Lecture 2: Supplemental Material - A First Look at VHDL

The objective of this supplemental material is to give you an early presentation of some of the most important concepts in VHDL. You should keep this document as a reference for future work on your course assignments.

```
-- entity & architecture template
library lib_name;
use lib_name.package_name.all;

entity entity_name is
  generic
  (
    generic_name : type_name := default;
    generic_name : type_name := default
  );
  port
  (
    port_name : in|out|inout|buffer|linkage type_name;
    port_name : in|out|inout|buffer|linkage type_name
  );
end entity_name;

architecture arch_name of entity_name is
  signal signal_name : type_name := default;
begin
  concurrent assignments and processes;
end arch_name;

-- component declaration
component component_name
  generic
  (
    generic_name : type_name := default;
    generic_name : type_name := default
  );
  port
  (
    port_name : in|out|inout|buffer|linkage type_name;
    port_name : in|out|inout|buffer|linkage type_name
  );
end component_name;

-- component instantiation
instance_name : component_name
  generic map
  (
    generic_name => value,
    generic_name => value
  );
  port map
  (
    port_name => value,
    port_name => value
  );
```
-- process template

process_name : process( signal_port_name, signal_port_name )
  variable var_name : type_name := default;
begin
...
end process process_name;

-- concurrent signal assignments

signal_name <= value;
signal_name <= transport value after time_value,
transport value after time_value;
access_name <= new type_name ( initial_value );
signal_name <= value1 when ( condition1 ) else
  value2 when ( condition2 ) else
  value3;
with expression select
signal_name <= value1 when choice1,
  value2 when choice2,
  value3 when others;

-- type declarations

type type_name is ( ENUM1, ENUM2, ENUM3 );
type type_name is range low_integer to high_integer
  units
    base_unit;
    unit1 = integer base_unit;
    unit2 = integer unit1;
end units;
type type_name is array ( low_index to high_index ) of element_type;
type type_name is array ( high_index downto low_index ) of element_type;
type type_name is array ( scalar_type1 range <> ) of element_type;
type type_name is array ( index1, index2 ) of element_type;
type type_name is
  record
    element_name : type_name;
    element_name : type_name;
  end record;
type record_type_name;
type pointer_type_name is access record_type_name;
type record_type_name is
  record
    next_record : pointer_type_name;
  end record;
type file_type_name is file of type_name;
subtype subtype_name is scalar_type range low to high;
subtype subtype_name is array_type( left downto/to right );
subtype subtype_name is resolution_fn type_name;

-- signal/constant/variable declarations
---------------------------------------------------------------
signal signal_name : type_name := default;
constant const_name : type_name := value;
variable var_name : type_name := default;
file file_id : file_type is in/out file_name;

-- procedure & function declaration
-------------------------------
procedure proc_name ( constant/variable/signal param : in/out/inout type_name );
procedure proc_name ( param1 : type_name; param2 : type_name );
function fn_name ( constant/variable/signal param : in/out/inout type_name ) return type_name;
function fn_name ( param1 : type_name; param2 : type_name ) return type_name;

-- procedure & function body
-------------------------------
procedure proc_name ( constant/variable/signal param : in/out/inout type_name ) is
  variable var_name : type_name := default;
begin
  statements;
end proc_name;
function fn_name ( constant/variable/signal param : in/out/inout type_name ) return type_name is
  variable var_name : type_name := default;
begin
  statements;
  return value;
end proc_name;

-- if statement
-------------------
if ( condition1 ) then
  statements;
elsif ( condition2 ) then
  statements;
else
  statements;
end if;

-- case statement
case signal/variable is
  when value1 => statements;
  when value2 => statements;
  when others => statements;
end case;

-- while loop
label : while ( condition ) loop
  statements;
end loop label;

-- for loop
label : for var_name in left to/downto right loop
  statements;
end loop label;

-- assert statement
assert ( condition )
  report string_value
  severity severity_value;

-- package declaration and body
package pkg_name is
  declarations;
end pkg_name;
package body pkg_name is
  definitions;
end pkg_name;

-- configurations
configuration cfg_name of entity_name is
  for arch_name
    end for;
end cfg_name;
configuration cfg_name of entity_name is
  for arch_name
    for instance_name : comp_name use entity entity_name ( architecture );
    for instance_name : comp_name
      for arch_name
        end for;
      end for;
    end for;
  end for;
configuration cfg_name of entity_name is
  for arch_name
    for instance_name : comp_name use configuration cfg_name2;
for others : comp_name use configuration cfg_name2;
for all : comp_name use configuration cfg_name2;
end for;
end cfg_name;