

Технологии проектирования компьютерных систем



Лекция 4. Типы данных



Определение

Тип - это множество значений с общим признаком.

VHDL - строго типизированный язык. Каждый объект объявляется со своим типом и может присваивать значение только данного типа. Благодаря этой особенности, программы VHDL имеют высокую надежность и обеспечивают экономию времени при отладке.

Подтип - подмножество значений данного типа.

Выделяют следующие типы данных языка VHDL:

- скалярные (`scalar_type`);
- составные (`composite_type`);
- указатели (`access_type`);
- файлы (`file_type`);
- защищенные (`protected_type`).

В лекции рассматриваются только те данные, которые поддерживаются средствами синтеза ПЛИС.

Скалярные типы

Скалярные типы создают значения, которые нельзя разбить на отдельные элементы или поля.

```
scalar_type_definition ::=  
enumeration_type_definition    -- перечисления;  
| integer_type_definition      -- целые значения;  
| floating_type_definition     -- действительные значения;  
| physical_type_definition     -- значения, имеющие размерность.
```

Все скалярные типы и их подтипы определяются через диапазон своих значений.

Перечисления

Перечисления состоят из списка значений, которые могут быть символами или идентификаторами.

```
enumeration_type_definition ::=  
(enumeration_literal { , enumeration_literal } )
```

```
enumeration_literal ::= identifier | character_literal
```

В САПР Quartus перечисления описывают по шаблону:

```
TYPE <name> IS (<enum_literal>, <enum_literal>, ...);
```

Например: **TYPE** LOGIC_VOLT **IS** ('0', '5', 'z', 'x');

```
TYPE MULTI_LEVEL_LOGIC is (LOW, HIGH, RISING, FALLING);
```

Вводимые имена перечислений не должны совпадать с
предопределенными в языке именами перечислимых типов.

Значения перечислений не должны совпадать.

Перечисления

Весь список значений пронумерован слева направо, начиная с нуля, то есть каждое значение имеет соответствующую позицию в списке:

TYPE LIGHT IS	(active,	off,	flashing);
Позиция:	0	1	2

Значения в разных перечислениях могут совпадать.

Предопределенные перечисления

Предопределенными перечислениями являются CHARACTER, BIT, BOOLEAN, SEVERITY_LEVEL, FILE_OPEN_KIND, и FILE_OPEN_STATUS, специфицированные в пакете "standard":

```
TYPE CHARACTER IS (NUL, SOH, ...);
```

```
TYPE BIT IS ('0', '1');
```

```
TYPE BOOLEAN IS (FALSE, TRUE);
```

```
TYPE SEVERITY_LEVEL IS (NOTE, WARNING, ERROR, FAILURE);
```

```
TYPE FILE_OPEN_KIND is (READ_MODE, WRITE_MODE, APPEND_MODE);
```

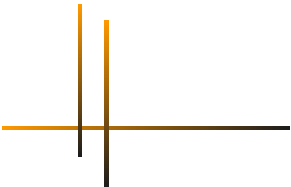
```
TYPE FILE_OPEN_STATUS is (OPEN_OK, STATUS_ERROR, NAME_ERROR, MODE_ERROR);
```

Примечание - Значения boolean (ложный и истинный) не идентичны логическим '0' и '1'.



Перечисления для цифровой техники

В пакете «std1164» predefinedены перечисления `std_ulogic` (неразрешимый логический тип с девятизначным алфавитом) и разрешимый подтип `std_logic`.



Перечисления `STD_ULOGIC` и `STD_LOGIC`

TYPE `STD_ULOGIC` IS (

'U',	-- неинициализированное значение;
'X',	-- неопределенное значение источника с малым выходным сопротивлением;
'0',	-- уровень «0» источника с малым выходным сопротивлением;
'1',	-- уровень «1» источника с малым выходным сопротивлением;
'Z',	-- высокоимпедансное состояние;
'W',	-- неопределенное значение источника с большим выходным сопротивлением;
'L',	-- уровень «0» источника с большим выходным сопротивлением;
'H',	-- уровень «1» источника с малым выходным сопротивлением;
'-'	-- произвольное);

Подтип `STD_LOGIC` определен как:

SUBTYPE `STD_LOGIC` IS **RESOLVED** `STD_ULOGIC`.

Целые числа

Тип целого задается через диапазон целых чисел.

integer_type_definition ::= range_constraint

Обычно этот диапазон находится между -2.147.483.648 и +2.147.483.647 (диапазон 32-разрядного целого).

В САПР QUARTUS целые числа описывают по шаблону:

TYPE <name> IS RANGE <low> TO <high>;

Пример описания целых чисел:

TYPE GROUP_INTEGER IS RANGE -1025 TO 1025;

Предопределенным целым типом является integer, который специфицирован в пакете "standard" как:

TYPE INTEGER IS RANGE -2147483648 TO 2147483647.

Целые числа

В пакете "standard» специфицированы и predefined подтипы natural и positive по шаблону:

```
SUBTYPE __subtype name IS __type_name RANGE __low_value TO  
__high_value;
```

```
SUBTYPE NATURAL IS INTEGER RANGE 0 TO INTEGER'HIGH;  
SUBTYPE POSITIVE IS INTEGER RANGE 1 TO INTEGER'HIGH.
```

Нельзя использовать имена predefined типов и подтипов для собственных определений.

Числа с плавающей запятой

Числа с плавающей запятой обеспечивают приближения к вещественным числам. Числа с плавающей запятой применяют для моделей, в которых погрешность вычисления значений не важна или не определена.

floating_type_definition ::= range_constraint

Примеры описания чисел с плавающей точкой:

TYPE RESULT IS RANGE 0.0 TO 11063.5;

SUBTYPE P_RESULT IS RESULT RANGE 2765.88 TO 8297.63.

Предопределенным типом с плавающей точкой является `real`, который специфицирован в пакете "standard" как:

TYPE REAL IS RANGE -1.0E38 TO 1.0E38.

Тип `real` обеспечивает 64-разрядное представление чисел с плавающей точкой (1 разряд - знак числа, 11 разрядов - порядок, 52 разряда - мантисса).

Физические типы

Физические типы создают числа с реальными размерностями, кратными некоторой базовой единице. Множество допустимых значений задается как диапазон целых чисел (базовых единиц).

Формат описания физических типов.

physical_type_definition ::=

range_constraint

units

base_unit_declaration { secondary_unit_declaration }

end units;

base_unit_declaration ::= identifier ;

secondary_unit_declaration ::= identifier = physical_literal;

physical_literal ::= [abstract_literal] unit_name.

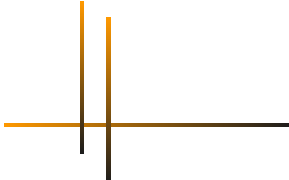
Физические типы



При объявлении физического типа сначала указывают базовую единицу размерности, а затем последующие единицы размерности как кратные базовой или предшествующим единицам.

Физические типы при описании устройств на ПЛИС обычно не применяются из-за сложности конструктивной реализации. Применяются при создании векторов входных воздействий.

Предопределенным физическим типом является `time`, который специфицирован в пакете "standard".



Предопределенный тип TIME

```
TYPE TIME IS RANGE -9223372036854775808 TO  
9223372036854775807
```

UNITS

```
fs;          -- femtosecond  
ps = 1000 fs;  -- picosecond  
ns = 1000 ps;  -- nanosecond  
us = 1000 ns;  -- microsecond  
ms = 1000 us;  -- millisecond  
sec = 1000 ms; -- second  
min= 60 sec;  -- minute  
hr = 60 min;  -- hour
```

```
END UNITS;
```

Составные типы

Составной тип - это группа значений под одним именем.

composite_type_definition ::=

array_type_definition -- массивы;

record_type_definition -- записи.

Массивы

Массивы объединяют элементы одного типа. Массивы могут иметь любую размерность. Тип элемента массива не может быть file. Можно задать тип массива как с неопределенными, так и с определенными границами. Задание типа с неопределенными границами дает возможность создавать массивы, имеющие один и тот же тип, но различные границы индексов.

array_type_definition ::=
unconstrained_array_definition|constrained_array_definiti
on

Описание массивов

Массивы рекомендуется описывать по шаблону:

```
TYPE __array_type_name IS ARRAY (INTEGER RANGE  $\langle \rangle$ ) OF  
__type_name;  
TYPE __array_type_name IS ARRAY (__integer DOWNTO __integer)  
OF __type_name;
```

Пример двух эквивалентных описаний двумерного массива 3*4:

```
TYPE ARRAY_M IS ARRAY (1 TO 3, 7 DOWNTO 4) OF POSITIVE;  
TYPE ARRAY_M IS ARRAY (INTEGER RANGE 1 TO 3, INTEGER  
RANGE 7 DOWNTO 4) OF POSITIVE;
```

Каждая пара границ массива должна иметь одинаковый тип. Элемент массива в свою очередь может быть массивом.

Описание массивов

Задание типа с неопределенными границами - очень удобная возможность для описания аппаратуры. Это дает как полную совместимость объектов данного типа, так и большую гибкость в параметризации описания. Примерами таких типов могут служить предопределенные типы `bit_vector` и `string`, специфицированные в пакете "standard»:

TYPE BIT_VECTOR IS ARRAY (NATURAL RANGE $\langle \rangle$) OF BIT;

TYPE STRING IS ARRAY (POSITIVE RANGE $\langle \rangle$) OF CHARACTER;

Направление и границы диапазона индексов не содержатся в определении указанных типов и должны быть указаны непосредственно при объявлении объектов данных типов.

Записи

Записи - это составной тип данных, элементы которых могут иметь различные типы.

```
record_type_definition ::=  
record  
    element_declaration  
    { element_declaration }  
end record
```

Все имена элементов (полей) записи должны быть различными.

Записи

Пример описания данных составного типа:

```
TYPE rec_type IS RECORD  
    hour      : INTEGER RANGE 0 TO 24;  
    min       : INTEGER RANGE 0 TO 60;  
    per       : day;  
END RECORD;
```

Тип данных `day` описан как:

```
TYPE day IS (morn, din, evn, night);
```

Функции преобразования типов данных

Пакет `arith` содержит четыре функции для преобразования типов данных `signed`, `unsigned`, `integer` и `std_ulogic`.

Функция `conv_integer` преобразует типы данных `integer`, `unsigned`, `signed` или `std_ulogic` к типу `integer`. Значения операндов функции `conv_integer` ограничивается диапазоном от -2147483647 до 2147483647, т.е. 31-битное представление значения `unsigned` или 32-битное представление значения `signed`.

Функция `conv_unsigned` преобразует типы данных `integer`, `unsigned`, `signed` или `std_ulogic` в тип `unsigned` с указанием разрядности.

Функция `conv_signed` преобразует типы данных `integer`, `unsigned`, `signed` или `std_ulogic` в тип `signed` с указанием разрядности.

Функция `conv_std_logic_vector` преобразует типы данных `integer`, `unsigned`, `signed` или `std_logic` в тип `std_logic_vector` с указанием разрядности.