#### Laboratório de Sistemas Digitais Aula Teórica-Prática 4

Ano Letivo 2018/19

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

Parametrização de componentes sequenciais

Arnaldo Oliveira, Augusto Silva, Iouliia Skliarova



#### Conteúdo

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



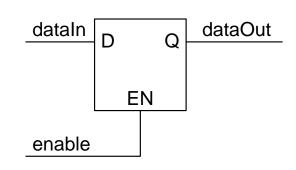
#### Módulo Sequencial Trivial – Latch D

```
use IEEE.STD LOGIC 1164.all;
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;
```

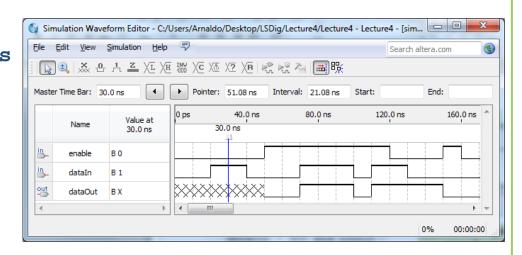
library IEEE;

end process;

end Behav;



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



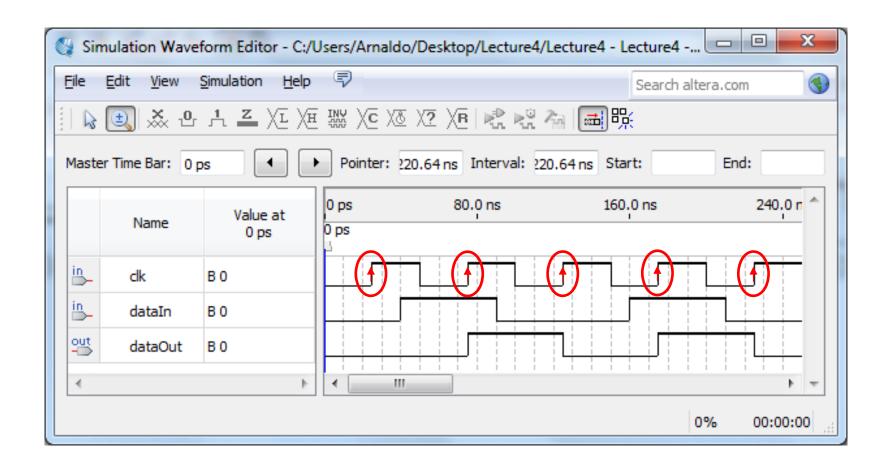
Porque razão o sinal dataOut surge inicialmente como "XXXXXXXXX"? É relevante? Como resolver?



#### Módulo Sequencial Simples com Clock Flip-flop tipo D

```
library IEEE;
use IEEE.STD LOGIC 1164.all;
                                                            dataOut
                                             dataln
entity FFD is
                                             clk
   port(clk : in std_logic;
        dataIn : in std logic;
        dataOut : out std logic);
                                             Porque razão apenas o
end FFD;
                                               clk é incluído na
                                             lista de sensibilidade?
architecture Behav of FFD is
begin
                                 clk'event and clk = '1' é
  process(clk)
                                equivalente a rising edge(clk)
  begin
    if (clk'event and clk = '1') then
      dataOut <= dataIn;</pre>
    end if;
                                 clk'event and clk = 0' é
  end process;
                                equivalente a falling edge(clk)
end Behav;
```

#### Simulação do Flip-Flop D



#### FF tipo D com Enable

dataIn

dataOut B 0

```
entity FFDEn is
                                                                   dataIn
    port(clk : in std logic;
            enable : in std_logic;
           dataIn : in std_logic;
                                                                   clk
                                                                               ΕN
           dataOut : out std logic);
end FFDEn;
                                                                   enable
architecture Behav of FFDEn is
begin
                                              💱 Simulation Waveform Editor - C:/Users/Arnaldo/Desktop/LSDig/Lecture4/Lecture4 - Lecture4 - [simulation/as... 😑 📳
                                               File Edit View Simulation Help 🐬
  process(clk)
                                                💫 📵 🗴 🖁 🛧 🚄 🕮 WE WE XE XE XE 🕏 😭 🦓 🔚 鶰 👺
  begin
                                               Master Time Bar: 0 ps
                                                             Pointer: 200.9 ns
                                                                            Interval: 200.9 ns
                                                                                160.0 ns
      if (rising edge(clk)) then
                                                         Value at
         if (enable = '1') then
                                                   enable B 0
           dataOut <= dataIn;</pre>
```

end if;

end if;

end process;

end Behav:



320.0

dataOut

Search altera.com

240.0 ns

#### FF tipo D com Enable e Reset

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

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

#### Registo de Tamanho Parametrizável

```
entity FFDNEnRst is

generic(N : positive := 8);

port(reset : in std_logic;

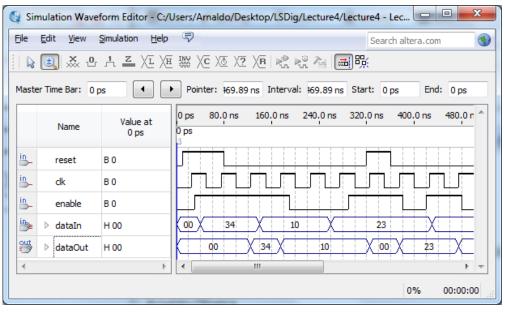
    clk : in std_logic;

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)

```
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;
end Behav;</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;
        clk
                : in std logic;
        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/gsi... 🖃 📳
  begin
                                                File Edit View Simulation Help
                                                 if (rising edge(clk)) then
       if (reset = '1') then
                                                Master Time Bar: 0 ps
                                                                   Pointer: 453, 26 ns
                                                                                        Start: 0 ps
         s_cntValue <= (others => '0');
       elsif (enable = '1') then
         s cntValue <= s cntValue + 1;</pre>
       end if:
                                                                           01 X 02
                                                   ▷ cntOut
    end if:
  end process;
  cntOut <= std logic vector(s cntValue);</pre>
end Behav:
```

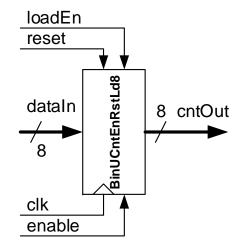
#### Contador Binário Crescente/Decrescente

```
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
               : in std logic;
 port(reset
                                                      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;
              : in std_logic;
       clk
       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 :
```

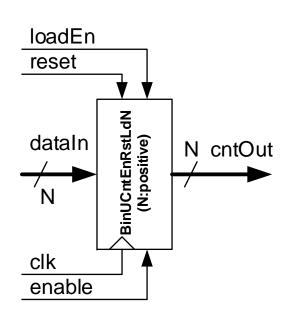
# Contador Binário com Entrada de Carregamento Paralelo

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));
```

## Parametrização do Número de Bits do Contador

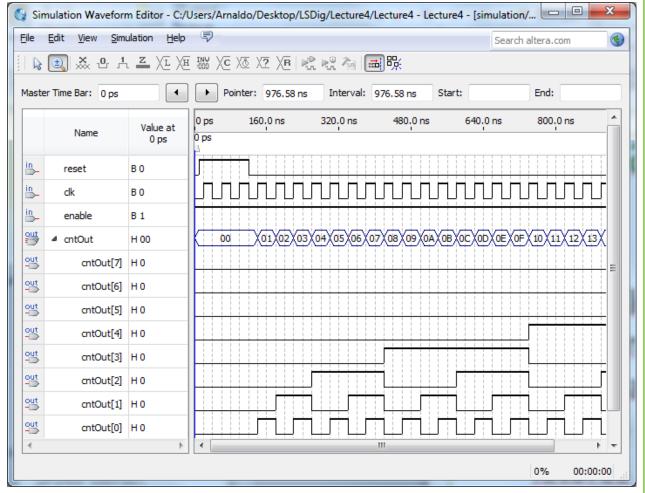


end BinUCntEnRstLdN;

```
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<sup>N</sup>

 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

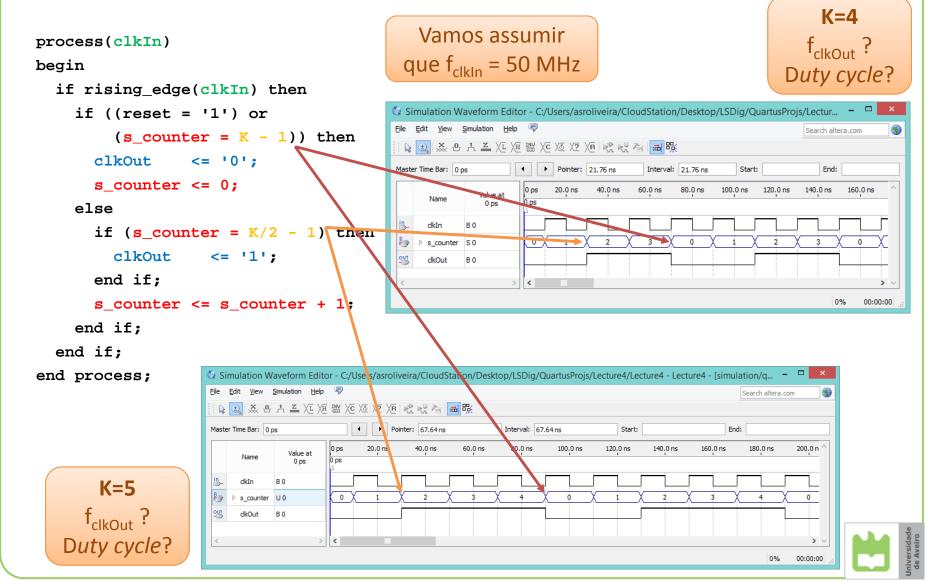
- A divisão da frequência de um sinal de relógio por <u>fatores inteiros</u> <u>arbitrários (K)</u> requer hardware "mais elaborado" (baseado num contador de modulo K)
- Exemplo de um módulo divisor de frequência <u>configurável</u> <u>estaticamente</u> (<u>K</u> fixado em *compile* time, aquando da 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
                                         FreqDivStatic (K:positive)
   process(clkIn)
                                              clkOut
                                   clkln
   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;
 end Behavioral;
   Contador free running de módulo K
clkOut <= '1' a "meio" da contagem</pre>
clkOut <= '0' no final da contagem</pre>
```

#### Simulação do Div. (Simples) de Freq.

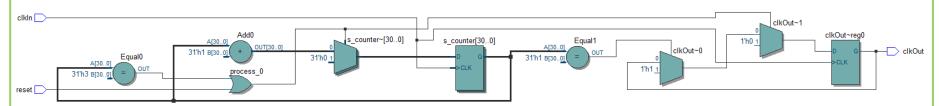


#### Divisor de Frequência Sintetizado

#### 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;
      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;
                                             divFactor
                                                              clkOut
use IEEE.NUMERIC STD.all;
entity FreqDivProg is
  generic(N : positive := 3);
                                             clkln
  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)

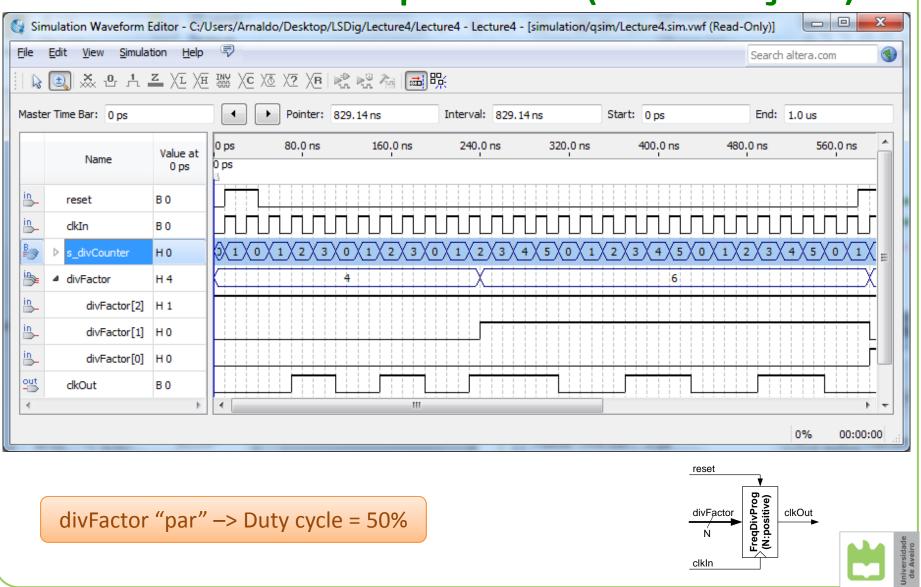
```
B 0
architecture Behavioral of FreqDivProg is
  signal s divCounter : unsigned(N-1 downto 0);
                                                                        ΗО
begin
                                                                divFactor
  process(clkIn)
                                                                  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';
                                                                    reset
         end if:
         s_divCounter <= s_divCounter + 1;</pre>
      end if;
                                                                    divFactor
                              Descrição semelhante ao
    end if;
                     FreqDivStatic, mas em que o fator
  end process;
                       de divisão é programável em run-time
end Behavioral;
                                                                    clkIn
```

80.0 ns

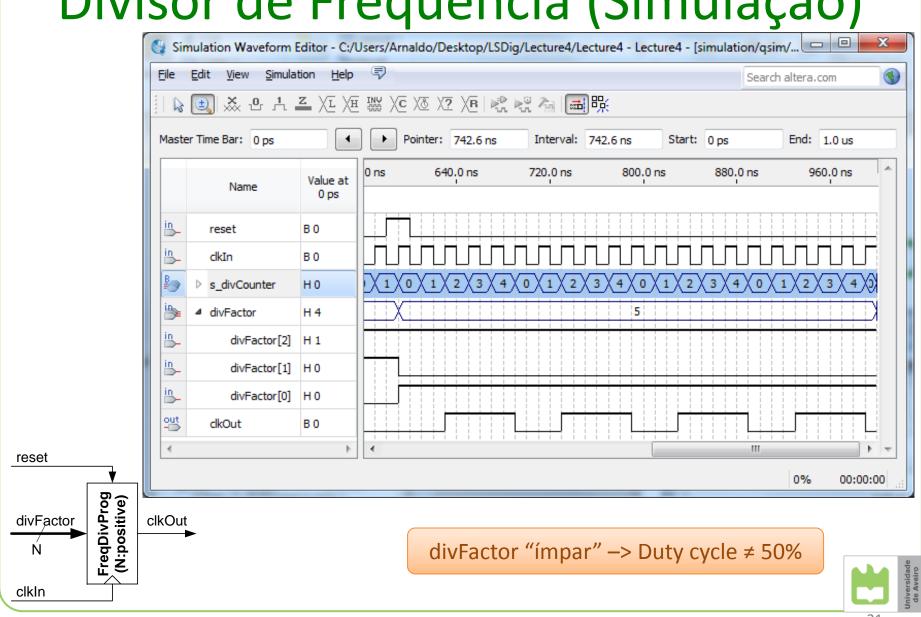
0 ps

Value at 0 ps

#### Divisor de Frequência (Simulação)

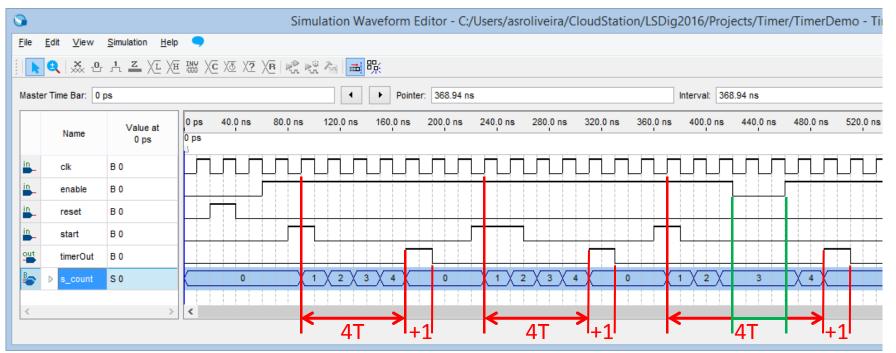


#### Divisor de Frequência (Simulação)



## 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")



## 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;
                  : in std logic;
         start
         timerOut : out std logic);
end TimerOnDelay;
        start
        reset
             Timer
                     timerOut
        clk
```

**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)

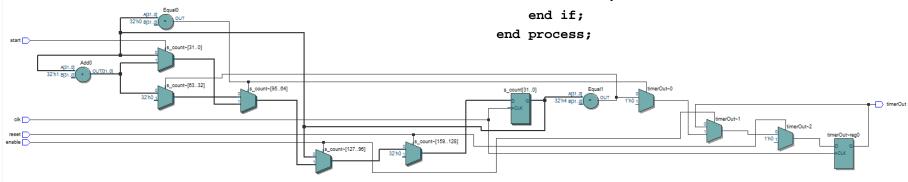
enable

```
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';
                              s count <= 0;
                          elsif (enable = '1') then
Teste do sinal "enable"
                              if (s count = 0) then
Se contador parado
                                  if (start = '1') then
Deteção de um novo disparo
                                       s count <= s count + 1;
                                  end if;
Desativação da saída
                                  timerOut <= '0';</pre>
                              else
                                  if (s count = (K - 1)) then
Deteção do final de contagem
                                       timerOut <= '1';
e ativação da saída
                                       s count <= 0;
                                  else
                                       timerOut <= '0';
Incremento do contador
"a meio" da contagem
                                       s_count <= s_count + 1;</pre>
                                  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

```
begin
    if (rising_edge(clk)) then
         if (reset = '1') then
             timerOut <= '0';</pre>
             s count <= 0;
         elsif (enable = '1') then
             if (s_count = 0) then
                 if (start = '1') then
                     s count <= s count + 1;
                 end if;
                 timerOut <= '0';
             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)

## 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 LSDig, deverá ser capaz de:
  - Modelar componentes sequenciais fundamentais em VHDL
    - Registos
    - Contadores
    - Divisores de frequência
    - Temporizadores
- No final do trabalho prático 5 de LSDig, 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