Testbench and Port Mapping

Testbench

In VHDL, a testbench is a module that instantiates the unit under test (UUT) and applies stimulus to it. The stimulus can be a set of input signals, a clock signal, or a reset signal. The testbench also monitors the output signals of the UUT and compares them to the expected results. The testbench can be used to verify the functionality of the UUT and to debug any issues that arise during simulation. There're some benefits of using a testbench:

- It allows you to verify the functionality of your design before you synthesize it.
- It allows you to test your design under different conditions and edge cases.
- It allows you to debug your design by monitoring the signals in the simulation.
- It allows you to automate the testing process by running a set of test cases automatically.

Types of Testbenches

There are several types of testbenches that you can use to test your design:

Simple Testbench

A simple testbench is a basic testbench that applies stimulus to the UUT and monitors the output signals. It is useful for testing simple designs that do not require complex stimulus or verification.

Process Statement Testbench

A process statement testbench is a testbench that uses a process statement to generate stimulus for the UUT. It is useful for testing designs that require sequential stimulus or verification.

Look-up Table Testbench

A look-up table testbench is a testbench that uses a look-up table to generate stimulus for the UUT. It is useful for testing designs that require complex stimulus or verification.

Port Mapping

Port Map is a VHDL construct that allows you to connect the ports of a component to signals in the testbench. It is used to instantiate the UUT in the testbench and to connect the input and output signals of the UUT to the stimulus and monitor signals in the testbench. The syntax of the port map is as follows:

```
UUT_inst : entity work.UUT
    port map (
        input_signal1 => stimulus_signal1,
        input_signal2 => stimulus_signal2,
        output_signal1 => monitor_signal1,
        output_signal2 => monitor_signal2
    );
```

Code above shows an example of a port map for a UUT with two input signals and two output signals. The input signals are connected to the stimulus signals in the testbench, and the output signals are connected to the monitor signals in the testbench.

Example

Here is an example of a testbench that instantiates a UUT with two input signals and two output signals and connects them to the stimulus and monitor signals in the testbench:

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

entity testbench is
end testbench;

architecture tb_arch of testbench is
    signal input_signal1 : std_logic;
    signal input_signal2 : std_logic;
    signal output_signal1 : std_logic;
    signal output_signal2 : std_logic;

    component UUT
        port (
            input_signal1 : in std_logic;
            input_signal2 : in std_logic;
            output_signal1 : out std_logic;
            output_signal2 : out std_logic
```

```
        );
    end component;

begin
    UUT_inst : UUT
        port map (
            input_signal1 => input_signal1,
            input_signal2 => input_signal2,
            output_signal1 => output_signal1,
            output_signal2 => output_signal2
        );

    -- Apply stimulus to the UUT
    input_signal1 <= '0';
    input_signal2 <= '1';

    -- Monitor the output signals of the UUT
    process
    begin
        wait for 10 ns;
        assert output_signal1 = '0' and output_signal2 = '1'
            report "Test failed"
            severity error;
        wait;
    end process;

end tb_arch;
```

In this example, the testbench instantiates a UUT with two input signals and two output signals and connects them to the stimulus and monitor signals in the testbench. The testbench applies stimulus to the UUT by setting the input signals to '0' and '1' and monitors the output signals of the UUT by comparing them to the expected results.

Revision #1
Created 22 September 2024 03:52:27 by GI
Updated 22 September 2024 03:52:46 by GI