



# دوره مقدماتی آشنایی با FPGA و VHDL

COMPONENT و PACKAGE

محمدرضا عزیزی

امیرعلی ابراهیمی

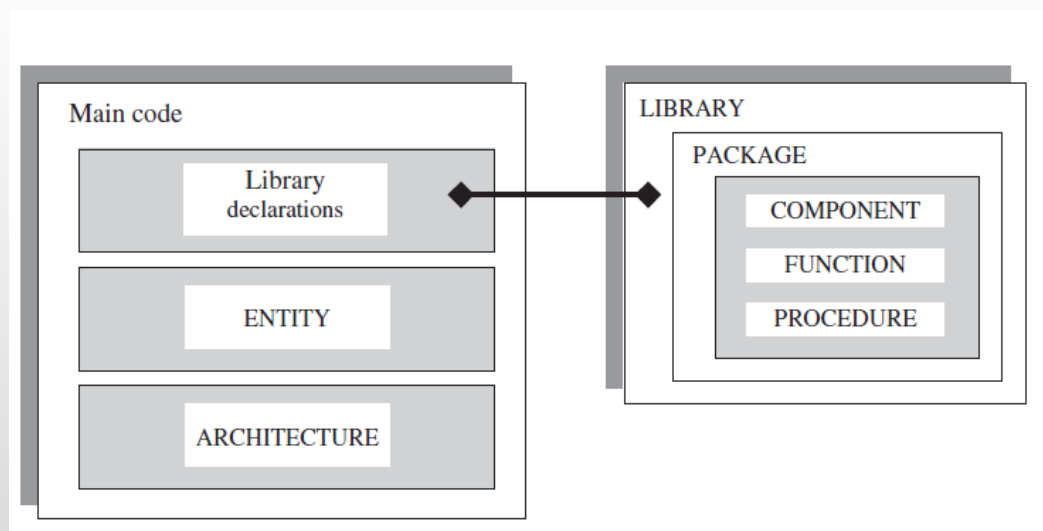
بهار ۱۳۹۷

## مقدمه (۱)

➤ در بخش اول درس، تمامی پس‌زمینه و تکنیک‌های کد VHDL را فراگرفتیم که شامل موارد زیر بودند:

- ساختار کد: فراخوانی کتابخانه‌ها، entity و architecture (فصل ۲)
- انواع داده (فصل ۳)
- عملگرها و صفت‌ها (فصل ۴)
- کد همروند (فصل ۵)
- کد ترتیبی (فصل ۶)
- ماشین‌حالت (فصل ۷)

➤ تمامی جزئیات کد سمت چپ



## مقدمه (۲)

➤ در این بخش، بخش دوم درس، بلوک‌های سازنده دیگری معرفی خواهد شد که در ساختن یک کتابخانه نقش دارند، که عبارتند از:

- Package ها (فصل ۸)
- Component ها (فصل ۸)
- Function ها (فصل ۹)
- Procedure ها (فصل ۹)

➤ این اجزاء جدید می‌توانند در داخل کد اصلی قرار گیرند (سمت چپ شکل) ولی از آنجایی که هدف استفاده از این اجزاء، قابلیت استفاده مجدد از کد و به اشتراک گذاری کد است، بهتر است که در داخل یک کتابخانه استفاده شوند. همچنین قابلیت پارتیشن‌بندی کد نیز در این حالت وجود دارد. (code partitioning)

➤ به طور خلاصه، قطعه کدهای پراستفاده در داخل FUNCTION، COMPONENT یا PROCEDURE نوشته شده، سپس داخل PACKAGE قرار گرفته و در نهایت داخل LIBRARY قرار می‌گیرند.

## (۱) PACKAGE

➤ قطعه کدهای پر استفاده در داخل FUNCTION، COMPONENT یا PROCEDURE نوشته شده، سپس داخل PACKAGE قرار گرفته و در نهایت داخل LIBRARY قرار می گیرند.

➤ اهمیت این تکنیک:

- پارتیشن بندی کردن کد
- به اشتراک گذاشتن کد
- استفاده مجدد از کد

➤ ساختار:

```
PACKAGE package_name IS
    (declarations)
END package_name;

[PACKAGE BODY package_name IS
    (FUNCTION and PROCEDURE descriptions)
END package_name;]
```

## (۲) PACKAGE

➤ بخش declarations:

■ COMPONENT

■ FUNCTION

■ PROCEDURE

■ TYPE

■ CONSTANT

➤ بخش body:

■ FUNCTION تعریف

■ PROCEDURE تعریف

## (۳) PACKAGE

مثال: ➤

```
1  -----
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  -----
5  PACKAGE my_package IS
6      TYPE state IS (st1, st2, st3, st4);
7      TYPE color IS (red, green, blue);
8      CONSTANT vec: STD_LOGIC_VECTOR(7 DOWNTO 0) := "11111111";
9  END my_package;
10 -----
```

## (۴) PACKAGE

مثال: ➤

```
1  -----
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  -----
5  PACKAGE my_package IS
6      TYPE state IS (st1, st2, st3, st4);
7      TYPE color IS (red, green, blue);
8      CONSTANT vec: STD_LOGIC_VECTOR(7 DOWNT0 0) := "11111111";
9      FUNCTION positive_edge(SIGNAL s: STD_LOGIC) RETURN BOOLEAN;
10 END my_package;
11 -----
12 PACKAGE BODY my_package IS
13     FUNCTION positive_edge(SIGNAL s: STD_LOGIC) RETURN BOOLEAN IS
14     BEGIN
15         RETURN (s'EVENT AND s='1');
16     END positive_edge;
17 END my_package;
18 -----
```

## (۵) PACKAGE

➤ پکیج‌های نوشته شده در دو مثال قبل، بعد از کامپایل، بخشی از کتابخانه *work* می‌شوند و نحوه استفاده از آن‌ها به این صورت است:

```
-----  
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
USE work.my_package.all;
```

```
-----  
ENTITY...
```

```
...
```

```
ARCHITECTURE...
```

```
...  
-----
```



## (۱) COMPONENT

- COMPONENT در واقع یک سختافزار طراحی شده به زبان VHDL است که شامل بخش‌های فراخوانی کتابخانه‌ها، ENTITY و ARCHITECTURE است.
- یک COMPONENT می‌تواند داخل یک کد دیگر قرار گیرد و به این ترتیب، طراحی مرتبه‌ای (hierarchical) به وجود آید.
- COMPONENT روش دیگری برای پارتیشن‌بندی کردن کد است.
- برای استفاده از یک COMPONENT ابتدا باید آن را فراخوانی کرد (COMPONENT declaration) و سپس از آن instance گرفت (COMPONENT instantiation).

## (۲) COMPONENT

➤ ساختار فراخوانی COMPONENT: (declaration)

```
COMPONENT component_name IS
  PORT (
    port_name : signal_mode signal_type;
    port_name : signal_mode signal_type;
    ...);
END COMPONENT;
```

➤ ساختار نمونه گرفتن از COMPONENT: (instantiation)

```
label: component_name PORT MAP (port_list);
```

▪ port\_list: لیست پورت‌های مدار اصلی که به پورت‌های مدار از پیش طراحی شده متصل می‌شوند.

## (۳) COMPONENT

➤ مثال: فرض کنید یک مدار معکوس کننده طراحی کرده ایم و می خواهیم از آن در یک طراحی دیگر استفاده کنیم.

----- COMPONENT declaration: -----

```
COMPONENT inverter IS
```

```
    PORT (a: IN STD_LOGIC; b: OUT STD_LOGIC);
```

```
END COMPONENT;
```

----- COMPONENT instantiation: -----

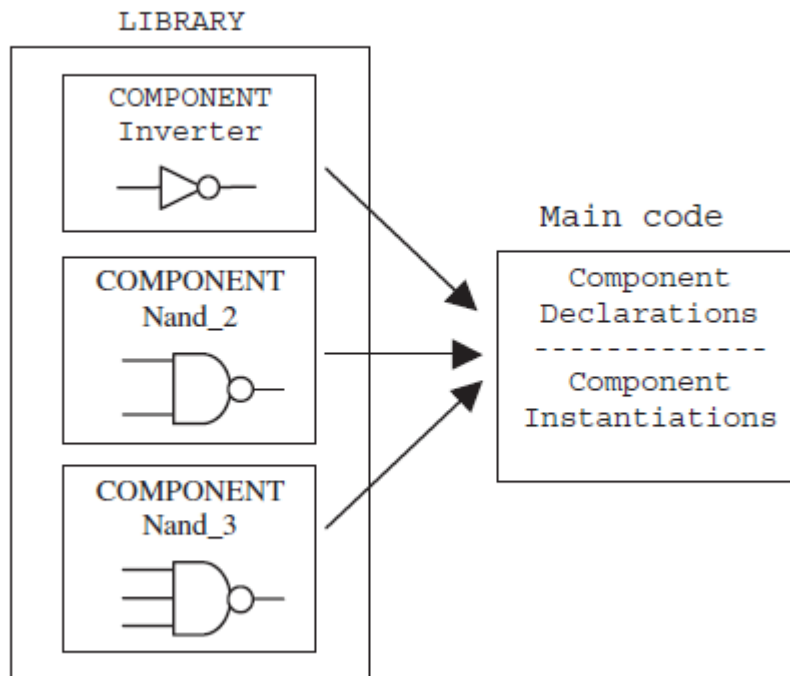
```
U1: inverter PORT MAP (x, y);
```

## (۴) COMPONENT

➤ روش اول استفاده از COMPONENT:

▪ فراخوانی و نمونه‌گیری در کد اصلی

➤ مثال:



## (۵) COMPONENT

```
1 ----- File inverter.vhd: -----  
2 LIBRARY ieee;  
3 USE ieee.std_logic_1164.all;  
4 -----  
5 ENTITY inverter IS  
6     PORT (a: IN STD_LOGIC; b: OUT STD_LOGIC);  
7 END inverter;  
8 -----  
9 ARCHITECTURE inverter OF inverter IS  
10 BEGIN  
11     b <= NOT a;  
12 END inverter;  
13 -----
```

## (۶) COMPONENT

```
1 ----- File nand_2.vhd: -----  
2 LIBRARY ieee;  
3 USE ieee.std_logic_1164.all;  
4 -----  
5 ENTITY nand_2 IS  
6     PORT (a, b: IN STD_LOGIC; c: OUT STD_LOGIC);  
7 END nand_2;  
8 -----  
9 ARCHITECTURE nand_2 OF nand_2 IS  
10 BEGIN  
11     c <= NOT (a AND b);  
12 END nand_2;  
13 -----
```

## (V) COMPONENT

```
1 ----- File nand_3.vhd: -----  
2 LIBRARY ieee;  
3 USE ieee.std_logic_1164.all;  
4 -----  
5 ENTITY nand_3 IS  
6     PORT (a, b, c: IN STD_LOGIC; d: OUT STD_LOGIC);  
7 END nand_3;  
8 -----  
9 ARCHITECTURE nand_3 OF nand_3 IS  
10 BEGIN  
11     d <= NOT (a AND b AND c);  
12 END nand_3;  
13 -----
```

## (A) COMPONENT

```
1 ----- File project.vhd: -----
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 -----
5 ENTITY project IS
6     PORT (a, b, c, d: IN STD_LOGIC;
7           x, y: OUT STD_LOGIC);
8 END project;
9 -----
10 ARCHITECTURE structural OF project IS
11 -----
12     COMPONENT inverter IS
13         PORT (a: IN STD_LOGIC; b: OUT STD_LOGIC);
14     END COMPONENT;
15 -----
16     COMPONENT nand_2 IS
17         PORT (a, b: IN STD_LOGIC; c: OUT STD_LOGIC);
18     END COMPONENT;
19 -----
20     COMPONENT nand_3 IS
21         PORT (a, b, c: IN STD_LOGIC; d: OUT STD_LOGIC);
22     END COMPONENT;
23 -----
24     SIGNAL w: STD_LOGIC;
25 BEGIN
26     U1: inverter PORT MAP (b, w);
27     U2: nand_2 PORT MAP (a, b, x);
28     U3: nand_3 PORT MAP (w, c, d, y);
29 END structural;
30 -----
```

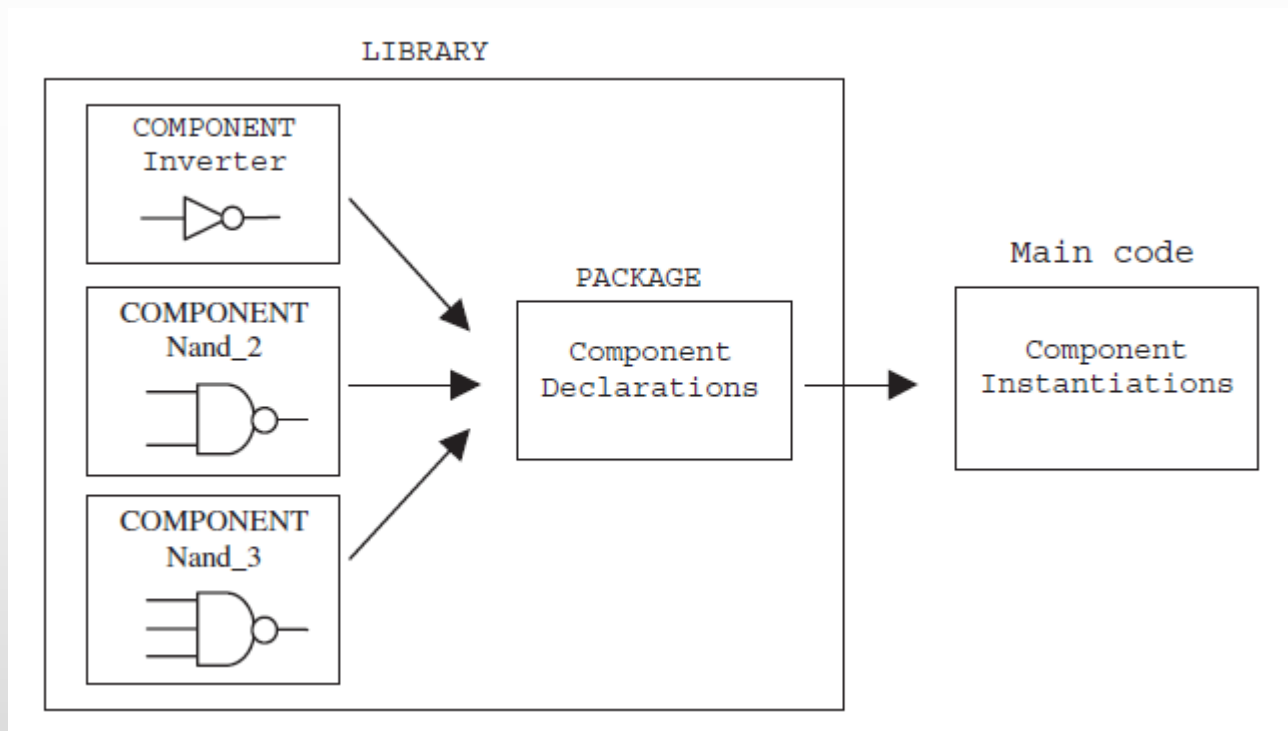


## (۹) COMPONENT

➤ روش دوم استفاده از COMPONENT:

▪ فراخوانی در یک پکیج

➤ مثال:



## (۱۰) COMPONENT

```
1 ----- File inverter.vhd: -----  
2 LIBRARY ieee;  
3 USE ieee.std_logic_1164.all;  
4 -----  
5 ENTITY inverter IS  
6     PORT (a: IN STD_LOGIC; b: OUT STD_LOGIC);  
7 END inverter;  
8 -----  
9 ARCHITECTURE inverter OF inverter IS  
10 BEGIN  
11     b <= NOT a;  
12 END inverter;  
13 -----
```

## (۱۱) COMPONENT

```
1 ----- File nand_2.vhd: -----
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 -----
5 ENTITY nand_2 IS
6     PORT (a, b: IN STD_LOGIC; c: OUT STD_LOGIC);
7 END nand_2;
8 -----
9 ARCHITECTURE nand_2 OF nand_2 IS
10 BEGIN
11     c <= NOT (a AND b);
12 END nand_2;
13 -----
```

## (۱۲) COMPONENT

```
1 ----- File nand_3.vhd: -----  
2 LIBRARY ieee;  
3 USE ieee.std_logic_1164.all;  
4 -----  
5 ENTITY nand_3 IS  
6     PORT (a, b, c: IN STD_LOGIC; d: OUT STD_LOGIC);  
7 END nand_3;  
8 -----  
9 ARCHITECTURE nand_3 OF nand_3 IS  
10 BEGIN  
11     d <= NOT (a AND b AND c);  
12 END nand_3;  
13 -----
```

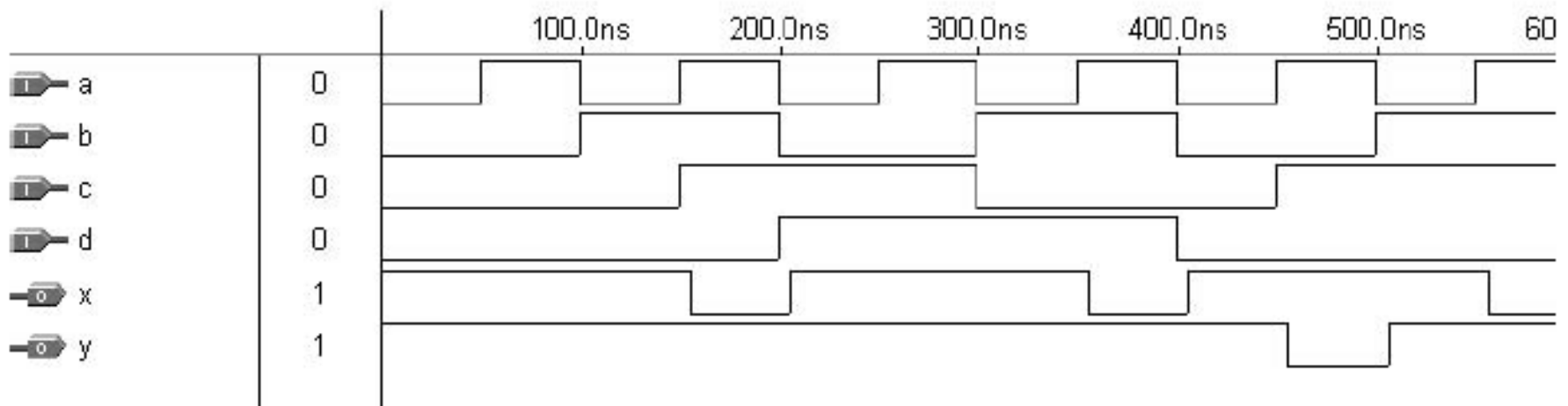
## (۱۳) COMPONENT

```
1  ----- File my_components.vhd: -----
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  -----
5  PACKAGE my_components IS
6      ----- inverter: -----
7      COMPONENT inverter IS
8          PORT (a: IN STD_LOGIC; b: OUT STD_LOGIC);
9      END COMPONENT;
10     ----- 2-input nand: ---
11     COMPONENT nand_2 IS
12         PORT (a, b: IN STD_LOGIC; c: OUT STD_LOGIC);
13     END COMPONENT;
14     ----- 3-input nand: ---
15     COMPONENT nand_3 IS
16         PORT (a, b, c: IN STD_LOGIC; d: OUT STD_LOGIC);
17     END COMPONENT;
18     -----
19 END my_components;
20 -----
```

## (۱۴) COMPONENT

```
1 ----- File project.vhd: -----
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 USE work.my_components.all;
5 -----
6 ENTITY project IS
7     PORT ( a, b, c, d: IN STD_LOGIC;
8           x, y: OUT STD_LOGIC);
9 END project;
10 -----
11 ARCHITECTURE structural OF project IS
12     SIGNAL w: STD_LOGIC;
13 BEGIN
14     U1: inverter PORT MAP (b, w);
15     U2: nand_2 PORT MAP (a, b, x);
16     U3: nand_3 PORT MAP (w, c, d, y);
17 END structural;
18 -----
```

## (۱۵) COMPONENT



## PORT MAP

```
COMPONENT inverter IS  
    PORT (a: IN STD_LOGIC; b: OUT STD_LOGIC);  
END COMPONENT;
```

:positional ➤

```
U1: inverter PORT MAP (x, y);
```

■ به ترتیب

:nominal ➤

```
U1: inverter PORT MAP (x=>a, y=>b);
```

➤ پورت‌ها می‌توانند به صورت متصل نشده باقی بمانند.

```
U2: my_circuit PORT MAP (x=>a, y=>b, w=>OPEN, z=>d);
```



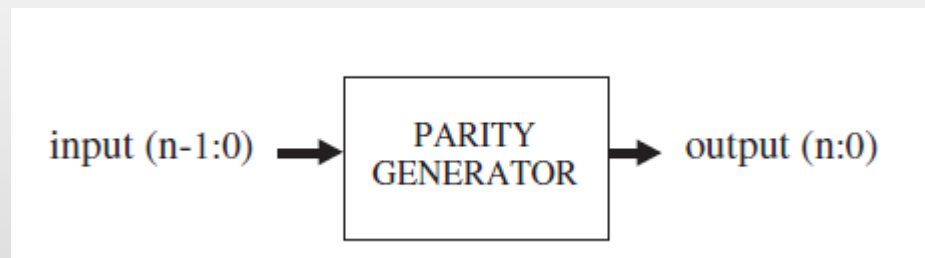
## (۱) GENERIC MAP

➤ با استفاده از GENERIC MAP می‌توان از واحدهای generic نمونه گرفت.

➤ ساختار:

```
label: compon_name GENERIC MAP (param. list) PORT MAP (port list);
```

➤ مثال:





## (۲) GENERIC MAP

```
1 ----- File parity_gen.vhd (component): -----
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 -----
5 ENTITY parity_gen IS
6     GENERIC (n : INTEGER := 7); -- default is 7
7     PORT ( input: IN BIT_VECTOR (n DOWNT0 0);
8           output: OUT BIT_VECTOR (n+1 DOWNT0 0));
9 END parity_gen;
10 -----
11 ARCHITECTURE parity OF parity_gen IS
12 BEGIN
13     PROCESS (input)
14         VARIABLE temp1: BIT;
15         VARIABLE temp2: BIT_VECTOR (output'RANGE);
16     BEGIN
17         temp1 := '0';
18         FOR i IN input'RANGE LOOP
19             temp1 := temp1 XOR input(i);
20             temp2(i) := input(i);
21         END LOOP;
22         temp2(output'HIGH) := temp1;
23         output <= temp2;
24     END PROCESS;
25 END parity;
26 -----
```

## (۳) GENERIC MAP

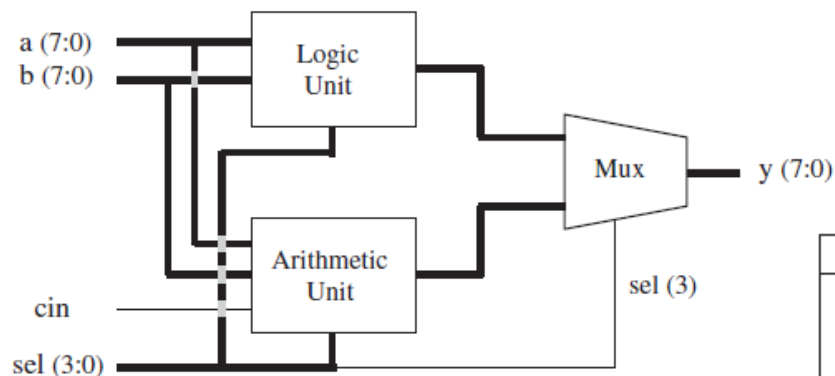
```
1 ----- File my_code.vhd (actual project): -----
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 -----
5 ENTITY my_code IS
6     GENERIC (n : POSITIVE := 2);           -- 2 will overwrite 7
7     PORT ( inp: IN BIT_VECTOR (n DOWNT0 0);
8           outp: OUT BIT_VECTOR (n+1 DOWNT0 0));
9 END my_code;
10 -----
11 ARCHITECTURE my_arch OF my_code IS
12 -----
13     COMPONENT parity_gen IS
14         GENERIC (n : POSITIVE);
15         PORT (input: IN BIT_VECTOR (n DOWNT0 0);
16              output: OUT BIT_VECTOR (n+1 DOWNT0 0));
17     END COMPONENT;
18 -----
19 BEGIN
20     C1: parity_gen GENERIC MAP(n) PORT MAP(inp, outp);
21 END my_arch;
22 -----
```

## (۴) GENERIC MAP

		100.0ns		200.0ns		300.0ns		400.0ns	
 inp	0 1	0	1	2	3	4	5	6	7 0
 outp	0 0	0	9	10	3	12	5	6	15 0

## مثال (۱)

➤ مثال: ALU (Arithmetic Logic Unit)



sel	Operation	Function	Unit
0000	$y \leq a$	Transfer a	Arithmetic
0001	$y \leq a+1$	Increment a	
0010	$y \leq a-1$	Decrement a	
0011	$y \leq b$	Transfer b	
0100	$y \leq b+1$	Increment b	
0101	$y \leq b-1$	Decrement b	
0110	$y \leq a+b$	Add a and b	
0111	$y \leq a+b+cin$	Add a and b with carry	
1000	$y \leq \text{NOT } a$	Complement a	Logic
1001	$y \leq \text{NOT } b$	Complement b	
1010	$y \leq a \text{ AND } b$	AND	
1011	$y \leq a \text{ OR } b$	OR	
1100	$y \leq a \text{ NAND } b$	NAND	
1101	$y \leq a \text{ NOR } b$	NOR	
1110	$y \leq a \text{ XOR } b$	XOR	
1111	$y \leq a \text{ XNOR } b$	XNOR	

## مثال (۲)

```
1 ----- COMPONENT arith_unit: -----
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 USE ieee.std_logic_unsigned.all;
5 -----
6 ENTITY arith_unit IS
7     PORT ( a, b: IN STD_LOGIC_VECTOR (7 DOWNT0 0);
8           sel: IN STD_LOGIC_VECTOR (2 DOWNT0 0);
9           cin: IN STD_LOGIC;
10          x: OUT STD_LOGIC_VECTOR (7 DOWNT0 0));
11 END arith_unit;
12 -----
13 ARCHITECTURE arith_unit OF arith_unit IS
14     SIGNAL arith, logic: STD_LOGIC_VECTOR (7 DOWNT0 0);
15 BEGIN
16     WITH sel SELECT
17         x <= a WHEN "000",
18         a+1 WHEN "001",
19         a-1 WHEN "010",
20         b WHEN "011",
21         b+1 WHEN "100",
22         b-1 WHEN "101",
23         a+b WHEN "110",
24         a+b+cin WHEN OTHERS;
25 END arith_unit;
26 -----
```

## مثال (۳)

```
1 ----- COMPONENT logic_unit: -----
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 -----
5 ENTITY logic_unit IS
6     PORT ( a, b: IN STD_LOGIC_VECTOR (7 DOWNT0 0);
7           sel: IN STD_LOGIC_VECTOR (2 DOWNT0 0);
8           x: OUT STD_LOGIC_VECTOR (7 DOWNT0 0));
9 END logic_unit;
10 -----
11 ARCHITECTURE logic_unit OF logic_unit IS
12 BEGIN
13     WITH sel SELECT
14         x <= NOT a WHEN "000",
15         NOT b WHEN "001",
16         a AND b WHEN "010",
17         a OR b WHEN "011",
18         a NAND b WHEN "100",
19         a NOR b WHEN "101",
20         a XOR b WHEN "110",
21         NOT (a XOR b) WHEN OTHERS;
22 END logic_unit;
23 -----
```

## مثال (۴)

```
1  ----- COMPONENT mux: -----  
2  LIBRARY ieee;  
3  USE ieee.std_logic_1164.all;  
4  -----  
5  ENTITY mux IS  
6      PORT ( a, b: IN STD_LOGIC_VECTOR (7 DOWNT0 0);  
7            sel: IN STD_LOGIC;  
8            x: OUT STD_LOGIC_VECTOR (7 DOWNT0 0));  
9  END mux;  
10 -----  
11 ARCHITECTURE mux OF mux IS  
12 BEGIN  
13     WITH sel SELECT  
14         x <= a WHEN '0',  
15         b WHEN OTHERS;  
16 END mux;  
17 -----
```



## مثال (۵)

```
1 ----- Project ALU (main code): -----
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 -----
5 ENTITY alu IS
6     PORT ( a, b: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
7           cin: IN STD_LOGIC;
8           sel: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
9           y: OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
10 END alu;
11 -----
12 ARCHITECTURE alu OF alu IS
13     -----
14     COMPONENT arith_unit IS
15         PORT ( a, b: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
16               cin: IN STD_LOGIC;
17               sel: IN STD_LOGIC_VECTOR(2 DOWNTO 0);
18               x: OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
19     END COMPONENT;
20     -----
```

## مثال (۶)

```
21     COMPONENT logic_unit IS
22         PORT ( a, b: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
23               sel: IN STD_LOGIC_VECTOR(2 DOWNTO 0);
24               x: OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
25     END COMPONENT;
26     -----
27     COMPONENT mux IS
28         PORT ( a, b: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
29               sel: IN STD_LOGIC;
30               x: OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
31     END COMPONENT;
32     -----
33     SIGNAL x1, x2: STD_LOGIC_VECTOR(7 DOWNTO 0);
34     -----
35 BEGIN
36     U1: arith_unit PORT MAP (a, b, cin, sel(2 DOWNTO 0), x1);
37     U2: logic_unit PORT MAP (a, b, sel(2 DOWNTO 0), x2);
38     U3: mux PORT MAP (x1, x2, sel(3), y);
39 END alu;
40 -----
```

## TEST BENCH

- یک ماژول (COMPONENT) بدون ورودی و خروجی که صرفاً وظیفه تست کردن یک ماژول دیگر را به عهده دارد.
- فقط در شبیه‌سازی استفاده می‌شود.
- ساختار:

**Entity** module\_tb is

**END Entity**;

**Architecture** arch **of** module\_tb is

( **Component definition** --@ @ @ @ @ @ )

**Begin**

**Instance of that module** --@ @ @ @ @ @

**Assigning Value to the Signals**

**End Arch**;

**پایان بخش PACKAGE و COMPONENT**