GONZAGA UNIVERSITY

VHDL:
Functions and Procedures

talarico@gonzaga.edu

GONZAGA UNIVERSITY

Subprograms

- FUNCTION
- PROCEDURE

} pieces of sequential VHDL code

- FUNCTIONs and PROCEDUREs have the same basic purpose
  - store commonly used pieces of code, so they can be reused and shared
- VHDL allow FUNCTIONs and PROCEDUREs to be overloaded

PROCESS and FUNCTION vs. PROCESS

- FUNCTIONs and PROCEDUREs are **mainly** intended for **LIBRARY allocation**
  (although it is possible to locate them in the main code)

- PROCESSes are intended for **immediate** use in the **main code**

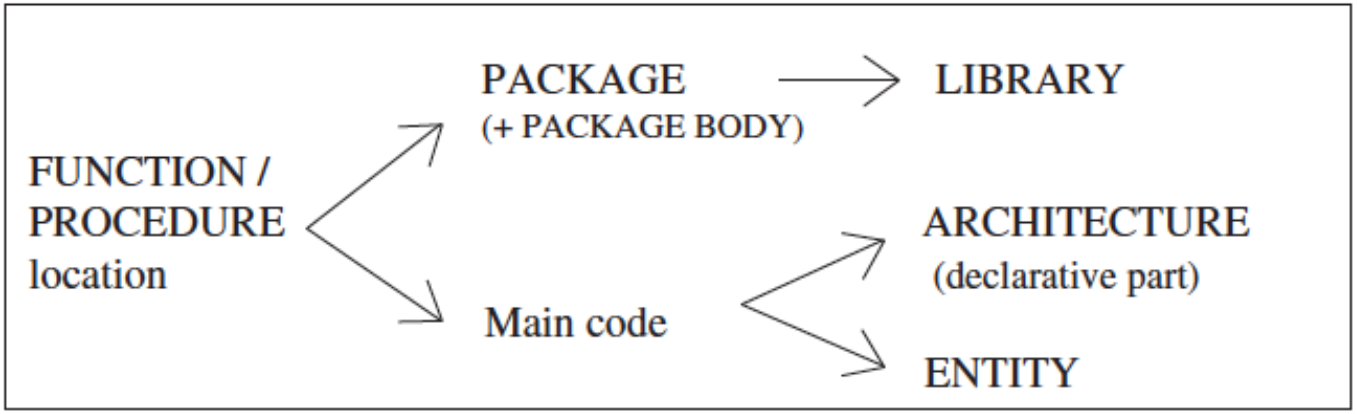
GONZAGA UNIVERSITY

FUNCTION vs. PROCEDURE

- FUNCTION
  - zero or more input parameters and a single return value
  - The input parameters can only be constants (default) or signals (variables are not allowed)
- PROCEDURE
  - Any number of IN, OUT, and INOUT parameters which can be constants, signals, variables
  - The default for OUT and INOUT parameters is variable
- A FUNCTION is called as part of an expression, while a PROCEDURE is a statement on its own

GONZAGA UNIVERSITY

FUNCTION/PROCEDURE location



GONZAGA UNIVERSITY

FUNCTION

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;

GONZAGA UNIVERSITY

FUNCTION: Example

Example of Function body:

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

The word CONSTANT can be omitted

do not enter RANGE or TO/DOWNTO

Examples of function calls:

```
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)
```

Examples: FUNCTION

```
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 located in the main

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

GONZAGA UNIVERSITY

PROCEDURE

Procedure Body

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

In the syntax above, `<parameter list>` specifies the procedure's input and output parameters; that is:

⟨parameter list⟩ = [CONSTANT] constant_name: mode type;

⟨parameter list⟩ = SIGNAL signal_name: mode type; or

⟨parameter list⟩ = VARIABLE variable_name: mode type;

GONZAGA UNIVERSITY

PROCEDURE: example

Example of PROCEDURE body:

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

NOT an issue with PROCEDUREs

Examples of procedure calls:

```
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
```

Examples: PROCEDURE

GONZAGA UNIVERSITY

```
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  STD_LOGIC;
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 ------------------------------------------------------
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 ------------------------------------------------------
```

PROCEDURE located in the main

Examples: PROCEDURE

GONZAGA UNIVERSITY

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

PROCEDURE located in a 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  STD_LOGIC;
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 ----------------------------------------
```

GONZAGA UNIVERSITY

ASSERT

```
ASSERT condition
[REPORT "message"]
[SEVERITY severity_level];
```

Example:

```
ASSERT a'LENGTH = b'LENGTH
REPORT "Error: vectors do not have same length!"
SEVERITY failure;
```