# COMBINATIONAL LOGIC

# DECODER
# ENCODER

## DIGITAL LOGIC DESIGN

**Iqra Chaudhary (Lecturer CS dept. NUML)**

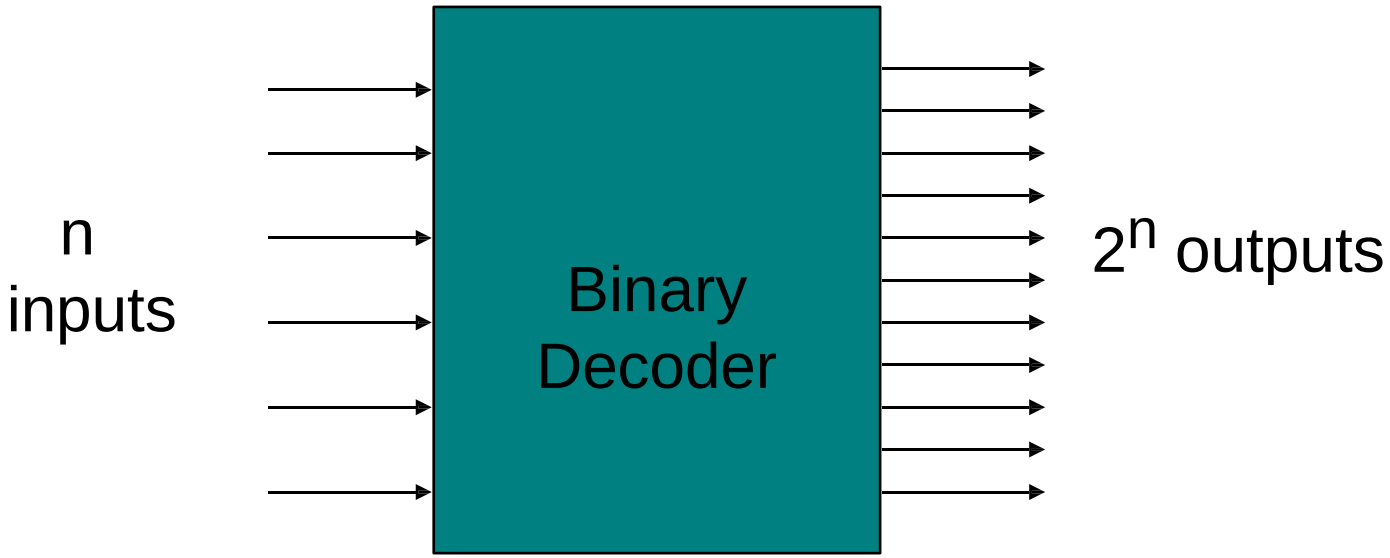
# Binary Decoder

Binary decoder has n input lines and $2^n$ output lines

Only one output is a 1 for any given input

Extract "*Information*" from the code

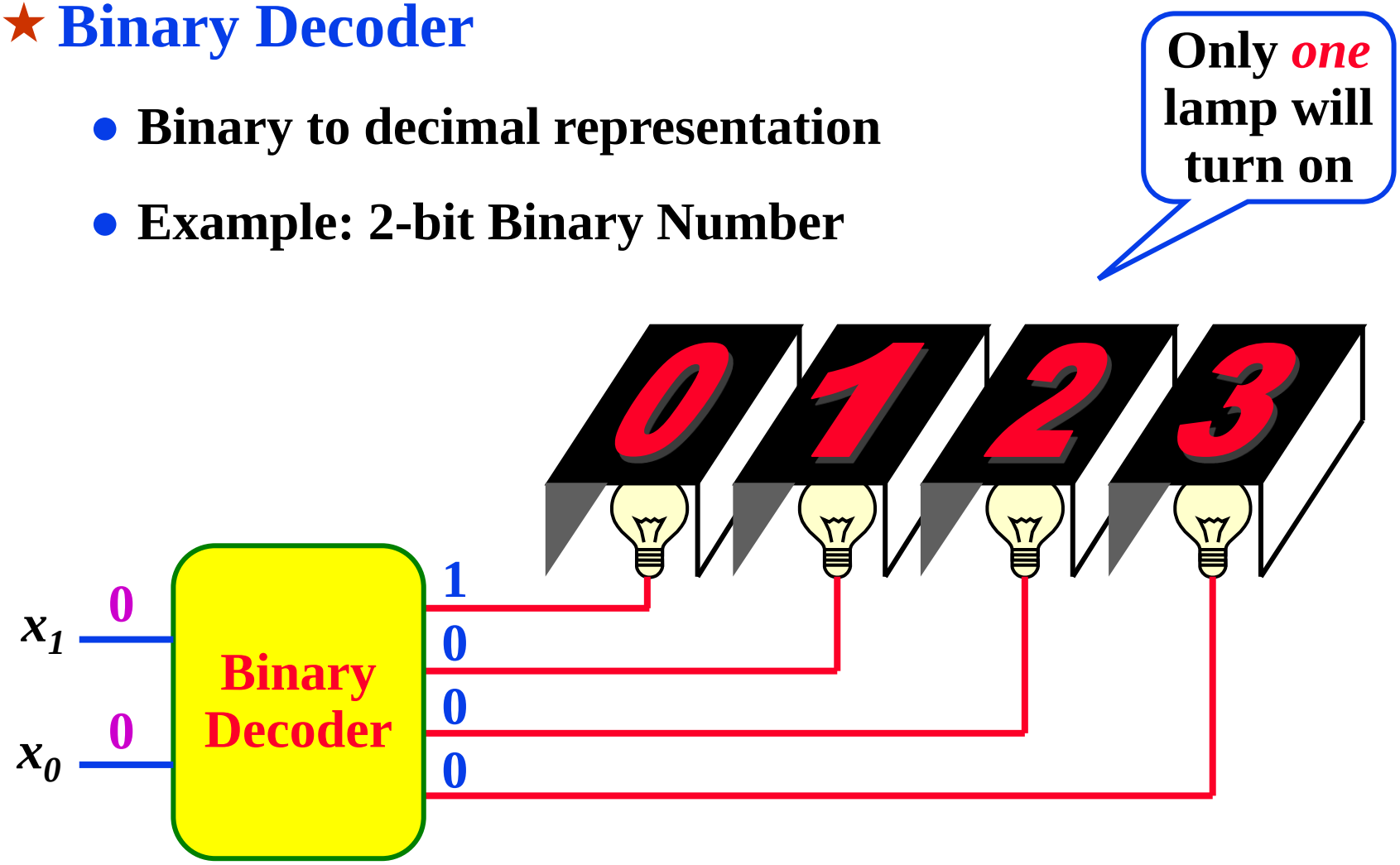
Can be used to implement logic circuits



# Decoders

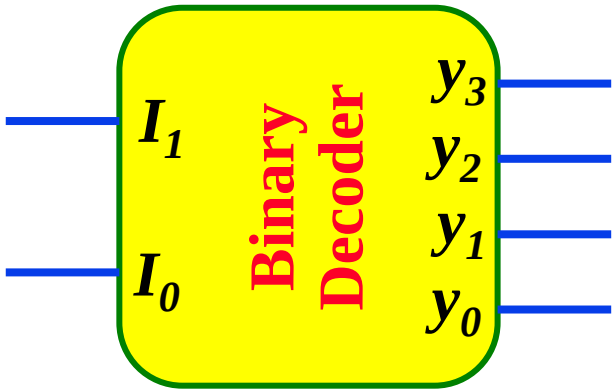
## ★ Binary Decoder

- Binary to decimal representation
- Example: 2-bit Binary Number

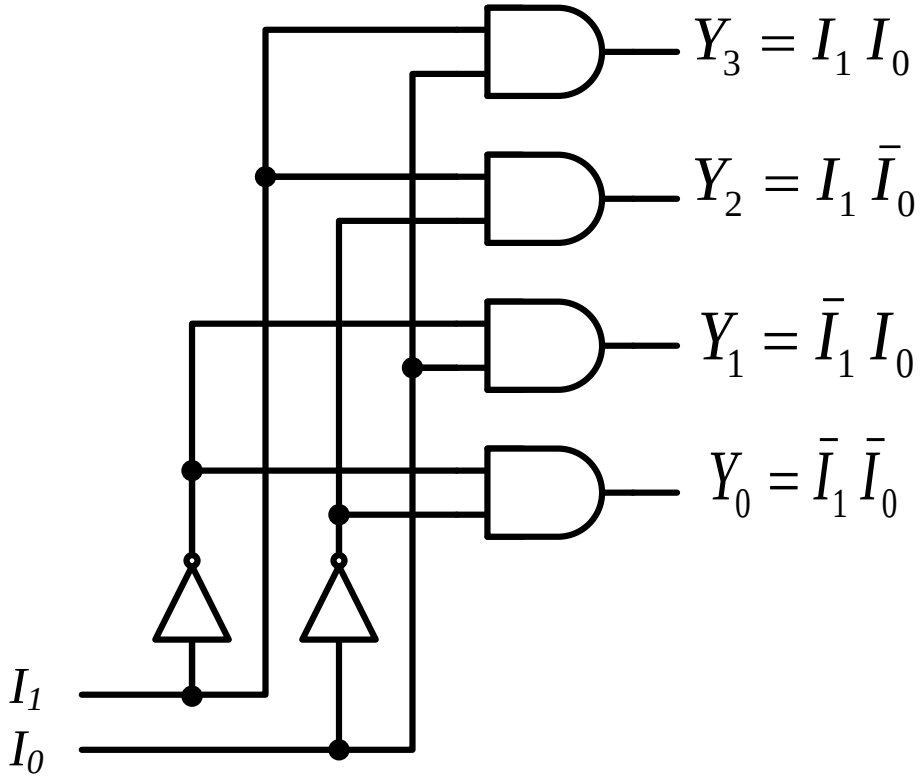


# Decoders

## ★ 2-to-4 Line Decoder




| $I_1$ | $I_0$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |




$$Y_3 = I_1 I_0 \qquad Y_2 = I_1 \bar{I}_0$$

$$Y_1 = \bar{I}_1 I_0 \qquad Y_0 = \bar{I}_1 \bar{I}_0$$

# Decoders

## ★ 3-to-8 Line Decoder

**Truth Table:**

| x | y | z | Y0 | Y1 | Y2 | Y3 | Y4 | Y5 | Y6 | Y7 |
|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

Y0 = x'y'z'

Y1 = x'y'z

Y2 = x'yz’

Y3 = x’yz

Y4 = xy'z’

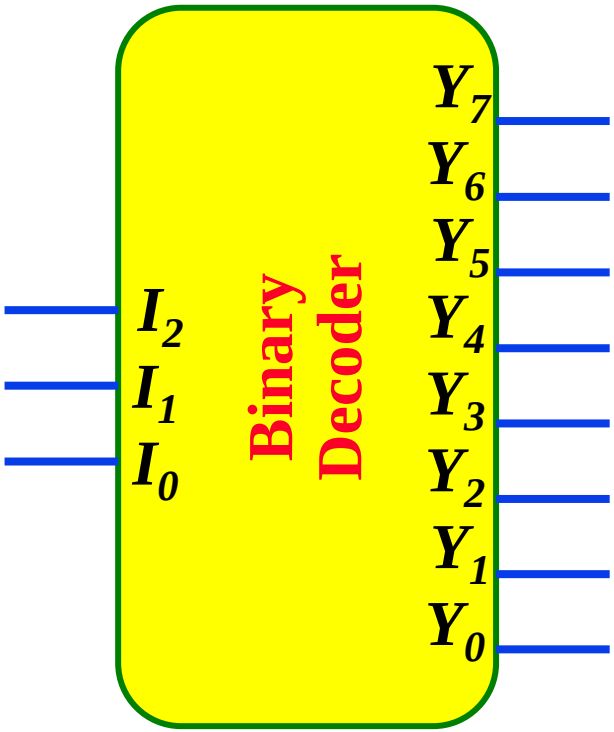
Y5 = xy'z

Y6 = xyz'

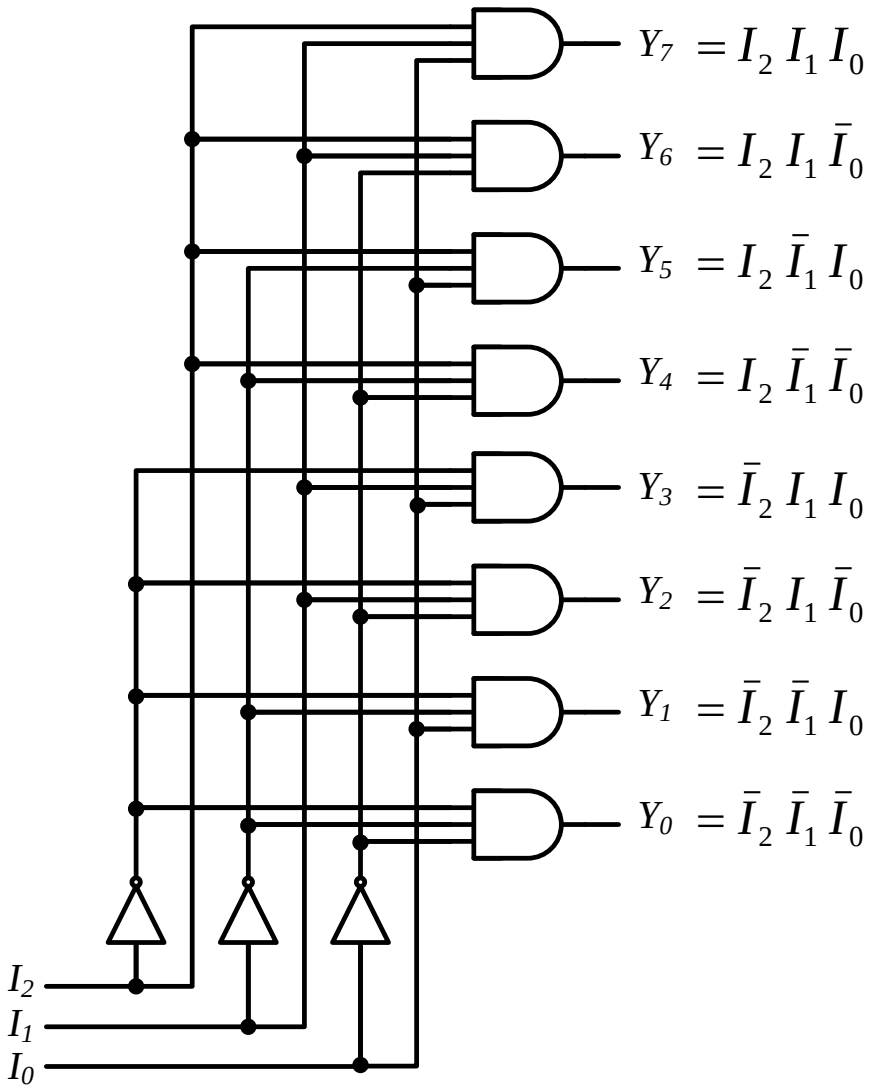
Y7 = xyz

# Decoders

## ★ 3-to-8 Line Decoder

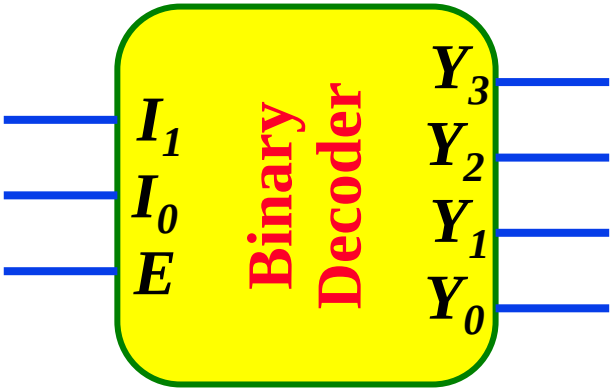



Each output is a minterm

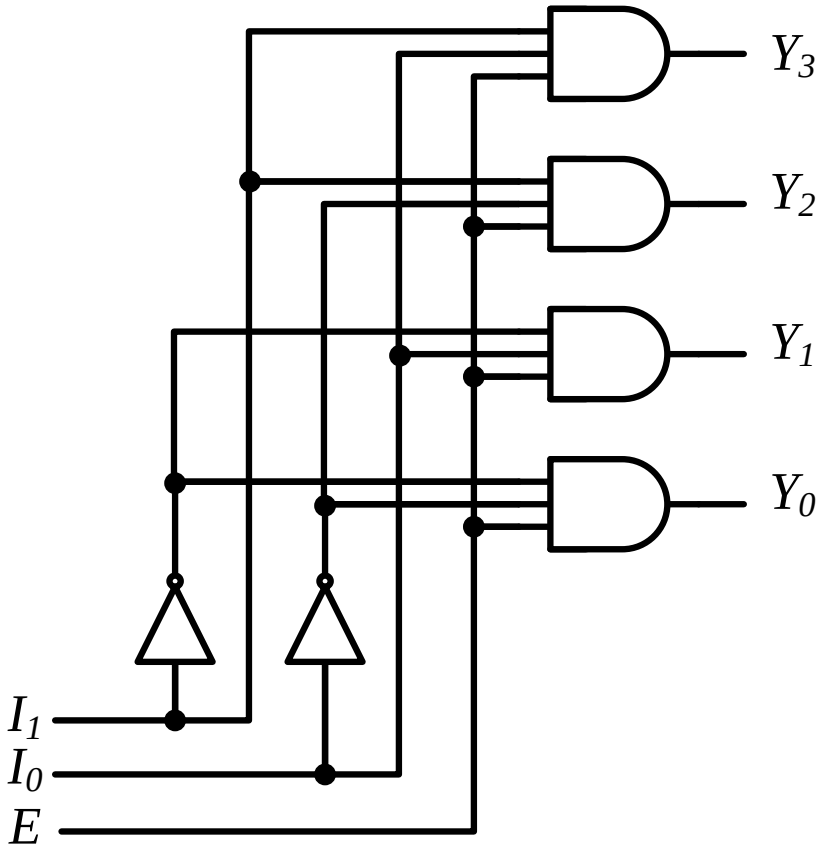


# Decoders

★ "*Enable*" Control



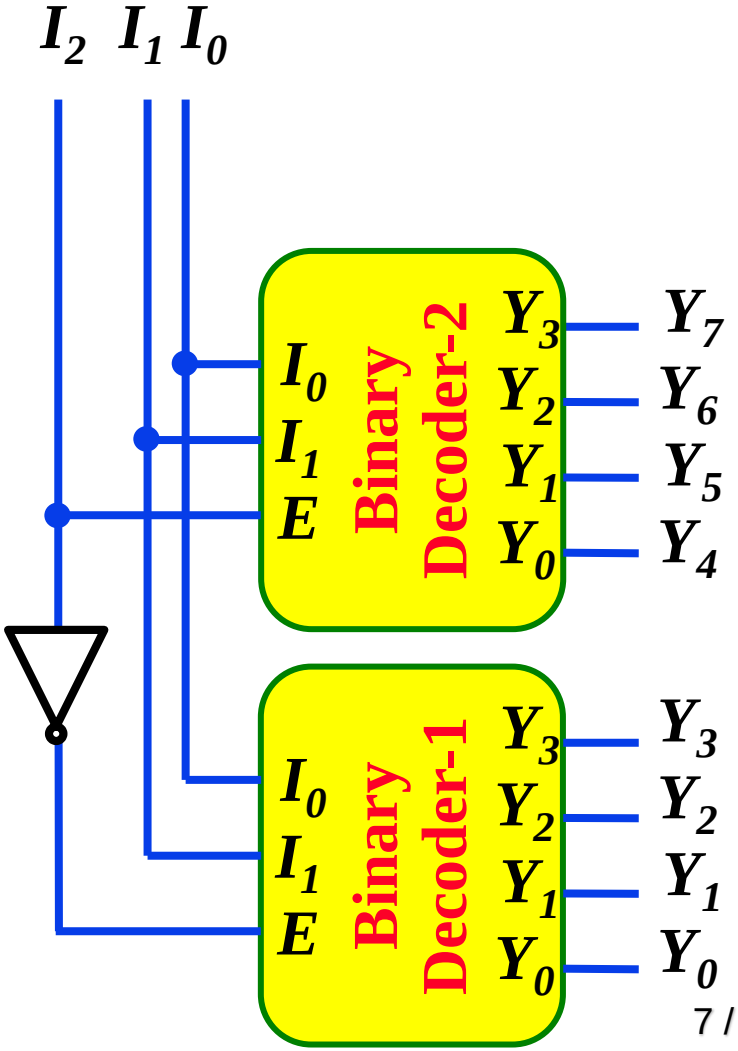

| $E$ | $I_1$ $I_0$ | $Y_3$ $Y_2$ $Y_1$ $Y_0$ |
|---|---|---|
| 0 | x x | 0 0 0 0 |
| 1 | 0 0 | 0 0 0 1 |
| 1 | 0 1 | 0 0 1 0 |
| 1 | 1 0 | 0 1 0 0 |
| 1 | 1 1 | 1 0 0 0 |



# Decoders

- ★ Use enable input to make two 2 to 4 decoders to make 3 to 8 decoder
- ★ In this example, only one decoder can be active at a time.

| $I_2$ | $I_1$ | $I_0$ | $Y_7$ | $Y_6$ | $Y_5$ | $Y_4$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |

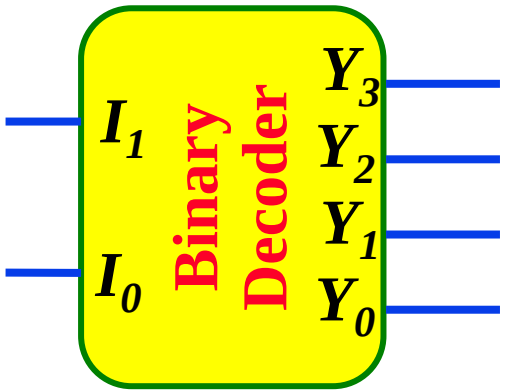


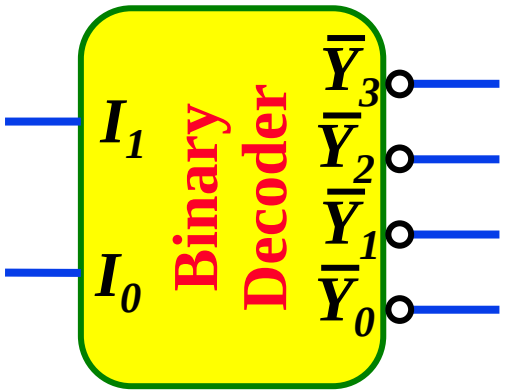
# Decoders using NAND gate

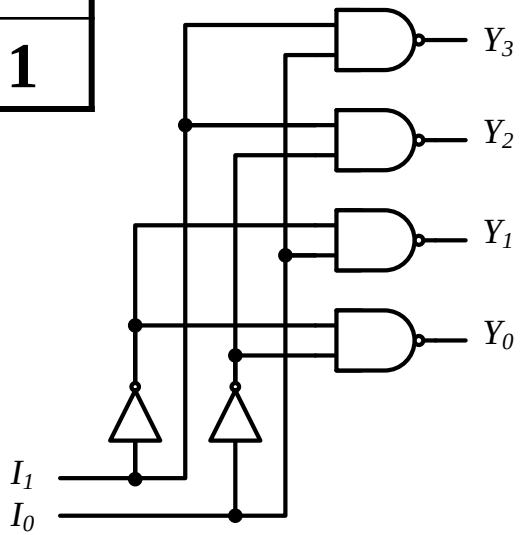
★ **Active-High/Active-Low(Note: use of NANDs only one 0 active!)**

| $I_1$ | $I_0$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

| $I_1$ | $I_0$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |



# Implement combinational circuit by Using Decoders

- ★ Each output is a minterm
- ★ All minterms are produced
- ★ Sum the required minterms

| x y z | | |
|---|---|---|
| $I_2\ I_1\ I_0$ | $S$ | $C$ |
| 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 |

**Example:** Full Adder

$$S(I_2,I_1,I_0) = \sum(1, 2, 4, 7) = \bar{I}_2\bar{I}_1I_0+\bar{I}_2I_1\bar{I}_0+I_2\bar{I}_1\bar{I}_0+I_2I_1I_0$$

$$C(I_2,I_1,I_0) = \sum(3, 5, 6, 7) = \bar{I}_2I_1I_0+I_2\bar{I}_1I_0+I_2I_1\bar{I}_0+I_2I_1I_0$$

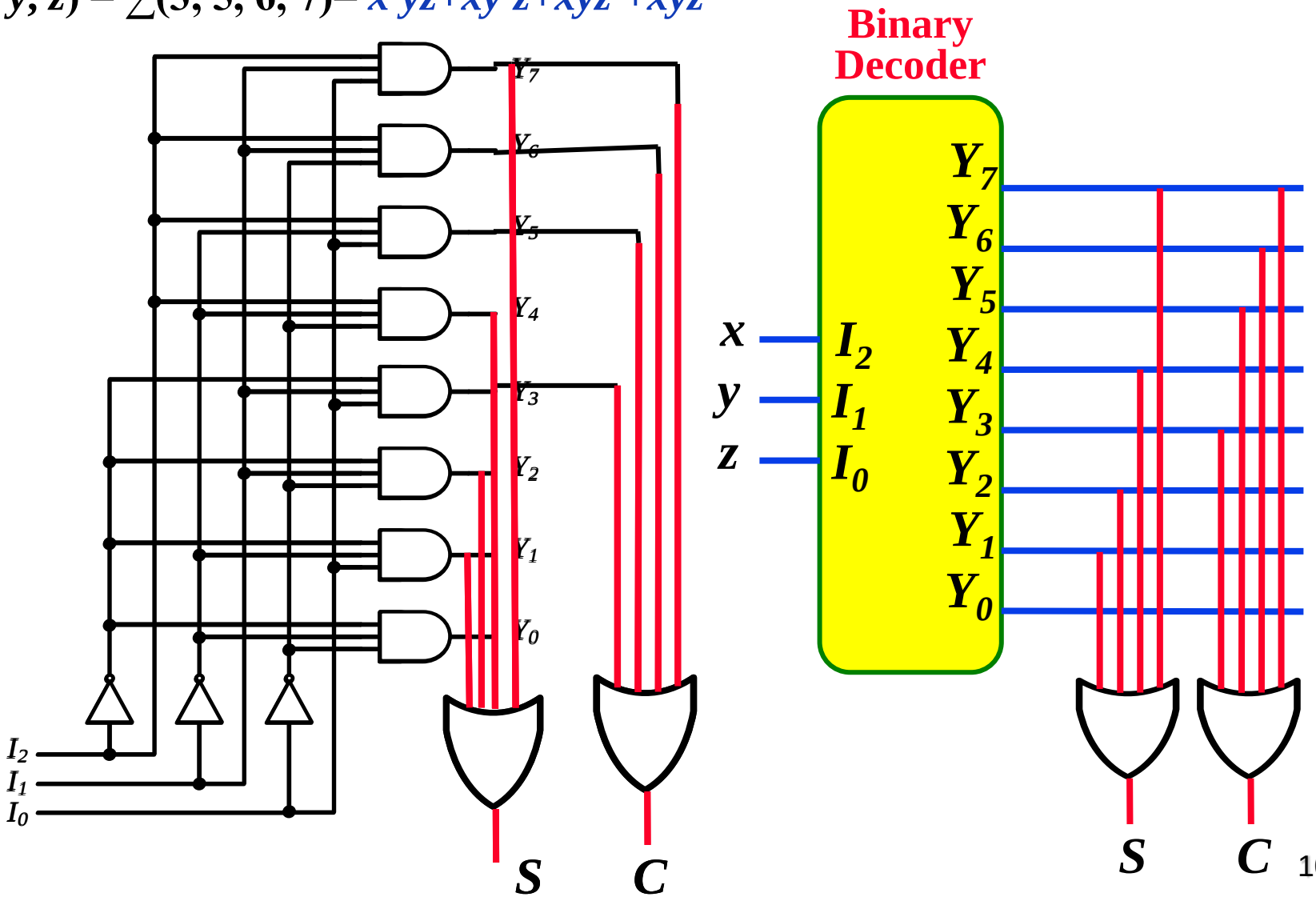
$$S(x,y,z) = x'y'z+x'yz'+xy'z'+xyz$$

$$C(x,y,z) = x'yz+xy'z+xyz'+xyz$$

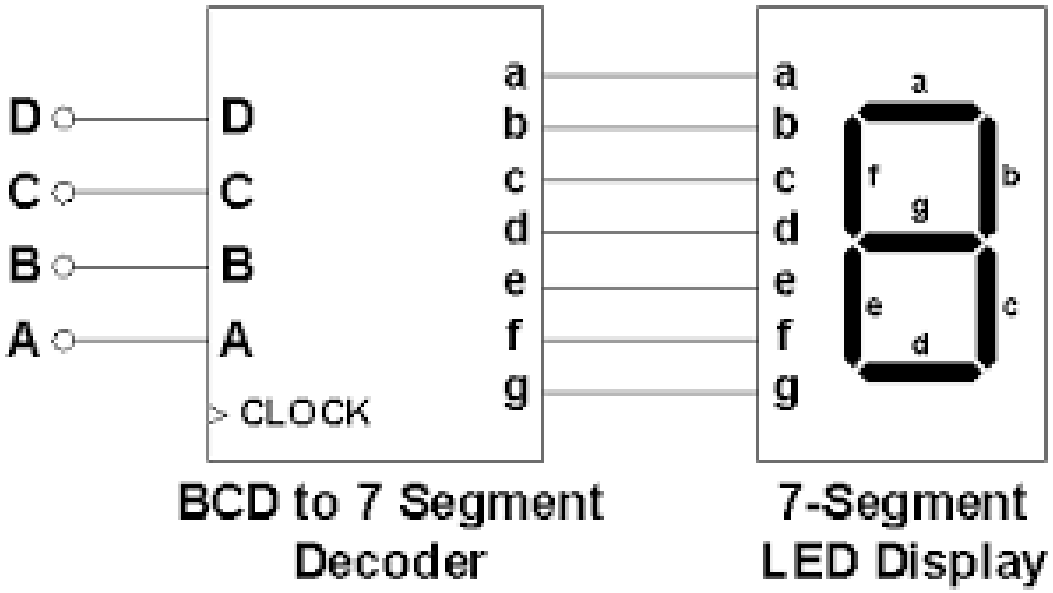
# Implement full adder by Using Decoders

$S(x, y, z) = \sum(1, 2, 4, 7) = xy'z'+x'yz'+x'y'z+xyz$

$C(x, y, z) = \sum(3, 5, 6, 7) = x'yz+xy'z+xyz'+xyz$

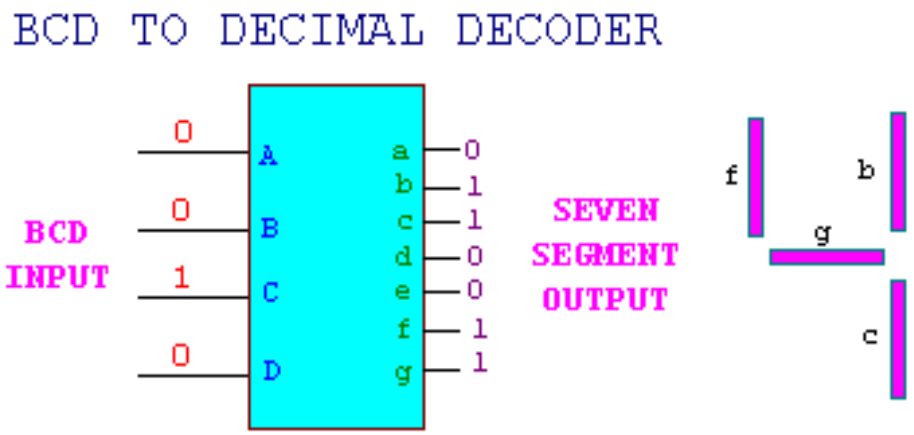


# BCD to Seven Segment Decoder



# BCD to Seven Segment Decoder

- Decoder inputs data in BCD form and converts it to a seven segment output.
- The IC does the decoding that is involved in activating the appropriate segment outputs (a-g) that is required to represent the binary number that is inputted.
- There are 4 binary inputs to the Decoder and seven output segments (a-g).




For example if the BCD number 0100 (4) is input into the Decoder, the Decoder must activate outputs (b, c, f, g) because they form the digit 4.

# Encoders

- ★ An encoder is a combinational logic circuit that essentially performs a "reverse" of decoder functions.
- ★ An encoder has $2^n$ number of input lines, only one of which input is activated at a given time and produces an n-bit output code , depending on which input is activated.
- ★ Put "*Information*" into code
- ★ An encoder accepts an active level on one of its inputs, representing digit, such as a decimal or octal digits, and converts it to a coded output such as BCD or binary. Encoders can also be devised to encode various symbols and alphabetic characters.

# Binary Encoders

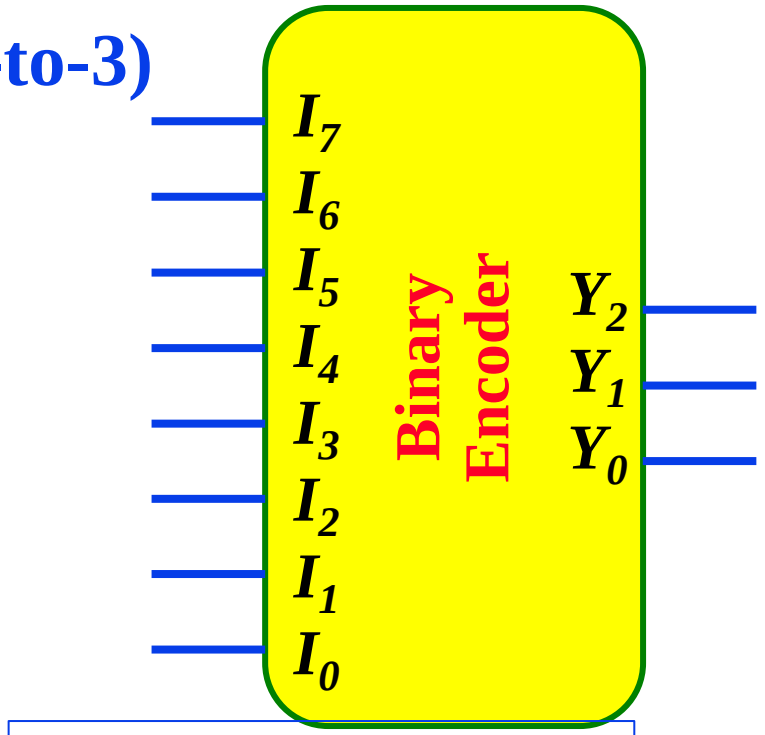
## ★ Octal-to-Binary Encoder (8-to-3)

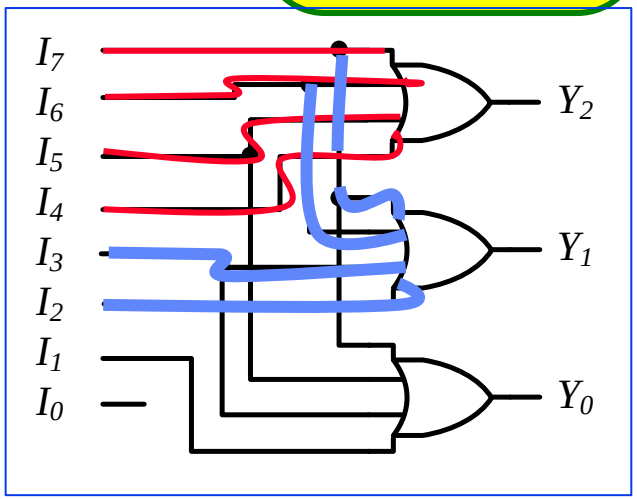
| $I_7$ | $I_6$ | $I_5$ | $I_4$ | $I_3$ | $I_2$ | $I_1$ | $I_0$ | $Y_2$ | $Y_1$ | $Y_0$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 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 |

$$Y_2 = I_7 + I_6 + I_5 + I_4$$

$$Y_1 = I_7 + I_6 + I_3 + I_2$$

$$Y_0 = I_7 + I_5 + I_3 + I_1$$

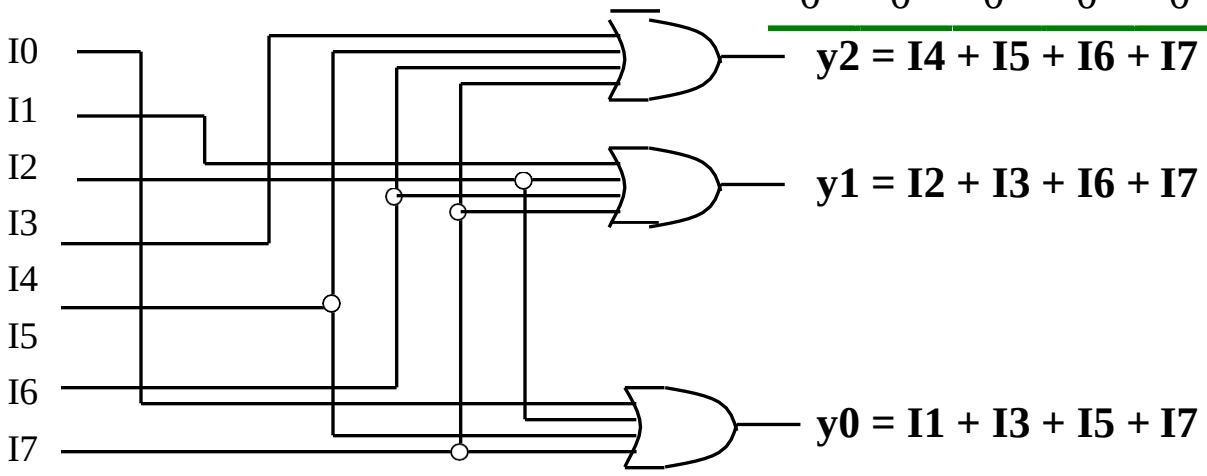


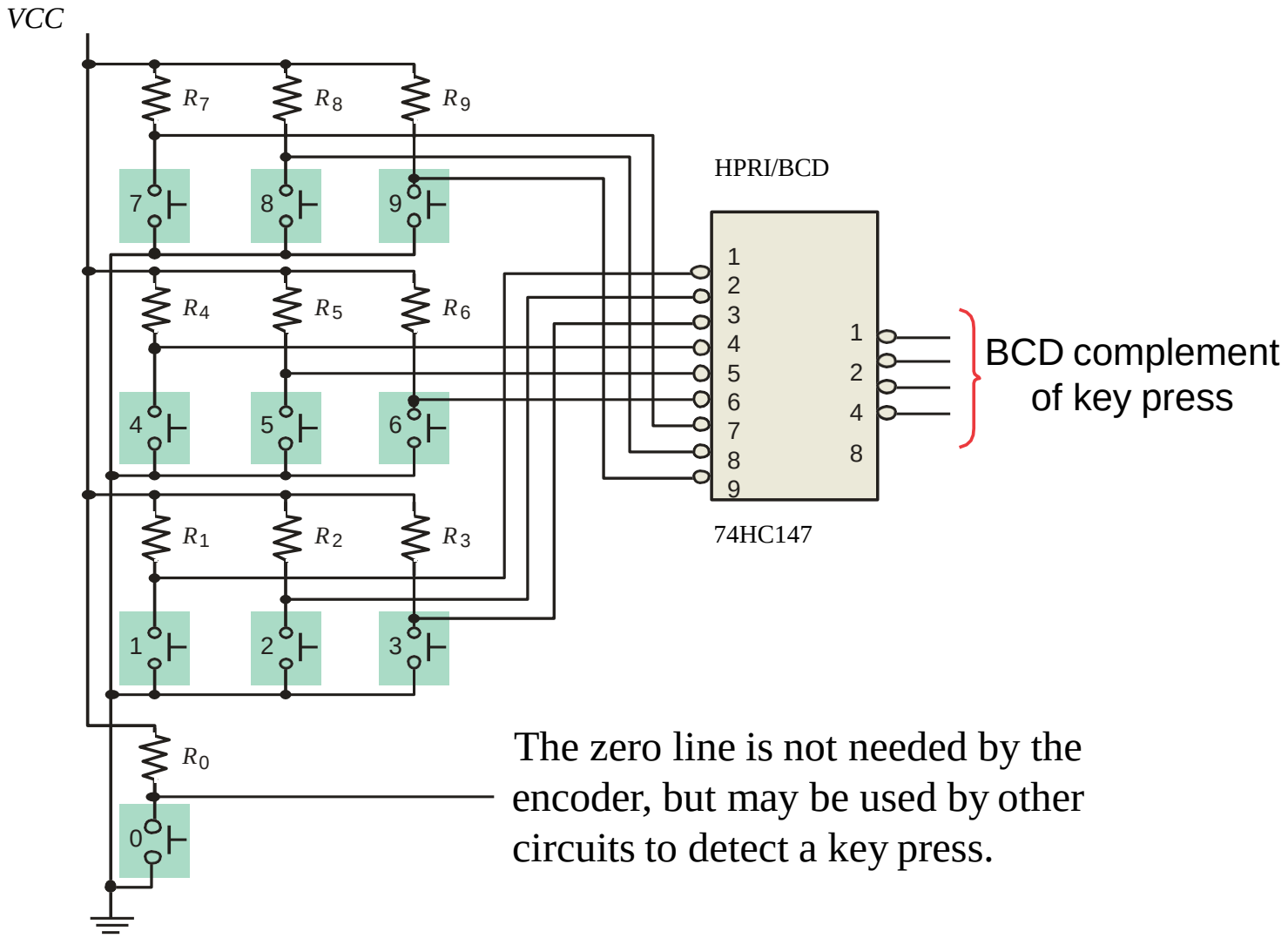
# Binary Encoders(repeat)

At any one time, only one input line has a value of 1.

| Inputs | | | | | | | | Outputs | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $I_0$ | $I_1$ | $I_2$ | $I_3$ | $I_4$ | $I_5$ | $I_6$ | $I_7$ | y2 | y1 | y0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |



# Keyboard encoder




The zero line is not needed by the encoder, but may be used by other circuits to detect a key press.

# Encoder / Decoder Pairs