



Circuitos Digitais II - 6882

Paulo Roberto de Oliveira

Universidade Estadual de Maringá
Departamento de Informática

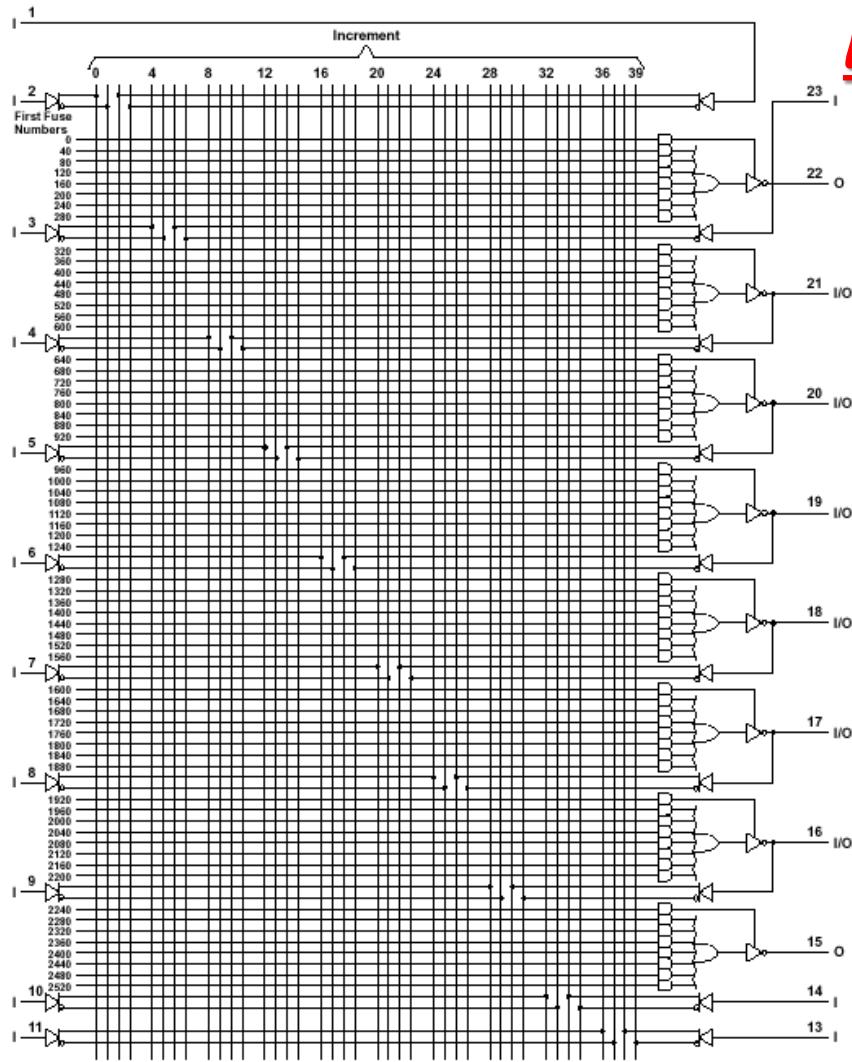
Bacharelado em Ciência da Computação

Aula de Hoje

- Revisão da aula anterior
- Características de Projeto em VHDL

Revisão

HDL - Linguagem de Descrição de Hardware



Motivação

Circuitos mais complexos:

- Difíceis de descrever com equações lógicas
- Difíceis de descrever com diagramas de portas lógicas

HDL - Linguagem de Descrição de Hardware

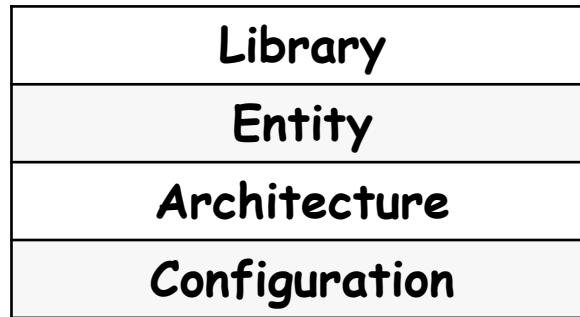
O que significa VHDL?

- ✓ Very High Speed Integrated Circuits
- ✓ Hardware
- ✓ Description
- ✓ Language

Linguagem de descrição de hardware
com ênfase em
circuitos integrados de altíssima velocidade.

HDL - Linguagem de Descrição de Hardware

Estrutura Básica de um Código em VHDL



- Library (Biblioteca): constantes, pacotes;
- Entity (Entidade): pinos de entrada e saída;
- Architecture (Arquitetura): implementações do projeto;
- Configuration (Configuração): define as arquiteturas que serão utilizadas.

HDL - Linguagem de Descrição de Hardware

Estrutura Básica de um Código em VHDL

```
LIBRARY IEEE;  
USE IEEE.STD_LOGIC_1164.all;  
USE IEEE.STD_LOGIC_UNSIGNED.all;
```

LIBRARY (PACOTES)

```
ENTITY exemplo IS  
PORT (  
    <descrição dos pinos de I/O>  
);  
END exemplo;
```

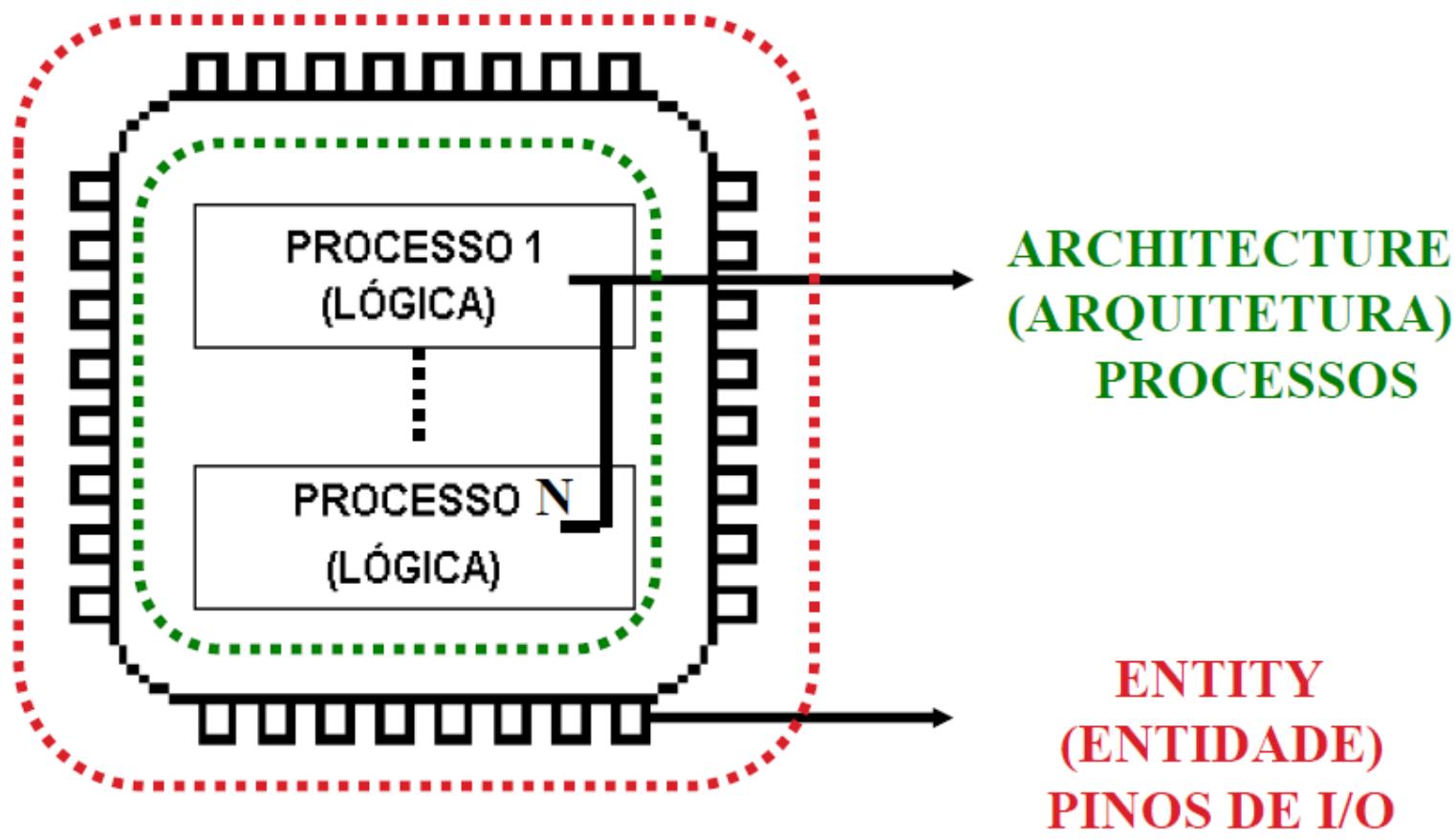
ENTITY (PINOS DE I/O)

```
ARCHITECTURE teste OF exemplo IS  
BEGIN  
    ...  
END teste;
```

ARCHITECTURE
(ARQUITETURA)

HDL - Linguagem de Descrição de Hardware

Estrutura Básica de um Código em VHDL



Aula de Hoje

- Características de Projeto

VHDL - Características de Projeto

Principais características

- Utilização de linguagens de descrição de hardware:
 - Abordagem top-down → projeto com início em níveis mais altos e posteriormente subdividido em blocos menores → consideração pela descrição comportamental do sistema;
 - Abordagem bottom-up → projeto iniciado por blocos mais simples e posteriormente construídos blocos mais complexos → consideração pela descrição estrutural do sistema.

VHDL - Características de Projeto

Principais características

- Reusabilidade → comunicação entre os diferentes componentes especificados, ou seja, criação de componentes para um determinado projeto pode ser reutilizado em outros projetos → Obtenção com utilização de pacotes e bibliotecas.
- Criação de novos pacotes e bibliotecas, além das bibliotecas já existentes pré-compiladas.
- Qualquer código VHDL simulado é associado a uma biblioteca especial ou de trabalho, na qual estão todos os componentes utilizados no projeto.

VHDL - Características de Projeto

Principais características

Palavras reservadas:

abs	body	entity	inertial
access	buffer	exit	inout
after	bus	file	is
alias	case	for	label
all	component	function	library
and	configuration	generate	linkage
architecture	constant	generic	literal
array	disconnect	group	loop
assert	downto	guarded	map
attribute	else	if	mod
begin	elseif	impure	nand
block	end	in	new

VHDL - Características de Projeto

Principais características

Palavras reservadas: (continuação)

next	postponed	ror	transport
nor	procedure	select	type
not	process	severity	unaffected
null	pure	shared	units
of	range	signal	until
on	record	sla	use
open	register	sll	variable
or	reject	sra	wait
others	rem	srl	when
out	report	subtype	while
package	return	then	with
port	rol	to	xor
			xnor

VHDL - Características de Projeto

Principais características

Alguns símbolos reservados:

<u>Símbolo</u>	<u>Significado</u>
+	adição
-	subtração
/	divisão
=	igualdade
<	menor do que
>	maior do que
>=	maior do que ou igual a
<=	menor do que ou igual a
<=	Associação de valor para sinais (recebe)

VHDL - Características de Projeto

Principais características

Alguns símbolos reservados: (continuação)

<u>Símbolo</u>	<u>Significado</u>
**	exponenciação
/=	desigualdade
&	concatenador
:	separação entre objeto e o tipo
:=	Associação de valor para variáveis (recebe)
;	terminador
--	comentário
=>	Seta indicando “então”
=>	Atribuição de valores únicos em vetores ou de vários valores em vetores junto com a cláusula “others” (recebe)

VHDL - Características de Projeto

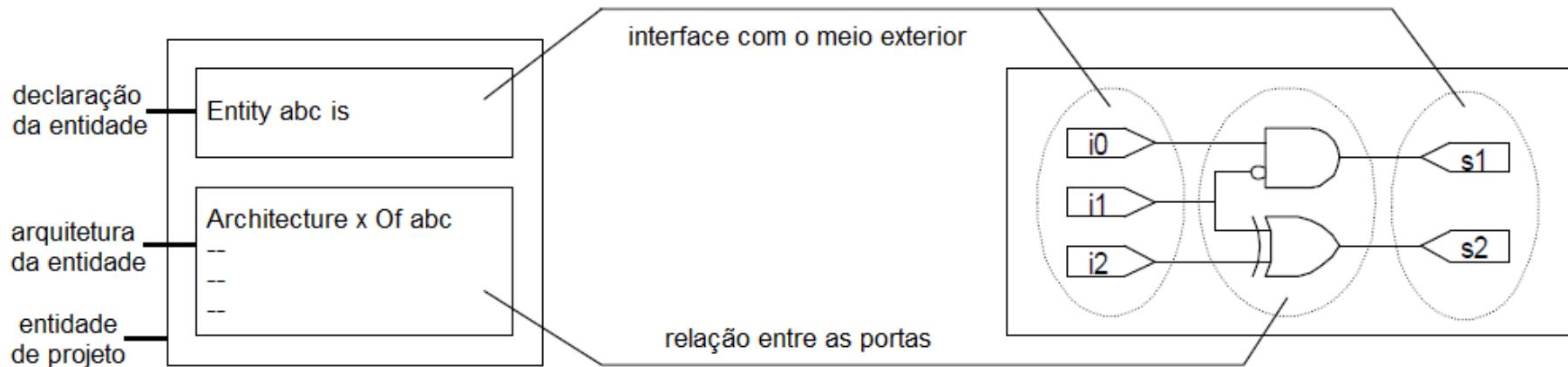
Estrutura ou entidade de projeto

- Representação desde uma simples porta lógica a um sistema completo.
- A criação de um projeto é composta de vários componentes.
- Cada componente é dividido em duas partes:
 - Declaração da Entidade (ENTITY) → definição da interface entre o componente e o ambiente exterior, isto é, definição das entradas e saídas do componente, especificadas com a declaração das portas (PORT).
 - Arquitetura da Entidade (ARCHITECTURE) → descrição da funcionalidade do componente, isto é, especificação do comportamento do componente em relação às suas entradas e saídas.

VHDL - Características de Projeto

Estrutura ou entidade de projeto

Exemplo:



VHDL - Características de Projeto

Declaração da entidade

Exemplo 01: Sintaxe da declaração da entidade

```
ENTITY nome IS
    PORT (entrada1, entrada2 : IN tipo_dado;      -- entradas
          saida : OUT tipo_dado);   -- saida
END nome;
```

- A declaração da entidade pode ser considerada uma interface do componente para o ambiente externo → somente as portas são visíveis a outros componentes.
- A declaração da entidade é composta por um nome identificador e por uma declaração de portas.

VHDL - Características de Projeto

Declaração da entidade

Exemplo 01:

- Nomes das entidades, subrotinas, variáveis, constantes e sinais devem usar:
 - apenas caracteres alfanuméricos e o caractere underline (_);
 - o primeiro caractere deve ser uma letra;
 - o último caractere não pode ser um caractere underline (_);
 - não são permitidos dois caracteres underline (_) consecutivos;
 - não há caso sensítivo → não há distinção entre letras maiúsculas e minúsculas.

Nota: nome, Nome e NOME são um mesmo identificador

VHDL - Características de Projeto

Declaração da entidade

Exemplo 01:

- Identificadores legais e ilegais em VHDL

NOMES	
rs_clk	✓
_rs_clk	✗
ab08B	✓
Sinal#1	✗
A_1023	✗
rs_clk_	✗
A_1023	✓
Circ_Dig2	✓
_Circ	✗
2Circ_Dig	✗

VHDL - Características de Projeto

Declaração da entidade

Exemplo 01: Modos de operação das portas → se o objeto será de leitura, escrita ou ambas.

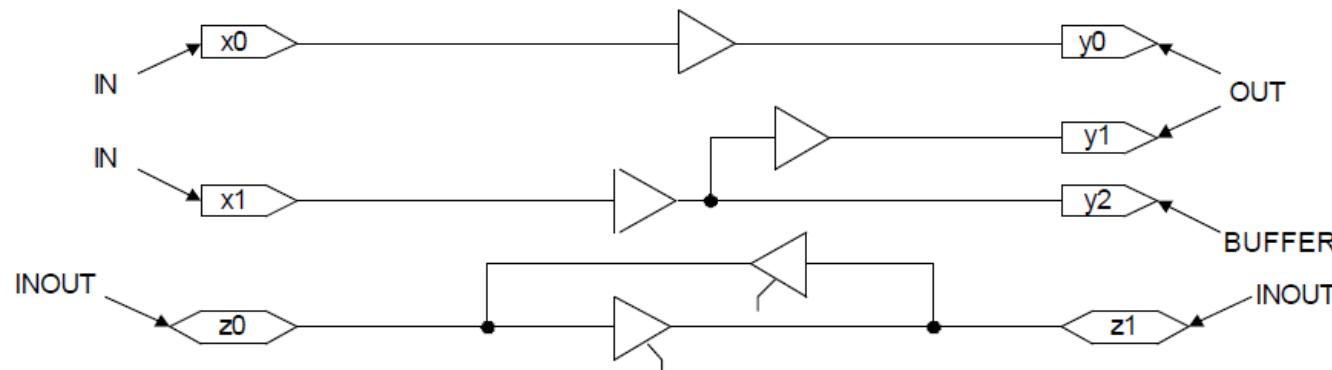
- Quatro modos:
 - IN → portas que operam exclusivamente como entradas. Por exemplo, os objetos 'entrada1' e 'entrada2' são entradas.
 - OUT → portas que operam exclusivamente como saídas. Por exemplo, o objeto 'saída' é uma saída.
 - BUFFER → porta de saída (similar a OUT) com possibilidade de realimentação interna.
 - INOUT → porta pode ser de entrada e saída ao mesmo tempo. Por exemplo, um barramento de dados.

VHDL - Características de Projeto

Declaração da entidade

Exemplo 02:

```
ENTITY entidade_abc IS
    PORT (x0, x1      : IN          tipo_a;      -- entradas
          y0, y1      : OUT         tipo_b;      -- saídas
          y2          : BUFFER     tipo_c;      -- saída
          z0, z1      : INOUT      tipo_d);     -- entrada / saída
END entidade_abc;
```



VHDL - Características de Projeto

Declaração da entidade

Exemplo 03: Sintaxe

```
ENTITY nome IS
    GENERIC (parametro1 : tipo_dado := valor);
    PORT (entrada1, entrada2 : IN tipo_dado;
          saida : OUT tipo_dado);
END nome;
```

- 'tipo_dado' deve ser um dos tipos propostos pela linguagem ou definidos pelo usuário. Por exemplo, BIT, INTEGER e TIME.
- Definição de constantes genéricas para o componente → empregadas no controle da operação: largura de banda, frequência de operação ou atraso de propagação → uso da palavra reservada GENERIC antes da declaração das portas.

VHDL - Características de Projeto

Declaração da entidade

Exemplo 04:

```
ENTITY entidade_abc IS
    GENERIC (n      : INTEGER := 5);
    PORT    (x0, x1 : IN      tipo_a;      -- entradas
              y0, y1 : OUT     tipo_b;      -- saídas
              y2 : BUFFER   tipo_c;      -- saída
              z0, z1 : inout  tipo_d;      -- entrada/saída
END entidade_abc;
```

- A declaração inicia com a palavra reservada “ENTITY” seguida do nome que a identifica.
- A palavra reservada “PORT” é empregada para definir o modo e tipo das portas de entrada e saída da descrição.
- A palavra reservada “GENERIC” permite a passagem de informações estáticas para uma unidade de projeto.

VHDL - Características de Projeto

Arquitetura da entidade

- Sintaxe da arquitetura da entidade:

Exemplo:

```
ARCHITECTURE nome_arch OF nome IS
[declaracoes]
BEGIN
  :
  comandos
  :
END nome_arch;
```

- 'nome_arch' → nome da arquitetura da entidade referindo ao comportamento de uma entidade identificada por 'nome'.
- 'declarações' → sinais, constantes ou componentes utilizados na arquitetura da entidade desenvolvida.
- 'comandos' → descrição da função do componente → especificações e operações que serão realizadas.

VHDL - Características de Projeto

Arquitetura da entidade

```
ARCHITECTURE nome_identificador OF entidade_abc IS
    --
    -- regiao de declaracoes:
    -- declaracoes de sinais e constantes
    -- declaracoes de componentes referenciais
    -- declaracao de corpo de subprogramas
    -- definicao de novos tipos de dados locais
    --
BEGIN
    --
    -- comandos concorrentes
    --
END;
```

VHDL - Características de Projeto

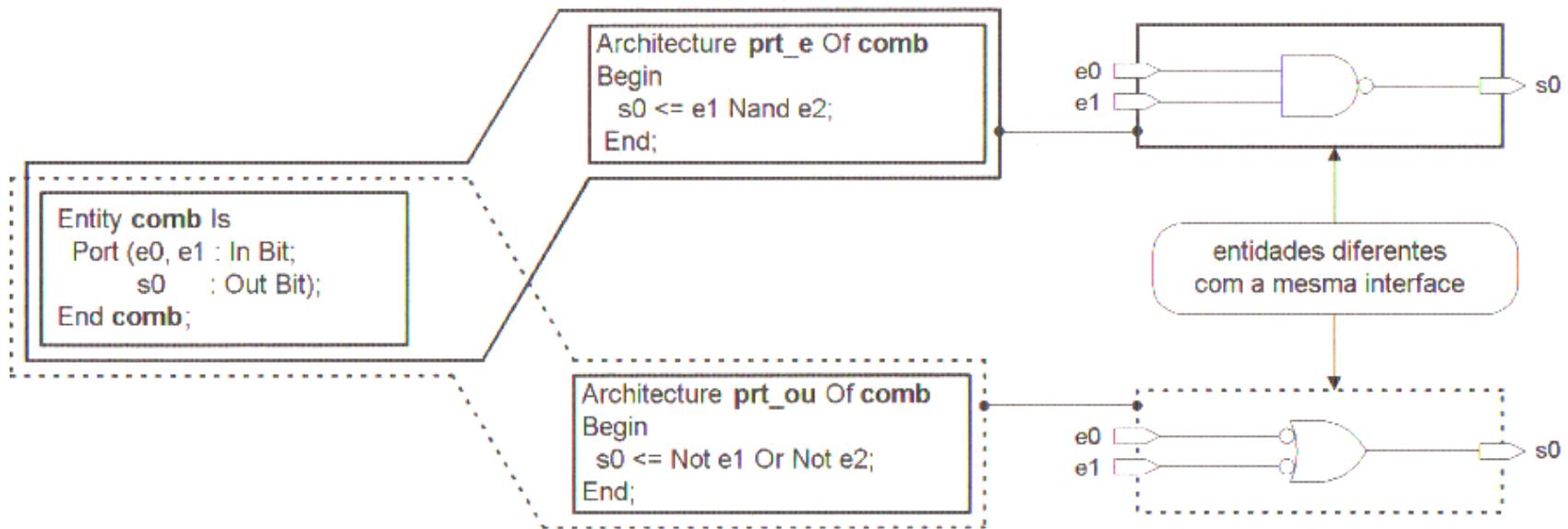
Arquitetura da entidade

- A arquitetura da entidade inicia-se com a palavra reservada "ARCHITECTURE" seguida de um nome identificador e o nome da declaração da entidade associada.
- As linhas subsequentes podem conter:
 - declarações de sinais e constantes;
 - declarações que identificam os componentes externos utilizados pela arquitetura da entidade;
 - descrição completa de subprogramas que são utilizados localmente pela arquitetura da entidade.
- Os comandos concorrentes que descrevem a entidade, propriamente, ficam entre as palavras reservadas "BEGIN" e "END".

VHDL - Características de Projeto

Arquitetura da entidade

- Exemplo de duas arquiteturas da entidade e uma única declaração de entidade:



- Nota: Muitas arquiteturas da entidade podem existir para uma mesma declaração de entidade, mas apenas uma delas pode estar ativa por vez.

VHDL - Características de Projeto

Estrutura de projeto

- Três tipos de modelagem da arquitetura da entidade:
 - Comportamental → descrição da funcionalidade do componente utilizando expressões e linguagem de alto nível.
 - Fluxo de Dados → descrição a nível de transferência entre registradores (RTL ou Register-Transfer Level) que especifica a própria expressão booleana do circuito.
 - Estrutural → descrição do componente com interconexão mais simples, construindo funções lógicas com o uso de elementos mais primitivos como portas lógicas e suas conexões.

VHDL - Características de Projeto

Modelagem comportamental

- Modelo fornece as relações entre as entradas e as saídas.
- Realização da síntese do projeto depende da forma como o código é escrito e da ferramenta utilizada → algumas construções de alto nível não são implementáveis em hardware.
- Geralmente o código em VHDL contém pelo menos um processo (palavra reservada PROCESS) e os comandos de um processo são executados de forma sequencial.
- Exemplo 01: Tabela verdade de uma porta AND

a	b	f
0	0	0
0	1	0
1	0	0
1	1	1

VHDL - Características de Projeto

Modelagem comportamental

- Exemplo 01: Tabela verdade de uma porta AND

➤ Declaração da entidade

```
ENTITY porta_and IS
    PORT (a, b : IN BIT;
          f : OUT BIT);
END porta_and;
```

Arquitetura da entidade

```
ARCHITECTURE logica OF porta_and IS
BEGIN
    PROCESS (a, b)
    BEGIN
        IF (a='1') AND (b='1') THEN
            f <= '1';
        ELSE
            f <= '0';
        END IF;
    END PROCESS;
END logica;
```

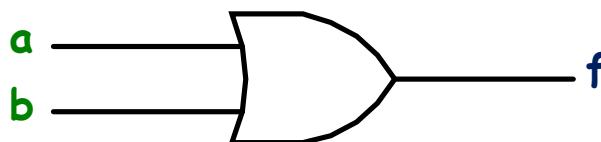
Entradas Saída

a	b	f
0	0	0
0	1	0
1	0	0
1	1	1

VHDL - Características de Projeto

Exemplo 02:

Considere a descrição da porta OR



Entradas Saída

a	b	f
0	0	0
0	1	1
1	0	1
1	1	1

Modelagem comportamental

Declaração
da
Entidade

Arquitetura
da
Entidade

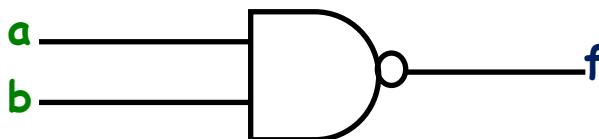
```
ENTITY porta_or IS -- exemplo
  PORT (a, b : IN BIT;
        f : OUT BIT);
END porta_or;

ARCHITECTURE logica OF porta_or IS
BEGIN
  PROCESS (a, b)
  BEGIN
    IF (a='0') AND (b='0') THEN
      f <= '0';
    ELSE
      f <= '1';
    END IF;
  END PROCESS;
END logica;
```

VHDL - Características de Projeto

Exemplo 03:

Considere a descrição da porta NAND



Entradas Saída

a	b	f
0	0	1
0	1	1
1	0	1
1	1	0

Modelagem comportamental

Declaração
da
Entidade

Arquitetura
da
Entidade

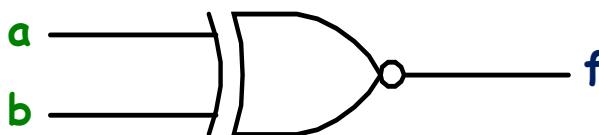
```
ENTITY porta_nand IS -- exemplo
  PORT (a, b : IN BIT;
        f : OUT BIT);
END porta_nand;

ARCHITECTURE logica OF porta_nand IS
BEGIN
  PROCESS (a, b)
  BEGIN
    IF (a='1') AND (b='1') THEN
      f <= '0';
    ELSE
      f <= '1';
    END IF;
  END PROCESS;
END logica;
```

VHDL - Características de Projeto

Exemplo 04:

Considere a descrição da porta XNOR



Entradas Saída

A red bracket labeled 'Entradas' spans under the input lines 'a' and 'b'. A blue bracket labeled 'Saída' spans under the output line 'f'.

a	b	f
0	0	1
0	1	0
1	0	0
1	1	1

Modelagem comportamental

Declaração
da
Entidade

Arquitetura
da
Entidade

```
ENTITY porta_xnor IS -- exemplo
  PORT (a, b : IN BIT;
        f : OUT BIT);
END porta_xnor;

ARCHITECTURE logica OF porta_xnor IS
BEGIN
  PROCESS (a, b)
  BEGIN
    IF (a=b) THEN
      f <= '1';
    ELSE
      f <= '0';
    END IF;
  END PROCESS;
END logica;
```

VHDL - Características de Projeto

Modelagem por fluxo de dados

- Também conhecida como descrição a nível de transferência entre registradores ou RTL (Register-Transfer Level).
- Especifica como os dados fluem de uma entrada ou registrador para um outro registrador ou saída do circuito através de um circuito combinacional.
- É similar à expressão booleana de um circuito.
- Exemplo 01: Tabela verdade de uma porta AND

a	b	f
0	0	0
0	1	0
1	0	0
1	1	1

VHDL - Características de Projeto

Modelagem por fluxo de dados

- Exemplo 01: Tabela verdade de uma porta AND

➤ Declaração da entidade:

```
ENTITY porta_and IS
    PORT (a, b : IN BIT;
          f : OUT BIT);
END porta_and;
```

➤ Arquitetura da entidade:

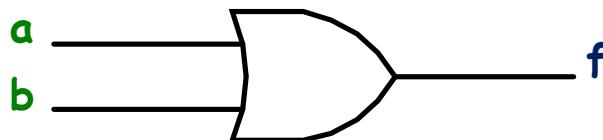
```
ARCHITECTURE logica OF porta_and IS
BEGIN
    f <= a AND b;
END logica;
```

Nota: Descrição completa de um componente → combinação entre declaração da entidade e arquitetura da entidade.

VHDL - Características de Projeto

Modelagem por fluxo de dados

- Exemplo 02: Considere a descrição da porta OR:



Entradas Saída

a	b	f
0	0	0
0	1	1
1	0	1
1	1	1

Declaração da Entidade

--exemplo

```
ENTITY porta_or IS  
  PORT(a, b : IN BIT;  
        f : OUT BIT);  
END porta_or;
```

Arquitetura da Entidade

```
ARCHITECTURE Logica OF porta_or IS  
BEGIN  
  f <= a OR b;  
END Logica;
```

VHDL - Características de Projeto

Modelagem por fluxo de dados

- Exemplo 03: Considere a descrição da porta NAND:



Entradas Saída

a	b	f
0	0	1
0	1	1
1	0	1
1	1	0

Declaração da Entidade

--exemplo

```
ENTITY porta_nand IS  
  PORT(a, b : IN BIT;  
        f : OUT BIT);  
END porta_nand;
```

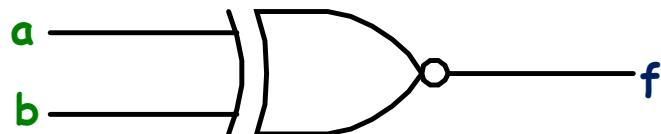
Arquitetura da Entidade

```
ARCHITECTURE Logica OF porta_nand IS  
BEGIN  
  f <= a NAND b;  
END Logica;
```

VHDL - Características de Projeto

Modelagem por fluxo de dados

- Exemplo 04: Considere a descrição da porta XNOR:



Entradas Saída

A red bracket groups the labels 'Entradas' and 'Saída' together, pointing to the first two columns of the truth table below.

a	b	f
0	0	1
0	1	0
1	0	0
1	1	1

Declaração da Entidade

--exemplo

```
ENTITY porta_xnor IS
    PORT(a, b : IN BIT;
          f : OUT BIT);
```

```
END porta_xnor;
```

Arquitetura da Entidade

```
ARCHITECTURE Logica OF porta_xnor IS
BEGIN
    f <= a XNOR b;
END Logica;
```

VHDL - Características de Projeto

Modelagem por fluxo de dados

- Exemplo 05: Implementação de um circuito gerador de paridade:

➤ Expressão booleana → $p = \overline{(i3 \oplus i2) \oplus (i1 \oplus i0)} = (i3 \odot i2) \odot (i1 \odot i0)$

➤ Declaração da entidade e arquitetura da entidade:

```
ENTITY gerador_par IS
    PORT (i3, i2, i1, i0 : IN BIT;
          p : OUT BIT);
END gerador_par;

ARCHITECTURE fluxo_de_dados OF gerador_par IS
BEGIN
    p <= i3 XNOR i2 XNOR i1 XNOR i0;
    -- p <= NOT (i3 XOR i2 XOR i1 XOR i0);
END fluxo_de_dados;
```

VHDL - Características de Projeto

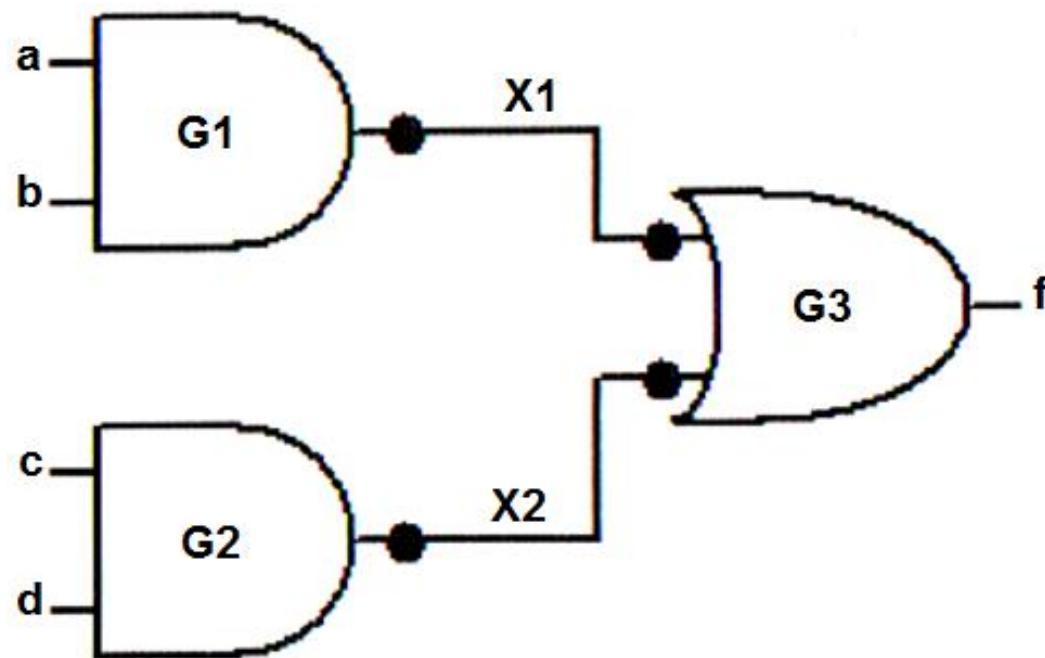
Modelagem estrutural

- Modelo descreve os circuitos sob a forma de diagramas lógicos, listando as portas que compõem o circuito e as suas interconexões.
- Vantagem → garantia de uma representação sintetizável do código, que pode ser traduzido para algum dispositivo físico programável.
- Algumas funcionalidades e características das portas são descritas em nível comportamental ou por fluxo de dados.

VHDL - Características de Projeto

Modelagem estrutural

- Exemplo 01: Implementação de um circuito simples 'Unidade1':



- Nota: Descrição de cada um dos componentes → duas portas AND (G1 e G2) e uma porta OR (G3).

VHDL - Características de Projeto

Modelagem estrutural

- Exemplo 01: Implementação de um circuito simples 'Unidade1':

➤ Declarações das entidades e arquiteturas das entidades:

```
ENTITY and2 IS
```

```
    PORT (x, y : IN BIT;
```

```
        z : OUT BIT);
```

```
END and2;
```

```
ARCHITECTURE logica OF and2 IS
```

```
BEGIN
```

```
    z <= x AND y;
```

```
END logica;
```

```
ENTITY or2 IS
```

```
    PORT (x, y : IN BIT;
```

```
        z : OUT BIT);
```

```
END or2;
```

```
ARCHITECTURE logica OF or2 IS
```

```
BEGIN
```

```
    z <= x OR y;
```

```
END logica;
```

- Maior reusabilidade dos componentes desenvolvidos reduzindo a quantidade de linhas de código escrito e evitando códigos repetidos.

VHDL - Características de Projeto

Modelagem estrutural

- Exemplo 01: Implementação de um circuito simples 'Unidade1':
 - Definição do circuito depende das modelagens comportamental ou por fluxo de dados e estrutural;
 - Declaração da entidade é a mesma do modelo comportamental ou por fluxo de dados com a declaração das portas de entrada e saída;
 - Arquitetura da entidade são declarados os componentes, bem como as suas entradas e saídas;
 - Utilização de sinais (X_1 e X_2) para representar os fios que interligam as portas, bem como guardam os valores dos resultados das duas portas AND;
 - Implementação do circuito com o mapeamento de portas (PORT MAP) → especificação do identificador ou label (G_1 , G_2 e G_3), da função que realizará e de suas entradas e saídas.

VHDL - Características de Projeto

Modelagem estrutural

- Exemplo 01: Implementação de um circuito simples 'Unidade1':

```
ENTITY Unidade1 IS
```

```
    PORT (a, b, c, d : IN BIT;  
          f : OUT BIT);
```

```
END Unidade1;
```

```
ARCHITECTURE estrutural OF Unidade1 IS
```

```
COMPONENT and2
```

```
    PORT (x, y : IN BIT;  
          z : OUT BIT);
```

```
END COMPONENT;
```

```
-- continuacao →
```

```
COMPONENT or2
```

```
          -- continuacao
```

```
    PORT (x, y : IN BIT;  
          z : OUT BIT);
```

```
END COMPONENT;
```

```
SIGNAL X1, X2 : BIT;
```

```
BEGIN
```

```
    G1 : and2 PORT MAP (a,b,X1);
```

```
    G2 : and2 PORT MAP (c,d,X2);
```

```
    G3 : or2 PORT MAP (X1,X2,f);
```

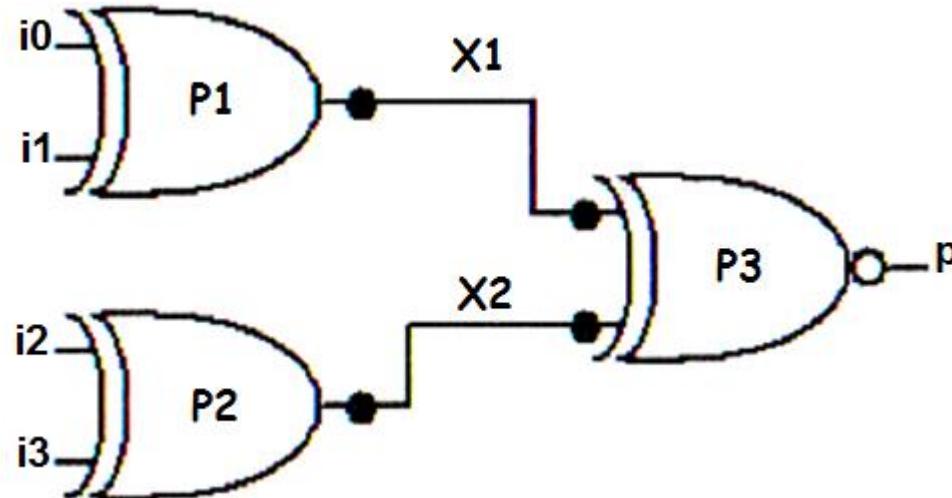
```
END estrutural;
```

VHDL - Características de Projeto

Modelagem estrutural

- Exemplo 02: Implementação de um circuito gerador de paridade

- A paridade indica a quantidade de dígitos '1' em um número binário:
 - ✓ Par \rightarrow número par de dígitos '1' $\therefore p = 1$
 - ✓ Ímpar \rightarrow número ímpar de dígitos '1' $\therefore p = 0$



- Nota: Descrição de cada um dos componentes \rightarrow duas portas XOR (P1 e P2) e uma porta XNOR (P3).

VHDL - Características de Projeto

Modelagem estrutural

- Exemplo 02: Implementação de um circuito gerador de paridade

➤ Declarações das entidades e arquiteturas das entidades:

```
ENTITY xor2 IS
```

```
    PORT (x, y : IN BIT;
```

```
        z : OUT BIT);
```

```
END xor2;
```

```
ARCHITECTURE logica OF xor2 IS
```

```
BEGIN
```

```
    z <= x XOR y;
```

```
END logica;
```

```
ENTITY xnor2 IS
```

```
    PORT (x, y : IN BIT;
```

```
        z : OUT BIT);
```

```
END xnor2;
```

```
ARCHITECTURE logica OF xnor2 IS
```

```
BEGIN
```

```
    z <= x XNOR y;
```

```
END logica;
```

- Maior reusabilidade dos componentes desenvolvidos reduzindo a quantidade de linhas de código escrito e evitando códigos repetidos.

VHDL - Características de Projeto

Modelagem estrutural

- Exemplo 02: Implementação de um circuito gerador de paridade:
 - Definição do circuito depende das modelagens comportamental ou por fluxo de dados e estrutural;
 - Declaração da entidade é a mesma do modelo comportamental ou por fluxo de dados com a declaração das portas de entrada e saída;
 - Arquitetura da entidade são declarados os componentes, bem como as suas entradas e saídas;
 - Utilização de sinais (X_1 e X_2) para representar os fios que interligam as portas, bem como guardam os valores dos resultados das duas portas XOR;
 - Implementação do circuito com o mapeamento de portas (PORT MAP) → especificação do identificador ou label (P_1 , P_2 e P_3), da função que realizará e de suas entradas e saídas.

VHDL - Características de Projeto

Modelagem estrutural

- Exemplo 02: Implementação de um circuito gerador de paridade:

```
ENTITY gerador_par IS
    PORT (i3, i2, i1, i0 : IN BIT;
          p : OUT BIT);
END gerador_par;

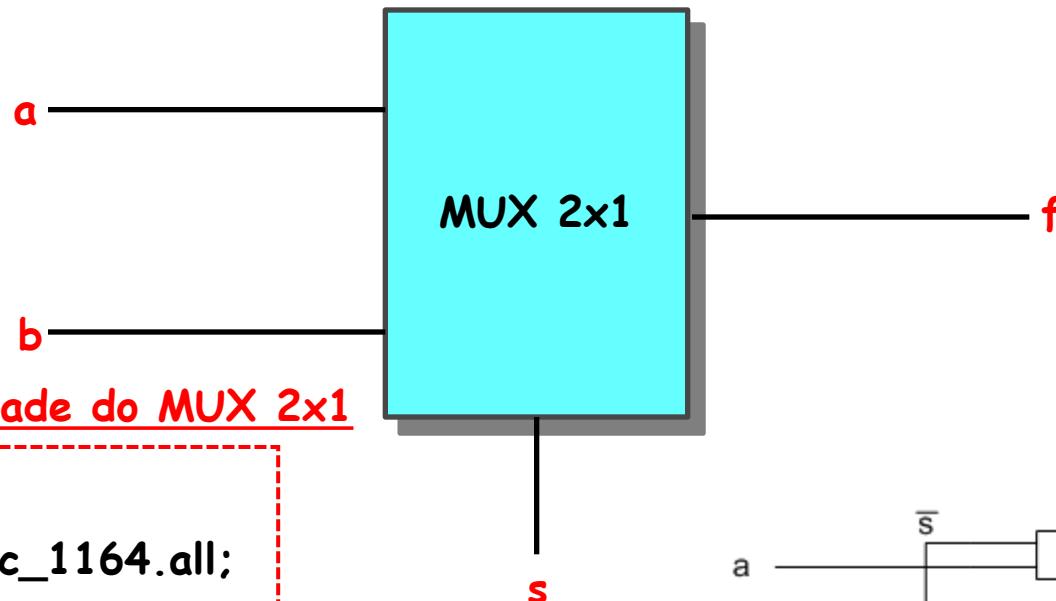
ARCHITECTURE estrutural OF gerador_par IS
COMPONENT xor2
    PORT (x, y : IN BIT;
          z : OUT BIT);
END COMPONENT;
-- continuacao →
```

```
COMPONENT xnor2           -- continuacao
    PORT (x, y : IN BIT;
          z : OUT BIT);
END COMPONENT;

SIGNAL X1, X2 : BIT;
BEGIN
    P1 : xor2 PORT MAP (i0,i1,X1);
    P2 : xor2 PORT MAP (i2,i3,X2);
    P3 : xnor2 PORT MAP (X1,X2,p);
END estrutural;
```

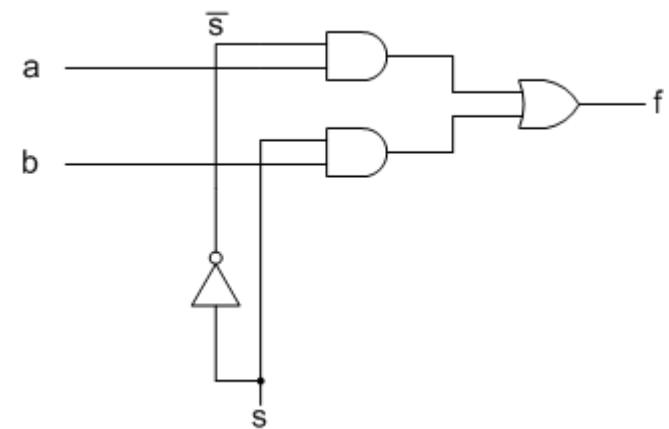
VHDL - Características de Projeto

Diferentes formas de descrever a arquitetura da entidade de um circuito



Declaração de Entidade do MUX 2x1

```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
ENTITY multiplexador IS  
    PORT (a, b : IN BIT;  
          s : IN BIT;  
          f : OUT BIT);  
END multiplexador;
```



VHDL - Características de Projeto

Descrição ou modelagem comportamental

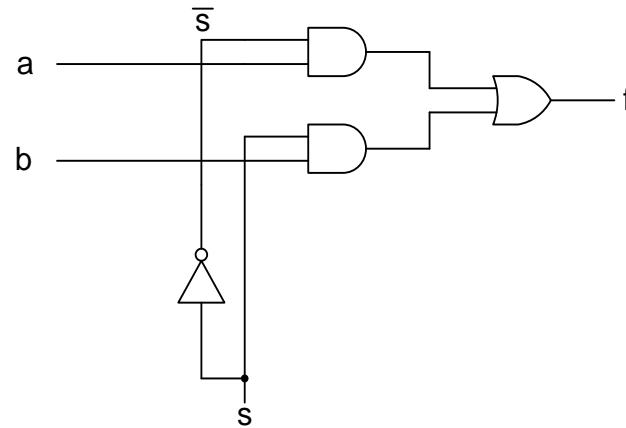
```
ARCHITECTURE comportamental OF multiplexador IS
BEGIN
    PROCESS (a, b, s)
        BEGIN
            IF s='0' THEN
                f<=a;
            ELSE
                f<=b;
            END IF;
        END PROCESS;
    END comportamental;
```

PROCESS:

- Todos os comandos em um processo (PROCESS) são executados de forma sequencial.
- O processo é ativado assim que um dos componentes da lista de sensibilidade é alterado.
- A lista de sensibilidade corresponde aos sinais que devem alterar a saída do circuito e é composta de todos os sinais de entrada (neste exemplo → a, b, s) para os circuitos combinacionais.

VHDL - Características de Projeto

Descrição ou modelagem por fluxo de dados (RTL)



```
ARCHITECTURE fluxo_de_dados OF multiplexador IS
BEGIN
    f <= ((NOT s ) AND a) OR (s AND b);
END fluxo_de_dados;
```

A descrição RTL especifica a própria expressão booleana do circuito

VHDL - Características de Projeto

Descrição ou modelagem estrutural

- Solução 01:

```
ARCHITECTURE estrutural OF multiplexador IS
```

```
SIGNAL t0, t1, t2 : BIT;
```

```
BEGIN
```

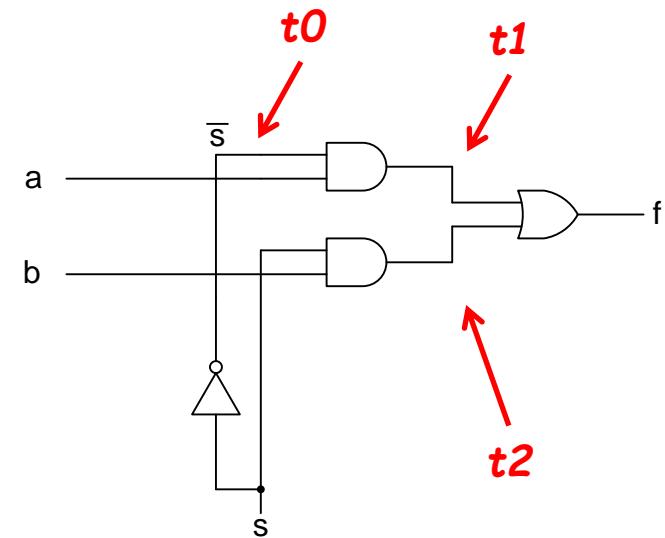
```
    t0 <= NOT s;
```

```
    t1 <= t0 AND a;
```

```
    t2 <= s AND b;
```

```
    f <= t1 OR t2;
```

```
END estrutural;
```



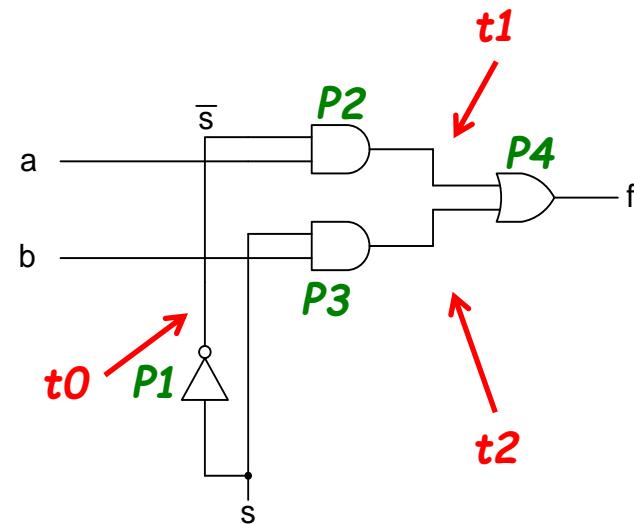
- A descrição estrutural especifica separadamente a função de cada porta no multiplexador.
- Essa descrição é baseada na expressão booleana do circuito.

VHDL - Características de Projeto

Descrição ou modelagem estrutural usando componente

- Solução 02:

```
ENTITY not1 IS
    PORT (x : IN BIT;
          z : OUT BIT);
END not1;
ARCHITECTURE logica OF not1 IS
BEGIN
    z <= NOT x;
END logica;
```

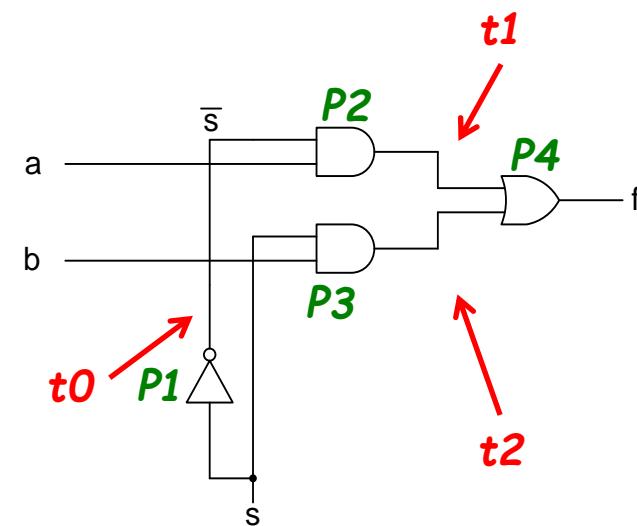


- A descrição estrutural especifica separadamente a função de cada porta no multiplexador.
- Essa descrição é baseada na expressão booleana do circuito.

VHDL - Características de Projeto

Descrição ou modelagem estrutural usando componente

```
ENTITY and2 IS
    PORT (x, y : IN BIT;
          z : OUT BIT);
END and2;
ARCHITECTURE logica OF and2 IS
BEGIN
    z <= x AND y;
END logica;
```

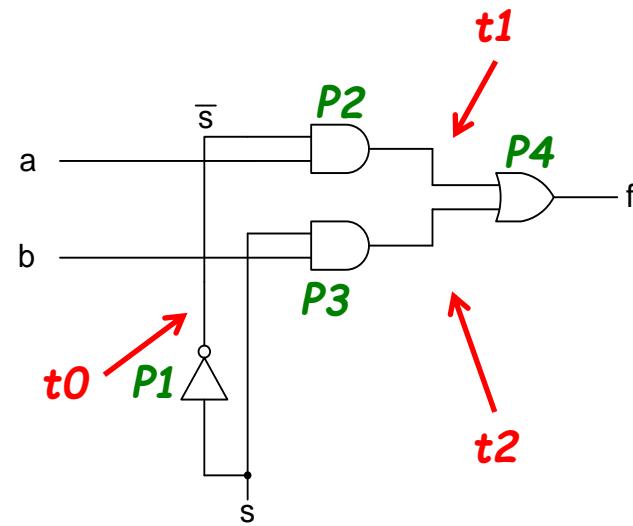


- A descrição estrutural especifica separadamente a função de cada porta no multiplexador.
- Essa descrição é baseada na expressão booleana do circuito.

VHDL - Características de Projeto

Descrição ou modelagem estrutural usando componente

```
ENTITY or2 IS
    PORT (x, y : IN BIT;
          z : OUT BIT);
END or2;
ARCHITECTURE logica OF or2 IS
BEGIN
    z <= x OR y;
END logica;
```



- A descrição estrutural especifica separadamente a função de cada porta no multiplexador.
- Essa descrição é baseada na expressão booleana do circuito.

VHDL - Características de Projeto

Descrição ou modelagem estrutural usando componente

```
ENTITY multiplexador IS
  PORT (a, b : IN BIT;
        s : IN BIT;
        f : OUT BIT);
END multiplexador;

ARCHITECTURE estrutural OF multiplexador IS
COMPONENT not1
  PORT (x : IN BIT;
        z : OUT BIT);
END component;

COMPONENT and2
  PORT (x, y : IN BIT;
        z : OUT BIT);
END component;
```

```
COMPONENT or2      -- continuacao
  PORT (x, y : IN BIT;
        z : OUT BIT);

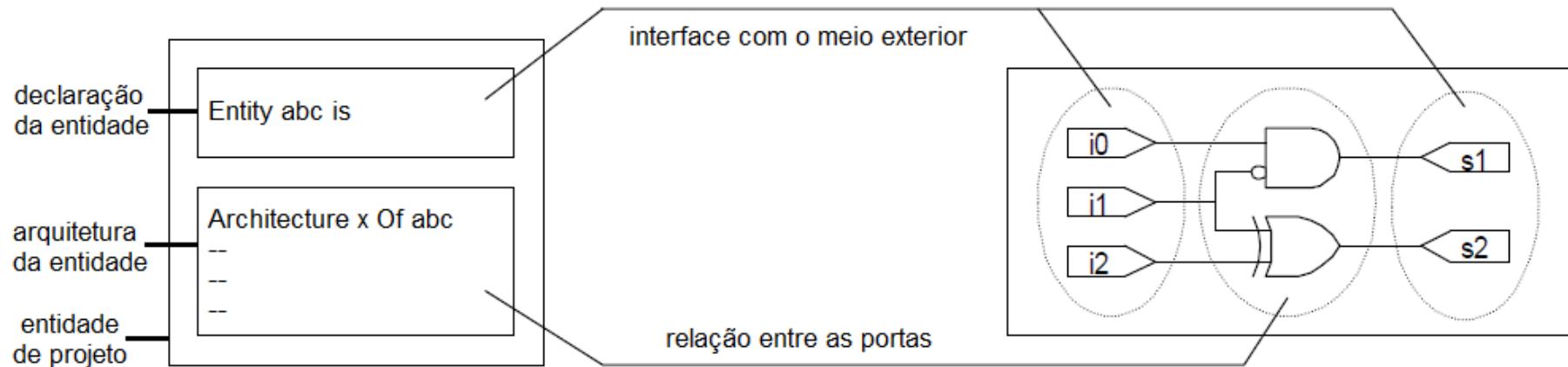
END component;

SIGNAL t0, t1, t2 : BIT;
BEGIN
  P1: not1 PORT MAP (s, t0);
  P2: and2 PORT MAP (t0, a, t1);
  P3: and2 PORT MAP (s, b, t2);
  P4: or2 PORT MAP (t1, t2, f);
END estrutural;
```

-- continuacao

Importante

Estrutura ou entidade de projeto



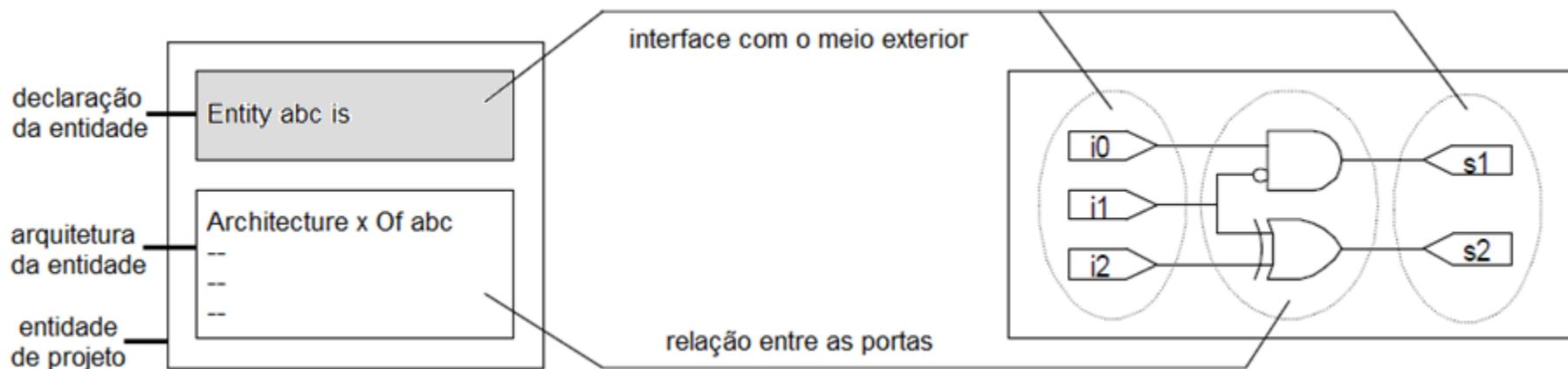
Importante

Estrutura ou entidade de projeto

- Pode representar desde uma simples porta lógica a um sistema completo
- Composta de duas partes:
 - Declaração da entidade → define portas de entrada e saída da descrição (equivalente ao símbolo de um bloco em captura esquemática)
 - Arquitetura da entidade → descreve as relações entre as portas (equivalente ao esquema contido no bloco em captura esquemática)

Importante

Declaração de entidade



```
ENTITY entidade_abc IS

    PORT (x0, x1      : IN      tipo_a;    -- entradas
          y0, y1      : OUT     tipo_b;    -- saídas
          y2          : BUFFER  tipo_c;    -- saída
          z0, z1      : INOUT   tipo_d);   -- entrada / saída

END entidade_abc;
```

Importante

Declaração de entidade

Declaração da entidade

ENTITY: inicia a declaração

PORT: define modo e tipo das portas

modo **IN** : entrada

modo **OUT** : saída

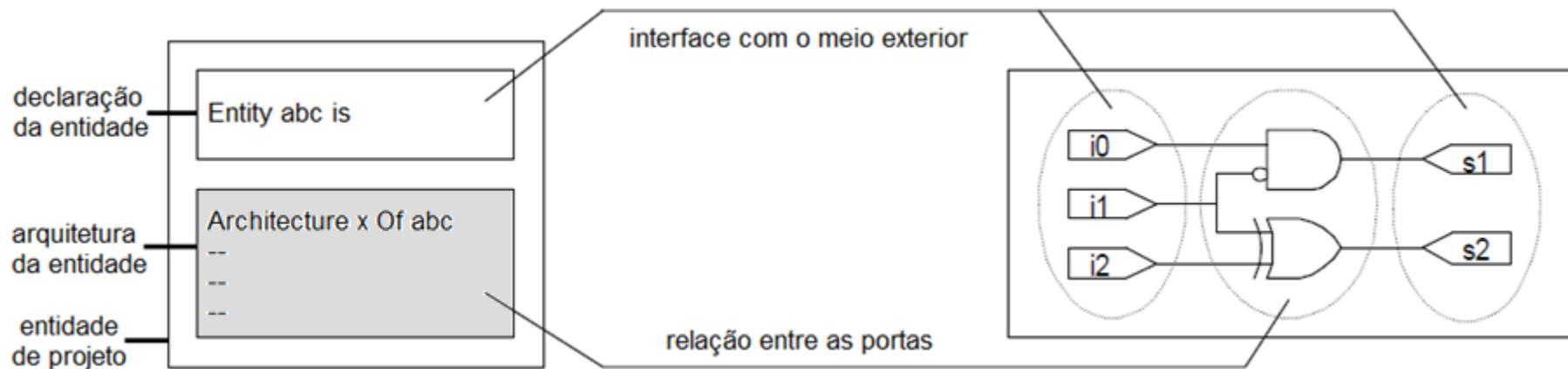
modo **BUFFER** : saída - pode ser referenciada
internamente

modo **INOUT** : bidirecional

END: termina a declaração

Importante

Arquitetura da entidade



```
ARCHITECTURE estilo_abc OF entidade_abc IS
  -- declaracoes de sinais e constantes
  -- declaracoes de componentes referenciais
  -- descricao de sub-programas locais
  -- definicao de novos tipos de dados locais
  --
BEGIN
  --
  -- declaracoes concorrentes
  --
END estilo_abc;
```

Importante

Arquitetura da entidade

Declaração da arquitetura

ARCHITECTURE: inicia a declaração

- Linhas que seguem podem conter:
 - declaração de sinais e constantes
 - declaração de componentes referenciados
 - descrição de subprogramas locais
 - definição de novos tipos

BEGIN: inicia a descrição

END: termina a descrição

Importante

Estrutura básica de um código em VHDL

```
LIBRARY IEEE;  
USE IEEE.STD_LOGIC_1164.all;  
USE IEEE.STD_LOGIC_UNSIGNED.all;
```

LIBRARY (PACOTES)

```
ENTITY exemplo IS  
PORT (  
    <descrição dos pinos de I/O>  
);  
END exemplo;
```

ENTITY (PINOS DE I/O)

```
ARCHITECTURE teste OF exemplo IS  
BEGIN  
    ...  
END teste;
```

ARCHITECTURE
(ARQUITETURA)

Importante

Descrição ou modelagem comportamental

- Declaração de um processo

...

```
ARCHITECTURE nome_da_arquitetura OF nome_da_entidade IS  
BEGIN
```

```
PROCESS (lista_de_sensibilidade)  
BEGIN
```

...

```
comandos;
```

...

```
END PROCESS;
```

```
END nome_da_arquitetura;
```

Importante

Descrição ou modelagem estrutural usando componente

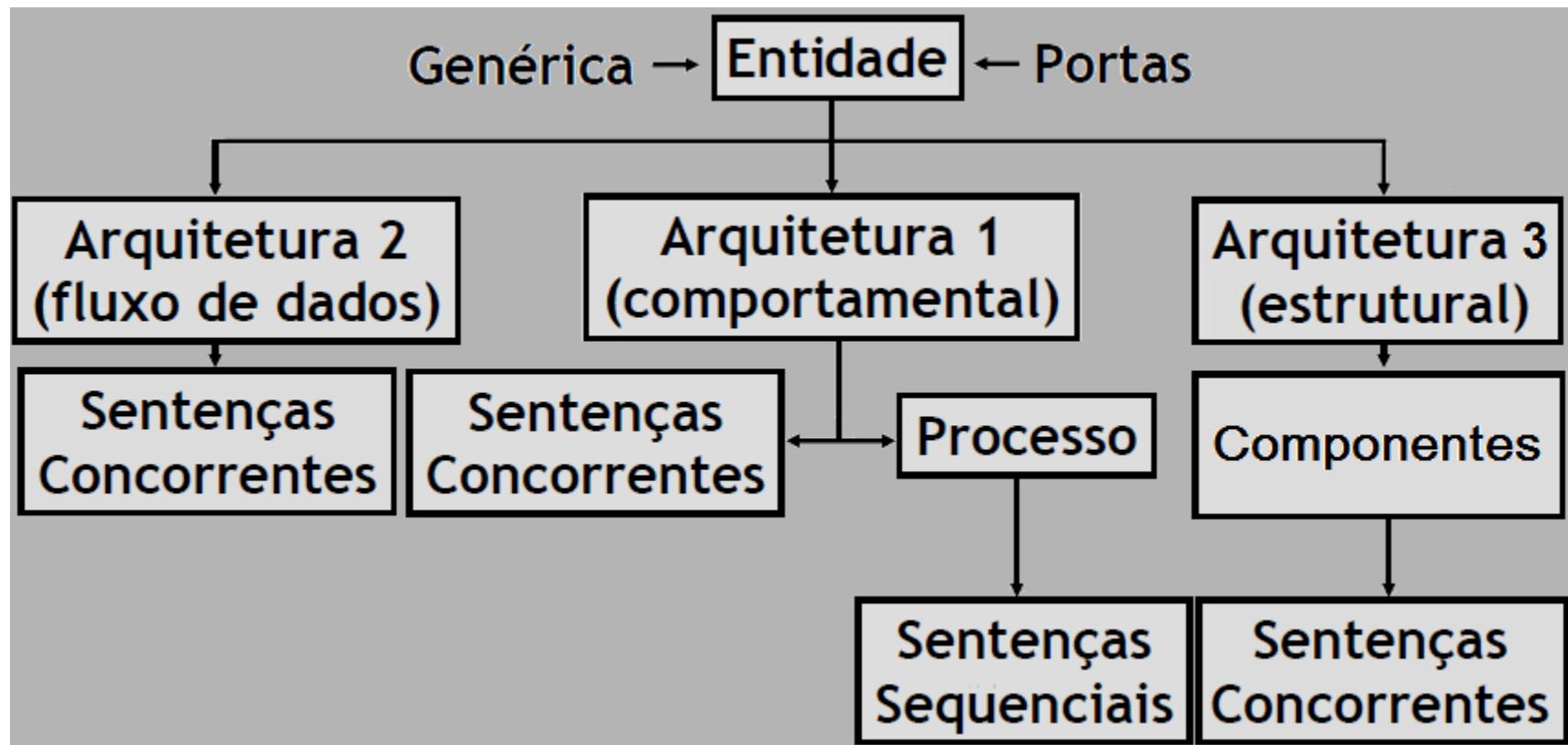
- Declaração e instanciação de um componente

```
...  
ARCHITECTURE nome_da_arquitetura OF nome_da_entidade IS  
COMPONENT nome_do_componente                                -- declaracao do componente  
    PORT (lista_de_porta_de_interface);  
END nome_do_componente;  
SIGNAL lista_de_sinal : tipo_de_dado;  
BEGIN  
    nome_da_instanciacao : nome_componente PORT MAP  
        (lista_de_associacao_de_porta);                      -- instanciacao do componente  
END nome_da_arquitetura;
```

- Nota: A declaração do componente é similar à declaração de entidade.

Importante

Estrutura básica de um código em VHDL



Resumo da Aula de Hoje

Tópicos mais importantes:

- Características de projeto

Próxima da Aula

- Tipos de Dados