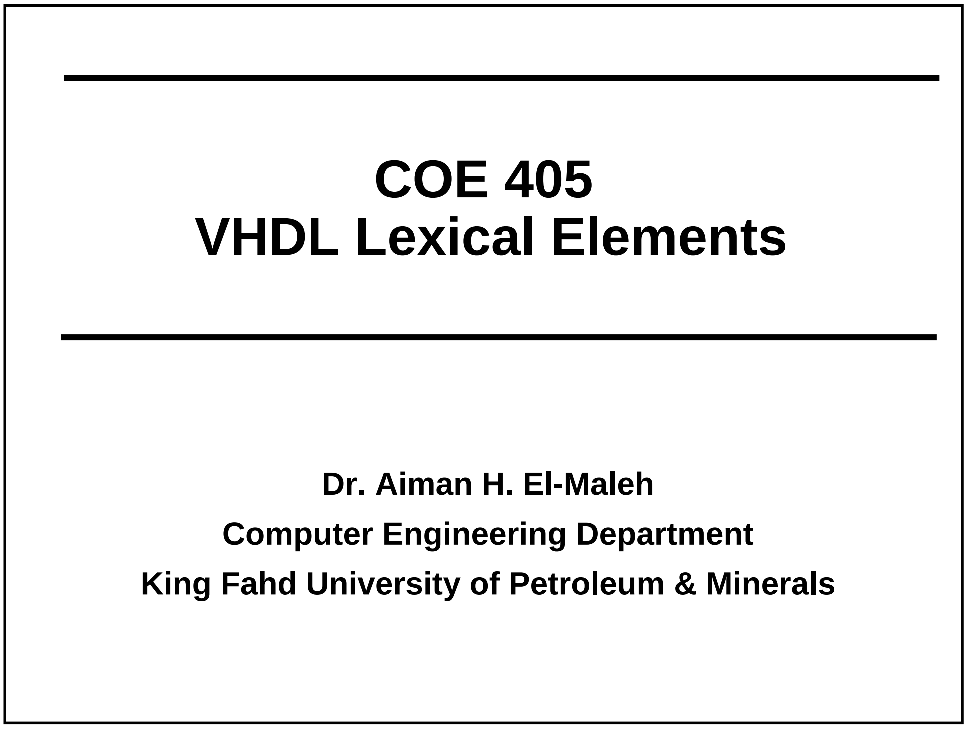
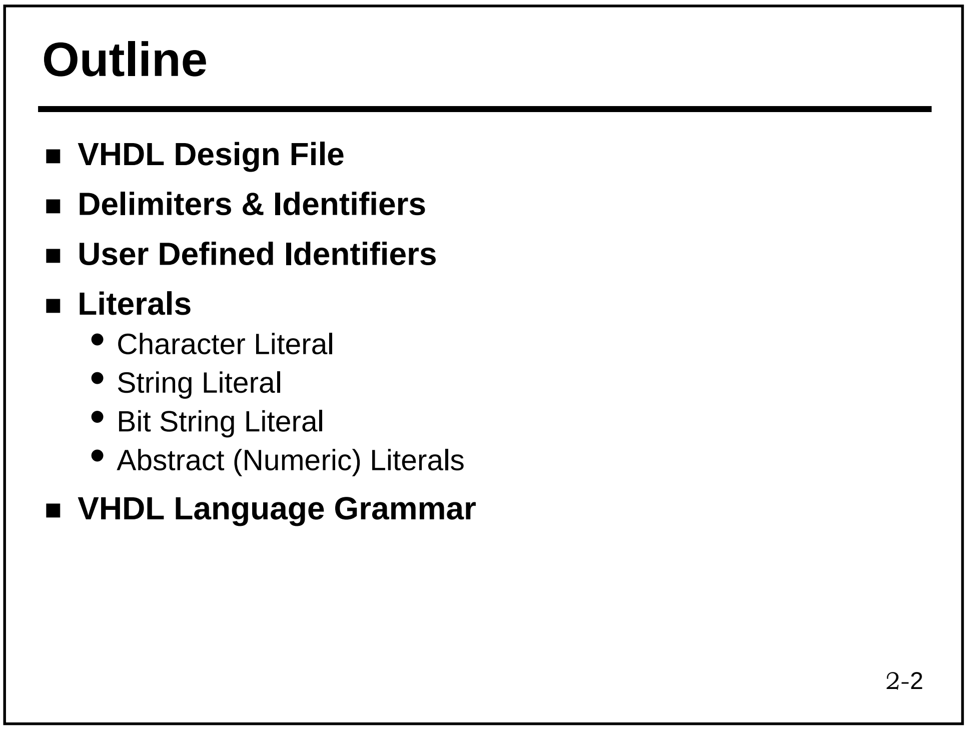


COE 405
VHDL Lexical Elements

Dr. Aiman H. El-Maleh
Computer Engineering Department
King Fahd University of Petroleum & Minerals



Outline

- **VHDL Design File**
- **Delimiters & Identifiers**
- **User Defined Identifiers**
- **Literals**
  - Character Literal
  - String Literal
  - Bit String Literal
  - Abstract (Numeric) Literals
- **VHDL Language Grammar**

2-2

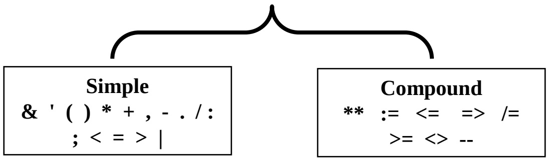
Design Files

- **Design file is a sequence of**
  - Lexical Elements
  - Separators
- **Separators**
  - Any # of separators allowed between lexical elements
    - Space character
    - Tab
    - Line Feed / Carriage Return (EOL)
- **Lexical Elements:**
  - Delimiters ``meaningful separator characters``
  - Identifiers
  - Literals
    - Character literal
    - String literal
    - Bit string literal
    - Abstract (numeric) literal

Delimiters & Identifiers

- **Delimiters are separators which have meaning**

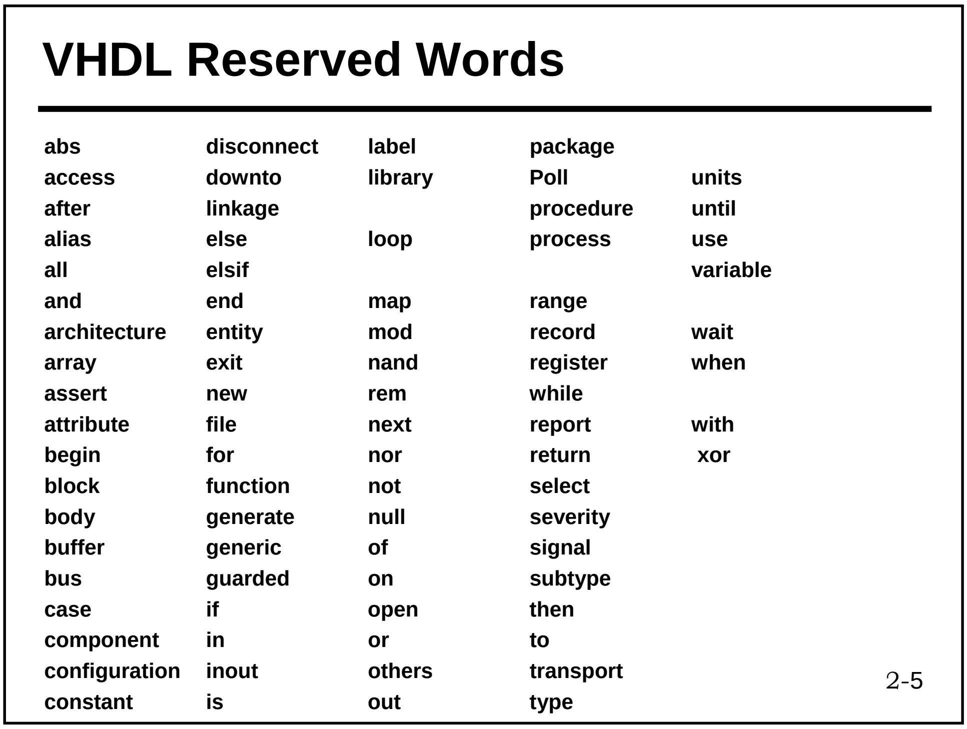
Delimiters



Simple
& ' ( ) * + , - . / :
; < = > |

Compound
** := <= => /=
>= <> --

- **Identifiers**
  - Key/Reserved words (No Declaration Required)
  - User-defined



VHDL Reserved Words

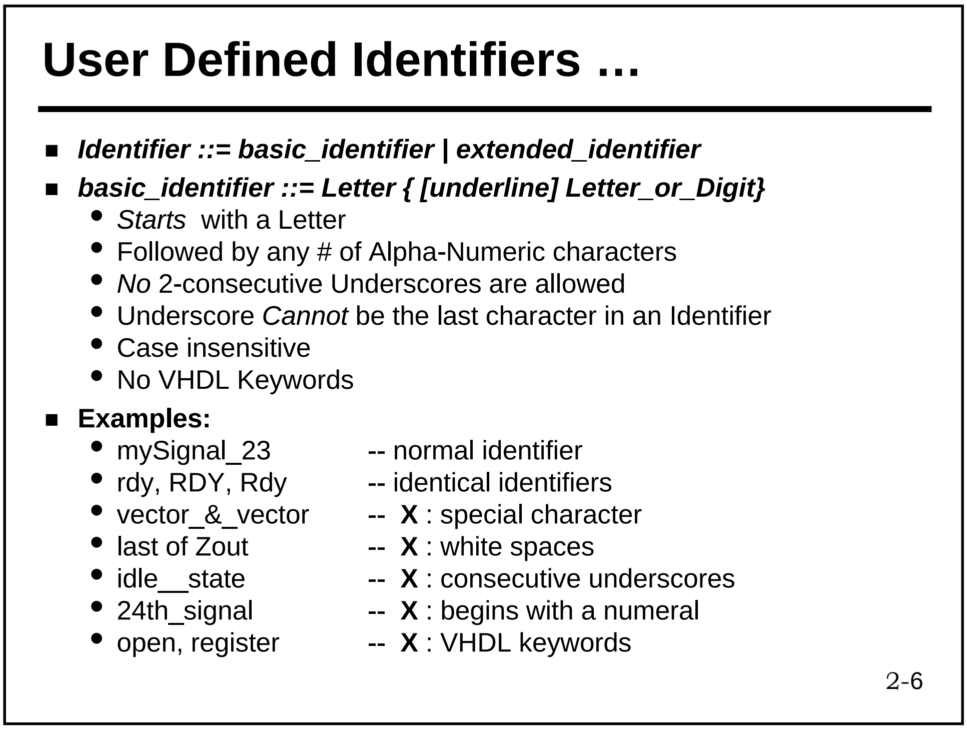
abs
access
after
alias
all
and
architecture
array
assert
attribute
begin
block
body
buffer
bus
case
component
configuration
constant

disconnect
downto
linkage
else
elsif
end
entity
exit
new
file
for
function
generate
generic
guarded
if
in
inout
is

label
library
loop
map
mod
nand
rem
next
nor
not
null
of
on
open
or
others
out

package
Poll
procedure
process
range
record
register
while
report
return
select
severity
signal
subtype
then
to
transport
type

units
until
use
variable
wait
when
with
xor



User Defined Identifiers …

- ***Identifier ::= basic_identifier | extended_identifier***
- ***basic_identifier ::= Letter { [underline] 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
  - Case insensitive
  - No VHDL Keywords
- **Examples:**
  - mySignal_23 -- normal identifier
  - rdy, RDY, Rdy -- identical identifiers
  - vector_&_vector -- **X** : special character
  - last of Zout -- **X** : white spaces
  - idle__state -- **X** : consecutive underscores
  - 24th_signal -- **X** : begins with a numeral
  - open, register -- **X** : VHDL keywords

… User Defined Identifiers

- ***extended_identifier ::= \ graphic_character {graphic_character} \***
  - Defined in VHDL’93
  - Enclosed in back slashes
  - Case sensitive
  - Graphical characters allowed
  - May contain spaces and consecutive underscores
  - VHDL keywords allowed
- ***Examples:***
  - [illegible] [illegible]
  - [illegible] [illegible]
  - [illegible] [illegible]
  - [illegible] [illegible]
  - [illegible] [illegible]
  - [illegible] [illegible]
  - [illegible] [illegible]

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

Literals …

- **Character Literal**
  - Single character enclosed in single quotes
  - Used to define constant values of objects of type Character
  - Literal values are Case Sensitive; ‘z’ NOT SAME as ‘Z’
  - Examples:
    - `A` `B` `e` ` ` `1` `9` `*` ………….etc.
- **String Literal**
  - Sequence of characters enclosed in double quotes
  - If a quotation char is required, 2 consecutive quotation marks are used
  - ***Strings must be typed on one line***
  - Longer strings are *Concatenated* from shorter ones using the **&** operator.

… Literals …

- **String Literal**
  - Examples:
    - ``A String`` -- 8-Char String
    - ```` -- Empty String
    - `` `` `` `` -- 4-Double Quotes -- String of Length 1
    - ``A+B=C;#3=$`` -- String with Special Chars
    - ``This is a Very Long String Literal`` **&**
      ``Formed By Concatenation``
- **Bit String Literal: is a String Literal**
  - Preceded by a ***Base Identifier*** ∈ **{B, O, X}** {**B** for Binary, **O** for Octal and **X** for Hex}
  - All Chars are only ***Digits*** {in the Base Number System} or ***Underscores***.
  - The length of the string does not include the number of Underscores.
  - *Used to specify initial contents of registers*
  - *Value of bit-string is equivalent to a string of Bits, however, <u>Interpreting</u> this value is a <u>User Choice</u>*

… Literals …

- **Bit String Literal**
  - Examples:
    - **B**``11011001``   -- Length 8
    - **B**``1101_1001``   -- Length 8
    - **X**``D9``   -- Length 8 (Equivalent to above string)
    - **O**``331``   -- Length 9
    - **X**``A``   -- Represents the String 1010

      -- Interpreted as +10 for unsigned representation
      -- Interpreted as -6 for signed 2`s complement representation

2-11

… Literals …

```
Entity Test is
end;
Architecture T1 of Test is
signal x, y, z, w : Bit_Vector(11 downto 0);
Begin
x <= "101011110011";
y <= B"1010_1111_0011";
z <= X"AF3";
w <= O"5363";
End T1;
```

When B symbol is used before a string, as many underscores as needed can be used inside the string.

2-12

… Literals …

- **Abstract (Numeric) Literals**
  - Default is decimal
  - Other bases are possible (Bases between 2 and 16)
  - Underscore char may be used to enhance readability
  - Scientific notations must have integer Exponent
  - Integer literals should not have base point
  - Integer literals should not have -ive Exponents
  - In Real literals, a base point must be followed by AT LEAST ONE DIGIT
  - No spaces are allowed
  - **Examples**:
    - 0   1   123_987_456   73E13   -- Integer
    - 0.0 2.5   2.7_456   73.0E-2   12.5E3   -- Real

2-13

… Literals

- **Special Case (Based Literals)**
  - General Base Abstract Literals (Including Decimal)
  - *Based_Literal::=Base#Based_Integer[Based_Integer]#[Exponent]*
  - *Based_Integer::=Extd_Digit { [Underline] 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#

2-14

VHDL Language Grammar

- **Formal grammar of the IEEE Standard 1076-1993 VHDL language in BNF format**
  - Appendix E
  - http://www.iis.ee.ethz.ch/~zimmi/download/vhdl93_syntax.html