Операторы

Набор операторов (операций) в VHDL обеспечивает возможность работы с предопределнными типами данных.

Список операций приведен в табл. 1.

Строки табл. 2 располагаются в порядке старшинства (от низшего к высшему) операторов. Операторы, находящиеся в одной строке, обладают одинаковым старшинством (приоритетом). Таким образом, операции нижней строки табл. 2 обладают наибольшим приоритетом и выполняются первыми, в частности, логический оператор not выполняется прежде других логических операторов.

Таблица 1

Операции языка VHDL

Обозначение	Название
not	логическое НЕ
and	логическое И
or	логическое ИЛИ
nand	логическое И-НЕ
nor	логическое ИЛИ-НЕ
xor	исключающее ИЛИ
xnor	эквивалентность
=	равно
/=	не равно
<	меньше
<=	меньше либо равно
>	больше
>=	больше либо равно
+	сложение, присвоение знака +
−	вычитание, присвоение знака -
&	конкатенация
*	умножение
/	деление
mod	модуль
rem	остаток
**	возведение в степень
abs	абсолютное значение

Исходя из контекста VHDL-кода следует отличать оператор <= (назначение сигнала) и оператор <= (меньше либо равно). Следует

также отличать унарные операции присвоения знака +,− от соответствующих бинарных операций сложения и вычитания.

Замечание. Для устранения неоднозначностей трактовки старшинства операций используются скобки.

Например, выражение «A nand B nand C» неверно (синтаксическая ошибка). Данное выражение не представляет трехвходовый элемент И-НЕ (трехвходовую NAND-ячейку). Правильная запись «not (A and B and C)». Запись «A nand (B nand C)» не есть то же самое, что «(A nand B) nand C».

Таблица 2

Классификация операций

Класс операций	Операции					
Логические	and	or	nand	nor	xor	xnor
Сравнения	=	/=	<	<=	>	>=
Сложения и конкатенации	+	-	&			
Присвоение знака	+	—				
Умножения	*	/	mod	rem		
Смешанные	**	abs	not			

Примеры арифметических операторов.

Сложение (+)

```
RealX2 + 2.0 -- если RealX2 есть сигнал типа
             -- вещественный, то число 2 должно
             -- быть записано как вещественное
1us + 3ns    -- 1003ns
```

Вычитание (-)

```
8.33 - 5     -- неправильно, оба числа должны
             -- быть одного типа
BusWidth - 1 -- допустимо, если BusWidth типа integer
```

Умножение (*)

4*SomeVal -- допустимо, если SomeVal
-- типа integer или time

Деление (/)

CLK/2 -- тип CLK - integer
5.0/2.0 -- результат - вещественное число 2.5
10ns/2ns — результат 5 типа integer, но не time

Модуль (mod)

6 **mod** 4 -- результат 2
6 **mod** (-4) -- результат -2
(-6) **mod** 4 -- результат 2

Остаток деления (rem)

6 **rem** 4 -- результат 2
6 **rem** (-4) -- результат 2
(-6) **rem** 4 -- результат -2

Экспонента (**)

c ** 0.5 -- неправильно в VHDL
A ** 2 -- эквивалентно A * A
B ** 3 -- эквивалентно B * B * B

Абсолютное значение (abs)

abs 1 -- результат 1
abs(-l) -- результат 1
abs(5 * (-2)) -- результат 10

В стандарте языка VHDL операции **rem** и **mod** определяются следующим образом.

Для операции L **rem** R должно выполняться соотношение

L = (L / R) * R + (L **rem** R),

где L/R - целая часть частного, (L **rem** R) - результат выполнения операции **rem** (остаток). Остаток имеет *знак операнда L.*

Для операции L **mod** R должно выполняться соотношение

L = N * R + (L **mod** R),

где N — некоторое целое число, (L **mod** R) - результат выполнения операции **mod.** Результат имеет *знак операнда R.*

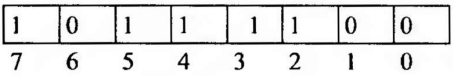
Следует быть внимательным при переходе от битовых векторов к числам:

Bit vector (0 **to** 7); -- возрастающий диапазон

1	0	1	1	1	1	0	0
0	1	2	3	4	5	6	7

Число 61 (десятичное), старший разряд справа

Bit vector (7 **downto** 0); -- убывающий диапазон



Число 188 (десятичное), старший разряд слева

Логические операторы выполняются для следующих типов данных:

- boolean;
- bit, bit_vector;
- std_logic, std_logic_vector;
- std_ulogic, std_ulogic_vector.

Логические операторы **and, or, xor** имеют одинаковое старшинство и выполняются слева направо в выражениях. Оператор **not** имеет более высокое старшинство и выполняется прежде других операторов. В сложных логических выражениях порядок выполнения операторов регулируется скобками. Рекомендуем читателю применять скобки в затруднительных случаях. Например, для выражения

Z <= A **and not** B **or** C;

будет только отрицание B, в выражении

Z <= A **and not** (B **or** C); будет отрицание подвыражения B **or** C, находящегося в скобках.

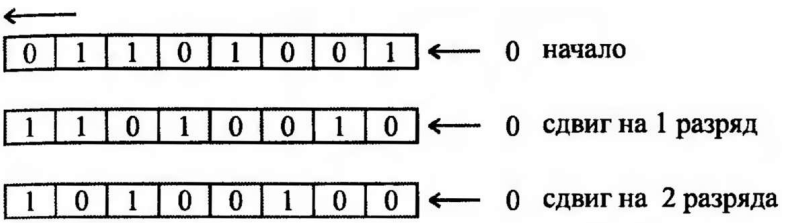
Операторы сдвига

Данные операторы поясним на примере семиразрядного битового вектора My Bus

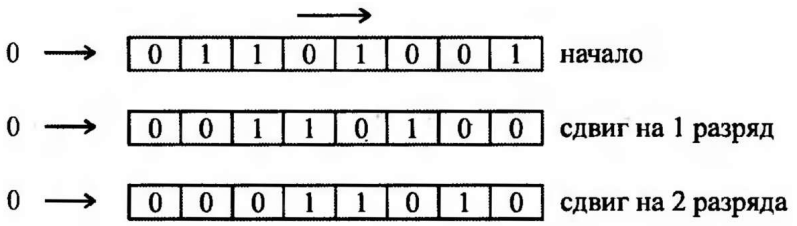
signal MyBus: bit_vector(7 **downto** 0) := "01101001";

Логический сдвиг влево (Shift Left Logical)

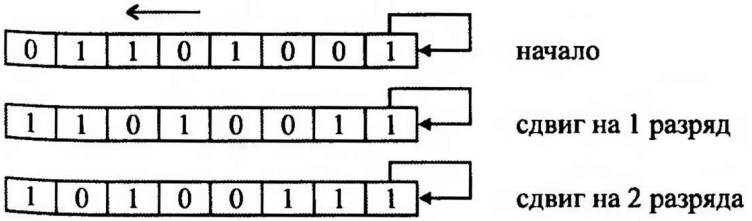
оператор **sll**



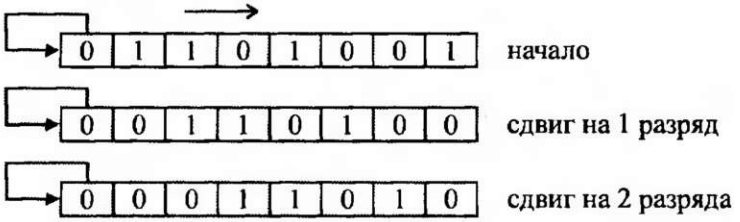
Логический сдвиг вправо (Shift Right Logical)
оператор **srl**



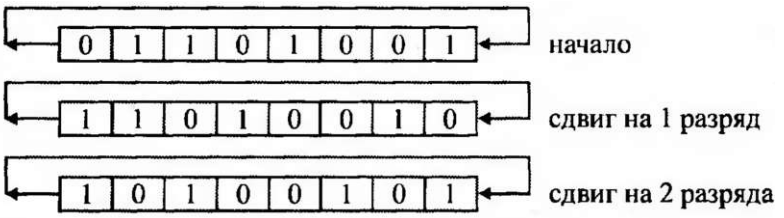
Арифметический сдвиг влево (Shift Left Arithmetic)
оператор **sla**



Арифметический сдвиг вправо (Shift Right Arithmetic)
оператор **sra**



Вращение логическое влево (Rotate Left Logical)
оператор **rol**



Вращение логическое вправо (Rotate Right Logical)
оператор **ror**