

---

# **Технологии проектирования компьютерных систем**

**Лекция 3. Алфавит языка VHDL и его  
лексические элементы**

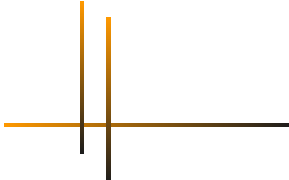
---



# Алфавит языка



Алфавит языка VHDL представляет собой набор символов, разрешенных к использованию и воспринимаемых компилятором. Алфавит языка составляют:

- символы из набора ISO 8859-1:1987 (International Organization for Standardization);
  - составные символы, воспринимаемые компилятором как один СИМВОЛ.
- 

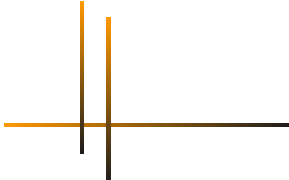
# Составные символы

Символ	Описание
$\leq$	Меньше или равно, присвоение
$\geq$	Больше или равно
$\Rightarrow$	Следует
$:=$	Присвоение
$\neq$	Не равно
$**$	Возведение в степень
$\langle \rangle$	Границы

# Лексические элементы



Текст на языке VHDL - это последовательность отдельных лексических элементов (лексем). Лексема - минимальное объединение символов, несущее смысл. Различают следующие виды лексем:

- разделитель и ограничитель;
  - идентификатор;
  - ключевое (зарезервированное) слово;
  - абстрактный литерал;
  - символьный литерал;
  - строковый литерал;
  - битовые строки;
  - комментарий.
- 

# Разделители и ограничители

Разделители и ограничители служат для разъединения (установки границ) лексических элементов (слов).

Разделителями служат символы: пробел, табуляция и конец строки. Количество разделителей не имеет значения.

Ограничители - это специальные одиночные символы (в основном наборе символов):

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

или составные (парные) символы.

# Идентификаторы

Идентификаторы - это простые пользовательские имена, которые присваиваются некоторому объекту.

Определение (в форме Бэкуса-Наура).

$$\text{identifier} ::= \text{letter} \{ [ \_ ] \text{letter} \mid \text{digit} \}$$

В программе идентификаторы могут конструироваться из строчных и прописных букв, цифр от 0 до 9 и символа подчеркивания '\_' (и только из них!). Кроме того, написание идентификаторов должно подчиняться следующим правилам:

- не может быть зарезервированным словом языка;
- должен начинаться с буквы (не с цифры);
- не может заканчиваться символом подчеркивания '\_';
- не может содержать двух последовательных символов подчеркивания '\_ \_';
- не может содержать внутри себя пробелы и специальные символы '-', '@', '%'.

# Идентификаторы

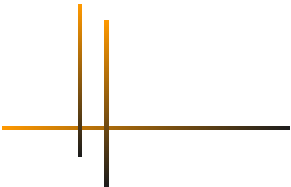
В VHDL-коде нет различия между прописными и строчными буквами. Так `ident1`, `IDENT1` и `Ident1` - это все одно и то же имя. Примеры идентификаторов приведены в таблице.

Правильные идентификаторы	Неправильные идентификаторы
<code>carry_out</code>	<code>7AB</code> (начинается с цифры)
<code>Dim_Sum</code>	<code>A@B</code> (специальный символ <code>@</code> )
<code>Count7SUB_2goX</code>	<code>SUM_</code> (заканчивается подчеркиванием)
<code>AaBBb</code>	<code>PI__A</code> (два подчеркивания подряд)
<code>ExampleOut</code>	<code>Example Out</code> (пробел не допустим)

# Ключевые (зарезервированные) слова



В VHDL-87 зарезервировано 81 ключевое слово, и VHDL-93 дополнительно введены 16 зарезервированных слов.





# Абстрактные литералы

Литералы представляют собой константы, непосредственно включаемые в текст программы в отличие от прочих данных — констант и переменных, обращение к которым осуществляется посредством ссылок. Литералы не могут быть изменены в тексте программы.

Имеются два класса абстрактных литералов:

- десятичные;
- целые.

Десятичным литералом является абстрактный литерал, содержащий точку.

Целым литералом является абстрактный литерал без точки.

```
abstract_literal ::= decimal_literal | based_literal
```

# Десятичные литералы

Десятичные литералы - абстрактные литералы, выраженные в десятичной системе счисления. Они могут быть целыми, реальными или целыми и реальными с экспонентой.

**decimal\_literal ::= integer [ .integer ] [ exponent ]**

**integer ::= digit { [ underline ] digit }**

**exponent ::= E [ + ] integer | E – integer**

Знак экспоненты E может быть строчным либо прописным.

Подчеркивание в десятичном литерале не является значащим.

Экспонента для целого литерала не должна иметь знак минус.

Средства синтеза ПЛИС допускают применение только целых литералов.

# Примеры описания десятичных литералов

## Целые числа

12    0    1E6    123\_456

## Реальные числа

12.0    0.0    0.456    3.14159\_26

## Реальные числа с экспонентой

1.34E-12    1.0E+6    6.023E+24

# Литералы с указанием основания системы счисления

Литерал с указанием основания системы счисления - абстрактный литерал, выраженный в форме, в которой явно указано основание системы счисления. Основание может быть от двух до шестнадцати.

**based\_literal ::=**

**base #based\_integer [ .based\_integer ] #[ exponent ]**

**base ::=integer**

**based\_integer ::=**

**extended\_digit { [ underline ] extended\_digit }**

**extended\_digit ::=digit |letter**

# Литералы с указанием основания системы счисления

Символ подчеркивания, вставленный между смежными цифрами литерала, не изменяет его значения. Основание и показатель должны быть записаны в десятичной системе счисления. В литерале могут использоваться буквы от А до F для указания цифр от десяти до пятнадцати. Знак экспоненты E может быть строчным либо прописным.

## *Примеры:*

Целочисленные литералы со значением 255:

2#1111\_1111#

16#FF#

016#0FF#

Целочисленные константы со значением 224:

16#E#E1

2#1110\_0000#

Вещественные константы со значением 4095.0:

16#F.FF#E+2

2#1.1111\_1111\_111#E11

# Символьные литералы

Символьные литералы формируются с помощью одного из 191 графических символов (включая пробел) между двумя символами апострофа. Символьный литерал имеет значение, которое принадлежит символьному типу.

**character\_literal ::= 'graphic\_character '**

*Примеры:*

'A' '\*' ''' ''

# Строковые литералы

Строковый литерал формируется как последовательность букв (возможно пустая), заключенных в двойные кавычки, которые применяют как строковые скобки.

**string\_literal ::= " {graphic\_character} "**

Значением строкового литерала является последовательность символов, соответствующих графическим символам константы строки, кроме кавычек.

Для включения кавычки в строку необходимо ввести две двойные кавычки.

Строковый литерал должен располагаться в одной строке. Для формирования "длинных" строковых литералов может быть употреблена операция конкатенации &.

Длина строкового литерала - количество символов в представленной последовательности.

# Строковые литералы

Примеры строковых литералов:

**"Установка времени слишком коротка "** --сообщение об ошибке.

**" "** -- пустой строковый литерал.

**" " "А" ""** -- три строковых литерала единичной длины.



# Битовые строки

Для задания значений битовым векторам можно применять не только строковые литералы ("111000"), но и более удобное представление в виде битовых строк в 2-ой(B), 8-ой(O) и 16-ой(X) формах с использованием символа '\_'.

Формат описания битовых строк.

**bit\_string\_literal ::= base\_specifier "[ bit\_value] "**

**base\_specifier ::= B | O | X**

**bit\_value ::= extended\_digit { [ underline] extended\_digit  
}**

**extended\_digit ::= digit | letter**

# Битовые строки

Вместо прописных букв В, О, Х допускается применять строчные буквы b, o, x.

Битовые строки формируются как последовательность цифр 0, ..., 9, А, ..., F (или а, ..., f) между двумя кавычками. Подчеркивание в таком литерале не является значащим.

Длина битовой строки - число бит в последовательности, представляющей литерал. Так, в частности, все литералы X"F\_FF", O"7777", B"1111\_1111\_1111" имеют длину 12 бит.

*Пример:*

**B"1111\_1111\_1111"**      --Эквивалент литералу строке  
**"111111111111"**.

**X"FFF"**      -- Эквивалент

**B"1111\_1111\_1111"**.

**O"777"**      -- Эквивалент **B"111\_111\_111"**.

**X"777"**      -- Эквивалент

**B"0111\_0111\_0111"**

# Комментарии

Комментарий начинается с двух смежных дефисов и продолжается до конца строки. Он может появляться в любой строке VHDL описания. Компилятор игнорирует текст, начиная с символов "--" до конца строки, т.е. комментарий может включать в себя символы, не входящие в алфавит языка (в частности, русские и украинские буквы).

*Примеры:*

- Последнее предложение отображает сообщения.**
- end ;-- Обработка строки закончена.**
- Длинный комментарий может быть разбит на**
- Две или больше последовательных строки.**
- Первые два дефиса запускают комментарий.**