# Machine Structure 02

# Chapter 01-part-02

# Combinatorial Circuits

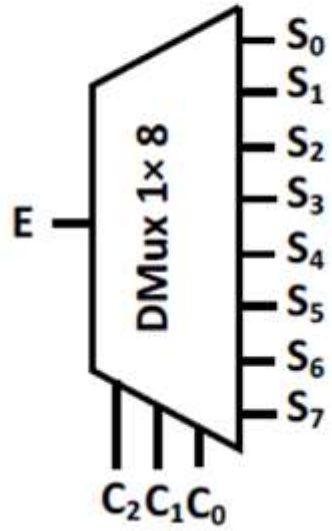
# Demultiplexers

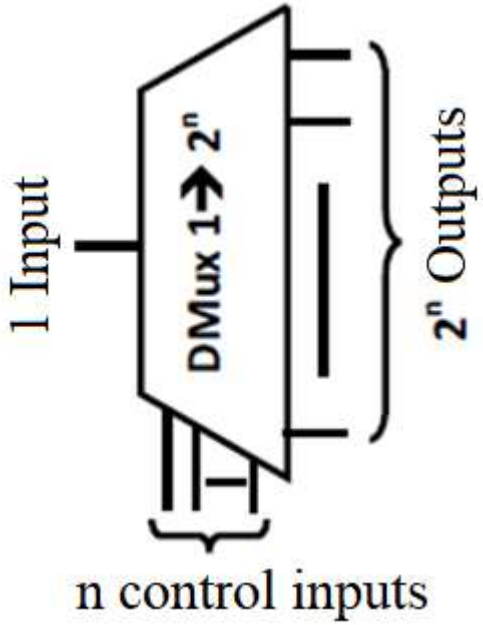
- Demultiplexer: It plays the reverse role of a multiplexer; it allows for passing information to one of the outputs based on the values of the control inputs. It has:
- **One** single input
- $2^n$ outputs
- **N** selection inputs (controls)

# Demultiplexers

- Definition: A demultiplexer is a combinational logic circuit that has a **1** single input, **n** control inputs, and **$2^n$** outputs, It allows routing the value of the input line to the output line indicated by its control inputs.
- Circuit synthesis (1x8 demultiplexer).

The Inputs/Outputs

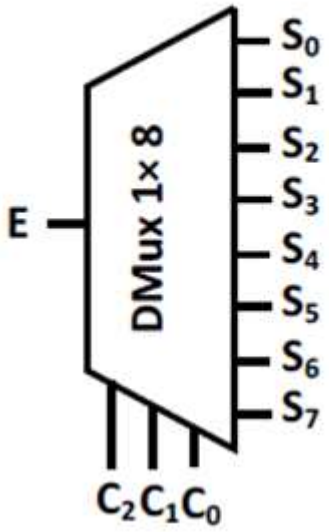


# Demultiplexers

- Definition: A demultiplexer is a combinational logic circuit that has a **1** single input, **n** control inputs, and **$2^n$** outputs, It allows routing the value of the input line to the output line indicated by its control inputs.
- Circuit synthesis (1x8 demultiplexer).

The Inputs/Outputs



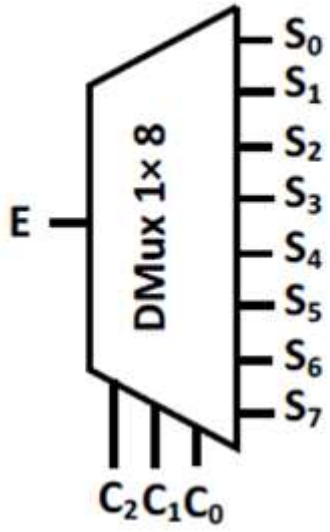

The truth table

| $C_2$ | $C_1$ | $C_0$ | $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | E | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | E | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | E | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | E | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | E | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | E | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | E | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | E |

# Demultiplexers

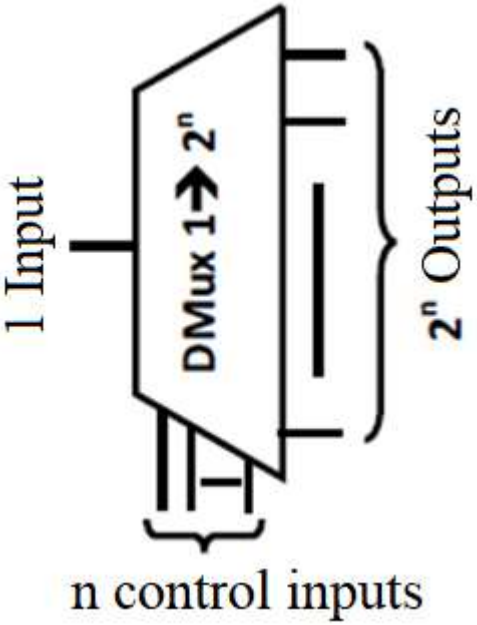
- Definition: A demultiplexer is a combinational logic circuit that has a **1** single input, **n** control inputs, and **$2^n$** outputs, It allows routing the value of the input line to the output line indicated by its control inputs.
- Circuit synthesis (1x8 demultiplexer).

The Inputs/Outputs




The truth table

| $C_2$ | $C_1$ | $C_0$ | $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | E | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | E | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | E | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | E | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | E | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | E | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | E | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | E |



# Demultiplexers

## The truth table

| $C_2$ | $C_1$ | $C_0$ | $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | E | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | E | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | E | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | E | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | E | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | E | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | E | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | E |

## Logical Fonctions

$$S_0 = \overline{C_2}.\overline{C_1}.\overline{C_0}.E$$

$$S_1 = \overline{C_2}.\overline{C_1}.C_0.E$$

$$S_2 = \overline{C_2}.C_1.\overline{C_0}.E$$

$$S_3 = \overline{C_2}.C_1.C_0.E$$

$$S_4 = C_2.\overline{C_1}.\overline{C_0}.E$$

$$S_5 = C_2.\overline{C_1}.C_0.E$$

$$S_6 = C_2.C_1.\overline{C_0}.E$$

$$S_7 = C_2.C_1.C_0.E$$

# Demultiplexers

Logical Fonctions

$$S_0 = \overline{C_2}.\overline{C_1}.\overline{C_0}.E$$

$$S_1 = \overline{C_2}.\overline{C_1}.C_0.E$$

$$S_2 = \overline{C_2}.C_1.\overline{C_0}.E$$

$$S_3 = \overline{C_2}.C_1.C_0.E$$

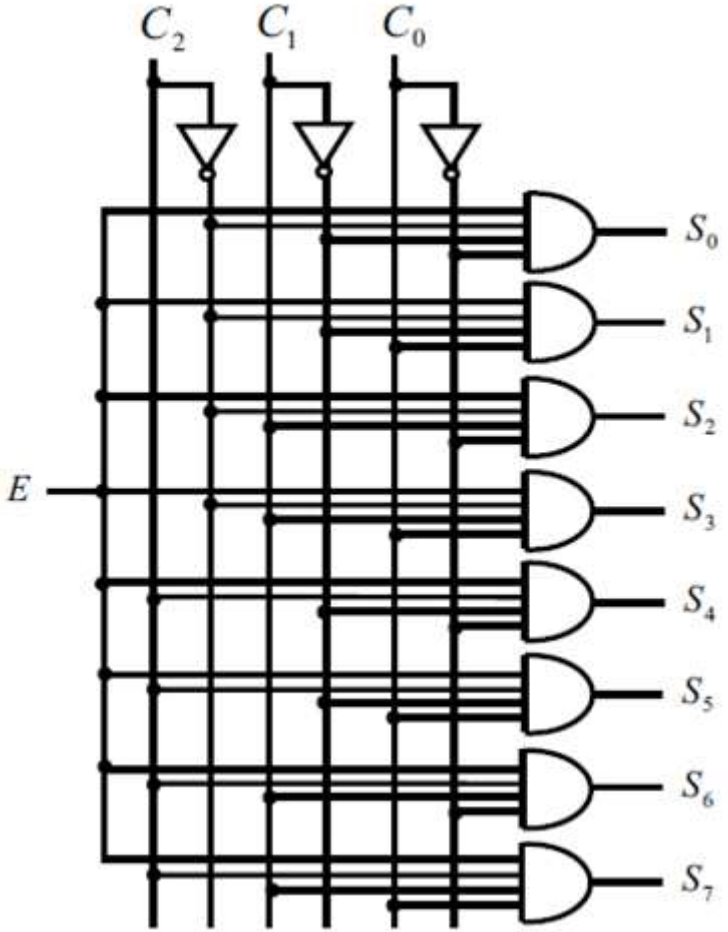
$$S_4 = C_2.\overline{C_1}.\overline{C_0}.E$$

$$S_5 = C_2.\overline{C_1}.C_0.E$$

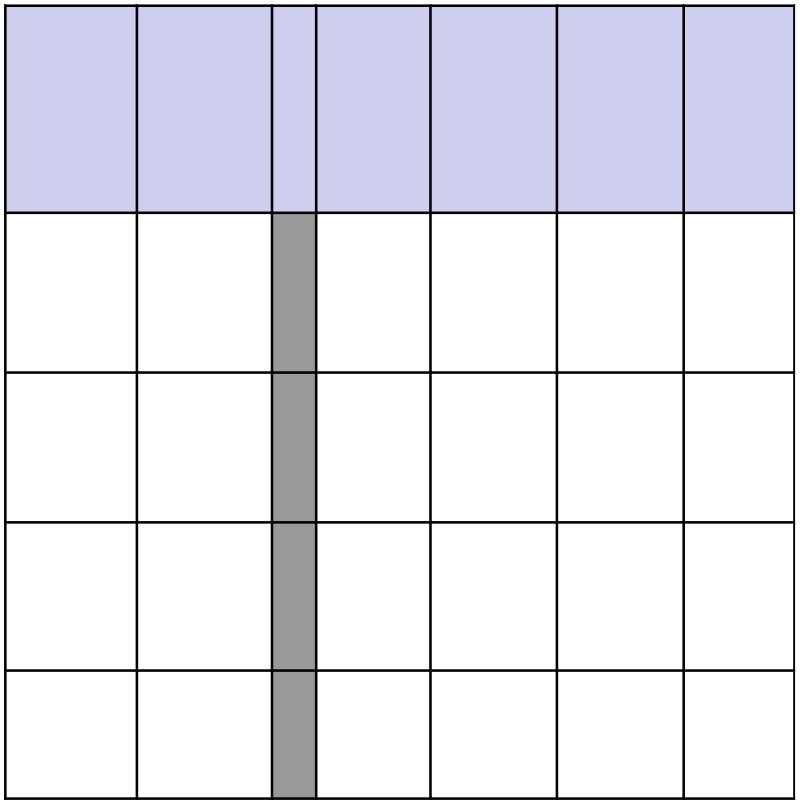
$$S_6 = C_2.C_1.\overline{C_0}.E$$

$$S_7 = C_2.C_1.C_0.E$$

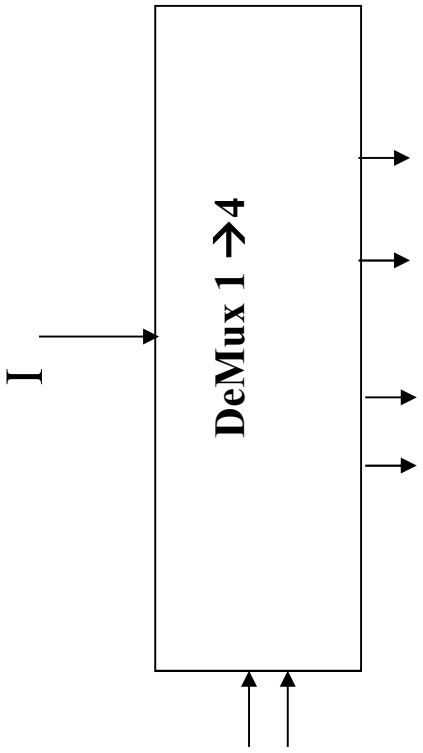
The circuit diagram



# Demultiplexers **1→4**



- Implementing a demultiplexer circuit. **1→4**



# Demultiplexers 1→4

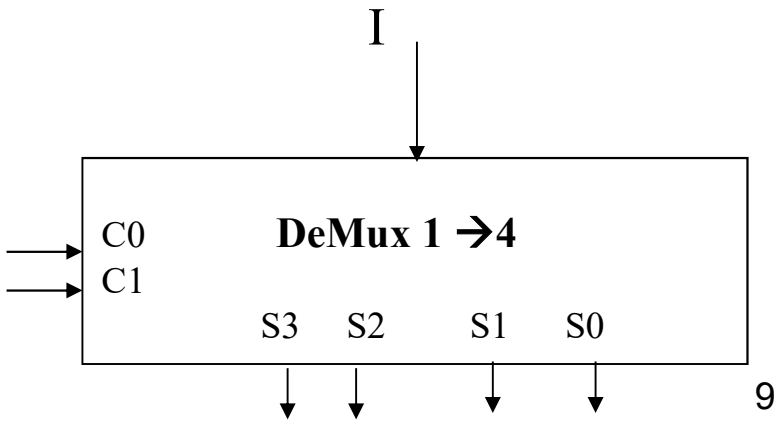
| C1 | C0 | | S3 | S2 | S1 | S0 |
|---|---|---|---|---|---|---|
| 0 | 0 | | 0 | 0 | 0 | i |
| 0 | 1 | | 0 | 0 | i | 0 |
| 1 | 0 | | 0 | i | 0 | 0 |
| 1 | 1 | | i | 0 | 0 | 0 |

$$S0 = \overline{C1}.\overline{C0}.(I)$$

$$S1 = \overline{C1}.C0.(I)$$

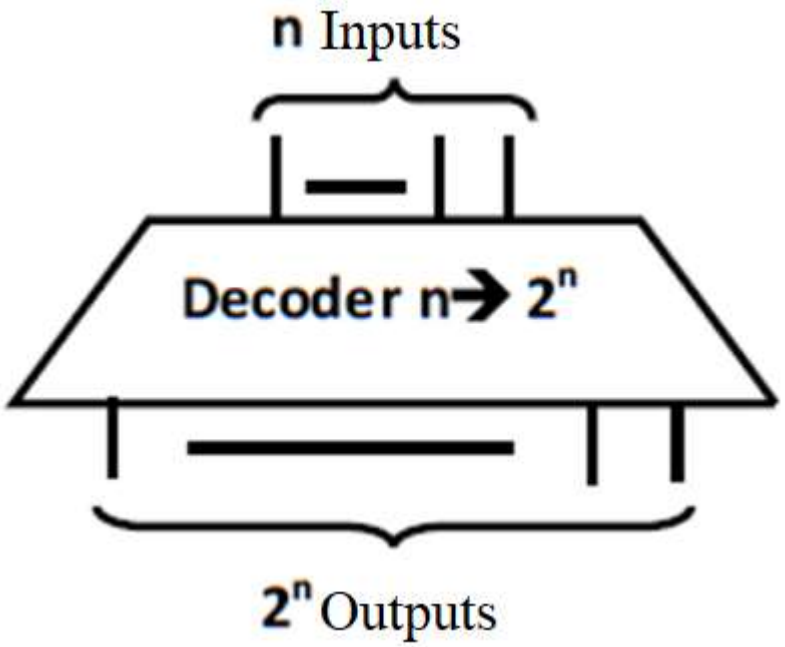
$$S2 = C1.\overline{C0}.(I)$$

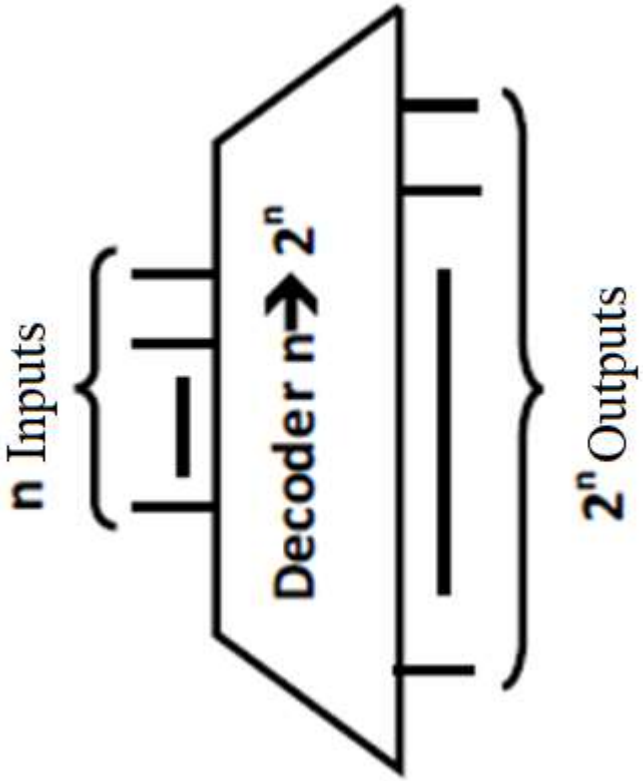
$$S3 = C1.C0.(I)$$



# Binary Decoder

Definition: A n-bit decoder is a combinational logic circuit with **n** inputs and **$2^n$** outputs. It enables the activation of the output line corresponding to the binary code present at the input.

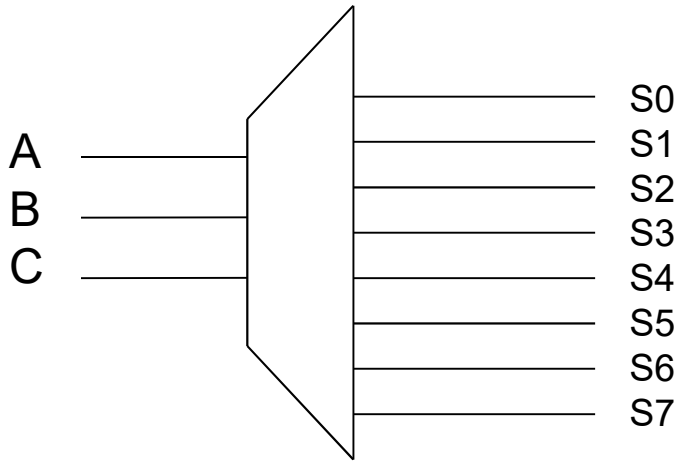


# Binary Decoder

It is a combinational circuit composed of:

- **N**: data inputs
- **$2^n$**: outputs
- For each input combination, only one output is active at a time.




**Exemple: A decoder 3→8**

# Binary Decoder

- Decoder with Enable Input Most decoders will have an enable input (EN).

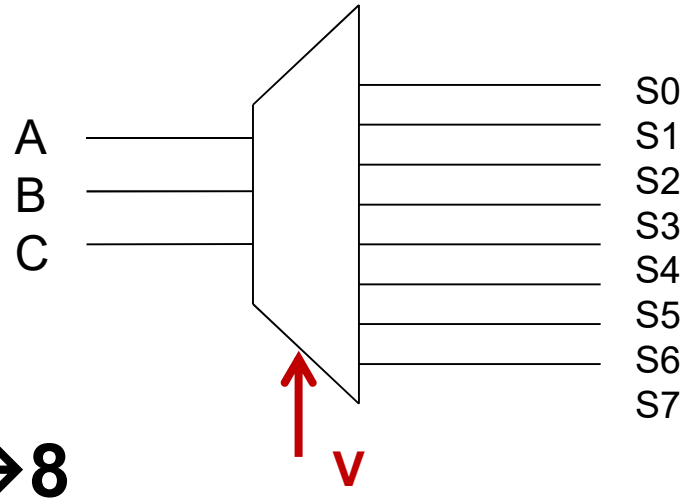
This input enables the operation of the decoder.

- If EN = 0, all outputs are set to 0.
- If EN = 1, the decoder operates normally.

We can also have:

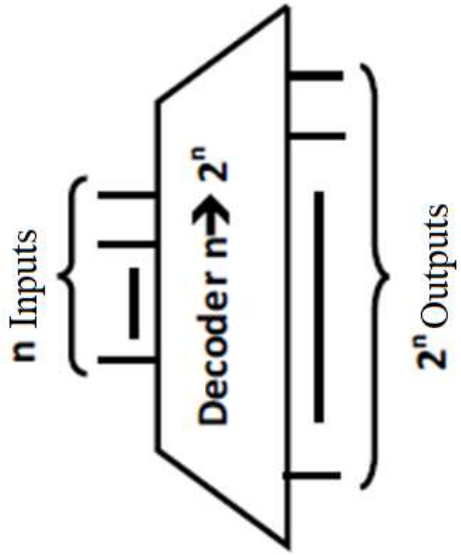
an inverted control

signal (EN).



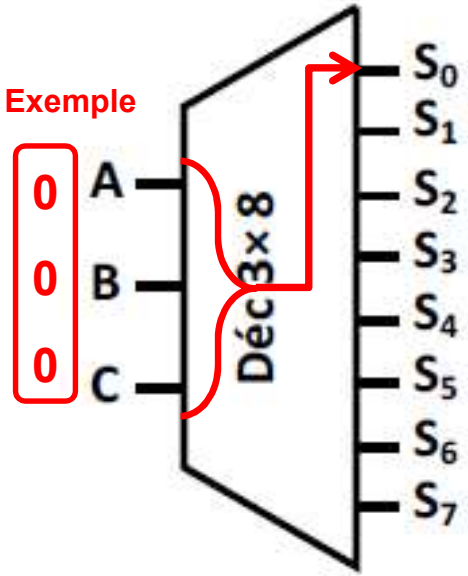

**Exemple: Un decoder 3→8**

# Binary Decoder




Circuit synthesis (3x8 decoder).

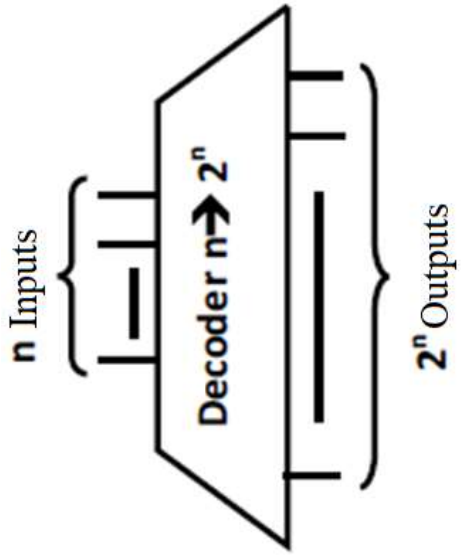
The Inputs/Outputs




The truth table

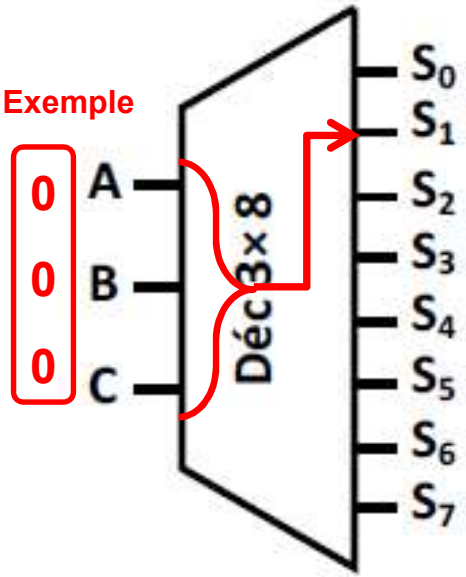
| A | B | C | $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 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 |

# Binary Decoder




Circuit synthesis (3x8 decoder).

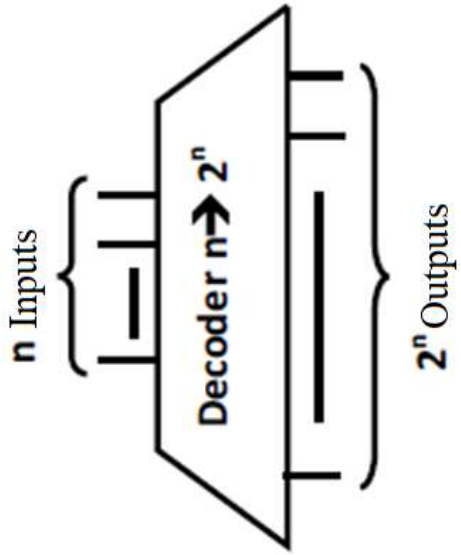
The Inputs/Outputs




The truth table

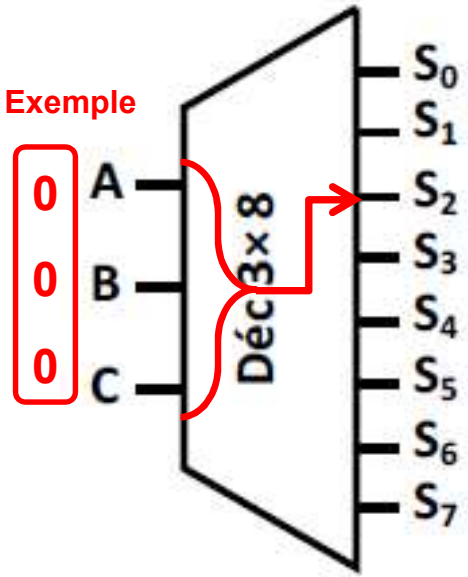
| A | B | C | $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **0** | **0** | **1** | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 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 |

# Binary Decoder




Circuit synthesis (3x8 decoder).

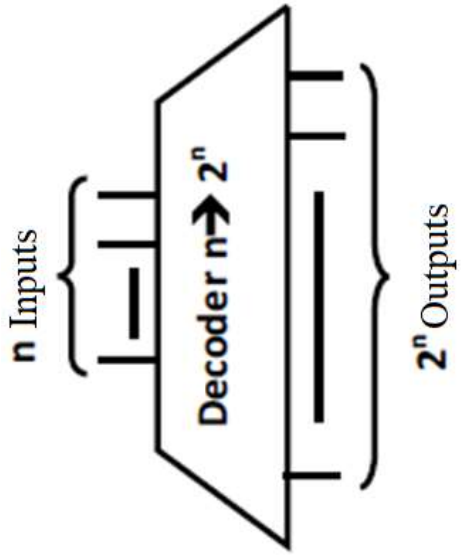
The Inputs/Outputs




The truth table

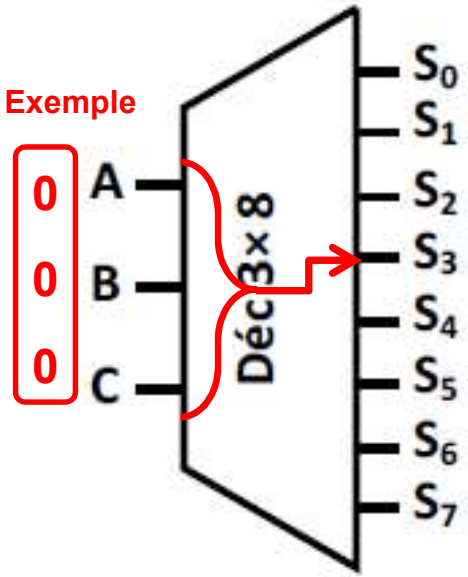
| A | B | C | $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **0** | **1** | **0** | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 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 |

# Binary Decoder




Circuit synthesis (3x8 decoder).

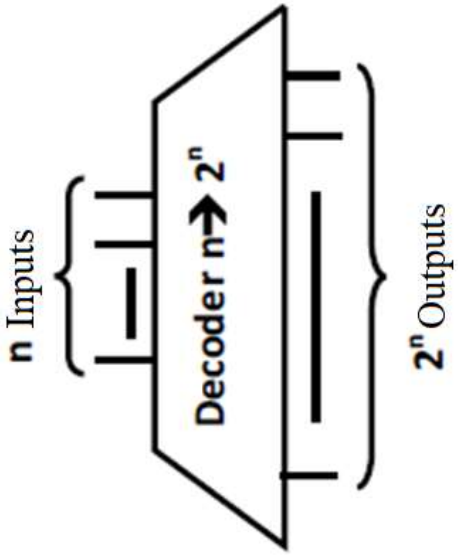
The Inputs/Outputs




The truth table

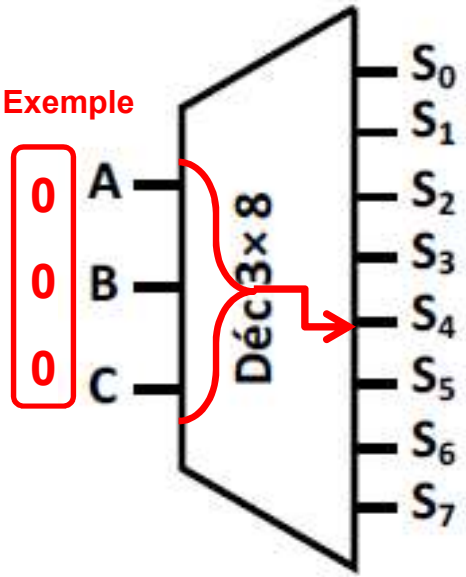
| A | B | C | $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **0** | **1** | **1** | 0 | 0 | 0 | **1** | 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 |

# Binary Decoder




Circuit synthesis (3x8 decoder).

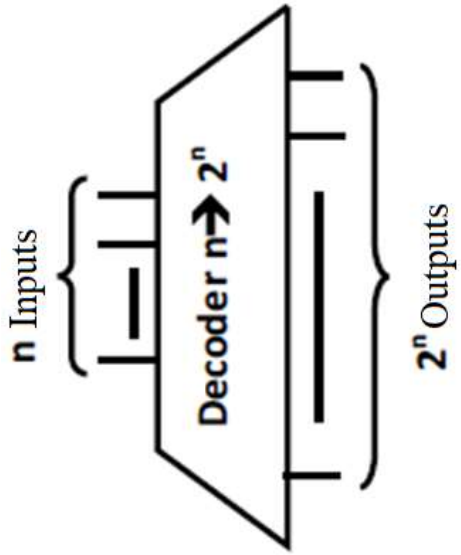
The Inputs/Outputs




The truth table

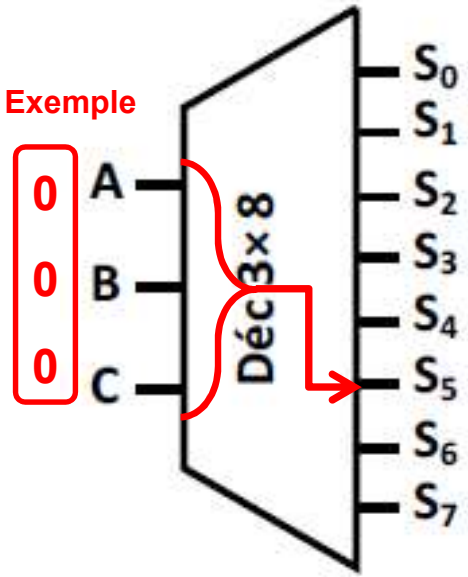
| A | B | C | $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 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 |

# Binary Decoder




Circuit synthesis (3x8 decoder).

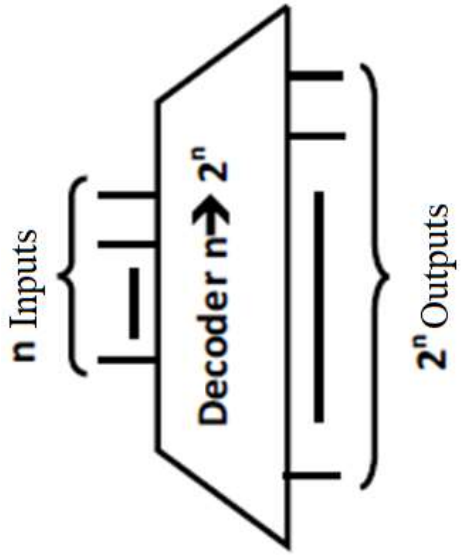
The Inputs/Outputs




The truth table

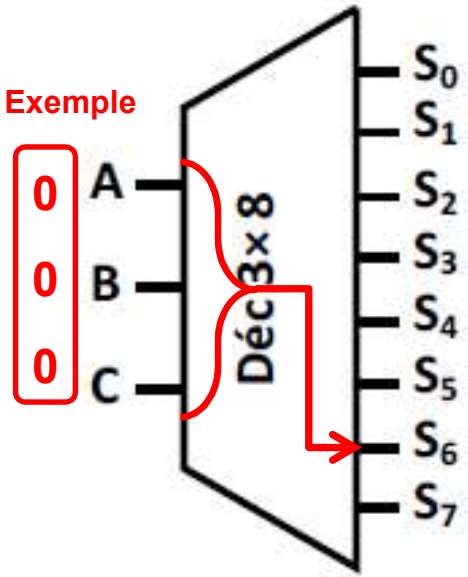
| A | B | C | $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 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 |

# Binary Decoder




Circuit synthesis (3x8 decoder).

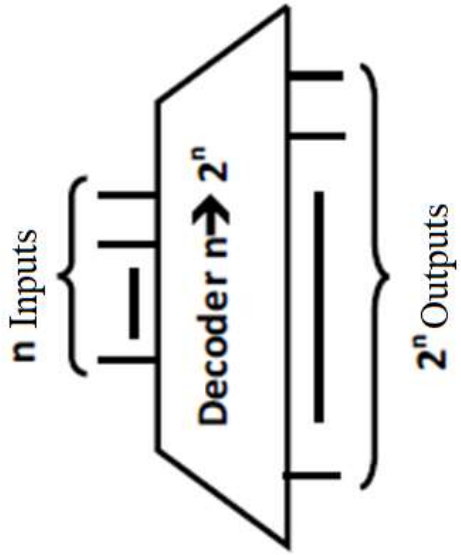
The Inputs/Outputs




The truth table

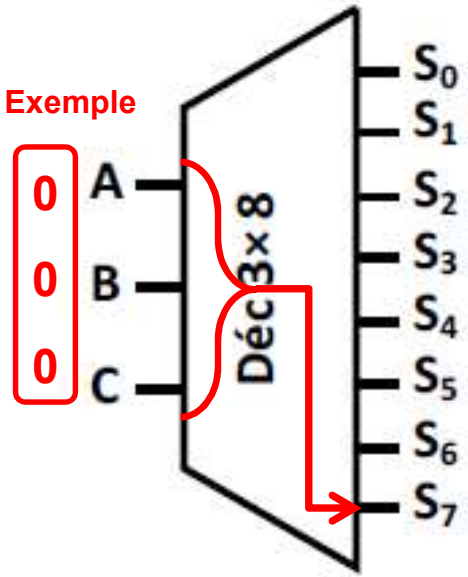
| A | B | C | $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 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 |

# Binary Decoder




## Circuit synthesis (3x8 decoder).

### The Inputs/Outputs




### The truth table

| A | B | C | $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

# Binary Decoder

Circuit synthesis (3x8 decoder).

The truth table

| A | B | C | $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

Logical expression

$$S_0 = \overline{A}.\overline{B}.\overline{C}$$

$$S_1 = \overline{A}.\overline{B}.C$$

$$S_2 = \overline{A}.B.\overline{C}$$

$$S_3 = \overline{A}.B.C$$

$$S_4 = A.\overline{B}.\overline{C}$$

$$S_5 = A.\overline{B}.C$$

$$S_6 = A.B.\overline{C}$$

$$S_7 = A.B.C$$

# Binary Decoder

Logical expression

$S_0 = \overline{A}.\overline{B}.\overline{C}$

$S_1 = \overline{A}.\overline{B}.C$

$S_2 = \overline{A}.B.\overline{C}$

$S_3 = \overline{A}.B.C$

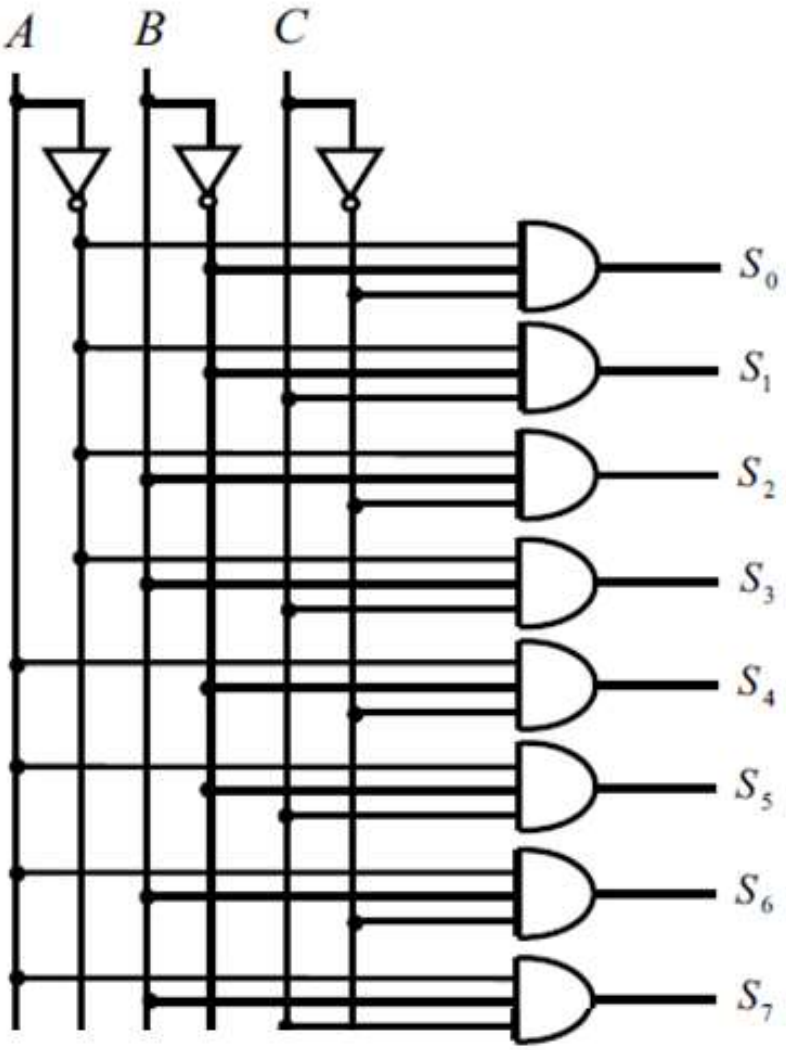
$S_4 = A.\overline{B}.\overline{C}$

$S_5 = A.\overline{B}.C$

$S_6 = A.B.\overline{C}$

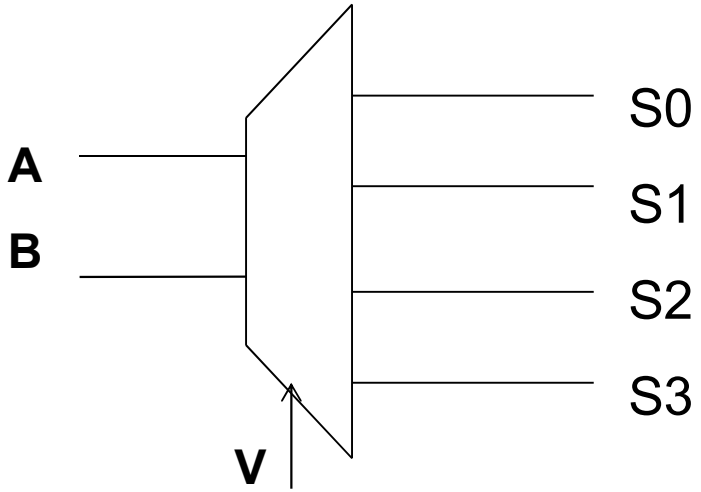
$S_7 = A.B.C$

The diagram circuit.



# 2→4 decoder with enable signal.

| V | A | B | | S0 | S1 | S2 | S3 |
|---|---|---|---|---|---|---|---|
| 0 | X | X | | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | | 0 | 0 | 0 | 1 |




$$S_0 = (\overline{A}.\overline{B}).V$$

$$S_1 = (\overline{A}.B).V$$

$$S_2 = (A.\overline{B}).V$$

$$S_3 = (A.B).V$$

# Logical Functions via Decoders:

- It is also possible to generate arbitrary logical functions using decoders and basic logic gates. Simply connect the variables of the function to be generated to the inputs of the decoder and connect the outputs corresponding to the different minterms of the function to the inputs of one or more basic logic gates.
- Example 1:
- Implement the function: $F(A, B, C) = \bar{B}C + \bar{A}B$
- using an 8x1 decoder.

# Logical Functions via Decoders:

Example 1: Implement $F(A,B,C)=\bar{B}C+\bar{A}B$ the function using an 8x1 decoder.

$F(A,B,C)=\bar{B}C+\bar{A}B$

# Logical Functions via Decoders:

Example 1: Implement $F(A,B,C)=\bar{B}C+\bar{A}B$ the function using an 8x1 decoder.

$$F(A,B,C)=\bar{B}C+\bar{A}B$$

$$F(A,B,C)=\bar{B}C(A+\bar{A})+\bar{A}B(C+\bar{C})$$

# Logical Functions via Decoders:

Example 1: Implement $F(A,B,C)=\bar{B}C+\bar{A}B$ the function using an 8x1 decoder.

$$F(A,B,C)=\bar{B}C+\bar{A}B$$

$$F(A,B,C)=\bar{B}C(A+\bar{A})+\bar{A}B(C+\bar{C})$$

$$F(A,B,C)=(\bar{A}\bar{B}C+A\bar{B}C)+(\bar{A}B\bar{C}+\bar{A}BC)$$

# Logical Functions via Decoders:

Example 1: Implement $F(A,B,C)=\bar{B}C+\bar{A}B$ the function using an 8x1 decoder.

$$F(A,B,C)=\bar{B}C+\bar{A}B$$

$$F(A,B,C)=\bar{B}C(A+\bar{A})+\bar{A}B(C+\bar{C})$$

$$F(A,B,C)=(\bar{A}\bar{B}C+A\bar{B}C)+(\bar{A}B\bar{C}+\bar{A}BC)$$

$$F(A,B,C)=\bar{A}\bar{B}C+\bar{A}B\bar{C}+\bar{A}BC+A\bar{B}C$$

# Logical Functions via Decoders:

Example 1: Implement $F(A,B,C)=\bar{B}C+\bar{A}B$ the function using an 8x1 decoder.

$$F(A,B,C)=\bar{B}C+\bar{A}B$$

$$F(A,B,C)=\bar{B}C(A+\bar{A})+\bar{A}B(C+\bar{C})$$

$$F(A,B,C)=(\bar{A}\bar{B}C+A\bar{B}C)+(\bar{A}B\bar{C}+\bar{A}BC)$$

$$F(A,B,C)=\bar{A}\bar{B}C+\bar{A}B\bar{C}+\bar{A}BC+A\bar{B}C$$

| A | B | C | F(A,B,C) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

# Logical Functions via Decoders:

Example 1: Implement $F(A,B,C)=\bar{B}C+\bar{A}B$ the function using an 8x1 decoder.

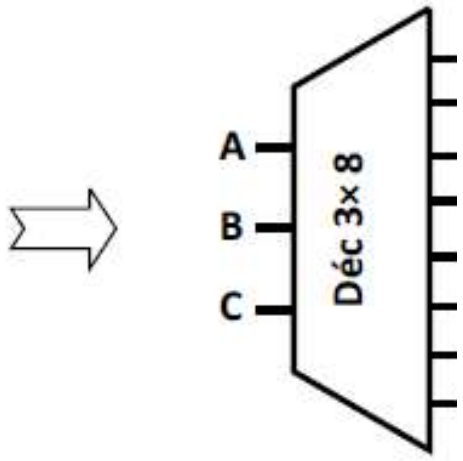
$$F(A,B,C)=\bar{B}C+\bar{A}B$$

$$F(A,B,C)=\bar{B}C(A+\bar{A})+\bar{A}B(C+\bar{C})$$

$$F(A,B,C)=(\bar{A}\bar{B}C+A\bar{B}C)+(\bar{A}B\bar{C}+\bar{A}BC)$$

$$F(A,B,C)=\bar{A}\bar{B}C+\bar{A}B\bar{C}+\bar{A}BC+A\bar{B}C$$

| A | B | C | F(A,B,C) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |



# Logical Functions via Decoders:

Example 1: Implement $F(A,B,C)=\bar{B}C+\bar{A}B$ the function using an 8x1 decoder.

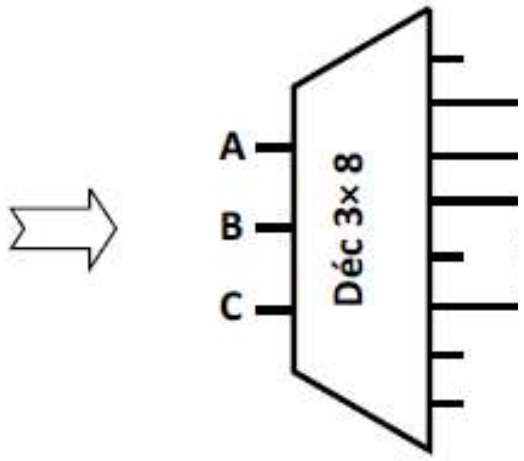
$$F(A,B,C)=\bar{B}C+\bar{A}B$$

$$F(A,B,C)=\bar{B}C(A+\bar{A})+\bar{A}B(C+\bar{C})$$

$$F(A,B,C)=(\bar{A}\bar{B}C+A\bar{B}C)+(\bar{A}B\bar{C}+\bar{A}BC)$$

$$F(A,B,C)=\bar{A}\bar{B}C+\bar{A}B\bar{C}+\bar{A}BC+A\bar{B}C$$

| A | B | C | F(A,B,C) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |



# Logical Functions via Decoders:

Example 1: Implement $F(A,B,C)=\bar{B}C+\bar{A}B$ the function using an 8x1 decoder.

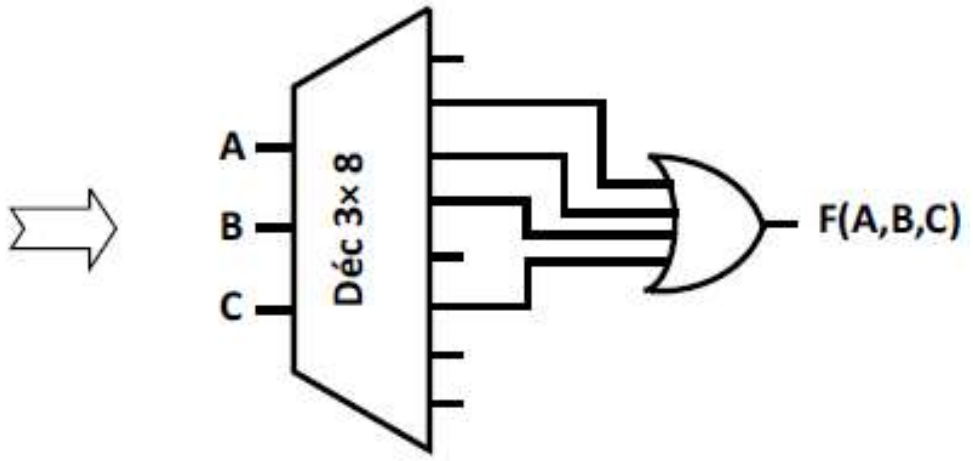
$$F(A,B,C)=\bar{B}C+\bar{A}B$$

$$F(A,B,C)=\bar{B}C(A+\bar{A})+\bar{A}B(C+\bar{C})$$

$$F(A,B,C)=(\bar{A}\bar{B}C+A\bar{B}C)+(\bar{A}B\bar{C}+\bar{A}BC)$$

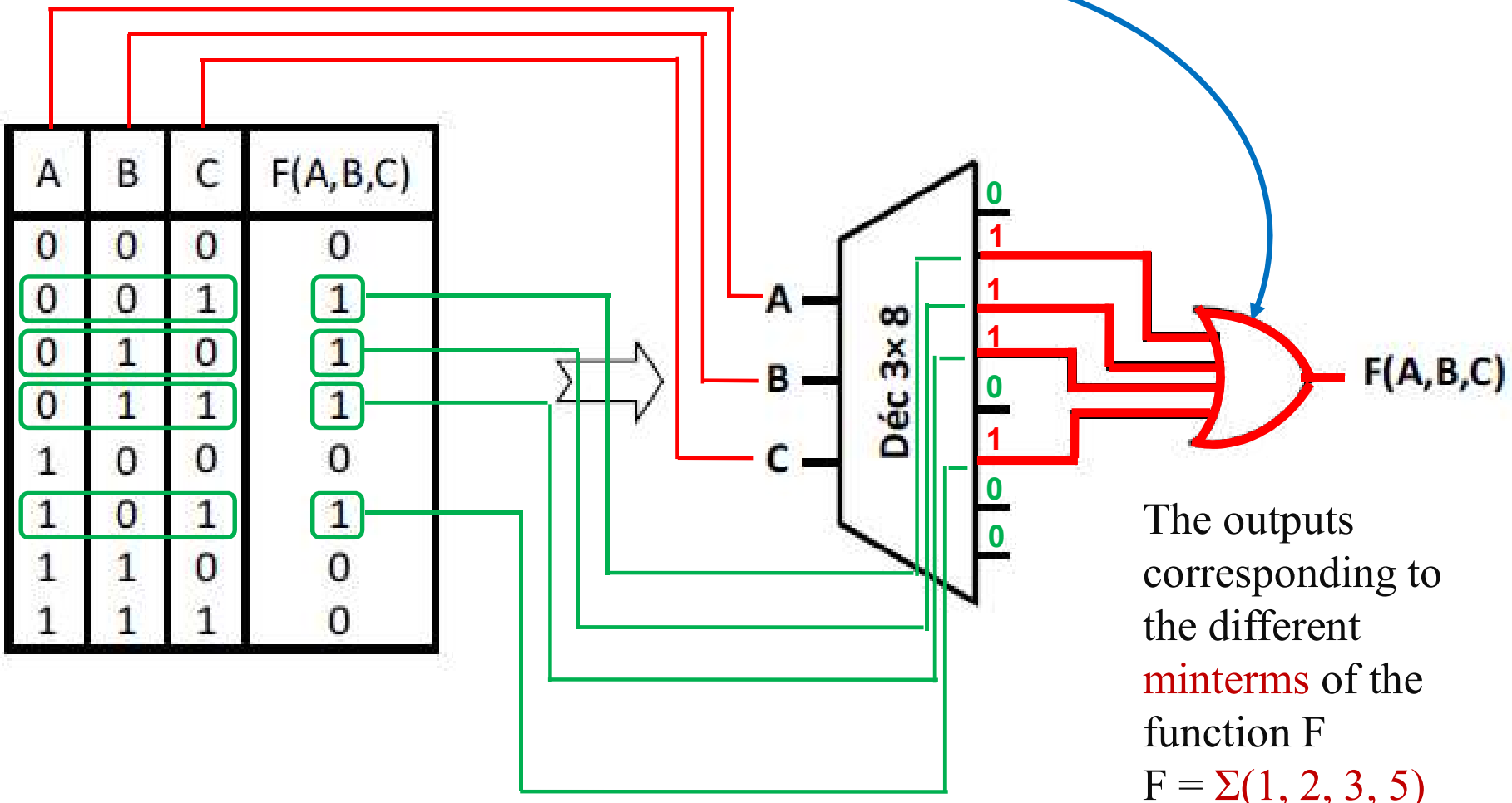
$$F(A,B,C)=\bar{A}\bar{B}C+\bar{A}B\bar{C}+\bar{A}BC+A\bar{B}C$$

| A | B | C | F(A,B,C) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |



# Logical Functions via Decoders:

$$F(A,B,C)=\bar{A}\,\bar{B}\,C+\bar{A}\,B\,\bar{C}+\bar{A}\,B\,C+A\,\bar{B}\,C$$




| A | B | C | F(A,B,C) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

The outputs corresponding to the different minterms of the function F

$F = \Sigma(1, 2, 3, 5)$

# Implementation of a full adder with 3→8 binary decoders.

- Reminder:

Truth table of a full adder for 1 bit.

| | | | | |
|---|---|---|---|---|
| 0+0 = 0 | | Carry | 0 | |
| 0+1 = 1 + 0 = 1 | | Carry | 0 | |
| 1 + 1 = 0 | | Carry | 1 | |
| 1+1+1 = 1 | | Carry | 1 | |

| $A_i$ | $B_i$ | $R_{i-1}$ | | $R_i$ | $S_i$ |
|---|---|---|---|---|---|
| 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 |

# Implementation of a full adder with 3→8 binary decoders.

$$S_i = \overline{A}_i.\overline{B}_i.R_{i-1} + \overline{A}_i.B_i.\overline{R}_{i-1} + A_i.\overline{B}_i.\overline{R}_{i-1} + A_i.B_i.R_{i-1}$$

**0 0 1 0 1 0 1 0 0 1 1 1**

$$R_i = \overline{A}_i B_i R_{i-1} + A_i \overline{B}_i R_{i-1} + A_i B_i \overline{R}_{i-1}. + A_i B_i R_{i-1}$$

| $A_i$ | $B_i$ | $R_{i-1}$ | | $R_i$ | $S_i$ |
|---|---|---|---|---|---|
| 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** |

# Implementation of a full adder with **3→8** binary decoders.

We supose $A=A_i$ , $B=B_i$ , $C=R_{i-1}C$

Then:

$$S_0 = \overline{A}.\overline{B}.\overline{C}, \qquad S_1 = \overline{A}.\overline{B}.C, \qquad S_2 = \overline{A}.B.\overline{C}, \qquad S_3 = \overline{A}.B.C,$$

$$S_4 = A.\overline{B}.\overline{C}, \qquad S_5 = A.\overline{B}.C, \qquad S_6 = A.B.\overline{C}, \qquad S_7 = A.B.C$$

$$R_i = S3 + S5 + S6 + S7$$

$$S_i = S1 + S2 + S4 + S7$$

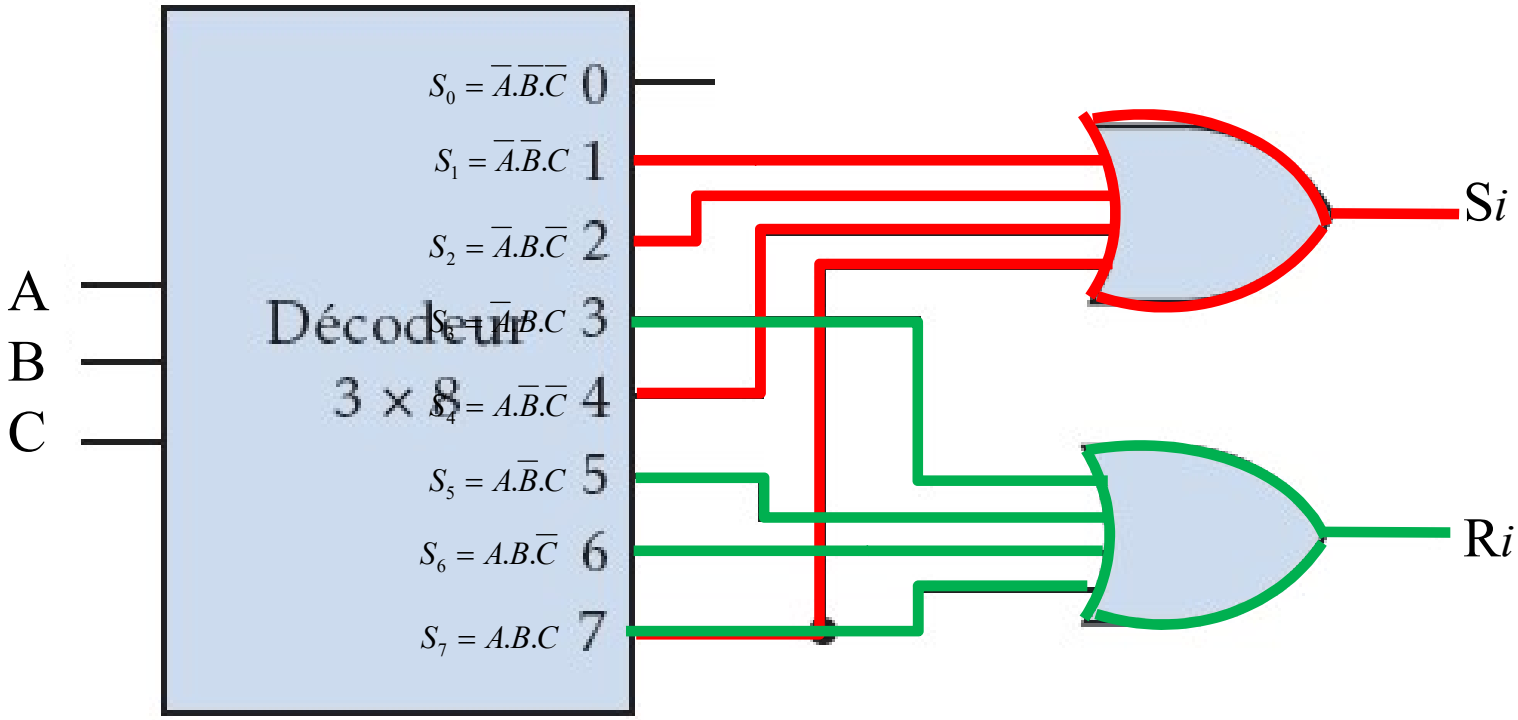
# Implementation of a full adder with 3→8 binary decoders.

We supose A=$A_i$ , B =$B_i$ , C=$R_{i-1}C$

Then:

$$S_i = S1 + S2 + S4 + S7$$

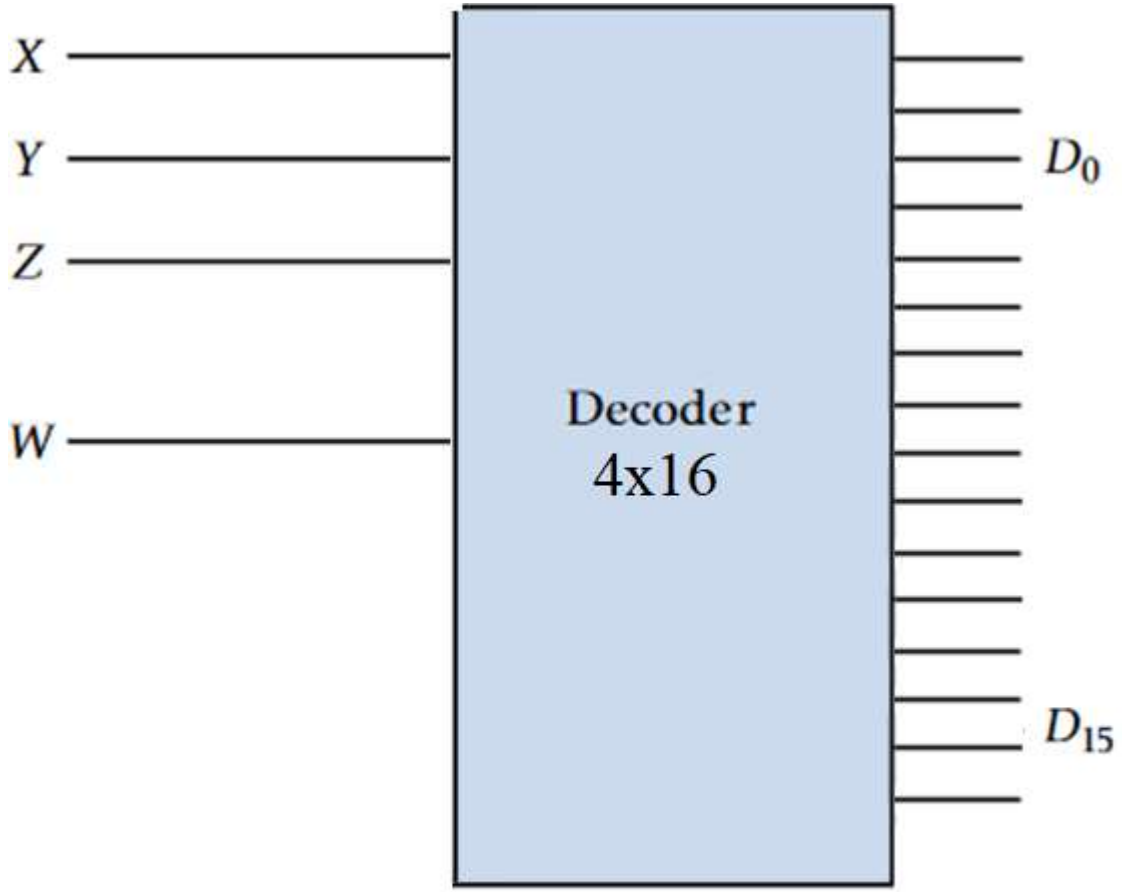
$$R_i = S3 + S5 + S6 + S7$$



# Synthesis of large binary decoders. Example: **3→8**

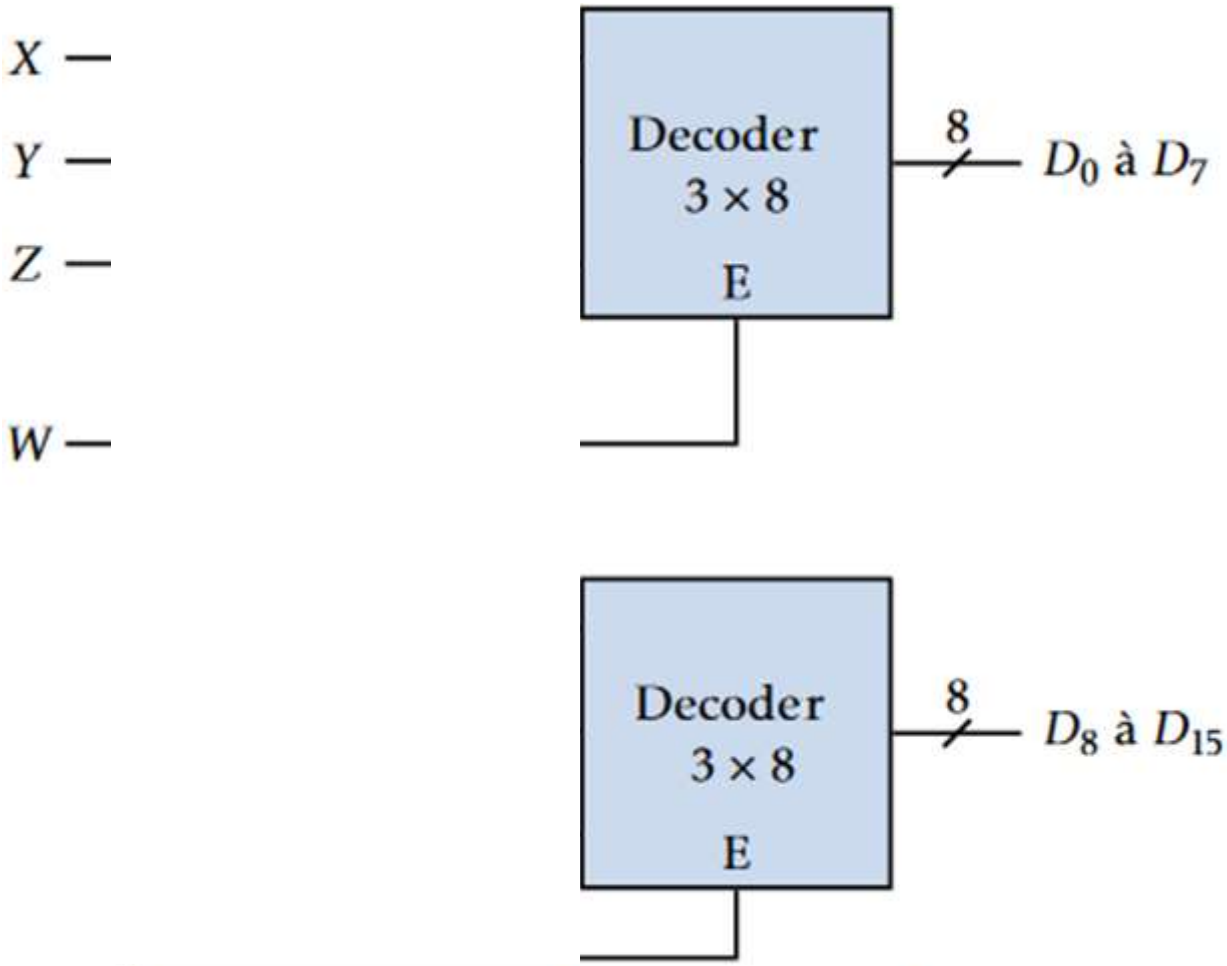
- Using 2 decoders 3 × 8 to create a 4 × 16 decoder.
- Decoders with enable inputs can be combined to create larger decoders.
- For example, we can use 2 decoders 3 × 8 to create a 4 × 16 decoder.
- The fourth variable is used to activate one or the other of the 3 × 8 decoders.
- In the following figure: The input W activates only one of the two decoders at a time.

# Synthesis of large binary decoders.



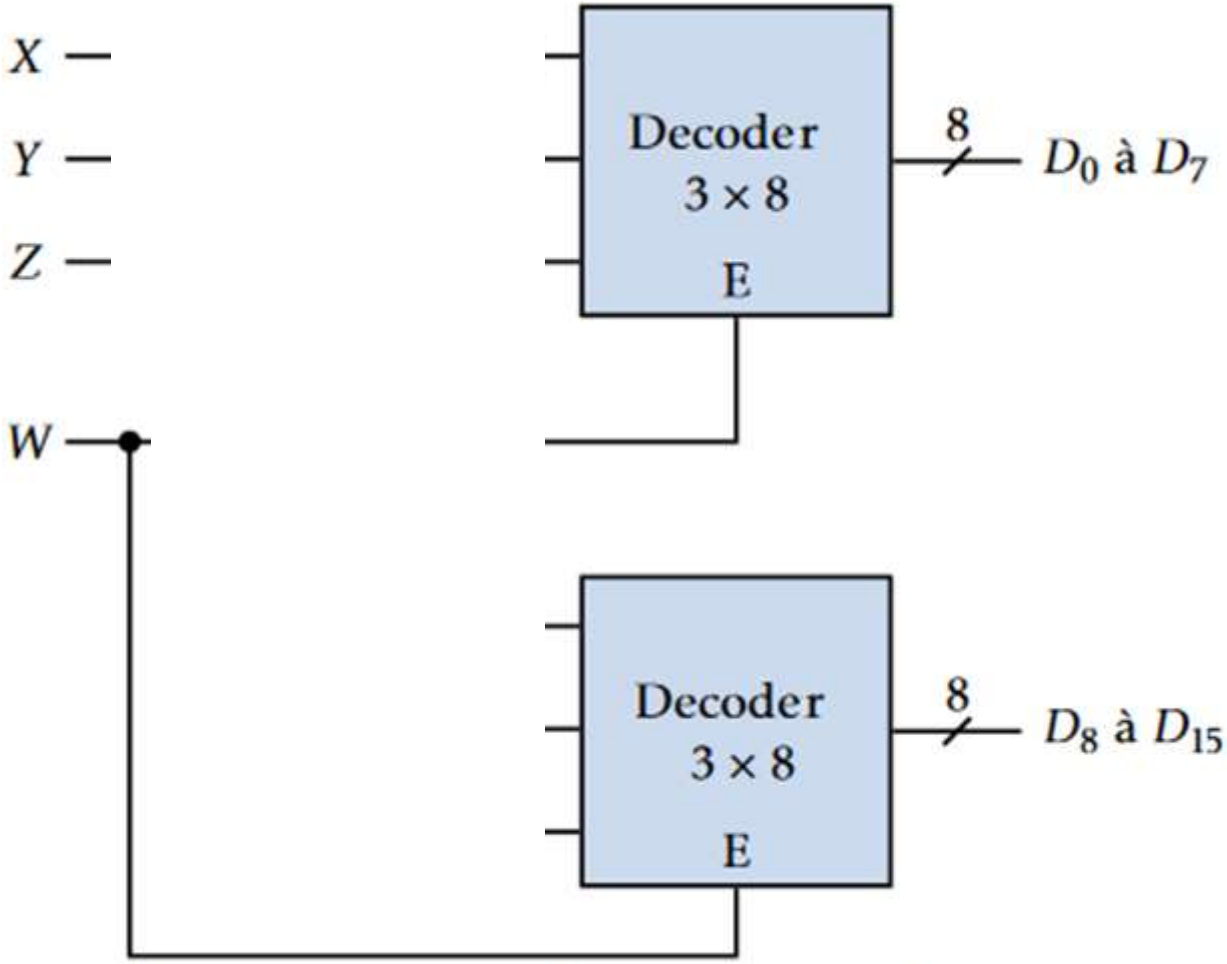

– 4x16 decoder

# Synthesis of large binary decoders.



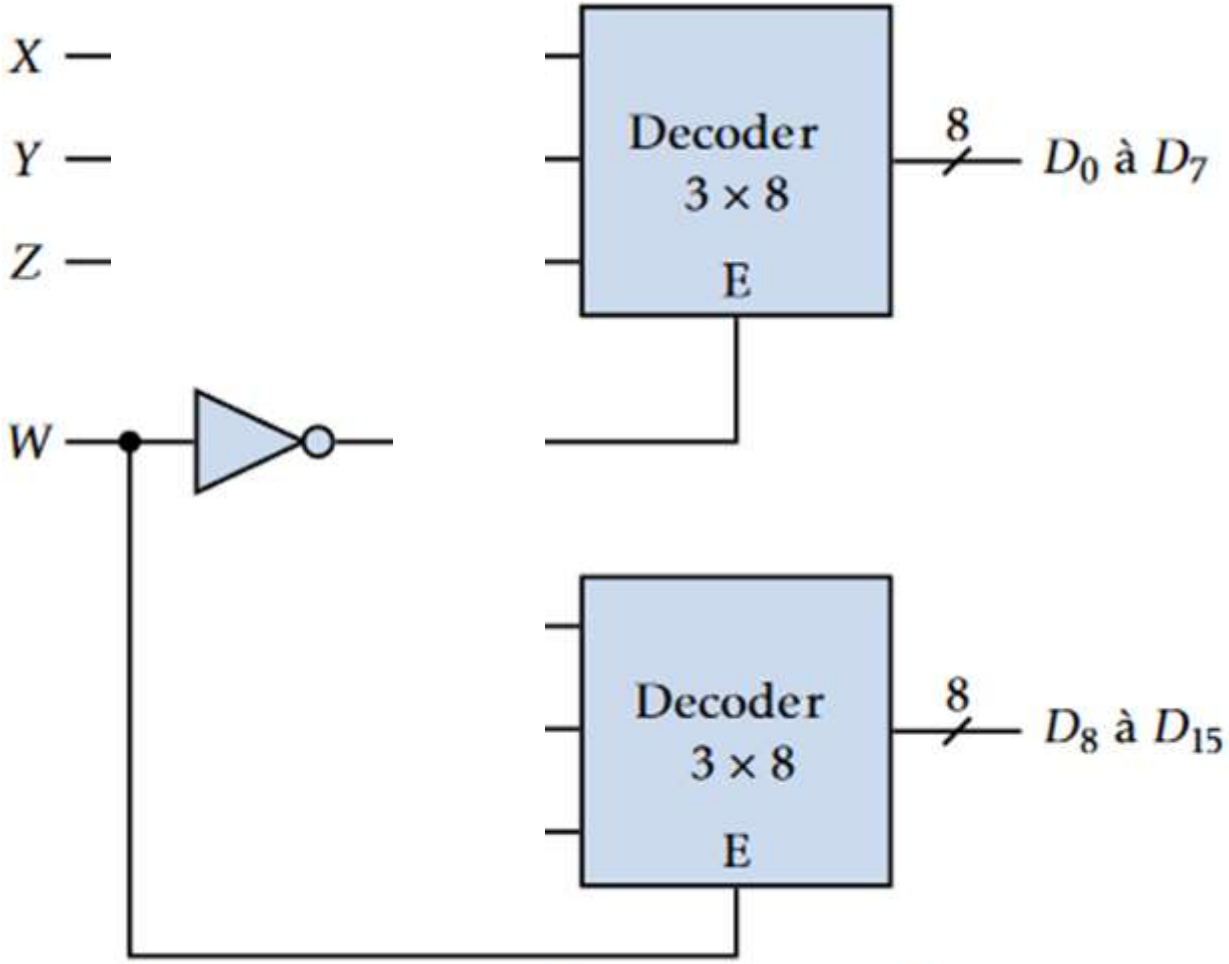

– 4x16 decoder created with two 3x8 decoders.

# Synthesis of large binary decoders.



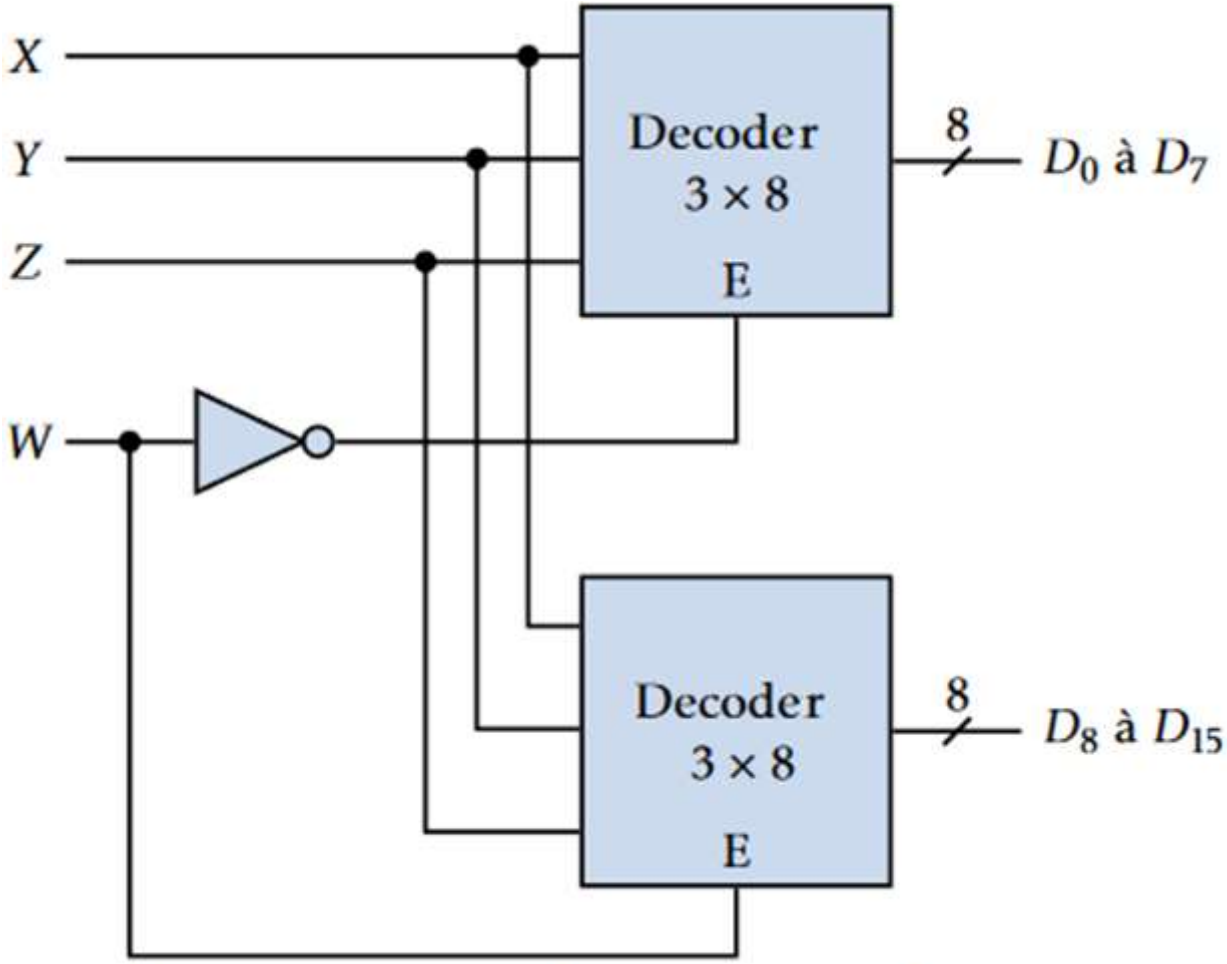

– 4x16 decoder created with two 3x8 decoders.

# Synthesis of large binary decoders.




– 4x16 decoder created with two 3x8 decoders.

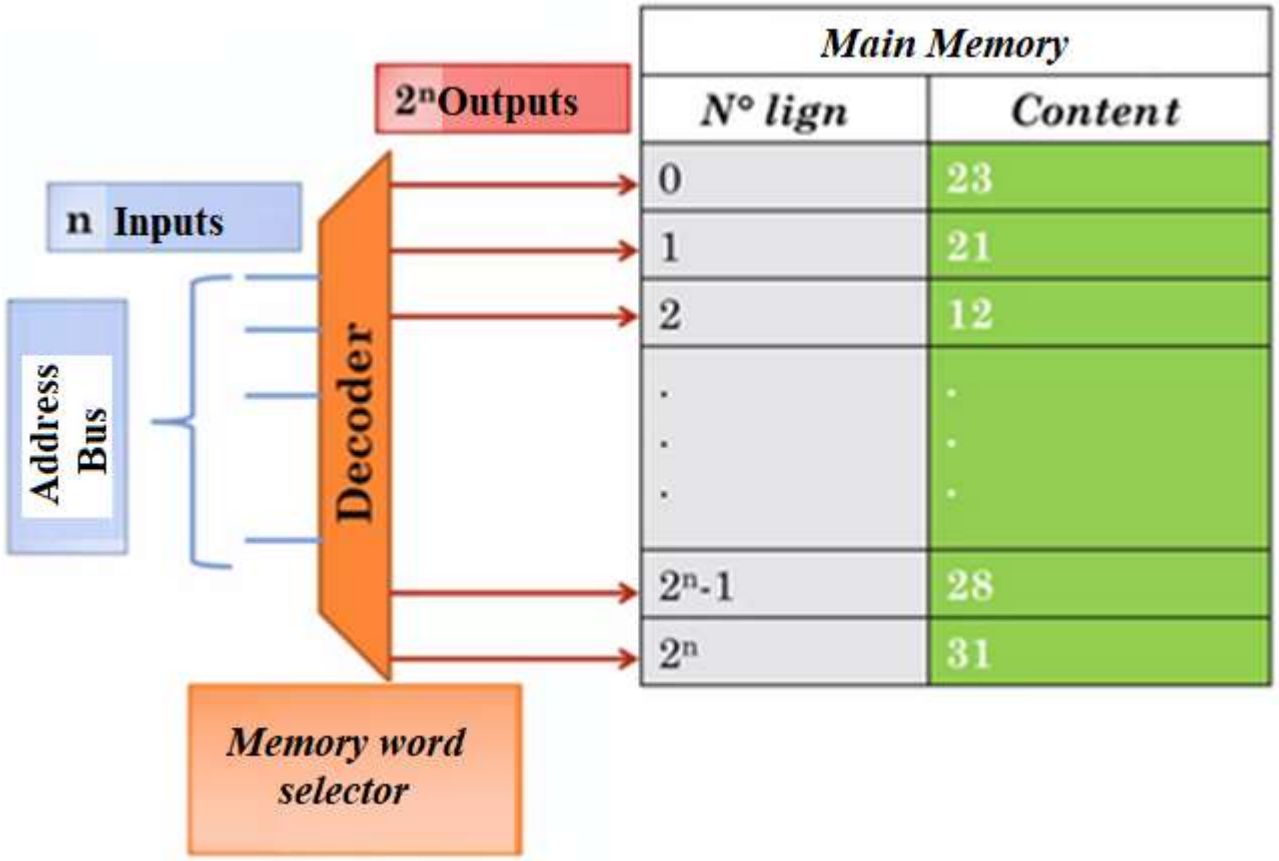
# Synthesis of large binary decoders.




– 4x16 decoder created with two 3x8 decoders.

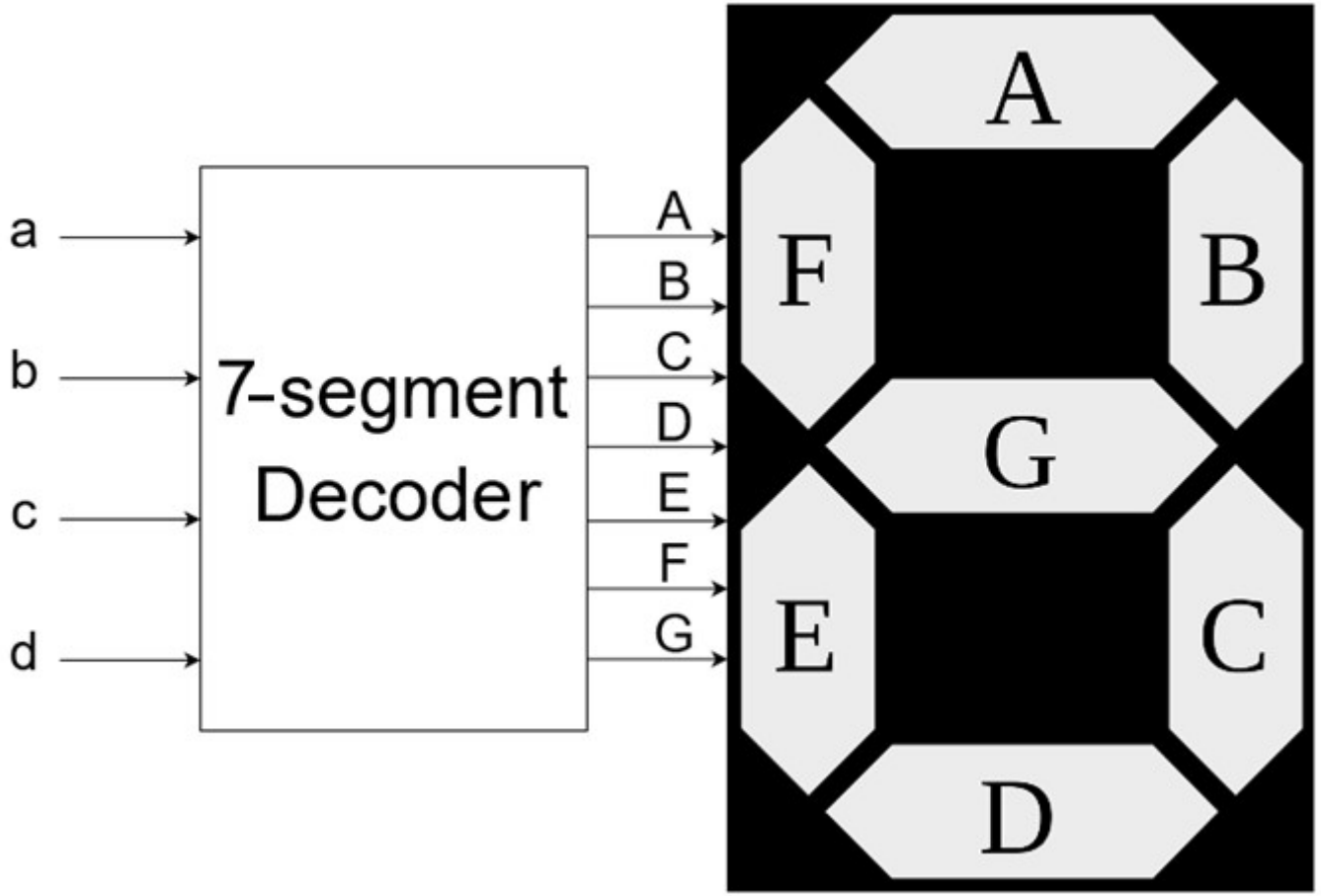
# Example of decoder application

- The decoder is an essential component at the input of the main memory.

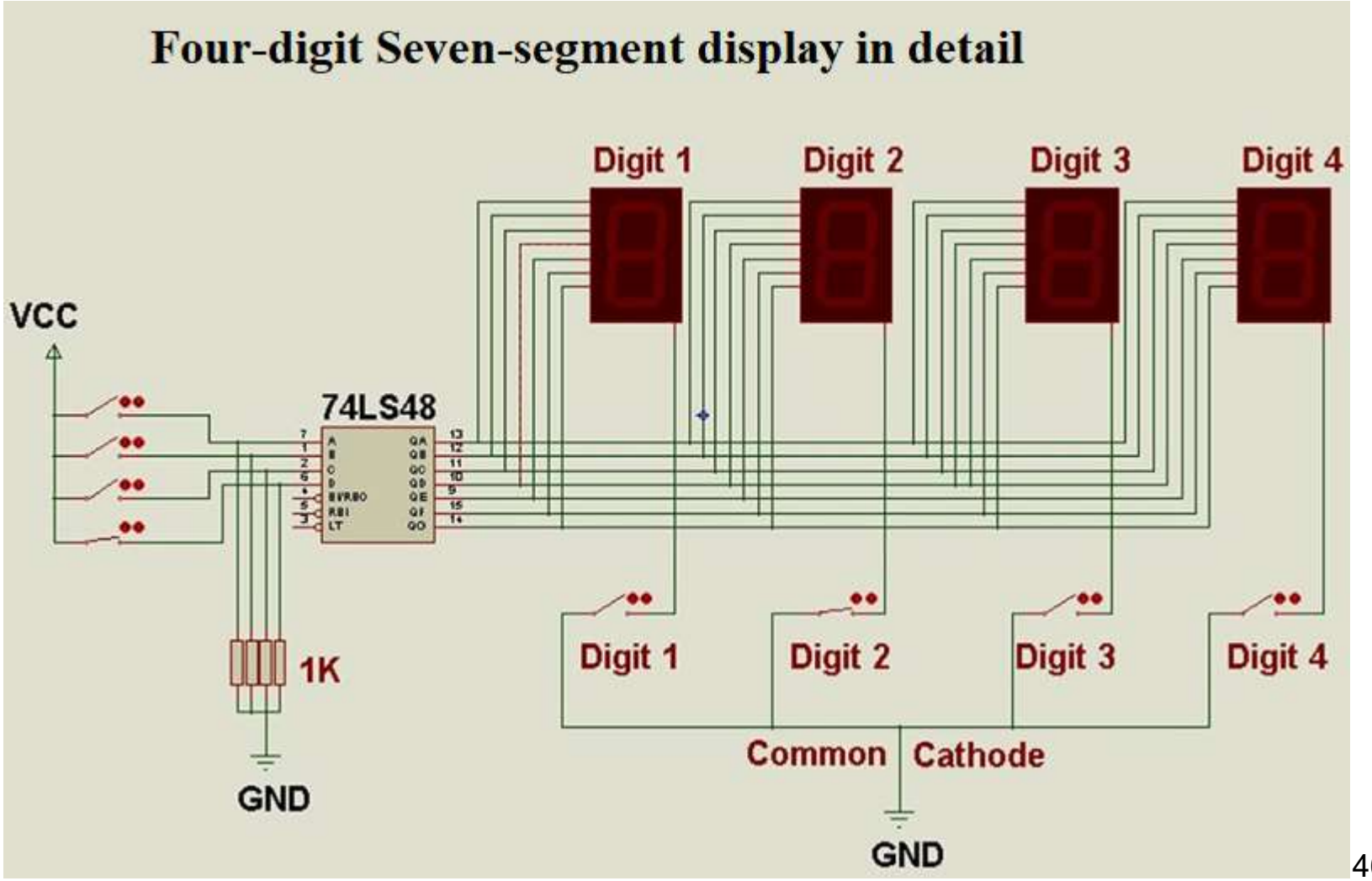


# Example of decoder application

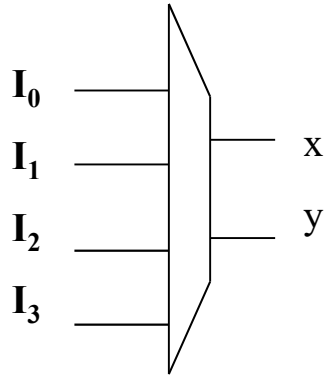
- Seven-segment display:



# Example of decoder application



# Binairy Encoder




- It plays the opposite role of a decoder. It has :
  - $2^n$ inputs.
  - N output.

**Example: Encoder 4→2**

- For each input combination, its number (in binary) will appear at the output.
- A coder generates the binary code equivalent to the numbers of an activated input.
- A coder detects which is the active input among $2^n$ of n standard inputs.

# Binairy Encoder

- For example, if input $I_2$ is active, it generates the output code 10.

- It is a device that performs the operation of a decoder: only one input among **M** is activated at a time, which corresponds to a binary number at the output.

- A priority encoder, if two inputs are active simultaneously, prioritizes one.

# Binairy Encoder

- Example: The truth table of an 8-to-3 encoder is shown in the following figure:

| $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ | $A_2$ | $A_1$ | $A_0$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

The truth table of Encoder 8→3

# Binairy Encoder

- **Exemple:**
- The truth table of an 8-to-3 encoder is shown in the following figure: The outputs are obtained with OR gates,
- For example, output $A_0 = 1$

  When inputs 1, 3, 5, or 7 are 1. We then obtain the following equations:

$A_0 = D_1 + D_3 + D_5 + D_7$

$A_1 = D_2 + D_3 + D_6 + D_7$

$A_2 = D_4 + D_5 + D_6 + D_7$

# Binairy Encoder

- So, the 8-to-3 encoder can be implemented with three 3-input OR gates.
- **Note:** Only one input must be activated at a time, otherwise, there will be an error.
- For example, If we simultaneously activate inputs $D_3$ and $D_6$, so $D_3 = D_6 = 1$, the output will be:

$A_2 = 1$, $A_1 = 1$ et $A_0 = 1$, which would mean that input 7 is activated.

$$A_0 = D_1 + D_3 + D_5 + D_7$$
$$A_1 = D_2 + D_3 + D_6 + D_7$$
$$A_2 = D_4 + D_5 + D_6 + D_7$$

# Binairy Encoder

The truth table of this encoder is shown in the following figure:

| $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ | $A_2$ | $A_1$ | $A_0$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

The truth table of Encoder 8→3

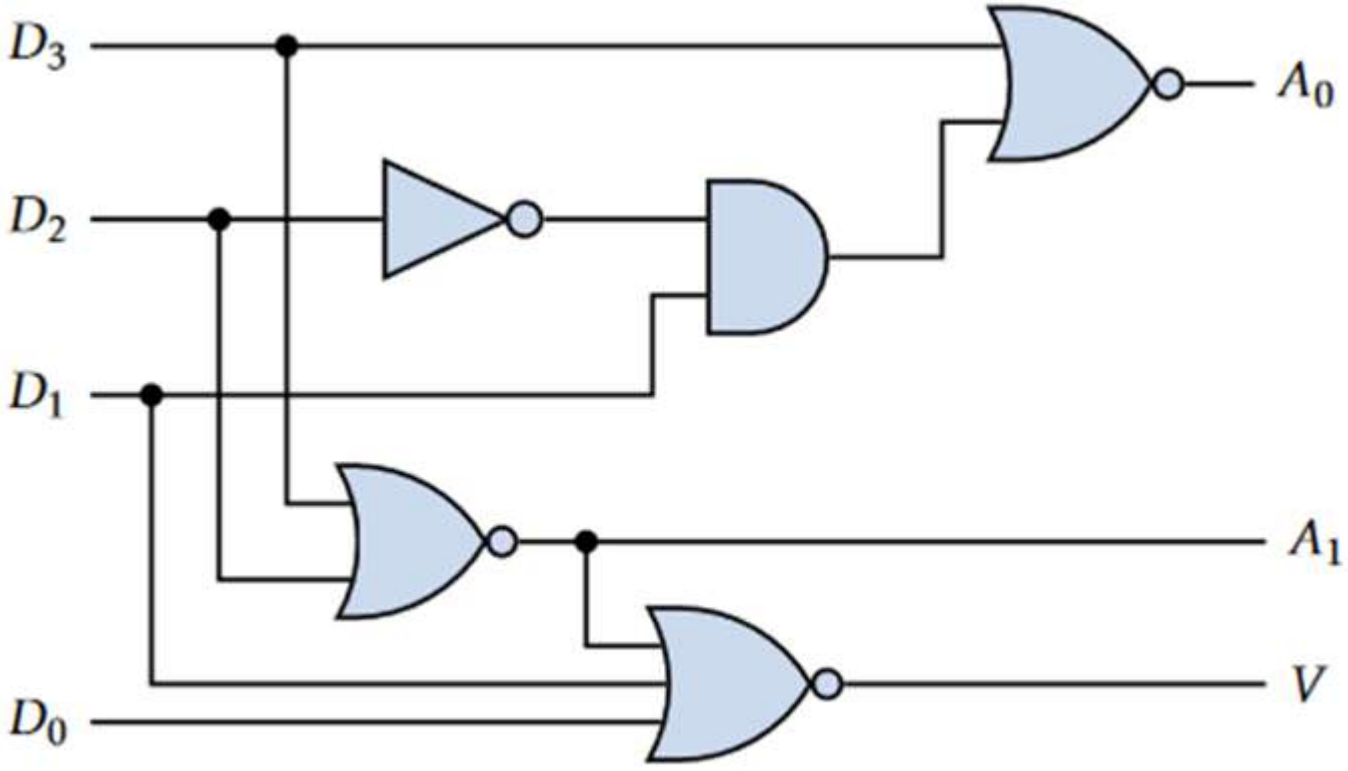
# Binairy Encoder

- To address this issue, we modify the encoder so that the highest input has priority: **it's a priority encoder**.
- Example 2: The truth table of a 4-to-2 encoder is shown in the figure. Note the use of don't-care conditions. We also have a validation output: V = 1 if any of the inputs are 1, otherwise V = 0. The circuit is shown in the figure.

| $D_0$ | $D_1$ | $D_2$ | $D_3$ | $A_1$ | $A_0$ | $V$ |
|---|---|---|---|---|---|---|
| 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 |

# Binairy Encoder

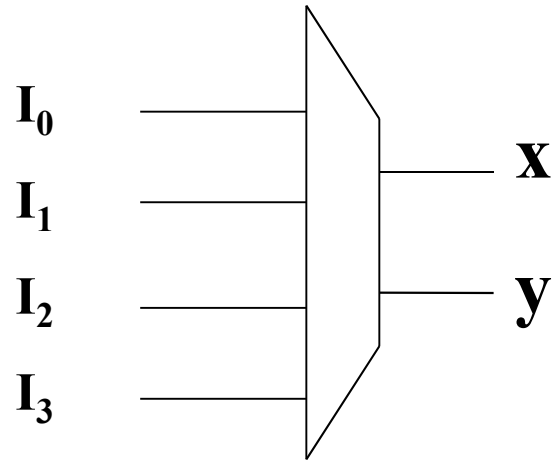
- The circuit is shown in the figure.




– 4-to-2 priority encoder

# Binairy Encoder **4→2**

| $I_0$ | $I_1$ | $I_2$ | $I_3$ | | x | y |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | 0 | 0 |
| 1 | x | x | x | | 0 | 0 |
| 0 | 1 | x | x | | 0 | 1 |
| 0 | 0 | 1 | x | | 1 | 0 |
| 0 | 0 | 0 | 1 | | 1 | 1 |




$$X = \overline{I0}.\overline{I1}.(I2 + I3)$$

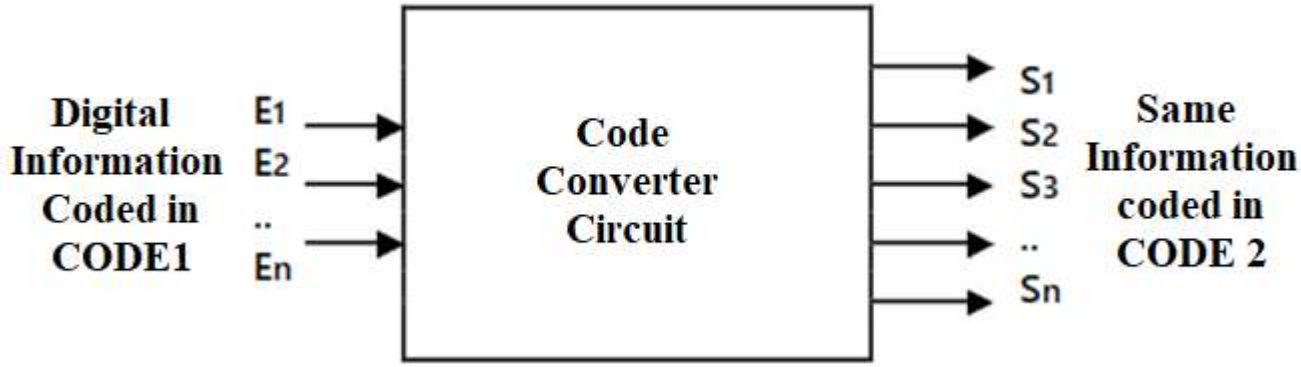
$$Y = \overline{I0}.(I1 + .\overline{I2}.I3)$$

# Transcoder

- Combinational transcode circuits (also known as code converters) fall into 3 categories:
  - Encoders,
  - Decoders,
  - Transcoders.
- All these logic circuits transform information present at their inputs in one format (code 1) into the same information present at their outputs in a different format (code 2).

# Transcoder

- We call:
- Encoder: a circuit with $2^n$ inputs and n outputs.
- Decoder: a circuit with n inputs and $2^n$ outputs, where only one is activated at a time.
- Transcoder: any other code converter circuit different from the previous ones, with P inputs and K outputs.



# Transcoder BCD/EXESS-3

Example:

The implementation of a transcoder:

We want to perform a transcoding from BCD code to Excess-3 code.

The input and output numbers are expressed in 4 bits, and this transcoder will be able to convert all digits from 0 to 9.

Solution: Step -1 in designing the transcoder: Writing the truth table:

# Transcoder BCD/EXESS-3

| Chiffre converti | Entrées (BCD) | | | | Sorties (XS 3) | | | |
|---|---|---|---|---|---|---|---|---|
| | $E_3$ | $E_2$ | $E_1$ | $E_0$ | $S_3$ | $S_2$ | $S_1$ | $S_0$ |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |

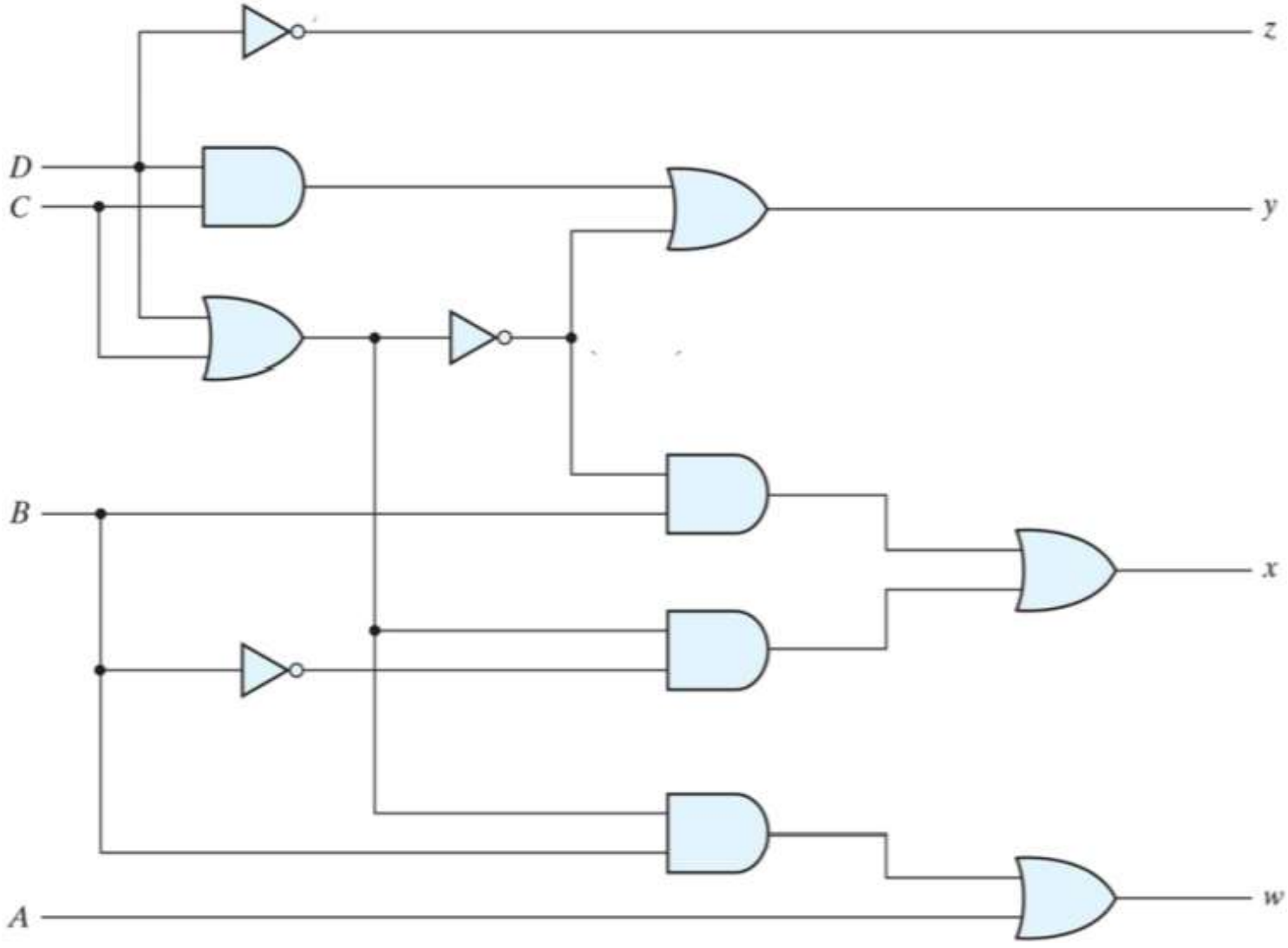
# Transcoder BCD/EXESS-3

- Step - 2 in designing the transcoder: Searching for and simplifying the equations of the outputs:
  S0=....., S1=....., S2=....., S3=.....
- Step - 3 in designing the transcoder: Drawing the logic diagram.
- Note: among the 16 possible combinations applicable to the 4 inputs of the transcoder, only 10 combinations will be used (to encode the 10 digits to be converted). The other 6 will never be present at the input of the transcoder. Crosses then appear in 6 cells of the Karnaugh maps of the outputs, which allows for a significant simplification of the logic equations.

# Transcoder BCD/EXESS-3

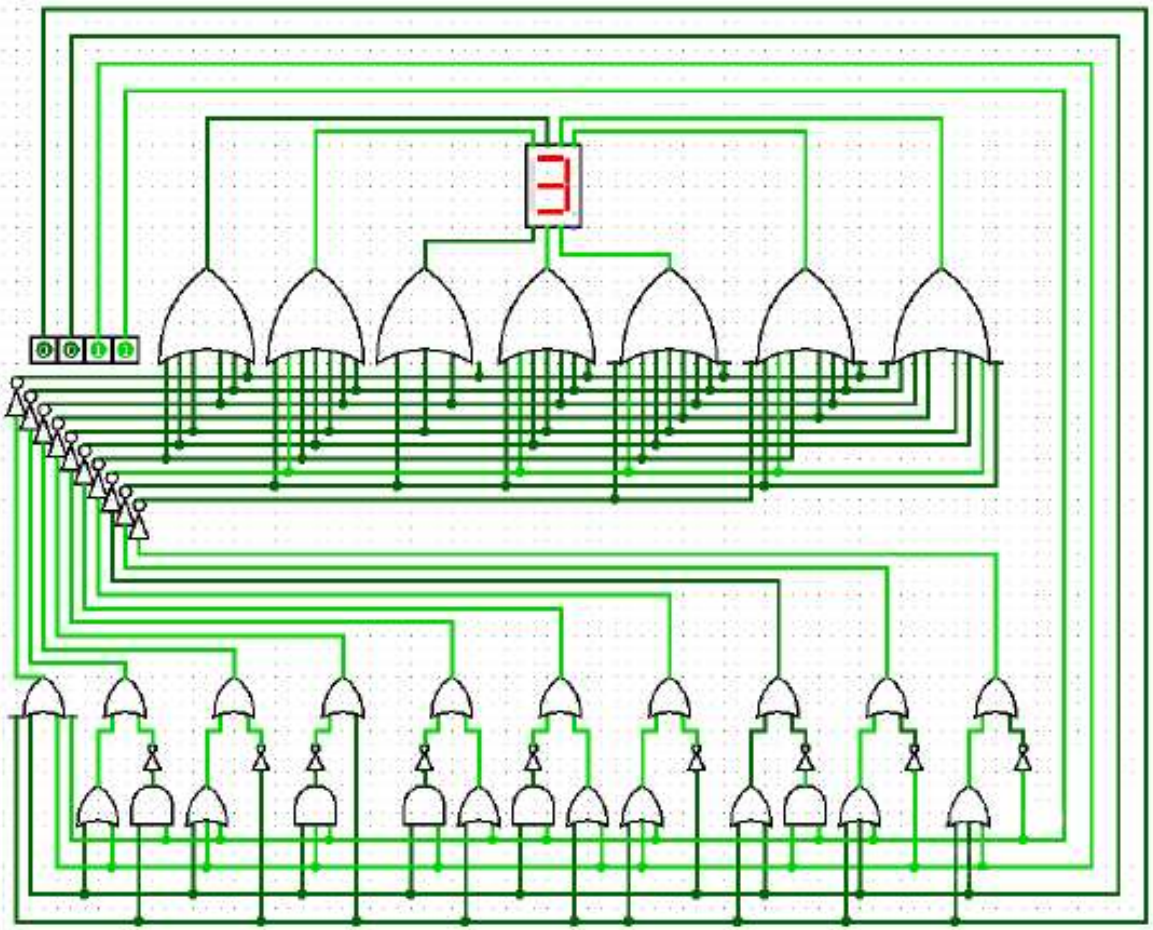
| A | B | C | D | | X | Y | Z | T |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 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 |
| 1 | 0 | 1 | 0 | | x | x | x | x |
| 1 | 0 | 1 | 1 | | x | x | x | x |
| 1 | 1 | 0 | 0 | | x | x | x | x |
| 1 | 1 | 0 | 1 | | x | x | x | x |
| 1 | 1 | 1 | 0 | | x | x | x | x |
| 1 | 1 | 1 | 1 | | x | x | x | x |

# Transcoder BCD/EXESS-3



# Transcoder BCD/7 segments

transcoder BCD/7 segments.



Transcoder BCD/7 segments.

# Transcoder BCD/EXESS-3

- Note: Among the transcoders found in integrated circuits:
- Decimal / BCD transcoders (74147 circuit)
- BCD / Decimal transcoders (7442, 7445, and 4028 circuits)
- XS 3 / Decimal transcoders (7443 circuit)
- Excess-3 Gray transcoders (Gray+3) / Decimal (7444 circuit)
- DCB / 7-segment display transcoders (7448, 7511, 4543, 4511 circuits)
- 5-bit binary / DCB transcoders (74185 circuit)
- DCB / 5-bit binary transcoders (74184 circuit)