Sistemas Digitais 2

Projeto lógico sequencial
Contadores e registradores de deslocamento

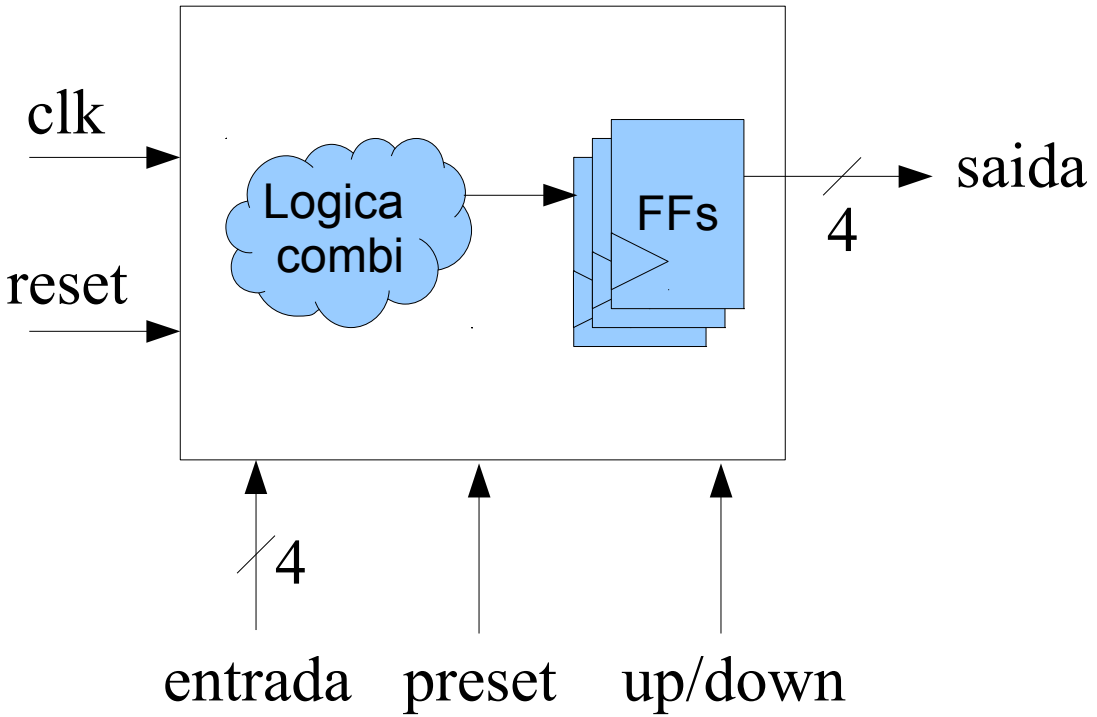
Prof. Daniel M. Muñoz Arboleda

FGA - UnB

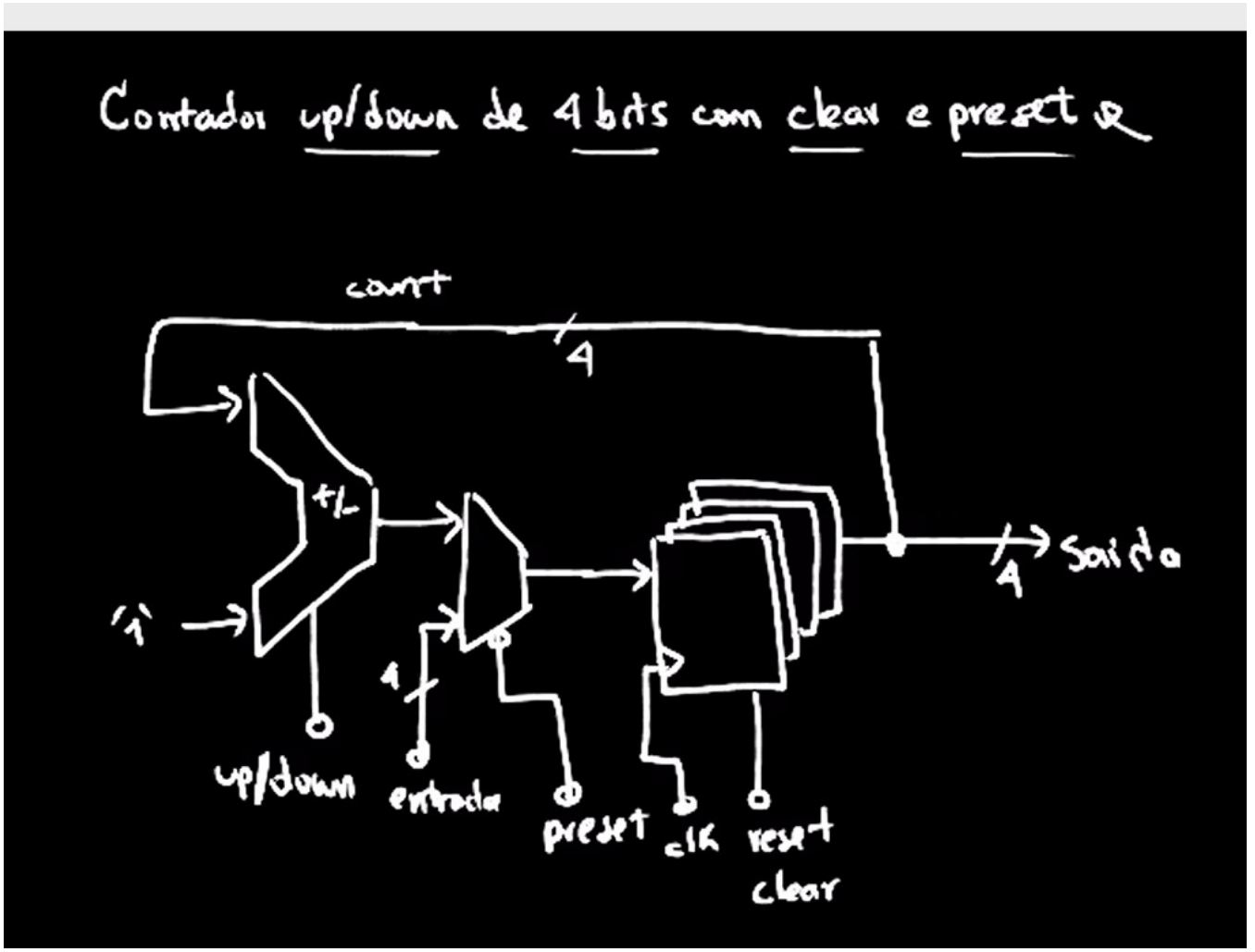
Agenda

- Contadores em VHDL
  - Exemplo contador up/down de 4 bits (0 a F) com clear e preset
  - Exemplo contador de 8 bits com enable e clear
  - Exemplo contador up/down de 4 bits com display de 7 segmentos
  - Exemplo divisor de clock em VHDL
- Registradores de deslocamento em VHDL
  - Exemplo de multiplicação por 2
  - Exemplo de divisão por 2

Contador up/down de 4bits com clear e preset



Contador up/down de 4bits com clear e preset



Descrição em VHDL de um contador up/down de 4bits com clear e preset

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;

entity contador_up_down_4bits is
    Port ( reset : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           entrada : in  STD_LOGIC_VECTOR (3 downto 0);
           preset : in  STD_LOGIC;
           updown : in  STD_LOGIC;
           saida : out  STD_LOGIC_VECTOR (3 downto 0));
end contador_up_down_4bits;
```

```vhdl
architecture Behavioral of contador_up_down_4bits is

signal count : STD_LOGIC_VECTOR (3 downto 0) := "0000";
begin

    saida <= count;

    process(clk,reset)
    begin
        if rising_edge(clk) then
            if reset = '1' then
                count <= (others=>'0');
            elsif preset='1' then
                count <= entrada;
            elsif updown='1' then
                count <= count + '1';
            elsif updown='0' then
                count <= count - '1';
            end if;
        end if;
    end process;

end Behavioral;
```

Descrição em VHDL de um contador up/down de 4bits com clear e preset

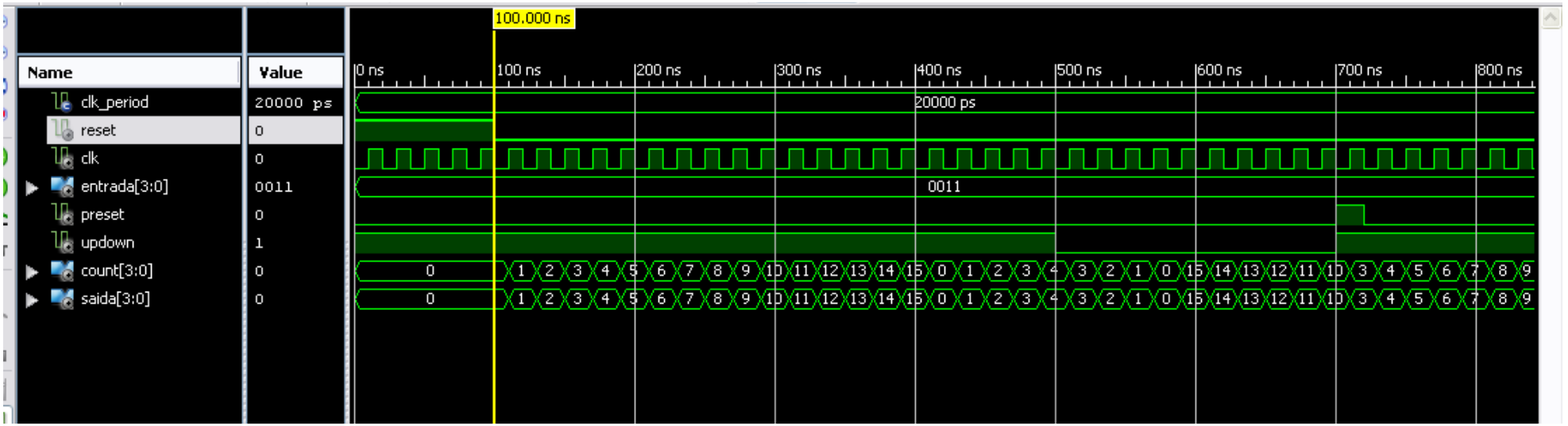
Testbench

```
41  -- Instantiate the Unit Under Test (UUT)
42     uut: contador_up_down_4bits PORT MAP (
43            reset => reset,
44            clk => clk,
45            entrada => entrada,
46            preset => preset,
47            updown => updown,
48            saida => saida
49          );
50
51     -- Clock process definitions
52     clk_process :process
53     begin
54          clk <= '0';
55          wait for clk_period/2;
56          clk <= '1';
57          wait for clk_period/2;
58     end process;
59
```

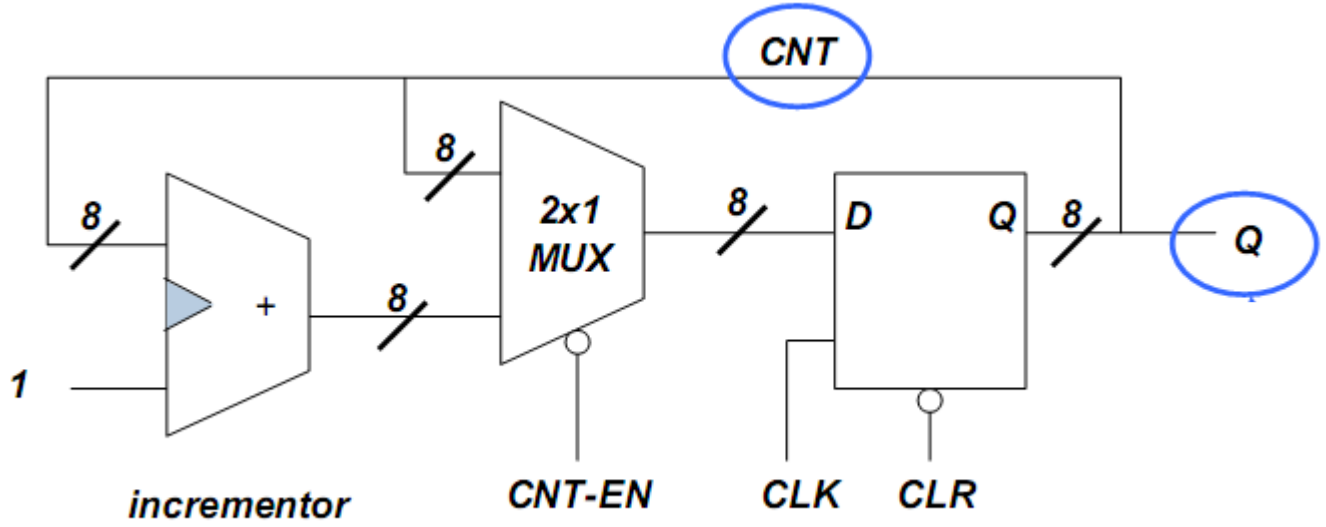
```
60     stim_proc: process
61     begin
62        reset <= '1';
63          entrada <= "0011";
64          updown <= '1'; -- incrementa
65          preset <= '0';
66        wait for 100 ns;
67          reset <= '0';
68
69        wait for clk_period*10;
70
71        updown <= '1'; wait for clk_period*10;
72        updown <= '0'; wait for clk_period*10;
73          preset <= '1'; updown <= '1'; wait for clk_period;
74          preset <= '0'; updown <= '1'; wait for clk_period*10;
75
76        wait;
77     end process;
```

Descrição em VHDL de um contador up/down de 4bits com clear e preset

Testbench



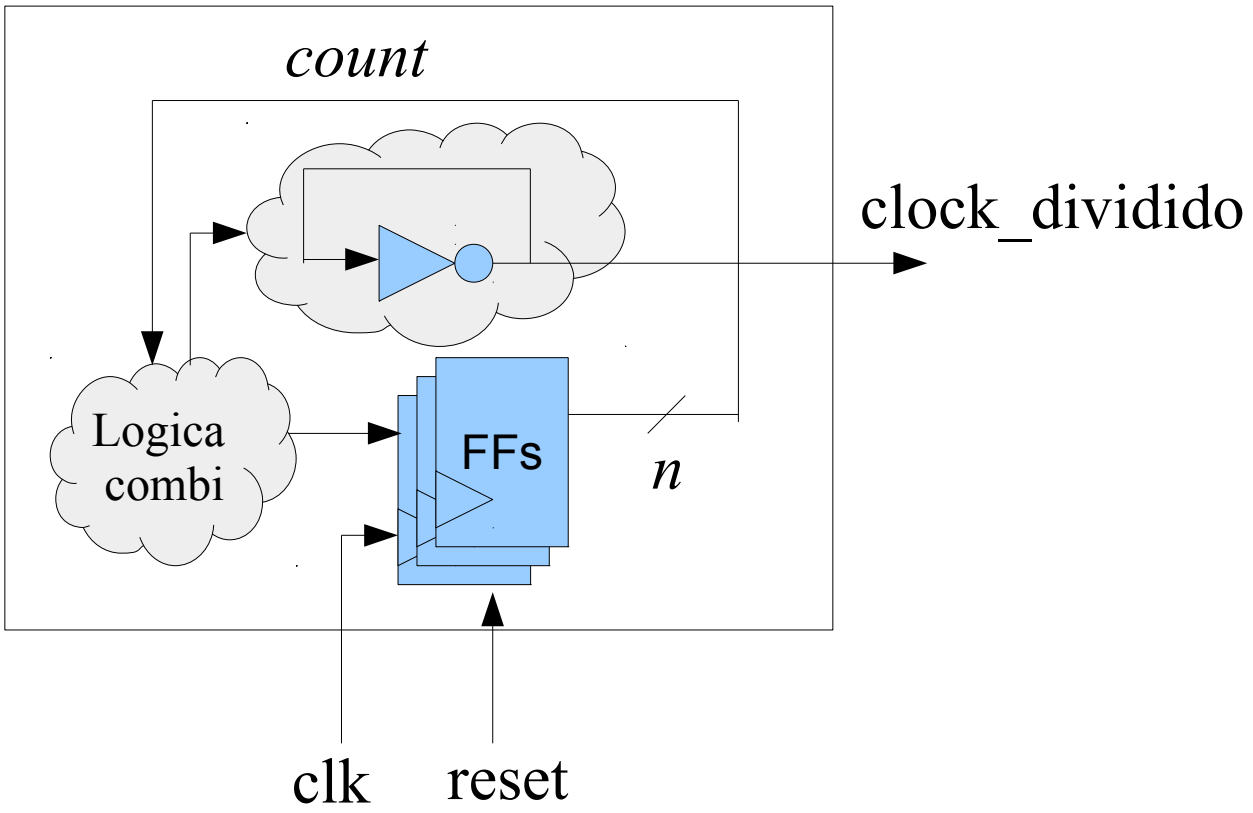
Contador de 8bits com enable e clear



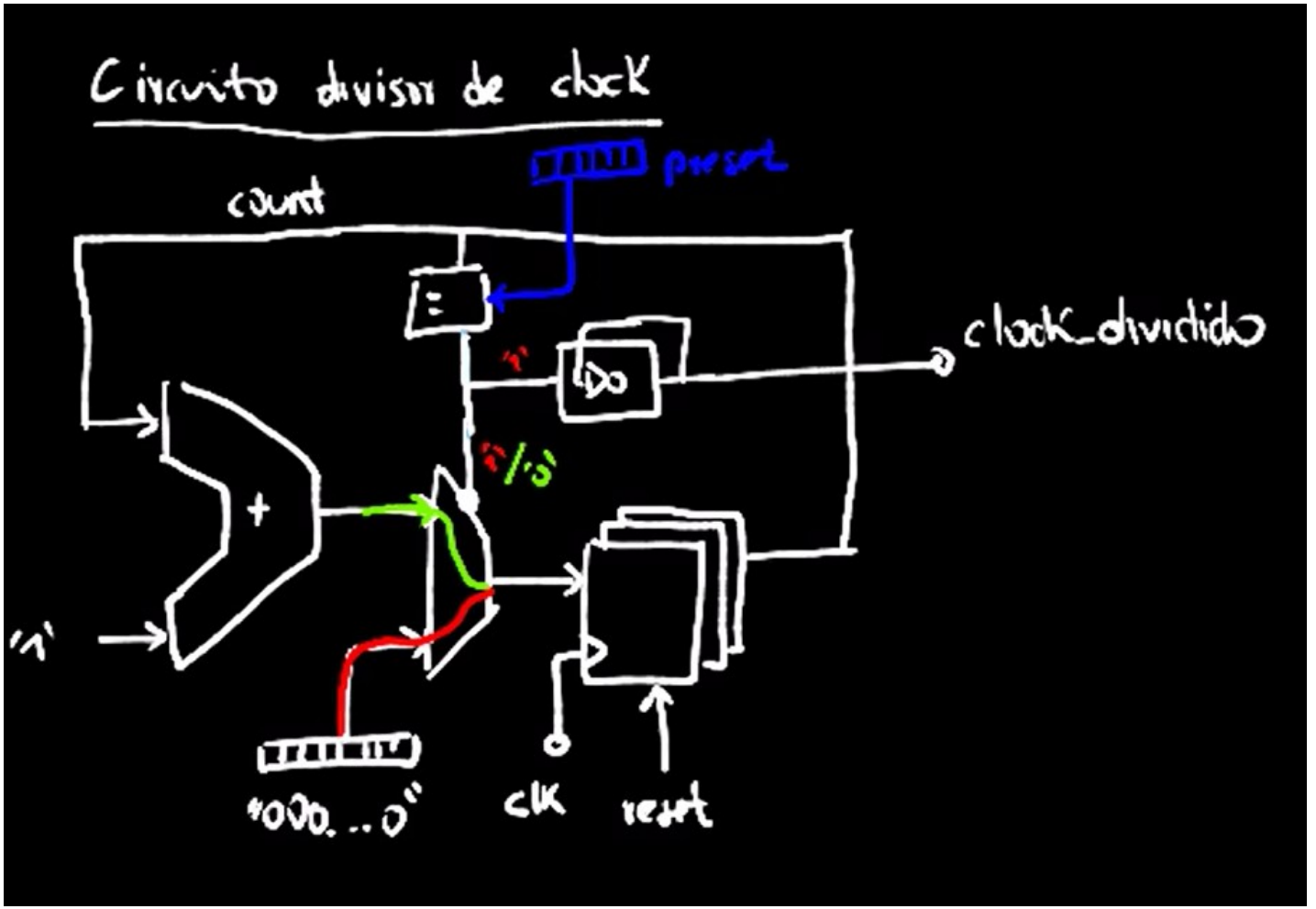
Descrição em VHDL de um contador de 8 bits com enable e clear

```
library ieee; use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity COUNTER is
 port (CLK,CNT_EN,CLR:in std_logic;
       Q              :out std_logic_vector(7 downto 0));
end COUNTER;
architecture BEHAVE of COUNTER is
  signal CNT:std_logic_vector(7 downto 0);
begin
 FIRST: process (CLK, CLR)
 begin
   if (CLR = '0') then
     CNT <= "00000000";
   elsif (CLK'event and CLK = '1') then
     if (CNT_EN = '0') then
         CNT <= CNT + '1';
     end if ;
   end if ;
 end process FIRST;
 Q <= CNT;
end BEHAVE;
```

Descrição em VHDL de um divisor de *clock*



Descrição em VHDL de um divisor de *clock*

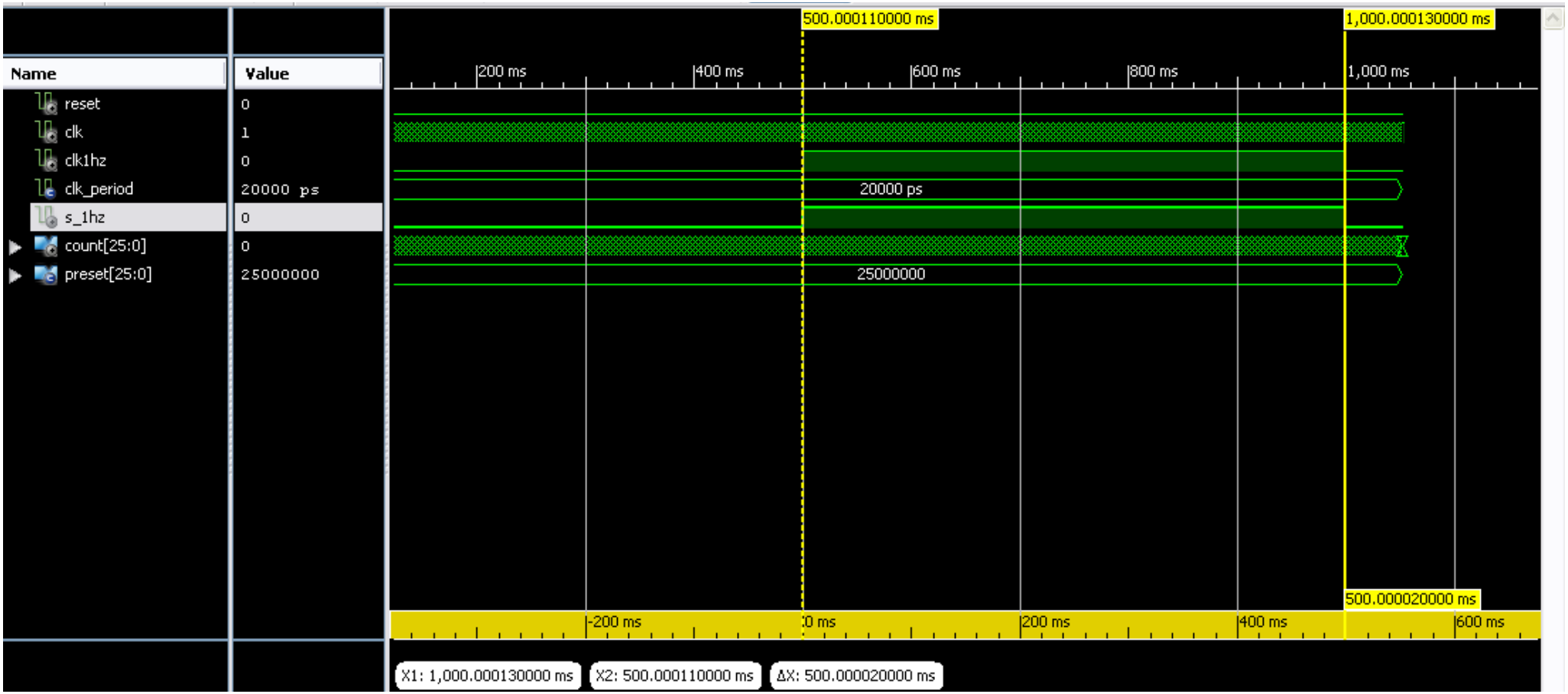


Descrição em VHDL de um divisor de *clock* de 1 seg

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_arith.ALL;
4  use IEEE.STD_LOGIC_unsigned.ALL;
5  entity divisor_clock is
6      Port ( reset : in  STD_LOGIC;
7             clk : in  STD_LOGIC;
8             clk1Hz : out  STD_LOGIC);
9  end divisor_clock;
10
11 architecture Behavioral of divisor_clock is
12 --constante preset = 25E6 para T=1seg
13 constant preset : std_logic_vector(25 downto 0) := "01011111010111100001000000";
14 signal count : std_logic_vector(25 downto 0) := (others => '0');
15 signal s_1Hz : std_logic := '0';
16
17 begin
18 clk1Hz <= s_1Hz;
19 process(clk,reset)
20 begin
21     if rising_edge(clk) then
22         if reset='1' then
23             s_1Hz <= '0';
24             count <= (others => '0');
25         elsif count=preset then
26             s_1Hz <= not s_1Hz;
27             count <= (others => '0');
28         else
29             count <= count + '1';
30         end if;
31     end if;
32 end process;
```

Descrição em VHDL de um divisor de *clock* de 1 seg

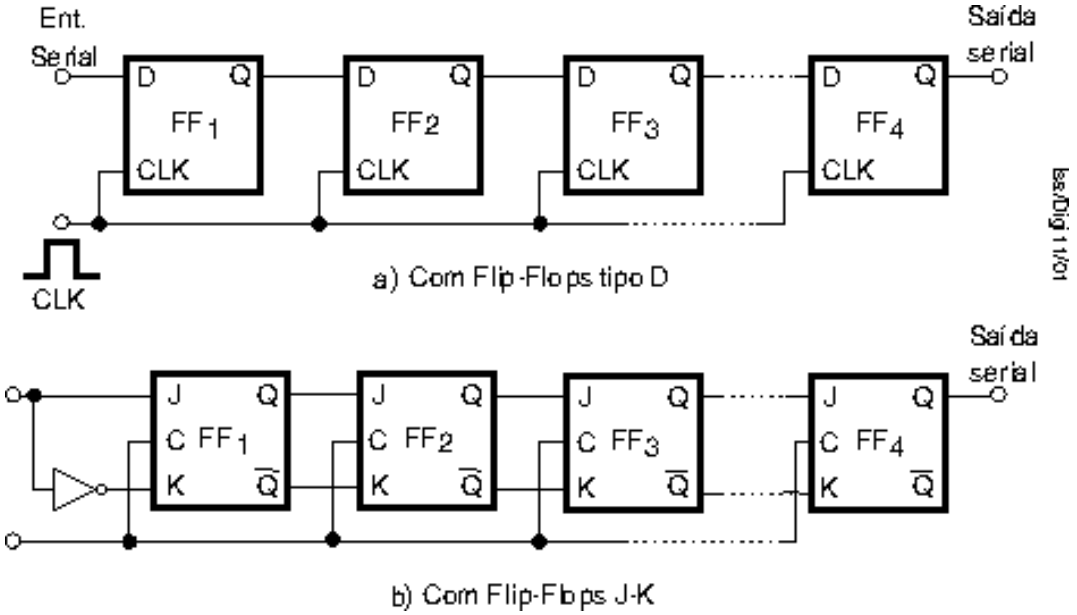
Testbench



Registradores de deslocamento – *Shift registers*

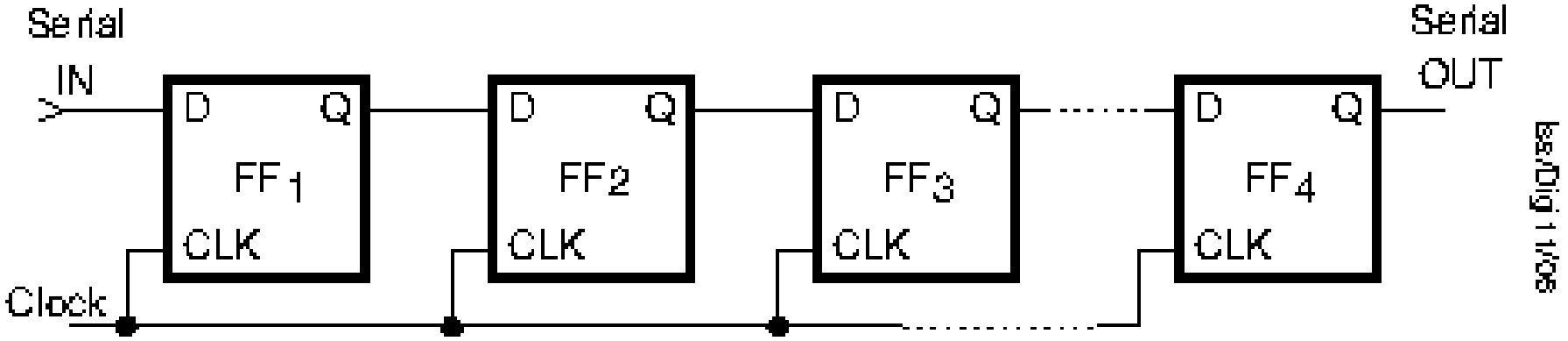
Registradores de deslocamento consistem em um conjunto de flip-flops que podem ser interligados para deslocar uma informação (bit) aplicada na entrada em uma posição à direita ou à esquerda a cada pulso de clock.

Podem ser utilizados para operações de criptografia, multiplicação e divisão em potência de 2, serialização e desserialização, etc.



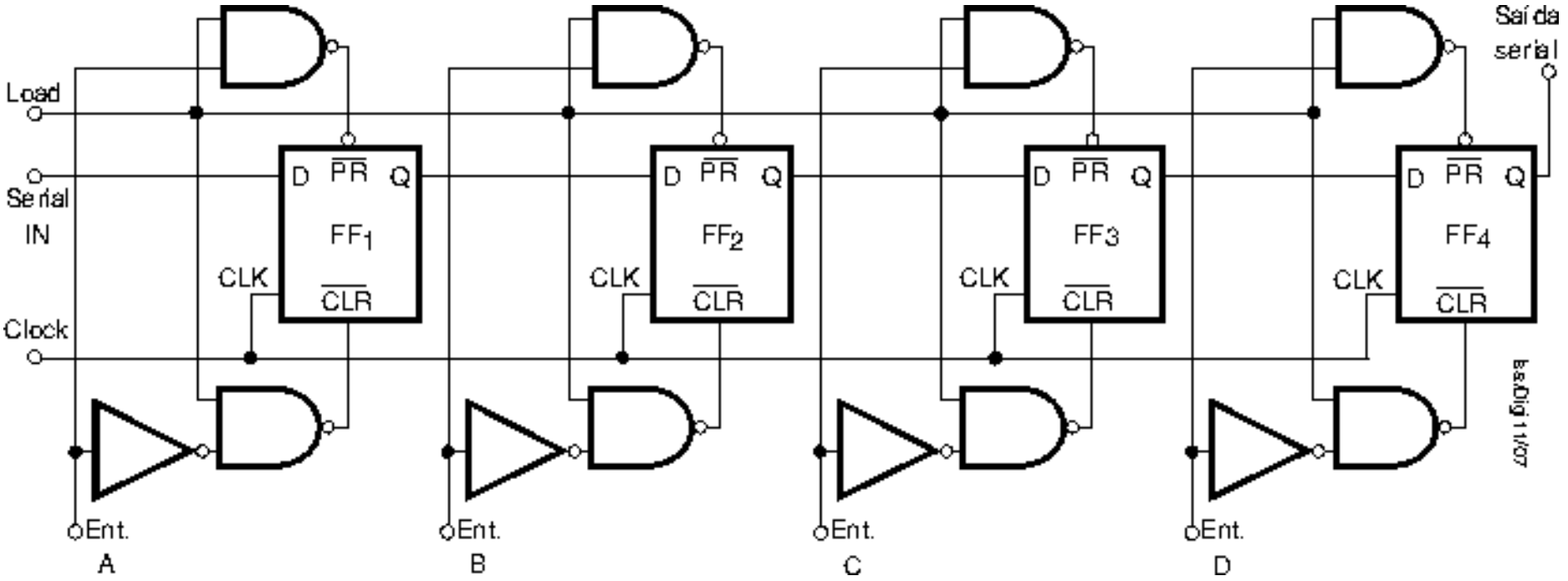
Tipos de registradores de deslocamento – *Shift registers*

a) SISO (Serial in/Serial out)



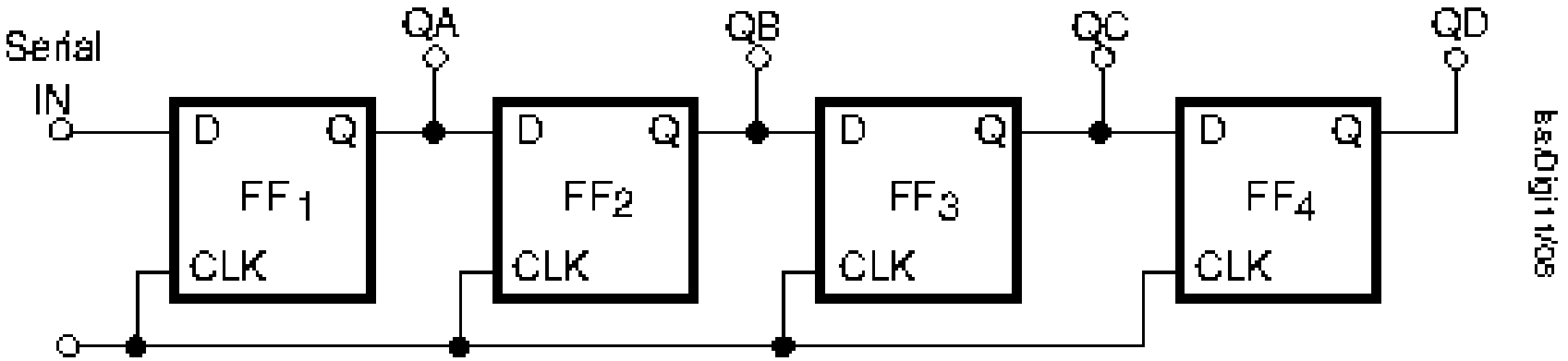
Tipos de registradores de deslocamento – *Shift registers*

b) PISO (Parallel in/Serial out)



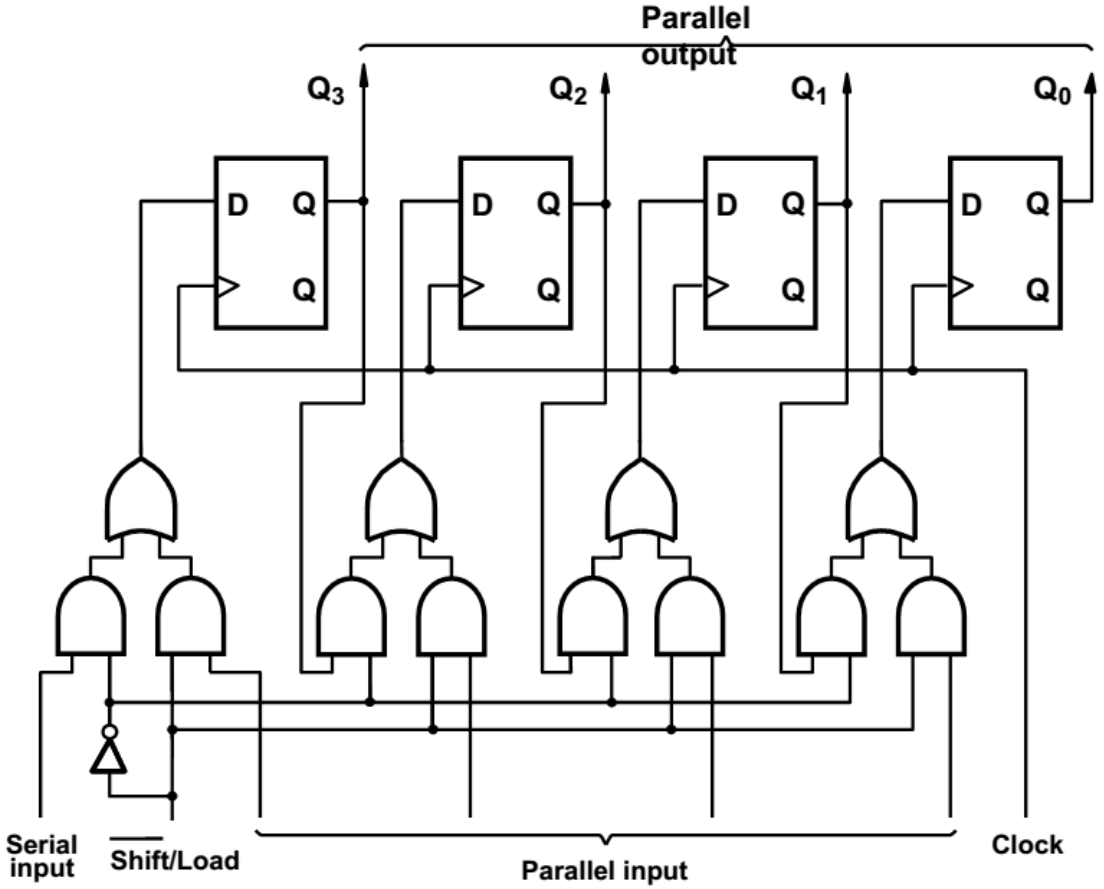
Tipos de registradores de deslocamento – *Shift registers*

c) SIPO (Serial in/Parallel out)



Tipos de registradores de deslocamento – *Shift registers*

d) PIPO (Parallel in/Parallel out)



Descrição em VHDL de um *shift register* SIPO de 4 bits

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity shift_reg_SIPO is
    Port ( reset : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           din : in  STD_LOGIC;
           saida : out  STD_LOGIC_VECTOR (3 downto 0));
end shift_reg_SIPO;

architecture Behavioral of shift_reg_SIPO is
signal A : STD_LOGIC_VECTOR (3 downto 0) := "0000";
begin
    process(clk)
    begin
        if rising_edge(clk) then
            A <= A(2 downto 0) & din;  -- shift left
        end if;
    end process;
    saida <= A;
end Behavioral;
```

Descrição em VHDL de um *shift register* SIPO de *N* bits

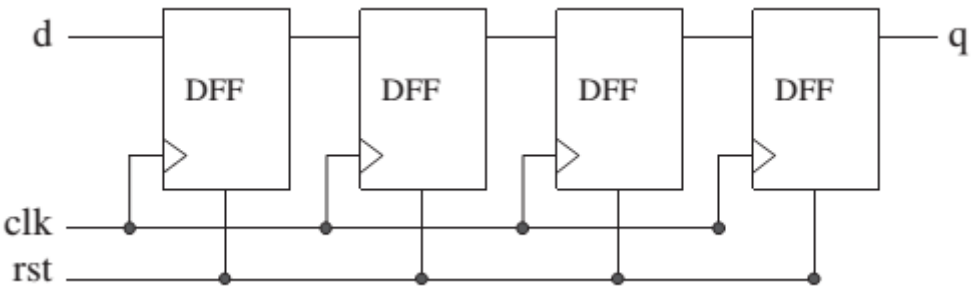
```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity shift_reg_SIPO is
5      Generic (N: integer := 4); -- numero de estágios
6      Port ( reset : in  STD_LOGIC;
7             clk : in  STD_LOGIC;
8             din : in  STD_LOGIC;
9             saida : out  STD_LOGIC_VECTOR (N-1 downto 0));
10 end shift_reg_SIPO;
11
12 architecture Behavioral of shift_reg_SIPO is
13 signal A : STD_LOGIC_VECTOR (N-1 downto 0) := (others=>'0');
14 begin
15     process(clk)
16     begin
17         if rising_edge(clk) then
18             A <= A(N-2 downto 0) & din;  -- shift left
19         end if;
20     end process;
21     saida <= A;
22 end Behavioral;
```

Descrição em VHDL de um *shift register* SISO de *N* bits

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity shift_reg_SISO is
    Generic (N: integer := 4); -- numero de estágios
    Port ( reset :  in  STD_LOGIC;
           clk   :  in  STD_LOGIC;
           din   :  in  STD_LOGIC;
           dout  : out  STD_LOGIC);
end shift_reg_SISO;

architecture Behavioral of shift_reg_SISO is
signal A : STD_LOGIC_VECTOR (N-1 downto 0) := (others=>'0');
begin
    process(clk,reset)
    begin
        if reset='1' then
            A <= (others=>'0');
        elsif rising_edge(clk) then
            A <= din & A(A'left downto 1);  -- shift right
        end if;
    end process;
    dout <= A(0);
end Behavioral;
```



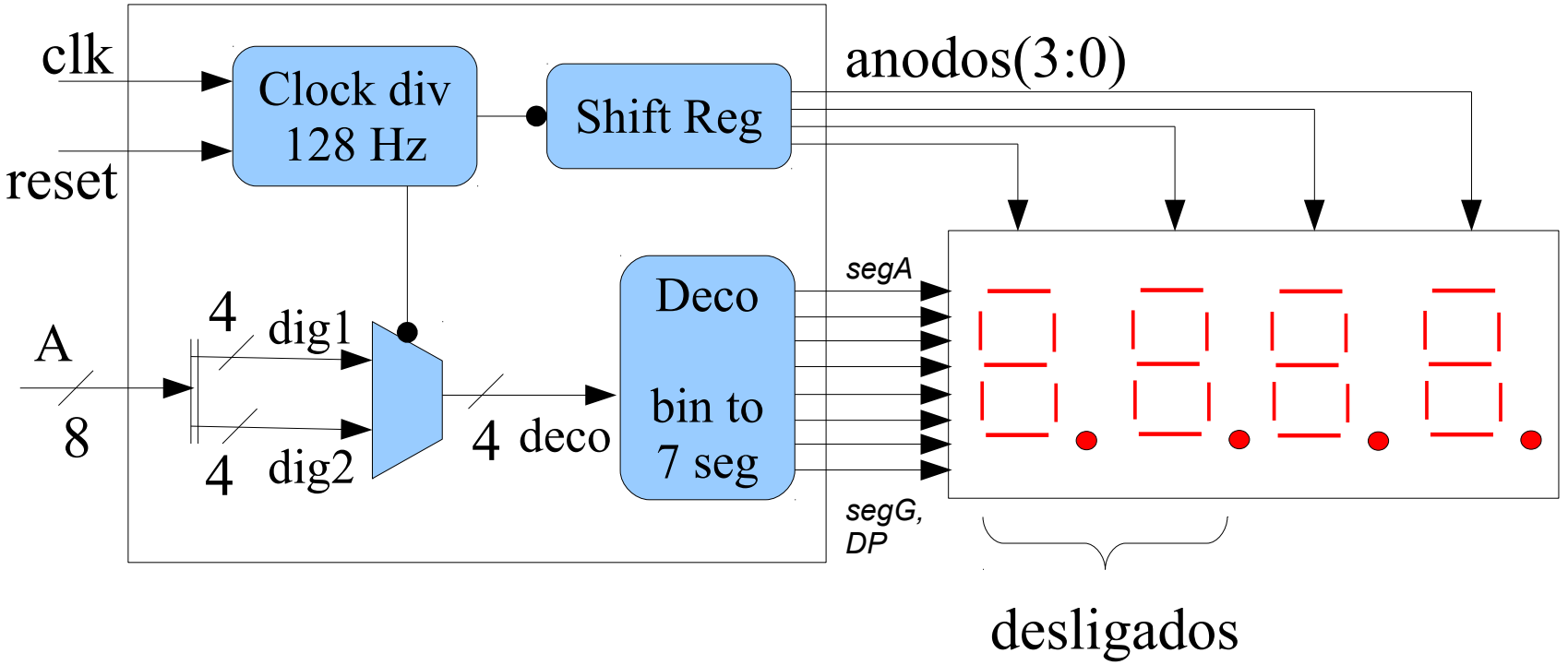
Descrição em VHDL de um *shift register* PISO de 8 bits

```
25  entity parallel_8b_shift is
26  port (
27      reset    :  in std_logic;
28      clk      :  in std_logic;
29      enable   :  in std_logic;
30      A        :  in std_logic_vector(7 downto 0);
31      outq     : out std_logic;
32      n_outq   : out std_logic
33  );
34  end parallel_8b_shift;
35
36  architecture behavioral of parallel_8b_shift is
37  signal s_reg : std_logic_vector(7 downto 0) := (others => '0');
38
39  begin
40
41  outq   <= s_reg(7);
42  n_outq <= not s_reg(7);
43
44  process (A,clk,enable,reset)
45  begin
46      if reset='0' then
47          s_reg <= A;
48      elsif rising_edge(clk) then
49          if enable='1' then
50              s_reg <= s_reg;
51          else
52              s_reg <= s_reg(7 downto 1) & '0';
53          end if;
54      end if;
55  end process;
56
57  end behavioral;
```

Descrição em VHDL de um *shift register* PIPO de *N* bits

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity shift_reg_SIPO is
5      Generic (N: integer := 8); -- numero de estágios
6      Port ( clk  : in STD_LOGIC;
7             R    : in STD_LOGIC_VECTOR(N-1 downto 0);
8             load : in STD_LOGIC;
9             din  : in STD_LOGIC;
10            Q: buffer STD_LOGIC_VECTOR (N-1 downto 0));
11 end shift_reg_SIPO;
12
13 architecture Behavioral of shift_reg_SIPO is
14 begin
15     process
16     begin
17         wait until cll'event and clk='1';
18         if load='1' then Q <= R;
19         else
20             desloca_direita: for i in 0 to N-2 loop
21                 Q(i) <= Q(i+1);
22             end loop;
23             Q(N-1)<=din;
24     end process;
25 end Behavioral;
```

Decodificador binário para 7 segmentos em anodo comum



VHDL decodificador binário para 7 segmentos (parte 1)

C:\Users\DanielMauricio\Desktop\VBshare\SD2\deco7seg\deco7seg.vhd - Notepad++

Arquivo Editar Localizar Visualizar Formatar Linguagem Configurações Macro Executar Plugins Janela ?

exemplo_shift_reg_SIPO_v2.vhd | exemplo_shift_reg_SIPO_v1.vhd | exemplo_shift_reg_SISO_v1.vhd | exemplo_shift_reg_PIPO_v1.vhd | deco7seg.vhd

```
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22  use IEEE.STD_LOGIC_arith.ALL;
23  use IEEE.STD_LOGIC_unsigned.ALL;
24
25  entity deco7seg is
26      Port ( reset : in  STD_LOGIC;
27             clk   : in  STD_LOGIC;
28             A   : in  STD_LOGIC_VECTOR (7 downto 0);
29             anodo_out : out  STD_LOGIC_VECTOR (3 downto 0);
30             segmentos : out  STD_LOGIC_VECTOR (7 downto 0));
31  end deco7seg;
32
33  architecture Behavioral of deco7seg is
34
35  constant preset : std_logic_vector(18 downto 0):= "1011111010111100001"; -- 128 Hz
36  signal count : std_logic_vector(18 downto 0):= (others=>'0');
37  signal control : std_logic_vector(1 downto 0):= "00";
38  signal dig1  : std_logic_vector (3 downto 0) := "0000";
39  signal dig2  : std_logic_vector (3 downto 0) := "0000";
40  signal dig3  : std_logic_vector (3 downto 0) := "0000";
41  signal dig4  : std_logic_vector (3 downto 0) := "0000";
42  signal deco  : std_logic_vector (3 downto 0) := "0000";
43
44  begin
```

VHDL decodificador binário para 7 segmentos (parte 2)

```
46      dig1 <= A(3 downto 0);
47      dig2 <= A(7 downto 4);
48      dig3 <= "0000";
49      dig4 <= "0000";
50
51      process(clk,reset)
52          variable anodos : std_logic_vector(4 downto 0) := "11110";
53      begin
54          if rising_edge(clk) then
55              if reset='1' then
56                  count <= (others=>'0');
57                  control <= "00";
58                  anodos := "11110";
59              elsif count=preset then
60                  count <= (others=>'0');
61                  control <= control+"01";
62                  anodos := anodos(3 downto 0) & '1';
63                  if anodos="01111" then
64                      anodos := "11110";
65                  end if;
66              else
67                  count <= count+'1';
68              end if;
69              anodo_out(3 downto 0) <= anodos(3 downto 0);
70          end if;
71      end process;
```

VHDL decodificador binário para 7 segmentos (parte 3)

```
73      with control select
74          deco <= dig1 when "00",
75                  dig2 when "01",
76                  dig3 when "10",
77                  dig4 when others;
78      process(deco)
79      begin
80          case deco is
81              when "0000" => segmentos <= "11000000"; --0
82              when "0001" => segmentos <= "11111001"; --1
83              when "0010" => segmentos <= "10100100"; --2
84              when "0011" => segmentos <= "10110000"; --3
85              when "0100" => segmentos <= "10011001"; --4
86              when "0101" => segmentos <= "10010010"; --5
87              when "0110" => segmentos <= "10000010"; --6
88              when "0111" => segmentos <= "11111000"; --7
89              when "1000" => segmentos <= "10000000"; --8
90              when "1001" => segmentos <= "10011000"; --9
91              when "1010" => segmentos <= "10001000"; --A
92              when "1011" => segmentos <= "10000011"; --b
93              when "1100" => segmentos <= "11000110"; --c
94              when "1101" => segmentos <= "10100001"; --d
95              when "1110" => segmentos <= "10000110"; --E
96              when others => segmentos <= "10001110"; --F
97          end case;
98      end process;
```

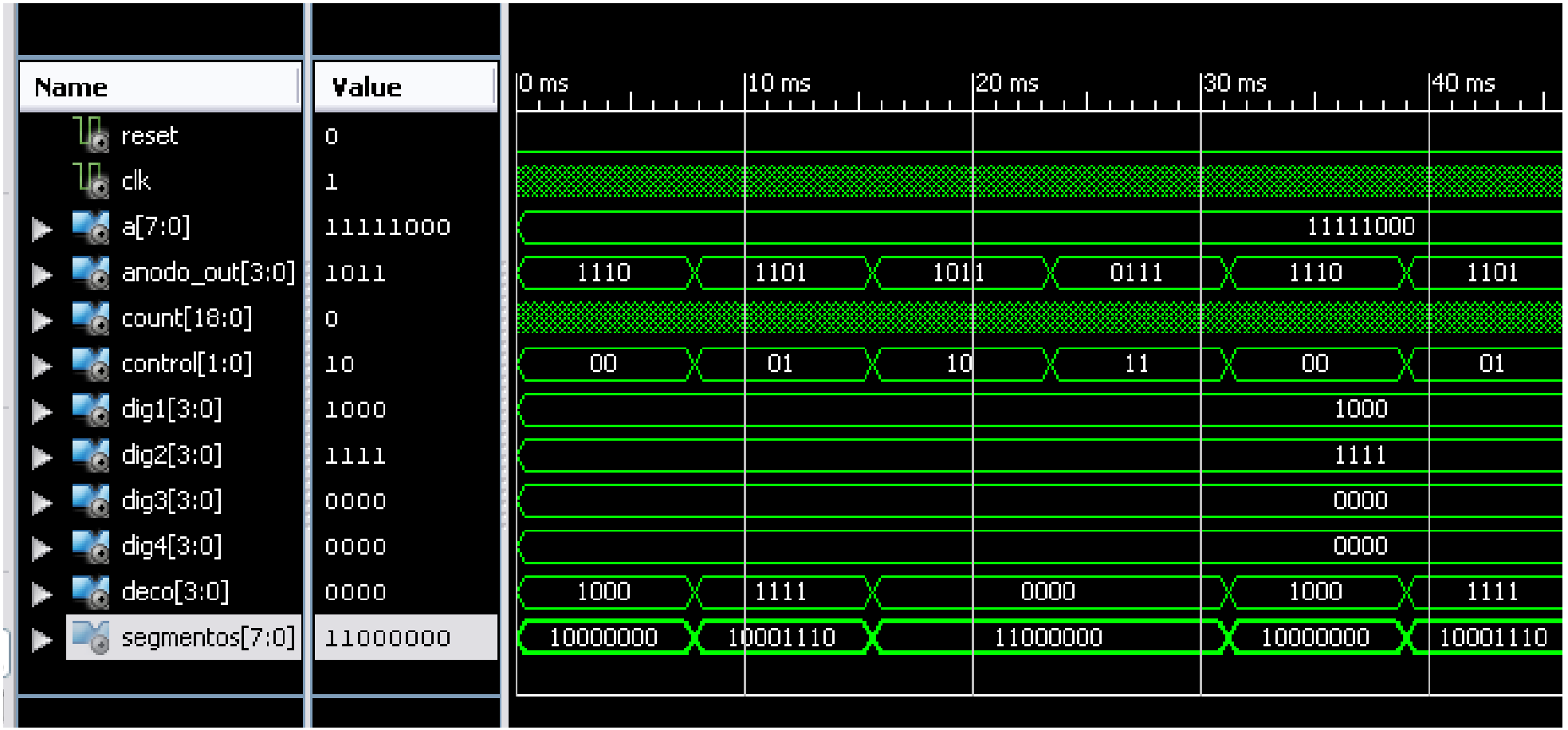
Testbench decodificador binário para 7 segmentos (parte 1)

```
38  ARCHITECTURE behavior OF tb_deco7seg IS
39
40      -- Component Declaration for the Unit Under Test (UUT)
41      COMPONENT deco7seg
42      PORT(
43           reset : IN  std_logic;
44           clk : IN  std_logic;
45           A : IN  std_logic_vector(7 downto 0);
46           anodo_out : OUT  std_logic_vector(3 downto 0);
47           segmentos : OUT  std_logic_vector(7 downto 0)
48          );
49      END COMPONENT;
50
51     --Inputs
52     signal reset : std_logic := '0';
53     signal clk : std_logic := '0';
54     signal A : std_logic_vector(7 downto 0) := (others => '0');
55
56      --Outputs
57     signal anodo_out : std_logic_vector(3 downto 0);
58     signal segmentos : std_logic_vector(7 downto 0);
59
60     -- Clock period definitions
61     constant clk_period : time := 20 ns;
62
63  BEGIN
```

Testbench decodificador binário para 7 segmentos (parte 2)

```
65      -- Instantiate the Unit Under Test (UUT)
66      uut: deco7seg PORT MAP (
67              reset => reset,
68              clk => clk,
69              A => A,
70              anodo_out => anodo_out,
71              segmentos => segmentos
72            );
73
74      -- Clock process definitions
75      clk_process :process
76      begin
77           clk <= '0';
78           wait for clk_period/2;
79           clk <= '1';
80           wait for clk_period/2;
81      end process;
82
83      -- Stimulus process
84      stim_proc: process
85      begin
86         reset <= '1'; wait for 100 ns;
87         reset <= '0'; wait for clk_period*10;
88         A <= "11111000"; wait;
89      end process;
```

Testbench decodificador binário para 7 segmentos (parte 3)



Atividade: implementar um contador BCD de 8 bits up/down e decodificador para 7 segmentos. O contador deve incrementar ou decrementar a uma frequência de 1 segundo. **(Dica: ver página p. 115 livro Volnei Pedroni)**