



UNIVERSIDADE FEDERAL DE SANTA CATARINA

Introdução à Linguagem VHDL

EEL5105 – Circuitos e Técnicas Digitais

Objetivos

- Primeiros passos em **VHDL**
- Estudar **exemplos básicos de descrição de hardware** em VHDL
- Implementar circuitos usando **VHDL** no **Emulador Web/Quartus II/DE2**

Introdução à Linguagem VHDL

Tarefas no Emulador

Testando no DE2

Tarefa Adicional

Introdução

- **VHDL - Visão Geral**
 - **VHDL** é uma linguagem para descrição de hardware.
 - **VHDL** = **V**HSIC **H**ardware **D**escription **L**anguage.
 - No final da década de 80, **VHDL** se tornou uma linguagem padrão para o **IEEE** (*Institute of Electrical and Electronic Engineers*).
 - Existem diversas ferramentas para simular e sintetizar (gerar hardware) circuitos descritos em **VHDL**.
 - Outras linguagens de descrição de hardware: Verilog, SystemC, AHDL, Handel-C, System Verilog, Abel, ...

Introdução

- **VHDL - Visão Geral**

- Permite descrever um circuito digital de diferentes formas (ex.: **estrutural**, **comportamental**, **fluxo de dados**).
- Descrições em **VHDL** podem então ser utilizadas para gerar **hardware** (configuração de um FPGA ou projeto de um circuito integrado, por exemplo).
- Descrições em **VHDL** podem ser **simuladas**, permitindo eliminar problemas antes da **síntese do hardware**.
- A geração de estímulos para simulação **VHDL** é comumente realizada por intermédio de **testbenches**, onde são definidos estímulos a serem aplicados ao circuito, dentre outras coisas.

Introdução

- VHDL - Visão Geral
 - **Anatomia** de um projeto simples em **VHDL**:

exemplo.vhd

Libraries and Packages

Imports necessários, tipicamente
IEEE e etc.

Entity

Principalmente a declaração de
ports (entradas e saídas).

Architecture

Descrição do **comportamento** ou
funcionalidade do sistema.

VHDL é *case insensitive*!

Introdução

- VHDL – Exemplo de código: **Majority Detector**
 - **Majority Detector**: saída em nível lógico alto sempre que a maioria dos bits de entrada estiver em nível lógico alto

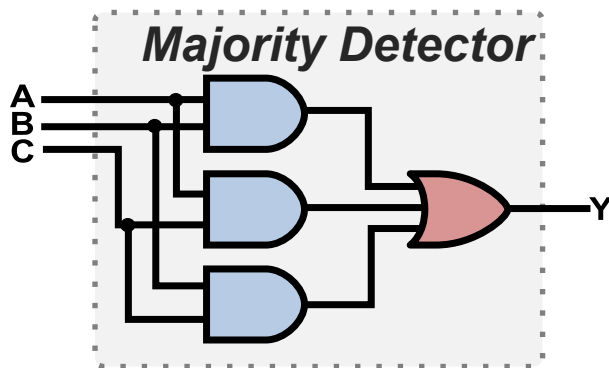


Tabela verdade

A B C	Y
0 0 0	0
0 0 1	0
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	1
1 1 0	1
1 1 1	1

$$Y = (A \text{ and } B) \text{ or } (A \text{ and } C) \text{ or } (B \text{ and } C)$$

$$Y = (A \cdot B) + (A \cdot C) + (B \cdot C)$$

Introdução

- VHDL – *Majority Detector*

LIBRARIES

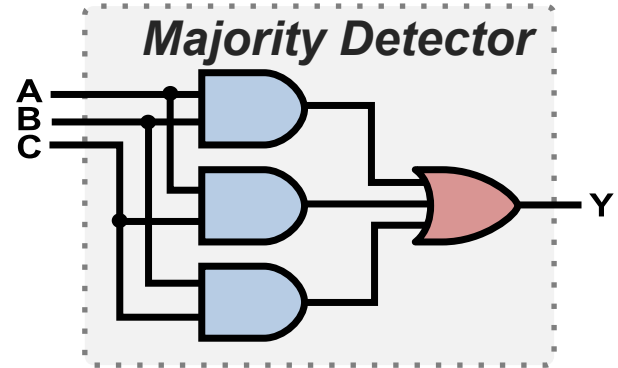
```
library IEEE;  
use IEEE.Std_Logic_1164.all;
```

ENTITY

```
entity majority is  
  port (A: in std_logic;  
        B: in std_logic;  
        C: in std_logic;  
        Y: out std_logic  
        );  
end majority;
```

ARCHITECTURE

```
architecture circuito_logico of majority is  
  signal D,E,F: std_logic;  
begin  
  Y <= D or E or F;  
  D <= A and B;  
  E <= A and C;  
  F <= B and C;  
end circuito_logico;
```



Introdução

- VHDL – *Majority Detector*

LIBRARIES

```
library IEEE;  
use IEEE.Std_Logic_1164.all;
```

ENTITY

```
entity majority is  
port (A: in std_logic;  
      B: in std_logic;  
      C: in std_logic;  
      Y: out std_logic  
      );  
end majority;
```

ARCHITECTURE

```
architecture circuito_logico of majority is  
  signal D,E,F: std_logic;  
begin  
  Y <= D or E or F;  
  D <= A and B;  
  E <= A and C;  
  F <= B and C;  
end circuito_logico;
```

majority.vhd

Arquivo deve ter mesmo nome da **entity**.

Architecture deve estar relacionada com a **entity**.

VHDL é *case insensitive*!

Introdução

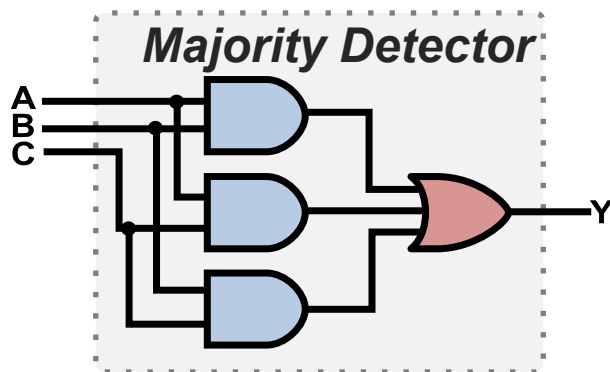
- VHDL – *Majority Detector*
 - **LIBRARIES** : bibliotecas necessárias.

```
library IEEE;  
use IEEE.Std_Logic_1164.all;
```

- Basicamente, essa biblioteca define os tipos **std_logic** e **std_logic_vector**, os quais são versões aperfeiçoadas dos tipos nativos **bit** e **bit_vector** do VHDL.
- **std_logic**: '0' ou '1' (com aspas simples).
Curiosidade: pode também assumir valores como 'U' (*uninitialized*), 'X' (*unknown*), 'Z' (*high impedance*), 'W' (*weak signal*), 'L', 'H', e '-'.
(com aspas duplas).
- **std_logic_vector**: "001010" ou "011" ou "01110" etc...
(com aspas duplas).

Introdução

- VHDL – *Majority Detector*
 - **ENTITY**: define as **ports** do circuito digital, ou seja, a **interface** entre a lógica implementada e o mundo externo.



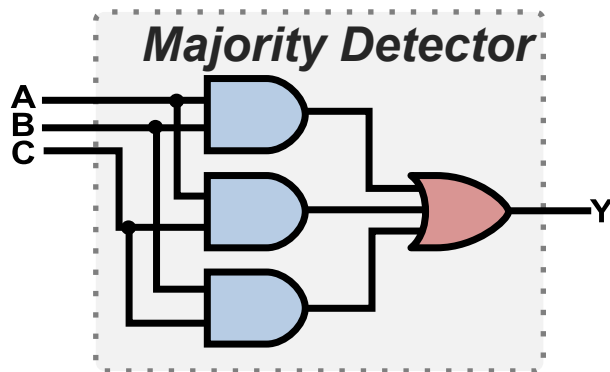
```
entity majority is
port (A: in std_logic;
      B: in std_logic;
      C: in std_logic;
      Y: out std_logic );
end majority;
```

Poderia ser também:

```
entity majority is
port (A, B, C: in std_logic;
      Y: out std_logic );
end majority;
```

Introdução

- VHDL – *Majority Detector*
 - **ENTITY**: define as **ports** do circuito digital, ou seja, a **interface** entre a lógica implementada e o mundo externo.



```
entity majority_is  
port (A: in std_logic;  
      B: in std_logic;  
      C: in std_logic;  
      Y: out std_logic);  
end majority;
```

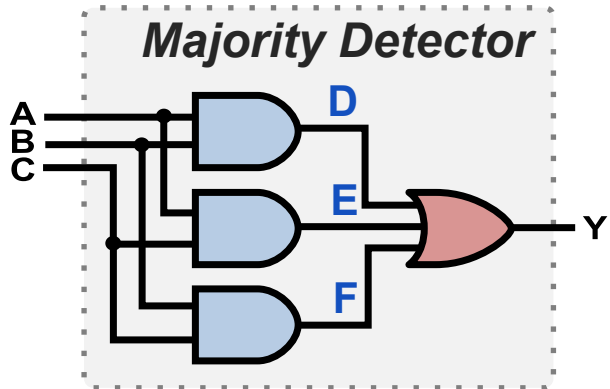
nome

direção

tipo

Introdução

- VHDL – *Majority Detector*
 - **ARCHITECTURE** : define a funcionalidade do circuito digital, utilizando as **ports** listadas na **ENTITY**, além de **signals** para fazer as conexões internas.



```
architecture circuito of majority is
    signal D,E,F: std_logic;
begin
    Y <= D or E or F;
    D <= A and B;
    E <= A and C;
    F <= B and C;
end circuito;
```

Introdução

- VHDL – *Majority Detector*
- **ARCHITECTURE**

```
architecture circuito of majority is  
→ signal D,E,F: std_logic;  
begin  
    Y <= D or E or F;  
    D <= A and B;  
    E <= A and C;  
    F <= B and C;  
end circuito;
```

Declaração de **signals**.

Operador de **atribuição**.

Operadores lógicos: not, and, nand, or, nor, xor e xnor.

Pouco importa a sequência das atribuições aqui, pois elas são **concorrentes**.

Introdução

- VHDL - Exemplo com `std_logic_vector`:

```
library IEEE;
use IEEE.Std_Logic_1164.all;
entity silly is
port (A: in std_logic_vector(7 downto 0);
      Y: out std_logic_vector(7 downto 0)
);
end silly;
architecture myarch of silly is
  signal AUX: std_logic_vector(3 downto 0);
begin
  Y <= A(7 downto 4) & AUX;
  AUX <= not A(3 downto 0);
end myarch;
```



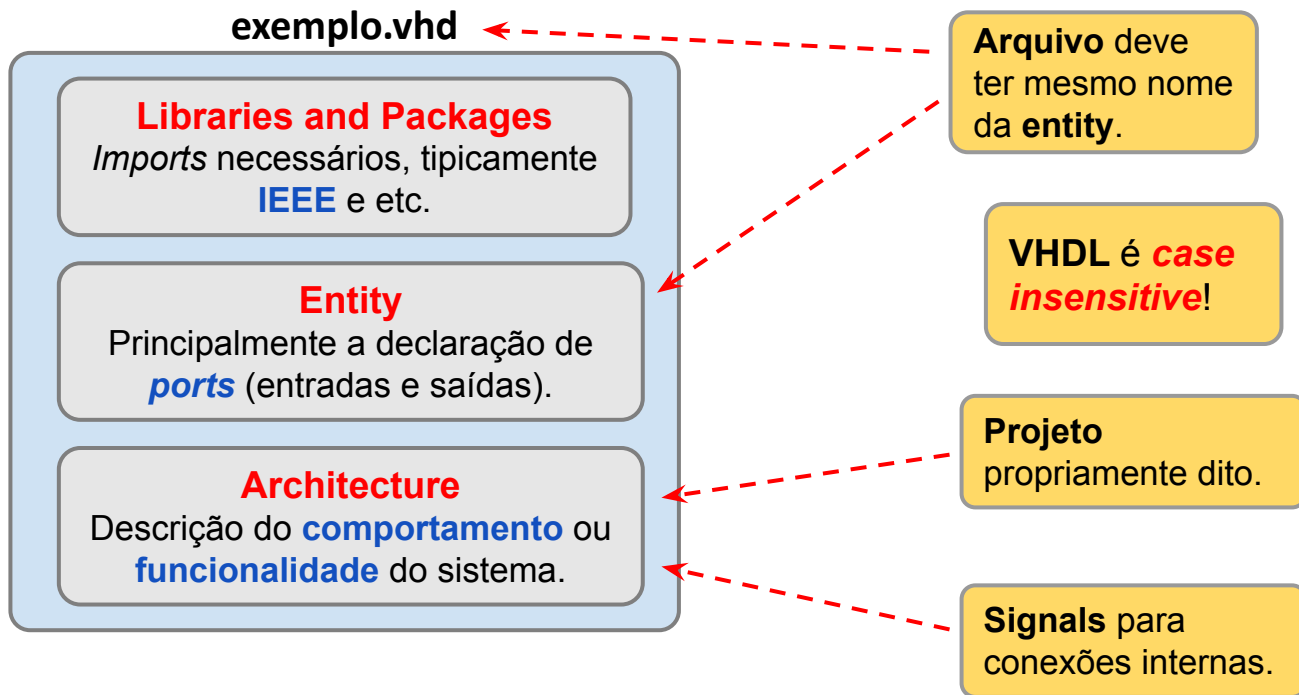
Pergunta que você deve ser capaz de responder:
o que faz esse circuito para A = 11110000?

Operador de **concatenação**.

Pouco importa a sequência das atribuições aqui, pois elas são **concorrentes**.

Introdução

- Em resumo:



Introdução à Linguagem VHDL

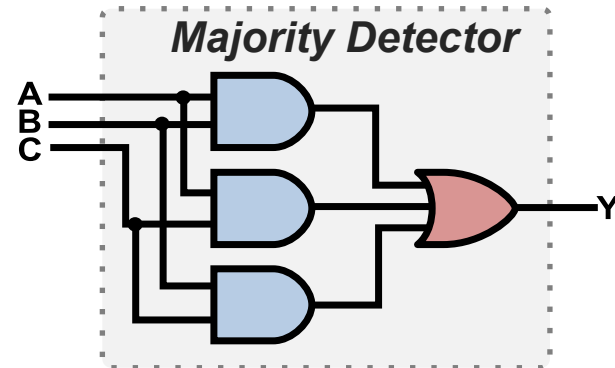
Tarefas no Emulador

Testando no DE2

Tarefa Adicional

Tarefas no Emulador

- **Tarefa 1:** Implementar o **Majority Detector** no **Emulador Web**.
 - Implementar o **majority detector** no arquivo **majority.vhd**, usando o código dado no início da aula.
 - **Atenção:** se o nome da **entity/arquivo** não for **usertop/usertop.vhd**, é preciso aplicar **Set Top Level** ao arquivo utilizado.
 - Além disso, **ports** precisam ser mapeadas para **chaves** e **leds** do **hardware** ou do **emulador**.



Tarefas no Emulador

- Tarefa 1: Implementar o **Majority Detector** no **Emulador Web**
- Alternativas para mapear **ports** no **emulador**:

Usar somente
uma das duas!

- Usar o mapeador de portas (**Mapper**) do emulador. - - - - - ➔
- Renomear as **ports** para:

```
SW: in std_logic_vector(17 downto 0);  
LEDR: out std_logic_vector(17 downto 0)
```

(mapeamento será feito para chaves **SW** e leds **LEDR**).

Dica: mudados os nomes das ports, você pode declarar

A, **B**, **C** e **Y** (entradas e saída
do **majority detector**) como
signals, e fazer: - - - - - ➔

```
A <= SW(0);  
B <= SW(1);  
C <= SW(2);  
LEDR(0) <= Y;
```

majority.vhd

Port Mapping

Only useful for emulation of the top level entity.

☐ Enable Port Mapping

A std_logic => SW ()

B std_logic => SW ()

C std_logic => SW ()

Y std_logic => LEDR ()

SAVE

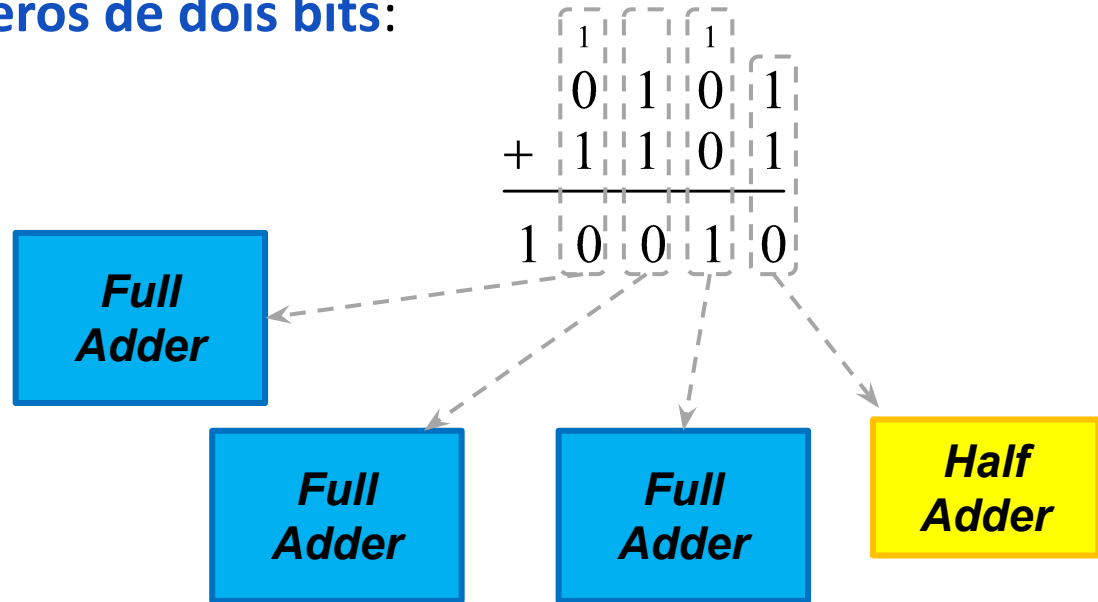
Tarefas no Emulador

- **Tarefa 2:** Implementar o código do exemplo “*silly*” para observar seu funcionamento no **Emulador**.
 - Novamente, se o nome da **entity/arquivo** não for **usertop/usertop.vhd**, é preciso aplicar **Set Top Level** ao arquivo utilizado.
 - **Ports** precisam ser mapeadas como na Tarefa 1.



Tarefas no Emulador

- Nas próximas tarefas, serão implementados um meio somador **half adder** (HA), um somador completo ou **full adder** (FA), e um somador de números de dois bits:

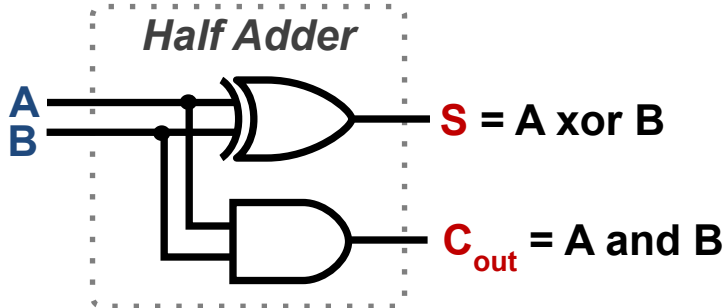


Tarefas no Emulador

- **Tarefa 3:** Implementar um meio somador ou *half adder* (HA)

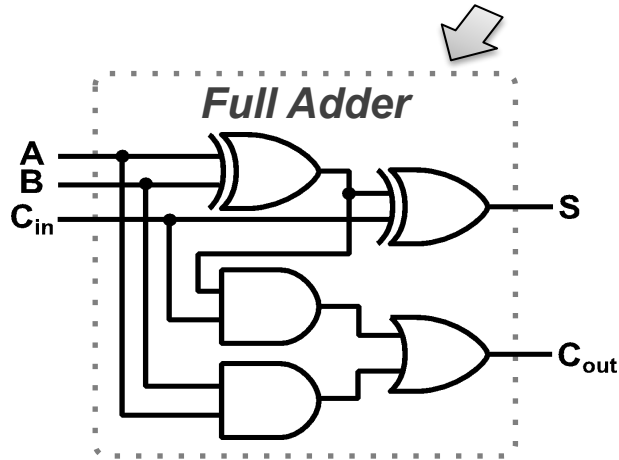
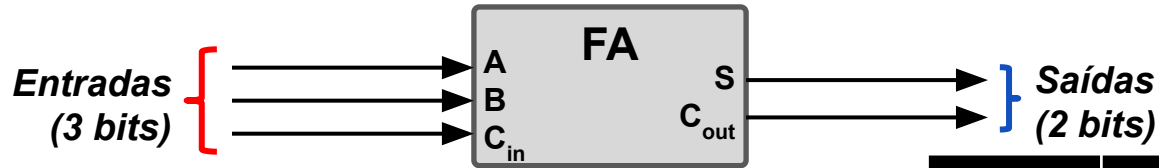


A	B	S	C _{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Tarefas no Emulador

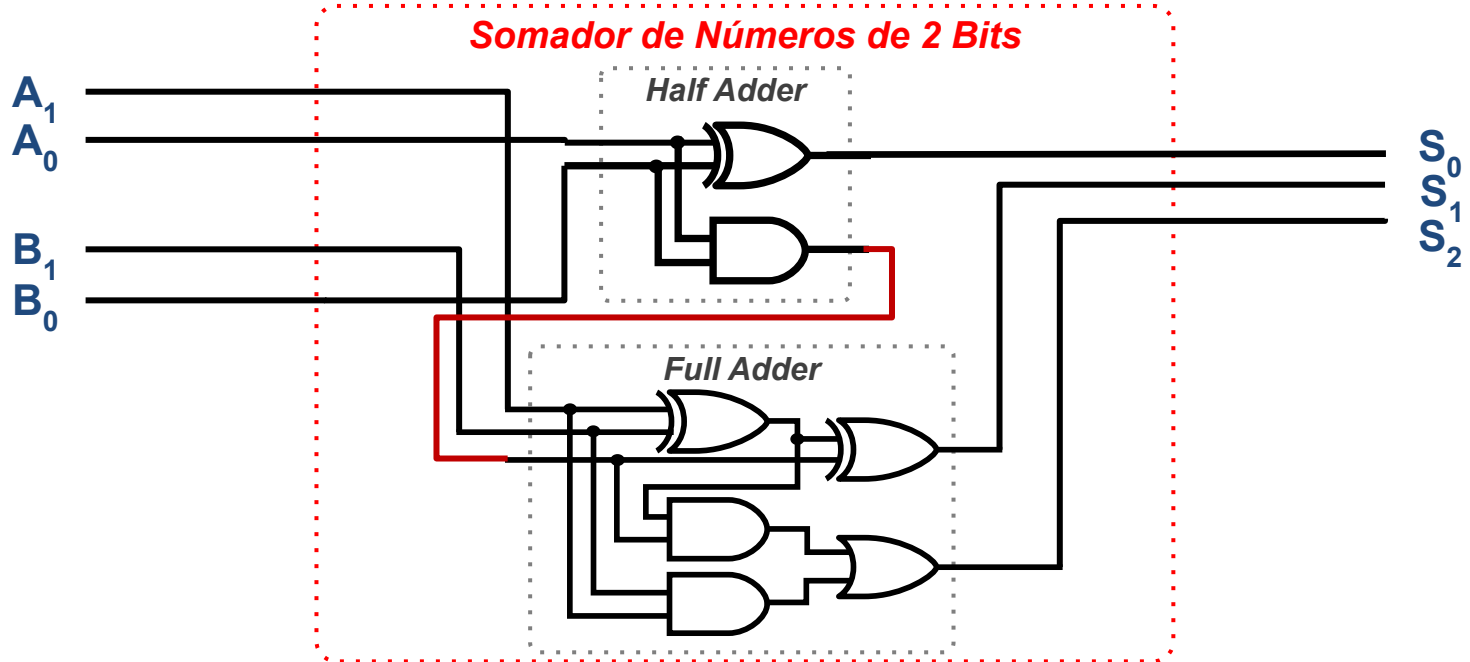
- **Tarefa 4:** Implementar um somador completo ou *full adder* (FA).
 - **Observação:** **signals** podem ser necessários para conexões internas



A B C_{in}	S	C_{out}
0 0 0	0	0
0 0 1	1	0
0 1 0	1	0
0 1 1	0	1
1 0 0	1	0
1 0 1	0	1
1 1 0	0	1
1 1 1	1	1

Tarefas no Emulador

- **Tarefa 5:** Faça a implementação de um somador de números de 2 bits, obtido associando um *half adder* com um *full adder*:



Introdução à Linguagem VHDL

Tarefas no Emulador

Testando no DE2

Tarefa Adicional

Testando no DE2

- Seguindo os passos do **Laboratório 1**, vamos colocar agora os projetos anteriores para funcionar no **DE2** usando o **Quartus II**.
- Para tal:
 - Configure o arquivo desejado como **Top Level** no **Emulador Web**.
 - Exporte o projeto para o **DE2/Quartus II**.
 - Abra o projeto exportado no **Quartus II**:
 - **File > Open project...**
 - **Processing > Start Compilation...**
 - Faça a gravação no **DE2**.
 - Ver próximos slides...

Testando no DE2



- **Gravação no DE2**

- A seguir, com o kit **DE2** ligado à tomada e ao computador, acesse **Tools > Programmer** no **Quartus II** para gravar seu projeto.



- Com o **Programmer** aberto:

1) Clique em  Auto Detect .

Caso **Auto Detect** esteja desativado ( Auto Detect), clique em  Hardware Setup... e selecione **USB Blaster** como hardware a ser utilizado.

2) Clique em  Start .

3) Pronto, agora é só testar o seu projeto no **DE2**.

Introdução à Linguagem VHDL

Tarefas no Emulador

Testando no DE2

Tarefa Adicional

A horizontal dotted line consisting of 30 small grey dots, extending across the width of the slide.

Tarefa Adicional

- **Tarefa 6:** Universalidade das operações **NAND** e **NOR**.
 - Obtenha a implementação equivalente somente com portas **NAND** do **Majority Detector** e faça a implementação de tal circuito no **Emulador/DE1-SoC/Quartus II**.
 - Realize o mesmo procedimento, agora com a representação equivalente com portas **NOR**.

- **Atenção:** para implementar 3 entradas em VHDL,

```
Saida <= not (Ent1 and Ent2 and Ent3)
```

De forma similar, para **NOR** é preciso:

```
Saida <= not (Ent1 or Ent2 or Ent3)
```

