



IC-UNICAMP

# MC 602

## Circuitos Lógicos e Organização de Computadores

IC/Unicamp

Prof Mario Côrtes

### Capítulo 5

## Sistemas de Numeração e Circuitos Aritméticos

# Tópicos

- Sistemas de numeração
  - Representação sinal-magnitude
  - Representação  $K_1$  e  $K_2$
  - Base Octal e Hexadecimal
  - BCD
- Circuitos aritméticos
  - Somadores half-adder e full-adder
  - Somador/subtrator
  - Somador com overflow
  - Carry Look Ahead

# Sistemas numéricos posicionais

- Decimal
  - Dígitos de 0 a 9

$$V(D) = d_{n-1} \cdot 10^{(n-1)} \dots + d_{n-2} \cdot 10^{(n-2)} + \dots + d_1 \cdot 10^1 + d_0 \cdot 10^0$$

$$V(D) = \sum_{i=0}^{n-1} d_i \cdot 10^i$$

- Binário
  - Dígitos 0 e 1

$$V(b) = b_{n-1} \cdot 2^{(n-1)} \dots + b_{n-2} \cdot 2^{(n-2)} + \dots + b_1 \cdot 2^1 + b_0 \cdot 2^0$$

$$V(b) = \sum_{i=0}^{n-1} b_i \cdot 2^i$$

# Bases de representação

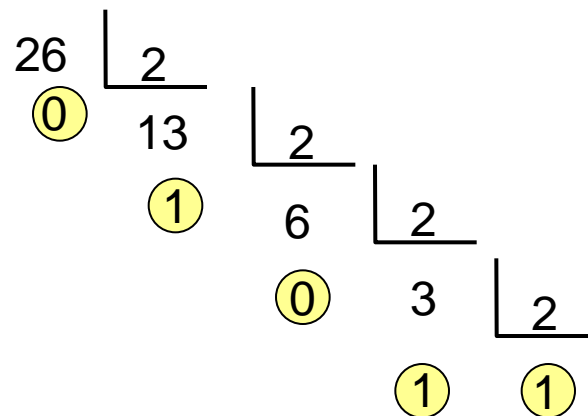
- Mais utilizadas: decimal (10), binária (2), octal (8), hexadecimal (16)
  - octal e hexa: “visões” convenientes do binário

Decimal	Binário	Octal	Hexa
0	000000	00	00
1	000001	01	01
2	000010	02	02
3	000011	03	03
4	000100	04	04
5	000101	05	05
6	000110	06	06
7	000111	07	07
8	001000	10	08
9	001001	11	09
10	001010	12	0A
11	001011	13	0B
12	001100	14	0C
13	001101	15	0D
14	001110	16	0E
15	001111	17	0F

Decimal	Binário	Octal	Hexa
16	010000	20	10
17	010001	21	11
18	010010	22	12
19	010011	23	13
20	010100	24	14
21	010101	25	15
22	010110	26	16
23	010111	27	17
24	011000	30	18
25	011001	31	19
26	011010	32	1A
27	011011	33	1B
28	011100	34	1C
29	011101	35	1D
30	011110	36	1E
31	011111	37	1F

# Conversão decimal $\leftrightarrow$ binário

- Binário  $\rightarrow$  Decimal
  - Basta somar as potências de 2
- Decimal  $\rightarrow$  Binário
  - Divisão sucessiva (regra geral)
  - Exemplo  $26_{10} \rightarrow 11010_2$



- ex: converter  $46_{10} \rightarrow$  base 3 ( $1201_3$ )

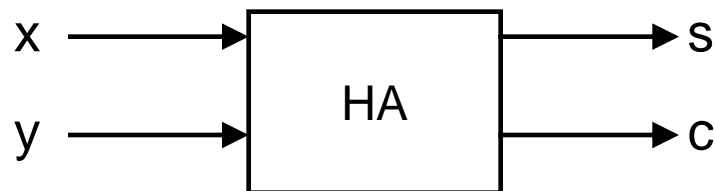
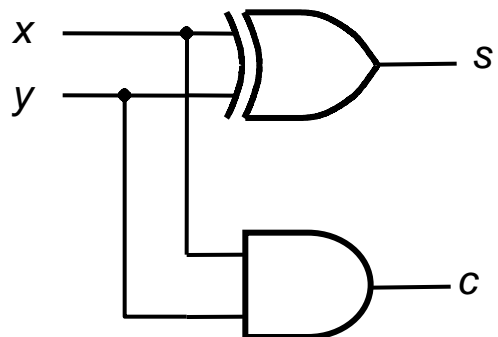
- Converter
  - 0101011010111 para octal
  - 0101011010111 para hexadecimal
  - $AF25_{16}$  para decimal
  - $AF25_{16}$  para octal
  - $5327_8$  para hexadecimal
  - $5327_8$  para decimal

# Soma de 1 bit

x	0	0	1	1
+ y	+ 0	+ 1	+ 0	+ 1
c s	0 0	0 1	0 1	1 0

x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

- $s = \bar{x} \cdot y + x \cdot \bar{y} = x \oplus y$
- $c = xy$



# Soma de palavras de bits

$$\begin{array}{r}
 X = x_4x_3x_2x_1x_0 \quad 01111 \quad (15)_{10} \\
 + Y = y_4y_3y_2y_1y_0 \quad 01010 \quad (10)_{10} \\
 \hline
 \quad \quad \quad 1110 \quad \leftarrow \text{carries gerados} \\
 \hline
 S = s_4s_3s_2s_1s_0 \quad 11001 \quad (25)_{10}
 \end{array}$$

$C_{in}$	0	0	0	0
X	0	0	1	1
+ y	+ 0	+ 1	+ 0	+ 1
$C_{out}$ S	0 0	0 1	0 1	1 0

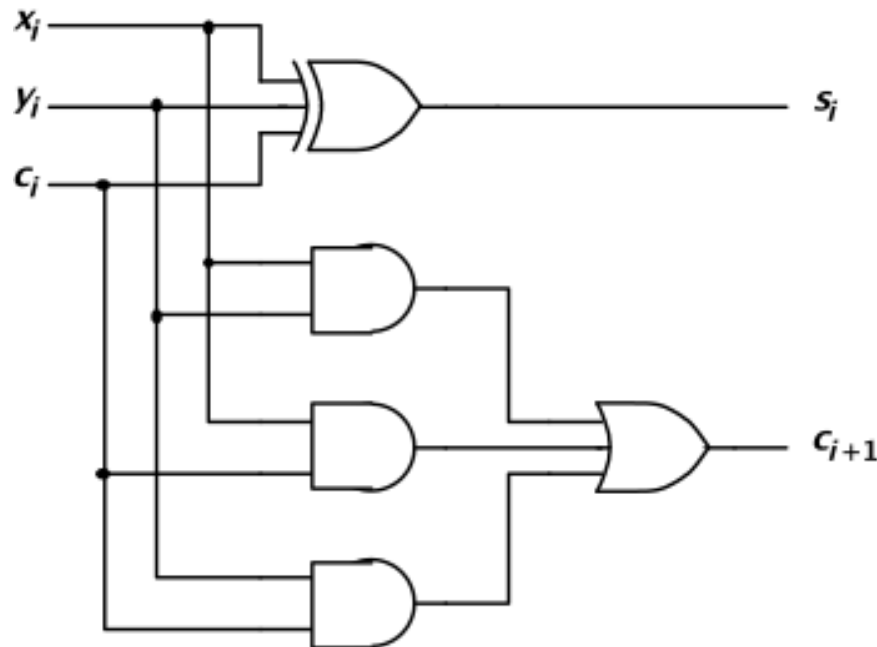
$C_{in}$	1	1	1	1
X	0	0	1	1
+ y	+ 0	+ 1	+ 0	+ 1
$C_{out}$ S	0 1	1 0	1 0	1 1

$C_{in}$	X	y	$C_{out}$	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	0	1



# Soma de palavras de bits: Full Adder

$c_i$	$x_i$	$y_i$	$c_{i+1}$	$s_i$
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



$x_i y_i$ $c_i$	00	01	11	10
0		1		1
1	1		1	

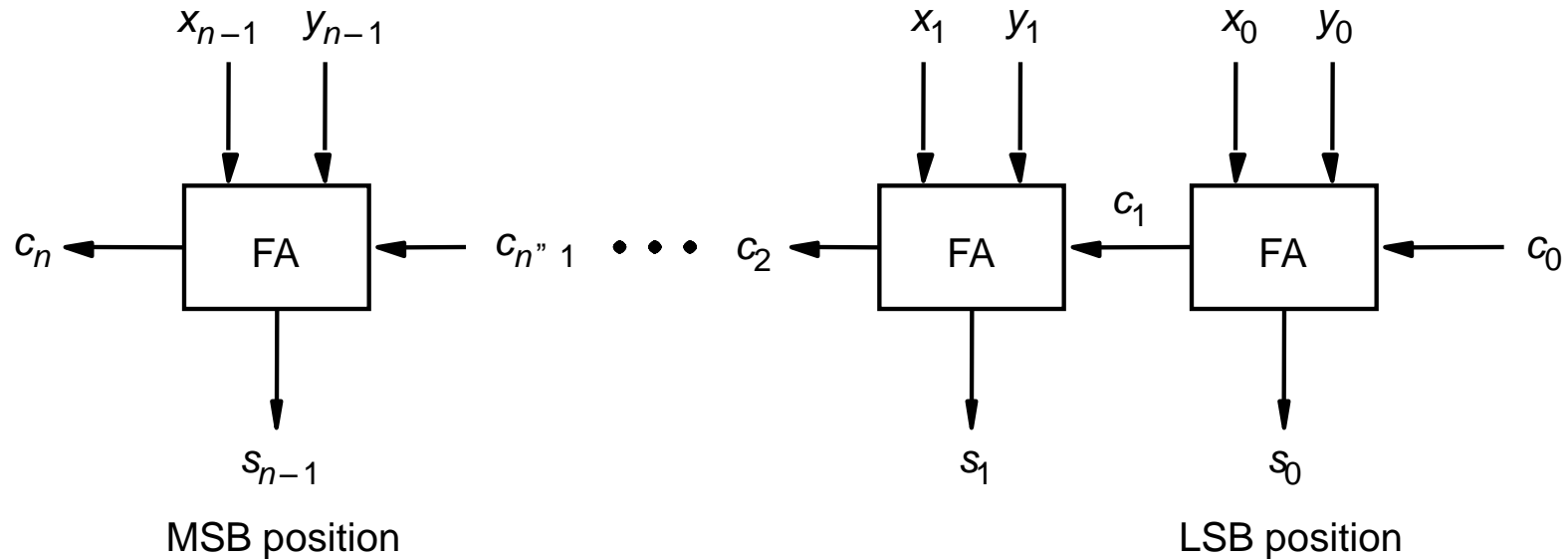
$$s_{i+1} = x_i \oplus y_i \oplus c_i$$

$x_i y_i$ $c_i$	00	01	11	10
0			1	
1		1	1	1

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

(b) Karnaugh maps

# Somador Ripple-Carry construído a partir de Full-Adders

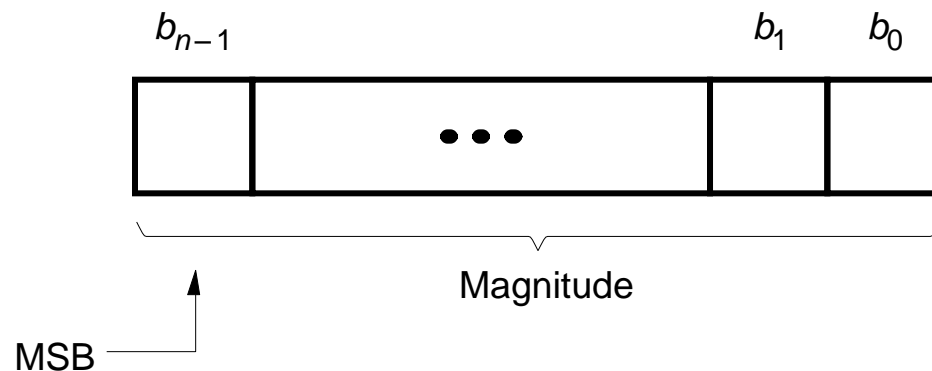




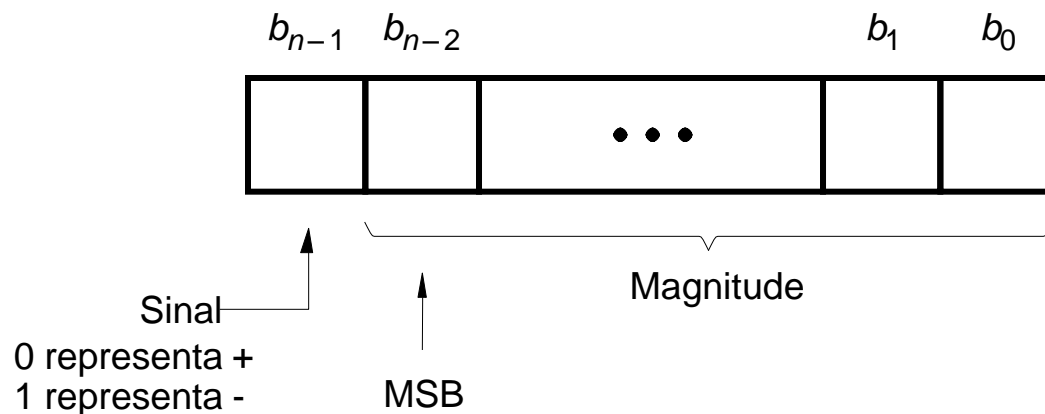
# Representação de números negativos

- Sinal magnitude
- Complemento de 1
- Complemento de 2

# Representação Sinal Magnitude



Número sem sinal



Número com sinal

# Representação Sinal Magnitude

- Sinal e Magnitude
  - 1 bit de sinal,  $N-1$  bits de magnitude
  - O bit de sinal é o mais significativo (mais a esquerda)
    - Número negativo: 1
    - Número positivo: 0
  - Exemplo, representação de  $\pm 5$  com 4-bit:
    - $-5 = 1101_2$
    - $+5 = 0101_2$
  - Intervalo de um número  $N$ -bit sinal/magnitude:
    - $[-(2^{N-1}-1), 2^{N-1}-1]$

- Exemplo:  $-5 + 5$

$$\begin{array}{r} 1101 \\ + \underline{0101} \\ 10010 \end{array}$$

- Duas representações para 0 ( $\pm 0$ ):

$$\begin{array}{l} 1000 \\ 0000 \end{array}$$

# Complemento de 1

## Complemento de 1 ( $K_1$ )

Em complemento de “Um” o número negativo  $K_1$ , com  $n$ -bits, é obtido subtraindo seu positivo  $P$  de  $2^n - 1$

$$K_1 = (2^n - 1) - P$$

Exemplo: se  $n = 4$  então:

$$K_1 = (2^4 - 1) - P$$

$$K_1 = (16 - 1) - P$$

$$K_1 = (1111)_2 - P$$

$$P = 7 \rightarrow K_1 = ?$$

$$7 = (0111)_2$$

$$-7 = (1111)_2 - (0111)_2$$

$$-7 = (1000)_2$$

# Complemento de 1

- Complemento de 1 ( $K_1$ )
  - Regra Prática

$$K_1 = (2^n - 1) - P$$

$$K_1 = 11\dots 11 - (p_{n-1} \dots p_0)$$

$$K_1 = \overline{p_{n-1} \dots p_0}$$



# Soma em complemento de 1

$$\begin{array}{r}
 (+5) \quad 0101 \\
 +(+2) \quad +0010 \\
 \hline
 (+7) \quad 0111
 \end{array}$$

$$\begin{array}{r}
 (-5) \quad 1010 \\
 +(+2) \quad +0010 \\
 \hline
 (-3) \quad 1100
 \end{array}$$

$$\begin{array}{r}
 (+5) \quad 0101 \\
 +(-2) \quad +1101 \\
 \hline
 (+3) \quad 10010 \\
 \begin{array}{l} \leftarrow \leftarrow 1 \\ \leftarrow \leftarrow \leftarrow 1 \end{array} \\
 \hline
 0011
 \end{array}$$

$$\begin{array}{r}
 (-5) \quad 1010 \\
 +(-2) \quad +1101 \\
 \hline
 (-7) \quad 10111 \\
 \begin{array}{l} \leftarrow \leftarrow 1 \\ \leftarrow \leftarrow \leftarrow 1 \end{array} \\
 \hline
 1000
 \end{array}$$

# Complemento de 2

## Complemento de 2 ( $K_2$ )

Em complemento de “Dois” o número negativo  $K$ , com  $n$ -bits, é obtido subtraindo seu positivo  $P$  de  $2^n$

$$K_2 = 2^n - P$$

Exemplo: se  $n = 4$  então:

$$K_2 = 2^4 - P$$

$$K_2 = 16 - P$$

$$K_2 = (10000)_2 - P$$

$$P = 7 \rightarrow K_2 = ?$$

$$7 = (0111)_2$$

$$-7 = (10000)_2 - (0111)_2$$

$$-7 = (1001)_2$$

# Complemento de 2

- Complemento de 2 ( $K_2$ )
  - Regra Prática

$$K_2 = 2^n - P \quad \longrightarrow \quad K_2 = (2^n - 1) + 1 - P$$

$$K_2 = (2^n - 1) - P + 1$$

$$K_2 = 11\dots 11 - (p_{n-1} \dots p_0) + 1$$

$$K_2 = (\overline{p_{n-1}} \dots \overline{p_0}) + 1 = K_1(P) + 1$$

# Complemento de 2

- Complemento de 2 ( $K_2$ )
  - Maior número positivo de 4-bit:  $0111_2$  ( $7_{10}$ )
  - Maior número negativo de 4-bit:  $1000_2$  ( $-2^3 = -8_{10}$ )
  - O most significant bit também indica o sinal (1 = negativo, 0 = positivo)
  - Intervalo de um número de  $N$ -bit:  $[-2^{N-1}, 2^{N-1}-1]$



# Representação de Números Negativos

<b>b<sub>3</sub> b<sub>2</sub> b<sub>1</sub> b<sub>0</sub></b>	<b>Sinal Magnitude</b>	<b>Complemento de 1</b>	<b>Complemento de 2</b>
0111	+7	7	+7
0110	+6	6	+6
0101	+5	5	+5
0100	+4	4	+4
0011	+3	3	+3
0010	+2	2	+2
0001	+1	1	+1
0000	+0	0	0
1000	-0	-7	-8
1001	-1	-6	-7
1010	-2	-5	-6
1011	-3	-4	-5
1100	-4	-3	-4
1101	-5	-2	-3
1110	-6	-1	-2
1111	-7	-0	-1

# Adição em $K_2$

$$\begin{array}{r}
 (+5) \quad 0101 \\
 + (+2) \quad +0010 \\
 \hline
 (+7) \quad 0111
 \end{array}$$

$$\begin{array}{r}
 (-5) \quad 1011 \\
 + (+2) \quad +0010 \\
 \hline
 (-3) \quad 1101
 \end{array}$$

$$\begin{array}{r}
 (+5) \quad 0101 \\
 + (-2) \quad +1110 \\
 \hline
 (+3) \quad 10011
 \end{array}$$

↑  
ignore

$$\begin{array}{r}
 (-5) \quad 1011 \\
 + (-2) \quad +1110 \\
 \hline
 (-7) \quad 11001
 \end{array}$$

↑  
ignore

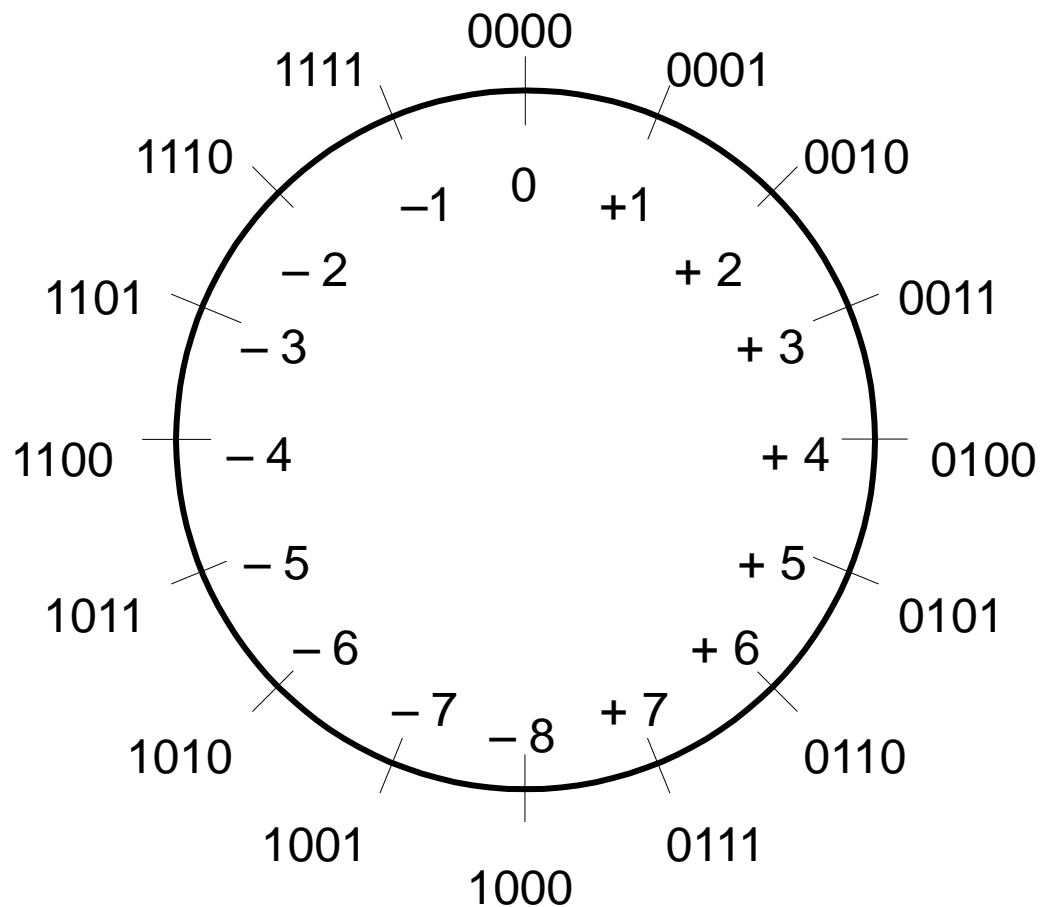
# Subtração em $K_2$



$$\begin{array}{r}
 (+5) \quad 0101 \\
 - (-2) \quad \underline{-1110} \\
 \hline
 (+7)
 \end{array}
 \Rightarrow
 \begin{array}{r}
 0101 \\
 + 0010 \\
 \hline
 0111
 \end{array}
 \quad
 \begin{array}{r}
 (+5) \quad 0101 \\
 - (+2) \quad \underline{-0010} \\
 \hline
 (+3)
 \end{array}
 \Rightarrow
 \begin{array}{r}
 0101 \\
 + 1110 \\
 \hline
 10011 \\
 \uparrow \\
 \text{ignore}
 \end{array}$$
  

$$\begin{array}{r}
 (-5) \quad 1011 \\
 - (-2) \quad \underline{-1110} \\
 \hline
 (-3)
 \end{array}
 \Rightarrow
 \begin{array}{r}
 1011 \\
 + 0010 \\
 \hline
 1101
 \end{array}
 \quad
 \begin{array}{r}
 (-5) \quad 1011 \\
 - (+2) \quad \underline{-0010} \\
 \hline
 (-7)
 \end{array}
 \Rightarrow
 \begin{array}{r}
 1011 \\
 + 1110 \\
 \hline
 11001 \\
 \uparrow \\
 \text{ignore}
 \end{array}$$

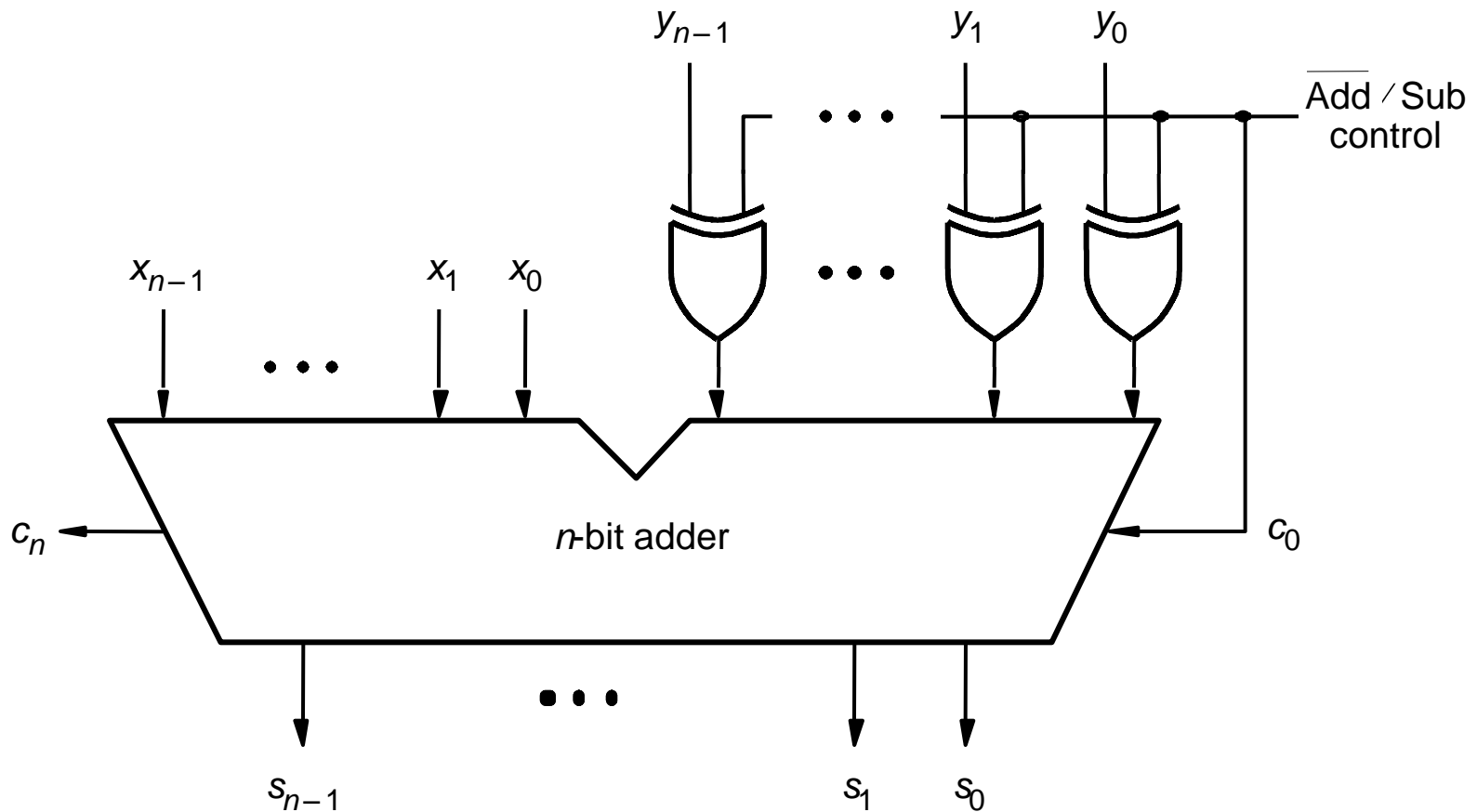
# K2: interpretação gráfica





# Somador/Subtrator

$$K_2 = (\overline{p_{n-1}} \dots \overline{p_0}) + 1 = K_1(P) + 1$$

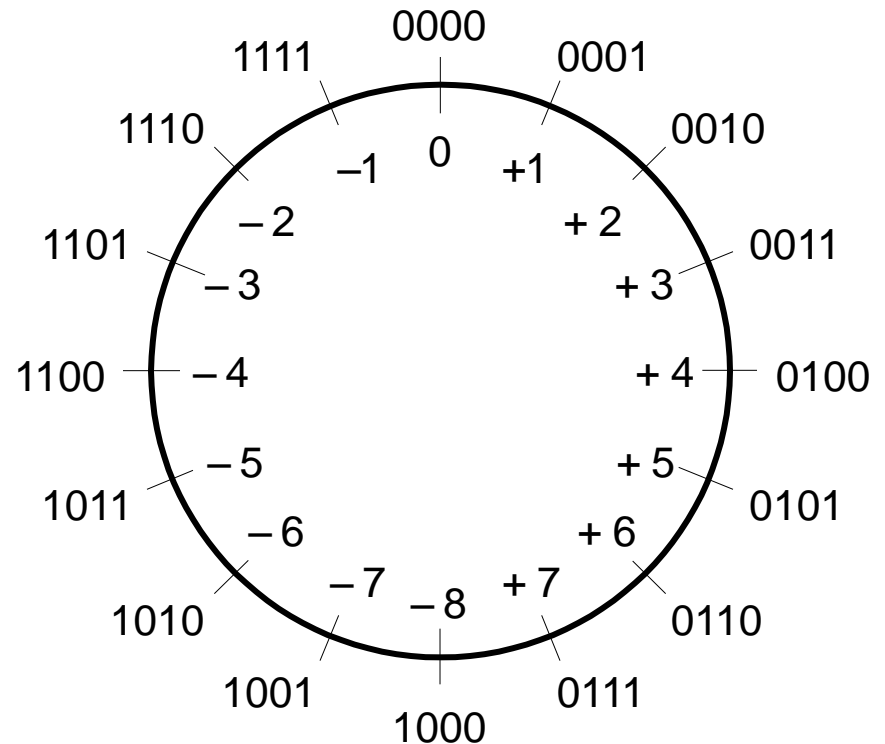


# Overflow (Soma)

- $A + B$ 
  - $\text{sign}(A) = \text{sign}(B)$ : overflow **possível**
  - $\text{sign}(A) \neq \text{sign}(B)$ : overflow impossível

$$\begin{array}{r}
 (+7) \quad 0111 \\
 + (+2) \quad +0010 \\
 \hline
 (+9) \quad 1001
 \end{array}$$

$$\begin{array}{r}
 (-7) \quad 1001 \\
 + (+2) \quad +0010 \\
 \hline
 (-5) \quad 1011
 \end{array}$$

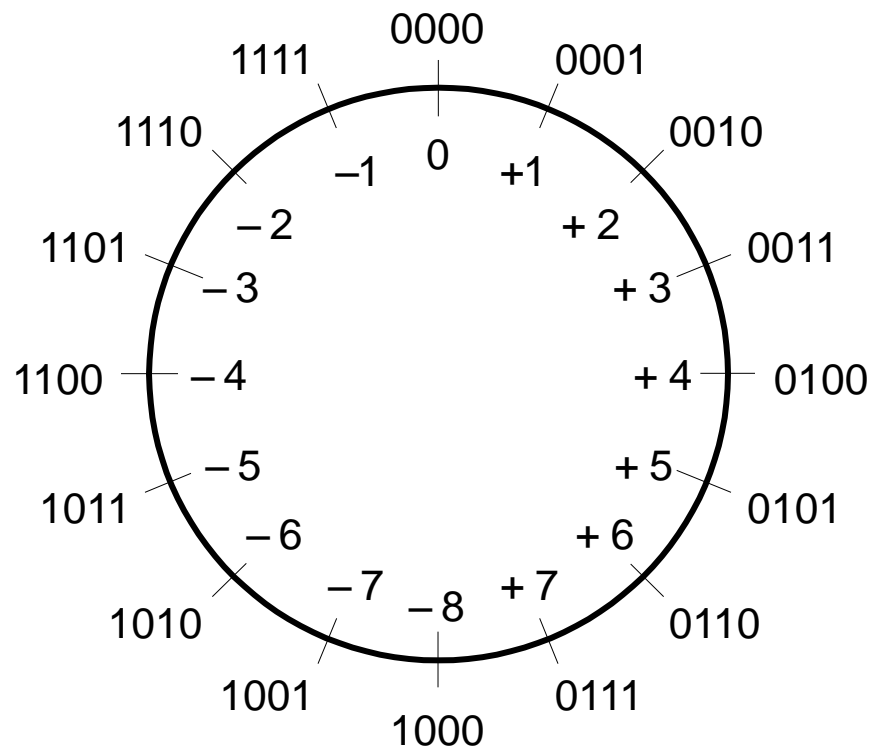


# Overflow (Subtração)

- $A - B = A + (-B)$ , reduz à soma
  - $\text{sign}(A) = \text{sign}(B)$ : overflow impossível
  - $\text{sign}(A) \neq \text{sign}(B)$ : overflow **possível**

$(+7)$	$(+7)$	$0111$
$- (+2)$	$+ (-2)$	$+ 1110$
$(+5)$	$(+5)$	$10101$

$(-7)$	$(-7)$	$1001$
$- (+2)$	$+ (-2)$	$+ 1110$
$(-5)$	$(-9)$	$10111$





# Resumo Overflow

A	B	S = A + B	OV
+	-	+	0
+	-	-	0
-	+	+	0
-	+	-	0
+	+	+	0
+	+	-	1
-	-	+	1
-	-	-	0

basta analisar soma, pois subtração pode ser transformada em soma

$$V = C_n(S) \text{ xor } C_{n-1}(S)$$

$$\begin{array}{r} 1 \\ 0 \text{ x..} \\ + 1 \text{ x..} \\ \hline 1 \text{ 0 x..} \end{array}$$

$$\begin{array}{r} 0 \\ 0 \text{ x..} \\ + 1 \text{ x..} \\ \hline 0 \text{ 1 x..} \end{array}$$

$$\begin{array}{r} 1 \\ 0 \text{ x..} \\ + 0 \text{ x..} \\ \hline 0 \text{ 1 x..} \end{array}$$

$$\begin{array}{r} 0 \\ 0 \text{ x..} \\ + 0 \text{ x..} \\ \hline 0 \text{ 0 x..} \end{array}$$

$$\begin{array}{r} 0 \\ 1 \text{ x..} \\ + 1 \text{ x..} \\ \hline 1 \text{ 0 x..} \end{array}$$

$$\begin{array}{r} 1 \\ 1 \text{ x..} \\ + 1 \text{ x..} \\ \hline 1 \text{ 1 x..} \end{array}$$



# Overflow em $K_2$

$$\begin{array}{r}
 (+7) \quad 0111 \\
 + (+2) \quad 0010 \\
 \hline
 (+9) \quad 1001 \\
 \\
 c_4 = 0 \\
 c_3 = 1
 \end{array}$$

$$\begin{array}{r}
 (-7) \quad 1001 \\
 + (+2) \quad 0010 \\
 \hline
 (-5) \quad 1011 \\
 \\
 c_4 = 0 \\
 c_3 = 0
 \end{array}$$

$$\begin{array}{r}
 (+7) \quad 0111 \\
 + (-2) \quad 1110 \\
 \hline
 (+5) \quad 10101 \\
 \\
 c_4 = 1 \\
 c_3 = 1
 \end{array}$$

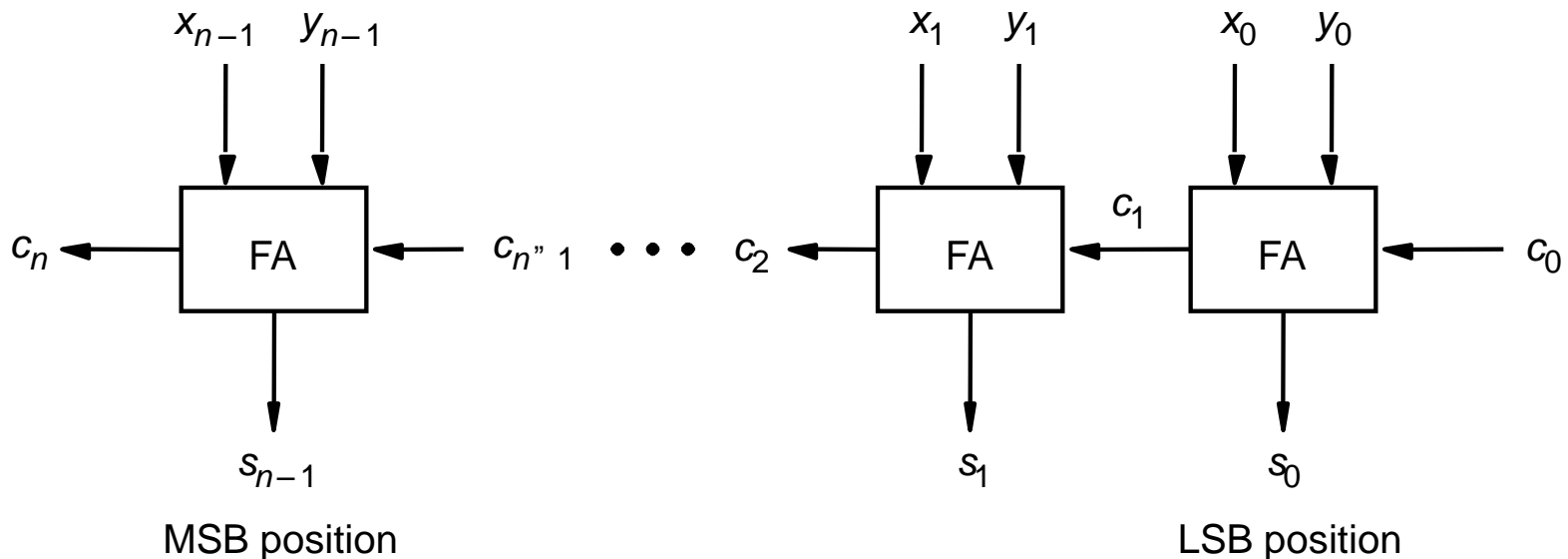
$$\begin{array}{r}
 (-7) \quad 1001 \\
 + (-2) \quad 1110 \\
 \hline
 (-9) \quad 10111 \\
 \\
 c_4 = 1 \\
 c_3 = 0
 \end{array}$$

# Somador Ripple Carry

- Atraso para um somador de n bits:

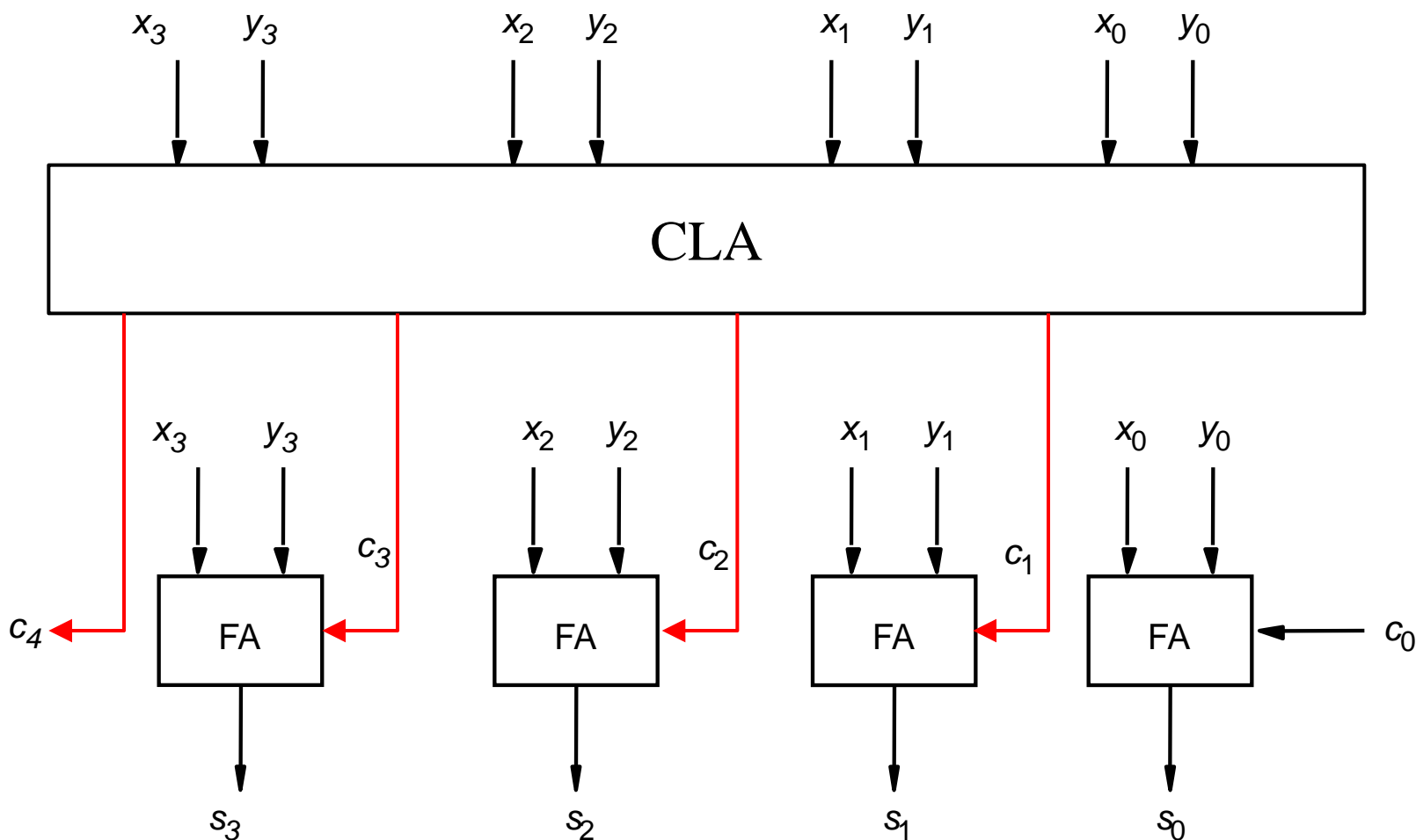
$$t_{\text{ripple}} = Nt_{FA}$$

Onde  $t_{FA}$  é o atraso de um full adder



# Antecipação de Carry: Carry Look Ahead (CLA)

- Aplicado para módulo de 4 bits



# CLA: Generate e Propagate

- Para gerar carries com atraso menor e fixo
- Observar para o bit  $i$ 
  - Carry é gerado sempre independente das entradas e dos carries de nível anterior:
    - $g_i = x_i y_i$
  - Carry é propagado sempre independente das entradas e dos carries de nível anterior:
    - $p_i = x_i + y_i$
    - observar que um carry de entrada é morto/killed se:
      - $\sim x_i \cdot \sim y_i$
      - Que é exatamente  $\sim p_i$



# CLA: Como gerar os carries a partir de g e p

$$C_1 = g_0 + p_0 C_0$$

$$C_2 = g_1 + p_1 C_1 \quad C_2 = g_1 + p_1 g_0 + p_1 p_0 C_0$$

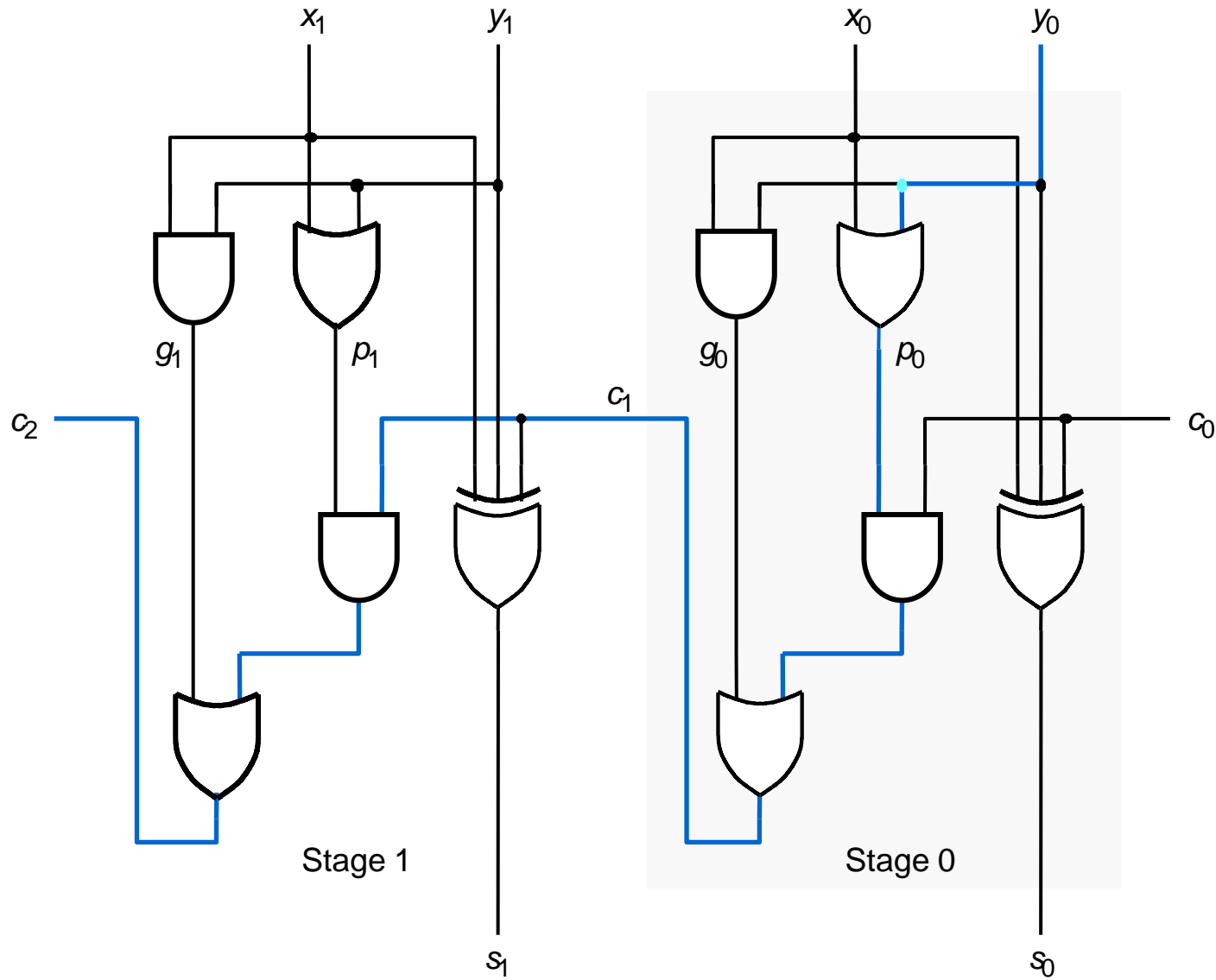
$$C_3 = g_2 + p_2 C_2 \quad C_3 =$$

$$C_4 = g_3 + p_3 C_3 \quad C_4 =$$

Atraso:

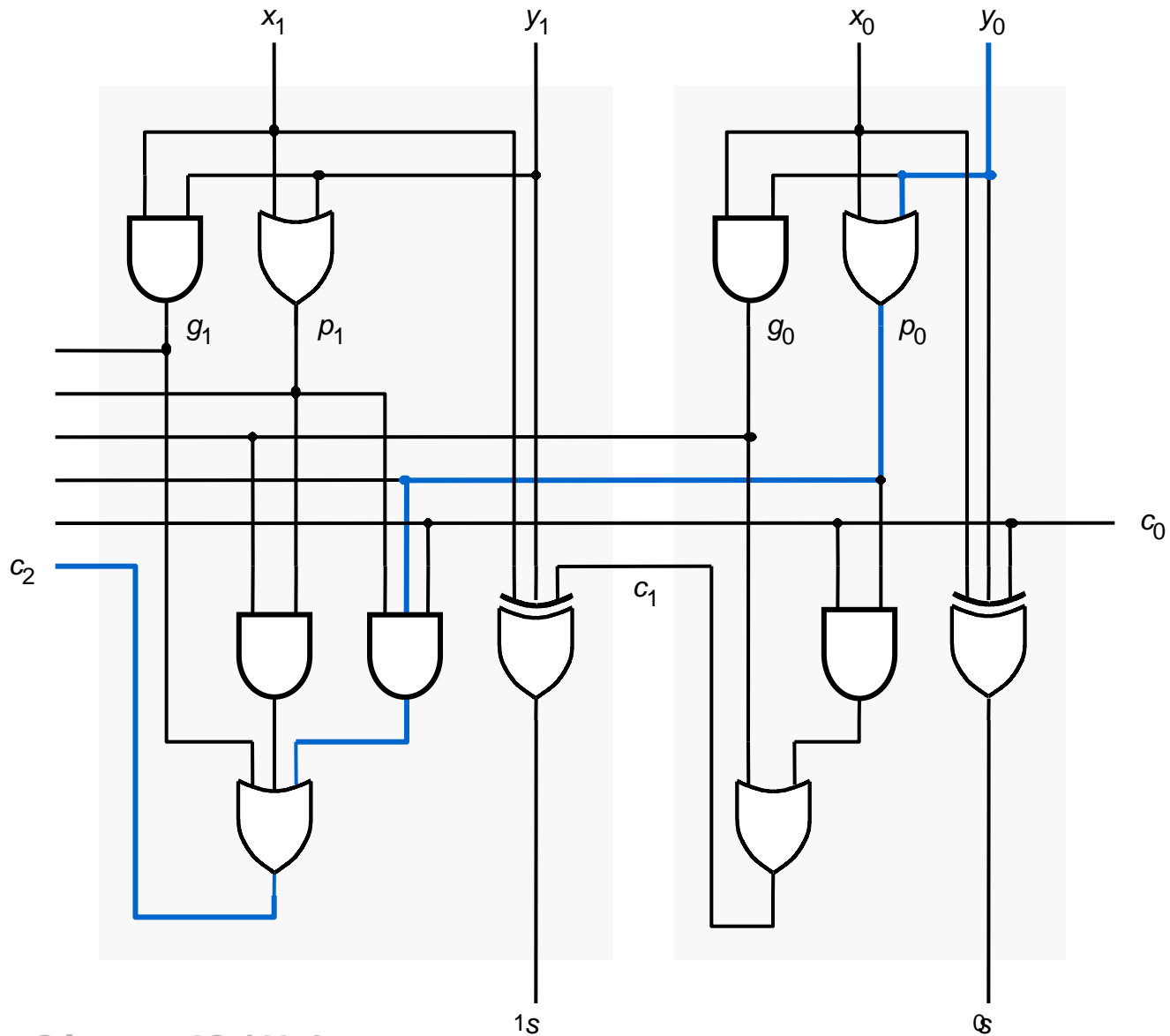
- entradas  $\Rightarrow g_i p_i$  (1G)
- $g_i p_i \Rightarrow$  carry (2G) : 1 AND seguido de 1 OR
- carry  $\Rightarrow$  saídas (2G)
- Total: 5G, independente de n
- Para  $n^o$  bits  $> 4$ , ver abordagem hierárquica na seção 5.4.1

# CLA: circuito p/ bits 0 e 1 carries encadeados

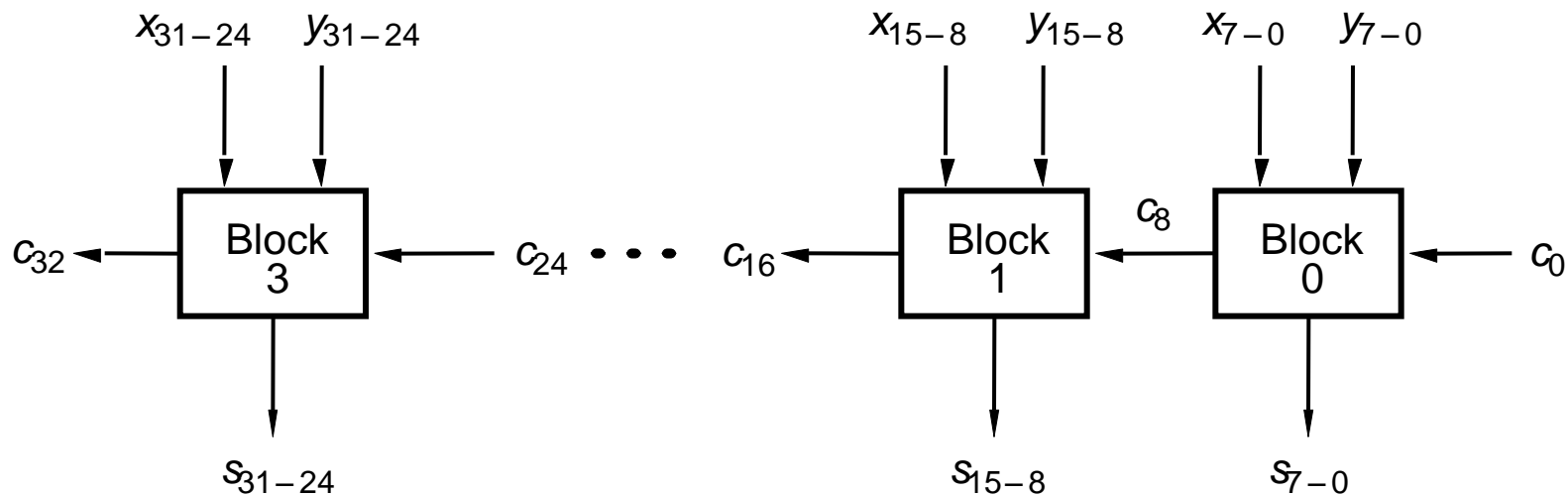


# CLA: circuito p/ bits 0 e 1

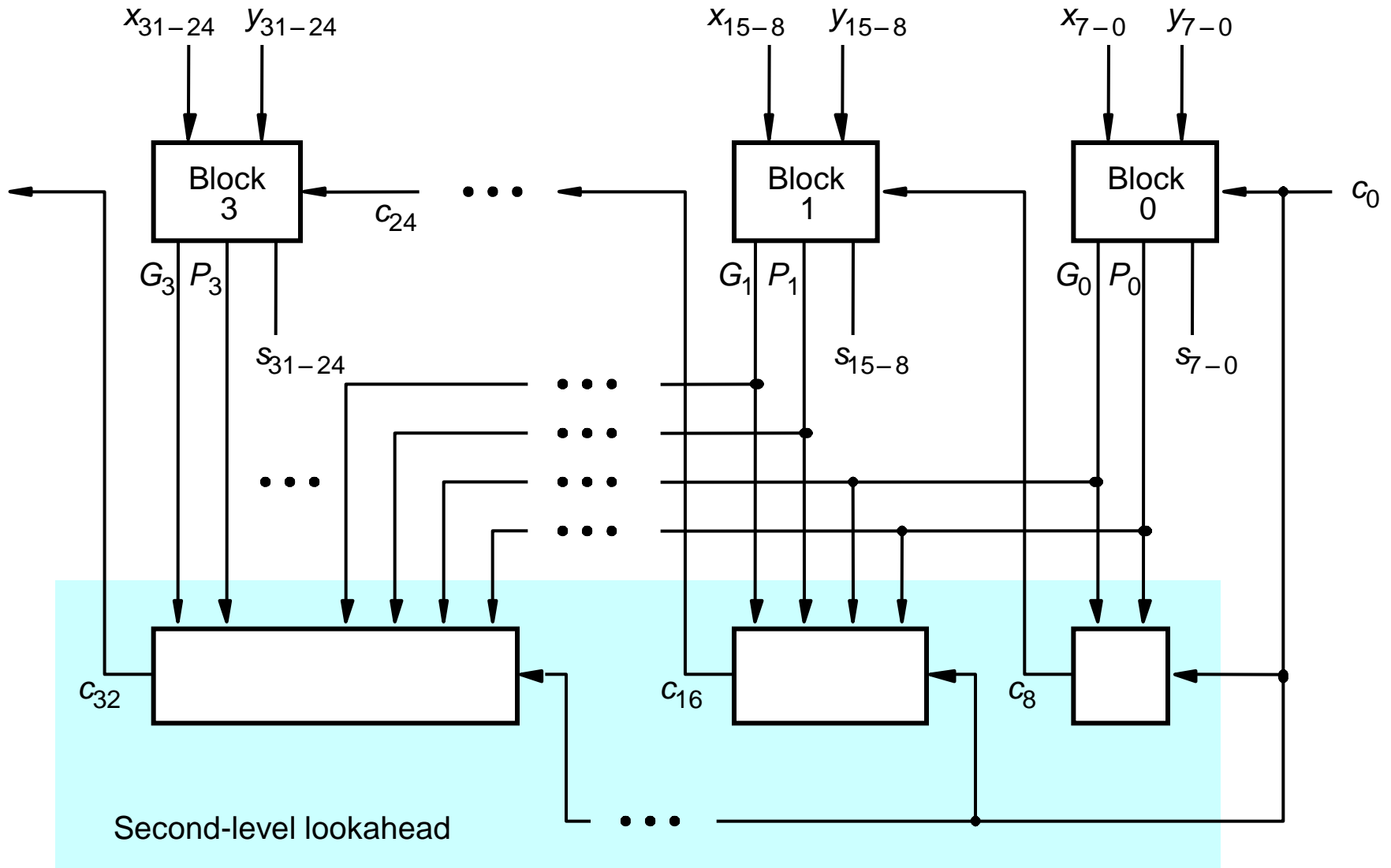
## circuito correto sem ripple



# CLA hierárquico: com ripple-carry entre blocos



# CLA hierárquico



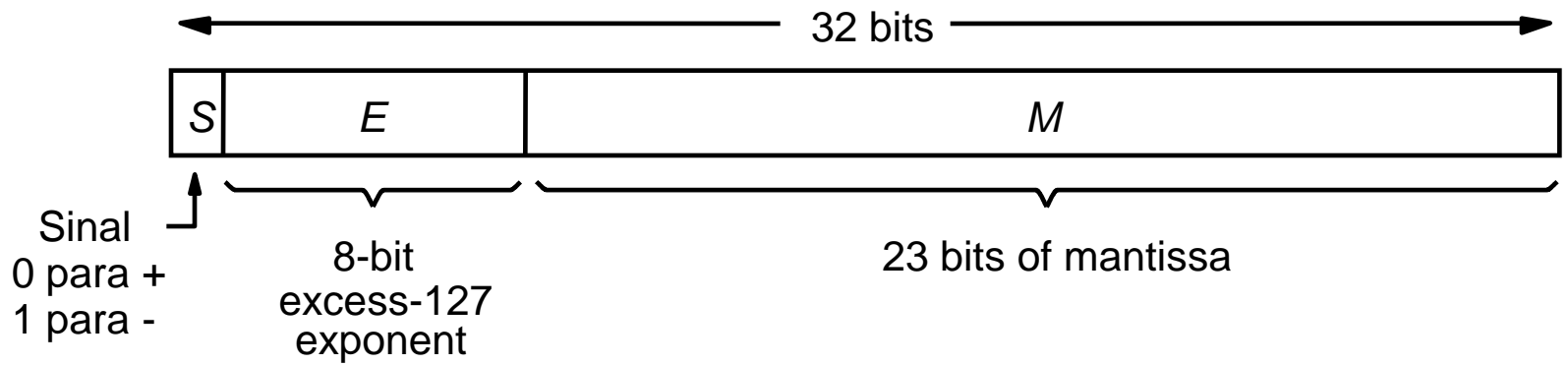
# Multiplicação de inteiros positivos

- Implementação totalmente combinacional (sequencial mais adiante) = mapeamento um-a-um com algoritmo convencional
  - implementação na seção 5.6.1

Multiplicando M (11)	1 1 1 0	
Multiplicador Q (14)	1 0 1 1	
<b>produto parcial 0</b>	1 1 1 0	
	+ 1 1 1 0	
<b>produto parcial 1</b>	1 0 1 0 1	
	+ 0 0 0 0	
<b>produto parcial 2</b>	0 1 0 1 0	
	+ 1 1 1 0	
Produto P (154)	1 0 0 1 1 0 1 0	

# Números fracionários: ponto flutuante

- $N = \text{mantissa} * R^{\text{expoente}}$
- Formato normalizado (IEEE 754 precisão simples):
  - mantissa = 1.xxxxxxxxxxxx (1 implícito)
  - expoente =  $E - 127$
- exemplo (para precisão simples)
  - $N = 13.25_{10} = 1101.01_2 = 1101.01_2 * 2^0 = 1.10101_2 * 2^3$
  - $S = 0$        $E = 127 + 3 = 130_{10} = 1000\ 0010$        $M = 10101$
  - 0      1000 0010      101010000000000000000000000000



# BCD: Binary Coded Decimal

- Exemplo:
  - $52_{10} = 0101\ 0010_{\text{BCD}}$
- Mais utilizado no passado em calculadoras
- Motivação: simplificar (ou evitar) conversões dec $\rightarrow$ bin na entrada de dados e bin $\rightarrow$ dec no display
- Ainda usado em casos especiais, apesar da complexidade na soma

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001



# Adição Usando BCD

$$\begin{array}{r}
 X \quad \quad 0111 \quad \quad 7 \\
 + Y \quad + 0101 \quad + 5 \\
 \hline
 Z \quad \quad 1100 \quad \quad 12 \\
 \quad \quad + 0110 \\
 \hline
 \text{carry} \rightarrow 10010 \\
 \quad \quad \underbrace{\quad\quad\quad} \\
 \quad \quad S = 2
 \end{array}$$

$$\begin{array}{r}
 X \quad \quad 1000 \quad \quad 8 \\
 + Y \quad + 1001 \quad + 9 \\
 \hline
 Z \quad \quad 10001 \quad \quad 17 \\
 \quad \quad + 0110 \\
 \hline
 \text{carry} \rightarrow 10111 \\
 \quad \quad \underbrace{\quad\quad\quad} \\
 \quad \quad S = 7
 \end{array}$$

Passou de 10? Remove 10:

$$\begin{aligned}
 S - 10 &= S - 9 - 1 \\
 &= S + K_2(9_{10}) - 1 \\
 &= S + K_1(9_{10}) + 1 - 1 \\
 &= S + \text{not}(1001_2) \\
 &= S + 0110_2 \\
 &= S + 6_{10}
 \end{aligned}$$

Raciocínio Alternativo

Passou de 10?

Remove 10 (carry=1)

$$\begin{aligned}
 S - 10 &= S - (16 - 6) \\
 &= S + 6 - 16 \\
 &= (S + 6) - 16
 \end{aligned}$$

soma

carry



# ASCII

- American Standard Code for Information Interchange → representação binária de caracteres alfabéticos+numéricos+controle

0	NUL
1	SOH
2	ST
3	ET
4	EOT
5	ENQ
6	ACK
7	BEL
8	BS
9	HT
10	LF
11	VT
12	FF
13	CR
14	SO
15	SI

16	DLE
17	DC1
18	DC2
19	DC3
20	DC4
21	NAK
22	SYN
23	ETB
24	CAN
25	EM
26	SUB
27	ESC
28	FS
29	GS
30	RS
31	US

32	SP
33	!
34	"
35	#
36	\$
37	%
38	&
39	'
40	
41	)
42	*
43	+
44	,
45	-
46	.
47	/

48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	:
59	;
60	<
61	=
62	>
63	?

64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O

80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	
89	Y
90	Z
91	[
92	\
93	]
94	^
95	_

96	`
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o

112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	
121	y
122	z
123	{
124	
125	}
126	~
127	DEL