Assignment Statements

- Concurrent Assignment Statement.
- Sequential Assignment Statement

Concurrent Assignment Statement

1- Simple Signal Assignment

It used for a logic or arithmetic expression.
The general form is:-

Signal_name <= expression;

Example:

```
y <= a AND b;
```

2- Selected Signal Assignment

It used to assign one of several values based on selection criterion used with keyword.
The general form is:-

With expression **select** Signal_name <= expression **when** constant_value;

Example: Modeling of a 4-1 multiplexer

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity mux4_1 is
port (sel: in std_logic_vector (1 downto 0);
a, b, c, d: in std_logic;
y: out std_logic);
end entity mux4_1;
architecture rtl of mux4_1 is
begin
with sel select
y <= a when “00”,
b when “01”,
c when “10”,
d when “11”,
a when others;
```

```
end architecture rtl;
```

3- Conditional Signal Assignment
It used to set a signal to one of several values.

The general form is:-

Signal_name <= expression **when** logic_expression **else** expression ;

Example: Modeling of a 4-1 multiplexer

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity mux4_1 is
port (sel: in std_logic_vector (1 downto 0);
a, b, c, d: in std_logic;
y: out std_logic);
end entity mux4_1;
architecture rtl of mux4_1 is
begin
y <= a when sel = "00" else
b when sel = "01" else
c when sel = "10" else
d; --when sel = "11"
end architecture rtl;
```

Sequential Assignment Statement

- Case Statement
- If-Then-Else-Statement

A process statement is the main construct that allows you to use sequential statements to describe the behavior of a system over time.
The syntax for a process statement is:

[*process_label*:] **process** (*sensitivity_list*)
Variable declarations
Begin

[if-then-else-statement]

[case-ststement]

End process;

1- Case Statement

General form:

Case expression **is**
When constant_value=> statement;
⋮
When others=> statement;

End case;

Example: Modeling of a 4-1 multiplexer

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity mux4_1 is
port (sel: in std_logic_vector (1 downto 0);
a, b, c, d: in std_logic;
y: out std_logic);
end entity mux4_1;
architecture rtl of mux4_1 is
begin
process (sel, a, b, c, d)
begin
case sel is
when "00" => y <= a;
when "01" => y <= b;
when "10" => y <= c;
when "11" => y <= d;
when others => y <= a;
end case;
end process;
end architecture rtl;
```

1- IF-THEN-ELSE Statement

General form:

IF expression **THEN** statement;
ELSIF expression **THEN** statement;
ELSE statement;
End IF;

Example: Modeling of a 4-1 multiplexer

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity mux4_1 is
```

```
port (sel: in std_logic_vector (1 downto 0);
a, b, c, d: in std_logic;
y: out std_logic);
end entity mux4_1;
architecture rtl of mux4_1 is
begin
process (sel, a, b, c, d)
begin
if (sel = “00”) then
y <= a;
elsif (sel = “01”) then
y <= b;
elsif (sel = “10”) then
y <= c;
else
y <= d;
end if;
end process;
end architecture rtl;
```