Integrated Circuits Design by FPGA

م.م. أحمد مؤيد عبدالحسين
جامعة الفرات الأوسط التقنية / الكلية التقنية الهندسية / نجف

Lecture 12

Functions

Objectives of this Lecture

- To define **Functions**
- To implement **Functions** in examples design.

Contents of this Lecture

- Introduction
- Function
- Function Location

Introduction

- **Functions** and **Procedures** are collectively called subprograms.
- From a construction point of view, they are very similar to a PROCESS. For they are the only pieces of sequential VHDL code, and thus employ the same sequential statements seen there (IF, CASE, and LOOP; WAIT is not allowed).
- However, from the applications point of view, there is a fundamental difference between a **PROCESS** and a **FUNCTION** or **PROCEDURE**. While the first is intended for immediate use in the main code, the others are intended mainly for LIBRARY allocation, that is, their purpose is to store commonly used pieces of code, so they can be reused or shared by other projects.

Function

- A **FUNCTION** is a section of sequential code. Its purpose is to create new functions to deal with commonly encountered problems, like data type conversions, logical operations, arithmetic computations, and new operators and attributes.
- By writing such code as a **FUNCTION**, it can be shared and reused, also partitioning the main code to be shorter and easier to understand.
- As already mentioned, a **FUNCTION** is very similar to a PROCESS. The same statements that can be used in a process (IF, WAIT, CASE, and LOOP) can also be used in a function, with the exception of WAIT.
- The other two prohibitions in a **Function** are SIGNAL declarations and COMPONENT instantiations.

Function

- To construct and use a **Function**, two parts are necessary: the function itself (**function body**) and a **call to the function**. Their syntaxes are shown below.

Function Body

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

In the syntax above, ⟨parameter list⟩ specifies the function's input parameters, that is:

⟨parameter list⟩ = [CONSTANT] constant_name: constant_type; or

⟨parameter list⟩ = SIGNAL signal_name: signal_type;

Function

- There can be any number of such parameters (even zero), which, as shown above, can only be CONSTANT (default) or SIGNAL (VARIABLES are not allowed).
- Their types can be any of the synthesizable data types studied in chapter 3 (BOOLEAN, STD_LOGIC, INTEGER, etc.). However, no range specification should be included (for example, do not enter RANGE when using INTEGER, or TO/DOWNTO when using STD_LOGIC_VECTOR).
- On the other hand, there is only one return value, whose type is specified by data_type.

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

Function Body

Function

- **Function Call:**

A function is called as part of an expression. The expression can obviously appear by itself or associated to a statement (either concurrent or sequential).

Examples of function calls:

```
x <= conv_integer(a);

y <= maximum(a, b);

IF x > maximum(a, b)
```

Function

Example 11.1: Function positive_edge( ):

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

Function Location

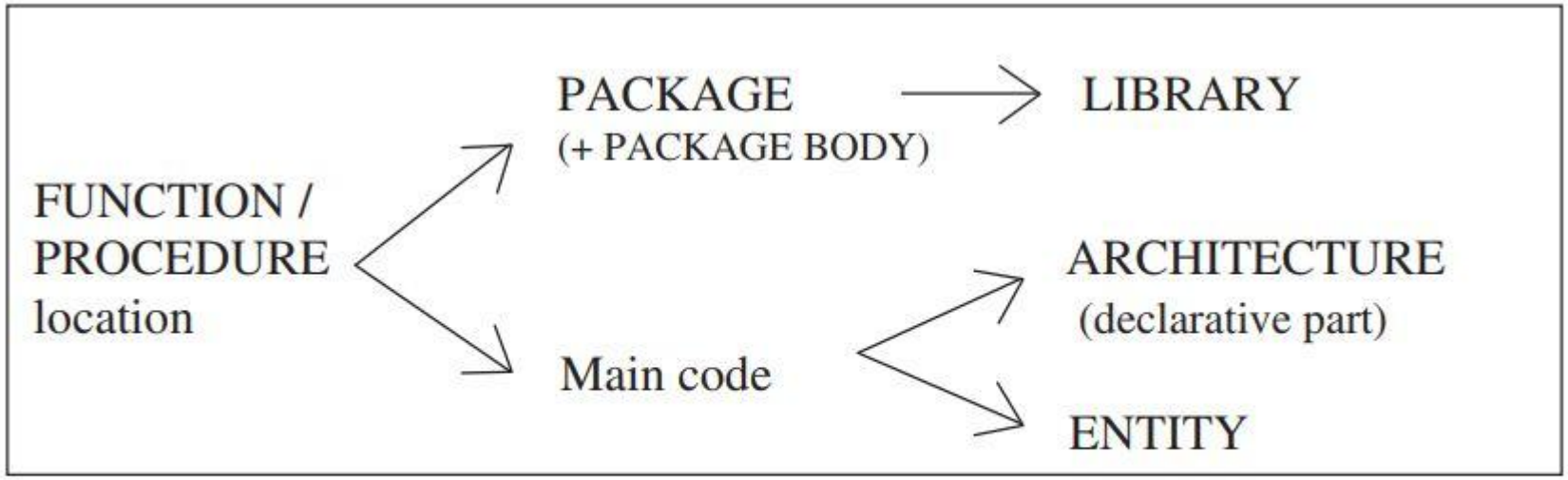

Figure 11.1
Typical locations of a FUNCTION or PROCEDURE.

Function Location

Example 11.3: FUNCTION Located in the Main Code

```
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 ------------------------------------------
```

Function Location

Example 11.4: FUNCTION Located in a 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 ---------------------------------------------
```

```
1  ------ Main code: ----------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  USE work.my_package.all;
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 ---------------------------------------------
```

Function Location

Example 11.5: FUNCTION Located in a Package : conv_integer( )

```
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 ---------------------------------------------
```

Function Location

Example 11.5: FUNCTION Located in a Package : conv_integer( )

```
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 ----------------------------------------------
```

Function Location

Example 11.6: FUNCTION Located in a Package : Overloaded "+" Operator

```
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;
```

```
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 ------------------------------------------------
```

Function Location

Example 11.6: FUNCTION Located in a Package : Overloaded “+” Operator

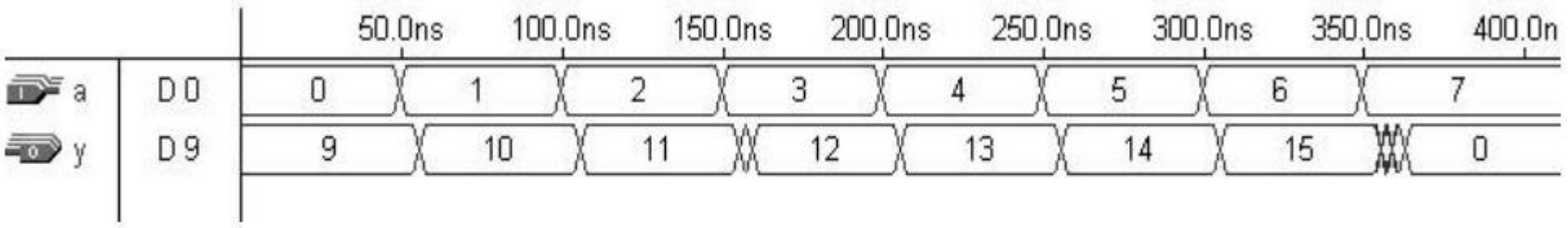

Figure 11.2
Simulation results of example 11.6.

Assignments

The assignments will be attached to your class room.

- Problem **11.2**

End of lecture 12
Any Questions ?