

Declarações generic e for/generate

Prof. MSc. André Macário Barros

15/04/2019

Declarações generic e for/generate

- Hoje você aprenderá a escrever códigos que requeiram o emprego de vetores de maneira flexível através das declarações generic e for/generate.
- Para tal um exemplo será abordado neste material.
- Há também outro exemplo, no capítulo 19 da referência, que é um circuito detector de paridade

Exemplo 1

Decodificador “1-hot” de 2 bits

- Consideremos aqui um decodificador “1-hot” de 2 bits que opere de acordo com a Figura 1 e a Tabela 1.



Figura 1

Tabela 1

x_1	x_0	y_3	y_2	y_1	y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Exemplo 1

Decodificador “1-hot” de 2 bits

- Utilizando as declarações condicionais concorrentes podemos escrever o código vhdl a seguir.

Exemplo 1

Decodificador “1-hot” de 2 bits

- 1 ENTITY address_decoder IS
- 3 PORT (x: IN BIT_VECTOR(1 DOWNTO 0);
- 4 y: OUT BIT_VECTOR(3 DOWNTO 0));
- 5 END address_decoder;
- 6 -----
- 7 ARCHITECTURE address_decoder OF address_decoder IS
- 8 BEGIN
- 9 y <= "0001" WHEN x = "00" ELSE
- 10 "0010" WHEN x = "01" ELSE
- 11 "0100" WHEN x = "10" ELSE
- 12 "1000";
- 17 END address_decoder;

Exemplo 2

Decodificador “1-hot” de 3 bits

- Consideremos agora um decodificador “1-hot” de 3 bits que opere de acordo com a Figura 2 e a Tabela 2.



Figura 2

Tabela 2

x_2	x_1	x_0	y_7	y_6	y_5	y_4	y_3	y_2	y_1	y_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
...										
1	1	1	1	0	0	0	0	0	0	0

Exemplo 2

Decodificador “1-hot” de 3 bits

- Também utilizando as declarações condicionais concorrentes podemos escrever o código vhdl a seguir.

Exemplo 2

Decodificador “1-hot” de 3 bits

```
• 1  ENTITY address_decoder IS
• 3      PORT (x: IN BIT_VECTOR(2 DOWNTO 0);
• 4            y: OUT BIT_VECTOR(7 DOWNTO 0));
• 5  END address_decoder;
• 6  -----
• 7  ARCHITECTURE address_decoder OF address_decoder IS
• 8  BEGIN
• 9      y <= "00000001" WHEN x = "000" ELSE
•10          "00000010" WHEN x = "001" ELSE
•11          "00000100" WHEN x = "010" ELSE
•12          "00001000" WHEN x = "011" ELSE
•13          "00010000" WHEN x = "100" ELSE
•14          "00100000" WHEN x = "101" ELSE
•15          "01000000" WHEN x = "110" ELSE
•16          "10000000";
•17 END address_decoder;
```


Exemplo 3

Decodificador genérico “1-hot” de n bits

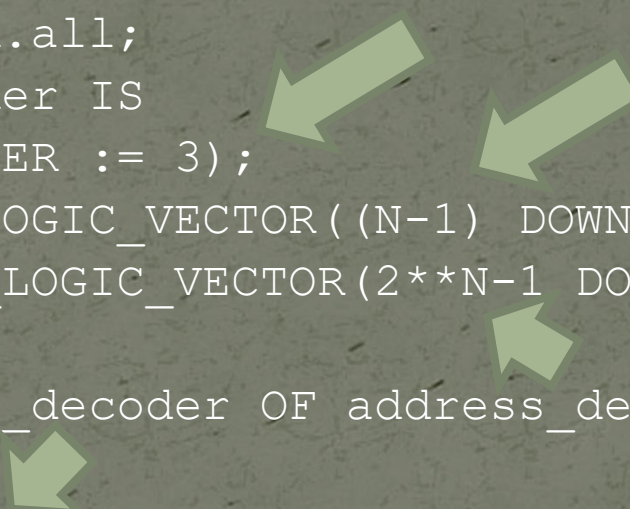
- Facilmente conseguimos constatar que, para um codificador com muitos bits o trabalho será grande caso a mesma lógica seja utilizada.
- E é para um cenário como estes que demonstraremos a utilidade das declarações generic e for/generate. Acompanhe como ao desenvolvermos um decodificador 1-hot genérico para n bits (Figura 3)



Figura 3

Exemplo 3

Decodificador “1-hot” genérico de n bits

- 01-library ieee;
 - 02-use ieee.std_logic_1164.all;
 - 03-use ieee.numeric_std.all;
 - 04-ENTITY address_decoder IS
 - 05- GENERIC (N: INTEGER := 3);
 - 06- PORT (x: IN STD_LOGIC_VECTOR((N-1) DOWNTO 0);
 - 07- y: OUT STD_LOGIC_VECTOR(2**N-1 DOWNTO 0));
 - 08-END address_decoder;
 - 09-ARCHITECTURE address_decoder OF address_decoder IS
 - 10-BEGIN
 - 11- gen: FOR i IN y'range GENERATE
 - 12- y(i) <= '1' WHEN
 - 13- ((std_logic_vector(to_unsigned(i, N))) = x)
 - 14- ELSE '0';
 - 15- END GENERATE;
 - 16-END address_decoder;
- 

Exemplo 3

Decodificador genérico “1-hot” de n bits

- Linha 3: é o local dentro da entity onde colocamos a declaração generic para criarmos as constantes com as quais serão trabalhadas na architecture de nosso código. Atente para colocar o generic antes do port.
- Linhas 6 e 7: observe que agora tanto a entrada x quanto a saída y podem ser expressas em um formato que independa do número de bits, pois os vetores são criados a partir da constante N .

Exemplo 3

Decodificador genérico “1-hot” de n bits

- Linhas 11 a 15: é o escopo dentro do qual poderemos escrever uma estrutura de repetição concorrente por meio da declaração `for/generate`. Observe que a sintaxe é bem parecida com a do `for` usado nos loops dos *testbenches*, porém com a palavra-chave `generate` em seu lugar e com um *label* igual ao usado nos mapeamentos dos *testbenches*. Portanto **gen** é um identificador qualquer podendo o mesmo ser substituído por qualquer outro desde que esteja dentro das normas do VHDL (até 26 caracteres, não começar por número, etc).

Exemplo 3

Decodificador genérico “1-hot” de n bits


- Linha 11: no capítulo 19 da referência há uma lista de atributos que podemos usar quando estamos trabalhando com vetores. O atributo `y'range` é um destes.
 - Para entender o **`y'range`**, experimente trocar a linha 11 pela linha a seguir:
 - `gen: FOR i IN 0 to 2**N-1 GENERATE`
 - Você observará que o comportamento do circuito será o mesmo

Exemplo 3

Decodificador genérico “1-hot” de n bits

- O código de testbench pode também ser escrito de uma forma flexível, também utilizando o conceito de generic, conforme pode ser visto no código de testbench a seguir.

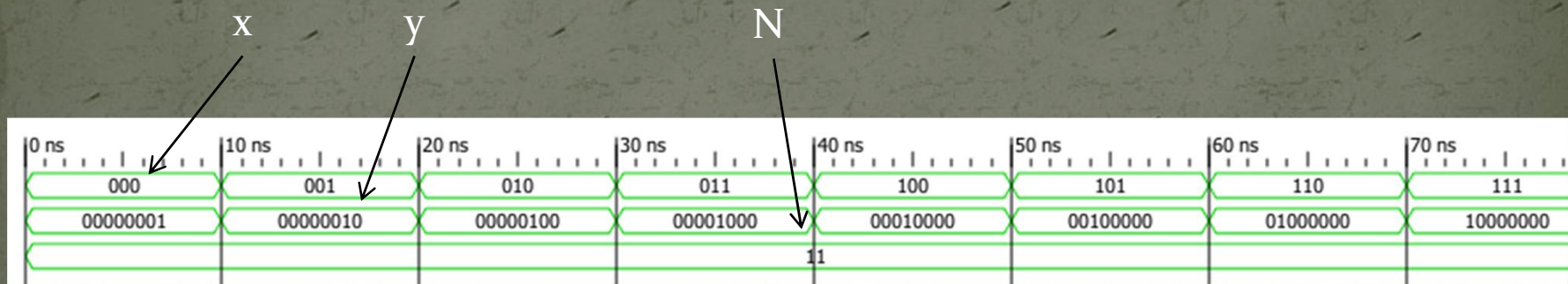
Exemplo 3: *testbench* do decodificador “1-hot” genérico de n bits



```
01-library ieee;
02-use ieee.std_logic_1164.all;
03-use ieee.numeric_std.all;
04-entity tb_ccto is
05-    generic(N: integer := 3);
06-end tb_ccto;
07-architecture arq of tb_ccto is
08-    signal tx: std_logic_vector((N-1) downto 0);
09-    signal ty: std_logic_vector((2**N - 1) downto 0);
10-begin
11-    uut: entity work.address_decoder(address_decoder)
12-        port map(tx, ty);
13-    process
14-        variable i: integer;
15-    begin
16-        for i in 0 to 2**N-1 loop
17-            tx <= std_logic_vector(to_unsigned(i, N));
18-            wait for 10 ns;
19-        end loop;
20-    end process;
21-end arq;
```

Exemplo 3

Decodificador genérico “1-hot” de n bits



Exercício: Detector de Paridade Par (ou Gerador de Paridade Ímpar)

- Desenvolva, simule e implemente um circuito detector de paridade par em um vetor com comprimento variável.
 - Para tal você necessitará das declarações generic e também da for/generate
 - Por exemplo, sua saída y deverá produzir '1' se o número de '1's existente no vetor de entrada for par. E, por consequência, a saída produzirá '0', caso o número de '1's existentes no vetor de entrada for ímpar.

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