

# Introdução aos Sistemas Digitais

## *Circuitos aritméticos*



Introdução aos Sistemas Digitais, Ioulia Skliarova



## Representação de números negativos

Represente o número  $-17_{10}$  em **sinal e módulo**, **complemento para 1** e **complemento para 2** com 8 bits.



Introdução aos Sistemas Digitais



## Adição de números

É bastante difícil construir um circuito digital que some dois números representados em **sinal e módulo** dado que é necessário comparar as magnitudes dos operandos para determinar o sinal do resultado.

Números em **complemento para 1** podem ser adicionados aplicando regras habituais de adição binária. Transportes para além do bit mais significativo devem ser somados ao resultado (para evitar que o zero seja contado duas vezes).

Números em **complemento para 2** podem ser adicionados aplicando regras habituais de adição binária ignorando transportes para além do bit mais significativo.

**Overflow** ocorre se a soma de dois números positivos produzir um resultado negativo, ou se a soma de dois números negativos produzir um resultado positivo.

Em complemento para 2 **overflow** ocorre se no bit mais significativo  $C_{in} \neq C_{out}$ .



Introdução aos Sistemas Digitais



## Subtração de números em complemento para 2

Números em complemento para 2 podem ser subtraídos complementando o segundo operando e realizando a operação de soma:

$$A - B = A + (-B)$$

**Exemplos:**

$$2 - 3 = 2 + (-3) = -1$$

$$-5 - 6 = -5 + (-6) = -11$$

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
-8	1	0	0	0
-7	1	0	0	1
-6	1	0	1	0
-5	1	0	1	1
-4	1	1	0	0
-3	1	1	0	1
-2	1	1	1	0
-1	1	1	1	1

$$\begin{array}{r} 000 \\ 0010 \\ + 1101 \\ \hline 1111 \end{array}$$

$$\begin{array}{r} 1010 \\ 1011 \\ + 1010 \\ \hline 0101 \end{array}$$

← overflow

Para somar e subtrair números em complemento para 2 precisamos de apenas um **circuito somador**.



Introdução aos Sistemas Digitais



## Circuitos somadores

Um **half adder** (**semi-somador**) soma dois operandos de 1 bit cada e produz um resultado de 2 bits que varia entre 0 e 2.

a	b	C <sub>out</sub>	hs
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$c_{out} = a \cdot b$$

$$hs = a \oplus b$$



Introdução aos Sistemas Digitais



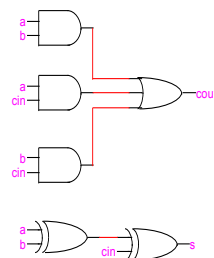
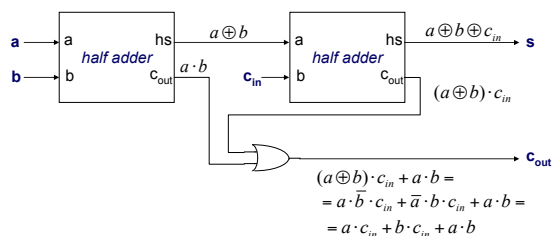
## Somador completo

Para somar operandos com mais que 1 bit temos que assegurar a transferência de transportes (*carries*) entre vários bits. Tal somador multi-bit pode ser construído à custa de **somadores completos** – **full adders**.

C <sub>in</sub>	a	b	C <sub>out</sub>	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$c_{out} = a \cdot b + a \cdot c_{in} + b \cdot c_{in}$$

$$s = a \oplus b \oplus c_{in}$$

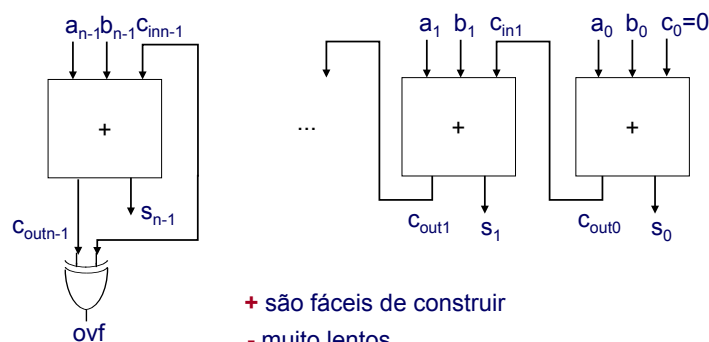


Introdução aos Sistemas Digitais



## Somadores em cascata (*ripple adders*)

Dois operandos binários de  $n$  bits podem ser somados com uma cascata de  $n$  somadores completos cada um dos quais calcula um bit do resultado.



**Circuito iterativo** – para calcular um resultado de  $n$  bits existem  $n$  módulos idênticos interligados em cascata em que cada um dos módulos “seguintes” faz cálculos com base nos resultados produzidos pelo módulo “anterior”.



Introdução aos Sistemas Digitais



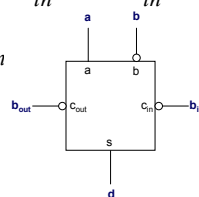
## Subtrator completo

Para subtrair operandos com mais que 1 bit temos que assegurar a transferência de transportes (*borrows*) entre vários bits. Tal subtrator multi-bit pode ser construído à custa de **subtratores completos**.

$b_{in}$	$a$	$b$	$b_{out}$	$d$
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

$$b_{out} = \bar{a} \cdot b + \bar{a} \cdot b_{in} + b \cdot b_{in}$$

$$d = a \oplus b \oplus b_{in}$$



Um somador completo pode realizar a operação de subtração  $a - b$  fazendo o cálculo de  $a + 1sc(b) + 1$  (dado que  $-b = 1sc(b) + 1$ ).

$$c_{out} = a \cdot b + a \cdot c_{in} + b \cdot c_{in} \quad \bar{b}_{out} = (a + \bar{b}) \cdot (a + \bar{b}_{in}) \cdot (\bar{b} + \bar{b}_{in}) =$$

$$s = a \oplus b \oplus c_{in} \quad = a \cdot \bar{b} + a \cdot \bar{b}_{in} + \bar{b} \cdot \bar{b}_{in}$$

$$d = a \oplus \bar{b} \oplus \bar{b}_{in}$$



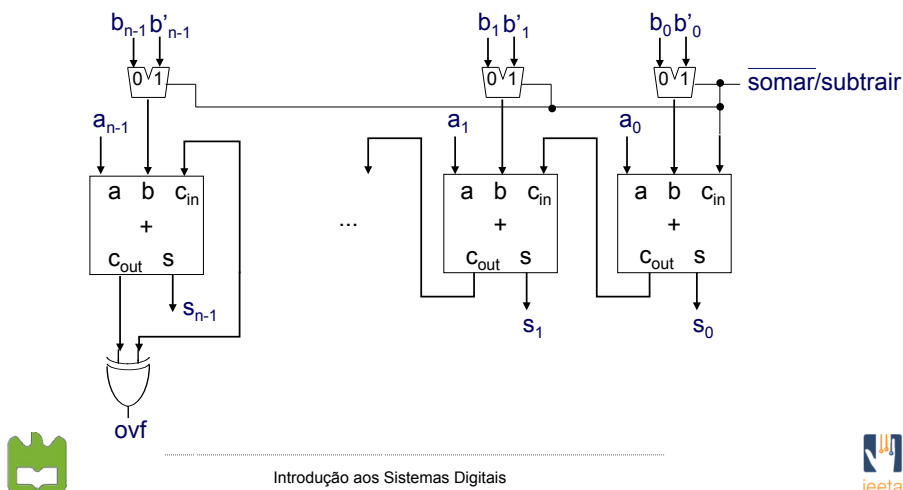
Introdução aos Sistemas Digitais



## Somador/subtrator em cascata

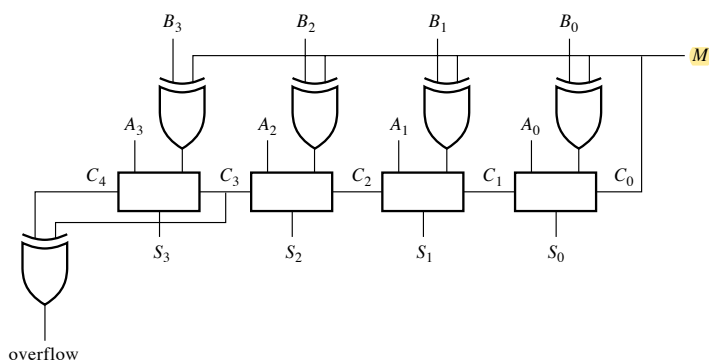
Para realizar a operação de **soma** devemos fornecer nas entradas **a**, **b**, e  $c_{in0} = 0$ .

Para realizar a operação de **subtração** devemos fornecer nas entradas **a**, e **b'**, e  $c_{in0} = 1$ .



## Somador/subtrator em cascata

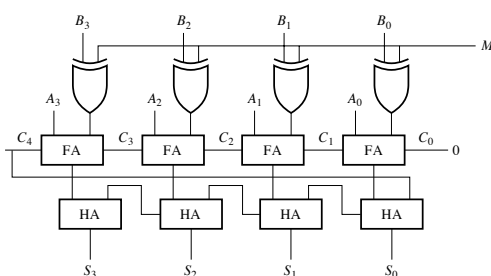
Implementação alternativa



## Exercícios

É possível construir um circuito que faça a soma de dois valores de 2 bits em complemento para 2 só com **2 níveis de atraso**?

Compare o circuito abaixo com o do slide anterior. Comprove que faz a soma de dois valores de 4 bits em complemento para 1 (a partir de somadores completos). Acrescente a lógica de detecção de overflow



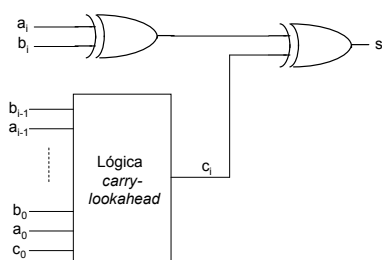
Introdução aos Sistemas Digitais



## Somadores *carry lookahead*

$$s_i = a_i \oplus b_i \oplus c_i$$

$$c_{i+1} = a_i \cdot b_i + a_i \cdot c_i + b_i \cdot c_i$$



Na fase  $i$  é **gerado um carry** se para alguma combinação de  $a_i$  e  $b_i$  é produzido  $c_{i+1}=1$ , independentemente das entradas  $a_0, \dots, a_{i-1}$ ,  $b_0, \dots, b_{i-1}$  e  $c_0$ .

**g** – sinal de **geração de carry**

$$g_i = a_i \cdot b_i$$

Na fase  $i$  é **propagado um carry** se é produzido  $c_{i+1}=1$  na presença de tal combinação de entradas  $a_0, \dots, a_{i-1}$ ,  $b_0, \dots, b_{i-1}$  e  $c_0$  que causam  $c_i = 1$ .

**p** – sinal de **propagação de carry**

$$p_i = a_i + b_i$$

$$c_{i+1} = g_i + p_i \cdot c_i$$



Introdução aos Sistemas Digitais



## Somadores *carry lookahead* (cont.)

Para um somador de 4 bits:

$$c_1 = g_0 + p_0 \cdot c_0$$

$$c_2 = g_1 + p_1 \cdot c_1 = g_1 + p_1 \cdot (g_0 + p_0 \cdot c_0) = g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot c_0$$

$$c_3 = g_2 + p_2 \cdot c_2 = g_2 + p_2 \cdot (g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot c_0) = \\ = g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot c_0$$

$$c_4 = g_3 + p_3 \cdot c_3 = g_3 + p_3 \cdot (g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot c_0) = \\ = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 + p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_0$$

Todos os sinais de transporte (*carry*) são calculados só com 3 níveis de atraso.

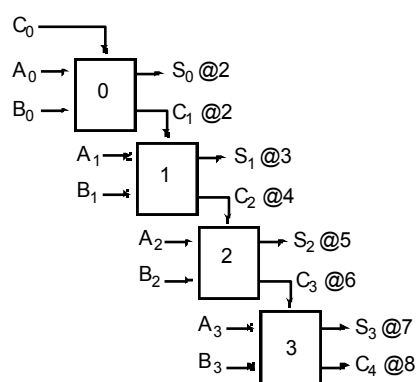


Introdução aos Sistemas Digitais



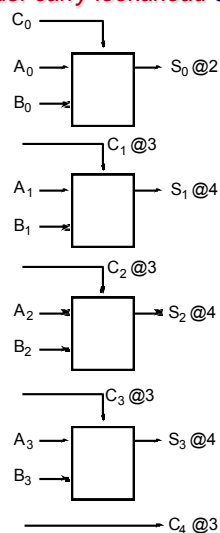
## Comparação de somadores

Somador *ripple* de 4 bits



No máximo 8 níveis de atraso

Somador *carry-lookahead* de 4 bits



No máximo 4 níveis de atraso

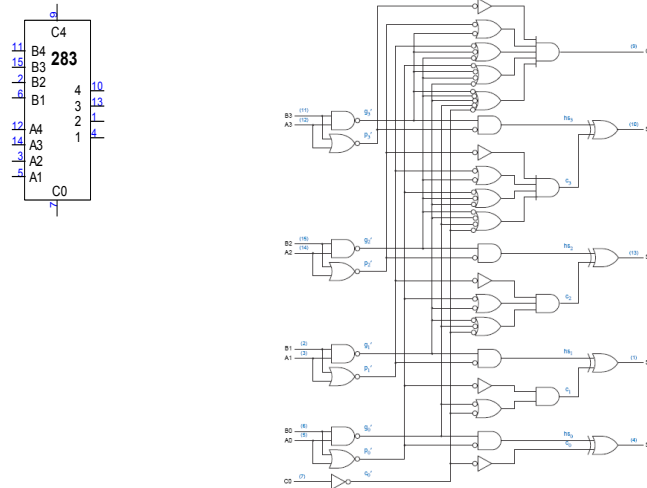


Introdução aos Sistemas Digitais



## Somadores comerciais

74x283 – somador *carry-lookahead* de 4 bits



Introdução aos Sistemas Digitais



## Somadores BCD

Se somar 2 dígitos BCD (e *carry*) pode-se obter uma soma que varia de 0 a 19

soma	binário		BCD	
	carry out	soma	carry out	soma
0	0	0000	0	0000
1	0	0001	0	0001
2	0	0010	0	0010
3	0	0011	0	0011
4	0	0100	0	0100
5	0	0101	0	0101
6	0	0110	0	0110
7	0	0111	0	0111
8	0	1000	0	1000
9	0	1001	0	1001
10	0	1010	1	0000
11	0	1011	1	0001
12	0	1100	1	0010
13	0	1101	1	0011
14	0	1110	1	0100
15	0	1111	1	0101
16	1	0000	1	0110
17	1	0001	1	0111
18	1	0010	1	1000
19	1	0011	1	1001

Casos que precisam de correção

Somar 6

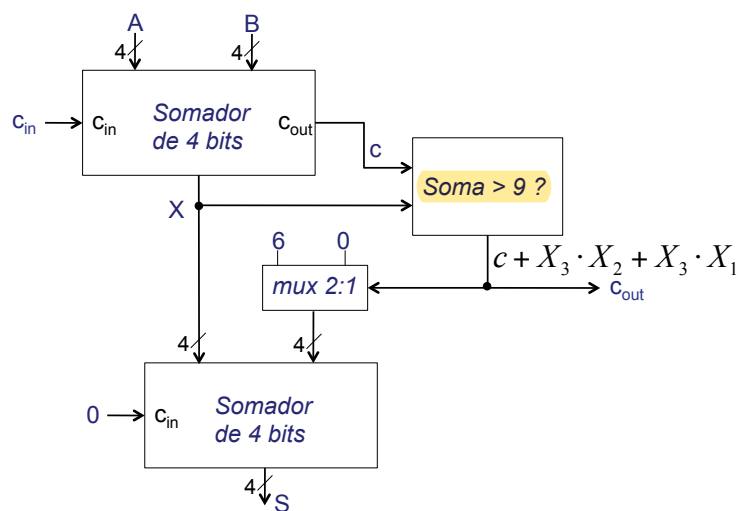


Introdução aos Sistemas Digitais





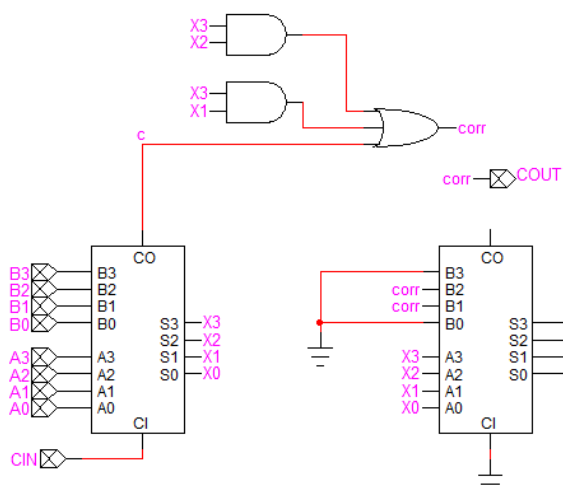
## Somadores BCD (cont.)



Introdução aos Sistemas Digitais



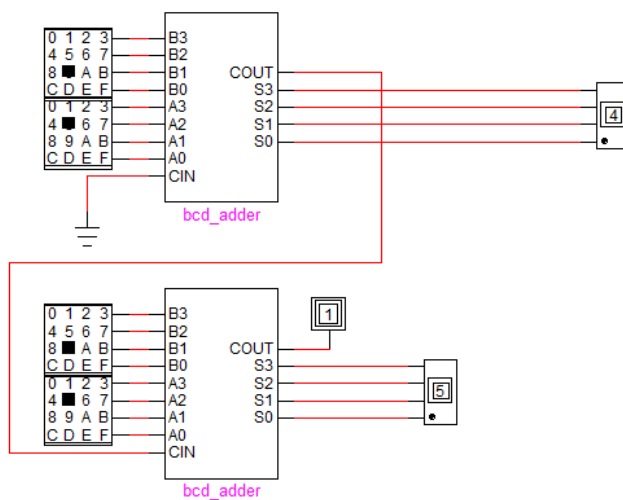
## Somadores BCD (cont.)



Introdução aos Sistemas Digitais



## Somadores BCD em cascata

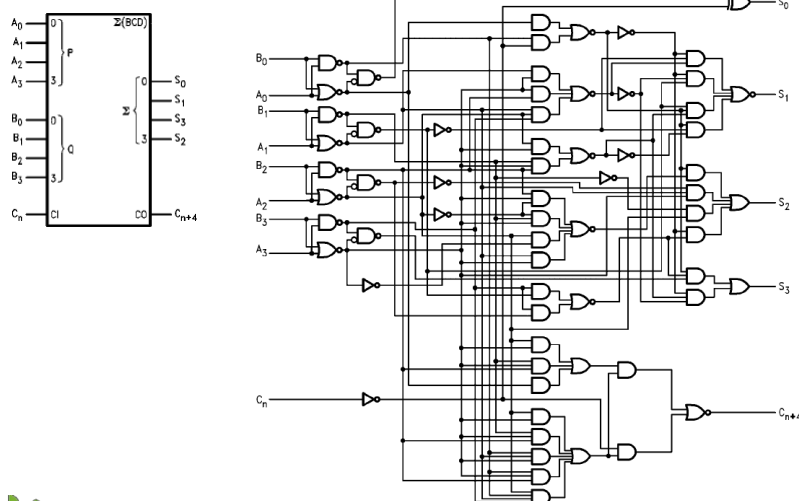


Introdução aos Sistemas Digitais



## Somadores BCD comerciais

**74x583** – somador BCD (com *carry-lookahead*) de 4 bits



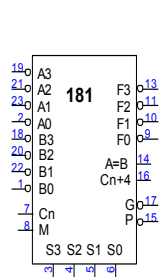
Introdução aos Sistemas Digitais



## Unidades aritméticas e lógicas

Uma **unidade aritmética e lógica (ALU – Arithmetic and Logic Unit)** é um dispositivo combinatório que executa qualquer operação aritmética ou lógica (de um conjunto predefinido) sobre um par de operandos de **b** bits.

A operação a executar é especificada com entradas de seleção de função.



S3	S2	S1	S0	M=0 (op. aritm.)	M=1 (op. lógica)
0	0	0	0	$F = A - 1 + \text{CIN}$	$F = \bar{A}$
0	0	0	1	$F = A \text{ AND } B - 1 + \text{CIN}$	$F = \bar{A} \text{ OR } B$
0	0	1	0	$F = A \text{ AND } \bar{B} - 1 + \text{CIN}$	$F = \bar{A} \text{ OR } \bar{B}$
0	0	1	1	$F = 1111 + \text{CIN}$	$F = 1111$
0	1	0	0	$F = A + (A \text{ OR } \bar{B}) + \text{CIN}$	$F = \bar{A} \text{ AND } \bar{B}$
0	1	0	1	$F = A \text{ AND } B + (A \text{ OR } \bar{B}) + \text{CIN}$	$F = \bar{B}$
0	1	1	0	$F = A - B - 1 + \text{CIN}$	$F = A \text{ XOR } B$
0	1	1	1	$F = A \text{ OR } \bar{B} + \text{CIN}$	$F = A \text{ OR } \bar{B}$
1	0	0	0	$F = A + (A \text{ OR } B) + \text{CIN}$	$F = \bar{A} \text{ AND } B$
1	0	0	1	$F = A + B + \text{CIN}$	$F = A \text{ XOR } B$
1	0	1	0	$F = A \text{ AND } \bar{B} + (A \text{ OR } B) + \text{CIN}$	$F = B$
1	0	1	1	$F = A \text{ OR } B + \text{CIN}$	$F = A \text{ OR } B$
1	1	0	0	$F = A + A + \text{CIN}$	$F = 0000$
1	1	0	1	$F = A \text{ AND } B + A + \text{CIN}$	$F = A \text{ AND } B$
1	1	1	0	$F = A \text{ AND } \bar{B} + A + \text{CIN}$	$F = A \text{ AND } B$
1	1	1	1	$F = A + \text{CIN}$	$F = A$



Introdução aos Sistemas Digitais



## Multiplicação de números sem sinal

Os processos de multiplicação no sistema binário obedecem às mesmas regras básicas existentes no sistema decimal.

**Exemplo:**

$$12 \times 13 = 156$$

	1	1	0	0	← multiplicando
×	1	1	0	1	← multiplicador
<hr/>					
		1	1	0	0
		0	0	0	0
	1	1	0	0	
1	1	0	0		
<hr/>					
1	0	0	1	1	1
					0

produtos parciais



Introdução aos Sistemas Digitais





## Exercícios

Calcule o resultado das operações seguintes em complemento para 2 com 4 bits de representação:

$$5 \times (-6)$$

$$-7 \times (-8)$$

