Laboratório de Sistemas Digitais Aula Teórica-Prática 4

Ano Letivo 2020/21

Modelação em VHDL de circuitos sequenciais elementares, contadores, divisores de frequência e temporizadores

Parametrização de componentes sequenciais



Conteúdo

- Modelação em VHDL de circuitos sequenciais
 - Latch D
 - Flip-flop tipo D
 - Registos
 - Parametrização
 - Contadores
 - Divisores de frequência
 - Temporizadores



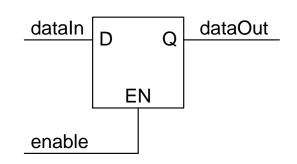
Módulo Sequencial Trivial – Latch D

```
entity LatchD is
   port(enable : in std logic;
        dataIn : in std logic;
        dataOut : out std logic);
end LatchD;
architecture Behav of LatchD is
begin
  process(enable, dataIn)
  begin
    if (enable = '1') then
      dataOut <= dataIn;</pre>
    end if:
  end process;
```

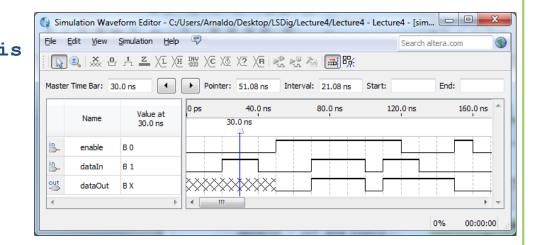
use IEEE.STD LOGIC 1164.all;

library IEEE;

end Behav:



A saída "segue" a entrada quando enable= \1'

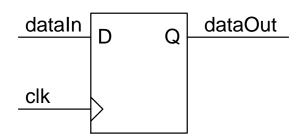


Porque razão o sinal dataOut surge inicialmente como "XXXXXXXXX"? É relevante? Como resolver?



Módulo Sequencial Simples com Clock Flip-flop tipo D

```
library IEEE;
use IEEE.STD LOGIC 1164.all;
entity FFD is
  port(clk : in std logic;
        dataIn : in std logic;
        dataOut : out std logic);
end FFD;
architecture Behav of FFD is
begin
 process(clk)
```



Porque razão apenas o clk é incluído na lista de sensibilidade?

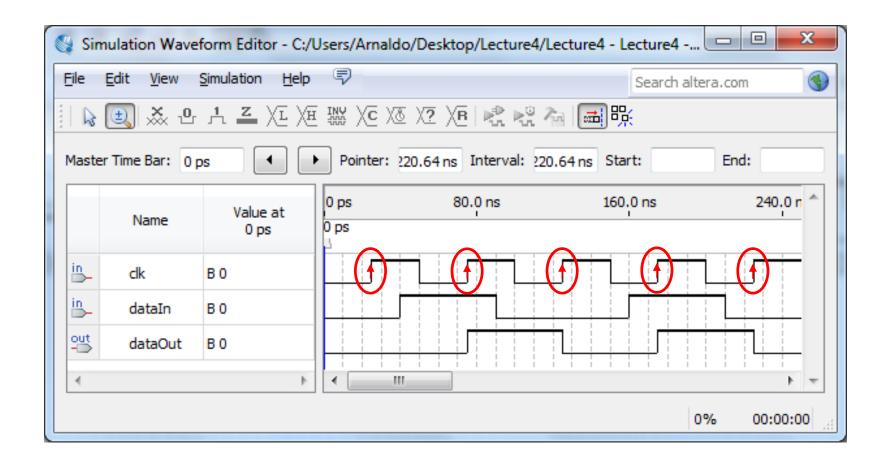
```
clk'event and clk = 1' é
 begin
    if (clk'event and clk = '1') then
      dataOut <= dataIn;</pre>
    end if:
  end process;
end Behav:
```

equivalente a rising edge (clk)

```
clk'event and clk = '0' é
equivalente a falling edge (clk)
```



Simulação do Flip-Flop D



FF tipo D com Enable

```
enable : in std_logic;
        dataIn : in std_logic;
        dataOut : out std logic);
end FFDEn;
architecture Behav of FFDEn is
begin
 process (clk)
  begin
    if (rising edge(clk)) then
      if (enable = '1') then
        dataOut <= dataIn;</pre>
      end if;
    end if;
  end process;
end Behav;
```

port(clk : in std logic;

entity FFDEn is

```
dataIn D Q dataOut

clk

EN

enable
```

```
🦭 Simulation Waveform Editor - C:/Users/Arnaldo/Desktop/LSDig/Lecture4/Lecture4 - Lecture4 - [simulation/qs... 😐 😐
File Edit View Simulation Help 🐬
                                                                           Search altera.com
  Master Time Bar: 0 ps
                            ▶ Pointer: 200.9 ns
                                                Interval: 200.9 ns
                                                                        240.0 ns
                                                                                        320.0
                  Value at
               В 0
       enable
               B 0
       dataIn
       dataOut
              B O
```



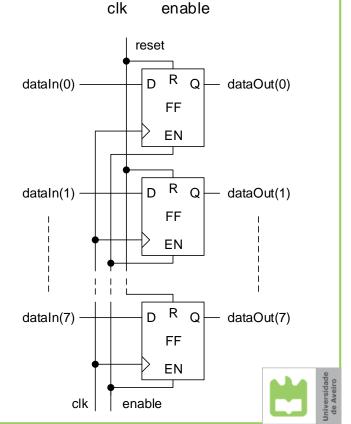
FF tipo D com Enable e Reset

```
entity FFDEnRst is
 port(reset : in std logic;
       clk : in std logic;
       enable : in std logic;
       dataIn : in std logic;
       dataOut : out std logic);
end FFDEnRst;
architecture BehavRAsyn of FFDEnRst is
begin
                          Variante com
  process(reset, clk)
                        Reset Assíncrono
  begin
    if (reset = '1') then
      dataOut <= '0';</pre>
    elsif (rising edge(clk)) then
      if (enable = '1') then
        dataOut <= dataIn;</pre>
      end if:
    end if;
  end process;
end BehavRAsyn;
```

```
architecture BehavRSync of FFDEnRst is
begin
                    Variante com
  process(clk)
                    Reset Síncrono
  begin
    if (rising edge(clk)) then
      if (reset = '1') then
        dataOut <= '0';</pre>
      elsif (enable = '1') then
        dataOut <= dataIn;</pre>
      end if;
    end if;
  end process;
                  reset
end BehavRSync;
                            R
                  dataIn
                                   dataOut
                  clk
                            FN
                  enable
```

```
Registo de 8 bits
entity FFD8EnRst is
 port(reset : in std logic;
       clk
              : in
                    std logic;
      enable
             : in
                    std logic;
      dataIn : in std logic vector(7 downto 0);
      dataOut : out std logic vector(7 downto 0));
end FFD8EnRst;
architecture Behav of FFD8EnRst is
begin
 process(clk)
 begin
                                  Exemplo com
    if (rising edge(clk)) then
                                 Reset Síncrono
      if (reset = '1') then
       dataOut <= (others => '0');
      elsif (enable = '1') then
       dataOut <= dataIn;</pre>
      end if;
   end if;
  end process;
```

end Behav:



reset

Registo

FFD8EnRst

dataOut

Registo de Tamanho Parametrizável

```
entity FFDNEnRst is

generic(N : positive := 8);

port(reset : in std_logic;

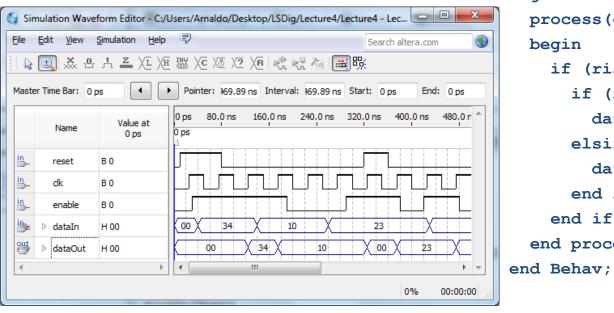
    clk : in std_logic;

    enable : in std_logic;

    dataIn : in std_logic_vector((N-1) downto 0);

    dataOut : out std_logic_vector((N-1) downto 0));

end FFDNEnRst;
```



architecture Behav of FFDNEnRst is begin

```
process(clk)
begin
  if (rising_edge(clk)) then
    if (reset = '1') then
       dataOut <= (others => '0');
    elsif (enable = '1') then
       dataOut <= dataIn;
    end if;
  end if;
end process;</pre>
Exemplo com
Reset Sincrono
```

Contador Binário Crescente de 8 bits com Enable e Reset Síncrono

```
entity BinUCntEnRst8 is
                                                                                      reset
  port(reset : in std logic;
              : in std logic;
         clk
                                                                                              BinUCntEnRst8
        enable : in std logic;
                                                                                                         cntOut
         cntOut : out std logic vector(7 downto 0));
end BinUCntEnRst8:
                                                             Porque razão é
                                                           necessário declarar
architecture Behav of BinUCntEnRst8 is
                                                                                     clk
                                                           o sinal s cntValue?
  signal s cntValue : unsigned(7 downto 0);
                                                                                     enable
begin
  process (clk)
                                                   💱 Simulation Waveform Editor - C:/Users/Arnaldo/Desktop/LSDig/Lecture4/Lecture4 - Lecture4 - [simulation/qsi... 🖵 📳
  begin
                                                   File Edit View Simulation Help 🐬
                                                                                                     Search altera.com
                                                    if (rising edge(clk)) then
       if (reset = '1') then
                                                    Master Time Bar: 0 ps
                                                                        Pointer: 453,26 ns
                                                                                               Start: 0 ns
          s cntValue <= (others => '0');
                                                                                 160.0 ns
                                                                                       240.0 ns
                                                                                              320.0 ns
                                                                                                     400.0 ns
                                                               Value at
       elsif (enable = '1') then
                                                        reset
          s cntValue <= s cntValue + 1;</pre>
       end if:
                                                        enable
                                                      01 X 02
     end if:
```

end process;

end Behav:

cntOut <= std logic vector(s cntValue);</pre>

00:00:00

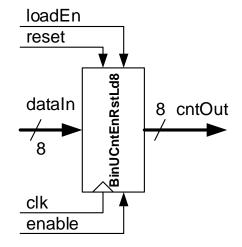
Contador Binário Crescente/Decrescente

```
architecture Behav of BinUDCntEnRst8 is
library IEEE;
                                                signal s cntValue : unsigned(7 downto 0);
use IEEE.STD LOGIC 1164.all;
                                              begin
use IEEE.NUMERIC STD.all;
                                                process(clk)
                                                begin
entity BinUDCntEnRst8 is
                                                   if (rising edge(clk)) then
port(reset : in std logic;
                                                       if (reset = '1') then
      clk : in std logic;
                                                         s cntValue <= (others => '0');
      enable : in std logic;
                                                      elsif (enable = '1') then
      upDown n : in std logic;
                                                          if (upDown n = '1') then
      cntOut : out std logic vector(7 downto 0));
                                                             s cntValue <= s cntValue + 1;</pre>
end BinUDCntEnRst8;
                                                          else
                      upDown_n
                                                             s cntValue <= s cntValue - 1;</pre>
                       reset
                                                          end if:
                                                      end if:
                             BinUDCntEnRst8
                                                   end if:
                                     cntOut
                                                end process;
                                                cntOut <= std logic vector(s cntValue);</pre>
                      clk
                                              end Behav :
                      enable
```

```
entity BinUCntEnRstLd8 is
  port(reset : in std logic;
       clk : in std logic;
       enable : in std logic;
       loadEn : in std logic;
       dataIn : in std logic vector(7 downto 0);
       cntOut : out std logic vector(7 downto 0));
end BinUCntEnRstLd8;
architecture Behav of BinUCntEnRstLd8 is
  signal s cntValue : unsigned(7 downto 0);
begin
  process(clk)
  begin
    if (rising edge(clk)) then
      if (reset = '1') then
        s cntValue <= (others => '0');
      elsif (enable = '1') then
        if (loadEn = '1') then
          s cntValue <= unsigned(dataIn);</pre>
        else
          s cntValue <= s cntValue + 1;</pre>
        end if;
      end if;
    end if;
  end process;
  cntOut <= std_logic_vector(s cntValue);</pre>
end Behav ;
```

Contador Binário com Entrada de Carregamento Paralelo

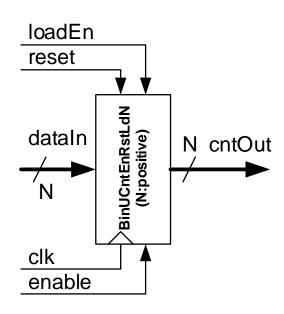
Sinais de controlo loadEn,
upDown_n e/ou outros podem ser
combinados no mesmo contador
de acordo com a prioridade
relativa pretendida



```
entity BinUCntEnRstLdN is
  generic(N : positive := 8);
port(reset : in std_logic;
    clk : in std_logic;
  enable : in std_logic;
  loadEn : in std_logic;
  dataIn : in std_logic_vector((N-1) downto 0);
  cntOut : out std_logic_vector((N-1) downto 0));
```

end BinUCntEnRstLdN;

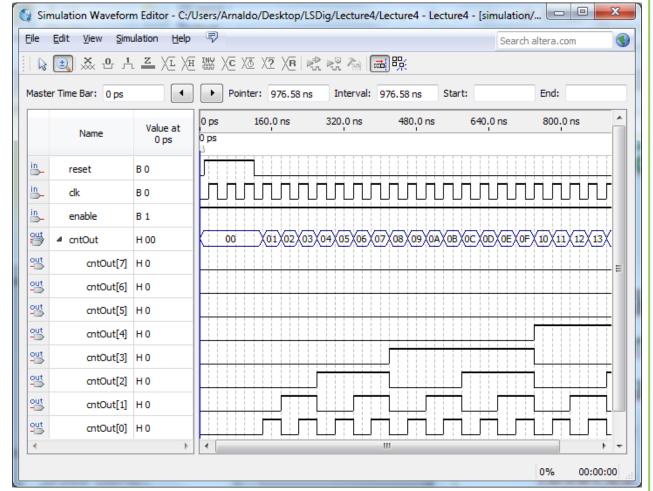
Parametrização do Número de Bits do Contador



```
architecture Behav of BinUCntEnRstLdN is
  signal s cntValue : unsigned((N-1) downto 0);
begin
  process (clk)
  begin
    if (rising edge(clk)) then
      if (reset = '1') then
        s cntValue <= (others => '0');
      elsif (enable = '1') then
        if (loadEn = '1') then
           s cntValue <= unsigned(dataIn);</pre>
        else
           s cntValue <= s cntValue + 1;</pre>
        end if:
      end if:
    end if;
  end process;
  cntOut <= std logic vector(s cntValue);</pre>
end Behav ;
```

Divisão da Frequência de um Sinal de Relógio (*clock*) por 2^N

A divisão da frequência de um sinal de relógio por fatores inteiros "potência de base 2" pode ser efetuada por um contador



Divisor (simples) de Frequência

- A divisão da frequência de um sinal de relógio por <u>fatores inteiros</u> <u>arbitrários (K)</u> requer hardware "mais elaborado" (baseado num contador de modulo <u>K</u>)
- Exemplo de um módulo divisor de frequência <u>configurável</u> <u>estaticamente</u> (<u>K</u> fixado em <u>compile</u> time, aquando da instanciação com generic map)

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;

entity FreqDivStatic is
  generic(K : positive := 4);
  port(reset : in std_logic;
        clkIn : in std_logic;
        clkOut : out std_logic);
end FreqDivStatic;
```

```
architecture Behavioral of FreqDivStatic is
   signal s counter : natural;
                                      reset
 begin
   process(clkIn)
                                 clkln
                                          clkOut
   begin
     if rising edge(clkIn) then
        if ((reset = '1') or
            (s counter = K - 1)) then
                  <= '0';
          clkOut
          s counter <= 0;</pre>
       else
          if (s counter = K/2 - 1) then
            clkOut <= '1';
         end if;
         s counter <= s counter + 1;</pre>
       end if:
     end if:
   end process;
 end Behavioral;
  Contador free running de módulo K
clkOut <= '1' a "meio" da contagem
```

clkOut <= '0' no final da contagem

Simulação do Div. (Simples) de Freq.

```
K=4
                                                                 Vamos assumir
process (clkIn)
                                                                                                                                   f<sub>clkOut</sub>?
                                                              que f_{clkin} = 50 \text{ MHz}
begin
                                                                                                                                Duty cycle?
   if rising edge(clkIn) then
      if ((reset = '1') or
                                                             Simulation Waveform Editor - C:/Users/asroliveira/CloudStation/Desktop/LSDig/QuartusProjs/Lectur...
                                                             File Edit View Simulation Help 🐬
                                                                                                                                   Search altera.com
             (s counter = K - 1) then
                                                              □ X 4 1 Z X X W X X X X X X X A □ □ □
                        <= '0';
          clkOut
                                                             Master Time Bar: 0 ps
                                                                                  ◆ Pointer: 21.76 ns
                                                                                                        Interval: 21.76 ns
          s counter <= 0;
                                                                                                       60.0 ns
                                                                                                                     100.0 ns
                                                                                                                            120.0 ns
      else
          if (s counter = K/2 - 1) then
                                                                   s counter S 0
             clkOut
                         <= '1':
                                                                   dkOut
          end if:
          s counter <= s counter + 1;
                                                                                                                                             00:00:00
      end if:
   end if:
end process;
                             💲 Simulation Waveform Editor - C:/Use.s/asroliveira/CloudStation/Desktop/LSDig/QuartusProjs/Lecture4/Lecture4 - Lecture4 - [simulation/q... 🕒 📙
                              File Edit View Simulation Help 🐬
                                                                                                                        Search altera.com
                               💫 📵 🕉 🗗 🚣 准 洭 洭 骤 ※ 🗷 💆 🗷 😹 鶰 鶰
                                                      ◆ Pointer: 67.64 ns
                                                                                Interval: 67.64 ns
                              Master Time Bar: 0 ps
                                                                                                    Start:
                                                                 40.0 ns
                                                                         60.0 ns
                                                                                         100.0 ns
                                                                                                         140.0 ns
                                                                                                 120.0 ns
                                                                                                                                 200.0 n
                                           Value at
          K=5
        f<sub>clkOut</sub>?
    Duty cycle?
```

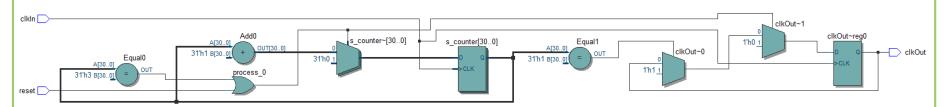
00:00:00

Divisor de Frequência Sintetizado

TPC

Análise do
 circuito
 resultante da
 síntese do
 modelo em VHDL

```
process(clkIn)
begin
   if rising edge(clkIn) then
      if ((reset = '1') or
           (s counter = K - 1)) then
        clkOut
                 <= '0';
        s counter <= 0;</pre>
      else
         if (s counter = K/2 - 1) then
             clkOut
                     <= '1';
         end if;
         s counter <= s counter + 1;</pre>
      end if:
   end if;
end process;
```



Divisor de Frequência Programável Dinamicamente (Entidade)

 Exemplo de um módulo divisor de frequência programável / configurável dinamicamente (em runtime através do porto divFactor)

reset

```
library IEEE;
use IEEE.STD LOGIC 1164.all;
                                                             clkOut
                                             divFactor
use IEEE.NUMERIC STD.all;
entity FreqDivProg is
                                             clkln
  generic(N : positive := 3);
 port(reset
                  : in std logic;
       divFactor : in std logic vector(N-1 downto 0);
       clkIn : in std logic;
                                           O número de bits "N" do fator de
       clkOut
                  : out std logic);
                                            divisão é fixado estaticamente
End FreqDivProg;
```

O valor do fator de divisão é definido dinamicamente

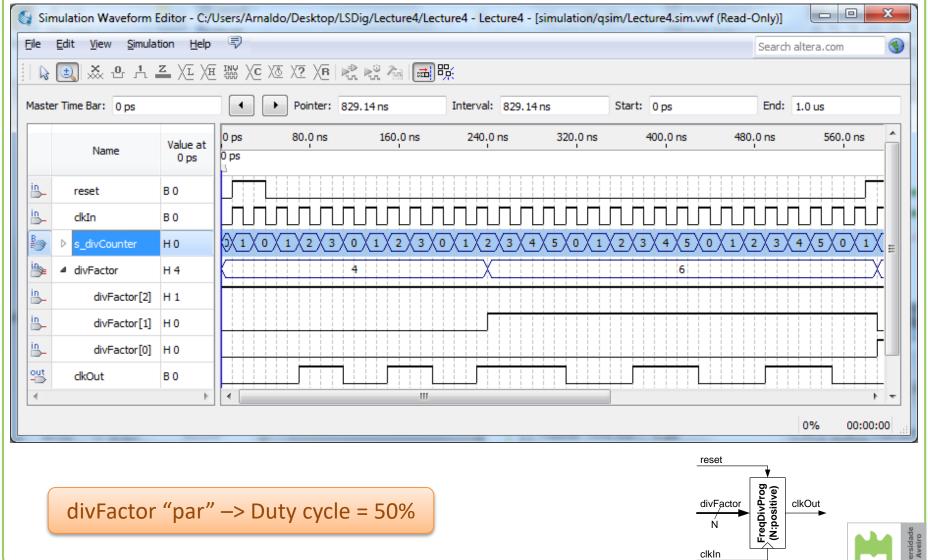
Div. de Freq. Programável Dinamicamente (Arquitetura) 0 ps 80.0 ns

```
reset
architecture Behavioral of FreqDivProg is
  signal s divCounter : unsigned(N-1 downto 0);
                                                                s_divCounter
begin
  process(clkIn)
                                                                  divFactor[2] H 1
  begin
                                                                  divFactor[1] H 0
                                                                  divFactor[0] H 0
    if (rising edge(clkIn)) then
                                                                dkOut
      if ((reset = '1') or
           (s divCounter >= unsigned(divFactor) - 1)) then
         clkOut
                       <= '0':
         s divCounter <= (others => '0');
      else
         if (s divCounter = (unsigned(divFactor)/2 - 1)) then
           clkOut <= '1';
         end if:
         s divCounter <= s divCounter + 1;</pre>
      end if:
                             Descrição semelhante ao
    end if;
                     FreqDivStatic, mas em que o fator
  end process;
                       de divisão é programável em run-time
end Behavioral:
```

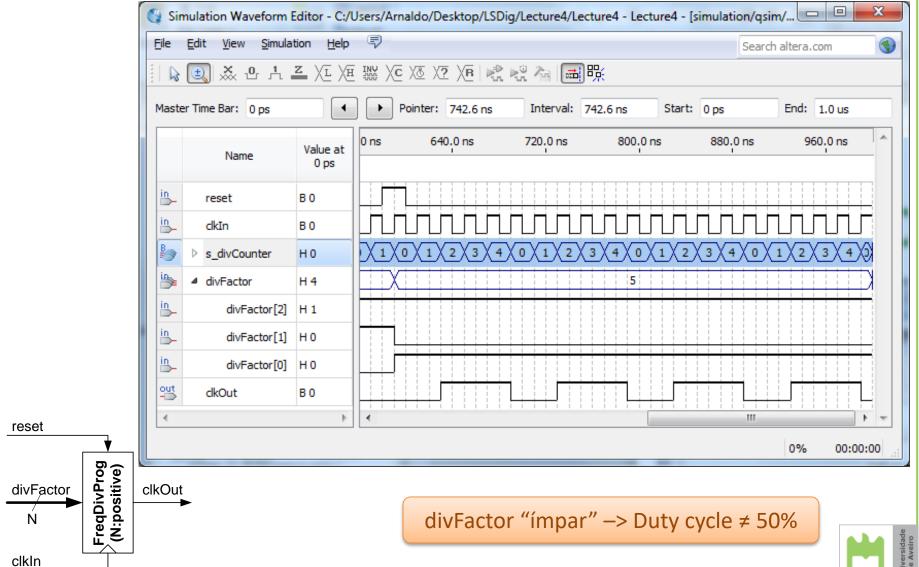
Value at 0 ps

ΗО

Divisor de Frequência (Simulação)

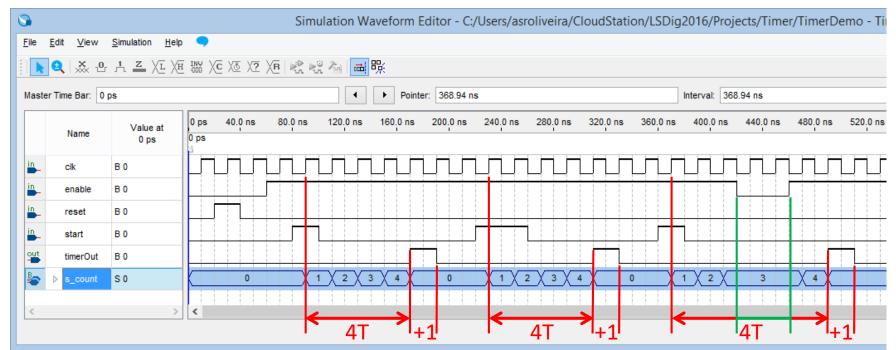


Divisor de Frequência (Simulação)



Temporizador (Exemplo de Comportamento e Simulação)

- Um temporizador é um módulo usado para medir tempo, ou para gerar um evento após ter decorrido um dado intervalo de tempo (depois do temporizador ter sido iniciado/disparado)
- Exemplo em que a saída ("timerOut") é ativada durante 1T, após um intervalo de tempo predefinido (4T) depois do disparo da entrada ("start")



T = período do sinal de relógio

Temporizador (Exemplo de Implementação em VHDL)

```
library ieee;
use ieee.std logic 1164.all;
entity TimerOnDelay is
    generic(K : positive := 5);
   port(clk : in std logic;
         reset : in std logic;
         enable : in std logic;
         start : in std logic;
         timerOut : out std logic);
end TimerOnDelay;
        start
        reset
               (K:positive)
                     timerOut
        clk
        enable
```

TPC: Desenvolver um novo temporizador em que a saída é ativada após o disparo do temporizador e desativada após ter decorrido o intervalo de tempo KT (com K programável dinamicamente)

```
architecture Behavioral of TimerOnDelay is
                  signal s count : integer := 0;
             begin
                  assert(K >= 2);
                                              K deve ser ≥ 2. Porquê?
                 process(clk)
                 begin
                      if (rising edge(clk)) then
                          if (reset = '1') then
Inicialização
                               timerOut <= '0';</pre>
                               s count <= 0;
Teste do sinal "enable"
                          elsif (enable = '1') then
                              if (s count = 0) then
Se contador parado
                                   if (start = '1') then
                                       s_count <= s count + 1;
Deteção de um novo disparo
                                   end if:
Desativação da saída
                                   timerOut <= '0';</pre>
                               else
                                   if (s count = (K - 1)) then
Deteção do final de contagem
                                        timerOut <= '1';</pre>
e ativação da saída
                                        s count <= 0;
                                   else
                                       timerOut <= '0';</pre>
Incremento do contador
                                       s count <= s count + 1;
"a meio" da contagem
                                   end if;
                               end if:
                          end if:
                      end if;
                 end process;
```

end Behavioral;

Temporizador Sintetizado

- TPC
 - Análise do circuito resultante da síntese do modelo em VHDL

s_count~[95..64]

```
elsif (enable = '1') then
             if (s count = 0) then
                 if (start = '1') then
                      s count <= s count + 1;
                 end if;
                 timerOut <= '0';</pre>
             else
                 if (s count = (K - 1)) then
                      timerOut <= '1';</pre>
                      s count <= 0;
                 else
                      timerOut <= '0';</pre>
                      s count <= s count + 1;</pre>
                 end if:
             end if:
        end if:
    end if:
end process;
```

process(clk)

if (rising_edge(clk)) then
 if (reset = '1') then

timerOut <= '0';
s count <= 0;</pre>

begin

Estrutura Geral de um Processo VHDL de um Temporizador

Tipos de temporizadores:

- Atraso "à operação"
 - Saída ativada após decorrido um tempo predefinido
- Atraso "à desoperação"
 - Saída desativada após decorrido um tempo predefinido

```
timerOut

clk
start

timerOut

clk
start

timerOut

Atraso à operação

timerOut

Atraso à desoperação
```

```
process(clk)
begin
  Em cada flanco ativo do sinal de relógio
    Se o sinal de reset estiver ativo
      Desativa saída
      Coloca contador a 0
    Senão, se o sinal de enable estiver ativo
      Se o temporizador estiver parado (contador = 0)
        Se a entrada start estiver ativa
           Desativa a saída / Ativa a saída
           Incrementa o contador
        Senão
           Desativa a saída
      Senão (contador /= 0)
        Se o contador tiver atingido o valor máximo (K-1)
           Ativa a saída / Desativa a saída
           Coloca contador a 0
        Senão
           Desativa a saída / Ativa a saída
           Incrementa o contador
```

Comentários Finais

- Todos os modelos apresentados podem ser usados da forma fornecida (módulo autónomo reutilizável com Entity + Architecture), ou, alternativamente, o processo que descreve a funcionalidade pode também ser integrado em módulos (arquiteturas) mais complexos com outros processos, instanciações de componentes e/ou atribuições concorrentes
- No final desta aula e do trabalho prático 4 de LSD, deverá ser capaz de:
 - Modelar componentes sequenciais fundamentais em VHDL
 - Registos
 - Contadores
 - Divisores de frequência
 - Temporizadores
- No final do trabalho prático 5 de LSD, deverá ser capaz de:
 - Modelar componentes parametrizáveis combinatórios e sequenciais em VHDL
- ... bons trabalhos práticos 4 e 5, disponíveis no site da UC 😊
 - elearning.ua.pt