Pacote e componentes

Prof. MSc. André Macário Barros

22/04/2019

Pacote e Componentes

- Hoje você aprenderá a escrever códigos que podem ser reutilizados por meio de uma biblioteca particular desenvolvida por você para este propósito.
- Para tal um exemplo com multiplexadores será abordado neste material.
- Há também outro exemplo, no capítulo 19 da referência, que é um somador completo com *n* bits, construído a partir de instâncias de um somador completo de 1 bit.

Pacote e Componentes

- No Exemplo 1 deste material será apresentado um multiplexador 2x1 simples, com o propósito de utilizá-lo como componente no Exemplo 3.
- No Exemplo 2, mostraremos uma forma de se projetar um multiplexador 4x2. Esta forma será confrontada com a solução apresentada no Exemplo 3.
- E finalmente no Exemplo 3, mostraremos como construir um multiplexador 4x2 a partir de multiplexadores 2x1, reutilizando para tal o multiplexador desenvolvido no Exemplo 1 na forma de componente.

Exemplo 1
Multiplexador "2x1"

- Consideremos aqui o mesmo multiplexador "2x1" descrito no material de sinal *versus* variável que opere de acordo com a Figura 1 e a Tabela 1.

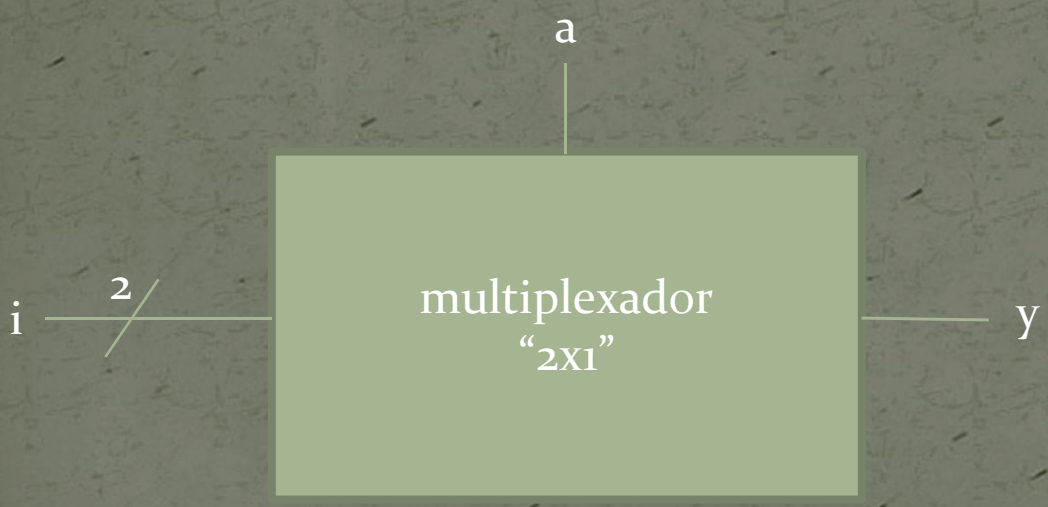

Figura 1

Tabela 1

a	*y*
0	i_0
1	i_1

Exemplo 1
Multiplexador "2x1"

- O código vhdl correspondente ao mux 2x1 está apresentado no slide a seguir.
- Como já comentado no material de generic e for/generate:
  - A linha comentada é através da soma padrão de mintermos (linha 10)
  - E as linhas 11 e 12 apresentam a função lógica da saída com declaração condicional concorrente, no caso when/else.

Exemplo 1 – Multiplexador "2x1"

```
01-library ieee;
02-use ieee.std_logic_1164.all;
03-entity ccto is
04-port(a: in std_logic;
05-     i: in std_logic_vector(1 downto 0);
06-    y: out std_logic);
07-end entity;
08-architecture arq of ccto is
09-begin
10---y <= (not(a) and i(0)) or (a and i(1));
11-  y <= i(0) when a='0' else
12-       i(1);
13-end arq;
```

Exemplo 2 – Multiplexador "4x2"

- Consideremos agora um multiplexador "4x2" que opere de acordo com a Figura 2 e a Tabela 2.

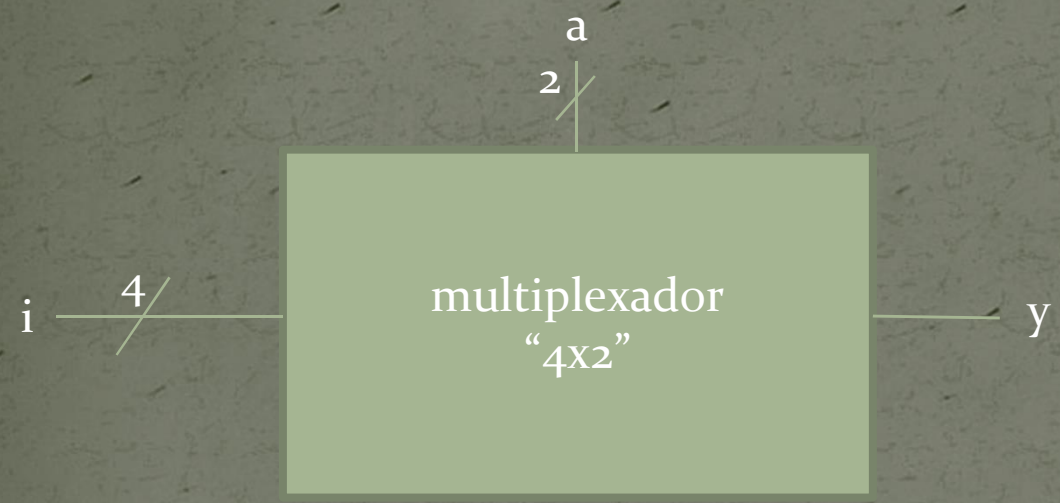

Figura 2

Tabela 2

a_1	a_0	y
0	0	i_0
0	1	i_1
1	0	i_2
1	1	i_3

Exemplo 2 – Multiplexador "4x2"

- Utilizando as declarações condicionais concorrentes, o código VHDL correspondente ao mux "4x2" está apresentado a seguir.

Exemplo 2 – Multiplexador "4x2"

```
01-library ieee;
02-use ieee.std_logic_1164.all;
03-entity mux_4x2 is
04-   port(a: in std_logic_vector(1 downto 0);
05-        i: in std_logic_vector(3 downto 0);
06-        y: out std_logic);
07-end entity;
08-architecture arq of mux_4x2 is
09-begin
10-y <= i(0) when a = "00" else
11-     i(1) when a = "01" else
12-     i(2) when a = "10" else
13-     i(3);
14-end arq;
```

Caminhos a partir daqui

- É possível, por meio da declaração generic, construirmos um multiplexador $2^n \times n$
  - Convidamos você a descobrir como
- Porém, vamos utilizar um outro conceito baseado em reutilização de código para demonstrarmos o uso de pacote e componentes.

Exemplo 3 – Mux "4x2" com mux "2x1" como componente

- A partir do mux "2x1" do Exemplo 1, podemos construir o sistema da Figura 3 para obtermos um mux "4x2", que cumpra com os estados da Tabela 2 (Exemplo 2).

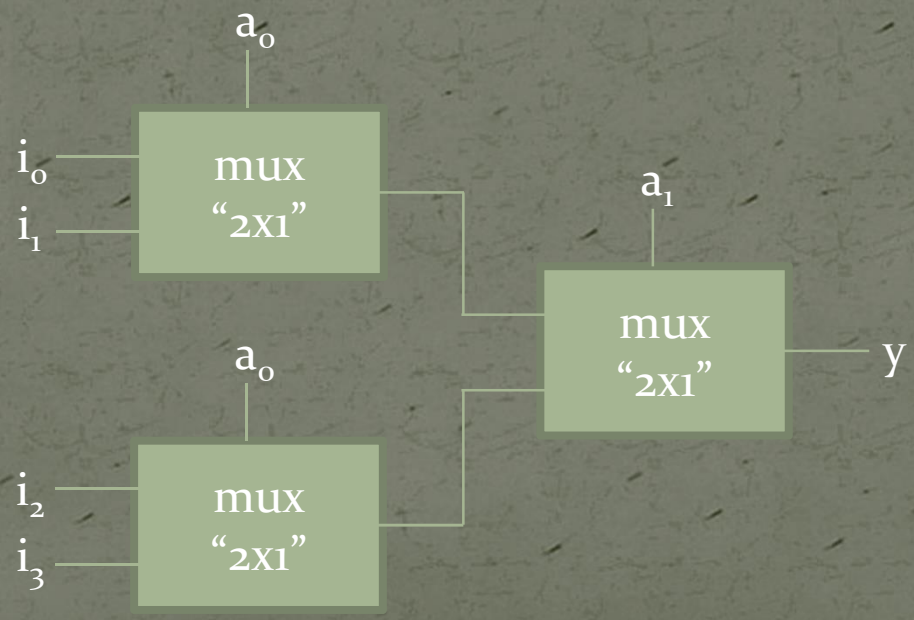

Figura 3

Exemplo 3 – Mux "4x2" com mux "2x1" como componente

- Em geral, a estrutura de um projeto que utilize pacote e componentes apresenta os arquivos da Figura 4.

meu_pacote.vhd
(esta é a sua biblioteca)

definição componente 1

definição componente 2
...
definição componente *n*

top.vhd
(*a top level entity*)

mapeamento componente 1

mapeamento componente 2
...
mapeamento componente *n*

componente_1.vhd
entity
+
architecture

componente_2.vhd
entity
+
architecture

componente_n.vhd
entity
+
architecture

Figura 4

Exemplo 3 – Mux "4x2" com mux "2x1" como componente

- Os arquivos apresentados na Figura 4 são descritos a seguir:
  - Os arquivos componente_1.vhd, componente_2.vhd, ..., componente_n.vhd, são os arquivos de cada sub-circuito que queremos que façam parte do circuito principal. Um sub-circuito, em nosso exemplo será o mux "2x1", composto por sua entity e sua architecture.
  - top.vhd é a "top level entity". É o local onde "desenhamos" o circuito principal, com seus sub-circuitos mapeados, chamados aqui de componentes.

Exemplo 3 – Mux "4x2" com mux "2x1" como componente

- Arquivos da Figura 4 (cont.):
  - E finalmente meu_pacote.vhd é a biblioteca que conterá as definições de cada componente.
    - A definição de um componente nada mais é do que a seção da entity de um circuito, porém com a palavra entity substituída por component

Exemplo 3 – Mux "4x2" com mux "2x1" como componente

- Iniciando então nossa montagem de nosso arquivo de projeto, iniciaremos pelos arquivos de componente.
- No nosso exemplo, o único componente que utilizaremos será o mux "2x1"
- Para inseri-lo em nosso projeto temos três caminhos:
  - Copiamos um projeto (File → copy project) que contenha o código vhd de um mux "2x1" (Exemplo 1)
  - Inserimos o componente em um projeto (Project → Add File)
  - Ou criamos um mux "2x1" do zero.
- Qualquer opção escolhida é necessário que tenhamos a prévia garantia de que seja um mux "2x1" correto

Exemplo 3 – Mux "4x2" com mux "2x1" como componente

- Em seguida, criaremos um arquivo de pacotes:
  - Insira um novo arquivo através da opção
    - Project → New Source → VHDL Package
    - Nomeie o arquivo como meu_pacote.vhd
- Na tela de edição com o novo arquivo de pacotes, selecione todo o arquivo pré-preenchido, delete-o e digite o código apresentado no slide a seguir

Exemplo 3 – Arquivo meu_pacote.vhd

```
01-library ieee;
02-use ieee.std_logic_1164.all;
03-package meu_pacote is
04-    component mux_2x1 is
05-        port(a: std_logic_vector(1 downto 0);
06-             i: std_logic_vector(3 downto 0);
07-             y: out std_logic);
08-    end component;
09-end package;
```

Exemplo 3 – Arquivo meu_pacote.vhd

- Linhas 03 e 09: **`package xxx is...`** e **`end package`** são os delimitadores do escopo dentro do qual as definições de cada componente serão feitas.
  - Observe que o arquivo de pacote contém outra parte, porém só iremos utilizá-la após aprendermos circuitos sequenciais

Exemplo 3 – Arquivo meu_pacote.vhd

- Linhas 04 a 08: é o local onde colocamos a definição de nosso componente.
  - Observe que a definição nada mais é do que a entity do componente, com a substituição da palavra entity por component
- Caso seu arquivo de projeto contenha mais componentes, isto é, mais sub-circuitos a serem mapeados, também insira suas respectivas definições dentro do escopo do pacote, isto é, entre as linhas 03 e 09 do arquivo meu_pacote.vhd.

Exemplo 3 – Arquivo top.vhd

- Para criarmos o arquivo top.vhd, vamos inicialmente redesenhar na Figura 5 o circuito da Figura 3 com sinais, s_0 e s_1, interligando os componentes.

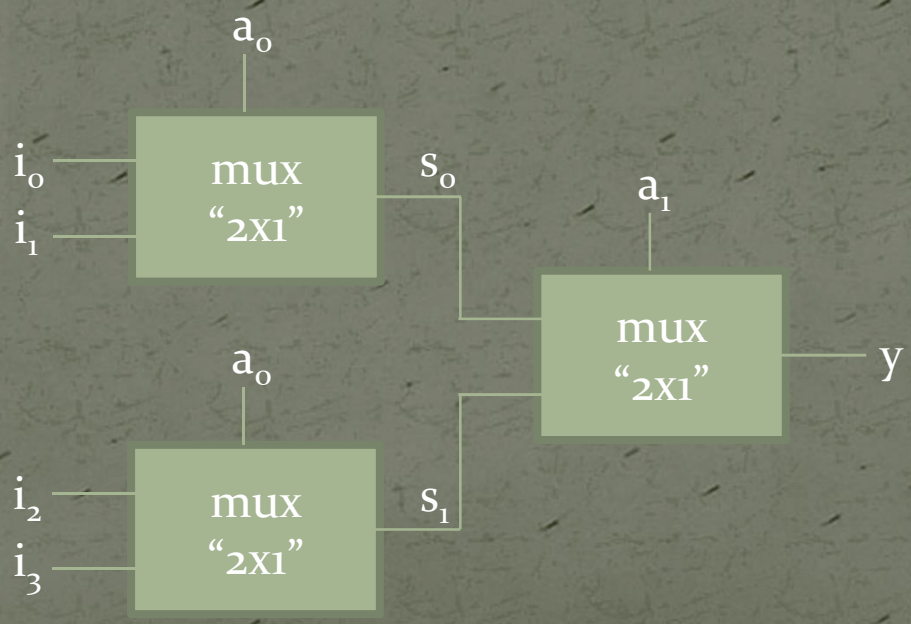

Figura 5

Exemplo 3 – Arquivo top.vhd

- Com base na Figura 5, podemos agora apresentar o arquivo top.vhd a seguir que conterá este sistema interconectado através de três mapeamentos.

Exemplo 3
Arquivo top.vhd

```
01-library ieee;
02-use ieee.std_logic_1164.all;
03-use work.meu_pacote.all;
04-entity top is
05-port(a: in std_logic_vector(1 downto 0);
06-     i: in std_logic_vector(3 downto 0);
07-     y: out std_logic);
08-end entity;
09-architecture arq of top is
10-   signal s: std_logic_vector(1 downto 0);
11-begin
12-   map_1: entity work.mux_2x1(arq)
13-          port map(a(0), i(1 downto 0), s(0));
14-   map_2: entity work.mux_2x1(arq)
15-          port map(a(0), i(3 downto 2), s(1));
16-   map_3: entity work.mux_2x1(arq)
17-          port map(a(1), s, y);
18-end arq;
```

Exemplo 3 – Arquivo top.vhd

- Linha 03: esta linha é necessária para que a top level entity localize os códigos vhdl de cada um dos componentes mapeados, ou seja, é nesta linha em que a sua biblioteca é incluída no projeto.
- Linha 10: nesta linha o vetor **s** com duas posições é declarado para os sinais $\mathbf{s}_0$ e $\mathbf{s}_1$ da Figura 5. A opção por vetor será explicada no terceiro mapeamento.

Exemplo 3 – Arquivo top.vhd

- Linha 12 e 13: é o mapeamento para as entradas $\mathbf{i_0}$ e $\mathbf{i_1}$ da Figura 5. Observe na Figura 5 que o mux "2x1":
  - recebe as entradas $\mathbf{i_0}$ e $\mathbf{i_1}$ por meio das duas primeiras posições do vetor de entrada **i** de quatro posições
  - fornece a saída para a posição zero do sinal s a depender de a_0.
- Linhas 14 e 15: é o mapeamento análogo ao anterior, porém destinado às entradas $\mathbf{i_2}$ e $\mathbf{i_3}$, fornecendo a saída para a posição um do sinal **s**, também dependendo de a_0.
- Linhas 16 e 17: é o mapeamento para o componente de saída, que recebe o resultado da multiplexação dos dois muxes anteriores feita por $\mathbf{a_0}$ e, por meio de $\mathbf{a_1}$, seleciona qual entrada será destinada à saída **y**.

Exemplo 3 – Arquivo top.vhd

- Linhas 16 e 17: é o mapeamento para o componente de saída, que recebe o resultado da multiplexação dos dois muxes anteriores feita por $\mathbf{a}_0$ e, por meio de $\mathbf{a}_1$, seleciona qual entrada será destinada à saída **y**.
  - Observe que as duas posições do sinal s foram usadas como entradas do mux "2x1" de saída. E foi por este motivo optou-se por trabalhar com o sinal s na forma de vetor.

Exemplo da referência

- Consulte também o exemplo do full-adder genérico presente no capítulo 19 da referência a fim de que você possa compreender melhor como utilizar o arquivo de pacote e componentes.

Referências

- Volnei Pedroni. **Eletrônica digital moderna e VHDL**. Elsevier, Rio de Janeiro, 2010.
  - Capítulos 19, 20 e 21
  - Há 12 exemplares na biblioteca
  - Número de chamada: 621.392 P372e