# VHDL Syntax Reference

By Prof. Taek M. Kwon  
ECE Dept, University of Minnesota Duluth

This summary is provided as a quick lookup table for searching the VHDL syntax and referencing a given example for your code. Please click on the topic to go to the corresponding page.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Signal Assignment</td>
<td>2</td>
</tr>
<tr>
<td>2. Variables</td>
<td>2</td>
</tr>
<tr>
<td>3. If-then-else Statement</td>
<td>3</td>
</tr>
<tr>
<td>3. Case Statement</td>
<td>3</td>
</tr>
<tr>
<td>5. For Loop</td>
<td>4</td>
</tr>
<tr>
<td>6. While Loop</td>
<td>4</td>
</tr>
<tr>
<td>7. Infinite Loop</td>
<td>5</td>
</tr>
<tr>
<td>8. Conditional Signal Assignment</td>
<td>5</td>
</tr>
<tr>
<td>9. Selected Signal Assignment</td>
<td>5</td>
</tr>
<tr>
<td>10. Wait Statements</td>
<td>6</td>
</tr>
</tbody>
</table>
1. Signal Assignment

signal R: std_logic;
signal Q: std_logic_vector(7 downto 0);

A single bit is enclosed using a single quotation:
R <= ‘1’;

Multiple bits are enclosed using a double quotation:
Q <= “10110000”;

Hexadecimals are represented using X”…”:
Q <= X”B0”;

2. Variables

Variables are objects used to store intermediate values between sequential VHDL statements. Variables are only allowed in processes, procedures, and functions, and they are always local to those functions. When a value is assigned to a variable, “:=” is used.

Example:

signal Grant, Select: std_logic;

process(Rst, Clk)
variable Q1, Q2, Q3: std_logic;
begin
if Rst='1' then
   Q1 := ‘0’; Q2 := ‘0’; Q3 := ‘0’;
elseif (Clk='1' and Clk'event) then
   Q1 := Grant;
   Q2 := Select;
   Q3 := Q1 or Q2;
end if;
end process;

Note: Both signals and variables carry data from place to place. However, you must always use signals to carry information between concurrent elements of your design.
3. If-then-else Statement

Syntax:

```plaintext
if Boolean_expr_1 then
    sequential_statements;
elsif Boolean_expr_2 then
    sequential_statements;
elsif Boolean_expr_3 then
    ...
elself
    sequential_statements;
end if;
```

Example:

```plaintext
process (a, b, m, n)
begin
    if m = n then
        r <= a + b;
    elsif m > 0 then
        r <= a - b;
    else
        r <= a + 1;
    end if;
end;
```

3. Case Statement

Syntax:

```plaintext
case sel is
    when choice_1 =>
        sequential_statements;
    when choice_2 =>
        sequential_statements;
    . . .
    when others =>
        sequential_statements;
end case;
```
Example:

```vhdl
    case sel is
    when "00" =>
        r <= a + b;
    when "10"
        r <= a - b;
    when others =>
        r <= a + 1;
    end case;
```

5. For Loop

Syntax:

```vhdl
    for index in loop_range loop
        sequential statements;
    end loop;
```

Example:

```vhdl
    constant MAX: integer := 8;
    signal a, b, y: std_logic_vector(MAX-1 downto 0);
    ...
    for i in (MAX-1) downto 0 loop
        y(i) <= a(i) xor b(i)
    end loop;
```

6. While Loop

Syntax:

```vhdl
    loop_name: while (condition) loop
        ---repeated statements
    end loop loop_name;
```

Example:

```vhdl
    while error_flag /= '1' and done /= '1' loop
        Clock <= not Clock;
        wait for CLK_PERIOD/2;
    end loop;
```
7. Infinite Loop

Syntax:

```vhdl
loop_name: loop

...  
   exit when (condition)
end loop loop_name;
```

Example:

```vhdl
loop
   Clock <= not Clock;
   wait for CLK_PERIOD/2;
   if done = ‘1’ or error_flag = ‘1’ then
      exit;
   end if;
end loop;
```

8. Conditional Signal Assignment

Syntax:

```vhdl
signal_name <= value_expr_1 when Boolean_expr_1 else
               value_expr_2 when Boolean_expr_2 else
               value_expr_3 when Boolean_expr_3 else
               ... value_expr_n
```

Example:

```vhdl
x <= a when (s=”00”) else
   b when (s=”01”) else
   c when (s=”10”) else
   d when (x=”11”) else
   ‘X’;
```

9. Selected Signal Assignment

Syntax:

```vhdl
with select_expression select
   signal_name <= value_expr_1 when choice_1,
```

```vhdl
value_expr_2 when choice_2,
....
value_expr_n when choice_n;

Example:

    with s select
      x <= a when "00",
             b when "01",
             c when "10",
             d when others;

10. Wait Statements

    wait on signals;
    wait until Boolean_expr;
    wait for time_expr;