VHDL

Comentários
Os comentários são iniciados pelo símbolo - -, não sendo possível comentar um bloco todo como por exemplo na linguagem C.
Exemplo:

```
-- ******************************
-- a demo circuit
-- ******************************
eq <= p0 or p1;  --sum term
p1 <= i0 and i1; --product term #1
```

Nomes
Os identificadores devem começar com uma letra (a linguagem não diferencia maiúsculas de minúsculas), pode conter dígitos decimais e underscore (não pode ter dois sucessivos e ele não pode ser o último dígito).

Library
É uma biblioteca que armazena as unidades de projeto analisadas

- Library “work”: biblioteca default para onde o projeto é mapeado.
- Library “ieee” é usada por muitos pacotes (packages) IEEE.

Package
Usado para adicionar tipos de dados, operadores, funções, etc.
Exemplo:

```
1- library ieee;
2- use ieee.std_logic_1164.all;
```

Linha 1: Invoca uma library chamada ieee.
Linha 2: faz o pacote std_logic_1164 visível à unidade de projeto subsequente.

(Este pacote é necessário para o tipo de dado std_logic e para std_logic_vector)

Declaração de Entidade
Especifica as portas de entrada e saída de um sistema.

```
entity entity_name is
   port(
      port_names: mode data_type;
      port_names: mode data_type;
      ...
      port_names: mode data_type
   );
end entity_name;
```

No campo (mode) pode ser:

- in: fluxo entrando no circuito
- out: fluxo saindo do circuito
- inout: bidirecional (usado com tristate buffer)

No campo de (data_type) pode ser std_logic ou std_logic_vector.

No caso do std_logic, ele é definido no pacote IEEE std_logic_1164, possuindo 9 valores: ('U', 'X', '0', '1', 'Z', 'W', 'L', 'H', '-'), mas em síntese usamos apenas '0', '1', e 'Z'.

- '0', '1': lógico 0 e lógico 1
- 'Z': alta-impedância, como em um buffer tristate
- 'L' , 'H': lógico 0 e lógico 1 fracos, como em wired-logic
- 'X', 'W': desconhecido (unknown) e desconhecido fraco (weak unknown)
- 'U': para não inicializado (uninitialized);
- '-': para don't-care.

No caso do std_logic_vector, ele é um array de elementos do tipo std_logic também definido no pacote IEEE std_logic_1164, e implica em um barramento (bus) em um circuito físico podendo ser declarado como:

signal a: std_logic_vector(7 downto 0); -> fortemente recomendado.
signal a: std_logic_vector(0 to 7);   -> menos recomendado.

Operadores lógicos
Os operadores lógicos definidos para os tipos std_logic e std_logic_vector são not,and,or e xor, e é necessário utilizar parênteses para indicar a precedência das operações.
Exemplo: (a and b) or (c and d).

Corpo da Arquitetura

```
architecture arch_name of entity_name is
   declarations;
begin
   concurrent statement;
   concurrent statement;
   concurrent statement;
   . . .
end arch_name;
```

Onde:
Signal declaration é as declarações de sinais. (Sinais podem ser pensados como fios internos - nets)
Concurrent statement se refere a cada comando concorrente. (Podem ser vistos como uma parte do circuito com específica funcionalidade e tempo).

```
architecture sop_arch of eq1 is
   signal p0, p1 : std_logic;
begin
   -- sum of two product terms
   eq <= p0 or p1;
   -- product terms
   p0 <= (not i0) and (not i1);
   p1 <= i0 and i1;
end sop_arch;
```

Instanciação de componentes
Possui a mesma declaração de entidade mas uma diferente arquitetura em que duas instâncias de eq1 são necessárias.

```
unit_label: entity lib_name.entity_name(arch_name)
    port map(
        formal_signal=>actual_signal,
        formal_signal=>actual_signal,
        . . .
```

Onde:

- unit_label: um nome único para identificar a instância
- lib_name: a biblioteca onde o componente reside;
- formal_signal: sinal usado na declaração da entidade do componente
- actual_signal: sinal usado na arquitetura corrente

Exemplo:

```
architecture struc_arch of eq2 is
   signal e0, e1: std_logic;
begin
   -- instantiate two 1-bit comparators
   eq_bit0_unit: entity work.eq1(sop_arch)
      port map(i0=>a(0), i1=>b(0), eq=>e0);
   eq_bit1_unit: entity work.eq1(sop_arch)
      port map(i0=>a(1), i1=>b(1), eq=>e1);
   -- a and b are equal if individual bits are equal
   aeqb <= e0 and e1;
end struc_arch;
```

Testbench
Um testbench é um programa em VHDL que funciona como uma bancada de testes “virtual”, que inclui o circuito para ser testado (uut: unit under test), os estímulos de entrada (por exemplo, algo como um gerador de função) e um monitor de saída para verificar o resultado,