IC-UNICAMP

MC 602

IC/Unicamp

2011s2

Prof Mario Côrtes

VHDL

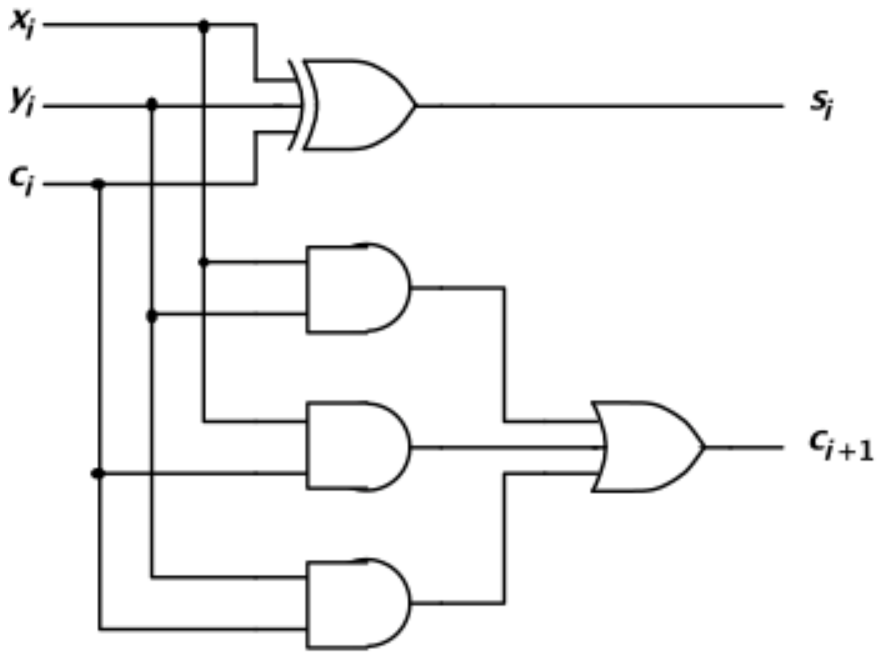
Circuitos Aritméticos

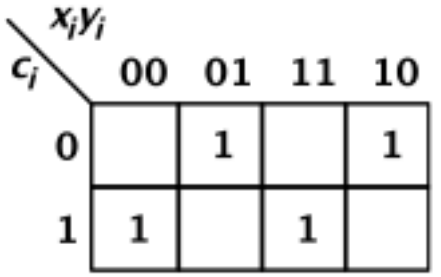
Tópicos

- Somador/subtrator
- Somador com overflow
- Diferentes implementações de somadores com VHDL

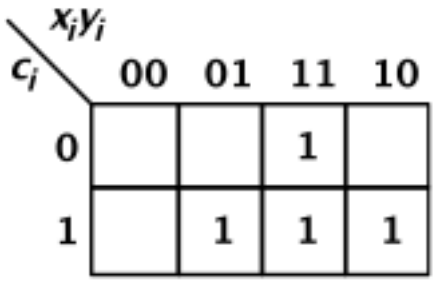
Full-adder

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





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



$c_{i+1} = x_iy_i + x_ic_i + y_ic_i$

(b) Karnaugh maps

IC-UNICAMP

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

IC-UNICAMP

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 ;
```

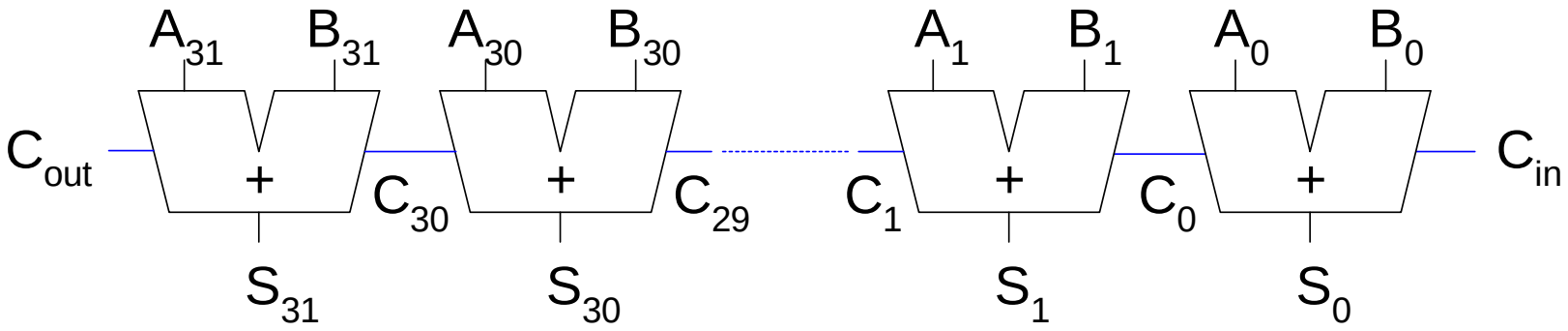
Figure 5.25 Declaration of a package

Somador Ripple Carry

- Atraso para um somador de n bits:

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

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



IC-UNICAMP

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

IC-UNICAMP

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

IC-UNICAMP

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 DOWNTO 0) ;
           S     : OUT  STD_LOGIC_VECTOR(15 DOWNTO 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

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

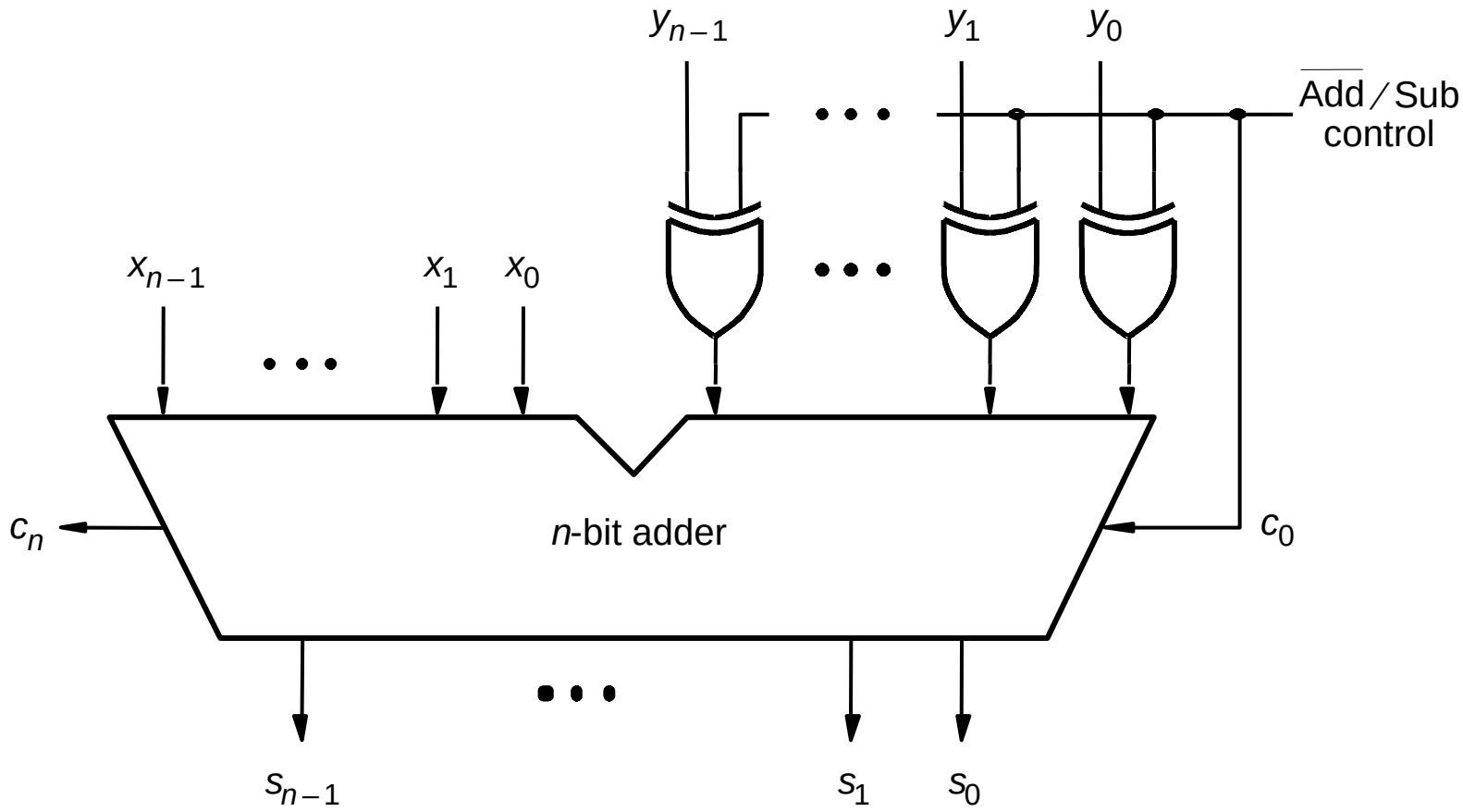


Figure 5.13 Adder/subtractor unit

IC-UNICAMP

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 ;
```

IC-UNICAMP

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 DOWNTO 0) ;
           S     : OUT  STD_LOGIC_VECTOR(15 DOWNTO 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 DOWNTO 0) ;
           S             : OUT  STD_LOGIC_VECTOR(15 DOWNTO 0) ;
           Cout,Overflow : OUT STD_LOGIC ) ;
END adder16 ;

ARCHITECTURE Behavior OF adder16 IS
    SIGNAL Sum : STD_LOGIC_VECTOR(16 DOWNTO 0) ;
BEGIN
    Sum <= ('0' & X) + Y + Cin ;
    S <= Sum(15 DOWNTO 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

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

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	
	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	
	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**

Somador de um Dígito BCD

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

$c_{out} = 1$?

$c_{out} = d_{out} + z_2\, z_3 + z_1\, z_3$

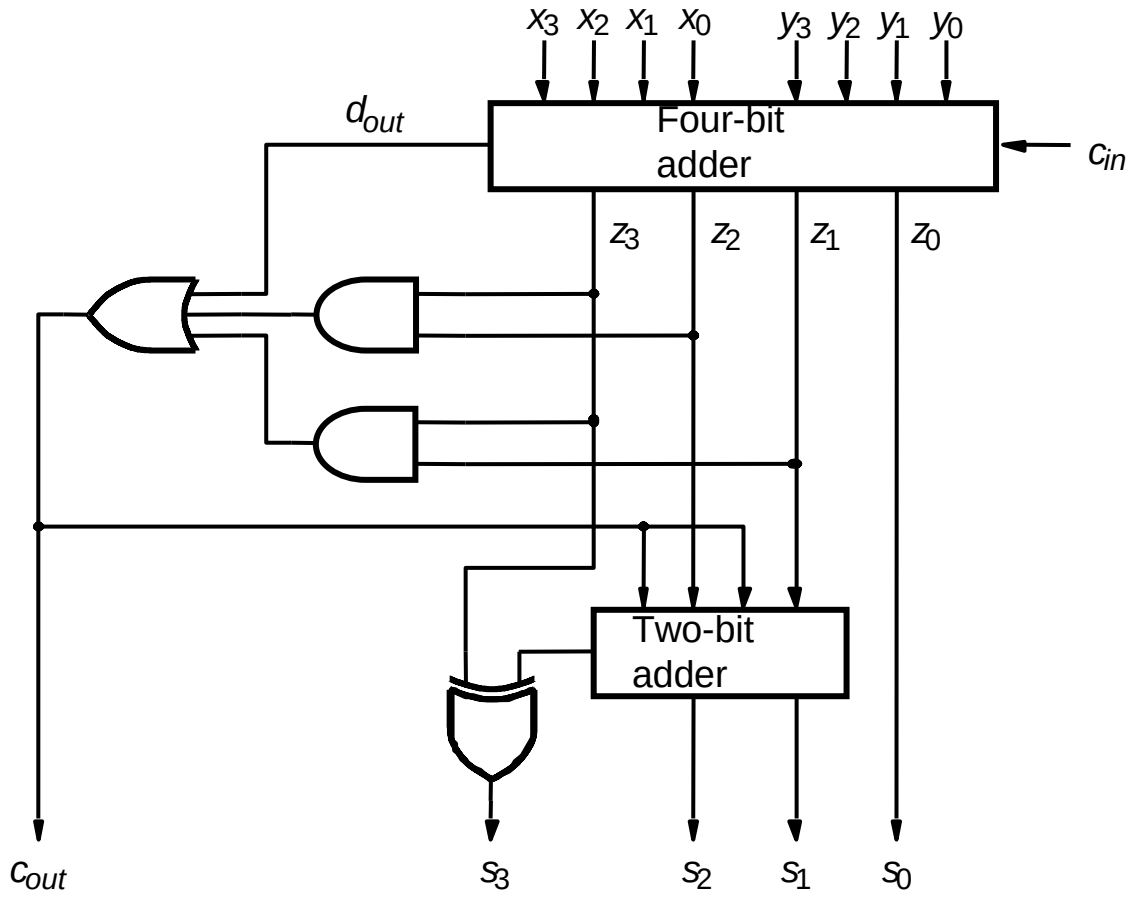


Figure 5.40 Circuit for a one-digit BCD adder

Somador em BCD

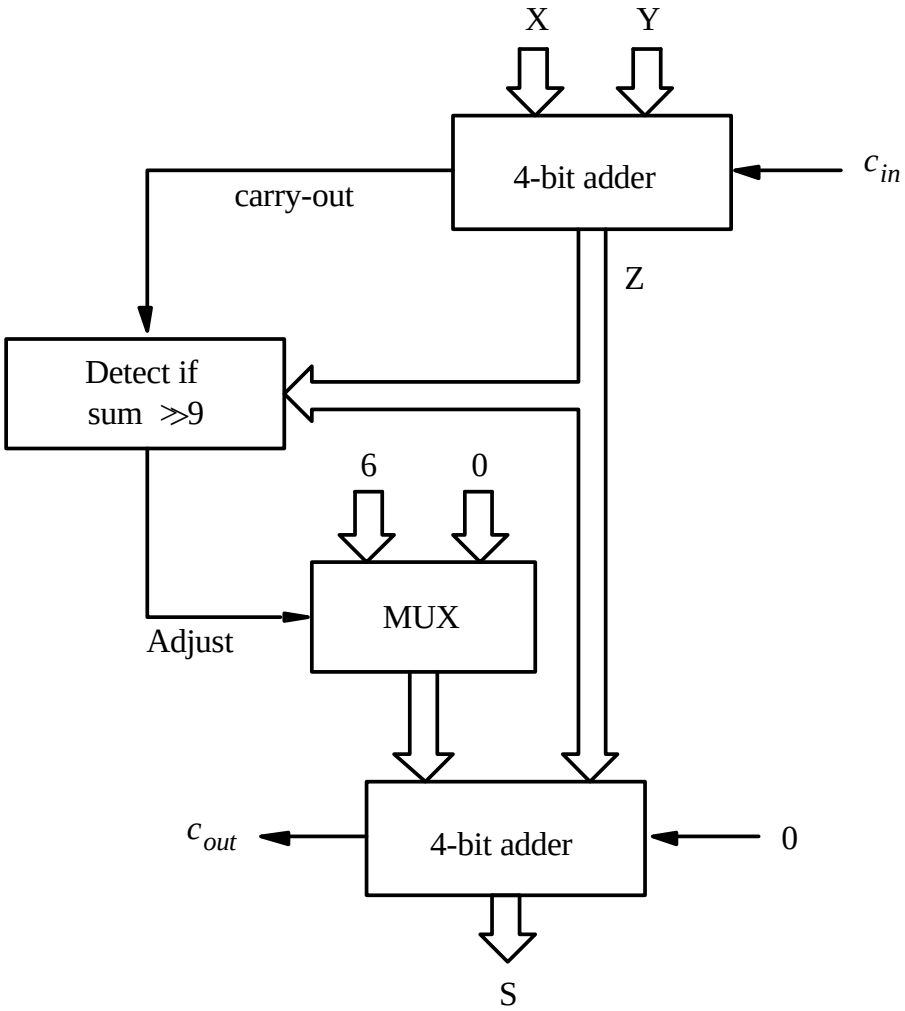


Figure 5.37 Block diagram for a one-digit BCD adder

IC-UNICAMP

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