# Process Statement

- A process statement contains sequential statements that describe the functionality of a portion *of an entity in sequential terms. The syntax of a process statement is*
- [ *process-label:* ] ***process [ ( sensitivity-list ) ]***

**begin**

- *sequential-statements; these are ->*
  - *variable-assignment-statement*
  - *signal-assignment-statement*

## Process Statement Cont..

*wait-statement*

*if-statement*

*case-statement*

*loop-statement*

*null-statement*

- *exit-statement*
- *next-statement*
- *assertion-statement*
- *procedure-call-statement*
- *return-statement.*
- **end process [ *process-label*];**

## *Variable Assignment Statement*

- Variables can be declared and used inside a process statement. A variable is assigned a value using the variable assignment statement that typically has the form
- *variable-object := expression;*

## *Variable Assignment Statement cont..*

- Consider the following process statement.

```
process (A)
   variable EVENTS_ON_A: INTEGER := 0;
begin
   EVENTS_ON_A := EVENTS_ON_A+1;
end process;
```

## *Signal Assignment Statement*

- Signals are assigned values using a signal assignment statement *The simplest form of a signal assignment statement is*
- *signal-object* ***<= expression [ after delay-value ];***
- A signal assignment statement can appear within a process or outside of a process. If it occurs outside of a process, it is considered to be a concurrent signal assignment statement.

## *Signal Assignment Statement cont..*

- When a signal assignment statement appears within a process, it is considered to be a sequential signal assignment statement . When a signal assignment statement is executed, the value of the expression is computed and this value is scheduled to be assigned to the signal after the specified delay.

## *Signal Assignment Statement cont..*

- If no after clause is specified, the delay is assumed to be a default delta delay.
- Some examples of signal assignment statements are
- COUNTER <= COUNTER+ "0010"; - Assign after a delta delay.
- PAR <= PAR **xor DIN after 12 ns;**
- Z <= (AO **and A1) or (BO and B1) or (CO and C1) after 6 ns;**

## *Wait Statement*

- The **wait statement provides an alternate way to suspend the execution of a process. There are three basic forms of the wait statement.**
- **wait on *sensitivity-list;***
- **wait until *boolean-expression ;***
- **wait for *time-expression ;***

## Wait Statement cont..

- They may also be combined in a single wait statement. For example,
- **wait on *sensitivity-list until boolean-expression for time-expression-,***
- Some examples of **wait statements are**
- **wait on A, B, C; -- statement 1**
- **wait until (A = B); -- statement 2**
- **wait for 10ns; -- statement 3**
- **wait on CLOCK for 20ns; -- statement 4**
- **wait until (SUM > 100) for 50 ms; -- statement 5**

## *If Statement*

- An if statement selects a sequence of statements for execution based on the value of a condition. The condition can be any expression that evaluates to a boolean value. The general form of an if statement is

**if** ***boolean-expression then*** *sequential-statements*

[ **elsif** ***boolean-expression then***

*sequential-statements ]*

[ **else** *sequential-statements ]*

**end if;**

## *Case Statement*

- The format of a case statement is
- **case *expression is***
  - **when *choices* => *sequential-statements***
  - **when *choices* => *sequential-statements*** [ **when others => *sequential-statements ]***
- **end case;**

## ***Null Statement***

The statement

**null;**

is a sequential statement that does not cause any action to take place and execution continues with the next statement. One example of this statement's use is in an if statement or in a case statement where for certain conditions, it may be useful or necessary to explicitly specify that no action needs to be performed.

## Loop Statement

- ***A loop statement is used to iterate through a set of sequential statements. The syntax of a loop statement is***

- [ *loop-label : ] iteration-scheme* ***loop***
  - *sequential-statements*
- **end loop [ *loop-label ] ;***

# Loop Statement cont..

- There are three types of iteration schemes. The first is the for iteration scheme that has the form

**for *identifier in range***

An example of this iteration scheme is

```
FACTORIAL := 1;
for NUMBER in 2 to N loop
    FACTORIAL := FACTORIAL * NUMBER;
end loop;
```

## *Loop Statement cont..*

- The second form of the iteration scheme is the while scheme that has the form

**while *boolean-expression***

An example of the while iteration scheme is

J:=0;SUM:=10;

WH-LOOP: **while J < 20 loop - This loop has a label, WH_LOOP.**

SUM *:= SUM * 2;*

J:=J+3;

**end loop;**

## *Loop Statement cont..*

- The third and final form of the iteration scheme is one where no iteration scheme is specified.
- In this form of **loop statement, all statements in the loop body are repeatedly executed until some other action causes it to exit the loop.**
- **These actions can be caused by an exit statement, a next statement, or a return statement.**

# Loop Statement cont..

- Here is an example.

```
SUM:=1;J:=0;
   L2: loop
   J:=J+21;
   SUM := SUM* 10;
   exit when SUM > 100;
end loop L2;
```

## Exit Statement

- The exit statement is a sequential statement that can be used only inside a loop. It causes execution to jump out of the innermost loop or the loop whose label is specified. The syntax for an exit statement is
- **exit [ *loop-label] [ when condition ]:***
- If no loop label is specified, the innermost loop is exited.

## *Next Statement*

- The next statement is also a sequential statement that can be used only inside a loop. The syntax is the same as that for the exit statement except that the keyword next replaces the keyword exit. Its syntax is
- **next [ *loop-label] [ when condition ];***
- The **next statement results in skipping the remaining statements in the current iteration of the specified loop and execution resumes with the first statement in the next iteration of this loop. If no loop label is specified, the innermost loop is assumed.**

## *Next Statement cont..*

- In contrast to the exit statement that causes the loop to be terminated (i.e., exits the specified loop), the next statement causes the current loop iteration of the specified loop to be prematurely terminated and execution resumes with the next iteration.

# *Next Statement cont..*

Here is an example.

```
for J in 10 downto 5 loop
    if (SUM < TOTAL_SUM) then
        SUM := SUM +2;
    elsif (SUM = TOTAL_SUM) then
        next;
    else
        null;
    end if;
        K:=K+1;
end loop;
```