4. Assert and Report Statement

4.1 Assert Statement

The assert statement is used for creating **self-checking testbenches**. It acts like an automated check that constantly monitors a condition. If the condition is **false**, it "asserts" a message, alerting us to a problem without requiring us to manually inspect the waveforms.

The full syntax of an assert statement is:

```
ASSERT condition REPORT "message" SEVERITY level;
```

- `ASSERT condition`: This is the boolean expression that you expect to be true. For example, `(actual_output = expected_output)`.
- `REPORT "message"`: This is the message that gets printed to the simulator's console **only if the condition is false**. It's used to provide context about the failure.
- `SEVERITY level`: This is a crucial part of the statement that tells the simulator how to react to the failure.

For example, if we want to implement the assert statement in our testbench from section 3.1.2, we can implement it as below:

```
tb1: process
    constant period    : time := 20 ns;
    begin
        a <= '0';
        b <= '0';
        wait for period;

        assert ((sum = '0') and (carry = '0'))
        report "tes gagal pada testcase ke-1" severity error;

        a <= '0';
        b <= '1';
        wait for period;

        assert ((sum = '1') and (carry = '0'))
        report "tes gagal pada testcase ke-2" severity error;
```

```
        a <= '1';
        b <= '0';
        wait for period;

        assert ((sum = '1') and (carry = '0'))
        report "tes gagal pada testcase ke-3" severity error;

        a <= '1';
        b <= '1';
        wait for period;

        assert ((sum = '0') and (carry = '1'))
        report "tes gagal pada testcase ke-4" severity error;

        wait; -- wait until the end of time
    end process;
```

The following is the output produced by the testbench:

```
Error: tes gagal pada testcase ke-1
Time: 20 ns  Iteration: 0  Process: /half_adder_tb/tb1
Error: tes gagal pada testcase ke-3
Time: 60 ns  Iteration: 0  Process: /half_adder_tb/tb1
Error: tes gagal pada testcase ke-4
Time: 80 ns  Iteration: 0  Process: /half_adder_tb/tb1
```

4.2 Severity Level

The `severity` level tells the simulator **how serious** the failed assertion is. There are four standard levels:

Level	Description	Simulator Action
`NOTE`	**Informational message**. Used for debugging, tracing, or indicating progress.	Prints the message and continues simulation.
`WARNING`	**Non-critical issue**. Something is unexpected or out of spec, but the design might still function.	Prints a warning and continues simulation. Increments a warning counter.
`ERROR`	**Functional failure**. The design's output is incorrect. This is the standard for a failed test.	Prints an error and continues simulation. Increments an error counter.

Level	Description	Simulator Action
`FAILURE`	**Catastrophic/Fatal error**. Something is fundamentally broken, making further simulation pointless.	Prints a failure message and immediately halts the simulation.

Note that if we don't explicitly specify which `severity` level used in a `report` statement, it automatically defaults to `note`.

Revision #7
Created 2025-09-15 16:23:21 UTC by RE
Updated 2025-09-16 10:14:29 UTC by RE