VHDL Cheat-Sheet

Basic Structure

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity entity_name is
    Generic(
       constant_name : integer := 0;
       constant_name : integer := 1
    );
    Port(
        signal_name : in std_logic;
        signal_name : out std_logic;
        signal_name : inout std_logic;
        signal_name : buffer std_logic
    );
end entity;

architecture architecture_name of entity_name is
    -- Declarations
begin
    -- Statements
end architecture;
```

Data Types

- **std_logic**: 'U' (uninitialized), 'X' (unknown), '0', '1', 'Z' (high impedance), 'W' (weak unknown), 'L' (weak 0), 'H' (weak 1), '-'.
- **std_logic_vector**: Array of `std_logic`.
- **integer**: Integer numbers.
- **bit**: '0' or '1'.
- **bit_vector**: Array of `bit`.

Operators

- **Arithmetic**: `+`, `-`, `*`, `/`
- **Relational**: `=`, `/=`, `<`, `<=`, `>`, `>=`
- **Logical**: `and`, `or`, `nand`, `nor`, `xor`, `not`

Concurrent Statements

- **Signal Assignment**: `<=`

```
signal_a <= signal_b;
```

- **Conditional Signal Assignment**:

```
signal_a <= signal_b when condition else signal_c;
```

- **Selected Signal Assignment**:

```
with selector select
    signal_a <= value1 when choice1,
                value2 when choice2,
                value3 when others;
```

- **Process**:

```
process (sensitivity_list)
begin
    -- Sequential statements
end process;
```

Sequential Statements (Inside Process)

- **Variable Assignment**: `:=`

```
variable_a := value;
```

- **If Statement**:

```
if condition then
    -- Statements
elsif another_condition then
    -- Statements
else
    -- Statements
end if;
```

- **Case Statement**:

```
case expression is
    when choice1 =>
        -- Statements
    when choice2 =>
        -- Statements
    when others =>
        -- Statements
end case;
```

- **Loop Statements**:
  - **For Loop**:

```
for i in range loop
    -- Statements
end loop;
```

– **While Loop**:

```
while condition loop
    -- Statements
end loop;
```

Functions

- **Function Declaration**:

```
function function_name (parameter_list) return return_type is
begin
    -- Function body
end function_name;
```

- **Example Function**:

```
function add (a, b: integer) return integer is
begin
    return a + b;
end add;
```

- **Calling a Function**:

```
variable result : integer;
result := add(5, 3);
```

- **Using Functions in Architectures**:

```
architecture Behavioral of example_entity is
    signal sum : integer;
begin
    sum <= add(5, 3);
end Behavioral;
```

Case-When Statements

- **Case Statement in Process**:

```
process (selector)
begin
    case selector is
        when "00" =>
            output <= "0001";
        when "01" =>
            output <= "0010";
        when "10" =>
```

```
            output <= "0100";
        when "11" =>
            output <= "1000";
        when others =>
            output <= "XXXX";
    end case;
end process;
```

- **Selected Signal Assignment**:

```
with selector select
    signal_a <= "0001" when "00",
                "0010" when "01",
                "0100" when "10",
                "1000" when "11",
                "XXXX" when others;
```

Component Instantiation

```
component component_name
    Port ( port_list : in std_logic_vector);
end component;

signal signal_name : std_logic_vector(size downto 0);

instance_name : component_name
    Port map (port_mapping);
```

Entity Instantiation

```
    use work.file_name.entity_name;

    instance_name : entity_name
        Generic map();
        Port map();
```

Packages and Libraries

- **Library Declaration**:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

- **Package Declaration**:

```
package package_name is
    -- Declarations
end package package_name;
```

- **Package Body**:

```
package body package_name is
    -- Definitions
end package body package_name;
```

- **Using a Package**:

```
library library_name;
use library_name.package_name.all;
```