

UNIOESTE
Ciência da Computação

Sistemas Digitais
Circuitos Combinacionais

Prof. Jorge Habib El Khouri
Prof. Antonio Marcos Hachisuca

2020/2021

Referências Bibliográficas

1. *Digital Fundamentals*, Thomas L. Floyd; Editora: Pearson; Edição: 11; Ano: 2015;
2. *Sistemas Digitais Princípios e Aplicações*, Ronald J. Tocci; Editora: Pearson; Edição: 11; Ano: 2011;
3. *Computer Organization and Design*, David A. Patterson; Editora: Elsevier; Edição: 1; Ano: 2017
4. *Digital Design: Principles and Practices*, John F. Wakerly; Editora: Pearson; Edição: 5; Ano: 2018;
5. *Guide to Assembly Language Programming in Linux*, Sivarama P. Dandamudi; Editora: Springer; Edição: 1; Ano: 2005.

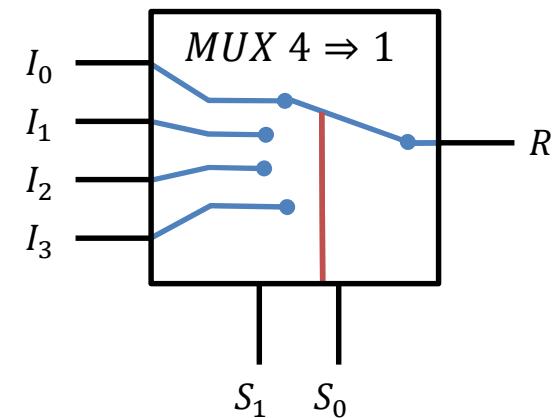
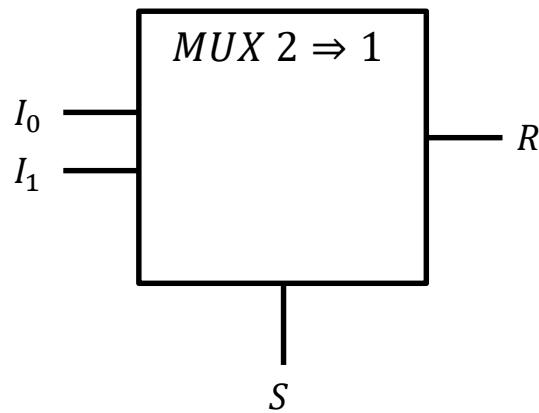
Sumário

- 1. Revisão – Sistemas de Numeração
- 2. Revisão – Representação de Dados
- 3. Revisão – Operações com Binários
- 4. Álgebra Booleana
- 5. Simplificação de Expressões
- 6. Mapa de Karnaugh
- 7. Elementos Lógicos Universais
- 8. Circuitos Combinacionais
- 9. Circuitos Sequenciais

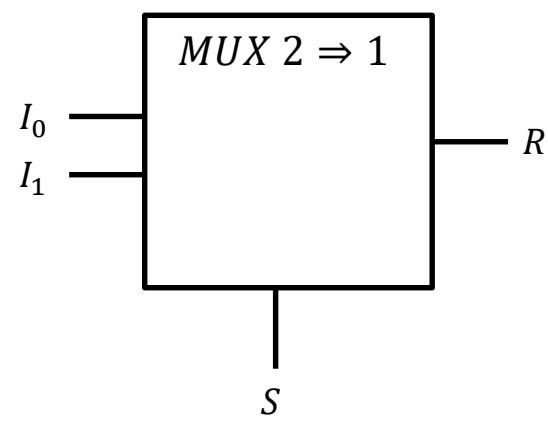
- 1. Somador / Subtrator
- 2. Comparadores
- 3. Codificador/decodificador
- 4. Multiplexador/Demux
- 5. Geradores de paridade
- 6. ULA
- 7. Multiplicadores / Divisores
- 8. PLD/PLA/ROM

Multiplexador

- Multiplexador é um circuito que seleciona, a partir de uma entrada específica, um dos sinais de entrada a ser transferido para a saída;
- Também referenciado como Multiplex, Multiplexer ou MUX.

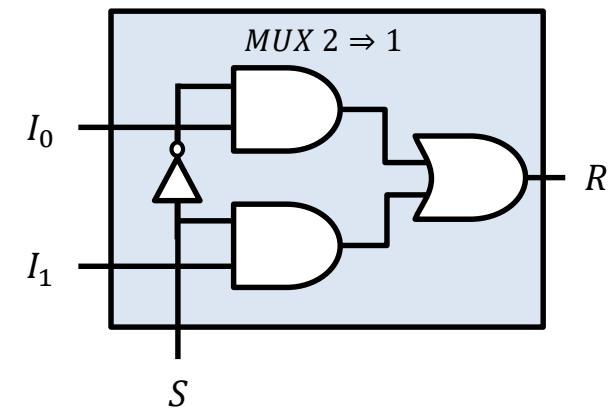


Multiplexador $MUX\ 2 \Rightarrow 1$

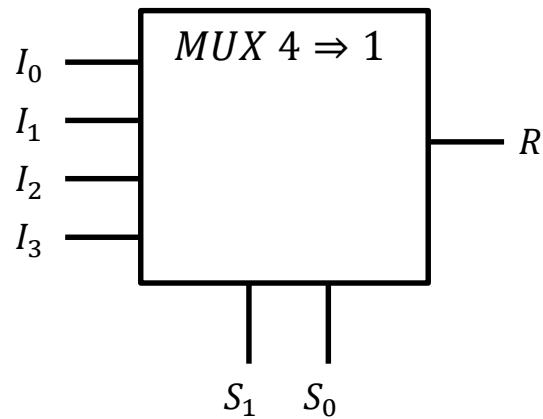


S	R
0	I_0
1	I_1

$$R = \bar{S}I_0 + SI_1$$

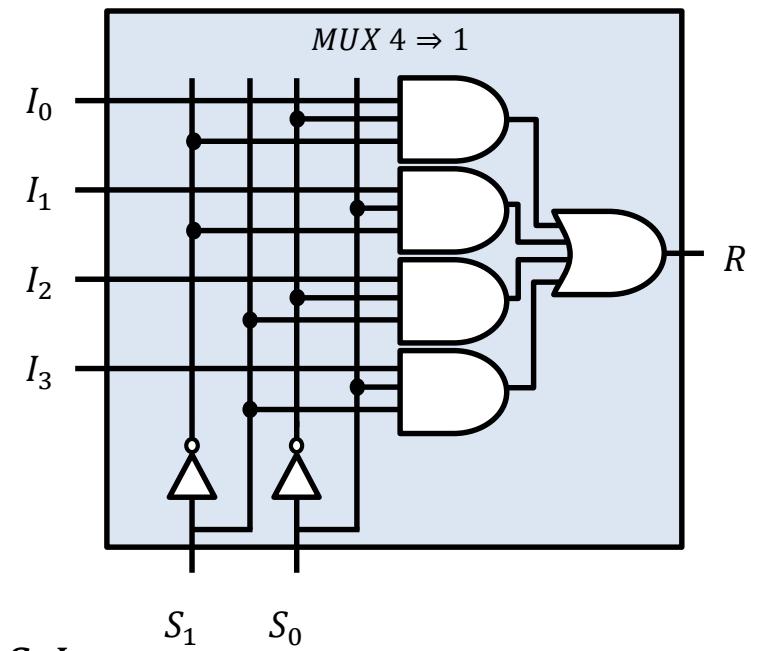


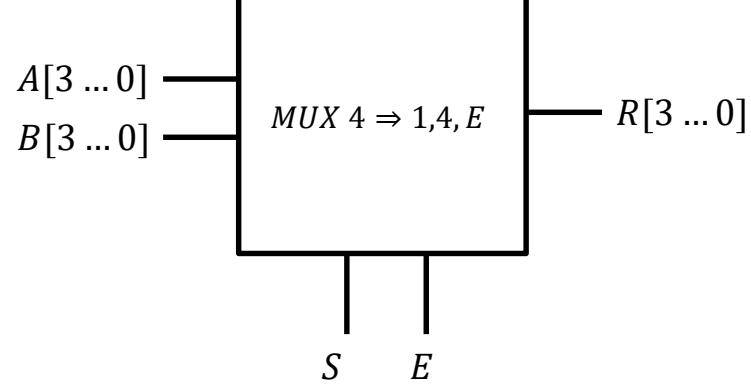
Multiplexador $MUX\ 4 \Rightarrow 1$



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

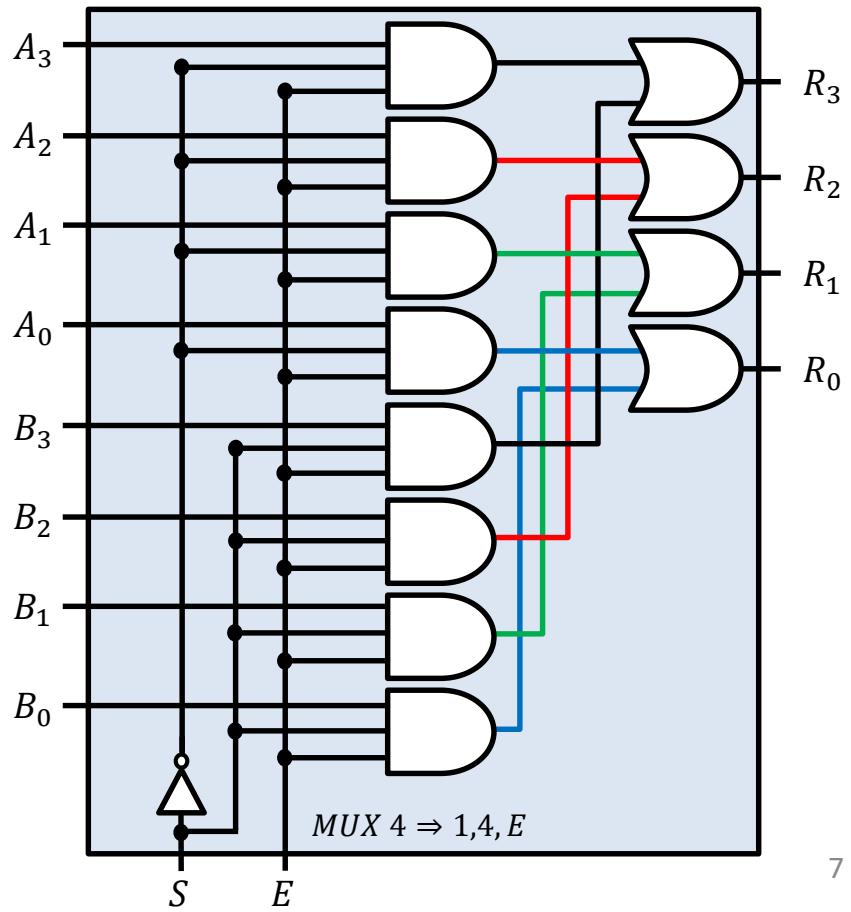
$$R = \bar{S}_1 \bar{S}_0 I_0 + \bar{S}_1 S_0 I_1 + S_1 \bar{S}_0 I_2 + S_1 S_0 I_3$$



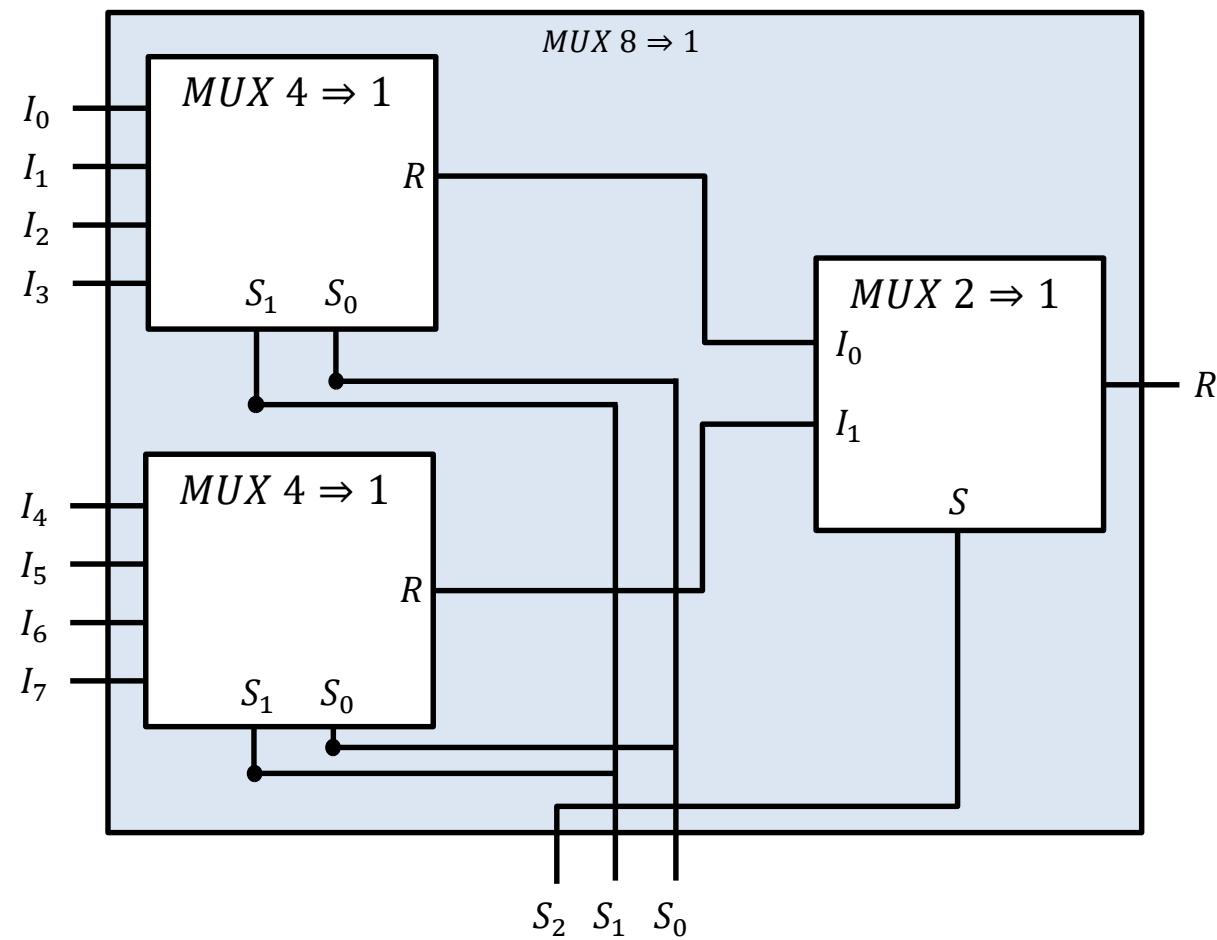


$$R = E\bar{S}A + ESB$$

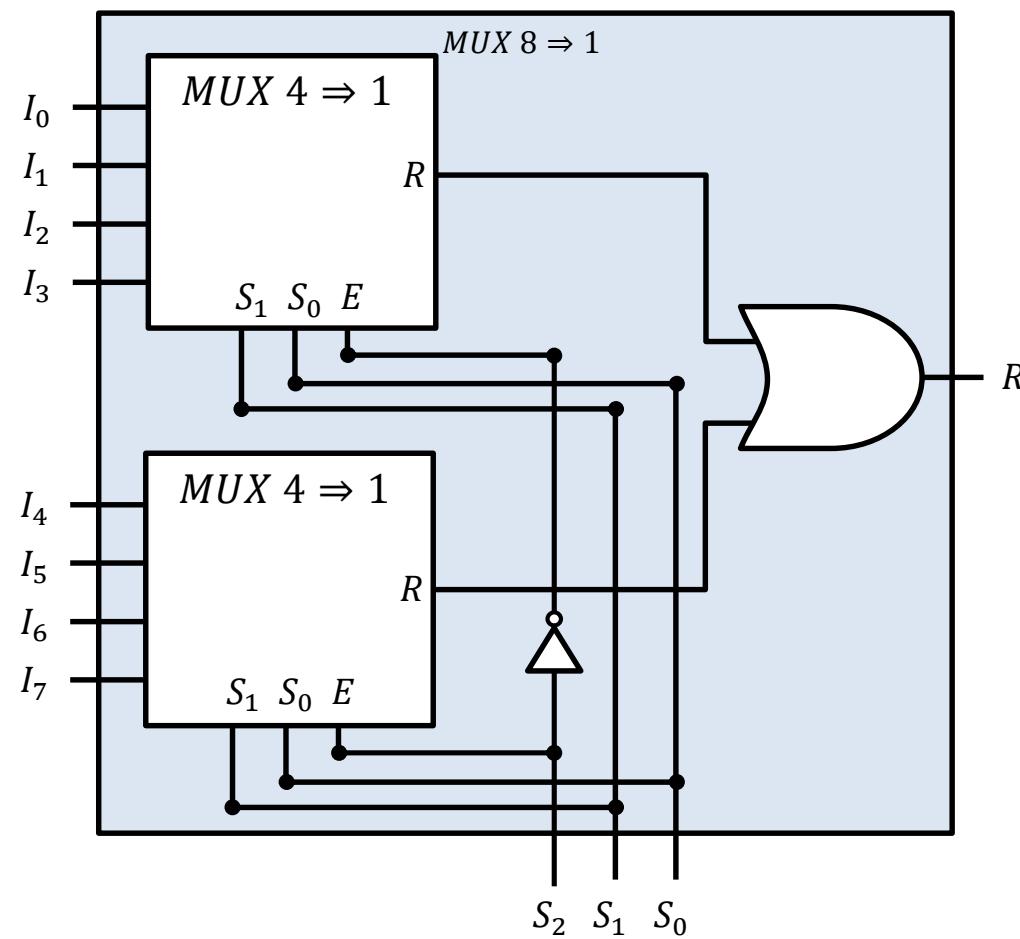
Multiplexador
 $MUX\ 2 \Rightarrow 1 - 4\ bits\ e\ Enable$



Multiplexador
 $MUX\ 8 \Rightarrow 1$



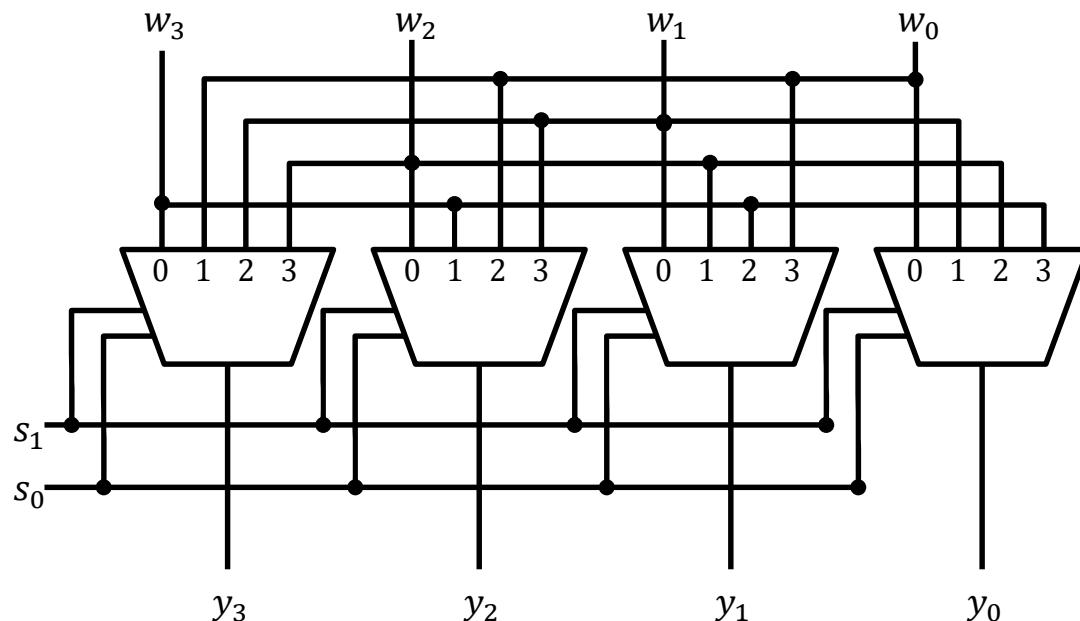
Multiplexador
 $MUX\ 8 \Rightarrow 1$



EXERCÍCIOS

Multiplexador Exercício

- O circuito abaixo é composto por 4 multiplexadores. As entradas são: uma palavra de dados $w[3 \dots 0]$ e dois bits de controle ($s_1 - s_0$). A saída é o vetor $y[3 \dots 0]$. Preencha a tabela verdade correspondente a este circuito, e interprete o que este circuito realiza. No preenchimento da tabela verdade utilizar os valores w_3, w_2, w_1 ou w_0 .

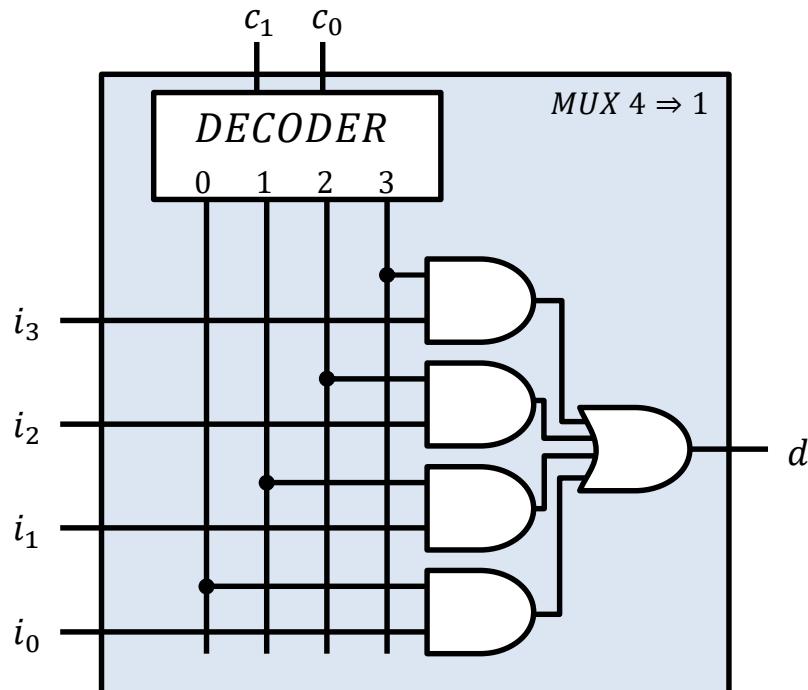


s_1	s_0	y_3	y_2	y_1	y_0
0	0	w_3	w_2	w_1	w_0
0	1	w_0	w_3	w_2	w_1
1	0	w_1	w_0	w_3	w_2
1	1	w_2	w_1	w_0	w_3

$$y = w <>> s$$

Multiplexador *Exercício*

2. Determine a saída d em função de c_1 e c_0 .



c_1	c_0	d
0	0	i_0
0	1	i_1
1	0	i_2
1	1	i_3

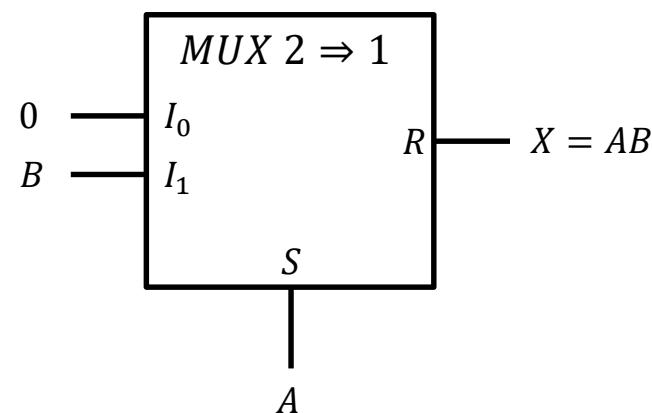
Multiplexador *Exercício*

3. Implementar as seguintes funções lógicas:

a) $Y = AB$

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

A	R
0	0
1	B



Multiplexador

Exercício

- Implementar as seguintes funções lógicas:

$$b) Y = A \oplus B$$

$$c) Y = A\bar{B} + \bar{B}\bar{C} + \bar{A}\bar{B}C$$

$$d) Y = \bar{C}D + ABC$$

- Implementar um circuito que entrega o maior entre três números inteiros de 4 bits:

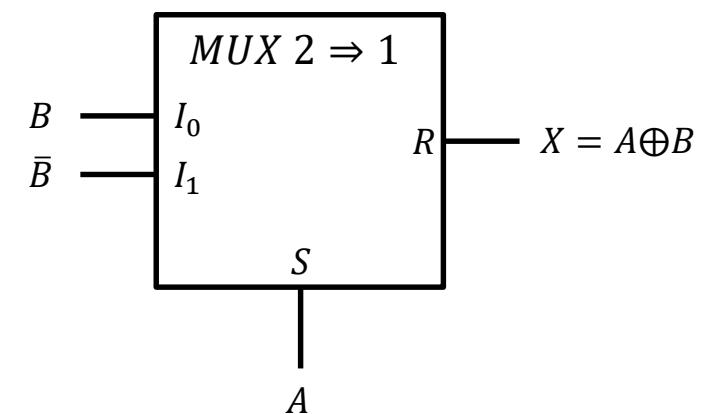
$$X = maior(A, B, C)$$

Multiplexador *Exercício*

b) $Y = A \oplus B$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

A	R
0	B
1	\bar{B}

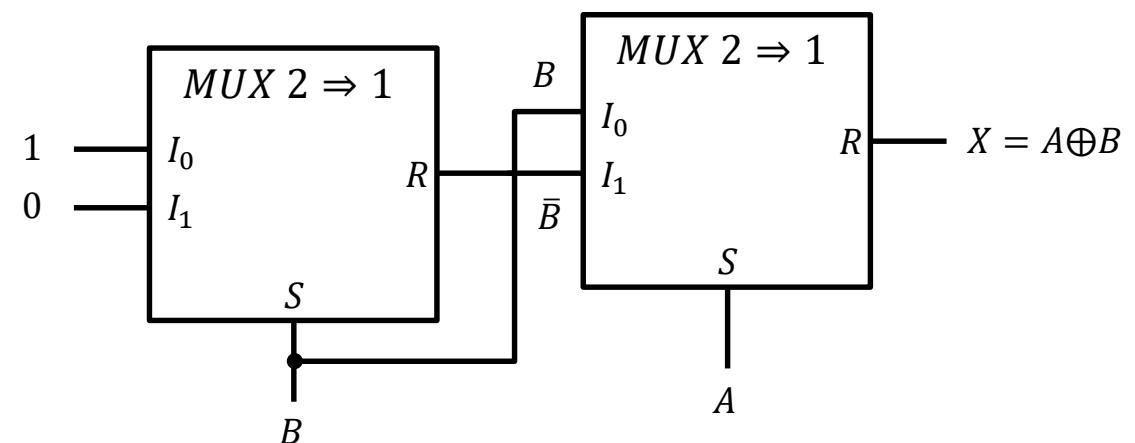


Multiplexador Exercício

$$b) Y = A \oplus B$$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

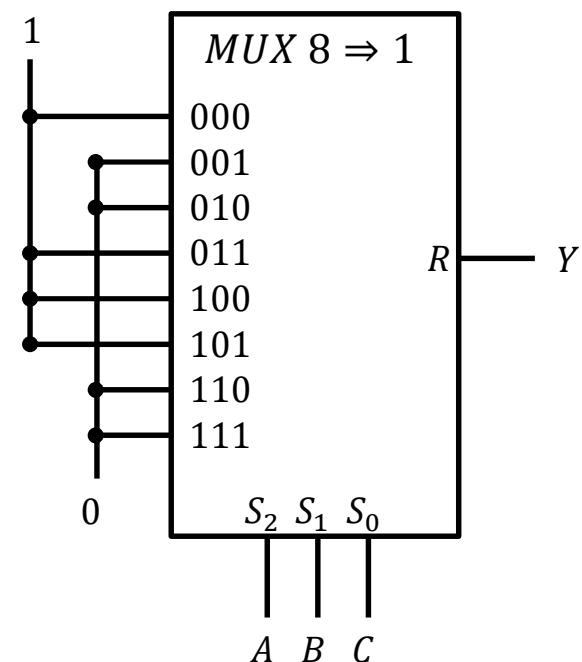
A	R
0	B
1	\bar{B}



Multiplexador Exercício

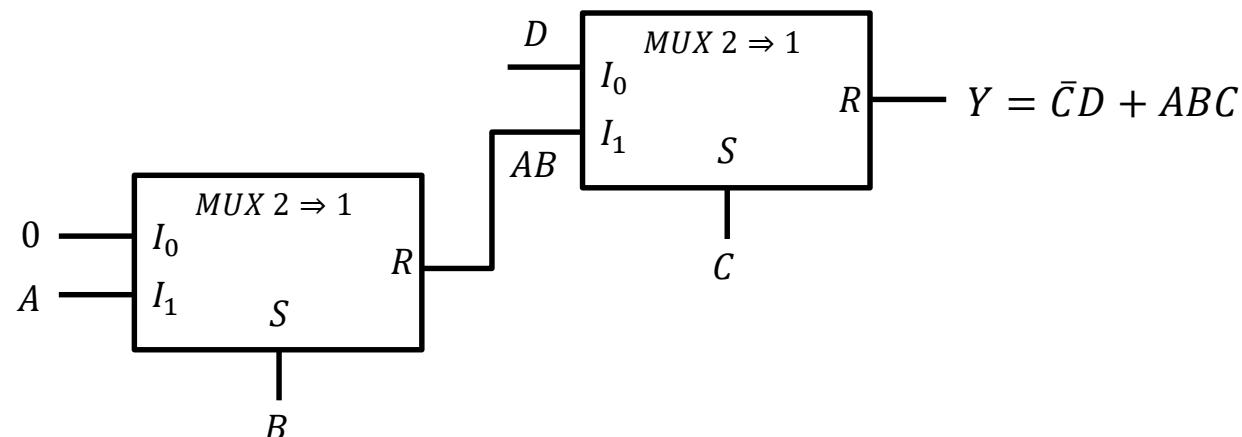
$$c) \ Y = A\bar{B} + \bar{B}\bar{C} + \bar{A}BC$$

	A	B	C	Y
$\bar{B}\bar{C}$	0	0	0	1
	0	0	1	0
	0	1	0	0
$\bar{A}BC$	0	1	1	1
$\bar{B}\bar{C}$ $A\bar{B}$	1	0	0	1
$A\bar{B}$	1	0	1	1
	1	1	0	0
	1	1	1	0



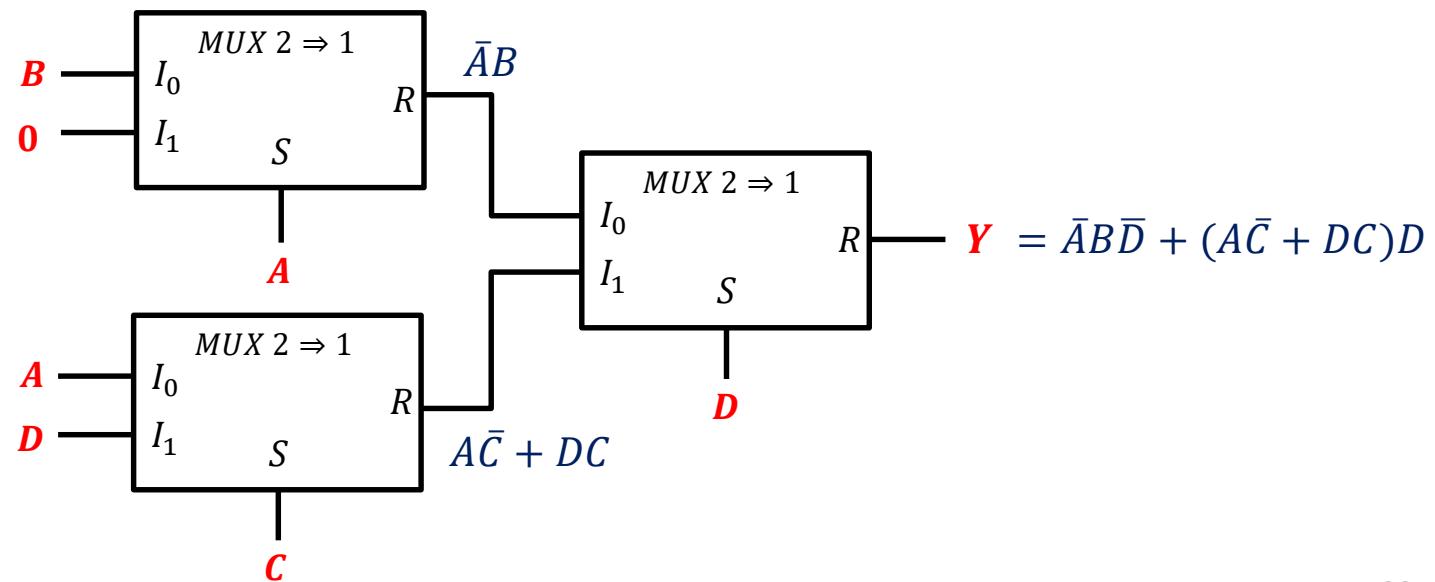
Multiplexador *Exercício*

d) $Y = \bar{C}D + ABC$



Multiplexador Exercício

e) Qual a função Y gerada abaixo?



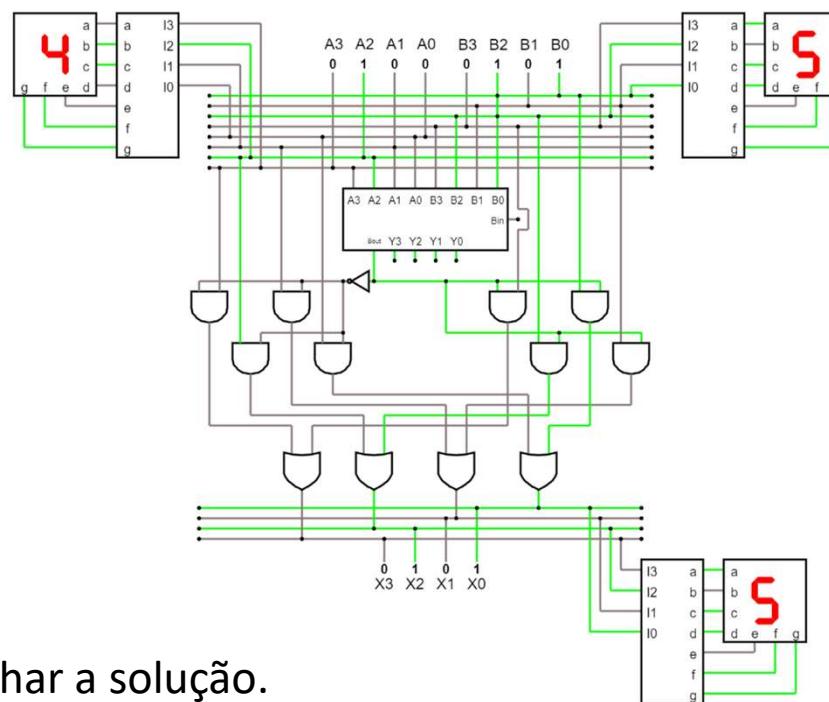
Multiplexador *Exercício*

Implementar um circuito que entrega o maior entre três números inteiros de 4 bits:

$$X = \text{maior}(A, B, C)$$

Multiplexador *Exercício*

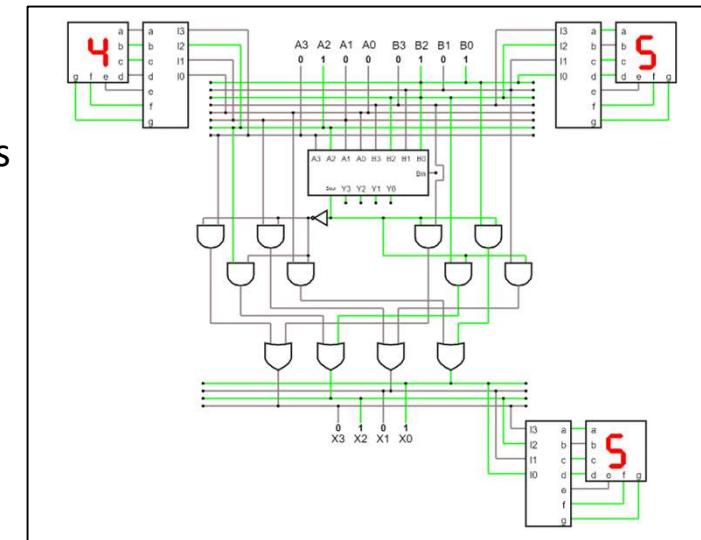
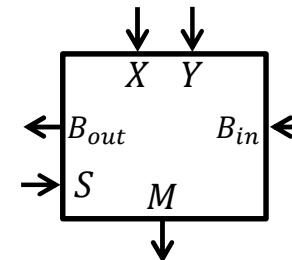
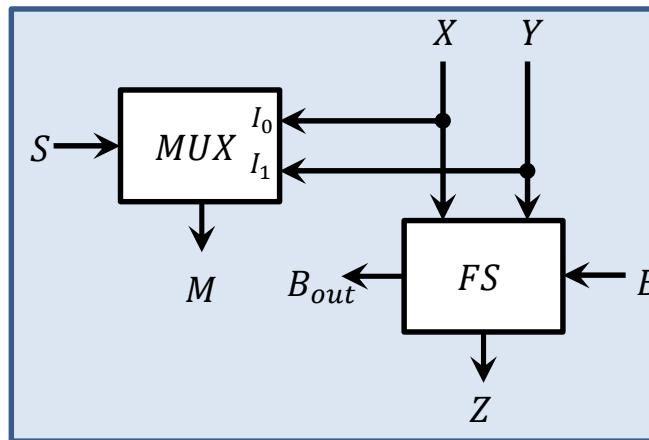
- ✓ A primeira versão da lógica do maior entre dois inteiros teve a seguinte estrutura:



- ✓ Vamos redesenhar a solução.

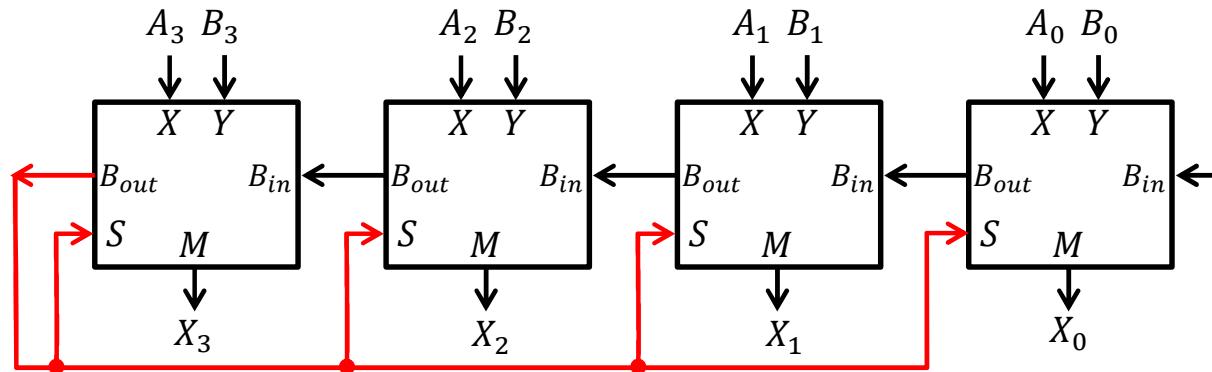
Multiplexador Exercício

- ✓ A nova proposta inclui a criação de um bloco funcional que combina um Subtrator Completo de 1 bit e um *MUX* 2:1;
 - O resultado do Subtrator será desprezado;
 - A saída será comandada pelo *MUX*;
 - A entrada seletora do *MUX* (*S*) escolherá entre as entradas *X* e *Y* do Subtrator;



Multiplexador Exercício

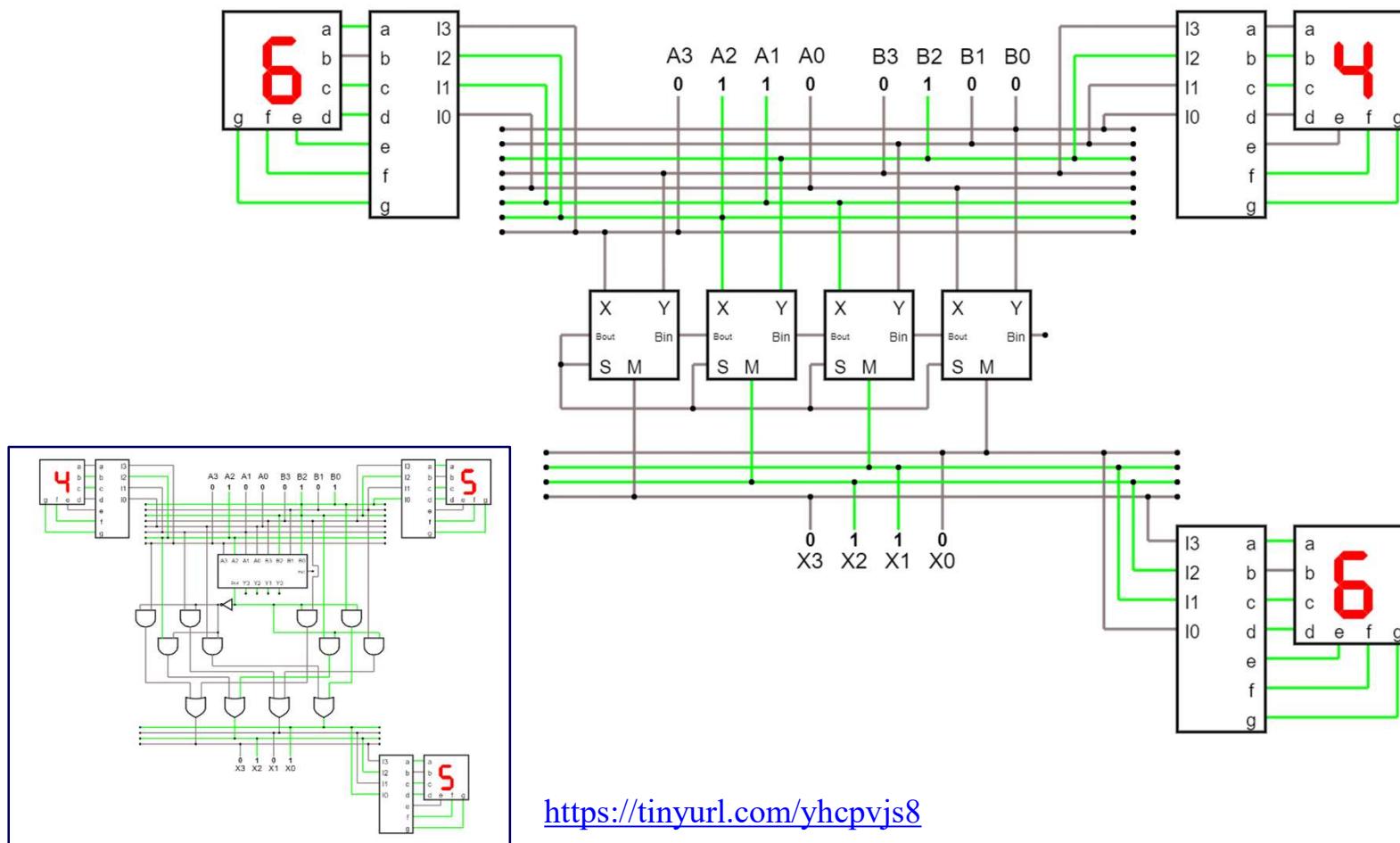
- ✓ A nova versão da lógica do maior entre dois inteiros tem a seguinte estrutura:



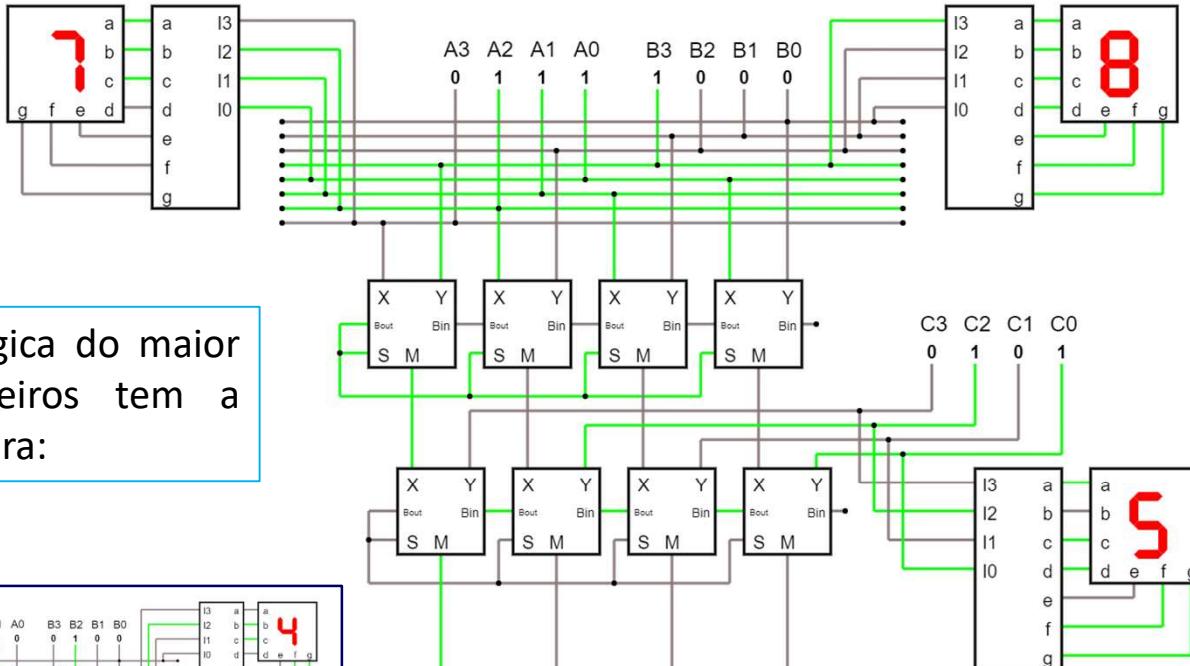
- ✓ O valor do último B_{out} comandará as saídas de cada Bloco.
 - Se o último $B_{out} == 0$ então entregará X , senão Y .

Multiplexador

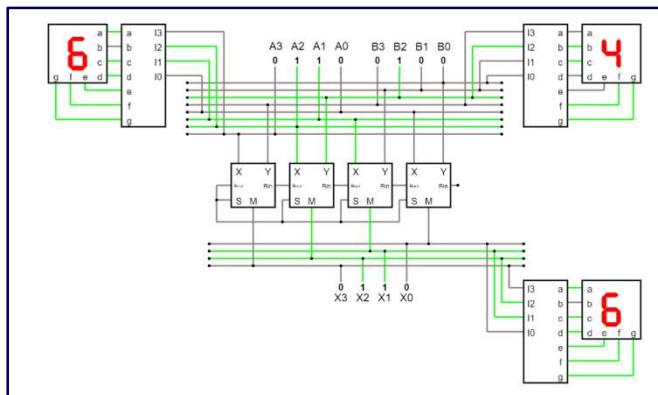
Exercício



Multiplexador Exercício



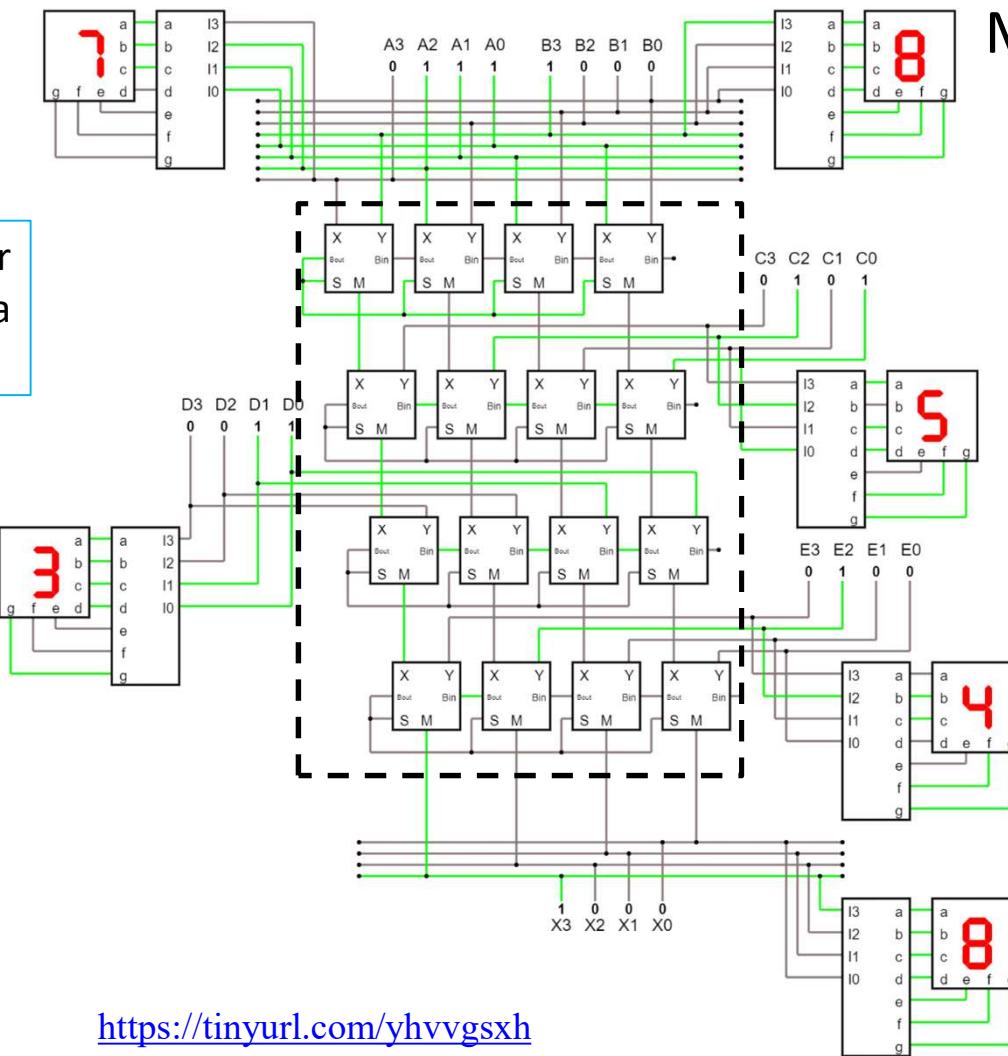
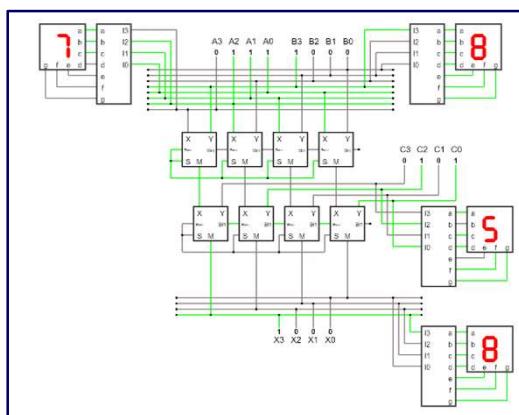
A versão da lógica do maior entre três inteiros tem a seguinte estrutura:



<https://tinyurl.com/yfbxddl3>

Multiplexador Exercício

A versão da lógica do maior entre cinco inteiros tem a seguinte estrutura:



<https://tinyurl.com/yhvvgсх>

Sumário

- 1. Revisão – Sistemas de Numeração
- 2. Revisão – Representação de Dados
- 3. Revisão – Operações com Binários
- 4. Álgebra Booleana
- 5. Simplificação de Expressões
- 6. Mapa de Karnaugh
- 7. Elementos Lógicos Universais
- 8. Circuitos Combinacionais
- 9. Circuitos Sequenciais

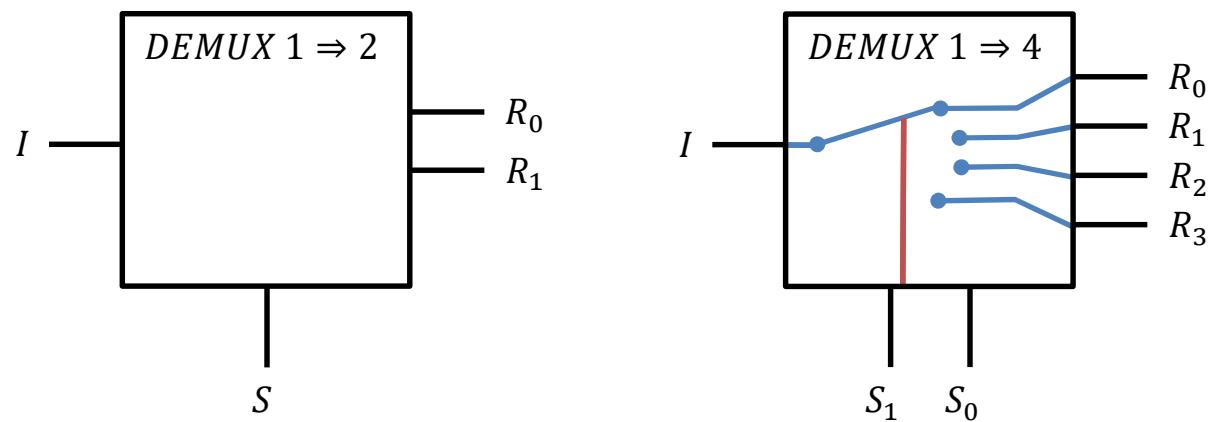
- 1. Somador / Subtrator
- 2. Comparadores
- 3. Codificador/decodificador
- 4. Multiplexador/Demux
- 5. Geradores de paridade
- 6. PLAs
- 7. Memórias ROM
- 8. ULA
- 9. Multiplicadores / Divisores

Demultiplexador

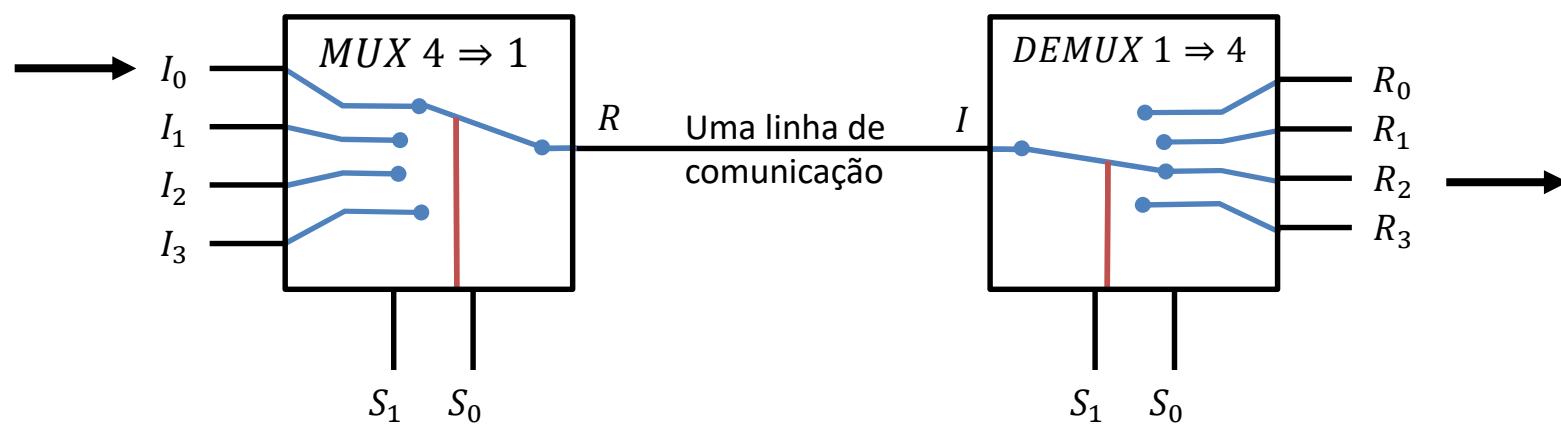
- Demultiplexador é um circuito que permite selecionar uma das entradas e transferi-la para a saída. É um distribuidor de dados;
- Esta característica de selecionar ou habilitar um sinal entre muitos, permite diversas aplicações em microprocessadores ou sistemas de controle, tais como:
 - Seleção de diferentes dispositivos de *I/O* para transferência de dados;
 - Escolha entre diferentes bancos de memória;
 - Habilitar diferentes linhas de chips de memória, de acordo com o endereço informado;
 - Habilitar diferentes unidades funcionais;
 - Sistemas de transmissão de dados síncronos;
 - Implementar expressões booleanas.

Demultiplexador

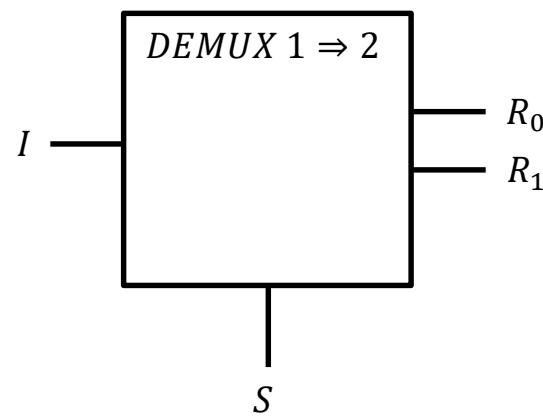
- Decoders podem também serem utilizados como demultiplexadores.
- Também referenciado como Demultiplex, Demultiplexer ou DEMUX.



Demultiplexador

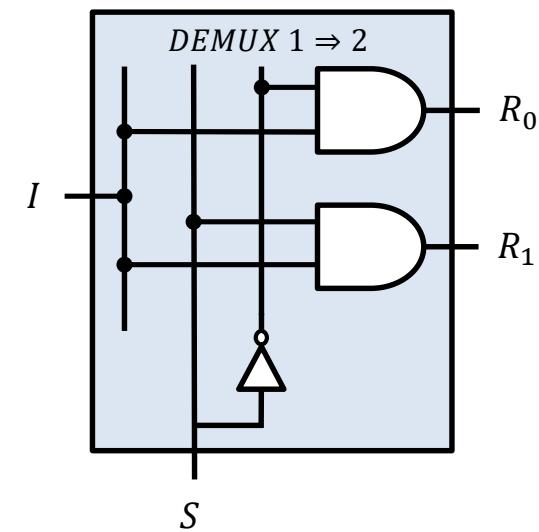


Demultiplexador $DEMUX\ 1 \Rightarrow 2$

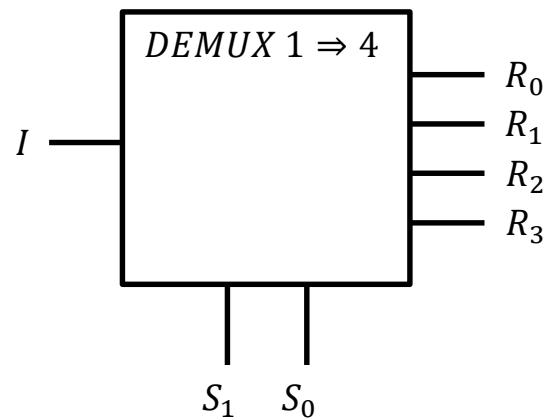


S	R_1	R_0
0	0	I
1	I	0

$$R_0 = \bar{S}I$$
$$R_1 = SI$$

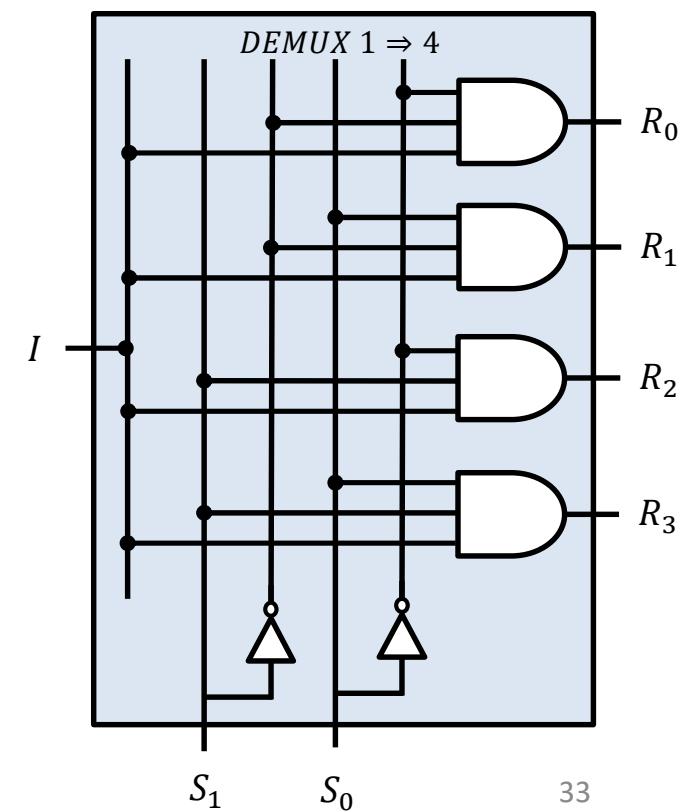


Demultiplexador *DEMUX 1 ⇒ 4*

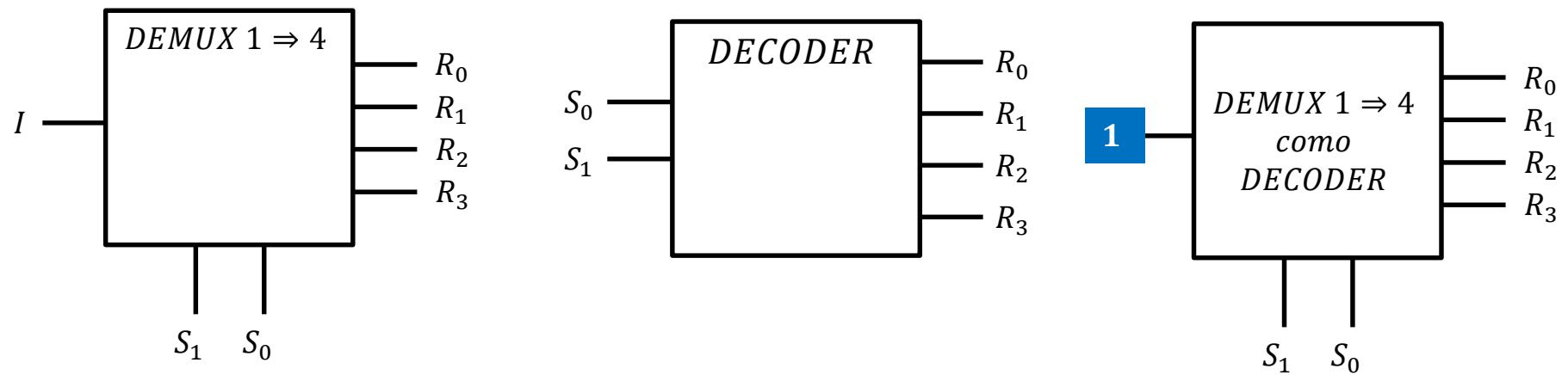


<i>S₁</i>	<i>S₀</i>	<i>R₃</i>	<i>R₂</i>	<i>R₁</i>	<i>R₀</i>
0	0	0	0	0	<i>I</i>
0	1	0	0	<i>I</i>	0
1	0	0	<i>I</i>	0	0
1	1	<i>I</i>	0	0	0

$$\begin{aligned}
 R_0 &= \bar{S}_1 \bar{S}_0 I \\
 R_1 &= \bar{S}_1 S_0 I \\
 R_2 &= S_1 \bar{S}_0 I \\
 R_3 &= S_1 S_0 I
 \end{aligned}$$



Demultiplexador *DEMUX 1 ⇒ 4 como DECODER*



$$R_0 = \bar{S}_1 \bar{S}_0 I$$

$$R_1 = \bar{S}_1 S_0 I$$

$$R_2 = S_1 \bar{S}_0 I$$

$$R_3 = S_1 S_0 I$$

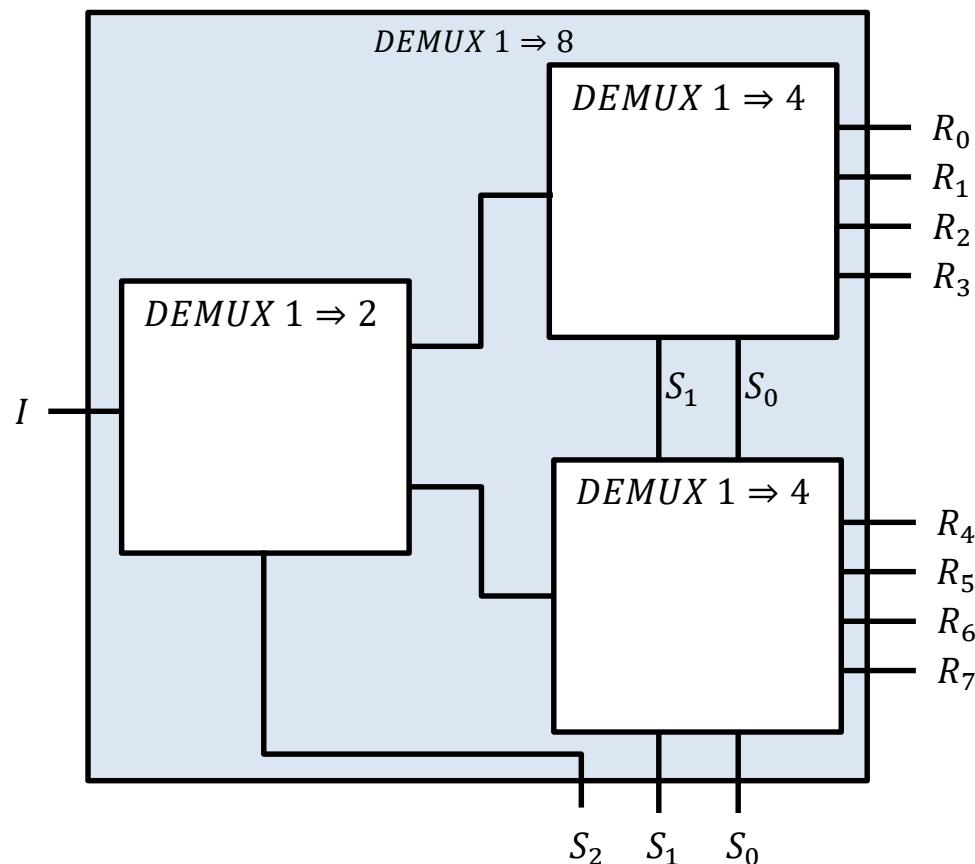
$$R_0 = \bar{S}_1 \bar{S}_0$$

$$R_1 = \bar{S}_1 S_0$$

$$R_2 = S_1 \bar{S}_0$$

$$R_3 = S_1 S_0$$

Demultiplexador
 $DEMUX\ 1 \Rightarrow 8$



Demultiplexador *Exercício*

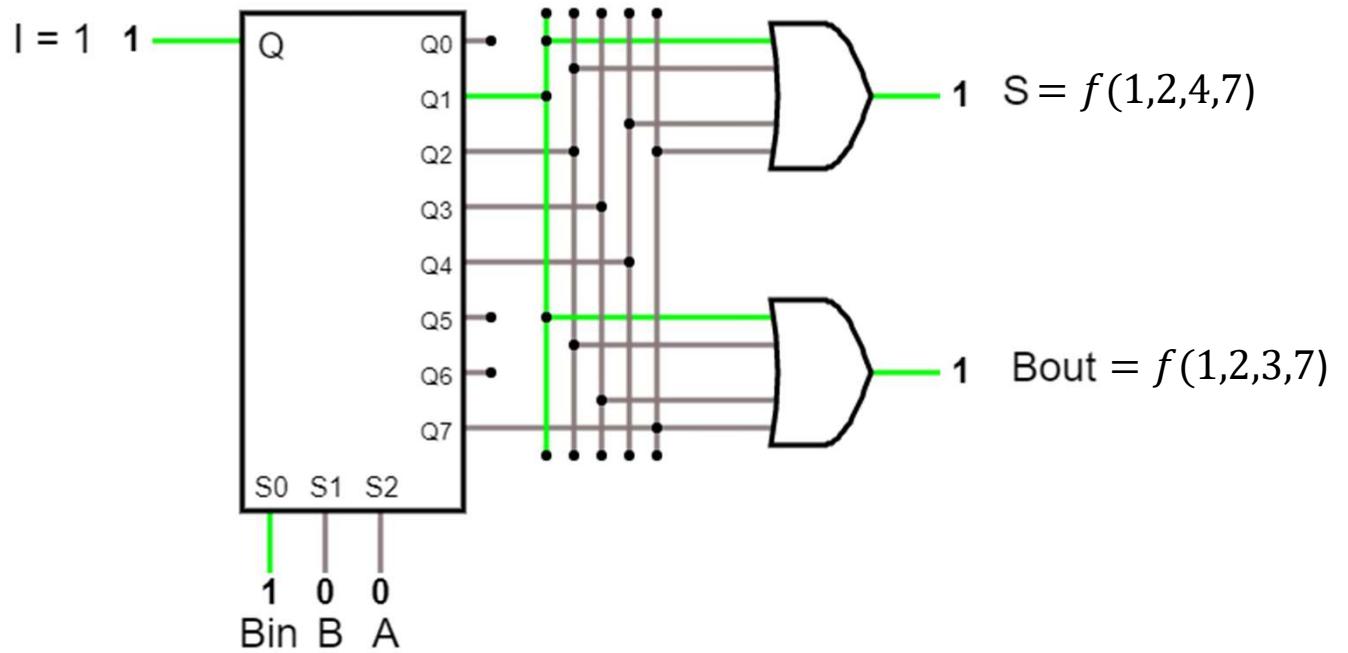
Implementar um Full Subtrator usando *DEMUX 1:8*:

B_{in}	A	B	B_{out}	S
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	0	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1

Demultiplexador Exercício

B_{in}	A	B	B_{out}	S
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	0	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1

A	B	B_{in}	B_{out}	S
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



<https://tinyurl.com/yz6mxa9b>

<https://www.electronicshub.org/demultiplexerdemux/>

Sumário

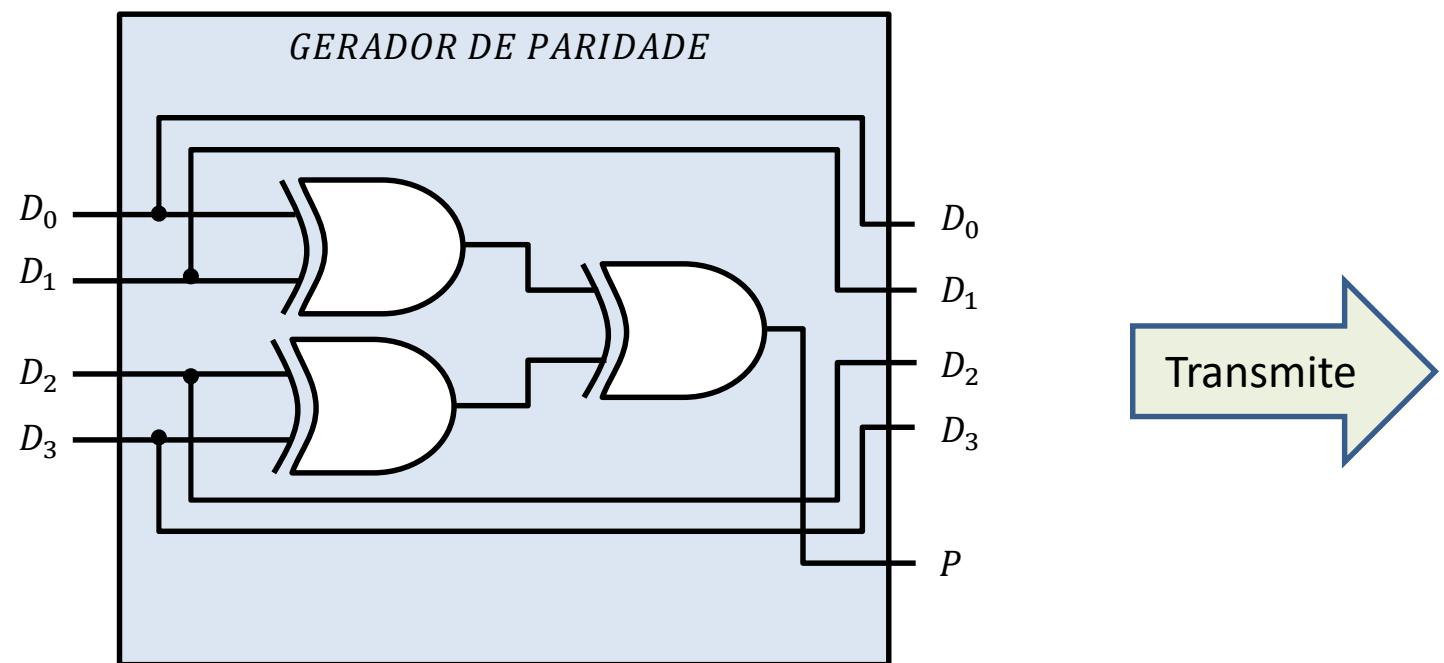
- 1. Revisão – Sistemas de Numeração
- 2. Revisão – Representação de Dados
- 3. Revisão – Operações com Binários
- 4. Álgebra Booleana
- 5. Simplificação de Expressões
- 6. Mapa de Karnaugh
- 7. Elementos Lógicos Universais
- 8. Circuitos Combinacionais
- 9. Circuitos Sequenciais

- 1. Somador / Subtrator
- 2. Comparadores
- 3. Codificador/decodificador
- 4. Multiplexador/Demux
- 5. Geradores de paridade
- 6. Circuitos Específicos
- 7. Multiplicadores / Divisores
- 8. ULA
- 9. PLD/PLA/ROM

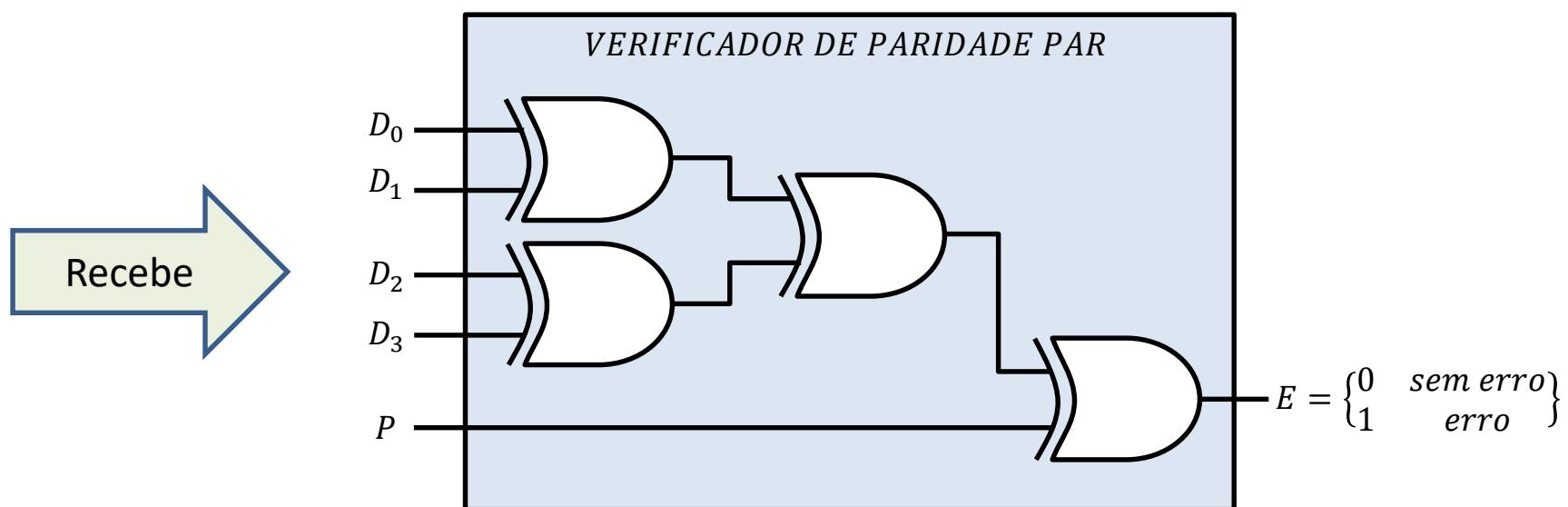
Gerador de Paridade

- Normalmente utilizado na comunicação de dados;
- Gerador de paridade - cria o *parity bit*, como *bit* extra, adicionado a uma determinada palavra;
- Verificador de paridade – calcula e compara a paridade de uma palavra a fim de garantir a correção do dado recebido.
- Paridade par: o valor do *bit* de paridade é determinado para que o número total de 1s na palavra seja par;
- Paridade ímpar: o valor do *bit* de paridade é determinado para que o número total de 1s na palavra seja ímpar;

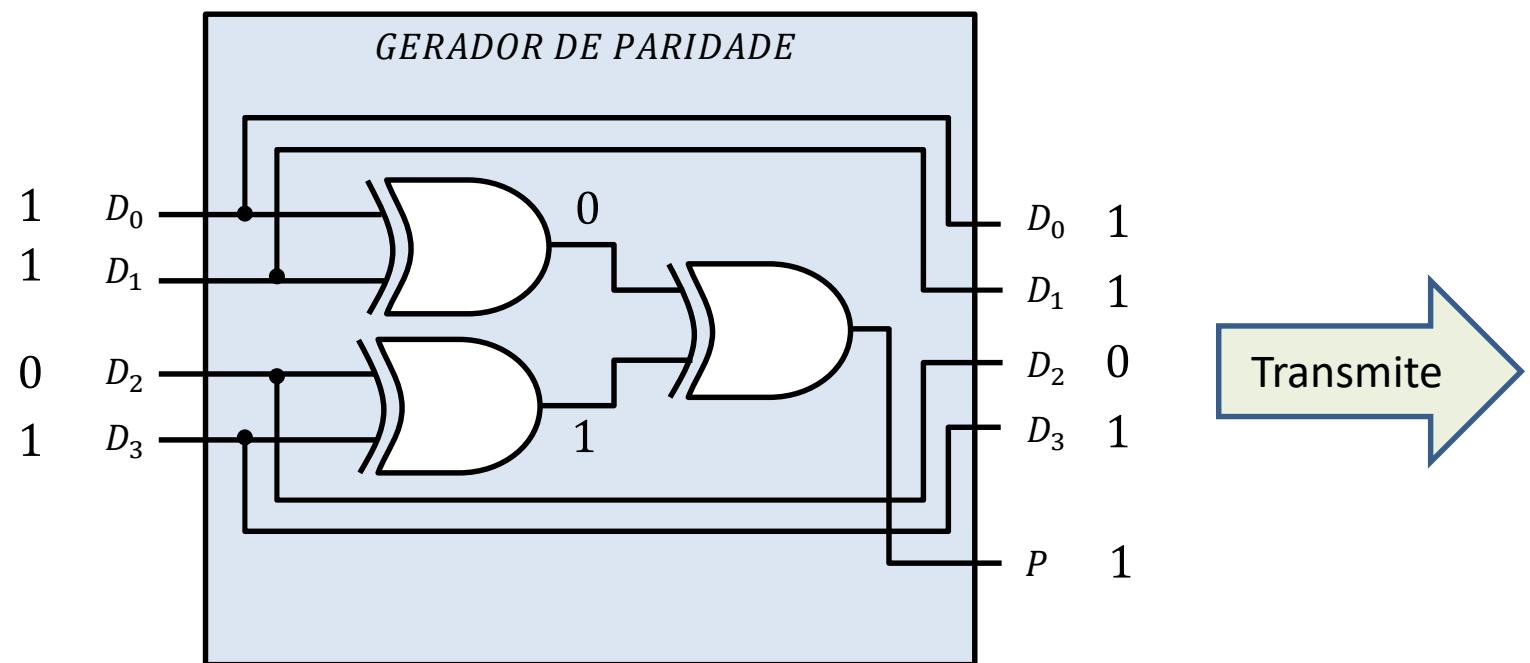
Gerador de Paridade Par



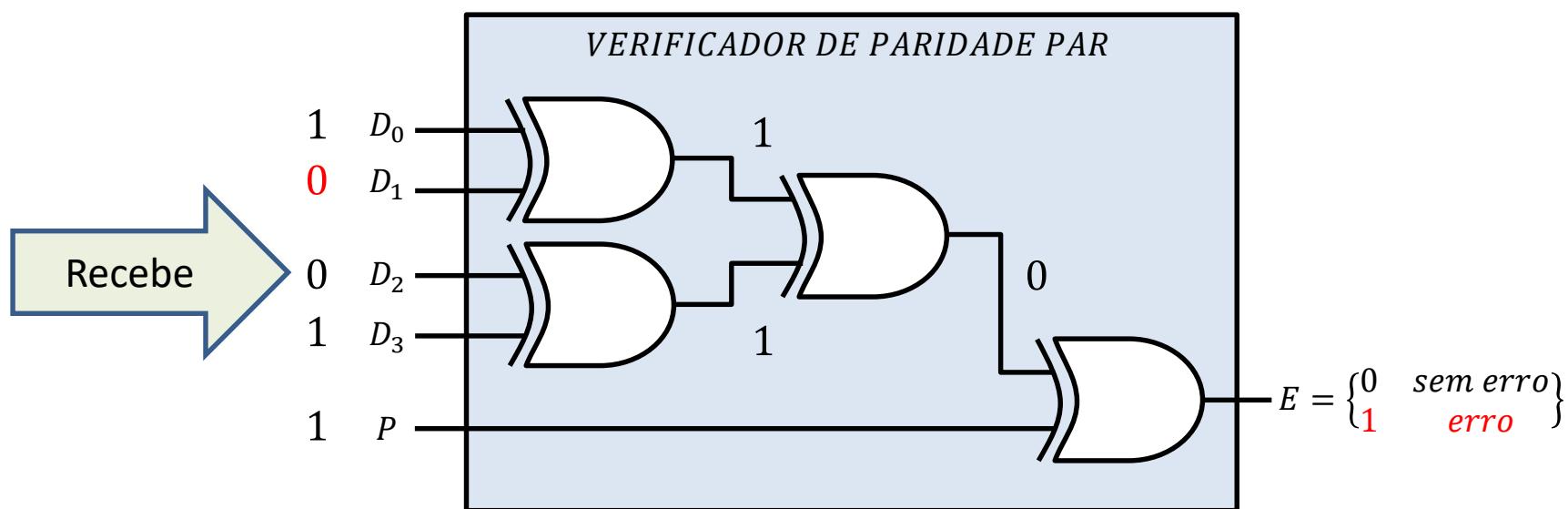
Verificador de Paridade Par



Gerador de Paridade Par Exemplo



Verificador de Paridade Par Exemplo

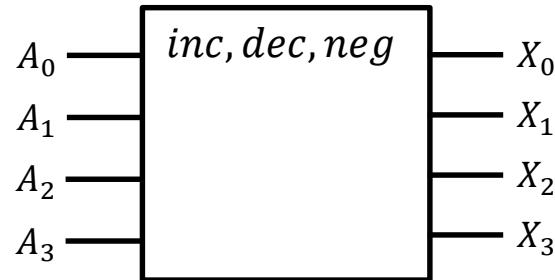


Sumário

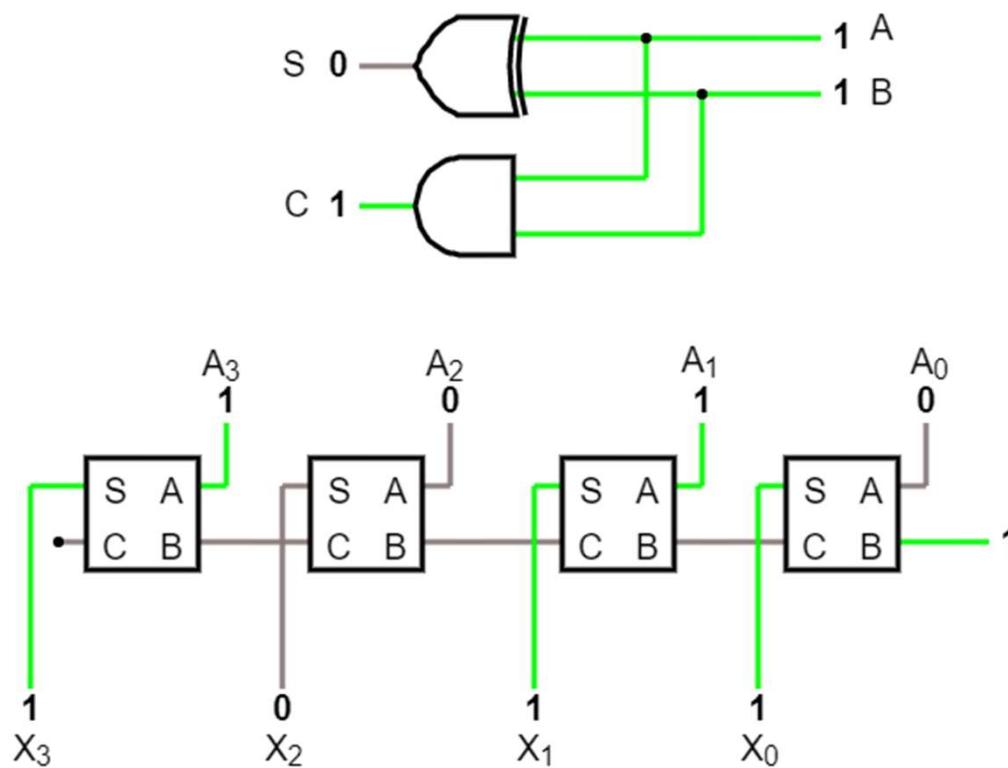
- 1. Revisão – Sistemas de Numeração
- 2. Revisão – Representação de Dados
- 3. Revisão – Operações com Binários
- 4. Álgebra Booleana
- 5. Simplificação de Expressões
- 6. Mapa de Karnaugh
- 7. Elementos Lógicos Universais
- 8. Circuitos Combinacionais
- 9. Circuitos Sequenciais

- 1. Somador / Subtrator
- 2. Comparadores
- 3. Codificador/decodificador
- 4. Multiplexador/Demux
- 5. Geradores de paridade
- 6. Circuitos Específicos
- 7. Multiplicadores / Divisores
- 8. ULA
- 9. PLD/PLA/ROM

inc, dec e neg exclusivos



Circuitos Específicos *inc*



Circuitos Específicos

inc

A_3	A_2	A_1	A_0	X_3	X_2	X_1	X_0
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0

	$\bar{A}_1\bar{A}_0$	\bar{A}_1A_0	A_1A_0	$A_1\bar{A}_0$
A_3A_2	00	01	11	10
$\bar{A}_3\bar{A}_2$	00	0	1	3
\bar{A}_3A_2	01	4	5	6
A_3A_2	11	1 12	1 13	1 14
$A_3\bar{A}_2$	10	1 8	1 9	1 11 1 10

$$A_3\bar{A}_1 + A_3\bar{A}_2A_1A_0 + A_3\bar{A}_0 + \bar{A}_3A_2A_1A_0$$

$$X_3 = A_3\bar{A}_1 + A_3\bar{A}_2A_1A_0 + A_3\bar{A}_0 + \bar{A}_3A_2A_1A_0$$

$$X_3 = A_3(\bar{A}_0 + \bar{A}_1) + A_1A_0(A_3 \oplus A_2)$$

Circuitos Específicos

inc

A_3	A_2	A_1	A_0	X_3	X_2	X_1	X_0
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0

	$\bar{A}_1\bar{A}_0$	\bar{A}_1A_0	A_1A_0	$A_1\bar{A}_0$
A_3A_2	00	01	11	10
$\bar{A}_3\bar{A}_2$	00	0	1	2
\bar{A}_3A_2	01	1	1	1
A_3A_2	11	1	1	1
$A_3\bar{A}_2$	10	8	9	10

$$A_2\bar{A}_1 + A_0A_1\bar{A}_2 + \bar{A}_0A_2$$

$$X_2 = A_2\bar{A}_1 + A_0A_1\bar{A}_2 + \bar{A}_0A_2$$

$$X_2 = A_2(\bar{A}_0 + \bar{A}_1) + A_0A_1\bar{A}_2$$

$$X_2 = A_2(\overline{A_0A_1}) + A_0A_1\bar{A}_2$$

Circuitos Específicos

inc

A_3	A_2	A_1	A_0	X_3	X_2	X_1	X_0
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0

	$\bar{A}_1\bar{A}_0$	\bar{A}_1A_0	A_1A_0	$A_1\bar{A}_0$
A_3A_2	00	01	11	10
$\bar{A}_3\bar{A}_2$	0	1	3	2
\bar{A}_3A_2	4	5	7	6
A_3A_2	12	13	15	14
$A_3\bar{A}_2$	8	9	11	10

$$\boxed{\bar{A}_1A_0} + \boxed{A_1\bar{A}_0}$$

$$X_1 = A_0 \oplus A_1$$

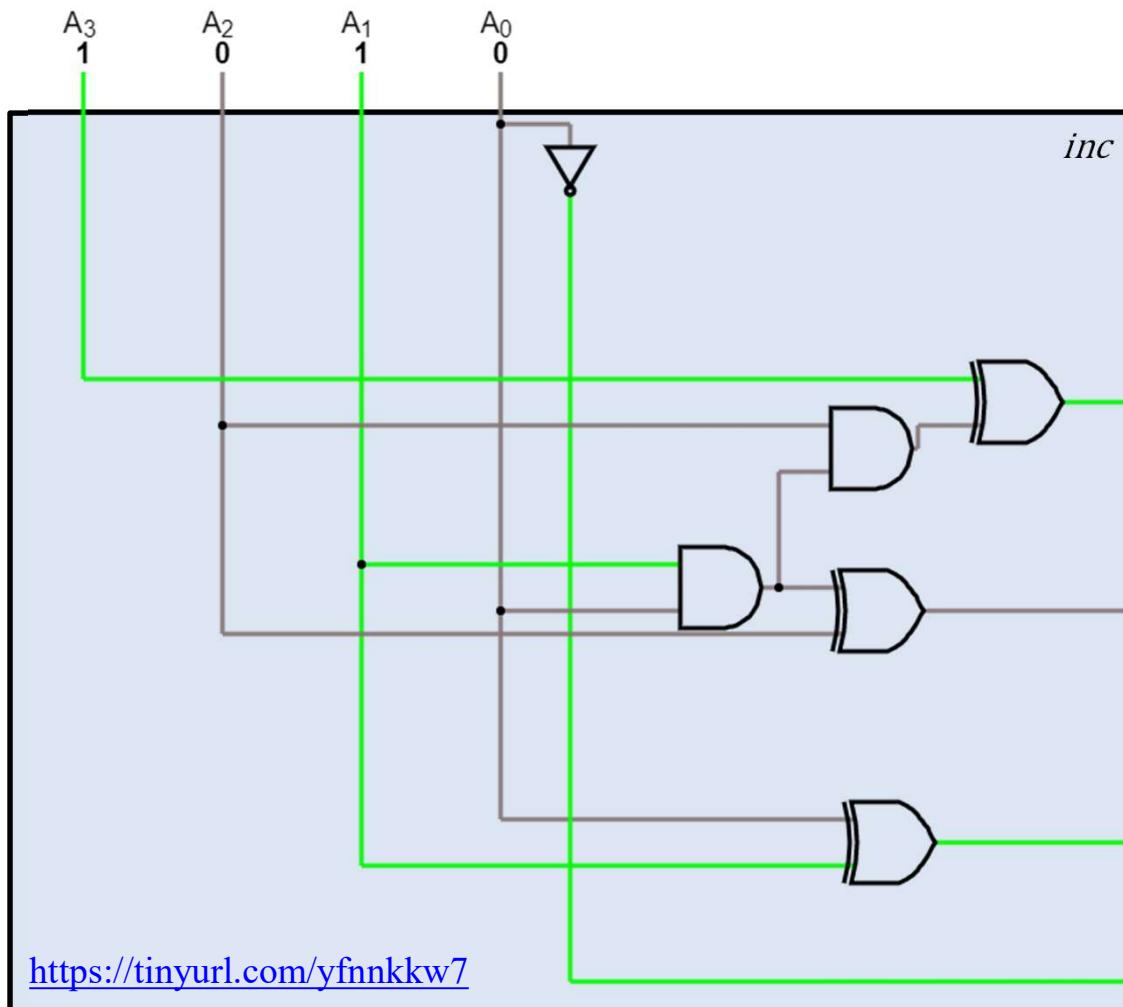
$$X_0 = \bar{A}_0$$

Circuitos Específicos

Exercício

- i. Implementar o circuito para a operação *inc*;
- ii. Obter as expressões para as operações *dec* e *neg* e implementar os respectivos circuitos;
- iii. Integrar os circuitos *inc* e *neg*;

Circuitos Específicos *inc*



$$X = A + 1$$

$$X_3 = A_3(\bar{A}_0 + \bar{A}_1) + A_0A_1(A_2 \oplus A_3)$$

$$X_3 = A_3(\overline{A_0A_1}) + A_0A_1(A_2 \oplus A_3)$$

$$X_3 = A_3 \oplus (A_0A_1A_2)$$

$$X_2 = A_2(\bar{A}_0 + \bar{A}_1) + A_0A_1\bar{A}_2$$

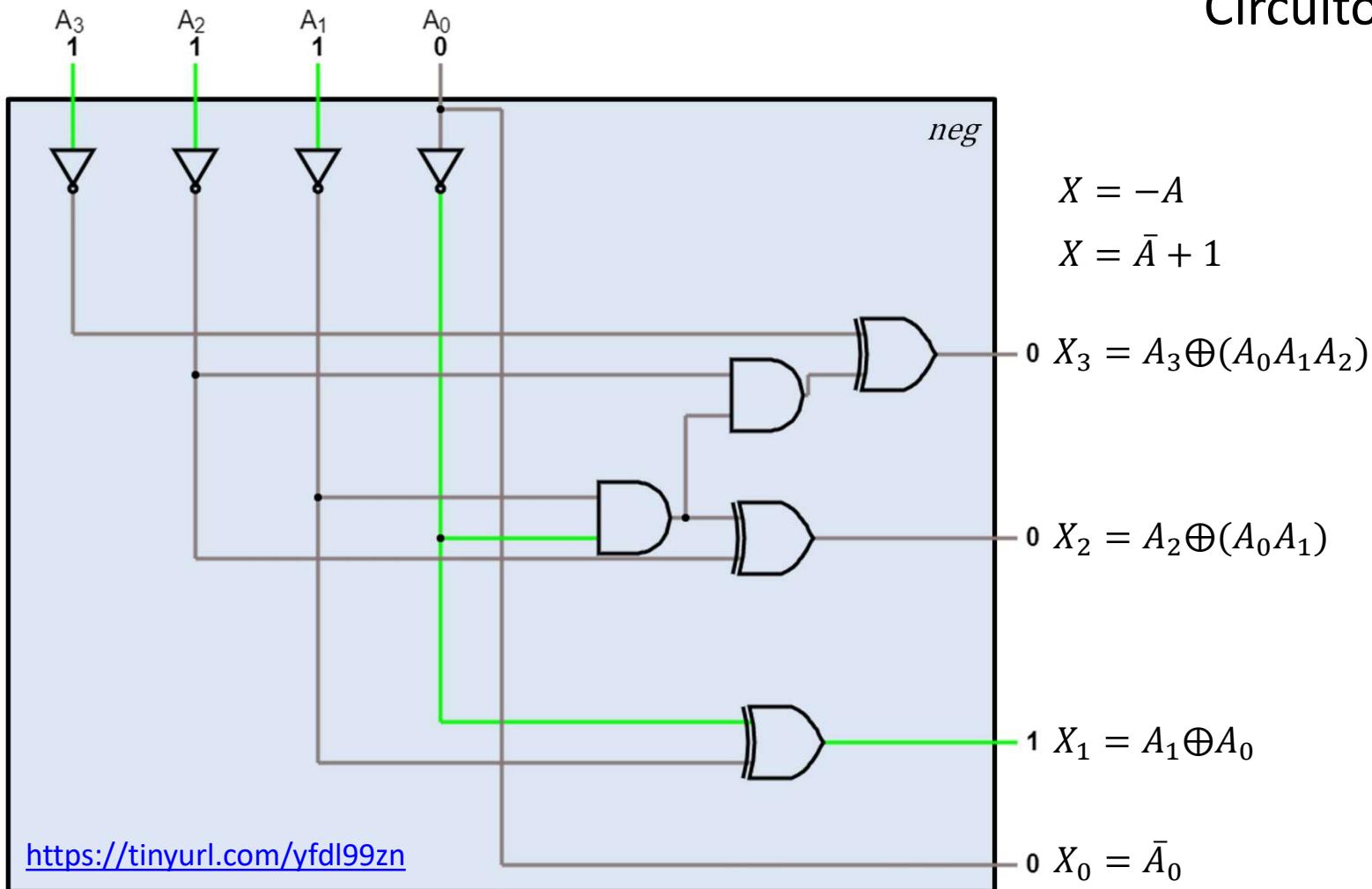
$$X_2 = A_2(\overline{A_0A_1}) + \bar{A}_2A_0A_1$$

$$X_2 = A_2 \oplus (A_0A_1)$$

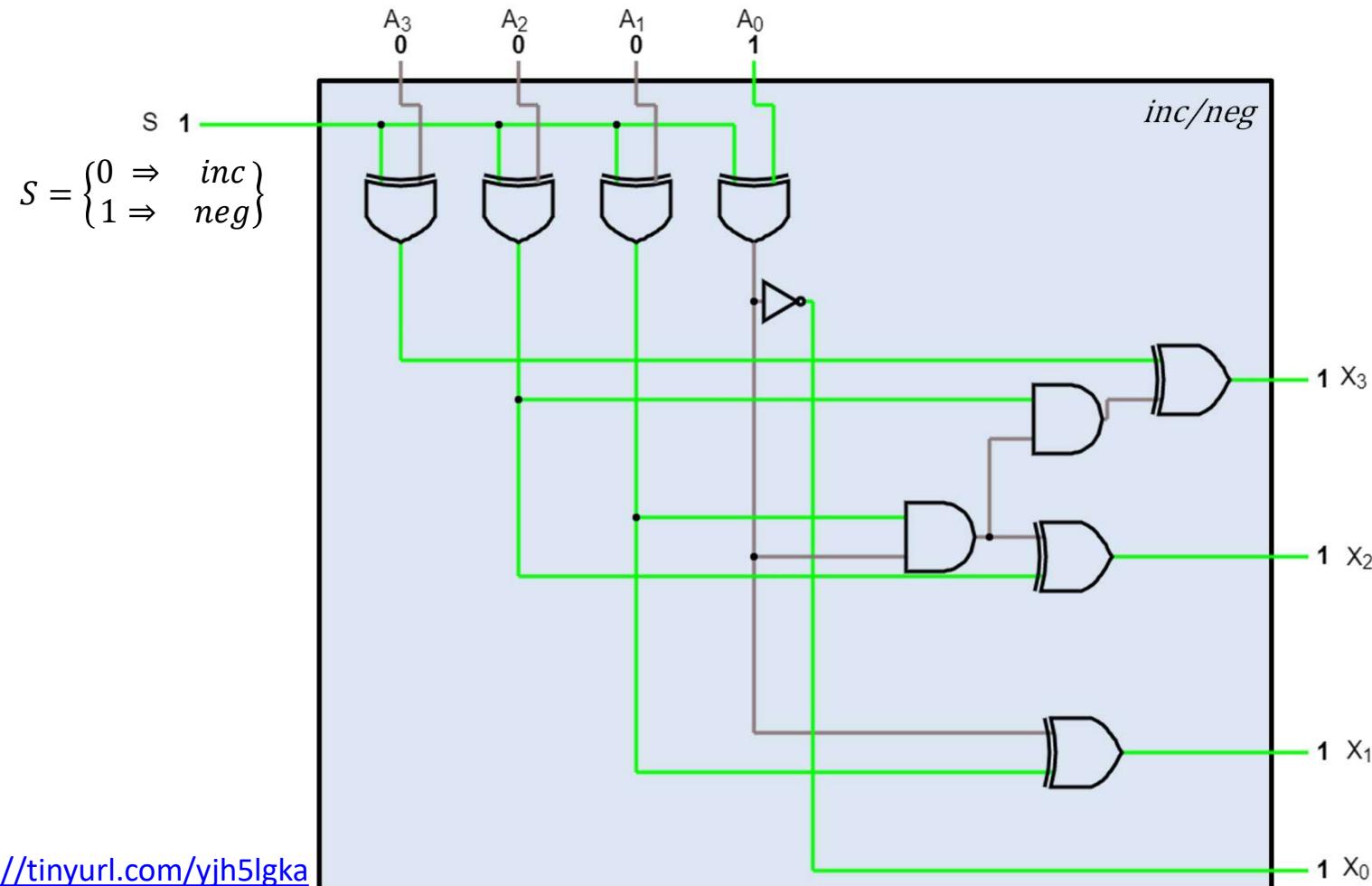
$$X_1 = A_1 \oplus A_0$$

$$X_0 = \bar{A}_0$$

Circuitos Específicos *neg*



Circuitos Específicos *inc/neg*



<https://tinyurl.com/yjh5lgka>

53

Circuitos Específicos

dec

A_3	A_2	A_1	A_0	X_3	X_2	X_1	X_0
0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1
0	0	1	1	0	0	1	0
0	1	0	0	0	0	1	1
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	1
0	1	1	1	0	1	1	0
1	0	0	0	0	1	1	1
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	1
1	0	1	1	1	0	1	0
1	1	0	0	1	0	1	1
1	1	0	1	1	1	0	0
1	1	1	0	1	1	0	1
1	1	1	1	1	1	1	0

	$\bar{A}_1\bar{A}_0$	\bar{A}_1A_0	A_1A_0	$A_1\bar{A}_0$
A_3A_2	00	01	11	10
$\bar{A}_3\bar{A}_2$	10	11	11	12
\bar{A}_3A_2	11	11	11	12
A_3A_2	12	13	13	14
$A_3\bar{A}_2$	11	11	11	10

$$\boxed{\bar{A}_1\bar{A}_0} + \boxed{A_1A_0}$$

$$X_1 = \overline{A_1 \oplus A_0} = A_1 \oplus \bar{A}_0$$

$$X_0 = \bar{A}_0$$

Circuitos Específicos

dec

A_3	A_2	A_1	A_0	X_3	X_2	X_1	X_0
0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1
0	0	1	1	0	0	1	0
0	1	0	0	0	0	1	1
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	1
0	1	1	1	0	1	1	0
1	0	0	0	0	1	1	1
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	1
1	0	1	1	1	0	1	0
1	1	0	0	1	0	1	1
1	1	0	1	1	1	0	0
1	1	1	0	1	1	0	1
1	1	1	1	1	1	1	0

	$\bar{A}_1\bar{A}_0$	\bar{A}_1A_0	A_1A_0	$A_1\bar{A}_0$
A_3A_2	00	01	11	10
$\bar{A}_3\bar{A}_2$	00	1	1	3
$A_3\bar{A}_2$	01	4	5	6
\bar{A}_3A_2	11	12	13	14
$A_3\bar{A}_2$	10	8	9	11

$$X_2 = A_2(\overline{A_0 + A_1}) + \bar{A}_0\bar{A}_1\bar{A}_2$$

$$A_2A_0 + \bar{A}_1\bar{A}_0\bar{A}_2 + A_1A_2$$

$$X_2 = A_2(\overline{\bar{A}_0\bar{A}_1}) + \bar{A}_2\bar{A}_0\bar{A}_1$$

$$X_2 = A_2A_0 + A_1A_2 + \bar{A}_1\bar{A}_0\bar{A}_2$$

$$X_2 = A_2 \oplus (\bar{A}_0\bar{A}_1)$$

$$X_2 = A_2(A_0 + A_1) + \bar{A}_0\bar{A}_1\bar{A}_2$$

Circuitos Específicos

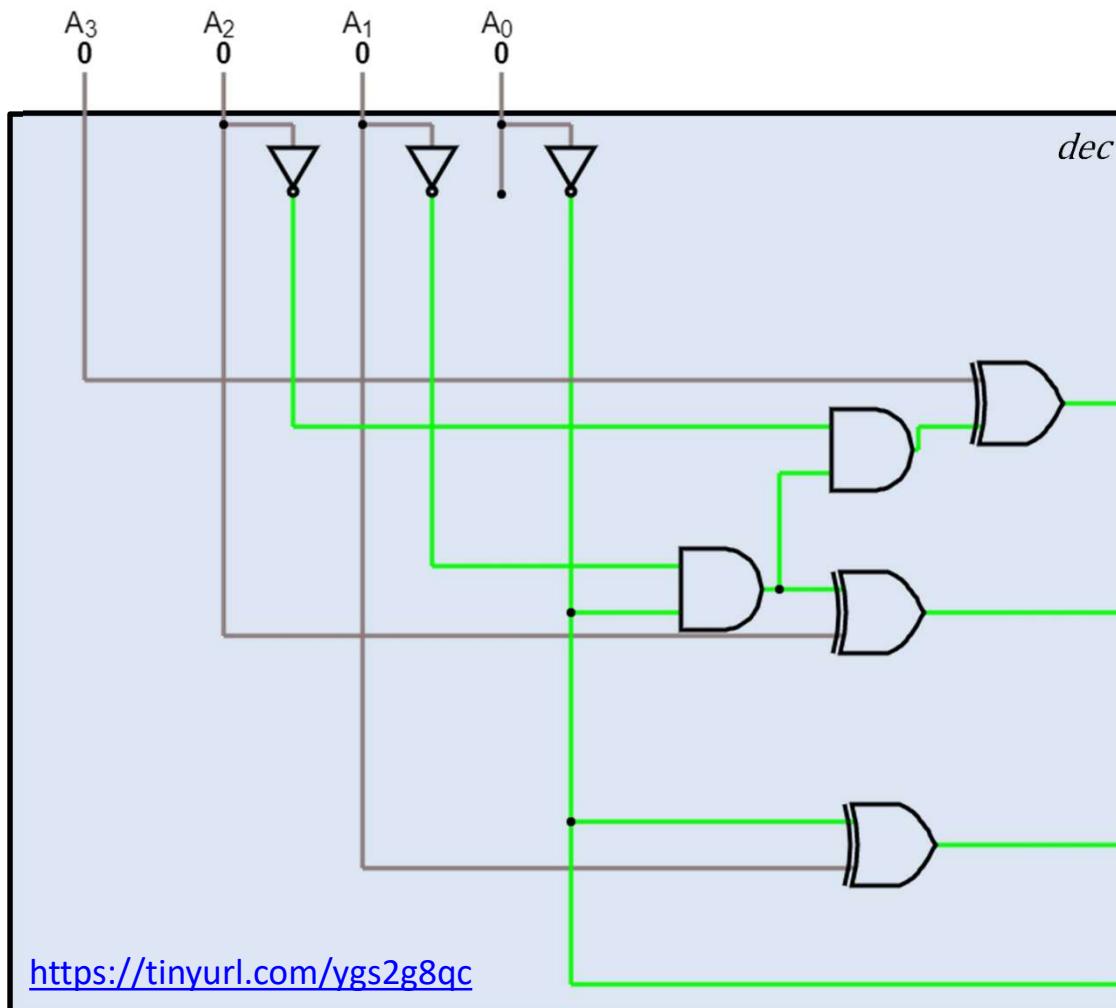
dec

A_3	A_2	A_1	A_0	X_3	X_2	X_1	X_0
0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1
0	0	1	1	0	0	1	0
0	1	0	0	0	0	1	1
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	1
0	1	1	1	0	1	1	0
1	0	0	0	0	1	1	1
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	1
1	0	1	1	1	0	1	0
1	1	0	0	1	0	1	1
1	1	0	1	1	1	0	0
1	1	1	0	1	1	0	1
1	1	1	1	1	1	1	0

	$\bar{A}_1\bar{A}_0$	\bar{A}_1A_0	A_1A_0	$A_1\bar{A}_0$
A_3A_2	00	01	11	10
$\bar{A}_3\bar{A}_2$	00	1	3	2
\bar{A}_3A_2	01	4	5	6
A_3A_2	11	1	1	1
$A_3\bar{A}_2$	10	12	13	14

$$\begin{aligned}
 X_3 &= A_3(\overline{A_0 + A_1 + A_2}) + \bar{A}_3\bar{A}_0\bar{A}_1\bar{A}_2 \quad \boxed{A_3A_2} + \boxed{A_3A_0} + \boxed{A_3A_1} + \boxed{\bar{A}_3\bar{A}_2\bar{A}_1\bar{A}_0} \\
 X_3 &= A_3(\overline{\bar{A}_0\bar{A}_1\bar{A}_2}) + \bar{A}_3\bar{A}_0\bar{A}_1\bar{A}_2 \quad X_3 = A_3A_2 + A_3A_0 + A_3A_1 + \bar{A}_3\bar{A}_2\bar{A}_1\bar{A}_0 \\
 X_3 &= A_3 \oplus (\bar{A}_0\bar{A}_1\bar{A}_2) \quad X_3 = A_3(A_0 + A_1 + A_2) + \bar{A}_0\bar{A}_1\bar{A}_2\bar{A}_3
 \end{aligned}$$

Circuitos Específicos *dec*



$$X = A - 1$$

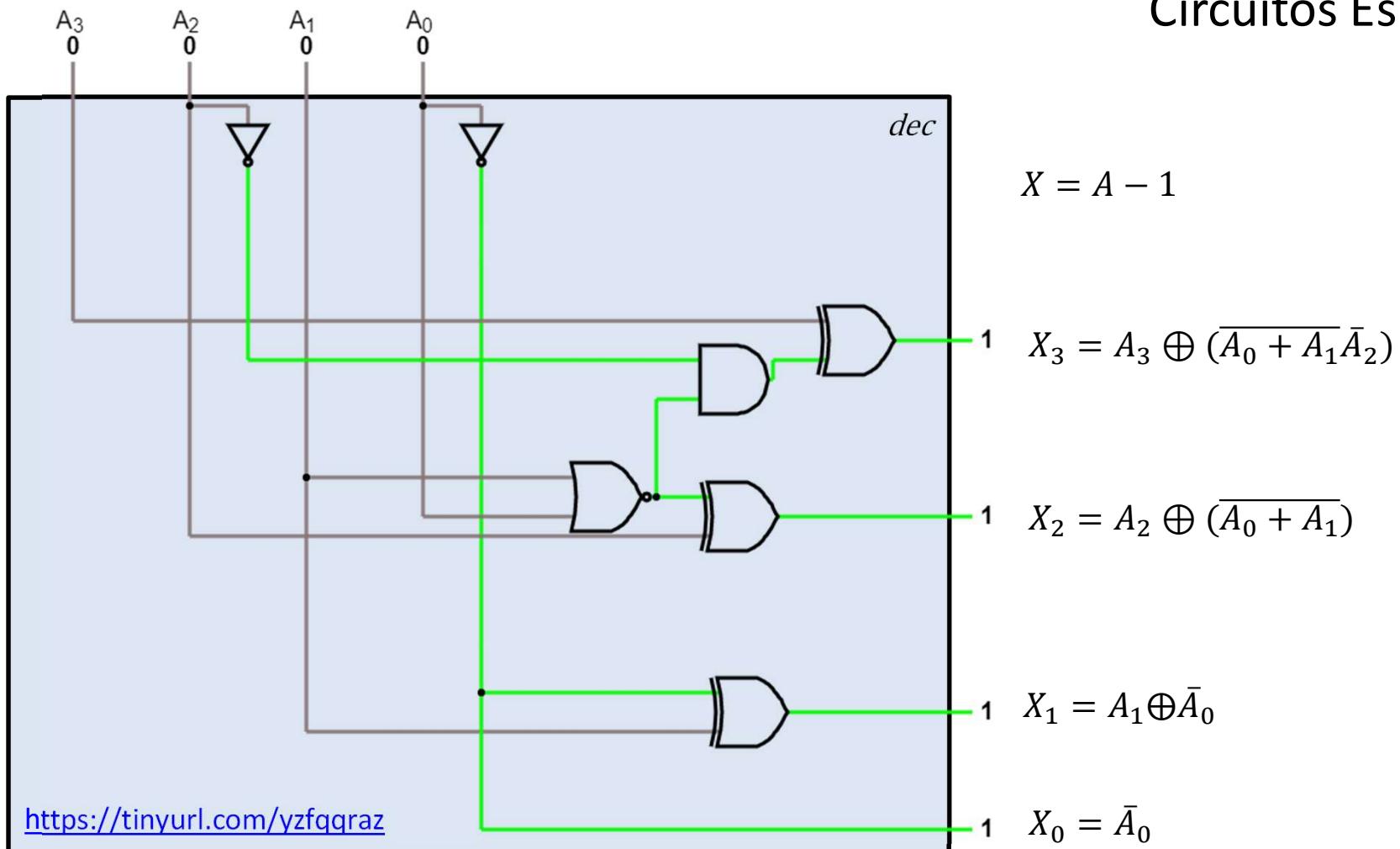
$$1 \quad X_3 = A_3 \oplus (\bar{A}_0 \bar{A}_1 \bar{A}_2)$$

$$1 \quad X_2 = A_2 \oplus (\bar{A}_0 \bar{A}_1)$$

$$1 \quad X_1 = A_1 \oplus \bar{A}_0$$

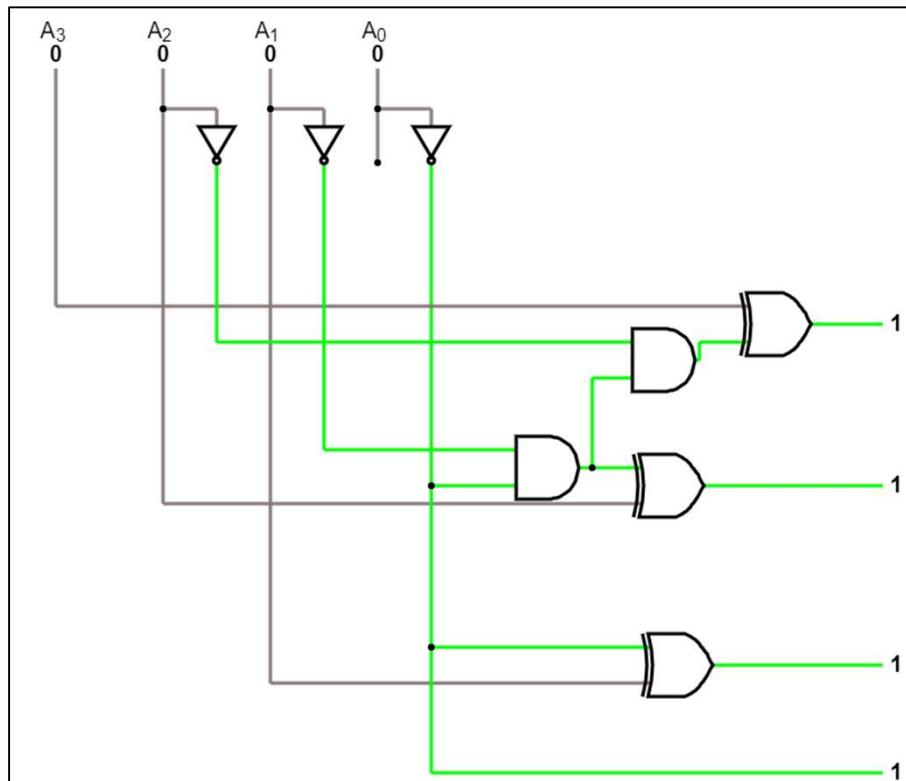
$$1 \quad X_0 = \bar{A}_0$$

Circuitos Específicos *dec*

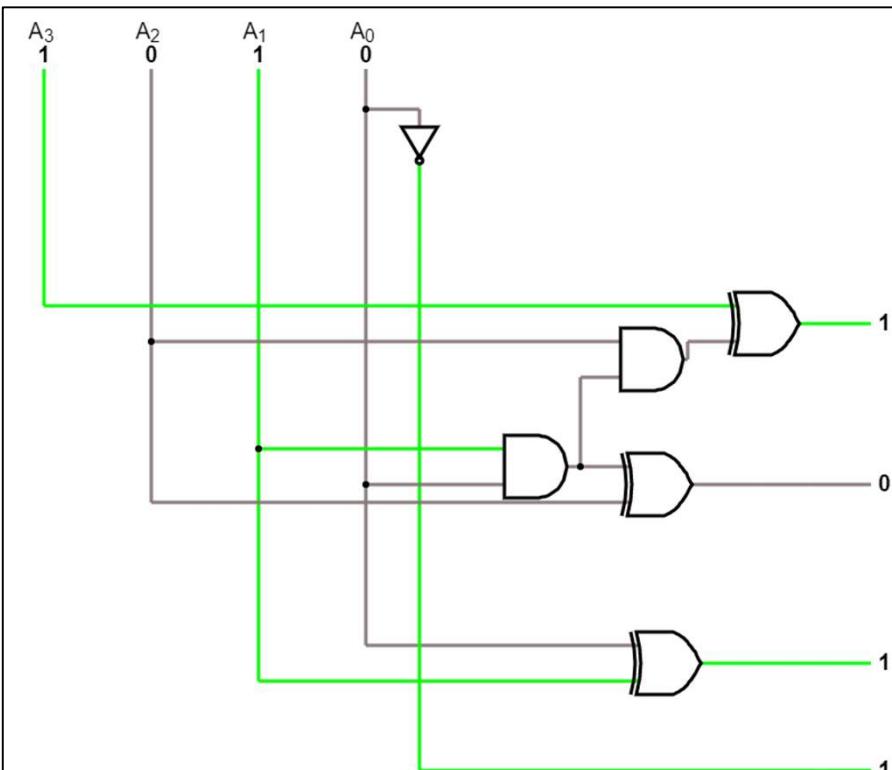


Circuitos Específicos *inc/dec*

$X = A - 1$



$X = A + 1$



<https://tinyurl.com/ygs2g8qc>

<https://tinyurl.com/yfnnkkw7>

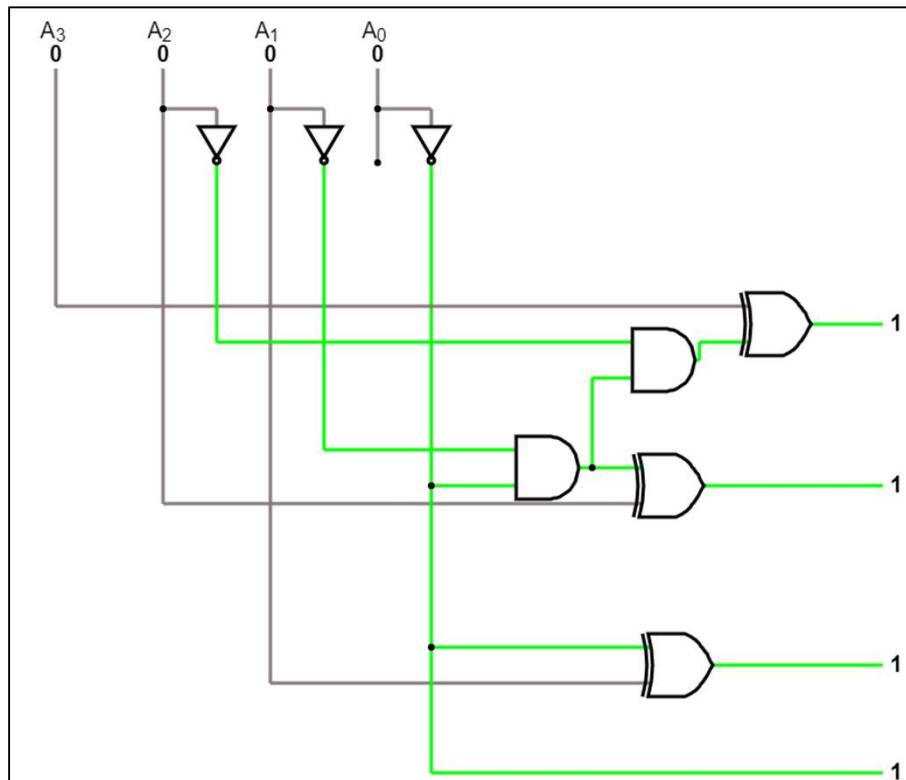
59

Circuitos Específicos

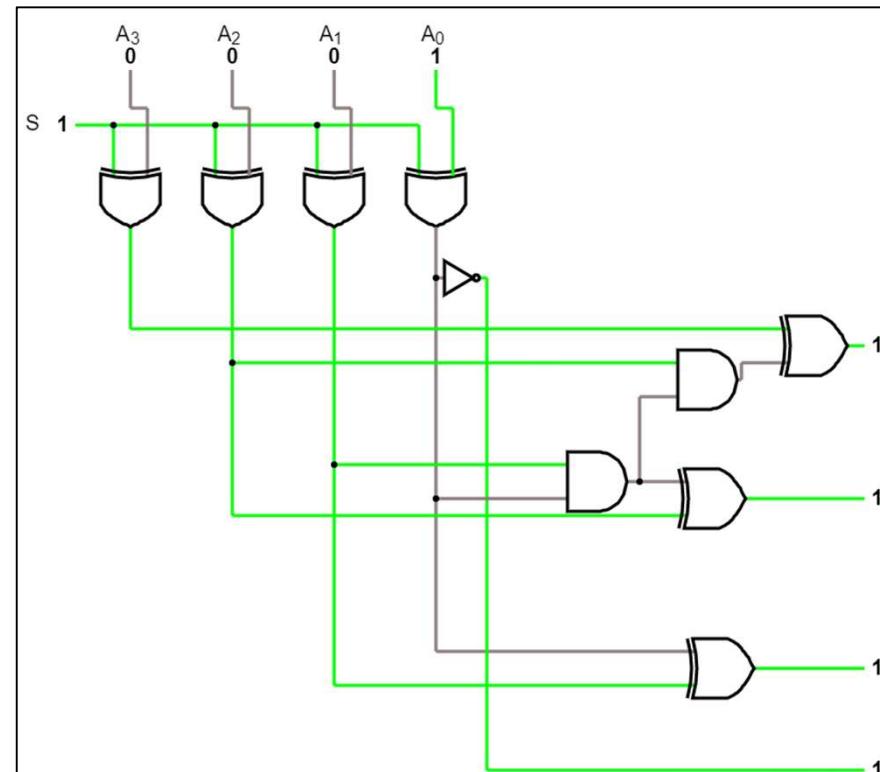
inc/dec/neg

$$X_s = \begin{bmatrix} A + 1 \\ -A \end{bmatrix}$$

$$X = A - 1$$



<https://tinyurl.com/ygs2g8qc>



<https://tinyurl.com/ye452j7p>

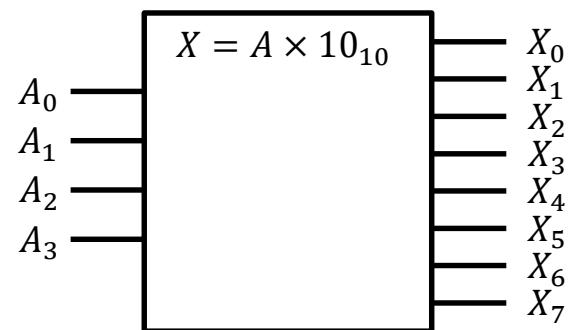
60

Exercício

- Propor uma versão integrada dos circuitos *inc/neg/dec*:

Exercício

- Elaborar um circuito que realize a multiplicação por 10:



Exercício
 $X = A \times 10$

$$X = A \times 10$$

$$X = A \times (2 + 8)$$

$$X = A \times 2 + A \times 8$$

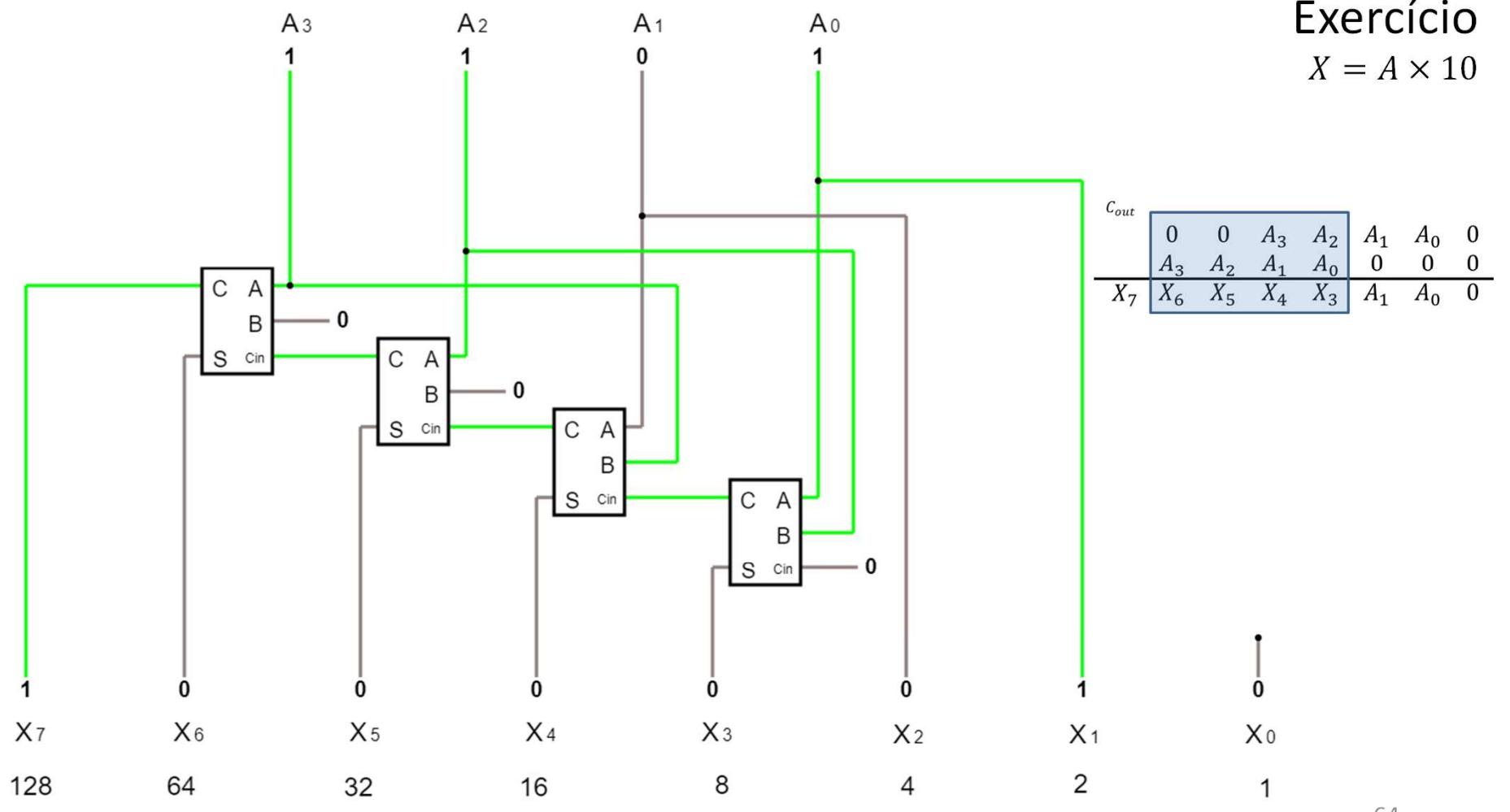
$$X = A \times 2^1 + A \times 2^3$$

$$X = A_3 A_2 A_1 A_0 \times 2^1 + A_3 A_2 A_1 A_0 \times 2^3$$

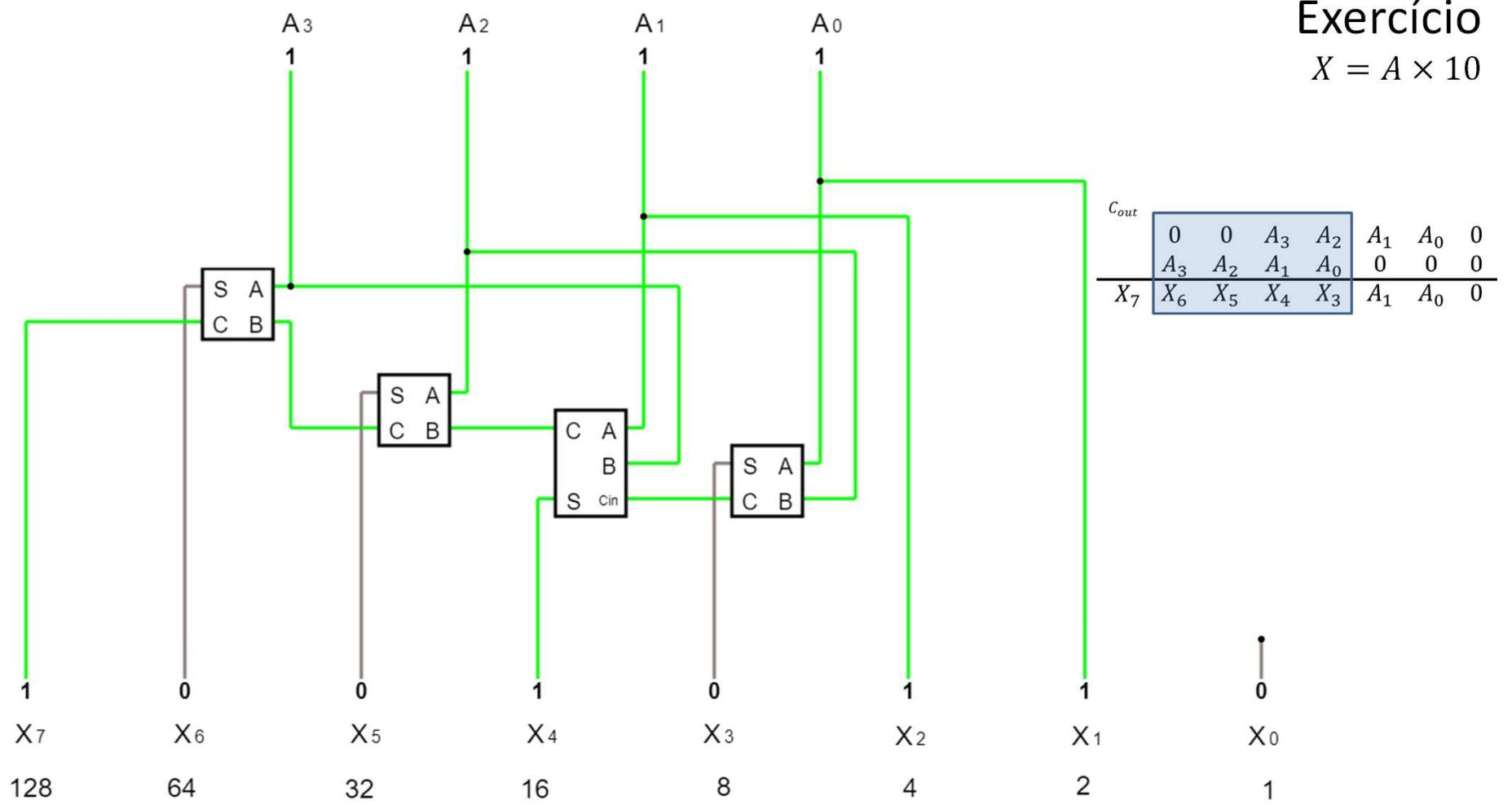
$$X = A_3 A_2 A_1 A_0 0 + A_3 A_2 A_1 A_0 000$$

c_{out}	0	0	A_3	A_2	A_1	A_0	0
	A_3	A_2	A_1	A_0	0	0	0
X_7	X_6	X_5	X_4	X_3	A_1	A_0	0

Exercício
 $X = A \times 10$



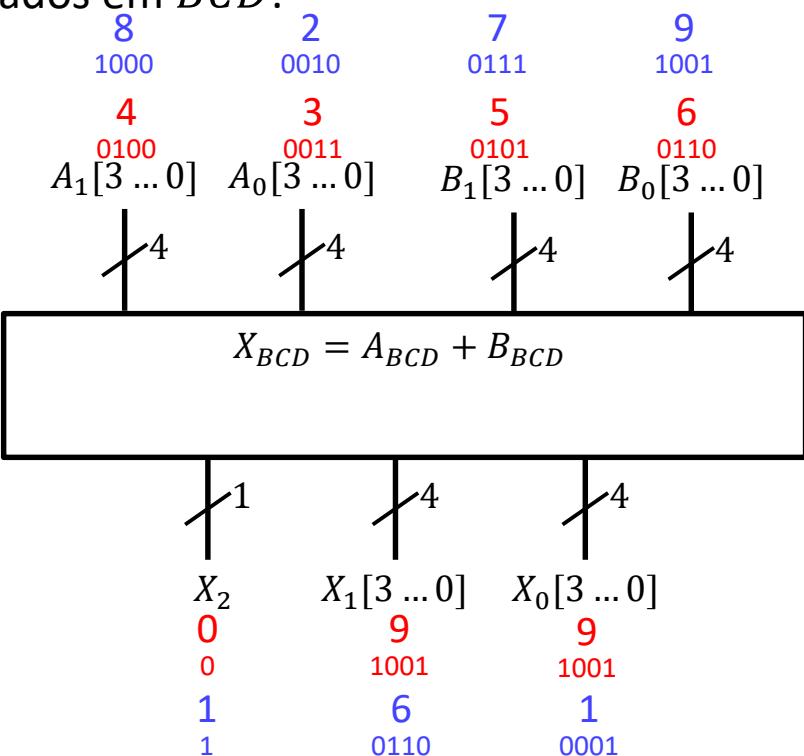
Exercício

$$X = A \times 10$$


<https://tinyurl.com/yj2t32z3>

Exercício

- Elaborar um circuito que realize a soma de dois inteiros de dois dígitos cada codificados em *BCD*:



$\begin{array}{r} \textcolor{red}{11} \\ 0110 \\ + 0010 \\ \hline \end{array}$	$\begin{array}{r} 6 \\ + 2 \\ \hline \end{array}$
$\begin{array}{r} - \\ - \\ \hline \end{array}$	
$\begin{array}{r} 1000 \\ \hline \end{array}$	$\begin{array}{r} 8 \\ \hline \end{array}$

$\begin{array}{r} \textcolor{red}{1} \\ 1001 \\ + 1000 \\ \hline \end{array}$	$\begin{array}{r} 9 \\ + 8 \\ \hline \end{array}$
$\begin{array}{r} - \\ - \\ \hline \end{array}$	
$\begin{array}{r} 0001 \\ \hline \end{array}$	$\begin{array}{r} 17 \\ \hline \end{array}$

$\begin{array}{r} + 0110 \\ \hline \end{array}$	$\begin{array}{r} \\ \hline \end{array}$
$\begin{array}{r} - \\ - \\ \hline \end{array}$	
$\begin{array}{r} 0001 \text{ } 0111 \\ \hline \end{array}$	$\begin{array}{r} 66 \\ \hline \end{array}$

Exercício

Soma em BCD

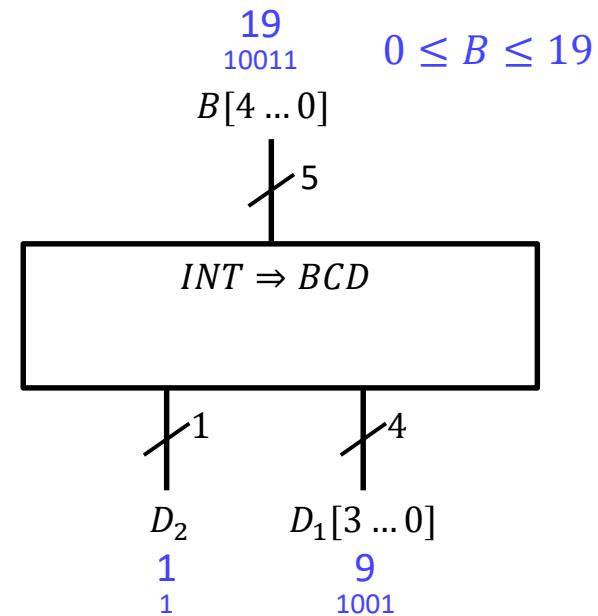
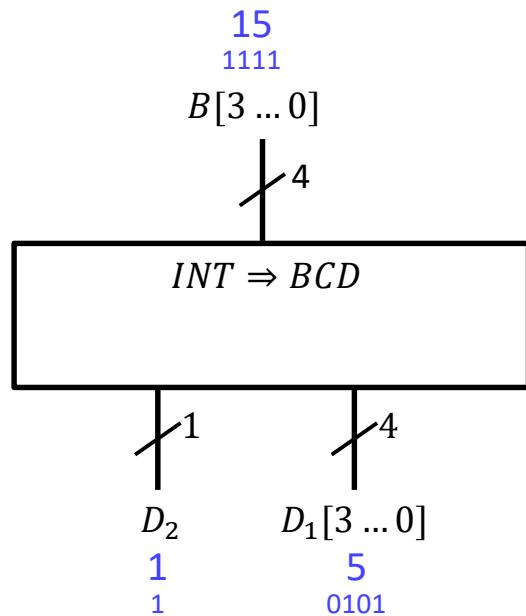
- ✓ A soma de inteiros sem sinal, padrão *BCD*, é feita através da soma binária dos dígitos correspondentes, realizando ajustes quando necessário;
- ✓ Os passos para a soma de números em BCD são:
 - ✓ Somar os dígitos usando a soma binária;
 - ✓ Se o dígito BCD resultante (4 bits) for menor ou igual a 9, então o resultado é válido;
 - ✓ Se o dígito BCD resultante (4 bits) for maior que 9, ou surgiu um *carry* final, então o resultado é inválido:
 - ❑ Adicionar 6 (0110) ao resultado de 4 bits a fim de saltar as seis combinações inválidas para BCD 8421;
 - ❑ Se ocorrer um *carry* após este ajuste, simplesmente adicione o *carry* ao próximo grupo de 4 bits;

$$\begin{array}{r} \textcolor{red}{11} \\ 0110 \quad 6 \\ + 0010 \quad + 2 \\ \hline \hline 1000 \quad 8 \end{array}$$

$$\begin{array}{r} \textcolor{red}{1} \\ 1001 \quad 9 \\ + 1000 \quad + 8 \\ \hline \hline 0001 \quad 17 \\ + 0110 \\ \hline \hline 0001 \textcolor{blue}{0111} \end{array}$$

Exercício

- Elaborar um circuito que converta um inteiro sem sinal de 4 bits para *BCD*:



Exercício

B_4	B_3	B_2	B_1	B_0	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	0	0	1	0
0	0	0	1	1	0	0	0	1	1
0	0	1	0	0	0	0	1	0	0
0	0	1	0	1	0	0	1	0	1
0	0	1	1	0	0	0	1	1	0
0	0	1	1	1	0	0	1	1	1
0	1	0	0	0	0	1	0	0	0
0	1	0	0	1	0	1	0	0	1
0	1	0	1	0	1	0	0	0	0
0	1	0	1	1	1	0	0	0	1
0	1	1	0	0	1	0	0	1	0
0	1	1	0	1	1	0	0	1	1
0	1	1	1	0	1	0	1	0	0
0	1	1	1	1	1	0	1	0	1
1	0	0	0	0	1	0	1	1	0
1	0	0	0	1	1	0	1	1	1
1	0	0	1	0	1	1	0	0	0
1	0	0	1	1	1	1	0	0	1

B_4	B_3	B_2	B_1	D_4	D_3	D_2	D_1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0
0	1	0	0	0	1	0	0
0	1	0	1	1	0	0	0
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	0	1	1	1	0	0

$$D_0 = B_0$$

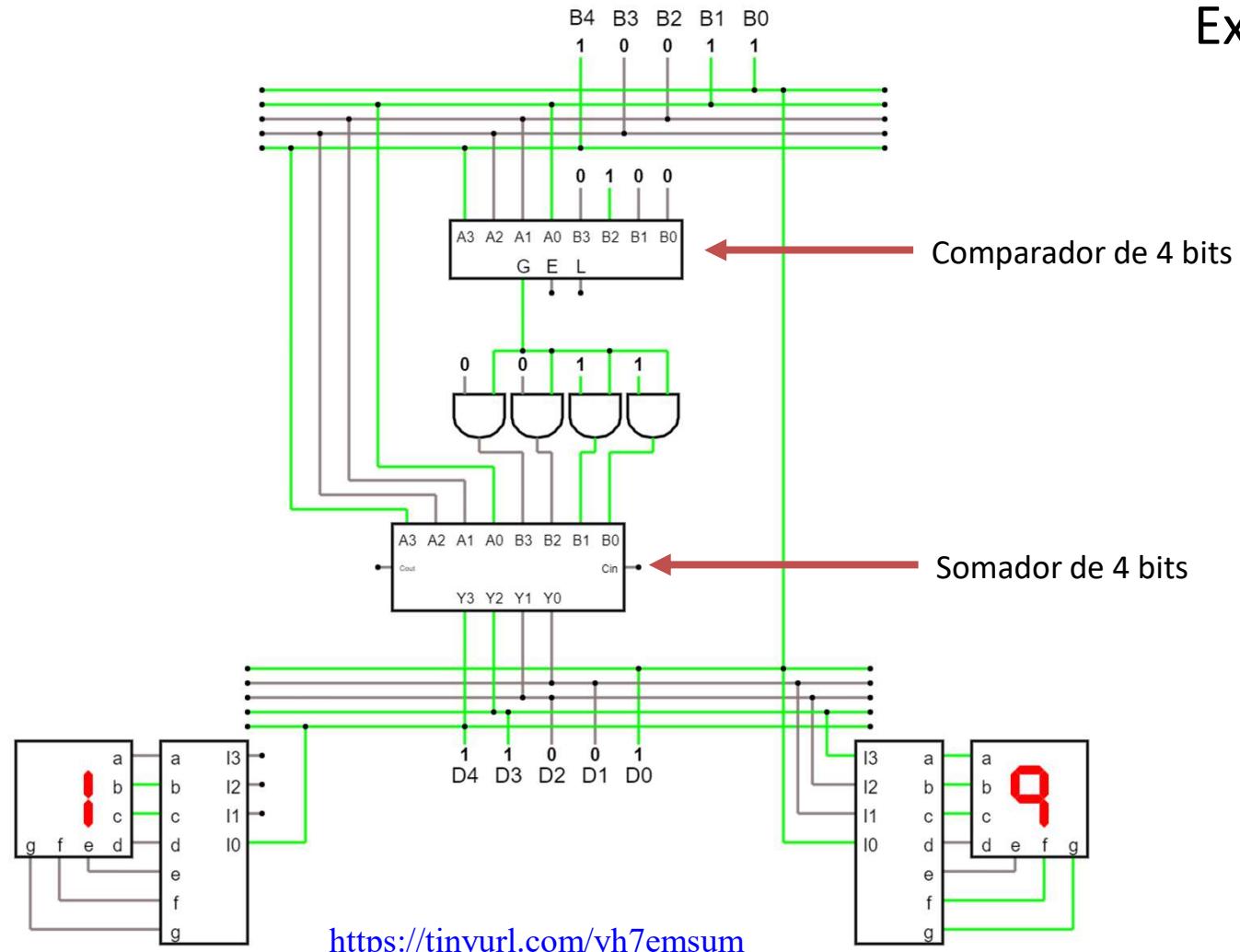
http://eng.staff.alexu.edu.eg/~mbanna/Logic_Circuit_Design_EE242/EE242_Class_Notes_%20logic%20design.pdf

$$B_4 B_3 B_2 B_1 \leq 0100 \Rightarrow \\ D_4 D_3 D_2 D_1 = B_4 B_3 B_2 B_1$$

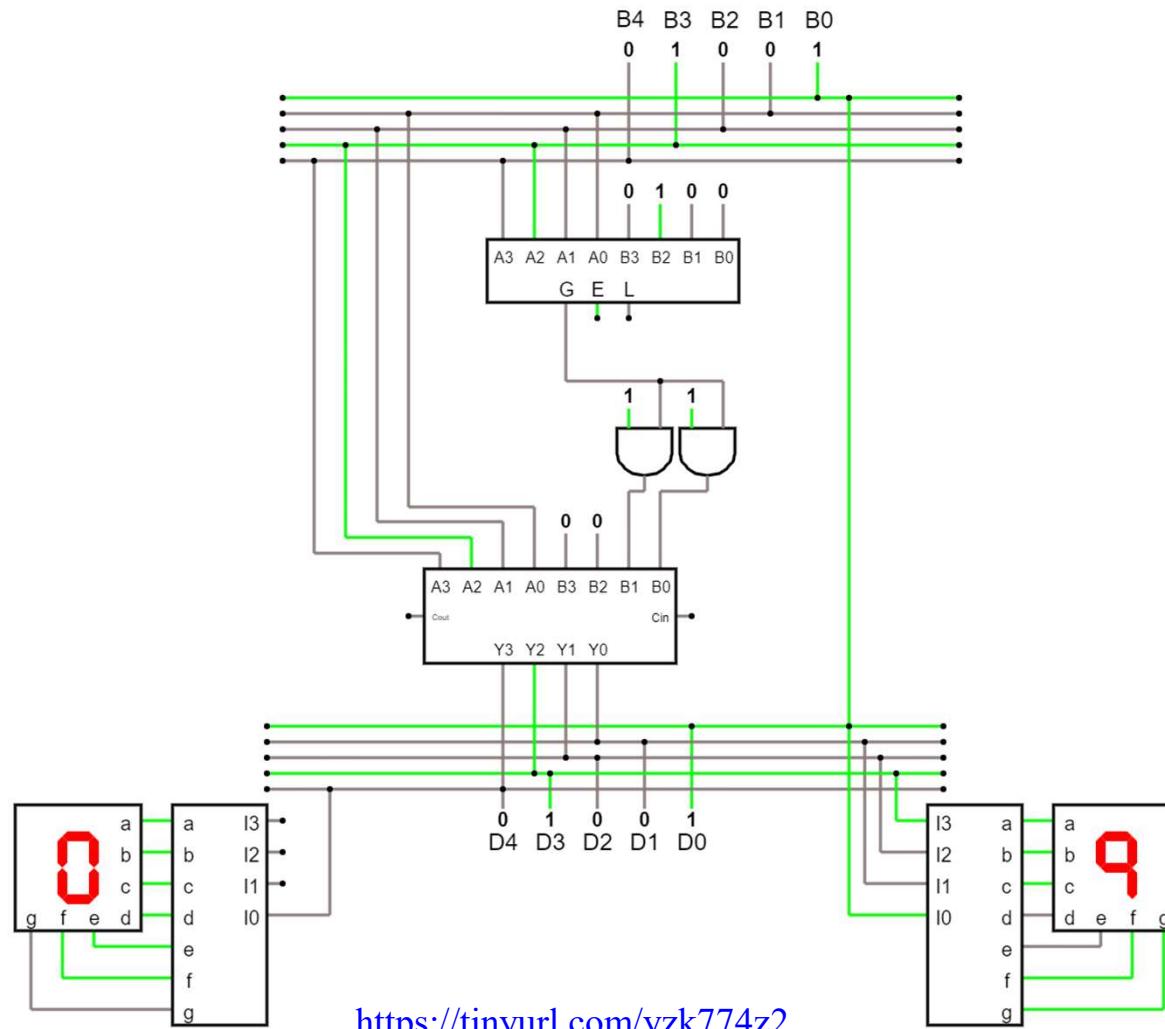
- ✓ Dedução da lógica a partir de análise;
 - ✓ Considerando como entrada válida $0 \leq B \leq 19$;

$$B_4B_3B_2B_1 > 0100 \Rightarrow \\ D_4D_3D_2D_1 = B_4B_3B_2B_1 + 0011$$

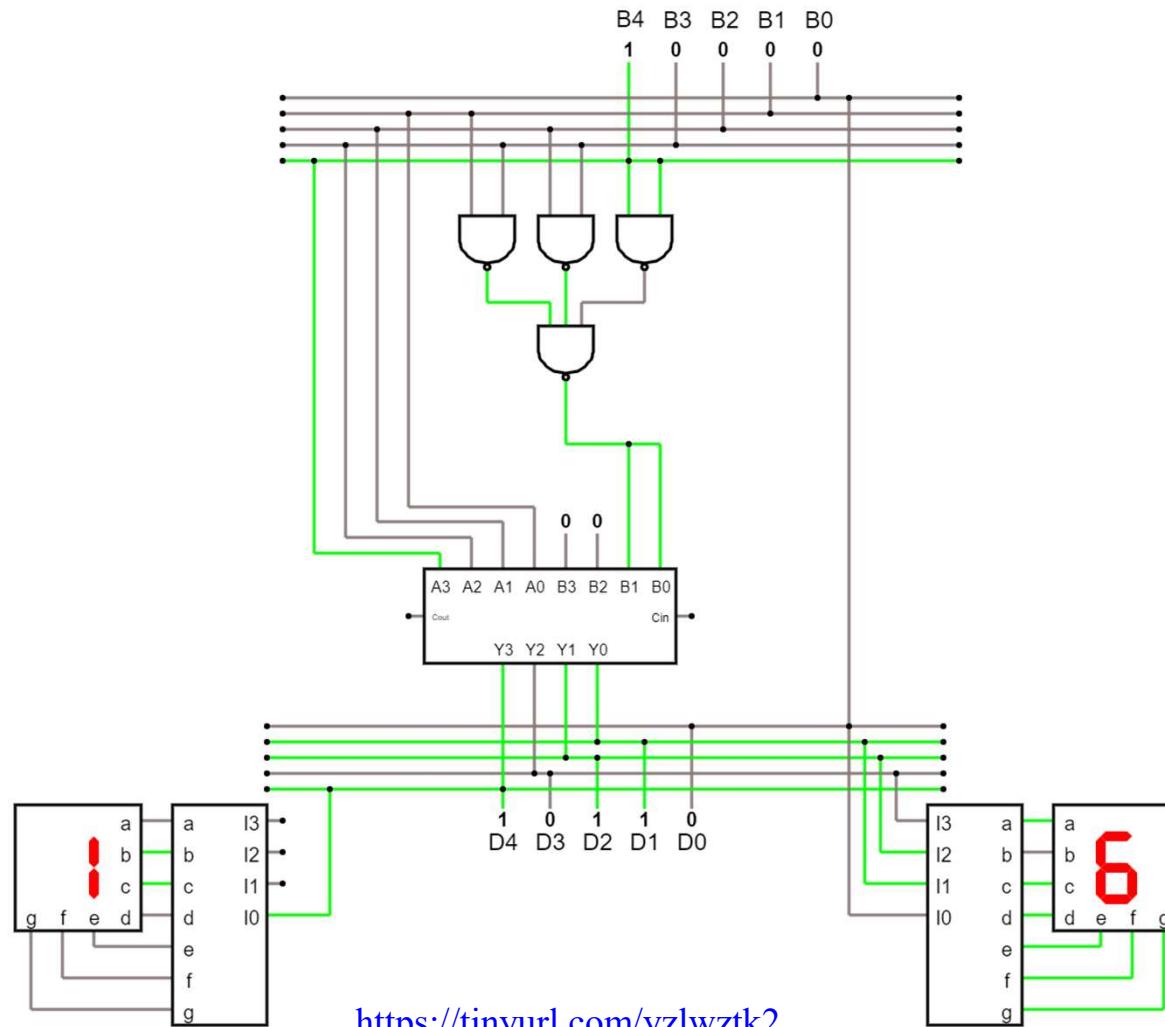
Exercício



Exercício



Exercício



Exercício

Obtenção das expressões de forma convencional – com mapa de Karnaugh:

$$D_4 = B_3B_1 + B_3B_2 + B_4$$

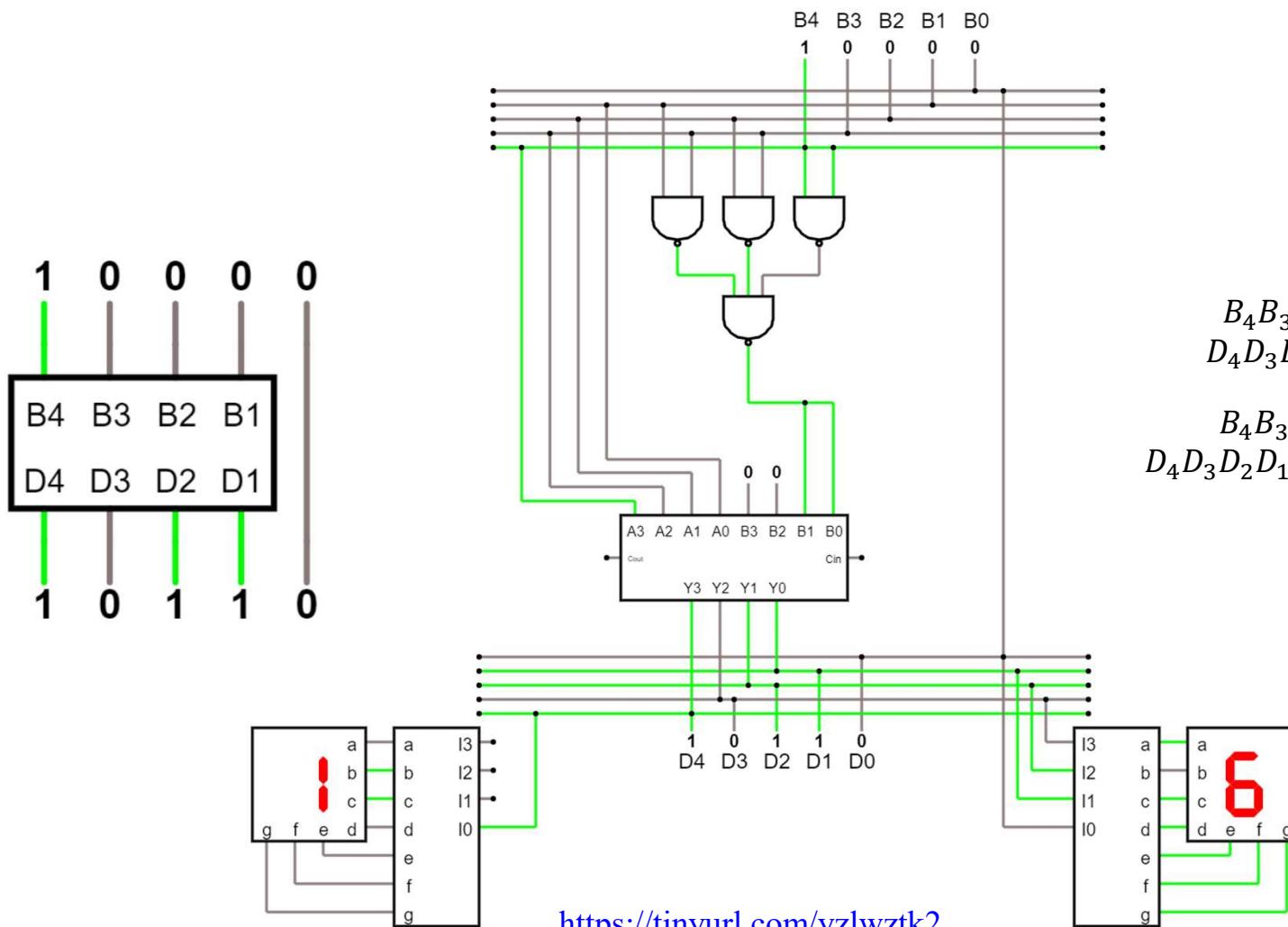
$$D_3 = B_4B_1 + B_3\bar{B}_2\bar{B}_1$$

$$D_2 = B_2B_1 + B_4\bar{B}_1 + \bar{B}_3B_2$$

$$D_1 = \bar{B}_4\bar{B}_3B_1 + B_3B_2\bar{B}_1 + B_4\bar{B}_1$$

$$D_0 = B_0$$

Exercício



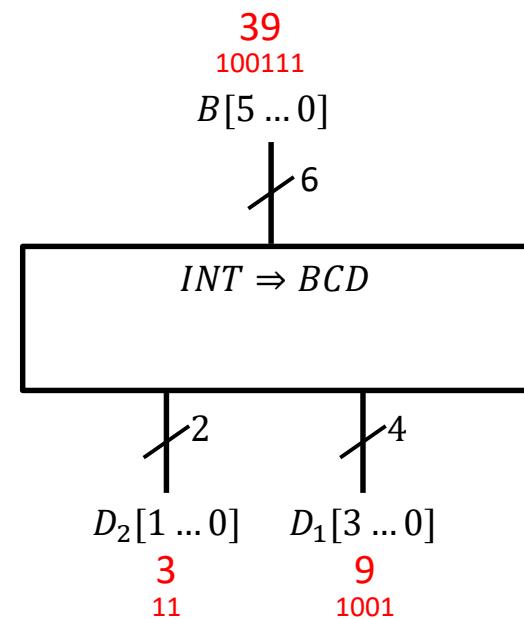
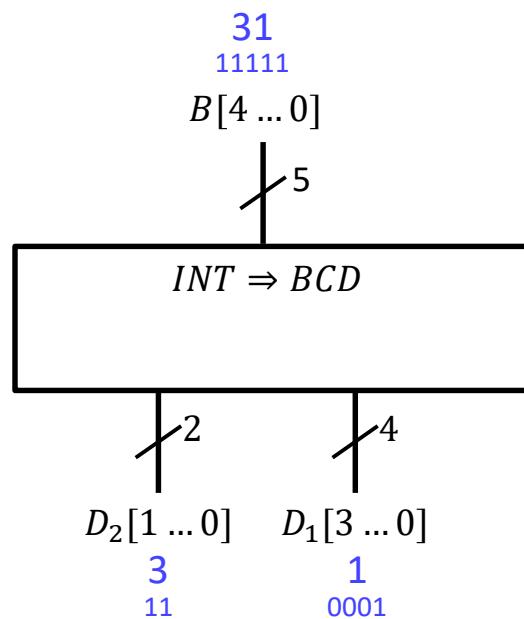
$$B_4 B_3 B_2 B_1 \leq 0100 \Rightarrow \\ D_4 D_3 D_2 D_1 = B_4 B_3 B_2 B_1$$

$$B_4 B_3 B_2 B_1 > 0100 \Rightarrow \\ D_4 D_3 D_2 D_1 = B_4 B_3 B_2 B_1 + 0011$$

<https://tinyurl.com/yzlwztk2>

Exercício

- Elaborar um circuito que converta um inteiro sem sinal de 5 bits para *BCD*:



Exercício

$$\begin{array}{r} 1 \ 1 \ 1 \ 1 \ 1 \\ \times 2^5 \\ \hline = 31 \end{array}$$

BCD.INT

$$0.1 \ 1 \ 1 \ 1 \ 1 \times 2^5 = 31$$

$$\boxed{1.} \ 1 \ 1 \ 1 \ 1 \times 2^4 = 16 + 15 = 31$$

$$\boxed{1 \ 1.} \ 1 \ 1 \ 1 \times 2^3 = 24 + 7 = 31$$

$$\boxed{1 \ 1 \ 1.} \ 1 \ 1 \times 2^2 = 28 + 3 = 31$$

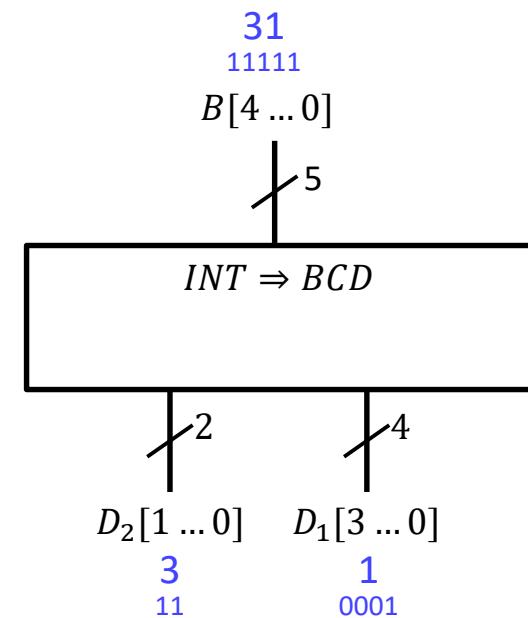
$$\boxed{1 \ 1 \ 1 \ 1.} \ 1 \times 2^1 = ??$$

BCD inválido: Dígito acima de 9. Neste caso teria que somar 6:

- $1111(15/??) + 0110(6) = \textcolor{blue}{1}0101(21/15);$

O mesmo efeito pode ser obtido fazendo o ajuste na etapa anterior: sempre que o valor exceder a 4, soma-se 3. Antes de deslocar o próximo bit:

- $111(7) + 011(3) = 1010(10) + \textit{bit deslocado} = 10101(21/15)$



Exercício

$$1 \ 1 \ 1 \ 1 \ 1 = 31$$

BCD.INT

$$0.1 \ 1 \ 1 \ 1 \ 1 \times 2^5 = 31$$

$$\boxed{1}.1 \ 1 \ 1 \ 1 \times 2^4 = 16 + 15 = 31$$

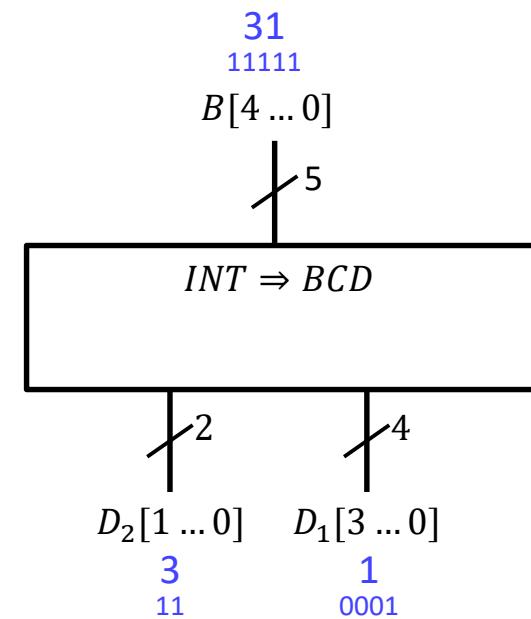
$$\boxed{1} \ 1.\boxed{1} \ 1 \ 1 \times 2^3 = 24 + 7 = 31$$

$$\boxed{1} \ 1 \ 1.\boxed{1} \ 1 \times 2^2 = 28 + 3 = 31$$

$$\begin{array}{|c|c|c|c|c|c|} \hline & 1 & 0 & 1 & 0 & 1 \\ \hline \end{array} \times 2^1 = 30 + 1 = 31$$

$$1 \ 0 \ \boxed{1 \ 0 \ 1 \ 1}.0 \times 2^0 = ??$$

$$\begin{aligned} B_4 B_3 B_2 B_1 &> 0100 \Rightarrow \\ D_4 D_3 D_2 D_1 &= B_4 B_3 B_2 B_1 + 0011 \end{aligned}$$



Exercício

$$1 \ 1 \ 1 \ 1 \ 1 = 31$$

BCD.INT

$$0.1 \ 1 \ 1 \ 1 \ 1 \times 2^5 = 31$$

$$\boxed{1.} \ 1 \ 1 \ 1 \ 1 \times 2^4 = 16 + 15 = 31$$

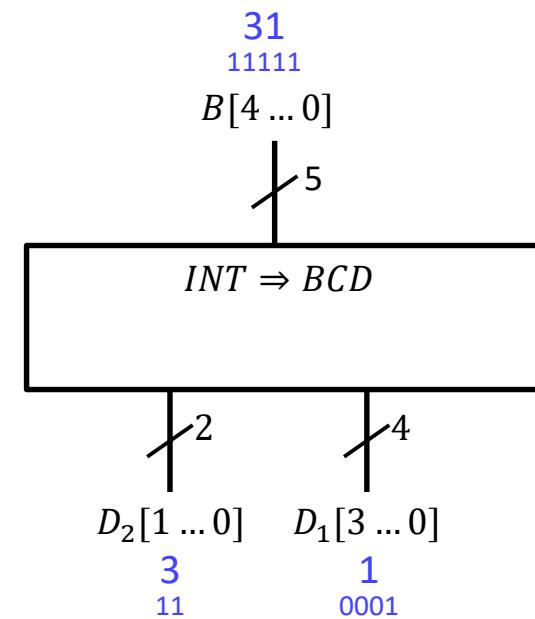
$$\boxed{1 \ 1.} \ 1 \ 1 \ 1 \times 2^3 = 24 + 7 = 31$$

$$\boxed{1 \ 1 \ 1.} \ 1 \ 1 \times 2^2 = 28 + 3 = 31$$

$$\begin{array}{r} \boxed{1 \ 0 \ 1 \ 0 \ 1.} \\ \hline 1 \ 0 \ 1 \ 0 \ 1.1 \end{array} \times 2^1 = 30 + 1 = 31$$

$$\begin{array}{r} \boxed{1 \ 1 \ 0 \ 0 \ 0 \ 1.} \\ \hline 1 \ 1 \ 0 \ 0 \ 0 \ 1.0 \end{array} \times 2^0 = 31$$

$$B_4 B_3 B_2 B_1 > 0100 \Rightarrow \\ D_4 D_3 D_2 D_1 = B_4 B_3 B_2 B_1 + 0011$$



Exercício

$$\begin{array}{r} 1 \ 1 \ 1 \ 1 \ 1 \\ \times 2^4 \\ \hline = 31 \end{array}$$

BCD.INT

$$0.1 \ 1 \ 1 \ 1 \ 1 \times 2^5 = 31$$

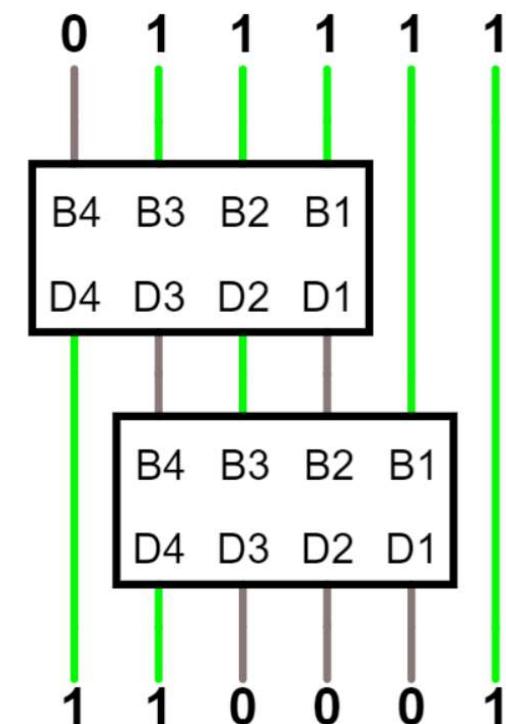
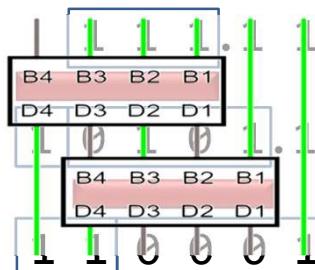
$$\boxed{1.1} \ 1 \ 1 \ 1 \times 2^4 = 16 + 15 = 31$$

$$\boxed{1 \ 1.1} \ 1 \ 1 \times 2^3 = 24 + 7 = 31$$

$$\begin{array}{c} \boxed{1 \ 1 \ 1} \\ \times 2^2 = 28 + 3 = 31 \end{array}$$

$$\begin{array}{c} \times 2^1 = 30 + 1 = 31 \\ \boxed{1 \ 1 \ 0 \ 0 \ 0} \end{array}$$

$$.0 \times 2^0 = 31$$

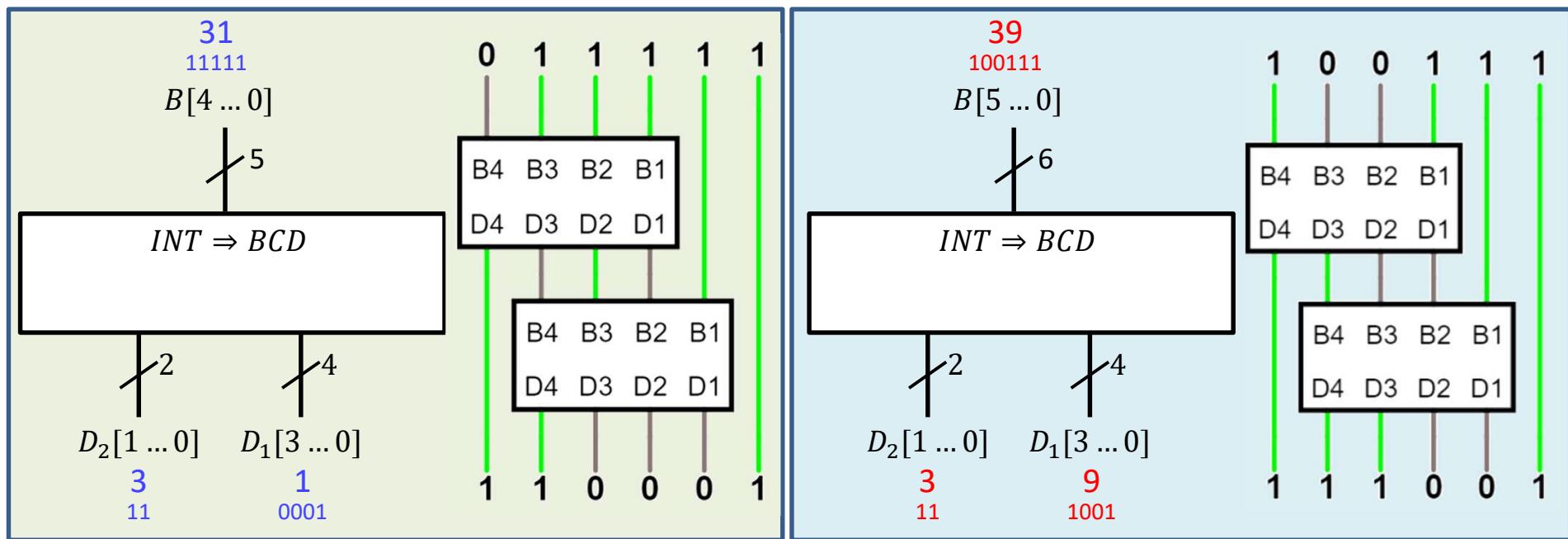


Exercício

✓ Resumo do método:

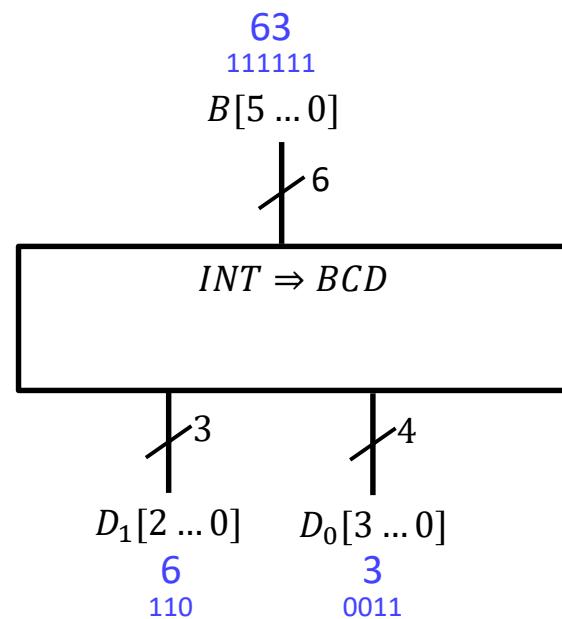
- O número B com n bits a ser convertido é transformado em um número real no formato: $0.M_B \times 2^n$;
- Este número será submetido a uma sequência de deslocamentos de bits a esquerda e de ajustes dos dígitos BCD ;
- A parte inteira deste número irá conter o valor de B transformado em BCD , enquanto a parte fracionária conterá o binário: $BCD.BIN$;
- Um dígito BCD , formado por 4 bits, precisará ser ajustado sempre que seu valor binário exceder a 4;
 - O ajuste é feito somando 4: $B_4B_3B_2B_1 > 0100 \Rightarrow D_4D_3D_2D_1 = B_4B_3B_2B_1 + 0011$
 - Este ajuste é feito pelo circuito construído para conversão de 5 bits;
- A lógica é encerrada quando todos os bits tiverem sido processados;

Exercício



Exercício

- Elaborar um circuito que converta um inteiro sem sinal de 6 bits para *BCD*:



Exercício

$$1 \ 1 \ 1 \ 1 \ 1 \ 1 = 63$$

BCD.INT

$$0.1 \ 1 \ 1 \ 1 \ 1 \ 1 \times 2^6 = 63$$

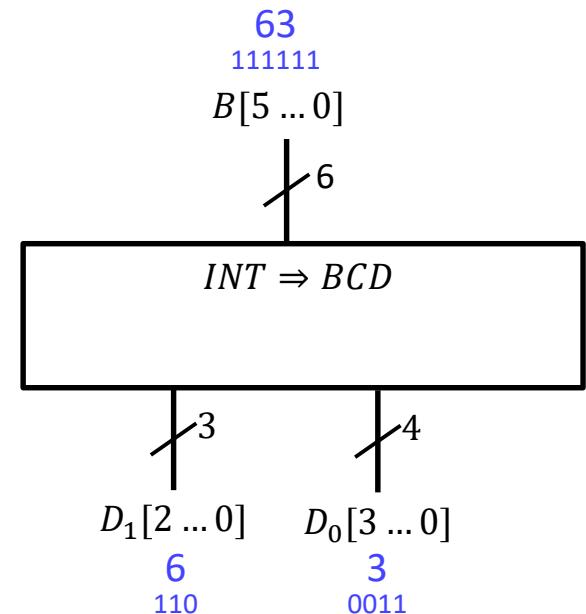
$$\boxed{1.} \ 1 \ 1 \ 1 \ 1 \ 1 \times 2^5 = 32 + 31 = 63$$

$$\boxed{1 \ 1.} \ 1 \ 1 \ 1 \ 1 \times 2^4 = 48 + 15 = 63$$

$$\boxed{1 \ 1 \ 1.} \ 1 \ 1 \ 1 \times 2^3 = 56 + 7 = 63$$

$$\boxed{1 \ 1 \ 1 \ 1.} \ 1 \ 1 \times 2^2 = ??$$

BCD inválido



Exercício

$$1 \ 1 \ 1 \ 1 \ 1 \ 1 = 63$$

BCD.INT

$$0.1 \ 1 \ 1 \ 1 \ 1 \ 1 \times 2^6 = 63$$

$$\boxed{1.} \ 1 \ 1 \ 1 \ 1 \ 1 \times 2^5 = 32 + 31 = 63$$

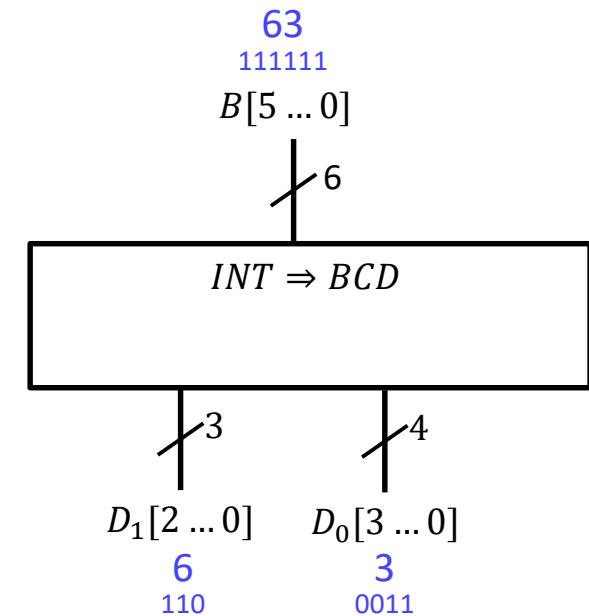
$$\boxed{1 \ 1.} \ 1 \ 1 \ 1 \ 1 \times 2^4 = 48 + 15 = 63$$

$$\boxed{1 \ 1 \ 1.} \ 1 \ 1 \ 1 \times 2^3 = 56 + 7 = 63$$

$$\begin{array}{r} \boxed{1 \ 0 \ 1 \ 0 \ 1.} \ 1 \ 1 \\ \hline 1 \ 0 \ 1 \ 0 \ 1 \ 1 \end{array} \times 2^2 = 60 + 3 = 63$$

$$\begin{array}{r} \boxed{1 \ 1 \ 0 \ 0 \ 0 \ 1.} \ 1 \\ \hline 1 \ 1 \ 0 \ 0 \ 0 \ 1 \ 1 \end{array} \times 2^1 = 62 + 1 = 63$$

$$\boxed{1 \ 1 \ 0 \ 0 \ 0 \ 1 \ 1} \times 2^0 = 63$$



Exercício

$$1 \ 1 \ 1 \ 1 \ 1 \ 1 = 63$$

BCD.INT

$$0.1 \ 1 \ 1 \ 1 \ 1 \ 1 \times 2^6 = 63$$

$$\boxed{1.} \ 1 \ 1 \ 1 \ 1 \ 1 \times 2^5 = 32 + 31 = 63$$

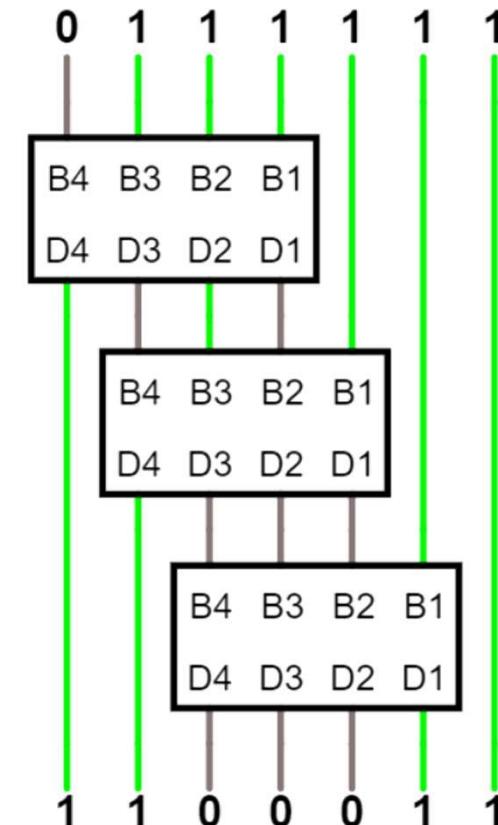
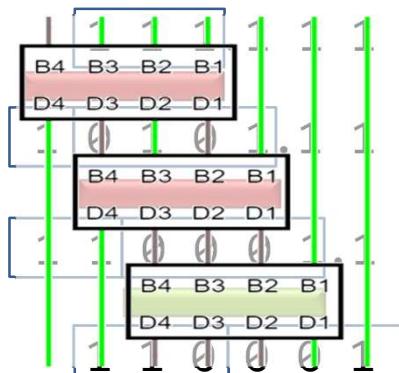
$$\boxed{1 \ 1.} \ 1 \ 1 \ 1 \ 1 \times 2^4 = 48 + 15 = 63$$

$$\times 2^3 = 56 + 7 = 63$$

$$\times 2^2 = 60 + 3 = 63$$

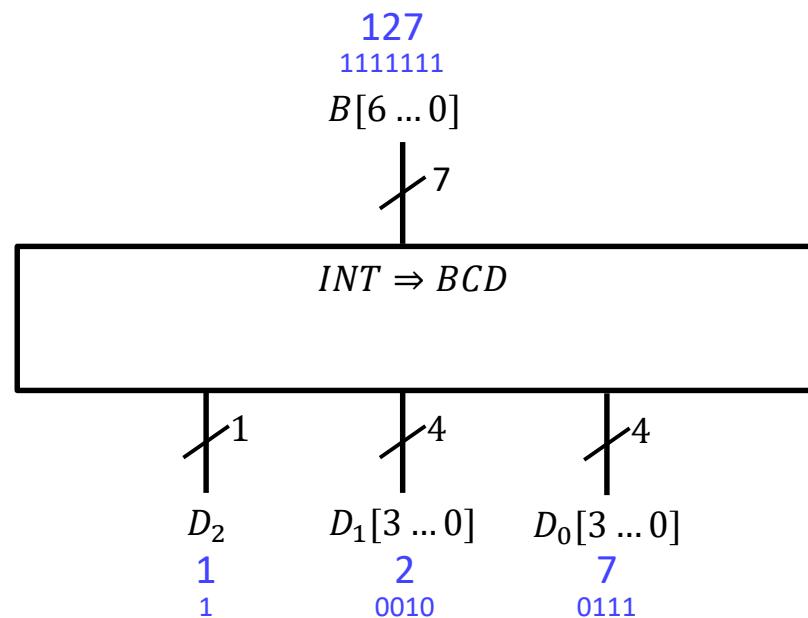
$$\times 2^1 = 62 + 1 = 63$$

$$1 \ 1 \ 0 \ 0 \ 0 \ 1 \times 2^0 = 63$$



Exercício

- Elaborar um circuito que converta um inteiro sem sinal de 7 bits para *BCD*:



Exercício

$$1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 = 127$$

BCD.INT

$$0.1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \times 2^7 = 127$$

$$\boxed{1.1} \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \times 2^6 = 64 + 63 = 127$$

$$\boxed{1 \ 1.1} \ 1 \ 1 \ 1 \ 1 \ 1 \times 2^5 = 96 + 31 = 127$$

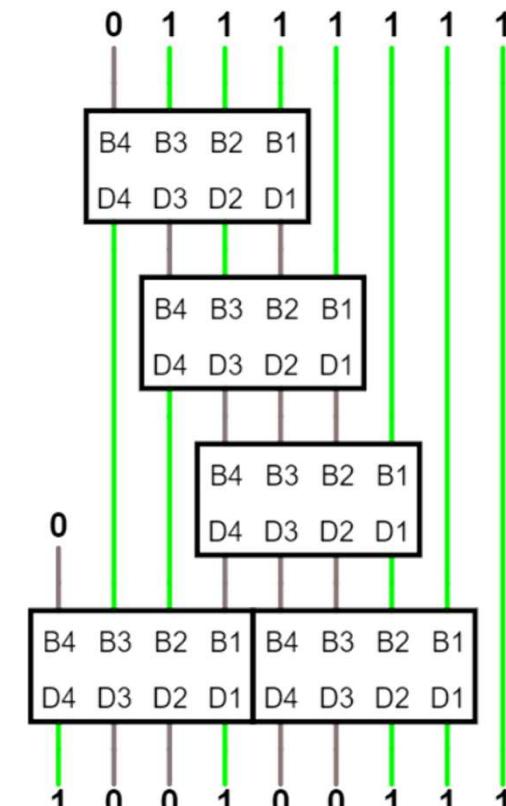
$$\boxed{1 \ 1 \ 1.1} \ 1 \ 1 \ 1 \ 1 \times 2^4 = 112 + 15 = 127$$

$$\begin{array}{r} \boxed{1 \ 0 \ 1 \ 0 \ 1.1} \\ \hline 1 \ 0 \ 1 \ 0 \ 1.1 \end{array} \ 1 \ 1 \times 2^3 = 120 + 7 = 127$$

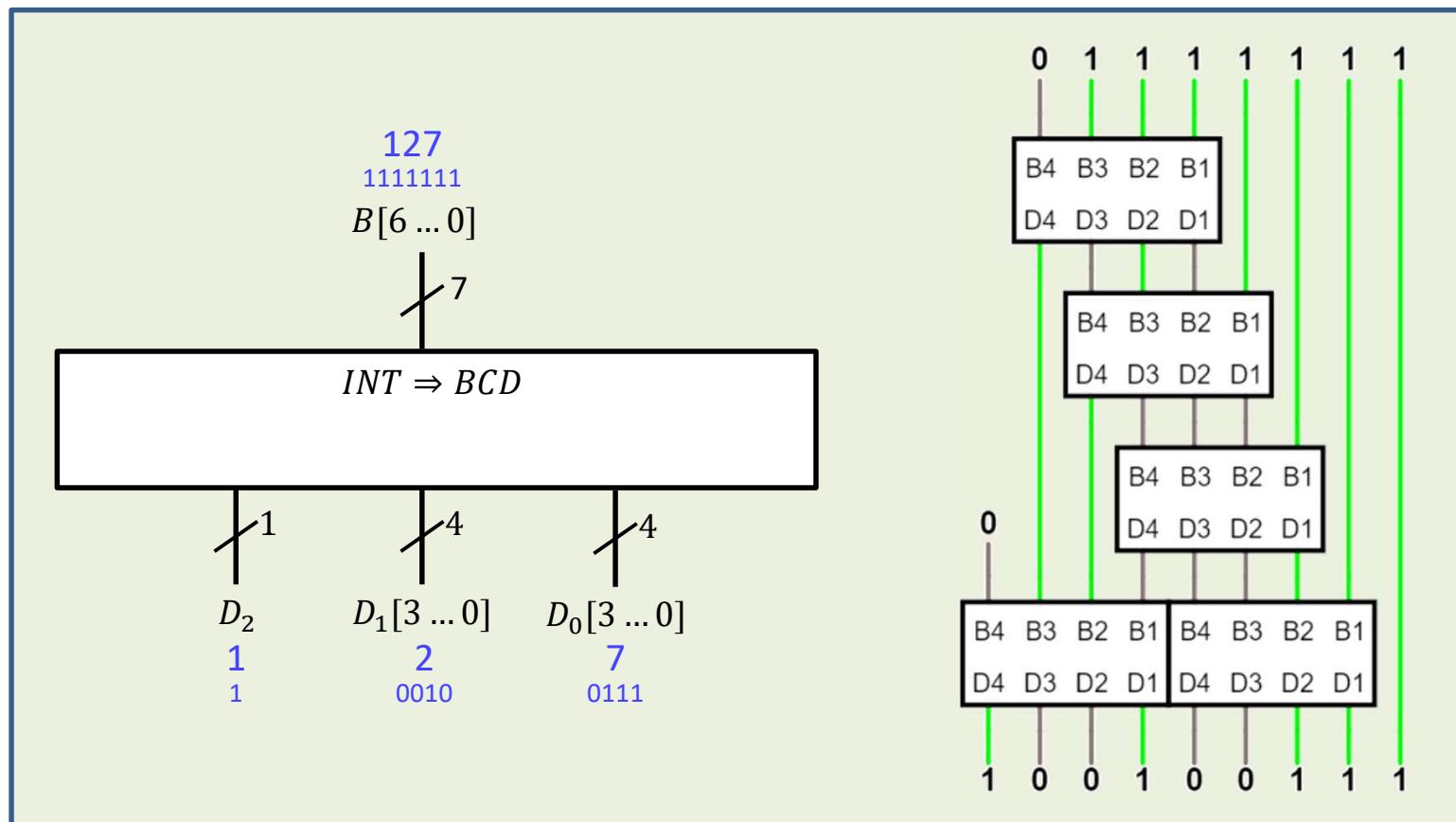
$$\begin{array}{r} \boxed{1 \ 1 \ 0 \ 0 \ 0 \ 1.1} \\ \hline 1 \ 1 \ 0 \ 0 \ 0 \ 1.1 \end{array} \ 1 \times 2^2 = 124 + 3 = 127$$

$$\begin{array}{r} \boxed{1 \ 1 \ 0 \ 0 \ 0 \ 1.1} \\ \hline 1 \ 1 \ 0 \ 0 \ 0 \ 1.1 \end{array} \ 1 \times 2^1 = 126 + 1 = 127$$

$$\begin{array}{r} \boxed{1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1.1} \\ \hline 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1.1 \end{array} \ 1 \times 2^0 = 127$$

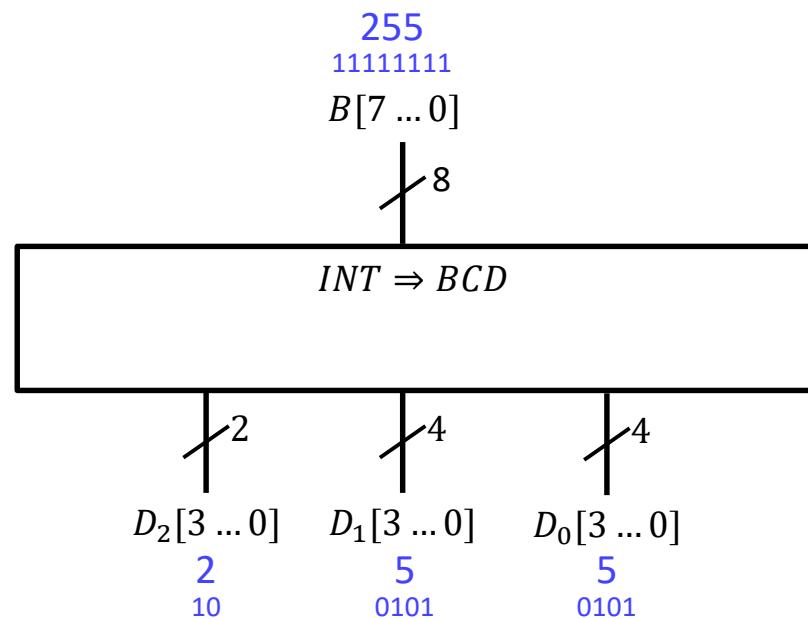


Exercício



Exercício

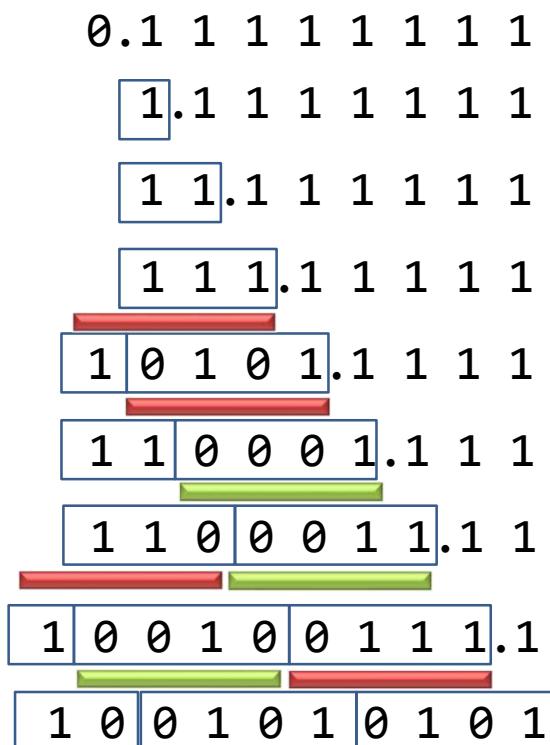
- Elaborar um circuito que converta um inteiro sem sinal de 8 bits para *BCD*:



Exercício

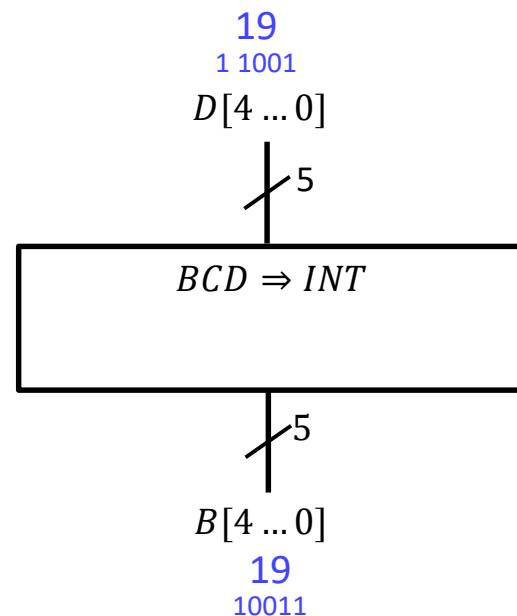
$$1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 = 255$$

BCD.INT



Exercício

- Elaborar um circuito que converte um número em BCD com 5 bits para binário:



Exercício

Como $B_0 = D_0$, então os demais bits de D é que definirão o padrão de conversão.

D_4	D_3	D_2	D_1	B_4	B_3	B_2	B_1
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0
0	1	0	1	X	X	X	X
0	1	1	0	X	X	X	X
0	1	1	1	X	X	X	X
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

$$D_4D_3D_2D_1 \leq 0100 \Rightarrow \\ B_4B_3B_2B_1 = D_4D_3D_2D_1$$

Entrada BCD inválida

$$D_4D_3D_2D_1 \geq 1000 \Rightarrow \\ B_4B_3B_2B_1 = D_4D_3D_2D_1 - 0011$$

Entrada BCD inválida

Exercício

	00	01	11	10
00		1	1	
01		X	X	X
11	1	X	X	X
10	1			1

$$X = \bar{A}D + A\bar{D} = A \oplus D$$

$$B_1 = D_1 \oplus D_4$$

	00	01	11	10
00				
01	1	X	X	X
11		X	X	X
10	1	1		1

$$X = \bar{A}B + A\bar{B}\bar{D} + A\bar{C}D$$

$$B_3 = \bar{D}_4D_3 + D_4\bar{D}_3\bar{D}_1 + D_4\bar{D}_2D_1 \quad B_4 = D_4D_3 + D_4D_2D_1$$

$$B_3 = \bar{D}_2(D_3 \oplus D_4) + \bar{D}_1D_2D_4$$

	00	01	11	10
00			1	1
01		X	X	X
11		X	X	X
10	1			1

$$X = \bar{A}C + C\bar{D} + A\bar{C}D$$

$$B_2 = \bar{D}_4D_2 + D_2\bar{D}_1 + D_4\bar{D}_2D_1$$

$$B_2 = D_4(D_1 \oplus D_2) + D_2\bar{D}_4$$

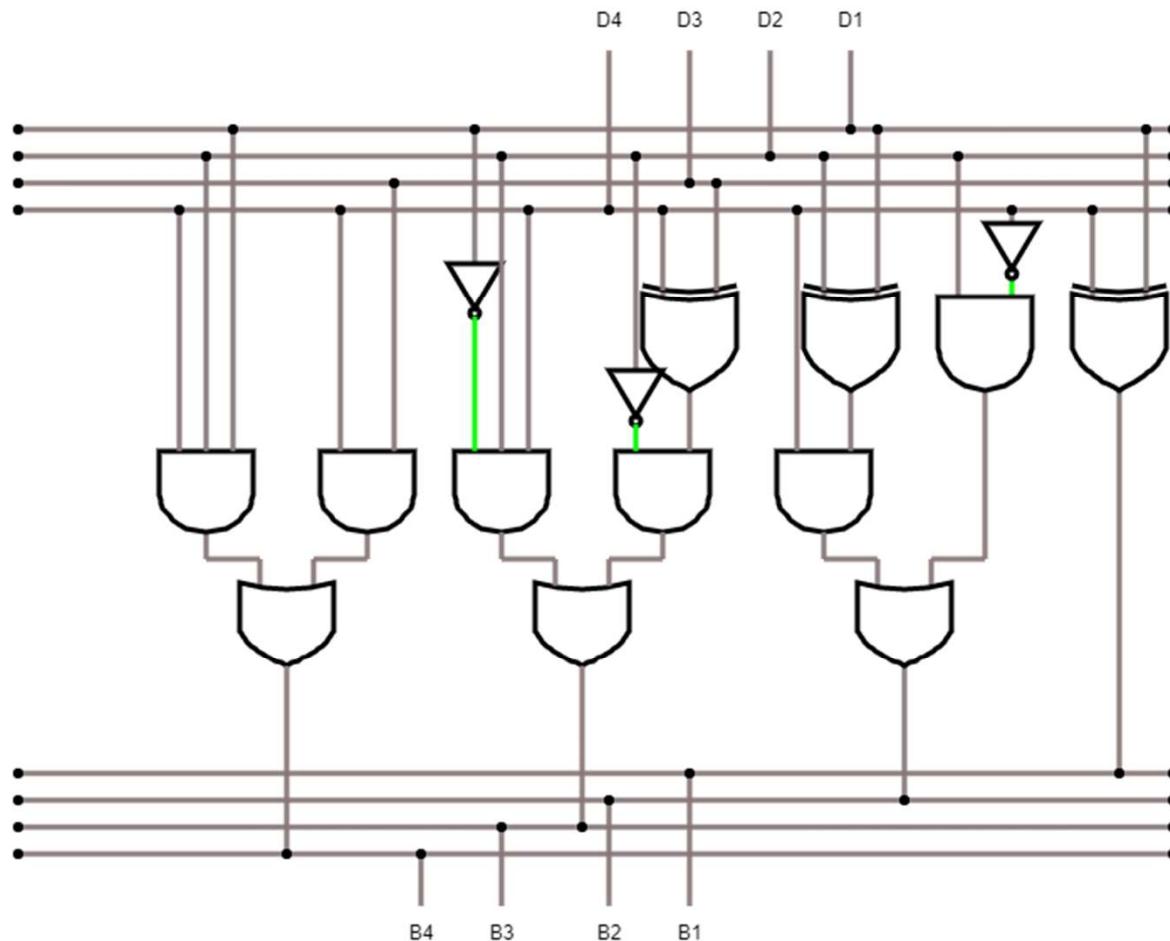
	00	01	11	10
00				
01		X	X	X
11	1	X	X	X
10			1	

$$X = AB + ACD$$

A	B	C	D
D ₄	D ₃	D ₂	D ₁
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

B ₄	B ₃	B ₂	B ₁
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
X	X	X	X
X	X	X	X
X	X	X	X
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
0	1	1	0
1	0	0	1
1	0	0	1
1	0	1	0
X	X	X	X
X	X	X	X
X	X	X	X

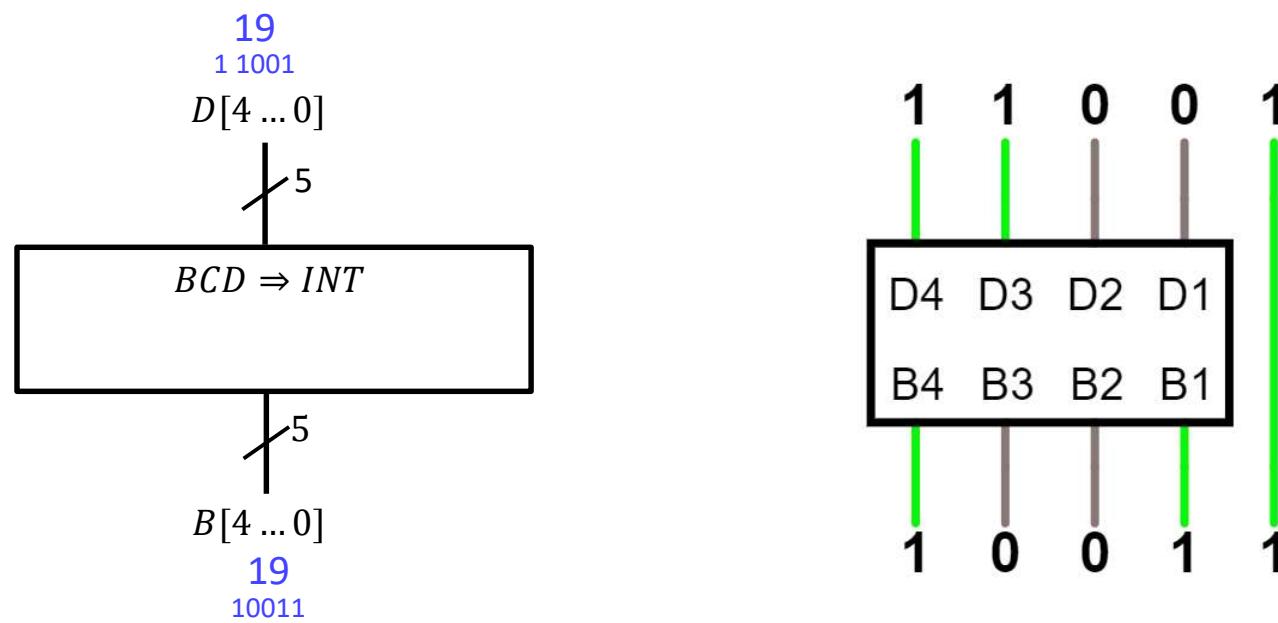
Exercício



1	1	0	0
D4	D3	D2	D1
B4	B3	B2	B1

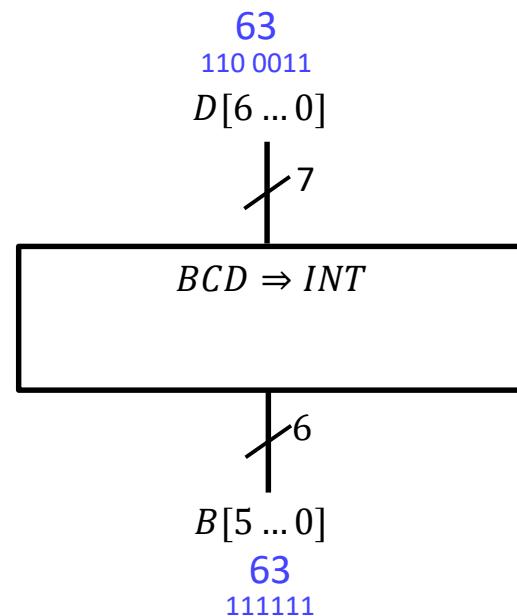
1	0	0	1
C4	C3	C2	C1

Exercício



Exercício

- Elaborar um circuito que converta um número em BCD para um binário de 6 bits:



Exercício

✓ Resumo do método:

- O número D em BCD a ser convertido é transformado em um número real no formato: $D.0 \times 2^0$;
- Este número será submetido a uma sequência de ajustes de dígitos BCD e deslocamentos de bits a direita;
- A parte fracionária deste número irá conter o valor de D transformado em *binário*, enquanto a parte inteira conterá o BCD : $BCD.BIN$;
- Antes de efetuar o deslocamento a direita, é verificado se os dígitos BCD , formados por 4 bits, precisarão ser ajustados;
 - O ajuste é feito subtraindo 3 sempre que seu valor exceder a 8: $D_4D_3D_2D_1 \geq 1000 \Rightarrow B_4B_3B_2B_1 = D_4D_3D_2D_1 - 0011$
 - Este ajuste é feito pelo circuito construído para conversão de 5 bits;
- A lógica é encerrada quando todos os bits tiverem sido processados, ou quando os bits remanescentes na parte BCD não precisarem mais de correção;

Exercício

$$1\ 1\ 0\ 0\ 0\ 1\ 1 = 63$$

BCD.INT

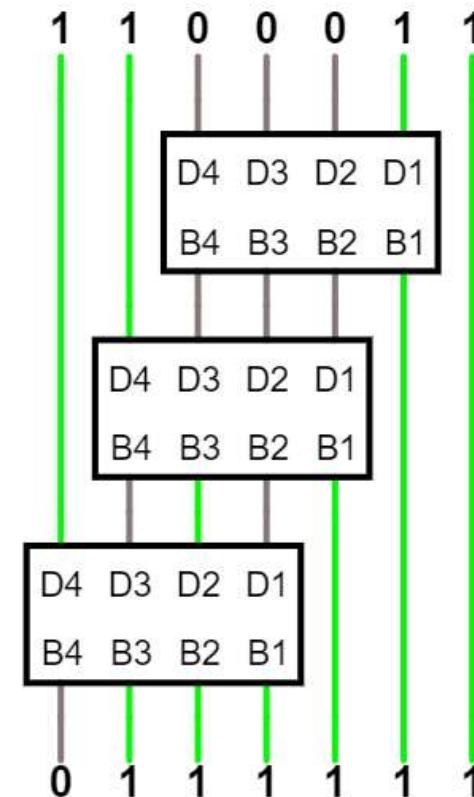
$$1\ 1\ 0\ 0\ 0\ 1\ 1 \times 2^0 = 63$$

$$1\ 1\ 0\ 0\ 0\ 1\ 1 \times 2^1 = 62 + 1 = 63$$

$$1\ 0\ 1\ 0\ 1\ 1\ 1 \times 2^2 = 60 + 3 = 63$$

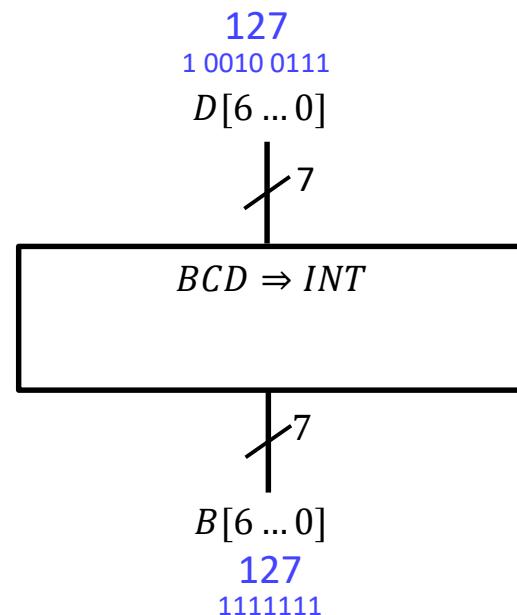
$$1\ 1\ 1\ 1\ 1\ 1\ 1 \times 2^3 = 56 + 7 = 63$$

$$D_4 D_3 D_2 D_1 \geq 1000 \Rightarrow \\ B_4 B_3 B_2 B_1 = D_4 D_3 D_2 D_1 - 0011$$



Exercício

- Elaborar um circuito que converta um número em BCD para um binário de 7 bits:



Exercício

1 0 0 1 0 0 1 1 1 = 127

BCD.INT

1 0 0 1 | 0 0 1 1

1 1 | 0 0 0 1

1 1 | 0 0 0 0

1 0 1 0 | 1.1 1 1

0 1 1 1 | 1.1 1 1 1

$$1. \times 2^0 = 127$$

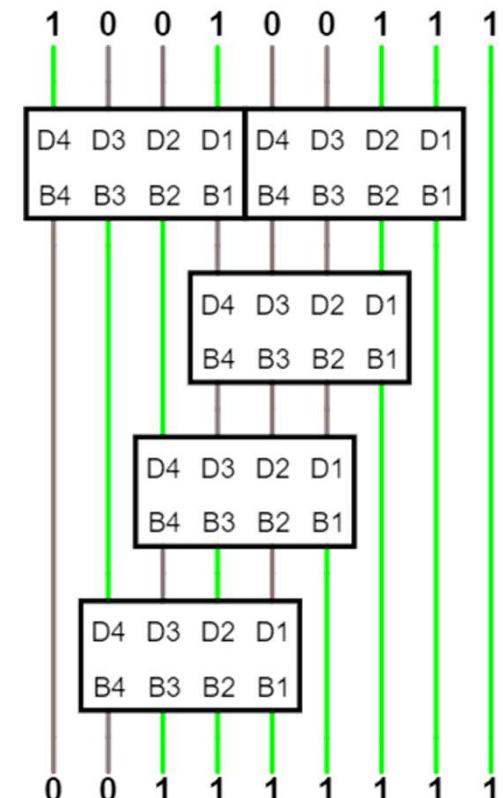
$$1.1 \times 2^1 = 126 + 1 = 127$$

$$1.1 1 \times 2^2 = 124 + 3 = 127$$

$$1.1 1 1 \times 2^3 = 120 + 7 = 127$$

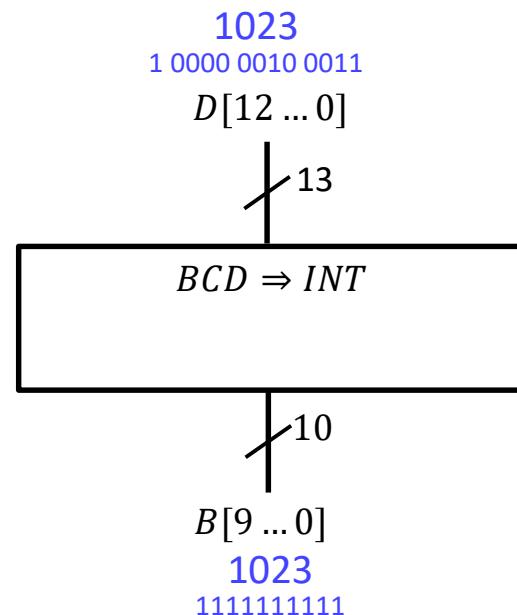
$$1.1 1 1 1 \times 2^4 = 112 + 15 = 127$$

$$\begin{aligned} D_4 D_3 D_2 D_1 &\geq 1000 \Rightarrow \\ B_4 B_3 B_2 B_1 &= D_4 D_3 D_2 D_1 - 0011 \end{aligned}$$



Exercício

- Elaborar um circuito que converta um número em *BCD* para um binário de 10 bits:



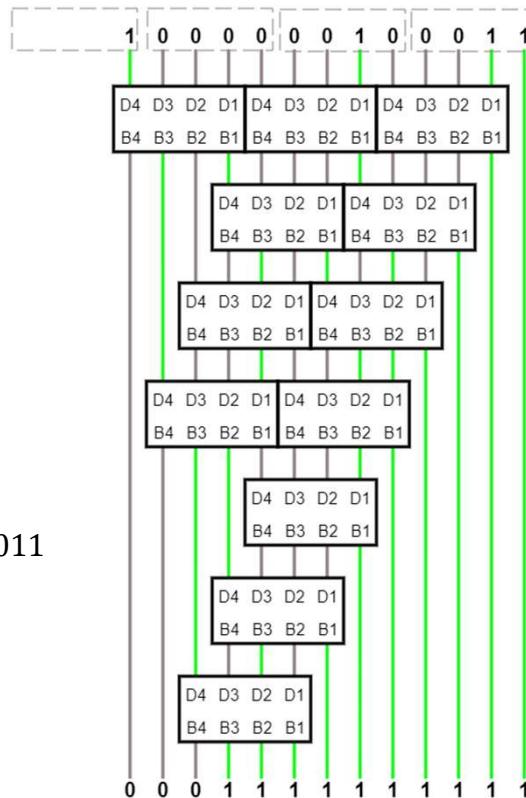
Exercício

1 0 0 0 0 0 0 1 0 0 0 1 1 = 1023

BCD.INT

1	0	0	0	0	0	0	1	0	0	0	1	1.
1	0	1	0	0	0	1	0	0	0	1	1	
1	0	0	1	0	1	0	1	0	1	1	1	
1	0	0	1	0	0	1	1	1	1	1	1	
1	1	0	0	0	1	1	.1	1	1	1	1	
1	1	0	0	0	1	.1	1	1	1	1	1	
1	0	1	0	1	.1	1	1	1	1	1	1	
1	1	1	1	1	.1	1	1	1	1	1	1	

$$D_4 D_3 D_2 D_1 \geq 1000 \Rightarrow \\ B_4 B_3 B_2 B_1 = D_4 D_3 D_2 D_1 - 0011$$

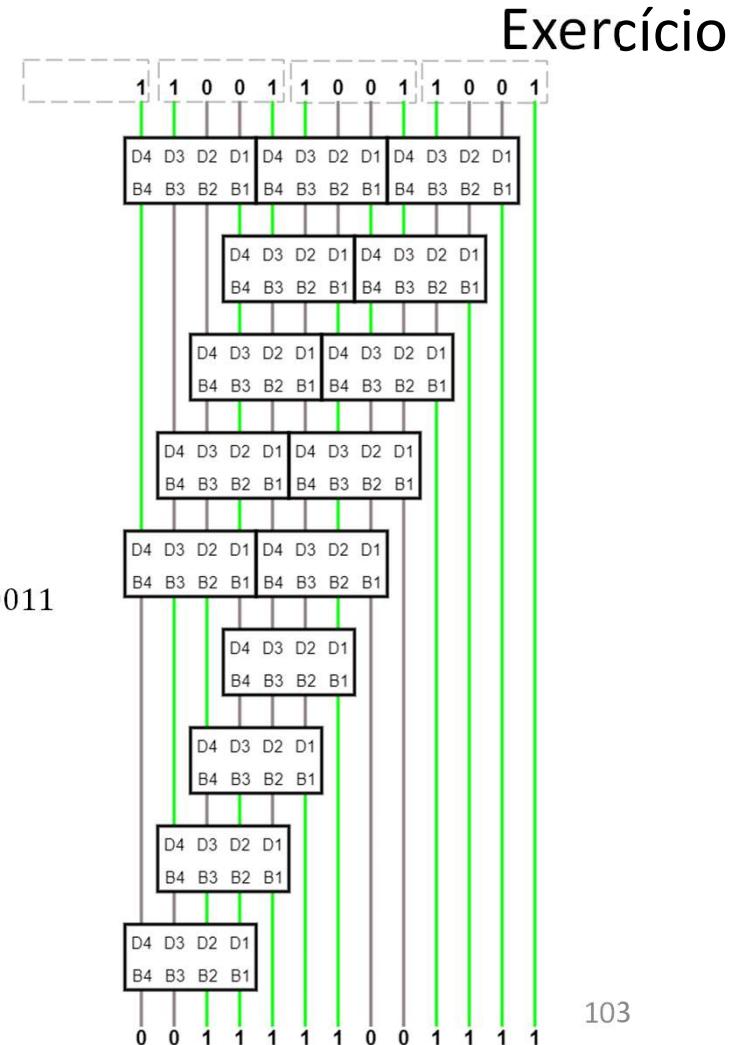


1 1 0 0 1 1 0 0 1 1 0 0 1 = 1999

1	1	0	0	1	1	0	0	1	1	0	0	1
1	0	0	1	1	0	0	1	1	0	0	1	.1
1	0	0	1	0	0	1	1	0	0	1	.1	1
1	0	0	1	0	0	1	0	0	1	.1	1	1
1	0	0	1	0	0	1	0	0	1	.1	1	1
1	0	0	1	0	0	1	0	0	0	1	1	1
0	1	1	0	0	0	1	0	0	1	1	1	1
0	1	1	0	0	0	1	0	0	1	1	1	1
0	1	0	1	0	1	1	0	0	1	1	1	1
0	0	1	1	1	1	.1	1	0	0	1	1	1
0	0	1	1	1	1	.1	1	0	0	1	1	1

$$B_4 B_3 B_2 B_1 = D_4 D_3 D_2 D_1 - 0011$$

= 0x7CF = 1999

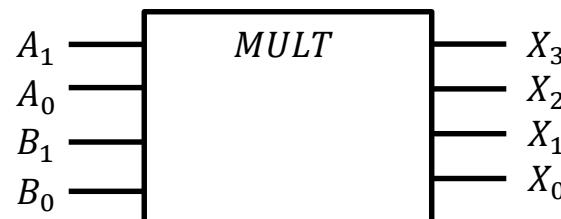


Sumário

- 1. Revisão – Sistemas de Numeração
- 2. Revisão – Representação de Dados
- 3. Revisão – Operações com Binários
- 4. Álgebra Booleana
- 5. Simplificação de Expressões
- 6. Mapa de Karnaugh
- 7. Elementos Lógicos Universais
- 8. Circuitos Combinacionais
- 9. Circuitos Sequenciais
- 1. Somador / Subtrator
- 2. Comparadores
- 3. Codificador/decodificador
- 4. Multiplexador/Demux
- 5. Geradores de paridade
- 6. Circuitos Específicos
- 7. Multiplicadores / Divisores
- 8. ULA
- 9. PLD/PLA/ROM

Multiplicação

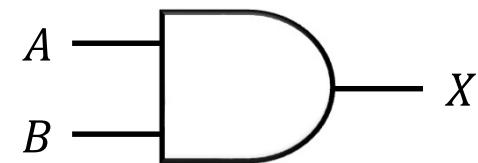
- As primeiras *ULAs* não incluíam o operação de multiplicação;
- Os programas realizavam esta operação mediante o uso de emulação por software;
- A multiplicação de dois inteiros sem sinal de n bits produz um resultado de $2 \times n$ bits;
- Por exemplo, a multiplicação de dois números inteiros sem sinal de 2 bits produziria um resultado de 4 bits:



Multiplicação

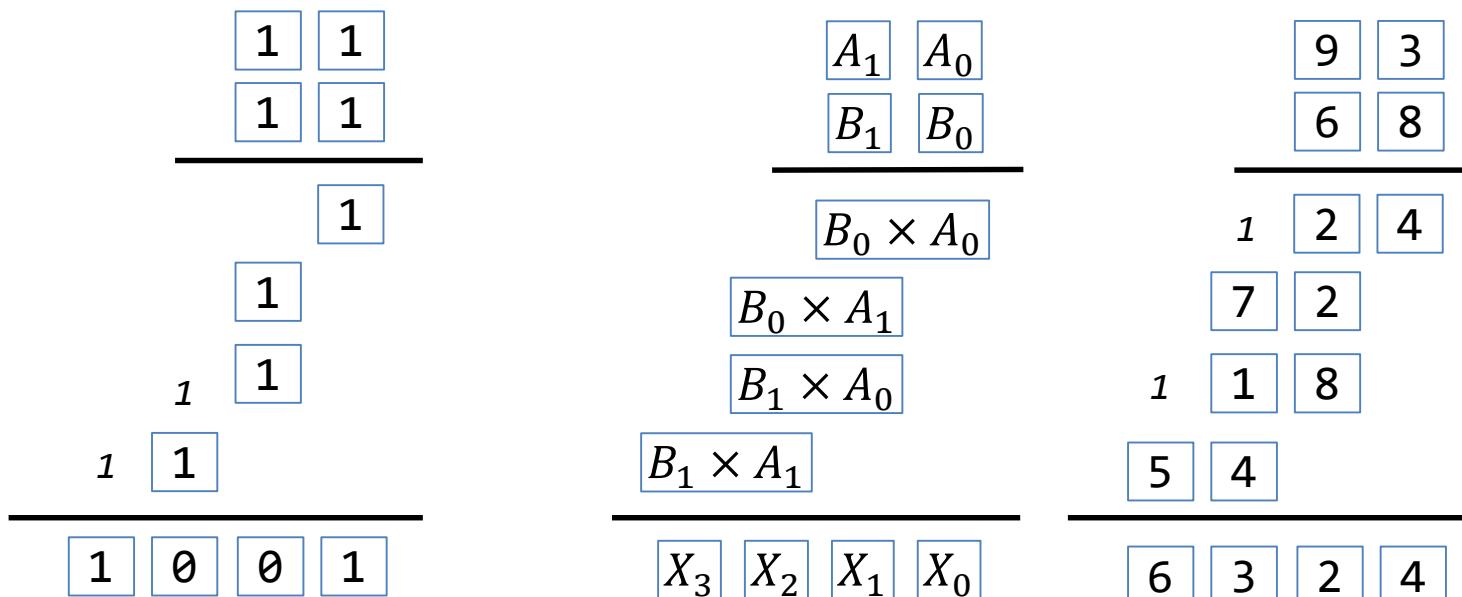
- A multiplicação de dois números inteiros sem sinal de 2 bits poderia ser feita partindo-se da multiplicação de inteiros de 1 bit;
- Multiplicação de inteiros de 1 bit:

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

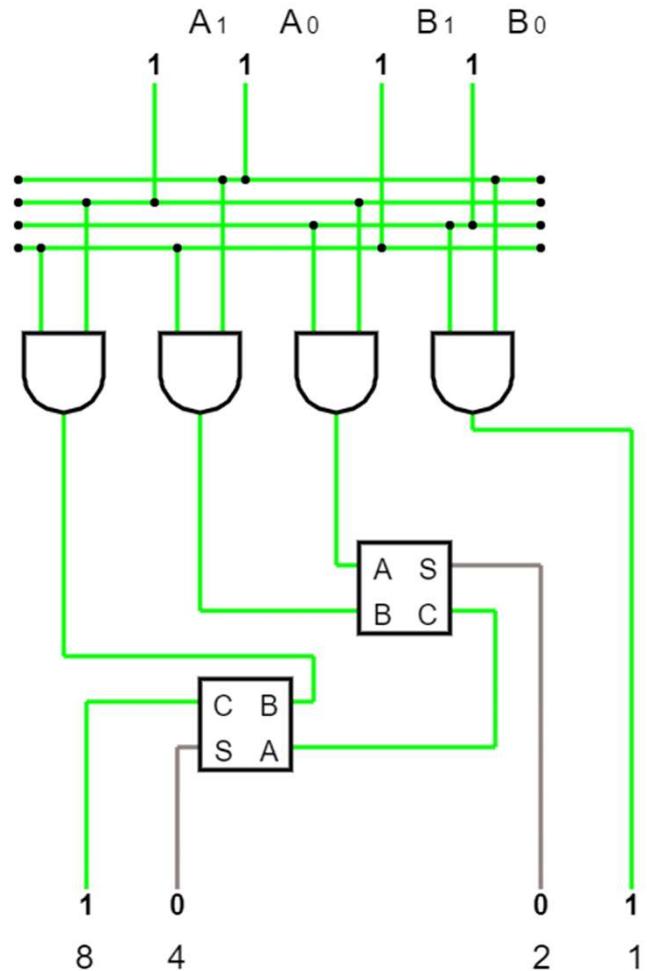


Multiplicação

- Utilizar o circuito de multiplicação de 1 bits;



Multiplicação



$$\begin{array}{r}
 \begin{matrix} 1 & 1 \\ 1 & 1 \end{matrix} \\
 \hline
 1
 \end{array}
 \quad
 \begin{array}{r}
 \begin{matrix} A_1 & A_0 \\ B_1 & B_0 \end{matrix} \\
 \hline
 B_0 \times A_0
 \end{array}
 \\
 \begin{array}{r}
 \begin{matrix} 1 \\ 1 \end{matrix} \\
 \hline
 B_0 \times A_1
 \end{array}
 \quad
 \begin{array}{r}
 \begin{matrix} B_1 \times A_0 \\ B_1 \times A_1 \end{matrix} \\
 \hline
 B_1 \times A_1
 \end{array}
 \\
 \begin{array}{r}
 \begin{matrix} 1 & 0 & 0 & 1 \\ X_3 & X_2 & X_1 & X_0 \end{matrix}
 \end{array}$$

Multiplicação

- Alternativa: Construir o circuito de multiplicação de 2 bits a partir da tabela verdade:

A_1	A_0	B_1	B_0		X_3	X_2	X_1	X_0
0	0	0	0		0	0	0	0
0	0	0	1		0	0	0	0
0	0	1	0		0	0	0	0
0	0	1	1		0	0	0	0
0	1	0	0		0	0	0	0
0	1	0	1		0	0	0	1
0	1	1	0		0	0	1	0
0	1	1	1		0	0	1	1
1	0	0	0		0	0	0	0
1	0	0	1		0	0	1	0
1	0	1	0		0	1	0	0
1	0	1	1		0	1	1	0
1	1	0	0		0	0	0	0
1	1	0	1		0	0	1	1
1	1	1	0		0	1	1	0
1	1	1	1		1	0	0	1

$$X_0 = A_0 B_0$$

$$X_1 = A_0 B_1 \oplus A_1 B_0$$

$$X_2 = A_1 B_1 \overline{A_0 B_0}$$

$$X_3 = A_1 B_1 A_0 B_0$$

Multiplicação

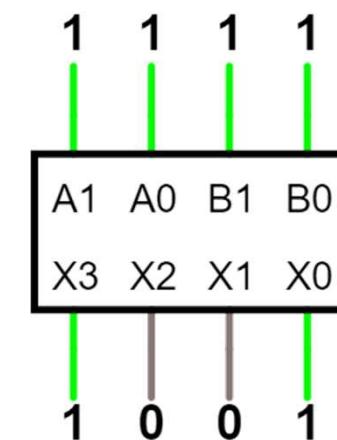
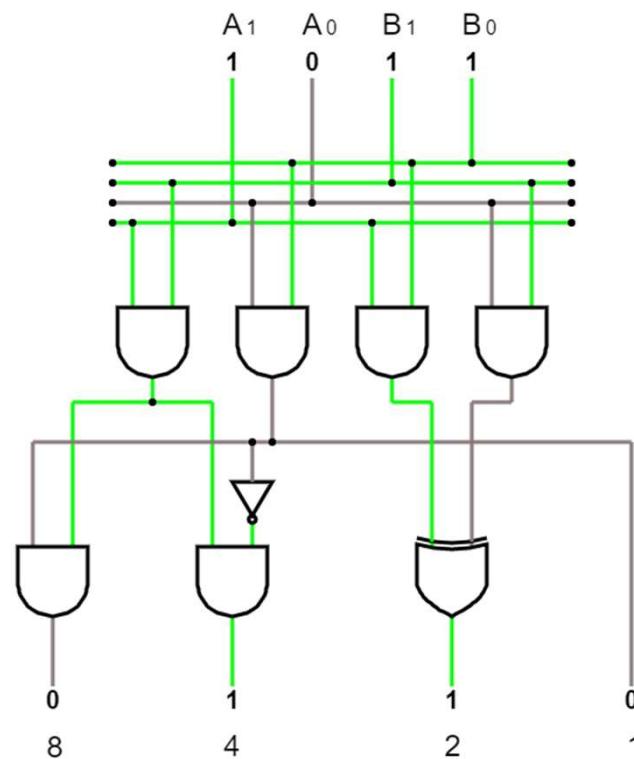
- Circuito lógico para a multiplicação de dois números inteiros sem sinal de 2 bits:

$$X_0 = A_0 B_0$$

$$X_1 = A_0 B_1 \oplus A_1 B_0$$

$$X_2 = A_1 B_1 \overline{A_0 B_0}$$

$$X_3 = A_1 B_1 A_0 B_0$$

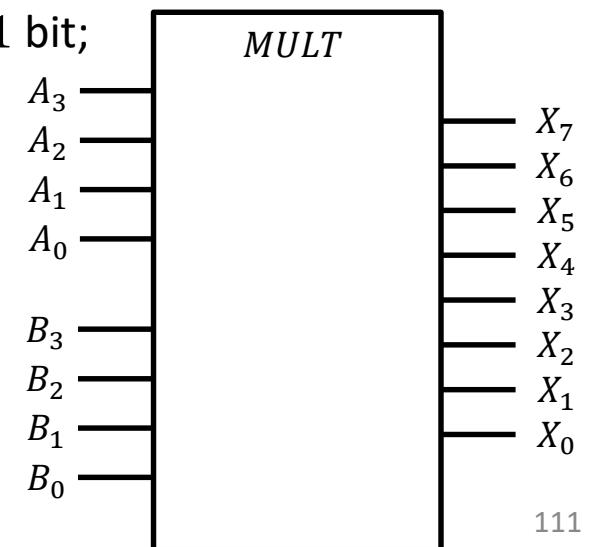


110

Multiplicação

- Como implementar a multiplicação de 4 bits?
- Diversos são os caminhos para a obtenção deste circuito;

- Utilizar o circuito de multiplicação de 2 bits;
- Utilizar o algoritmo baseado em somas e deslocamentos;
- Utilizar uma matriz de somador estendido de 1 bit;
- E muitas outras técnicas;

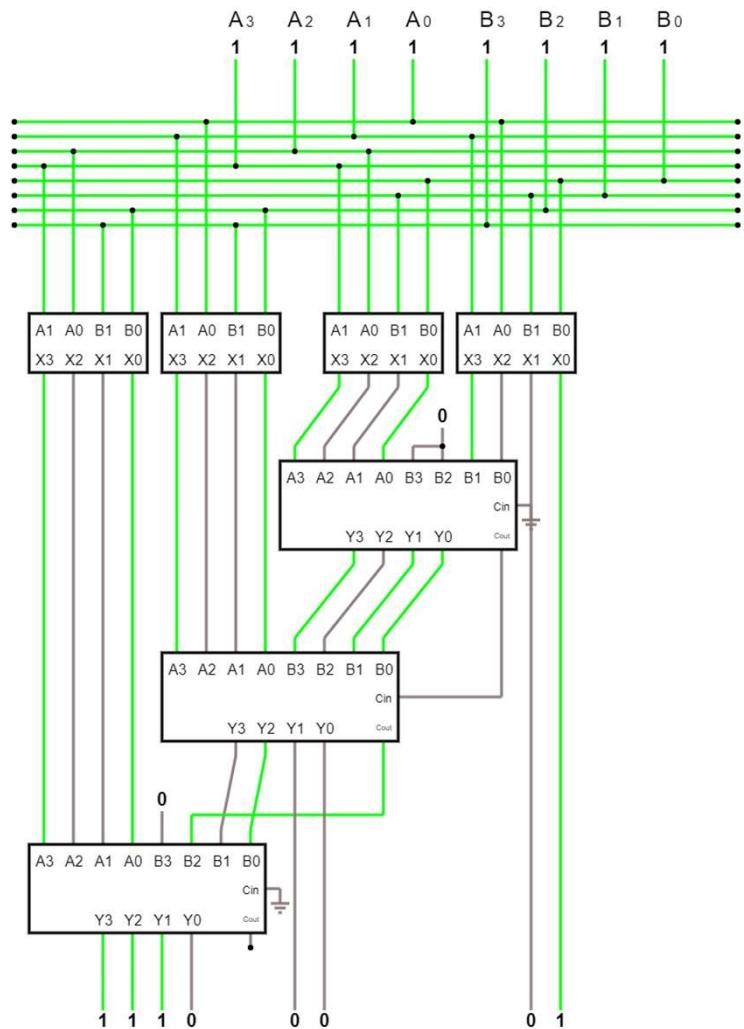


Multiplicação

- Utilizar o circuito de multiplicação de 2 bits;

$$\begin{array}{r} A_3A_2 \quad A_1A_0 \\ B_3B_2 \quad B_1B_0 \\ \hline B_1B_0 \times A_1A_0 \\ B_1B_0 \times A_3A_2 \\ B_3B_2 \times A_1A_0 \\ \hline B_3B_2 \times A_3A_2 \\ \hline X_7X_6 \quad X_5X_4 \quad X_3X_2 \quad X_1X_0 \end{array}$$
$$\begin{array}{r} 9 \quad 3 \\ 6 \quad 8 \\ \hline 1 \quad 2 \quad 4 \\ 7 \quad 2 \\ 1 \quad 1 \quad 8 \\ \hline 5 \quad 4 \\ \hline 6 \quad 3 \quad 2 \quad 4 \end{array}$$

Multiplicação



$$\begin{array}{|c|c|} \hline A_3 & A_2 \\ \hline \end{array} \quad \begin{array}{|c|c|} \hline A_1 & A_0 \\ \hline \end{array}$$

$$\begin{array}{|c|c|} \hline B_3 & B_2 \\ \hline \end{array} \quad \begin{array}{|c|c|} \hline B_1 & B_0 \\ \hline \end{array}$$

$$B_1B_0 \times A_1A_0$$

$$B_1B_0 \times A_3A_2$$

$$B_3B_2 \times A_1A_0$$

$$B_3B_2 \times A_3A_2$$

$$X_7X_6 \quad X_5X_4 \quad X_3X_2 \quad X_1X_0$$

Multiplicação

- Utilizar o algoritmo baseado em somas e deslocamentos;

$$\begin{array}{r} & A_3 & A_2 & A_1 & A_0 \\ & B_3 & B_2 & B_1 & B_0 \\ \hline & A_3B_0 & A_2B_0 & A_1B_0 & A_0B_0 \\ & A_3B_1 & A_2B_1 & A_1B_1 & A_0B_1 \\ & A_3B_2 & A_2B_2 & A_1B_2 & A_0B_2 \\ & A_3B_3 & A_2B_3 & A_1B_3 & A_0B_3 \\ \hline X_7 & X_6 & X_5 & X_4 & X_3 & X_2 & X_1 & X_0 \end{array}$$

AB_0
 AB_1
 AB_2
 AB_3

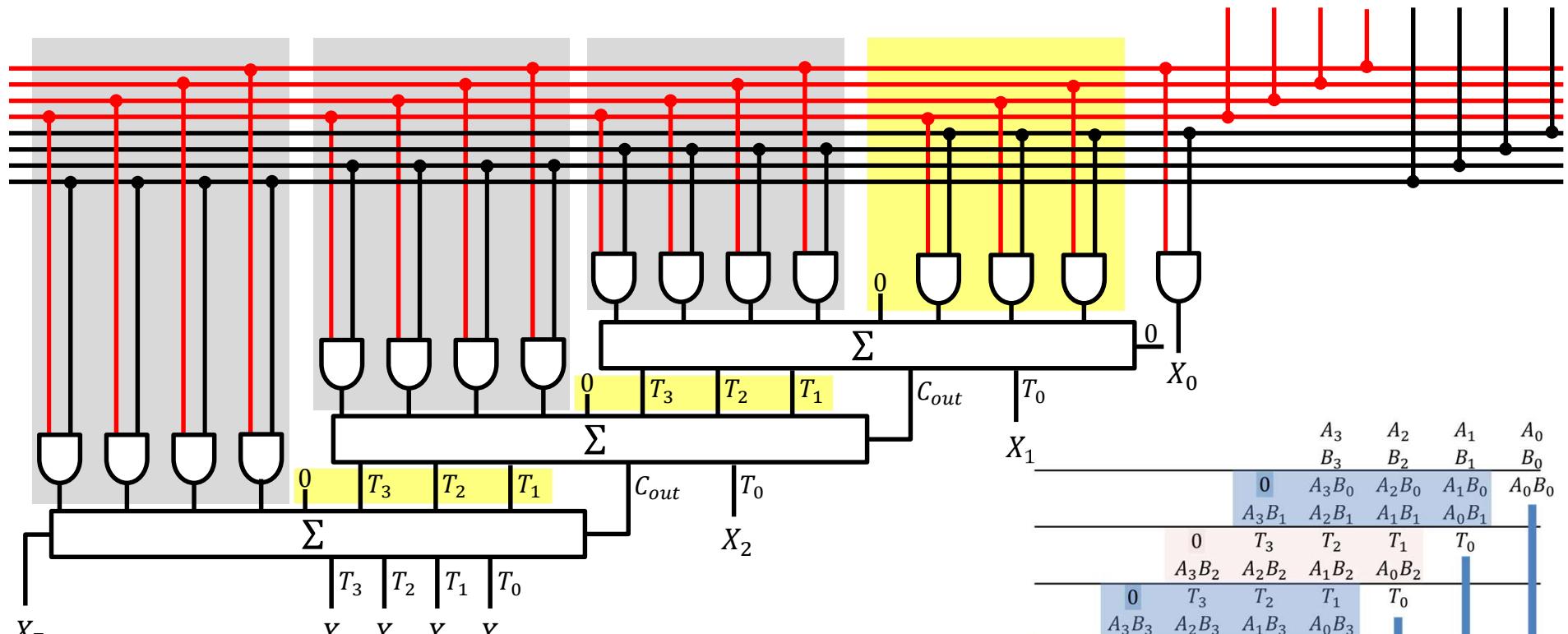
Vamos desdobrar a multiplicação em somas parciais de dois binários.

Multiplicação

	A_3	A_2	A_1	A_0	
	B_3	B_2	B_1	B_0	
	0	A_3B_0	A_2B_0	A_1B_0	A_0B_0
	A_3B_1	A_2B_1	A_1B_1	A_0B_1	
	0	T_3	T_2	T_1	T_0
	A_3B_2	A_2B_2	A_1B_2	A_0B_2	
	0	T_3	T_2	T_1	T_0
	A_3B_3	A_2B_3	A_1B_3	A_0B_3	
C_{out}	T_3	T_2	T_1	T_0	
X_7	X_6	X_5	X_4	X_3	
					X_2
					X_1
					X_0

Multiplicação

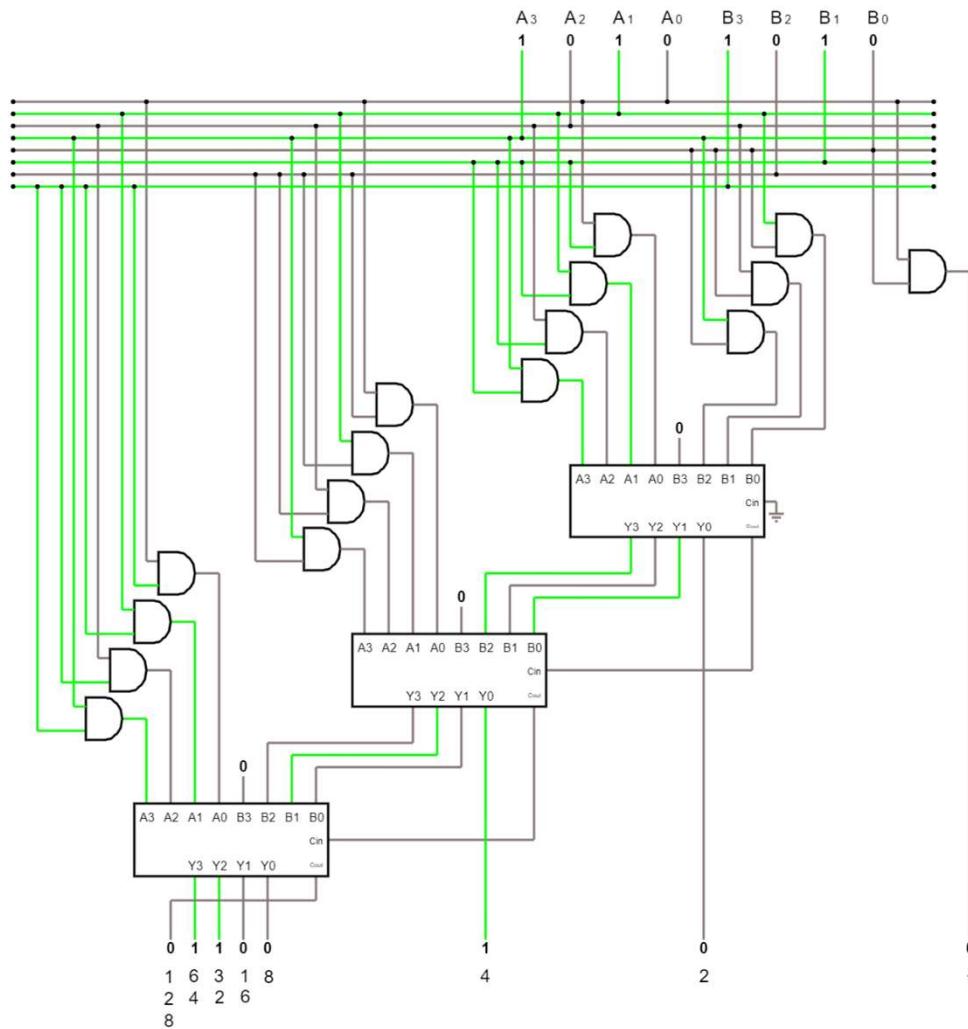
$A_3 \ A_2 \ A_1 \ A_0 \ B_3 \ B_2 \ B_1 \ B_0$



A_3	A_2	A_1	A_0
B_3	B_2	B_1	B_0
0	A_3B_0	A_2B_0	A_1B_0
A_3B_1	A_2B_1	A_1B_1	A_0B_1
0	T_3	T_2	T_1
A_3B_2	A_2B_2	A_1B_2	A_0B_2
0	T_3	T_2	T_1
A_3B_3	A_2B_3	A_1B_3	A_0B_3
C_{out}	T_3	T_2	T_1
X_7	X_6	X_5	X_4
			X_3
			X_2
			X_1
			X_0

116

Multiplicação



A_3	A_2	A_1	A_0
B_3	B_2	B_1	B_0
0	A_3B_0	A_2B_0	A_1B_0
A_3B_1	A_2B_1	A_1B_1	A_0B_1
0	T_3	T_2	T_1
A_3B_2	A_2B_2	A_1B_2	A_0B_2
0	T_3	T_2	T_1
A_3B_3	A_2B_3	A_1B_3	A_0B_3
C_{out}	T_3	T_2	T_1
X_7	X_6	X_5	X_4
			X_3
			X_2
			X_1
			X_0

117

Multiplicação

- Utilizar uma matriz de somador estendido de 1 bit:
- Na sequência, uma análise dos detalhes do processo de multiplicação:

Multiplicação

$$\begin{array}{cccc}
 & A_3 & A_2 & A_1 & A_0 \\
 & B_3 & B_2 & B_1 & B_0 \\
 \hline
 A_3B_0 & A_2B_0 & A_1B_0 & A_0B_0 \\
 A_3B_1 & A_2B_1 & A_1B_1 & A_0B_1 \\
 A_3B_2 & A_2B_2 & A_1B_2 & A_0B_2 \\
 \hline
 A_3B_3 & A_2B_3 & A_1B_3 & A_0B_3 \\
 \hline
 X_7 & X_6 & X_5 & X_4 & X_3 & X_2 & X_1 & X_0
 \end{array}$$

AB_0
 AB_1
 AB_2
 AB_3

Vamos transformar em uma sucessão de somas:

$$\begin{array}{cccc}
 & 0 & 0 & 0 & 0 \\
 & A_3B_0 & A_2B_0 & A_1B_0 & A_0B_0 \\
 A_3B_1 & A_2B_1 & A_1B_1 & A_0B_1 \\
 A_3B_2 & A_2B_2 & A_1B_2 & A_0B_2 \\
 \hline
 A_3B_3 & A_2B_3 & A_1B_3 & A_0B_3 \\
 \hline
 X_7 & X_6 & X_5 & X_4 & X_3 & X_2 & X_1 & X_0
 \end{array}$$

0
 AB_0
 AB_1
 AB_2
 AB_3

Multiplicação

$$B_0 = \begin{cases} 0 \Rightarrow \\ 1 \Rightarrow \end{cases}$$

	0	0	0	0
	0	0	0	0
	A_3	A_2	A_1	A_0
C	T_3	T_2	T_1	T_0

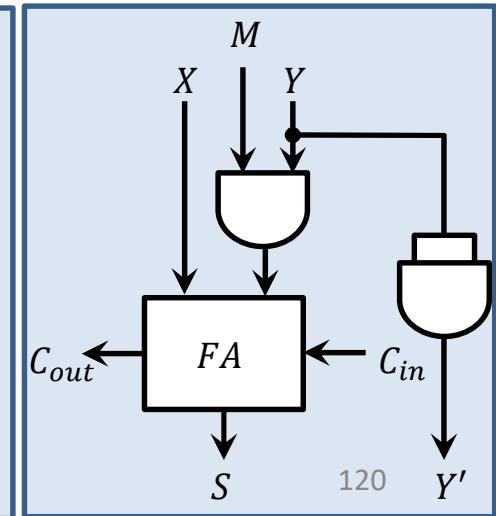
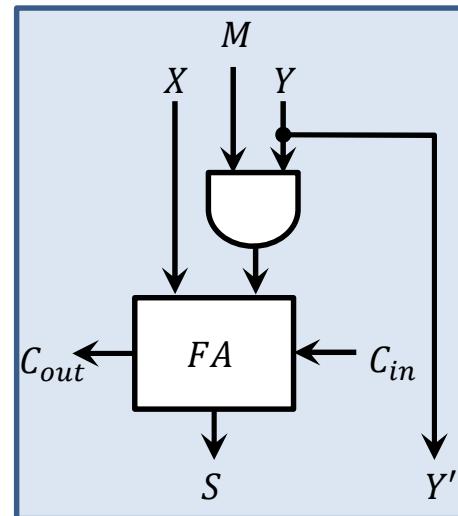
$$B_1 = \begin{cases} 0 \Rightarrow & 0 & 0 & 0 & 0 \\ 1 \Rightarrow & A_3 & A_2 & A_1 & A_0 \end{cases}$$

$$B_2 = \left\{ \begin{array}{l} 0 \Rightarrow \begin{array}{cccc} 0 & 0 & 0 & 0 \\ A_3 & A_2 & A_1 & A_0 \end{array} \\ 1 \Rightarrow \begin{array}{ccccc} C & T_3 & T_2 & T_1 & T_0 \end{array} \end{array} \right.$$

$$B_3 = \begin{cases} 0 \Rightarrow & 0 & 0 & 0 & 0 \\ 1 \Rightarrow & A_3 & A_2 & A_1 & A_0 \end{cases}$$

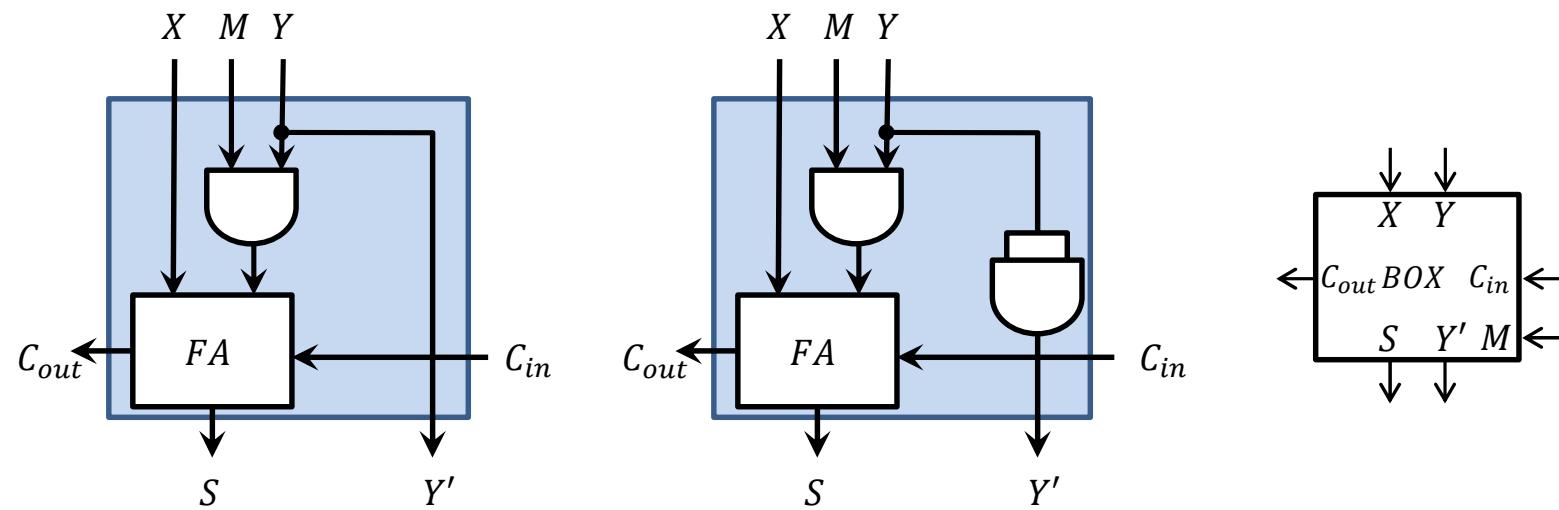
C	T_3	T_2	T_1	T_0			
\downarrow							
X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0

			0	0	0	0	0	0
			A_3B_0	A_2B_0	A_1B_0	A_0B_0		0
			A_3B_1	A_2B_1	A_1B_1	A_0B_1		AB_0
			A_3B_2	A_2B_2	A_1B_2	A_0B_2		AB_1
			A_3B_3	A_2B_3	A_1B_3	A_0B_3		AB_2
								AB_3
X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0	



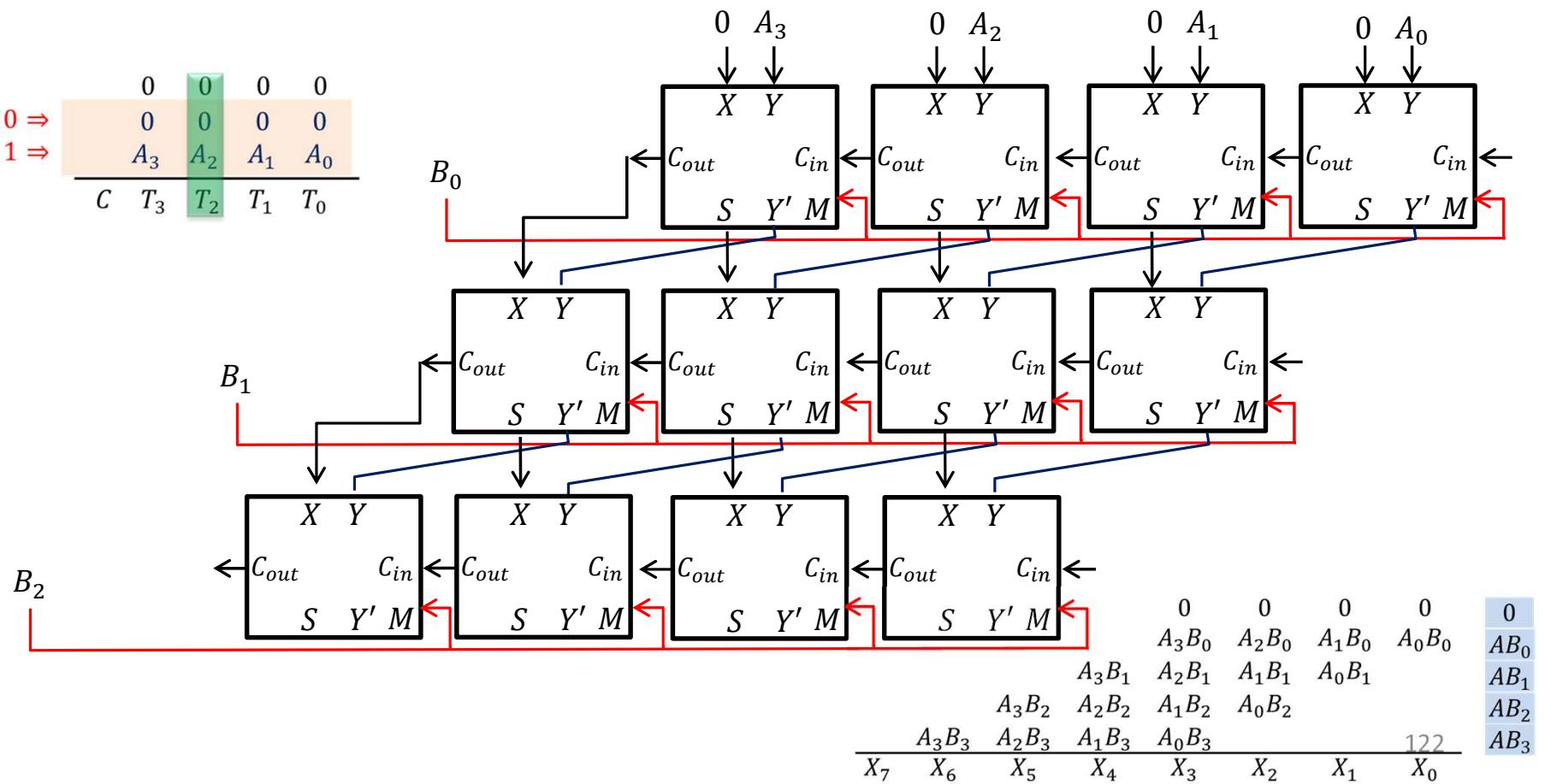
Multiplicação

- ✓ A proposta inclui a criação de um bloco funcional que combina um Somador Completo de 1 bit e um *AND*;
 - A entrada M da porta *AND* selecionará o segundo operando do Somador: entre 0 e Y ;
 - Adicionalmente, a *BOX* replica Y na saída;

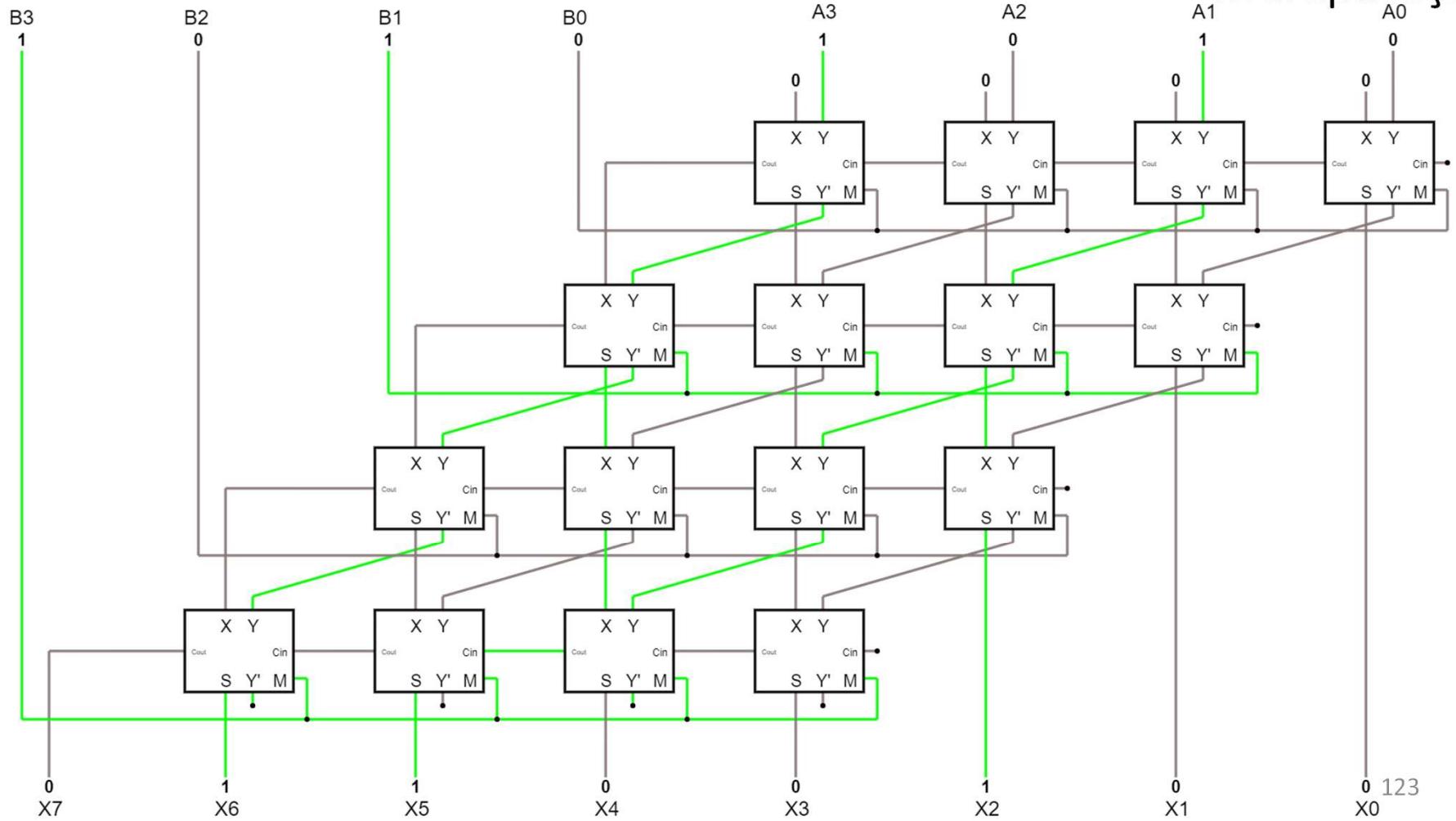


Multiplicação

$$B_0 = \left\{ \begin{array}{l} 0 \Rightarrow \\ 1 \Rightarrow \end{array} \right. \begin{array}{ccccc} 0 & | & 0 & 0 & 0 \\ 0 & | & 0 & 0 & 0 \\ A_3 & | & A_2 & A_1 & A_0 \end{array}$$



Multiplicação



Multiplicação

Exercício

- Implementar um multiplicador de 8 bits utilizando uma matriz de somador estendido de 1 bit.

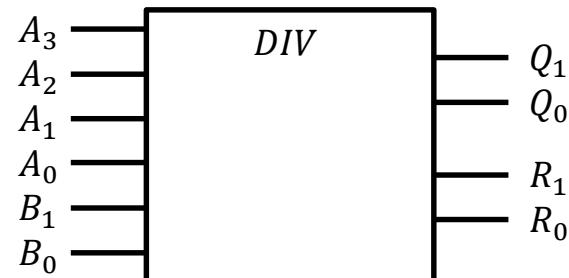
Sumário

- 1. Revisão – Sistemas de Numeração
- 2. Revisão – Representação de Dados
- 3. Revisão – Operações com Binários
- 4. Álgebra Booleana
- 5. Simplificação de Expressões
- 6. Mapa de Karnaugh
- 7. Elementos Lógicos Universais
- 8. Circuitos Combinacionais
- 9. Circuitos Sequenciais

- 1. Somador / Subtrator
- 2. Comparadores
- 3. Codificador/decodificador
- 4. Multiplexador/Demux
- 5. Geradores de paridade
- 6. Circuitos Específicos
- 7. Multiplicadores / Divisores
- 8. ULA
- 9. PLD/PLA/ROM

Multiplicação

- Do modo semelhante à multiplicação, as primeiras *ULAs* não incluíam a operação de divisão;
- Os programas realizavam esta operação mediante o uso de emulação por software;
- A divisão ocorre normalmente entre um numerador de $2 \times n$ bits por denominador de n bits, gerando assim um quociente de n bits;
- Adicionalmente, os algoritmos podem fornecer o resto no mesmo processo;
- Por exemplo, a divisão entre dois números inteiros sem sinal de 4 e 2 bits produziria dois resultados de 2 bits:



Divisão

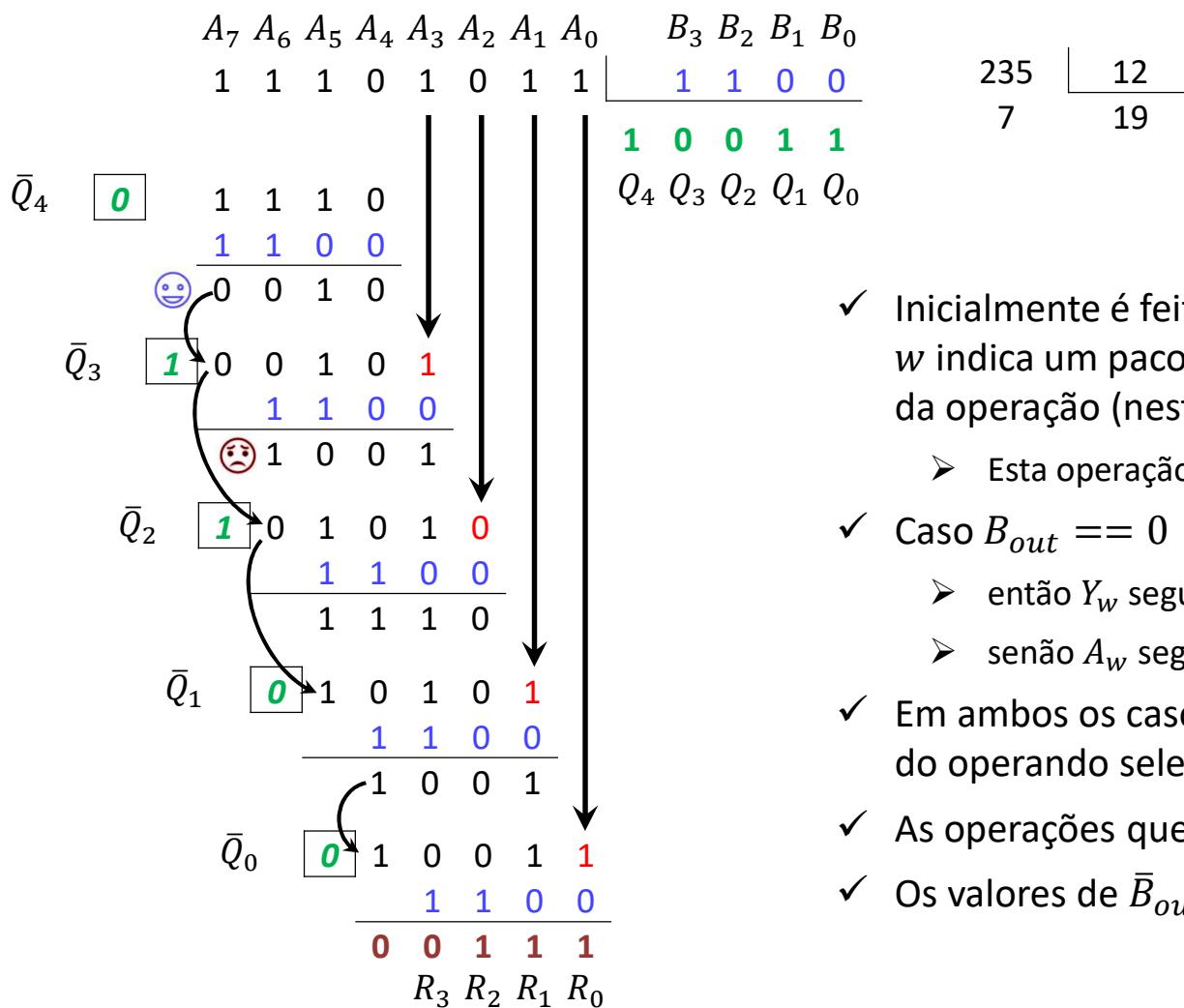
- Exemplo livre da divisão entre dois números inteiros sem sinal de 8 e 4 bits:

$$\begin{array}{r}
 A_7 \ A_6 \ A_5 \ A_4 \ A_3 \ A_2 \ A_1 \ A_0 \\
 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1 \\
 \underline{1 \ 1 \ 0 \ 0} \\
 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \\
 \underline{1 \ 1 \ 0 \ 0} \\
 0 \ 1 \ 0 \ 0 \ 1 \ 1 \\
 \underline{1 \ 1 \ 0 \ 0} \\
 0 \ 0 \ 1 \ 1 \ 1
 \end{array}
 \quad
 \begin{array}{r}
 B_3 \ B_2 \ B_1 \ B_0 \\
 1 \ 1 \ 0 \ 0 \\
 \hline
 1 \ 0 \ 0 \ 1 \ 1
 \end{array}$$

$$\begin{array}{r}
 235 \quad | \quad 12 \\
 7 \qquad \qquad \qquad 19
 \end{array}$$

- Na sequência vamos adaptar este processo trazendo elementos funcionais conhecidos:

Divisão



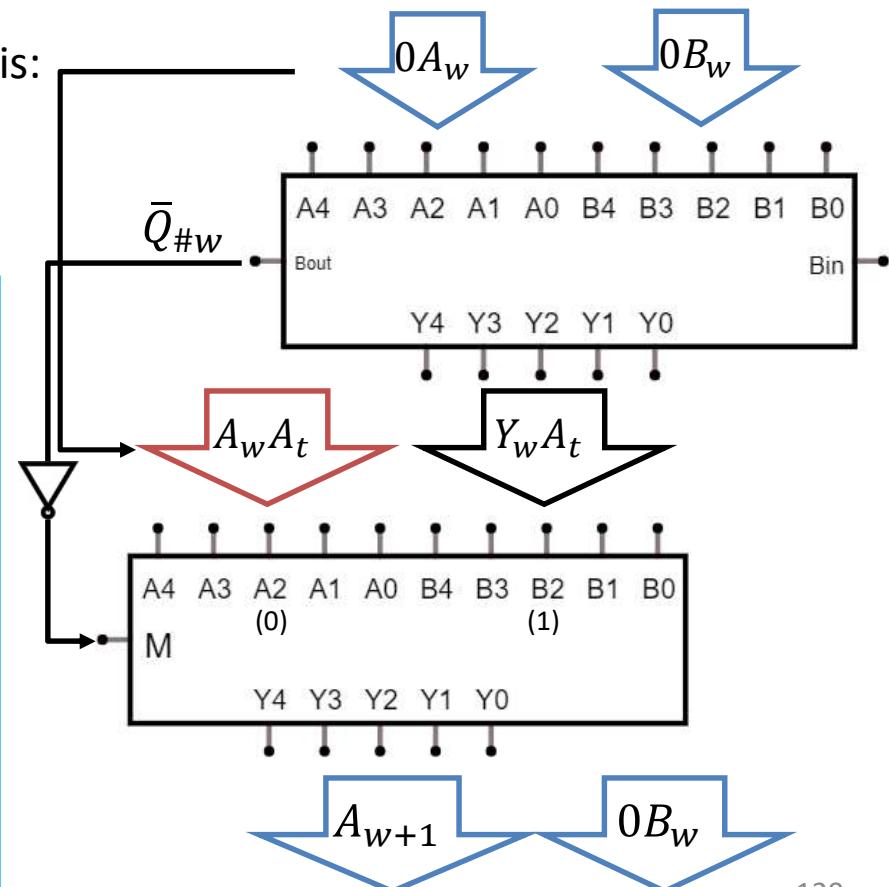
$$\begin{array}{r} 235 \\ 7 \end{array} \quad \begin{array}{r} 12 \\ 19 \end{array}$$

- ✓ Inicialmente é feita a operação $Y_w = A_w - B_w$, onde w indica um pacote de bits equivalente a capacidade da operação (neste exemplo $w = 4\text{bits}$);
 - Esta operação gera um valor para o *Borrow* (B_{out});
- ✓ Caso $B_{out} == 0$
 - então Y_w segue para a próxima operação,
 - senão A_w seguirá;
- ✓ Em ambos os casos um dígito A_t é acrescido a direita do operando selecionado ($X_w A_t = A_{w+1}$);
- ✓ As operações que seguem trabalham com $w + 1$ bits;
- ✓ Os valores de \bar{B}_{out} fornecem os bits do quociente;

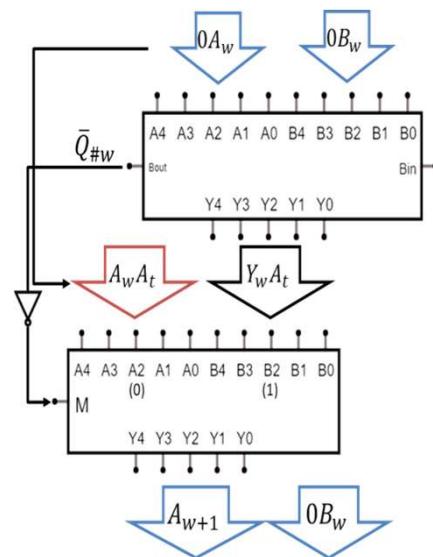
Divisão

- ✓ Vamos lançar mão de dois blocos funcionais:
 - ✓ Subtrator de 5 bits;
 - ✓ Multiplex de 5 bits.

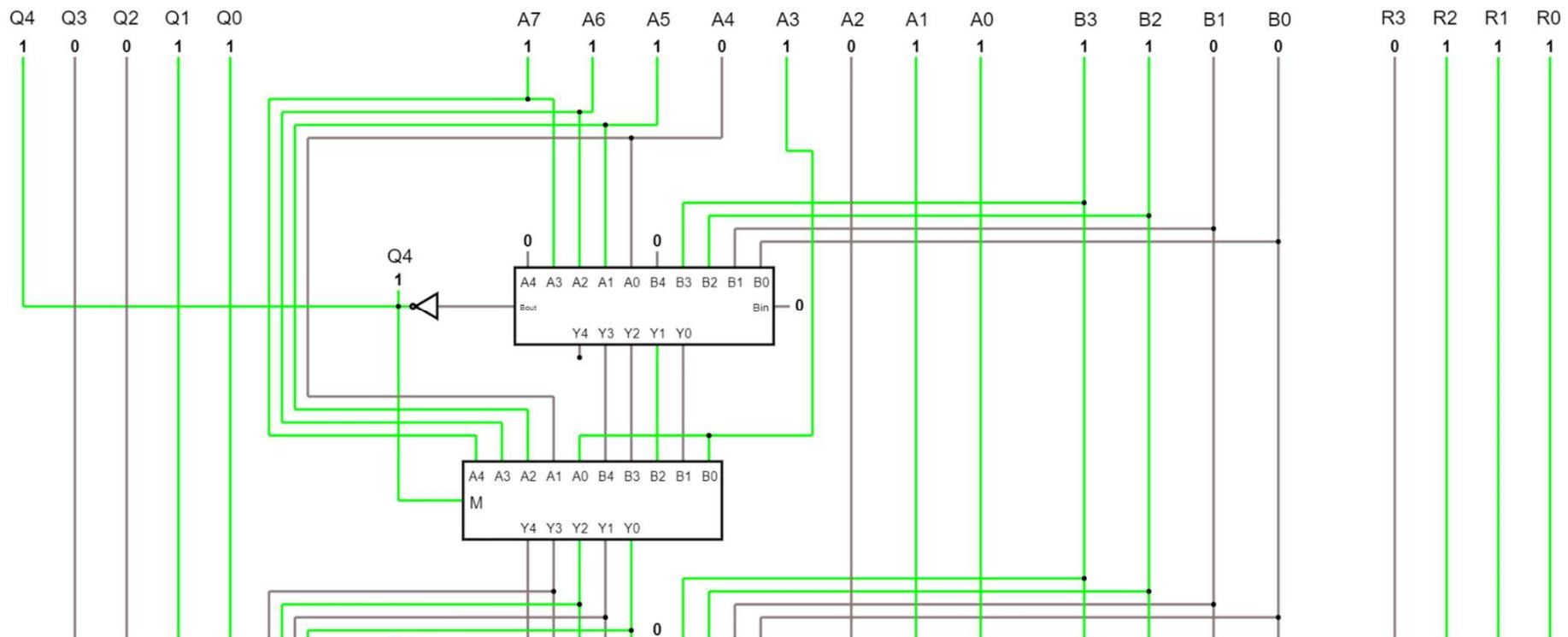
- ✓ Inicialmente é feita a operação $Y_w = A_w - B_w$, onde w indica um pacote de bits equivalente a capacidade da operação (neste exemplo $w = 4\text{bits}$);
 - Esta operação gera um valor para o *Borrow* (B_{out});
- ✓ Caso $B_{out} == 0$
 - então Y_w segue para a próxima operação,
 - senão A_w seguirá;
- ✓ Em ambos os casos um dígito A_t é acrescido a direita do operando selecionado ($X_w A_t = A_{w+1}$);
- ✓ As operações que seguem trabalham com $w + 1$ bits;
- ✓ Os valores de B_{out} fornecem os bits do quociente;



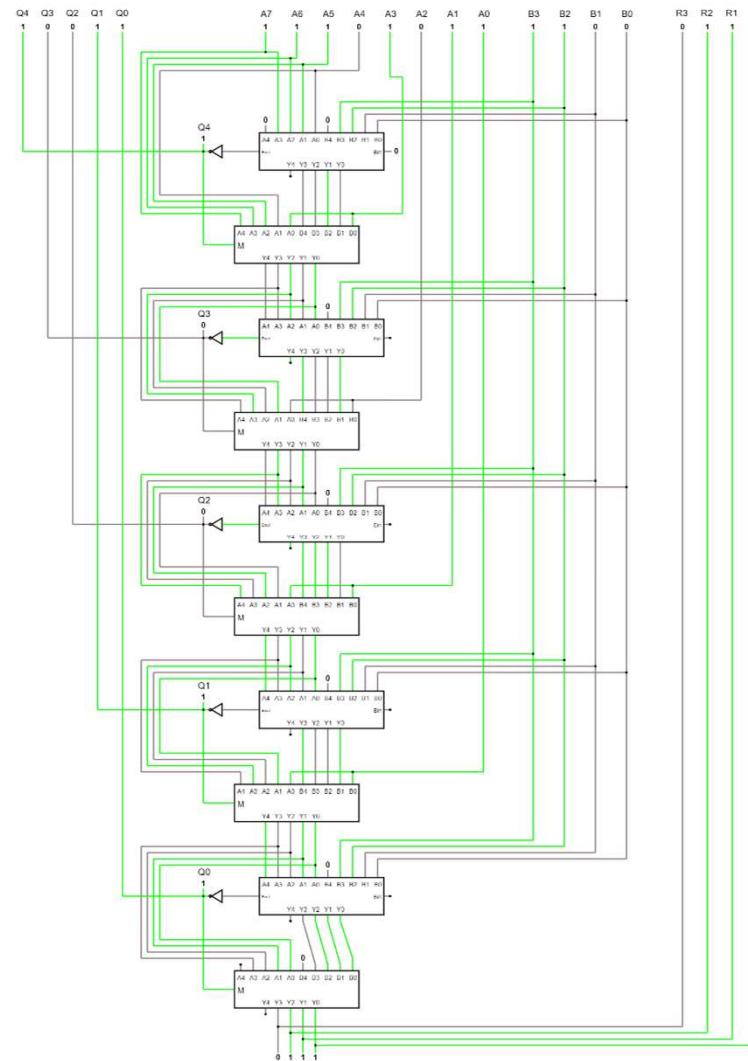
Divisão



Divisão

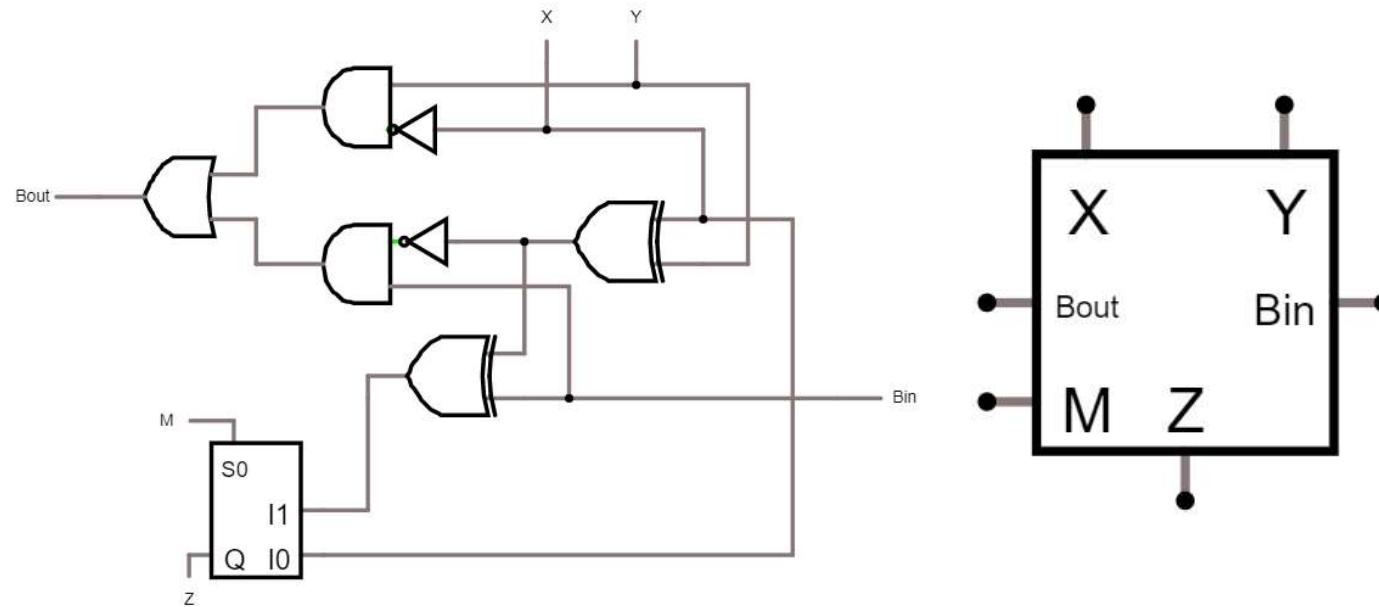


Divisão



Divisão

- ✓ É possível refazer o circuito de divisão utilizando um *array* de Subtrator de 1 bit integrado com um Multiplex;
- ✓ Criaremos um bloco funcional, onde uma chave M permite selecionar a saída Z entre o resultado da subtração e a entrada X ;

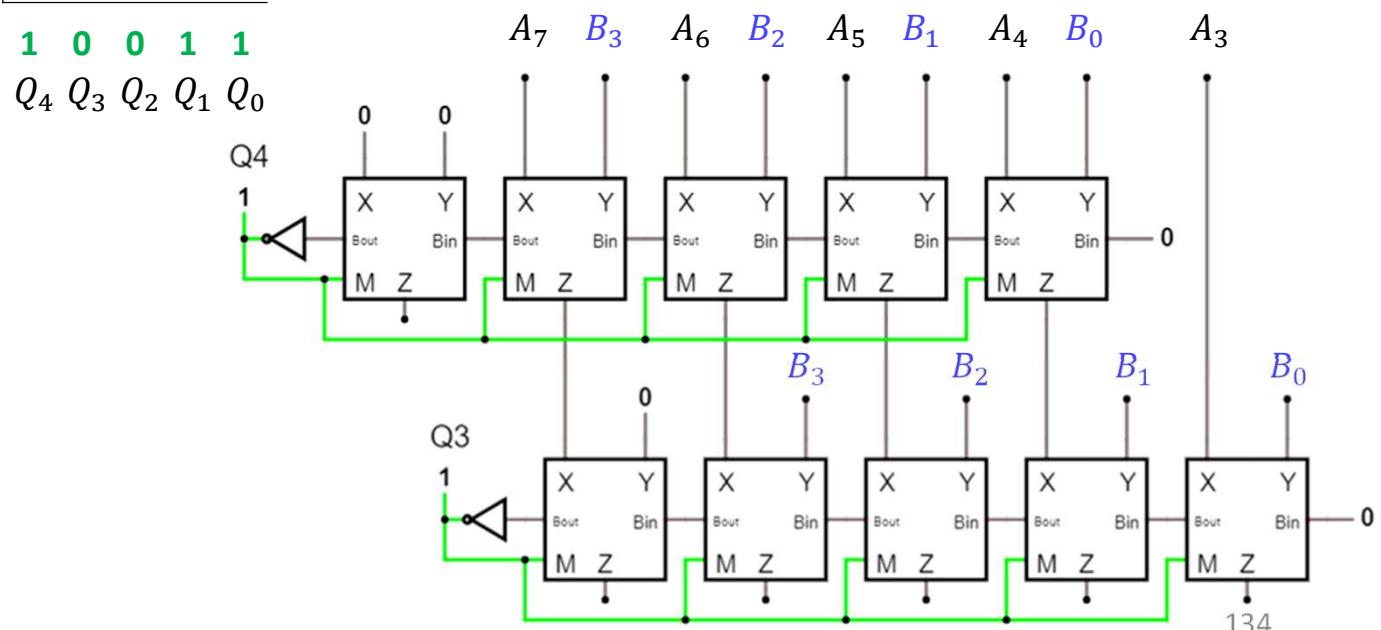


Divisão

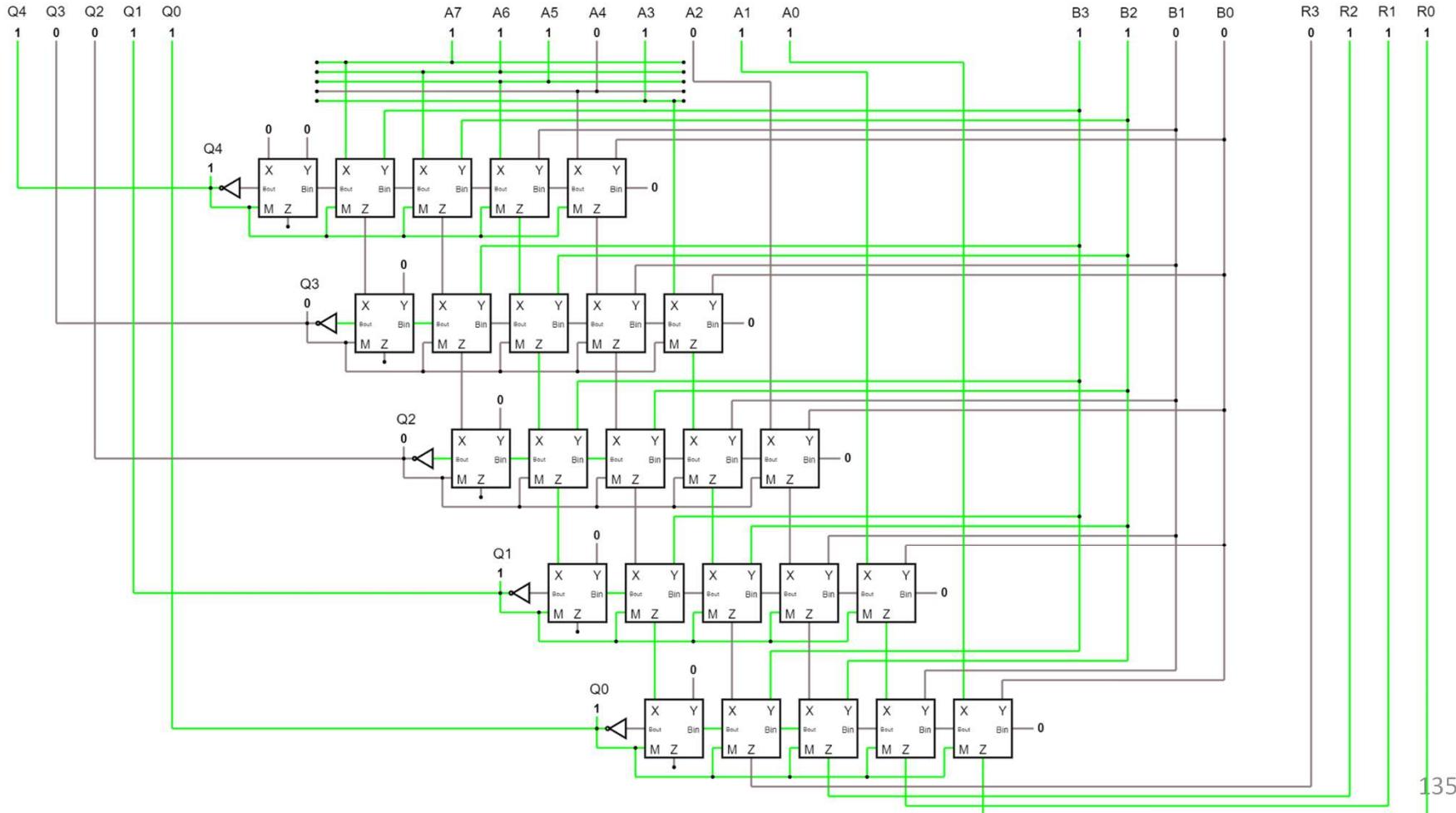
- ✓ O último B_{out} indicará se a linha toda deverá selecionar o resultado da subtração ou o operando X ;

A Karnaugh map diagram illustrating the simplification of the Q_4 term. The top row shows minterms $A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0$ with values 1, 1, 1, 0, 1, 0, 1, 1. The bottom row shows the resulting terms $B_3 B_2 B_1 B_0$ with values 1, 1, 0, 0. A vertical arrow points from the Q_4 term in the first column of the K-map to the Q_3 term in the second column.

Q_4	1	1	1	0	\bar{Q}_4	1	1	0	0	\bar{Q}_3	0	0	1	0	Q_3	0	0	1	1	Q_4	1	0	0	1	1	Q_0
0	1	1	1	0	1	1	0	0	0	1	0	0	1	0	1	0	0	1	1	Q_4	1	0	0	1	1	Q_0

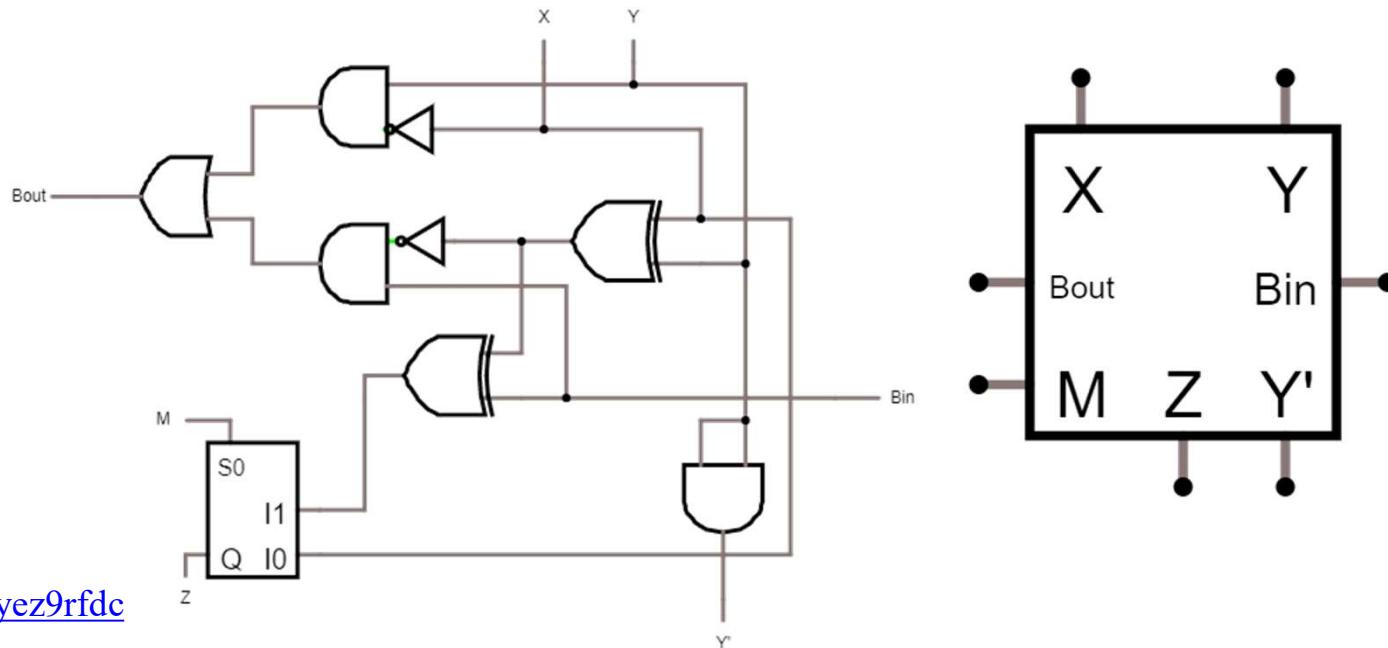


Divisão



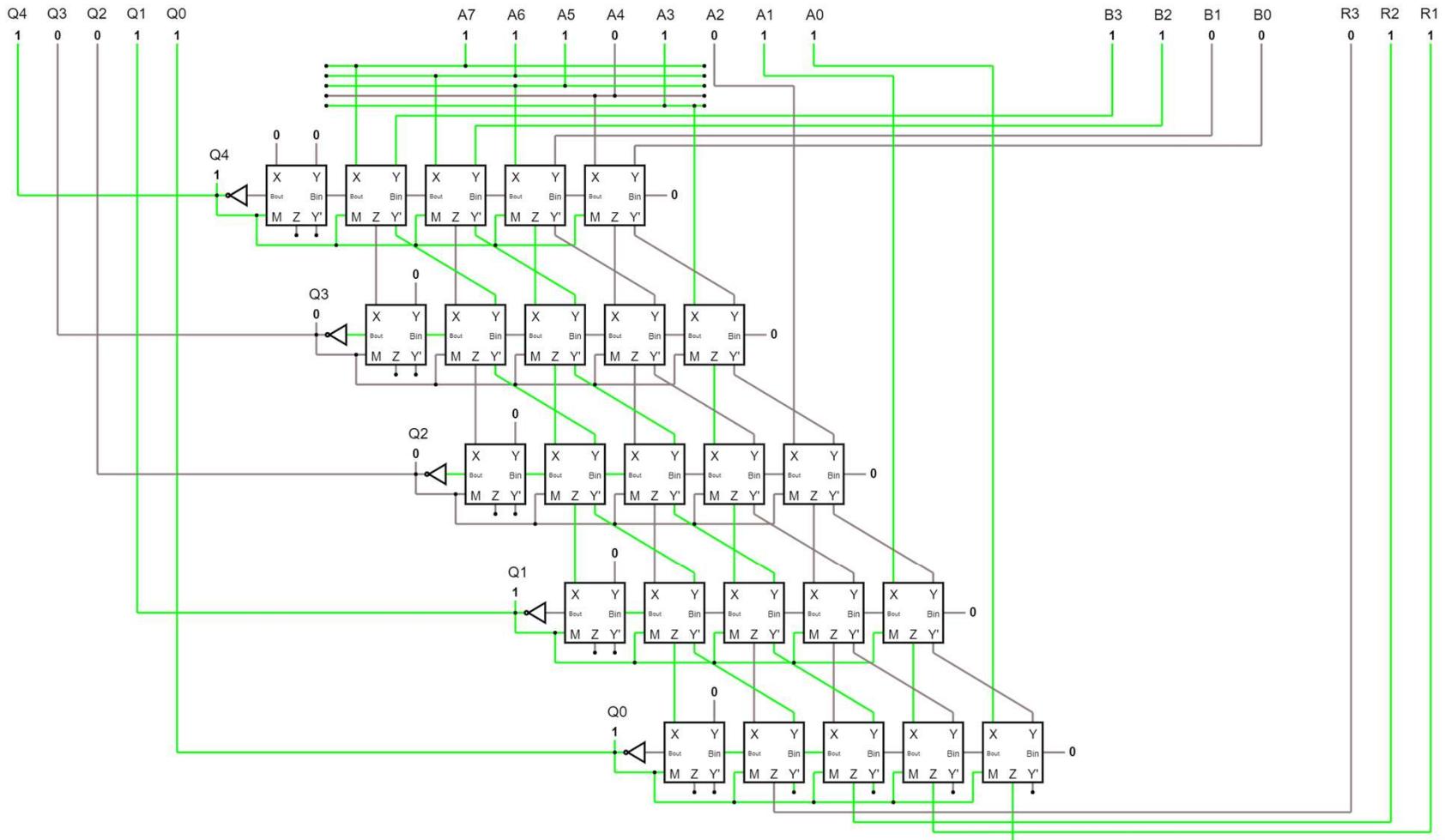
Divisão

- ✓ Uma nova versão do Bloco Funcional pode incluir uma saída adicional replicando a entrada Y (valor de B);
- ✓ Assim, um linha do *array* fornece o divisor da próxima etapa;

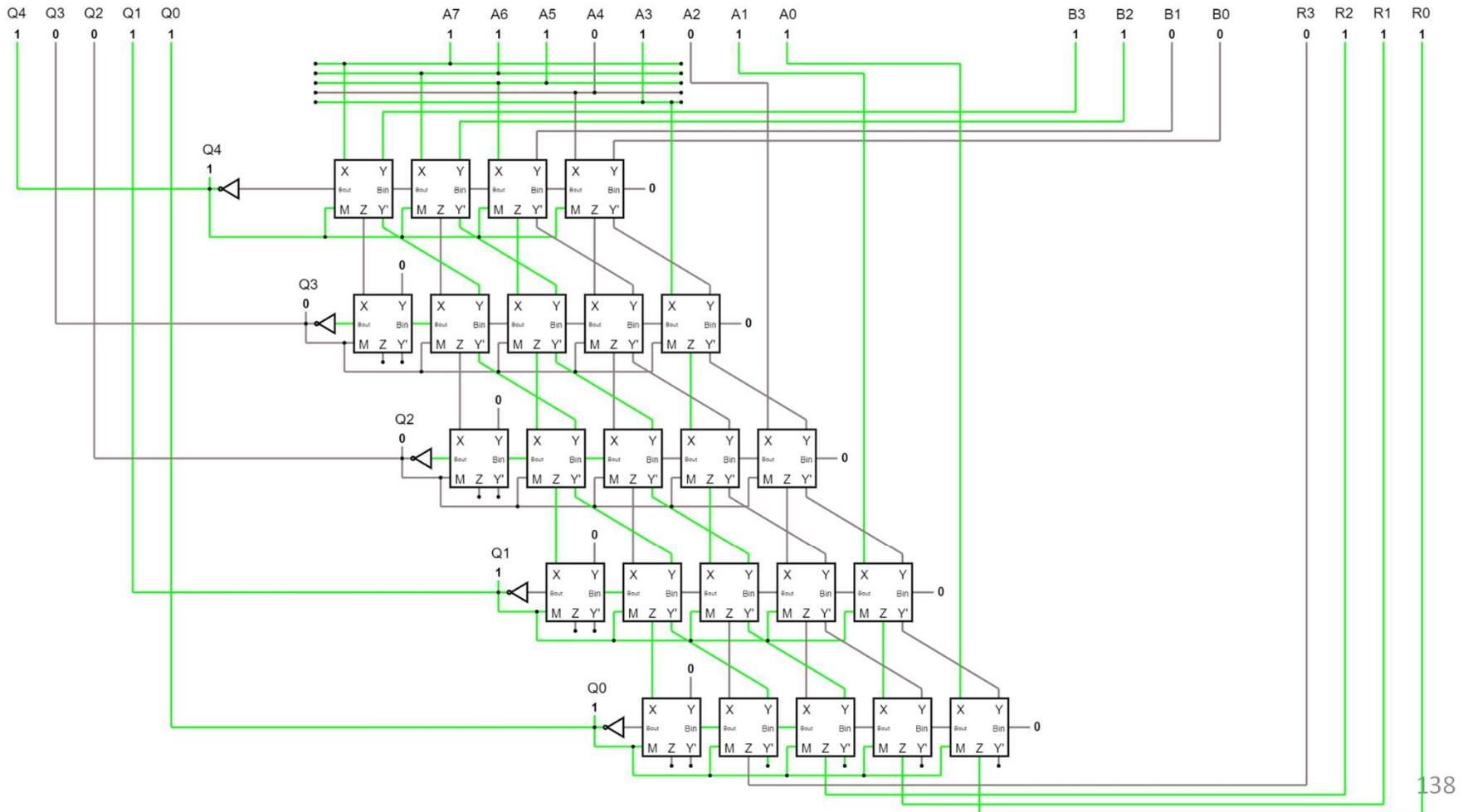


<https://tinyurl.com/yez9rfdc>

Divisão



Divisão



Divisão

Exercício

- Implementar um divisor de 16 bits por 8 bits utilizando uma matriz de subtrator estendido de 1 bit.

Sumário

- 1. Revisão – Sistemas de Numeração
- 2. Revisão – Representação de Dados
- 3. Revisão – Operações com Binários
- 4. Álgebra Booleana
- 5. Simplificação de Expressões
- 6. Mapa de Karnaugh
- 7. Elementos Lógicos Universais
- 8. Circuitos Combinacionais
- 9. Circuitos Sequenciais

- 1. Somador / Subtrator
- 2. Comparadores
- 3. Codificador/decodificador
- 4. Multiplexador/Demux
- 5. Geradores de paridade
- 6. Circuitos Específicos
- 7. Multiplicadores / Divisores
- 8. ULA
- 9. PLD/PLA/PAL/FPGA/ROM