

PCS 3115

Sistemas Digitais I

Códigos

Prof. Dr. Marcos A. Simplicio Jr.

versão: 3.0 (Jan/2016)

Adaptado por Glauber (2018)

CÓDIGOS NUMÉRICOS

Códigos Numéricos

- Conjunto de cadeias com n bits: cadeias diferentes representam coisas diferentes;
 - Para n bits existem 2^n códigos válidos de comprimento fixo
- Exemplo: códigos numéricos
 - Computadores: operam com bits
 - Humanos: preferem operar com decimais
 - → Códigos permitem codificar 10 símbolos decimais (0-9) em termos de bits

Códigos Numéricos

- **Pergunta 1:** quantos bits são necessários para representar 10 dígitos (símbolos)?
 - Resposta: 4 bits → 16 possibilidades (6 códigos não são usados)
- **Pergunta 2:** Quantos diferentes códigos numéricos (i.e., para 10 dígitos) são possíveis de serem criados com esse número de bits?
 - Resposta: $16!/(16-10)! = 16*15*...*7 \sim 29$ milhões
 - Corresponde a um arranjo simples (ordem dos elementos é considerada)
- **Pergunta 3:** Qual o código numérico mais imediato e fácil de ser compreendido ?
 - Resposta: código que representa os dígitos (símbolos) de 0 a 9 através da representação em binário (4 bits) dos números de 0 a 9
Nome dado: BCD (*binary-coded decimal*) ou “8421”.

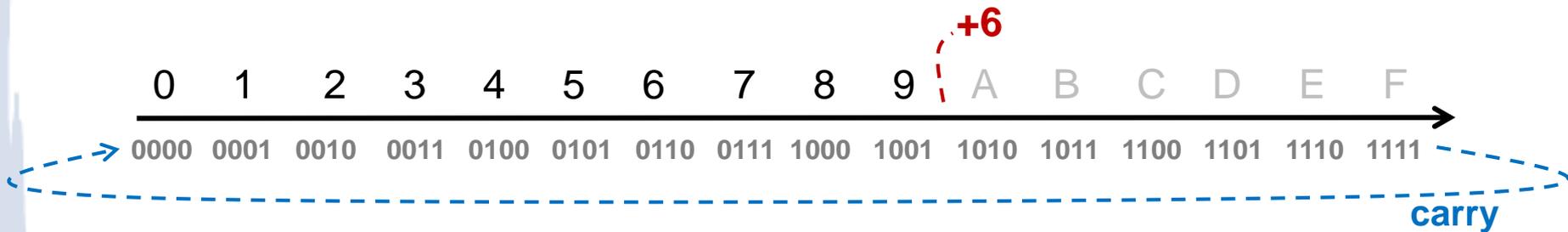
BCD – Binary Coded Decimal

- Cadeias de 4 bits;
 - Código 8421: indica peso de cada um dos bits na cadeia
- Variantes
 - *Packed BCD*: 2 dígitos BCD em 1 byte → 00 a 99;
 - BCD com sinal: um dígito extra representando “+” ou “–”

Dígito decimal	BCD (8421)	Dígito decimal	BCD (8421)
0	0000	8	1000
1	0001	9	1001
2	0010	não usado	1010
3	0011		1011
4	0100		1100
5	0101		1101
6	0110		1110
7	0111		1111

BCD – Binary Coded Decimal

- **Adição:** similar à de números binários de 4 bits
 - Mas com correção se resultado ultrapassar 1001: somar 6



- Exemplos:

5	0101		8	1000		4	0100
<u>+ 9</u>	<u>+ 1001</u>		<u>+ 8</u>	<u>+ 1000</u>		<u>+ 5</u>	<u>+ 0101</u>
14	1110	> 1001	16	1 0000	> 1001	9	1001
	<u>+ 0110</u>			<u>+ 0110</u>			
10+4	1 0100		10+6	1 0110			

A red dashed arrow labeled 'correção' points from the 'correção' text to the '> 1001' boxes in the third row.

Conversão Binário-BCD

- Passamos pra decimal: $1011_2 = 11_{10} = 00010001$
- Algoritmo **Double-Dabble**. Ex: 11110011 em BCD?

0000	0000	0000	11110011	Initialization
0000	0000	0001	11100110	Shift
0000	0000	0011	11001100	Shift
0000	0000	0111	10011000	Shift
0000	0000	1010	10011000	Add 3 to ONES, since it was 7
0000	0001	0101	00110000	Shift
0000	0001	1000	00110000	Add 3 to ONES, since it was 5
0000	0011	0000	01100000	Shift
0000	0110	0000	11000000	Shift
0000	1001	0000	11000000	Add 3 to TENS, since it was 6
0001	0010	0001	10000000	Shift
0010	0100	0011	00000000	Shift
2	4	3		

Código 2421

- Código **ponderado**: Nome indica peso de cada um dos bits na cadeia
 - Ex.: $5 = 1011$, pois $2*1+4*0+2*1+1+1 = 5$
- **Propriedade: auto-complementar**
 - Ao inverter os bits do código de um determinado dígito, obtém-se o código do complemento de 9 daquele dígito
 - Ex.: $\text{Compl}_9(8) = 1 \rightarrow$ código de 8 é 1110 (o inverso de 0001)

Dígito decimal	2421	Dígito decimal	2421
0	0000	9	1111
1	0001	8	1110
2	0010	7	1101
3	0011	6	1100
4	0100	5	1011
Não usado (mas poderia ser)	0101 (5) 0110 (6) 0111 (7)	Não usado (mas poderia ser)	1010 (4) 1001 (3) 1000 (2)

Código Excesso-3

- Equivalente a BCD + 3
- **Propriedade: auto-complementar**
 - Ex.: $\text{Compl}_9(8) = 1 \rightarrow$ código de 8 é 1110 (o inverso de 0001)
- Vantagem sobre 2421: aritmética similar a BCD
 - **Lição de casa:** pesquisar/deduzir como fazer

Dígito decimal	Excesso-3	Dígito decimal	Excesso-3
Não usado	0000	5	1000
	0001	6	1001
	0010	7	1010
0	0011	8	1011
1	0100	9	1100
2	0101	Não usado	1101
3	0110		1110
4	0111		1111

Código 2 entre 5

- Código de 5 bits:
 - Usam-se todos os códigos em que (número de bits 1) = 2
- **Propriedade:** correção de erros
 - Em uma transmissão digital, sempre se esperam dois bits '1': se houver um erro na transmissão e um bit for invertido, erro é facilmente detectado.

Dígito decimal	2 entre 5	Dígito decimal	2 entre 5
0	00011	6	10001
1	00101	7	10010
2	00110	8	10100
3	01001	9	11000
4	01010	Não usados: todos os códigos que não tenham dois bits 1	
5	01100		

Código 1 entre n

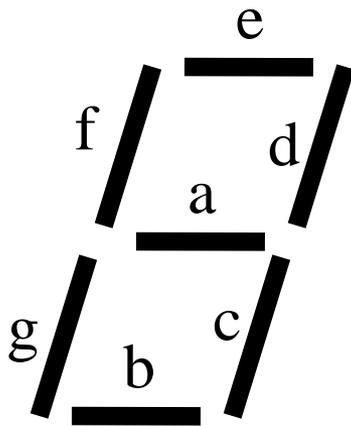
- Código de n bits:
 - A posição do bit indica o valor representado
- **Propriedade:** correção de erros; facilita seleção
 - Grande possibilidade de detectar inversão de bits
 - Apenas 10 códigos válidos dentre 1024 possíveis
 - Sinal de habilitação de circuito de seleção pode ser ligado diretamente a código

Dígito decimal	1 entre 10	Dígito decimal	1 entre 10
0	0000000001	5	0000100000
1	0000000010	6	0001000000
2	0000000100	7	0010000000
3	0000001000	8	0100000000
4	0000010000	9	1000000000

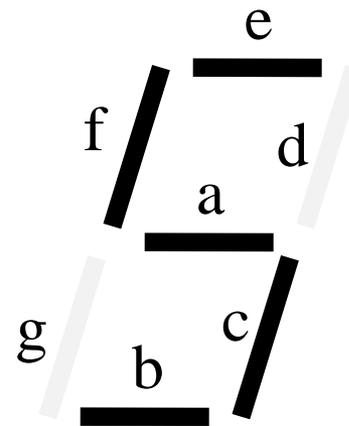
Não usados: todos os códigos que não tenham somente um bit 1

Código 7 Segmentos

- **Problema:** criar código para iluminar LEDs correspondentes em display de 7 segmentos
- Objetivo: evitar necessidade de decodificadores



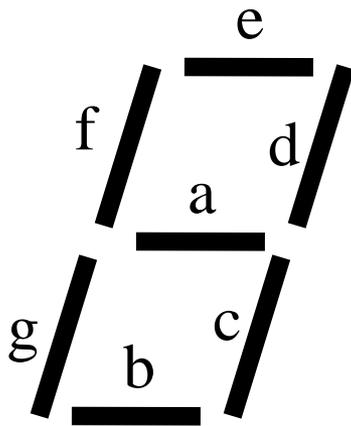
Exemplo: 5



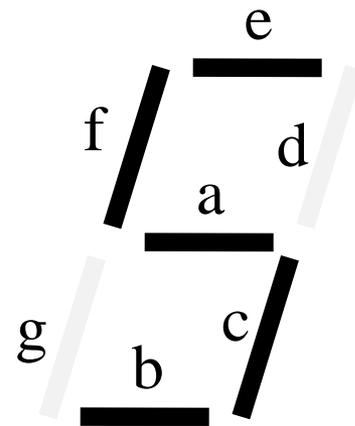
- Quantos bits? Qual o código para cada valor decimal?

Código 7 Segmentos

- Utilizado para iluminar LEDs correspondentes em display de 7 segmentos



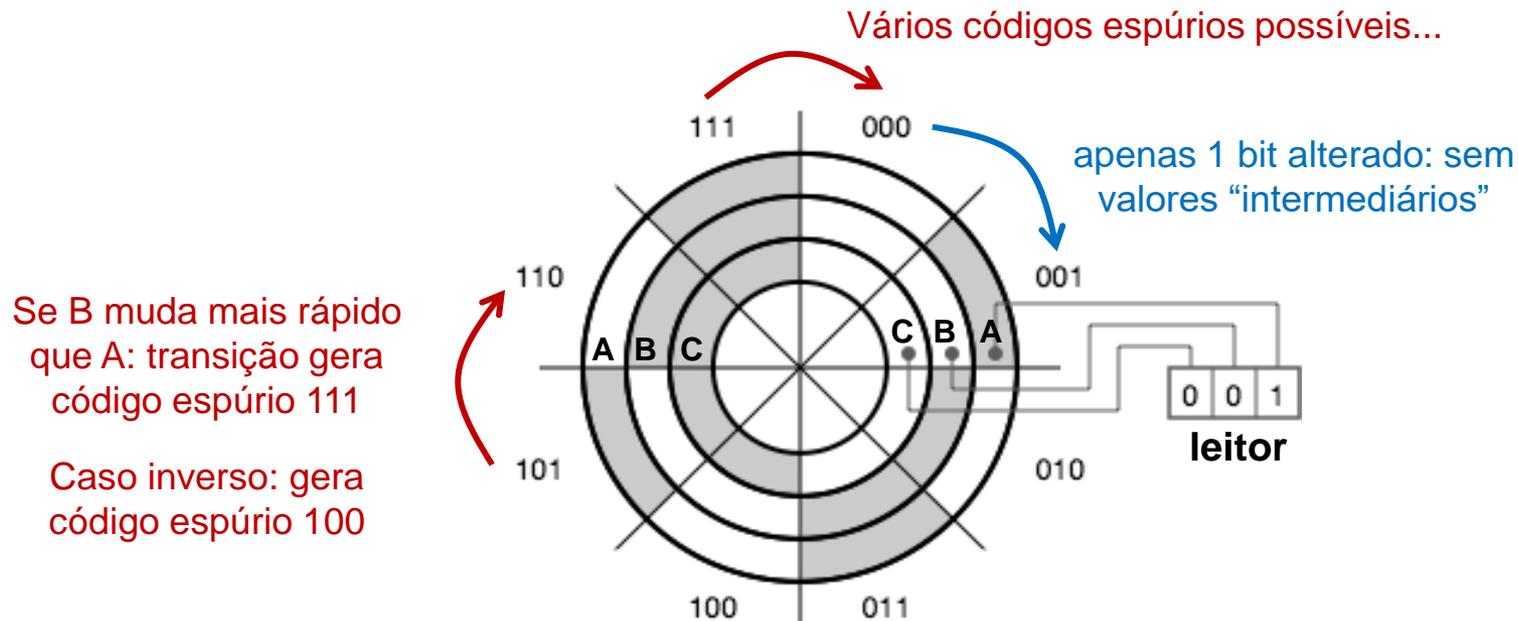
Exemplo: 5
→ 1110110
abcdefg



Dígito decimal	7 Segmentos	Dígito decimal	7 Segmentos
0	0111111	5	1110110
1	0011000	6	1110011
2	1101101	7	0011100
3	1111100	8	1111111
4	1011010	9	1011110

Código de Gray

- Cenário: aplicações eletromecânicas (ex.: copiadora, freio automotivo, etc.)
 - Valor digital no leitor indica posição mecânica
- **Problema:**
 - **Valor “na transição”** pode ser interpretado (incorretamente) como a posição atual



Código de Gray

- Propriedades:

- Apenas **um bit alterado** entre um código e seu sucessor

- Construção:

- 0 | 1

- 00 01 | 11 10

- 000 001 011 010 | 110 111 101 100

0s ↗

↖ 1s

← Parte-se de 0 e 1

← A 2a. metade é a sequência invertida

Dígito decimal	BCD	Gray	Dígito decimal	BCD	Gray
0	000	000	4	100	110
1	001	001	5	101	111
2	010	011	6	110	101
3	011	010	7	111	100

Código de Gray

- Propriedades:
 - Apenas **um bit alterado** entre um código e seu sucessor
- Conversão Binário-Gray
 - Enumere os bits da direita para a esquerda
 - Se bit $i = \text{bit}(i+1)$, então bit i em Gray é **0**; senão, é **1**. Manter bit mais significativo
- Equivalentemente: $X_{\text{gray}} = X_{\text{bin}} \text{ xor } (X_{\text{bin}} \gg 1)$

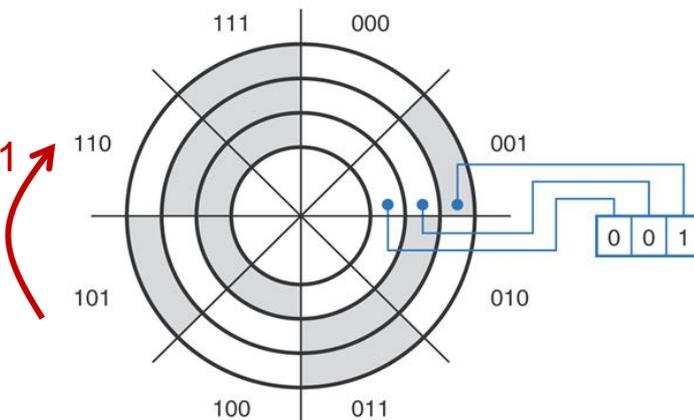
Dígito decimal	BCD	Gray	Dígito decimal	BCD	Gray
0	000	000	4	100	110
1	001	001	5	101	111
2	010	011	6	110	101
3	011	010	7	111	100

Código de Gray

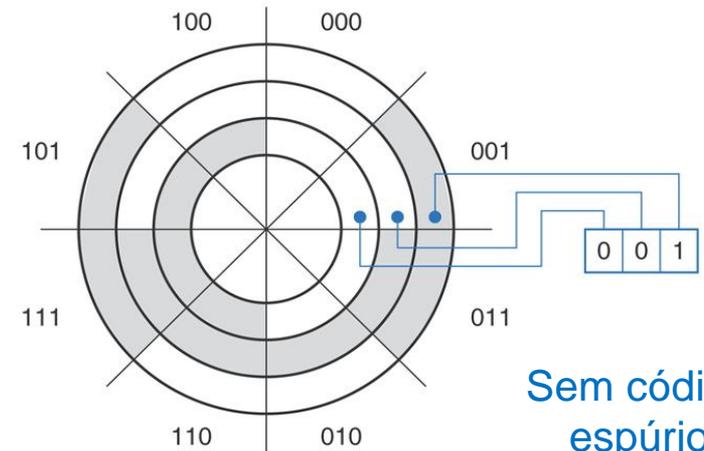
- Não aparecem valores espúrios na transição entre posições do disco



códigos espúrios:
100 ou 111



Codificação BCD



Codificação Gray

Código de Gray

- Patente US 2,632,058 – “Pulse Code Communication” (1953).
Inventor: Frank Gray, Bell Labs

Patented Mar. 17, 1953	2,632,058
UNITED STATES PATENT OFFICE	
2,632,058	
PULSE CODE COMMUNICATION	
Frank Gray, East Orange, N. J., assignor to Bell Telephone Laboratories, Incorporated, New York, N. Y., a corporation of New York	
Application November 13, 1947, Serial No. 785,697	
16 Claims. (Cl. 179—15)	
<p>1</p> <p>This invention relates to pulse code transmission and particularly to the coding of a message signal in a novel code and to the decoding thereof.</p>	<p>2</p> <p>der control of the signal to a particular aperture row and thereupon swept laterally along this row, a train of current pulses may be drawn from the collector whose location on the time scale is in accordance with the arrangement of the 1's and 0's in the binary number whose value is equal to the value of the signal sample being coded.</p>
<p>In communication by pulse code transmissions the instantaneous amplitudes of a message to be transmitted are successively sampled and each of the successive samples is translated into a code group of on-or-off pulses. By reason of the on-or-off character of the pulses, such a code is denoted a binary code. The number of pulse positions in a code group is the same from</p>	<p>5</p> <p>10</p> <p>It is a characteristic of the conventional binary number notation that a value change of unity may be reflected in the binary number notation by a simultaneous change in several of the</p>



Exercícios

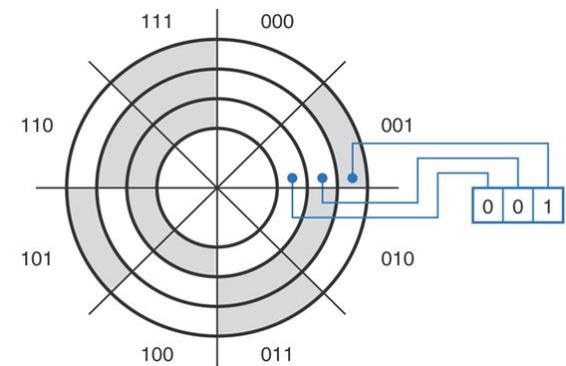
1) Converta os números decimais para BCD e 2421:

a) 742

b) 268

2) Some os números 742 e 268 em BCD

3) Quantas “fronteiras ruins” existem no disco de codificação binária de 3 bits?



4) Responda a questão 3 para um disco de n bits, em função de n

Exercícios: Respostas

1) Converta os números decimais para BCD e 2421:

a) **742** → BCD: 0111 0100 0010 2421: 1101 0100 0010

b) **268** → BCD: 0010 0110 1000 2421: 0010 1100 1110

2) Some os números 742 e 268 em BCD

742	¹¹ 0111	¹ 0100	0010	
+ 268	+ 0010	+ 0110	+ 1000	
<u>1010</u>	<u>1001</u>	<u>1010</u>	<u>1010</u>	> 1001
	+ 0000	+ 0110	+ 0110	→ +6
	¹¹ <u>1010</u>	0001	0000	
	+ 0110			
	¹ 0000			
	1	0	1	0

Exercícios: Respostas

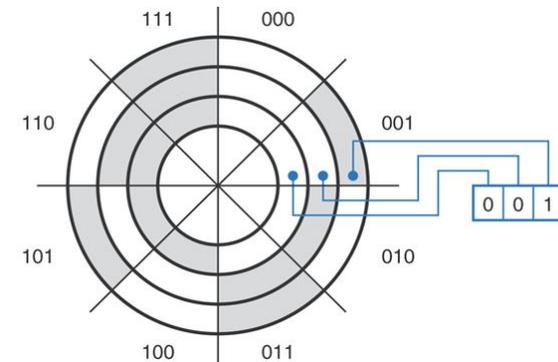
3) Quantas “fronteiras ruins” existem no disco de codificação BCD de 3 bits?)

001 → 010

101 → 110

011 → 100

111 → 000



4) Responda a questão 3 para um disco de n bits, em função de n

→ Quando bit menos significativo (LSB) muda de 0 para 1, não há problema. O problema é quando ele muda de 1 para 0, gerando carries, o que altera pelo menos um bit mais alto. Logo, metade das fronteiras (2^{n-1}) geram problemas

CÓDIGOS PARA CARACTERES

Código ASCII

- Informação processada por computador: bits
- Então como representar texto...?
 - Usa-se um código: tabela que especifica representação binária para um determinado conjunto de símbolos.
 - Comumente: ASCII (*American Standard Code for Information Interchange*),
 - Código alfanumérico: letras do alfabeto, números, símbolos, sinais e alguns caracteres não-imprimíveis de controle
 - 7 bits: 128 caracteres diferentes (letras com acentos não inclusas)
 - Pronúncia: o correto é “ASKI”, não “ASK2”

Código ASCII

		<i>b₆b₅b₄ (column)</i>							
<i>b₃b₂b₁b₀</i>	<i>Row (hex)</i>	<i>000 0</i>	<i>001 1</i>	<i>010 2</i>	<i>011 3</i>	<i>100 4</i>	<i>101 5</i>	<i>110 6</i>	<i>111 7</i>
0000	0	NUL	DLE	SP	0	@	P	~	p
0001	1	SOH	DC1	!	1	A	Q	a	q
0010	2	STX	DC2	"	2	B	R	b	r
0011	3	ETX	DC3	#	3	C	S	c	s
0100	4	EOT	DC4	\$	4	D	T	d	t
0101	5	ENQ	NAK	%	5	E	U	e	u
0110	6	ACK	SYN	&	6	F	V	f	v
0111	7	BEL	ETB	'	7	G	W	g	w
1000	8	BS	CAN	(8	H	X	h	x
1001	9	HT	EM)	9	I	Y	i	y
1010	A	LF	SUB	*	:	J	Z	j	z
1011	B	VT	ESC	+	;	K	[k	{
1100	C	FF	FS	,	<	L	\	l	
1101	D	CR	GS	-	=	M]	m	}
1110	E	SO	RS	.	>	N	^	n	~
1111	F	SI	US	/	?	O	_	o	DEL

Código ASCII

Control codes

NUL	Null	DLE	Data link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Negative acknowledge
ACK	Acknowledge	SYN	Synchronize
BEL	Bell	ETB	End transmitted block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SO	Shift out	RS	Record separator
SI	Shift in	US	Unit separator
SP	Space	DEL	Delete or rubout

Unicode

- ASCII não suporta caracteres com acentos...
 - Apenas 7 bits: desenvolvido para alfabeto inglês
- Unicode: suporte a caracteres em múltiplos idiomas
 - 32 bits: ~4 bilhões de símbolos possíveis
 - Construído para ser compatível com ASCII: códigos de 00 a 7F representam os mesmos símbolos em Unicode e ASCII
 - Padrão ISO/IEC 10646: define 3 métodos de codificação UTF-8, UTF-16 e UTF-32
 - UTF = Unicode Transformation Format
 - Maiores informações: www.unicode.org

APÊNDICE

Somador BCD

- Relembrando: código BCD
 - 4 bits, representação direta binária de 0 a 9
- Soma BCD: usa a adição binária com 4 bits, mas não é idêntica a ela...
 - Requer correção dos valores inválidos:
 - Aqueles acima de 9
 - Requer correção do “vai-um decimal”:
 - Diferente do vai-um hexadecimal (de 4 bits)

Somador BCD

Comparação entre soma binária (4bits) e BCD

Resultado da soma de 4 bits	Soma Hexa	Vai-um Hexa	Soma BCD	Vai-um BCD	Correção
0 a 9	0 a 9	0	0 a 9	0	-
10 a 15	10-15 (A-F)	0	0 a 5	1	Soma 6 $C_{BCD}=1$
16 a 19	0-3	1	6 a 9	1	Soma 6 $C_{BCD}=C_{hexa}$

↑
Obs.: 9+9+1 (“vem-um”)

Somador BCD

- Correção da soma BCD em relação ao resultado hexadecimal:

- Se resultado da soma entre A e F, OU
- Se $\text{vai-um}_{\text{hexa}} = 1$

Lógica adicional sobre saída do somador hexa

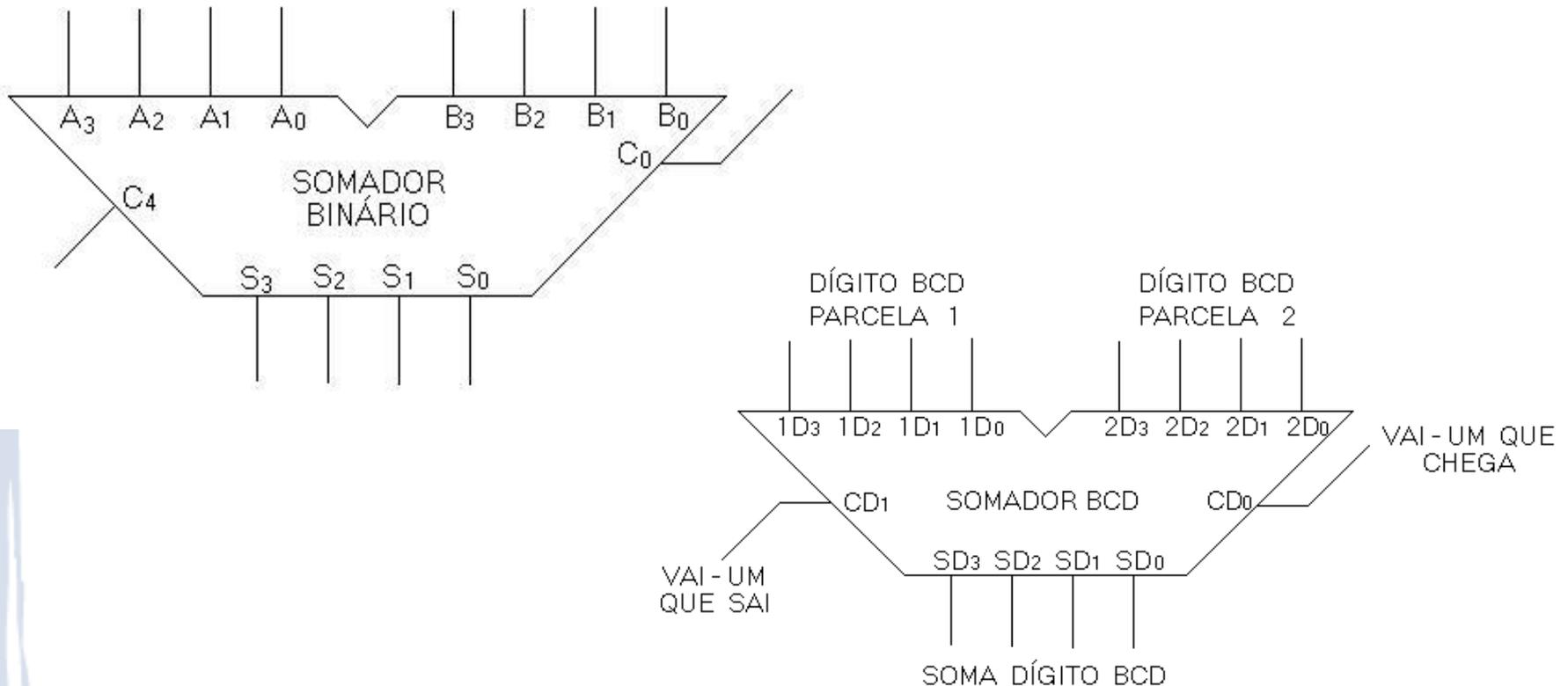
- Nesse caso, ação a tomar:

- **Somar 6 (0110) à soma hexa**
- $\text{Vai-um}_{\text{BCD}} = 1$

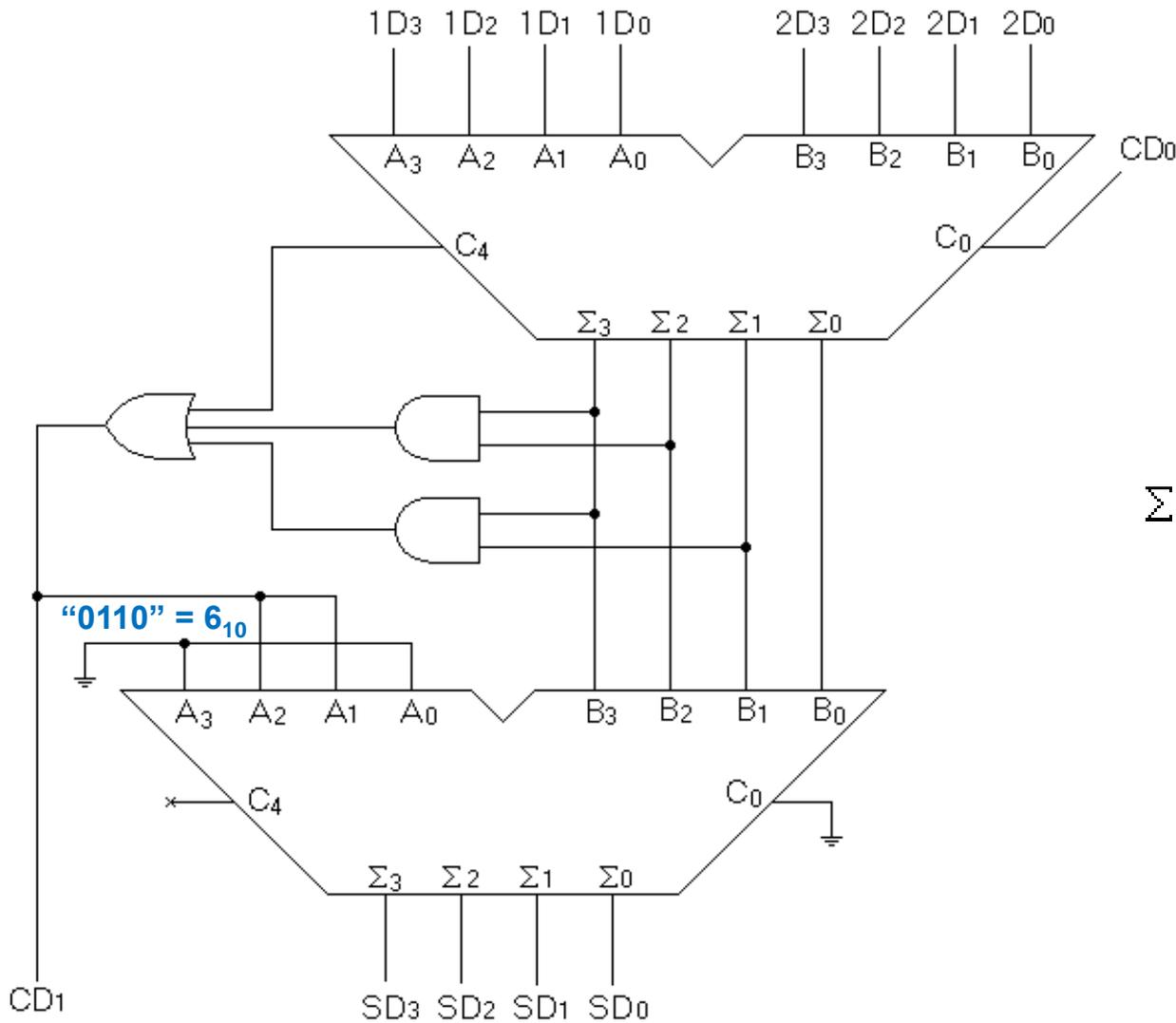
Necessário um segundo somador

Somador BCD

- Desafio: obter um Somador BCD a partir de um Somador Binário de (4 bits)



Somador BCD



10-16

		$\Sigma_3 \Sigma_2$		
$\Sigma_1 \Sigma_0$	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	0	0	1	1
10	0	0	1	1