

# Testbench Architecture

## Models, Assert, and Report

## Testbench Architecture Models

As mentioned in the previous section, there are three main testbench architecture models:

### Simple Testbench

Works for simple designs with a few inputs and outputs. Values are applied to the inputs, and the outputs are monitored. Each input value is applied with a delay to allow the UUT to process the input and generate the output. This resembles the data-flow style in VHDL, where input signals are directly assigned using `<=`, and changes are triggered after specific times using the `after` keyword.

```
begin
  -- Apply values to the input signals with delays
  input_signal1 <= '0', '1' after 10 ns, '0' after 20 ns;
  input_signal2 <= '1', '0' after 10 ns, '0' after 20 ns;
  -- Monitor the output signals
  assert output_signal1 = '1' and output_signal2 = '0'
    report "Test failed"
    severity error;
  wait;
end process;
```

### Process Statement Testbench

This resembles the behavioral style in VHDL, where a process statement is used, and each line within the process is executed sequentially.

```

begin
  -- Stimulus process
  stimulus : process
  begin
    input_signal1 <= '0';
    input_signal2 <= '1';
    wait for 10 ns;
    input_signal1 <= '1';
    input_signal2 <= '0';
    wait for 10 ns;
    input_signal1 <= '0';
    input_signal2 <= '0';
    wait for 10 ns;
    wait;
  end process stimulus;

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

end process;

```

## Look-up Table Testbench

This extends the process statement approach by storing input combinations in a lookup table (either signal or constant) and assigning values in a for-loop within the process statement.

```

begin
  -- Lookup table for input signals
  type input_table is array (natural range <>) of std_logic_vector(1 downto 0);
  constant input_values : input_table := (
    "00", "01", "10", "11"

```

```

);

-- Stimulus process
stimulus : process
begin
    for i in input_values'range loop
        input_signal1 <= input_values(i)(0);
        input_signal2 <= input_values(i)(1);
        wait for 10 ns;
    end loop;
    wait;
end process stimulus;

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

end process;

```

# Assert and Report

Since testbench are for simulation purposes, it is important to include assertions and reports to verify the correctness of the design. The assert statement checks if a condition is true and reports an error if it is false. The report statement is used to display a message when the condition is false. Assert more likely `printf` in C language.

```

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

```

severity error;

wait;

end process;

---

Revision #1

Created 22 September 2024 03:53:29 by GI

Updated 22 September 2024 03:53:40 by GI