

# **Projetos de Sistemas Digitais com VHDL - Parte I**

**VHDL**

**Modelamento Estrutural**

**Prof. Dr. Leonardo Mesquita**

**Departamento de Engenharia Elétrica**

**DEE/FE-G/UNESP**

**Versão 1.0 - 2005**

# Como desenvolver modelos VHDL para circuitos esquemáticos?

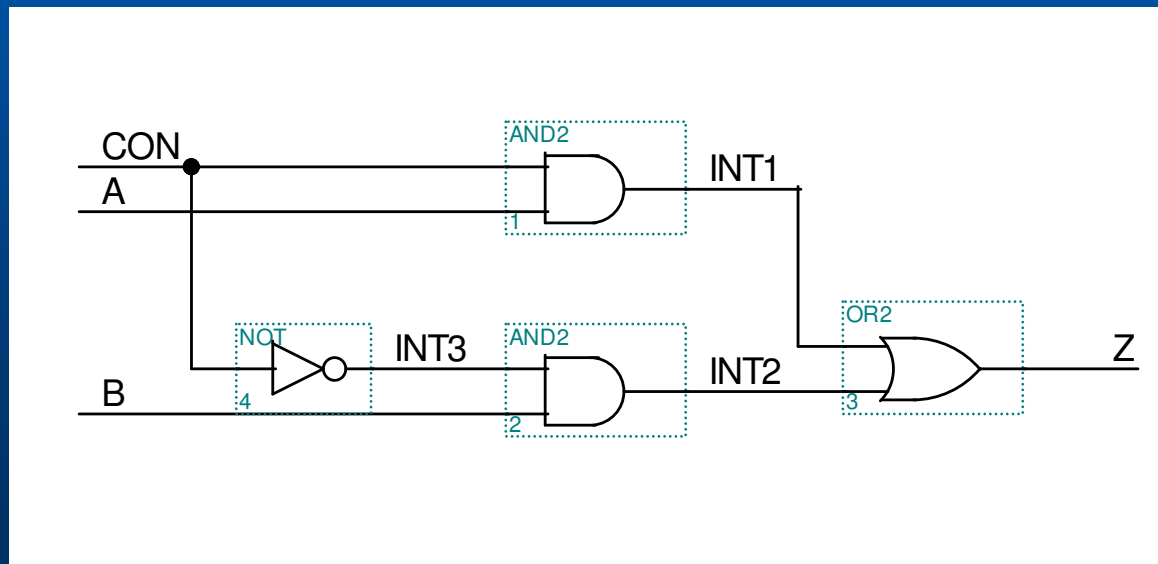
Usando o estilo de modelamento denominado Modelamento estrutural.

Define o comportamento de uma interface definindo-se os componentes que pertencem a essa interface e suas interconexões.

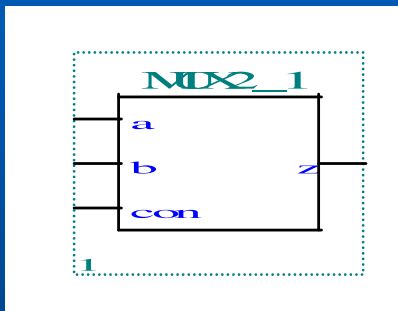
Reutilização de códigos (projetos) já desenvolvidos usando-se este estilo de modelamento.

# Exemplo 01

Analisar o circuito esquemático apresentado abaixo representando um circuito multiplexador 2X1. Observe:



## Descrição VHDL MUX 2X1



```
--circuito multiplexador 2X1 usando estilo de modelamento estrutural

--interface
entity mux2_1 is
    port(a,b,con:in bit;
          z:out bit);
end mux2_1;

--decrição
architecture structure of mux2_1 is

--declarações dos operadores lógicos
    component and2p
        port(a,b:in bit; z:out bit);
    end component;

    component or2p
        port(a,b:in bit; z:out bit);
    end component;

    component neg
        port(a:in bit; not_a:out bit);
    end component;

--sinais internos a interface
    signal int1,int2,int3: bit;

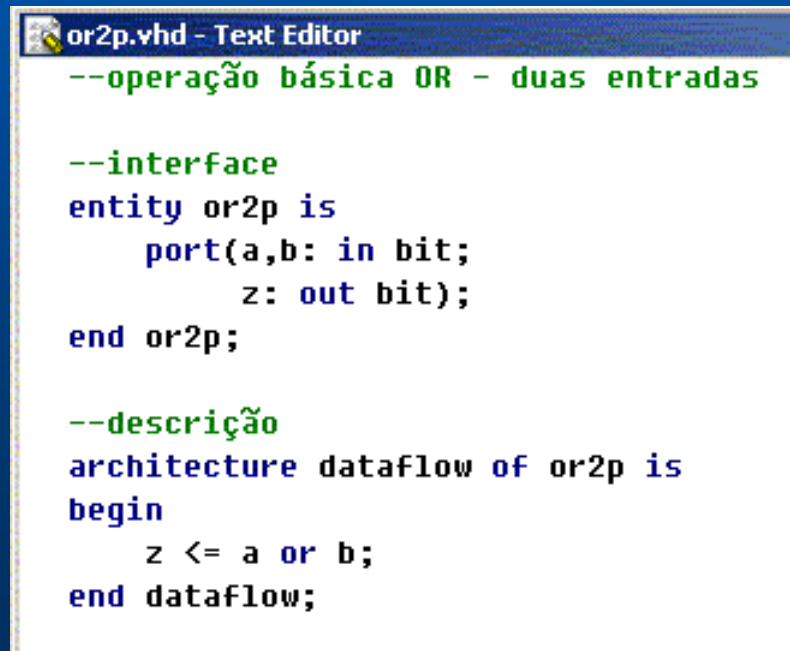
begin --operadores lógicos do circuito esquemático

    n1:neg port map(con,int3);
    a1:and2p port map(con,a,int1);
    a2:and2p port map(int3,b,int2);
    o2:or2p port map(int1,int2,z);

end structure;
```

# Considerações

Todos os componentes usados na descrição de um modelo VHDL com estilo estrutural devem ter sido criados previamente.



```
or2p.vhd - Text Editor
--operação básica OR - duas entradas

--interface
entity or2p is
    port(a,b: in bit;
          z: out bit);
end or2p;

--descrição
architecture dataflow of or2p is
begin
    z <= a or b;
end dataflow;
```

# Component Declaration

Usada para declarar todos os componentes que serão usados posteriormente na descrição do circuito lógico.

```
component <name>  
    port( ..... );  
end component;
```

component / end component são palavras reservadas do VHDL.

<name> qualquer identificador válido para VHDL.

port (...), os pinos do componente devem possuir os mesmos nomes, modelos e tipos dos previamente declarados como pinos de interface.

# Component Declaration

```
--declarações dos operadores lógicos  
component and2p  
    port(a,b:in bit; z:out bit);  
end component;
```

```
component or2p  
    port(a,b:in bit; z:out bit);  
end component;
```

```
component neg  
    port(a:in bit; not_a:out bit);  
end component;
```

Uma configuração *default* automaticamente associa componentes e *design entities* que possuem a mesma interface, ou seja, mesmo nome e pinos (portas).

# Component Instatiation Statement

Tais instruções são usadas para descrever o esquemático lógico implementado usando VHDL.

```
a1:and2p port map(con,a,int1);
```

Cada instrução posiciona um dado componente, de forma ordenada, para produzir o circuito esquemático a ser implementado com o modelo VHDL.



# Component Instatiation Statement

```
label:component_name port map ( .....);
```

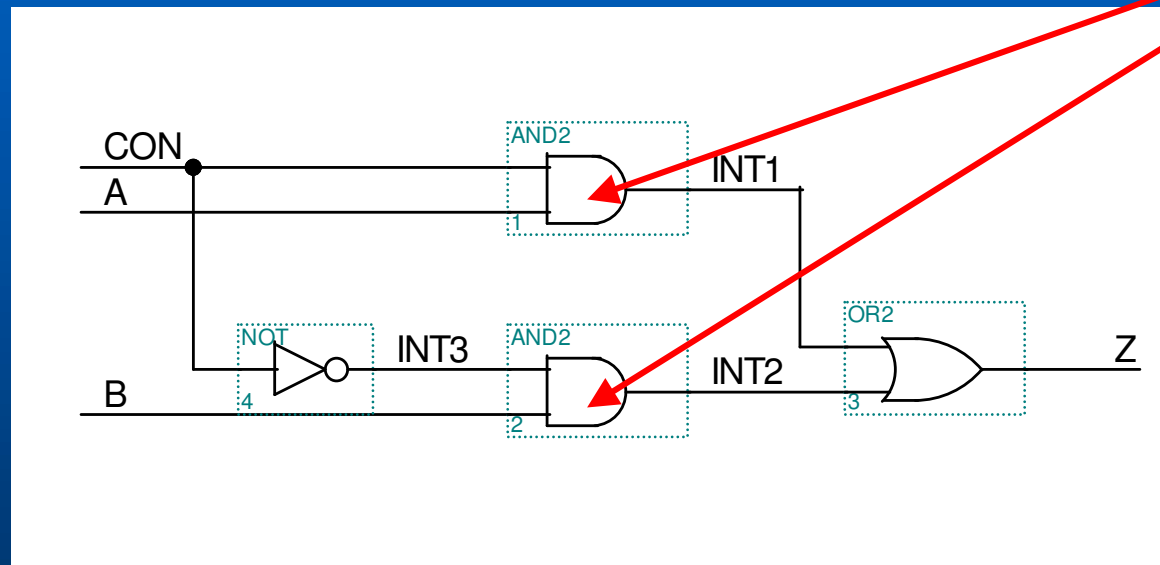
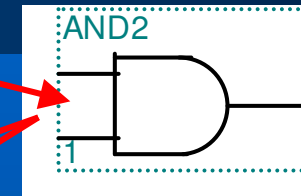
label pode ser qualquer identificador válido de VHDL.

component name nome usado para declarar uma entidade de projeto (*design entity*) previamente desenvolvido.

Port map palavra reservada da linguagem, usada para descrever como o componente virtual é conectado com o resto do sistema.

```
component and2p
```

```
    port(a,b:in bit; z:out bit);  
end component;
```



```
a1:and2p port map(con,a,int1);  
a2:and2p port map(int3,b,int2);
```

# Descrição do Componente a1

```
component and2p
  port(a,b:in bit; z:out bit);
end component;
```

```
a1:and2p port map(con,a,int1);
```



**Port map** define o mapeamento entre quais os sinais que estão conectados a quais pinos de entrada ou saída do componente.

Na instrução do componente a1 esta sendo usado uma associação denominada **associação por posição**, sendo que neste tipo de associação é válida a posição da descrição dos sinais.

Outro tipo de associação também válida em VHDL é a denominada **associação por nomes**.

Neste tipo de associação o nome do sinal é diretamente associado ao nome do pino onde o mesmo será conectado.

```
port_name => signal_name
```

Observe a descrição do componente a1 usado este tipo de associação:

```
a1:and2p port map(a=>con,b=>a,z=>int1);
```

Ambos os tipos de associação podem ser usadas em uma única instrução.

```
a1:and2p port map(con, b=>a, z=>int1);
```

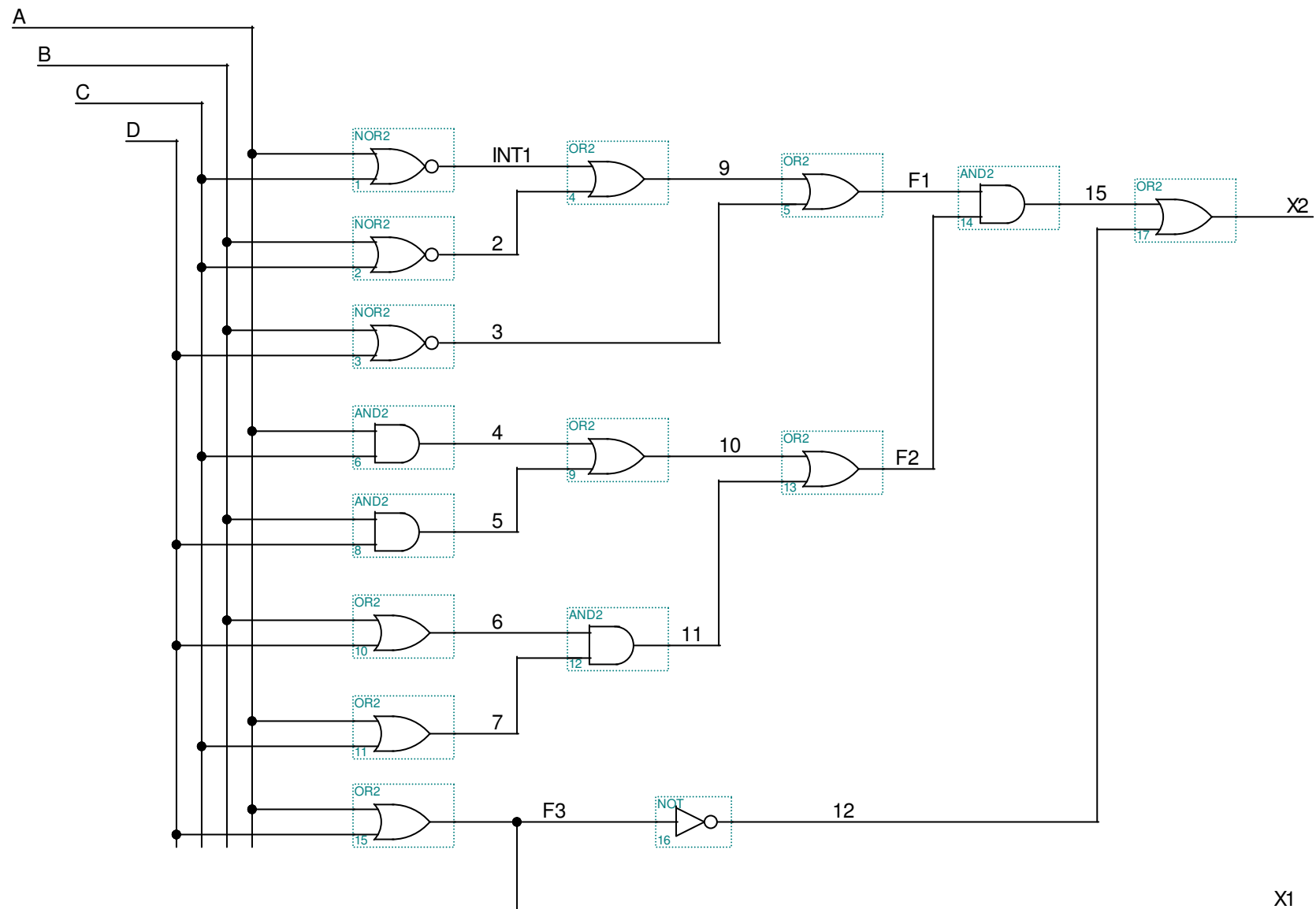
associação por posição

associação por nomes

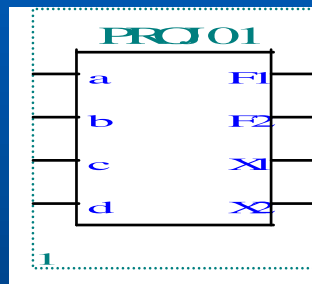
Mas sempre a associação por posição deve vir antes de qualquer associação por nomes.

A instrução de posicionamento (localização) de componentes é uma instrução concorrente.

# Exemplo 02



# Descrição Modelo VHDL



```
entity proj01 is
    port(a,b,c,d:in bit; F1,F2,X1:buffer bit; X2:out bit);
end proj01;
architecture structure of proj01 is
    signal int1,int2,int3,int4,int5,int6,int7,int9,int10,int11,int12,int15: bit;
    component and2p
        port(a,b:in bit; z:out bit);
    end component;
    component or2p
        port(a,b:in bit; z:out bit);
    end component;
    component neg
        port(a:in bit; not_a:out bit);
    end component;
    component nor2p
        port(a,b:in bit; z:out bit);
    end component;
begin
    nor1:nor2p port map(a,c,int1);
    nor2:nor2p port map(b,c,int2);
    nor3:nor2p port map(b,d,int3);
    a1:and2p port map(a,c,int4);
    a2:and2p port map(b,d,int5);
    a3:and2p port map(int6,int7,int11);
    a4:and2p port map(F1,F2,int15);
    o1:or2p port map(b,d,int6);
    o2:or2p port map(a,c,int7);
    o3:or2p port map(a,d,X1);
    o4:or2p port map(int1,int2,int9);
    o5:or2p port map(int4,int5,int10);
    o6:or2p port map(int9,int3,F1);
    o7:or2p port map(int10,int11,F2);
    o8:or2p port map(int15,int12,X2);
    n1:neg port map(X1,int12);
end structure;
```

# Estruturas Hierárquicas

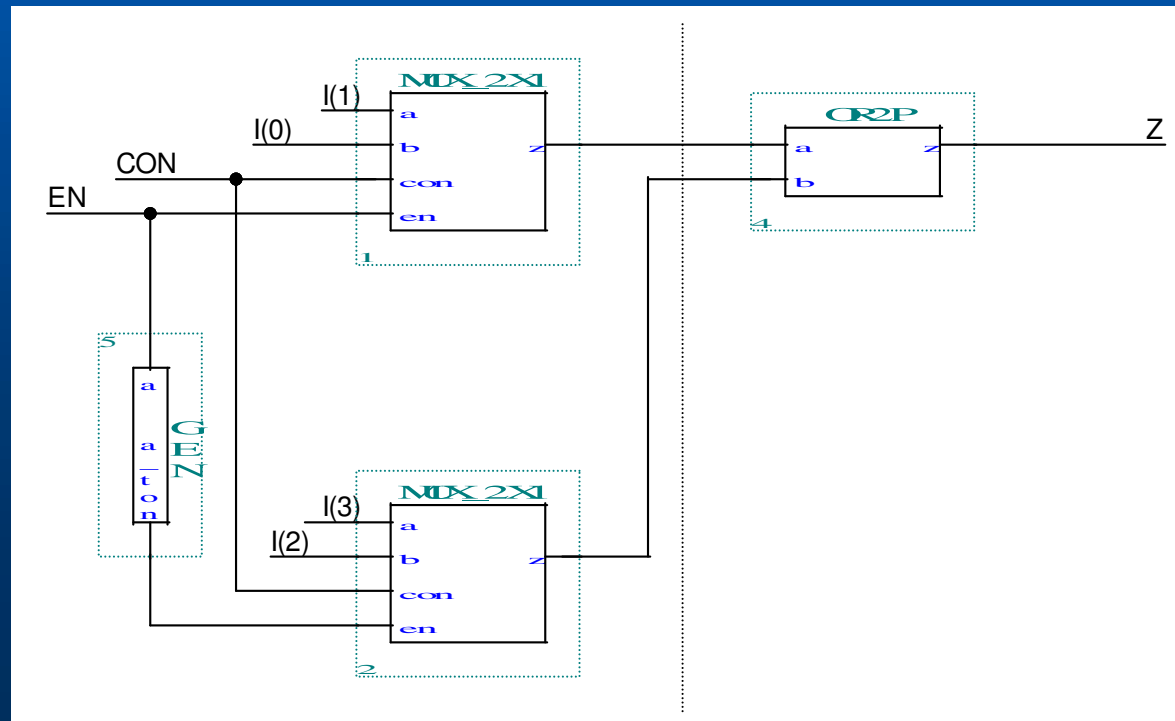
Entidades de projeto podem usar instruções de componentes, e serem posteriormente usadas para descrever um outro projeto.

A descrição estrutural hierárquica é uma poderosa ferramenta de modelamento de VHDL porque fornece um mecanismo para decompor complexos sistemas digitais em projetos menores mais simples.

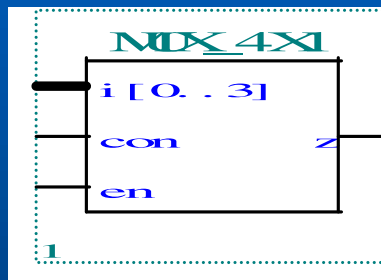


# Projeto de um MUX 4X1

O projeto foi desenvolvido a partir de um circuito multiplexador 2X1, e de portas lógicas adicionais desenvolvidas anteriormente.



# Modelo MUX 4X1



```
--circuito multiplexador 4X1 com modelo estrutural
```

```
--interface
```

```
entity mux_4X1 is
```

```
    port(i:in bit_vector(0 to 3);
```

```
          con,en:in bit;
```

```
          z:out bit);
```

```
end mux_4X1;
```

```
--descrição
```

```
architecture structure of mux_4X1 is
```

```
    signal int1,int2,int3: bit;
```

```
    component mux_2x1
```

```
        port(a,b,con,en:in bit;z:out bit);
```

```
    end component;
```

```
    component or2p
```

```
        port(a,b:in bit;z:out bit);
```

```
    end component;
```

```
    component neg
```

```
        port(a:in bit;not_a:out bit);
```

```
    end component;
```

```
begin
```

```
    mux1:mux_2x1 port map(i(1),i(0),con,en,int2);
```

```
    mux2:mux_2x1 port map(i(3),i(2),con,int1,int3);
```

```
    n1:neg port map(en,int1);
```

```
    o1:or2p port map(int2,int3,z);
```

```
end structure;
```

# Packages

Packages são usados para armazenar informação que posteriormente pode ser usada por várias entidades de projetos, tais como: declaração de componentes, alguns tipos de sinais, ....

```
package nome is
    --declaração de componente
    --declaração de sinal
    --declaração de constante
    --declaração de tipo e sub-tipo
end package;
```

# Packages

Facilita a reutilização de códigos previamente desenvolvidos.

Ajuda a descentralizar o desenvolvimento de códigos muito complexos.

Habilita o uso de várias declarações a todas as entidades de projeto que estão sendo desenvolvidas.

# Packages

Para habilitar o uso de packages dentro de um modelo VHDL o seguinte comando deve ser usado:

```
use work.nome.all;
```

Indica que todo o conteúdo do pacote está disponível para ser usado.

Biblioteca onde está localizado o package desenvolvido

# Packages

--pacote de blocos lógicos

package blocos\_logicos is

component and2p

port(a,b:in bit;z:out bit);

end component;

component or2p

port(a,b:in bit;z:out bit);

end component;

component neg

port(a:in bit; not\_a:out bit);

end component;

component nor2p

port(a,b:in bit;z:out bit);

end component;

component mux\_2x1

port(a,b,con,en:in bit;z:out bit);

end component;

end package;

# Packages

```
--uso da biblioteca blocos lógicos
```

```
use work.blocos_logicos.all;
```

```
--interface
```

```
entity proj02 is  
    port(a,b,c:in bit;  
          z:out bit);  
end proj02;
```

```
architecture structure of proj02 is  
    signal int1,int2: bit;  
begin  
    o1:or2p port map(a,b,int1);  
    a1:and2p port map(a,c,int2);  
    nor1:nor2p port map(int1,int2,z);  
end structure;
```