LaPSE

Laboratório de Prototipação Digital
e Sistemas Embarcados

Universidade do Vale do Rio dos Sinos - UNISINOS

Prototipação em *PLDs*

Técnicas de Implementação em *VHDL*

Autor: Prof. Rodrigo Marques de Figueiredo

Agenda

- Restrições
- Recomendações
- Biblioteca Xilinx (Primitivas)

Restrições

- Em descrição de *hardware* deve-se tomar cuidado com o *hardware* e não com a simplicidade do código, pois muitas vezes isso causa dualidades que não podem ou são resolvidas de maneira errônea pelo sintetizador;

- Para evitar problemas provenientes dessas dualidades deve-se seguir algumas restrições.

Restrições

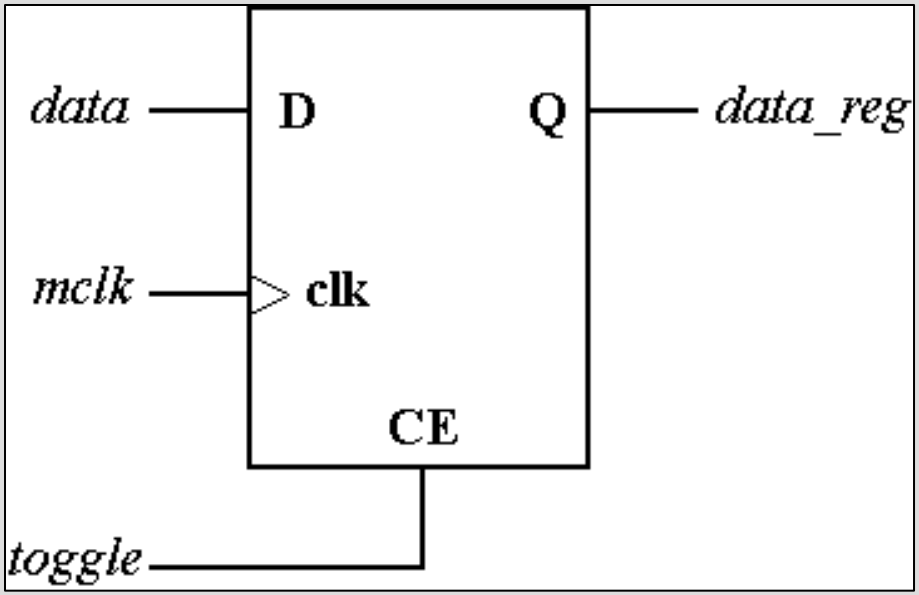
- **Acumuladores ou registradores:**

- Sempre registrá-los utilizando como referência o clock do sistema para aquela determinada área de *hardware*;

- Ou seja, só haverá espaço para a utilização do atributo *event* para os sinais de *clock*, caso outro sinal registre um determinado dado deve-se utilizar um circuito de sincronismo, como o do exemplo a seguir:

Restrições

- **Circuito de Sincronismo:**



Restrições

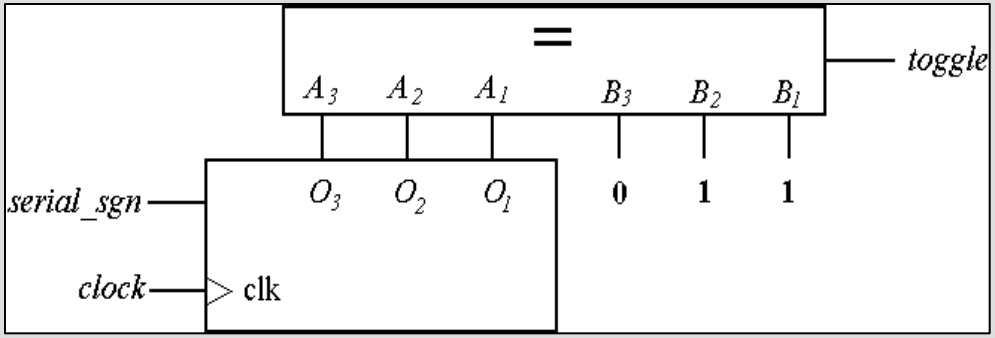
- **Detectores de Borda:**

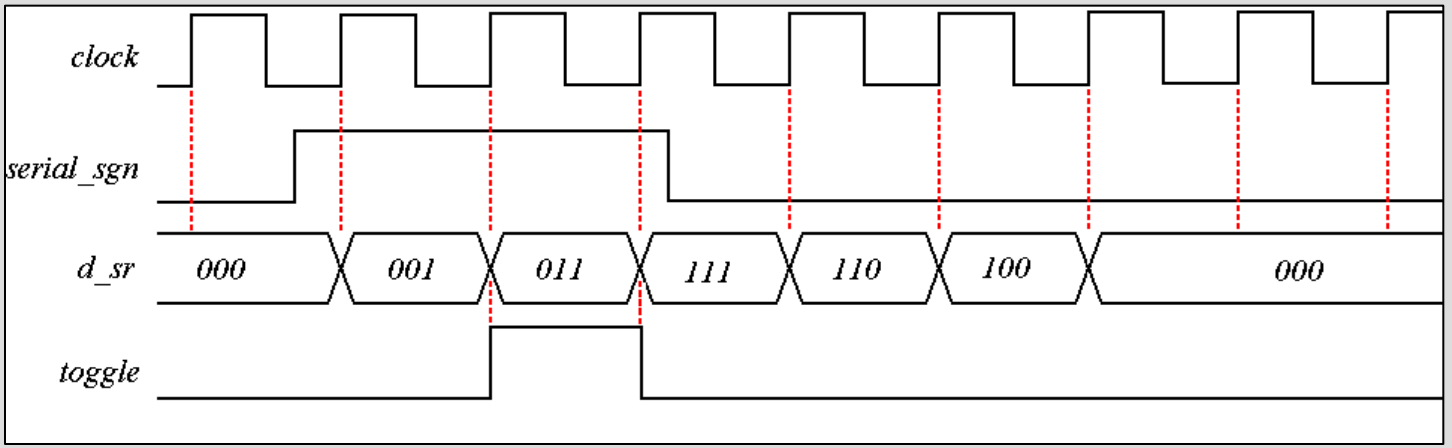
- Com a restrição de não utilizar indiscriminadamente o atributo *event*, fica a questão de como fazer um detector de borda sem a utilização do mesmo;

- A solução é fazer um detector de borda amostrando o sinal de entrada. Com a opção por generic deste ter saída em nível ou por toggle.

Restrições

- **Detector de Borda:**





Restrições

- ***Shift-Registers*:**

- Para implementar um *shift-register* em *VHDL* deve-se utilizar o comportamento procedural de comandos dentro de um processo, fazendo um código com a seguinte construção:

```
<sinal_sr>(0)          <= <sinal_serial>;
<sinal_sr>(7 downto 1) <= <sinal_sr>(6 downto 0);
```

Restrições

- ***Shift-Registers*:**

- Esta decrição para o sintetizador da *Xilinx* (ISE) é inferida diretamente em uma LUTSR para este trecho de *hardware*, o que gera um ganho na redução do *hardware* da implementação.

Este fato é muito importante para otimização de espaço em hardware e minimização de área em projetos com estas restrições.

Restrições

- **Multiplexadores:**

- Usar sempre o comando **when** para todas as ordens de multiplexação, quando esta for assíncrona;
- Quando a multiplexação for obrigatoriamente síncrona, deve-se então utilizar o comando case a partir da 3ª ordem de multiplexação inclusive.

Restrições

- **Comando *For*:**

- Em códigos sintetizáveis usado somente no *set* e *reset* de matrizes e barramentos interdependentes, ou em gerações rotas de implementação (**for...generate**);

- Para códigos não sintetizáveis pode ser utilizado normalmente sem quaisquer restrições.

Restrições

- **Comando *While*:**

- Nunca utilizar este comando para gerar um código sintetizável, pois este não possui inferência coerente;

- Para testes deve-se tomar muito cuidado em utilizar este comando, uma vez que seu comportamento é sucetível a diversos fatores, o torna o seu uso muito criterioso e por conseqüência de implementação demorada. Evitar este comando mesmo em códigos não sintetizáveis.

Recomendações

- As recomendações visam alertar o projetista para a prevenção de possíveis problemas de implementação e ocupação de área de hardware;

- Outro tipo de problema a ser evitado é a performance em termos computacionais, pois melhorando o roteamento melhora-se a velocidade máxima de operação do *hardware*.

Recomendações

- **Operações Aritméticas:**

- Como VHDL é descrição de *hardware* e em neste tipo de implementação as operações matemáticas não são de fácil implementação no que se trata de ocupação de área, deve-se então ter em mente o seguinte:

  Em *hardware* existe apenas 1 função diretamente implementável que é a soma;

  A multiplicação pode ser dada de maneira indireta, através da técnica de multiplicar analógicamente e decodificar para digital.

Recomendações

- **Soma / Subtração:**

- Otimizadas para uso diretamente em VHDL;

- Deve-se cuidar o uso da subtração no caso de números negativos (atentar para o complemento de 2);

- Também atentar para a necessidade ou não do uso da **library** *signed* e *unsigned*.

Recomendações

- **Multiplicação:**

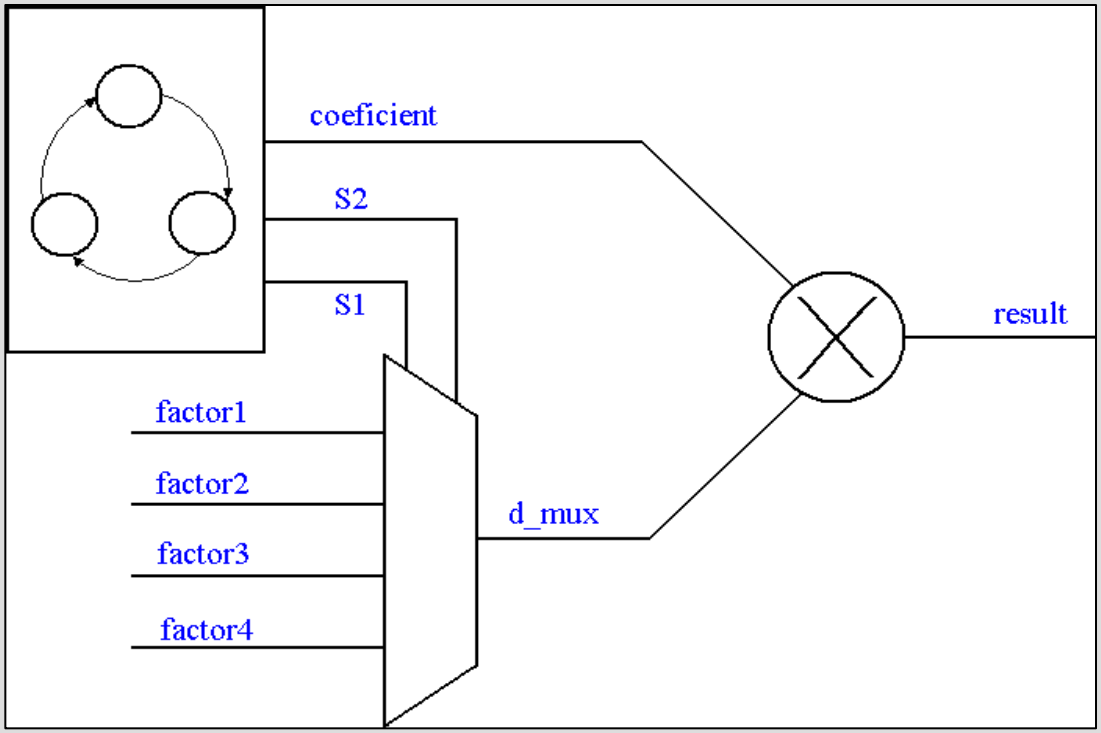
- Pode-se utilizar a multiplicação diretamente no código com um símbolo (*);

- Para obter-se performance e otimização de área em *hardware* deve-se utilizar a instanciação de uma primitiva *hardcore*;

- Com esta abordagem pode-se inclusive conseguir-se um reaproveitamento de multiplicadores.

Recomendações

- **Multiplicação:**



Recomendações

- **Divisão:**

- Deve-se ter o mesmo cuidado dado a multiplicação;

- Exige-se uma atenção extra para com o truncamento de bits que deve ser feito de maneira criteriosa;

- Não existe uma primitiva para divisões, mas sim técnicas para o uso de multiplicadores associados a divisores binários (*shift* de *bits*).

Recomendações

- ***Encoders* / *Decoders*:**

- Sempre que possível (99,99% das vezes) utilizar o comando case para implementar um encoder/decoder;

- Pois o sintetizador reconhece neste comando o indicador para o uso de uma estrutura de uma ULA (encoder), o que provê uma maior otimização na síntese para *hardware*.

Recomendações

- **Máquinas de Estado Finito:**

- Usar sempre máquinas de estado usuais, Mealy e Moore

- Observar a aplicação para qual as características destas são utilizadas;

- Dar preferência a máquina Moore, pois esta tem melhor performance em FPGAs (esta possui uma simplificação de *hardware* maior na arquitetura que estes utilizam para sua implementação).

Recomendações

- ***Generate*:**

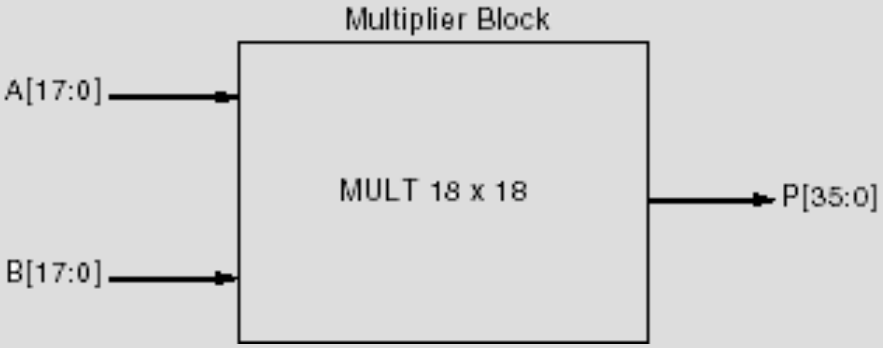
- Usar sempre que possível este comando, pois assim pode-se parametrizar o código (além de deixá-lo mais limpo)

- Cria previsões rotas de debug no hardware, e diminui o tempo de preparo de setup de teste;

- O *hardware* descrito pode ser alterado de maneira simples.

Biblioteca *Xilinx*

- **Multiplicadores:**




```
MULT18X18_inst : MULT18X18
  port  (
     P   : out std_logic_vector(35 downto 0);
     A   : in  std_logic_vector(17 downto 0);
     B   : in  std_logic_vector(17 downto 0)
  );
```

Biblioteca *Xilinx*

- **Utilização da Biblioteca *Xilinx*:**
  - Para utilizar-se as bibliotecas *Xilinx* deve-se utilizar a *library* ***unisim*;**
  - *No sintetizador a simples declaração da library já suficiente para a mesma passar a ser reconhecida*
  - *Já no simulador será necessário (caso não este não possua) a instalação de um pacote de suporte a compilação e simulação das primitivas.*
  - Esta *library* é declarada da seguinte forma:

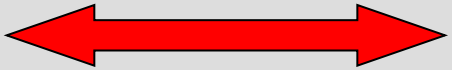
library *unisim;*

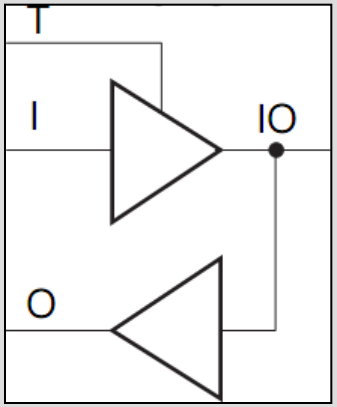
use *unisim.vcomponents.****all****;*

Biblioteca *Xilinx*

- ***IOBs*:**

```
IBUFGDS_inst : IBUFGDS
   generic (
      IOSTANDARD : string := "DEFAULT"
   );
   port  (
      O   : out std_logic;
      I    : in   std_logic;
      IB  : in   std_logic
   );
```



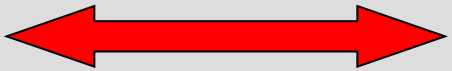


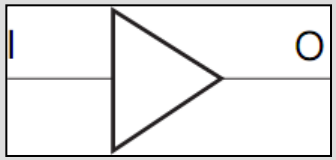
Diferencial

Biblioteca *Xilinx*

- ***IOBs*:**

```
IBUFG_inst : IBUFG
generic (
   IOSTANDARD : string := "DEFAULT"
);
port (
   O   : out std_logic;
   I    : in std_logic
);
```

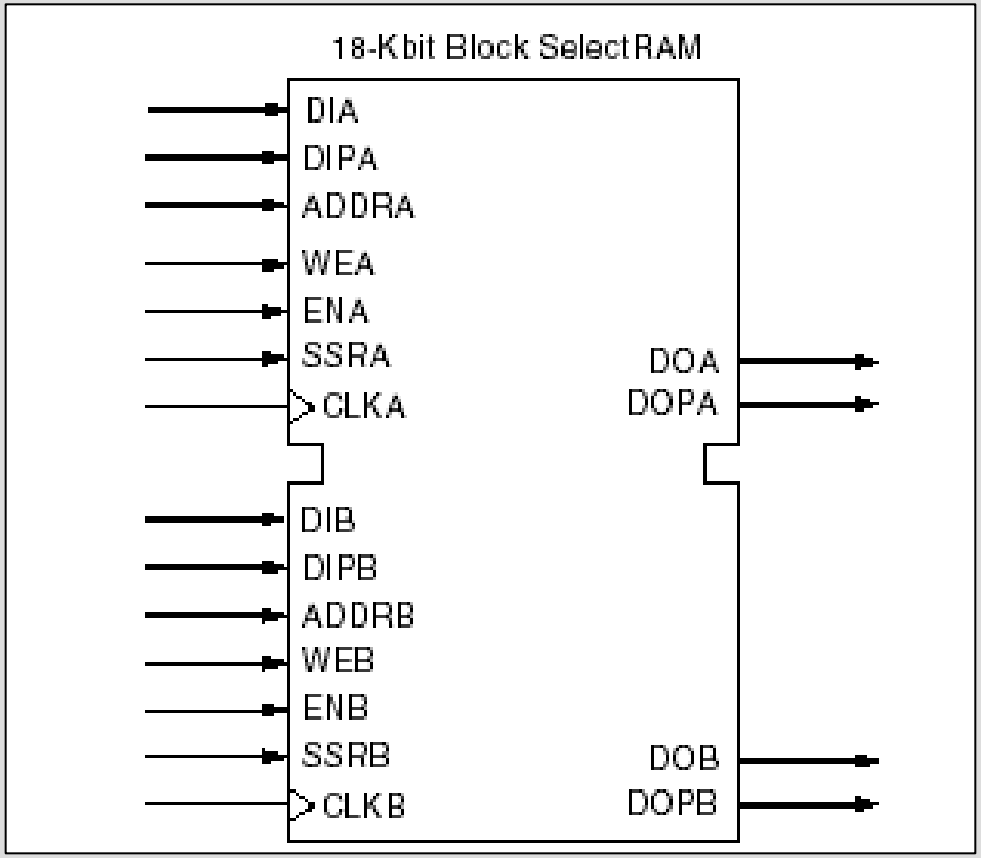




Single-end

Biblioteca *Xilinx*

- ***Block RAM* (*BRAM*):**



Biblioteca *Xilinx*

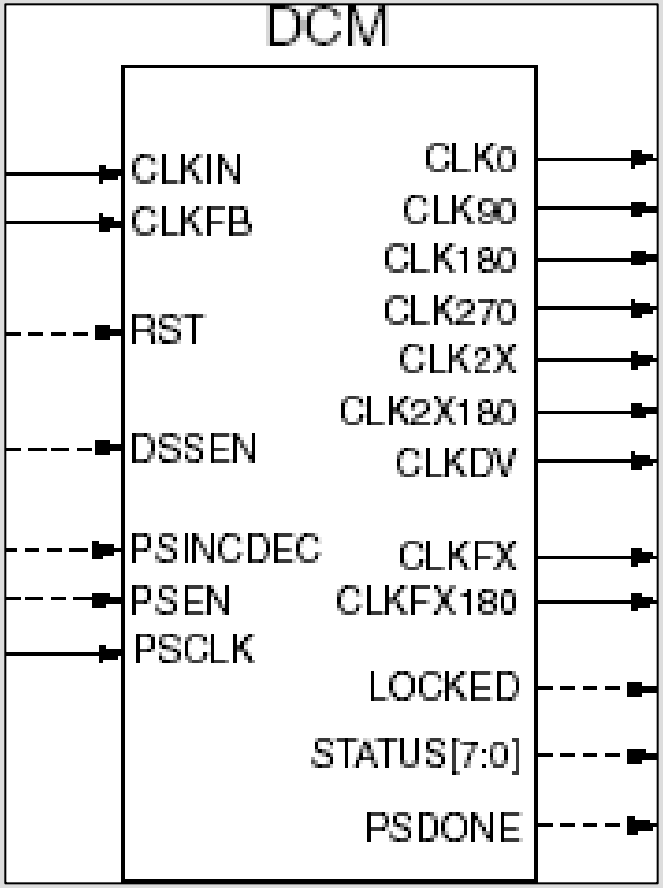
- ***Block RAM* (*BRAM*):**

```
RAMB16_S18_S18_inst : RAMB16_S18_S18
  generic  (
    INIT_A : bit_vector(19 downto 0) := x"00000";
    INIT_B : bit_vector(19 downto 0) := x"00000;
    SRVAL_A : bit_vector(19 downto 0) := x"00000";
    SRVAL_B : bit_vector(19 downto 0) := x"00000";
    WRITE_MODE_A : string := "WRITE_FIRST";
    WRITE_MODE_B : string := "WRITE_FIRST";
    SIM_COLLISION_CHECK : string :=  "ALL";
    -- Espaço de memória RAM
    INIT_00 : bit_vector(255 downto 0) := x"00...00";
.
.
.
    INIT_3F : bit_vector(255 downto 0) := x"00...00";
    -- Espaço de memória RAM (paridade)
   INITP_00 : bit_vector(255 downto 0) := x"00..00";
.
.
.
    INITP_07 : bit_vector(255 downto 0) := x"00...00"
  )
```

```
port  (
  DOA      : out std_logic_vector(15 downto 0);
  DOB      : out std_logic_vector(15 downto 0);
  DOPA     : out std_logic_vector(1 downto 0);
  DOPB     : out std_logic_vector(1 downto 0);
  ADDRA    : in std_logic_vector(9 downto 0);
  ADDRB    : in std_logic_vector(9 downto 0);
  CLKA     : in std_logic;
  CLKB     : in std_logic;
  DIA      : in std_logic_vector(15 downto 0);
  DIB      : in std_logic_vector(15 downto 0);
  DIPA     : in std_logic_vector(1 downto 0);
  DIPB     : in std_logic_vector(1 downto 0);
  ENA      : in std_logic;
  ENB      : in std_logic;
  SSRA     : in std_logic;
  SSRB     : in std_logic;
  WEA      : in std_logic;
  WEB      : in std_logic
);
```

Biblioteca *Xilinx*

- ***Digital Clock Manager* (*DCM*):**



Biblioteca *Xilinx*

- ***Digital Clock Manager* (*DCM*):**

```
DCM_inst : DCM
  generic (
    CLKDV_DIVIDE                : real := 2.0;
    CLKFX_DIVIDE                : integer := 1;
    CLKFX_MULTIPLY              : integer:= 4;
    CLKIN_DIVIDE_BY_2           : boolean := FALSE;
    CLKIN_PERIOD                : real := 0.0;
    CLKOUT_PHASE_SHIFT          : string := “NONE”;
    CLK_FEEDBACK                : string := "1X";
    DESKEW_ADJUS                : string := "SYSTEM_SYNCHRONOUS";
    DFS_FREQUENCY_MODE          : string := "LOW";
    DLL_FREQUENCY_MODE          : string := "LOW";
    DUTY_CYCLE_CORRECTION       : string := TRUE;
    FACTORY_JF                  : std_logic_vector(15 downto 0) := X"C080";
    PHASE_SHIFT                 : integer := 0;
    STARTUP_WAIT                : boolean := FALSE
)
```

```
port (
    clk0                : out std_logic;
    clk180      : out std_logic;
    clk270      : out std_logic;
    clk2x       : out std_logic;
    clk2x180    : out std_logic;
    clk90       : out std_logic;
    clkdv       : out std_logic;
    clkfx       : out std_logic;
    clkfx180    : out std_logic;
    locked      : out std_logic;
    psdone      : out std_logic;
    status      : out std_logic_vector(7 downto 0);
    clkfb       : in  std_logic;
    clkin       : in  std_logic;
    psclk       : in  std_logic;
    psen        : in  std_logic;
    psincdec    : in  std_logic;
    rst                 : in  std_logic
  );
```

UNISINOS
Somos infinitas possibilidades

LaPSE
Laboratório de Prototipação Digital e Sistemas Embarcados

LaPSE

Laboratório de Prototipação Digital
e Sistemas Embarcados

Universidade do Vale do Rio dos Sinos - UNISINOS

Prototipação em *PLDs*

Obrigado!