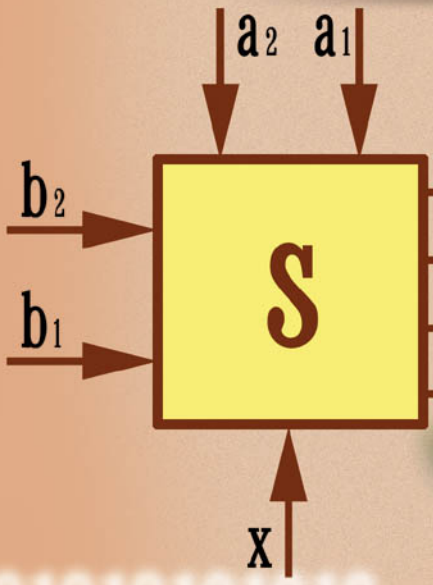
П. Н. БИБИЛО

«СОЛОН-Р»

ОСНОВЫ VHDL ЯЗЫКА



Altera
Cadence
Graphics
Mentor
Synopsys
XILINX
и др.

$S=f(a_2a_1,b_2b_1,x)$

Основы языка VHDL

Эффективная работа в современных САПР, разработанных ведущими зарубежными фирмами в области микроэлектроники (XILINX, Altera, Cadence, Synopsys, Mentor Graphics и др.), требует знания языка VHDL – исходного языка описания проектов.

Книга доктора технических наук, профессора Бибило П.Н. предназначена для первоначального ознакомления с языком VHDL, являющимся международным стандартом в области автоматизации проектирования микроэлектронных систем на современной элементной базе сверхбольших интегральных схем (СБИС) – как на заказных СБИС, так и на программируемых логических интегральных схемах (ПЛИС).

Рецензенты:

доктор технических наук А.А. Петровский,

доктор технических наук А.А. Прихожий,

кандидат технических наук Д.И. Черемисинов

Приглашаем к сотрудничеству **Авторов** по тематике САПР:

E-mail: Solon.Pub@relcom.ru

тел.: (095) 254-44-10
(095) 252-36-96

ISBN 5-93455-056-X

© “СОЛОН-Р”, 2010
© П.Н.Бибило, 2010

Оглавление

Глава 1

Основные элементы языка VHDL

1.1. Структурное и поведенческое описание цифровой системы

Цифровая система может быть описана на уровне поведения - функций, реализуемых системой, и на структурном уровне.

Структурное описание - это описание системы в виде совокупности компонент (подсхем, элементов) и связей между компонентами.

Поведенческое описание - это описание системы при помощи некоторых процедур на уровне зависимостей выходов от входов. Иначе говоря, поведенческое описание задает алгоритм, реализуемый системой.

Цифровая система может быть сложной, как, например, СБИС, (микропроцессор) или весьма простой, как логический элемент (вентиль).

Естественно, некоторые компоненты системы в структурном описании могут состоять из нескольких частей - компонент более низкого уровня иерархии описания. Представляя отношение вхождения подсхем в схемы в виде графа, можно получить дерево (граф) иерархии описания всей системы.

Например, пусть цифровая система **S** (рис. 1.1) реализует следующий алгоритм. На входные полюсы системы **S** подаются два двухразрядных числа **a**=(a2,a1), **b**=(b2,b1), где a2, b2 - старшие разряды чисел **a**, **b** соответственно и x - управляющий сигнал [3].

Если x=1, система **S** должна перемножить числа **a**, **b** и выдать четырехразрядное число **d**=(d4,d3,d2,d1), где **d= a×b**.

Если x=0, то числа **a, b** должны быть сложены, при этом в разряде d4 всегда должен быть нуль, в разряде d3 - перенос c2, в разряде d2 - старший разряд суммы s2, в разряде d1 - младший разряд суммы s1.

Предполагается, что

(a2,a1) + (b2,b1)= (c2,s2,s1).

Мы описали алгоритм, который должна реализовать система **S**, на русском языке. Однако для автоматизированного (компьютерного) проектирования требуется формальная спецификация.

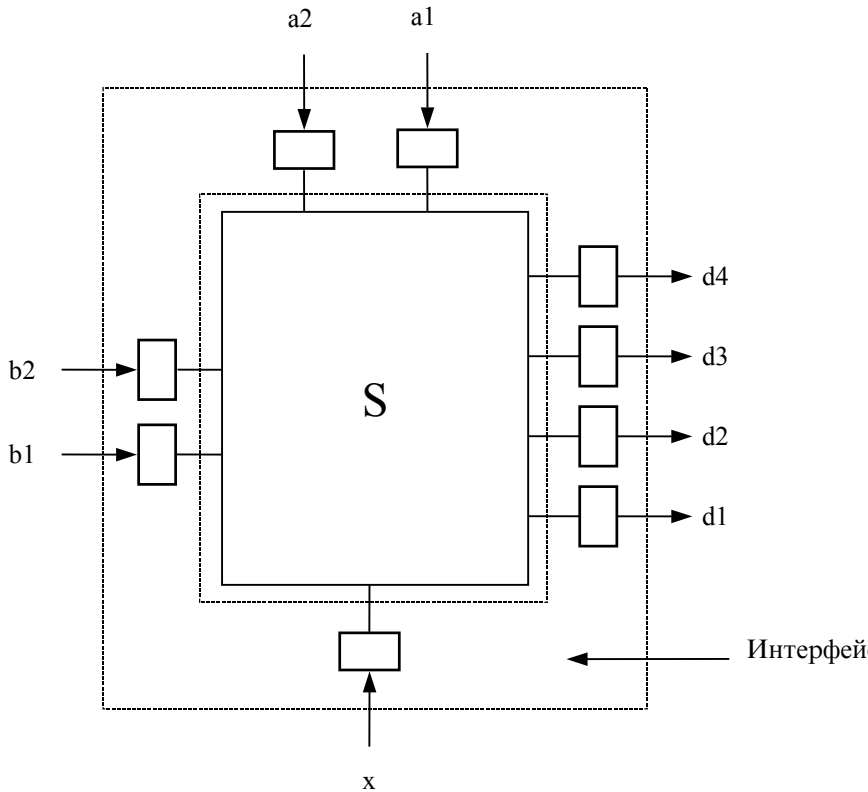


Рис.1.1. Система **S** и ее интерфейс

Формальные спецификации должны быть записаны на соответствующем формальном языке. Позже будет показано, что алгоритм, реализуемый системой **S**, может быть очень коротко описан на языке VHDL. Если же рассматривать структурный уровень описания системы **S**, то можно легко увидеть, что в систему **S** входит *двухразрядный сумматор* и *двухразрядный умножитель*.

Двухразрядный сумматор - это устройство для сложения двухразрядных чисел, *двухразрядный умножитель* - устройство для перемножения двух чисел, каждое из которых имеет только два

разряда. В систему **S** должно входить также простейшее устройство управления и схема дизъюнктивного объединения выходных сигналов.

Структура системы **S** изображена на рис. 1.2, где **adder_2** - двухразрядный сумматор, **mult_2** - двухразрядный умножитель, **yy** - устройство управления, **dd** - схема дизъюнктивного формирования

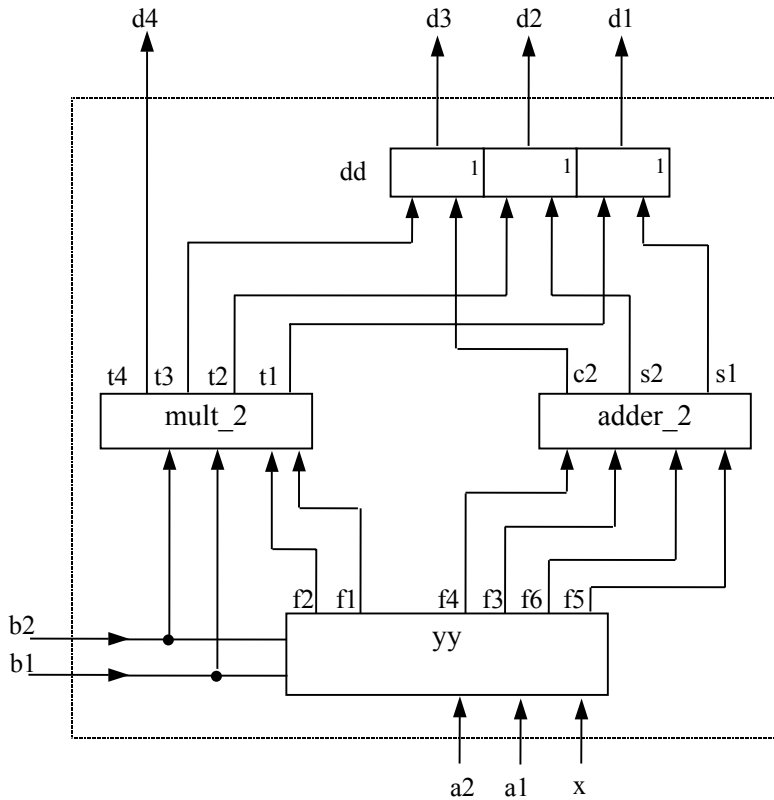


Рис. 1.2. Структура цифровой системы **S**

выходных сигналов. Схему **dd** образуют три дизъюнктора. Устройство управления **yy** является весьма простым и функционирует следующим образом:

если x=0, то

(a2,a1)=(f4,f3),
(b2,b1)=(f6,f5),

(f2,f1)=(0,0),

т.е. числа **a, b** подаются на вход сумматора **adder_2**.

Если же x=1, то

(f4,f3)=(0,0),
(f6,f5)=(0,0),
(a2,a1)=(f2,f1),

т.е. числа **a, b** подаются на вход умножителя **mult_2**.

Пусть описание системы **S** имеет имя VLSI_1. Тогда иерархия структурного описания системы **S** будет иметь вид (рис.1.3)

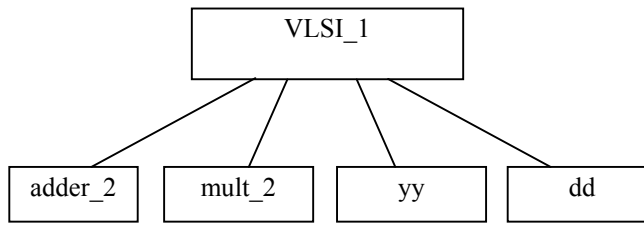


Рис. 1.3. Дерево иерархии структурного описания системы **S**

Рассмотрим двухразрядный умножитель **mult_2.** На вход данной схемы (рис. 1.4) поступают сигналы r1, r0, s1, s0. Сигналы r1, r0 интерпретируются как двухразрядное целое число **r=**(r1,r0), сигналы s1, s0 **-** как двухразрядное целое число **s=**(s1,s0).

Выходные сигналы в t3, t2, t1, t0 представляют собой разряды числа **t=**(t3, t2, t1, t0) - произведения чисел **s**, **r**:

$$(t3, t2, t1, t0) = (r1,r0)\times(s1,s0).$$

В схему входят элементы двух типов - **add1** и **and2**.

Элемент **and2** представляет собой логический элемент И - двухвходовый конъюнктор.

Заметим, что на языке VHDL знак & употребляется не для обозначения логической операции "И" (конъюнкции), а для операции конкатенации векторов.

Оператором логической конъюнкции служит оператор **and**, поэтому описание функции элемента **and2** на языке VHDL выглядит следующим образом:

```
y <= x1 and x2;
```

где x1, x2 - имена входных сигналов, y - имя выходного сигнала, **<=** - оператор назначения сигнала (будет рассмотрен позже).

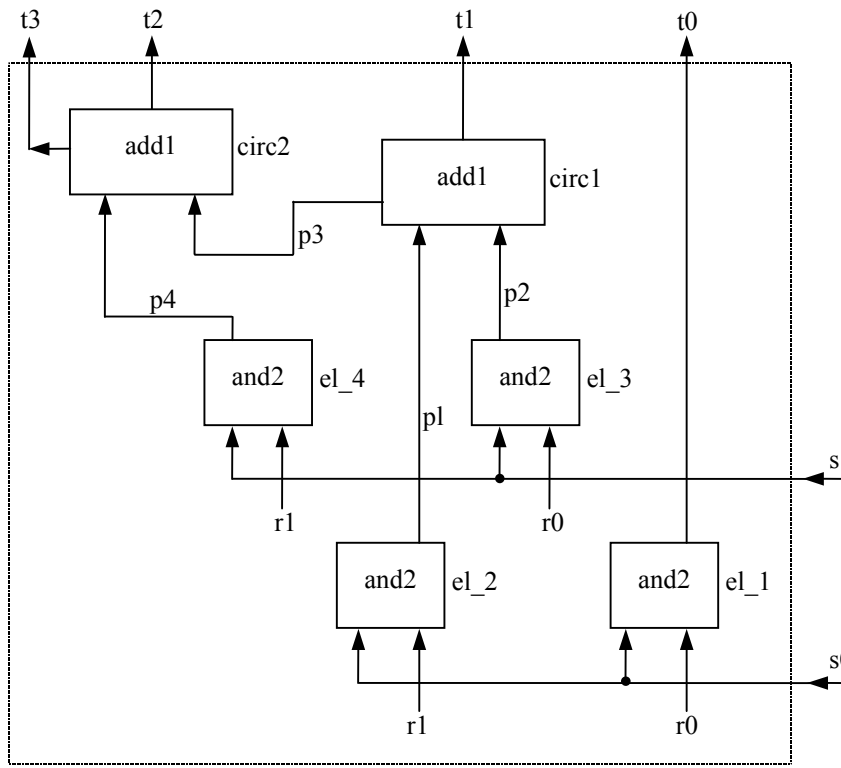


Рис. 1.4. Двухразрядный умножитель mult_2

Элемент **add1** представляет собой одноразрядный полусумматор, функционирование которого описывается таблицей истинности (табл. 1.1).

Таблица 1.1

b1	b2	c	s1
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

В данном случае b1, b2 интерпретируются как одноразрядные числа, s1 - сумма, c - перенос в следующий разряд.

В привычной математической записи булевы функции c, s1 могут быть представлены следующим образом:

$$s1 = b1 \oplus b2 = (\ \overline{b1} \wedge b2\ ) \vee (\ b1 \wedge \overline{b2}\ ),$$
$$c = b1 \wedge b2.$$

В языке VHDL функционирование элемента **add1** записывается следующим образом:

```
s1<=((b1 and (not b2)) or (not b1) and b2));
c<=b1 and b2;
```

где **or** - оператор логической дизъюнкции;

and - оператор логической конъюнкции;

not - оператор отрицания.

Итак, в дерево проекта схемы умножителя, которую назовем **mult_2**, входят элемент and2 и подсхема add1. Если использовать логические элементы **or2** (двухвходовый дизъюнктор) и **inv** (инвертор) и **and2** (двухвходовый конъюнктор) для реализации подсхемы **add1,** то дерево проекта будет трехуровневым.

Элементы **and2**, **or2**, **inv** являются *листьями* проекта. Листья проекта не имеют составных частей и называются *примитивами* проекта. Примитив описывается только на поведенческом уровне.

Объектами проекта для двухразрядного умножителя являются **mult_2**, **add1**, **and2.**

Обозначение корня дерева (**mult_2**) является именем проекта.

Каждый объект проекта имеет два различных типа описаний:

- описание объекта "в целом" (**entity**);
- описание архитектуры объекта (**architecture**).

Упрощенно можно сказать, что описание объекта "в целом" состоит из имени объекта и описания портов (входов, выходов) объекта. Описание объекта "в целом" в языке VHDL носит название "интерфейс" объекта. Чтобы отличать один объект проекта от другого, термин entity будет пониматься иногда и как объект проекта.

Для сигналов, подаваемых, снимаемых с портов указывается вид (режим, направление) сигнала: входной (**in**), выходной (**out**) и его тип.

Описание объекта проекта **and2** имеет вид.

```
entity and2 is                  -- декларация имени объекта проекта
port (x1,x2: in BIT;            -- декларация входных портов
          y: out BIT);          -- декларация выходного порта
end and2;

architecture functional of and2 is -- декларация архитектуры
begin
y <= x1 and x2;                 -- описание функции объекта
end functional;
```

В тексте данной программы имеются комментарии. Комментарий начинается двумя смежными дефисами и продолжается до конца строки.

В данном примере BIT - это тип сигнала. Архитектурное тело может определять поведение объекта проекта непосредственно (быть примитивом), либо представлять собой структурную декомпозицию на более простые компоненты. Описание объекта проекта **add1** выглядит следующим образом:

```
entity add1 is
port (b1,b2 : in BIT;
      c1,s1 : out BIT);
end add1;

architecture struct_1 of add1 is
begin
s1<= ((b1 and (not b2)) or ((not b1) and b2));
c1<= b1 and b2;
end struct_1;
```

Описание объекта проекта **mult_2** выглядит следующим образом:

```
entity mult_2 is
port (s1,s0,r1,r0 : in BIT;
       t3,t2,t1,t0 : out BIT);
end mult_2;
architecture structure of mult_2 is
component
```

```
add1
port (b1,b2: in BIT;
    c1,s1: out BIT);
end component;

signal p1,p2,p3,p4 : BIT;
begin
t0 <= r0 and s0;          -- элемент el_1
p2 <= r0 and s1;          -- элемент el_3
p1 <= r1 and s0;          -- элемент el_2
p4 <= r1 and s1;          -- элемент el_4
circ1: add1
port map (p1, p2, p3,t1);
circ2: add1
port map (p3,p4,t3,t2);
end structure;
```

В описании архитектуры объявляются (декларируются) две подсхемы (компоненты). После ключевого слова **begin** приводятся экземпляры описаний. Каждый экземпляр имеет уникальную метку (circ1, circ2 – метки). Каждый экземпляр имеет карту портов (**port map**). Карта портов отражает связь между входами, выходами описаний компонента и экземплярами компонента. Заметим, что в данном описании мы использовали понятие компонента (подсхемы) для add1, в то время как логические элементы И схемы мы описали на функциональном уровне, не используя понятие компонента. Возможность проведения таких смешанных описаний является важной полезной особенностью языка VHDL. Данная гибкость весьма удобна при проектировании на начальных этапах, когда важно получить точное алгоритмическое описание, не вдаваясь в детали структурной организации некоторых частей проекта.

Аналогично можно рассмотреть двухразрядный сумматор **adder_2** (рис. 1.5), состоящий из двух подсхем **add1, add2**, где **add1 -** это уже известный нам одноразрядный полусумматор, а **add2 -** одноразрядный сумматор, функционирование которого описывается следующей таблицей истинности (табл. 1.2):

Таблица 1.2

c1	a1	a2	c2	s2
0	0	0	0	0
0	0	1	0	1

0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

В дерево проекта для подсхемы **adder_2** входят подсхемы add1, add2, а VHDL-описание имеет вид

```
entity adder_2 is
port (a1, b1, a2,b2 : in BIT;
      c2,s2,s1 : out BIT);
end adder_2;


architecture structure of adder_2 is
component
add1
port (b1,b2: in BIT;
     c1,s1: out BIT);
end component;
component  add2
port(c1, a1,a2:in BIT;
     c2,s2:out BIT);
end component;
signal c1: BIT;
begin
circ1: add1
port map (b1,b2, c1,s1);
circ2: add2
port map (c1,a1,a2,c2,s2);
end structure;
```

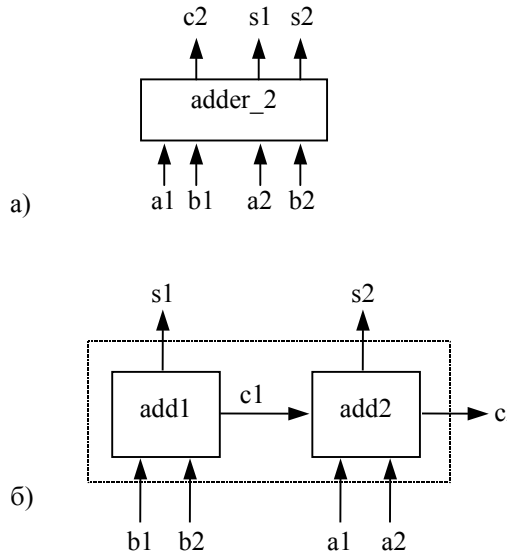


Рис. 1.5. Двухразрядный сумматор
(a1,b1)+(a2,b2)=(c2,s2,s1):
а - условное обозначение; б - схема в виде каскадного
соединения одноразрядного полусумматора add1 и
одноразрядного сумматора add2

Можно заметить, что в различные подсхемы входит **add1**, при этом подсхема **add1**, входящая в сумматор **adder_2**, является листом проекта и поэтому описана на поведенческом уровне. Подсхема **add1**, входящая в умножитель **mult_2**, описана на структурном уровне. Предлагаем читателю самостоятельно описать подсхему **add1** в виде объекта проекта, используя примитивы **and2**, **or2**, **inv.**

Итак, VHDL-код для структурного описания системы **S** (см. рис. 1.2) выглядит следующим образом:

```
entity vlsi_1 is
port (a2, a1, b2,b1,x:in BIT;
      d4,d3,d2,d1: out BIT);
end vlsi_1;
architecture structure of vlsi_1 is
component                          -- декларация компонента
adder_2
port (a1,b1,a2,b2: in BIT;
```

```
      c2,s2,s1: out BIT);
end component;
component mult_2
port(s1,s0, r1,r0: in BIT;
     t3,t2,t1,t0: out BIT);
end component;
component dd
port (x1,x2,x3,x4,x5,x6 : in BIT;
      y1,y2,y3 : out BIT);
end component;
component yy
port( a2,a1,b2,b1,x : in BIT;
     f6,f5,f4,f3,f2,f1 : out bit);
end component;

signal f1,f2,f3,f4,f5,f6,t4,t3,t2,t1,c2,s2,s1: BIT;
                    -- декларация внутренних сигналов
begin
circ1: yy
port map ( a2,a1, b2,b1, x, f6,f5,f4,f3,f2,f1);
circ2: mult_2
port map ( f2,f1, b2,b1, d4,t3,t2,t1);
circ3: adder_2
port map ( f4,f3, f6,f5,c2,s2,s1);
circ4: dd
port map ( s1,t1,s2,t2,c2,t3, d1,d2,d3);
end structure;
```

Дерево проекта для системы **S** изображено на рис. 1.6.

Приведем один из вариантов алгоритмического описания системы **S** в целом. Следует обратить внимание на то, что входные и выходные сигналы интерпретируются как целые числа, что позволяет сделать алгоритмическое описание весьма компактным.

```
entity vlsi_1 is
port (a,b : in integer range 0 to 3;
     x : in BIT;
     D : out integer  range 0 to 15);
end vlsi_1;

architecture functional of vlsi_1 is
```

```
signal e: integer range 0 to 15;
begin
p0: process(a,b,x)
begin
  if (x='0') then
e <= a + b;
  elsif ( x = '1') then
e <= a * b ;
  end if;
end process;
D <= e;
end functional;
```

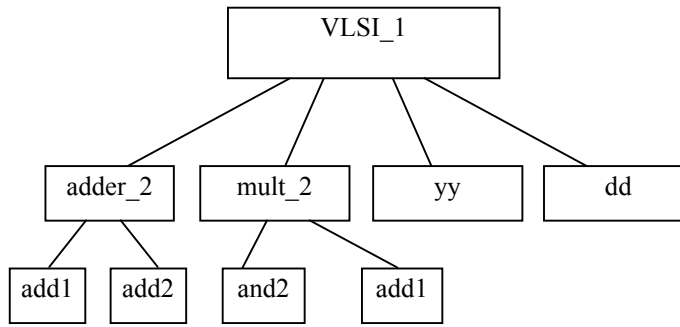


Рис.1.6. Дерево проекта цифровой системы **S**

Как видно, оно значительно компактнее структурного описания. Очевидно, что переход от алгоритмического описания к структурному представляет значительный практический интерес. К сожалению, автоматический переход возможен только для подмножества языка VHDL. Такое подмножество языка называется *синтезируемым*.

Получение функционально-структурной схемы по ее алгоритмическому описанию называется *высокоуровневым синтезом* в отличие от *логического* синтеза, когда по функционально-структурному описанию цифровой системы надо получить логическую схему из заданных базисных логических элементов. Программу, осуществляющую по VHDL-описанию синтез схемы, например схемы FPGA, часто называют *компилятором*. Однако в системах моделирования VHDL-кодов под компиляцией также понимается преобразование VHDL-кода в промежуточный язык, с которым оперируют непосредственно программы моделирования. Как и для языков программирования, сборку откомпилированных модулей осуществляет программа LINK. Программа, осуществляющая проверку синтаксической корректности, называется VHDL-*анализатором.*

Приведенное VHDL-описание системы **S** станет понятным позже, когда будут введены типы данных и основные операторы языка VHDL - операторы процессов, назначения сигналов и др.

1.2. Лексические элементы и типы данных

Лексические элементы, разделители, операторы.

Текст на языке VHDL есть последовательность раздельных лексических элементов, таких как

- идентификатор;
- разделитель;
- ключевое (зарезервированное) слово;
- абстрактный литерал;
- символьный литерал;
- строка литералов;
- битовая строка литералов;
- комментарий.

Смежные лексические элементы разделяются

- разделителями;
- концами строк;
- знаками форматирования.

Оператор есть один из следующих специальных символов:

& ( ) * + ? - . / : ; < = > |

Составной оператор есть композиция двух смежных специаль-
ных символов

=> ** := /= >= <= < >

Пример. VHDL-предложение

A <= B and C;

имеет шесть лексических элементов "A", "<=", "B", "and", "C", ";" .

Два из шести лексических элементов являются операторами:

" <= " (составной оператор назначения сигнала);

" ; "

В качестве разделителей в данном примере используются пробелы, однако нет необходимости иметь разделитель между оператором " ; " и лексическим элементом "C".

Комментарий начинается с двух **смежных** дефисов и продолжается до конца строки. Комментарии не учитываются при моделировании VHDL-описания.

Идентификаторы

Определение.

identifier ::= letter { [underline] letter_or_digit }

Внимание! Здесь и далее формальная запись синтаксических конструкций языка VHDL основывается на формах Бэкуса-Наура, употребление которых разъясняется в Приложении 1. Далее важная информация, на которую следует обратить особое внимание, будет сопровождаться только восклицательным знаком.

Идентификаторы употребляются как пользовательские имена и ключевые слова.

Идентификатор должен начинаться с буквы (не цифры). Может употребляться кроме букв и цифр, знак подчеркивания. Два подряд
идущих подчеркивания не допускаются.

! В VHDL-коде нет различия между прописными и строчными буквами.

AbC7 эквивалентно aBC7,
A_3 не эквивалентно A3.

Идентификатор не должен оканчиваться подчеркиванием.

Пример.

Правильные Идентификаторы	Неправильные Идентификаторы
carry_OUT	7AB (начинается с цифры)
Dim_Sum	A@B (специальный символ @)
Count7SUB_2goX	SUM_(кончается подчеркиванием)
AaBBb	PI_ _A (два подчеркивания подряд)

Зарезервированные (ключевые) слова

Как и многие другие языки программирования, VHDL имеет *ключевые* (*специальные)* слова.

Список ключевых слов

abs	access	after	Alias	all
and	architectur e	array	Assert	attribute
begin	block	body	Buffer	bus
case	component	configuratio n	Constant	disconnect
downto	else	elsif	End	entity
exit	file	for	Function	generate
generic	guarded	if	In	inout
is	label	library	Linkage	loop
map	mod	nand	New	next
nor	not	null	Of	on
open	or	others	Out	package
port	procedure	process	range	record
register	rem	report	return	select
severity	signal	subtype	then	to
transport	type	units	until	use
variable	wait	when	while	with
xor				

В стандарт VHDL'93 добавлены следующие слова:

group	impure	inertial	literal	postponed
pure	reject	rol	ror	shared
sla	sll	sra	srl	unaffected

xnor

Литералы

Классификация литералов языка VHDL приведена на рис. 1.7.

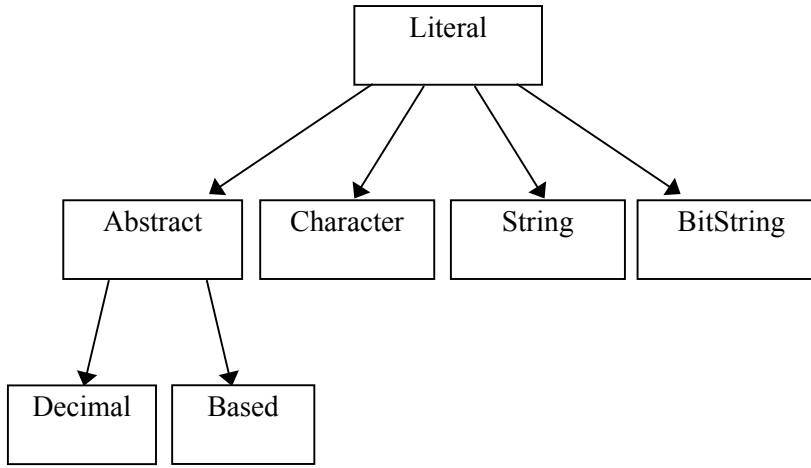


Рис. 1.7. Классификация литералов

Десятичный литерал может быть целым, вещественным или вещественным с экспонентой.

Определение.

decimal_literal ::= integer [**.** integer] [exponent]
integer ::= digit { [underline] digit }
exponent ::= E [+] integer | E - integer

Примеры.

Целые литералы: 21, 0, 1E2, 3e4, 123_000.

Вещественные литералы: 11.0, 0.0, 0.468, 3.141_592_6.

Вещественные литералы с экспонентой: 1.23E-11, 1.0E+4, 3.024E+23.

Знак экспоненты E может быть строчным либо прописным. Подчеркивание в десятичном литерале не является значащим. Экспонента для целого литерала не должна иметь знак минус.

Базовый литерал указывает на систему счисления: - от двоичной