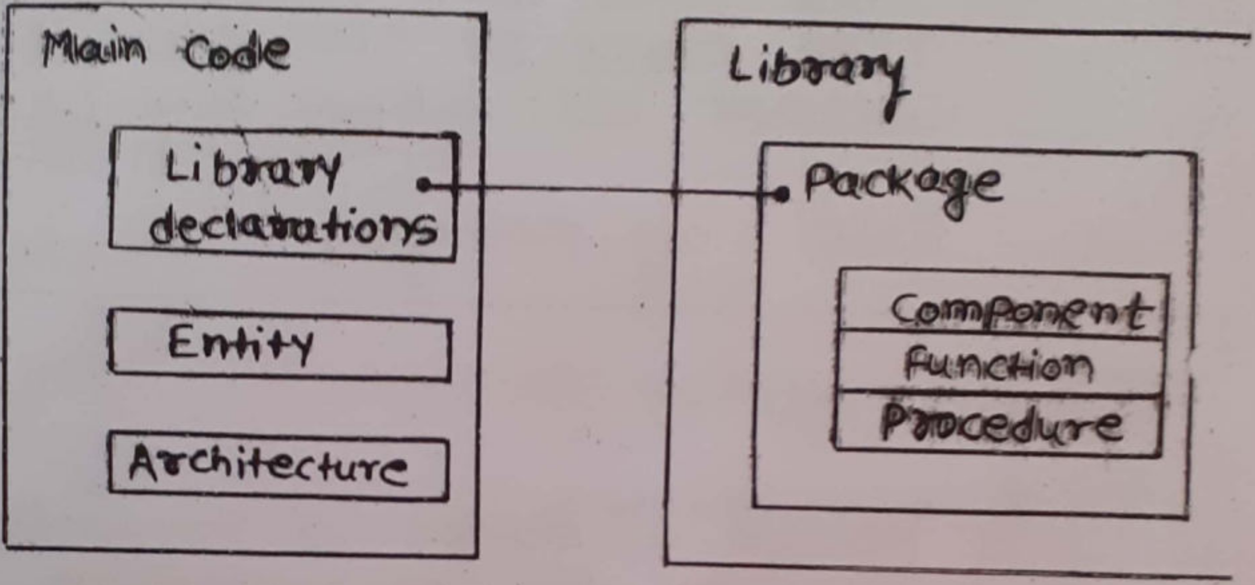
Sub programs → It defines sequential algorithm that performs certain computatio
It consist of procedures and functions.

PACKAGE



Frequently used pieces of VHDL code are usually written in the form of Components, procedures & functions. Such a codes are placed inside package & compiled into destination library.
This allows Code partioning, code sharing and code reuse.

SYNTAX

```
PACKAGE package_name IS
   (declarations)
 END package_name;

[PACKAGE BODY package_name IS
   ( Function & procedure descriptions)
    END package_name;]
```

Example :-

```
LIBRARY ieee ;
use ieee.std-logic-1164.all ;

PACKAGE my-package IS
TYPE state IS (St1, St2, St3, St4);
TYPE Colour IS (red, green, blue);
CONSTANT vec : Std-logic-Vector (7 downto 0)
                  := "11111111";
END my-package ;
```

FUNCTION ⇒

A function is section of sequential code. It is used for data type conversions, logical operations, arithmetic computations & attributes.

SYNTAX :-

```
FUNCTION function-name <parameter list>
  RETURN data-type IS
   [ declarations]
   BEGIN
    ( sequential statements)
   END function-name ;
```

Parameter list specifies the functions input parameters.

<parameter list> = [CONSTANT] const-name : const-type
<Parameter list> = [SIGNAL] signal-name : signal-type

Variables are not allowed

Example ⇒

```
FUNCTION f1 (a, b : INTEGER ;
                  c : std-logic )
  RETURN Boolean IS
  BEGIN
    ( sequential statements )
  END f1 ;
```

FUNCTION CALL ⇒

Examples

X <= Conv- integer (a) → converts a to intiger

Y <= maximum (a, b) → returns largest of a & b

IF X > maximum (a, b) → Compares X to the largest of a, b.

- Function positive-edge ( )

function body

```
FUNCTION positive-edge (SIGNAL Clk : std-logic )
 RETURN BOOLEAN IS
 BEGIN
   RETURN ( Clk'event AND Clk = '1');
 END positive-edge ;
```

function call

```
 ....
 IF positive-edge (clk) Then ....
```

The function above detects a positive (rising) Clock edge. It is similar to
IF (Clk'event and Clk = '1') Statement.

FUNCTION LOCATION

Function Located in main Code

```
Library ieee;
use ieee.std-logic-1164.all;

Entity dff is
  Port (d, clk, rst : IN std-logic;
              q : out std-logic);
 End dff;

Architecture dff OF dff is

Function positive-edge (signal s: std-logic)
 Return Boolean IS
 Begin
     Return s'event and s='1';
  end positive-edge;

 Begin
     process (clk, rst)
  Begin
      If (rst ='1') then q <='0';
  elsif positive-edge (clk) then
               q < = d;

      end if;
     end process;
    end dff;
```

Scanned by CamScanner

FUNCTION Located in a Package

```
 ------ Package ------
Library ieee;
Use   ieee. std-logic-1164. all;

PACKAGE my-package IS
  FUNCTION  positive-edge (SIGNAL S: std-logi
  RETURN  BOOLEAN;
 END  my-package;

PACKAGE BODY my-package IS
 FUNCTION positive-edge (SIGNAL S: std-logi
 RETURN  BOOLEAN  IS
 BEGIN
  RETURN  S'event and s = '1';
  END positive-edge;
 END   my-package;
```

```
 ------ Main code ----------
Library ieee;
USE  ieee.std-logic-1164.all;
USE work.my-package.all;

Entity dff IS
  PORT (d, clK, rst : IN std-logic;
          q : OUT std-logic);
End dff;

Architecture my-arch OF dff IS
Begin
 Process (clK, rst)
 Begin
  If (Rst = '1') then
     q <= '0':
```

```
elsif positive-edge (clk) Then
    q <= d ;
end if ;
end process ;
end my-arch ;
```

PROCEDURE

procedure can return more than one value.

Procedure Body

```
PROCEDURE procedure-name <parameter list,
  IS
 [declarations]
 BEGIN
   (Sequential statements)
 End  procedure-name ;
```

parameter list specifies input and outputs
they may be constant, signal, Variable

procedure can have any number of
IN, OUT, or INOUT parameters

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

```
Library ieee;
Use     ieee.std-logic-1164.all;

Entity  min-max Is
  Generic  (limit : Integer := 255);
  Port     (ena : In  Bit;
            inp1, inp2 : In Integer Range
                              0 to limit;
           min-out, max-out : out Integer Range
                              0 to limit);
end  min-max;

Architecture  my-arch of min-max is

Procedure  Sort (Signal in1, in2 : In integer
                    range  0 to limit;
           signal min, max : OUT Integer
                    range  0 to  limit) is
  Begin
    If  (in1 > in2) then
        max <= in1;
        min  <= in2;
    else
          max <= in2;
          min  <= in1;
   end if;
   end  Sort;
Begin
 Process (ena)
 Begin
If  (ena = '1') then sort (inp1, inp2, min-out,
      max out);
 end if;
end process;
end   my arch.
```

```
Type op-code is ( Add. sub. mul. div );
....
Procedure ALU ( A, B : in integer ;
                op : in op code ;
                z : out integer ) is
begin
    case op is
    when Add => z := A+B ;
    when sub => z := A-B ;
    when mul => z := A * B;
    when Div => z := A/B ;
    when others  null ;
     end case ;
    end ALU ;
```

Parameters may be constants, variables, or signals. & modes may be in, out, inout. If class of parameter is not expiicitly specified by defult a constant if parameter is mode in otherwise it is variable if parameter is of mode <u>out or inout</u>.

PROCEDURE

	FUNCTION	PROCEDURE
i)	A function has zero or more input paramet--ers and a single return value. The input parameters can be constants 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
ii)	A function is called as part of an expre--ssion	procedure is a statemer on its own.
iii)	Requires zero simulation time	may or may not need zero simulation time.
iv)	Debugging & maintai--nance of models consisting function is easy	Debugging & maintainanc of models consisting proc-edures can be very difficult.
v)	Recommended for writing models	Not recommended for writing models
vi)	Wait and compo--nent are not synthesizable	Wait and component are not synthesizabl

possible location of function & procedur is same may be in package or main code - in entity or architecture.