Procedure Call

Contrary to a FUNCTION, which is called as part of an expression, a PROCEDURE call is a statement on its own. It can appear by itself or associated to a statement (either concurrent or sequential).

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
```

Procedure Location

The typical locations of a PROCEDURE are the same as those of a FUNCTION.

Again, though it is usually placed in a PACKAGE (for code partitioning, code reuse, and code sharing purposes), it can also be located in the main code (either in the ENTITY or in the declarative part of the ARCHITECTURE).

When placed in a PACKAGE, a PACKAGE BODY is then necessary, which must contain the body of each PROCEDURE declared in the declarative part of the PACKAGE.

Example: PROCEDURE Located in the Main Code

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

Example: PROCEDURE Located in a 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 ----------------------------------------------
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 ---------------------------------------------
```

Example:

```
1 ---------Package:-----------------------------------------------
2 PACKAGE my_package IS
3 PROCEDURE min_max (SIGNAL a, b, c: IN INTEGER;
4                                     SIGNAL min, max: OUT INTEGER);
5 END PACKAGE;
6 ---------------------------------------------------------------
7 PACKAGE BODY my_package IS
8 PROCEDURE min_max (SIGNAL a, b, c: IN INTEGER RANGE 0 TO 255;
9             SIGNAL min, max: OUT INTEGER RANGE 0 TO 255) IS
10 BEGIN
11     IF (a>=b) THEN
12             IF (a>=c) THEN max <= a;
13                     IF (b>=c) THEN min <= c;
14                             ELSE min <= b;
15                     END IF;
16             ELSE
17              max <= c;
18             min <= b;
19             END IF;
20     ELSE
21             IF (b>=c) THEN max <= b;
22                     IF (a>=c) THEN min <= c;
23                     ELSE min <= a;
24                     END IF;
25             ELSE
26             max <= c;
27             min <= a;
28             END IF;
29 END IF;
30 END PACKAGE BODY;
31 END my_package;
32 ---------------------------------------------------------------
```

```
1 ---------Main code:----------------------------
2 USE work.my_package.all;
3 -----------------------------------------------
4 ENTITY comparator IS
5       PORT (a, b, c: IN INTEGER RANGE -256 TO 255;
6                min, max: OUT INTEGER RANGE -256 TO 255);
7 END ENTITY;
8 -----------------------------------------------
9 ARCHITECTURE comparator OF comparator IS
10 BEGIN
11              min_max(a, b, c, min, max);
12 END ARCHITECTURE;
13 -----------------------------------------------
```

FUNCTION versus PROCEDURE Summary

A FUNCTION has zero or more input parameters and a single return value. The input parameters can only be CONSTANTS (default) or SIGNALS (VARIABLES are not allowed).

A PROCEDURE can have any number of IN, OUT, and INOUT parameters, which can be SIGNALS, VARIABLES, or CONSTANTS. For input parameters (mode IN) the default is CONSTANT, whereas for output parameters (mode OUT or INOUT) the default is VARIABLE.

A FUNCTION is called as part of an expression, while a PROCEDURE is a statement on its own.

In both, WAIT and COMPONENTS are not synthesizable.

The possible locations of FUNCTIONS and PROCEDURES are the same. Though they are usually placed in PACKAGES (for code partitioning, code sharing, and code reuse purposes), they can also be located in the main code (either inside the ARCHITECTURE or inside the ENTITY).

When placed in a PACKAGE, then a PACKAGE BODY is necessary, which should contain the body of each FUNCTION and/or PROCEDURE declared in the PACKAGE.