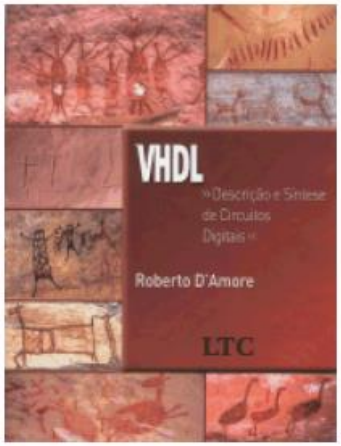
SEL5752/SEL0632 – Linguagens de Descrição de Hardware
Aula 7 – Bibliotecas e Pacotes

Prof. Dr. Maximiliam Luppe

Livro adotado:

VHDL - Descrição e Síntese de Circuitos Digitais
Roberto d'Amore

ISBN 85-216-1452-7
Editora LTC www.ltceditora.com.br



Para informações adicionais consulte: **www.ele.ita.br/~damore/vhdl**

Bibliotecas e Pacotes

Tópicos

- Bibliotecas
- Pacotes
- Ordem de análise na compilação

Bibliotecas

- **Bibliotecas de projeto**, ou *design libraries*:
  - local de armazenamento das unidades de projeto compiladas

- **Bibliotecas**:

- são diretórios criados com o auxílio da ferramenta de trabalho

- **Referência a uma biblioteca na descrição**:
  - feita por nome lógico

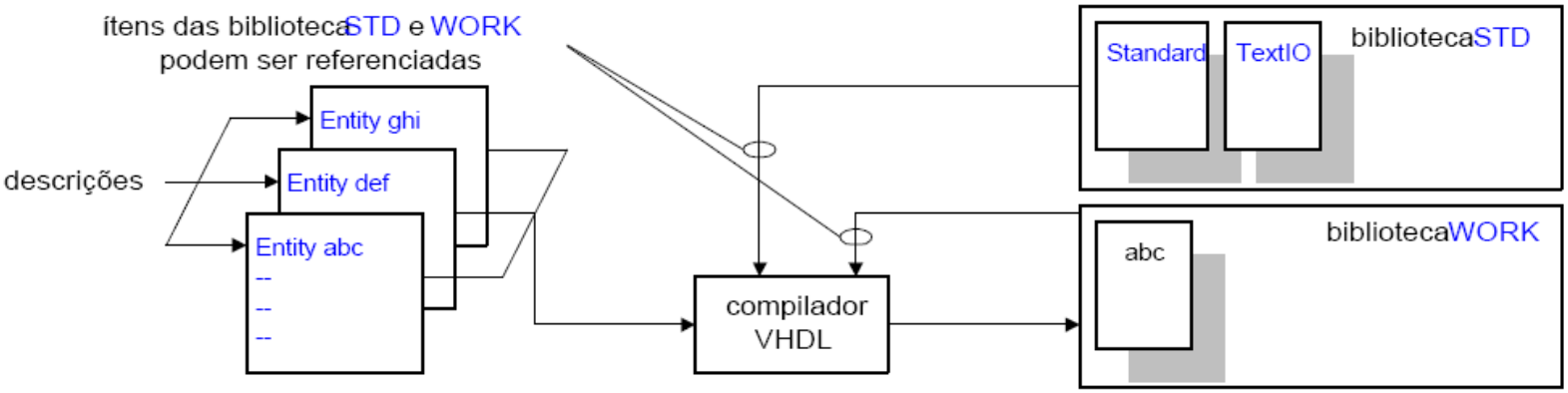
- **Nome lógico**:
  - identifica o caminho completo local onde a informação é armazenada

- **Emprego de uma biblioteca**:

```
LIBRARY nome_biblioteca;                    -- biblioteca pode ser referenciada
```

Bibliotecas

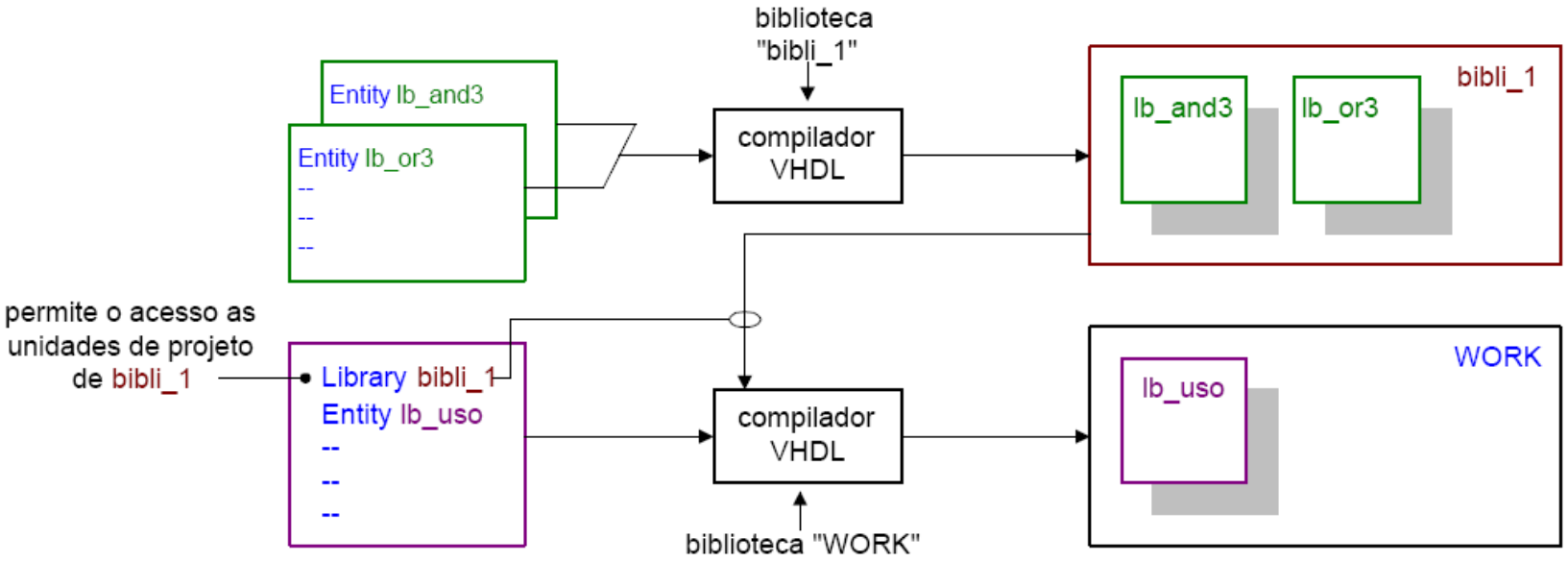
- **WORK**:
  - biblioteca padrão para o armazenamento das unidades de projeto
- **STD**
  - biblioteca pré-definida em VHDL
  - contém pacotes pré-definidos STANDARD TEXTIO
- **Implicitamente declaradas em qualquer descrição**
  - unidades armazenadas sempre podem ser referenciadas



Bibliotecas - exemplo

- **Entidades lb_and3 lb_or3**
  - compiladas / armazenadas em bibli_1

- **Clausula LIBRARY bibli_1**
  - biblioteca bibli_1 pode ser referenciada no projeto



Bibliotecas - exemplo

bibli_1

```
1  ENTITY lb_or3 IS
2    PORT (i0, i1, i2 :  IN  BIT;
3          s          :  OUT BIT);
4  END lb_or3;
5  ARCHITECTURE teste OF lb_or3 IS
6  BEGIN
7    s <= i0 OR i1 OR i2;
8  END teste;
```

```
1  ENTITY lb_and3 IS
2    PORT (i0, i1, i2 :  IN  BIT;
3          s          :  OUT BIT);
4  END lb_and3;
5  ARCHITECTURE teste OF lb_and3 IS
6  BEGIN
7    s <= i0 AND i1 AND i2;
8  END teste;
```

work

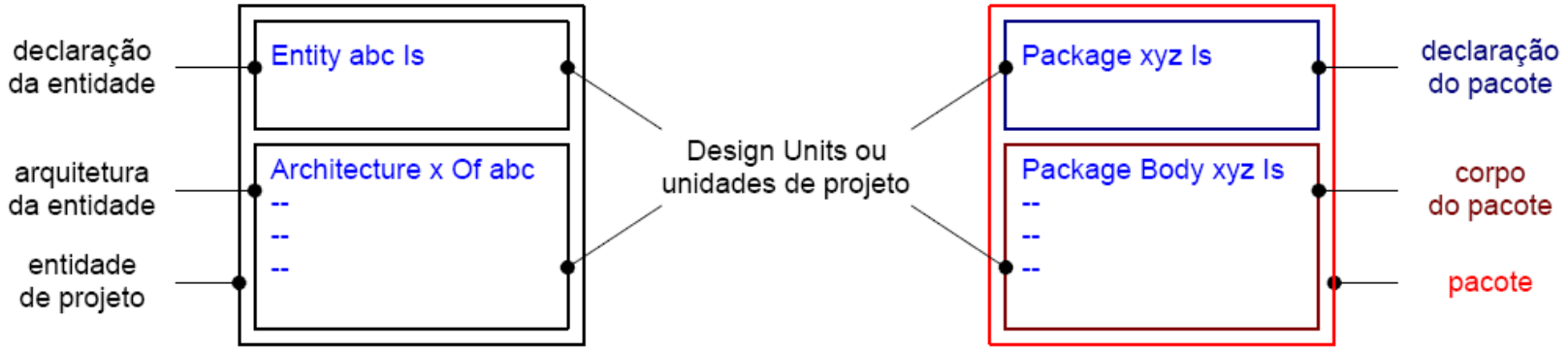
```
1  LIBRARY bibli_1;   -- biblioteca bibli_1 pode ser referenciada no projeto
2
3  ENTITY lb_uso IS
4    PORT (a, b, c :  IN  BIT;
5          s_e, s_ou  :  OUT BIT);
6  END lb_uso;
7
8  ARCHITECTURE teste OF lb_uso IS
9    COMPONENT lb_and3  -- resultado da compilacao se encontra em bibli_1
10     PORT (i0, i1, i2 : IN  BIT; s  :  OUT BIT);
11   END COMPONENT;
12
13   COMPONENT lb_or3   -- resultado da compilacao se encontra em bibli_1
14     PORT (i0, i1, i2 : IN  BIT; s  :  OUT BIT);
15   END COMPONENT;
16 BEGIN
17   x1: lb_and3 PORT MAP(a, b, c, s_e);
18   x2: lb_or3  PORT MAP(a, b, c, s_ou);
19 END teste;
```

Pacotes

- **Pacotes** → local para o armazenamento:
  - constantes,
  - subprogramas
  - declarações (tipos sinais globais componentes)

- **Pacotes** consistem de duas partes:
  - declaração do pacote
  - corpo do pacote (opcional)

- **Declaração do pacote** e **corpo do pacote**:
  - são unidades de projeto (devem ser compiladas)



Pacotes

- **Declaração do pacote**
  - declaração de: constantes tipos sinais globais
    subprogramas componentes arquivos

```
PACKAGE nome_pacote IS
  --
  -- declaracao de subprogramas, constantes, componentes,
  --               tipos, subtipos, sinais globais
  --
END nome_pacote;
```

- **Corpo do pacote:**
  - corpo de funções procedimentos

```
PACKAGE BODY nome_pacote IS
  --
  -- corpo de subprogramas
  --
END nome_pacote;
```

Pacotes

- **Cláusula USE**
  - torna visíveis os itens de um pacote
  - especifica:
    - biblioteca que o pacote se encontra
    - nome do pacote
    - item selecionado (ALL todos itens visíveis)

```
LIBRARY nome_biblioteca;                    -- biblioteca pode ser referenciada

USE nome_biblioteca.nome_pacote.ALL;        -- todos itens do pacote visiveis
USE nome_biblioteca.nome_pacote.item_abc    -- unicamente item_abc visivel
```

- **Segunda opção**:
  - acesso direto ao item, (necessita da referência completa)

```
LIBRARY nome_biblioteca;                  -- inclue a biblioteca no projeto

nome_biblioteca.nome_pacote.item_abc      -- acesso direto ao item item_abc
```

Pacotes - exemplos

- **Pacotes contendo unicamente a declaração:**

- definição de uma constante:

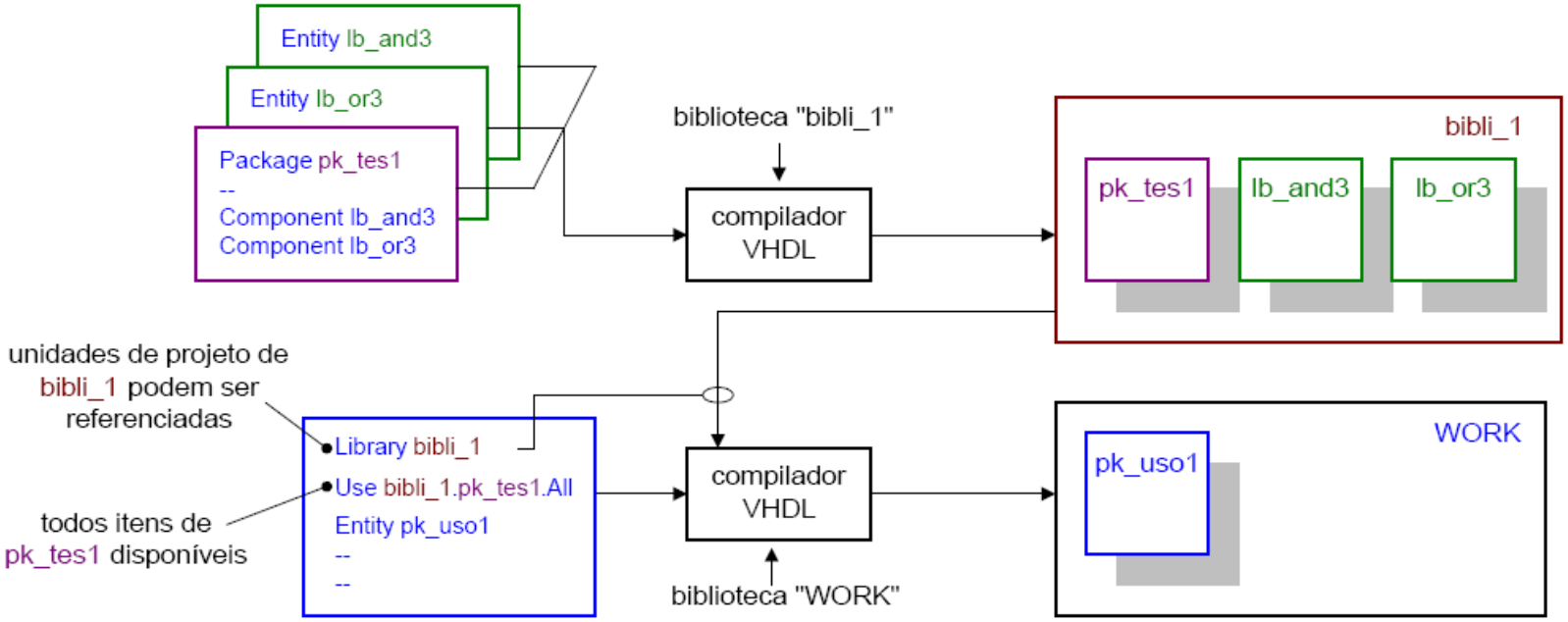
```
1 PACKAGE pack_0 IS
2   CONSTANT x1 : INTEGER := 7;
3 END pack_0;
```

- declaração de dois componentes

```
1 PACKAGE pk_tes1 IS        -- declaracao do pacote
2   COMPONENT lb_and3 PORT (i0, i1, i2 : IN  BIT; s : OUT BIT);
3   END COMPONENT;
4
5   COMPONENT lb_or3 PORT (i0, i1, i2 : IN  BIT; s : OUT BIT);
6   END COMPONENT;
7 END pk_tes1;
```

Pacotes - exemplo (empregando a cláusula USE)

- **lb_and3 lb_or3 pk_tes1**: compilados / armazenados em bibli_1
- **Pacote pk_tes1**: contém a declaração dos componentes
- **Clausula LIBRARY bibli_1**: biblioteca bibli_1 pode ser referenciada
- **Cláusula USE bibli_1.pk_tes1.ALL**: torna todos itens do pacote visivies



Pacotes - exemplo (empregando a cláusula USE)

bibli_1

```
1  ENTITY lb_or3 IS
2    PORT (i0, i1, i2 :  IN  BIT;
3          s          :  OUT BIT);
4  END lb_or3;
5  ARCHITECTURE teste OF lb_or3 IS
6  BEGIN
7    s <= i0 OR i1 OR i2;
8  END teste;
```

```
1  ENTITY lb_or3 IS
2    PORT (i0, i1, i2 :  IN  BIT;
3          s          :  OUT BIT);
4  END lb_or3;
5  ARCHITECTURE teste OF lb_and3 IS
6  BEGIN
7    s <= i0 AND i1 AND i2;
8  END teste;
```

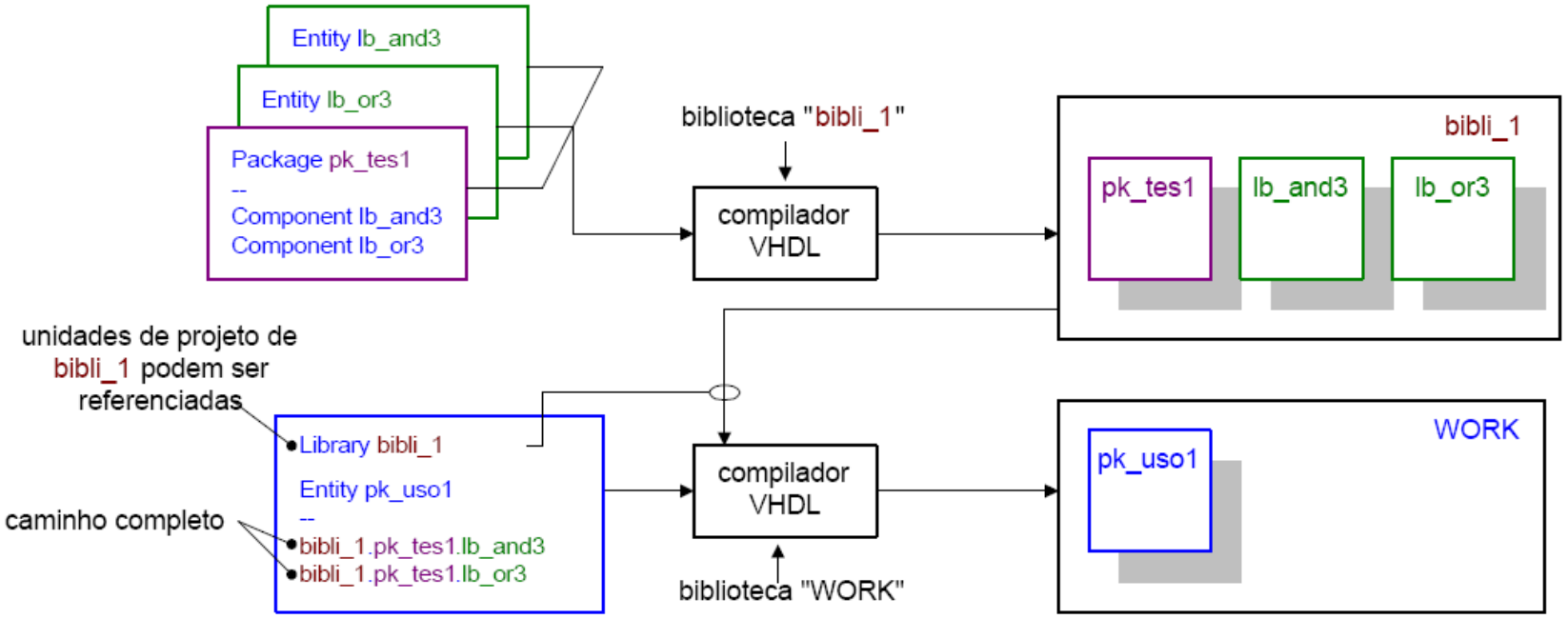
```
1  PACKAGE pk_tes1 IS        -- declaracao do pacote
2    COMPONENT lb_and3 PORT (i0, i1, i2 : IN  BIT; s : OUT BIT);
3    END COMPONENT;
4
5    COMPONENT lb_or3 PORT (i0, i1, i2 : IN  BIT; s : OUT BIT);
6    END COMPONENT;
7  END pk_tes1;
```

work

```
1  LIBRARY bibli_1;           -- biblioteca bibli_1 pode ser referenciada
2  USE bibli_1.pk_tes1.ALL;   -- torna todos itens do pacote visiveis
3
4  ENTITY pk_uso1 IS
5    PORT (a, b, c :  IN  BIT;
6          s_e, s_ou  :  OUT BIT);
7  END pk_uso1;
8
9  ARCHITECTURE teste OF pk_uso1 IS
10 BEGIN
11   x1: lb_and3 PORT MAP(a, b, c, s_e);
12   x2: lb_or3  PORT MAP(a, b, c, s_ou);
13 END teste;
```

Pacotes - exemplo (sem empregar a cláusula USE - referência direta)

- **lb_and3 lb_or3 pk_tes1**: compilados / armazenados em bibli_1
- **Pacote pk_tes1**: contém a declaração dos componentes
- **Clausula LIBRARY bibli_1**: biblioteca bibli_1 pode ser referenciada
- **bibli_1.pk_tes1. lb_and3 e bibli_1.pk_tes1. lb_or3** : caminho completo do item



Pacotes - exemplo (sem a cláusula USE)

bibli_1

```
1  ENTITY lb_or3 IS
2    PORT (i0, i1, i2 :  IN  BIT;
3          s          :  OUT BIT);
4  END lb_or3;
5  ARCHITECTURE teste OF lb_or3 IS
6  BEGIN
7    s <= i0 OR i1 OR i2;
8  END teste;
```

```
1  ENTITY lb_and3 IS
2    PORT (i0, i1, i2 :  IN  BIT;
3          s          :  OUT BIT);
4  END lb_and3;
5  ARCHITECTURE teste OF lb_and3 IS
6  BEGIN
7    s <= i0 AND i1 AND i2;
8  END teste;
```

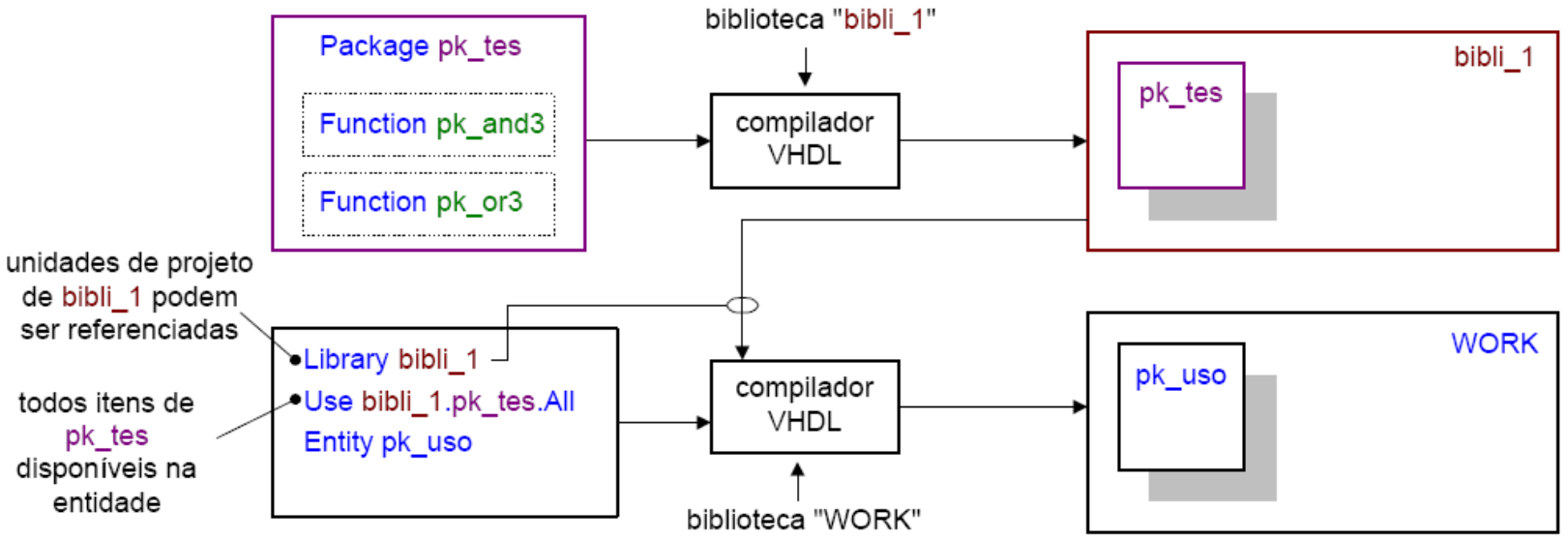
```
1  PACKAGE pk_tes1  IS       -- declaracao do pacote
2    COMPONENT lb_and3 PORT (i0, i1, i2 : IN  BIT; s : OUT BIT);
3    END COMPONENT;
4
5    COMPONENT lb_or3 PORT (i0, i1, i2 : IN  BIT; s : OUT BIT);
6    END COMPONENT;
7  END pk_tes1;
```

work

```
1  LIBRARY bibli_1;   -- inclue a biblioteca bibli_1 no projeto
2
3  ENTITY pk_uso2 IS
4    PORT (a, b, c :  IN  BIT;
5          s_e, s_ou  :  OUT BIT);
6  END pk_uso2;
7
8  ARCHITECTURE teste OF pk_uso2 IS
9  BEGIN
10   x1: bibli_1.pk_tes1.lb_and3 PORT MAP(a, b, c, s_e);
11   x2: bibli_1.pk_tes1.lb_or3  PORT MAP(a, b, c, s_ou);
12 END teste;
```

Pacotes - exemplo (pacote com corpo - conteúdo: duas funções)

- **Pacote pk_tes**: declaração e corpo das funções pk_and3 pk_or3
- **pk_tes**: compilado / armazenado em bibli_1
- **Clausula LIBRARY bibli_1**: biblioteca bibli_1 pode ser ref. no projeto
- **Cláusula USE bibli_1.pk_tes1.ALL**: torna todos itens do pacote visivies



Pacotes - exemplo (pacote com corpo - duas funções)

bibli_1

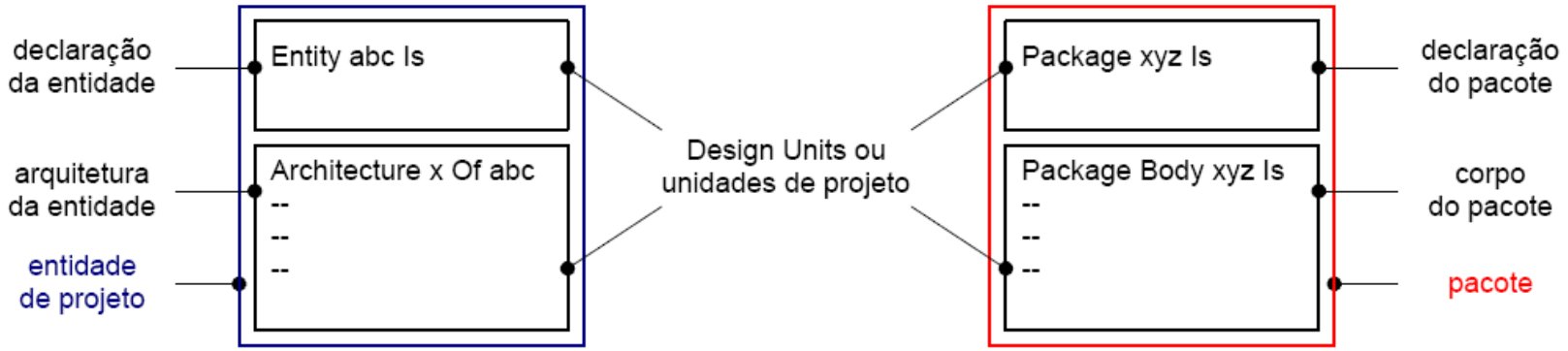
```
1  PACKAGE pk_tes IS        -- declaracao do pacote
2    FUNCTION pk_and3 (SIGNAL i0, i1, i2 : IN  BIT) RETURN BIT;
3    FUNCTION pk_or3  (SIGNAL i0, i1, i2 : IN  BIT) RETURN BIT;
4  END pk_tes;
5
6  PACKAGE BODY pk_tes IS  -- corpo do pacote
7    FUNCTION pk_and3 (SIGNAL i0, i1, i2 : IN  BIT) RETURN BIT IS
8    BEGIN
9      RETURN (i0 AND i1 AND i2);
10   END pk_and3;
11
12   FUNCTION pk_or3  (SIGNAL i0, i1, i2 : IN  BIT) RETURN BIT IS
13   BEGIN
14     RETURN (i0 OR i1 OR i2);
15   END pk_or3;
16 END pk_tes;
```

work

```
1  LIBRARY bibli_1;         -- biblioteca bibli_1 pode ser referenciada projeto
2  USE bibli_1.pk_tes.ALL; -- torna todos itens do pacote visiveis
3
4  ENTITY pk_uso IS
5    PORT (a, b, c    :  IN  BIT;
6          s_e, s_ou :  OUT BIT);
7  END pk_uso;
8
9  ARCHITECTURE teste OF pk_uso IS
10 BEGIN
11   s_e  <= pk_and3(a, b, c);
12   s_ou <= pk_or3(a, b, c);
13 END teste;
```

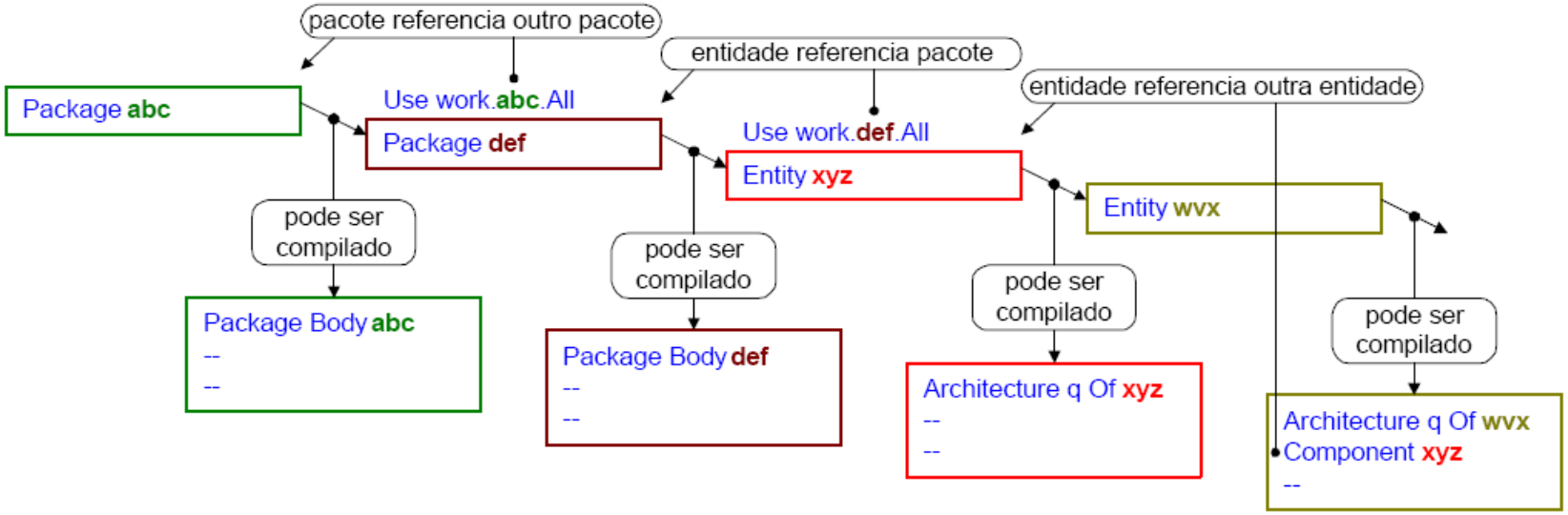
Ordem de análise na compilação

- **Compilação**: orientada por arquivos
- Um **arquivo**: pode conter uma ou mais unidades de projetos
- **Entidades**: declaração e arquitetura - podem ser compiladas isoladamente
- **Pacotes**: declaração e corpo - podem ser compilados isoladamente
- **Seqüência das operações de compilação**
  - deve observar a dependência entre as unidades compiladas



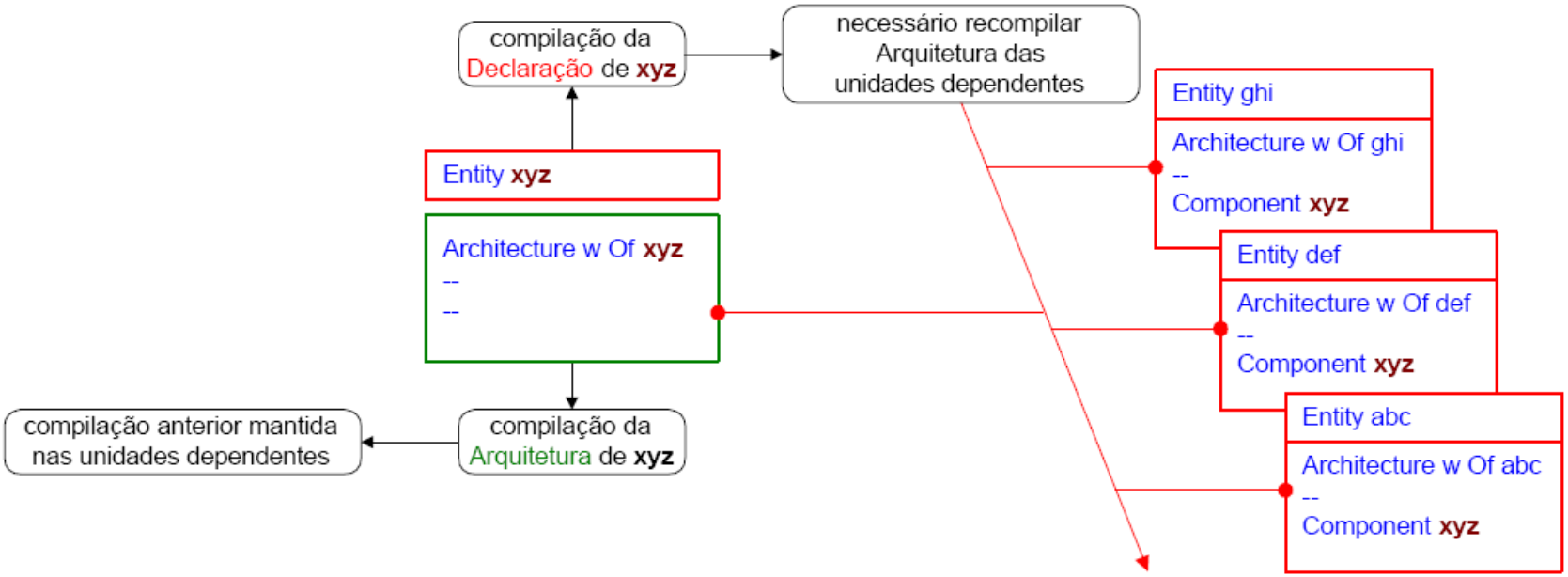
Ordem de análise na compilação

- **Compilação da declaração de um pacote** → realizada antes das unidades de projeto dependentes
- **Compilação da declaração de uma entidade** → realizada antes das unidades de projeto dependentes
- **Corpo de um pacote** → compilado após declaração
- **Arquitetura de uma entidade** → compilada após declaração
- **Corpo e Arquitetura** → necessários na simulação



Dependências na compilação - (entidades)

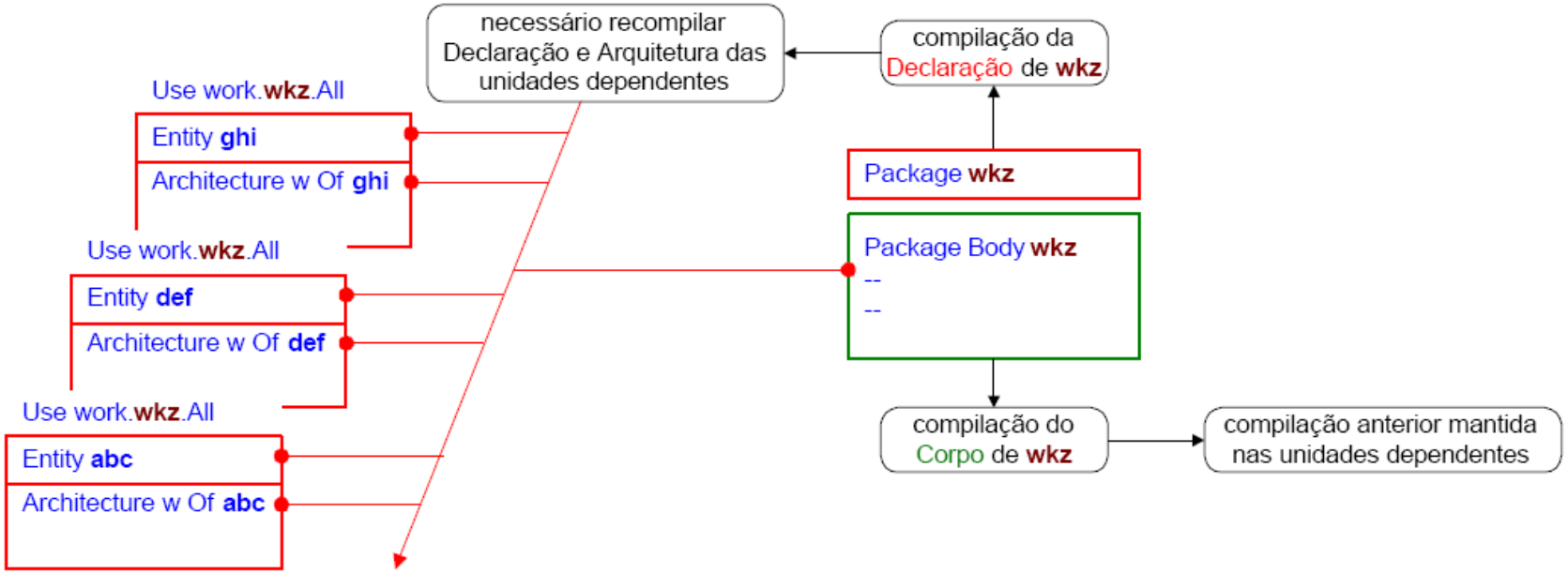
- **Declaração de uma entidade** **alterada e recompilada** →
  - necessário recompilar:
    - todas arquiteturas desta entidade (caso tenha mais de uma arquitetura)
    - todas arquiteturas que referenciam a entidade (componentes)
- **Arquitetura da entidade** **alterada e recompilada** →
  - nenhum outra recompilação é necessária



Dependências na compilação - (pacotes)

- **Declaração de um pacote** **alterado e recompilado →**
  - necessário recompilar:
    - todas unidades de projeto dependentes:
      declarações e arquiteturas entidades declarações pacotes e corpo de pacotes

- **Corpo do pacote** **alterado e recompilado →**
  - nenhum outra recompilação é necessária



Dependências na compilação - (exemplo)

- **Arquivo contendo duas unidades de projeto**
- **Não existe dependência entre as unidades**
- **Arquiteturas compiladas por último**

```
 1 -- arquivo dep_ver0.vhd: duas entidades
 2 ENTITY or3 IS
 3   PORT (i0, i1, i2 : IN BIT; s : OUT BIT);
 4 END or3;
 5
 6 ENTITY and3 IS
 7   PORT (i0, i1, i2 : IN BIT; s : OUT BIT);
 8 END and3;
 9
10 ARCHITECTURE teste OF or3 IS
11   BEGIN s <= i0 OR i1 OR i2;
12 END teste;
13
14 ARCHITECTURE teste OF and3 IS
15   BEGIN s <= i0 AND i1 AND i2;
16 END teste;
```

Dependências na compilação - (exemplo 1)

- **Entidade pk_uso4 emprega a função and3 do pacote pk_1**
- **Seqüência de compilação**:
  - declaração pacote declaração entidade
- **Corpo do pacote**: pode ser a última unidade compilada

```
 2 PACKAGE pk_1 IS    -- declaracao do pacote
 3   FUNCTION and3 (SIGNAL i0, i1, i2 : IN  BIT) RETURN BIT;
 4 END pk_1;
 5
 6 USE WORK.pk_1.ALL; -- torna todos itens do pacote pk_1 visiveis
 7 ENTITY pk_uso4 IS
 8   PORT (a, b, c : IN  BIT; s_e : OUT BIT);
 9 END pk_uso4;
10
11 ARCHITECTURE teste OF pk_uso4 IS
12 BEGIN
13   s_e  <= and3(a, b, c);
14 END teste;
15
16 PACKAGE BODY pk_1 IS  -- corpo do pacote necessario para simulacao
17   FUNCTION and3 (SIGNAL i0, i1, i2 : IN  BIT) RETURN BIT IS
18   BEGIN RETURN (i0 AND i1 AND i2);
19   END and3;
20 END pk_1;
```

Dependências na compilação - (exemplo 2)

- **Cláusula USE**: necessária a unidade que segue (veja linhas 6 e 11)

```
 2 PACKAGE pk_2 IS     -- declaracao do pacote
 3   FUNCTION and2 (SIGNAL i0, i1 : IN  BIT) RETURN BIT;
 4 END pk_2;
 5
 6 USE WORK.pk_2.ALL; -- torna itens do pacote pk_2 visiveis para pk_uso5
 7 ENTITY pk_uso5 IS
 8   PORT (a, b : IN  BIT; s_e5 : OUT BIT);
 9 END pk_uso5;
10
11 USE WORK.pk_2.ALL; -- torna itens do pacote pk_2 visiveis para pk_uso6
12 ENTITY pk_uso6 IS
13   PORT (a, b : IN  BIT; s_e6 : OUT BIT);
14 END pk_uso6;
15
16 ARCHITECTURE teste OF pk_uso5 IS
17 BEGIN  s_e5 <= and2(a, b);
18 END teste;
19
20 ARCHITECTURE teste OF pk_uso6 IS
21 BEGIN  s_e6 <= and2(a, b);
22 END teste;
23
24 PACKAGE BODY pk_2 IS  -- corpo do pacote pk_1
25   FUNCTION and2 (SIGNAL i0, i1 : IN  BIT) RETURN BIT IS
26   BEGIN RETURN (i0 AND i1);
27   END and2;
28 END pk_2;
```

Dependências na compilação - (conclusão)

- **Organização de arquivos contendo:**
  - declaração com a arquitetura (caso de entidades)
  - declaração com o corpo (caso de pacotes)

  não é a mais vantajosa para a condução de um projeto mais complexo

- **Arquivos contendo isoladamente**:
  - declaração (caso de entidades)
  - arquitetura (caso de entidades)
  - declaração (caso de pacotes)
  - corpo (caso de pacotes)

  pode ser mais vantajosa