

c) 74

d) 08

# QUICK QUIZ (POLL)

. The full form of ECL is _____

a) Emitter-collector logic

b) Emitter-complementary logic

c) Emitter-coupled logic

d) Emitter-cored logic

# QUICK QUIZ (POLL)

Which logic is the fastest of all the logic families?

a) TTL

b) ECL

c) HTL

d) DTL

# TTL vs CMOS

- TTL
  - faster (some versions)
  - strong drive capability
  - rugged
- CMOS
  - lower power consumption
  - simpler to make
  - greater packing density
  - better noise immunity

# QUICK QUIZ (POLL)

**Which of the following is not the advantage of MOS gates?**

a) Low power dissipation

b) Small size

c) Good immunity to noise

d) High switching speeds

# DIGITAL ELECTRONICS: ECE 213


**Topic:** Digital IC Logic Families

**UNIT III:** Introduction to Combinational Logic Circuits and Logic Families

**Lecture No.:** 21

Prepared By: Irfan Ahmad Pindoo

Assistant Professor

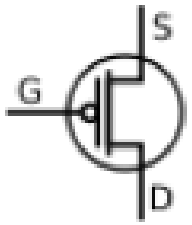
VLSI Design, ECE

School of Computer Science and Engineering

# CMOS as a Switch

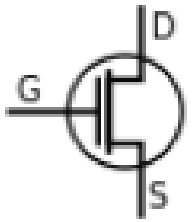
PMOS

ON➔ Gate input Low
OFF➔ Gate input High




NMOS

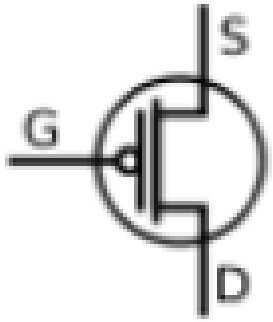
OFF➔ Gate input Low
ON➔ Gate input High



# QUICK QUIZ (POLL)

If the input applied to the gate is GND, what would be the output of the following transistor?

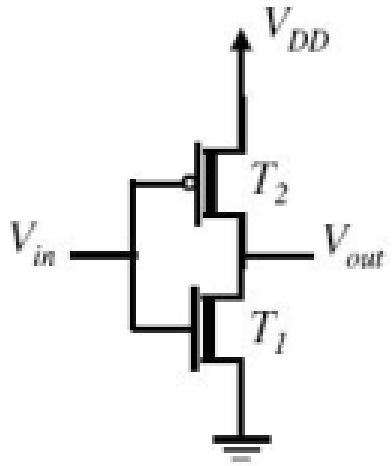
a) ON state
b) OFF state
c) GND
d) None of these



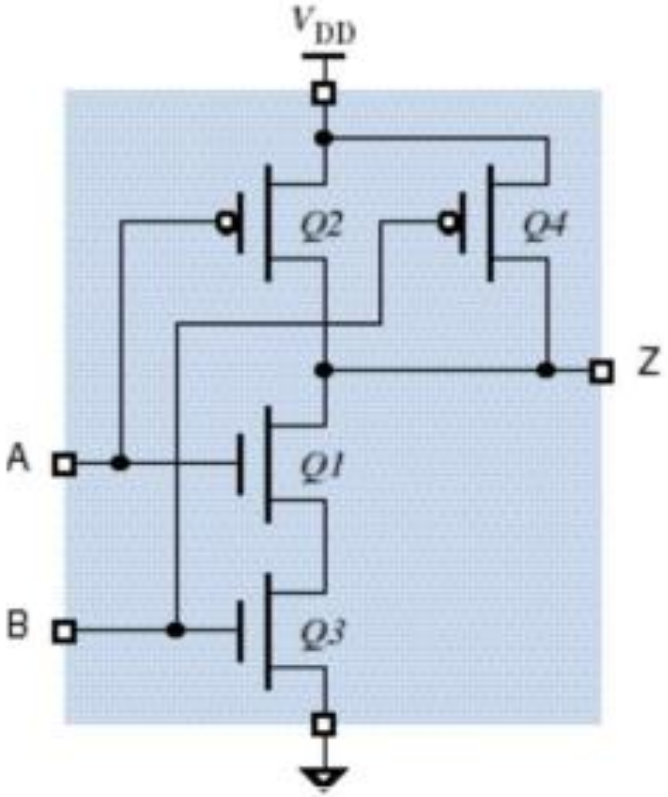
# CMOS as a NOT Gate

| *Truth Table* | *Integrated Circuit* |
|---|---|

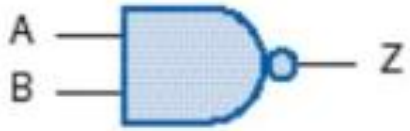
| $V_{in}$ | $T_1$ | $T_2$ | $V_{out}$ |
|---|---|---|---|
| 0 | *off* | *on* | 1 |
| 1 | *off* | *off* | 0 |



# CMOS as a NAND Gate

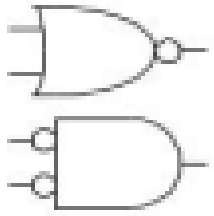


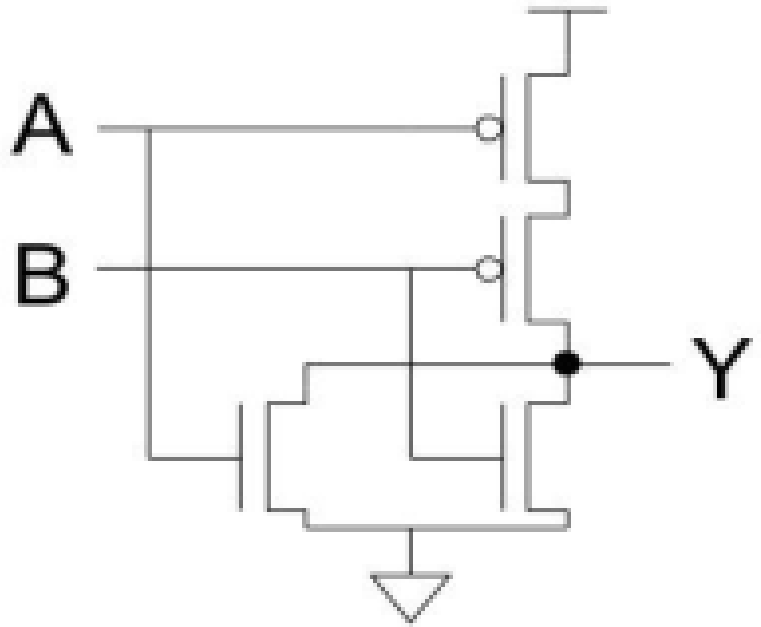

| A | B | Q1 | Q2 | Q3 | Q4 | Z |
|---|---|---|---|---|---|---|
| L | L | off | on | off | on | H |
| L | H | off | on | on | off | H |
| H | L | on | off | off | on | H |
| H | H | on | off | on | off | L |



# CMOS as a NOR Gate

| A | B | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

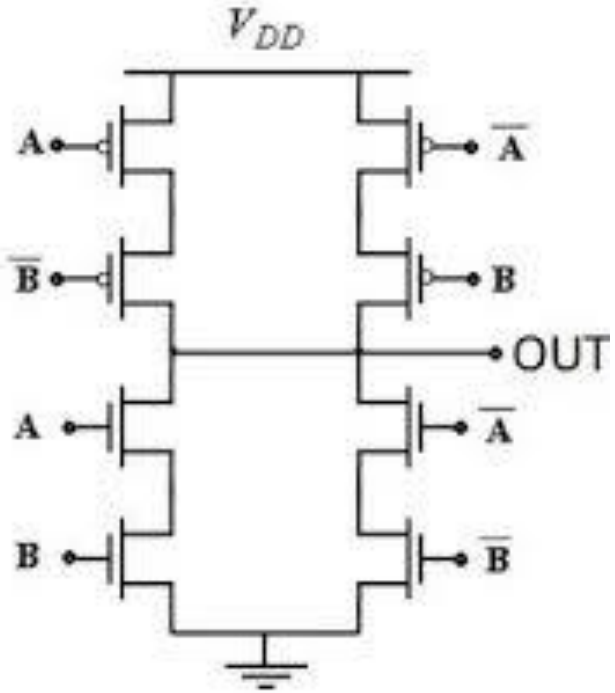




# QUICK QUIZ (POLL)

Identify the logic gate provided below?

a) AND gate
b) OR gate
c) XNOR gate
d) XOR gate



# Comparison of Logic Families

| Parameter | CMOS | TTL | ECL |
|---|---|---|---|
| Basic gate | NAND/NOR | NAND | OR/NOR |
| Fan-out | >50 | 10 | 25 |
| Power per gate (mW) | 1 @ 1 MHz | 1 - 22 | 4 - 55 |
| Noise immunity | Excellent | Very good | Good |
| $t_{PD}$ (ns) | 1 - 200 | 1.5 – 33 | 1 - 4 |

# DIGITAL ELECTRONICS: ECE 213


**Topic:** PRACTICE QUESTIONS.

**UNIT III:** Introduction to Combinational Logic Circuits and Logic Families

**Lecture No.:** 22

Prepared By: Irfan Ahmad Pindoo

Assistant Professor

VLSI Design, ECE

School of Computer Science and Engineering

# PRACTICE QUESTIONS:

# PRACTICE QUESTIONS:

Q. Design a combinational circuit that converts:

a) 4-bit binary to Gray code.

b) 4- bit Gray code to binary

# PRACTICE QUESTIONS:

Q. Design a combinational circuit that converts:

a) 4-bit binary to Gray code.

b) 4- bit Gray code to binary

| Binary | | | | Gray Code | | | |
|---|---|---|---|---|---|---|---|
| $b_3$ | $b_2$ | $b_1$ | $b_0$ | $g_3$ | $g_2$ | $g_1$ | $g_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

# PRACTICE QUESTIONS:

Q. Design a combinational circuit that converts:

a) 4-bit binary to Gray code.

b) 4- bit Gray code to binary

| Binary | | | | Gray Code | | | |
|---|---|---|---|---|---|---|---|
| $b_3$ | $b_2$ | $b_1$ | $b_0$ | $g_3$ | $g_2$ | $g_1$ | $g_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

K-map for $G_0$

| $B_3B_2$ \ $B_1B_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 0 | 1 |
| 01 | 0 | 1 | 0 | 1 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 0 | 1 | 0 | 1 |

$G_0 = B_1'B_0 + B_1B_0'$

$G_0 = B_0 \oplus B_1$

K-map for $G_1$

| $B_3B_2$ \ $B_1B_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 1 | 1 |
| 01 | 1 | 1 | 0 | 0 |
| 11 | 1 | 1 | 0 | 0 |
| 10 | 0 | 0 | 1 | 1 |

$G_1 = B_1'B_2 + B_1B_2'$

$G_2 = B_1 \oplus B_2$

K-map for $G_2$

| $B_3B_2$ \ $B_1B_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 1 | 1 | 1 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 1 | 1 | 1 | 1 |

$G_2 = B_3'B_2 + B_3B_2'$

$G_2 = B_2 \oplus B_3$

And, G3 = B3

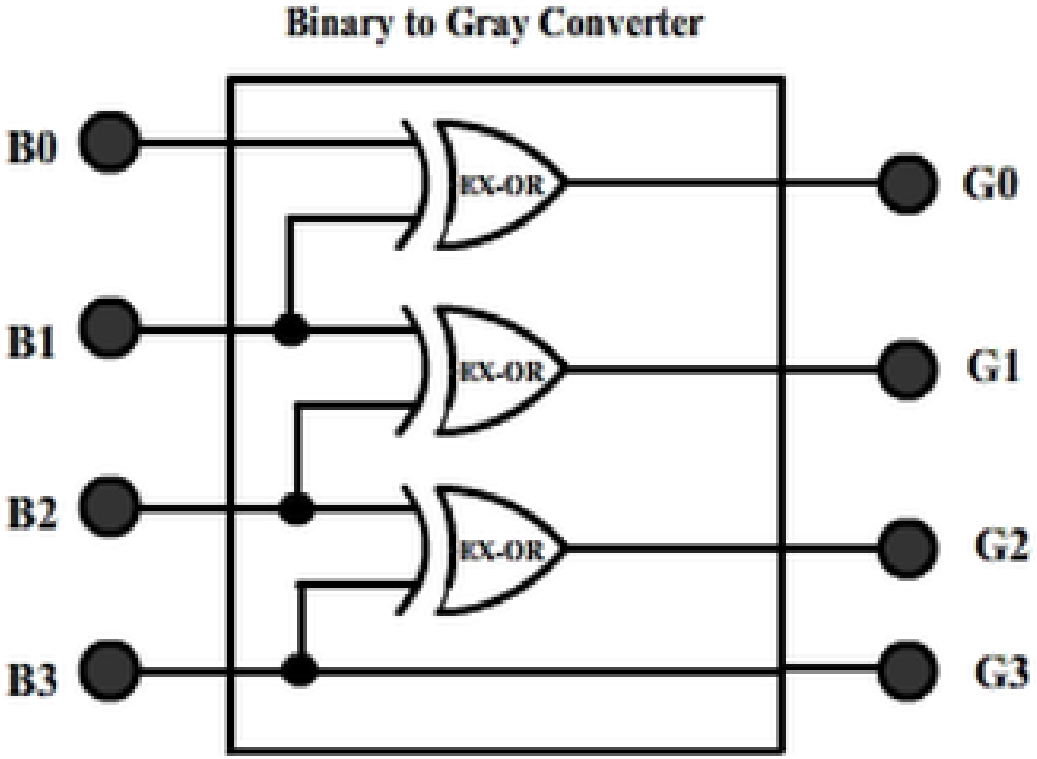
# PRACTICE QUESTIONS:

Q. Design a combinational circuit that converts:

a) 4-bit binary to Gray code.

b) 4- bit Gray code to binary

| Binary | | | | Gray Code | | | |
|---|---|---|---|---|---|---|---|
| $b_3$ | $b_2$ | $b_1$ | $b_0$ | $g_3$ | $g_2$ | $g_1$ | $g_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |



# PRACTICE QUESTIONS:

Implement the Boolean function $F = xy + x'y' + y'z$ using:

a) Decoder

b) Multiplexer

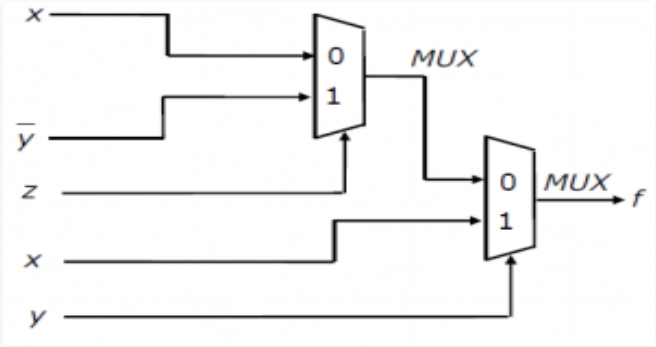
# PRACTICE QUESTIONS:

Implement the Boolean function $F = xy + x'y' + y'z$ using:

a) Decoder

b) Multiplexer

# GATE Problem




Consider the circuit above. Which one of the following options correctly represents f (x, y, z)?

A xz' + xy + y'z

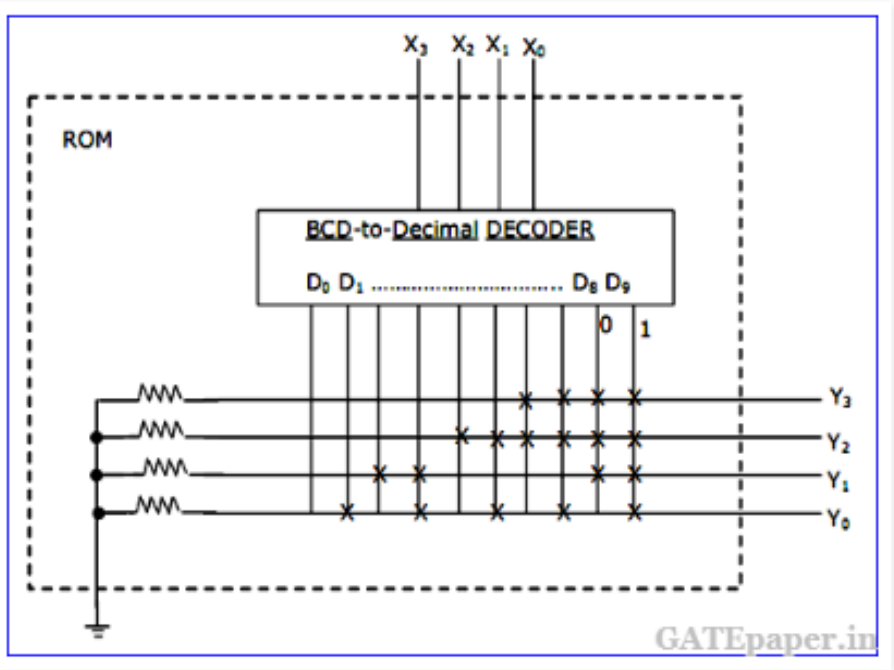
B xz' + xy + (yz)'

C xz + xy + (yz)'

D xz + xy' + y'z

# GATE Problem

If the inputs $X_3, X_2, X_1, X_0$ to the ROM in the figure are 8-4-2-1 BCD numbers, then the outputs $Y_3Y_2Y_1Y_0$ are




a. Gary code numbers
b. 2-4-2-1 BCD numbers
c. Excess-3 code numbers
d. None of the above

# GATE Problem

In the following truth table, V = 1 if and only if the input is valid.

| Inputs | | | | Outputs | | |
|---|---|---|---|---|---|---|
| $D_0$ | $D_1$ | $D_2$ | $D_3$ | $X_0$ | $X_1$ | V |
| 0 | 0 | 0 | 0 | x | x | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| x | 1 | 0 | 0 | 0 | 1 | 1 |
| x | x | 1 | 0 | 1 | 0 | 1 |
| x | x | x | 1 | 1 | 1 | 1 |

What function does the truth table represent?

a) Multiplexer b) Encoder c) Decoder d) Priority Encoder

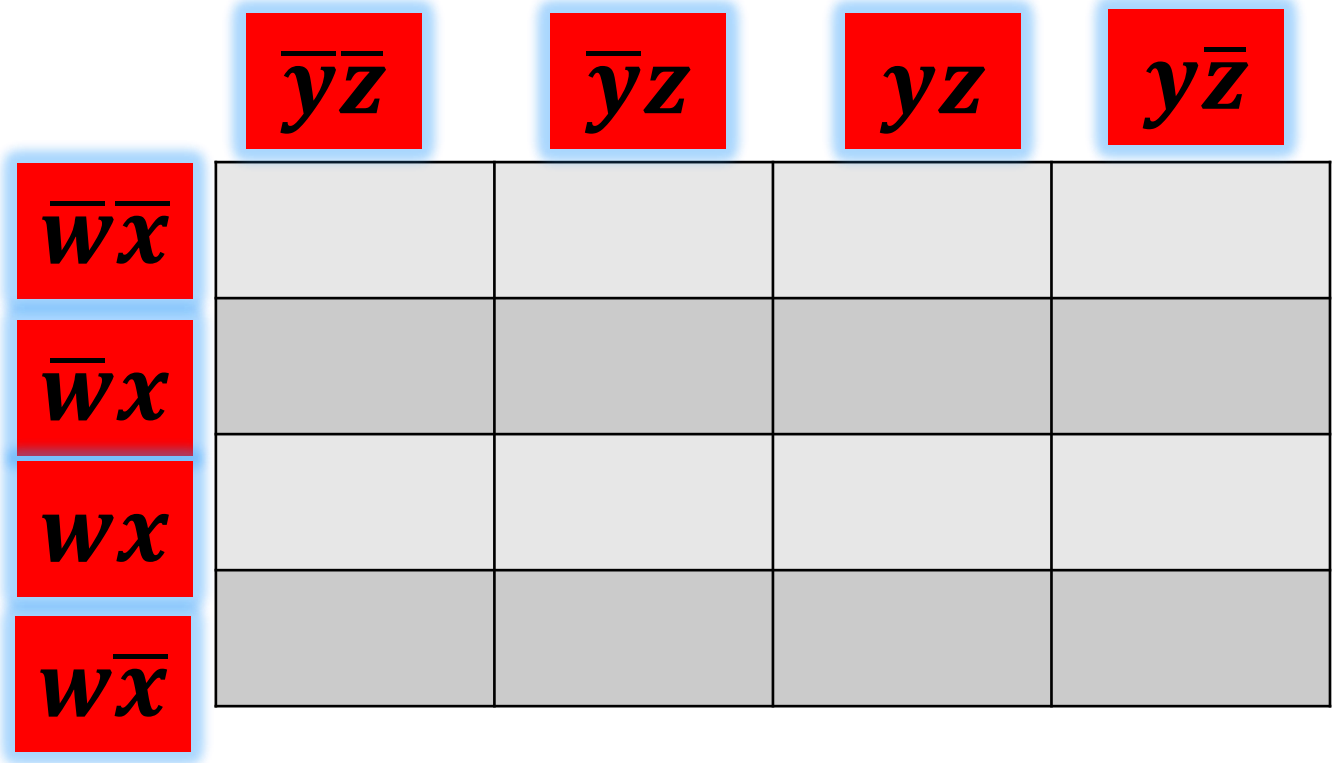
# PRACTICE QUESTIONS:

Find essential prime implicants of following functions, using K-map:

(a) $F(w, x, y, z) = \Sigma(0, 2, 5, 7, 8, 10, 12, 13, 14, 15)$

(b) $F(A, B, C, D) = \Sigma(0, 2, 3, 5, 7, 8, 10, 11, 14, 15)$

(c)* $F(A, B, C, D) = \Sigma(1, 3, 4, 5, 10, 11, 12, 13, 14, 15)$



# PRACTICE QUESTIONS:

Find essential prime implicants of following functions, using K-map:

(a) $F(w, x, y, z) = \Sigma(0, 2, 5, 7, 8, 10, 12, 13, 14, 15)$

(b) $F(A, B, C, D) = \Sigma(0, 2, 3, 5, 7, 8, 10, 11, 14, 15)$

(c)* $F(A, B, C, D) = \Sigma(1, 3, 4, 5, 10, 11, 12, 13, 14, 15)$