621.396.69(07)
М

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Таганрогский государственный радиотехнический университет

КОНСПЕКТ ЛЕКЦИЙ
ПО КУРСУ

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ В МИКРОЭЛЕКТРОНИКЕ

(часть 2)

Для студентов специальности 2205

ФЭП

Таганрог 2002

ЯЗЫК ОПИСАНИЯ АППАРАТУРЫ VHDL

В начале 80-х годов министерство обороны США финансировало разработку многоуровневого языка VHDL, стандартизовало его и обязало поставщиков цифровых микросхем представлять в составе документации их описание на VHDL.

В связи с возлагаемой на VHDL особой ролью, интерес к нему в США и Европе огромен. Созданы американская и европейская группы, занимающиеся всем комплексом вопросов, связанных с внедрением VHDL, один из которых: создание мощных систем моделирования, использующих в качестве входного языка VHDL.

В России также поддерживаются работы по развитию и внедрению VHDL: РосНИИИС, институт МИЭМ, Томский политехнический институт, НИИ "Квант", Ассоциация разработчиков заинтересованных в применении VHDL.

(HDL – Hardware Description Languages – язык описания аппаратуры).

Общие сведения.

Язык VHDL представляет собой набор формальных записей, которые могут быть использованы на всех этапах разработки цифровых схем. На этом языке возможно как поведенческое, так и структурное и потоковое описание цифровых схем.

Язык VHDL используется: для моделирования цифровых схем, проектирования ПЛИС, БМК, заказных ИС.

VHDL, это язык, близкий по синтаксису и семантике к современным языкам программирования типа Паскаль, С и др.

I. Алфавит языка

Это набор символов, разрешенных к использованию и воспринимаемых компилятором. В алфавит языка входят:

1. Латинские строчные и прописные буквы: A, B, C, … Z и a, b, c, … z.
2. Цифры от 0 до 9.
3. Символы подчеркивания « _ ».

Только из этих символов могут конструироваться идентификаторы. Они должны подчинятся следующим правилам:

- индентификатор не может быть зарезервированным словом языка;
- индентификатор не может заканчиваться символом подчеркивания;
- индентификатор не может содержать двух последовательных символов подчеркивания;
- индентификатор должен начинаться с буквы.

Корректные идентификаторы:

cont, clock2, full_add.

Некорректные идентификаторы:

1clock, _adder, add_ _sub, entity*(зарезервированное слово).*

4. Символ “пробел”, символ табуляции, символ новой строки.

Символы являются разделителями слов в конструкциях языка. Количество разделителей не имеет значения. Таким образом, следующие выражения будут для компилятора эквивалентны:

Count:=2+2; count := 2

Count :=2 + 2; $\overset{+}{2}$;

5. Специальные символы, участвующие в построении конструкций языка:
   + - * / = < > . , ( ) : ; # ‘ “ |
6. Составные символы, воспринимаемые компилятором как один символ:
   <= >= => := /=

Комментарии: Примером комментария является два символа тире “- -”. Компилятор игнорирует текст, начиная с символов “- -” и до конца строки. Таким образом комментарий может включать символы, не входящие в алфавит языка (например русские буквы).

Числа.

В стандарте языка определены числа как целого, так и вещественного типа. Средства синтеза ПЛИС допускают применение только целых чисел. Целое число в VHDL может быть представлено в одной из четырех систем счисления: двоичной, десятичной, восьмеричной, шестнадцатеричной.

Символы.

Запись символа представляет собой собственно символ, заключенный в кавычки:

‘A’ ‘*’

Строки.

Представляют собой набор символов, заключенных в двойные кавычки:

“A string”

II.Типы данных.

Каждый тип данных в VHDL имеет определенный набор принимаемых значений и набор допустимых операций.

Простые типы.

Следующие типы являются предопределёнными:

1. BOOLEAN(логический) – могут принимать значения FALSE(ложь) *(эквивалентно 0)* и TRUE(истина) *(эквивалентно 1)*;
2. INTEGER(целый) - 32-разрядные числа со знаком;
3. BIT(битовый) – содержит значение 0 или 1;
4. STD_LOGIC(битовый) – представляет один бит данных . Объекты данного типа могут принимать 9 состояний. Данный тип определён стандартом IEEE_1164 для замены типа BIT.
5. STD_ULOGIC(только для одного источника), то же, что и STD_LOGIC, только в нем не определена функция разрешения, используемая для определения значения сигнала, имеющего несколько источников (драйверов).
6. ENUMERATED(перечислимый)–используется для задания пользовательских типов.
7. SEVERITY_LEVEL - перечислимый тип, используется только в операторе ASSERT.
8. CHARACTER – символьный тип.

Сложные типы.

К сложным типам относятся массивы(ARRAY).

Следующие типы – массивы являются предопределёнными:

1. BIT_VECTOR – одномерный массив элементов типа BIT;
2. STD_LOGIC_VECTOR – одномерный массив элементов типа STD_LOGIC;
3. STD_ULOGIC - одномерный массив элементов типа STD_ULOGIC;
4. STRING – одномерный массив элементов типа CHARACTER.

Границы диапазонов должны быть указаны при объявлении объектов данных типов.

Entity – объект.

Описание простых типов.

Тип BOOLEAN. Такой тип имею т константы, переменные и сигналы.

Пример:

```
Process (a,b)
Variable Cond(имя):Boolean;
 BEGIN
Cond := a > b;
 If Cond then
Output <= ‘1’;
 Else
Output <= ‘0’;
END IF;
END Process.
```

Значения типа BOOLEAN могут участвовать в выражениях =, /=, <, <=, >, >= и в логических операциях: AND (и); OR (или); NAND (и-не); NOR (или-не); XOR (исключающее или); NOT (инвертор).

Тип INTEGER.

Этот тип применяется в арифметических выражениях. По умолчанию объекты этого типа имеют размерность 32 бита. Для меньшего размера используется ключевое слово RANGE:

SIGNALX: Integer Range -127 to 127 (эта конструкция определяет X как 8-битное число).

(32 бита: 2^{31}-1 = от – 2147483647 до +2147483647)

Кроме того, можно определить ограниченный целый тип, используя конструкцию:

Type имя_типа is Range диапазон_индексов;

Диапазон индексов определяется следующим образом:

m To n

n Downto m,

где m, n – целочисленные константы, m <= n.

Пример:

Type byte_int 0 to 255;

Type Signed_word_int is Range -32768 to 32768;

Type Bit_index is Range 31 Downto 0;

Запись числа:

Constant min: integer :=0;

Constant group: integer :=13_452; -- эквивалентно 13452;

Значения типа integer могут участвовать в выражениях =, /=, <, <=, >, >=.

Результат выражений имеет тип BOOLEAN.

Для операндов типа integer допустимы операторы +, –, ABS. Результат выражения имеет тип integer.

Тип BIT.

Участвует в выражениях =, /=, <, <=, >, >= (результат выражения имеет тип BOLEAN), и в логических операциях AND; OR; NAND; NOR; XOR; NOT.

Тип STD_LOGIC.

Объекты этого типа могут принимать 9 значений:

0, 1, Z, –, L, H, U, X, W.

Для синтеза логических схем используются первые четыре:

‘0’ – логический ноль;

‘1’ – логическая единица;

‘Z’ – третье состояние;

‘–’ – не подключен.

Участвуют в операторах отношения и логических операциях.

Тип SEVERITY_LEVEL.

Переменные этого типа принимают значения:
NOTE;
WARNING;
ERROR;
FAILURE.

Тип CHARACTER.

Значением объекта данного типа может быть любой символ из набора ASCII (128 первых символов).

III. Массивы.

Массив представляет собой упорядоченную структуру однотипных данных.

При синтезе ПЛИС используются, в основном, одномерные массивы ограниченного и неограниченного типов.

1. Объявление ограниченного типа «массив» имеет вид:
Type имя_типа is
Array (диапазон индексов)
OF тип_элемента;
Диапазон индексов определяется явным заданием границ диапазона:
m To n
n Downto m (m < n).
2. Объявление неограниченного типа «массив» имеет вид:
Type имя_типа is
Array(тип индекса)
OF тип_элемента;
Тип индекса определяется:
подтип Range <>
где подтип может быть:
Integer – диапазон $-(2^{31} - 1) \ldots 2^{31} - 1$;
Natural – $0 \ldots 2^{31} - 1$;
Positive – $1 \ldots 2^{31} - 1$.

Примеры:
1) Объявление ограниченного массива:
Type Word(*имя типа*) is array (31 Downto 0) of STD_LOGIC;
Type Register is Array (byte Range 0 To 132) of Integer;
2) Объявление неограниченного массива:
Type Logic(*имя типа*) is Array (integer Range <>) Of BOOLEAN;
В языке имеется несколько предопределенных типов «массив»:
Type String (*строки*) is array (positive Range <>) Of CHARACTER;
Type STD_LOGIC_VECTOR is Array (Natural Range <>) of STD_LOGIC;

IV. Операторы VHDL

Основы синтаксиса.

Операторы VHDL записываются с учетом правил:

а) каждый оператор – это последовательность слов, содержащих буквы английского алфавита, цифры и знаки пунктуации;

б) слова разделяются произвольным количеством пробелов, табуляций и переводов строки;

в) операторы разделяются символом «;».

Для указания системы счисления для констант могут быть применены спецификаторы:

B – двоичная система счисления, B «0011»;

O – восьмеричная система счисления, O «3760»;

H – шестнадцатеричная система счисления, H «F6A0».

Объекты.

Объекты являются контейнерами для хранения различных значений в рамках модели. Каждый объект характеризуется типом и классом.

Типы подразделяются на предопределенные и определенные пользователем. Тип показывает, какого рода данные может содержать объект.

Класс показывает, что можно сделать с данными, содержащимися в объекте.

В VHDL определены следующие классы объектов:

– Constant – константы. Значение константы определяется при ее объявлении и не может быть изменено. Имеет все типы.

– Variable – переменные. Значение переменной меняется везде, где встречается присваивание данной переменной. Имеет все типы.

– Signal – сигналы. Сигнал представляет значение, передаваемое по проводам. Сигналы имеют ограниченный набор типов. Обычно Bit, Bit_Vector, STD_LOGIC, STD_LOGIC_VECTOR, Integer.

Синтаксис объявления объектов:

Constant {name[, name]}: Type [(index_range)]:= initial_value;

Variable (name): Type (index_range) := initial_value;

Signal (name): Type (index_range);

Диапазон значений индексов задается в виде:

Int_value to int_value или

Int_value Downto Int_value.

Атрибуты (свойства).

Определяют характеристики объекта к которым они относятся. Подразделяются, в основном, на предопределенные атрибуты. Для обращения к атрибутам используют символ «’». Например: A1’left.

В VHDL определены следующие атрибуты:
‘ left – левая граница диапазона индексов массива;
‘ right – правая граница диапазона индексов массива;
‘ low – нижняя граница диапазона индексов массива;
‘ high – верхняя граница диапазона индексов массива;
‘ range - диапазон индексов массива;
‘ reverse_range – обратный диапазон индексов массива;
‘ length – ширина диапазона индексов массива.

Компоненты.

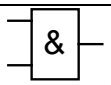
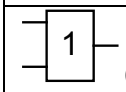
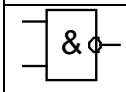
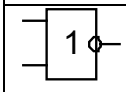
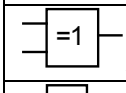
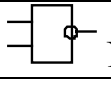
Объявление компонента определяет интерфейс и модели на VHDL (Entity или Architecture). В момент моделирования должны существовать объявления Entity и Architecture для компонентов, которые не только объявлены, но и установлены в схему. Это позволяет задавать объявления библиотечных элементов, а реальные их описания (объявления Entity и Architecture) задавать по мере использования этих элементов.

Объявление компонента записывается в виде:

```
Component name
[port (port_list);]
end component;
```

Выражения.

Выражения могут содержать следующие операторы:

Category	Operator	Пояснения
Boolean	 AND (И)	y1<=a1 and b1
	 OR (ИЛИ)	y2<=a2 or b2
	 NAND(И-НЕ)	y3<=a5 nand b5
	 NOR(ИЛИ-НЕ)	y4<=a6 nor b6
	 XOR(Искл.ИЛИ)	y3<=a3 xor b3
	 NOT(НЕ)	y5<=not a5
Comparison(сравнение)	=	Равенство
	/=	Неравенство
	<	Меньше
	<=	Меньше или равно
	>	Больше
	>=	Больше или равно
Arithmetic	ABS	Абсолютное значение
	+	Сложение

	–	Вычитание
	*	Умножение
	/	Деление
	MOD	Модуль
	REM	Остаток от деления
Concatenation	&	объединение

Порядок вычисления выражений определяется приоритетом операторов:

AND, OR, NAND, NOR, XOR – самый низкий приоритет.

=, /, <, <=, >, >=

+, –, & - средний приоритет.

ABS, NOT – Высший приоритет

Операторы с более высоким приоритетом выполняются раньше. Чтобы изменить такой порядок, используются скобки.

Операторы.

С помощью операторов описывается алгоритм, определяющий функционирование схемы. Операторы могут находиться в теле функции, процедуры, или процесса.

Wait…until ждать до тех пор пока (не)

Wait…for ждать в течение

WAIT UNTIL condition; - Приостанавливает выполнение процесса до момента выполнения условия;

Signal <= Expression; - оператор назначения значения сигнала. Устанавливает значение равным выражению справа;

Variable := Expression; - оператор присвоения значения переменной. Устанавливает значение переменной равным выражению справа;

Procedure_name (параметры); - оператор вызова процедуры. Состоит из имени процедуры и списка фактических параметров.

Оператор условия IF (если) используется для ветвления алгоритма по различным условиям.

Оператор выбора CASE задаёт ветвление алгоритма.

Оператор цикла LOOP позволяет многократно выполнять последовательность операторов. Диапазон значений задаётся в виде Value1 TO Value2 или Value1 DOWNTO Value2. Переменная цикла последовательно принимает значения из заданного диапазона. Количество итераций равно количеству значений в диапазоне.

Использование оператора LOOP. Пусть три микросхемы выполняют функцию «И»:

```
Y1<=A1 AND B1;
Y2<=A2 AND B2;
Y3<=A3 AND B3;
```

С помощью оператора цикла эту запись можно упростить:

```
FOR i  in 1 to 3 LOOP
Y(i)<=A(i) AND B(i);
END LOOP;
```

Оператор RETURN expression; возвращает значение из функции.

Оператор NULL – пустой оператор, не выполняет никаких действий.

Интерфейс объекта.

Полное VHDL–описание объекта состоит как минимум из двух описаний: описания интерфейса объекта и описания тела объекта (описание архитектуры).

Интерфейс описывается в объявлении объекта: ENTITY DECLARATION и определяет входы и выходы объекта, его входные и выходные порты PORTS и параметры настройки GENERIC. Параметры настройки отражают тот факт, что некоторые объекты могут иметь управляющие входы, с помощью которых может производиться настройка объектов, в частности, задаваться время задержки.

Например, у объекта Q1 три входных порта X1, X2, X3 и два выхода Y1, Y2.

Описание его интерфейса на VHDL имеет вид:

```
ENTITY Q1 is
        Port (X1, X2, X3: IN REAL; Y1, Y2: OUT REAL);
END Q1;
```

Порты объекта характеризуются направлением потока информации. Они могут быть:

- входными (IN);
- выходными (OUT);
- двунаправленными (INOUT);
- двунаправленными буферными (BUFFER);
- связными (LINKAGE).

А также имеют тип, характеризующий значения поступающих на них сигналов:

- целый (INTEGER);
- вещественный (REAL);
- битовый (BIT);
- символьный (CHARACTER).

Тело объекта описывает его структуру или поведение, и содержится в описании ARCHITECTURE.

Средства VHDL базируются на представлении о том, что описываемый объект ENTITY представляет собой структуру из компонент COMPONENT, соединяемых линиями связи. Каждая компонента, в свою очередь, является объектом и может состоять из компонент низшего уровня (иерархия объектов). Взаимодействуют объекты путём передачи сигналов SIGNAL по линиям связи.

Описание структуры объекта строится как описание связей компонент, каждая из которых имеет имя, тип и карты портов. Карта портов PORT MAP определяет соответствие портов компонент поступающим на них сигналам.

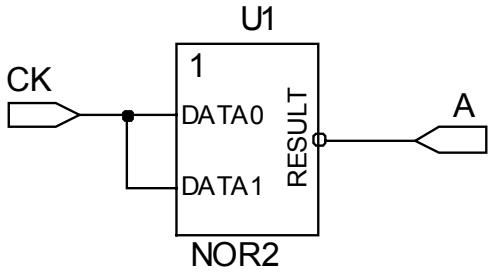
Можно интерпретировать карту портов как разъём, на который приходят сигналы и в который вставляется объект-компонента.

Принятая в VHDL форма описания связей компонент имеет вид:

Имя: тип связь (сигнал, порт);

Например:

U1: NOR2 PORT MAP (DATA0=>CK, DATA1=>CK, RESULT=>A);



ОПИСАНИЕ ОБЪЕКТА

Пусть имеется объект F. Он имеет два входа A1 и A2 и два выхода B1 и B2. Таблица истинности для объекта F:

Входы		Выходы	
A0	A1	B1	B2
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Сначала в VHDL-описании задаются библиотеки.

```
Library ieee;
Use ieee.std_logic_1164.all;
```

Затем объявляется объект F. Входы и выходы могут объявляться как одиночные порты или как шины.

1. Объявление входов и выходов как одиночных портов:

```
Entity F is
Port (A0, A1: in std_logic;
B1, B2: out std_logic);
End F;
```

2. Объявление входов в виде шины:

```
Entity F is
Port (A: in std_logic_vector (1 downto 0);
B1, B2: out std_logic);
End F;
```

ПОВЕДЕНЧЕСКОЕ ОПИСАНИЕ ОБЪЕКТА

При поведенческом описании объекта внутренняя архитектура объекта неизвестна. Известен только принцип функционирования, который может быть задан, например, в виде таблицы функционирования или таблицы истинности.

Описание объекта F, поведение которого задано в виде таблицы истинности:

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity F is
Port (a0, a1: in std_logic;
B1, b2: out std_logic);
End F;

Architecture Behavior of F is
Begin
Process (A0, A1)
Begin
Case (A0&A1) is
When “00”! “01”! “10” => B1<=’0’; B2<=’1’;
When “11” => B1<=’1’; B2<=’0’;
End case;
End process;
End behavior;
```

Другая форма записи поведенческого описания объекта F:

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity F is
Port (A: in std_logic_vector (1 downto 0);
B1, b2: out std_logic);
End F;

Architecture Behavior of F is
Begin
```

```
Process (A)
Begin
Case A is
When “00” => b1<=’0’; b2<=’1’;
When “01” => b1<=’0’; b2<=’1’;
When “10” => b1<=’0’; b2<=’1’;
When “11” => b1<=’1’; b2<=’0’;
End case;
End process;
End behavior;
```

Другая форма поведенческого описания объекта F:

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity F is
Port (A: in std_logic_vector (1 downto 0);
B1, b2: out std_logic);
End F;

Architecture Behavior of F is
Begin
Process (A)
Begin
If a = “00” then F<=’0’;
Elsif a =”01” then F<=’0’;
Elsif a =”10” then F<=’0’;
Elsif a =”11” then F<=’1’;
End if;
End process;
End behavior;
```

Другая форма записи поведенческого описания объекта F:

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity F is
Port (a0, a1: in std_logic;
B1, b2: out std_logic);
End F;

Architecture Behavior of F is
Begin
```

```
Process
Begin
Wait on (A0, A1)
If (A0=’1’ and (A1=’1’)
Then B1<=’1’; B2<=’0’;
Else B1<=’0’; B2<=’1’;
End if;
End process;
End behavior;
```

ПОТОКОВАЯ ФОРМА

В потоковой форме объект представляется в виде последовательности булевых операций И, ИЛИ, НЕ и др. Объект F может иметь различные варианты функциональных схем. Выберем вариант, когда вентили включены последовательно.

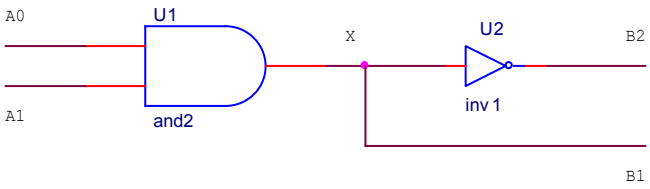


Рис.

Потоковая форма описания включает обозначение промежуточного узла X и введение строчки signal X: std_logic; Полная потоковая форма описания объекта F имеет вид:

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity F is
Port (a0, a1: in std_logic;
B1, b2: out std_logic);
Enf F;

Architecture F_A of F is
Signal X: std_logic;
Begin
X <= a0 and a1;
B2 <= not (X);
B1 <= X;
End;
```

Сигнал X введен потому, что порт B1 описании иинтерфейса объявлен выходным, то есть с него нельзя считывать сигнал и запись B2 <= not B1 была бы некорректной.

Введем задержки в распространение сигнала. Пусть задержка на выход B1 равна 10 нс и на выход B2 – 15 нс. Тогда:

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity F is
Port (a0, a1: in std_logic;
B1, b2: out std_logic);
Enf F;

Architecture F_B of  F is
Signal X: std_logic;
Begin
X <= a0 and a1 after 10ns;
B2 <= not (X) after 5ns;
B1 <= X;
End;
```

СТРУКТУРНОЕ ОПИСАНИЕ АРХИТЕКТУРЫ

Описание представляет собой архитектуру объекта как набор компонент, соединенных между собой и обменивающихся сигналами. Компоненты представляют собой библиотечные элементы. Их функции описаны на языке VHDL.

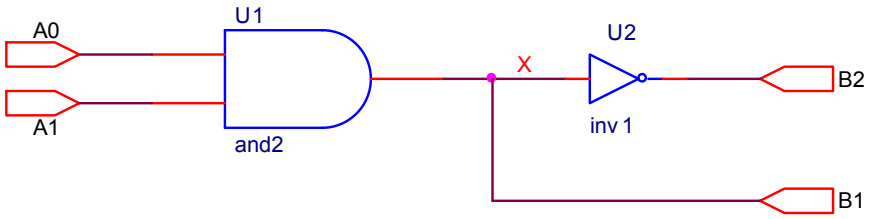


Рис.

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
ENTITY SCHEMATIC1 IS PORT (
A0 : IN std_logic;
A1 : IN std_logic;
B1 : OUT std_logic;
B2 : OUT std_logic
); END SCHEMATIC1;
```

```
ARCHITECTURE STRUCTURE OF SCHEMATIC1 IS

-- COMPONENTS

COMPONENT and2
PORT (
DATA0 : IN std_logic;
DATA1 : IN std_logic;
RESULT : OUT std_logic
); END COMPONENT;

COMPONENT inv1
PORT (
RESULT : OUT std_logic;
DATA : IN std_logic
); END COMPONENT;

-- SIGNALS

SIGNAL X : std_logic;

-- INSTANCE ATTRIBUTES

-- GATE INSTANCES

BEGIN
B1<=X;
U1 : and2     PORT MAP(
DATA0 => A0,
DATA1 => A1,
RESULT => X
);
U2 : inv1     PORT MAP(
RESULT => B2,
DATA => X
);
END STRUCTURE;
```

ЗАДЕРЖКИ СИГНАЛОВ

Объект с задержкой можно представить состоящим из двух: идеального элемента и элемента задержкки:

В языке VHDL встроены две модели задержки: инерциальная и транспортная.

Инерциальная модель предполагает, что элемент не реагирует на сигналы, длительность которых меньше порога, равного времени задержки элемента.

Транспортная модель не имеет такого ограничения.

Инерциальная модель по умолчанию встроена в оператор назначения сигнала языка VHDL. Например, оператор назначения

```
Y <= X1 and X2 after 10ns;
```

Описывает работу вентиля 2И т соответствует инерциальной модели.

Указание на использование транспортной модели обеспечивается ключевым словом Transport в правой части оператора назначения. Например:

```
YT <= transport X1 and X2 after 10ns;
```

Он отображает транспортную модель задержки вентиля.

Задержка может быть задана не константой, а выражением. Для этого ее надо задать как параметр настройки в описании интерфейса объекта. Например:

```
Entity 12 is
Generic (T: time=10ns);     - параметр настройки T равен 10 нс
Port (X1, X2: in std_logic;
Y: out std_logic);
End 12;

Architecture A1_inert of 12 is
Begin
Y <= X1 and X2 after T;
End A1_inert;
```

АТРИБУТЫ СИГНАЛОВ И КОНТРОЛЬ ЗАПРЕЩЕННЫХ СОСТОЯНИЙ

Описания систем может содержать информацию о запрещенных ситуациях, например, о недопустимых комбинациях сигналов на входах объектов, рекомендуемых частотах импульсов и т.д.

Средством отображения информации о запрещенных ситуациях в языке VHDL является оператор утверждения (оператор контроля, оператор аномалии) ***assert*** (утверждать). В нем, кроме контролируемого условия, которое не должно быть нарушено, то есть должно быть истинным, записывается сообщение Report о нарушении и уровень ошибки severity.

Например, дляя R-S-триггера запрещенной ситуацией является наличие двух нулей на входах: S=0, R=0. Эта запрещенная ситуация записывается предложением:

```
Assert not (S='0' and R='0')
```

Предопределенные атрибуты сигналов

	Тип результата	Примечание
S'TRANSACTION	BIT	S изменяется каждый раз, когда S активен
S'STABLE	BOOLEAN	TRUE (1) , если не было событий за интервал времени T
S'DELAYED	SIGNAL	Предыдущее значение S в момент NOW-T
S'ACTIVE	BOOLEAN	TRUE, если сигнал активен
S'EVENT	BOOLEAN	TRUE, если происходит событие S
S'LAST_EVENT	TIME	Время последнего события S

Например, в вентиле 2И возникает риск сбоя, когда фронт одного сигнала перекрывает срез другого. Эта запрещенная ситуация записывается следующим образом:

```
Architecture C1 of 12 is
Signal Z: std_logic :='0'          --запись исходного состояния сигнала
Begin
Process (X1, X2)
Z<=X1 and X2;
Assert not (Z='0' and not Z'STABLE and Z'DELAYED (10NS)='0');
Report "риск сбоя в вентиле 12";
Severity    warning;
Y<= transport Z after   10ns;
End C1;
```

ПАКЕТЫ

Описание пакета VHDL задается ключевым словом ***PACKAGE*** и используется, чтобы собирать часто используемые элементы конструкции для применения в других проектах. Пакет состоит из описания пакета и дополнительного тела пакета (Package body).

Описание пакета содержит:

1. Объявление типов;
2. Объявление констант;
3. Описания сигналов;

4. Объявления процедур и функций;
5. Объявление компонентов.

Если пакет содержит описания подпрограмм (функция или процедур) или определяет константы, величина которых не задана, то в дополнение к описанию необходимо тело пакета.

Можно выделить три типовых узла ИС:

1. Логические схемы или комбинационные;
2. Регистровые схемы или последовательностные (типовой представитель – триггер);
3. Цифровые автоматы (например, автомат Мура).

Рассмотрим примеры описания на языке VHDL комбинационных и регистровых схем.

МУЛЬТИПЛЕКСОРЫ
(КОМБИНАЦИОННАЯ СХЕМА)

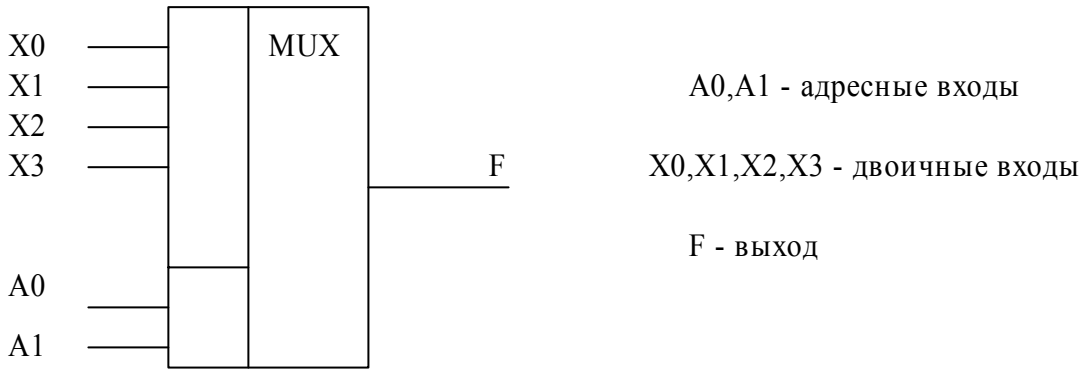


Таблица функционирования мультиплексора:

X0	X1	X2	X3	A0	A1	F
X0				0	0	X0
	X1			0	1	X1
		X2		1	0	X2
			X3	1	1	X3

```
Lidrary ieee;
Use ieee.std_logic_1164.all;

Entity mux is
Port (A: in std_logic_vector (1 downto 0);
```

```
X0, X1, X2, X3: in std_logic;
F: out std_logic);

Architecture behavior of mux is
Begin
Process (A, X0, X1, X2, X3)
Begin
If A=”00” then F<=X0;
Elsif  A=”01” then F<=X1;
Elsif  A=”11” then F<=X2;
Elsif  A=”11” then F<=X3;
End if;
End process;
End behavior;
```

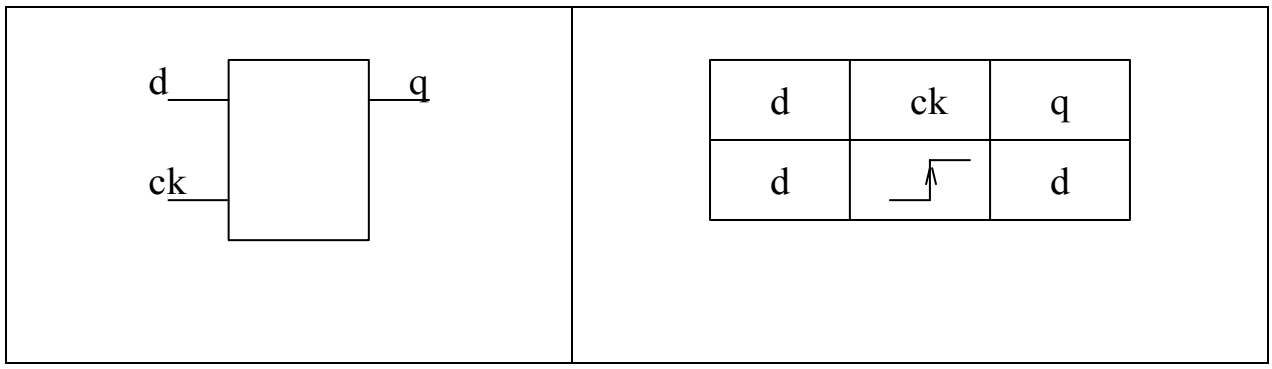
Другая форма записи:

```
Lidrary ieee;
Use ieee.std_logic_1164.all;

Entity mux is
Port (A: in std_logic_vector (1 downto 0);
X0, X1, X2, X3: in std_logic;
F: out std_logic);

Architecture behavior of mux is
Begin
Process (A, X0, X1, X2, X3)
Begin
Case A is
When “00” => F<=X0;
When “01” => F<=X1;
When “10” => F<=X2;
When “11” => F<=X3;
End case;
End process;
End behavior;
```

21
РЕГИСТРОВЫЕ СХЕМЫ

Триггер D-типа

Триггер работает по положительному фронту импульса. При этом происходит запись информации со входа D-триггера на его выход Q.



d	ck	q
d	↑	d

VHDL-файл имеет следующее описание:

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity dtype is
Port (d, ck: in std_logic;
        q: out std_logic);
end dtype;

architecture behavior of dtype is
begin
process (d, ck)
begin
if ck=’1’ and ck’event then
q<=d;
end if;
end process;
end behavior;
```

Другая форма записи триггера:

```
Library ieee;
Use ieee.std_logic_1164.all;
```

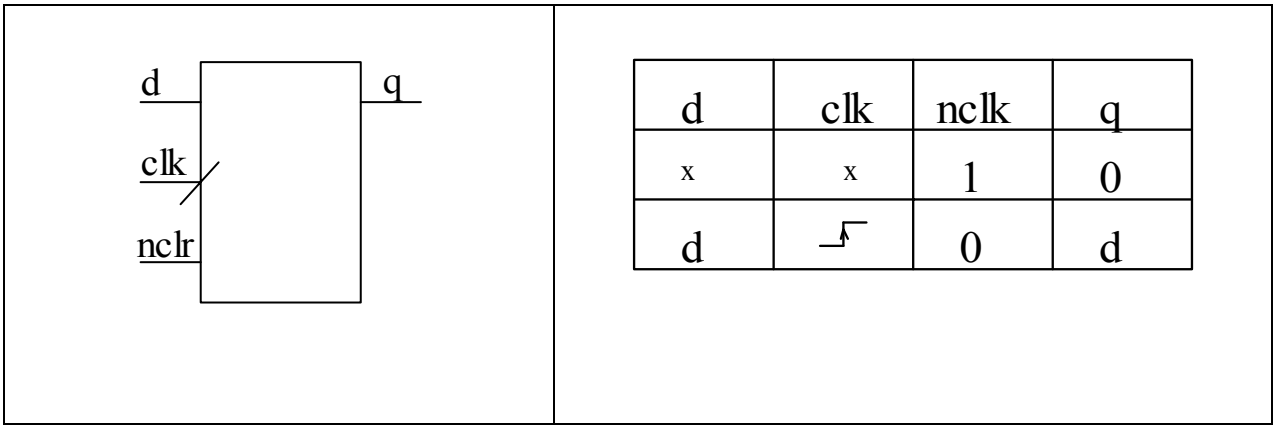
```
Entity dtype is
Port (d, ck: in std_logic;
        q: out std_logic);
end dtype;

architecture behavior of dtype is
begin
B1: Block
(ck’event and ck=’1’)
begin
q<=guarded d;
end block B1;
end behavior;
```

Другая форма записи триггера:

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity dtype is
Port (d, ck: in std_logic;
        q: out std_logic);
end dtype;

architecture behavior of dtype is
begin
process
begin
waint until ck’event and ck=’1’;
q<=d;
end process;
end behavior;
```

23

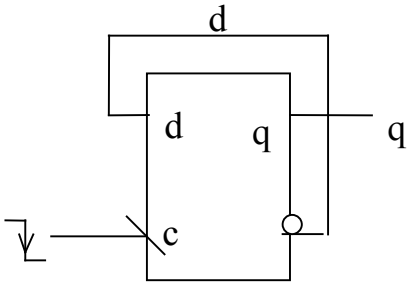
D-триггер с асинхронным сбросом



d	clk	nclk	q
x	x	1	0
d	↓	0	d

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity tr is
Port ( d, clk, nclr: in std_logic;
        q: out std_logic);
end tr;

architecture beh of tr is
Begin
Process (nclr, clk)
Begin
If nclr='0' then
q<='0';
elsif clk='1' and clk'event       --elsif clk='1' and not clk'stable  - можно и так
then q<=d;
End if;
End process;
End beh;
```

Т - триггер



Т-триггер – это счетный триггер. Он срабатывает по отрицательному фронту тактового импульса. Его описание приведено ниже:

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity T is
Port ( c: in std_logic;
        q: out std_logic);
end T;

architecture beh of T is
Signal d: std_logic :='0';
Begin
Process (c)
Begin
if c='0' and c'event then
d<= not d; q<=d;
End if;
End process;
End beh;
```