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;
```

Revision #1
Created 4 October 2024 02:01:10 by GI
Updated 4 October 2024 02:03:13 by GI