Проектирование цифровых устройств на языке VHDL

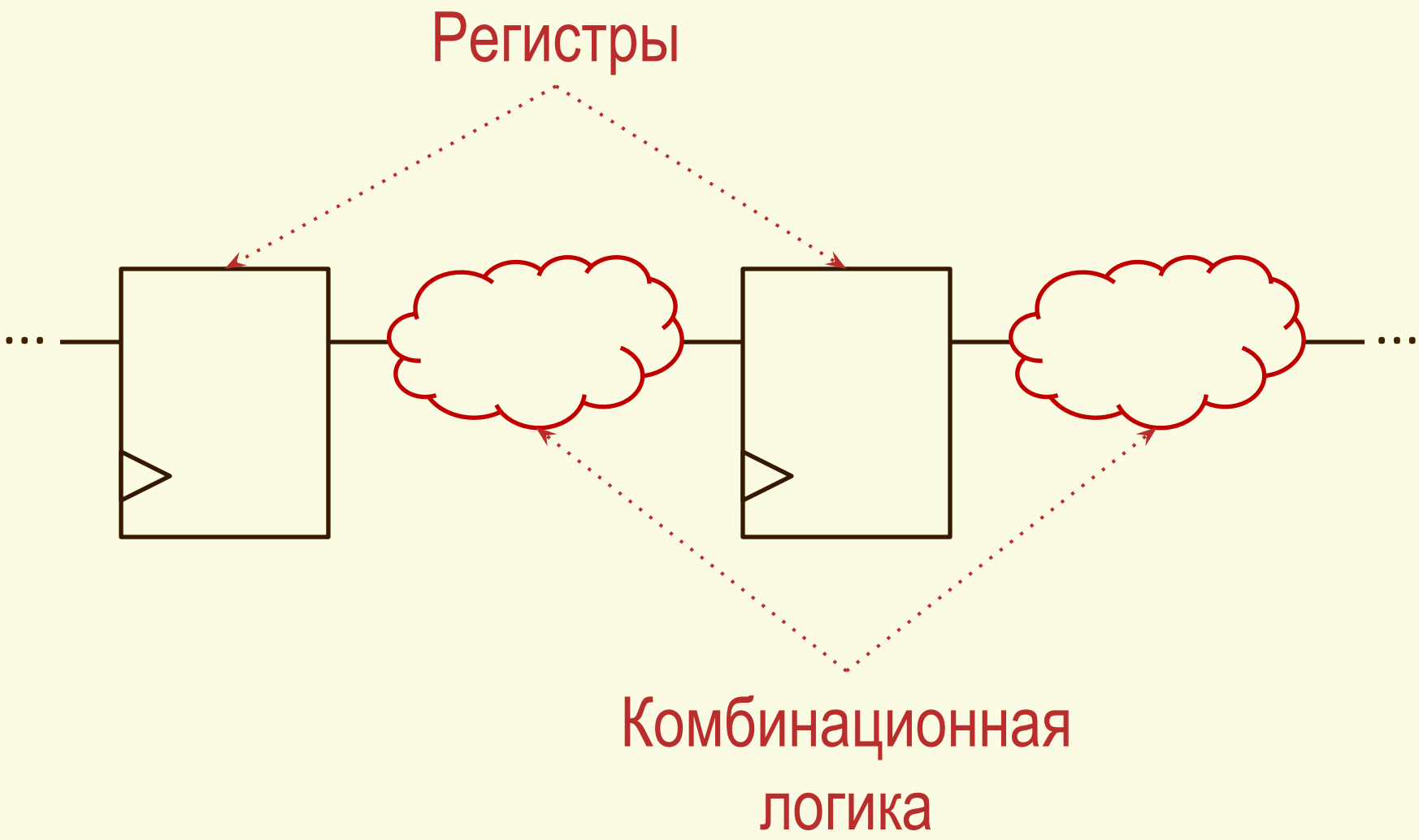
Описание комбинационных схем

Комбинационные схемы

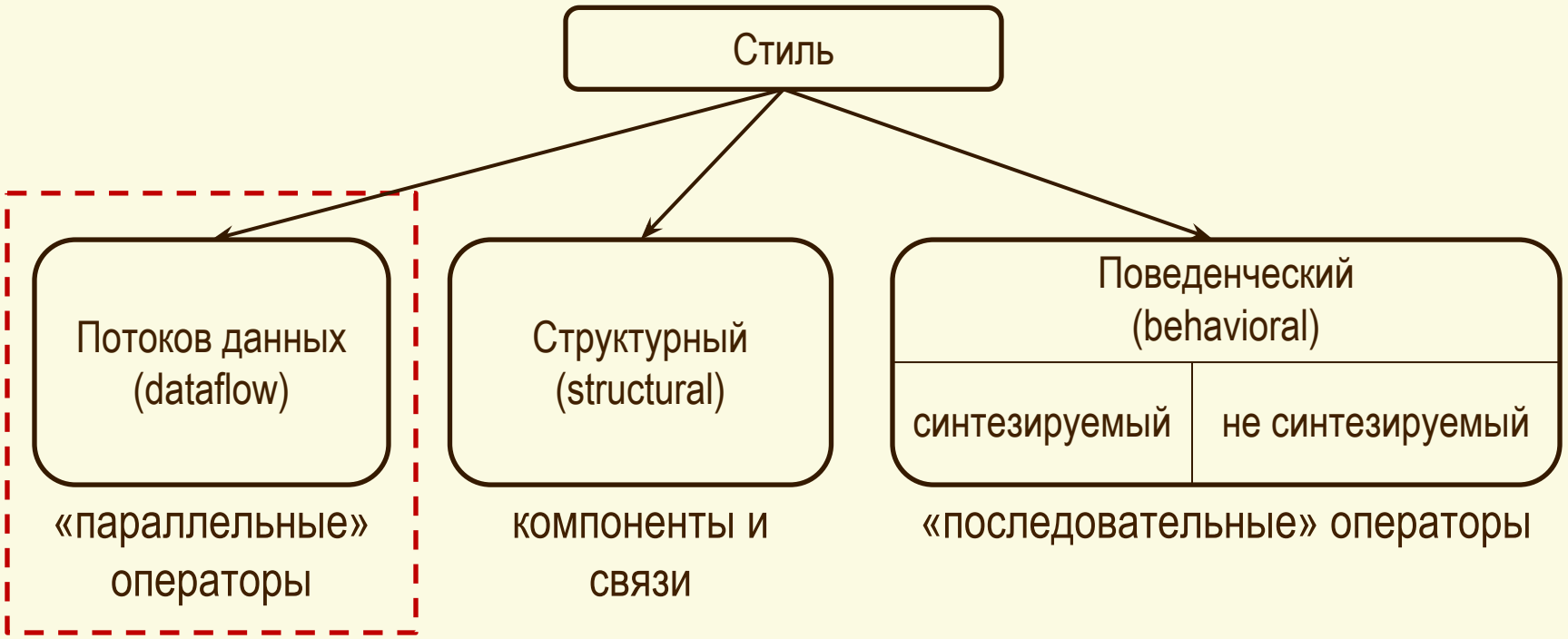
- Сдвиговые
  - Сдвиг влево / вправо
  - Вращение влево / вправо
- Логические
  - Реализация булевых функций
- Арифметические
  - Сложение, вычитание
  - Умножение, деление
- Схемотехнические
  - Мультиплексоры, демультиплексоры
  - Шифраторы, дешифраторы

Классификация условная

Структура цифрового устройства

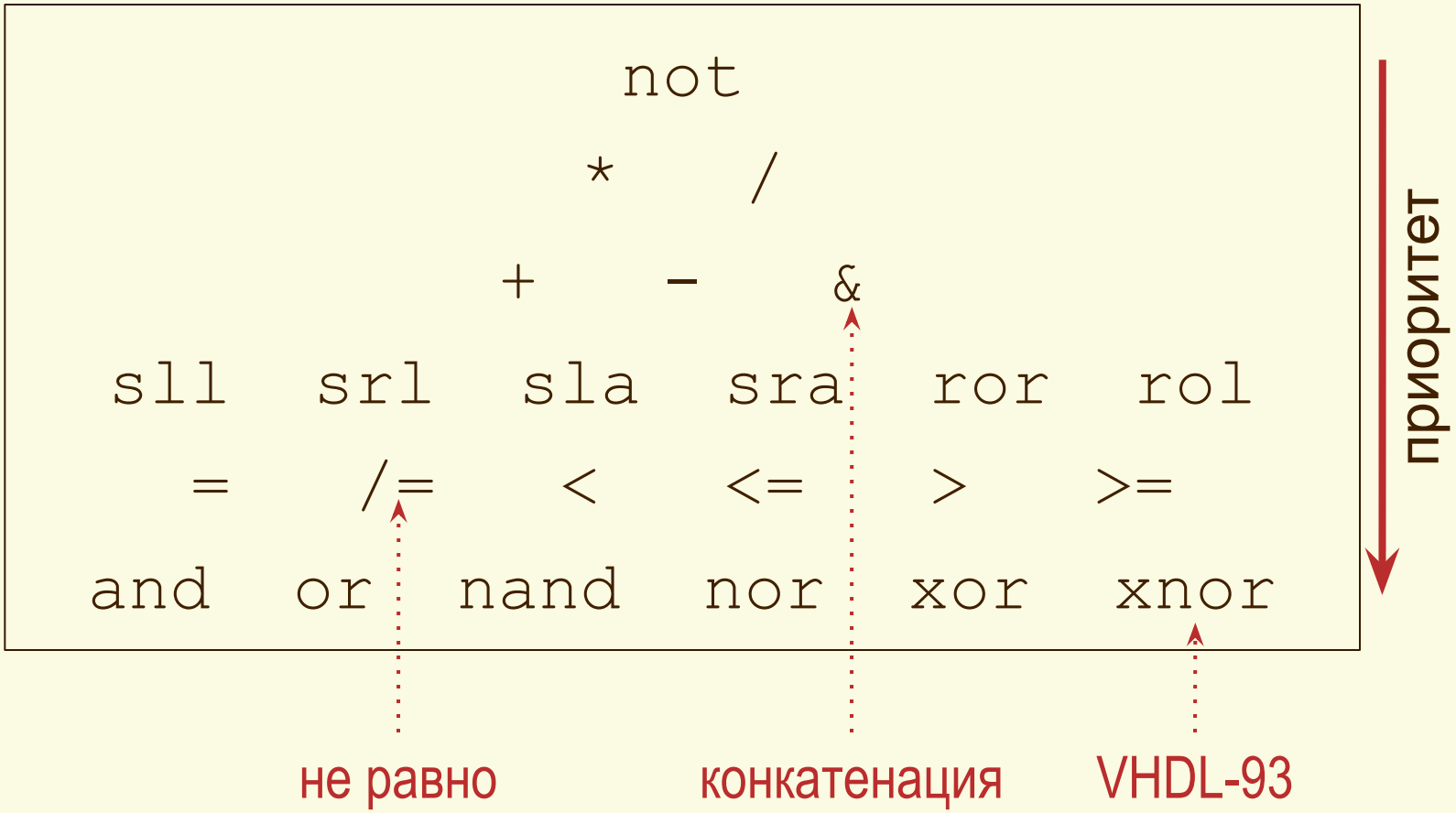


Стили описания на VHDL



Арифметические и логические операторы, операторы отношений

Основные операторы



Арифметические операторы

- Умножение (*)
- Деление (/)
- Сложение (+)
- Вычитание (–)

- Приоритет операций обычный (арифметический)
- Результат синтеза – **комбинационная** схема

Пакеты IEEE

- **std_logic_1164**
  - Стандарт IEEE
  - Определяет типы std_logic и std_logic_vector
    - Пример: std_logic_vector(7 downto 0)
  - Описывает логические операции
- **std_logic_arith**
  - Не является стандартом (разработан Synopsys)
  - Определяет типы signed (знаковое целое) и unsigned (беззнаковое целое)
    - Пример: unsigned(7 downto 0)
  - Описывает арифметические операции

Пакеты IEEE

- **std_logic_unsigned**
  - Не является стандартом (разработан Synopsys)
  - Заставляет компилятор интерпретировать std_logic_vector как unsigned
- **std_logic_signed**
  - Не является стандартом (разработан Synopsys)
  - Заставляет компилятор интерпретировать std_logic_vector как signed
- Не используйте одновременно std_logic_unsigned и std_logic_signed
  - Преобразования типов можно выполнять явно

Пакеты IEEE

- **numeric_std**
  - Стандарт IEEE
  - Определяет типы signed и unsigned
  - Описывает арифметические операции
  - Несовместим с пакетом std_logic_arith

Подключение арифметических пакетов IEEE

- Работа с беззнаковыми целыми числами (unsigned)

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;                -- определение арифметических операций
use ieee.std_logic_unsigned.all;             -- std_logic_vector <=> unsigned
```

- Работа со знаковыми целыми числами (signed)

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;                -- определение арифметических операций
use ieee.std_logic_signed.all;               -- std_logic_vector <=> signed
```

- Работа как с беззнаковыми, так и со знаковыми целыми числами

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;                -- определение арифметических операций
```

Преобразование типов данных

		numeric_std	std_logic_arith
Type Conversion			
std_logic_vector	-> unsigned	unsigned(*arg*)	unsigned(*arg*)
std_logic_vector	-> signed	signed(*arg*)	signed(*arg*)
unsigned	-> std_logic_vector	std_logic_vector(*arg*)	std_logic_vector(*arg*)
signed	-> std_logic_vector	std_logic_vector(*arg*)	std_logic_vector(*arg*)
integer	-> unsigned	to_unsigned(*arg*, *size*)	conv_unsigned(*arg*, *size*)
integer	-> signed	to_signed(*arg*, *size*)	conv_signed(*arg*, *size*)
unsigned	-> integer	to_integer(*arg*)	conv_integer(*arg*)
signed	-> integer	to_integer(*arg*)	conv_integer(*arg*)
integer	-> std_logic_vector	integer -> unsigned/signed ->std_logic_vector	
std_logic_vector	-> integer	std_logic_vector -> unsigned/signed ->integer	
unsigned + unsigned	-> std_logic_vector	std_logic_vector(*arg1* + *arg2*)	*arg1* + *arg2*
signed + signed	-> std_logic_vector	std_logic_vector(*arg1* + *arg2*)	*arg1* + *arg2*
Resizing			
unsigned		resize(*arg*, *size*)	conv_unsigned(*arg*, *size*)
signed		resize(*arg*, *size*)	conv_signed(*arg*, *size*)

Присваивание

Присваивание

- Простое присваивание
- Условное присваивание (when-else)
- Выборочное присваивание (with-select-when)

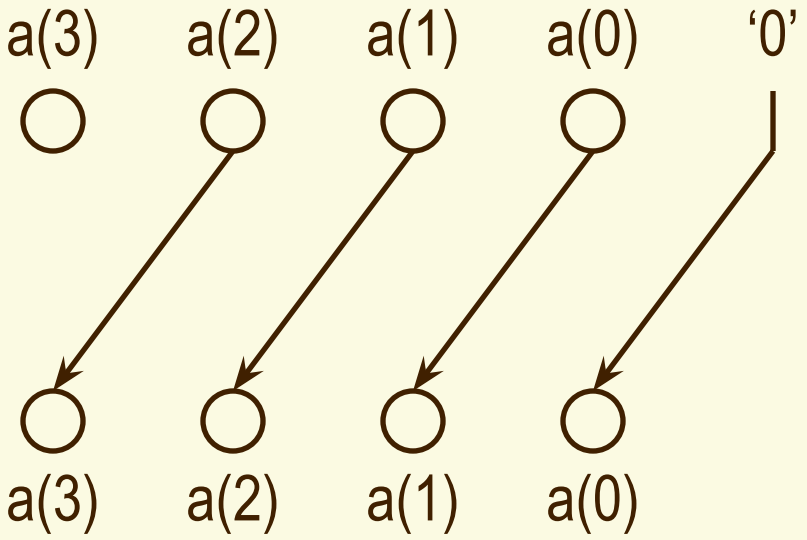
Простое присваивание

- Оператор **<=**
- Присваивает значение выражения сигналу

```
sig_name <= expression;
```

Простое присваивание – пример

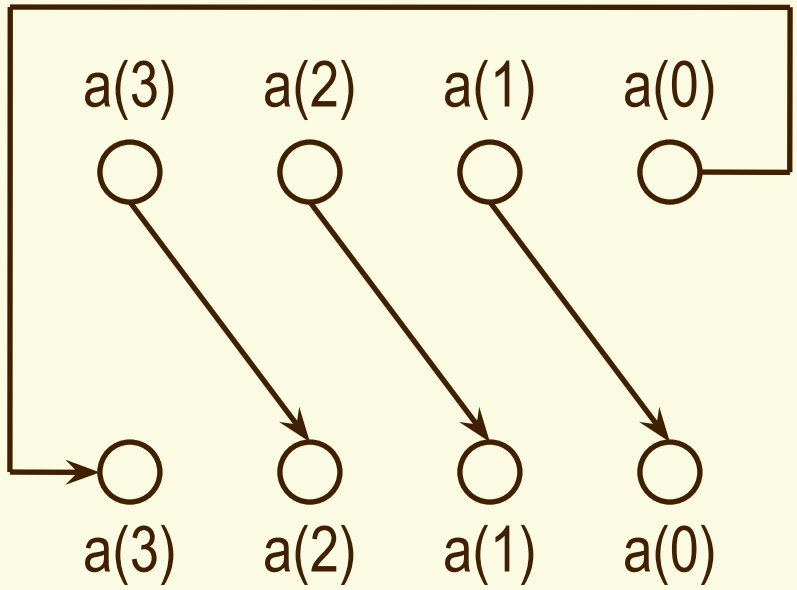
- Сдвиг влево (дополнение нулем)




```
a_shl <= a(2 downto 0) & ‘0’;
```

Простое присваивание – пример

- Вращение вправо




```
a_ror <= a(0) & a(3 downto 1);
```

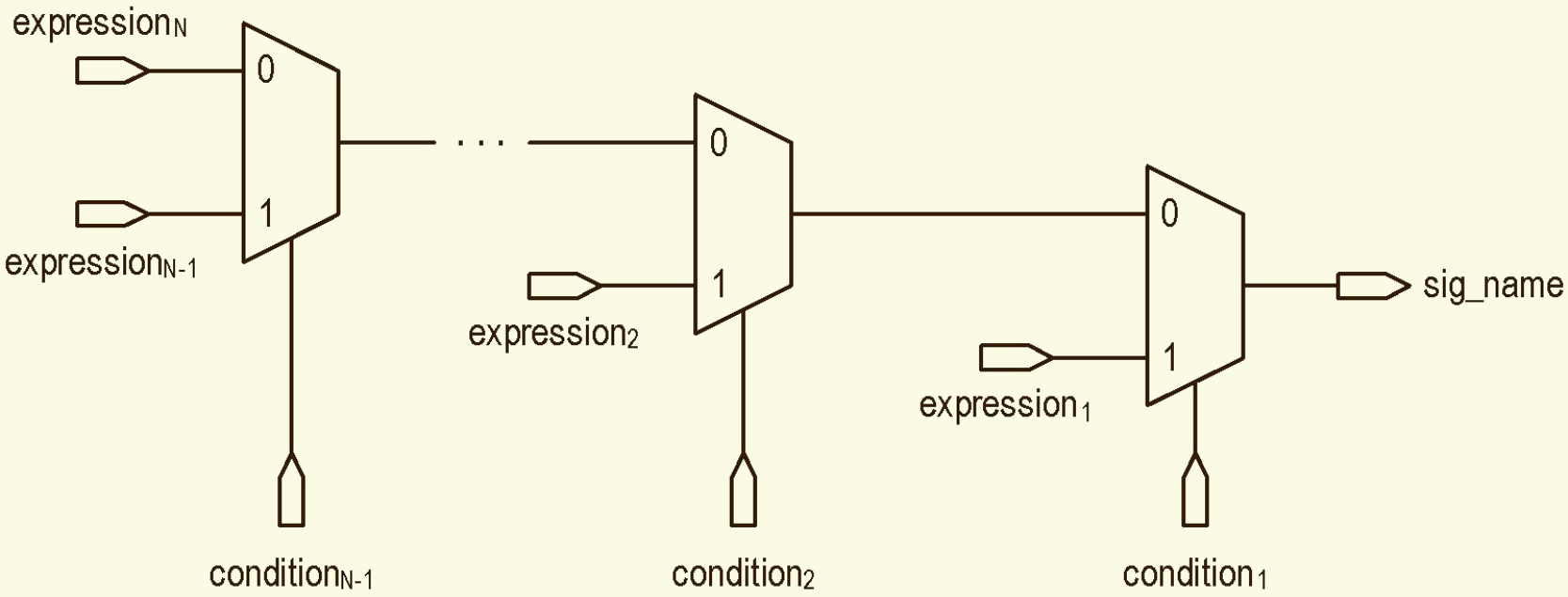
Условное присваивание

- Оператор **<=** с конструкцией **when-else**
  - Аналог – оператор if-else
- Присваивается первое значение, для которого условие истинно
  - Если все условия ложны, используется безусловная ветвь **else**
  - Если все условия ложны и безусловная ветвь else отсутствует, синтезируется **триггер-защелка**

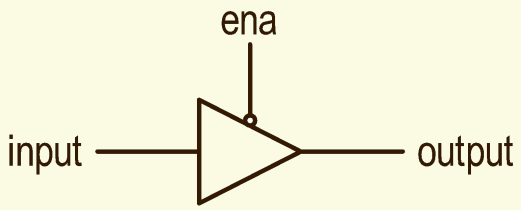
```
sig_name <=expression1 when condition1 else
                       …
                       expressionN-1 when conditionN-1
else
                                  ← безусловная ветвь else
                       expressionN;
```

Условное присваивание

sig_name <=expression$_1$ **when** condition$_1$ **else**
…
expression$_{N-1}$ **when** condition$_{N-1}$
else
expression$_N$;



Условное присваивание – пример




```
library ieee;
use ieee.std_logic_1164.all;

entity tri_state is
    port (  ena:        in std_logic;
            input:      in std_logic_vector(7 downto 0);
            output:     out std_logic_vector(7 downto 0));
end tri_state;

architecture dataflow of tri_state is
begin
    output <=  input when ena = '0' else
                    (others => 'Z');
end architecture;
```

все явно не указанные компоненты вектора

Управляемое присваивание

- Оператор **<=** с конструкцией **with-select-when**
  - Аналог – оператор switch-case
- Присваиваемое значение зависит от управляющего выражения

```
with choise_expression select
    sig_name <=expression1 when choise1,
                …
               expressionN-1 when choiseN-1,
               expressionN when others;
```

разделяются запятыми

Управляемое присваивание

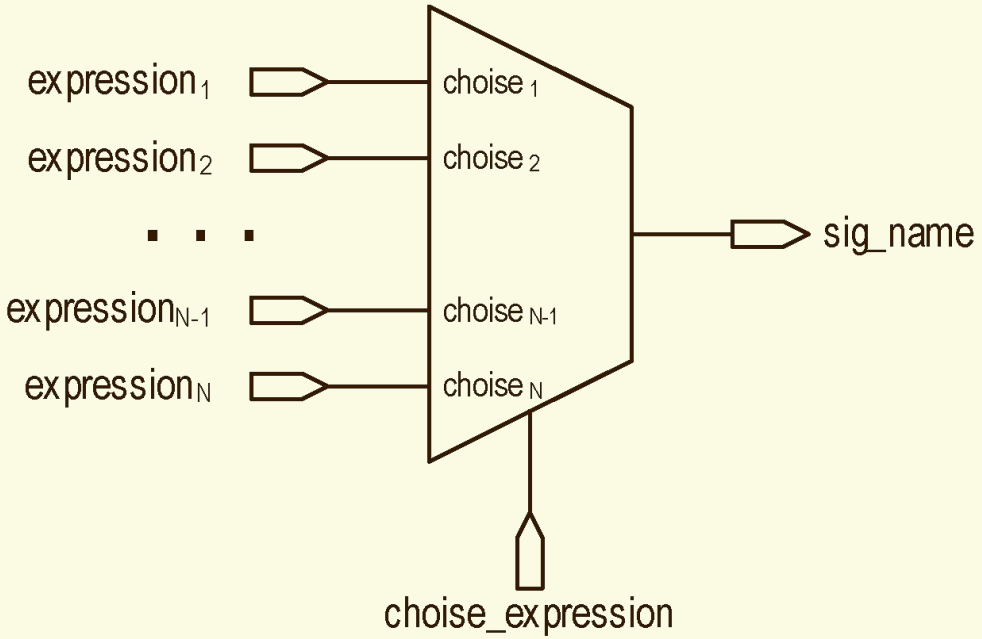
with choise_expression **select**

sig_name **<=**expression$_1$ **when** choise$_1$,

…

expression$_{N-1}$ **when** choise$_{N-1}$,

expression$_N$ **when others**;



Описание вариантов выбора (choise$_i$)

- Указание одиночного значения
- Перечисление нескольких значений

  … **when** choise$_1$ | choise$_2$ | … | choise$_N$,
- Задание диапазона значений

  … **when** choise$_1$ **to** choise$_2$,
- Ключевое слово **others** – все остальные случаи
  - Всегда используйте others – возможных вариантов больше, чем ‘0’ и ‘1’

```
with sel select
    result <= a   when “000”,
                  b   when “011” to “110”,
                  c   when “001” | “111”,
                  d   when others;
```

Простые правила

- Для описания чисто комбинационной логики используйте только параллельные операторы

Простое присваивание	Простые логические операции (и, или, не, несложные функции) Простые арифметические операции (сложение, вычитание, умножение) Сдвиги / вращения на постоянное число разрядов
Условное присваивание Управляемое присваивание	Мультиплексоры / демультиплексоры Шифраторы / дешифраторы Тристабильные буферы

Генерация выражений

Генерация выражений

- Конструкция **for-generate**

```
label:    for var in range generate
                concurrent statements
          end generate;
```

- Конструкция **if-generate**

```
label:    if condition generate
                concurrent statements
          end generate;
```

Генерация выражений – пример

- Подсчет четности вектора
- Интерфейс объекта:
  - Вход – 8-разрядный вектор data
  - Выход – бит четности parity
- Функциональность объекта:

$$\text{parity} = \left(\sum_{i=0}^{7} \text{data}(i) \right)_{\bmod 2}$$

Генерация выражений – пример

```
library ieee;
use ieee.std_logic_1164.all;

entity par_count is
        port (      data:           in std_logic_vector(7 downto 0);
                    parity:     out std_logic);
end par_count;

architecture dataflow of par_count is
        signal temp: std_logic_vector(7 downto 0);
begin
        temp(0) <= data(0);

gen: for i in 1 to 7 generate
                    temp(i) <= temp(i-1) xor data(i);
              end generate;

        parity <= temp(7);
end architecture;
```