While Loop

While Loop

The **`while` loop** is used where the number of repetitions is not known from the start. A `while` loop continues to execute as long as a specified condition is `true`.

Syntax

The basic structure of a `while` loop is:

```
loop_label: while <condition> loop
    -- Going through...
    -- IMPORTANT: Must include logic to eventually make the condition false!
end loop loop_label;
```

- **`<condition>`** is a **boolean expression** that is checked before each iteration. If it's `true`, the loop body executes. If it's `false`, the loop terminates.

Warning: Infinite Loops

A `while` loop can create an **infinite loop**. This occurs if the loop's condition never becomes `false`. Unlike in C, Java, Python etc. where infinite loops are harmless, VHDL **shouldn't have these**.

- In a simulation, an infinite loop will cause the simulator to hang and never finish.
- In synthesis, it can be interpreted as a feedback path that creates a latch, or the synthesizer might fail with an error.

To avoid this, ensure that the logic inside the loop will eventually cause the condition to become `false`.

Example: Finding the First '1'

In this example, we'll search a vector from left to right (from the most significant bit) to find the position of the first bit that is a '1'.

Notice that unlike a `for` loop, we must **manually declare and update** our index variable (`i`).

```
-- Inside a process...
p_Find_First_One: process(a_vector)
    -- We must declare our own index variable for a while loop
    variable i : integer := a_vector'left; -- Start at the leftmost bit
    variable i_position : integer := -1; -- -1 if no 1 is found
begin
    -- Reset variables for each run of the process
    i_position := -1;
    i := a_vector'left;

    FIRST_ONE: while (i >= a_vector'right) loop

        if a_vector(i) = '1' then
            i_position := i;
            exit FIRST_ONE; -- Quit if found
        end if;

        -- Manually decrement the index to check the next bit
        i := i - 1;

    end loop SEARCH_LOOP;

    -- Assign the result to a signal
    found_position_signal <= i_position;

end process p_Find_First_One;
```

In this code, the loop continues as long as `i` is within the vector's bounds. If a '1' is found, the `exit` statement terminates the loop early. If no '1' is found, the loop completes naturally when `i` goes out of bounds.

Revision #1
Created 2025-10-03 17:09:05 UTC by JD
Updated 2025-10-03 17:09:39 UTC by JD