Loop Control: Next & Exit Statements

The following are two additional statements that can be used to control the looping construct:

Next

The `next` statement is used to skip the remaining code in the current iteration of the loop and proceed to the next iteration. In a for-loop, the index variable will be incremented or decremented automatically before the next iteration. In a while loop, this behavior depends on the logic applied. There are two ways to use the `next` statement.

```
next when (condition);
if (condition) then
    next;
end if;
```

Exit

The `exit` statement is used to stop the loop forcibly. When this statement is executed, the loop will terminate immediately, and the program will continue executing the code that follows the loop. There are two ways to use the `exit` statement.

```
exit when (condition);
if (condition) then
    exit;
end if;
```

Example

Below is an example of a for-loop that uses the `next` and `exit` statements.

```
process
    variable i: integer := 0;
begin
    for i in 0 to 7 loop
        if i = 3 then
            next; -- Skip the rest of the code in this iteration
        elsif i = 5 then
            exit; -- Exit the loop
        end if;
        reg(i) <= reg(i+1);
    end loop;
end process;
```

This VHDL code snippet demonstrates the use of a for-loop within a process statement. Here's the detailed explanation:

1. **Process Declaration**:
   - The process block is declared, and a variable `i` of type integer is initialized to 0.
2. **For-Loop**:
   - The for-loop iterates over the range from 0 to 7, inclusive. The loop variable `i` is automatically created and incremented with each iteration.
3. **Conditional Statements**:
   - Inside the loop, there are two conditional checks:
     - `if i = 3 then`: If `i` equals 3, the `next` statement is executed. This causes the loop to skip the remaining code in the current iteration and proceed to the next iteration.
     - `elsif i = 5 then`: If `i` equals 5, the `exit` statement is executed. This causes the loop to terminate immediately, and the process continues with the code following the loop.
4. **Register Assignment**:
   - If neither of the above conditions is met, the code `reg(i) <= reg(i+1);` is executed. This assigns the value of `reg(i+1)` to `reg(i)`.
5. **End Loop and Process**:
   - The loop ends after completing the iterations from 0 to 7 unless exited early by the `exit` statement.
   - The process block ends after the loop.

Summary

- The loop iterates from 0 to 7.
- When `i` is 3, the loop skips the current iteration.
- When `i` is 5, the loop exits.
- For other values of `i`, `reg(i)` is assigned the value of `reg(i+1)`.

Revision #1
Created 1 March 2025 15:07:58 by GI
Updated 1 March 2025 15:08:14 by GI