



**INSTITUTO  
FEDERAL**

Santa Catarina

---

Câmpus  
São José

## **Dispositivos Lógicos Programáveis II**

Consumo de área em bin2bcd e contagem binária

**Arthur Cadore Matuella Barcella e Gabriel Luiz Espindola Pedro**

02 de Abril de 2023

# Sumário

|   |   |
|---|---|
| 1. Descrição de desenvolvimento .....                       | 3 |
| 2. Conceitos teóricos utilizados .....                      | 3 |
| 3. Implementação com somador BCD: .....                     | 3 |
| 4. Implementação com somador binário e conversor BCD: ..... | 3 |
| 5. Conclusão: .....   | 3 |
| 6. Códigos VHDL utilizados - Parte 1: .....                 | 3 |
| 7. Códigos VHDL utilizados - Parte 2: .....                 | 4 |
| 7.1. bin2bcd .....  | 4 |
| 7.2. binAdder .....   | 5 |
| 7.3. bcd2ssd: .....   | 5 |
| 7.4. Project-1 (declaração de componentes): .....           | 6 |

## 1. Descrição de desenvolvimento



Figura 1: Definições de  $x_1[n]$  e  $x_2[n]$

Figura elaborada pelo autor

## 2. Conceitos teóricos utilizados

## 3. Implementação com somador BCD:

Para implementação da primeira parte da atividade, implementamos quatro códigos VHDL para a contagem ocorrer diretamente em BCD.

Para isso, utilizamos os códigos `bcd_counter`, `bcd2ssd`, `bcd2ssd` e `project1`.

## 4. Implementação com somador binário e conversor BCD:

Para realizar a segunda etapa da atividade, implementamos quatro códigos VHDL para a contagem ocorrer em binário (de maneira mais simples), e em seguida realizar sua conversão para BCD. Para isso, utilizamos os códigos `bin2bcd`, `binAdder` e `bcd2ssd`, além de um código que declara os componentes utilizados, chamado `project1`.

O código `bin2bcd` é responsável por converter um número binário de 8 bits para BCD, dividindo o número em centenas, dezenas e unidades.

O código `binAdder` é responsável por somar dois números binários de 8 bits, e o código `bcd2ssd` é responsável por converter um número BCD para um display de 7 segmentos.

Por fim, o código `project1` declara os componentes utilizados e realiza a conexão entre eles.

## 5. Conclusão:

Podemos concluir que a implementação X é mais rápida devido ao tempo de propagação amostrado em cada um dos casos. Também podemos concluir que a implementação Y é mais eficiente em termos de área, pois o consumo de área foi menor em relação à implementação X.

| Implementacao | Área | Tempo de propagação |
|---------------|------|---------------------|
| Parte 1       |      |                     |
| Parte 2       |      |                     |

## 6. Códigos VHDL utilizados - Parte 1:

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4  use ieee.math_real.all;
5
6  entity bcd_adder_4d is
7      port(
8          b0, b1: in std_logic_vector(3 downto 0);
9          a0, a1: in std_logic_vector(3 downto 0);
10         s0, s1, s2: out std_logic_vector(3 downto 0)
11     );
12 end entity bcd_adder_4d;
13
14 architecture bcd_adder_4d of bcd_adder_4d is
15     component bcd_adder is
16         port(
17             cin: in std_logic;
18             a, b: in std_logic_vector(3 downto 0);
19             s: out std_logic_vector(3 downto 0);
20             cout: out std_logic
21         );
22     end component;
23
24     signal c0, c1: std_logic;
25 begin
26     bcd_adder_0: bcd_adder port map(
27         cin => '0',
28         a => a0,
29         b => b0,
30         s => s0,
31         cout => c0
32     );
33
34     bcd_adder_1: bcd_adder port map(
35         cin => c0,
36         a => a1,
37         b => b1,
38         s => s1,
39         cout => c1
40     );
41
42     s2 <= "000" & c1;
43
44 end architecture bcd_adder_4d;

```

## 7. Códigos VHDL utilizados - Parte 2:

### 7.1. bin2bcd

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity bin2bcd is
6      port (

```

```

7      A      : in  std_logic_vector (7 downto 0);
8      sd, su, sc : out std_logic_vector (3 downto 0)
9    );
10 end entity;
11
12 architecture ifsc_v1 of bin2bcd is
13     signal A_uns      : unsigned (7 downto 0);
14     signal sd_uns, su_uns, sc_uns : unsigned (7 downto 0);
15
16 begin
17     A_uns <= unsigned(A);
18     sc_uns <= A_uns/100;
19     sd_uns <= A_uns/10;
20     su_uns <= A_uns rem 10;
21     sc     <= std_logic_vector(resize(sc_uns, 4));
22     sd     <= std_logic_vector(resize(sd_uns, 4));
23     su     <= std_logic_vector(resize(su_uns, 4));
24 end architecture;

```

## 7.2. binAdder

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity binAdder is
6      generic(
7          Nlength: integer := 7
8      );
9      port(
10
11         in_1: in std_logic_vector(Nlength - 1 downto 0);
12         in_2: in std_logic_vector(Nlength - 1 downto 0);
13         out_1: out std_logic_vector(Nlength downto 0)
14
15     );
16 end binAdder;
17
18 architecture v1 of binAdder is
19
20     signal bin1, bin2: unsigned(Nlength downto 0);
21     signal out_bin: unsigned(Nlength downto 0);
22
23 begin
24
25     bin1 <= resize(unsigned(in_1), Nlength + 1);
26     bin2 <= resize(unsigned(in_2), Nlength + 1);
27
28     out_bin <= bin1 + bin2;
29     out_1 <= std_logic_vector(out_bin);
30
31 end v1;

```

## 7.3. bcd2ssd:

```

1  library ieee;

```

```

2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity bcd2ssd is
6      port (
7          bcd      : in std_logic_vector(3 downto 0);
8          ssd_out   : out std_logic_vector(6 downto 0);
9          ac_ccn    : in std_logic
10     );
11 end entity bcd2ssd;
12
13 architecture bcd2ssd_v1 of bcd2ssd is
14
15     signal ssd : std_logic_vector(6 downto 0);
16     signal bcd_int : integer range 0 to 9;
17
18 begin
19
20     ssd_out <= ssd when ac_ccn = '1' else
21         not ssd;
22     bcd_int <= to_integer(unsigned(bcd));
23
24     with bcd_int select ssd <=
25         "0111111" when 0,
26         "0000110" when 1,
27         "1011011" when 2,
28         "1001111" when 3,
29         "1100110" when 4,
30         "1101101" when 5,
31         "1111101" when 6,
32         "0000111" when 7,
33         "1111111" when 8,
34         "1101111" when 9,
35         -- Character "E" when others:
36         "1111001" when others;
37
38 end architecture bcd2ssd_v1;

```

## 7.4. Project-1 (declaração de componentes):

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity project1 is
6      port(
7          clk      : in    std_logic;
8          reset    : in    std_logic;
9
10         input1    : in std_logic_vector(6 downto 0);
11         input2    : in std_logic_vector(6 downto 0);
12
13         ssd_unit   : out std_logic_vector(6 downto 0);
14         ssd_decimal : out std_logic_vector(6 downto 0);
15         ssd_centena : out std_logic_vector(6 downto 0)

```

```

16 );
17 end entity;
18
19 architecture ifsc of project1 is
20
21     component div_clk is
22         generic (
23             div : natural := 50
24         );
25         port (
26             clk_in  : in    std_logic;
27             rst     : in    std_logic;
28             clk_out : out   std_logic
29         );
30     end component div_clk;
31
32     component bin2bcd is
33         port (
34             A      : in  std_logic_vector (7 downto 0);
35             sd, su, sc : out std_logic_vector (3 downto 0)
36         );
37     end component bin2bcd ;
38
39     component bcd2ssd is
40         port (
41             bcd      : in    std_logic_vector(3 downto 0);
42             ssd_out  : out   std_logic_vector(6 downto 0);
43             ac_ccn   : in    std_logic
44         );
45     end component bcd2ssd;
46
47     component binAdder is
48         generic (
49             Nlength : natural := 7
50         );
51         port(
52             in_1: in std_logic_vector(Nlength - 1 downto 0);
53             in_2: in std_logic_vector(Nlength - 1 downto 0);
54             out_1: out std_logic_vector(Nlength downto 0)
55         );
56     end component binAdder;
57
58     signal adder_out : std_logic_vector(7 downto 0);
59     signal bcd_out0, bcd_out1, bcd_out2 : std_logic_vector(3 downto 0);
60     signal ac_ccn0, ac_ccn1, ac_ccn2 : std_logic;
61
62 begin
63
64     adder : component binAdder
65         generic map(
66             Nlength => 7
67         )
68         port map (
69             in_1  => input1,
70             in_2  => input2,
71             out_1  => adder_out
72         );

```

```

73
74 bin2bcd_1 : component bin2bcd
75     port map (
76         A => adder_out,
77         su => bcd_out0,
78         sd => bcd_out1,
79         sc => bcd_out2
80     );
81
82 bcd2ssd_1 : component bcd2ssd
83     port map (
84         bcd      => bcd_out0,
85         ssd_out => ssd_unit,
86         ac_ccn  => ac_ccn0
87     );
88
89 bcd2ssd_2 : component bcd2ssd
90     port map (
91         bcd      => bcd_out1,
92         ssd_out => ssd_decimal,
93         ac_ccn  => ac_ccn1
94     );
95
96 bcd2ssd_3 : component bcd2ssd
97     port map (
98         bcd      => bcd_out2,
99         ssd_out => ssd_centena,
100        ac_ccn  => ac_ccn2
101    );
102
103
104 end architecture;

```