Технологии проектирования компьютерных систем

Лекция 6. Выражения

Определение

Выражение - это формула, которая используется для вычисления нового значения, или одиночный термин, имеющий значение.

Выражение можно рассматривать как совокупность бинарных выражений, имеющих левый операнд, правый операнд и оператор, связывающий левый и правый операнды (x+y). В результате вычисления бинарного выражения получается новый операнд, который вступает далее в бинарные отношения со своими соседями.

Унарное выражение рассматривают как бинарное выражение, в котором отсутствует левый операнд (x).

Константы могут употребляться в выражениях.

Операторы

Класс операций	Операции						
Логические	and	nand	or	nor	xor	xnor	not
Сравнения	=	/=	<	<=	>	>=	
Сложения и конкатенации	+	-	&				
Присвоение знака	+	-					
Сдвига	sll	srl	sla	sra	rol	ror	
Умножения и деления	*	/	mod	rem			
Смешанные	**	abs					

Операторы

Предопределенные операторы языка приведены по классам. При необходимости эти операторы можно перегружать: переопределять их семантику и расширять область их применимости для различных типов.

Строки в таблице располагаются в порядке старшинства (от низшего к высшему) операторов. Операторы, находящиеся в одной строке, обладают одинаковым старшинством (приоритетом). Таким образом, операции нижней строки в таблице обладают наибольшим приоритетом и выполняются первыми.

Логические операторы

Определение:

logical_operator ::= and | nand | or | nor | xor | nxor | not

Логические операторы выполняют следующие функции: and - логическое 'и'; nand - логическое 'и-не'; or - логическое 'или'; nor - логическое 'или-не'; xor - логическое 'исключающее или'; nxor - логическое 'исключающее или-не'; not - логическое отрицание.

В логических бинарных выражениях массивы должны быть одинаковой длины. Вычисления здесь производятся над парами элементов, равно отстоящими от левой границы. Результатом является массив, индексация элементов в котором совпадает с индексацией элементов левого операнда, то есть их подтипы совпадают.

Оператор not является логическим унарным.

Логические операторы

Логические операторы выполняются для следующих типов данных:

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

Логические операторы and, nand, or, nor, xor, nxor имеют одинаковое старшинство и выполняются слева направо в выражениях. Операция not имеет более высокое старшинство и выполняется прежде других операторов. В сложных логических выражениях порядок выполнения операторов регулируется скобками. Рекомендуется применять скобки в затруднительных случаях.

Логические операторы

Оператор называется перезагруженным (overloaded), если для него создано более одного функционального определения для различных типов данных. Например, оператор AND определен для 7 типов данных. Оператор AND может быть дополнительно описан для других типов данных.

В языке VHDL можно применять три способа вызова операторов:

- префиксный;
- инфиксный;
- с использованием квалифицирующего выражения.

Логические операторы

Три способа вызова операторов:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY operat IS
 PORT (  op1, op2        : IN     std_logic_vector(3 downto 0);
              res1, res2, res3     : OUT std_logic_vector(3 downto 0));
END operat ;
ARCHITECTURE maxpld OF operat IS
BEGIN
 res1 <= op1 AND op2;
 res2 <= "AND"(op1,op2);
 res3 <= std_logic_vector'(op1 AND op2);
END maxpld;
```

Операторы сравнения

Определение:

relational_operator ::= = | / = | < | <= | > | > =

Операторы сравнения предназначены для выполнения следующих операций: = - равно; /= - не равно; < - меньше; <= - меньше-равно; > - больше; >= - больше-равно.

Операторы сравнения выполняются для следующих типов данных:

- std_logic_vector;
- std_ulogic_vector;
- signed;
- unsigned;
- integer.

Операторы сдвига

В VHDL-93 были введены предопределенные операторы сдвига. Операторы сдвига можно использовать, когда левым операндом является одномерный массив из элементов типа bit (bit_vector) или boolean, а правым операндом является любое неотрицательное целое.

shift_operator ::= sll | srl | sla | sra | rol | ror

sll (shift left logical) - сдвиг левый логический. Освобождающиеся элементы массива заполняются значением, определенным по умолчанию для данного типа (для типа bit это '0').

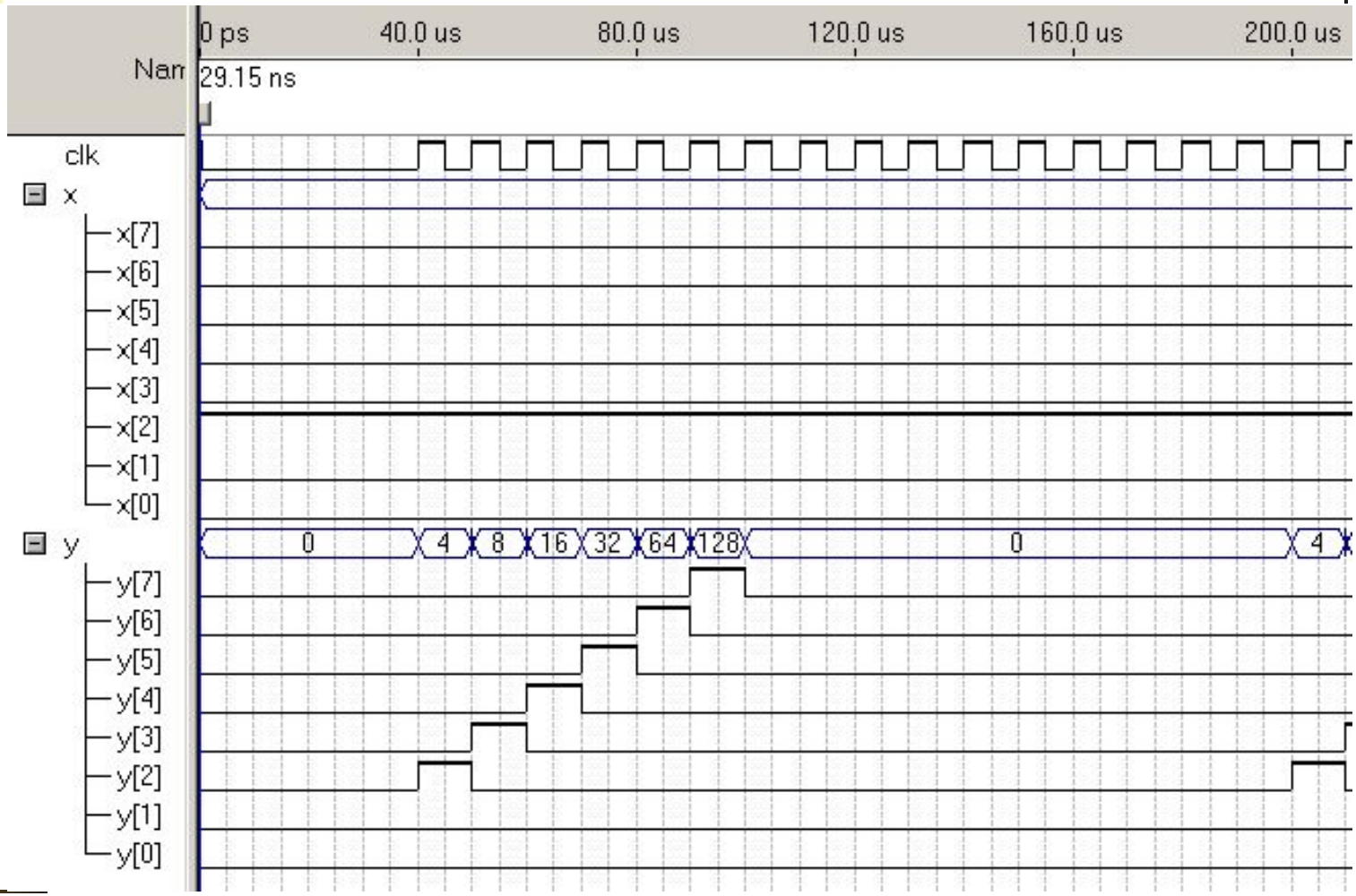
srl (shift right logical) - сдвиг правый логический. Освобождающиеся элементы массива заполняются значением, определенным по умолчанию для данного типа.

Операторы сдвига

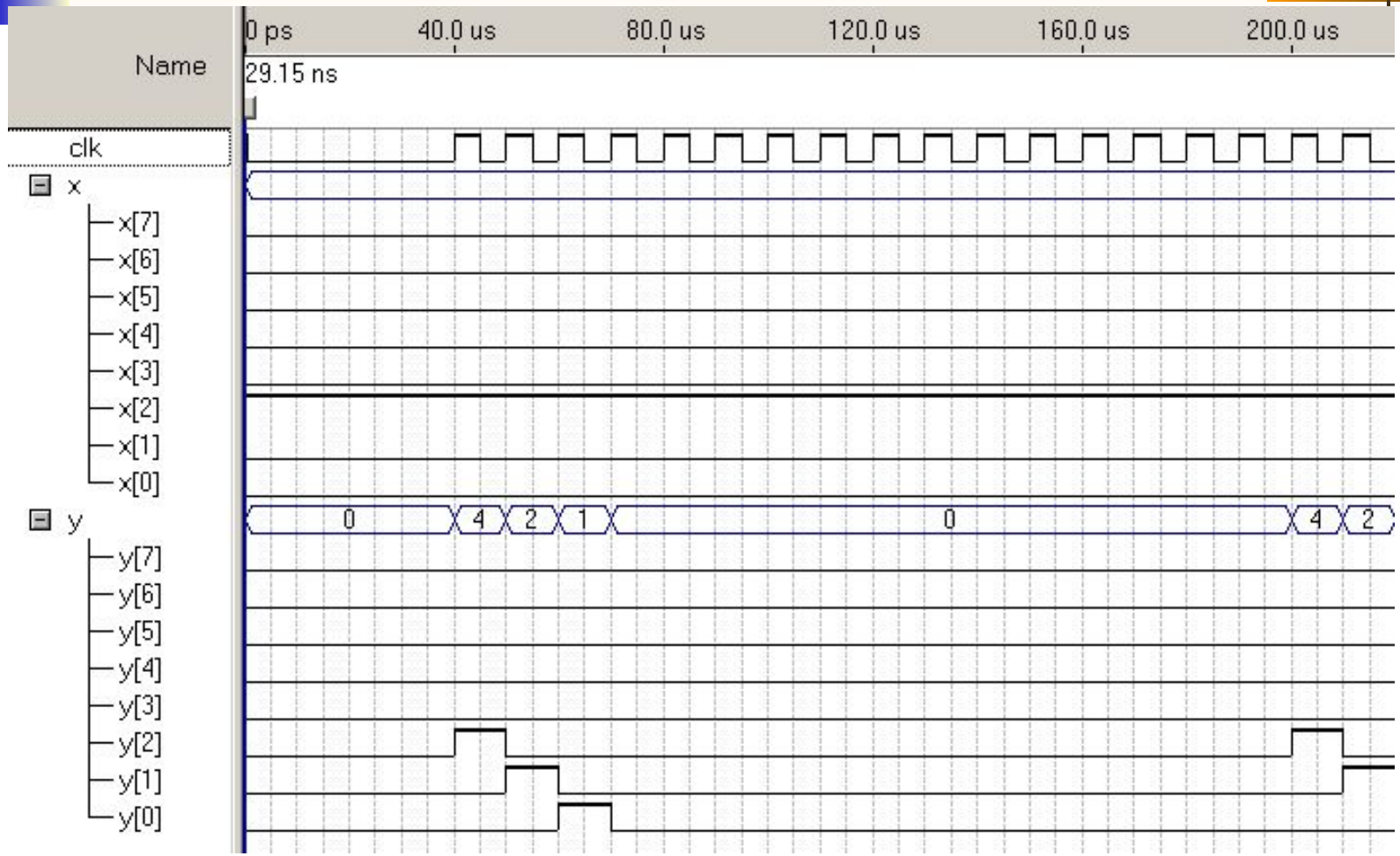
Рассмотрим временные диаграммы сдвига, полученные по описанию:

```
ENTITY vsll IS
PORT ( clk    : IN bit;
   x  : IN BIT_VECTOR(7 DOWNTO 0);
   y  : OUT BIT_VECTOR(7 DOWNTO 0));
END vsll;
ARCHITECTURE arch OF vsll IS
  SIGNAL shift :integer RANGE 0 TO 15;
    BEGIN
      PROCESS (clk)
          BEGIN
   IF (clk'EVENT AND clk = '1')
    THENshift <= shift +1; y <= x sll shift;  ELSE null;
              END IF;   END PROCESS;  END arch;
```

Оператор сдвига sll



Оператор сдвига srl

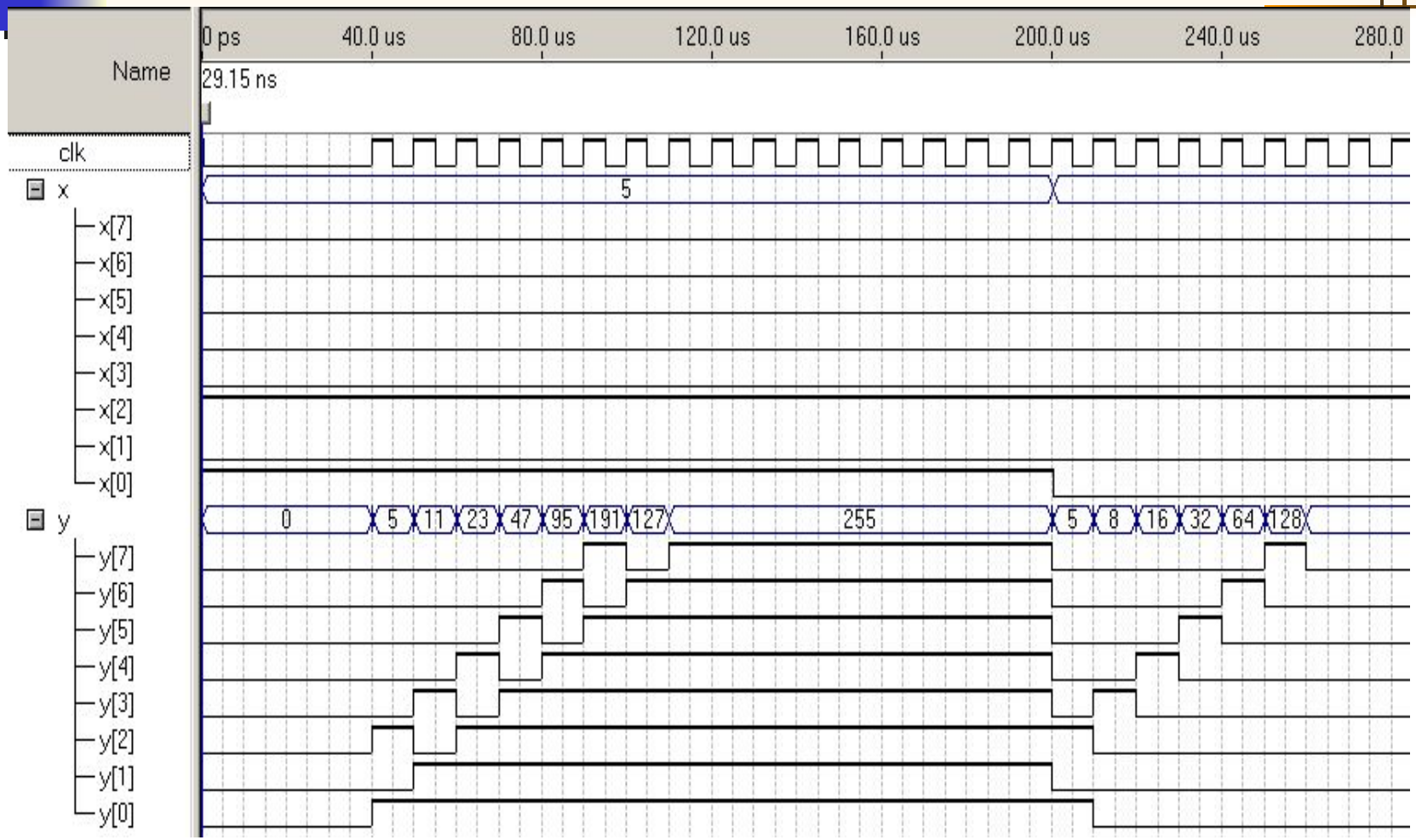


Операторы сдвига sla и sra

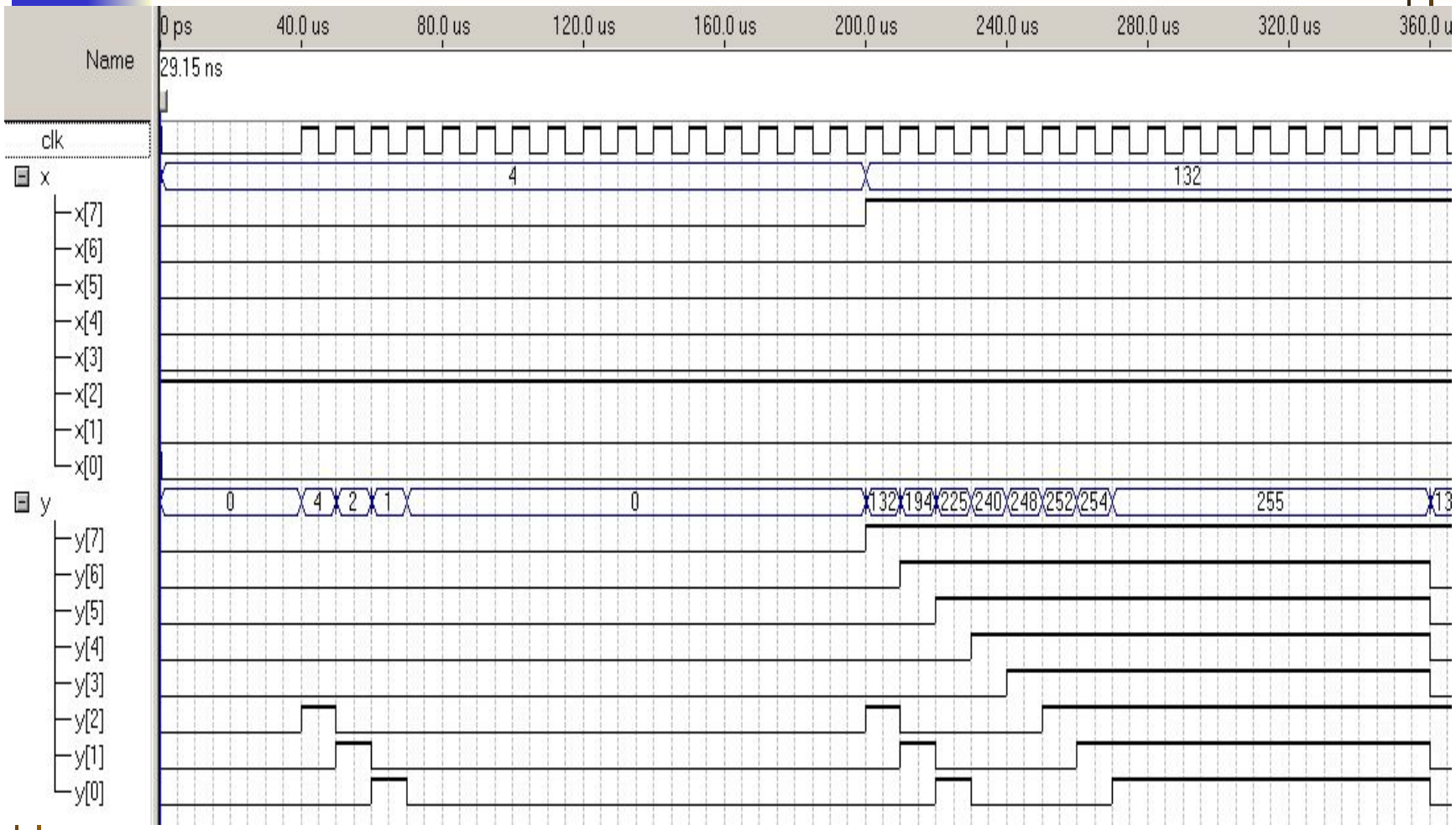
sla (shift left arithmetic) - сдвиг левый арифметический. Освобождающиеся элементы заполняются значениями крайнего правого элемента массива.

sra (shift right arithmetic) - сдвиг правый арифметический. Освобождающиеся элементы заполняются значениями крайнего левого элемента массива.

Оператор сдвига sla



Оператор сдвига sra



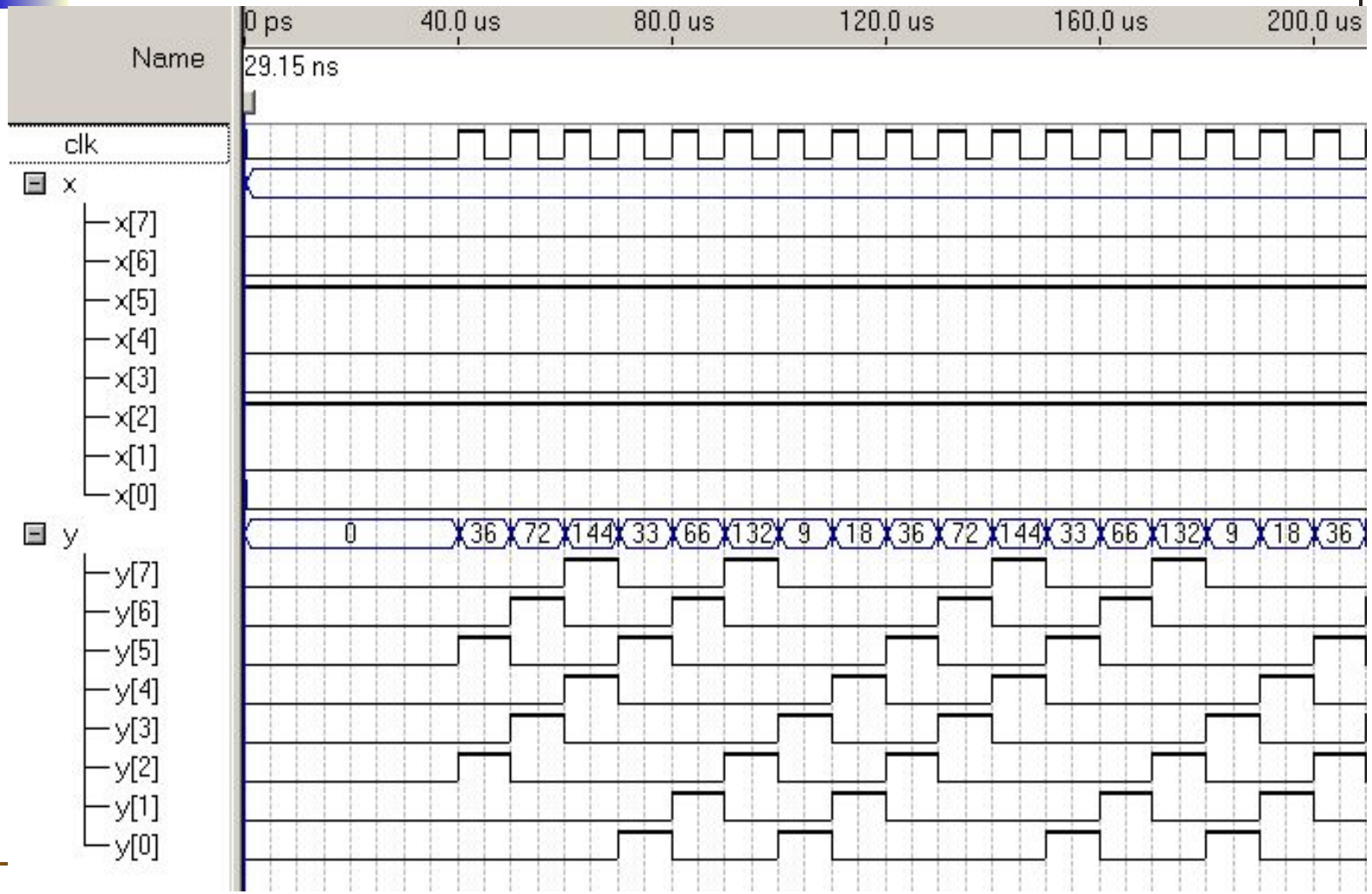
Операторы сдвига rol и ror

rol (rotate left logical) - сдвиг циклический левый логический.

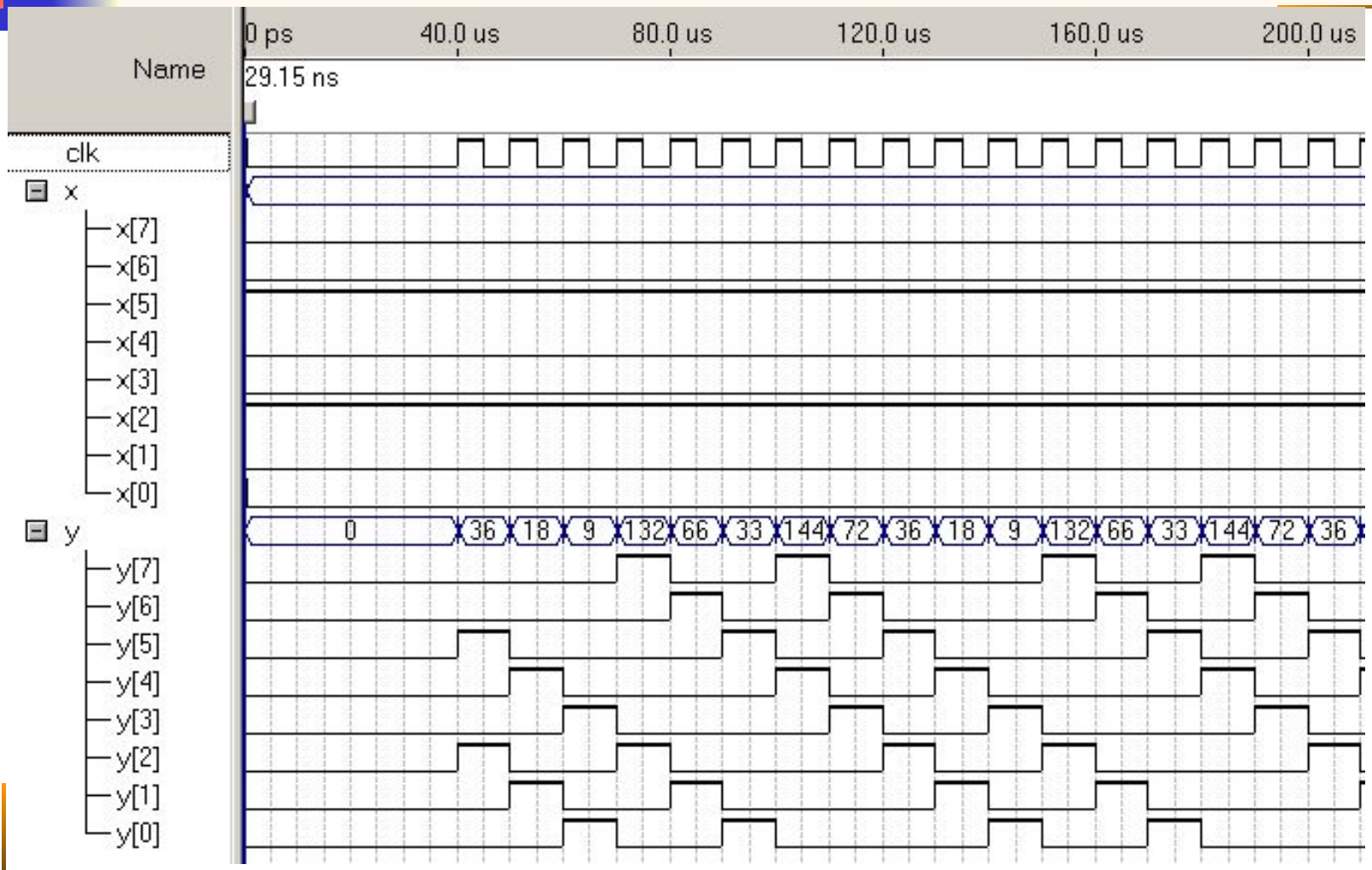
ror (rotate right logical) - сдвиг циклический правый логический.

Операторы сдвига имеют одинаковое старшинство с мультипликативными операторами.

Оператор сдвига rol



Оператор сдвига ror



Аддитивные операторы

Определение:

adding_operator ::= + | - | &

Аддитивные операторы предназначены для выполнения операций суммирования, вычитания и конкатенации.

Операторы суммирования и вычитания выполняются для следующих типов данных:

- std_logic_vector;
- std_ulogic_vector;
- integer;
- signed;
- unsigned.

Аддитивные операторы

Оператор конкатенации & служит для объединения двух одномерных массивов, одномерного массива и скаляра, двух скаляров. Любой скаляр здесь рассматривается как одномерный массив, элементы которого индексируются в диапазоне 1 To 1. Скаляры и элементы массивов, участвующие в конкатенации, могут быть любого типа, но эти типы должны совпадать.

В результате конкатенации образуется одномерный массив большей длины. Индекс этого массива непрерывен и значение его левого индекса совпадает со значением левого индекса левого операнда. Конкатенация является удобным средством при описании устройств на регистровом уровне для объединения различных разрядов нескольких регистров в один вектор.

В VHDL-93 допустимо объединять только восходящие (n1 To n2), или нисходящие (nl Downto n2) массивы.

Мультипликативные операторы

Определение:

multiplying_operator ::= * | / | mod | rem

Мультипликативные предназначены для выполнения операций умножения, деления, нахождения модуля и остатка от деления.

Функции указанных операторов находятся в пакете ieee.numetic_std.

Оператор * и / поддерживается для следующих типов данных:

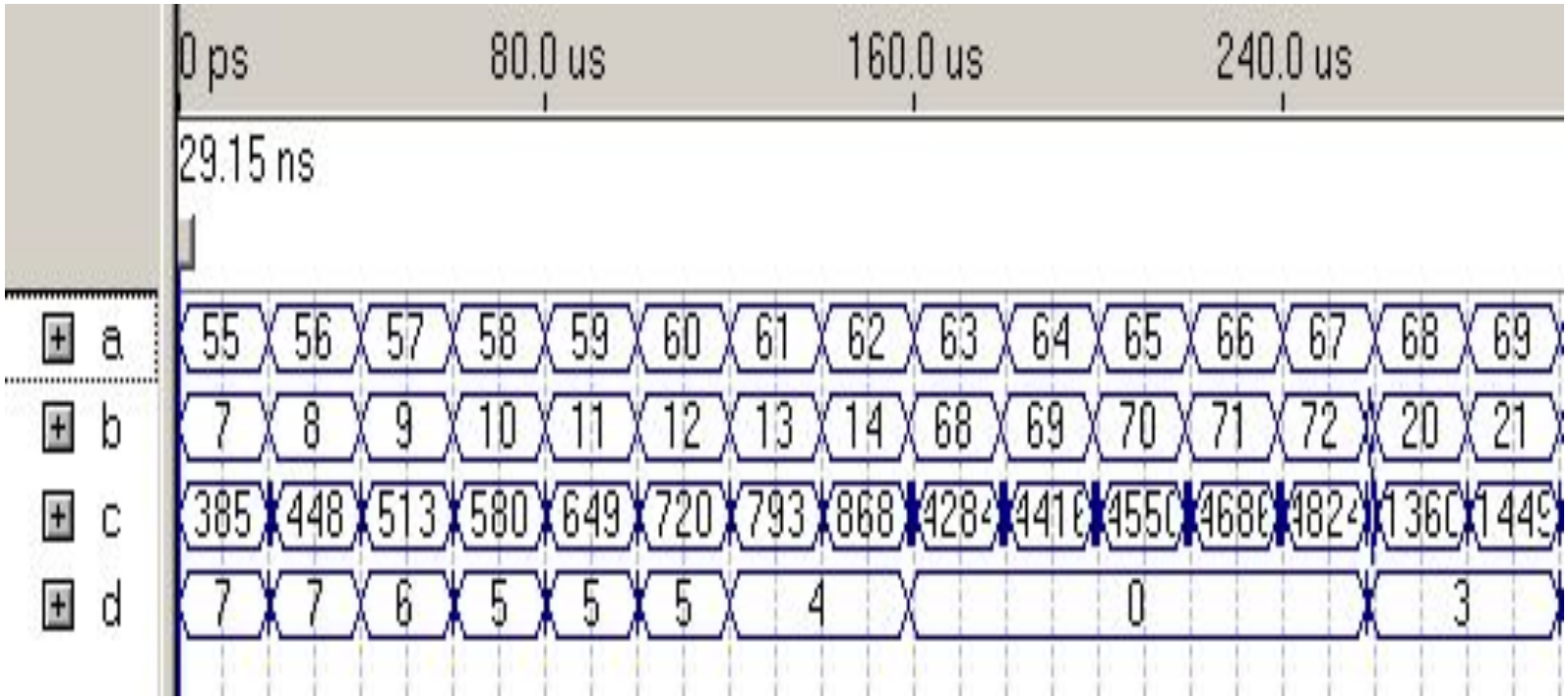
- std_logic_vector;
- std_ulogic_vector;
- integer;
- signed;
- unsigned.

Операторы * и /

Рассмотрим действие операторов * и / по описанию цифрового устройства:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL, ieee.numeric.std.all;
ENTITY div IS
   PORT (a  : IN  unsigned (15 downto 0);
--разрядность делимого должна быть больше
  b  : IN  unsigned (7 downto 0);
  c  : OUT unsigned (23 downto 0);
  d  : OUT unsigned (15 downto 0));
--разрядность произведения необходимо предварительно рассчитать
    END ENTITY div;
ARCHITECTURE arch OF div IS
BEGIN
      c <= a*b; d <= a/b;
END;
```

Операторы * и /



Из анализа временных диаграмм следует, что при делении формируется только целая часть результата.

Мультипликативные операторы

Для операций остатка от деления и модуля выполняются следующие условия.

A = (A/B)*B + (A rem B),

где выражение (A rem B) имеет знак A, а абсолютное значение меньше абсолютной величины B.

Для целочисленного деления выполняется тождество:

(-A)/B = - (A/B) = A/(-B)

Выражение (A mod B) имеет знак B, абсолютное значение меньше абсолютной величины B, и определяется отношением:

A = B*целое + (A mod B).

Мультипликативные операторы

Рассмотрим значения операторов mod и rem для различных чисел.

5 **rem** 3 = 2;
5 **mod** 3 = 2.

(–5) **rem** 3 = –2;
(–5) **mod** 3 = 1.

(–5) **rem** (–3) = –2;
(–5) **mod** (–3) = –2.

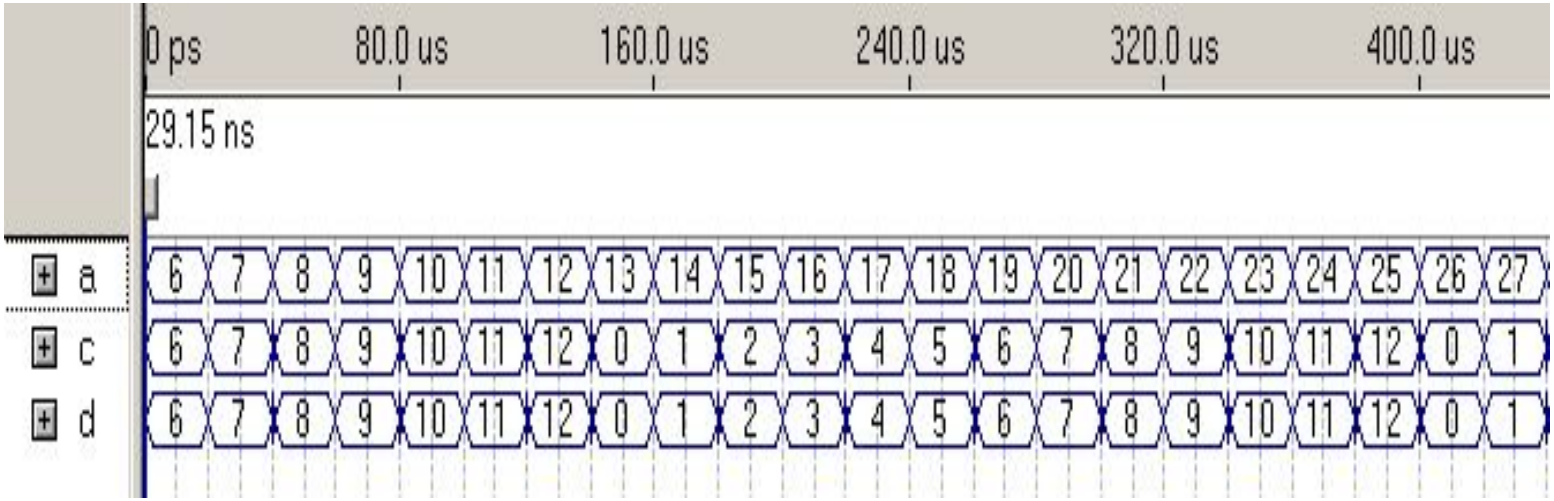
5 **rem** (–3) = 2;
5 **mod** (–3) = –1.

Мультипликативные операторы

Действие мультипликативных операторов рассмотрим по описанию и временным диаграммам работы.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL, ieee.NUMERIC_STD.all;
ENTITY vmod IS
 PORT (  a  : IN  unsigned (7 downto 0);
            c,d  : OUT unsigned (7 downto 0));
END ENTITY vmod;
ARCHITECTURE arch OF vmod IS
 BEGIN
       c <= a mod 13;
       d <= a rem  13;
END;
```

Мультипликативные операторы



Как следует из временных диаграмм значения mod и rem для положительных чисел совпадает.

Знаковые операторы

Определение:

sign ::= + | -

Знаковые операнды допустимы для операндов, имеющих скалярные типы. Они не могут следовать непосредственно за мультипликативными, аддитивными операторами или смешанным оператором "экспонента".

Эти выражения ошибочны:
(A*-B), (C/+D), (F**-G), (H+-I).

Здесь необходимы скобки:
A * (- B), C / (+ D), F ** (- G), H + (-I).

Смешанные операторы

Определение:

miscellaneous_operator ::= ** | abs

Смешанные операнды предназначены для возведения числа в степень или получения абсолютного значения.

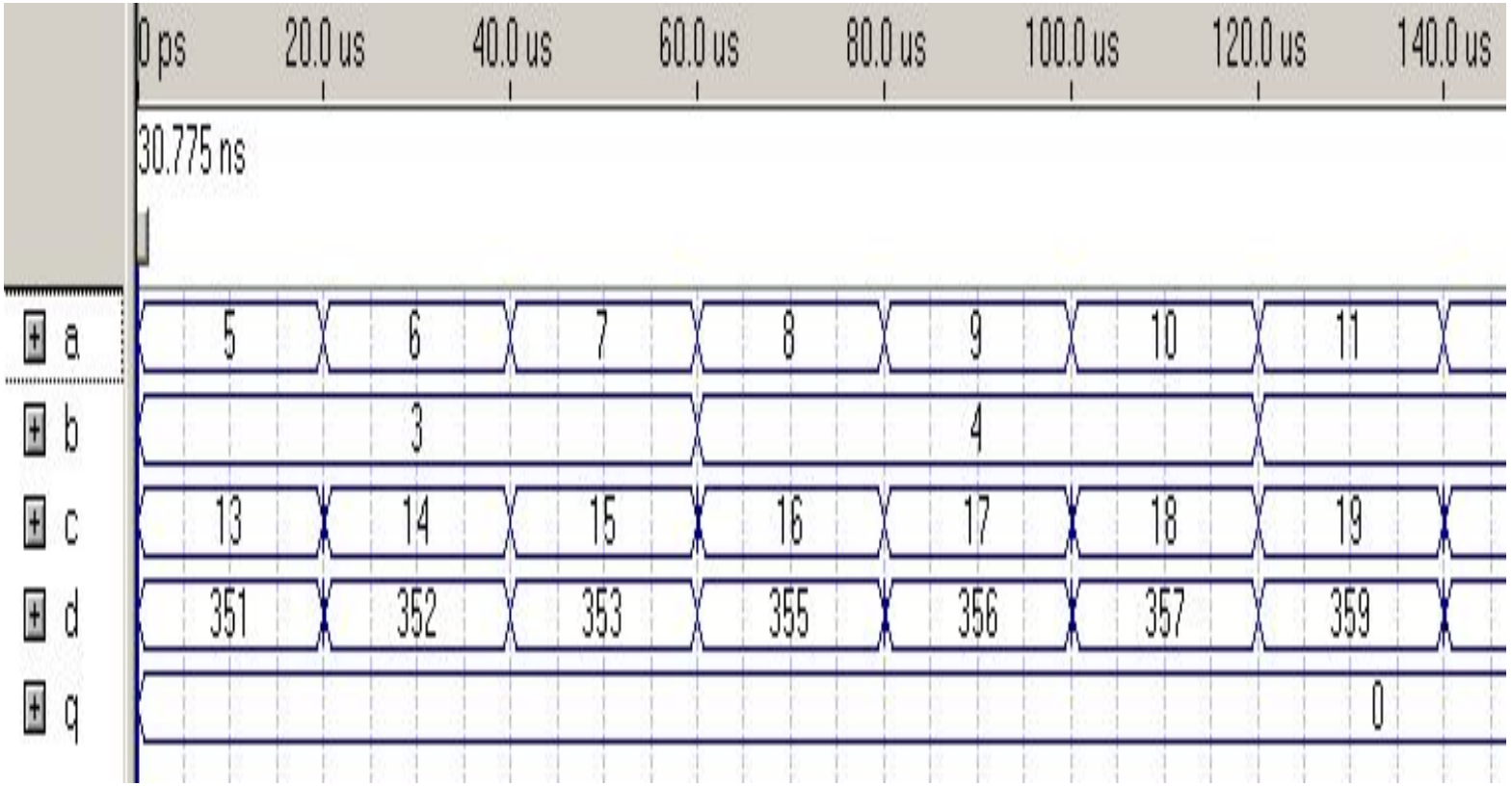
Смешанные операторы

Оператор ** находит ограниченное применение при описание цифровых устройств из-за сложности технической реализации. Возводить в степень допускается только целые числа, причем показатель степени должен быть декларирован в параметрах Entity.

Смешанные операторы

```
library ieee;
use ieee.std_logic_1164.all, ieee.std_logic_unsigned.all;
entity stepen is
     generic (const :integer := 3);
     port( a, b :in  std_logic_vector (2**(const+2) downto 0);
      c,d,q        :out std_logic_vector (2**(const+2) downto 0));
 end entity;
 architecture rtl of stepen is
     constant koef :integer := 2**const;
   begin
            c <= koef + a;  d <= a+b+7**const;
            -- q <= 6** koef;  q<= const**koef; q<= a**const;
            -- q<= const**4;  q<= 5**4;
   end architecture;
```

Смешанные операторы



Операнды

Определение.

primary ::=

name | literal | aggregate | function_call |

qualified_expression | type_conversion | allocator | (expression)

Перечисленные выше операнды могут участвовать в выражениях.

Операнд Name

В качестве операнда используют шесть форм имен, которые были рассмотрены:

name ::=

simple_name -- простое имя;
| operator_symbol -- символ оператора;
| selected_name -- селективное имя;
| indexed_name -- индексное имя;
| slice_name -- вырезка имени;
| attribute_name -- имя атрибута.

Операнд Literal

Литерал служит для задания значений объектов языка. В VHDL имеется пять типов литералов.

literal ::=

numeric_ literal -- числа (целые и реальные): 0, 2E4, 3.14 ns;

|enumeration_literal -- перечисления: '0', ram;

| string_ literal -- строки (строковый литерал): "are not";

| bit_string_literal -- битовые строки: b"111_111»;

| null -- значение указателя (access_type) в никуда.

Числа могут быть представлены в виде абстрактных и физических литералов.

numeric_literal ::=

abstract_literal

| physical_literal

Операнд Literal

Литералы перечисления - это литералы, применяемые для описания типа данных - перечисления. В качестве этих литералов применяют идентификаторы и символьные литералы.

Строковые литералы и битовые строки были рассмотрены в разделе “Лексемы”.

Литерал null представляет собой нулевое значение для любого типа.

Операнд Aggregate

Агрегат - это базовая операция, объединяющая одно или несколько значений в массив или запись.

Элементы этого объединения связываются (ассоциируются) при вычислениях (например, при присвоении агрегата одномерному массиву) позиционно (в этом случае конструкция choices => отсутствует), или поименованно.

aggregate ::= (element_association { , element_association })
element_association ::= [choices =>] expression
choices ::= choice { | choice }
choice ::=
simple_expression
| discrete_range
| element_simple_name
| others.

Операнд Aggregate

При именованном присвоении сначала формируется цель - множество элементов массива, затем, после символов "=>" - то, что надо записать в эту цель, и далее, через разделитель ", ", формируется следующая цель.

Позиционное связывание в агрегатах должно предшествовать поименованным ассоциациям. Поименованная ассоциация "Others =>" может быть использована (при необходимости) только в конце агрегата. Допустимо только однократное связывание. Если агрегат имеет единственное значение, то оно должно быть связано поименованно.

Операнд Aggregate

Опишем, для примера, присвоение значений переменной:
VARIABLE q : BIT_VECTOR (0 **TO** 3).
Возможно 7 форм присвоения значений с помощью агрегатов:

1. позиционная **q := ('0', '1', '1', '0');**

2. именованная **q := (0=>'0', 2=> '1', 1=> '1', 3=> '0');**

3. смешанная **q := ('0', '1', 1=> '1', 3=> '0');**

4. с использованием вырезки имен

q := (0 =>'0', 1 TO 2 =>'1', 3=> '0');

Операнд Aggregate

5. с использованием зарезервированного слова OTHERS (слово OTHERS можно использовать в случае применения только позиционной или именованной ассоциации)

q := (1 TO 2 =>'1', OTHERS => '0');

q := ('0', '1', OTHERS => '0');

6. с применением только слова OTHERS, если необходимо всем разрядам установить одно и то же значение

q := (OTHERS => '0');

q := (OTHERS => '1');

7. с перечислением разрядов, которым следует установить одинаковые значения. Перечисляемые разряды отделяют друг от друга вертикальной чертой

q := (0|3 =>'0', 1|2 => '1').

Операнд Function Call

Вызов функции приводит к выполнению тела функции. Он определяет имя функции, которая будет вызвана, и фактические параметры, которые должны быть связаны с формальными параметрами функции. В результате выполнения функции формируется значение с типом, определенным при объявлении функции.

function_call:: =
function_name [(actual_parameter_part)]

Каждому формальному параметру должен соответствовать фактический параметр.

Операнд Qualified Expression

Позволяет однозначно указывать тип или подтип операнда.

qualified_expression ::= type_mark'(expression)| type_mark'aggregate

Необходимость использования квалифицирующего выражения возникает при наличии перегруженных (overload) операторов, функций или операндов, то есть когда, например, существуют одинаково именованные операторы, работающие с различными типами операндов и, естественно, имеющие различное функционирование.

Операнд Type Conversion

Конверсия типа используется для перевода одного типа данных в другой тип данных.

type_conversion ::= type_mark (expression)

Операнд Allocator

Предназначен для создания анонимных объектов, доступ к которым осуществляется через указатели (access) на эти объекты.

allocator ::= new subtype_indication | new qualified_expression

Использование динамически размещаемых объектов удобно не только при задании необходимого объема аппаратуры (ROM, RAM и пр.), но и при моделировании таких структур, как FIFO, LIFO. Сами динамически размещаемые объекты, естественно, должны быть детерминированных размеров.

Статическое выражение

Для вычисления значений констант, параметров настройки и начальных значений всех других объектов языка используются выражения, которые являются статическими (Static Expression), то есть вычисляются единожды в процессе выполнения программы. Кроме этого, статические выражения участвуют при определении типов.