VHDL Language Reference

LIBRARY and USE clauses:

```
LIBRARY __library_name;
USE __library_name.__package_name.ALL;
```

Some frequently used library and package names:

ieee.std_logic_1164	(for STD_LOGIC)
ieee.std_logic_unsigned	(to treat STD_LOGIC as INTEGER)
ieee.std_logic_arith	(contains conversion functions)
lpm.lpm_components	

Entity declaration:

```
ENTITY __entity_name IS
    PORT(
        __input_name, __input_name      : IN        STD_LOGIC;
        __input_vector_name             : IN STD_LOGIC_VECTOR(max downto min);
        __output_name, __output_name    : OUT       STD_LOGIC;
        __output_name, __output_name    : BUFFER    STD_LOGIC;
        __output_vector_name            : OUT STD_LOGIC_VECTOR(max downto min));
END __entity_name;
```

Architecture body:

```
ARCHITECTURE a OF __entity_name IS
    signal, type, and component declarations
BEGIN
     statements (e.g., signal assignment, process, or component instantiation)
END a;
```

Concurrent Signal Assignment statement (global):

```
__signal_name <= expression;
```

Variable Assignment statement (local to a process):

```
__variable_name := expression;
```

Selected Signal Assignment Statement:

```
WITH signal_name SELECT
    assigned_signal <=constant_value_1   WHEN expression_1,
                      constant_value_2   WHEN expression_2,
                      constant_value_n   WHEN expression_n,
                      default_value      WHEN others;
```

Signal declaration (before BEGIN in architecture body):

```
SIGNAL  __signal_name1 : STD_LOGIC;
SIGNAL  __signal_name2 : STD_LOGIC_VECTOR(max downto min);
```

Component declaration (before BEGIN in architecture body):

```
COMPONENT __component_name
    PORT(
        __input_name, __input_name      : IN        __type;
        __output_name, __output_name    : OUT       __type);
END COMPONENT;
```

Component instantiation statement (inside architecture body):

```
__instance_name: component_name
    GENERIC MAP (   parameter name => parameter value,
                    parameter name => parameter value)
    PORT MAP (  __component_port => __user_port,
                __component_port => __user_port);
```

LPM Component instantiation statement:

```
__instance_name: __lpm_component_name
    GENERIC MAP (   LPM_PARAMETER => __value,
                    LPM_PARAMETER => __value)
    PORT MAP (  __component_port => __user_port,
                __component_port => __user_port);
```

GENERATE statement:

```
__generate_label:
FOR __index_variable IN __range GENERATE
   __statement;
   __statement;
END GENERATE;
```

PROCESS statement:

```
PROCESS (sensitivity list)
    VARIABLE __variable_name : __type;
BEGIN
    statements; (e.g., IF, CASE, etc.)
END PROCESS;
```

IF statement (must be inside a process):

```
IF __expression THEN
   __statement;
   __statement;
ELSIF __expression THEN
   __statement;
   __statement;
ELSE
   __statement;
   __statement;
END IF;
```

CASE statement (must be inside a PROCESS):

```
CASE __expression IS
    WHEN __case1 =>
       __statement;
       __statement;
    WHEN __case2 =>
       __statement;
       __statement;
    WHEN others =>
       __statement;
       __statement;
END CASE;
```

Evaluating a clock edge

```
IF(clock'EVENT and clock = '1')THEN  -- positive-edge clock
      __statements
END IF;
```

Clearing a vector of unspecified width:

```
q <= (others => '0');
```

Setting a vector of unspecified width:

```
q <= (others => '1');          -- Vector elements all HIGH
q <= (others => 'Z');          -- High-impedance state of tristate buffer
```

Constants:

STD_LOGIC or BIT	single quotes (e.g., '0', '1')
STD_LOGIC_VECTOR or BIT_VECTOR	double quotes (e.g., "0111")
INTEGER	no quotes (e.g., 255)