

*UNIOESTE*  
*Ciência da Computação*

*Sistemas Digitais*  
*Circuitos Combinacionais*

*Prof. Jorge Habib El Khouri*  
*Prof. Antonio Marcos Hachisuca*

2020

# 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.

# Regras Básicas da Álgebra Booleana

## **Comutativa**

$$A + B = B + A$$

$$AB = BA$$

## **Associativa**

$$(A + B) + C = A + (B + C)$$

$$(AB)C = A(BC)$$

## **Distributiva**

$$A(B + C) = AB + AC$$

$$A + 0 = A$$

$$A + 1 = 1$$

$$A + A = A$$

$$A + \bar{A} = 1$$

$$\bar{\bar{A}} = A$$

$$A \cdot 0 = 0$$

$$A \cdot 1 = A$$

$$A \cdot A = A$$

$$A \cdot \bar{A} = 0$$

$$\overline{X \cdot Y \cdot Z} = \bar{X} + \bar{Y} + \bar{Z}$$

$$A \oplus 0 = A$$

$$A \oplus 1 = \bar{A}$$

$$A \oplus A = 0$$

$$A \oplus \bar{A} = 1$$

$$A + AB = A$$

$$A + \bar{A}B = A + B$$

$$(A + B)(A + C) = A + BC$$

$$A \oplus B = A\bar{B} + \bar{A}B$$

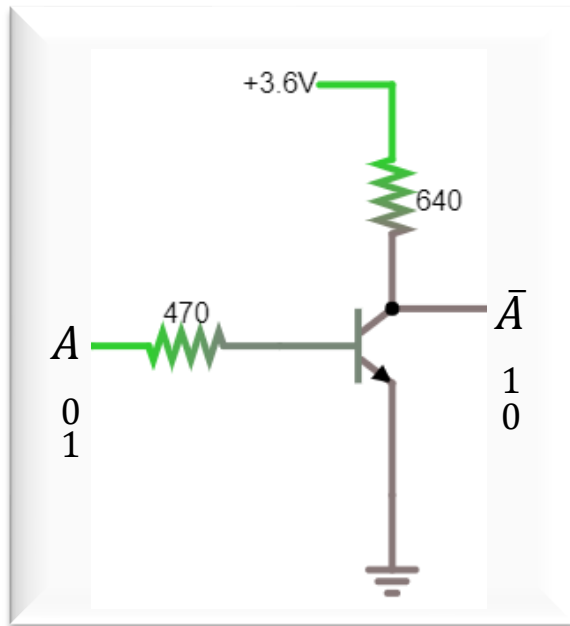
$$\overline{A \oplus B} = AB + \bar{A}\bar{B}$$

$$\overline{X + Y + Z} = \bar{X} \cdot \bar{Y} \cdot \bar{Z}$$

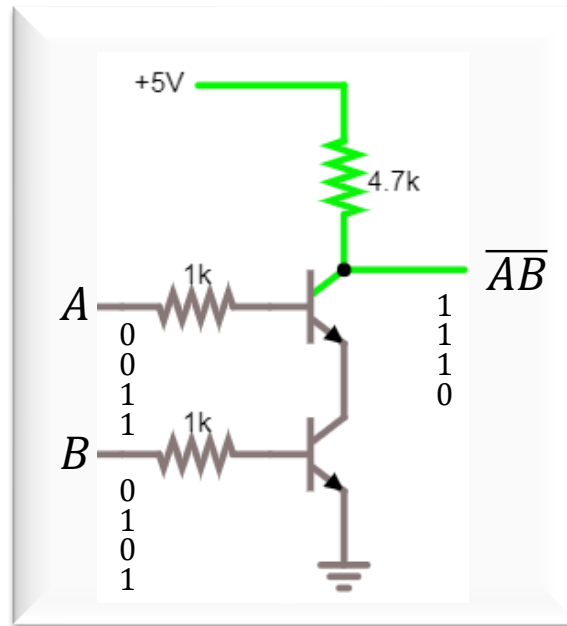


# Elementos Lógicos Universais

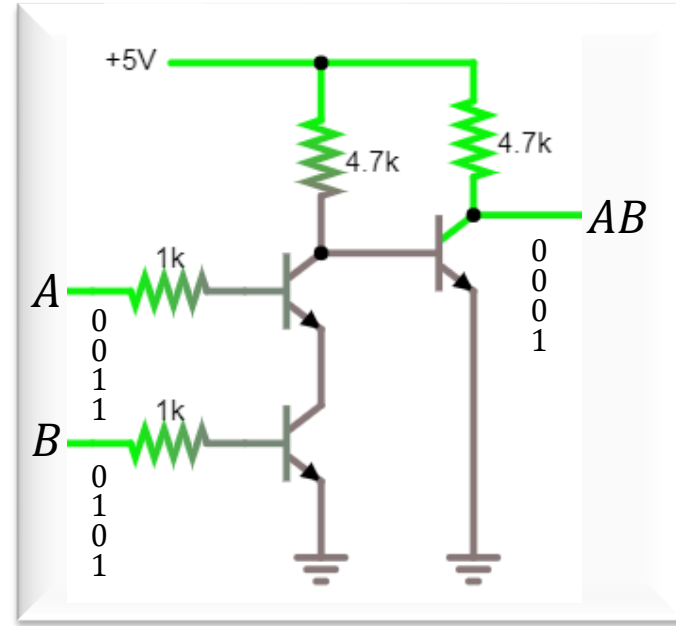
- A porta NAND é considerada universal porque pode ser utilizada para produzir todas as demais portas lógicas;
- A implementação de algumas Portas pode ser feita como segue:



NOT (1T)



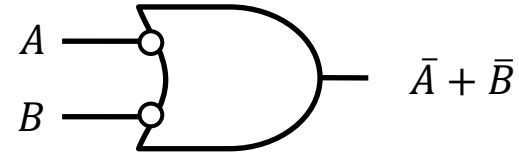
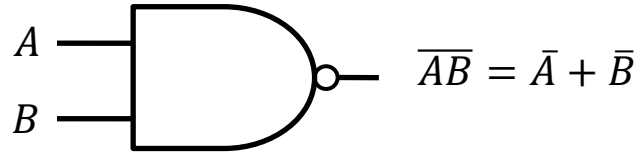
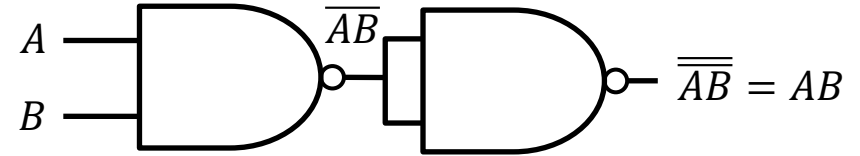
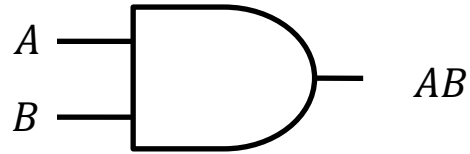
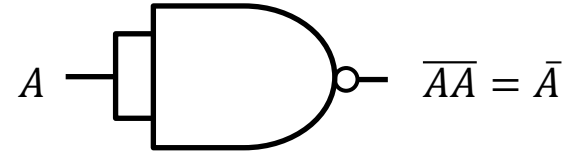
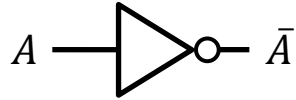
NAND (2T)



AND (3T)

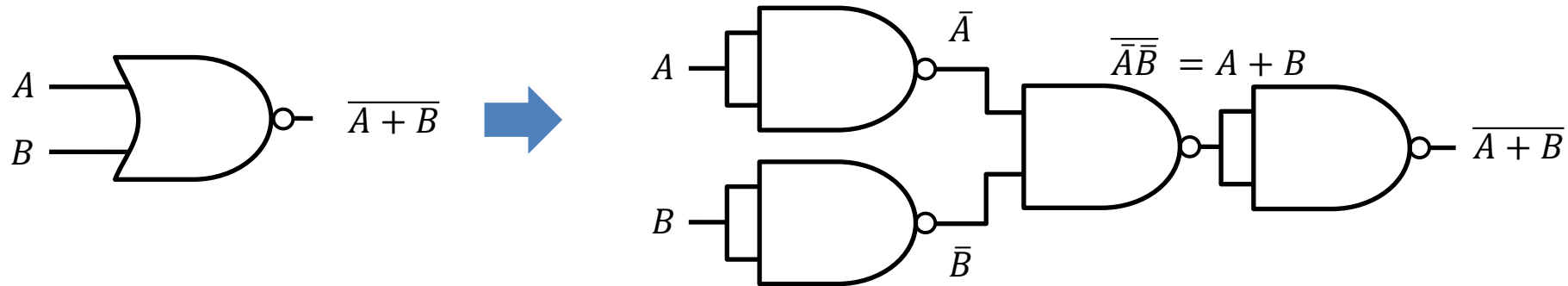
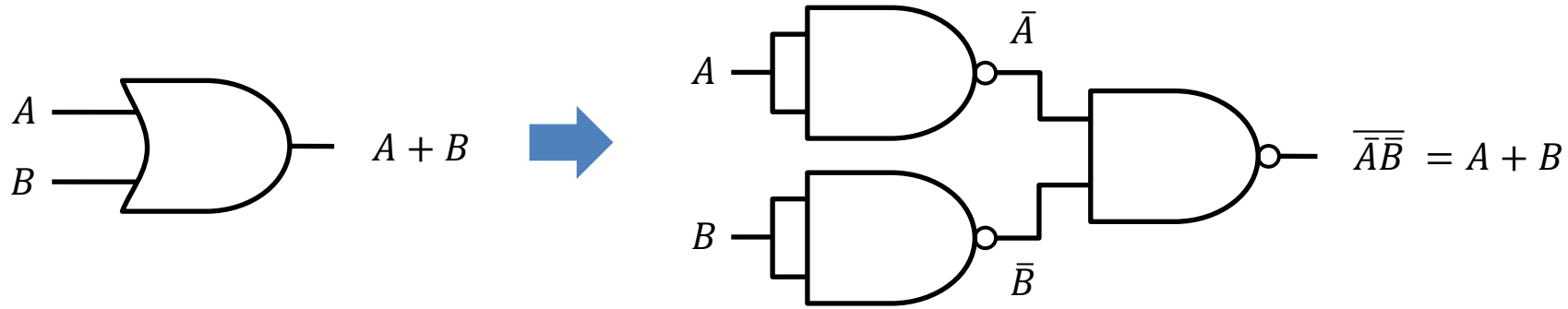
# Equivalência de Portas

## Porta NAND



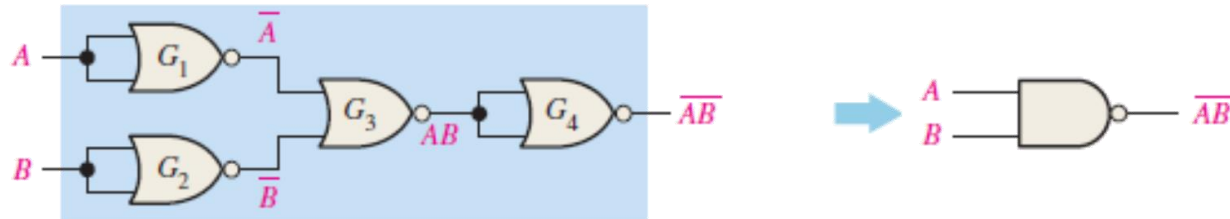
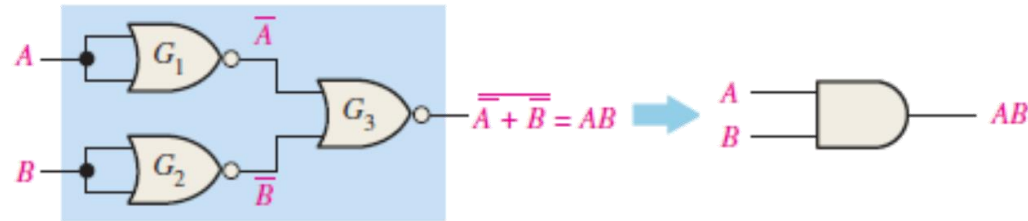
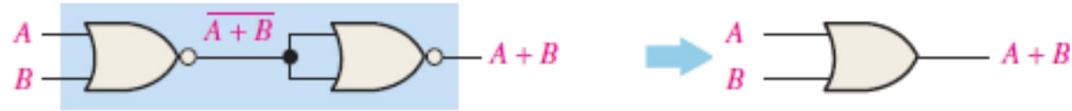
# Equivalência de Portas

## Porta NAND



# Equivalência de Portas

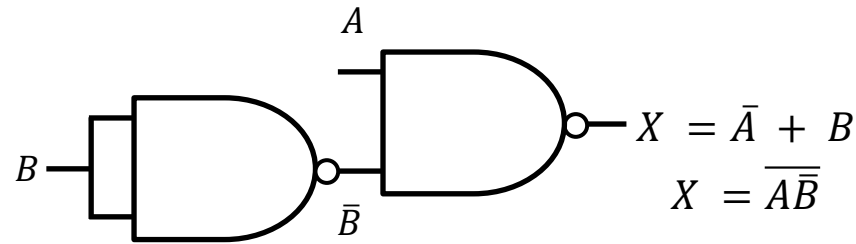
## Porta NOR



# Equivalência de Portas

## Exemplo

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



$$X = A * (B * B)$$

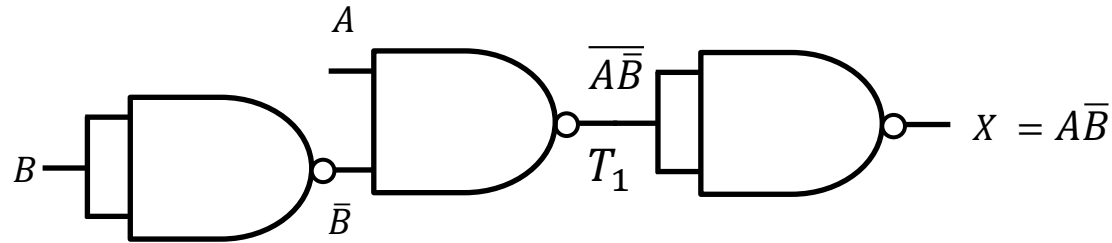
*\* representando o NAND*



# Equivalência de Portas

## Exemplo

$$X = A\bar{B}$$



$$T_1 = A * (B * B)$$

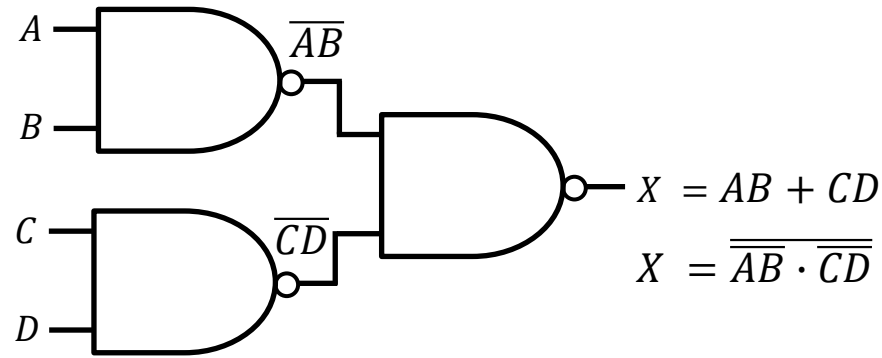
$$X = T_1 * T_1$$

*\* representando o NAND*

# Equivalência de Portas

## Exemplo

$$X = AB + CD$$



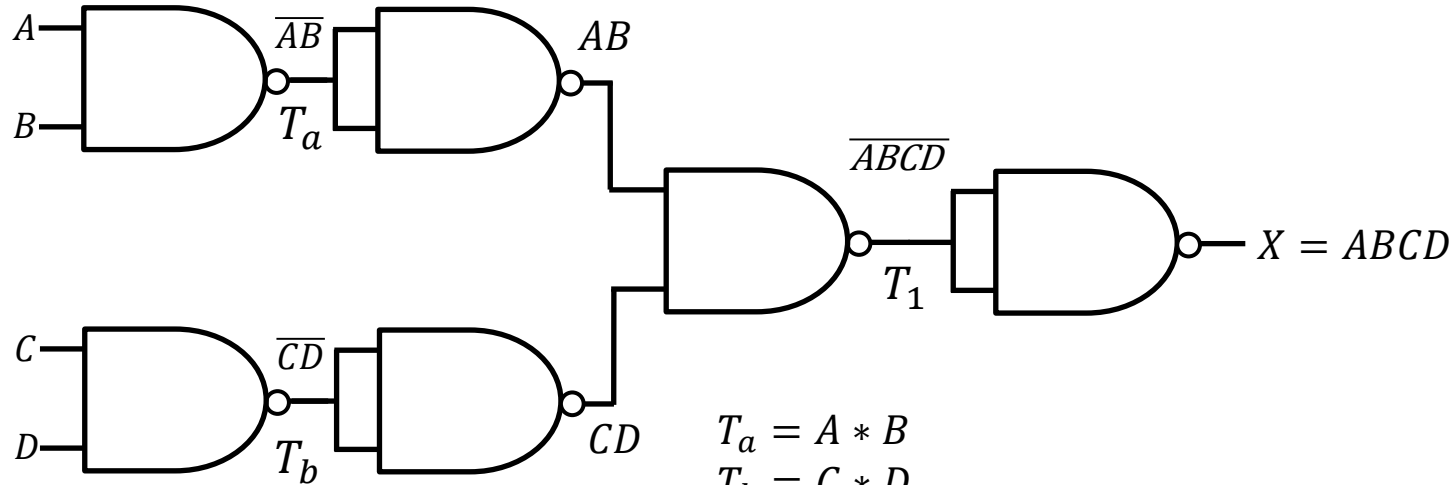
$$X = (A * B) * (C * D)$$

*\* representando o NAND*

# Equivalência de Portas

## Exemplo

$$X = ABCD$$



$$T_a = A * B$$

$$T_b = C * D$$

$$T_1 = (T_a * T_a) * (T_b * T_b)$$

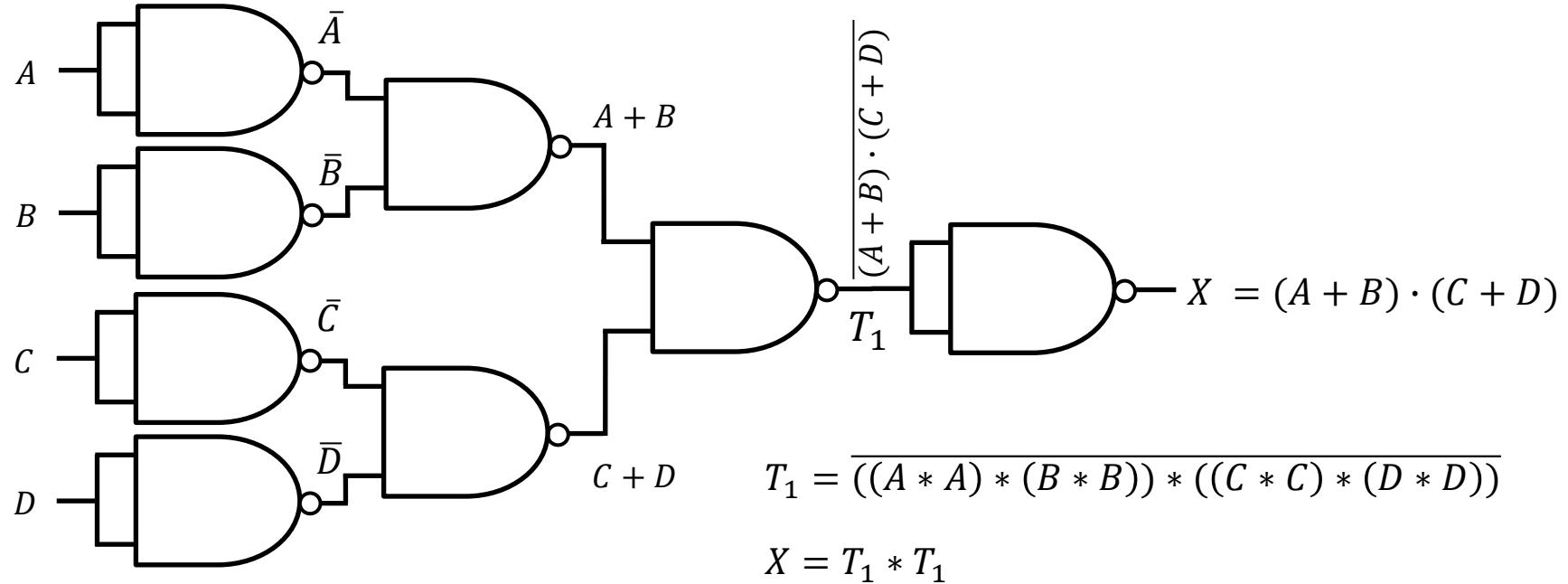
$$X = T_1 * T_1$$

*\* representando o NAND*

# Equivalência de Portas

## Exemplo

$$X = (A + B) \cdot (C + D)$$



*\* representando o NAND*

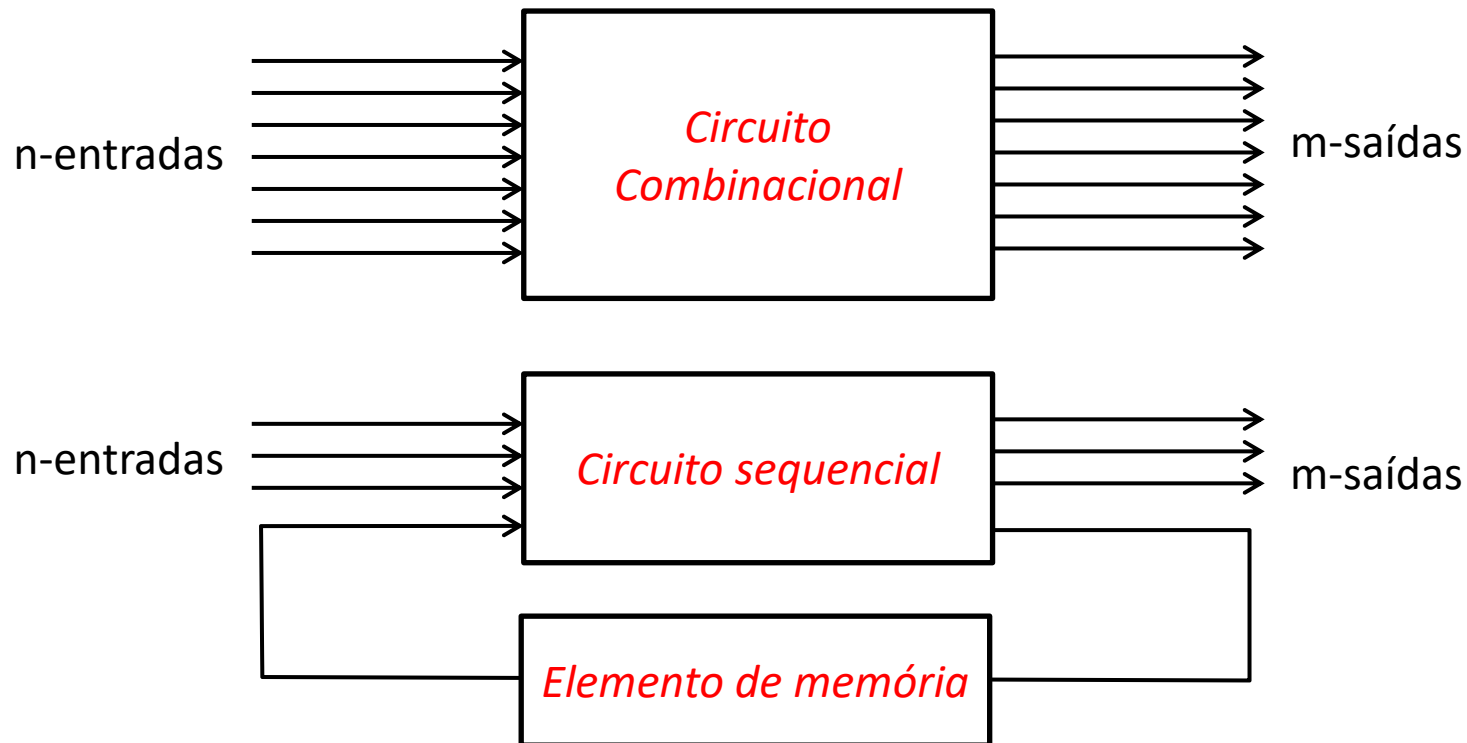
# CIRCUITOS COMBINACIONAIS

- Um circuito combinacional consiste em portas lógicas cujas saídas, em qualquer momento, são determinadas pela combinação dos valores das entradas
- Para  $n$  variáveis de entrada, existem  $2^n$  combinações de entrada binária possíveis
- Para cada combinação binária das variáveis de entrada, existe uma saída possível

# Circuitos Combinacionais vs circuitos sequenciais

- Os circuitos combinacionais não possuem memória interna
  - O valor de saída depende apenas dos valores atuais de entrada
- Os circuitos sequenciais contem lógica combinacional, e elementos de memória (usados para armazenar estados de circuito)
  - As saídas dependem dos valores de entrada atuais e dos valores de entrada anteriores (mantidos nos elementos de memória)

# Circuitos Combinacionais vs circuitos sequenciais





# Circuitos Combinacionais vs circuitos sequenciais

## CIRCUITOS COMBINACIONAIS

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

## CIRCUITOS SEQUENCIAIS

1. Latches
2. Flip-Flop
3. Registradores
4. Contadores
5. Máquina de Estados
6. Geradores de clock
7. Memória RAM
8. Sequenciadores



# SOMADOR

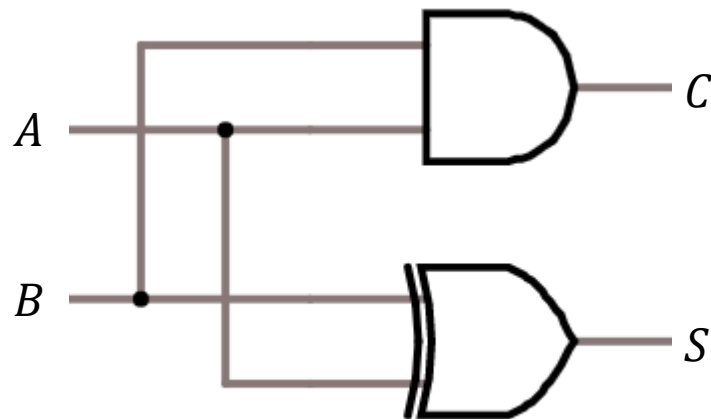
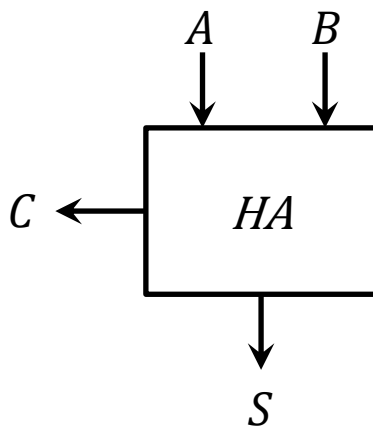
# Circuito Somador *Half-Adder*

$$\begin{array}{r} C \\ + \\ A \\ B \\ \hline S \end{array}$$

$A$	$B$	$S$	$C$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S = A \oplus B$$

$$C = AB$$



# Circuito Somador

## Full-Adder

$C_{in}$	$A$	$B$	$S$	$C_{out}$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = ??$$

$$\begin{array}{r} C_{out} \quad C_{in} \\ + \quad A \\ \quad B \\ \hline S \end{array}$$

$$C_{out} = AB\bar{C}_{in} + \bar{A}BC_{in} + A\bar{B}C_{in} + ABC_{in}$$

$$C_{out} = AB(\bar{C}_{in} + C_{in}) + C_{in}(\bar{A}B + A\bar{B})$$

$$C_{out} = AB + C_{in}(A \oplus B)$$

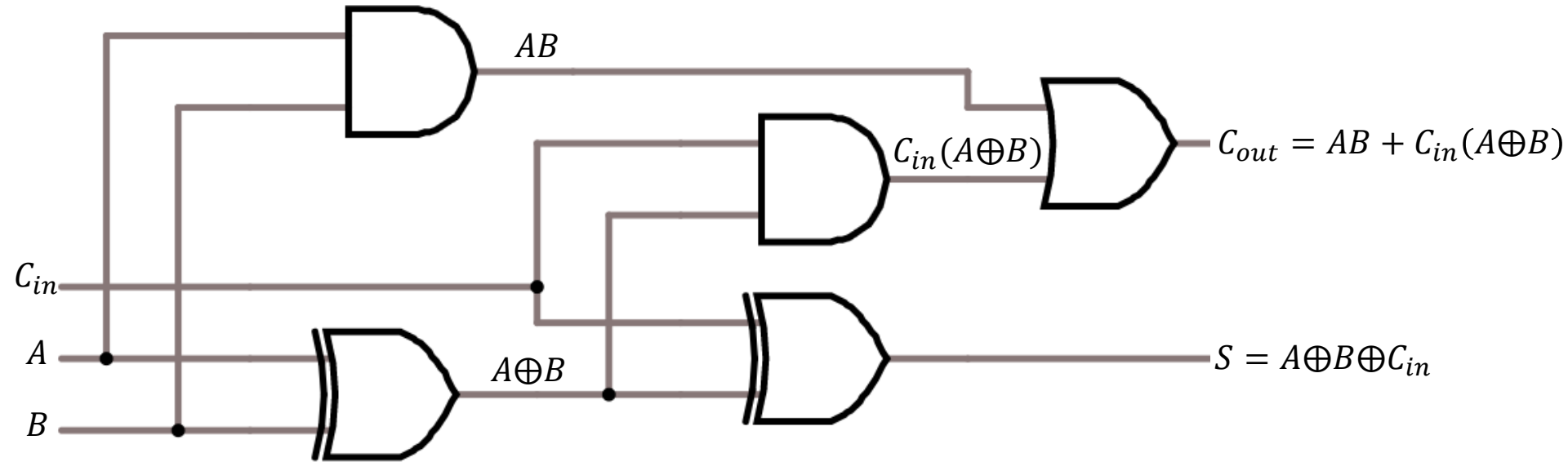
		$\bar{C}_{in}$	$C_{in}$
		0	1
$\bar{A}\bar{B}$	$AB$	0	1
$\bar{A}B$	$AB$	2	3
$A\bar{B}$	$AB$	4	5
$AB$	$AB$	6	7

$$\boxed{AB} + \boxed{AC_{in}} + \boxed{BC_{in}}$$

$$C_{out} = AB + C_{in}(A + B)$$

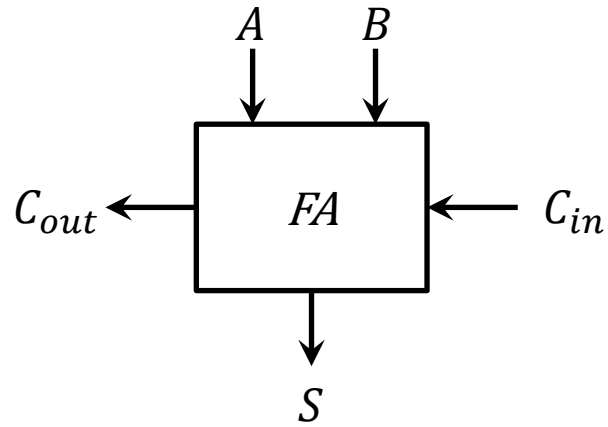
# Circuito Somador

## *Full-Adder*



# Circuito Somador

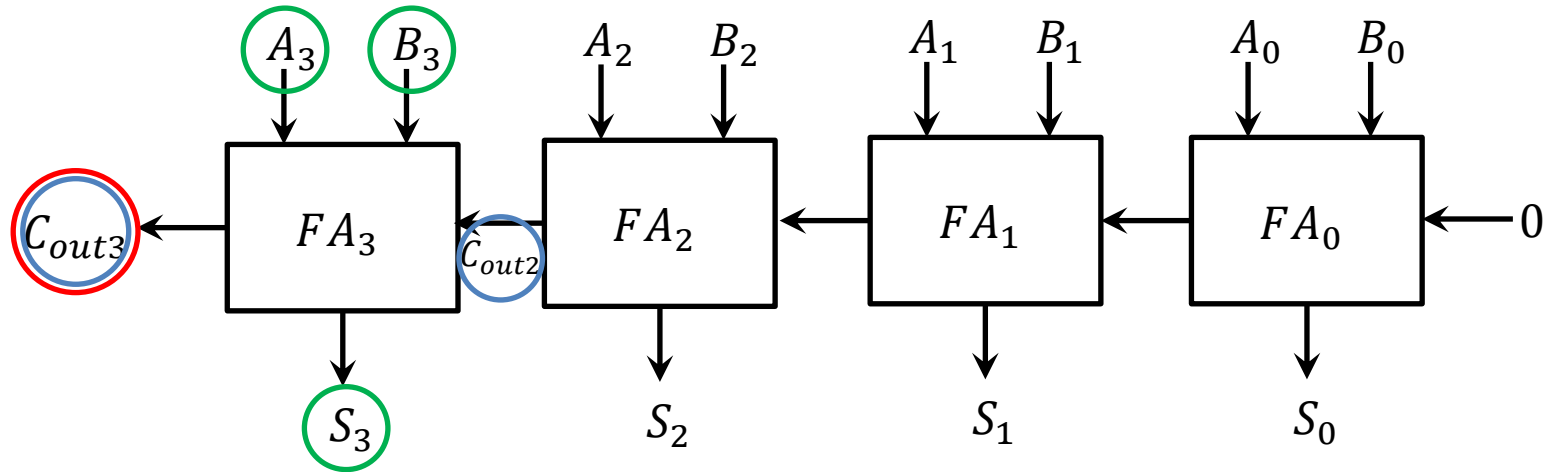
## *Full-Adder*



# Circuito Somador

## Full-Adder

Soma de dois números de 4 bits:  $S_{0...3} = A_{0...3} + B_{0...3}$

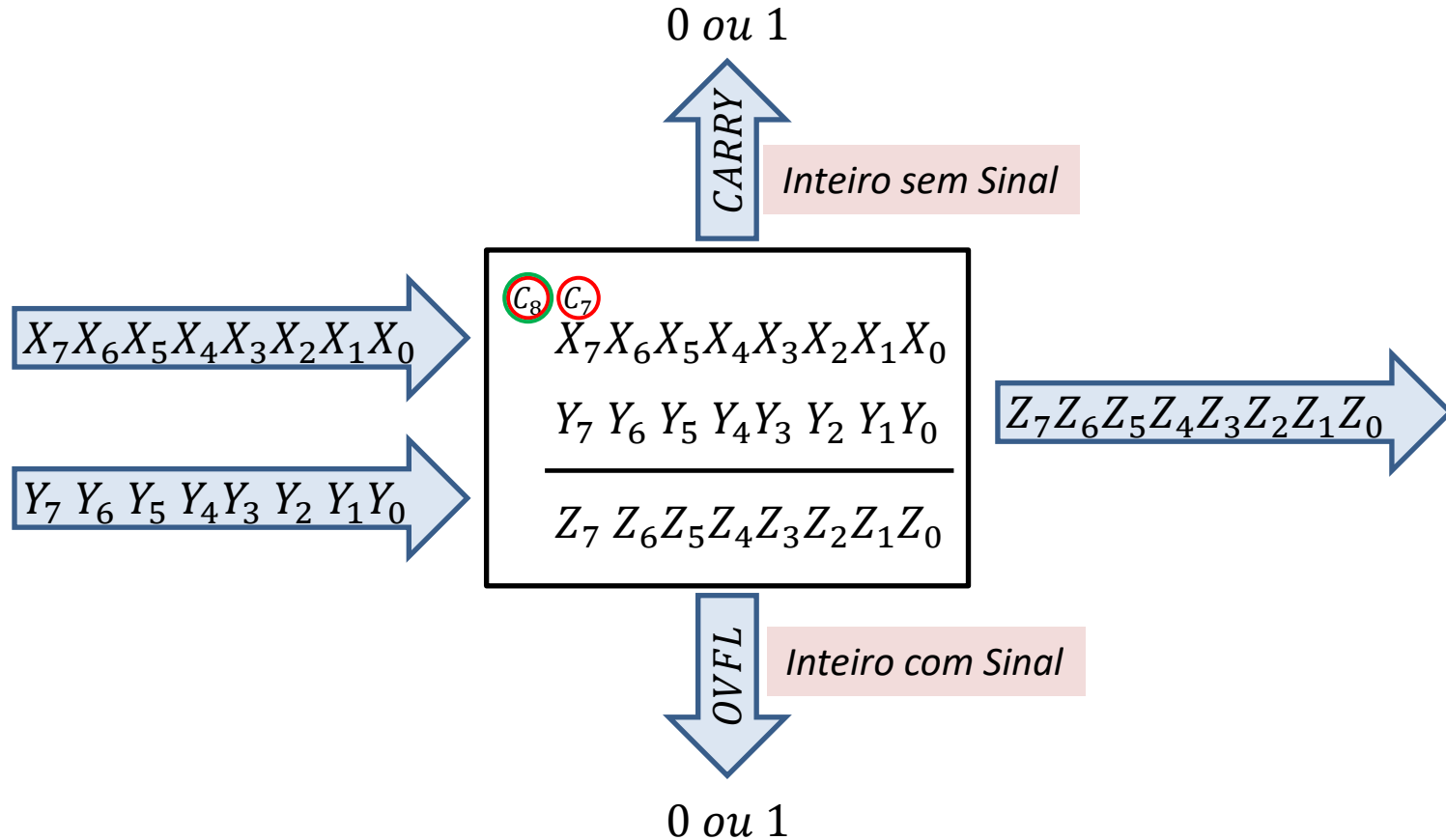


*CARRY* = flag para soma de inteiros sem sinal: se 1 então Erro

*OVFL* = flag para soma de inteiros com sinal: se 1 então Erro

*OVFL* = flag para soma de inteiros com sinal: se 1 então Erro

# Soma de Inteiros





# Soma de Inteiros com Sinal OVERFLOW

<i>SINAIS</i>			
<i>A</i>	<i>B</i>	<i>R</i>	<i>OVFL</i>
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

$$OVFL = AB\bar{R} + \bar{A}\bar{B}R$$

$$OVFL = A_3B_3\bar{S}_3 + \bar{A}_3\bar{B}_3S_3$$

		$\bar{R}$	$R$
	$AB \backslash C$	0	1
$\bar{A}\bar{B}$	00	0	1
$\bar{A}B$	01	2	3
$AB$	11	6	7
$A\bar{B}$	10	4	5

$$AB\bar{R} + \bar{A}\bar{B}R$$

# Soma de Inteiros com Sinal

## OVERFLOW

$C_3C_2C_1C_0$

<i>SINAIS</i>				
<i>A</i>	<i>B</i>	<i>R</i>	<i>OVFL</i>	$C_3 \oplus C_2$
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	1	1
1	1	1	0	0

$\begin{array}{r} \underline{0001} \\ 0101 \\ + 0001 \\ \hline 0110 \end{array}$

$\begin{array}{r} \underline{\textcolor{red}{01}11} \\ 0111 \\ + 0001 \\ \hline 1000 \end{array}$

$\begin{array}{r} \underline{1101} \\ 0101 \\ + 1101 \\ \hline 0010 \end{array}$

$\begin{array}{r} \underline{0001} \\ 0101 \\ + 1001 \\ \hline 1110 \end{array}$

$\begin{array}{r} \underline{1111} \\ 1111 \\ + 0001 \\ \hline 0000 \end{array}$

$\begin{array}{r} \underline{0011} \\ 1011 \\ + 0001 \\ \hline 1100 \end{array}$

$\begin{array}{r} \underline{\textcolor{red}{10}01} \\ 1101 \\ + 1001 \\ \hline 0110 \end{array}$

$\begin{array}{r} \underline{1101} \\ 1101 \\ + 1101 \\ \hline 1010 \end{array}$

$$OVFL = C_3 \oplus C_2$$

# Circuito Subtrator

## Full-Subtractor

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

$$S = A \oplus B \oplus B_{in}$$

$$B_{out} = ??$$

		$\overline{B_{in}}$	$B_{in}$
		0	1
$AB$	$\overline{A}\overline{B}$	0	1
	$\overline{A}B$	1	1
	$AB$	6	7
	$A\overline{B}$	4	5

$$\overline{A}B + BB_{in} + \overline{A}B_{in}$$

$$B_{out} = \overline{A}B + B_{in}(\overline{A} + B)$$

-1	0	-1	-1	-1	-1
-	0	-	0	-	0
1		0		1	
1		1		0	

$$B_{out} = \overline{A}\overline{B}\overline{B_{in}} + \overline{A}\overline{B}B_{in} + \overline{A}BB_{in} + ABB_{in}$$

$$B_{out} = \overline{A}B(B_{in} + \overline{B_{in}}) + B_{in}(\overline{A}\overline{B} + AB)$$

$$B_{out} = \overline{A}B + B_{in}(\overline{A} \oplus \overline{B})$$

# Circuito Subtrator

## Full-Subtractor

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

$$S = A \oplus B \oplus B_{in}$$

$$B_{out} = ??$$

		$\overline{B_{in}}$	$B_{in}$
$AB \backslash C$		0	1
$\overline{A}\overline{B}$	00	0	1 <sub>1</sub>
$\overline{A}B$	01	1 <sub>2</sub>	1 <sub>3</sub>
$AB$	11	6	1 <sub>7</sub>
$A\overline{B}$	10	4	5

$$\overline{A}B + AB B_{in} + \overline{A}\overline{B} B_{in}$$

$$B_{out} = \overline{A}B + B_{in}(AB + \overline{A}\overline{B}) \quad B_{out} = \overline{A}B + B_{in}(\overline{A \oplus B})$$


-1	0	-1	-1	-1	-1
-	0	-	0	-	0
1		0		1	
1		1		0	

# REPRESENTAÇÃO DE DADOS

## INTEIROS SEM E COM SINAL

# Representação de Dados

## *Inteiros*

- Como obter um Binário que represente um número inteiro?
  - Inteiro sem sinal e Inteiro com sinal;
  - Para os casos exemplos será utilizado  $N = 4bits$ ;
  - Inteiro sem sinal contém apenas a magnitude do número:
    - o binário correspondente pode ser obtido pela conversão do número para a base 2;
- 
- $X = 8$                        $BIN = 1000$
  - $X = 13$                       $BIN = 1101$
  - $X = 19$                       $BIN = 10011$  
  - $BIN = 1001$                  $X = 9$
  - $BIN = 0000$                  $X = 0$

Assim, a faixa de representação é dada por:

$$0 \leq X_4 \leq 15$$

$$0 \leq X_{16} \leq 65535$$

$$0 \leq X_N \leq 2^N - 1$$

# Representação de Dados

## *Inteiros*

- Para número inteiro com sinal há a necessidade de codificar o sinal e a magnitude:
- O método Complemento de 2 ou  $C - 2$  consiste em:
- Se o sinal de  $X$  for positivo, converter o número para a base 2 e verificar se o *MSB* resultou em 0:

$X = +4$       0100       $\rightarrow$  Sinal Ok.

$X = +9$       1001       $\rightarrow$  Sinal inconsistente

$X = +0$       0000       $\rightarrow$  Sinal Ok.

$X = +7$       0111       $\rightarrow$  Sinal Ok.

# Representação de Dados

## Inteiros

- Se o sinal de  $X$  for negativo, seguir o processo indicado abaixo e verificar se o *MSB* resultou em 1:

$$\begin{array}{rcl}
 X = -5 & (|X|)_2 : & 0101 \\
 & 0 \Leftrightarrow 1 : & 1010 \\
 & + & 1 \\
 & \text{----} & \\
 \text{BIN:} & & 1011 \quad \checkmark
 \end{array}$$

$$\begin{array}{rcl}
 X = -0 & (|X|)_2 : & 0000 \\
 & 0 \Leftrightarrow 1 : & 1111 \\
 & + & 1 \\
 & \text{----} & \\
 \text{BIN:} & & 0000 \quad \checkmark
 \end{array}$$

$$\begin{array}{rcl}
 X = -10 & (|X|)_2 : & 1010 \\
 & 0 \Leftrightarrow 1 : & 0101 \\
 & + & 1 \\
 & \text{----} & \\
 \text{BIN:} & & 0110 \quad \times
 \end{array}$$

$$\begin{array}{rcl}
 X = -1 & (|X|)_2 : & 0001 \\
 & 0 \Leftrightarrow 1 : & 1110 \\
 & + & 1 \\
 & \text{----} & \\
 \text{BIN:} & & 1111 \quad \checkmark
 \end{array}$$

$$\begin{array}{rcl}
 X = -8 & (|X|)_2 : & 1000 \\
 & 0 \Leftrightarrow 1 : & 0111 \\
 & + & 1 \\
 & \text{----} & \\
 \text{BIN:} & & 1000 \quad \checkmark
 \end{array}$$

$$\begin{array}{rcl}
 X = -9 & (|X|)_2 : & 1001 \\
 & 0 \Leftrightarrow 1 : & 0110 \\
 & + & 1 \\
 & \text{----} & \\
 \text{BIN:} & & 0111 \quad \times
 \end{array}$$



Assim, a faixa de representação para  $C - 2$  é dada por:

$$-8 \leq X_4 \leq +7$$

$$-32768 \leq X_{16} \leq +32767$$

$$-2^{N-1} \leq X_N \leq +(2^{N-1} - 1)$$

O método  $C - 2$  proporciona a mesma representação para o  $+0$  e  $-0$ .

# Representação de Dados

## *Inteiros*

- Decodificar os seguintes binários que representam inteiros com sinal em  $C - 2$ :

$BIN = 0010$

$X = +2$

$BIN = 1101$

$X = -3$

$BIN:$	1101
-	1
	----
	1100
$0 \Leftrightarrow 1:$	<b>0011</b> ( $ X _2$ )

$BIN:$	1101
$0 \Leftrightarrow 1:$	0010
	+ 1
	----
$( X )_2:$	<b>0011</b>

$BIN = 1001$

$X = -7$

$BIN:$	1001
$0 \Leftrightarrow 1:$	0110
	+ 1
	----
$( X )_2:$	<b>0111</b>

$BIN = 0111$

$X = +7$

$BIN = 1100$

$X = -4$

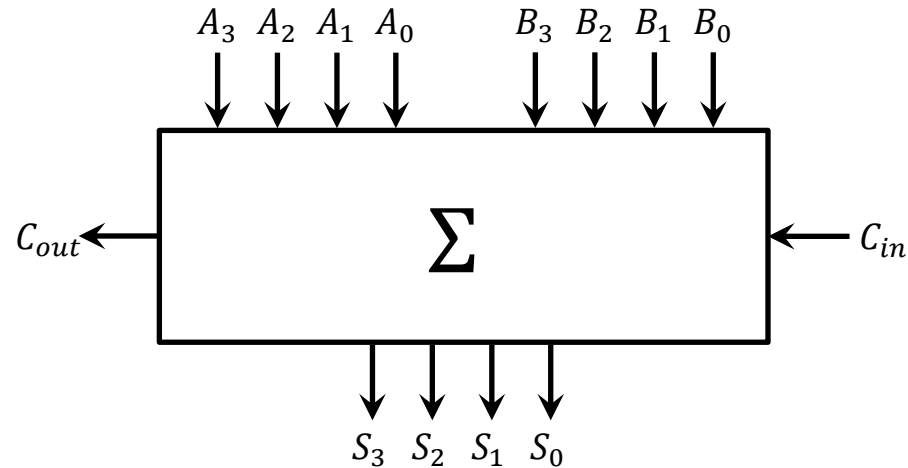
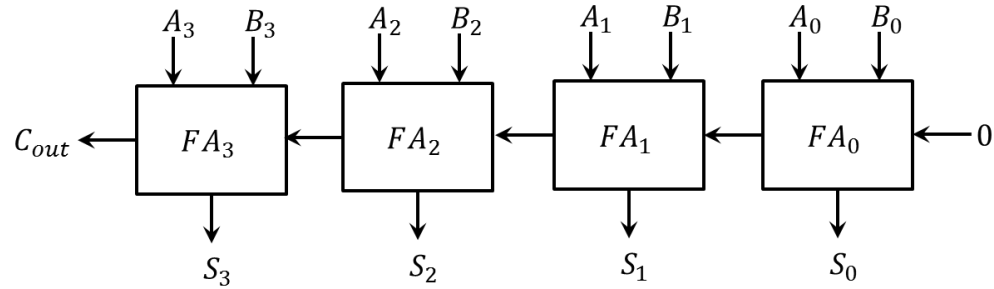
$BIN:$	1100
$0 \Leftrightarrow 1:$	0011
	+ 1
	----
$( X )_2:$	<b>0100</b>

# ENCADEANDO SOMADORES

## INTEGRANDO SOMADOR/SUBTRATOR

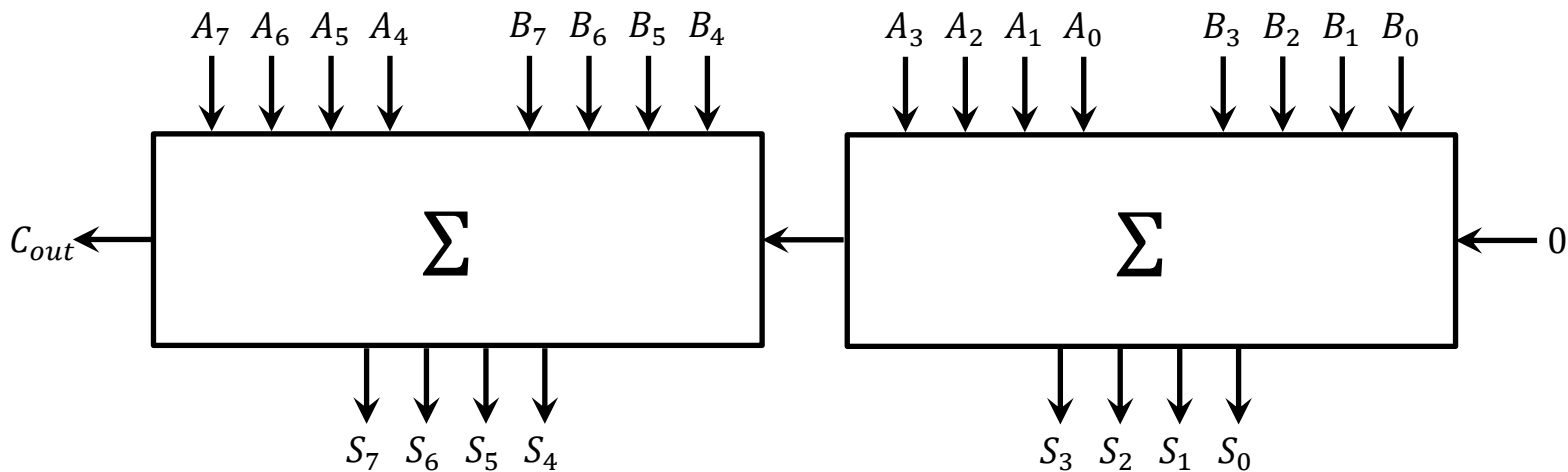
# Circuito Somador

## *Somador de 4 Bits*



# Circuito Somador

## *Somador de 8 Bits*



# Integrando Somador/Subtrator

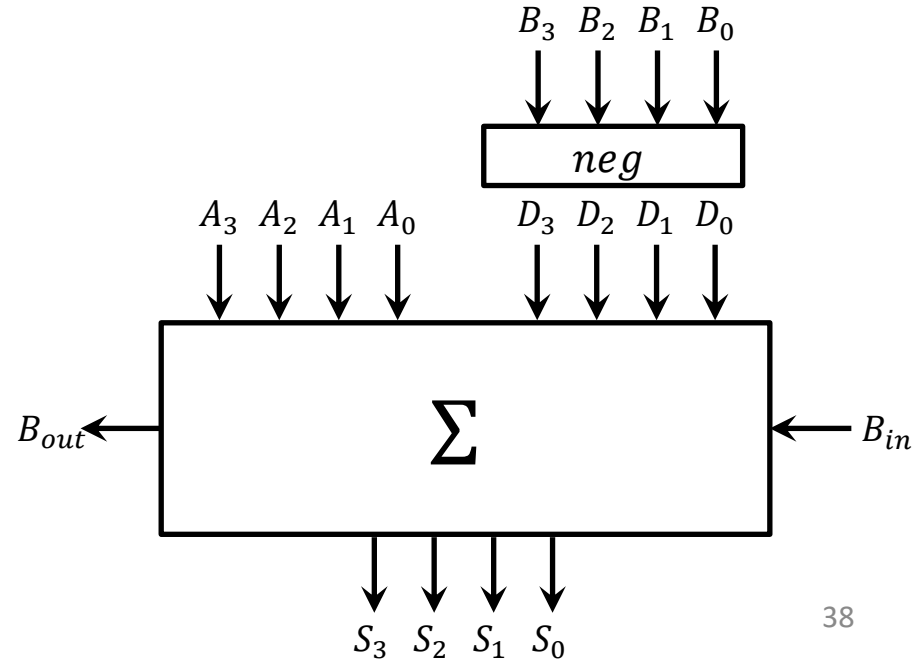
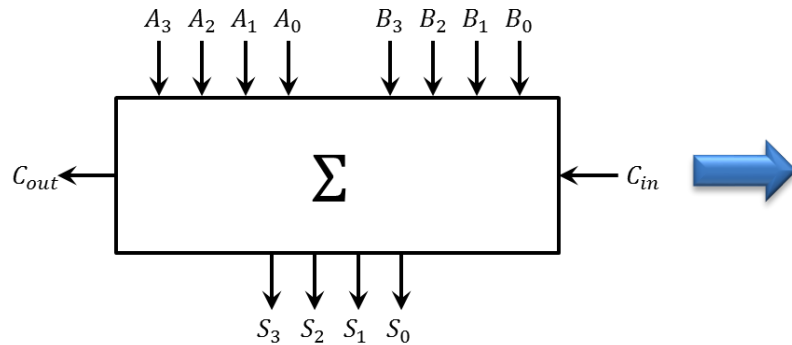
## Somador de 4 Bits

É possível implementar o *SUBTRATOR* com o *SOMADOR*?

$$S[3 \dots 0] = A[3 \dots 0] - B[3..0]$$



$$S[3 \dots 0] = A[3 \dots 0] + (-B[3..0])$$



# Circuito Subtrator

## *Subtrator de 4 Bits*

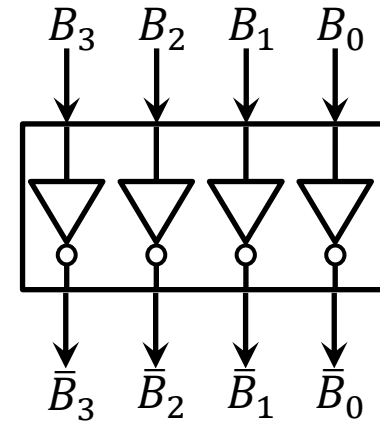
É possível implementar o *SUBTRATOR* com o *SOMADOR*?

$$D[3 \dots 0] = -B[3..0]$$

A transformação deverá converter um inteiro sem sinal em seu equivalente negativo em  $C - 2$ .

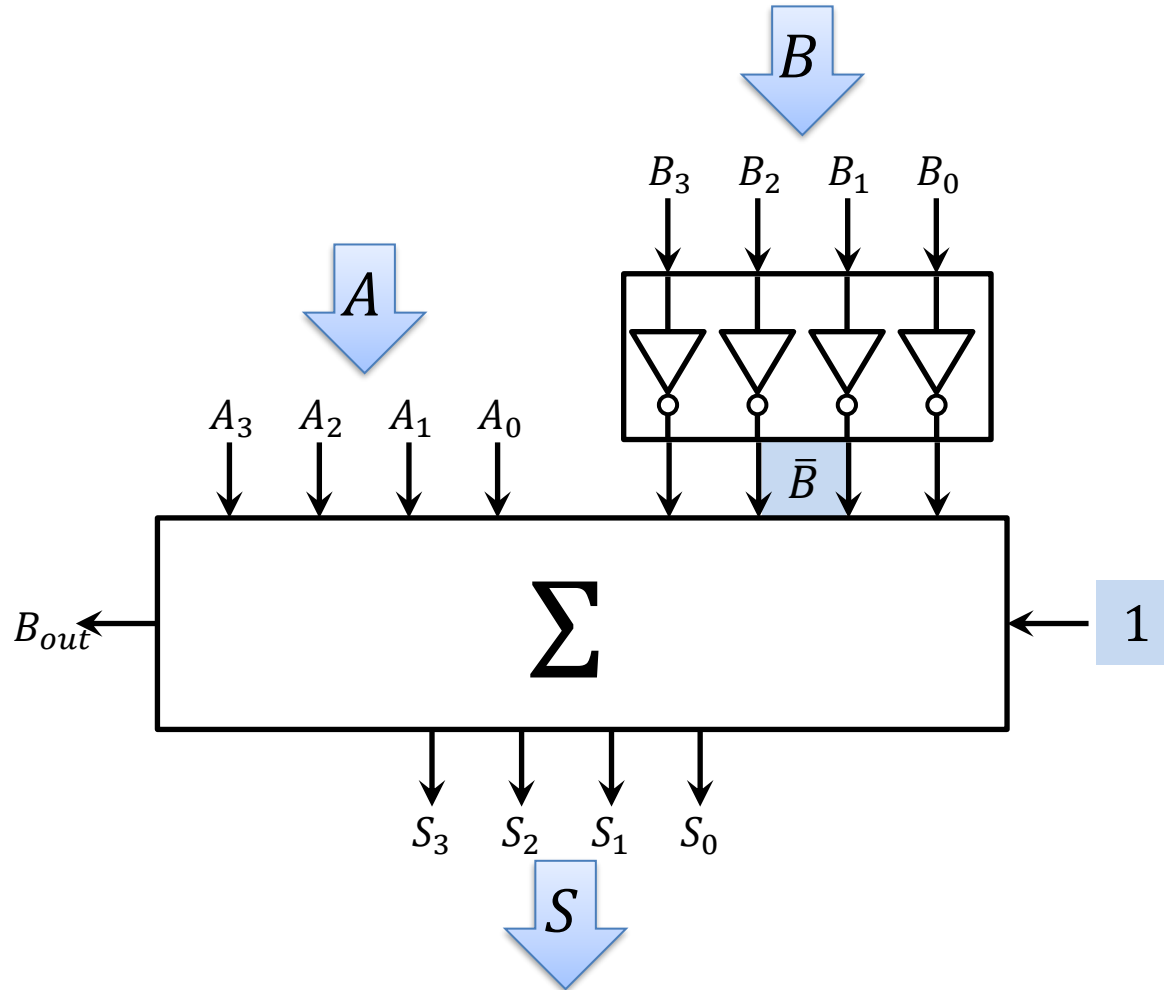
Exemplo: 5 em  $-5$ .

$$\begin{array}{r} B = 0101 \\ \bar{B} = 1010 \\ + \quad 1 \\ \hline -B = 1011 \end{array}$$



# Circuito Subtrator

## *Subtrator de 4 Bits*



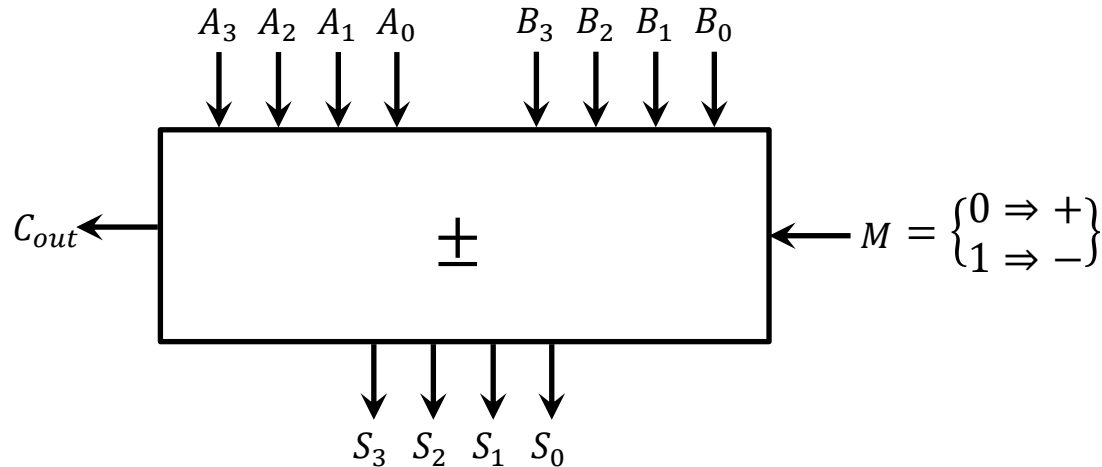
$$\bar{B} + 1 = -B$$

Em  $C - 2$

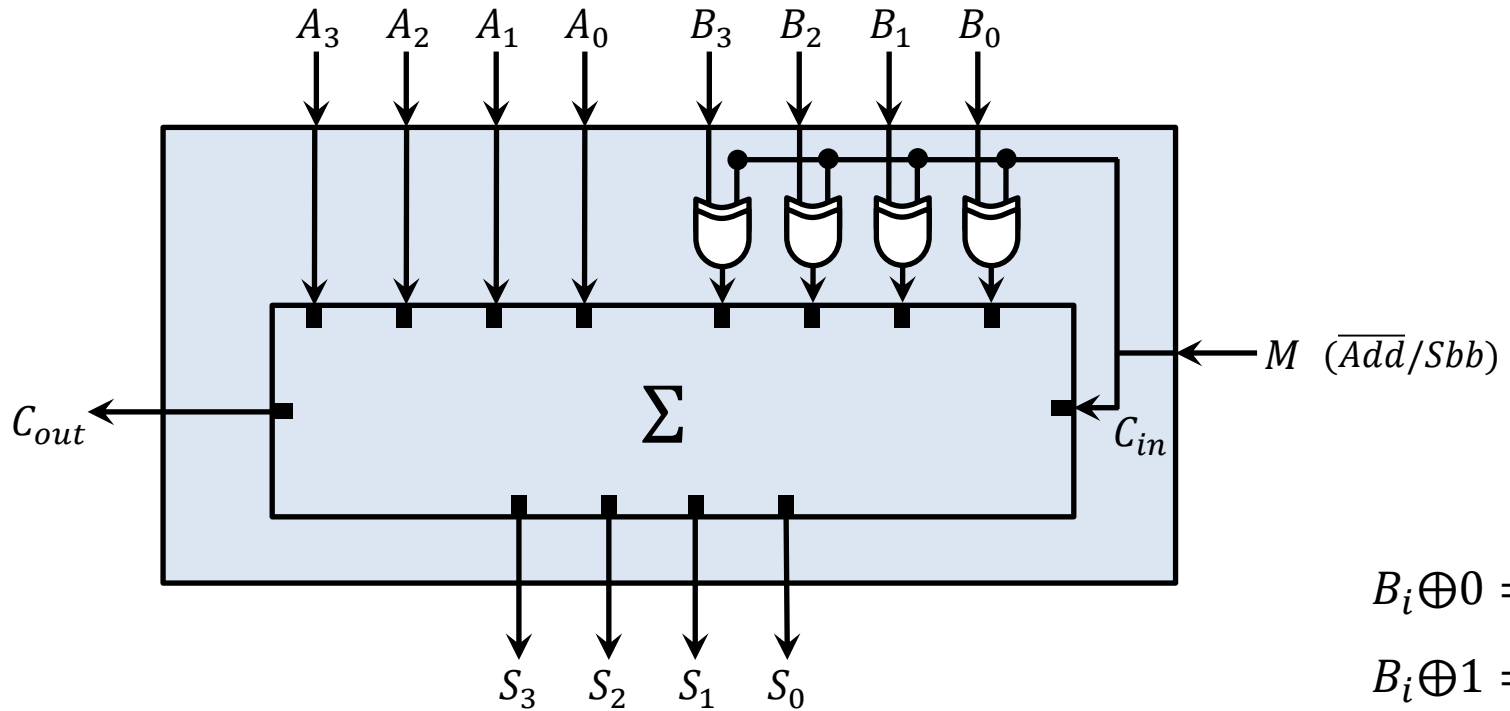


# Circuito Somador/Subtrator

Podemos pensar em um circuito que integra a *SOMA* e *SUBTRAÇÃO*, acrescentando um *flag*  $M$  que seleciona a operação desejada.



# Circuito Somador/Subtrator

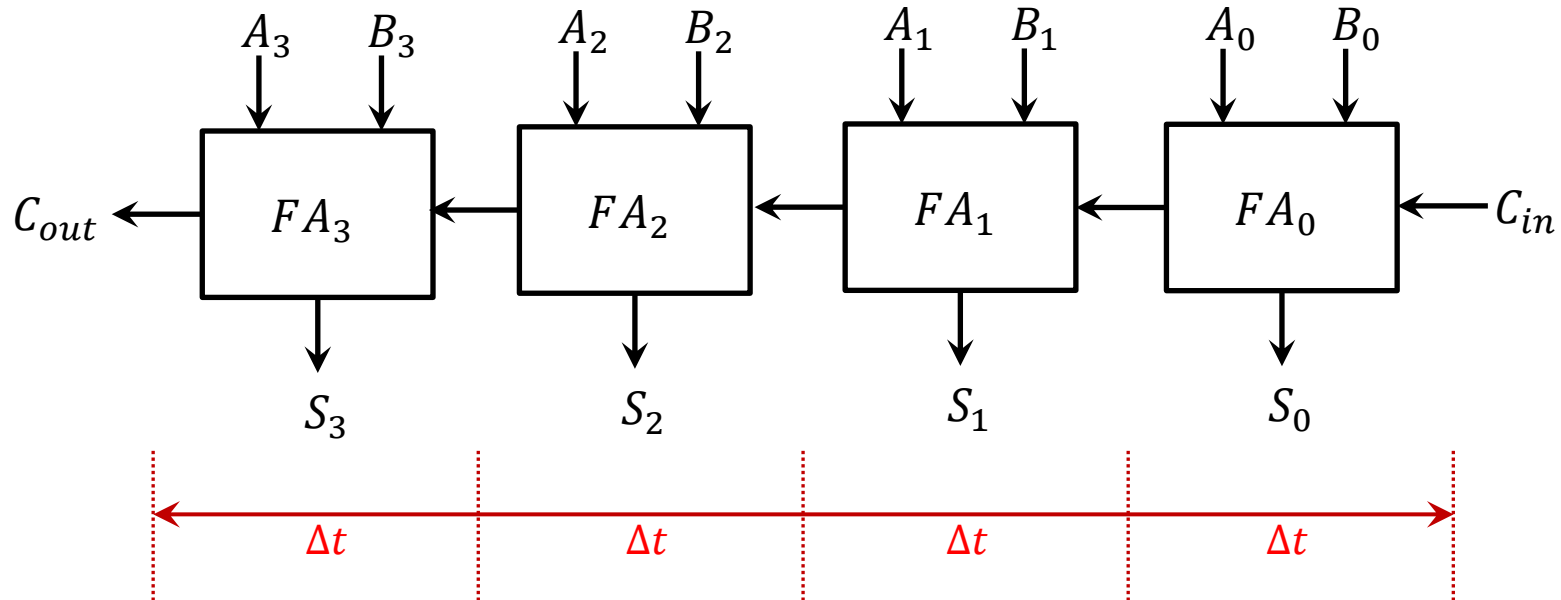


# OTIMIZAÇÃO DO SOMADOR

# Circuito Somador/Subtrator

## Otimização

- O circuito *SOMADOR* forma um importante componente da Unidade de Aritmética e Lógica;
- Assim, é importante avaliar a possibilidade de otimização da lógica utilizada.



# Circuito Somador/Subtrator

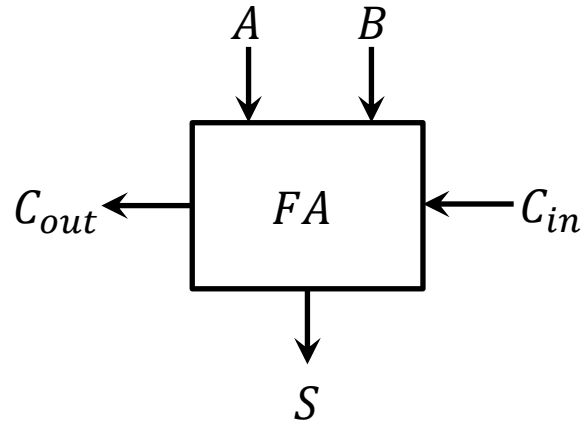
## Otimização

- Nesta arquitetura, denominada *Ripple Carry Adder* (Somador com Propagação de *Carry*), cada estágio consome um tempo  $\Delta t$ , em função da dependência do *Carry* produzido pelo estágio anterior;
- A estratégia é buscar uma forma de obtenção do *Carry* em paralelo com cálculo da soma;
- A nova arquitetura, denominada *Look-Ahead Carry Adder* (Somador com Antecipação do *Carry*), implementa uma lógica que calcula o *Carry* final em paralelo com as somas de cada estágio;

# Circuito Somador/Subtrator

## Otimização

- Observando um estágio do *SOMADOR*, temos:



- O  $C_{out}$  será 1 em uma destas situações:
  - ❑ Quando  $A$  e  $B$  forem 1; ou
  - ❑ Quando  $A$  ou  $B$  for 1, sendo  $C_{in}$  também 1;
- Assim, duas variáveis de apoio são criadas:
  - ❑ *Carry generation*:  $C_g = AB$
  - ❑ *Carry propagation*:  $C_p = A + B$

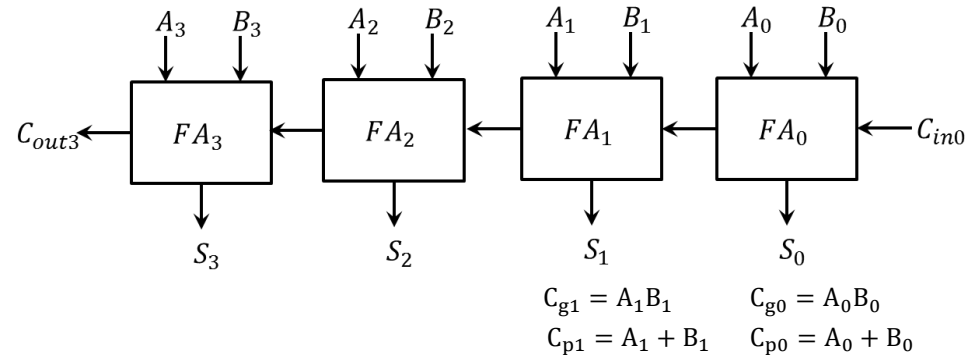


$$C_{out} = C_g + C_p C_{in}$$

# Circuito Somador/Subtrator

## Otimização

Para cada estágio, começando em  $FA_0$ , serão calculados os respectivos  $C_{out}$ , até alcançarmos o  $C_{out}$  final.



- Estágio  $FA_0$

$$C_{out0} = C_{g0} + C_{p0}C_{in0}$$

- Estágio  $FA_1$

$$C_{in1} = C_{out0}$$

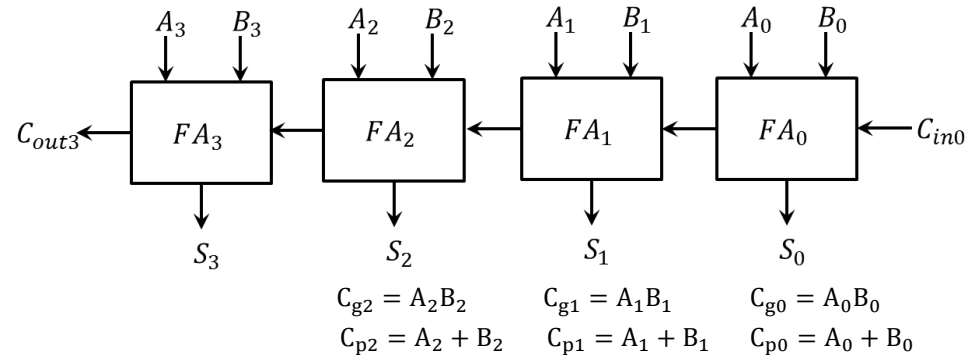
$$C_{out1} = C_{g1} + C_{p1}C_{in1} = C_{g1} + C_{p1}C_{out0} = C_{g1} + C_{p1}(C_{g0} + C_{p0}C_{in0})$$

$$C_{out1} = C_{g1} + C_{p1}C_{g0} + C_{p1}C_{p0}C_{in0}$$

# Circuito Somador/Subtrator

## Otimização

Para cada estágio, começando em  $FA_0$ , serão calculados os respectivos  $C_{out}$ , até alcançarmos o  $C_{out}$  final.



- Estágio  $FA_2$

$$C_{in2} = C_{out1}$$

$$C_{out1} = C_{g1} + C_{p1}C_{g0} + C_{p1}C_{p0}C_{in0}$$

$$C_{out2} = C_{g2} + C_{p2}C_{in2} = C_{g2} + C_{p2}C_{out1} = C_{g2} + C_{p2}(C_{g1} + C_{p1}C_{g0} + C_{p1}C_{p0}C_{in0})$$

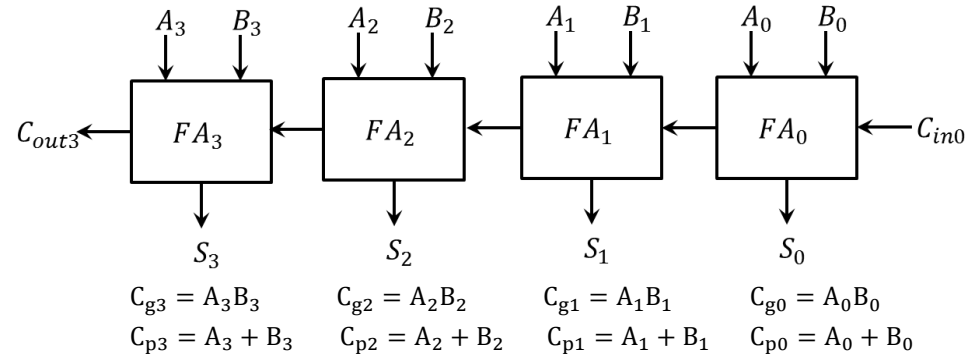
$$C_{out2} = C_{g2} + C_{p2}C_{g1} + C_{p2}C_{p1}C_{g0} + C_{p2}C_{p1}C_{p0}C_{in0}$$



# Circuito Somador/Subtrator

## Otimização

Para cada estágio, começando em  $FA_0$ , serão calculados os respectivos  $C_{out}$ , até alcançarmos o  $C_{out}$  final.



- Estágio  $FA_3$

$$C_{in3} = C_{out2}$$

$$C_{out2} = C_{g2} + C_{p2}C_{g1} + C_{p2}C_{p1}C_{g0} + C_{p2}C_{p1}C_{p0}C_{in0}$$

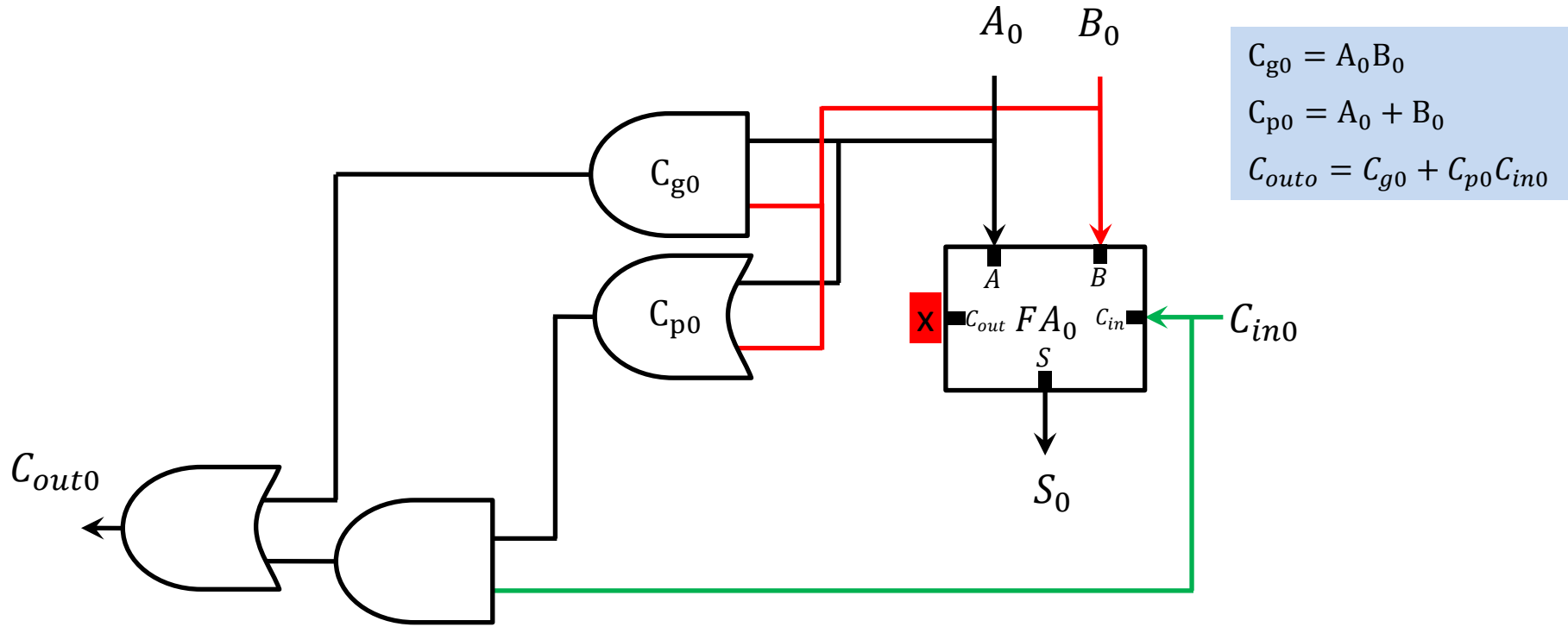
$$C_{out3} = C_{g3} + C_{p3}C_{in3} = C_{g3} + C_{p3}C_{out2}$$

$$C_{out3} = C_{g3} + C_{p3}(C_{g2} + C_{p2}C_{g1} + C_{p2}C_{p1}C_{g0} + C_{p2}C_{p1}C_{p0}C_{in0})$$

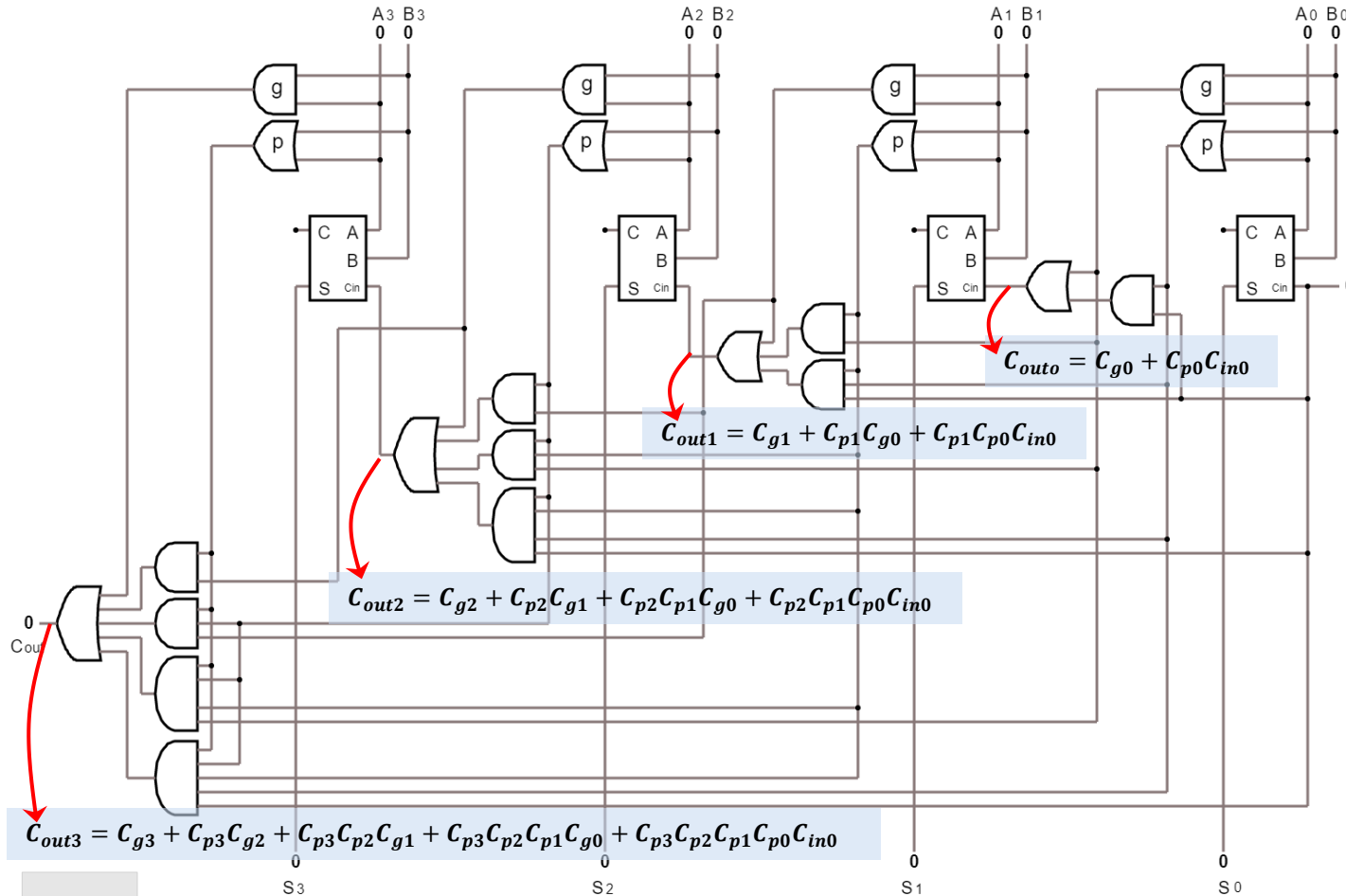
$$C_{out3} = C_{g3} + C_{p3}C_{g2} + C_{p3}C_{p2}C_{g1} + C_{p3}C_{p2}C_{p1}C_{g0} + C_{p3}C_{p2}C_{p1}C_{p0}C_{in0}$$

# Circuito Somador/Subtrator

## *Look-Ahead Carry*



# Circuito Somador *Look-Ahead Carry*



# Circuito Somador/Subtrator

## *Look-Ahead Carry*

