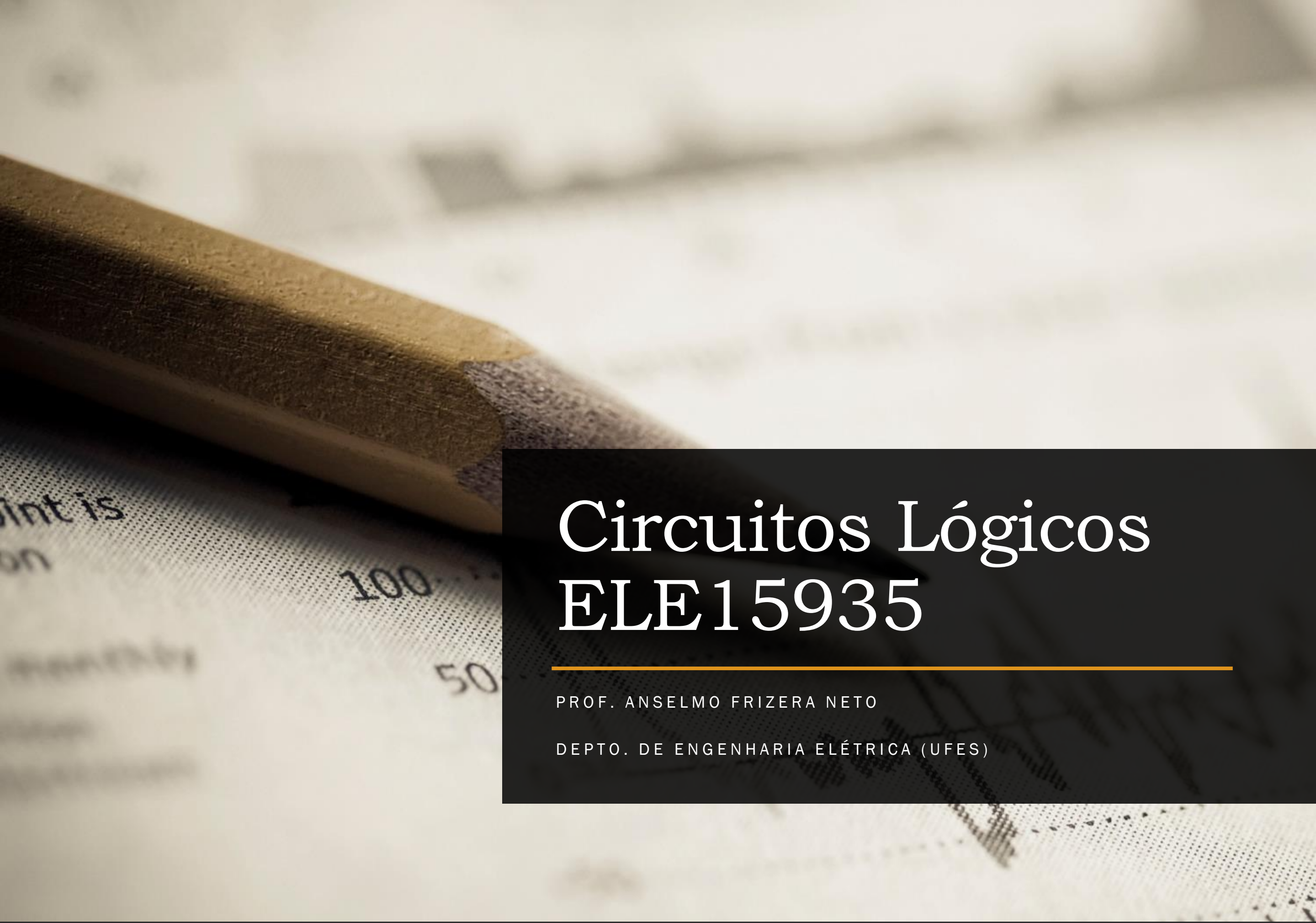


Circuitos Lógicos ELE15935

PROF. ANSELMO FRIZERA NETO

DEPTO. DE ENGENHARIA ELÉTRICA (UFES)

Lab. 5. Replicated structure

Tópicos da aula de hoje:

- For-generate statement
- For-loop statement

Bibliografia:

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

Overview

Many digital circuits exhibit patterned structure

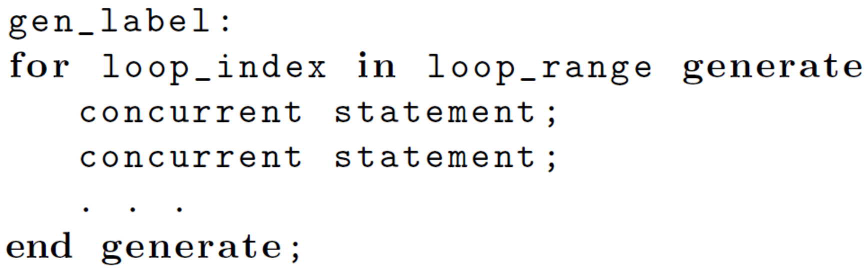
- Repetitive composition of basic blocks
- 1-D cascading chain
- 2-D mesh
- e.g., ripple adder

Loop construct

- Concurrent statement: for-generate
- Sequential statement: for-loop

For-generate statement

Syntax:



```
gen_label:
for loop_index in loop_range generate
    concurrent statement;
    concurrent statement;
    . . .
end generate;
```

loop_range:

- must be static (known before synthesis)
- Index takes values from loop_range

Loop “unrolled” in synthesis

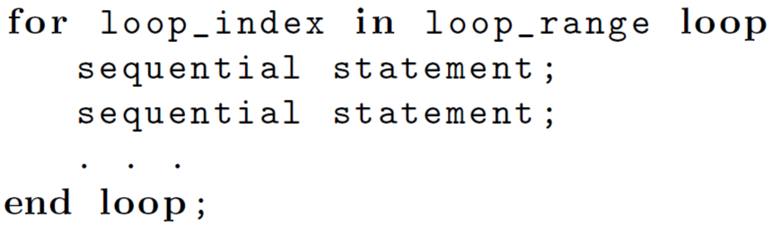
For-loop statement

Similar to the for-generate statement but is a sequential statement:

- It can only be used within a process

May lead to large complex circuit (e.g., nested if-then-else statement with loop)

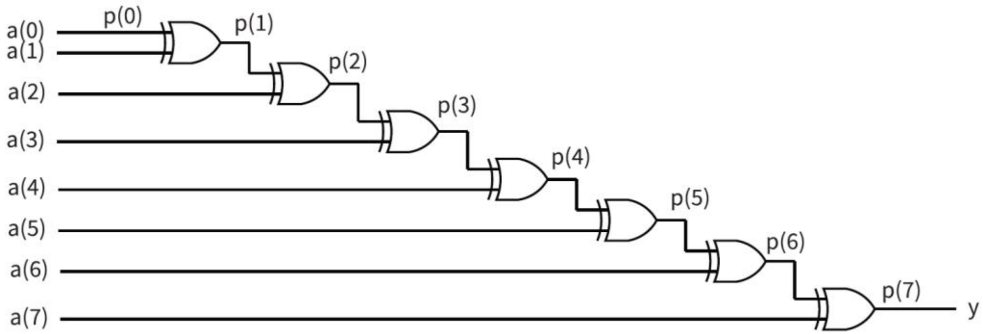
Syntax:

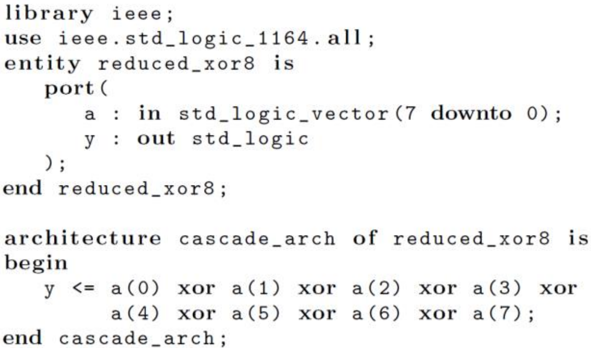


```
for loop_index in loop_range loop
   sequential statement;
   sequential statement;
   . . .
end loop;
```

Example

$$a_7 \oplus a_6 \oplus a_5 \oplus a_4 \oplus a_3 \oplus a_2 \oplus a_1 \oplus a_0$$



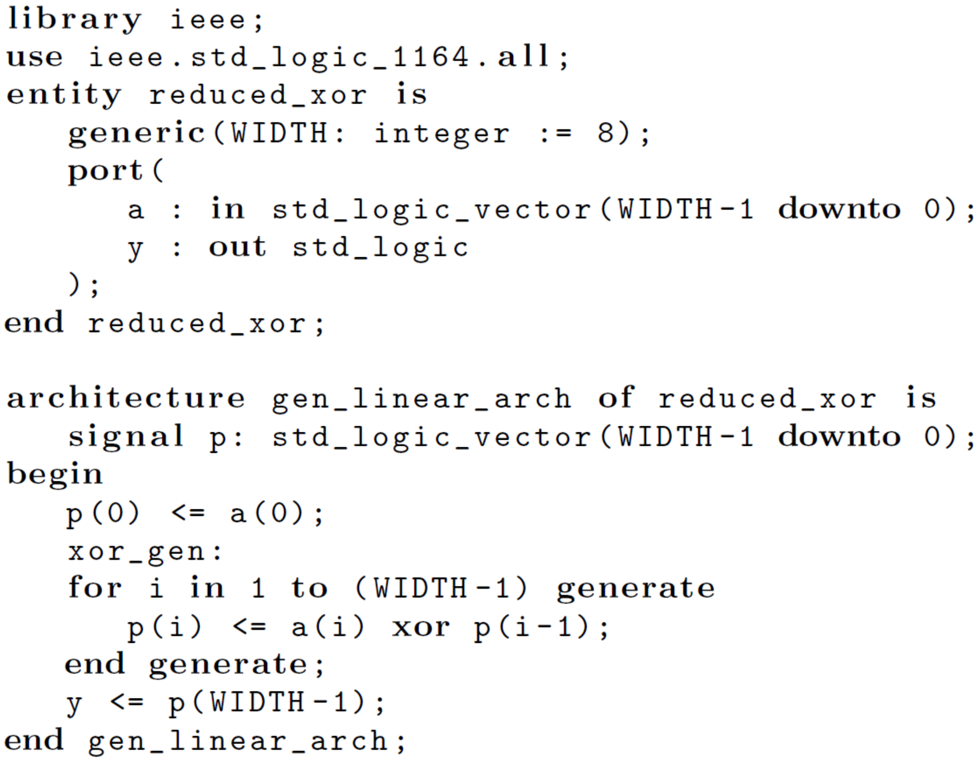


```
library ieee;
use ieee.std_logic_1164.all;
entity reduced_xor8 is
   port(
      a : in std_logic_vector(7 downto 0);
      y : out std_logic
   );
end reduced_xor8;

architecture cascade_arch of reduced_xor8 is
begin
   y <= a(0) xor a(1) xor a(2) xor a(3) xor
        a(4) xor a(5) xor a(6) xor a(7);
end cascade_arch;
```

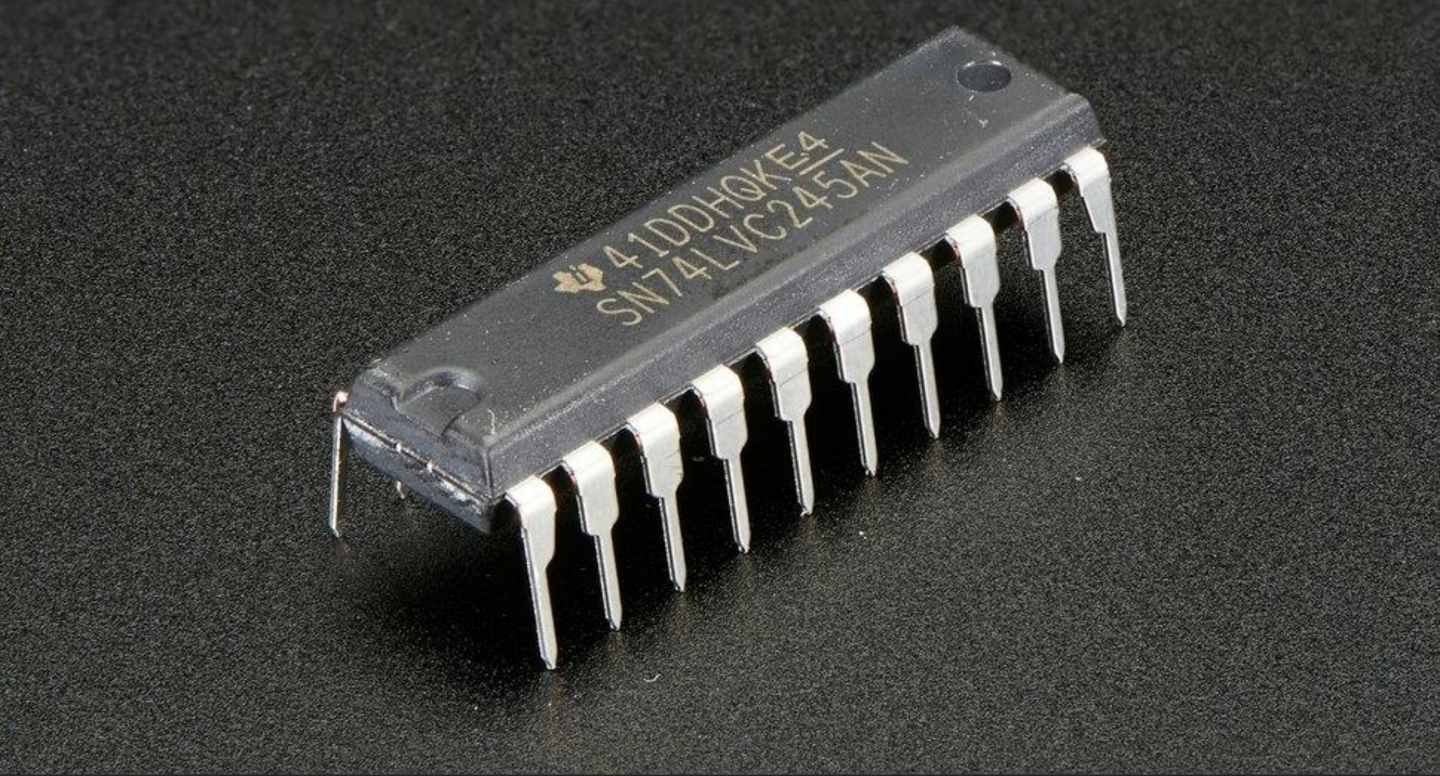
Difficult to use generics and “parametrize”

Parameterized reduced-xor circuit



```
library ieee;
use ieee.std_logic_1164.all;
entity reduced_xor is
   generic(WIDTH: integer := 8);
   port(
      a : in std_logic_vector(WIDTH-1 downto 0);
      y : out std_logic
   );
end reduced_xor;

architecture gen_linear_arch of reduced_xor is
   signal p: std_logic_vector(WIDTH-1 downto 0);
begin
   p(0) <= a(0);
   xor_gen:
   for i in 1 to (WIDTH-1) generate
      p(i) <= a(i) xor p(i-1);
   end generate;
   y <= p(WIDTH-1);
end gen_linear_arch;
```



Atividade de hoje

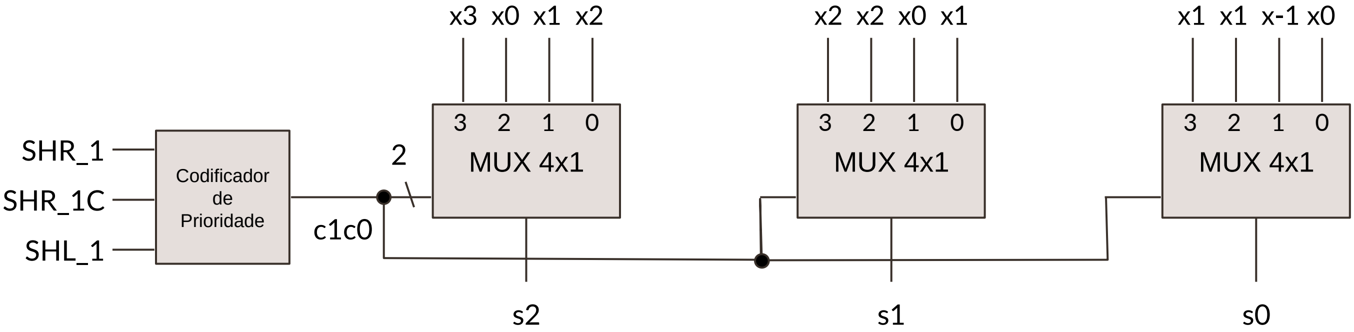
PROJETO DE DESLOCADOR COM MÚLTIPLAS FUNÇÕES

Projeto de deslocador com múltiplas funções

Projete um Deslocador de 3 bits que realize as seguintes funções (em ordem de prioridade decrescente):

- Deslocamento para a direita de 1 bit (entrada de controle SHR_1 = 1).
- Deslocamento circular para a direita de 1 bit (entrada de controle SHR_1C = 1).
- Deslocamento para a esquerda de 1 bit (entrada de controle SHL_1 = 1)
- Não deslocar

Utilize as chaves como entradas de controle e de dados e os LEDs discretos como saídas



Código:
Deslocador usando FOR GENERATE

Codificador de Prioridade:

```
library ieee;
use ieee.std_logic_1164.all;
entity prio_encoder32 is
   port(
      r: in std_logic_vector(2 downto 0);
      code: out std_logic_vector(1 downto 0)
   );
end prio_encoder32;

architecture cond_arch of prio_encoder32 is
begin
   code <=  "11" when (r(2)='1') else
            "10" when (r(1)='1') else
            "01" when (r(0)='1') else
            "00";
end cond_arch;
```

Código:
Deslocador usando FOR GENERATE

MUX:

```
library ieee;
use ieee.std_logic_1164.all;
entity mux4 is
   port(
      dado : in std_logic_vector(3 downto 0);
      s: in std_logic_vector(1 downto 0);
      x: out std_logic
   );
end mux4;
```

```
architecture case_arch of mux4 is
begin
   process(s, dado)
   begin
      case s is
         when "00" =>
            x <= dado(0);
         when "01" =>
             x <= dado(1);
         when "10" =>
             x <= dado(2);
         when others =>
             x <= dado(3);
      end case;
   end process;
end case_arch;
```

Código:
Deslocador usando FOR GENERATE

MUX:

```
library ieee;
use ieee.std_logic_1164.all;
entity top is
   port(
      sw  : in  std_logic_vector(7 downto 0); -- 8
switches -> 0 a 4 xN ; 5 a 7 SH
      led : out std_logic_vector(2 downto 0) -- 3vred
LED
   );
end top;

architecture struc_arch of top is
signal prio : std_logic_vector(1 downto 0);
begin

  prio_cod : entity work.prio_encoder32(cond_arch)
      port map(
         r    => sw(7 downto 5),
         code => prio(1 downto 0)
      );
```

```
  r1 : for i in 0 to 2 generate
     r2 : if (i = 2) generate
           mux0 : entity work.mux4(case_arch)
                 port map(
                     dado(3) => sw(i+2),
                     dado(2) => sw(i-1),
                     dado(1) => sw(i),
                     dado(0) => sw(i+1),
                     s => prio(1 downto 0),
                     x => led(i)
                    );
         end generate r2;
     r3 : if (i < 2) generate
          mux0 : entity work.mux4(case_arch)
              port map(
                  dado(3) => sw(i+2),
                  dado(2) => sw(i+2),
                  dado(1) => sw(i),
                  dado(0) => sw(i+1),
                  s => prio(1 downto 0),
                  x => led(i)
                 );
       end generate r3;
    end generate r1;
end struc_arch;
```