دوره مقدماتی آشنایی با FPGA و VHDL

FUNCTION و PROCEDURE

محمدرضا عزیزی

امیرعلی ابراهیمی

بهار ۱۳۹۷

مقدمه

- به FUNCTION ها و PROCEDURE ها، *زیربرنامه (subprogram)* می‌گویند.
- همانند بلوک PROCESS، FUNCTION و PROCEDURE بلوک‌های کد ترتیبی هستند و می‌توان از دستورات ترتیبی (IF، CASE و LOOP) در داخل آن‌ها استفاده کرد.
  - استفاده از WAIT درون FUNCTION و PROCEDURE مجاز نیست.
- از لحاظ کاربرد، PROCESS جهت استفاده بلافاصله (immediate) از یک مجموعه دستورات است در حالی که از FUNCTION و PROCEDURE جهت ذخیره قطعه کدهایی که زیاد استفاده می‌شوند و استفاده مجدد از آن‌ها و یا به اشتراک‌گذاری آن‌ها در پروژه‌های مختلف است.

(۱) FUNCTION

- قطعه کد ترتیبی
- هدف FUNCTION ایجاد عملکردهای جدید جهت استفاده در مسائلی چون:
  - تبدیل نوع داده‌ها
  - عملگرهای منطقی
  - محاسبات ریاضی
  - عملگرها و صفت‌های جدید
- دستورات قابل استفاده:
  - IF، CASE و LOOP
- ممنوعیت‌ها:
  - WAIT
  - SIGNAL declaration
  - COMPONENT instantiation

(۲) FUNCTION

- جهت ساخت و استفاده از یک FUNCTION، دو بخش ضروری وجود دارد:
  - تعریف یا بدنه FUNCTION
  - فراخوانی FUNCTION
- ساختار بدنه FUNCTION:

```
FUNCTION function_name [<parameter list>] RETURN data_type IS
    [declarations]
BEGIN
    (sequential statements)
END function_name;
```

- <parameter list>: لیست **ورودی**های تابع
  - <parameter list> = [CONSTANT] constant_name: constant_type; -- default
  - <parameter list> = SIGNAL signal_name: signal_type;
- تعداد ورودی‌ها می‌تواند هر تعدادی باشد (حتی ۰) اما مقدار بازگشتی فقط یک مقدار است.
- در لیست ورودی‌ها نباید از RANGE برای INTEGER و TO/DOWNTO برای وکتورها استفاده کرد.

(۳) FUNCTION

- مثال:

```
FUNCTION f1 (a, b: INTEGER; SIGNAL c: STD_LOGIC_VECTOR)
     RETURN BOOLEAN IS
BEGIN
     (sequential statements)
END f1;
```

(۴) FUNCTION

➢ فراخوانی FUNCTION:

```
x <= conv_integer(a);          -- converts a to an integer
                               -- (expression appears by itself)
y <= maximum(a, b);            -- returns the largest of a and b
                               -- (expression appears by itself)
IF x > maximum(a, b) ...       -- compares x to the largest of a, b
                               -- (expression associated to a
                               -- statement)
```

(۵) FUNCTION

- مثال: Function positive_edge( ):
  - این تابع، لبه بالارونده سیگنال ساعت را تشخیص می‌دهد.

```
------ Function body: --------------------------------
FUNCTION positive_edge(SIGNAL s: STD_LOGIC) RETURN BOOLEAN IS
BEGIN
     RETURN (s'EVENT AND s='1');
END positive_edge;
------ Function call: --------------------------------
...
IF positive_edge(clk) THEN...
...
-------------------------------------------------------
```

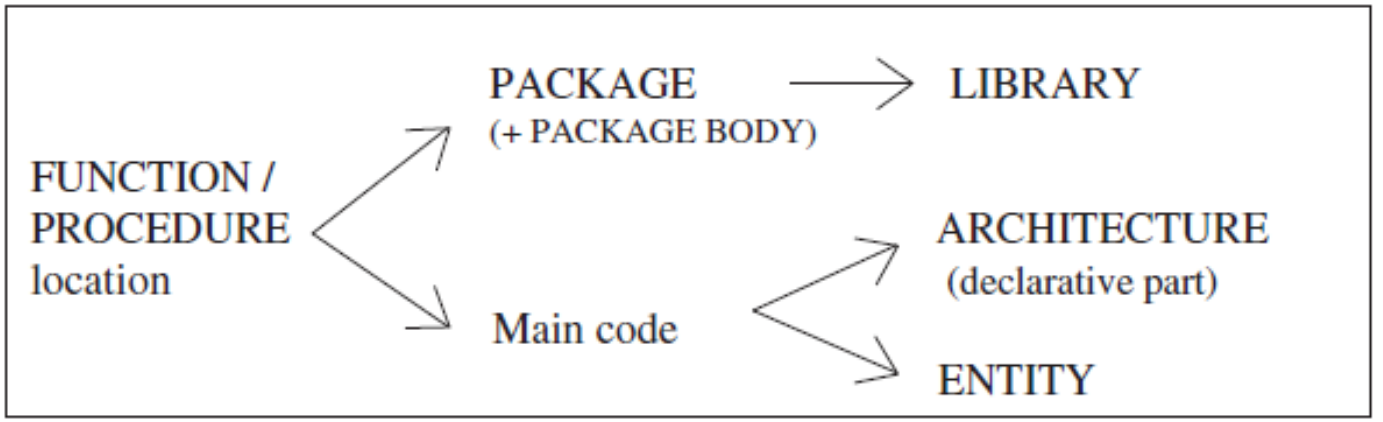
(۶) FUNCTION

- مثال: Function conv_integer( ):
  - تابع زیر، یک ورودی با نوع داده STD_LOGIC_VECTOR را به INTEGER تبدیل می‌کند.
  - این تابع، generic است و از آن می‌توان برای هر ورودی با هر اندازه ای استفاده کرد.

```
------ Function body: -------------------------------
FUNCTION conv_integer (SIGNAL vector: STD_LOGIC_VECTOR)
          RETURN INTEGER IS
     VARIABLE result: INTEGER RANGE 0 TO 2**vector'LENGTH-1;
BEGIN
     IF (vector(vector'HIGH)='1') THEN result:=1;
     ELSE result:=0;
     END IF;
     FOR i IN (vector'HIGH-1) DOWNTO (vector'LOW) LOOP
          result:=result*2;
          IF(vector(i)='1') THEN result:=result+1;
          END IF;
     END LOOP;
     RETURN result;
END conv_integer;
------ Function call: -------------------------------
...
y <= conv_integer(a);
...
-----------------------------------------------------
```

(V) FUNCTION

- مکان FUNCTION (یا PROCEDURE):



- از آن‌جایی که هدف نوشتن FUNCTION استفاده مجدد از کد و یا به اشتراک‌گذاری آن است، معمولا از روش اول (نوشتن FUNCTION در PACKAGE) استفاده می‌شود.

(۸) FUNCTION

- مثال: FUNCTION در کد اصلی:

```
1 ------------------------------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ------------------------------------------------
5 ENTITY dff IS
6      PORT ( d, clk, rst: IN STD_LOGIC;
7             q: OUT STD_LOGIC);
8 END dff;
9 ------------------------------------------------
10 ARCHITECTURE my_arch OF dff IS
11 ------------------------------------------------
12     FUNCTION positive_edge(SIGNAL s: STD_LOGIC)
13          RETURN BOOLEAN IS
14     BEGIN
15          RETURN s'EVENT AND s='1';
16     END positive_edge;
17 ------------------------------------------------
18 BEGIN
19     PROCESS (clk, rst)
20     BEGIN
21          IF (rst='1') THEN q <= '0';
22          ELSIF positive_edge(clk) THEN q <= d;
23          END IF;
24     END PROCESS;
25 END my_arch;
26 ------------------------------------------------
```

(۹) FUNCTION

➢ مثال: FUNCTION در PACKAGE:

```
1 ------- Package: ------------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ---------------------------------------------
5 PACKAGE my_package IS
6      FUNCTION positive_edge(SIGNAL s: STD_LOGIC) RETURN BOOLEAN;
7 END my_package;
8 ---------------------------------------------
9 PACKAGE BODY my_package IS
10     FUNCTION positive_edge(SIGNAL s: STD_LOGIC)
11          RETURN BOOLEAN IS
12     BEGIN
13          RETURN s'EVENT AND s='1';
14     END positive_edge;
15 END my_package;
16 ---------------------------------------------
```

(۱۰) FUNCTION

```
1 ------ Main code: ----------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 USE work.my_package.all;       -- *IMP*
5 ----------------------------------------------
6 ENTITY dff IS
7      PORT ( d, clk, rst: IN STD_LOGIC;
8             q: OUT STD_LOGIC);
9 END dff;
10 ---------------------------------------------
11 ARCHITECTURE my_arch OF dff IS
12 BEGIN
13     PROCESS (clk, rst)
14     BEGIN
15          IF (rst='1') THEN q <= '0';
16          ELSIF positive_edge(clk) THEN q <= d;
17          END IF;
18     END PROCESS;
19 END my_arch;
20 ---------------------------------------------
```

(۱۱) FUNCTION

➢ مثال: conv_integer( ) در PACKAGE:

```
1 --------- Package: ---------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ---------------------------------------------
5 PACKAGE my_package IS
6      FUNCTION conv_integer (SIGNAL vector: STD_LOGIC_VECTOR)
7           RETURN INTEGER;
8 END my_package;
9 ---------------------------------------------
10 PACKAGE BODY my_package IS
11     FUNCTION conv_integer (SIGNAL vector: STD_LOGIC_VECTOR)
12               RETURN INTEGER IS
13          VARIABLE result: INTEGER RANGE 0 TO 2**vector'LENGTH-1;
14     BEGIN
15          IF (vector(vector'HIGH)='1') THEN result:=1;
16          ELSE result:=0;
17          END IF;
18          FOR i IN (vector'HIGH-1) DOWNTO (vector'LOW) LOOP
19               result:=result*2;
20               IF(vector(i)='1') THEN result:=result+1;
21               END IF;
22          END LOOP;
23          RETURN result;
24     END conv_integer;
25 END my_package;
26 ---------------------------------------------
```

(۱۲) FUNCTION

```
1 -------- Main code: --------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 USE work.my_package.all;
5 ----------------------------------------------
6 ENTITY conv_int2 IS
7       PORT ( a: IN STD_LOGIC_VECTOR(0 TO 3);
8              y: OUT INTEGER RANGE 0 TO 15);
9 END conv_int2;
10 ---------------------------------------------
11 ARCHITECTURE my_arch OF conv_int2 IS
12 BEGIN
13      y <= conv_integer(a);
14 END my_arch;
15 ----------------------------------------------
```

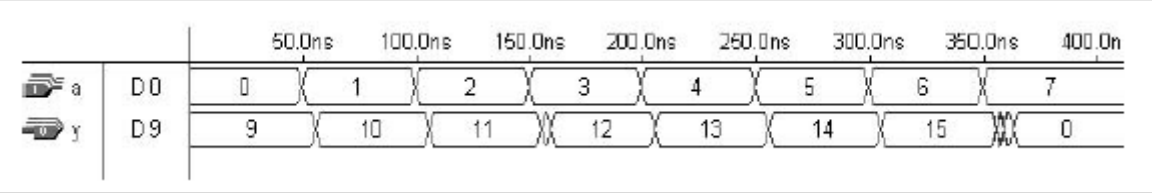
(۱۳) FUNCTION

- مثال: سربارگذاری (overload) عملگر "+" جهت انجام عمل جمع بر روی دو STD_LOGIC_VECTOR:

```
1 -------- Package: ----------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ---------------------------------------------
5 PACKAGE my_package IS
6      FUNCTION "+" (a, b: STD_LOGIC_VECTOR)
7           RETURN STD_LOGIC_VECTOR;
8 END my_package;
9 ---------------------------------------------
10 PACKAGE BODY my_package IS
11     FUNCTION "+" (a, b: STD_LOGIC_VECTOR)
12               RETURN STD_LOGIC_VECTOR IS
13          VARIABLE result: STD_LOGIC_VECTOR;
14          VARIABLE carry: STD_LOGIC;
15     BEGIN
16          carry := '0';
17          FOR i IN a'REVERSE_RANGE LOOP
18               result(i) := a(i) XOR b(i) XOR carry;
19               carry := (a(i) AND b(i)) OR (a(i) AND carry) OR
20                        (b(i) AND carry);
21          END LOOP;
22          RETURN result;
23     END "+";
24 END my_package;
25 ---------------------------------------------
```

(۱۴) FUNCTION

```
1 --------- Main code: -------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 USE work.my_package.all;
5 ---------------------------------------------
6 ENTITY add_bit IS
7     PORT ( a: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
8            y: OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
9 END add_bit;
10 ---------------------------------------------
11 ARCHITECTURE my_arch OF add_bit IS
12     CONSTANT b: STD_LOGIC_VECTOR(3 DOWNTO 0) := "0011";
13     CONSTANT c: STD_LOGIC_VECTOR(3 DOWNTO 0) := "0110";
14 BEGIN
15     y <= a + b + c; -- overloaded "+" operator
16 END my_arch;
17 ---------------------------------------------
```



(۱۵) FUNCTION

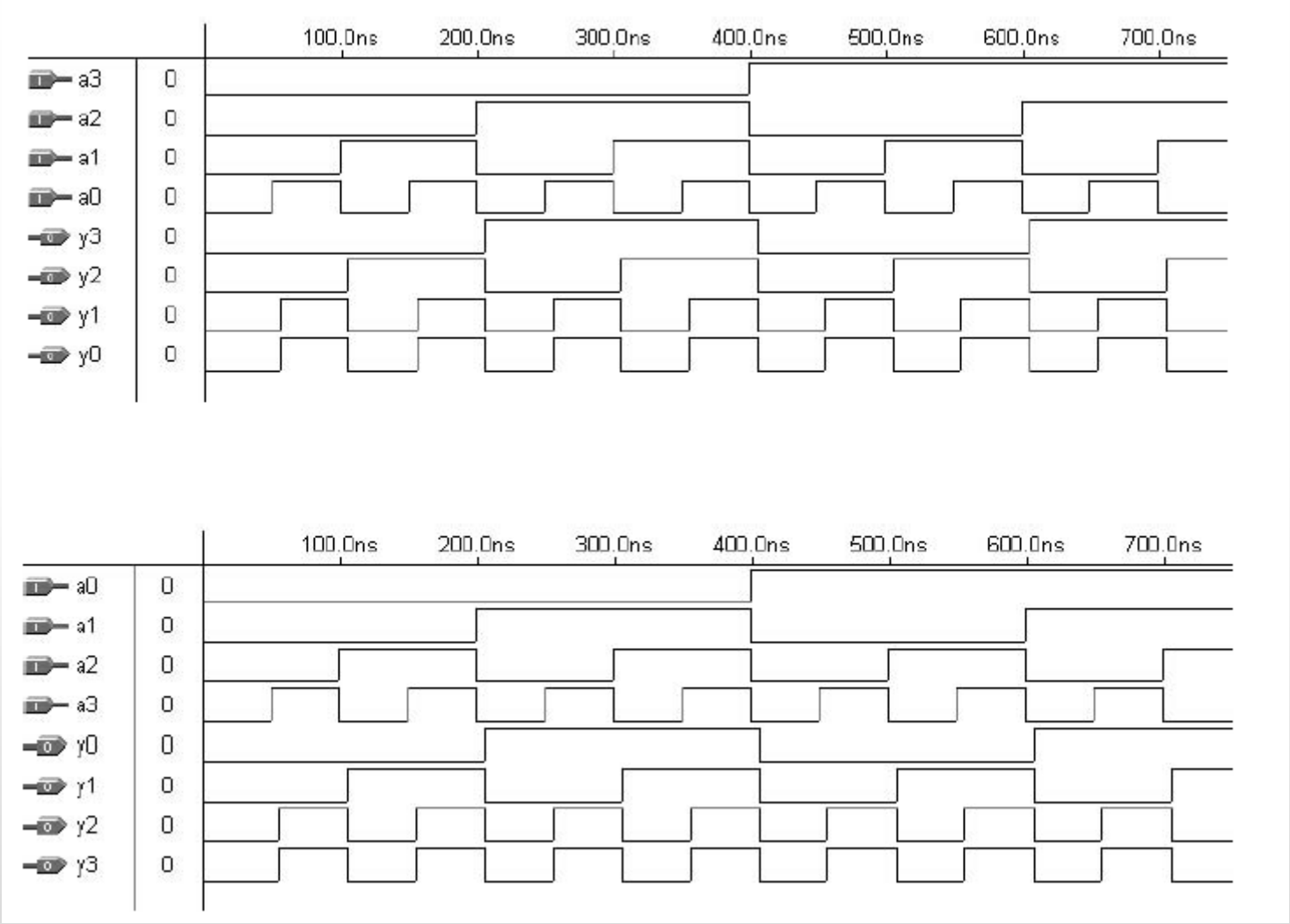
- مثال: شیفت حسابی (Arithmetic Shift):
  - آرگومان ورودی اول: وکتوری که باید شیفت داده شود.
  - آرگومان ورودی دوم: مقدار شیفت

```
1 ------------------------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ------------------------------------------
5 ENTITY shift_left IS
6      GENERIC (size: INTEGER := 4);
7      PORT ( a: IN STD_LOGIC_VECTOR(size-1 DOWNTO 0);
8             x, y, z: OUT STD_LOGIC_VECTOR(size-1 DOWNTO 0));
9 END shift_left;
10 ------------------------------------------
11 ARCHITECTURE behavior OF shift_left IS
12 ------------------------------------------
13     FUNCTION slar (arg1: STD_LOGIC_VECTOR; arg2: NATURAL)
14                RETURN STD_LOGIC_VECTOR IS
15          VARIABLE input: STD_LOGIC_VECTOR(size-1 DOWNTO 0) := arg1;
16          CONSTANT size : INTEGER := arg1'LENGTH;
17          VARIABLE copy: STD_LOGIC_VECTOR(size-1 DOWNTO 0)
18                      := (OTHERS => arg1(arg1'RIGHT));
19          VARIABLE result: STD_LOGIC_VECTOR(size-1 DOWNTO 0);
```

(۱۶) FUNCTION

```
20      BEGIN
21           IF (arg2 >= size-1) THEN result := copy;
22           ELSE result := input(size-1-arg2 DOWNTO 1) &
23                          copy(arg2 DOWNTO 0);
24           END IF;
25           RETURN result;
26      END slar;
27      ---------------------------------------
28 BEGIN
29      x <= slar(a, 0);
30      y <= slar(a, 1);
31      z <= slar(a, 2);
32 END behavior;
33 ---------------------------------------
```

(۱۷) FUNCTION


100.0ns
200.0ns
300.0ns
400.0ns
500.0ns
600.0ns
700.0ns
a3
a2
a1
a0
y3
y2
y1
y0
0
100.0ns
200.0ns
300.0ns
400.0ns
500.0ns
600.0ns
700.0ns
a0
a1
a2
a3
y0
y1
y2
y3
0

(۱) PROCEDURE

- هدف و نحوه عملکرد یک PROCEDURE بسیار شبیه به FUNCTION است.
- تنها تفاوت PROCEDURE این است که برخلاف FUNCTION (که تنها یک مقدار خروجی می‌تواند داشته باشد) می‌تواند چندین مقدار خروجی داشته باشد.
- دارای دو قسمت است:
  - بدنه PROCEDURE
  - فراخوانی PROCEDURE
- بدنه PROCEDURE:

```
PROCEDURE procedure_name [<parameter list>] IS
    [declarations]
BEGIN
    (sequential statements)
END procedure_name;
```

(۲) PROCEDURE

- <parameter list> لیست ورودی و خروجی‌های PROCEDURE را مشخص می‌کند.
  - <parameter list> = [CONSTANT] constant_name: mode type;
  - <parameter list> = SIGNAL signal_name: mode type; or
  - <parameter list> = VARIABLE variable_name: mode type;
- یک PROCEDURE می‌تواند هر تعداد ورودی(IN)، خروجی(OUT) یا ورودی/خروجی (IN/OUT) داشته باشد.
- این ورودی، خروجی و ورودی/خروجی‌ها می‌توانند SIGNAL، VARIABLE یا CONSTANT باشند.
- برای ورودی، مد پیش‌فرض CONSTANT است و برای خروجی و ورودی/خروجی‌ها VARIABLE.

(۳) PROCEDURE

- همان‌طور که قبلا دیدیم، WAIT، تعریف SIGNAL و COMPONENT در داخل FUNCTION غیرمجاز است.
  - این قانون در رابطه با PROCEDURE نیز صادق است با این تفاوت که می‌توان در PROCEDURE، SIGNAL تعریف کرد، اما در این صورت PROCEDURE باید در بدنه یک PROCESS فراخوانی شود.
- علاوه بر WAIT، هر حالت تشخیص لبه دیگری نیز در PROCEDURE غیرقابل سنتز است.
  - یعنی بر خلاف یک FUNCTION، یک PROCEDURE قابل سنتز نباید رجیستر تولید کند.

(۴) PROCEDURE

- مثال:

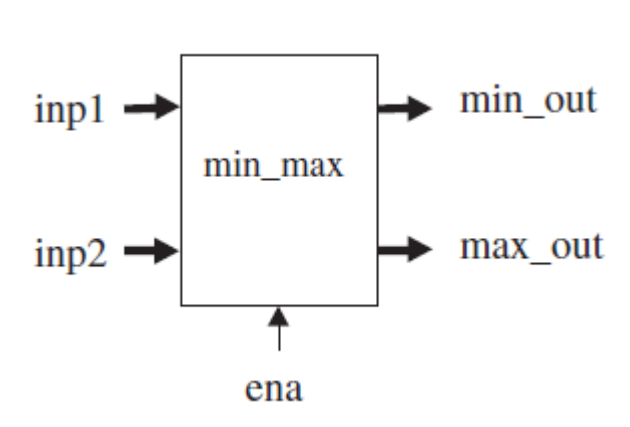
```
PROCEDURE my_procedure ( a: IN BIT; SIGNAL b, c: IN BIT;
                         SIGNAL x: OUT BIT_VECTOR(7 DOWNTO 0);
                         SIGNAL y: INOUT INTEGER RANGE 0 TO 99) IS
BEGIN
...
END my_procedure;
```

- فراخوانی PROCEDURE:

```
compute_min_max(in1, in2, 1n3, out1, out2);
                                -- statement by itself
divide(dividend, divisor, quotient, remainder);
                                -- statement by itself
IF (a>b) THEN compute_min_max(in1, in2, 1n3, out1, out2);
                                -- procedure call associated to
                                -- another statement
```

(۵) PROCEDURE

- مکان PROCEDURE:
  - کاملا مشابه با FUNCTION
- مثال: PROCEDURE در کد اصلی:
  - این PROCEDURE دو ورودی بدون علامت ۸ بیتی دریافت کرده، آن‌ها را به صورت صعودی مرتب کرده و به ترتیب در خروجی‌ها قرار می‌دهد.



(۶) PROCEDURE

```
1 ------------------------------------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ------------------------------------------------------
5 ENTITY min_max IS
6       GENERIC (limit : INTEGER := 255);
7       PORT ( ena: IN BIT;
8              inp1, inp2: IN INTEGER RANGE 0 TO limit;
9              min_out, max_out: OUT INTEGER RANGE 0 TO limit);
10 END min_max;
11 ------------------------------------------------------
```

(V) PROCEDURE

```
12 ARCHITECTURE my_architecture OF min_max IS
13     --------------------------
14     PROCEDURE sort (SIGNAL in1, in2: IN INTEGER RANGE 0 TO limit;
15                     SIGNAL min, max: OUT INTEGER RANGE 0 TO limit) IS
16     BEGIN
17          IF (in1 > in2) THEN
18               max <= in1;
19               min <= in2;
20          ELSE
21               max <= in2;
22               min <= in1;
23          END IF;
24     END sort;
25     --------------------------
26 BEGIN
27     PROCESS (ena)
28     BEGIN
29          IF (ena='1') THEN sort (inp1, inp2, min_out, max_out);
30          END IF;
31     END PROCESS;
32 END my_architecture;
33 ----------------------------------------------------
```

(۸) PROCEDURE

➢ مثال: PROCEDURE در PACKAGE:

```
1 ------------ Package: --------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ---------------------------------------
5 PACKAGE my_package IS
6      CONSTANT limit: INTEGER := 255;
7      PROCEDURE sort (SIGNAL in1, in2: IN INTEGER RANGE 0 TO limit;
8      SIGNAL min, max: OUT INTEGER RANGE 0 TO limit);
9 END my_package;
10 ---------------------------------------
11 PACKAGE BODY my_package IS
12     PROCEDURE sort (SIGNAL in1, in2: IN INTEGER RANGE 0 TO limit;
13                     SIGNAL min, max: OUT INTEGER RANGE 0 TO limit) IS
14     BEGIN
15          IF (in1 > in2) THEN
16               max <= in1;
17               min <= in2;
18          ELSE
19               max <= in2;
20               min <= in1;
21          END IF;
22     END sort;
23 END my_package;
24 ------------------------------------------------
```

(۹) PROCEDURE

➢ مثال: PROCEDURE در PACKAGE:

```
1 --------- Main code: ----------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 USE work.my_package.all;
5 ---------------------------------------
6 ENTITY min_max IS
7      GENERIC (limit: INTEGER := 255);
8      PORT ( ena: IN BIT;
9             inp1, inp2: IN INTEGER RANGE 0 TO limit;
10            min_out, max_out: OUT INTEGER RANGE 0 TO limit);
11 END min_max;
12 ---------------------------------------
13 ARCHITECTURE my_architecture OF min_max IS
14 BEGIN
15     PROCESS (ena)
16     BEGIN
17          IF (ena='1') THEN sort (inp1, inp2, min_out, max_out);
18          END IF;
19     END PROCESS;
20 END my_architecture;
21 -----------------------------------------------
```

(۱) RESOLUTION FUNCTION

➢ مثال:

```
TYPE v4l IS (‘X’,’0’,’1’,’Z’);
ENTITY multiplexer IS
     PORT (a, b, s : IN v4l;
           w : OUT v4l
           );
END ENTITY;
--
-- Does not compile!
ARCHITECTURE wired OF multiplexer IS
     SIGNAL y : v4l;
BEGIN
     T1: y <= a WHEN s=‘0’ ELSE 'Z';
     T2: y <= b WHEN s=‘1’ ELSE 'Z';
     w <= y;
END ARCHITECTURE wired;
```

(۲) RESOLUTION FUNCTION

- بازه گسسته یک وکتور:

```
TYPE v4l IS (‘X’,’0’,’1’,’Z’);
TYPE v4l_2d IS ARRAY (v4l, v4l) OF v4l;
CONSTANT v4l_nand2_table : v4l_2d := (
--       X   0   1   Z
        (‘X’,’1’,’X’,’X’), -- X
        (‘1’,’1’,’1’,’1’), -- 0
        (‘X’,’1’,’0’,’X’), -- 1
        (‘X’,’1’,’X’,’X’)); -- Z

CONSTANT v4l_nand2_table : v4l_2d := (
‘0’ => (OTHERS => ‘1’),
‘1’ => (‘0’ => ‘1’, ‘1’ => ‘0’, OTHERS => ‘X’),
OTHERS => (‘0’ => ‘1’, OTHERS => ‘X’) );
```

(۳) RESOLUTION FUNCTION

- resolve کردن دو مقدار:

```
TYPE v4l IS (‘X’,’0’,’1’,’Z’);
TYPE v4l_2d IS ARRAY (v4l, v4l) OF v4l;
TYPE v4l_vector IS ARRAY (NATURAL RANGE <>) OF v4l;
FUNCTION wire (a, b : v4l) RETURN v4l IS
     CONSTANT v4l_wire_table : v4l_2d := (
               ‘X’ => (‘X’, ‘X’, ‘X’, ‘X’),
               ‘0’ => (‘X’, ‘0’, ‘X’, ‘0’),
               ‘1’ => (‘X’, ‘X’, ‘1’, ‘1’),
               ‘Z’ => (‘X’, ‘0’, ‘1’, ‘Z’));
BEGIN
     RETURN v4l_wire_table (a, b)
END wire;
```

(۴) RESOLUTION FUNCTION

- resolve کردن چند مقدار (resolution function):

```
FUNCTION wiring ( drivers : v4l_vector) RETURN v4l IS
     VARIABLE accumulate : v4l := 'Z';
BEGIN
     FOR i IN drivers'RANGE LOOP
          accumulate := wire (accumulate, drivers(i));
     END LOOP;
     RETURN accumulate;
END wiring;
```

(۵) RESOLUTION FUNCTION

- استفاده از تابع resolution function:

```
ARCHITECTURE wired OF multiplexer IS
     SIGNAL y : wiring v4l;          -- using ‘wiring’ function
BEGIN
     T1: y <= a WHEN s=‘0’ ELSE 'Z';
     T2: y <= b WHEN s=‘1’ ELSE 'Z';
     w <= y;
END ARCHITECTURE wired;
-- Can be compiled now!
```

(۶) RESOLUTION FUNCTION

- رفع یک مشکل!
  - در این حالت برای استفاده از حالت `resolve` شده نوع‌داده `v41`، باید سیگنال‌ها یا متغیرها را به صورت `wiring v41` تعریف کرد.
  - رفع این مشکل:

```
SUBTYPE wired_v41 IS wiring v41;
```

این تعریف زیرنوع را در یک `PACKAGE` قرار داده و از این پس برای استفاده از حالت `resolve` شده نوع داده `v41`، از زیرنوع `wired_v41` استفاده می‌کنیم.

مطالعه بیشتر

- بخش ۱۱.۶ (ASSERT)
  - صفحه ۲۷۰ مرجع اصلی
- مثال ۱۱.۸ (ضرب‌کننده)
  - صفحه ۲۶۳ مرجع اصلی درس

پایان بخش FUNCTION و PROCEDURE