VHDL BASIC ELEMENTS

INTRODUCTION

VHDL Basic elements

Identifiers

- Basic identifiers
- Extended identifiers

Data Objects

- Constant
- Variable
- Signal
- File

Data Types

- Scalar
- Composite
- Access
- File type

Identifiers

□ Can be defined as the identification of any data type

Basic identifier: is composed of sequence of ne or more character.

1. It must begin with alphabetic character (a-z or A-Z).
2. It is case insensitive.
3. It can contain alphanumeric and underscores.
4. It cannot contain spaces.
5. No two consecutive underscores are allowed.
6. No VHDL keyword allowed.

Identifiers

- Extended identifiers: is a sequence of characters written between two backslashes. (/ ……. /)

1. Any character can be allowed like @,%,$,#,!.
2. Case sensitive .
3. It may contain spaces and consecutive undescores.
4. VHDL keywords are allowed.

Data Objects

- A data object holds a value of a specific type.

  eg. Variable count: integer

  Eg. Constant count: integer

Constant: this type f object can hold a single value. The value cannot be changed during the time of simulation. Eg. Constant count: integer:=5

(Deferred constant: constant without initialized value)

Data Objects

- Variable: this type of object an hold different values at different times. It is declared within a block, process, procedure or function.

eg. Variable count: integer

eg. Variable sum: integer range 0 to 100:=10

- Signal: It holds a list of values, that include the current value of the signal, and set of possible future values that are to be appeared on the signal.

eg. Signal clock: bit

Signal clock: bit:=10ns

Data Objects

- File: this class contains a sequence of values. Values can be read or written to the file using read and write procedures.
- Eg. File math : text open read_mode is "/usr/home/add.doc.

DATA TYPES

- Every data object in VHDL can hold a value that belongs to set of values. This set of values is specified by using type declarations.

Categorized into 4 major categories:

- Scalar types
- Composite types
- Access types
- File types

a.SCALAR TYPES

□ Values belonging to these types appear in sequential order. i.e. these types are ordered.

i. Enumeration: An enumeration type declaration defines a type that has a set of user-defined values consisting of identifiers and character literals.

□ SYNTAX:

TYPE type_name ***IS*** (   )

□ E.g.

TYPE MVL ***IS*** (‘u’,’0’,’1’,’z’);

ii. Integer: These values fall within a specified integer range.

- □ SYNTAX:

TYPE type_name ***IS*** range value;

- □ E.g.

TYPE INDEX ***IS*** range 0 to15;

iii. Physical: Contains values that represent measurement of physical quantity like, time, length, volume, or current.

- □ SYNTAX:

TYPE type_name ***IS*** range value;

- □ E.g.

TYPE CURRENT ***IS*** range 0 to 1e9;

Units nA;

iv. Floating point: Has set of values in a given range of real numbers.

- □ SYNTAX:

TYPE type_name ***IS*** range value;

- □ E.g.

TYPE itl_voltage ***IS*** range -5.5 to -1.4;

b. COMPOSITE TYPES

□ Composite types represent a collection of values.

i. Array type: An object of an array type consists of that have same type of elements.

□ SYNTAX:

TYPE type_name ***IS ARRAY*** values;

□ E.g.

□ ***TYPE*** address_word ***IS ARRAY*** (0 to 63) ***of*** bit;

ii. Record type

- It consists of elements of different data type.
- SYNTAX:

TYPE type_name ***IS RECORD***

Values;

--;

END RECORD;

- E.g.

TYPE birthday ***IS RECORD***

Day: integer range (0 to 31);

Month: month_1;

END RECORD;

C. ACCESS TYPES

Values belonging to an access type are pointers to a dynamically allocated object of some other type.

□ E.g. ***TYPE*** PTR ***IS ACCESS*** MODULE;

TYPE FIFO ***IS ARRAY*** (0 to 63, 0 to 7) of BIT;

d. **FILE TYPES**

Objects of file types represent files in host environment.

□ SYNTAX:

TYPE file_type_name ***IS FILE OF*** type_name;

□ E.g.

TYPE VECTORS ***IS FILE OF*** BIT_VECTOR;

TYPE NAME ***IS FILE OF*** STRING;

Note: Data types are also classified as

□ Pre-defined data types

□ User defined data types

User –defined data types

Integer

Real

Enumerated type

Array

Record

Pre-defined data types.

Bit

Boolean

Integer

Real

Natural

Physical

Character

Signed

Unsigned

Data Types

Predefined data types.

- □ **bit** values: '0', '1'
- □ **boolean** values: TRUE, FALSE
- □ **integer** values: -(231) to +(231 - 1)

- □ **std_logic** values: 'U','X','1','0','Z','W','H','L','-'

U' = uninitialized
'X' = unknown
'W' = weak 'X‘
'Z' = floating
'H'/'L' = weak '1'/'0‘
'-' = don't care

- □ **Std_logic_vector** (n downto 0);
- □ **Std_logic_vector** (0 upto n);

OPERATORS

- Adding operator
- Multiplication operators
- Logical operators
- Relational or comparision operators
- Shift operators
- Miscellaneous operators

1. Adding operations:

- These are + - &

- Where & is the concatenation operator ,it can used for array type or element type.

- Eg.

  X=’1’ & “ 1011”

  Results in “11011”

2. Multiplication operators:

- These are / * mod rem

- Suppose mod A/B=Where mod operator gives remainder of A/B ,with sign of B value.

  Eg. 15 mod -7 =-1

- Suppose rem A/B= where rem operator gives Remainder of A/B with sign of A value.

  Eg. 15 rem -7 =1

3. Logical operators

- The seven logical operators are :

  AND, OR, NOT, NAND, NOR, XOR, XNOR.

  Eg. A And B;

- Note: A nand B nand C is illegal..This problem can be avoided by using parenthesis.

  i.e (A nand B) nand C

4. Relational or comparison operators

- These are:

  = /= < < = > >=

- Eg.

  mvl'('u')>mvl'('z');

  "VHDL" < "VHDL 92"

5. Shift operators

These are:

□ Sll – shift left logical

□ Srl – shift right logical

□ Sla –shift left arithmatic

□ Sra –shift right arithmetic

□ Rol –rotate left

□ Ror –rotate right

Example

For x=”1100”

- Sll 2 is “0000”   --vacated bits filled with ‘0’
- Srl 3 is “0001”   --vacated bits filled with ‘0’
- Sla 2 is “0000”   -- filled with the rightmost bit
- Sra 2 is “1111”   -- filled with the leftmost bit
- Rol 2 is “0011”   --rotate left
- Ror 3 is “1001”   --rotate right

- Sll -2 is “0011”   -- srl 2 operation performed
- Srl -3 is “0000”   -- sll 3 operation performed
- Sla -2 is “1111”   -- sra 2 operation performed
- Sra -2 is “0000”   -- sla 2 operation performed
- Rol -2 is “0011”   -- ror 2 operation performed
- Ror -3 is “0110”   -- rol 3 operation performed

6. Miscellaneous operators

These are: Abs **

- □ The abs (absolute) operator is defined for any numeric type.
- □ The **(exponentiation) operator is defined for left operand to be of integer or floating point type,and for right operand(i.e the exponent) to be of integer type only.
- □ Note: the **NOT** operator has same precedence as above two operators.

SUBTYPE

- □ A subtype is a type with a constraint.
- □ The constraint specifies the subset of values for the subtype. This type is called the base type of subtype.
- □ SYNTAX:

  SUBTYPE type_name ***IS*** data_type;

- □ Eg.

  SUBTYPE My_integer ***IS*** integer range 48 to 145;

Behavior model(**Sequential Statements**)

- □ wait statement
- □ assertion statement
- □ report statement
- □ signal assignment statement
- □ variable assignment statement
- □ procedure call statement
- □ if statement
- □ case statement
- □ loop statement
- □ next statement
- □ exit statement
- □ return statement
- □ null statement