

# Laboratório de Sistemas Digitais

## Aula Teórico-Prática 2

Ano Letivo 2023/24

Introdução à linguagem VHDL

Modelação de  
componentes combinatórios



# Conteúdo

- Aspectos básicos de VHDL
  - Finalidade, funcionalidades e tipos de construções
  - Tipos de dados
  - Comentários
  - Identificadores
  - Exemplo simples de um multiplexador 2→1
- Componentes combinatórios
  - Diferentes abordagens de modelação em VHDL

# VHDL – Aspectos Básicos

- Very high speed integrated circuits Hardware Description Language (IEEE std 1076)
- *Hardware Description Language* (HDL)
  - Não é uma linguagem de programação de software!!! Diferente paradigma!!!
  - Modelação, simulação e síntese de sistemas digitais
- Permite descrever o comportamento e a estrutura de hardware digital
  - Utilizando construções típicas das linguagens de programação
- Utilizada com ferramentas de projeto assistido por computador para simulação, síntese e implementação
  - Permite ganhos de produtividade apreciáveis sobre métodos de projeto baseados em captura de diagramas lógicos (embora este também possua algumas vantagens!)

# VHDL – Aspectos Básicos

- Disponibiliza abstrações para as construções típicas do hardware
  - *Entities* (interface de um módulo)
  - *Architectures* (implementação de um módulo)
  - *Signals* (sinais internos de um módulo)
  - *Ports* (sinais da interface de um módulo)
  - Tipos de dados orientados para o hardware (para representar níveis lógicos, vetores de bits de diversos tamanhos, noção de tempo, etc.)
  - Concorrência (para modelar o paralelismo do hardware)
  - ...

# Tipos de Dados em VHDL

- Tipo de dados mais utilizado (para representar valores lógicos em sistemas digitais)
  - std\_logic (1 bit) / std\_logic\_vector (vários bits - barramentos)  
(tipos definidos na biblioteca IEEE.STD\_LOGIC\_1164)
    - '0' – Nível baixo**
    - '1' – Nível Alto**
    - 'L' – Nível baixo passivo**
    - 'H' – Nível alto passivo**
    - 'Z' – Tri-state (alta impedância)**
    - '-' – Don't care**
    - 'X' – Conflito (entre '0' e '1')**
    - 'W' – Conflito (entre 'L' e 'H')**
    - 'U' – Não inicializado**
- Outros (a abordar mais tarde)
  - unsigned, signed
  - integer
  - enumerated
  - boolean
  - character
  - time

# Exemplo Simples em VHDL (Mux 2->1)

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.all;
```

Inclusão de bibliotecas

```
entity Mux2_1 is
```

```
  port(sel      : in  std_logic;
```

```
        input0  : in  std_logic;
```

```
        input1  : in  std_logic;
```

```
        muxOut  : out std_logic);
```

```
end Mux2_1;
```

Entidade – definição da interface do módulo

```
architecture Equations of Mux2_1 is
```

```
  signal s_and0Out, s_and1Out : std_logic;
```

Declaração de sinais internos da arquitetura

```
begin
```

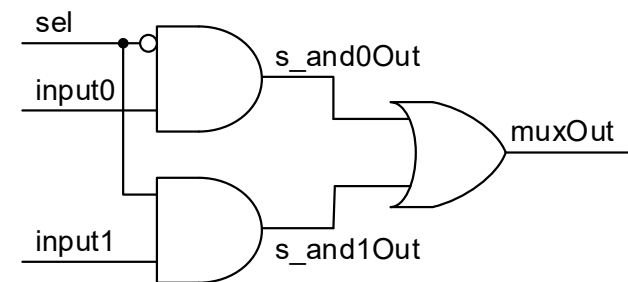
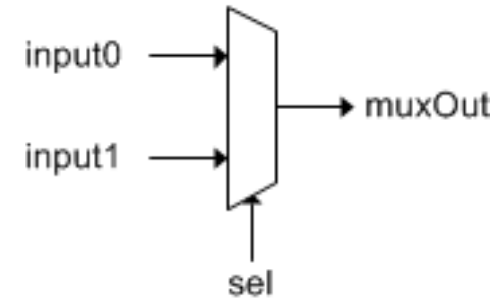
```
  s_and0Out <= not sel and input0;
```

```
  s_and1Out <=      sel and input1;
```

```
  muxOut    <= s_and0Out or s_and1Out;
```

```
end Equations;
```

Arquitetura – definição da implementação do módulo



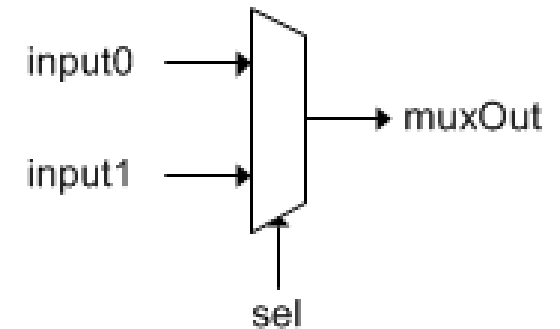
# Arquitetura Alternativa p/o Mux 2->1

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

Inclusão de bibliotecas

```
entity Mux2_1 is  
    port(sel      : in  std_logic;  
          input0   : in  std_logic;  
          input1   : in  std_logic;  
          muxOut    : out std_logic);  
end Mux2_1;
```

Entidade – definição da interface do módulo



```
architecture Behavioral of Mux2_1 is  
begin  
    process(sel, input0, input1)  
    begin  
        if (sel = '0') then  
            muxOut <= input0;  
        else  
            muxOut <= input1;  
        end if;  
    end process;  
end Behavioral;
```

Arquitetura – definição da implementação do módulo

# Mux 2->1 de Barramentos de 8 bits

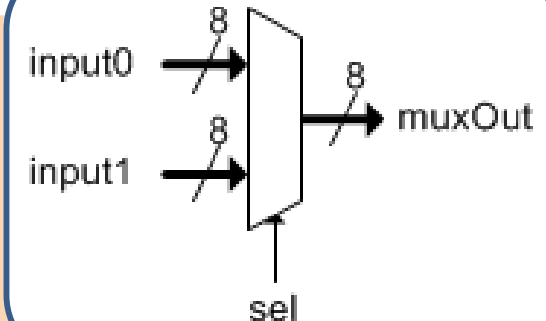
```
library IEEE;  
use IEEE.STD_LOGIC_1164.all;
```

Inclusão de bibliotecas

```
entity Mux2_1 is  
    port(sel      : in  std_logic;  
          input0   : in  std_logic_vector(7 downto 0);  
          input1   : in  std_logic_vector(7 downto 0);  
          muxOut    : out std_logic_vector(7 downto 0));  
end Mux2_1;
```

Entidade – definição da interface do módulo  
(entradas e saídas de dados passam a ser vetores)

```
architecture Behavioral of Mux2_1 is  
begin  
    process(sel, input0, input1)  
    begin  
        if (sel = '0') then  
            muxOut <= input0;  
        else  
            muxOut <= input1;  
        end if;  
    end process;  
end Behavioral;
```



Arquitetura – definição da implementação do módulo  
(a descrição comportamental é a mesma, i.e. o comportamento é semelhante)



# VHDL – comentários

-- comentário de uma linha

/\* comentário em bloco (de várias linhas)

or\_gate: entity work.OR2Gate(Behavioral)

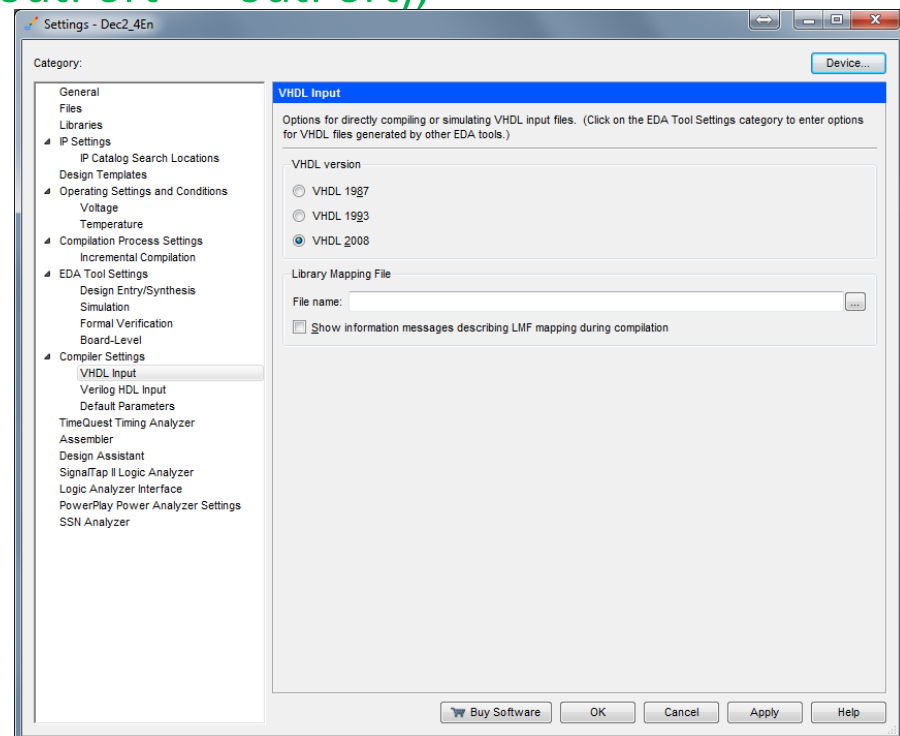
port map (inPort0 => inPort0, inPort1 => inPort1, outPort => s\_orOut);

not\_gate: entity work.NOTGate(Behavioral)

port map (inPort => s\_orOut, outPort => outPort);

\*/

Comentários em bloco só são suportados a partir de VHDL 2008 (pode ser especificado nas opções de síntese – menu “*Assignments→Settings→Compiler Settings→VHDL Input*” no *Quartus Prime*)



# VHDL – identificadores

- VHDL é uma linguagem que não faz distinção entre letras minúsculas e maiúsculas (*not case-sensitive*)
- Identificadores:
  - servem para nomear itens num modelo VHDL
  - podem ser de comprimento arbitrário
  - só podem incluir letras ('A'-'Z', 'a'-'z'), dígitos ('0'-'9'), e '\_'
  - devem começar com uma letra
  - não podem conter '\_' no fim
  - não podem incluir dois '\_' seguidos
  - não podem usar palavras reservadas de VHDL

# VHDL – palavras reservadas (2008)

abs access after alias all and architecture array assert assume assume_guarantee attribute  begin block body buffer bus  case component configuration constant context cover	default disconnect downto  else elsif end entity exit  fairness file for force function  generate generic group guarded  if impure in inertial inout is	label library linkage literal loop  map mod  nand new next nor not null  of on open or others out	package parameter port postponed procedure process property protected pure  range record register reject release rem report restrict restrict_guarantee return rol ror  select sequence severity shared signal	sla sll sra srl strong subtype  then to transport type  unaffected units until use  variable vmode vprop vunit  wait when while with  xnor xor
--	---	--	---	--

Além destas, não usar os nomes “input” e “output”, nem quaisquer outras palavras reservadas em Verilog!

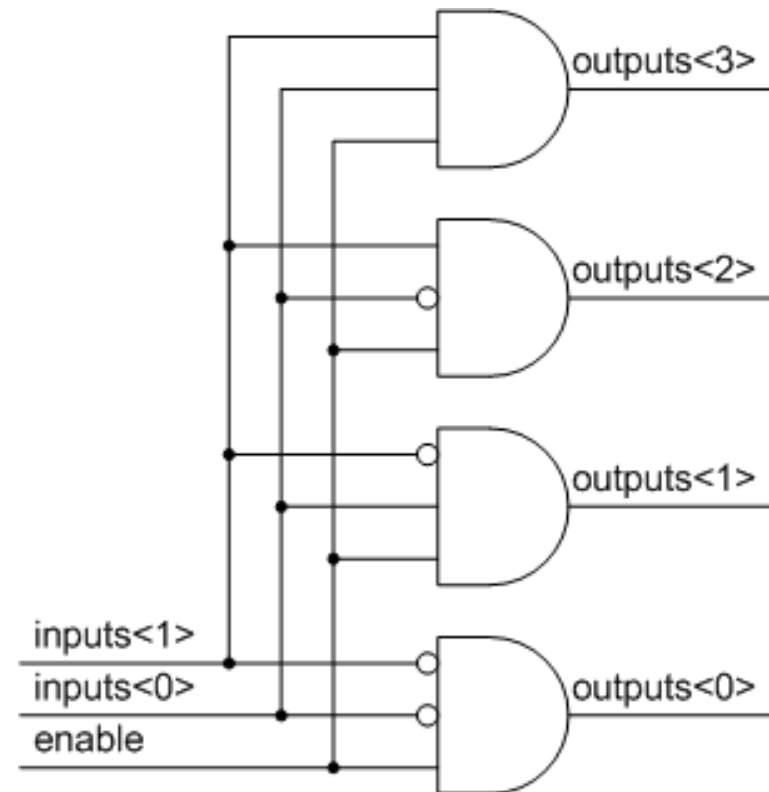
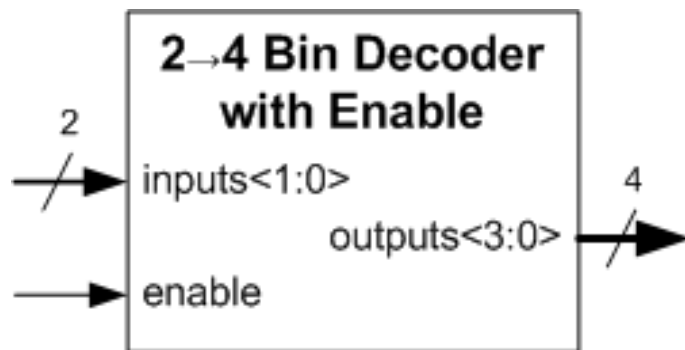


# Componentes Combinatórios

- O que é um componente combinatório?
  - Circuito cujas saídas só dependem do valor das entradas em cada momento
- Exemplos
  - Portas lógicas elementares
  - Descodificadores / codificadores
  - Multiplexadores
  - Comparadores
  - Somadores / subtratores
  - Etc...

# Exemplo de um Componente Combinatório

- Descodificador Binário 2→4 com entrada de ativação (*enable*)



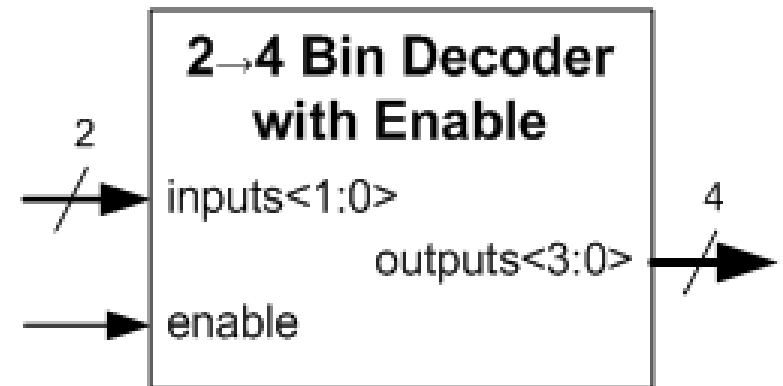
enable	inputs<1>	inputs<0>	outputs<3>	outputs<2>	outputs<1>	outputs<0>
0	-	-	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

# Modelo VHDL do Descodificador 2→4

- *Entity*

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

```
entity Dec2_4En is  
    port(enable : in std_logic;  
          inputs : in std_logic_vector (1 downto 0);  
          outputs : out std_logic_vector (3 downto 0));  
end Dec2_4En;
```



Para que serve a biblioteca  
**IEEE.STD\_LOGIC\_1164?**

# Modelo VHDL do Descodificador 2→4

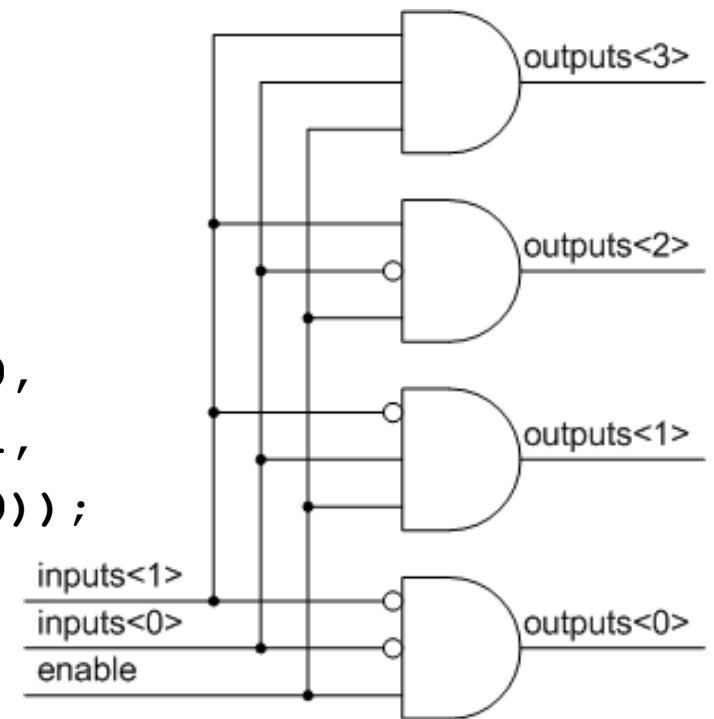
- *Architecture*
  - Diversas abordagens de modelação possíveis em VHDL
    - Instanciar e interligar as portas lógicas necessárias (estrutural)
    - Escrever as equações lógicas para cada saída
    - Descrever o comportamento usando
      - Atribuições condicionais concorrentes
      - Um processo com construções “if...then...else”
      - Um processo com construções “case... when”

Vamos apresentar e analisar as diversas abordagens...

# Arquitetura 1 (Estrutural)

```
architecture Structural of Dec2_4En is
    signal s_nInput0, s_nInput1 : std_logic;
begin
    s_nInput0 <= not inputs(0);
    s_nInput1 <= not inputs(1);
    and_0 : entity work.AND3(ArchName)
        port map(input0 => enable,
                  input1 => s_nInput0,
                  input2 => s_nInput1,
                  andout => outputs(0));

    ...
    and_3 : entity work.AND3(ArchName)
        port map(input0 => enable,
                  input1 => inputs(0),
                  input2 => inputs(1),
                  andout => outputs(3));
end Structural;
```





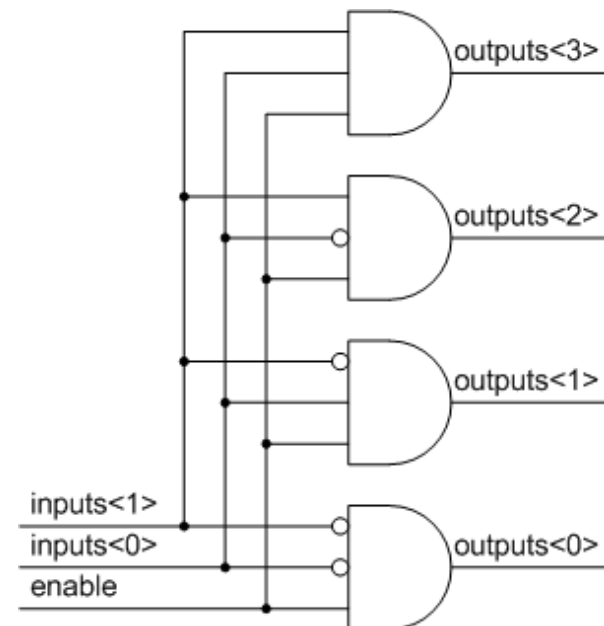
# Arquitetura 2 (Equações Lógicas)

architecture BehavEquations of Dec2\_4En is  
begin

```
    outputs(0) <= enable and (not inputs(1)) and (not inputs(0));  
    outputs(1) <= enable and (not inputs(1)) and (    inputs(0));  
    outputs(2) <= enable and (    inputs(1)) and (not inputs(0));  
    outputs(3) <= enable and (    inputs(1)) and (    inputs(0));
```

end BehavEquations;

Estas atribuições são concorrentes.  
O que significa isto?  
Porquê?



# Arquitetura 3

## (Atribuições Condicionais)

```
architecture BehavAssign of Dec2_4En is  
begin
```

```
    outputs <= "0000" when (enable = '0') else  
                "0001" when (inputs = "00") else  
                "0010" when (inputs = "01") else  
                "0100" when (inputs = "10") else  
                "1000";
```

```
end BehavAssign;
```

1 bit (std\_logic)

String de bits  
(std\_logic\_vector)

No “*Quartus Prime*” este tipo de atribuições condicionais só pode ser usado diretamente no corpo da arquitetura (fora de processos)!!!

enable	inputs<1>	inputs<0>	outputs<3>	outputs<2>	outputs<1>	outputs<0>
0	-	-	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0



# Arquitetura 4

## (processo com if...then...else)

```
architecture BehavProc1 of Dec2_4En is
begin
  process(enable, inputs)
  begin
    if (enable = '0') then
      outputs <= "0000";
    else
      if (inputs = "00") then
        outputs <= "0001";
      elsif (inputs = "01") then
        outputs <= "0010";
      elsif (inputs = "10") then
        outputs <= "0100";
      else
        outputs <= "1000";
      end if;
    end if;
  end process;
end BehavProc1;
```

**(enable, inputs) é a lista de sensibilidade do processo**

O que significa?

Lista dos sinais/portos dos quais depende o processo (i.e. lista dos sinais/portos cujos eventos afetam os valores calculados pelo processo)

enable	inputs<1>	inputs<0>	outputs<3>	outputs<2>	outputs<1>	outputs<0>
0	-	-	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0



# Arquitetura 5

## (processo com case...when)

```
architecture BehavProc2 of Dec2_4En is
begin
    process(enable, inputs)
    begin
        if (enable = '0') then
            outputs <= "0000";
        else
            case inputs is
            when "00" =>
                outputs <= "0001";
            when "01" =>
                outputs <= "0010";
            when "10" =>
                outputs <= "0100";
            when others =>
                outputs <= "1000";
            end case;
        end if;
    end process;
end BehavProc2;
```

Porquê usar **others** em vez de "11" neste caso?

enable	inputs<1>	inputs<0>	outputs<3>	outputs<2>	outputs<1>	outputs<0>
0	-	-	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0



# Aspetos Fundamentais

- Corpo da arquitetura pode conter:
  - Instanciação e interligação de componentes
  - Atribuições concorrentes (funções lógicas)
  - Atribuições concorrentes condicionais
  - Processos
- Num processo relativo a um componente combinatório
  - A lista de sensibilidade deve incluir todos os sinais aos quais o processo é sensível (entradas que influenciam as saídas)
  - As saídas devem estar especificadas para todas as combinações das entradas
    - Todos os “if ... then” (ou “elsif”) devem possuir um “else” (ou “elsif”) – ou seja, deve existir um “else” para cada “if ... then”

# Comentários Finais

- No final desta aula e dos trabalhos práticos 1 e 2 de LSD, deverá ser capaz de:
  - Compreender e utilizar as construções mais elementares da linguagem VHDL
  - Criar projetos simples baseados em VHDL e diagramas esquemáticos, sintetizá-los, implementá-los e testá-los em FPGA
  - Descrever qualquer componente combinatório em VHDL usando a abordagem mais adequada
    - Selecionar os sinais que devem ser incluídos na lista de sensibilidade de um processo
- ... bons trabalhos práticos 1 e 2, disponíveis no site da UC ☺
  - [elearning.ua.pt](http://elearning.ua.pt)