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

IC-UNICAMP

Sistemas numéricos posicionais

- Decimal
  - Dígitos de 0 a 9

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

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

- Binário
  - Dígitos 0 e 1

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

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

IC-UNICAMP

Bases de representação

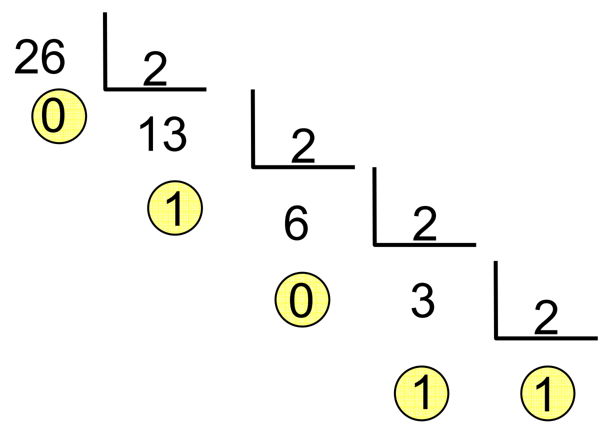
- Mais utilizadas: decimal (10), binária (2), octal (8), hexadecimal (16)
  - octal e hexa: “visões” convenientes do binário

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 ←→ binário

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



- ex: converter 46_{10} → base 3 (1201_3)

IC-UNICAMP

Conversões

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

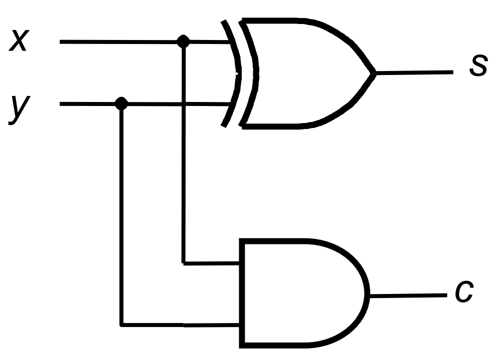
IC-UNICAMP

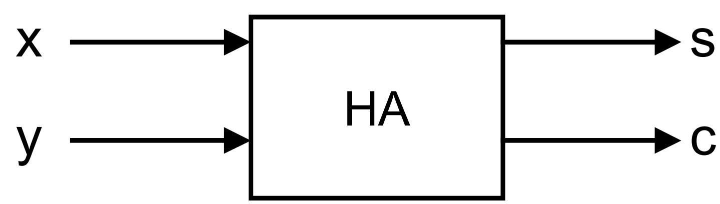
Soma de 1 bit

$$\begin{array}{cc} & x \\ + & y \\ \hline c & s \end{array} \qquad \begin{array}{cc} & 0 \\ + & 0 \\ \hline 0 & 0 \end{array} \qquad \begin{array}{cc} & 0 \\ + & 1 \\ \hline 0 & 1 \end{array} \qquad \begin{array}{cc} & 1 \\ + & 0 \\ \hline 0 & 1 \end{array} \qquad \begin{array}{cc} & 1 \\ + & 1 \\ \hline 1 & 0 \end{array}$$

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

- $s = \overline{x} . y + x . \overline{y} = x \oplus y$
- $c = xy$





IC-UNICAMP

Soma de palavras de bits

$$X = x_4x_3x_2x_1x_0 \quad 0\,1\,1\,1\,1 \quad (15)_{10}$$

$$+\,Y = y_4y_3y_2y_1y_0 \quad 0\,1\,0\,1\,0 \quad (10)_{10}$$

1 1 1 0 ← carries gerados

$$S = s_4s_3s_2s_1s_0 \quad 1\,1\,0\,0\,1 \quad (25)_{10}$$

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

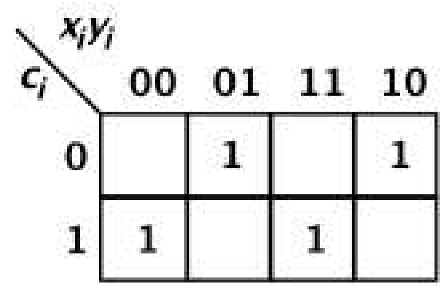
c_{in}	1	1	1	1
x	0	0	1	1
$+\ y$	+ 0	+ 1	+ 0	+ 1
$c_{out}\ s$	0 1	1 0	1 0	1 1

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	1	1

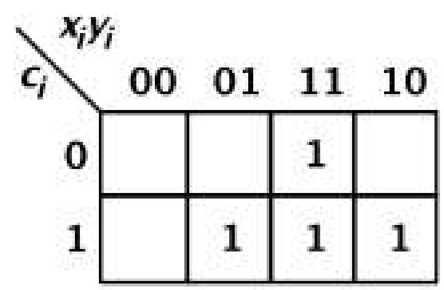
IC-UNICAMP

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

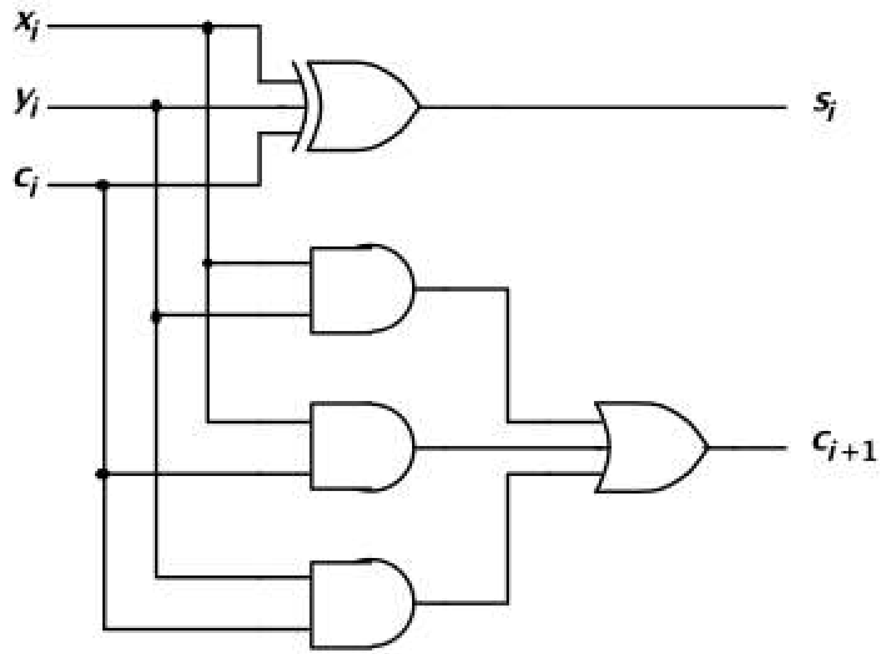


$s_{i+1} = x_i \oplus y_i \oplus c_i$



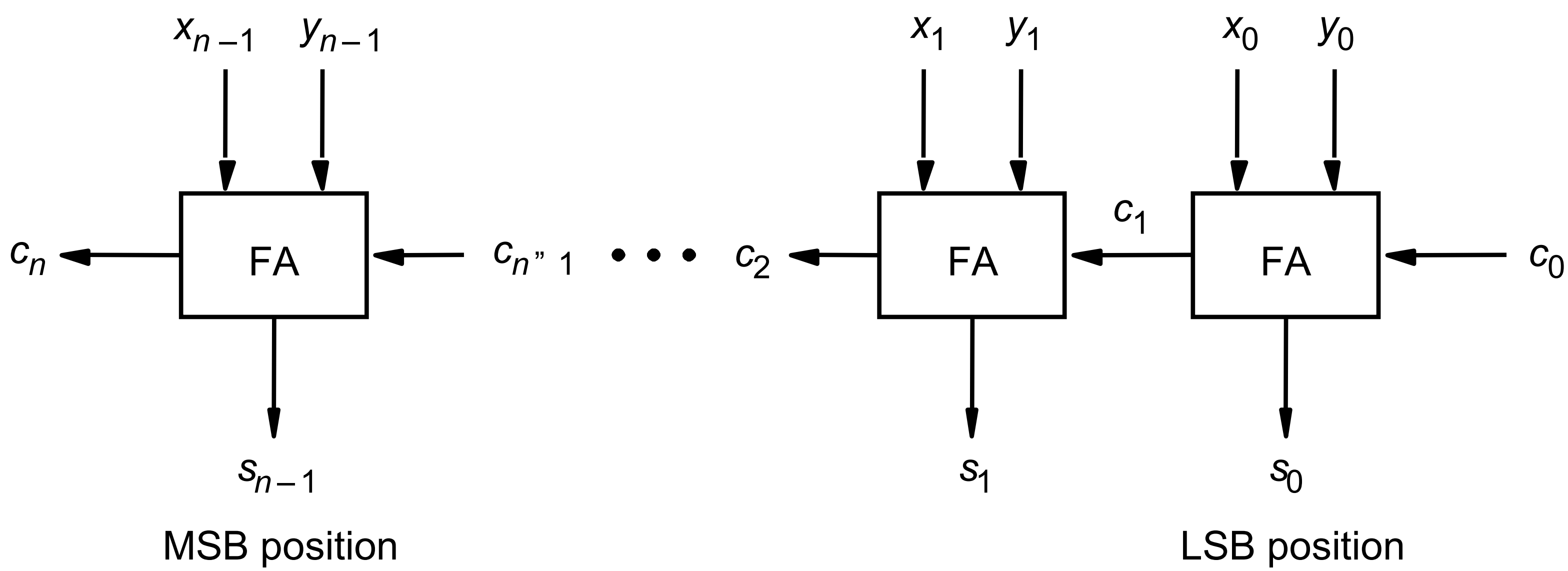
$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$

(b) Karnaugh maps



Somador Ripple-Carry construído a partir de Full-Adders

IC-UNICAMP



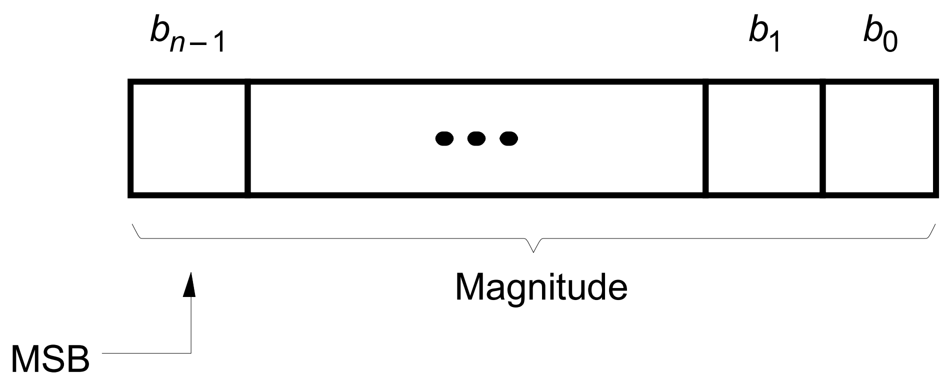
- obs: como X e Y são inteiros positivos, se Cn=1 houve overflow
  - resultado não cabe em n bits

IC-UNICAMP

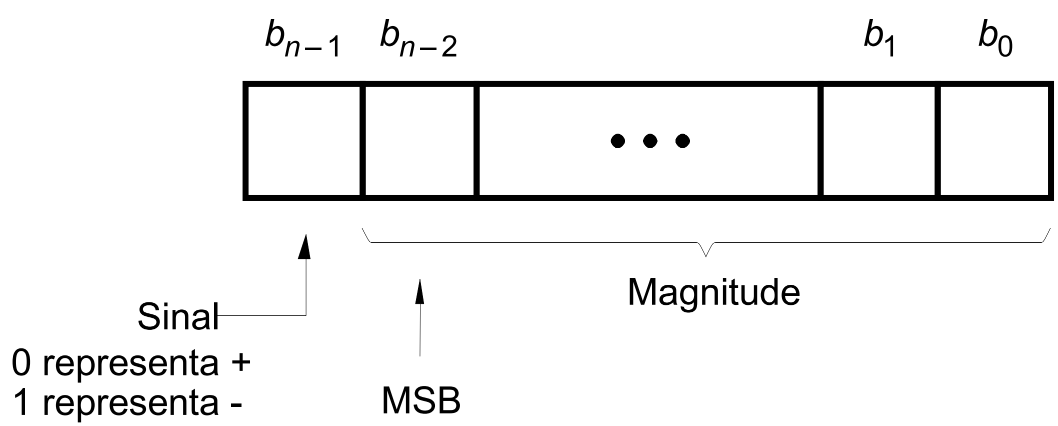
Representação de números negativos

- 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 signal, 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:

$$[-(2^{N-1}-1), 2^{N-1}-1]$$

Adição e Subtração Sinal e Magnitude

IC-UNICAMP

- Exemplo: -5 + 5

```
     1 1 0 1
+    0 1 0 1
   1 0 0 1 0
```

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

1000
0000

Complemento de 1

Complemento de 1 (K_1)

Em complemento de "Um" o número negativo K_1, com n-bits, é obtido subtraíndo 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 -> K_1 = ?

$$7 = (0111)_2$$
$$-7 = (1111)_2 - (0111)_2$$
$$-7 = (1000)_2$$

Complemento de 1

- Complemento de 1 (K_1)
  - Regra Prática

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

$$K_1 = 11\ldots11 - (p_{n-1} \ldots p_0)$$

$$K_1 = (\overline{p_{n-1}} \ldots \overline{p_0})$$

  - Intervalo de um número *N*-bit (mesmo que sinal/magnitude):

$$[-(2^{N-1}-1),\ 2^{N-1}-1]$$

IC-UNICAMP

Soma em complemento de 1

$$\begin{array}{r} (+5) \\ +\,(+2) \\ \hline (+7) \end{array} \qquad \begin{array}{r} 0\,1\,0\,1 \\ +\,0\,0\,1\,0 \\ \hline 0\,1\,1\,1 \end{array} \qquad\qquad \begin{array}{r} (-5) \\ +\,(+2) \\ \hline (-3) \end{array} \qquad \begin{array}{r} 1\,0\,1\,0 \\ +\,0\,0\,1\,0 \\ \hline 1\,1\,0\,0 \end{array}$$

$$\begin{array}{r} (+5) \\ +\,(-2) \\ \hline (+3) \end{array} \qquad \begin{array}{r} 0\,1\,0\,1 \\ +\,1\,1\,0\,1 \\ \hline 1\,0\,0\,1\,0 \\ 1 \\ \hline 0\,0\,1\,1 \end{array} \qquad\qquad \begin{array}{r} (-5) \\ +\,(-2) \\ \hline (-7) \end{array} \qquad \begin{array}{r} 1\,0\,1\,0 \\ +\,1\,1\,0\,1 \\ \hline 1\,0\,1\,1\,1 \\ 1 \\ \hline 1\,0\,0\,0 \end{array}$$

Complemento de 2

Complemento de 2 (K_2)

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

$$K_2 = 2^n - P$$

Exemplo: se n = 4 então:

$K_2 = 2^4 – P$
$K_2 = 16 – P$
$K_2 = (10000)_2 - P$

$P = 7 \rightarrow K_2 = ?$
$7 = (0111)_2$
$-7 = (10000)_2 - (0111)_2$
$-7 = (1001)_2$

IC-UNICAMP

Complemento de 2

- Complemento de 2 (K_2)
  - Regra Prática

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

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

$$K_2 = 11\ldots11 - (p_{n-1} \ldots p_0) + 1$$

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

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

IC-UNICAMP

b_3 b_2 b_1 b_0	Sinal Magnitude	Complemento de 1	Complemento de 2
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

IC-UNICAMP

Adição em K_2

$$\begin{array}{r} (+5) \\ +\ (+2) \\ \hline (+7) \end{array} \qquad \begin{array}{r} 0\,1\,0\,1 \\ +\ 0\,0\,1\,0 \\ \hline 0\,1\,1\,1 \end{array} \qquad\qquad \begin{array}{r} (-5) \\ +\ (+2) \\ \hline (-3) \end{array} \qquad \begin{array}{r} 1\,0\,1\,1 \\ +\ 0\,0\,1\,0 \\ \hline 1\,1\,0\,1 \end{array}$$

$$\begin{array}{r} (+5) \\ +\ (-2) \\ \hline (+3) \end{array} \qquad \begin{array}{r} 0\,1\,0\,1 \\ +\ 1\,1\,1\,0 \\ \hline 1\,0\,0\,1\,1 \end{array} \qquad\qquad \begin{array}{r} (-5) \\ +\ (-2) \\ \hline (-7) \end{array} \qquad \begin{array}{r} 1\,0\,1\,1 \\ +\ 1\,1\,1\,0 \\ \hline 1\,1\,0\,0\,1 \end{array}$$

ignore

ignore

IC-UNICAMP

Subtração em K_2

$$\begin{array}{r} (+5) \\ -\;(-2) \\ \hline (+7) \end{array} \qquad \begin{array}{r} 0\,1\,0\,1 \\ -\;1\,1\,1\,0 \\ \hline \end{array} \quad \Longrightarrow \quad \begin{array}{r} 0\,1\,0\,1 \\ +\;0\,0\,1\,0 \\ \hline 0\,1\,1\,1 \end{array}$$

$$\begin{array}{r} (+5) \\ -\;(+2) \\ \hline (+3) \end{array} \qquad \begin{array}{r} 0\,1\,0\,1 \\ -\;0\,0\,1\,0 \\ \hline \end{array} \quad \Longrightarrow \quad \begin{array}{r} 0\,1\,0\,1 \\ +\;1\,1\,1\,0 \\ \hline 1\,0\,0\,1\,1 \end{array}$$

ignore

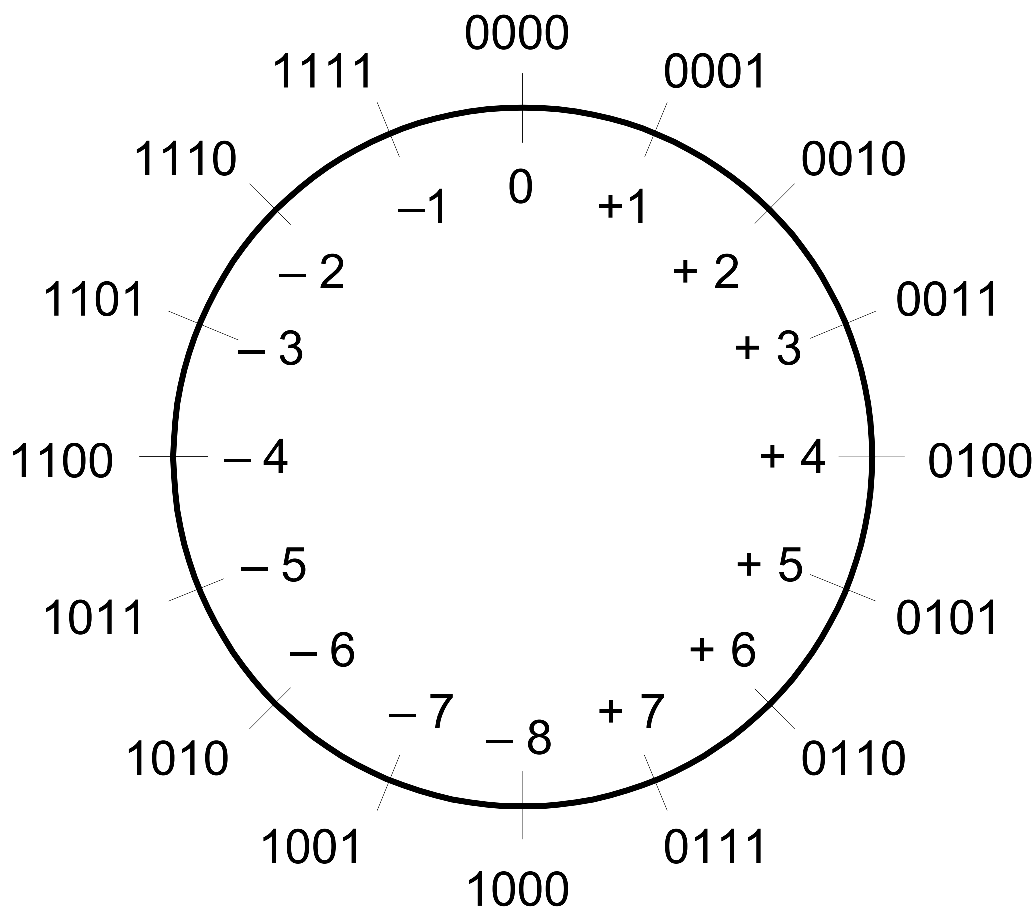
$$\begin{array}{r} (-5) \\ -\;(-2) \\ \hline (-3) \end{array} \qquad \begin{array}{r} 1\,0\,1\,1 \\ -\;1\,1\,1\,0 \\ \hline \end{array} \quad \Longrightarrow \quad \begin{array}{r} 1\,0\,1\,1 \\ +\;0\,0\,1\,0 \\ \hline 1\,1\,0\,1 \end{array}$$

$$\begin{array}{r} (-5) \\ -\;(+2) \\ \hline (-7) \end{array} \qquad \begin{array}{r} 1\,0\,1\,1 \\ -\;0\,0\,1\,0 \\ \hline \end{array} \quad \Longrightarrow \quad \begin{array}{r} 1\,0\,1\,1 \\ +\;1\,1\,1\,0 \\ \hline 1\,1\,0\,0\,1 \end{array}$$

ignore

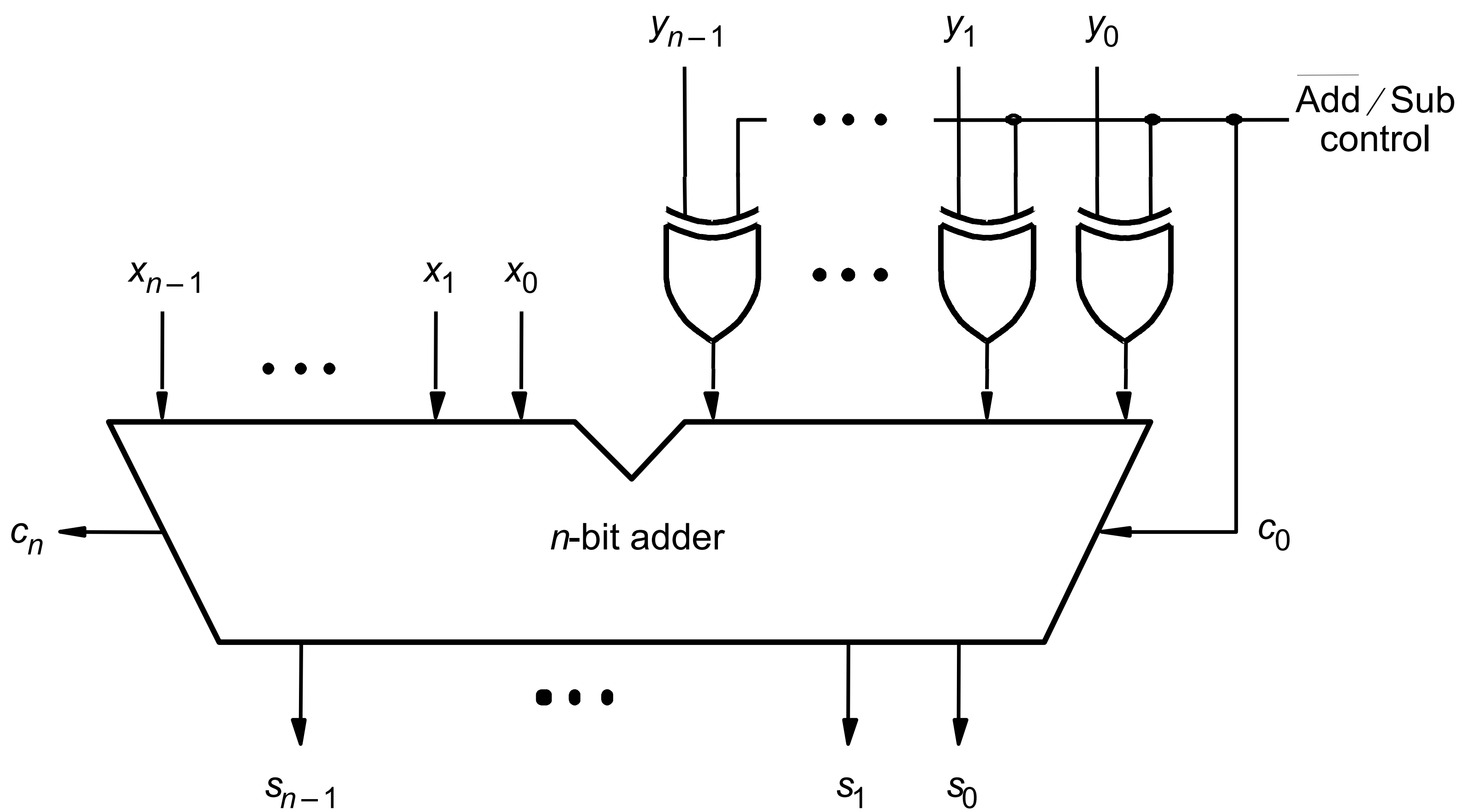
IC-UNICAMP

K2: interpretação gráfica



Somador/Subtrator

$$K_2 = (\overline{p}_{n-1} \ldots \overline{p}_0) + 1 = K_1(P) + 1$$



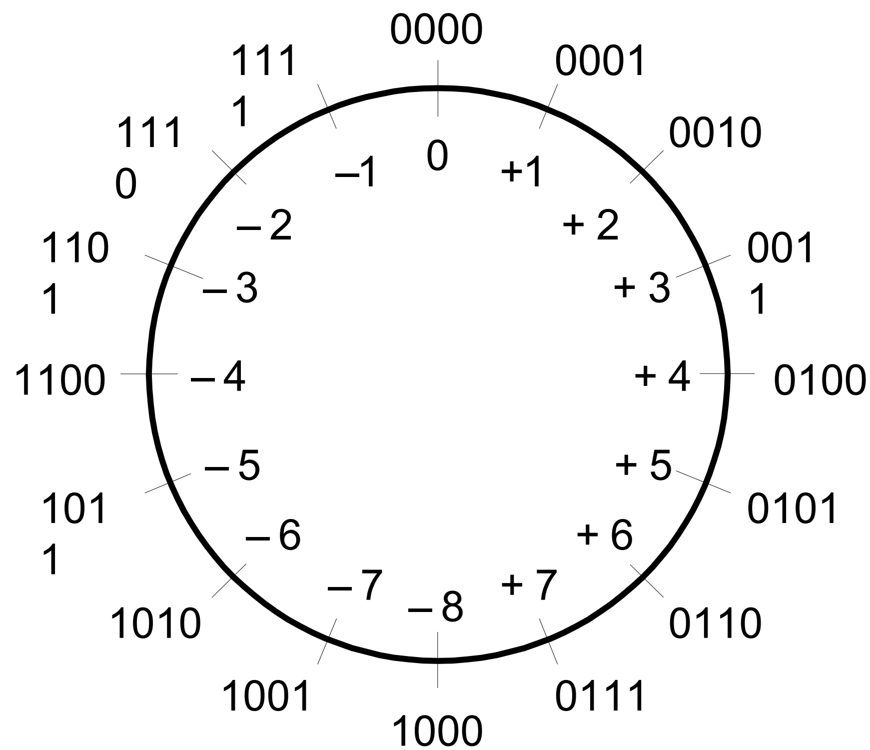
IC-UNICAMP

Overflow (Soma)

- A + B
  - sign(A) = sign(B): overflow possível
  - sign(A) ≠ sign(B): overflow impossivel

$$\begin{array}{r} (+7) \\ +\ (+2) \\ \hline (+9) \end{array} \qquad \begin{array}{r} 0\,1\,1\,1 \\ +\ 0\,0\,1\,0 \\ \hline 1\,0\,0\,1 \end{array}$$

$$\begin{array}{r} (-7) \\ +\ (+2) \\ \hline (-5) \end{array} \qquad \begin{array}{r} 1\,0\,0\,1 \\ +\ 0\,0\,1\,0 \\ \hline 1\,0\,1\,1 \end{array}$$



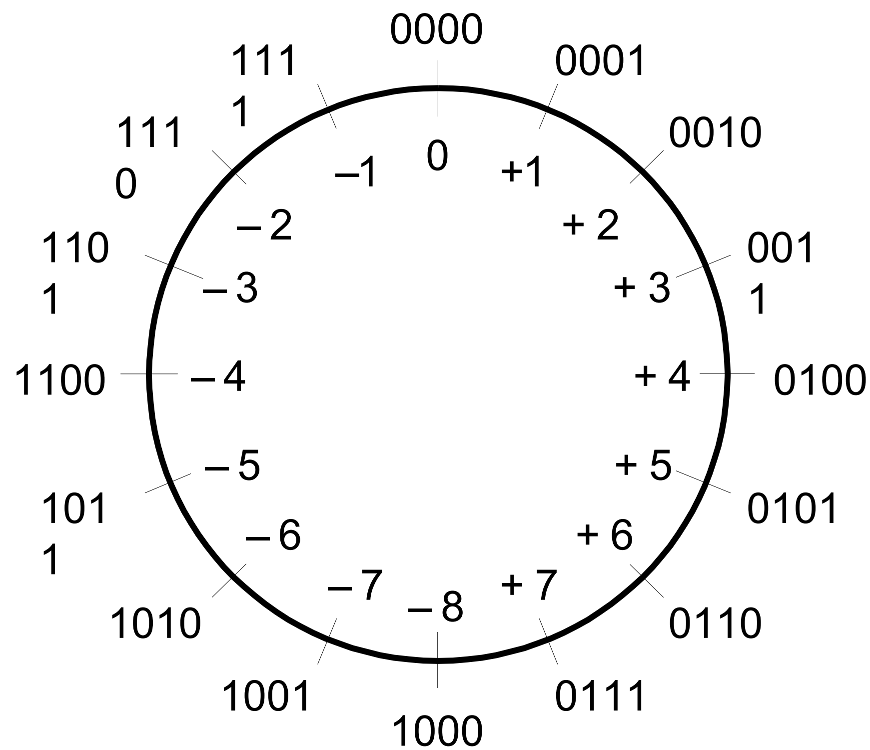
Overflow (Subtração)

IC-UNICAMP

- A - B = A + (-B), reduz à soma
  - sign(A) = sign(B): overflow impossível
  - sign(A) ≠ sign(B): overflow possível

(+ 7)	(+ 7)	0 1 1 1
- (+2)	+ (–2)	+ 1 1 1 0
(+ 5)	(+ 5)	1 0 1 0 1

(–7)	(–7)	1 0 0 1
- (+2)	+ (– 2)	+ 1 1 1 0
(–5)	(–9)	1 0 1 1 1



IC-UNICAMP

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

```
   1
  0 x..
+ 1 x..
-------
1 0 x..
```

```
   0
  0 x..
+ 1 x..
-------
0 1 x..
```

```
   1
  0 x..
+ 0 x..
-------
0 1 x..
```

```
   0
  0 x..
+ 0 x..
-------
0 0 x..
```

```
   0
  1 x..
+ 1 x..
-------
1 0 x..
```

```
   1
  1 x..
+ 1 x..
-------
1 1 x..
```

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

IC-UNICAMP

Overflow em K_2

$$\begin{array}{r} (+7) \\ +\ (+2) \\ \hline (+9) \end{array} \qquad \begin{array}{r} 0\,1\,1\,1 \\ +\ 0\,0\,1\,0 \\ \hline 1\,0\,0\,1 \end{array}$$

$c_4 = 0$
$c_3 = 1$

$$\begin{array}{r} (-7) \\ +\ (+2) \\ \hline (-5) \end{array} \qquad \begin{array}{r} 1\,0\,0\,1 \\ +\ 0\,0\,1\,0 \\ \hline 1\,0\,1\,1 \end{array}$$

$c_4 = 0$
$c_3 = 0$

$$\begin{array}{r} (+7) \\ +\ (-2) \\ \hline (+5) \end{array} \qquad \begin{array}{r} 0\,1\,1\,1 \\ +\ 1\,1\,1\,0 \\ \hline 1\,0\,1\,0\,1 \end{array}$$

$c_4 = 1$
$c_3 = 1$

$$\begin{array}{r} (-7) \\ +\ (-2) \\ \hline (-9) \end{array} \qquad \begin{array}{r} 1\,0\,0\,1 \\ +\ 1\,1\,1\,0 \\ \hline 1\,0\,1\,1\,1 \end{array}$$

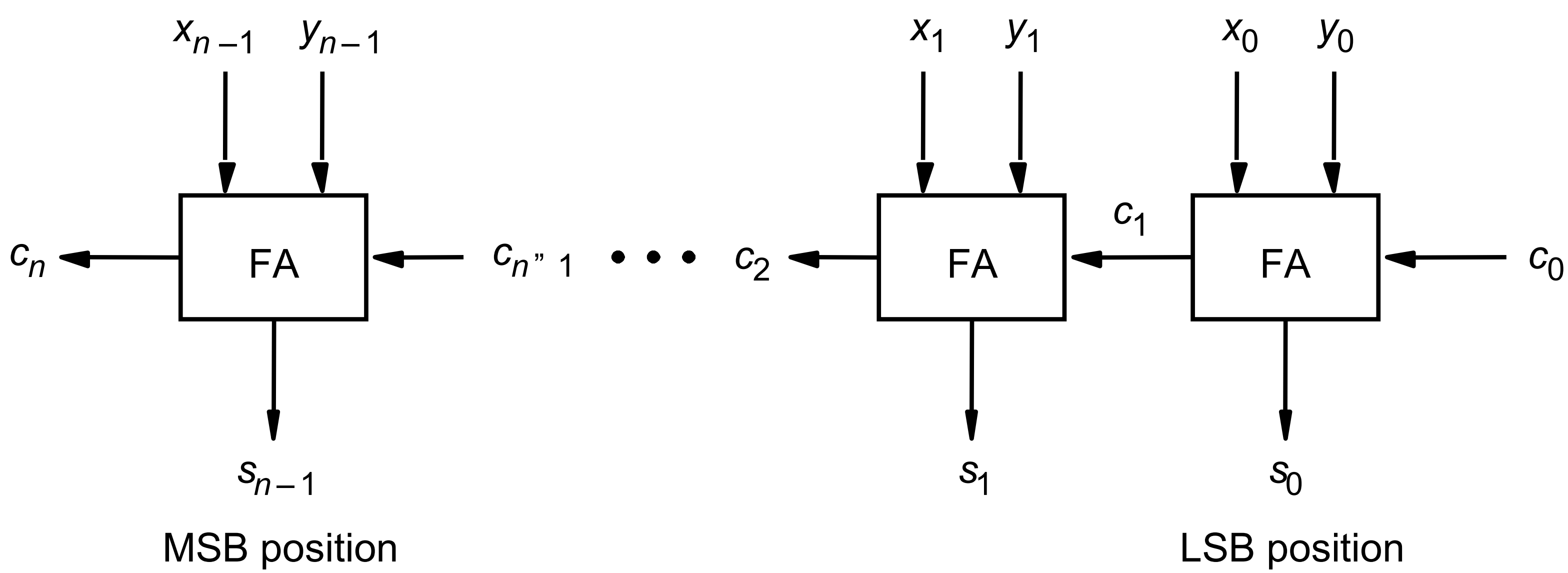
$c_4 = 1$
$c_3 = 0$

Somador Ripple Carry

- Atraso para um somador de n bits:

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

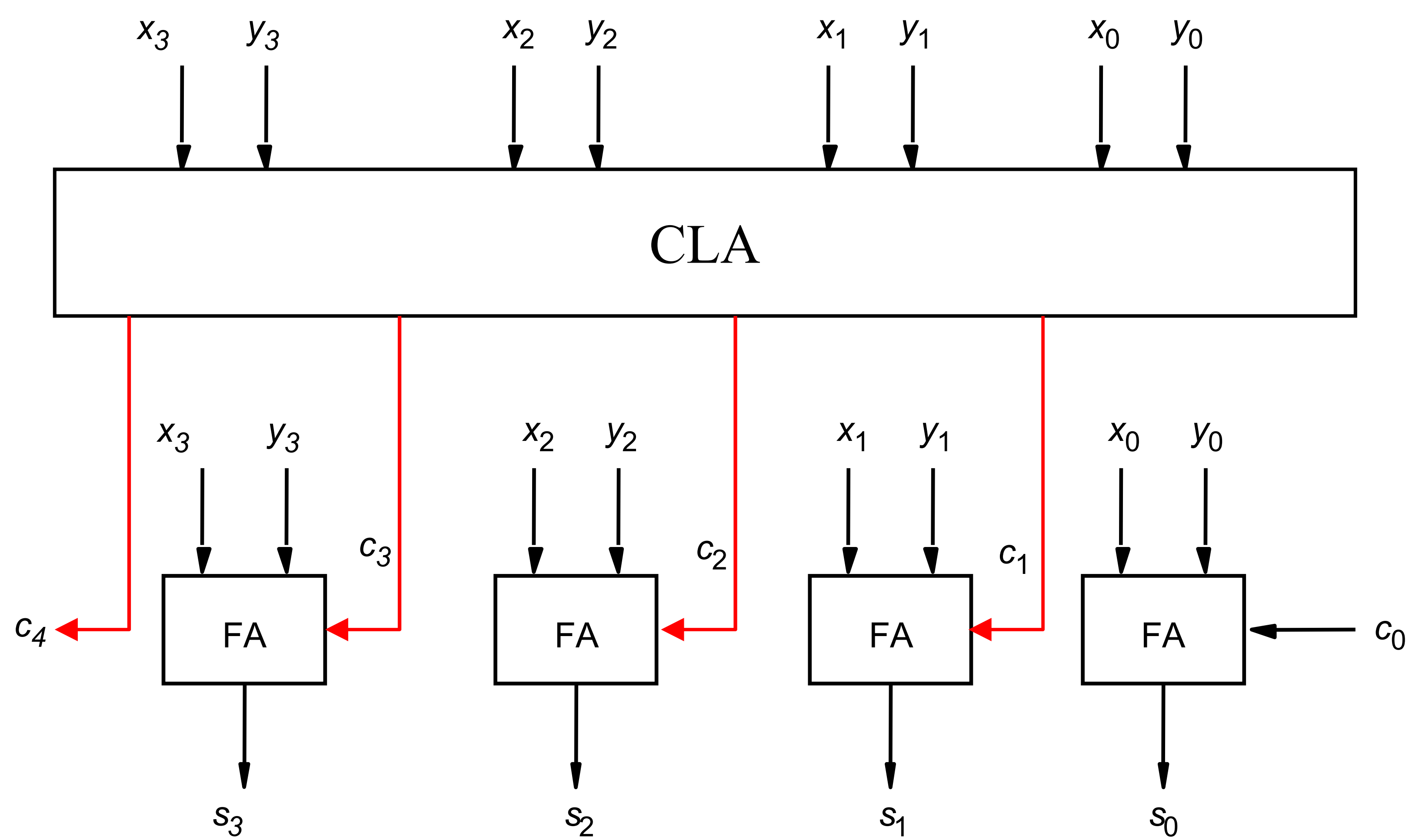
Onde t_{FA} é o atraso de um full adder



Antecipação de Carry: Carry Look Ahead (CLA)

IC-UNICAMP

- 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:
      - $\sim x_i \,.\, \sim y_i$
      - Que é exatamente $\sim p_i$

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

$c_1 = g_0 + p_0c_0$

$c_2 = g_1 + p_1c_1$ $\quad c_2 = g_1 + p_1\, g_0 + p_1\, p_0c_0$

$c_3 = g_2 + p_2c_2$ $\quad c_3 =$

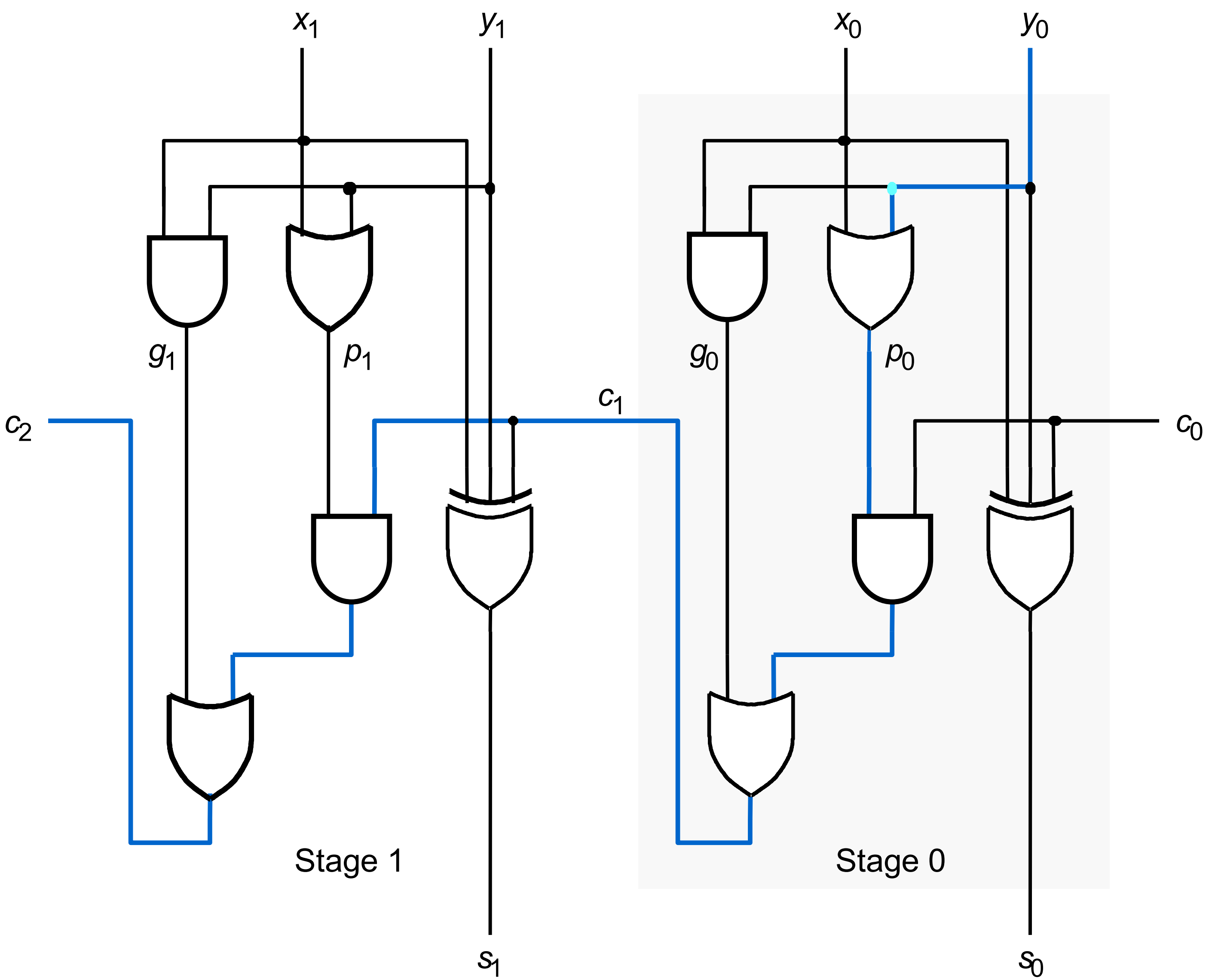
$c_4 = g_3 + p_3c_3$ $\quad c_4 =$

Atraso:

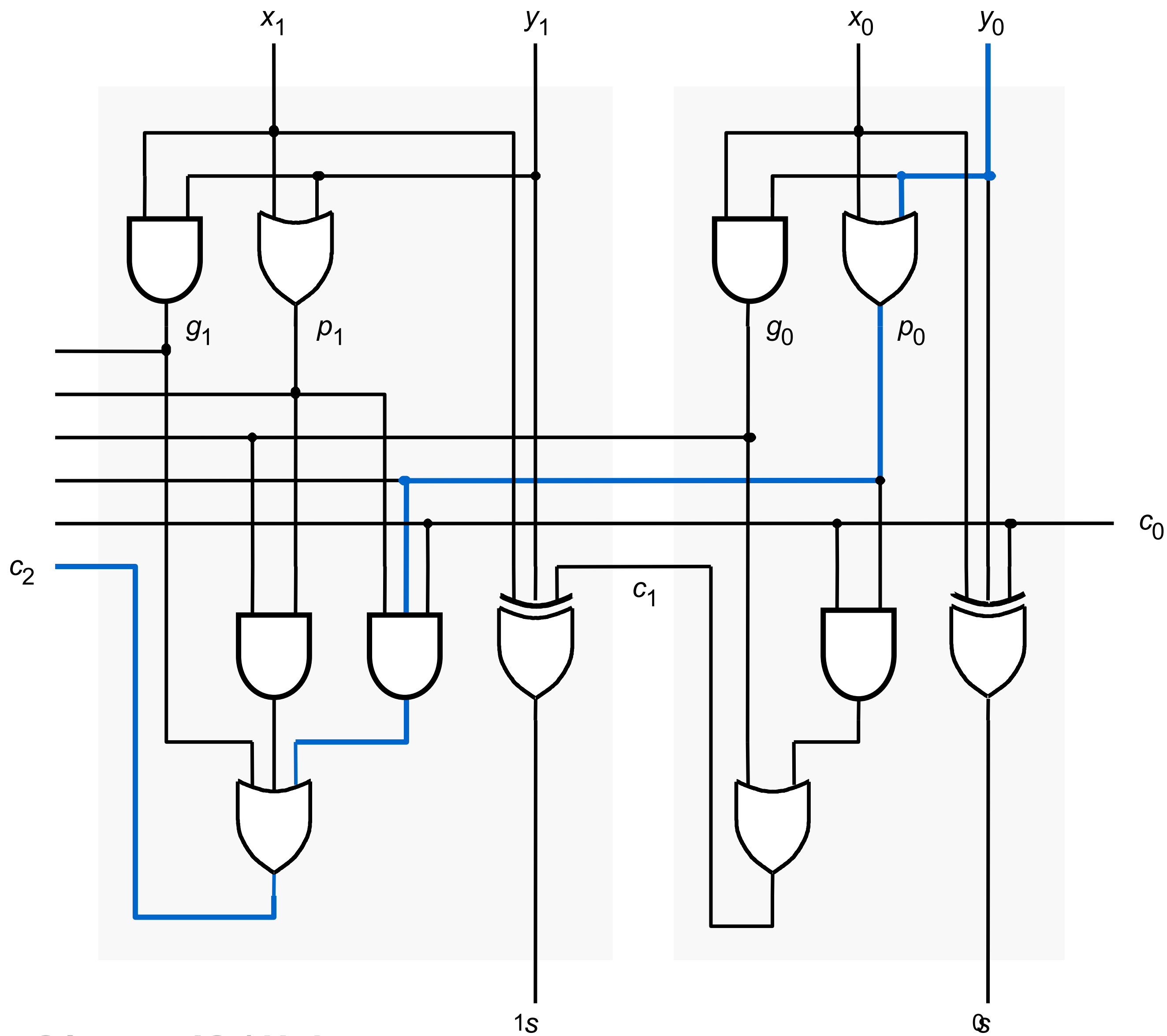
- entradas $\Rightarrow$ $g_i\, p_i$ (1G)
- $g_i\, p_i$ $\Rightarrow$ carry (2G) : 1 AND seguido de 1 OR
- carry $\Rightarrow$ saídas (2G)

- Total: 5G, independente de n
- Para nº bits > 4, ver abordagem hierárquica na seção 5.4.1

CLA: circuito p/ bits 0 e 1 carries encadeados

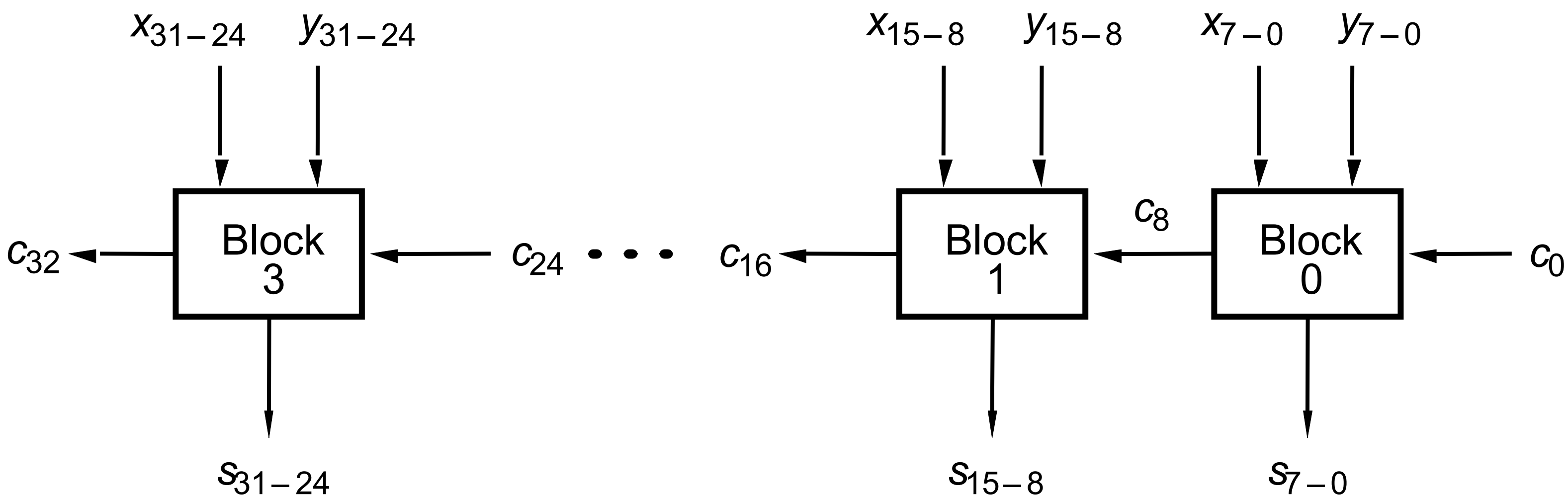


CLA: circuito p/ bits 0 e 1
circuito correto sem ripple



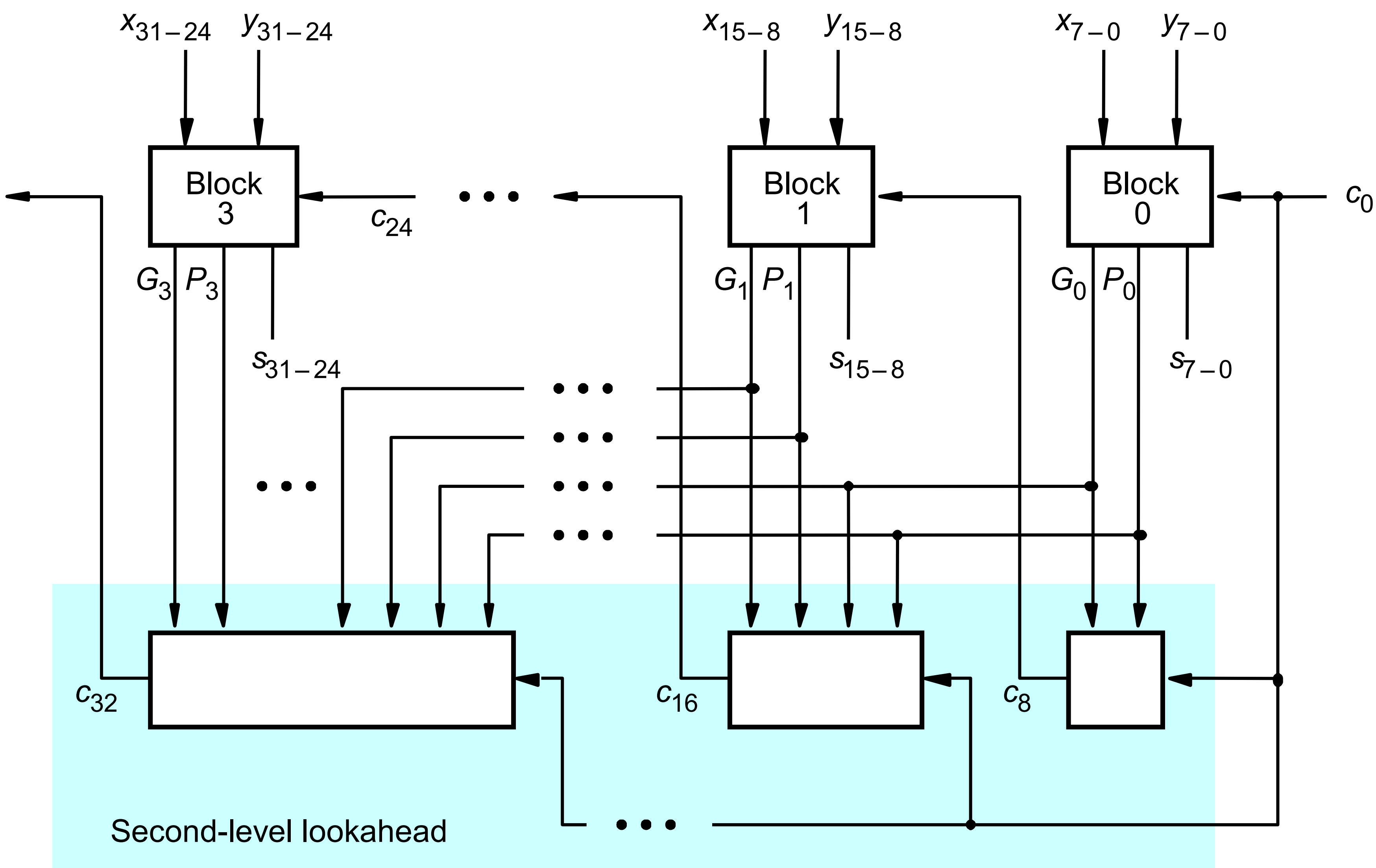
IC-UNICAMP

CLA hierárquico: com ripple-carry entre blocos



IC-UNICAMP

CLA hierárquico



IC-UNICAMP

Multiplicação de inteiros positivos

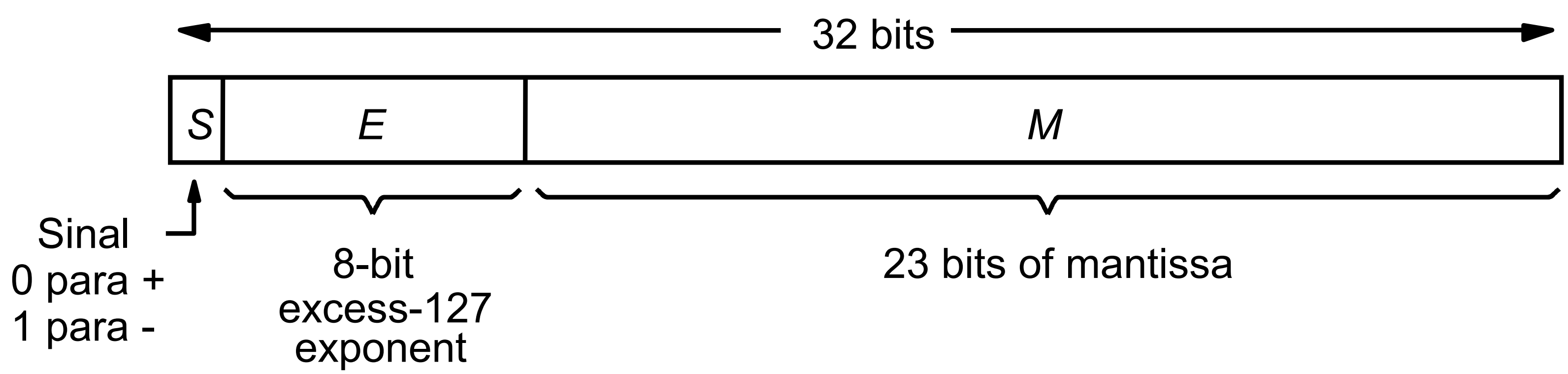
- Implementação totalmente combinacional (sequencial mais adiante) = mapeamento um-a-um com algoritmo convencional
  - implementação na seção 5.6.1

Multiplicando M	(11)	1 1 1 0
Multiplicador Q	(14)	´ 1 0 1 1
produto parcial 0		1 1 1 0
		+ 1 1 1 0
produto parcial 1		1 0 1 0 1
		+ 0 0 0 0
produto parcial 2		0 1 0 1 0
		+ 1 1 1 0
Produto P	(154)	1 0 0 1 1 0 1 0

IC-UNICAMP

Números fracionários: ponto flutuante

- N = mantissa * R $^{\text{expoente}}$
- Formato normalizado (IEEE 754 precisão simples):
  - mantissa = 1.xxxxxxxxxxxxx (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 10101000000000000000000



IC-UNICAMP

BCD: Binary Coded Decimal

- Exemplo:
  - $52_{10} = 0101\ 0010_{BCD}$
- Mais utilizado no passado em calculadoras
- Motivação: simplificar (ou evitar) conversões dec→bin na entrada de dados e bin→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

IC-UNICAMP

X	0 1 1 1	7
+ Y	+ 0 1 0 1	+ 5
Z	1 1 0 0	12
	+ 0 1 1 0	
carry →	1 0 0 1 0	

$\underbrace{0010}_{S = 2}$

X	1 0 0 0	8
+ Y	+ 1 0 0 1	+ 9
Z	1 0 0 0 1	17
	+ 0 1 1 0	
carry →	1 0 1 1 1	

$\underbrace{0111}_{S = 7}$

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

IC-UNICAMP

ASCII

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

0	NUL	16	DLE	32	SP	48	0	64	@	80	P	96	`	112	p
1	SOH	17	DC1	33	!	49	1	65	A	81	Q	97	a	113	q
2	ST	18	DC2	34	"	50	2	66	B	82	R	98	b	114	r
3	ET	19	DC3	35	#	51	3	67	C	83	S	99	c	115	s
4	EOT	20	DC4	36	$	52	4	68	D	84	T	100	d	116	t
5	ENQ	21	NAK	37	%	53	5	69	E	85	U	101	e	117	u
6	ACK	22	SYN	38	&	54	6	70	F	86	V	102	f	118	v
7	BEL	23	ETB	39	'	55	7	71	G	87	W	103	g	119	w
8	BS	24	CAN	40		56	8	72	H	88		104	h	120	
9	HT	25	EM	41	)	57	9	73	I	89	Y	105	i	121	y
10	LF	26	SUB	42	*	58	:	74	J	90	Z	106	j	122	z
11	VT	27	ESC	43	+	59	;	75	K	91	[	107	k	123	{
12	FF	28	FS	44	,	60	<	76	L	92	\	108	l	124	|
13	CR	29	GS	45	-	61	=	77	M	93	]	109	m	125	}
14	SO	30	RS	46	.	62	>	78	N	94	^	110	n	126	~
15	SI	31	US	47	/	63	?	79	O	95	_	111	o	127	DEL