



Aula 02 – Linguagem de hardware e síntese de circuitos



Tópicos da aula

- **Introdução ao VHDL**
- **Definição de Entidade e Arquitetura**
- **Operadores Lógicos e Aritméticos**
- **Tipos de dados**



O que é VHDL?

- ❏ **É uma linguagem de descrição de hardware**
- ❏ **Desenvolvida a partir da necessidade do Departamento de Defesa (DoD) dos EUA para unificar a documentação de projetos de seus fornecedores no contexto do programa VHSIC, substituindo os diagrama esquemáticos**
- ❏ **VHDL = VHSIC + HDL**
 - ❏ VHSIC = **V**ery High Speed Integrated Circuit
 - ❏ HDL = **H**ardware **D**escription **L**anguage






“O VHDL” ou “a VHDL”?

- ❏ Assim como toda e qualquer linguagem de programação, em português, costuma-se “masculinizar” o gênero da linguagem
 - ❏ o C
 - ❏ o Delphi
 - ❏ o JAVA
 - ❏ o assembly
 - ❏ o VHDL






Cronologia e uso do VHDL

Marcos históricos

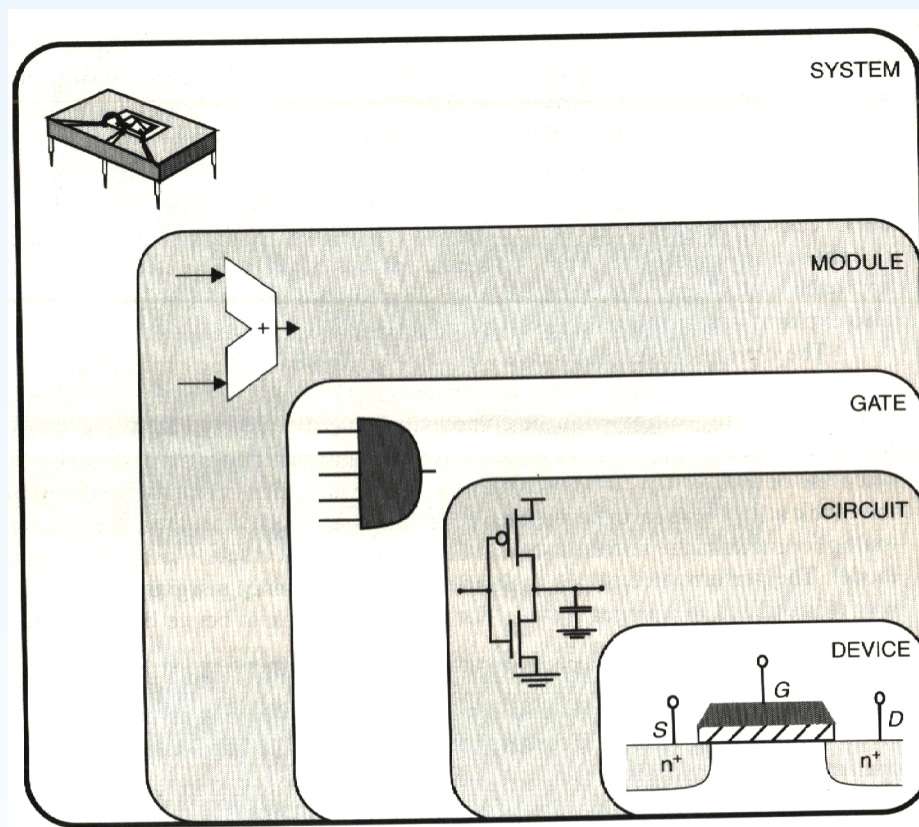
-  1980: criação da linguagem
-  1987: padronização pelo IEEE (IEEE Std 1076-1987)
-  1993: revisão do padrão

Uso

-  Documentação
-  Simulação
-  Síntese

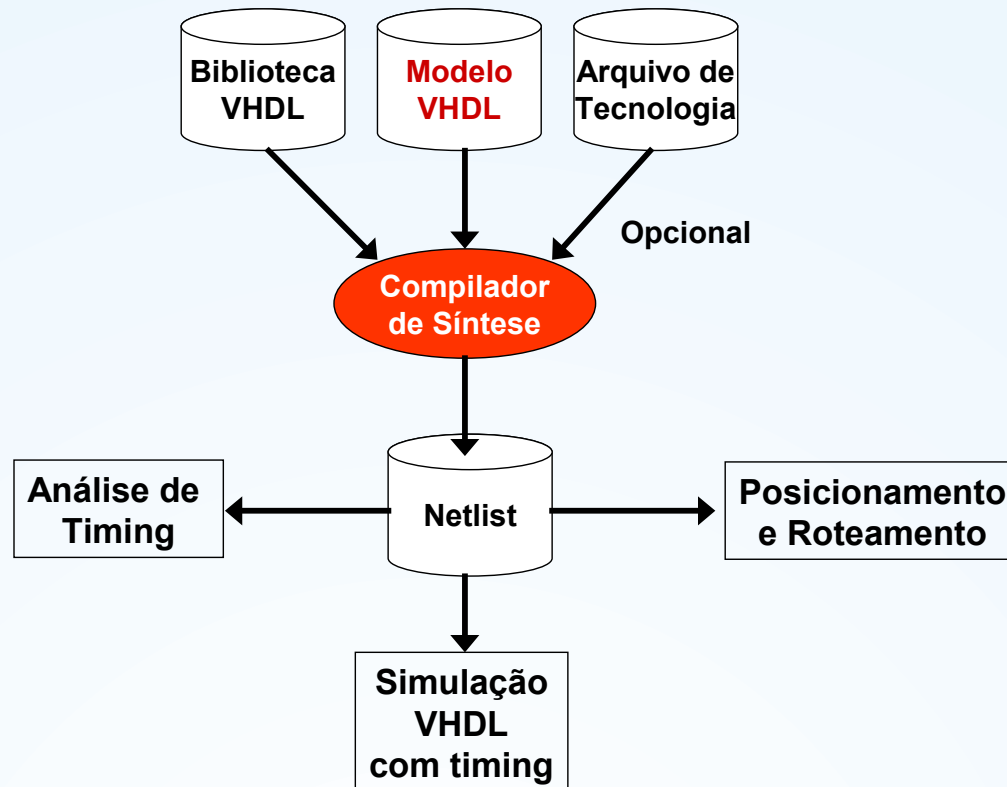


Níveis de abstração em circuitos digitais





Síntese de um modelo VHDL





Universidade Federal
de Santa Catarina

Linguagem de descrição de hardware

Aula II: 8 de 22

Ambiente de desenvolvimento (Síntese)

Quartus II 64-Bit - C:/Projetos_Altera/Linguagem_Hardware/aulas/aula_02/and2 - and2

File Edit View Project Assignments Processing Tools Window Help

Search altera.com

Project Navigator

Entity

Cyclone IV GX: AUTO

PORTA_AND

Hierarchy Files Design Uni

Tasks

Flow: Compilation Customize...

Task

- Compile Design
- Analysis & Synthesis
 - Edit Settings
 - View Report
- Analysis & Elaboration
- Partition Merge

Entity Declaration:

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4 use IEEE.STD_LOGIC_UNSIGNED.ALL;
5 use IEEE.STD_LOGIC_ARITH.ALL;
6
7 ENTITY PORTA_AND IS
8
9     PORT (
10         A      : IN STD_LOGIC;
11         B      : IN STD_LOGIC;
12         C      : OUT STD_LOGIC
13     );
14 END PORTA_AND;
15
16
17 ARCHITECTURE behavioral OF PORTA_AND IS
18
19 BEGIN
20
21     C <= A and B;
22
23 END behavioral;
24
25
26
27
```

Messages

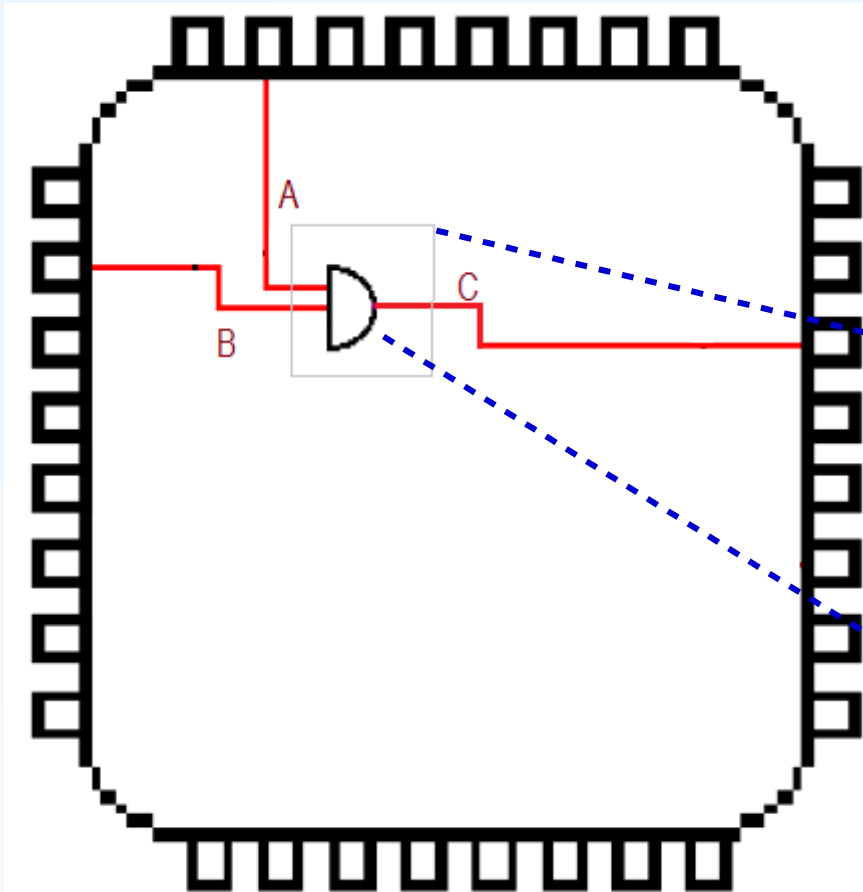
Type	ID	Message
Information	204019	Generated file and2_vhd.sdo in folder "C:/Projetos_Altera/Linguagem_Hardware/aulas/aula_02/simulation/modelsim/" for EDA simulation tool
Information		Quartus II 64-Bit EDA Netlist Writer was successful. 0 errors, 0 warnings
Information	293000	Quartus II Full Compilation was successful. 0 errors, 12 warnings

System (1) / Processing (123) /

100% 00:00:26



ENTIDADE e corpo (ARQUITETURA) do componente



```
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.numeric_std.all;  
use IEEE.STD_LOGIC_UNSIGNED.ALL;  
use IEEE.STD_LOGIC_ARITH.ALL;  
  
ENTITY and2 IS  
    PORT (  
        A      : IN STD_LOGIC;  
        B      : IN STD_LOGIC;  
        C      : OUT STD_LOGIC  
    );  
END and2;  
  
ARCHITECTURE behavioral OF and2 IS  
  
BEGIN  
    C <= A and B;  
  
END behavioral;
```



Aspectos gerais do VHDL

- ❏ **Dois conjuntos de construtores**
 - ❏ Simulação
 - ❏ Síntese
- ❏ **Características**
 - ❏ É baseado em palavras reservadas (**BEGIN**, **END**, ...)
 - ❏ É insensível à caixa (BEGIN = Begin = begin)
 - ❏ Declarações terminadas por ponto e vírgula “;”
 - ❏ Comentários marcados precedidos por duplo hífen “--”



Exercícios

- ❏ Implementar uma porta lógica OR 2x1
- ❏ Implementar uma porta lógica AND 4x1



Sinais INTERNOS e EXTERNOS

- ❏ **EXTERNOS:** são sinais de entrada e saída do componente
- ❏ **INTERNOS:** são sinais usados para conectar os pinos de dois ou mais componentes dentro de um mesmo FPGA, ou conectar sinais internos de um componente

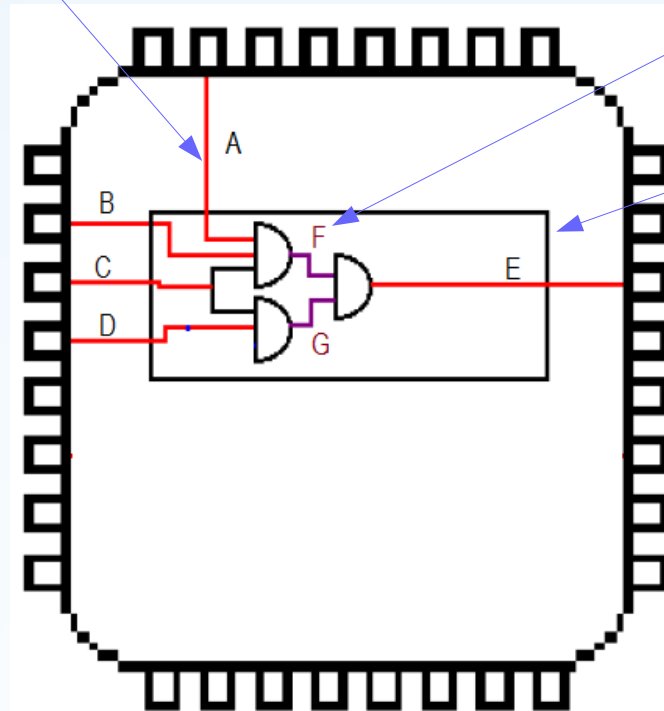


Sinais INTERNOS e EXTERNOS

Sinais **externos** do componente (A, B, C, D, E)

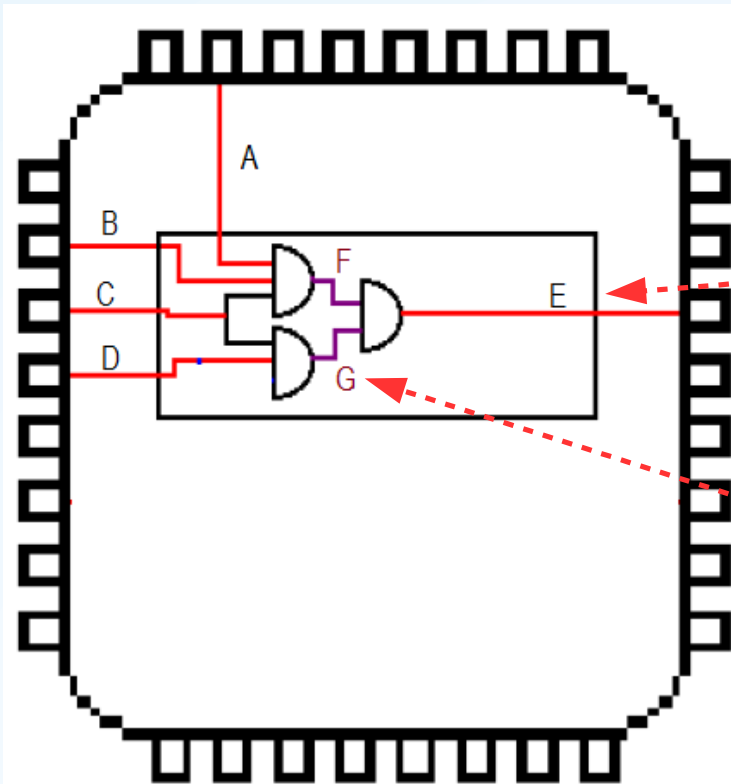
Sinais **internos** do componente (F, G)

Componente





Exemplo: declaração de sinais



```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

ENTITY COMPONENTE IS

    PORT (
        A      : IN STD_LOGIC;
        B      : IN STD_LOGIC;
        C      : IN STD_LOGIC;
        D      : IN STD_LOGIC;
        E      : OUT STD_LOGIC
    );
END COMPONENTE;

ARCHITECTURE behavioral OF COMPONENTE IS

    signal F : STD_LOGIC;
    signal G : STD_LOGIC;

BEGIN

    F <= A and B and C;
    G <= C and D;
    E <= F and G;

END behavioral;
```



Classes de Objetos

 **CONSTANT**

 **VARIABLE**

 **SIGNAL**



Exemplo: Classes de Objetos

```
ARCHITECTURE behavioral OF COMPONENTE IS

    signal F : STD_LOGIC;
    signal G : STD_LOGIC;

    constant K : integer := 7;

BEGIN
```




Tipos Escalares

Tipos definidos	Valor	Exemplos
STD_LOGIC	1, 0, Z	
BOOLEAN	Verdadeiro, falso	TRUE, FALSE
INTEGER	$(-2^{31} - 1) \leq x \leq (2^{31} - 1)$	123
NATURAL	$0 \leq x \leq (2^{31} - 1)$	
REAL	$-3,65^{47} \leq x \leq 3,65^{47}$	
TIME	Pico, nano, micro, mili, etc.	1 us



Universidade Federal
de Santa Catarina

Tipos Compostos

Tipos definidos	Valor	Exemplos
STD_LOGIC_VECTOR	1, 0, Z	"011010010100"
STRING	Tipo caracter	"texto qualquer"



Universidade Federal
de Santa Catarina

Exercícios

- ❏ Implementar uma porta lógica OR 4x1 usando `STD_LOGIC_VECTOR`
- ❏ Implementar uma porta lógica AND 8x1 usando `STD_LOGIC_VECTOR`



Definição de novos tipos

- A linguagem permite a criação de novos tipos
- Aplicações:
 - facilitar a leitura do código: estados de uma máquina
 - definir novos tipos físicos: resistência, capacitância etc.
 - novos tipos compostos: definição de memórias

- Declaração: palavra reservada **TYPE**

- Exemplo:

```
TYPE estado IS (parado, inicio, caso_1, caso_2, caso_3);  
SIGNAL abc : estado := parado;
```

- é definido um novo tipo **estado**

```
TYPE estado IS (parado, inicio, caso_1, caso_2, caso_3);
```

- declaração de um sinal do tipo **estado**

```
SIGNAL abc : estado := parado;
```

- valores possíveis para o sinal **abc**: **parado**, **inicio**, **caso_1**, **caso_2**, **caso_3**



Operadores

- **Divididos em classes:**
 - as classes definem a precedência dos operadores
 - operadores de uma mesma classe: igual precedência
- **Maior precedência:** classe diversos
- **Menor precedência:** classe lógicos
- **Operador not:** operador lógico; está na classe diversos devido à precedência

classe	operadores
lógicos	and or nand nor xor xnor
relacionais	= /= < <= > >=
deslocamento	sll srl sla sra rol ror
adição	+ - &
sinal	+ -
multiplicação	* / mod rem
diversos	** abs not



FIM AULA II