Function

In VHDL, a **function** is a subprogram used to perform calculations or data processing that **returns a single value as a result**. Functions in VHDL are similar to functions in traditional programming languages such as C or Java, where the main purpose is to compute a value based on the input arguments. Unlike procedures, functions **must return exactly one value** and **cannot modify signals or variables outside the function directly**. They are typically used for operations like arithmetic, logical computations, or data conversion.

Functions improve code readability, reusability, and modularity in complex VHDL designs.

Function Declaration

A function is defined using a function declaration. This declaration specifies:

- The function name
- Input parameters (if any)
- The return data type

Below is an example of a function declaration in VHDL:

```
function Function_Name(parameter1: data_type; parameter2: data_type) return return_type is
begin
  -- Function code block
  return value;
end Function_Name;
```

Parameters in Function

A function can accept input parameters only. These parameters are used to pass data into the function to be processed. Unlike procedures, functions cannot have out or inout parameters. All data returned from a function must be provided through the return statement.

Function Body (Code Block)

The body of a function contains the logic used to compute and return a value. This may include arithmetic operations, conditions, or other expressions. The function must include a return statement that provides the final result.

Function Call

A function is called by using its name and passing the required arguments. The returned value can be directly assigned to a signal or variable.

```
variable1 := Function_Name(input_value1, input_value2);
```

Example Code

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity Function_Example is
  port(
    clk : in std_logic;         -- Clock input
    result_out : out integer    -- Output to observe the result
  );
end Function_Example;

architecture Behavioral of Function_Example is

  -- Signal declarations
  signal sigA : integer := 10;
  signal sigB : integer := 20;
  signal function_result : integer;

  -- Function Declaration
  function adder(
    A: integer;
    B: integer
  ) return integer is
  begin
    return A + B;
  end function;

begin

  process(clk)
  begin
    if rising_edge(clk) then
      function_result <= adder(sigA, sigB);  -- Call the function
```

```
    end if;
  end process;

  -- Output assignment
  result_out <= function_result;

end Behavioral;
```

Revision #1
Created 2025-10-23 05:47:15 UTC by CH
Updated 2025-10-23 05:55:02 UTC by CH