

Projeto com Circuitos Reconfiguráveis

Projeto lógico sequencial divisor de *clock*, *c*ontadores e *shift registers*

Prof. Daniel M. Muñoz Arboleda

FGA - UnB



Descrição em VHDL de um divisor de clock

```
⊟entity clock div is
 7
   ⊟port (
8
        reset : in std logic;
9
        clk : in std logic;
        Hz 1 : out std logic;
10
        Hz 2 : out std logic;
11
12
        Hz 5 : out std logic;
13
        Hz 10 : out std logic
14
15
     end clock div;
16
17
   Farchitecture behavioral of clock div is
18
        signal preset 1Hz : unsigned(25 downto 0) := "101111110101111000010000000";
       signal preset 2Hz : unsigned(25 downto 0) := "00101111101011110000100000";
19
20
        signal preset 5Hz : unsigned (25 downto 0) := "000100110001001011010000000";
       signal preset 10Hz: unsigned(25 downto 0) := "000010011000100101101000000";
21
22
        signal count 1Hz : unsigned(25 downto 0) := (others => '0');
23
        signal count 2Hz : unsigned(25 downto 0) := (others => '0');
24
        signal count 5Hz : unsigned(25 downto 0) := (others => '0');
25
        signal count 10Hz : unsigned(25 downto 0) := (others => '0');
26
        signal s 1Hz : std logic;
       signal s 2Hz : std logic;
27
       signal s_5Hz : std_logic;
28
29
        signal s 10Hz
                         : std logic;
30
   ⊟begin
31
32
    Hz 1 <= s 1Hz;
33
    Hz 2 <= s 2Hz;
34
    Hz 5 <= s 5Hz;
35
    Hz 10 <= s 10Hz;
```



Descrição em VHDL de um divisor de clock

```
37
         -- counter frequency 1Hz
38
         process (reset, clk)
39
         begin
40
             if reset = '0' then
41
                 count 1Hz <= preset 1Hz;
42
                 s 1Hz <= '0';
43
             elsif rising edge(clk) then
44
                 if count 1Hz = 0 then
45
                      s 1Hz
                               <= not s 1Hz;
46
                      count 1Hz <= preset 1Hz;
47
48
                      count 1Hz <= count 1Hz - 1;
49
                 end if;
50
             end if;
51
         end process;
```

```
| parallel_ob_stillt.vtiu | setialstillt.vtiu | Clock_div.vtiu | ilip_liop.vtiu
69
          -- counter frequency 5Hz
70
          process (reset, clk)
71
          begin
72
               if reset = '0' then
73
                    count 5Hz <= preset 5Hz;
74
                    s 5Hz
                               <= '0';
75
               elsif rising edge(clk) then
76
                    if count 5Hz = 0 then
77
                        s 5Hz
                                 \leq not s 5Hz;
78
                        count 5Hz <= preset 5Hz;
79
                   else
80
                        count 5Hz <= count 5Hz - 1;
81
                   end if;
               end if:
82
83
          end process;
84
```

```
37
         -- counter frequency 1Hz
38
         process (reset, clk)
39
         begin
40
             if reset = '0' then
41
                  count 1Hz <= preset 1Hz;
42
                  s 1Hz
                        <= '0';
43
             elsif rising edge(clk) then
44
                 if count 1Hz = 0 then
45
                      s 1Hz
                                <= not s 1Hz;
46
                      count 1Hz <= preset 1Hz;
47
48
                      count 1Hz <= count 1Hz - 1;
49
                 end if;
50
             end if;
51
         end process;
52
```

```
84
85
         -- counter frequency 10Hz
86
         process (reset, clk)
87
         begin
88
              if reset = '0' then
                  count 10Hz <= preset 10Hz;
90
                  s 10Hz
                             <= '0';
91
              elsif rising edge(clk) then
92
                  if count 10Hz = 0 then
93
                                 <= not s 10Hz;
                      s 10Hz
94
                      count 10Hz <= preset 10Hz;
95
                  else
                      count 10Hz <= count 10Hz - 1;
97
                  end if:
98
              end if;
         end process;
```



Descrição em VHDL de um contador de 4bits (exemplo 1)

```
library IEEE;
21
                                                                     process (clk, reset)
                                                                 43
                                                                      begin
22
     use IEEE.std logic 1164.all;
                                                                 44
                                                                          if reset='1' then
     use IEEE.numeric std.all;
23
                                                                              s count <= "1111";
                                                                 45
2.4
                                                                 46
                                                                          elsif rising edge(clk) then
25
    □entity contador 4bit is
                                                                             if up down = '1' then
                                                                 47
    bort (
26
                                                                 48
                                                                                 s count <= s count + 1;
27
                  : in std logic;
         reset
                                                                             elsif up down = '0' then
                                                                 49
28
                : in std logic;
                                                                 50
                                                                                 s count <= s count - 1;
         clk
                                                                 51 白
                                                                              else
        up down : in std logic;
29
                                                                 52
                                                                                 s count <= s count;
30
         Q out : out std logic vector(3 downto 0)
                                                                 53
                                                                              end if;
     );
31
                                                                 54
                                                                          end if:
32
     end contador 4bit;
                                                                 55
                                                                      end process;
33
                                                                 56
34
                                                                 57
                                                                      end behavioral:
    □architecture behavioral of contador 4bit is
35
36
     signal s count : unsigned(3 downto 0) := (others => '1111');
37
38
    ⊟begin
39
40
     Q out <= std logic vector(s count);
```

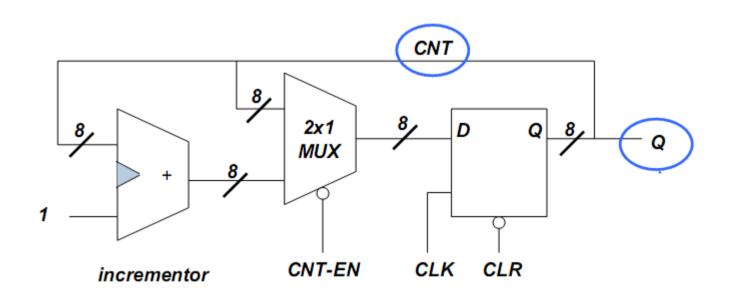


Descrição em VHDL de um contador de 4bits (exemplo 2)

```
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;
                     :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 \le CNT + '1';
     end if :
   end if;
 end process FIRST;
Q \le CNT;
end BEHAVE;
```



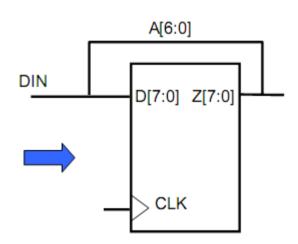
Descrição em VHDL de um contador de 4bits (exemplo 2 cont.)





Descrição em VHDL de um shift register de 8 bits serial

```
library ieee;
use ieee.std logic 1164.all;
entity SHIFTER is
port (CLK, DIN: in std logic;
     Z: out std logic vector(7 downto 0));
end SHIFTER;
architecture RTL of SHIFTER is
signal A: std logic vector(7 downto 0);
begin
process (CLK)
begin
if (CLK'event and CLK='1') then
   A <= A (6 downto 0) & DIN; -- shift left
end if;
end process;
Z \leq A;
end RTL
```





Descrição em VHDL de um shift register de 8 bits paralelo

```
⊟entity parallel 8b shift is
26 |port (
27
                : in std logic;
                : in std logic;
29
       enable : in std logic;
30
                 : in std logic vector(7 downto 0);
31
                : out std logic;
         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
   ⊟begin
40
41
     outq <= s reg(7);
42
     n outq \leq not s reg(7);
43
    process (A,clk,enable,reset)
45
     begin
46
         if reset='0' then
             s req <= A;
48
         elsif rising edge(clk) then
49
             if enable='1' then
50
                 s req <= s req;
52
                 s reg <= s reg (7 downto 1) & '0';
53
             end if;
54
         end if:
     end process;
56
     end behavioral;
```