There are four types of objects in VHDL:
- Constants
- Variables
- Signals
- Files

The scope of an object is as follows:
- Objects declared in a package are available to all VHDL descriptions that use that package.
- Objects declared in an entity are available to all architectures associated with that entity.
- Objects declared in an architecture are available to all statements in that architecture.
- Objects declared in a process are available only within that process.
VHDL Objects: Constants

- Name assigned to a specific value of a type
- Allow for easy update and readability
- Declaration of constant may omit value so that the value assignment may be deferred
  - Facilitates reconfiguration
- Declaration syntax:

  ```vhdl
  CONSTANT constant_name : type_name [:= value];
  ```

- Declaration examples:

  ```vhdl
  CONSTANT PI : REAL := 3.14;
  CONSTANT SPEED : INTEGER;
  ```
**VHDL Objects: Variables**

- Provide convenient mechanism for local storage
  - E.g. loop counters, intermediate values

- Scope is process in which they are declared

- All variable assignments take place immediately
  - No delta or user specified delay is incurred

- Declaration syntax:

```vhdl
VARIABLE variable_name : type_name [:= value];
```

- Declaration examples:

```vhdl
VARIABLE opcode : BIT_VECTOR(3 DOWNTO 0) := "0000";
VARIABLE freq : INTEGER;
```
VHDL Objects: Signals

- Used for communication between VHDL components
- Real, physical signals in system often mapped to VHDL signals
- ALL VHDL signal assignments require either delta cycle or user-specified delay before new value is assumed

Declaration syntax:

```
SIGNAL signal_name : type_name [:= value];
```

Declaration and assignment examples:

```
SIGNAL brdy : BIT;
brdy <= '0' AFTER 5ns, '1' AFTER 10ns;
```
Signals and Variables

- This example highlights the difference between signals and variables

ARCHITECTURE test1 OF mux IS
  SIGNAL x : BIT := '1';
  SIGNAL y : BIT := '0';
BEGIN
  PROCESS (in_sig, x, y)
  BEGIN
    x <= in_sig XOR y;
    y <= in_sig XOR x;
  END PROCESS;
END test1;

ARCHITECTURE test2 OF mux IS
  SIGNAL y : BIT := '0';
BEGIN
  PROCESS (in_sig, y)
  VARIABLE x : BIT := '1';
  BEGIN
    x := in_sig XOR y;
    y <= in_sig XOR x;
  END PROCESS;
END test2;

- Assuming a 1 to 0 transition on in_sig, what are the resulting values for y in the both cases?
A key difference between variables and signals is the assignment delay

ARCHITECTURE sig_ex OF test IS
  PROCESS (a, b, c, out_1)
  BEGIN
    out_1 <= a NAND b;
    out_2 <= out_1 XOR c;
  END PROCESS;
END sig_ex;

<table>
<thead>
<tr>
<th>Time</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>out_1</th>
<th>out_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1+d</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1+2d</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
ARCHITECTURE var_ex OF test IS
BEGIN
   PROCESS (a, b, c)
   VARIABLE out_3 : BIT;
   BEGIN
      out_3 := a NAND b;
      out_4 <= out_3 XOR c;
   END PROCESS;
END var_ex;

Time   a  b  c  out_3  out_4
------  ---- ---- ---- ---- ----
  0     0  1  1    1    0
  1     1  1  1    0    0
1+d    1  1  1    0    1
VHDL Objects: Files

- Files provide a way for a VHDL design to communicate with the host environment.
- File declarations make a file available for use to a design.
- Files can be opened for reading and writing.
  - In VHDL87, files are opened and closed when their associated objects come into and out of scope.
  - In VHDL93 explicit `FILE_OPEN()` and `FILE_CLOSE()` procedures were added.
- The package STANDARD defines basic file I/O routines for VHDL types.
All declarations of VHDL ports, signals, and variables must specify their corresponding type or subtype.
VHDL Data Types: Scalar Types

- **Integer**
  - Minimum range for any implementation as defined by standard: -2,147,483,647 to 2,147,483,647
  - Example assignments to a variable of type integer:

```vhdl
ARCHITECTURE test_int OF test IS
BEGIN
  PROCESS (X)
  VARIABLE a: INTEGER;
  BEGIN
    a := 1;  -- OK
    a := -1;  -- OK
    a := 1.0;  -- illegal
  END PROCESS;
END test_int;
```
ARCHITECTURE test_real OF test IS
BEGIN
  PROCESS (X)
    VARIABLE a: REAL;
  BEGIN
    a := 1.3; -- OK
    a := -7.5; -- OK
    a := 1; -- illegal
    a := 1.7E13; -- OK
    a := 5.3 ns; -- illegal
  END PROCESS;
END test_real;

VHDL Data Types: Scalar Types (Cont.)

• Real
  - Minimum range for any implementation as defined by standard: -1.0E38 to 1.0E38
  - Example assignments to a variable of type real:


**Enumerated**
- User specifies list of possible values
- Example declaration and usage of enumerated data type:

```vhdl
TYPE binary IS ( ON, OFF );
... some statements ...
ARCHITECTURE test_enum OF test IS
BEGIN
    VARIABLE a: binary;
    BEGIN
        a := ON;  -- OK
        ... more statements ...
        a := OFF;  -- OK
        ... more statements ...
    END PROCESS;
END test_enum;
```
Physical
- Require associated units
- Range must be specified
- Example of physical type declaration:

```vhdl
TYPE resistance IS RANGE 0 TO 10000000

UNITS
  ohm;  -- ohm
  Kohm = 1000 ohm;  -- i.e. 1 kΩ
  Mohm = 1000 kohm;  -- i.e. 1 MΩ
END UNITS;
```

- Time is the only physical type predefined in VHDL standard
VHDL Data Types: Composite Types

• Array
  – Used to group elements of the same type into a single VHDL object
  – Range may be unconstrained in declaration
    • Range would then be constrained when array is used
  – Example declaration for one-dimensional array (vector):
    ```vhdl
    TYPE data_bus IS ARRAY(0 TO 31) OF BIT;
    ```

0...element indices...31

<table>
<thead>
<tr>
<th>0</th>
<th>...array values...</th>
<th>1</th>
</tr>
</thead>
</table>

VARIABLE X : data_bus;
VARIABLE Y : BIT;

Y := X(12);  -- Y gets value of element at index 12
Example one-dimensional array using DOWNTO:

```
TYPE reg_type IS ARRAY(15 DOWNTO 0) OF BIT;
```

```
VARIABLE X : reg_type;
VARIABLE Y : BIT;

Y := X(4);  -- Y gets value of element at index 4
```

DOWNTO keyword must be used if leftmost index is greater than rightmost index.
• Records
  – Used to group elements of possibly different types into a single VHDL object
  – Elements are indexed via field names
  – Examples of record declaration and usage:

```vhdl
TYPE binary IS ( ON, OFF );
TYPE switch_info IS
  RECORD
    status : BINARY;
    IDnumber : INTEGER;
  END RECORD;

VARIABLE switch : switch_info;
switch.status := ON;  -- status of the switch
switch.IDnumber := 30;  -- e.g. number of the switch
```
**VHDL Data Types: Subtypes**

- **Subtype**
  - Allows for user defined constraints on a data type
    - e.g. a subtype based on an unconstrained VHDL type
  - May include entire range of base type
  - Assignments that are out of the subtype range are illegal
    - Range violation detected at run time rather than compile time because only base type is checked at compile time

- Subtype declaration syntax:

```vhdl
SUBTYPE name IS base_type RANGE <user range>;
```

- Subtype example:

```vhdl
SUBTYPE first_ten IS INTEGER RANGE 0 TO 9;
```