VHDL Lexical Elements

Design File = Sequence of

- **Lexical Elements &&**
- **Separators**

(a) Separators: *Any # of Separators Allowed Between Lexical Elements*

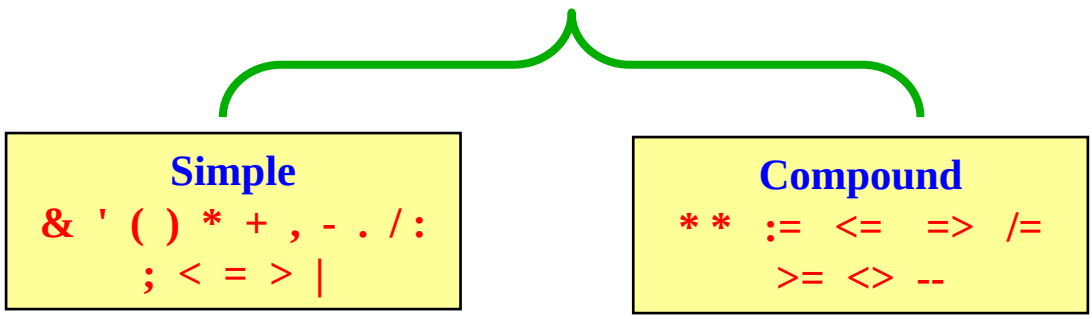
1. Space character
2. Tab
3. Line Feed / Carriage Return (EOL)

(b) Lexical Elements:

1. Delimiters `` *Meaningful Separator Characters*``
2. Identifiers
3. Literals
   - (i) Character Literal
   - (ii) String Literal
   - (iii) Bit String Literal
   - (iv) Abstract (Numeric) Literal

VHDL Lexical Elements

(i) Delimiters = *Separators Which Have Meaning*



(ii) Identifiers:

1. Key/Reserved Words (*No Declaration Required*)
2. User-Defined

VHDL Lexical Elements

Reserved Words

abs	disconnect	label	package	
access	downto	library	Poll	units
after		linkage	procedure	until
alias	else	loop	process	use
all	elsif		variable	
and	end	map	range	
architecture	entity	mod	record	wait
array	exit	nand	register	when
assert		new	rem	while
attribute	file	next	report	with
begin	for	nor	return	xor
block	function	not	select	
body	generate	null	severity	
buffer	generic	of	signal	
bus	guarded	on	subtype	
case	if	open	then	
component	in	or	to	
configuration	inout	others	transport	
constant	is	out	type	

VHDL Lexical Elements

User-Defined Identifiers

Identifier ::= Letter{[underscore]Letter_or_Digit}

- ***Starts*** with a Letter
- Followed by any # of Alpha-Numeric Characters
- ***No*** 2-Consecutive Underscores are Allowed
- Underscore ***Cannot*** be the Last Character in an Identifier

Comments

- Start with - - ``*2 Consecutive Dashes*``
- Comment Must be the LAST Lexical Element on the Line
- IF Line starts with - - , It is a Full-Line Comment.

Examples:

- - This is a Full-Line Comment

C := A*B; - - This is an In-Line Comment

VHDL Lexical Elements

Literals

(i) Character Literal

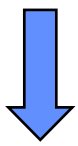
- ***Single*** Character Enclosed in ***Single Quotes***
- Used to Define *Constant* Values of *Objects of Type Character*

Examples:

`A` `B` `e` `` `1` `9` `*` ……etc.

(ii) String Literal

- ***Sequence*** of Characters Enclosed in ***Double Quotes***
- IF a Quotation Char is Required, 2 Consecutive Quotation Marks Are Used



  - *No **2** Strings are Allowed on the same Line*

Examples:

``A String`` -- *8-Char String*

`````` -- *Empty String*

`` `` `` `` -- *4-Double Quotes -- String of Length 1*

``A+B=C;#3=$`` -- *String with Special Chars*
``````

VHDL Lexical Elements (Literals)

(ii) String Literal (Contd)

- ***Strings Must Be Typed on One Line***
- Longer Strings Are *Concatenated* from Shorter Ones Using the **&** operator.

Examples:

``This is a Very Long String Literal`` **&**

``Formed By Concatenation``

(iii) Bit String Literals

- *Is a String Literal Where*
  - All Chars are Only Digits or Underscore.
  - Is Preceded By A Base Identifier ∈ {B, O, X}
  - The Length of the String Does Not Include the Number of UnderScores

Examples:

- **B**``11011001`` -- Length 8
- **B**``1101_1001`` -- Length 8
- **X**``D9`` -- Length 8 (Equiv to Above String)
- **O**``331`` -- Length 9

VHDL Lexical Elements

(iii) Bit String Literals (Contd)

- *Used to Specify Initial Contents of Registers*
- *Value of Bit-String is Equivalent to a String of Bits, However, Interpreting This Value is a User Choice*

Examples:

- X``A`` -- Represents the String 1010
  -- Interpreted as +10 For Unsigned Representation
  -- Interpreted as -6 For Signed 2`s Complement Representation

(iv) Abstract (Numeric) Literals

1. *Default is Decimal*
2. *Other Bases Are Possible (Bases Between 2 and 16)*
3. *Underscore Char May Be Used to Enhance Readability*
4. *Scientific Notations Must Have Integer Exponent*
5. *Integer Literals Should Not Have Base Point*
6. *Integer Literals Should Not Have -ive Exponents*
7. *In Real Literals, a Base Point Must Be Followed By AT LEAST ONE DIGIT*
8. *No Spaces Are Allowed*

VHDL Lexical Elements

(iv) Abstract (Numeric) Literals (Contd)

Examples:

```
0     1      123_987_456     73E13    -- Integers
0.0   2.5    2.7_456   73E-2   12.5E3   -- Reals
```

Special Case (BASED LITERALS)

- ***General Base*** *Abstract Literals (Including Decimal)*

Based_Literal::=Base#Based_Integer[.Based_Integer]#[Exponent]

Based_Integer::=Extd_Digit{[Underscode] Extd_Digit}

Extd_Digit::=digit | Letters_A-F

- *Both* **Base** *and* **Exponent** *are Expressed in Decimal*
- **Base** *Must be Between 2 & 16*
- *Digits Are Extended to Use the HEX Characters A-F*

Examples:

The Following Represent Integer Value of 196

2#1100_0100# , 16#C4#

4#301#E1 , 10#196#

The Following Represent Real Value of 4095.0

2#1.1111_1111_111#E11 , 16#F.FF#E2

10#4095.0#