



MC 613

IC/Unicamp

2013s1

Prof Guido Araújo

Prof Mario Côrtes

Circuitos Aritméticos

Tópicos

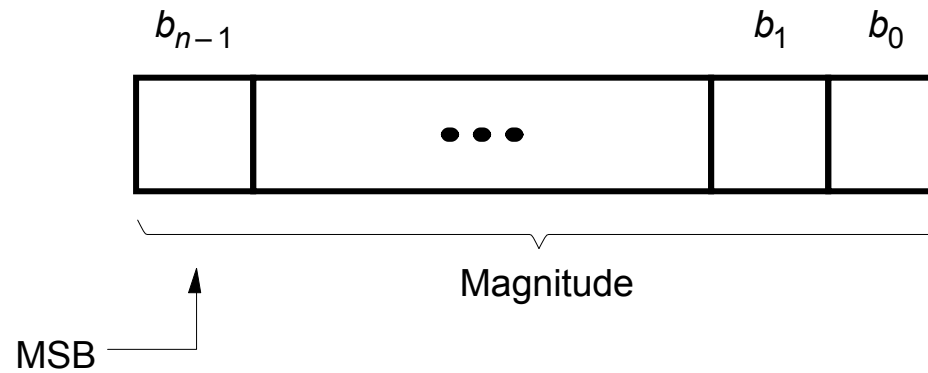
- Representação sinal-magnitude
- Representação K_1 e K_2
- Somadores half-adder e full-adder
- Somador/subtrator
- Somador com overflow

Representação de Números Negativos

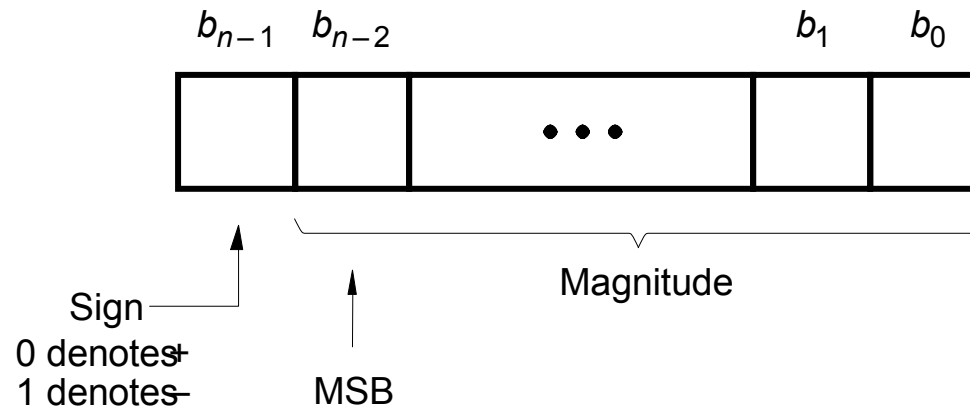
b_3 b_2 b_1 b_0	Sinal Magnitude	Complemento de 1	Complemento de 2
0111	+7	7	+7
0110	+6	6	+6
0101	+5	5	+5
0100	+4	4	+4
0011	+3	3	+3
0010	+2	2	+2
0001	+1	1	+1
0000	+0	0	0
1000	-0	-7	-8
1001	-1	-6	-7
1010	-2	-5	-6
1011	-3	-4	-5
1100	-4	-3	-4
1101	-5	-2	-3
1110	-6	-1	-2
1111	-7	-0	-1

Table 5.2 Interpretation of four-bit signed integers

Representação Sinal Magnitude



(a) Unsigned number



(b) Signed number

Figure 5.8 Formats for representation of integers

Representação de Números Negativos

- 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 ± 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

- Exemplo: $-5 + 5$

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

- Duas representações para 0 (± 0):

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

Representação de Números Negativos

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:

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

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

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

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

$$7 = (0111)_2$$

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

$$-7 = (1000)_2$$



Representação de Números Negativos

- Complemento de 1 (K_1)
 - Regra Prática

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

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

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

Representação de Números Negativos

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:

$$K_2 = 2^4 - P$$

$$K_2 = 16 - P$$

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

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

$$7 = (0111)_2$$

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

$$-7 = (1001)_2$$

Representação de Números Negativos

- Complemento de 2 (K_2)
 - Regra Prática

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

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

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



Representação de Números Negativos

- 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: $[-2^{N-1}, 2^{N-1}-1]$

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

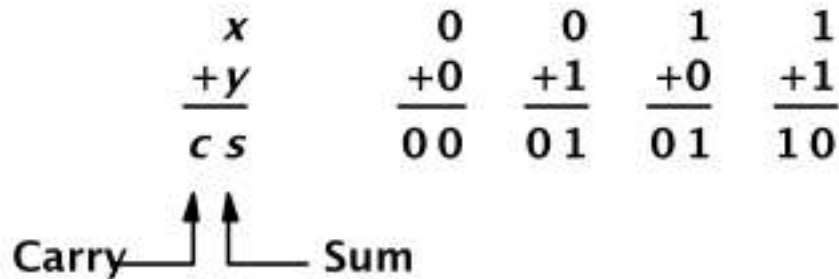
Figure 5.10 Examples of 2's complement addition

Subtração em K_2

$\begin{array}{r} (+5) \quad 0101 \\ - (-2) \quad \underline{1110} \\ \hline (+7) \end{array}$	\Rightarrow	$\begin{array}{r} 0101 \\ + 0010 \\ \hline 0111 \end{array}$	$\begin{array}{r} (+5) \quad 0101 \\ - (+2) \quad \underline{0010} \\ \hline (+3) \end{array}$	\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	$\begin{array}{r} 1011 \\ + 0010 \\ \hline 1101 \end{array}$	$\begin{array}{r} (-5) \quad 1011 \\ - (+2) \quad \underline{0010} \\ \hline (-7) \end{array}$	\Rightarrow	$\begin{array}{r} 1011 \\ + 1110 \\ \hline 11001 \\ \uparrow \\ \text{ignore} \end{array}$

Figure 5.11 Examples of 2's complement subtraction

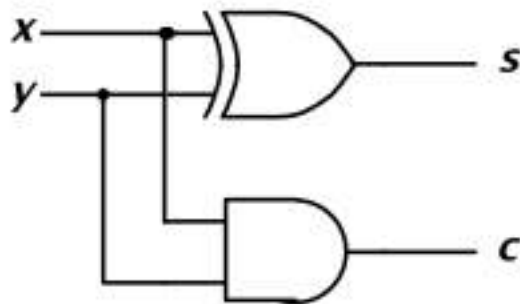
Half-adder



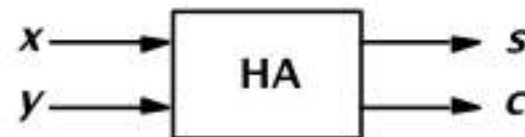
(a) The four possible cases

x	y	Carry c	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

(b) Truth table

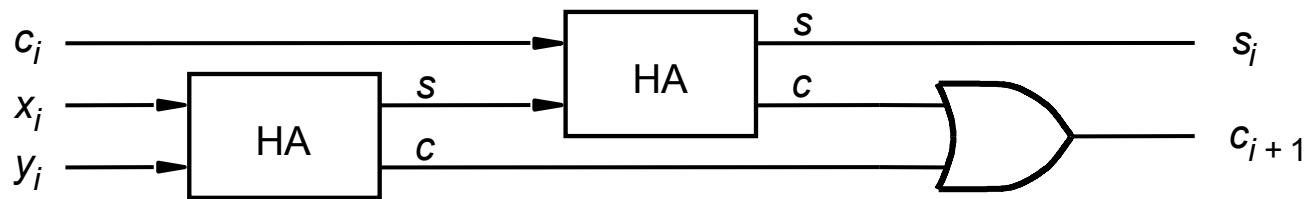


(c) Circuit

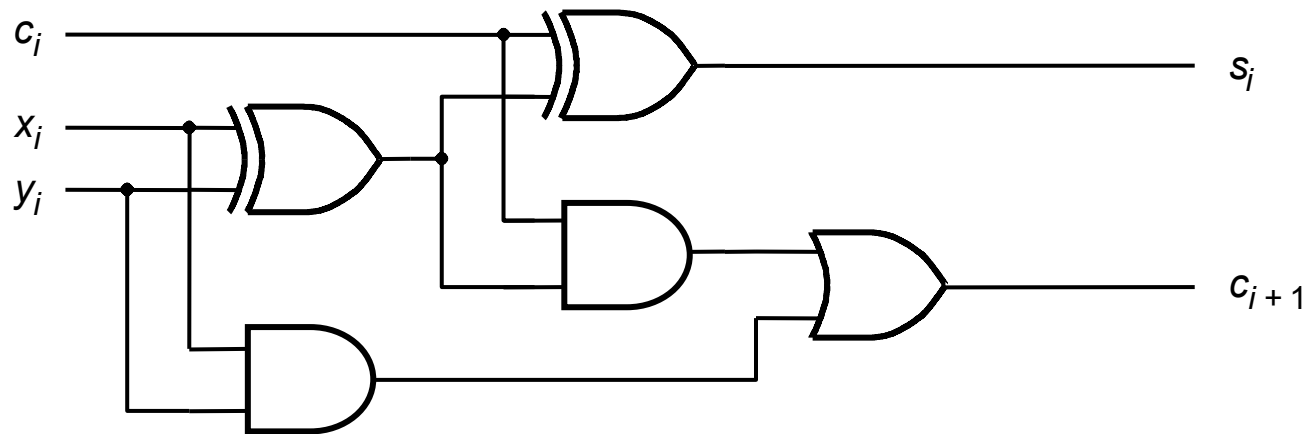


(d) Graphical symbol

Somador com Half-adder



(a) Block diagram



(b) Detailed diagram

Figure 5.5 A decomposed implementation of the full-adder circuit

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

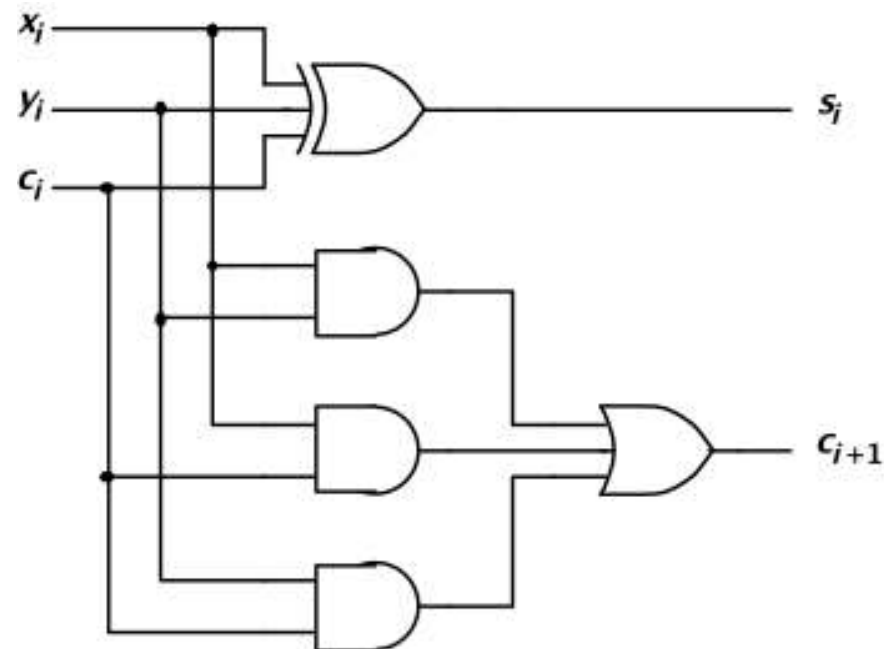
$c_i \backslash x_i y_i$	00	01	11	10
0		1		1
1	1		1	

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

$c_i \backslash x_i y_i$	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



Full-adder (VHDL)

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY fulladd IS
    PORT ( Cin, x, y : IN      STD_LOGIC ;
          s, Cout : OUT     STD_LOGIC ) ;
END fulladd ;

ARCHITECTURE LogicFunc OF fulladd IS
BEGIN
    s <= x XOR y XOR Cin ;
    Cout <= (x AND y) OR (Cin AND x) OR (Cin AND y) ;
END LogicFunc ;
```

Figure 5.23 VHDL code for the full-adder



Full-adder Package (VHDL)

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

PACKAGE fulladd_package IS
  COMPONENT fulladd
    PORT (Cin, x, y      : IN  STD_LOGIC ;
          s, Cout       : OUT  STD_LOGIC ) ;
  END COMPONENT ;
END fulladd_package ;
```

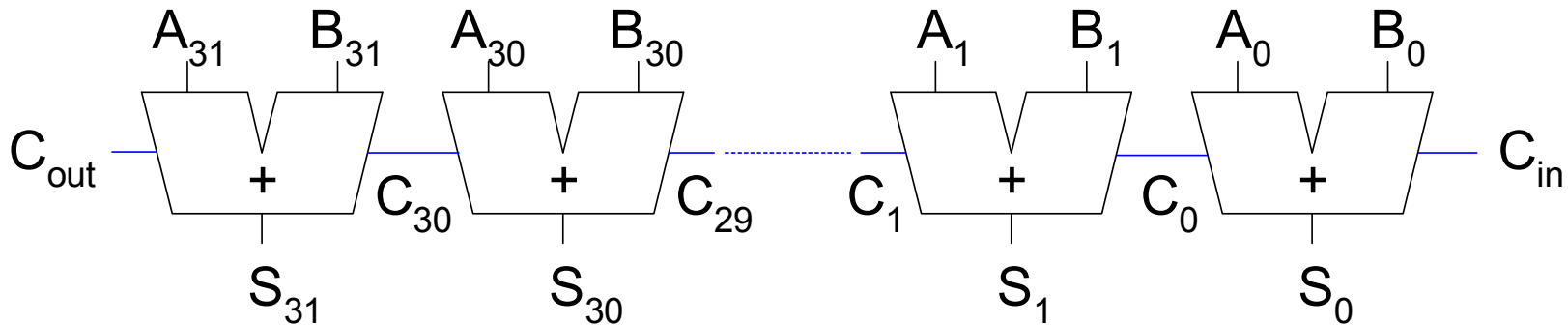
Figure 5.25 Declaration of a package

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





4-bit Ripple Carry Adder (sinais)

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE work.fulladd_package.all ;

ENTITY adder4 IS
    PORT ( Cin           : IN     STD_LOGIC ;
          x3, x2, x1, x0 : IN     STD_LOGIC ;
          y3, y2, y1, y0 : IN     STD_LOGIC ;
          s3, s2, s1, s0 : OUT    STD_LOGIC ;
          Cout           : OUT    STD_LOGIC ) ;
END adder4 ;

ARCHITECTURE Structure OF adder4 IS
    SIGNAL c1, c2, c3 : STD_LOGIC ;
BEGIN
    stage0: fulladd PORT MAP ( Cin, x0, y0, s0, c1 ) ;
    stage1: fulladd PORT MAP ( c1, x1, y1, s1, c2 ) ;
    stage2: fulladd PORT MAP ( c2, x2, y2, s2, c3 ) ;
    stage3: fulladd PORT MAP (
        x => x3, y => y3, Cin => c3, Cout => Cout, s => s3 ) ;
END Structure ;
```

Figure 5.26 Using a package for the four-bit adder



4-bit Ripple Carry Adder (vetores)

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE work.fulladd_package.all ;

ENTITY adder4 IS
    PORT (Cin      : IN      STD_LOGIC ;
          X, Y     : IN      STD_LOGIC_VECTOR(3 DOWNTO 0) ;
          S        : OUT     STD_LOGIC_VECTOR(3 DOWNTO 0) ;
          Cout     : OUT     STD_LOGIC ) ;
END adder4 ;

ARCHITECTURE Structure OF adder4 IS
    SIGNAL C : STD_LOGIC_VECTOR(1 TO 3) ;
BEGIN
    stage0: fulladd PORT MAP ( Cin, X(0), Y(0), S(0), C(1) ) ;
    stage1: fulladd PORT MAP ( C(1), X(1), Y(1), S(1), C(2) ) ;
    stage2: fulladd PORT MAP ( C(2), X(2), Y(2), S(2), C(3) ) ;
    stage3: fulladd PORT MAP ( C(3), X(3), Y(3), S(3), Cout ) ;
END Structure ;
```

Figure 5.27 A four-bit adder defined using multibit signals

Descrição Comportamental

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_signed.all ;

ENTITY adder16 IS
    PORT ( X, Y : IN     STD_LOGIC_VECTOR(15 DOWNT0 0) ;
          S      : OUT   STD_LOGIC_VECTOR(15 DOWNT0 0) ) ;
END adder16 ;

ARCHITECTURE Behavior OF adder16 IS
BEGIN
    S <= X + Y ;
END Behavior ;
```

Figure 5.28 VHDL code for a 16-bit adder

Somador/Subtrator

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

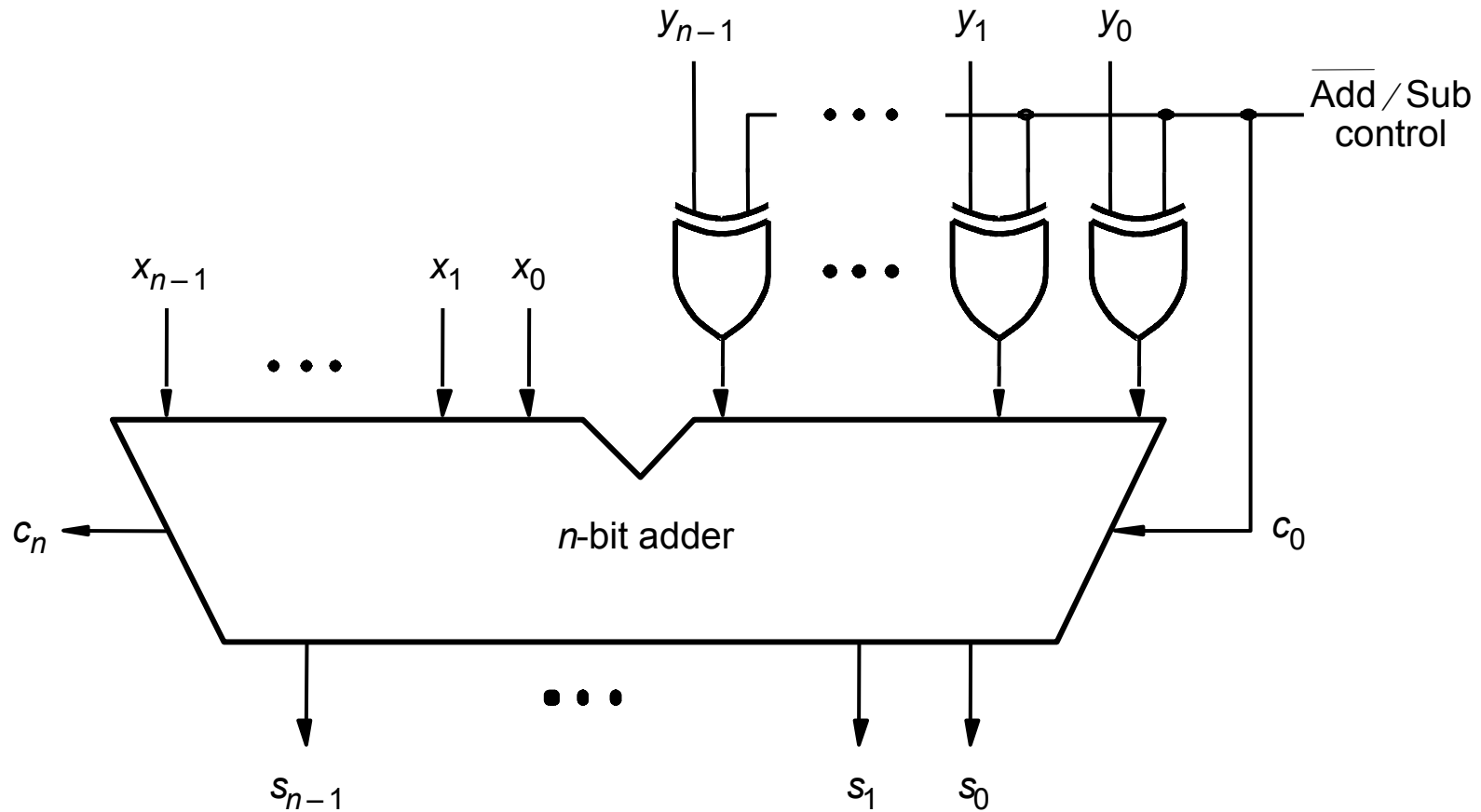


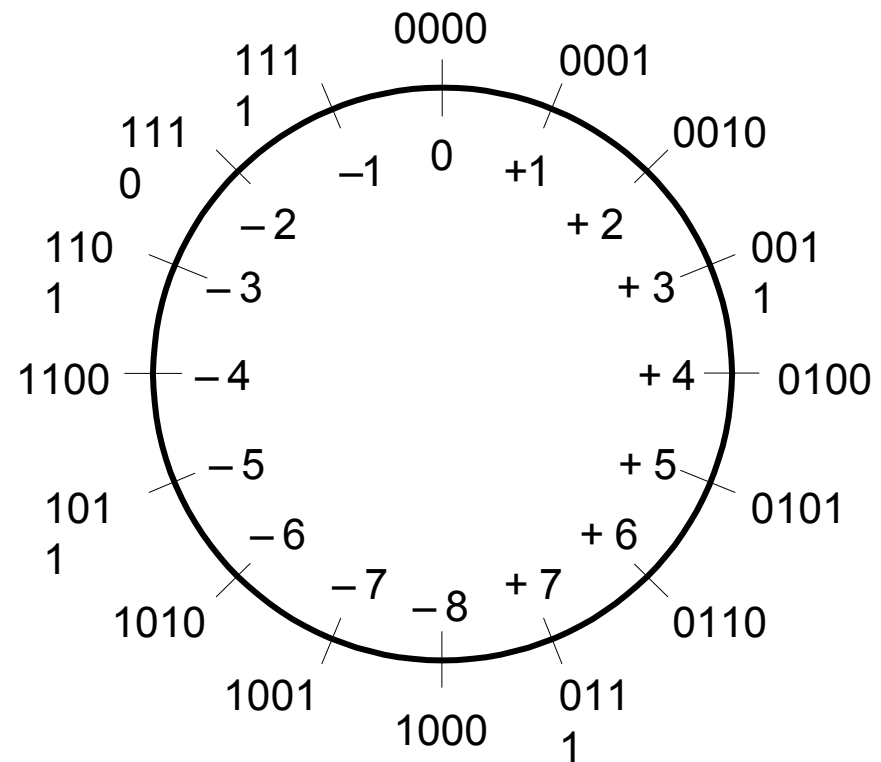
Figure 5.13 Adder/subtractor unit

Overflow (Soma)

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

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

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

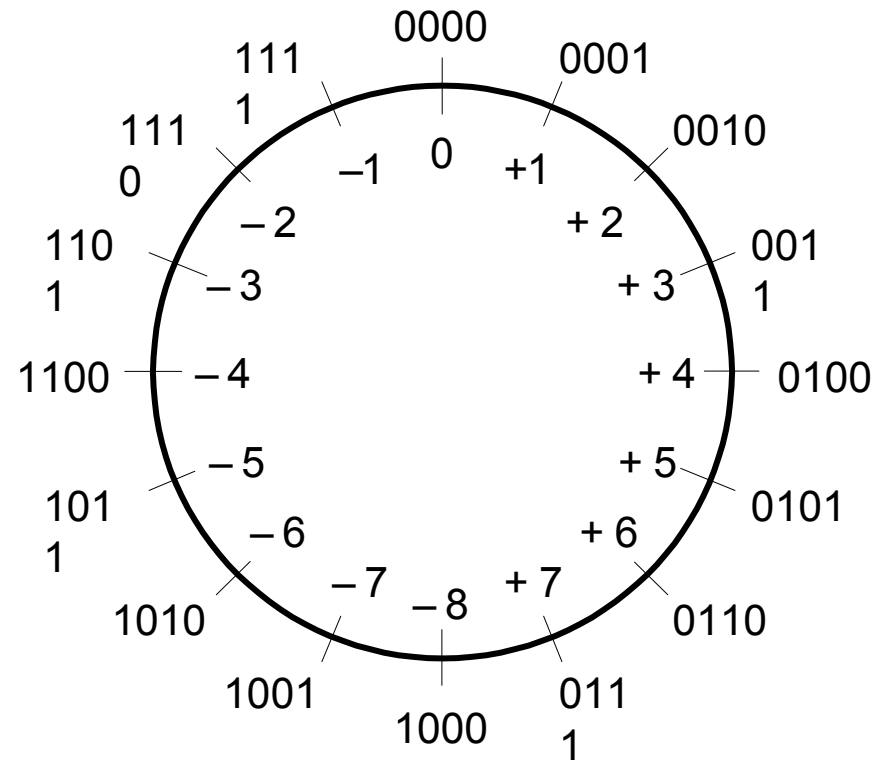


Overflow (Subtração)

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

$\begin{array}{r} (+7) \\ - (+2) \\ \hline (+5) \end{array}$	$\begin{array}{r} (+7) \\ + (-2) \\ \hline (+5) \end{array}$	$\begin{array}{r} 0111 \\ + 1110 \\ \hline 10101 \end{array}$
--	--	---

$\begin{array}{r} (-7) \\ - (+2) \\ \hline (-5) \end{array}$	$\begin{array}{r} (-7) \\ + (-2) \\ \hline (-9) \end{array}$	$\begin{array}{r} 1001 \\ + 1110 \\ \hline 10111 \end{array}$
--	--	---



Resumo Overflow

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

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

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

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

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

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

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

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

Overflow em K_2

(+ 7)	0 1 1 1	(-7)	1 0 0 1
+ (+ 2)	+ 0 0 1 0	+ (+ 2)	+ 0 0 1 0
<u> </u>	<u> </u>	<u> </u>	<u> </u>
(+ 9)	1 0 0 1	(-5)	1 0 1 1
	$c_4 = 0$		$c_4 = 0$
	$c_3 = 1$		$c_3 = 0$
(+ 7)	0 1 1 1	(-7)	1 0 0 1
+ (-2)	+ 1 1 1 0	+ (-2)	+ 1 1 1 0
<u> </u>	<u> </u>	<u> </u>	<u> </u>
(+ 5)	1 0 1 0 1	(-9)	1 0 1 1 1
	$c_4 = 1$		$c_4 = 1$
	$c_3 = 1$		$c_3 = 0$

Figure 5.14 Examples of determination of overflow



4-bit Ripple Carry Adder (vetores) + overflow

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE work.fulladd_package.all ;

ENTITY adder4 IS
    PORT (Cin      : IN      STD_LOGIC ;
          X, Y     : IN      STD_LOGIC_VECTOR(3 DOWNTO 0) ;
          S        : OUT     STD_LOGIC_VECTOR(3 DOWNTO 0) ;
          Cout, Overflow : OUT  STD_LOGIC ) ;
END adder4 ;

ARCHITECTURE Structure OF adder4 IS
    SIGNAL C : STD_LOGIC_VECTOR(1 TO 4) ;
BEGIN
    stage0: fulladd PORT MAP ( Cin, X(0), Y(0), S(0), C(1) ) ;
    stage1: fulladd PORT MAP ( C(1), X(1), Y(1), S(1), C(2) ) ;
    stage2: fulladd PORT MAP ( C(2), X(2), Y(2), S(2), C(3) ) ;
    stage3: fulladd PORT MAP ( C(3), X(3), Y(3), S(3), C(4) ) ;
    Overflow <= C(3) XOR C(4) ;
    Cout <= C(4) ;
END Structure ;
```

Descrição Comportamental

Como incluir overflow?

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_signed.all ;

ENTITY adder16 IS
    PORT ( X, Y : IN     STD_LOGIC_VECTOR(15 DOWNT0 0) ;
          S      : OUT   STD_LOGIC_VECTOR(15 DOWNT0 0) ) ;
END adder16 ;

ARCHITECTURE Behavior OF adder16 IS
BEGIN
    S <= X + Y ;
END Behavior ;
```

Figure 5.28 VHDL code for a 16-bit adder



IC-UNICAMP

16-bit Adder com Overflow

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_signed.all ;

ENTITY adder16 IS
    PORT ( Cin          : IN    STD_LOGIC ;
          X, Y          : IN    STD_LOGIC_VECTOR(15 DOWNT0 0) ;
          S             : OUT   STD_LOGIC_VECTOR(15 DOWNT0 0) ;
          Cout,Overflow : OUT   STD_LOGIC ) ;
END adder16 ;

ARCHITECTURE Behavior OF adder16 IS
    SIGNAL Sum : STD_LOGIC_VECTOR(16 DOWNT0 0) ;
BEGIN
    Sum <= ('0' & X) + Y + Cin ;
    S <= Sum(15 DOWNT0 0) ;
    Cout <= Sum(16) ;
    Overflow <= Sum(16) XOR X(15) XOR Y(15) XOR Sum(15) ;
END Behavior ;
```

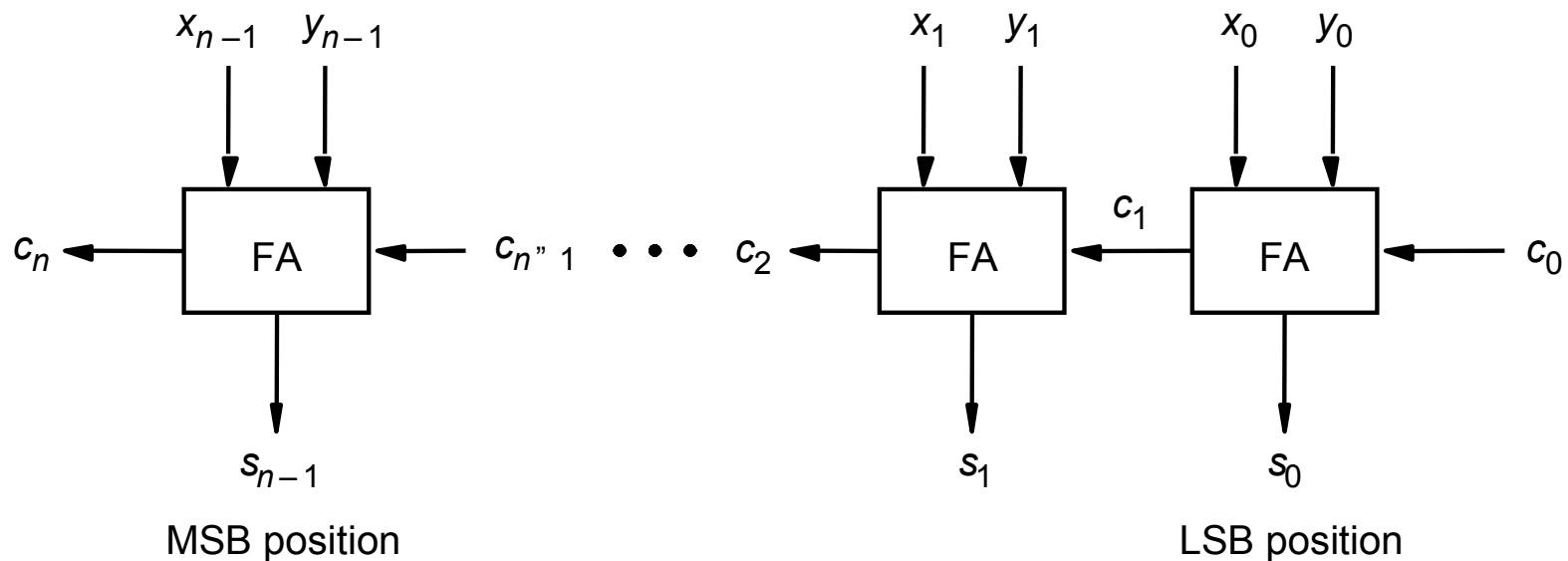
Figure 5.29 A 16-bit adder with carry and overflow

Somador Ripple Carry

- Atraso para um somador de n bits:

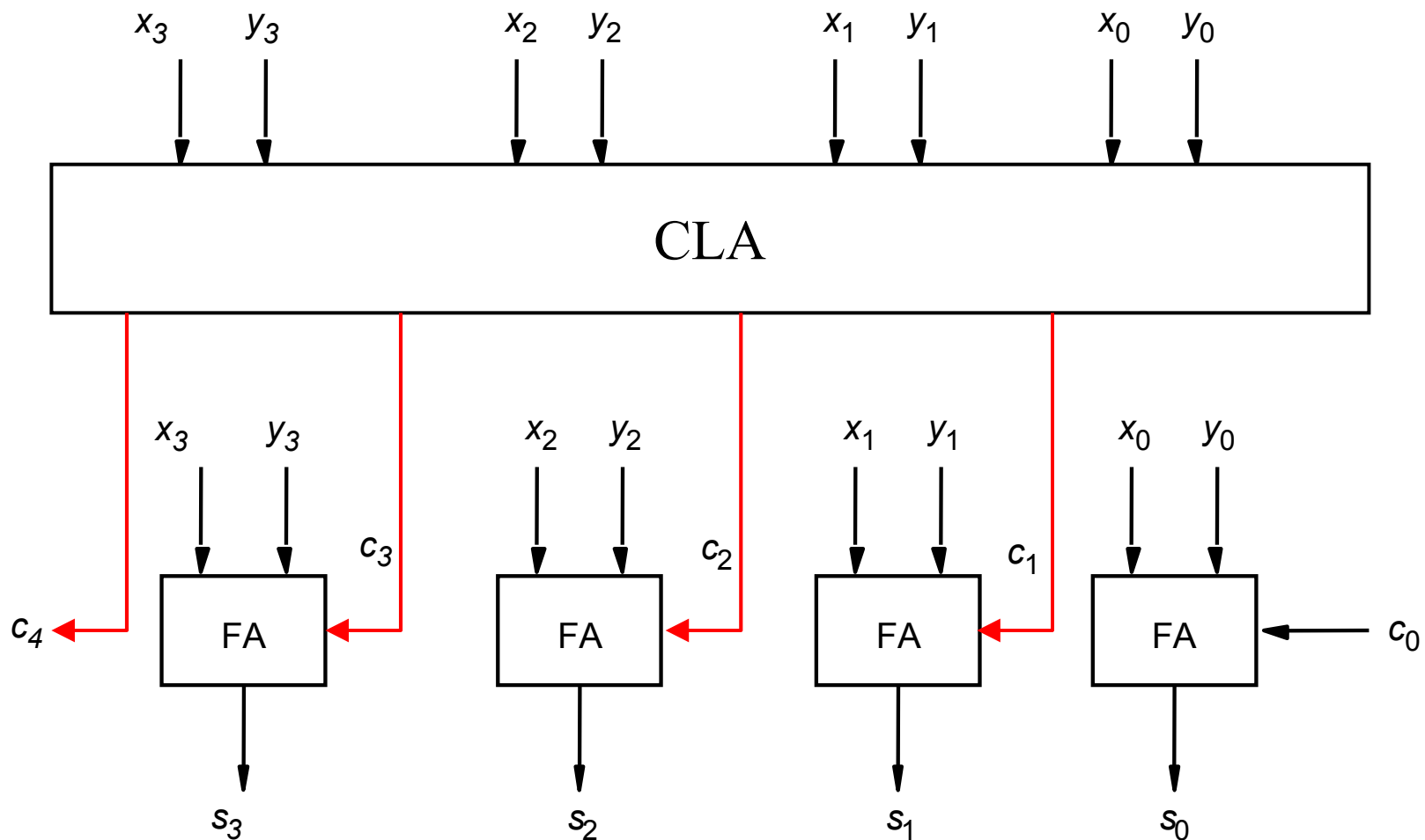
$$t_{\text{ripple}} = Nt_{FA}$$

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

Codificação em BCD

“No mundo há **10** tipos de pessoas: as que sabem contar em binário e as que não sabem”

BCD

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

Table 5.3 Binary-coded decimal digits

Adição Usando BCD

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

$$\begin{array}{r}
 X \quad \quad 1000 \quad \quad 8 \\
 + Y \quad + 1001 \quad + 9 \\
 \hline
 Z \quad \quad 10001 \quad \quad 17 \\
 \quad \quad + 0110 \\
 \hline
 \text{carry} \rightarrow 10111 \\
 \quad \quad \underbrace{\hspace{2cm}} \\
 \quad \quad S = 7
 \end{array}$$

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

Somador em BCD

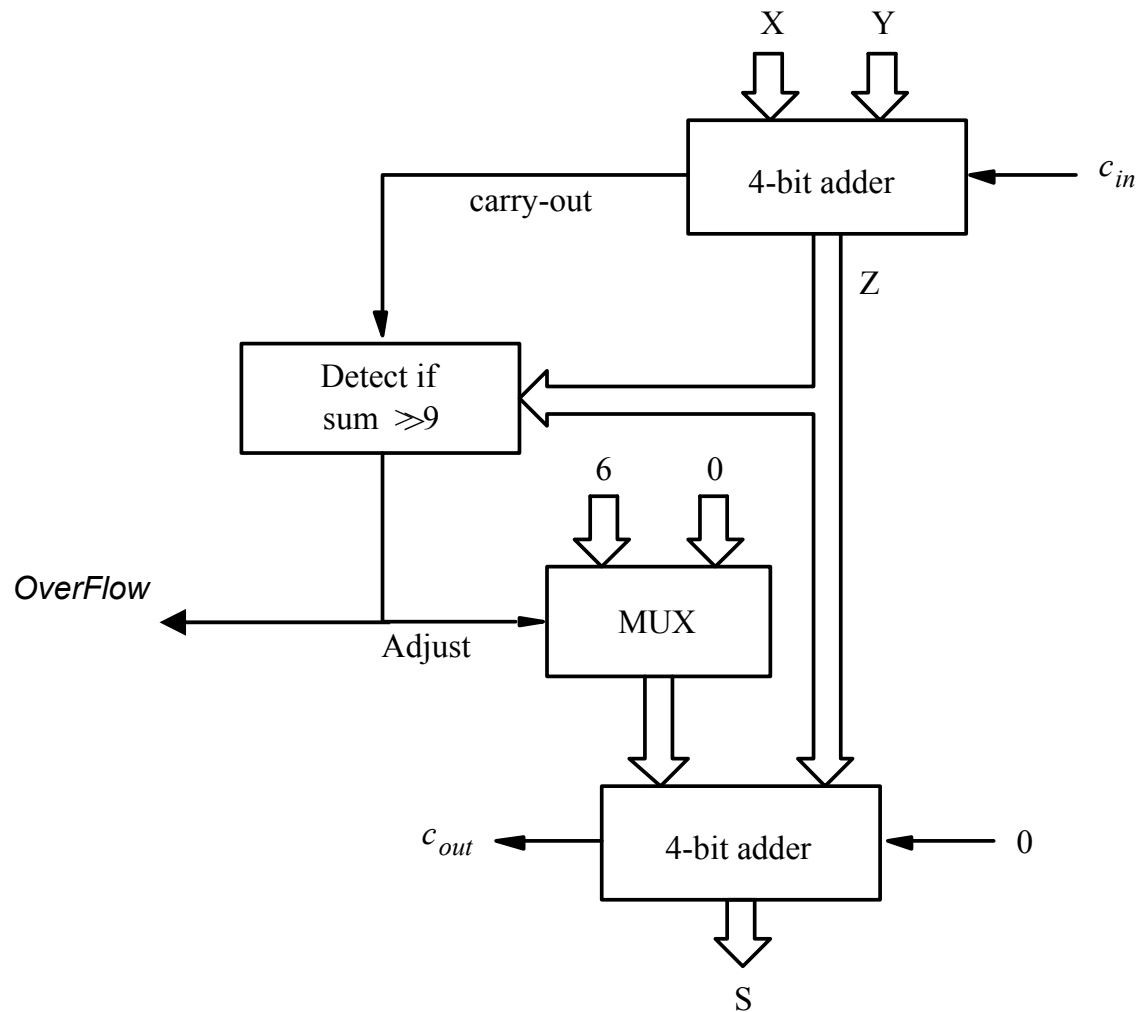


Figure 5.37 Block diagram for a one-digit BCD adder

Somador de um Dígito BCD

$$\begin{array}{r}
 z_3 \ z_2 \ z_1 \ z_0 \\
 + 0 \ 1 \ 1 \ 0 \\
 \hline
 \end{array}$$

$C_{out} = 1 ?$

$$C_{out} = d_{out} + z_2 z_3 + z_1 z_3$$

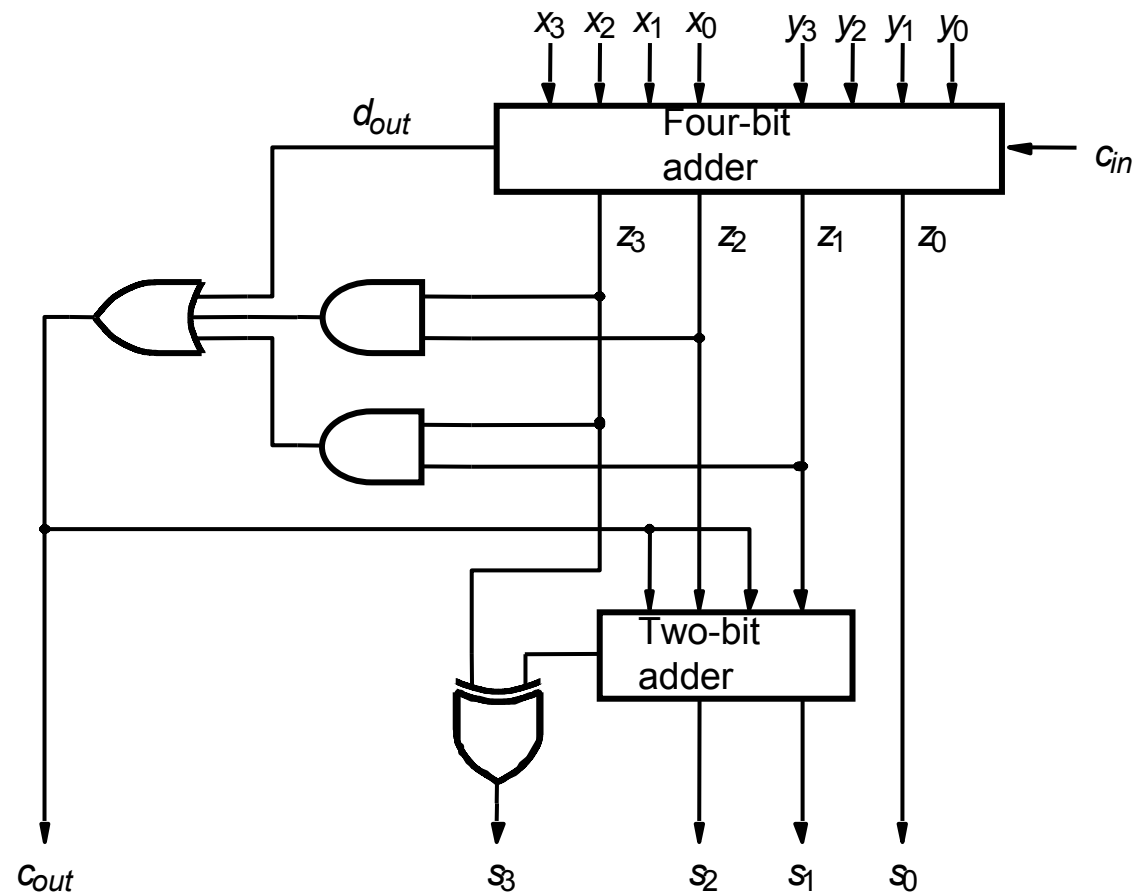


Figure 5.40 Circuit for a one-digit BCD adder

Somador BCD

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_unsigned.all ;

ENTITY BCD IS
    PORT ( X, Y: IN STD_LOGIC_VECTOR(3 DOWNTO 0) ;
          S: OUT STD_LOGIC_VECTOR(4 DOWNTO 0) ) ;
END BCD ;

ARCHITECTURE Behavior OF BCD IS
    SIGNAL Z : STD_LOGIC_VECTOR(4 DOWNTO 0) ;
    SIGNAL Adjust : STD_LOGIC ;
BEGIN
    Z <= ('0' & X) + Y ;
    Adjust <= '1' WHEN Z > 9 ELSE '0' ;
    S <= Z WHEN (Adjust = '0') ELSE Z + 6 ;
END Behavior ;
```

Figure 5.38 VHDL code for a one-digit BCD adder