Procedure

In VHDL, a *procedure* is a type of language construct used to group several statements and specific tasks into a single block of code. Procedures help in organizing and simplifying the understanding of complex VHDL designs.

A procedure in VHDL is similar to a void function in languages like C. It performs a specific task but does not return a value. Instead, it may modify signals or variables passed to it as parameters, allowing designers to reuse code and improve readability.

Procedure Declaration

A procedure is defined using a procedure declaration. This declaration specifies the procedure name, any required parameters (if any), and the data type that is returned (if applicable). Below is an example of a procedure declaration in VHDL:

```
procedure Nama_Procedure(parameter1: tipe_data; parameter2: tipe_data) return tipe_data
  is
  begin
  -- Blok kode procedure
end Nama_Procedure;
```

Parameters in Procedure
A procedure can accept parameters as arguments. These parameters are used to pass data into the procedure so that it can be processed. The required parameters are defined in the procedure declaration and can be of different modes such as in, out, or inout, depending on whether the data is being read, written, or both.

Procedure Body (Code Block)
The body of a procedure is the section where the tasks to be performed by the procedure are written. Within this block, you can write statements to perform various operations such as calculations, condition checks, data manipulation, and more. This is where the logic of the procedure is implemented, similar to the body of a function in other programming languages.

Procedure Call

To use a procedure, it can be called from the main part of the design or from within another procedure. A procedure is invoked by providing arguments that match the parameters defined in its declaration. Below is an example of how a procedure is used in VHDL.

```
variable1 := Nama_Procedure(nilai_parameter1, nilai_parameter2);
```

Example Code

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

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

architecture Behavioral of Procedure_Example is

  -- Signal declarations
  signal sigA : integer := 5;
  signal sigB : integer := 7;
  signal adder_result : integer;

  procedure adder(
    A, B : in integer;       -- Input parameters
    Hasil : out integer      -- Output parameter (will write to adder_result)
  ) is
  begin
    Hasil := A + B;
  end procedure;

begin

  process(clk)
  begin
    if rising_edge(clk) then
      adder(sigA, sigB, adder_result);  -- Call the procedure
    end if;
  end process;
```

```
  -- Output assignment
  result_out <= adder_result;

end Behavioral;
```

Revision #2
Created 2025-10-23 05:24:51 UTC by CH
Updated 2025-10-23 05:46:54 UTC by CH