Introduction to Digital Systems

Part II (4 lectures)

2022/2023

Combinational Logic Blocks

Arnaldo Oliveira, Augusto Silva, Iouliia Skliarova

Lecture 5 contents

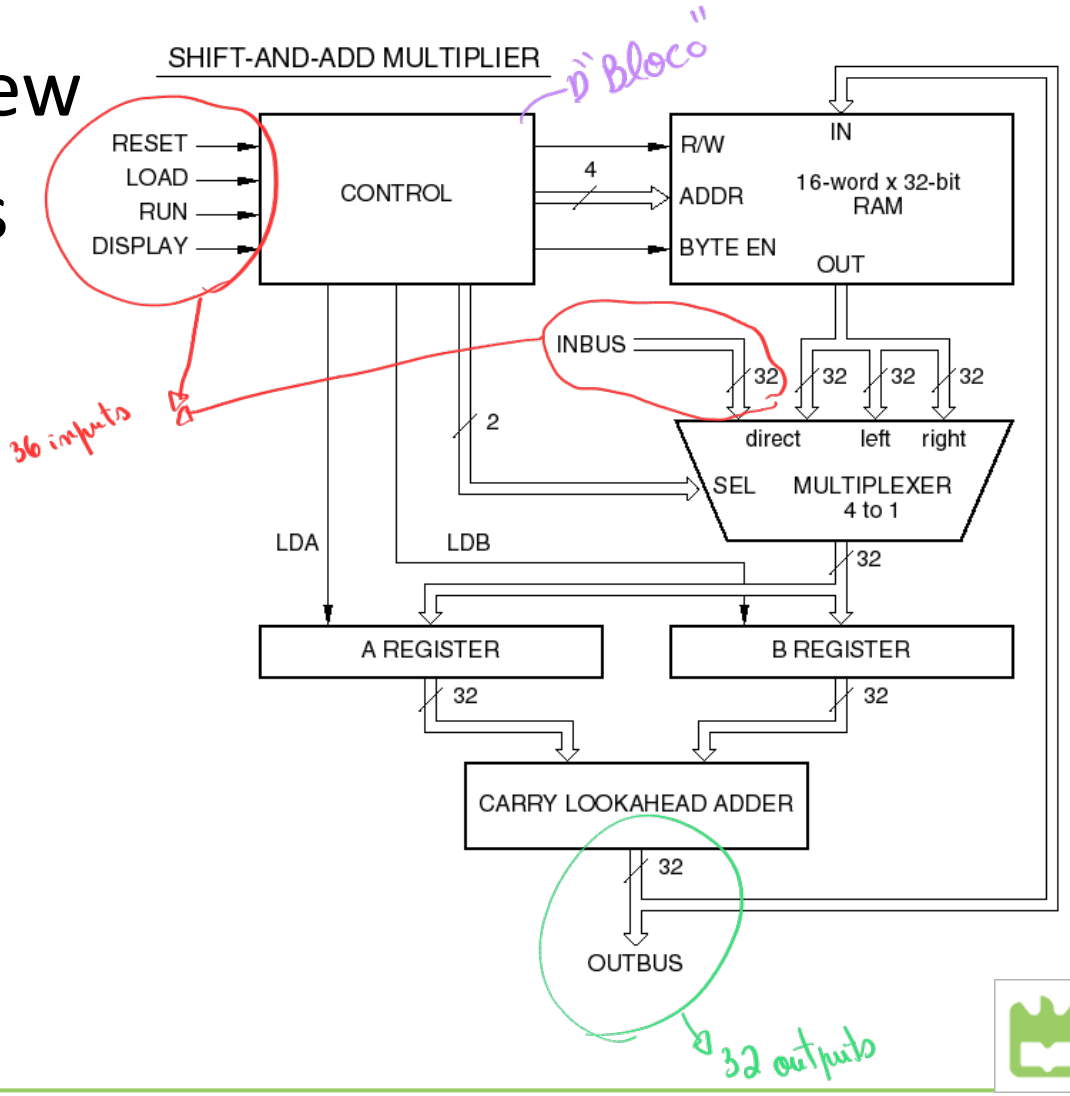
- About project documentation
- Block oriented combinational logic design
- Decoders
- Encoders

Documentation

- Essential in the whole project development cycle
- Block diagrams
- Logic circuits
- Hardware Description Languages
  - (VHDL, Verilog)
- Timing diagrams
- Electrical circuits
- Component specs

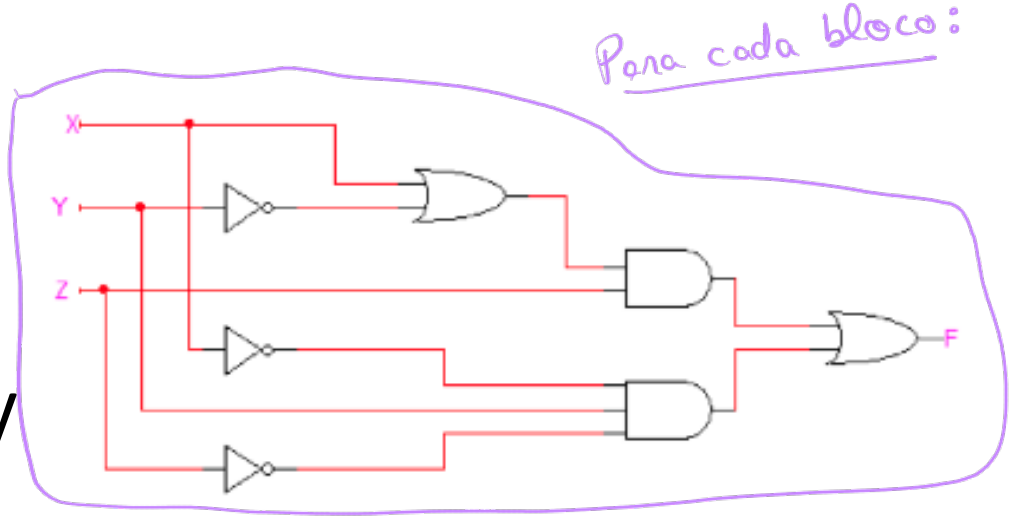
Block Diagrams

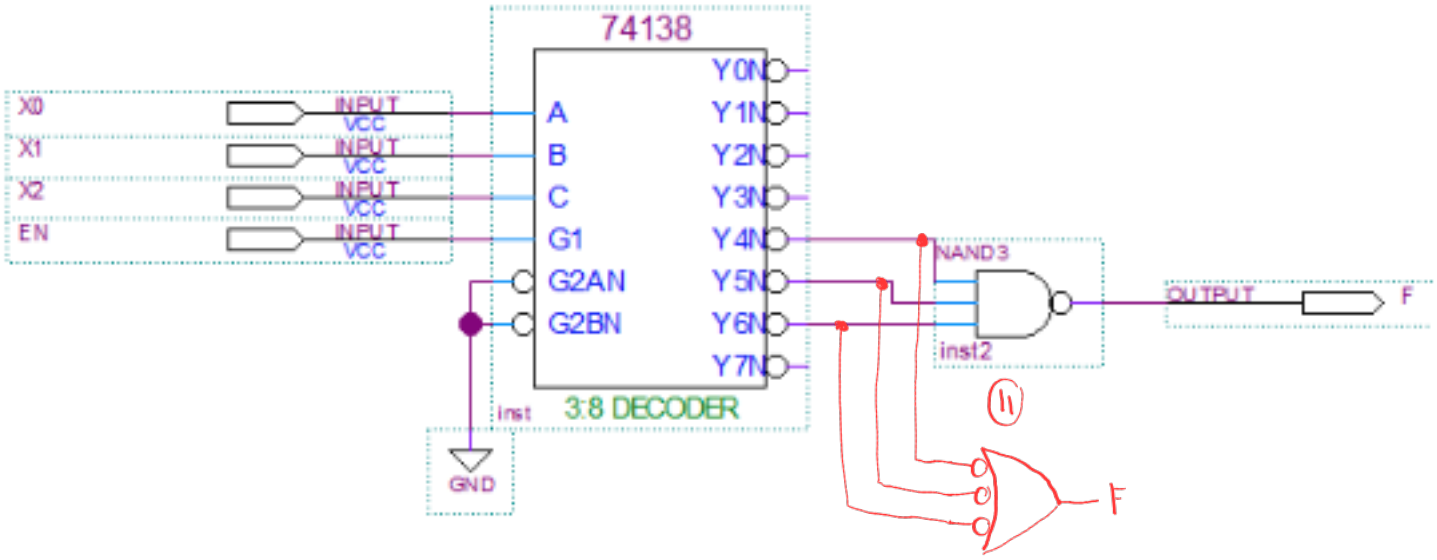
- Architectural view
- Functional hints



Logic Circuits

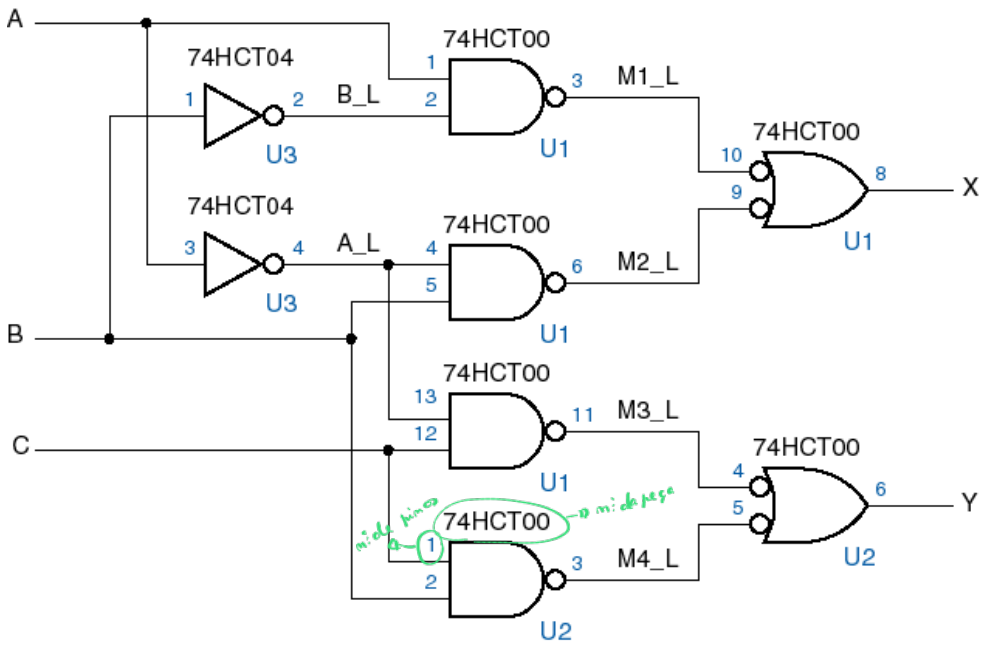
- Elementary gates
- Logic blocks
- Signal naming only





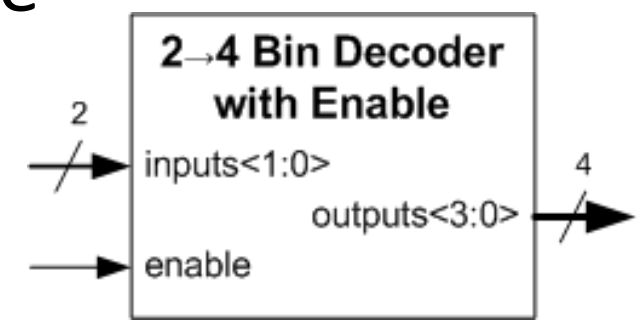
Electrical Circuits

- Component references
- Pin references
- Signal naming
- Connectors
- …



HDL's

- Coming soon …
- Modeling hardware with code
- A flavor of VHDL

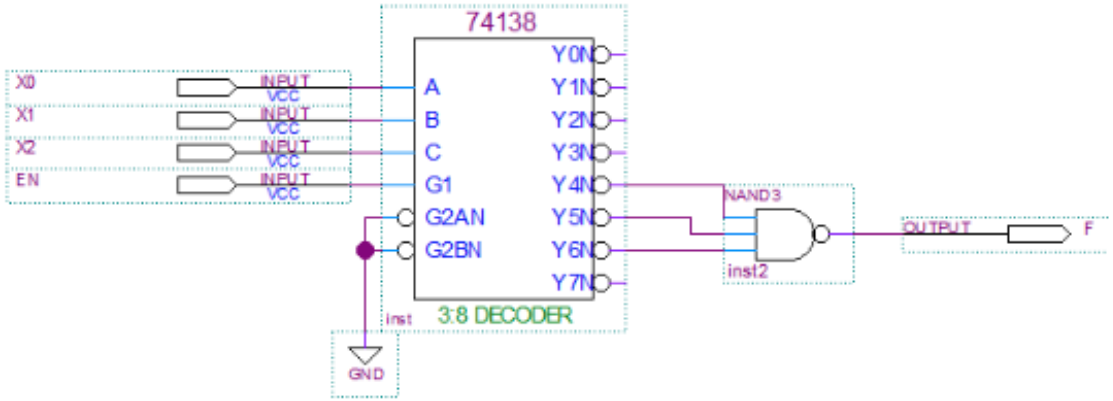


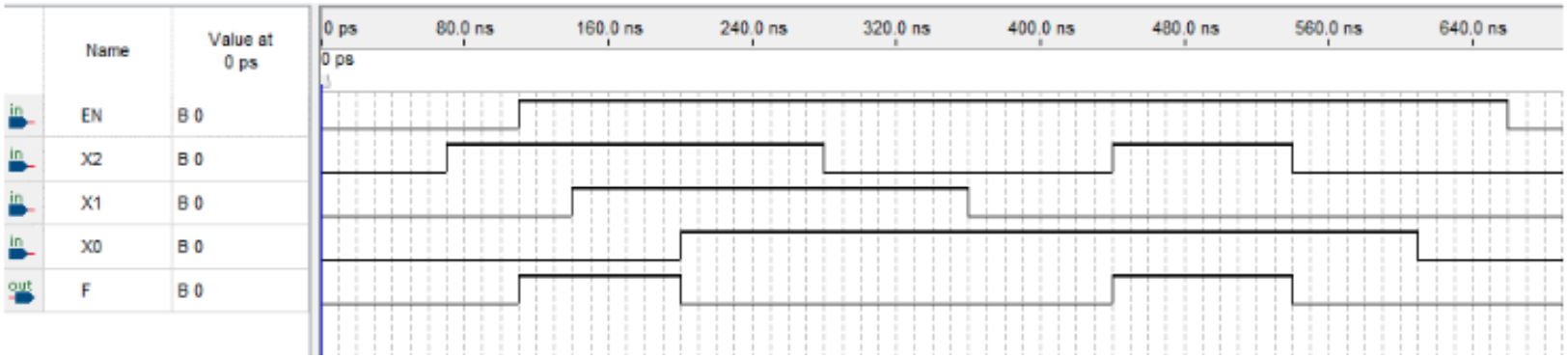

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity Dec2_4En is
  port(enable  : in  std_logic;
       inputs  : in  std_logic_vector (1 downto 0);
       outputs : out std_logic_vector (3 downto 0));
end Dec2_4En;
architecture BehavAssign of Dec2_4En is
begin
    outputs <= "0000" when (enable = '0')  else
               "0001" when (inputs = "00") else
               "0010" when (inputs = "01") else
               "0100" when (inputs = "10") else
               "1000";
end BehavAssign
```

Timing Diagrams

- A core skill for analysis and simulation





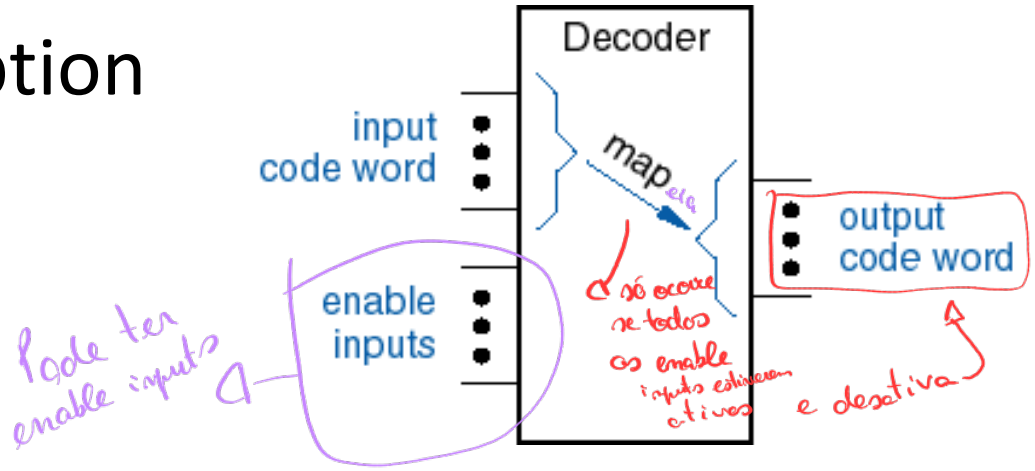
Beyond elementary gates

- Combinational logic blocks
- Encapsulation of specific behavior within a functional block
  - Decoders / Encoders
  - Multiplexers / Demutiplexers
  - Arithmetic blocks
    - Adders / Subtractors
    - Comparators
    - Multipliers
    - Arithmetic Logic Units (ALU)

Vamos estudar

Decoders

- Generic description

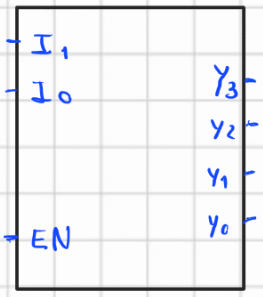


- A decoder implements a mapping between and input code word and output code word
- Behavior is externally controlled by ”enable” inputs

Descodificador binário ($n : 2^n$)

→ Apenas 1 saída pode estar ativa!

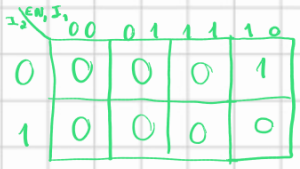
dec. 2:4 ou dec 3:8 ou dec 4:16



EN	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

0 no EN, desativa os saídas. → Ignora os inputs

Já quando EN = 1, ele depende das entradas



M.K.

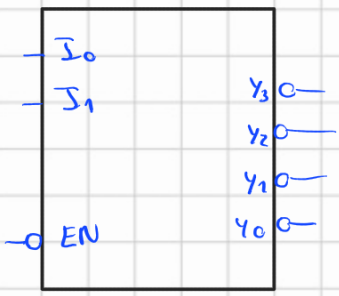
$$y_0 = EN \cdot \overline{I_1} \cdot \overline{I_0}$$

$$y_1 = EN \cdot \overline{I_1} \cdot I_0$$

$$y_2 = EN \cdot I_1 \cdot \overline{I_0}$$

$$y_3 = EN \cdot I_1 \cdot I_0$$

dec 2:4

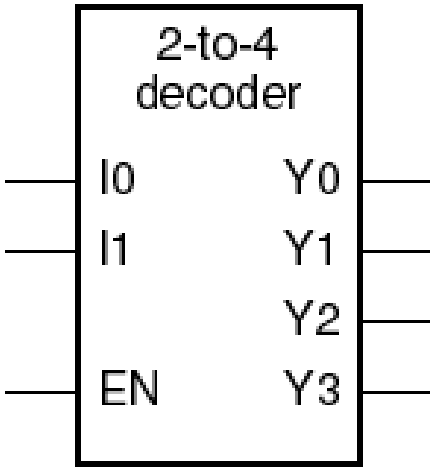


EN_L (negado!)	I_1	I_0	Y_3_L	Y_2_L	Y_1_L	Y_0_L
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1
1	X	X	1	1	1	1

Ativa como 0

n:2^n decoders

- Restrict the #inputs – #outputs relation to n:2^n
- Impose a "1 out of 2^n" output code
- Then we have a standard n:2^n binary decoder



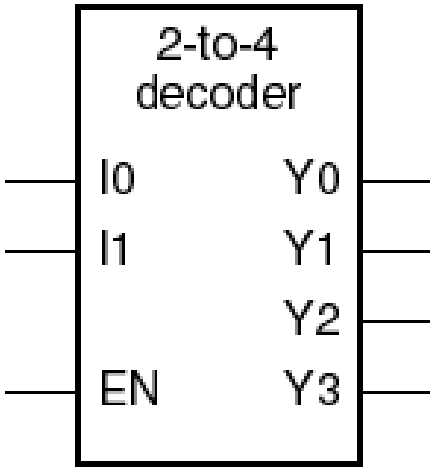
Inputs			*Outputs*			
EN	I1	I0	Y3	Y2	Y1	Y0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

"x" (don't care)

2:4 Decoder

- Things we have to know:

Block diagram



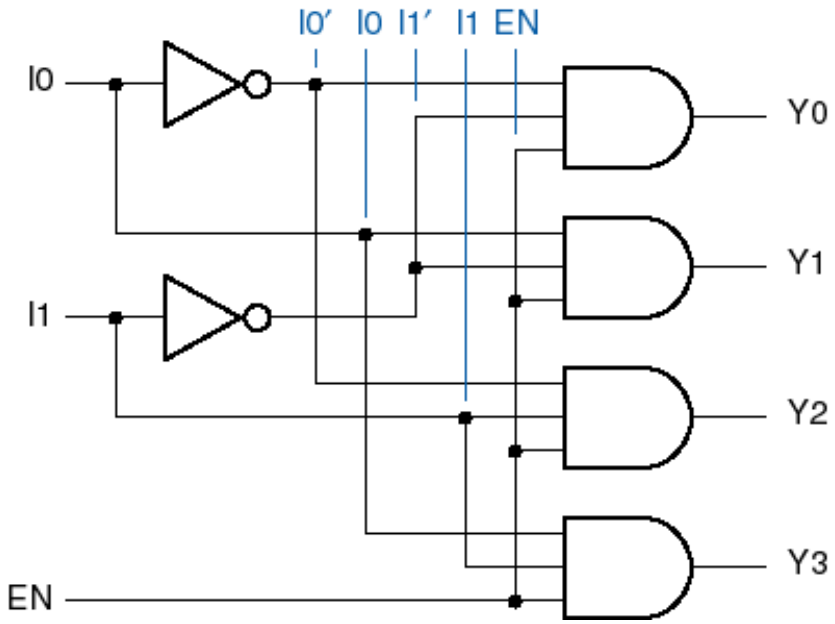
Functional Truth Table

Inputs			*Outputs*			
EN	I1	I0	Y3	Y2	Y1	Y0
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

"1" out of 4 code words
Look at the diagonal layout

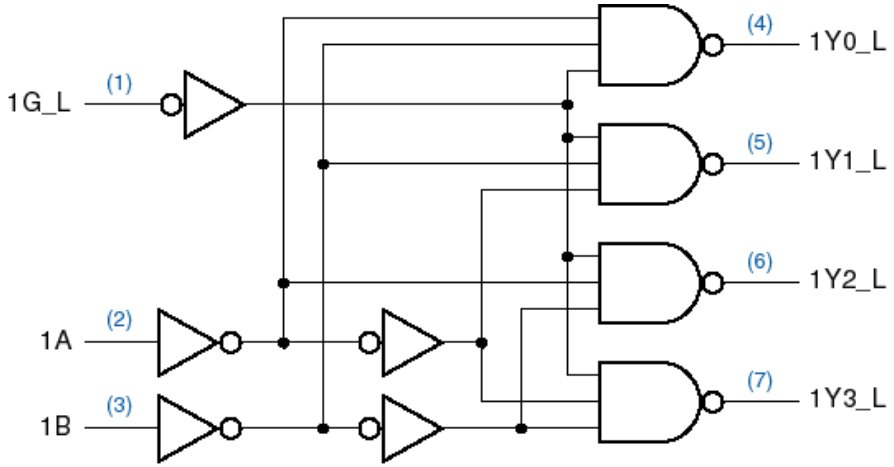
2:4 Decoder

- Things we have to know:
  - Write the output equations
  - Be aware of the role of the EN input



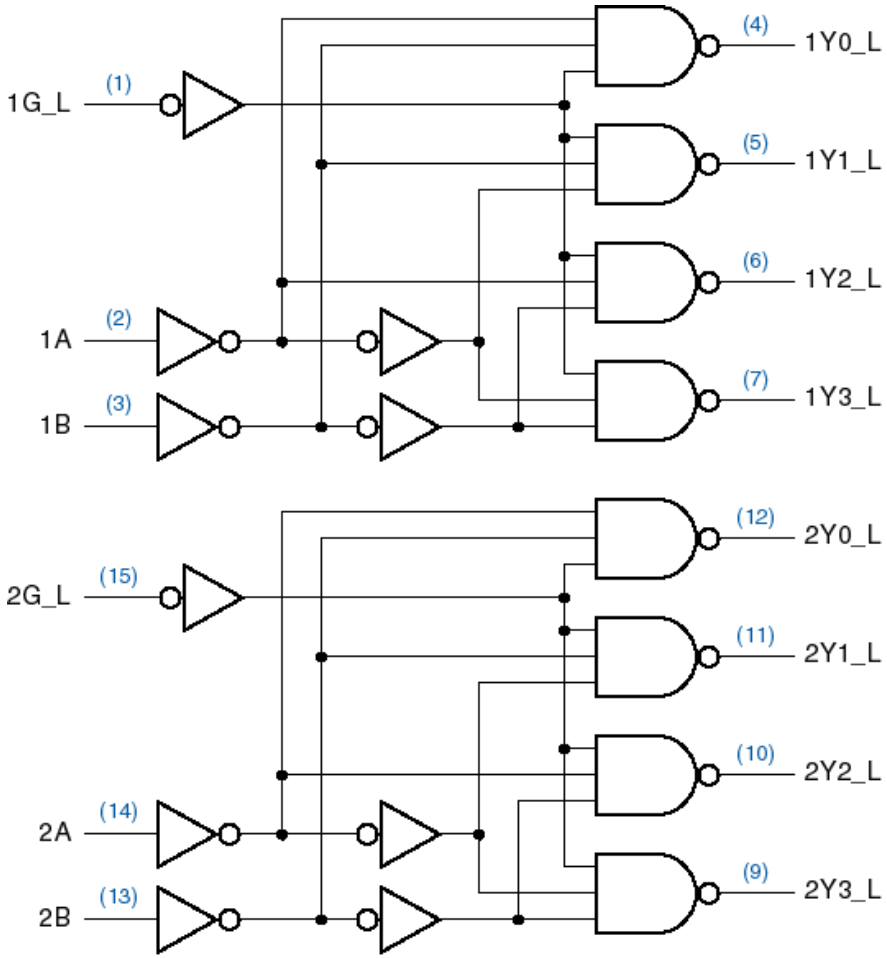
2:4 Decoder

- Active-low version
- Write the output equations
- Build the truth table

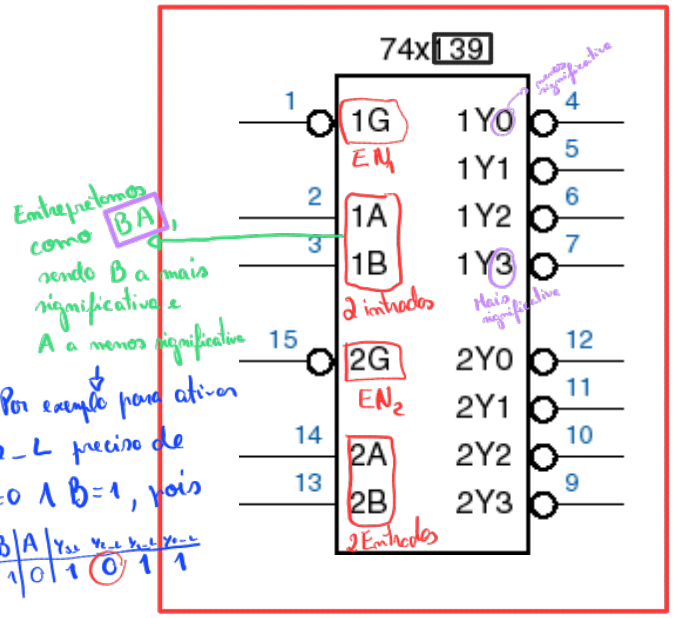


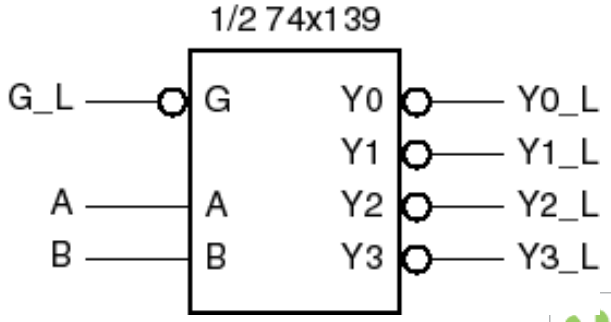
As saídas são negadas pois, para fazer funções basta adicionar uma NAND

Commercial Models



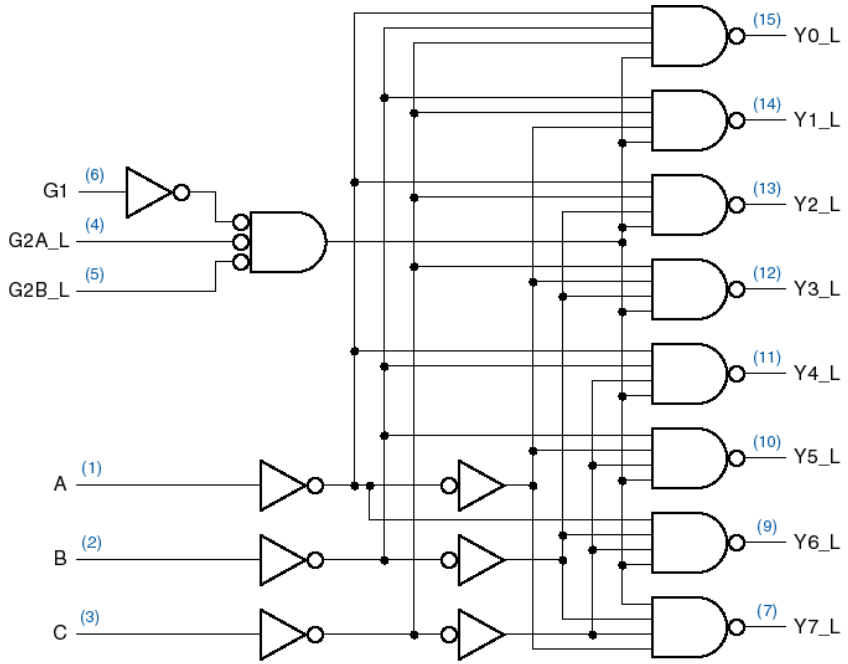
Dec. 2:4, dual. Contém 2

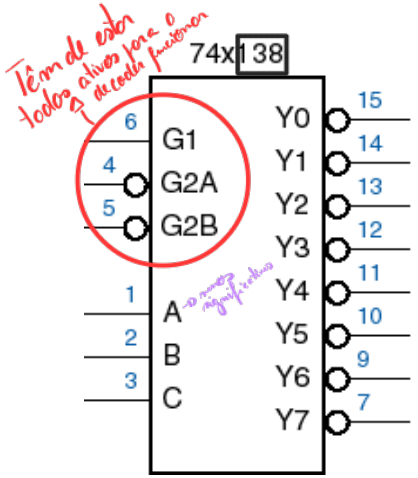




3:8 Decoder

- Twofold increase in the number of output products



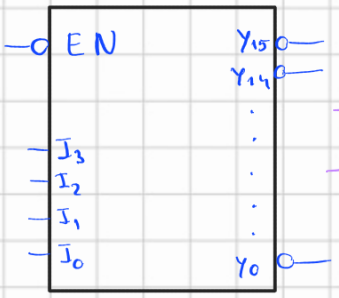


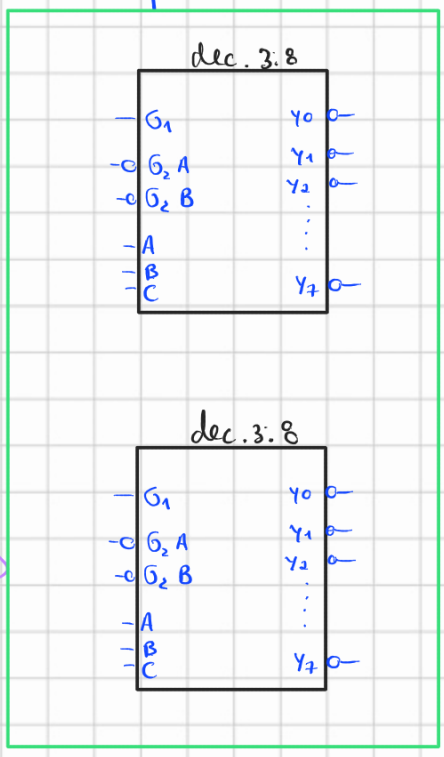
Build the Truth Table for the "enabled" mode

G1	G2A_L	G2B_L	C	B	A	Saída ativa
0	X	X	X	X	X	nenhuma
1	0	0	0	0	0	Y0_L
			⋮			
			1	1	1	Y7_L
1	1	X	X	X	X	nenhuma
1	X	1	X	X	X	nenhuma

dec 4:16, apenas com dec. 3:8

Queremos!

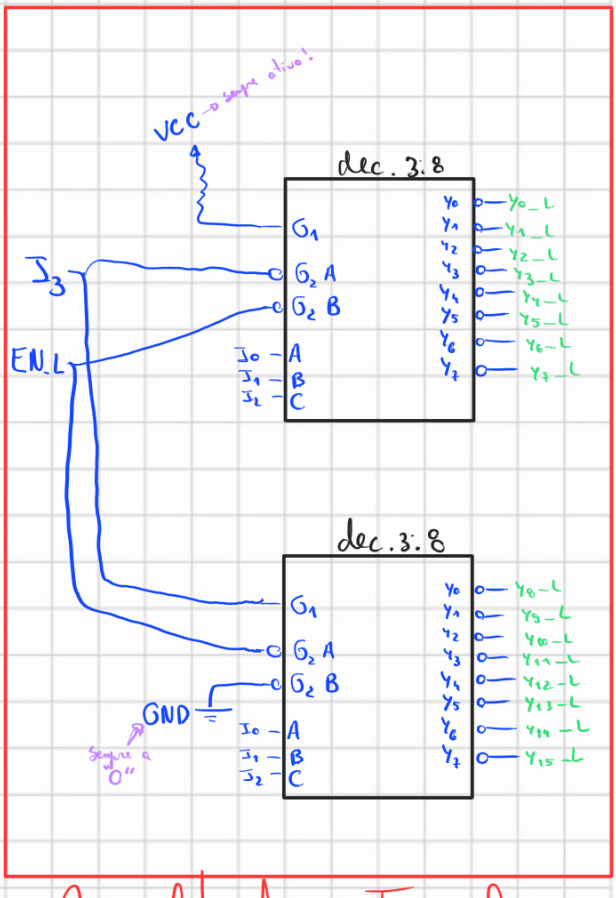




EN_L	I_3	I_2	I_1	I_0	Saída ativa
"0"	0	0	0	0	Y_0_L
	0	0	0	1	Y_1_L
	0	0	1	0	Y_2_L
	0	0	1	1	Y_3_L
	0	1	0	0	Y_4_L
	0	1	0	1	Y_5_L
	0	1	1	0	Y_6_L
	0	1	1	1	Y_7_L
	1	0	0	0	Y_8_L
	1	0	0	1	Y_9_L
	1	0	1	0	Y_{10}_L
	1	0	1	1	Y_{11}_L
	1	1	0	0	Y_{12}_L
	1	1	0	1	Y_{13}_L
	1	1	1	0	Y_{14}_L
	1	1	1	1	Y_{15}_L
1	X	X	X	X	nenhuma

Quando $\overline{I_3}=0$

Quando $\overline{I_3}=1$

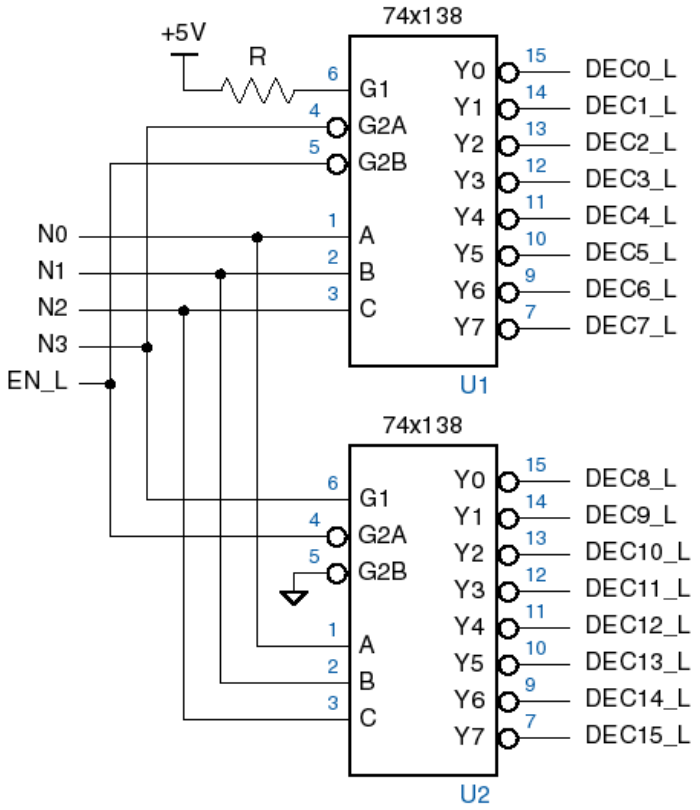


Resultado Final

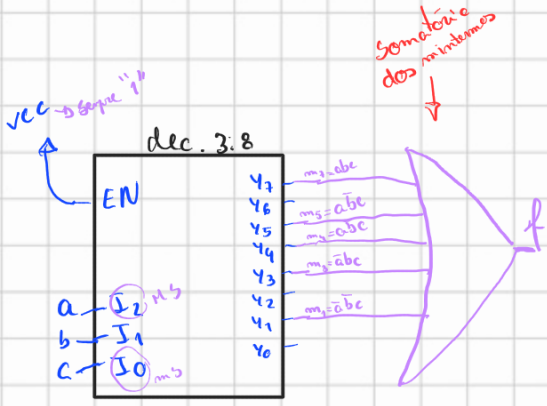
Também é possível fazer um dec. 4:16 só com dec. 2:4 (com 5)

Scaling-up

- Decoder cascades
  - 4:16 with 2x(3:8)
  - Look at the role of the enable inputs
  - Figure out other cascading structures:
    - 4:16 from 2:4 Decoder blocks
    - 5:32 from 2:4 + 3:8 Decoder blocks



$$f(a,b,c) = a\bar{b} + c$$
$$\underset{MS}{a}\ \underset{}{b}\ \underset{ms}{c} \qquad 1\,0\,1$$

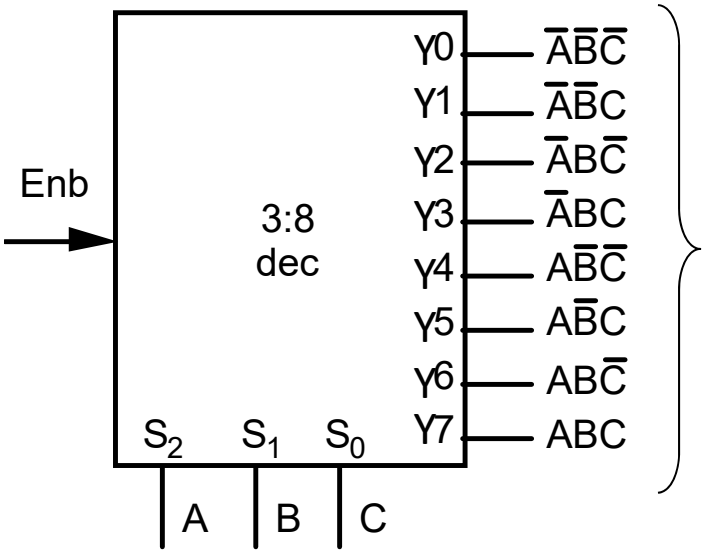


	a	b	c	f
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

$$f(a,b,c) = \sum m(1,3,4,5,7)$$
$$= \bar{a}\bar{b}c + \bar{a}bc + a\bar{b}\bar{c} + a\bar{b}c + abc$$

The Decoder as a minterm generator

- Recall the "internals" of the decoding block



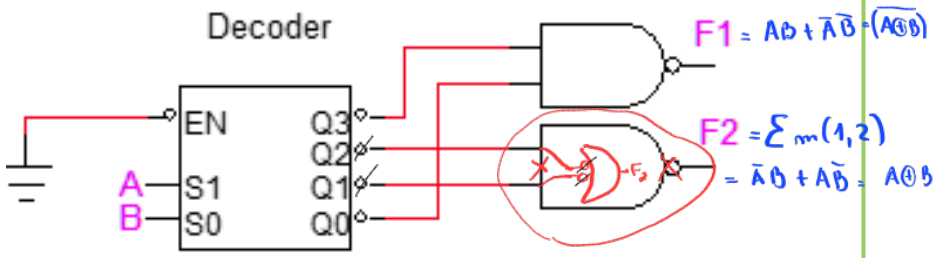
Each output equation is an "enabled" minterm

$$Y_i = Enb.m_i(A,B,C) \qquad i = 0,\ldots,2^n - 1$$

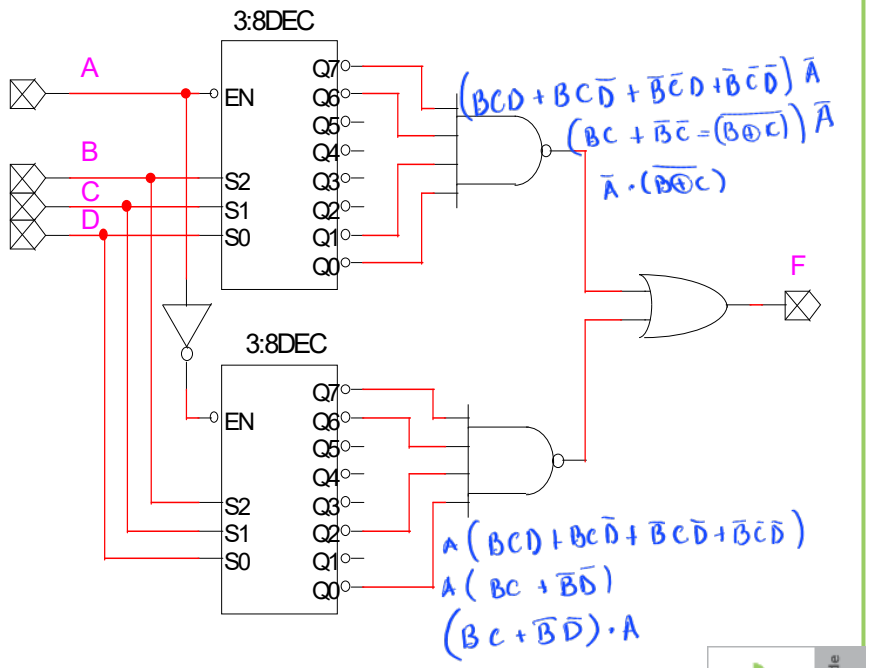
- A n:2^n binary decoder is an implicit minterm generator for any *n* variables Boolean function

Canonical Implementations

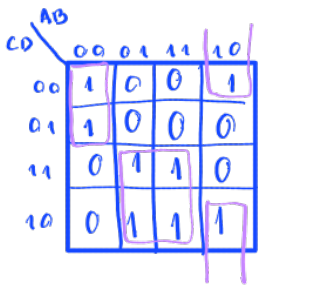
- Write the SOP form for F_1 and F_2



- Obtain a minimal SOP form *F(A,B,C,D)*

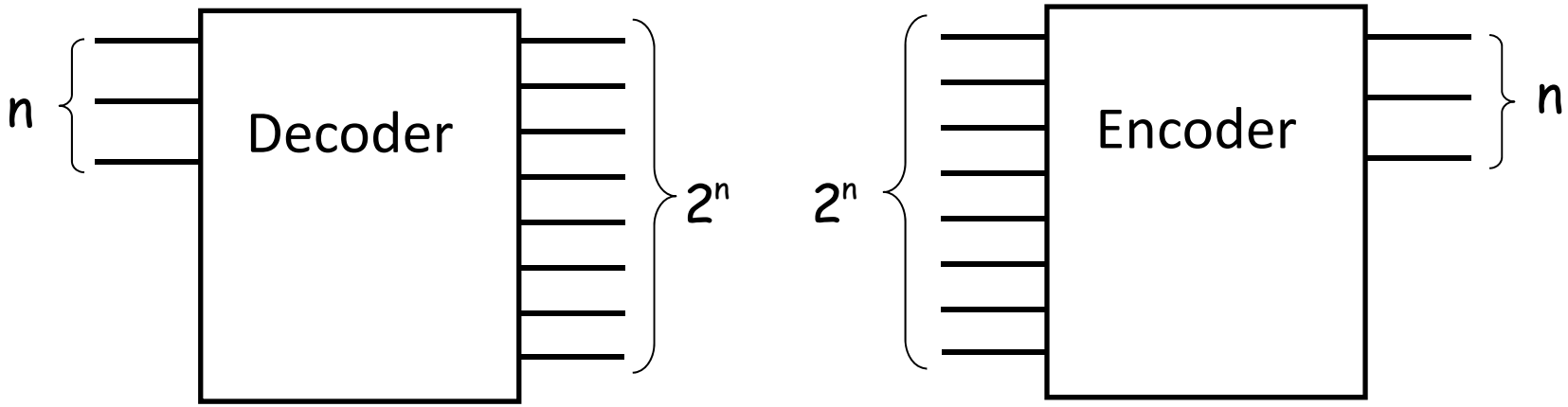


A	B	C	D	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1



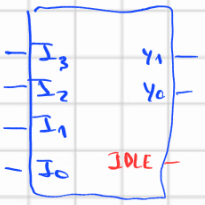
Encoders

- Functional inverse of the decoder



enc $2^m : 4$

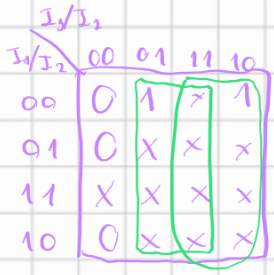
enc $4 : 2$



I_3	I_2	I_1	I_0	Y_1	Y_0	IDLE
0	0	0	1	0	0	
0	0	1	0	0	1	
0	1	0	0	1	0	
1	0	0	0	1	1	
0	0	0	0	0	0	1

Prob!

↳ inválido



Não é necessário mas pode dar geito

$Y_1 = I_3 + I_2$

2 OR
1 NAND

$Y_0 = I_3 + I_1$

$IDLE = \overline{I_3}\,\overline{I_2}\,\overline{I_1}\,\overline{I_0}$

Para um enc 8:3 é basicamente igual, mas com expressões maiores

Nestes encoders, não resolvemos o problema de colocarmos 2 inputs

↓ Para isso, usamos um sistema de prioridades!

I_3	I_2	I_1	I_0	Y_1	Y_0	IDLE
1	X	X	X	1	1	
0	1	X	X	1	0	
0	0	1	X	0	1	
0	0	0	1	0	0	
0	0	0	0	0	0	1

não esquecer!

Teorema → $a + \bar{a}\,b = a + b$

$Y_1 = I_3 + \overline{I_3}\,I_2 = I_3 + I_2$

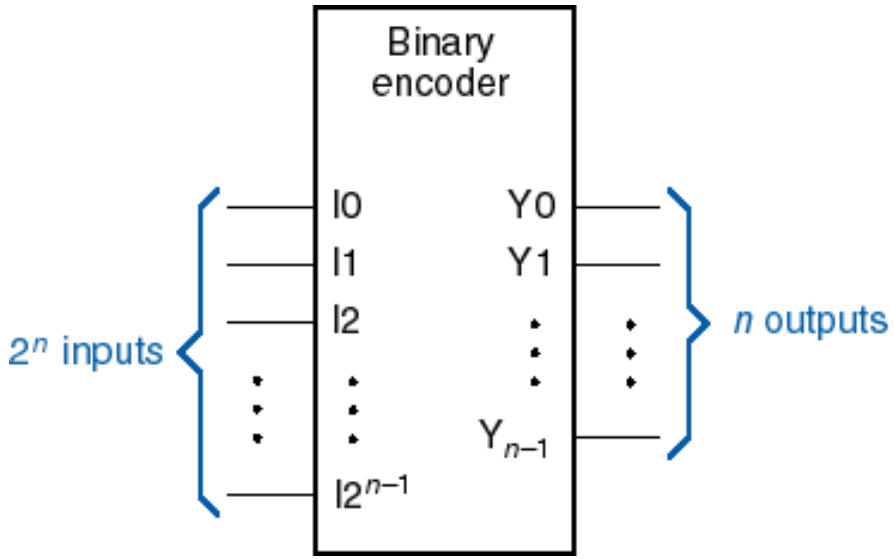
$Y_0 = I_3 + \cancel{\overline{I_3}}\,\overline{I_2}\,I_1 = I_3 + \overline{I_2}\,I_1$

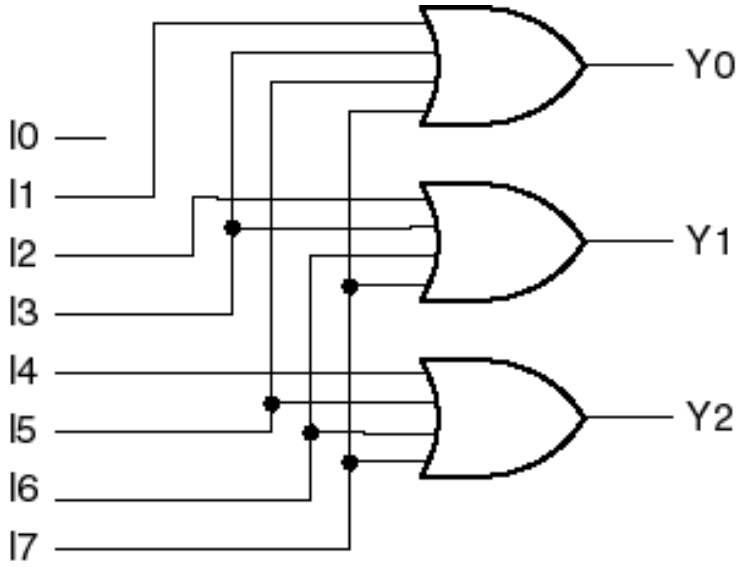
Usamos o teorema!

$IDLE = \overline{I_3}\,\overline{I_2}\,\overline{I_1}\,\overline{I_0}$

“Naïve” Binary encoders

- Why naïve?



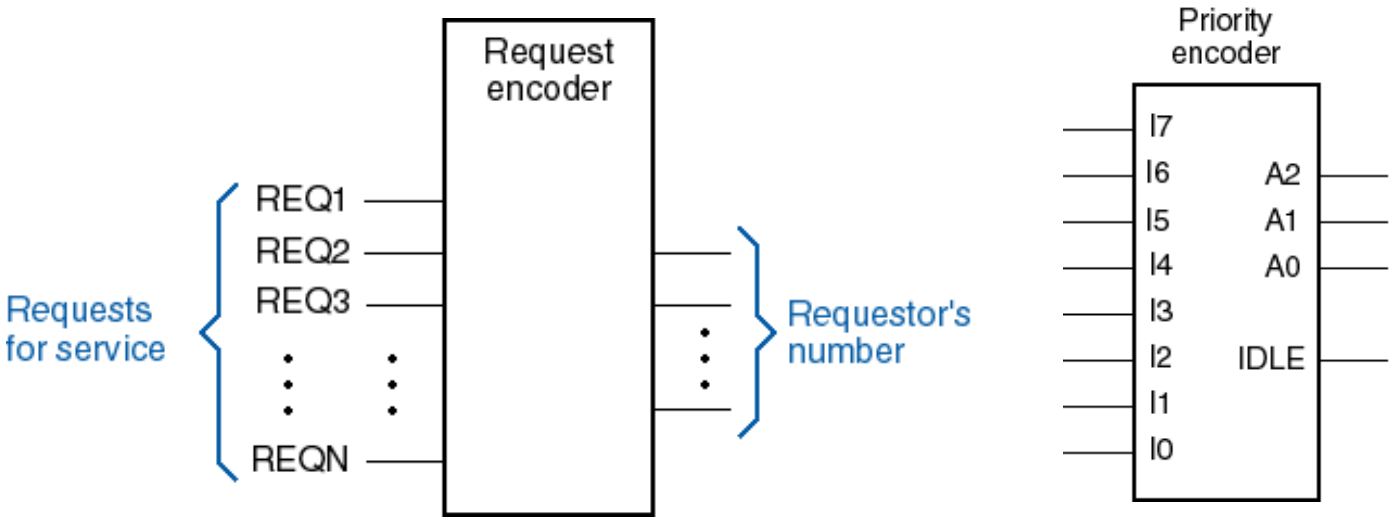


Write the truth table

$$Y_0 = I_1 + I_3 + I_5 + I_7$$
$$Y_1 = I_2 + I_3 + I_6 + I_7$$
$$Y_2 = I_4 + I_5 + I_6 + I_7$$

The Priority issue

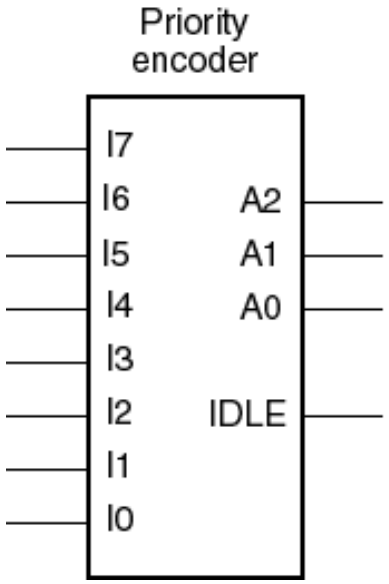
- In the previous "naïve" encoder what happens when I_3 and I_5 are asserted?



- Conflicts are resolved using a priority strategy

The Priority Encoder

- Let's define the following set of internal signals H_n : H_0, H_1, …, H_7



$$H_7 = I_7$$
$$H_6 = I_6.\bar{I}_7$$
$$H_5 = I_5.\bar{I}_6\bar{I}_7$$
$$\vdots$$
$$H_0 = \underline{I_0.\bar{I}_1}\bar{I}_2 \square \ \bar{I}_6\bar{I}_7$$

$$IDLE = \sum_{n=0}^{2^n-1} I_n = \prod_{n=0}^{2^n-1} \bar{I}_n$$

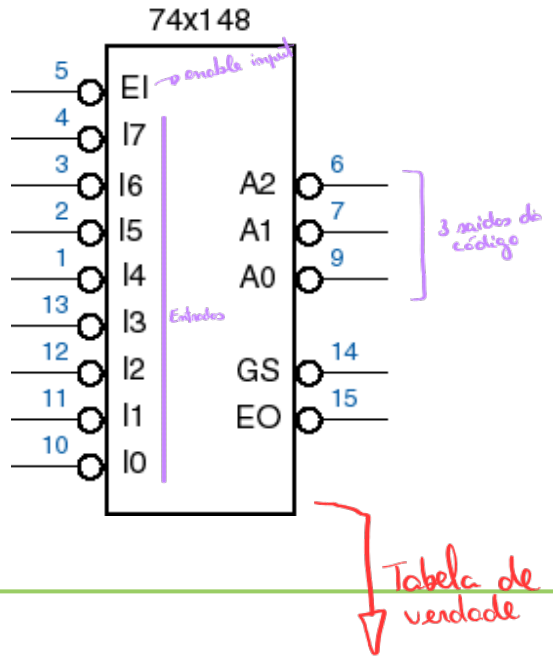
- Verify that priority is achieved when

$$A_0 = H_1 + H_3 + H_5 + H_7$$
$$A_1 = H_2 + H_3 + H_6 + H_7$$
$$A_2 = H_4 + H_5 + H_6 + H_7$$

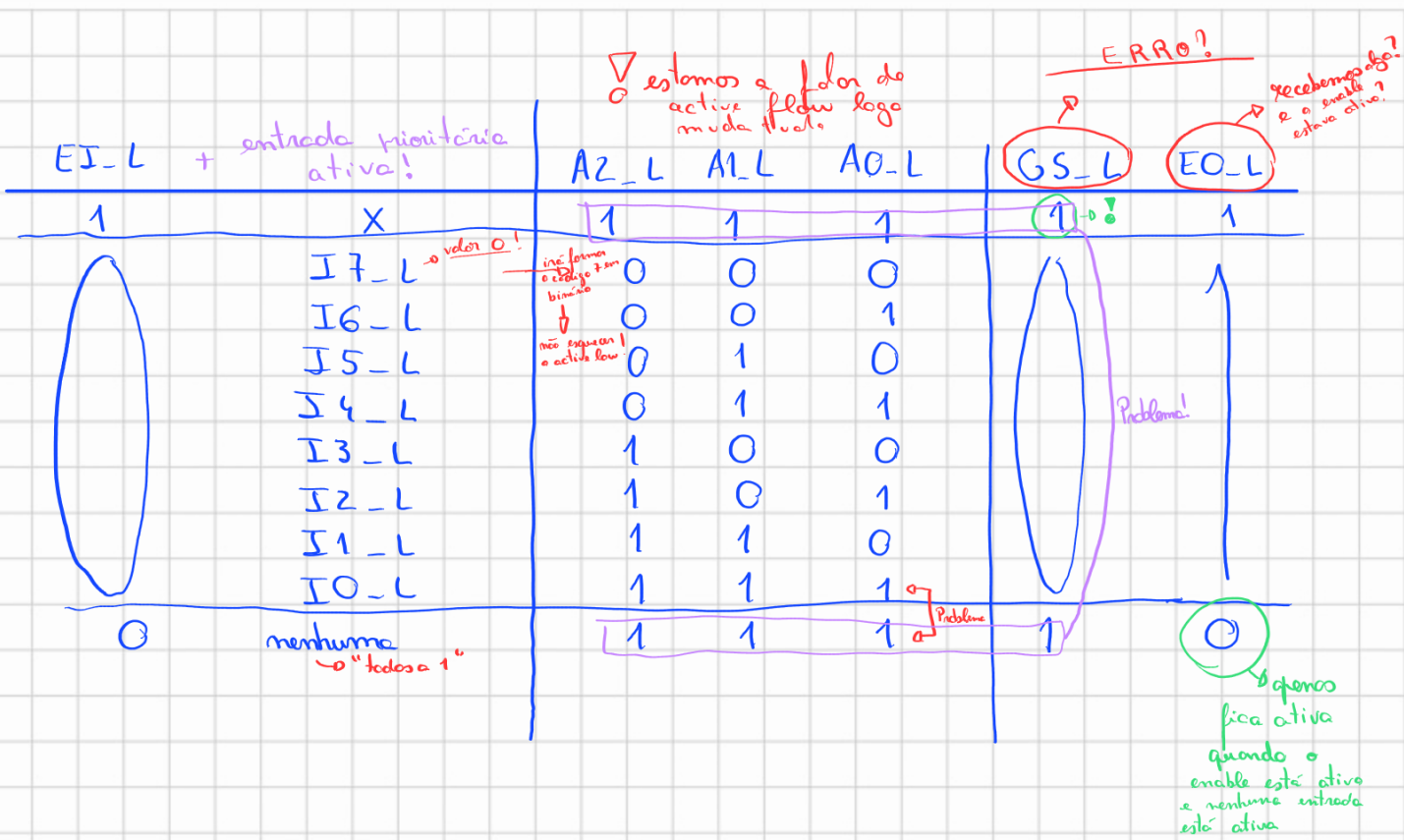
- What's the role of the IDLE output?

A commercial Model

- Active-low I/O
- Enable Input
- “Got Something Output”
- Enable Output

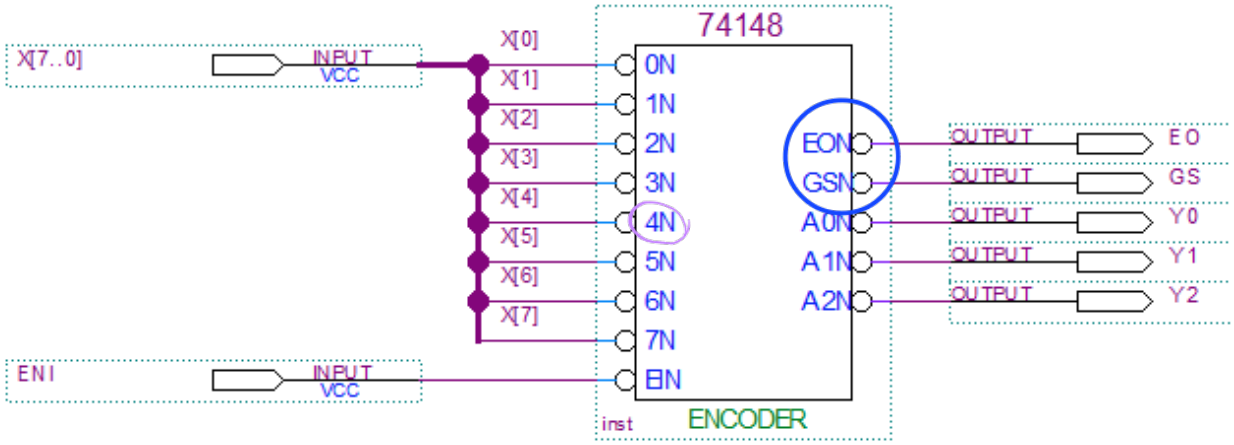


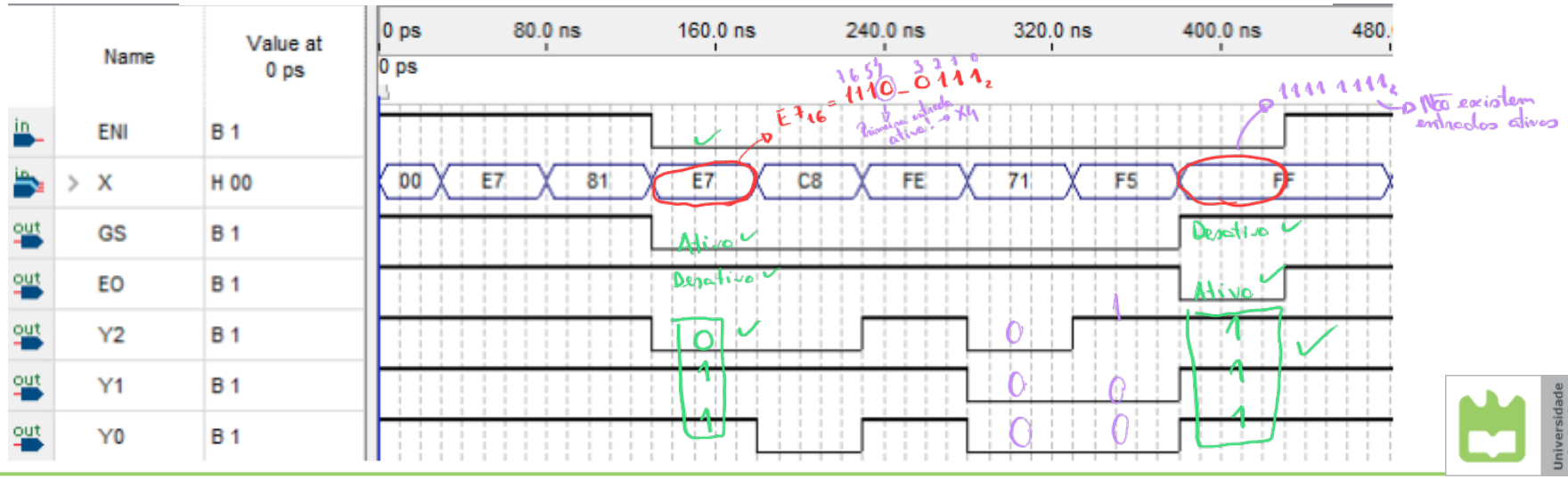
Inputs									Outputs				
EI_L	I0_L	I1_L	I2_L	I3_L	I4_L	I5_L	I6_L	I7_L	A2_L	A1_L	A0_L	GS_L	EO_L
1	x	x	x	x	x	x	x	x	1	1	1	1	1
0	x	x	x	x	x	x	x	0	0	0	0	0	1
0	x	x	x	x	x	x	0	1	0	0	1	0	1
0	x	x	x	x	x	0	1	1	0	1	0	0	1
0	x	x	x	x	0	1	1	1	0	1	1	0	1
0	x	x	x	0	1	1	1	1	1	0	0	0	1
0	x	x	0	1	1	1	1	1	1	0	1	0	1
0	x	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0


estamos a falar do active low logo muda tudo.
ERRO?
recebemos algo? e o enable estava ativo?
EI_L + entrada prioritária ativa!
A2_L A1_L A0_L
GS_L
EO_L
1
X
1 1 1 1 1
I7_L → valor 0!
I6_L
I5_L
I4_L
I3_L
I2_L
I1_L
I0_L
0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1
irá formar o código 7 em binário
não esquecer! o activo low
Problema!
Problema
0
nenhuma
→ "todos a 1"
1 1 1 1
0
fica ativa quando o enable está ativo e nenhuma entrada está ativa

Exercise

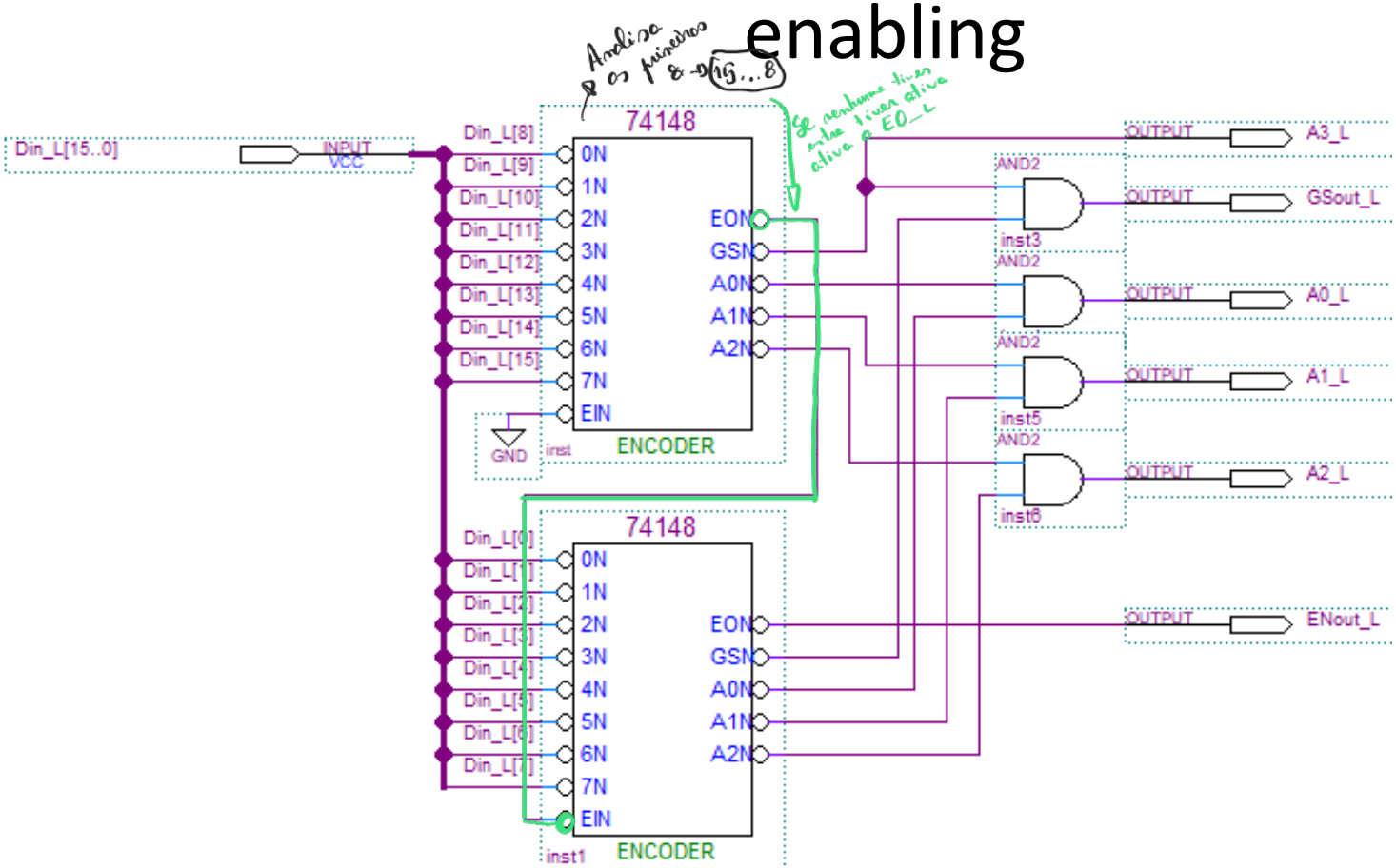
- Explain the timing diagram





Scaling-up: PE16to4

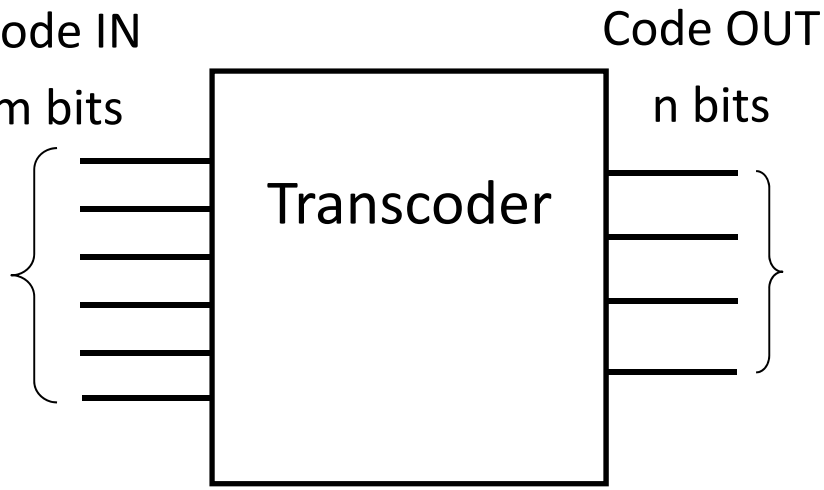
- Encoder cascading
- Note the priority enabling

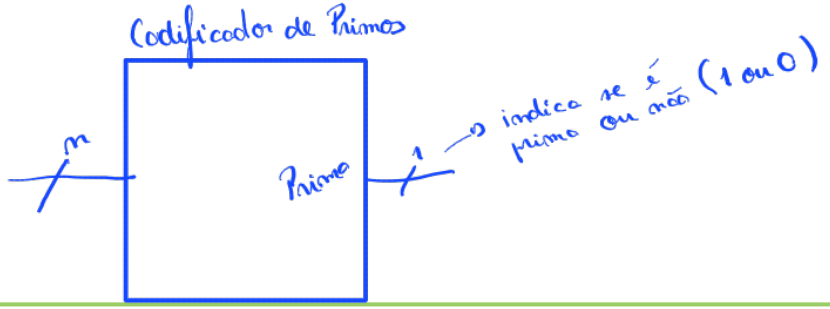


Generic Decoding

- "Transcoding"

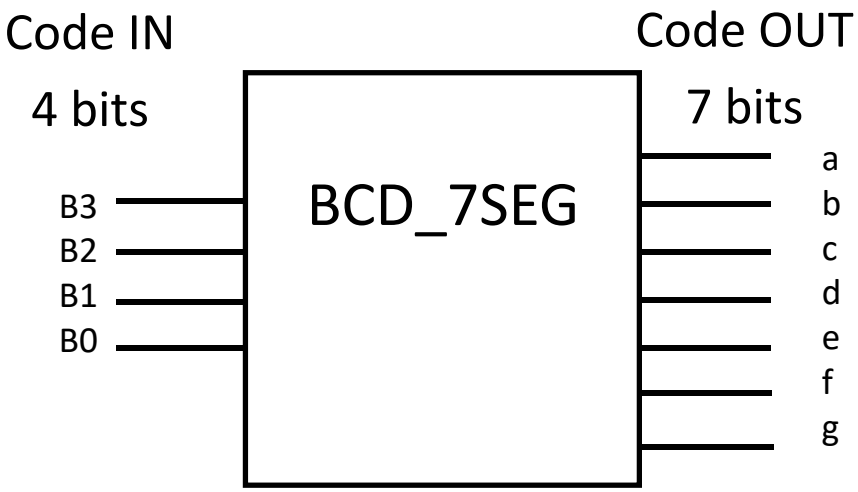
Code IN	Code OUT
BCD_{8421}	7-Segs
BCD_{8421}	Gray
Others ...	





Example

- BCD_{8421} to 7-Segment decoder



- Write the truth table
- Obtain a minimal SOP for the segment “a”

Final Remarks

- Always recall
  - The block symbol
  - The types of inputs and outputs
    - Data
    - Control
  - The truth table
  - The output equations
- Design with encapsulated logic requires mastering all the functional details of each block