Universidade Federal da Fronteira Sul
Curso de Ciência da Computação
Campus Chapecó

VHDL

Prof. Geomar A Schreiner
gschreiner@uffs.edu.br

By Caimi

Introdução

VHDL: Uma linguagem para descrever sistemas digitais

Outras linguagens de descrição de hardware:

VERILOG, AHDL, Handel-C, SDL, ISP, Esterel, … (existem dezenas)

Originalmente para especificar hardware, hoje, simulação e síntese, também!

Origem:

DoD 1980

Linguagem para documentar e descrever hardware "Very High Speed Integrated Circuits" (VHSIC), iniciado em 1980.

VHDL ➔ **V**HSIC **H**ardware **D**escription **L**anguage

Padrão IEEE em 1986 (Institute of Electrical and Electronics Engineers).

Em setembro de 2008 foi aprovada a mais recente versão, IEEE 1076-2008.

Linguagem utilizada mundialmente por empresas de CAD (simulação, síntese, propriedade intelectual). Verilog muito usada nos EUA.

Introdução

- **Benefícios**
  - Especificação do sistema digital:
    - Projetos independentes da tecnologia (implementação física é postergada)
    - Ferramentas de CAD compatíveis entre si
    - Flexibilidade: re-utilização, escolha de ferramentas e fornecedores
    - Facilidade de atualização dos projetos
    - Permite explorar, em um nível mais alto de abstração (em relação a álgebra de boole)
    - Permite, através de simulação, verificar o comportamento do sistema digital

Introdução

- **Benefícios (cont...)**
  - Nível físico:
    - Reduz tempo de projeto (favorece níveis mais abstratos de projeto)
    - Reduz custo
    - Elimina erros de baixo nível

    Conseqüência: reduz “time-to-market”

Introdução

- **Desvantagens**
  - Hardware gerado é menos otimizado
  - Controlabilidade/Observabilidade de projeto reduzidas
  - Falta de pessoal treinado para desenvolver com a linguagem.

Introdução

- **Características**
  - **Suporte para sentenças concorrentes:**

    No projeto real de sistemas digitais todos os elementos do sistema estão ativos simultaneamente e realizam suas tarefas ao mesmo tempo
  - **Suporte para sentenças Seqüenciais:**

    Permite controle seqüencial como em um progama comum (isto é, case, if-then-else, loop, etc.)
  - **Suporte para Bibliotecas:**

    Primitivas definidas pelo usuário e pré-definidas pelo sistema podem residir em uma biblioteca.
  - **Suporte a Projeto Hierárquico**
  - **Independente de Tecnologia**

Introdução

- **Características (cont...)**
  - **Projeto Genérico:**

    Descrições genéricas são configuráveis em tamanho, características físicas, temporização, condições de operação, etc.
  - **Uso de subprogramas:**

    Habilidade de definir e usar funções e procedimentos;

    Subprogramas são utilizados para a conversão explícita de tipos, redefinição de operadores, etc.

Introdução

Características (cont...)

- **Suporta declaração de tipos:**

  não está limitada a tipos de dados como Bit ou Booleanos, permite tipos inteiros, de ponto flutuante, enumerados, assim como tipos definidos pelos usuários.

  possibilita definição de novos operadores para os novos tipos criados pelo usuário.

- **Controle de Temporização :**

  Habilidade para especificar temporização em todos os níveis.

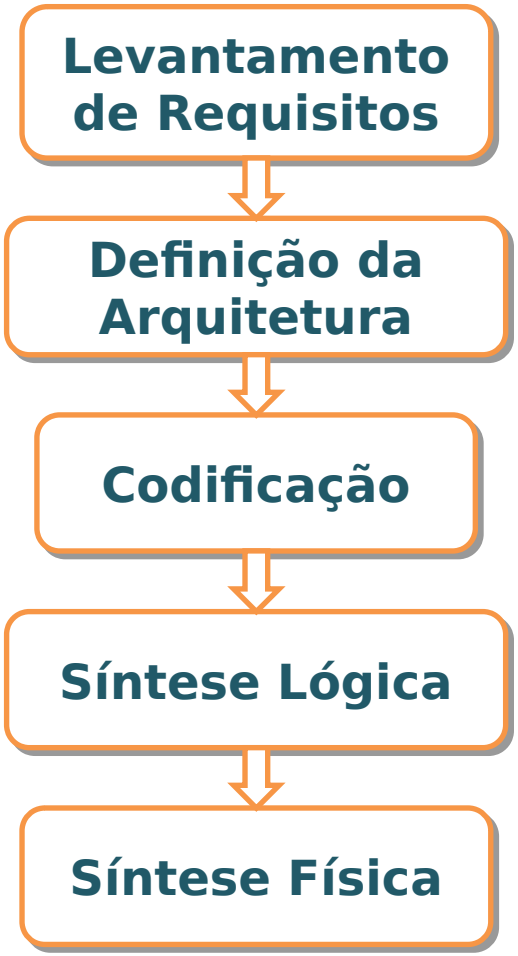
  Construções para detecção de rampa do sinal (subida ou descida), especificação de atraso, etc. estão disponíveis.

Introdução

- ✓ Permite descrever hardware em diversos níveis de abstração
  - Algorítmico (também chamado Comportamental)
  - Transferência entre registradores (RTL)
  - Nível lógico com atrasos unitários
  - Nível lógico com atrasos arbitrários
  - Estrutural (interconexão entre componentes)

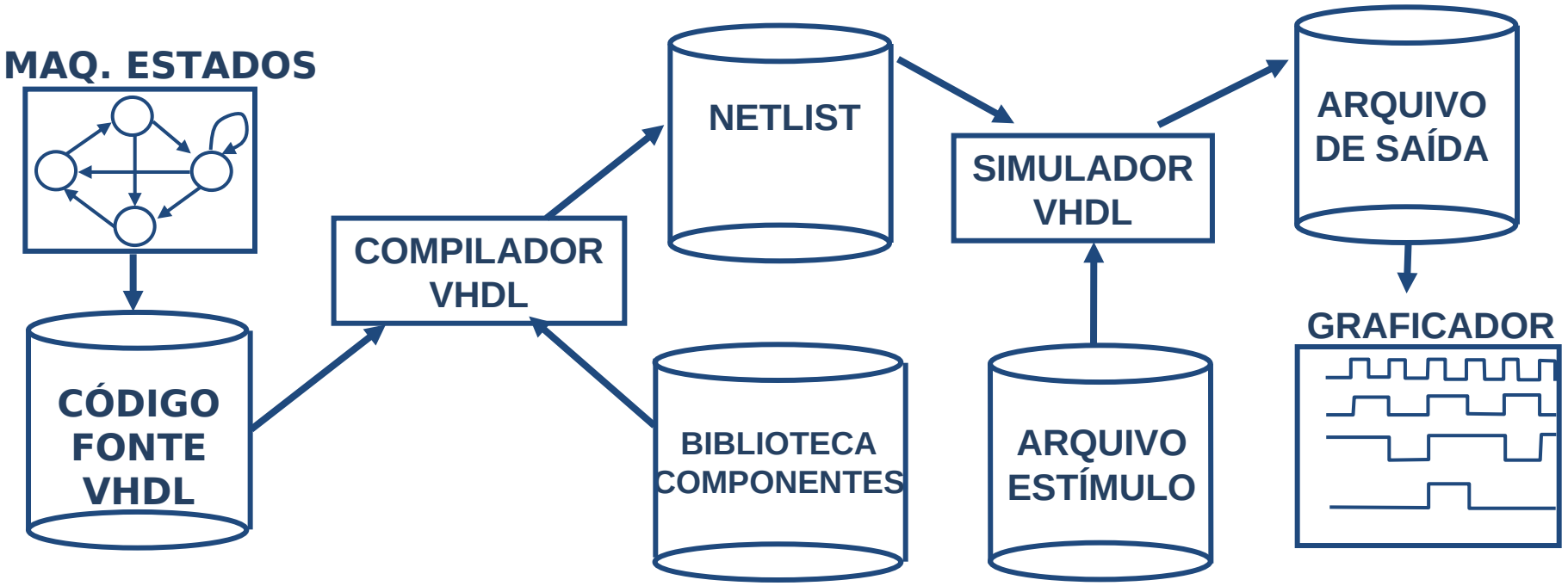
Introdução

✓ Fluxo de projeto (resumido)



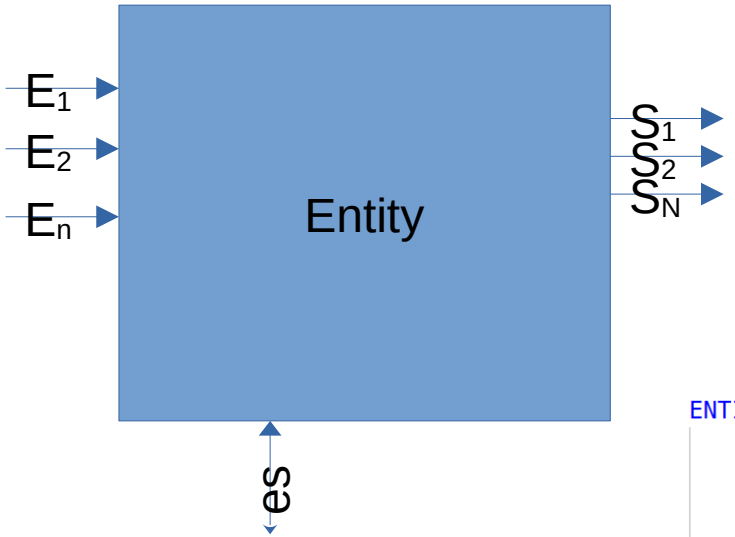
Introdução

- ✓ **Codificando... Compilando... Simulando...**



Representação de um sistema em VHDL

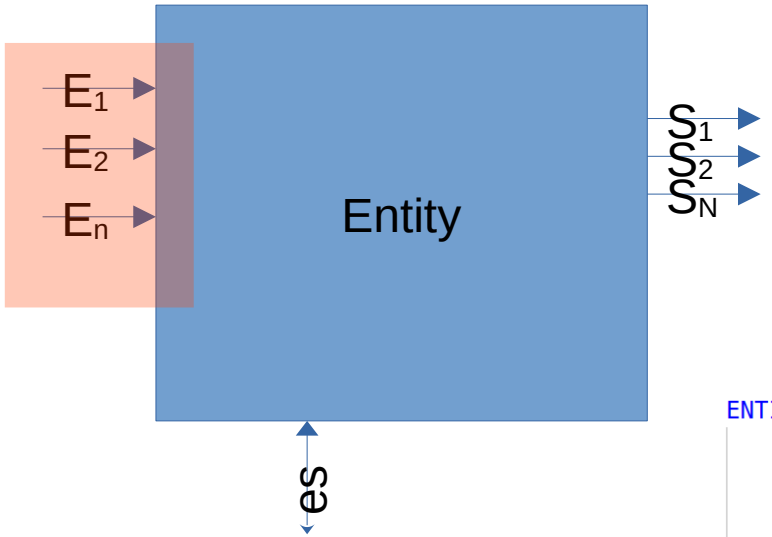
- ✓ Cada módulo tem sua própria "*entity*" e "*architecture*".




```
ENTITY nome_entidade IS
    PORT (
        E1 : IN STD_LOGIC_VECTOR (2 DOWNTO 0); -- Input
        E2 : IN STD_LOGIC; -- Input
        En : IN STD_LOGIC; -- Input
        S1 : OUT STD_LOGIC_VECTOR (7 DOWNTO 0); -- Output
        S2 : OUT STD_LOGIC; -- Output
        SN : OUT STD_LOGIC -- Output
    );
END ENTITY nome_entidade;
```

Representação de um sistema em VHDL

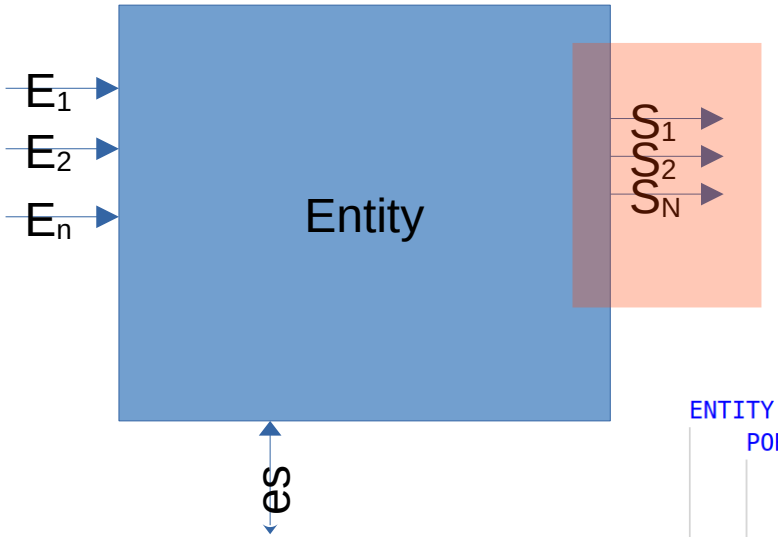
- ✓ Cada módulo tem sua própria "*entity*" e "*architecture*".




```
ENTITY nome_entidade IS
    PORT (
        E1 : IN STD_LOGIC_VECTOR (2 DOWNTO 0); -- Input
        E2 : IN STD_LOGIC; -- Input
        En : IN STD_LOGIC; -- Input
        S1 : OUT STD_LOGIC_VECTOR (7 DOWNTO 0); -- Output
        S2 : OUT STD_LOGIC; -- Output
        SN : OUT STD_LOGIC -- Output
    );
END ENTITY nome_entidade;
```

Representação de um sistema em VHDL

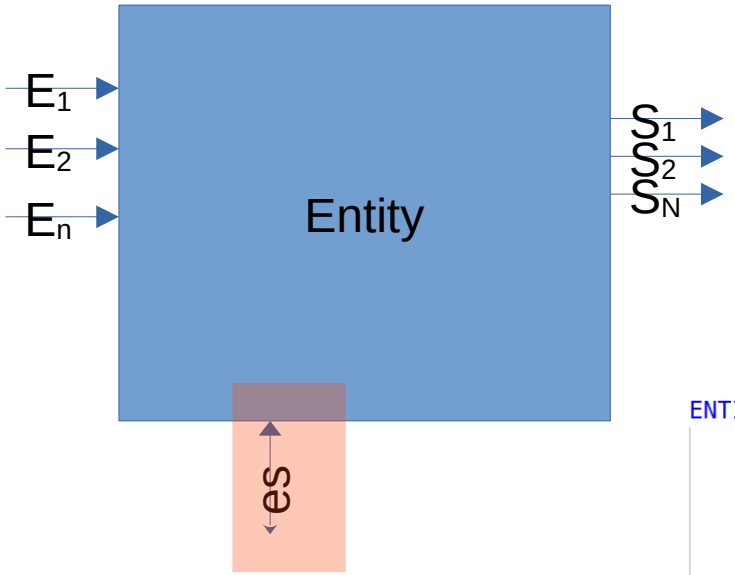
- ✓ Cada módulo tem sua própria "*entity*" e "*architecture*".




```
ENTITY nome_entidade IS
    PORT (
        E1 : IN STD_LOGIC_VECTOR (2 DOWNTO 0); -- Input
        E2 : IN STD_LOGIC; -- Input
        En : IN STD_LOGIC; -- Input
        S1 : OUT STD_LOGIC_VECTOR (7 DOWNTO 0); -- Output
        S2 : OUT STD_LOGIC; -- Output
        SN : OUT STD_LOGIC -- Output
    );
END ENTITY nome_entidade;
```

Representação de um sistema em VHDL

- ✓ Cada módulo tem sua própria "*entity*" e "*architecture*".



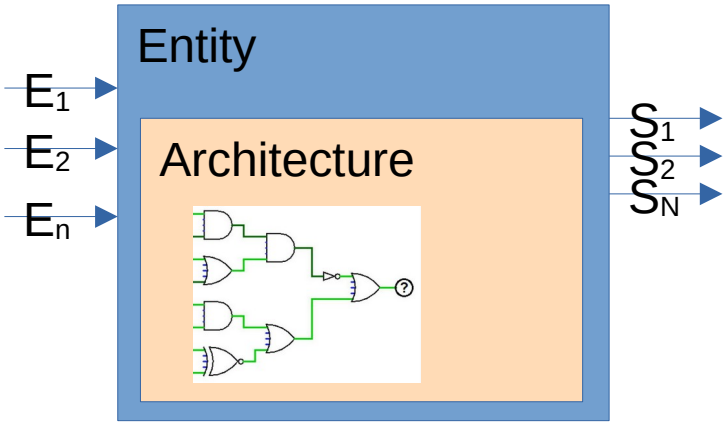

```
ENTITY nome_entidade IS
    PORT (
        E1 : IN STD_LOGIC_VECTOR (2 DOWNTO 0); -- Input
        E2 : IN STD_LOGIC; -- Input
        En : IN STD_LOGIC; -- Input
        S1 : OUT STD_LOGIC_VECTOR (7 DOWNTO 0); -- Output
        S2 : OUT STD_LOGIC; -- Output
        SN : OUT STD_LOGIC -- Output
    );
END ENTITY nome_entidade;
```

Representação de um sistema em VHDL

- ✓ Cada módulo tem sua própria "*entity*" e "*architecture*".
- ✓ As arquiteturas podem ser descritas tanto a nível comportamental quanto estrutural ou uma mistura.

  Comportamental: Algorítmica ou fluxo de dados;

  Estrutural: indica componentes e conexões;




```
ARCHITECTURE nome_arquitetura OF nome_entidade IS
BEGIN

    --BLOCO DE CÓDIGOS

END nome_arquitetura;
```

Representação de um sistema em VHDL

- ✓ Cada módulo tem sua própria "*entity*" e "*architecture*".
- ✓ As arquiteturas podem ser descritas tanto a nível comportamental quanto estrutural ou uma mistura.

  Comportamental: Algorítmica ou fluxo de dados;

  Estrutural: indica componentes e conexões;

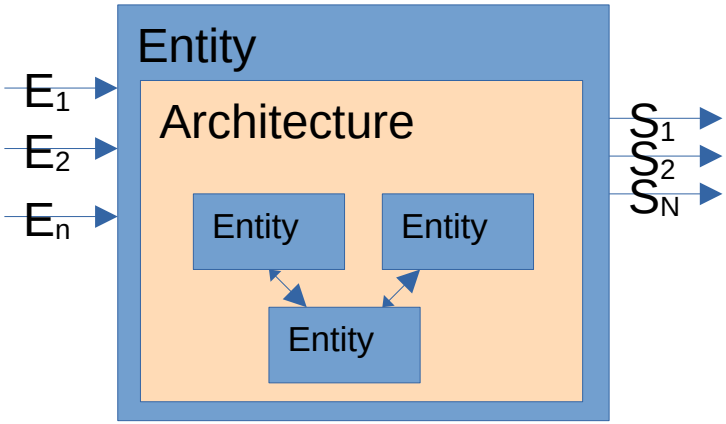

```
ARCHITECTURE nome_arquitetura OF nome_entidade IS
BEGIN

    --BLOCO DE CÓDIGOS

END nome_arquitetura;
```

Representação de um sistema em VHDL

- ✓ Cada módulo tem sua própria "*entity*" e "*architecture*".
- ✓ As arquiteturas podem ser descritas tanto a nível comportamental quanto estrutural ou uma mistura.

Comportamental: Algorítmica ou fluxo de dados;

Estrutural: indica componentes e conexões;




```
ARCHITECTURE nome_arquitetura OF nome_entidade IS
BEGIN

    --BLOCO DE CÓDIGOS
                architecture behav2 of adder is
END nome_arquit     component full_adder is
                        port(
                            a_e,b_e: in std_logic;
                            carry_in : in std_logic;
                            carry, sum: out std_logic
                        );
                    end component full_adder;
```

Representação de um sistema em VHDL

- ✓ Cada módulo tem sua própria "*entity*" e "*architecture*".
- ✓ As arquiteturas podem ser descritas tanto a nível comportamental quanto estrutural ou uma mistura.

  Comportamental: Algorítmica ou fluxo de dados;

  Estrutural: indica componentes e conexões;
- ✓ Toda a comunicação ocorre através das portas declaradas em cada *entity*, observando-se o tipo, tamanho, se é sinal ou barramento e a direção.
- ✓ Várias funções e tipos básicos são armazenados em bibliotecas (*library*). A biblioteca "IEEE" sempre é incluída.

```
library ieee;
use ieee.std_logic_1164.all;
```

Representação de um sistema em VHDL

- ✓ Cada módulo tem sua própria "*entity*" e "*architecture*".
- ✓ As arquiteturas podem ser descritas tanto a nível comportamental quanto estrutural ou uma mistura.

  Comportamental: Algorítmica ou fluxo de dados;

  Estrutural: indica componentes e conexões;
- ✓ Toda a comunicação ocorre através das portas declaradas em cada *entity*, observando-se o tipo, tamanho, se é sinal ou barramento e a direção.
- ✓ Várias funções e tipos básicos são armazenados em bibliotecas (*library*). A biblioteca "IEEE" sempre é incluída.
- ✓ Biblioteca do usuário (default): work. Todos os arquivos contidos no diretório de trabalho fazem parte da biblioteca do usuário.

Representação de um sistema em VHDL

- ✓ VHDL não é case sensitive

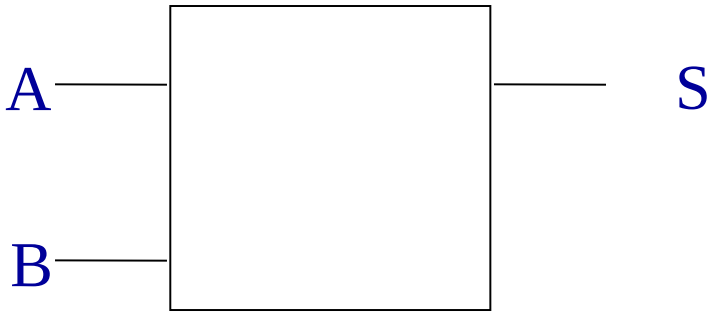
  Bit ↔ bit ↔ BIT
- ✓ Comentários: dois hifens adjascentes (--)

  -- esta linha é um comentário

Representação de um sistema em VHDL

Entity

- ✓ *Especifica somente a interface*
- ✓ *Não contém definição do comportamento*
- ✓ *Direção: in, out, inout, buffer*

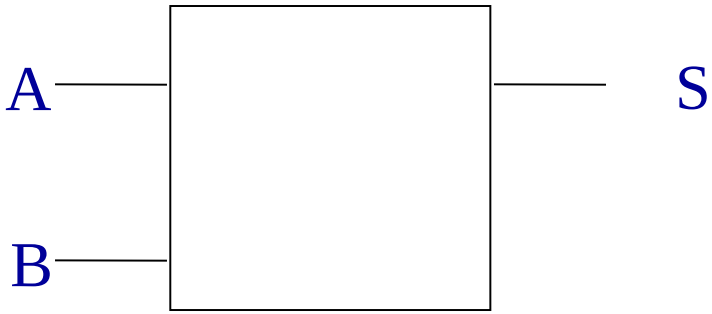



```
entity hw is
    port (
        A: in STD_LOGIC;
        B: in STD_LOGIC;
        S: out STD_LOGIC
    );
end hw;
```

Representação de um sistema em VHDL

Entity

- ✓ *Especifica somente a interface*
- ✓ *Não contém definição do comportamento*
- ✓ *Direção: in, out, inout, buffer*

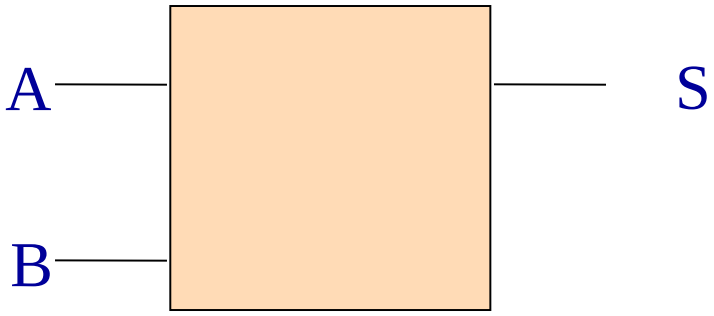



```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity hw is
5       port (
6           A: in STD_LOGIC;
7           B: in STD_LOGIC;
8           S: out STD_LOGIC
9       );
10  end hw;
```

Representação de um sistema em VHDL

Entity

- ✓ *Especifica somente a interface*
- ✓ *Não contém definição do comportamento*
- ✓ *Direção: in, out, inout, buffer*

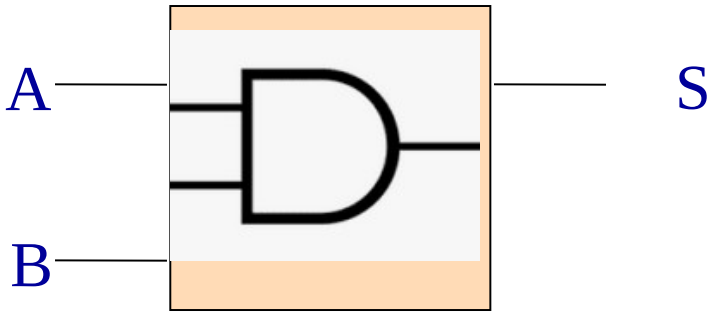



```
architecture comportamento of hw is
begin



end comportamento;
```

Representação de um sistema em VHDL

Entity

- ✓ *Especifica somente a interface*
- ✓ *Não contém definição do comportamento*
- ✓ *Direção: in, out, inout, buffer*

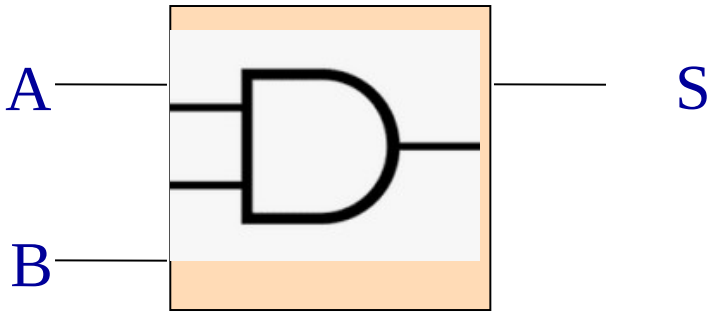



```
architecture comportamento of hw is
begin



end comportamento;
```

Representação de um sistema em VHDL

Entity

- ✓ *Especifica somente a interface*
- ✓ *Não contém definição do comportamento*
- ✓ *Direção: in, out, inout, buffer*

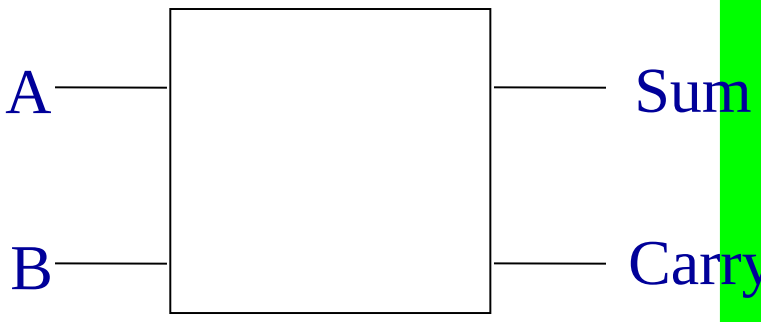



```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity hw is
5       port (
6           A: in STD_LOGIC;
7           B: in STD_LOGIC;
8           S: out STD_LOGIC
9       );
10  end hw;
11
12  architecture comportamento of hw is
13  begin
14
15      S <= A and B;
16
17  end comportamento;
```

Representação de um sistema em VHDL

Entity

- ✓ *Especifica somente a interface*
- ✓ *Não contém definição do comportamento*
- ✓ *Direção: in, out, inout, buffer*

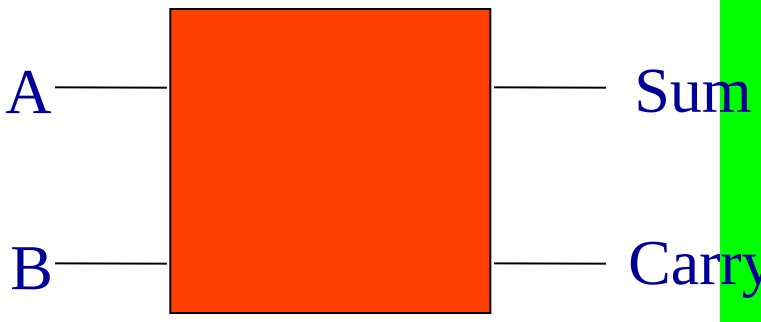



```
entity halfadder is
        port (
                A: in STD_LOGIC;
                B: in STD_LOGIC;
                sum: out STD_LOGIC;
                carry: out STD_LOGIC
        );
end halfadder;
```

Representação de um sistema em VHDL

Entity

- ✓ *Especifica somente a interface*
- ✓ *Não contém definição do comportamento*
- ✓ *Direção: in, out, inout, buffer*




```
entity halfadder is
        port (
                A: in STD_LOGIC;
                B: in STD_LOGIC;
                sum: out STD_LOGIC;
                carry: out STD_LOGIC
        );
end halfadder;
```

Representação de um sistema em VHDL

Architecture

- ✓ *Especifica o comportamento da entity*
- ✓ *Deve ser associada a uma entity específica*
- ✓ *Uma entity pode ter várias architectures*

```
architecture comp of halfadd is
   begin
        sum <= A xor B;
        carry <= A and B;
   end comp;
```

Representação de um sistema em VHDL

Entity:

```
entity nome is
  [generic (lista de parâmetros);]
  [port (lista de parâmetros);]
  [ declarações ]
  [begin  sentenças ]
end [ entity ] [ nome ];
```

Architecture:

```
architecture nome of
    nome_entidade is
   [ declarações ]
   begin
    [ sentenças concorrentes ]
end [ architecture ] [ nome ];
```

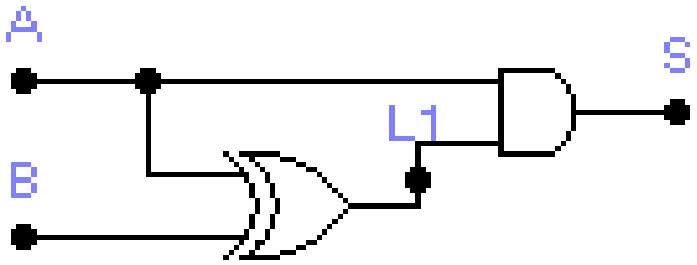
Representação de um sistema em VHDL

Descrições:

a) algorítmica

b) fluxo de dados

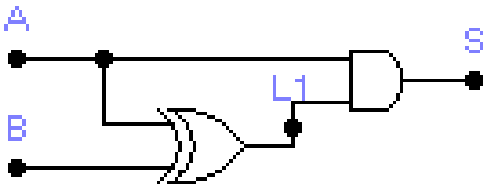
c) estrutural



Representação de um sistema em VHDL

a) algorítmica

```
entity comportamento is
   port (
         A: in STD_LOGIC;
         B: in
   STD_LOGIC;
         S: out
   STD_LOGIC
   );
end
```




```
architecture comport_algor of
                    comportamento is
begin
   process(A,B)
   begin
         if(B < A) then
               s <= '1';
         else
               s <= '0';
         end if;
      end process;
end comport_algor;
```

Representação de um sistema em VHDL

Primitiva de base (concorrência): process

Observar diferença entre variável e sinal:

Variável: interna ao processo, do tipo natural, atribuição IMEDIATA

Sinal: global, com atribuição ao término do processo

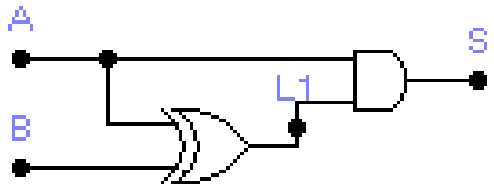
- ✓ Notar que na declaração do processo há uma lista de ativação

Significado: o processo está em espera até um sinal da lista de ativação mudar.

Representação de um sistema em VHDL

b) fluxo de dados

```
entity comportamento is
   port (
            A: in STD_LOGIC;
            B: in STD_LOGIC;
            S: out STD_LOGIC
   );
end comportamento;
```

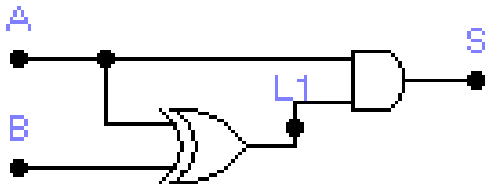



```
architecture comport_fluxo of
                        comportamento is
begin
   s <= ‘1’ when B < A
            else '0';
end comport_fluxo ;
```

Representação de um sistema em VHDL

c) estrutural

```
entity comportamento is
   port (
            A: in STD_LOGIC;
            B: in STD_LOGIC;
            S: out STD_LOGIC
   );
end comportamento;
```




```
architecture comp_estrutural of
                    comportamento is
   signal L1: STD_LOGIC;
   component  XOR2 is
    port (A1,B1: in std_logic; X1: out
std_logic);
    end component;
   component  And2 is
    port (A2,B2: in std_logic; X2: out
std_logic);
    end component
begin
   U1:  xor2 port map (A,B,L1);
   U2:  and2 port map (A,L1,S);
end comp_estrutural ;
```

(existem outros 2 arquivos com os pares entidade arquitetura para AND2 e XOR2)

Sintaxe VHDL

1. Faça a descrição de uma porta lógica OU de 4 entradas
2. Refaça o exercício acima, considerando agora que as entradas são um array
3. Dados os esquemáticos, obtenha descrições VHDL compatíveis