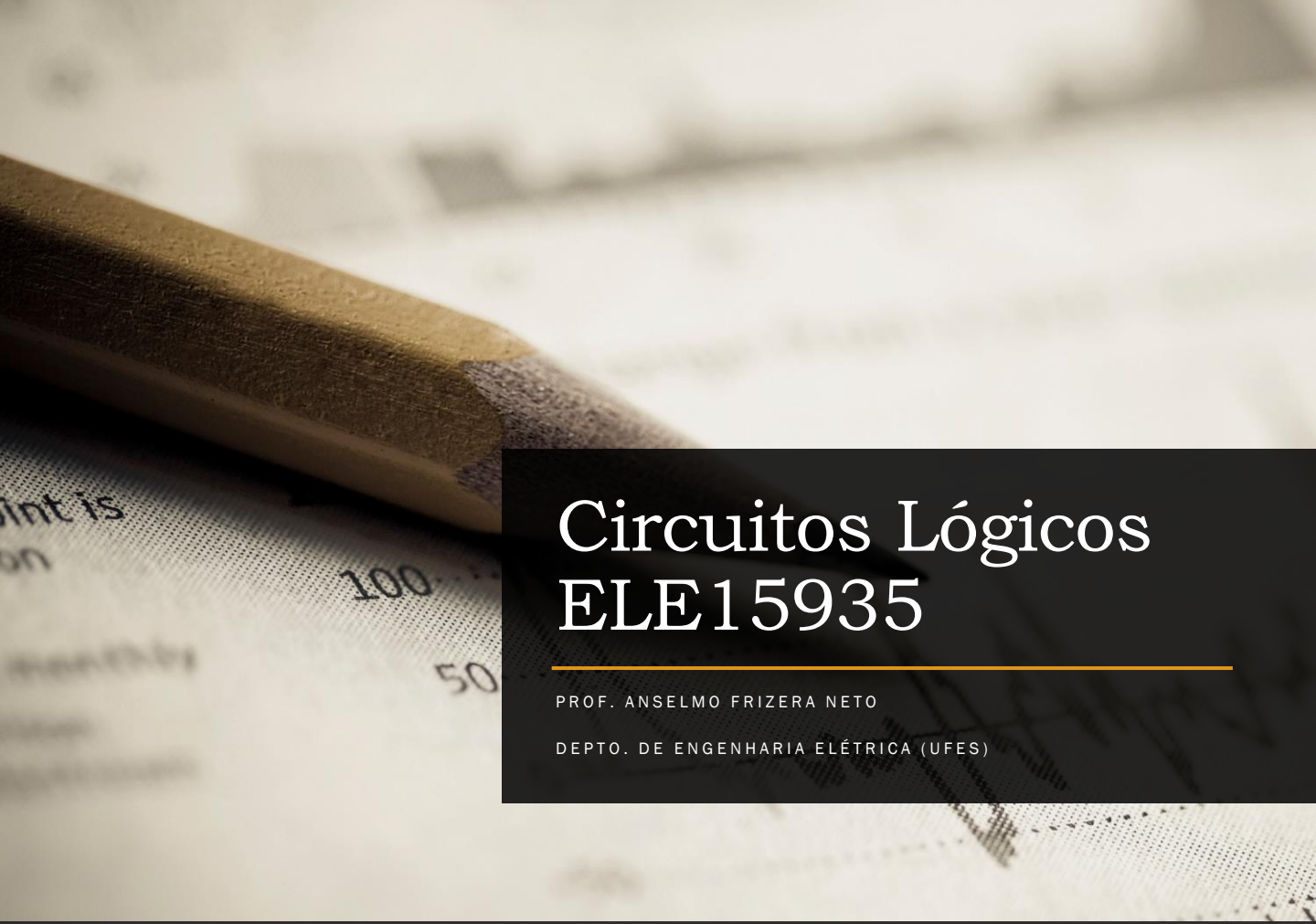


Circuitos Lógicos ELE15935

PROF. ANSELMO FRIZERA NETO

DEPTO. DE ENGENHARIA ELÉTRICA (UFES)

Lab. 4. Constants and generics

Tópicos da aula de hoje:

- Constants
- Generics

Bibliografia:

- P. Chu, “FPGA Prototyping by VHDL Examples”, Capítulo 3

Constants

Constant values frequently used in expressions and array boundaries

Easier to understand by replacing “hard literals” with symbolic constants

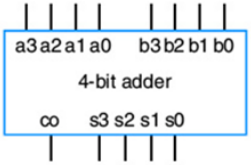
Syntax

- constant const_name : data_type := value_expression ;

E.g.,

- constant DATA_WIDTH : integer := 8;
- constant DATA_RANGE : integer := 2** DATA_WIDTH - 1;

Example: adder with “hard literals”



	a_3	a_2	a_1	a_0
+	b_3	b_2	b_1	b_0
cout	sum_3	sum_2	sum_1	sum_0

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity add_w_carry is
   port(
      a, b : in  std_logic_vector(3 downto 0);
      cout : out std_logic;
      sum  : out std_logic_vector(3 downto 0)
   );
end add_w_carry;
```

Para “caber” o vai_1 (cout)

Para usar o “+”

```
architecture hard_arch of add_w_carry is
   signal a_ext, b_ext, sum_ext : unsigned(4 downto 0);
begin
   a_ext   <= unsigned('0' & a);
   b_ext   <= unsigned('0' & b);
   sum_ext <= a_ext + b_ext;
   sum     <= std_logic_vector(sum_ext(3 downto 0));
   cout    <= sum_ext(4);
end hard_arch;
```

Gera as saídas em std_logic

Example: adder with symbolic constant

```
architecture const_arch of add_w_carry is
    constant N : integer := 4;
    signal a_ext, b_ext, sum_ext : unsigned(N downto 0);
begin
    a_ext   <= unsigned('0' & a);
    b_ext   <= unsigned('0' & b);
    sum_ext <= a_ext + b_ext;
    sum     <= std_logic_vector(sum_ext(N - 1 downto 0));
    cout    <= sum_ext(N);
end const_arch;
```

Generics

Mechanism to pass info into a component

- Declared in entity declaration
- used as a constant in port declaration and architecture body

Assigned a value when the component is instantiated

The value is fixed after component instantiation

```
entity entity_name is
    generic(
        generic_name: data_type := default_value;
        generic_name: data_type := default_value;
        . . .
        generic_name: data_type := default_value
    )
    port(
        port_name: mode data_type;
        . . .
    );
end entity_name;
```

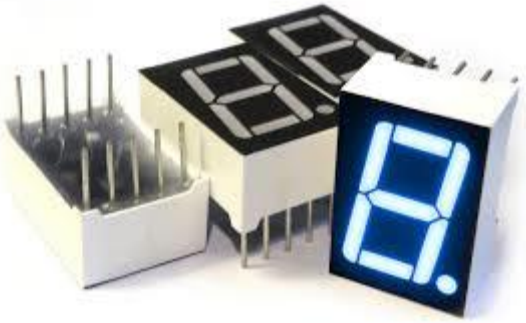
Adder using a generic

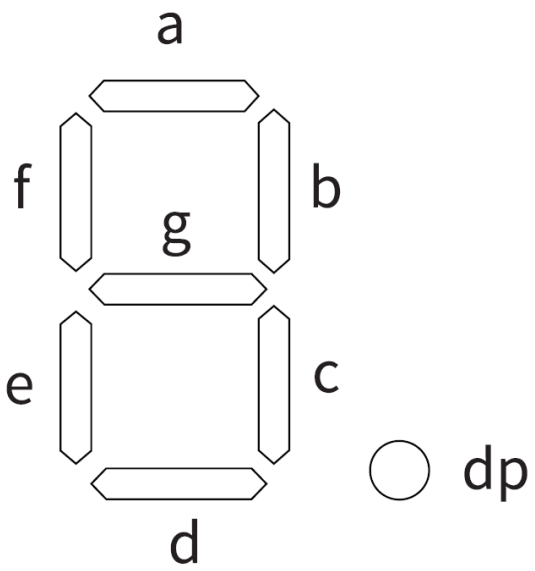
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity gen_add_w_carry is
   generic(N : integer := 4);
   port(
      a, b : in  std_logic_vector(N - 1 downto 0);
      cout : out std_logic;
      sum  : out std_logic_vector(N - 1 downto 0)
   );
end gen_add_w_carry;

architecture arch of gen_add_w_carry is
   signal a_ext, b_ext, sum_ext : unsigned(N downto 0);
begin
   a_ext   <= unsigned('0' & a);
   b_ext   <= unsigned('0' & b);
   sum_ext <= a_ext + b_ext;
   sum     <= std_logic_vector(sum_ext(N - 1 downto 0));
   cout    <= sum_ext(N);
end arch
```

Example: adder component instantiation

```
signal a4, b4, sum4    : unsigned(3 downto 0);
signal a8, b8, sum8    : unsigned(7 downto 0);
signal a16, b16, sum16: unsigned(15 downto 0);
signal c4, c8, c16     : std_logic;

. . .
-- instantiate 8-bit adder
adder_8_unit: work.gen_add_w_carry(arch)
   generic map(N=>8)
   port map(a=>a8, b=>b8, cout=>c8, sum=>sum8));
-- instantiate 16-bit adder
adder_16_unit: work.gen_add_w_carry(arch)
   generic map(N=>16)
   port map(a=>a16, b=>b16, cout=>c16, sum=>sum16));
-- instantiate 4-bit adder
-- (generic mapping omitted, default value 4 used)
adder_4_unit: work.gen_add_w_carry(arch)
   port map(a=>a4, b=>b4, cout=>c4, sum=>sum4));
```

Faltou o entity aqui?



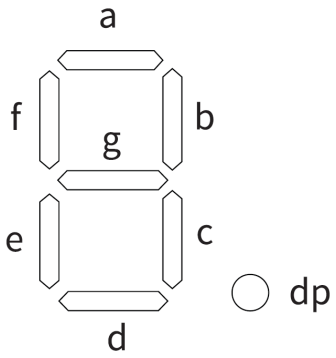


Atividade de hoje

Trabalhando com Display de 7 segmentos

Como funciona o Display de 7 segmentos

N.L. = 0 liga o segmento

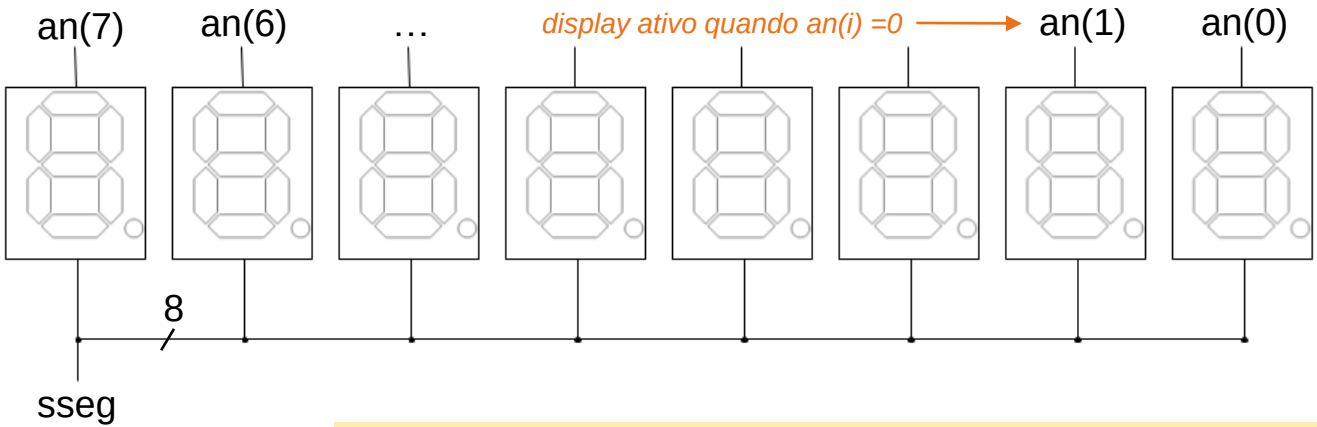


Representação de sinais hexadecimais

```
library ieee;
use ieee.std_logic_1164.all;
entity hex_to_sseg is
   port(
      hex  : in  std_logic_vector(3 downto 0);
      dp   : in  std_logic;
      sseg : out std_logic_vector(7 downto 0)
   );
end hex_to_sseg;

architecture arch of hex_to_sseg is
begin
   with hex select
      sseg(6 downto 0) <=
         "1000000" when "0000",
         "1111001" when "0001",
         "0100100" when "0010",
         "0110000" when "0011",
         "0011001" when "0100",
         "0010010" when "0101",
         "0000010" when "0110",
         "1111000" when "0111",
         "0000000" when "1000",
         "0010000" when "1001",
         "0001000" when "1010", --a
         "0000011" when "1011", --b
         "1000110" when "1100", --c
         "0100001" when "1101", --d
         "0000110" when "1110", --e
         "0001110" when others; --f
   sseg(7) <= dp;
end arch;
```

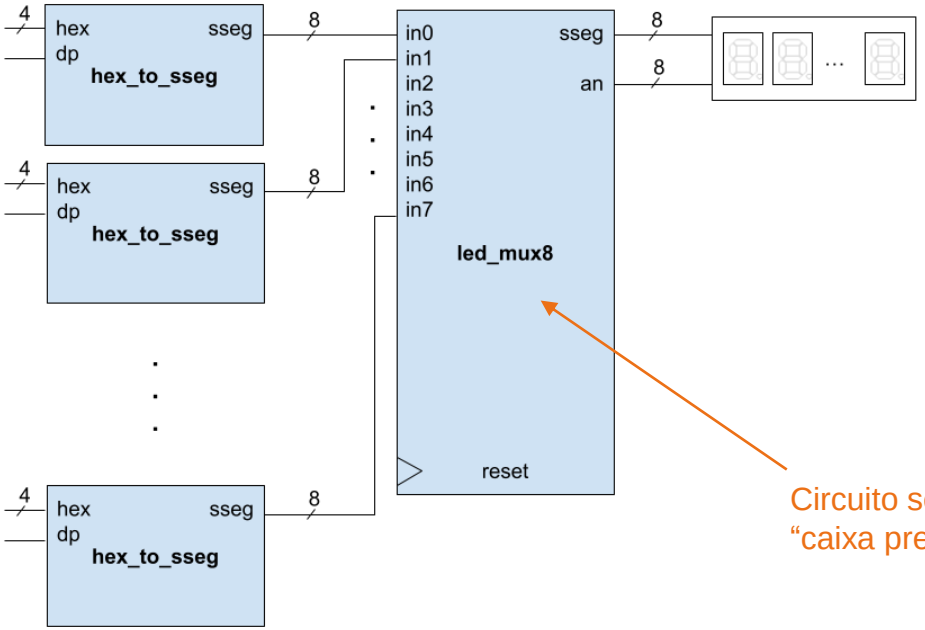
Displays na Placa Nexys A7

8 displays de 7 segmentos que funcionam da seguinte forma:



O mesmo sinal de dados vai para todos os displays!
Para mandar dados diferentes para cada display, precisamos de realizar uma varredura dos displays no tempo:
→ circuito sequencial!

Escrevendo em vários displays…



Atividade de hoje

Utilizar os códigos ***hex_to_sseg*** e ***led_mux8*** para escrever nos 8 displays:

- Displays 0 e 4 → 0 valor em hexadecimal das chaves sw(3 downto 0)
- Displays 1 e 5 → 0 valor em hexadecimal das chaves sw(7 downto 4)
- Displays 2 e 6 → 0 valor em hexadecimal das chaves sw(11 downto 8)
- Displays 3 e 7 → 0 valor em hexadecimal das chaves sw(15 downto 12)

Desafio:

- Display 0 → 0 valor em hexadecimal das chaves sw(3 downto 0)
- Display 1 → 0 valor em hexadecimal das chaves sw(7 downto 4)
- Display 2 → 0 valor em hexadecimal das chaves sw(11 downto 8)
- Display 3 → 0 valor em hexadecimal das chaves sw(15 downto 12)
- Display 4 → 0 valor em hexadecimal das chaves sw(3 downto 0) + 1
- Display 5 → 0 valor em hexadecimal das chaves sw(7 downto 4) + 1
- Display 6 → 0 valor em hexadecimal das chaves sw(11 downto 8) + 1
- Display 7 → 0 valor em hexadecimal das chaves sw(15 downto 12) + 1