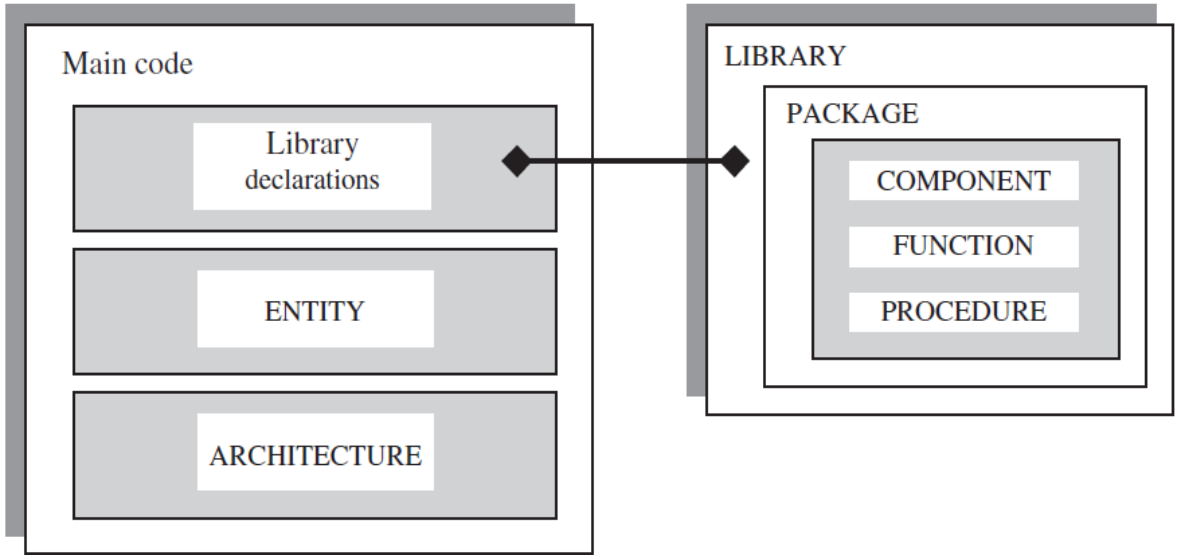
VHDL – Package (Private Sub-Library)

Introduction

- In order to allow common pieces of code to be reused, shared and partitioning, it is more usual to place them in a **LIBRARY**, which is helpful when dealing with long codes.
- Generally, frequently used pieces of code can be written in the form of **COMPONENTS**, **FUNCTIONS**, or **PROCEDURES**, then placed in a **PACKAGE**, which is finally compiled into the destination LIBRARY.
- As we have already seen, that at least three LIBRARIES are generally needed in a design: ***ieee, std,*** and ***work***. In this chapter we will learn how to construct our own sub-libraries using **Package** (VHDL element), which can then be added to the ***work*** LIBRARY.



PACKAGE

- As mentioned, the importance of this technique is that it allows code partitioning, code sharing, and code reuse.
- Package syntax is composed of two parts (must have the same name):
  - ✓ Package declarative part – **mandatory**.
  - ✓ Package Body - necessary only when **FUNCTION** or **PROCEDURE** are declared in the upper part, in which case it must contain the descriptions (bodies) of the subprograms.
- The declarative part can contain the following elements:

  COMPONENT, **FUNCTION**, **PROCEDURE**, **TYPE**, **CONSTANT**, etc.
- Package syntax:

```
PACKAGE package_name IS
    -- package_declarative_part
END package_name;


PACKAGE BODY package_name IS
    -- FUNCTION and PROCEDURE descriptions
END package_name;
```

PACKAGE – example 1

Package with Declarative part only:

```
PACKAGE example IS

  TYPE byte IS RANGE 0 TO 255;
  SUBTYPE nibble IS byte RANGE 0 TO 15;
  CONSTANT byte_ff : byte := 255;
  SIGNAL addend : nibble;

  COMPONENT byte_adder
    PORT (a, b : IN byte;
          z : OUT byte; overflow : OUT boolean);
  END COMPONENT;

END PACKAGE example;
```

PACKAGE – example 2

Package with Declarative and Body parts:

```
PACKAGE filt_cmp IS
  TYPE stae_type IS (idle,tap1,tap2,tap3,tap4);
  FUNCTION compare (variable a, b : integer) RETURN boolean;
END PACKAGE filt_cmp;


PACKAGE BODY filt_cmp IS
  FUNCTION compare (variable a, b : integer) IS
    VARIABLE  temp : BOOLEAN;
    BEGIN
      IF  a<b THEN
        temp := true;
      ELSE
        temp := false;
      END IF;
      RETURN temp;
  END FUNCTION compare;
END PACKAGE BODY filt_cmp;
```