Computer System Architecture

DR. Howida Youssry

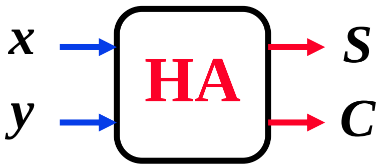
Functions and Functional Blocks

- **The functions considered are those found to be very useful in design**
- **Corresponding to each of the functions is a combinational circuit implementation called a *functional block*.**

Binary Adder

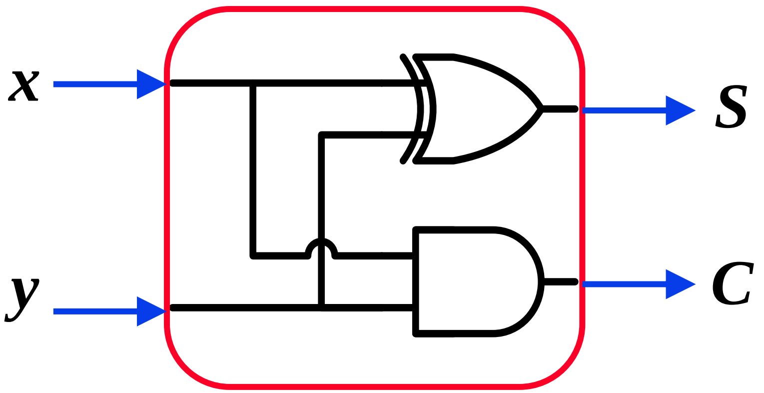
★ Half Adder

- Adds 1-bit plus 1-bit
- Produces Sum and Carry



	x
$+$	y
C	S

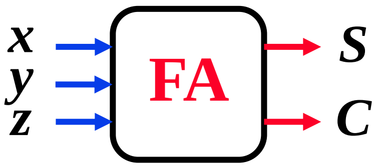
x y	C S
0 0	0 0
0 1	0 1
1 0	0 1
1 1	1 0



Binary Adder

★ Full Adder

- Adds 1-bit plus 1-bit plus 1-bit
- Produces Sum and Carry



$$\begin{array}{r} x \\ +\ y \\ +\ z \\ \hline C\ S \end{array}$$

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

			y	y
	0	1	0	1
x	1	0	1	0
		z	z	

$S = xy'z'+x'yz'+x'y'z+xyz = x \oplus y \oplus z$

			y	y
	0	0	1	0
x	0	1	1	1
		z	z	

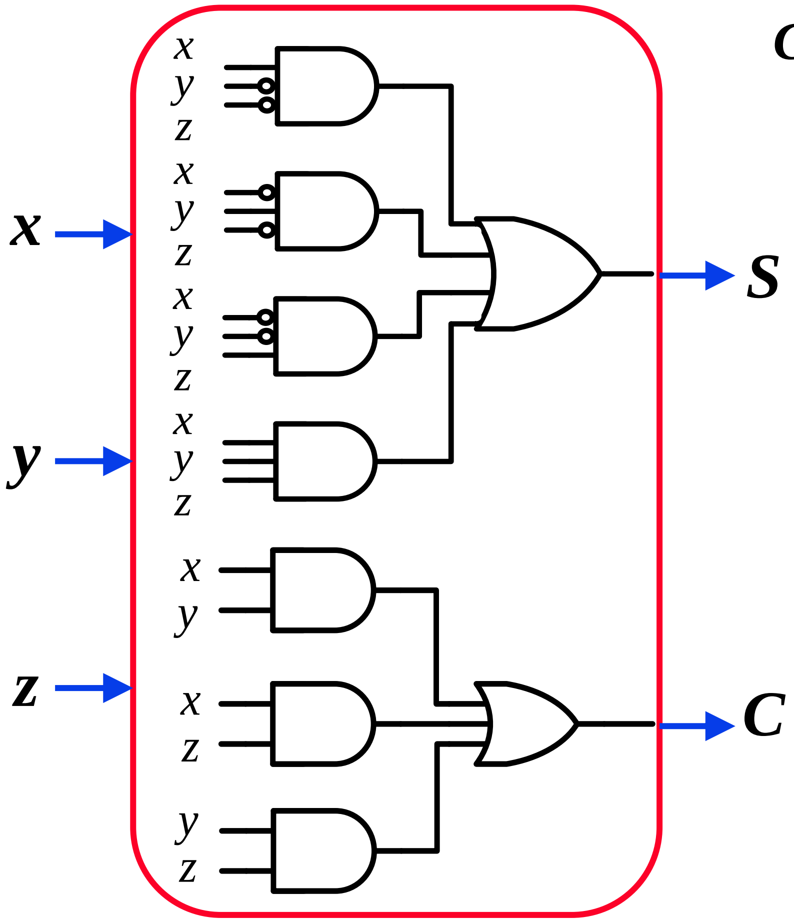
$C = xy + xz + yz$

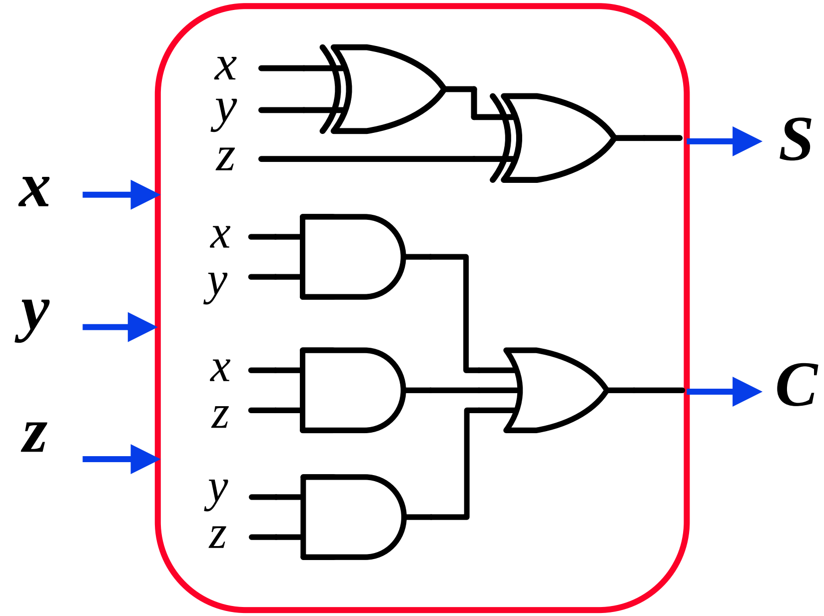
Binary Adder

★ Full Adder

$$S = xy'z' + x'yz' + x'y'z + xyz = x \oplus y \oplus z$$

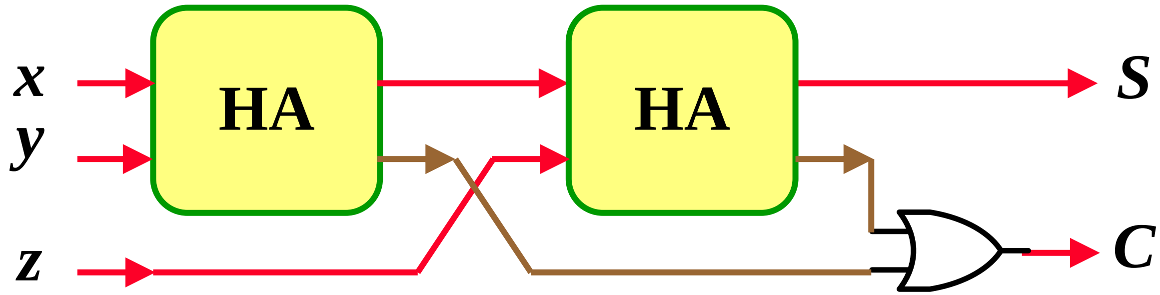
$$C = xy + xz + yz$$

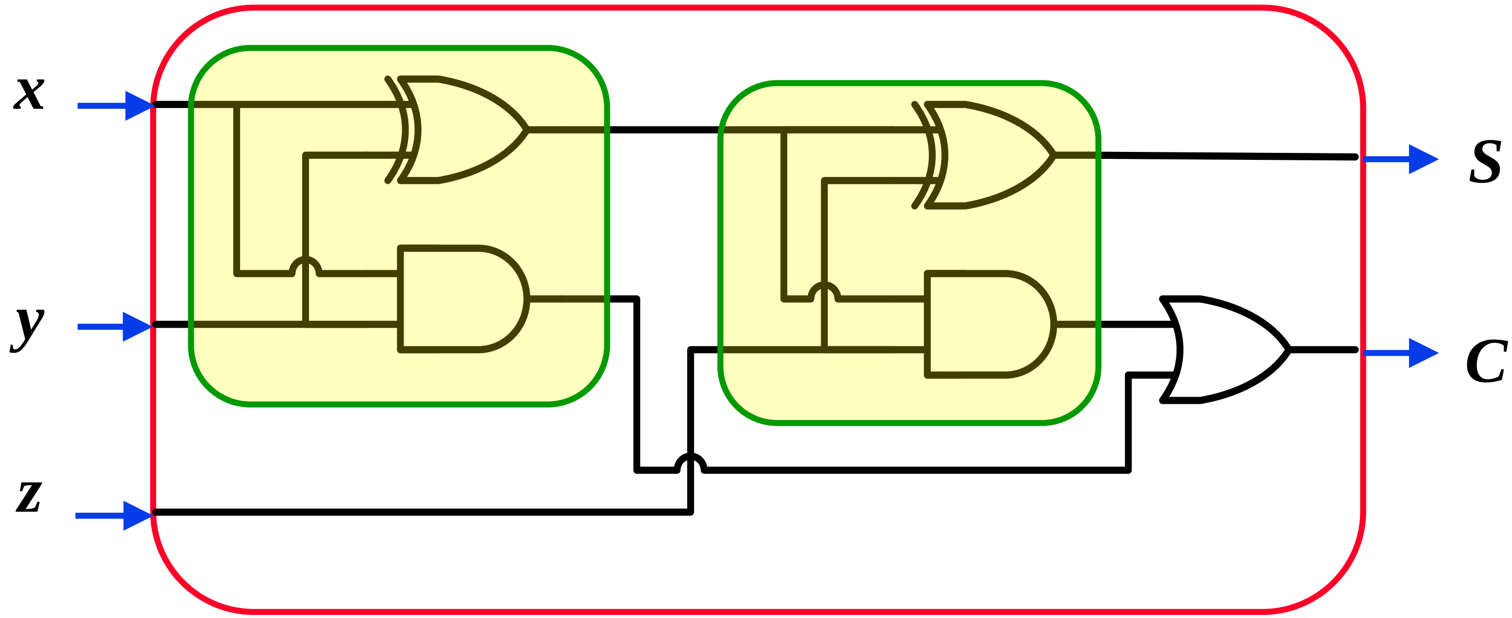




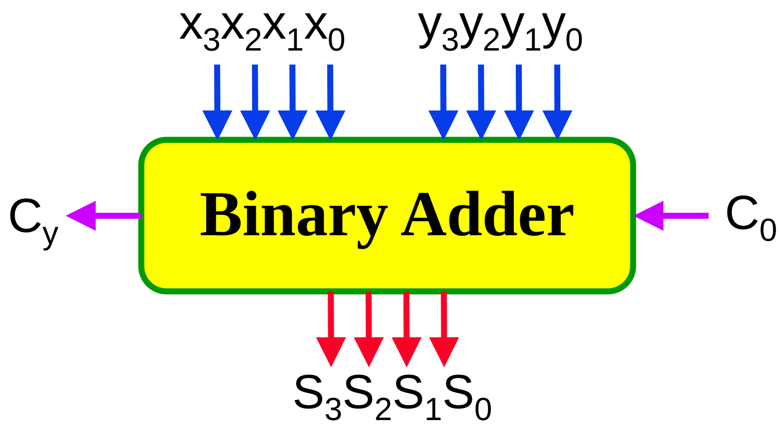
Binary Adder

★ **Full Adder**

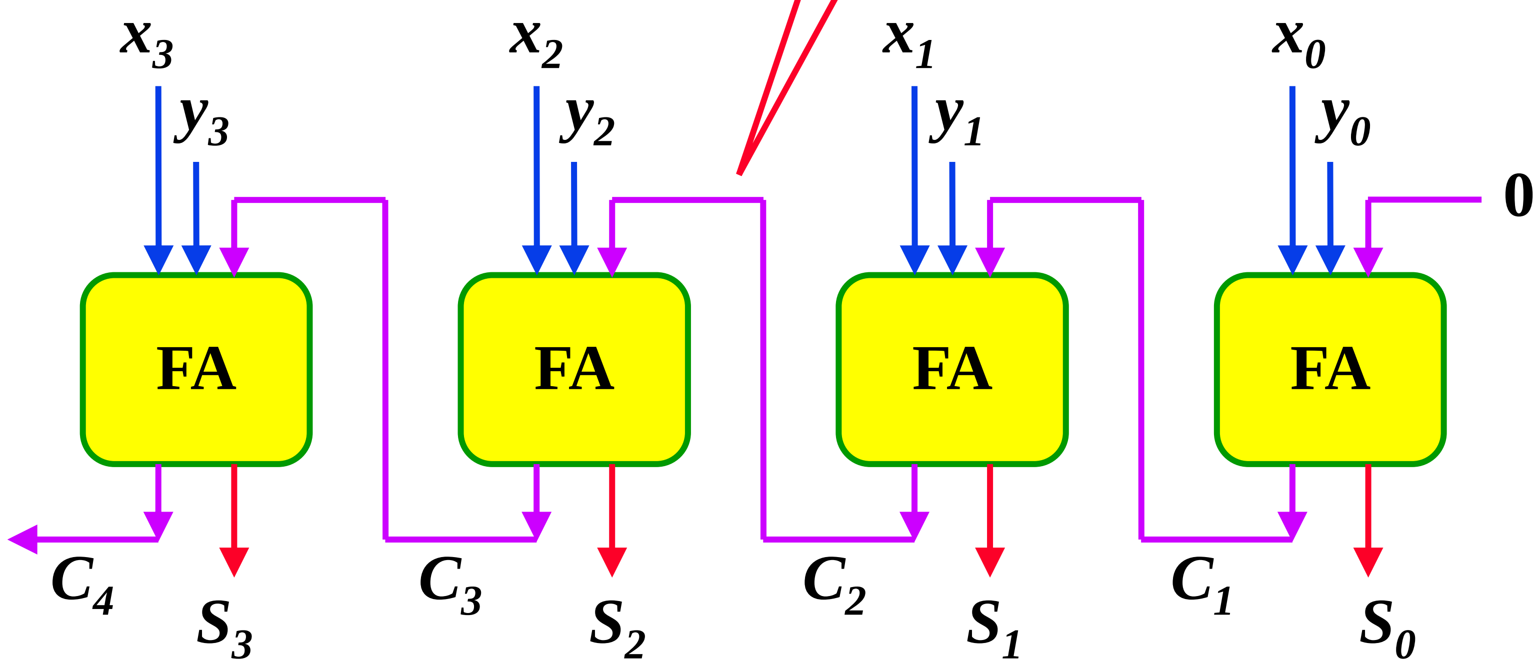




Binary Adder

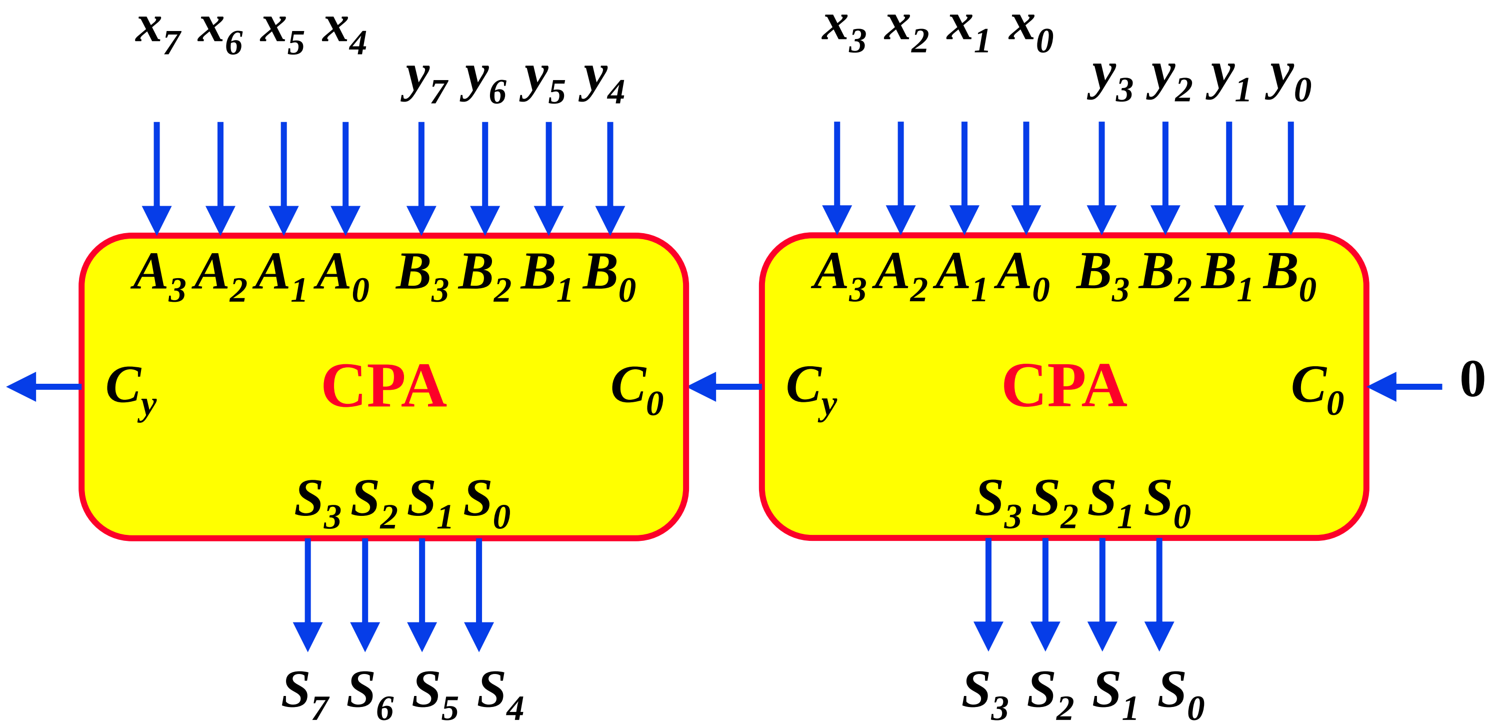


$$\begin{array}{ccccc} & c_3 & c_2 & c_1 & \\ + & x_3 & x_2 & x_1 & x_0 \\ + & y_3 & y_2 & y_1 & y_0 \\ \hline Cy & S_3 & S_2 & S_1 & S_0 \end{array}$$



Binary Adder

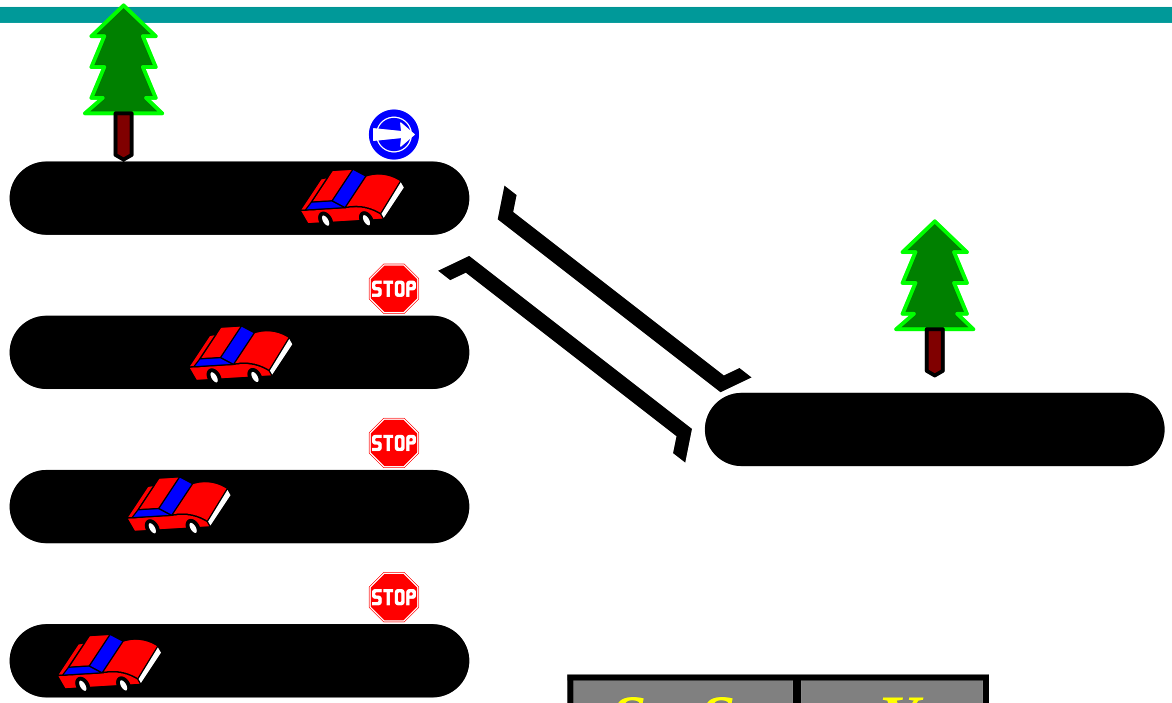
★ **Carry Propagate Adder**



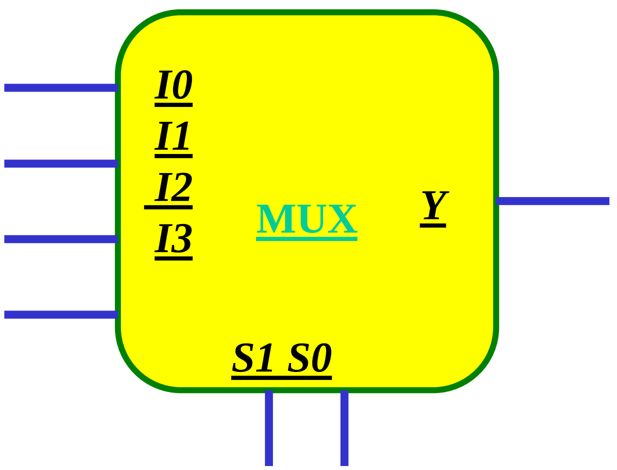
Selecting

- **Selecting of data or information is a critical function in digital systems and computers**
- **Circuits that perform selecting have:**
  - **A set of information inputs from which the selection is made**
  - **A single output**
  - **A set of control lines for making the selection**
- **Logic circuits that perform selecting are called *multiplexers***
- **Selecting can also be done by three-state logic or transmission gates**

Multiplexers

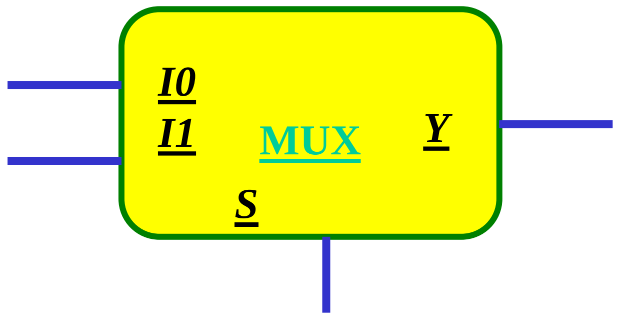


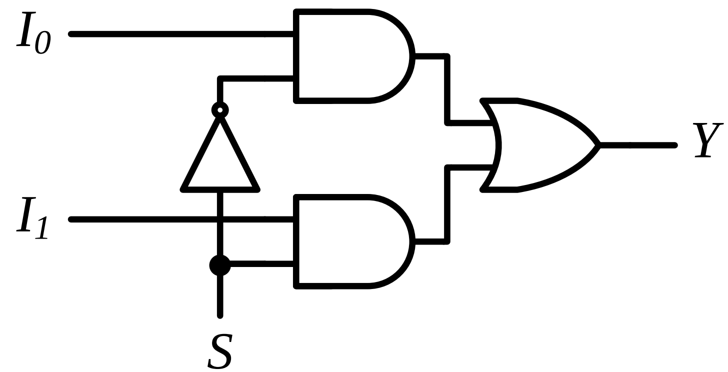
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



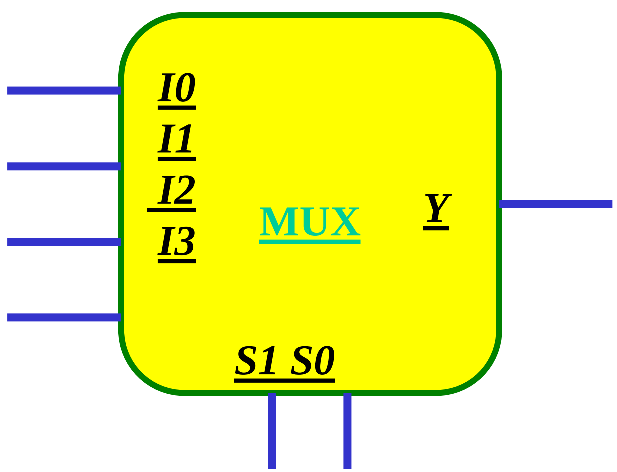
Multiplexers

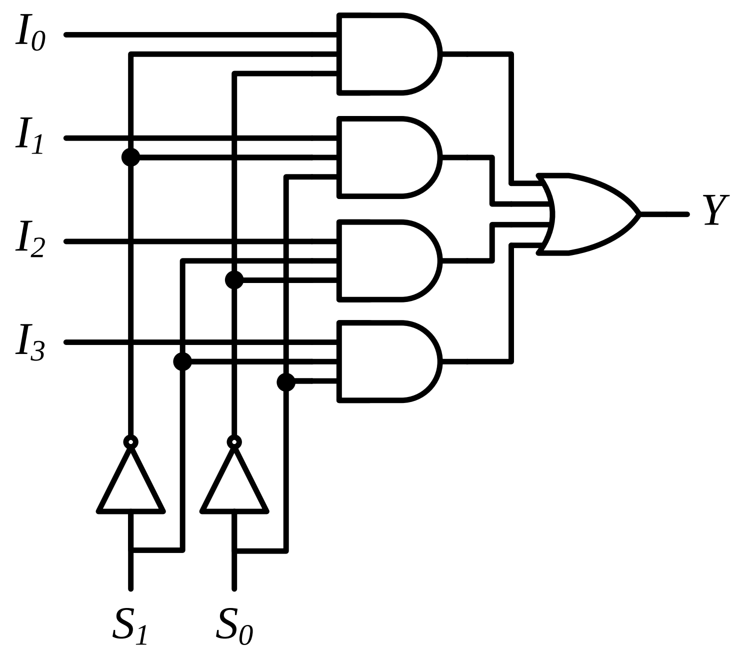
- 2-to-1 MUX





- 4-to-1 MUX

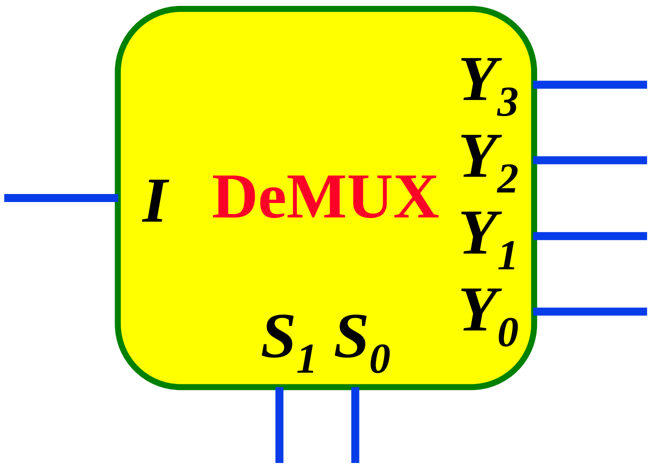


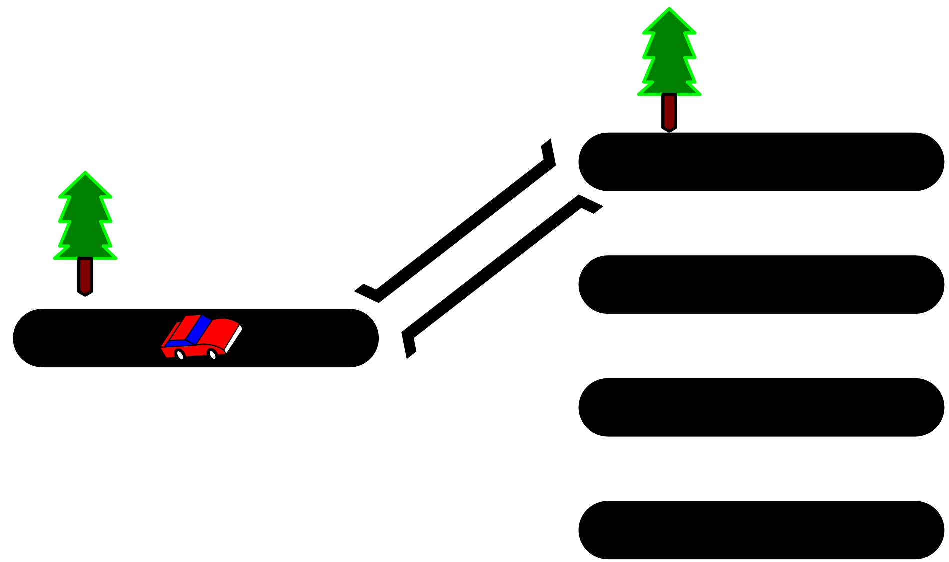


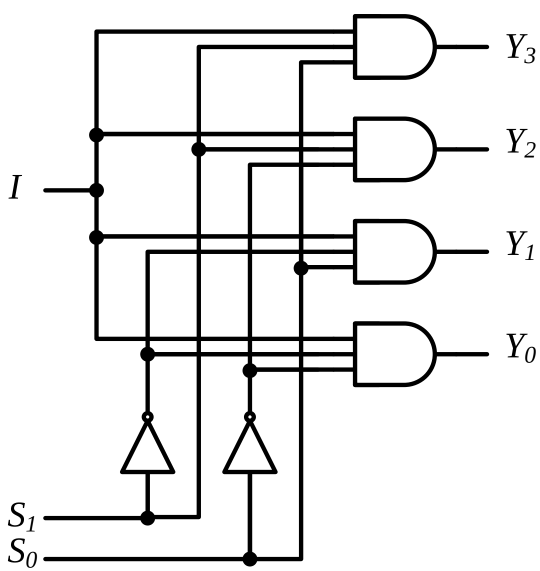
Multiplexers

- **A multiplexer selects information from an input line and directs the information to an output line**
- **A typical multiplexer has *n* control inputs (S_{n-1}, … S_0) called *selection inputs*, 2^n information inputs (I_{2^n-1}, … I_0), and one output Y**
- **A multiplexer can be designed to have *m* information inputs with $m < 2^n$ as well as *n* selection inputs**

DeMultiplexers

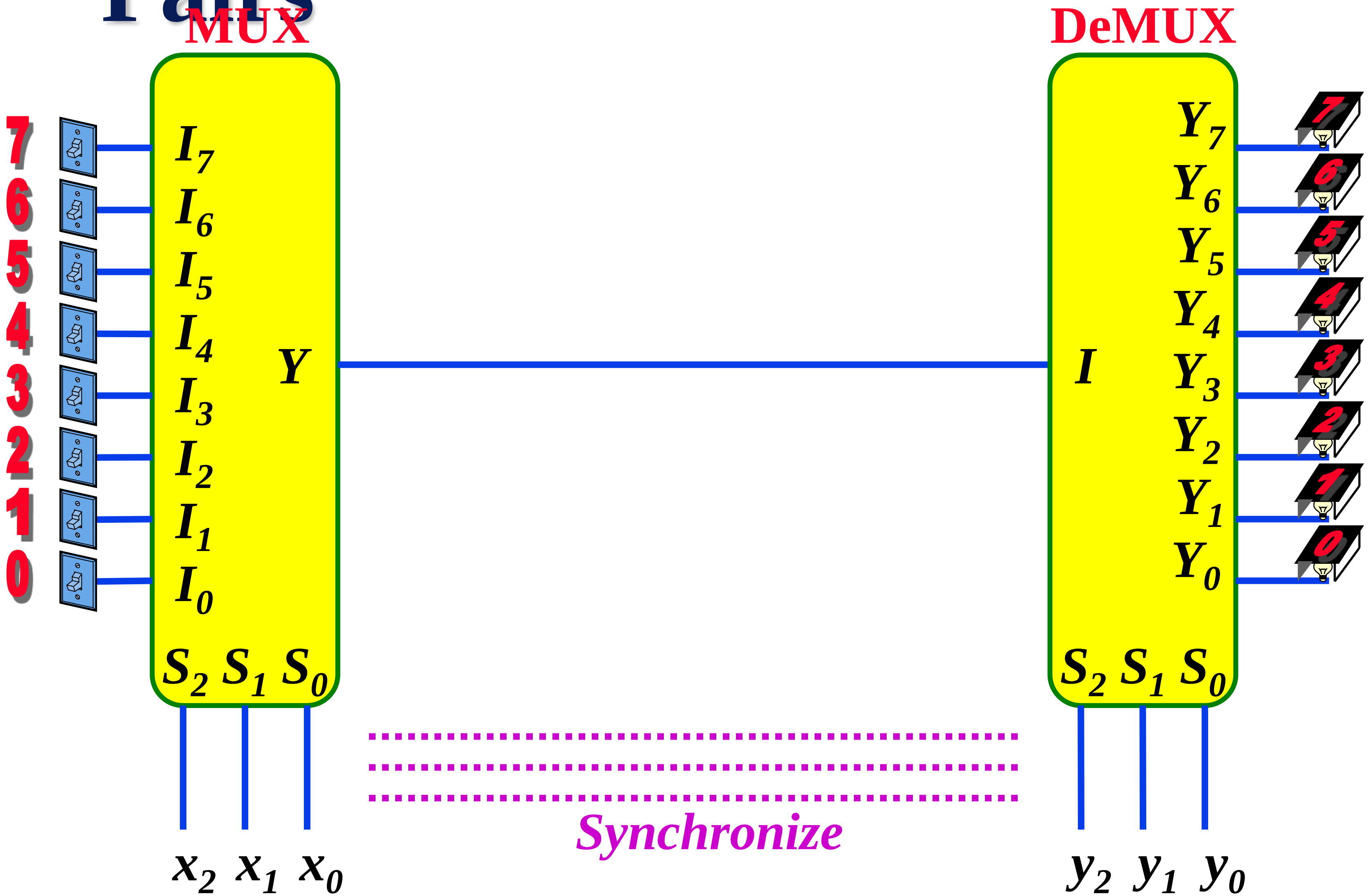






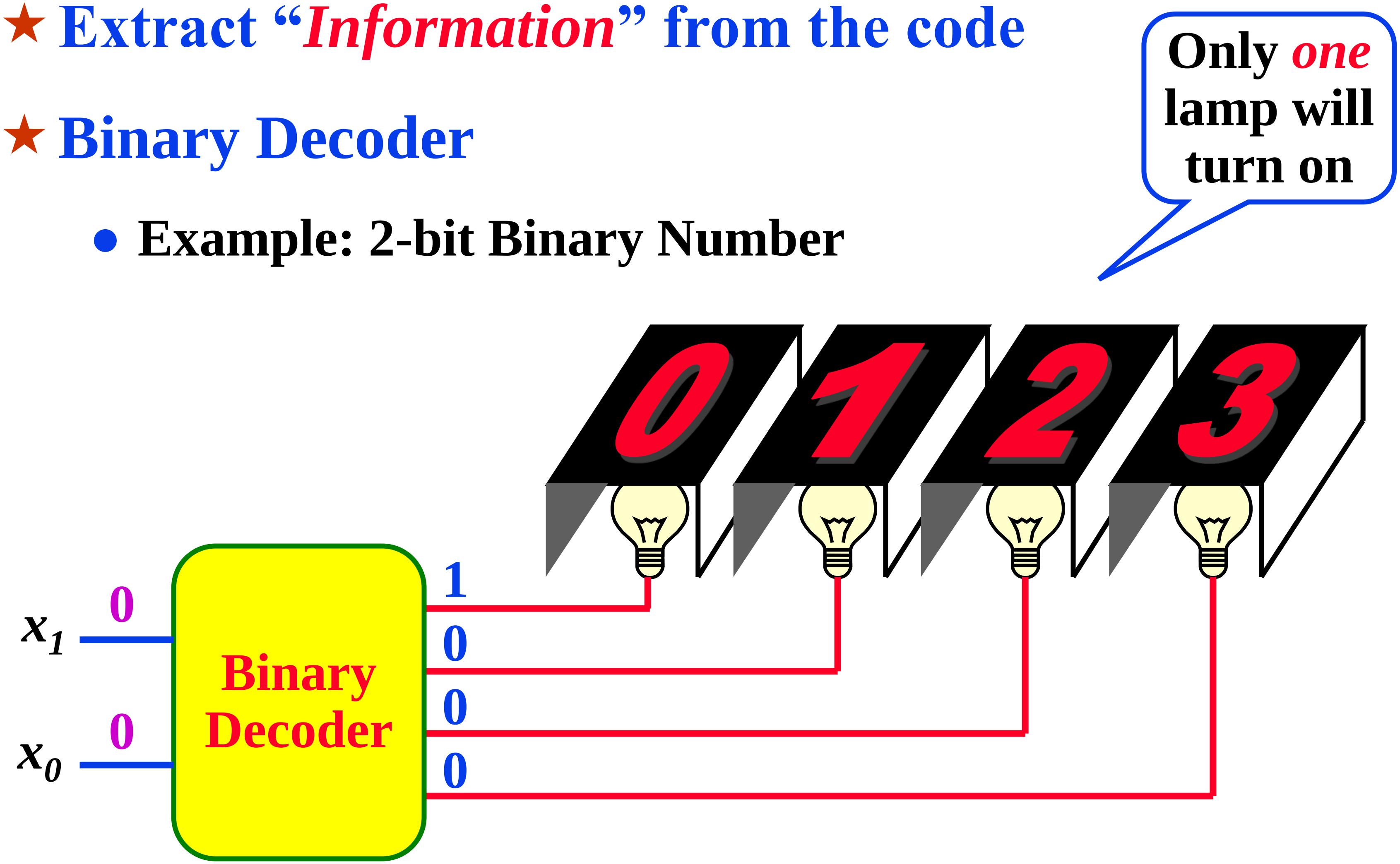
S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0

Multiplexer / DeMultiplexer Pairs



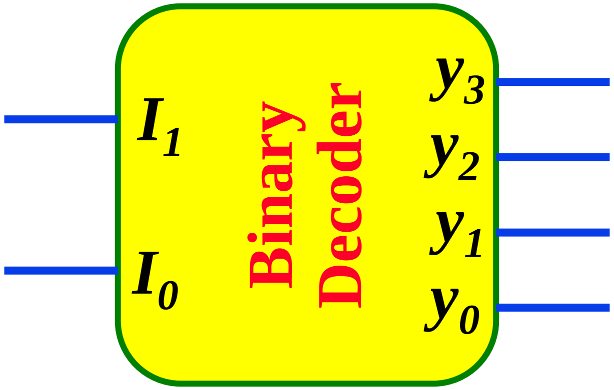
Decoders

- ★ Extract "*Information*" from the code
- ★ Binary Decoder
  - Example: 2-bit Binary Number

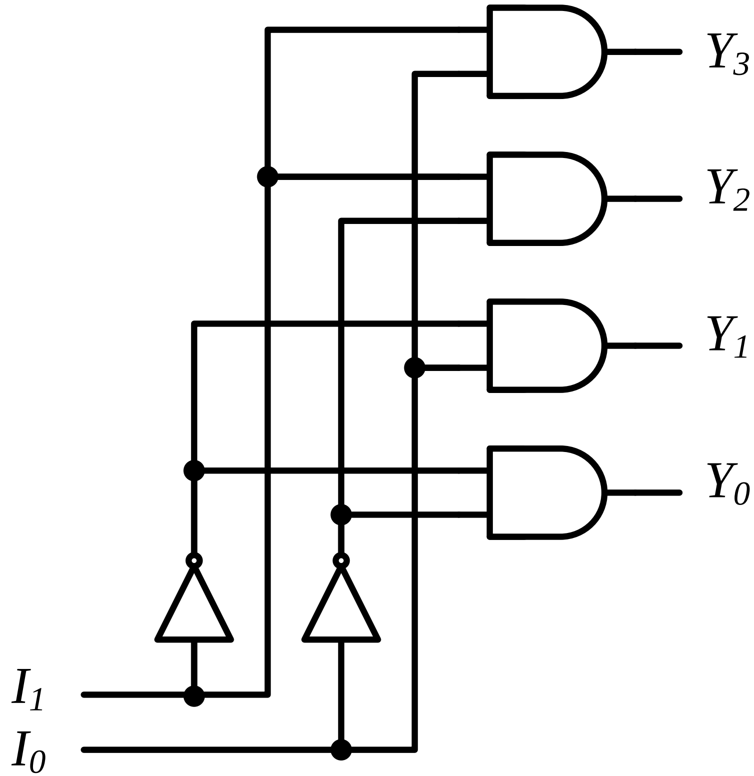


Decoders

★ **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

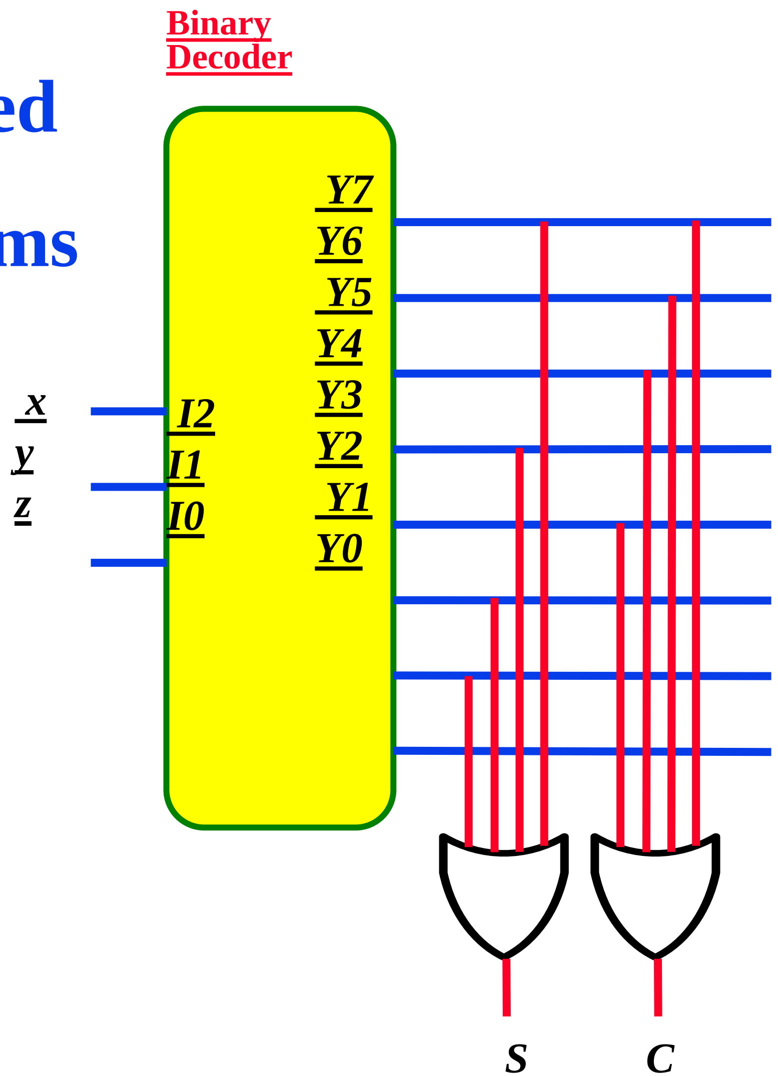


$$Y_3 = I_1\, I_0 \qquad Y_2 = I_1\, \bar{I}_0$$

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

Implementation Using Decoders

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

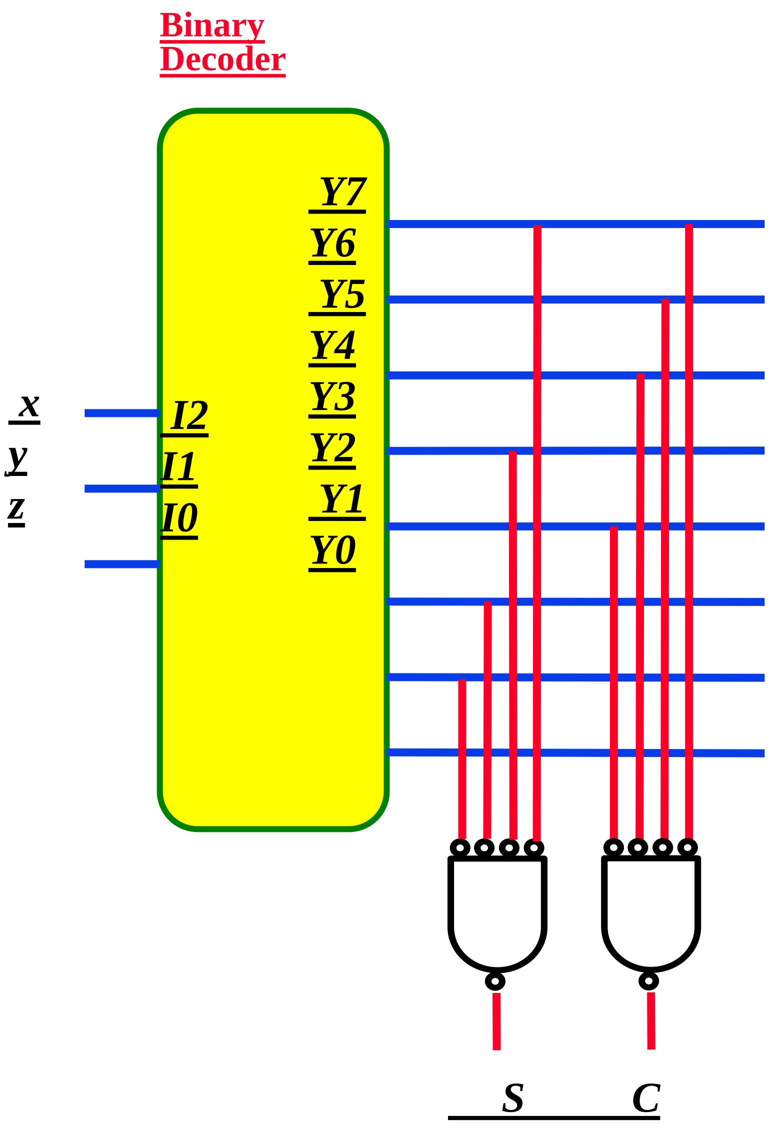


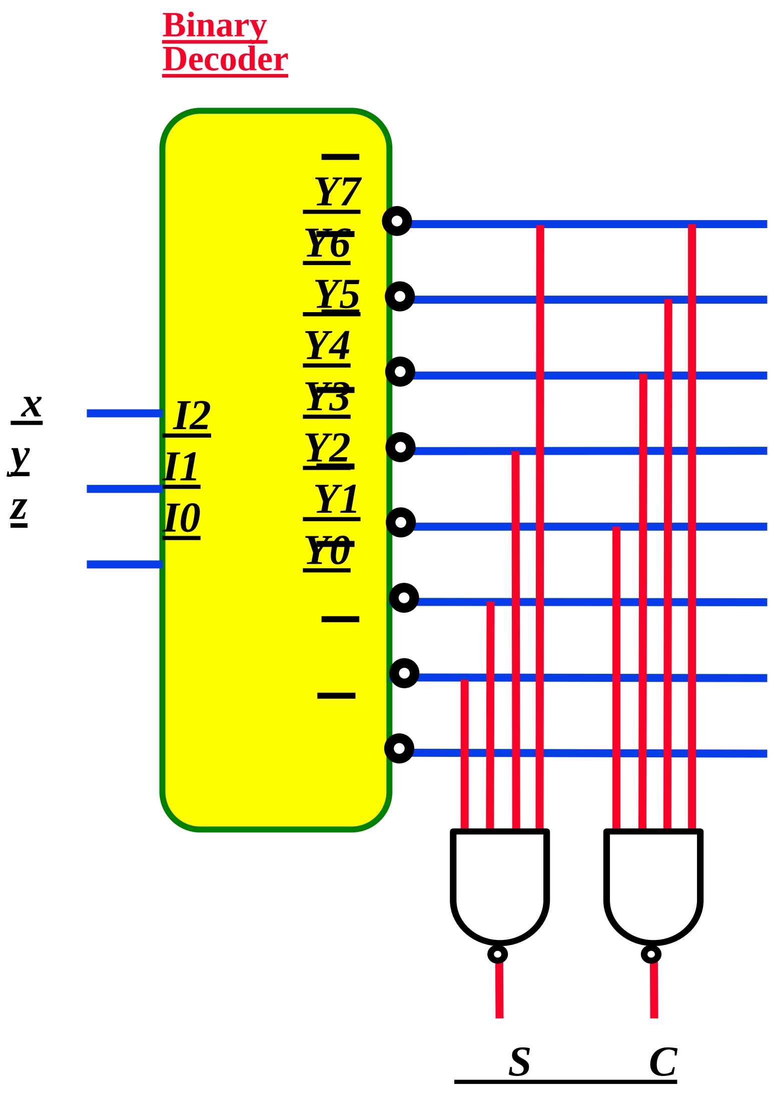
Example: **Full Adder**

$$S(x, y, z) = \sum(1, 2, 4, 7)$$

$$C(x, y, z) = \sum(3, 5, 6, 7)$$

Implementation Using Decoders





Decoding

- **Decoding - the conversion of an *n*-bit input code to an *m*-bit output code with $n \le m \le 2^n$ such that each valid code word produces a unique output code**
- **Circuits that perform decoding are called *decoders***
- **Here, functional blocks for decoding are**
  - **called *n*-to-*m* line decoders, where $m \le 2^n$, and**
  - **generate 2^n (or fewer) minterms for the *n* input variables**

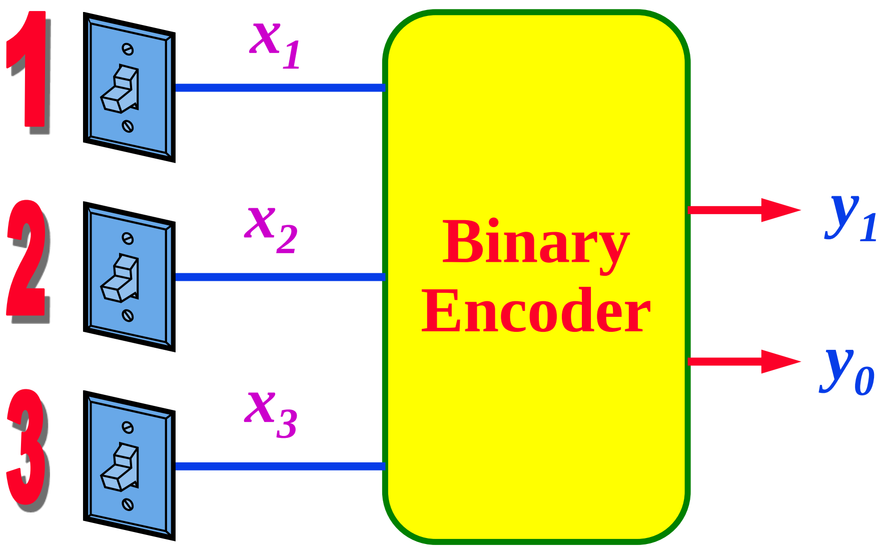
Encoders

★ **Put "*Information*" into code**

★ **Binary Encoder**

- **Example: 4-to-2 Binary Encoder**

Only *one* switch should be activated at a time

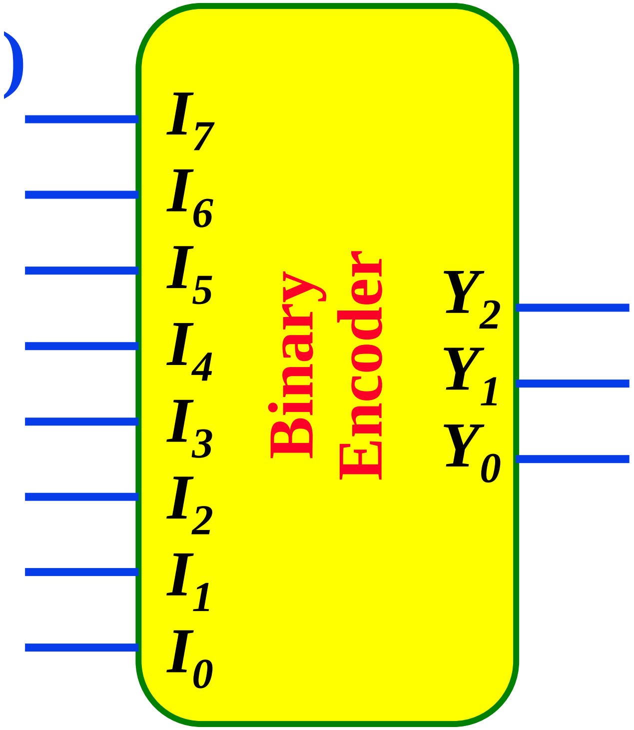


x_3	x_2	x_1	y_1	y_0
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
1	0	0	1	1

Encoders

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

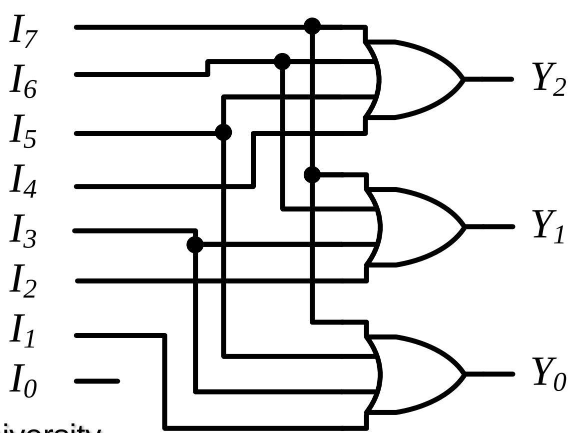
I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1



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

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

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



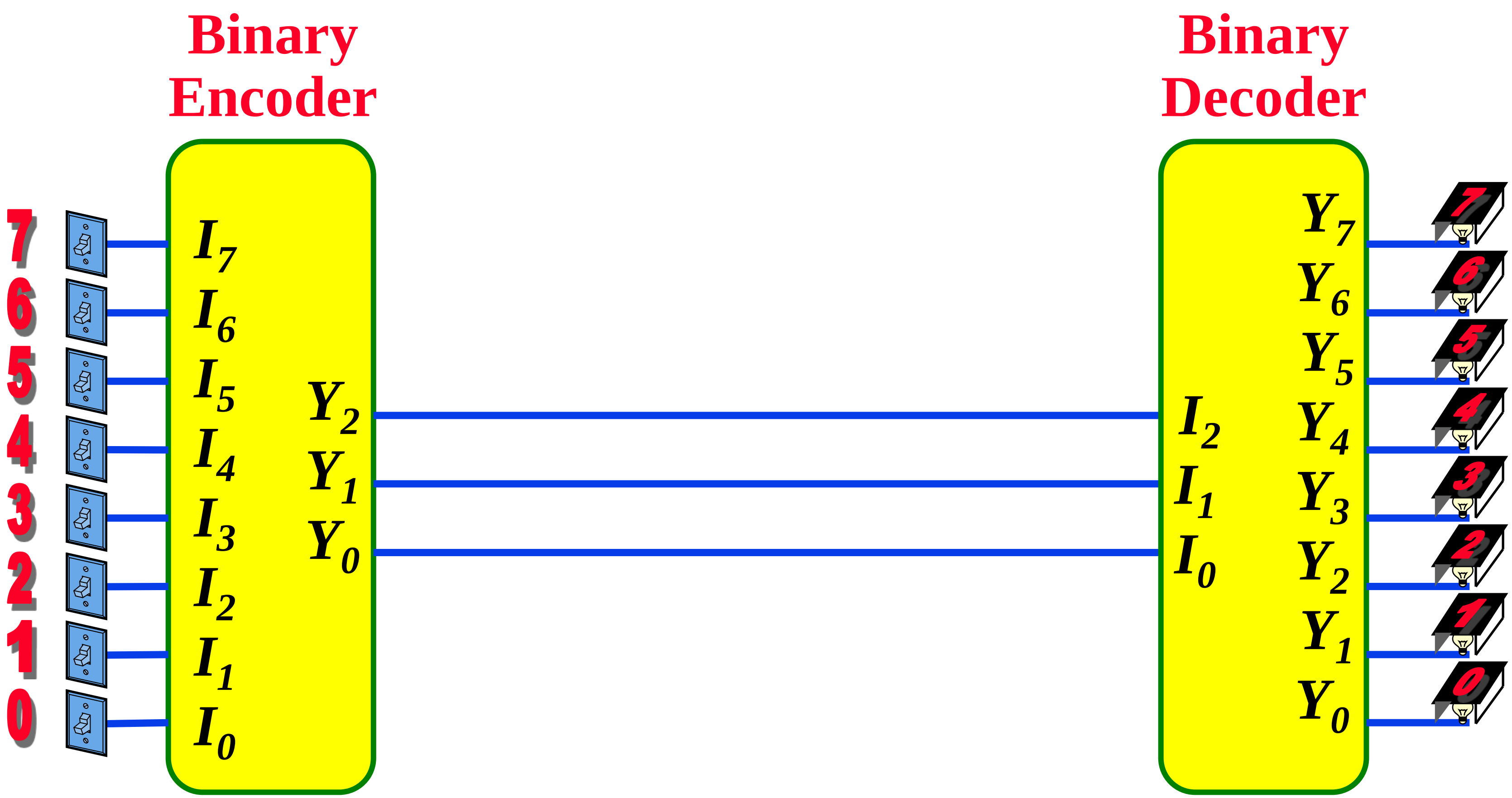
Encoding

- Encoding - the opposite of decoding - the conversion of an m-bit input code to a n-bit output code with $n \leq m \leq 2^n$ such that each valid code word produces a unique output code
- Circuits that perform encoding are called *encoders*
- An encoder has 2^n (or fewer) input lines and n output lines which generate the binary code corresponding to the input values
- Typically, an encoder converts a code containing exactly one bit that is 1 to a binary code corres-ponding to the position in which the 1 appears.

Encoder Example

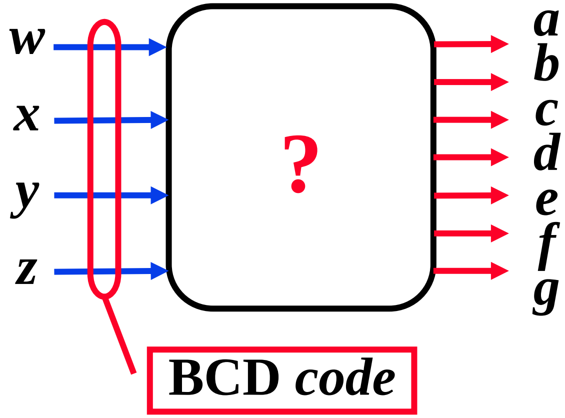
- A decimal-to-BCD encoder
  - Inputs: 10 bits corresponding to decimal digits 0 through 9, (D_0, …, D_9)
  - Outputs: 4 bits with BCD codes
  - Function: If input bit D_i is a 1, then the output (A_3, A_2, A_1, A_0) is the BCD code for i,
- The truth table could be formed, but alternatively, the equations for each of the four outputs can be obtained directly.

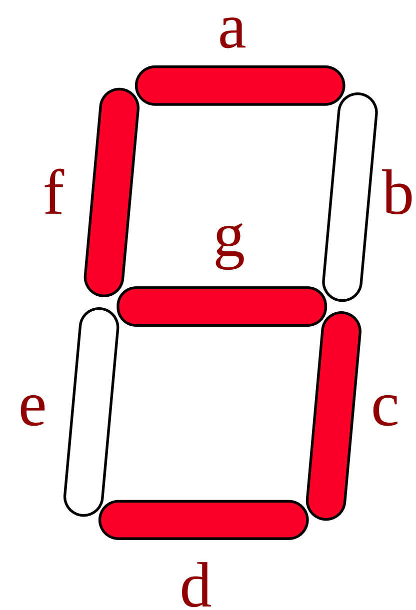
Encoder / Decoder Pairs

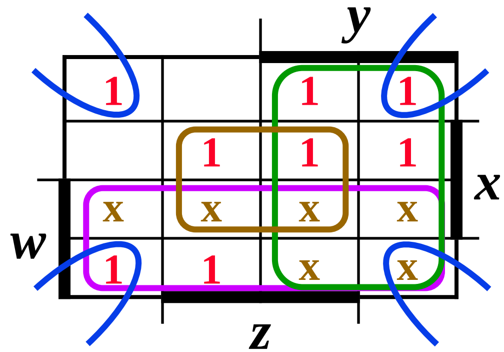


Seven-Segment Decoder

w x y z	*a b c d e f g*
0 0 0 0	1 1 1 1 1 1 0
0 0 0 1	0 1 1 0 0 0 0
0 0 1 0	1 1 0 1 1 0 1
0 0 1 1	1 1 1 1 0 0 1
0 1 0 0	0 1 1 0 0 1 1
0 1 0 1	1 0 1 1 0 1 1
0 1 1 0	1 0 1 1 1 1 1
0 1 1 1	1 1 1 0 0 0 0
1 0 0 0	1 1 1 1 1 1 1
1 0 0 1	1 1 1 1 0 1 1
1 0 1 0	x x x x x x x
1 0 1 1	x x x x x x x
1 1 0 0	x x x x x x x
1 1 0 1	x x x x x x x
1 1 1 0	x x x x x x x
1 1 1 1	x x x x x x x







$a = w + y + xz + x'z'$

$b = \ldots$

$c = \ldots$

$d = \ldots$

Combinational Function Implementation

- **Alternative implementation techniques:**
  - **Decoders and OR gates**
  - **Multiplexers (and inverter)**
  - **ROMs**
  - **PLAs**
  - **PALs**
  - **Lookup Tables**
- **Can be referred to as *structured implementation methods* since a specific underlying structure is assumed in each case**