Module 7: Procedure, Function, and Impure Function

- Procedure and Function
- Procedure, Function, and Impure Function Synthesis

Procedure and Function

Procedure in VHDL

In VHDL, a "procedure" is a language construct used to group multiple statements and specific tasks into a single block of code. Procedures help organize and understand complex VHDL designs.

Procedure Declaration

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

```
procedure procedure_name(param1: in type1; param2: out type2) is
begin
    -- Procedure body
end procedure_name;
```

Procedures can accept parameters as arguments. These parameters are used to send data into the procedure for processing. The required parameters can be defined in the procedure declaration.

The code block in a procedure is where the tasks to be performed by the procedure are placed. You can write statements in the procedure code block to perform various operations. These statements can include calculations, tests, data manipulation, etc.

Procedure Call

To use a procedure, you can call it from the main part of the design or another procedure. Calling a procedure is done by providing arguments that match the parameters defined in the procedure declaration. Here is an example of using a procedure:

```
variable_name := procedure_name(arg1, arg2);
```

In this example, `arg1` and `arg2` are the arguments passed to the procedure, and the result of the procedure is assigned to `variable_name`. Here's an example of calling a procedure:

```
procedure add_numbers(a: in integer; b: in integer; sum: out integer) is
begin
    sum := a + b;
```

```
end add_numbers;


-- Calling the procedure
variable_name := add_numbers(5, 3);
```

Function in VHDL

In VHDL (VHSIC Hardware Description Language), "function" and "impure function" are two concepts used to create subprograms that can be used in hardware descriptions. The following is an explanation of both:

Function

Procedures in VHDL do not return values directly. Instead, they can modify the values of their parameters or variables within their scope. If a return value is needed, a function should be used instead.

Functions in VHDL are subprograms used to perform calculations or data processing that return a value as a result. You can think of them as mathematical functions in programming. Functions can have input arguments (parameters) that are used in the calculation. The result of the function will depend on the values of the input arguments provided. Here is an example of function usage in VHDL:

```
function function_name(param1: in type1; param2: in type2) return type3 is
begin
    -- Function body
    return result;
end function_name;
```

The code above just shows the basic structure of a function in VHDL. The actual implementation of the function will depend on the specific task it is designed to perform. Here's an example of a function that calculates the sum of two numbers:

```
function add_numbers(a: in integer; b: in integer) return integer is
begin
    return a + b;
end add_numbers;


-- Using the function
variable_name := add_numbers(5, 3);
```

Impure Function

An impure function is a function that can have properties that are unpredictable or changeable when executed. This means that the result of an impure function can depend on external factors unknown to the program, such as the time at which it is executed, random values, or global variables that can change.

An impure function is usually used when its result depends on values outside the input arguments and may change over time. Impure functions cannot be used in hardware descriptions that are deterministic or synchronous, as is commonly expected in VHDL.

```
function random_number return integer is
begin
    return integer'image(random(0, 100));
end random_number;
```

Procedure, Function, and Impure Function Synthesis

In VHDL, both "functions" and "procedures" can be used in the description of hardware. However, it should be understood that hardware synthesis is usually more suitable for implementations based on deterministic and synchronous behavior. Therefore, there are some restrictions on the use of functions and procedures in the context of synthesis:

Procedures

Procedures in VHDL perform tasks without returning values. They can also be used in hardware descriptions to organize operations and code. Hardware synthesis usually replaces a procedure call with a corresponding physical action in the target hardware. Therefore, deterministic procedures can be synthesized. However, there are some limitations in the use of procedures that depend on time streams or behaviors that are difficult to predict. Some VHDL compilers may not support the synthesis of such procedures.

Functions

VHDL functions that do not have impure properties (e.g., produce deterministic values based on input arguments alone) can usually be synthesized well.

Impure Functions

Impure functions, which produce results that are not predictable or depend on external factors, are usually not suitable for deterministic hardware synthesis. Impure functions that depend on random or non-deterministic behavior will not synthesize well because the resulting hardware must be deterministic and predictable.

So, while functions and procedures can be used in hardware descriptions and can be synthesized if they meet specific requirements, impure functions are not usually suitable for VHDL synthesis.

Difference Between It All

Criteria	Procedure	Function	Impure Function
Destination	Performing tasks without returning values	Returns the calculated values	Returning values with unpredictable properties
Arguments	Can have input and output arguments	Can have input arguments only	Can have input arguments only
Return value	No return value	Returns a value	Returns a value
Usage	Used for organizing code and operations	Used for calculations and data processing	Used for calculations and data processing with unpredictable properties
Example	Procedure to add two numbers	Function to add two numbers	Function to generate random numbers
Synthesis	Can be synthesized if deterministic	Can be synthesized if deterministic	Usually not suitable for synthesis

Example

Procedure

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Adder is
    port (
        A, B: in std_logic;
        Sum: out std_logic
    );
end entity;

architecture RTL of Adder is
    procedure add_numbers(a: in std_logic; b: in std_logic; sum: out std_logic) is
    begin
        sum <= a xor b;
    end add_numbers;
begin
    process (A, B)
    begin
        add_numbers(A, B, Sum);
    end process;
end architecture;
```

Function

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Adder is
    port (
        A, B: in std_logic;
        Sum: out std_logic
    );
end entity;

architecture RTL of Adder is
    function add_numbers(a: in std_logic; b: in std_logic) return std_logic is
    begin
        return a xor b;
    end add_numbers;
begin
    process (A, B)
    begin
        Sum <= add_numbers(A, B);
    end process;
end architecture;
```

Impure Function

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.MATH_REAL.ALL;

entity Adder is
    port (
        Sum: out std_logic
    );
end entity;

architecture RTL of Adder is
    function random_number return std_logic is
    begin
        return REAL'(uniform(0.0, 1.0) > 0.5);
    end random_number;
```

```
begin
    process
    begin
        Sum <= random_number;
    end process;
end architecture;
```