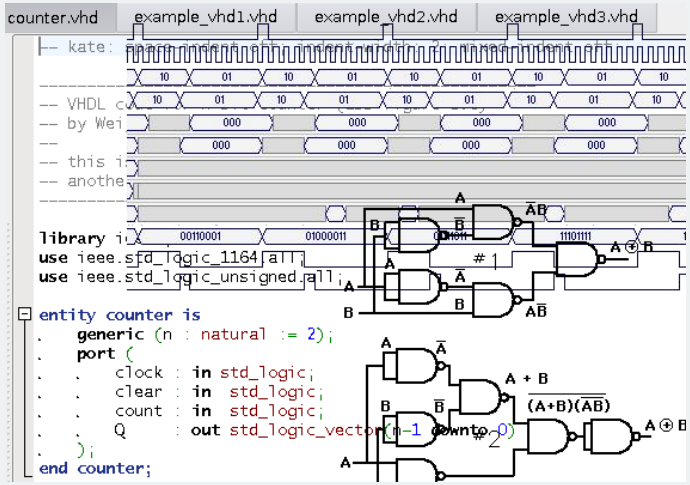
МИЭТ
Национальный исследовательский университет «МИЭТ»
Кафедра ПКИМС

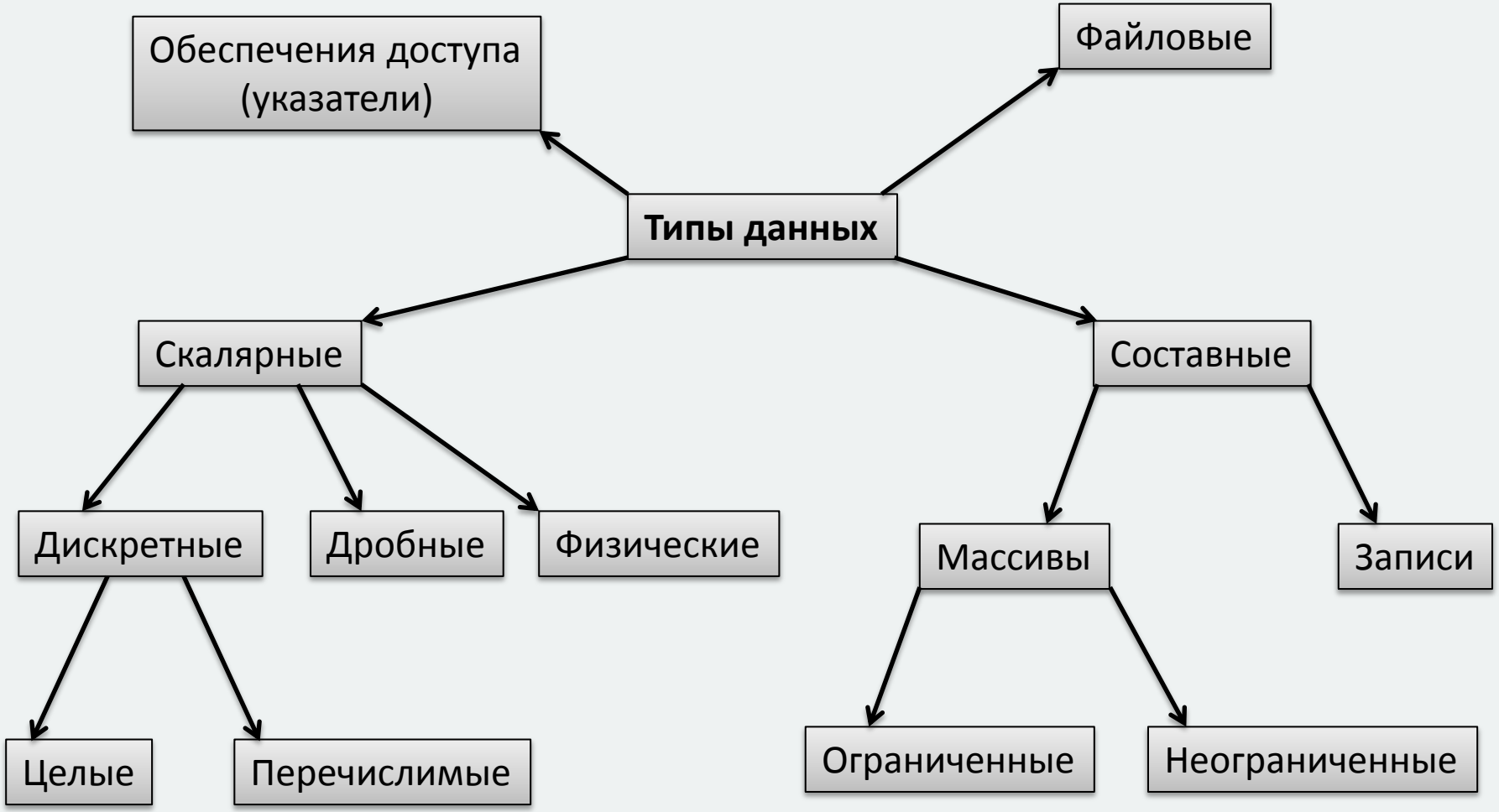
Лингвистические средства проектирования



Лекция 4

Поведенческое архитектурное тело. Конфигурация.

Типы типов данных



Предопределённые типы данных в языке VHDL

Скалярные:

`NATURAL, POSITIVE, INTEGER, REAL`

Перечислимые:

`BIT``, CHARACTER, BOOLEAN`

Векторные:

`BIT_VECTOR``, STRING`

Физические:

`TIME`

Целочисленные типы данных

```
INTEGER     от -(2^31 - 1) до +(2^31 - 1)

NATURAL     от 0 до +(2^31 - 1)

POSITIVE    от 1 до +(2^31 - 1)
```

```
type small_int is range -1023 to 1024;
```

или

```
subtype small_int is Integer range -1023 to +1024;
```

или

```
variable small_int: Integer is range -1023 to +1024;
```

Перечислимый тип данных

Перечислимые типы:

```
type BIT is ('0', '1');

type severity_level is (note, warning, error, failure);


type status is (unknown, idle, work);
```

Использование перечислимых типов:

```
signal A: bit;
variable B: bit;

...
A <= '1';
B := '0';
```

```
signal S1: status;
variable S2: status;

...
A <= idle;
B := work;
```

Физические типы данных

Имя физического типа данных

Минимальное значение

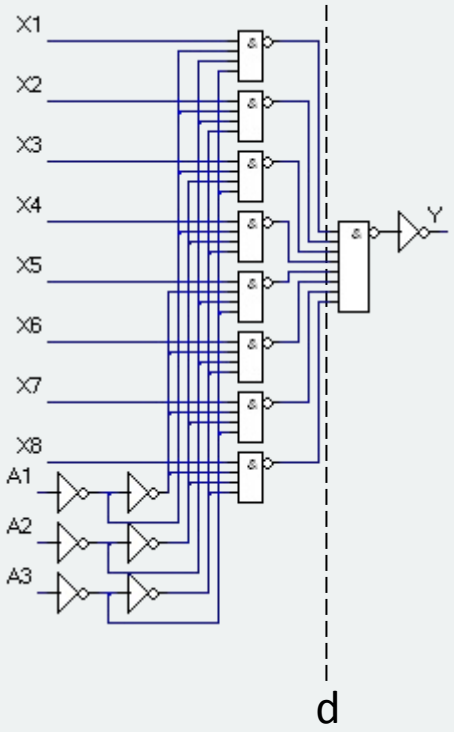
Максимальное значение

```
type DISTANCE is range 0 to 20e+12
units
   nm;
   um = 1e+3 nm;
   mm = 1e+3 um;
   cm = 10 mm;
   dm = 10 cm;
   m  = 10 dm;
   km = 1e+3 m;
end units DISTANCE;
```

Имя минимального юнита

Определение последующих имён юнитов на основе предыдущих

Векторные типы данных




```
   ...
signal d: std_logic_vector(1 to 8);
   ...

begin

   ...
nand4 port map(..., d(1));
nand4 port map(..., d(2));
nand4 port map(..., d(3));
   ...
nand4 port map(..., d(8));

nand8 port map(d(1), d(2), ..., o1);
   ...
```

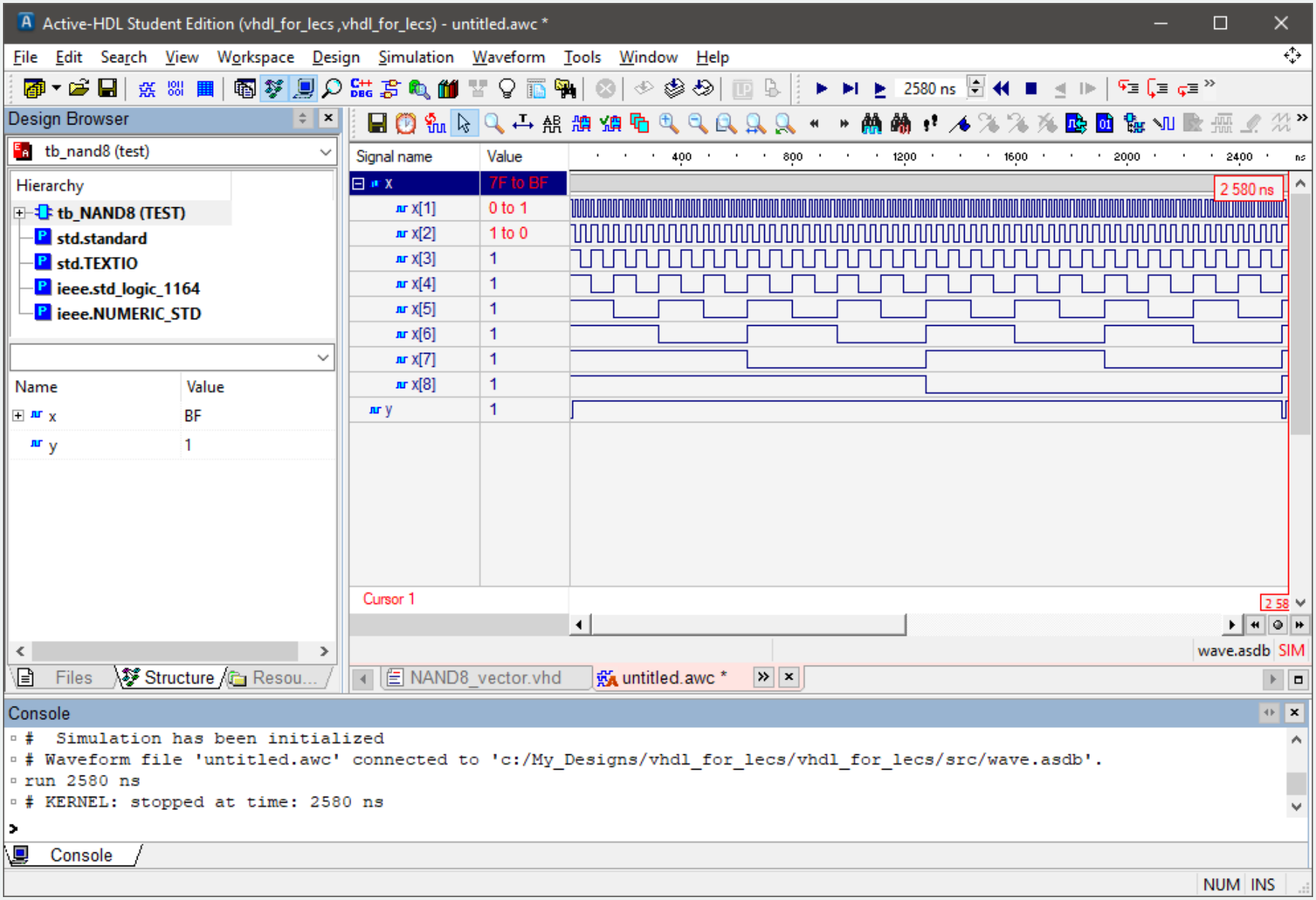
Описание элемента с векторными типами данных

```
entity nand8 is
  port(x: in std_logic_vector(1 to 8);
       y: out std_logic);
end nand8;

architecture RTL of nand8 is
begin
  y <= not(x(1) and x(2) and ... and x(8));
end RTL;
```

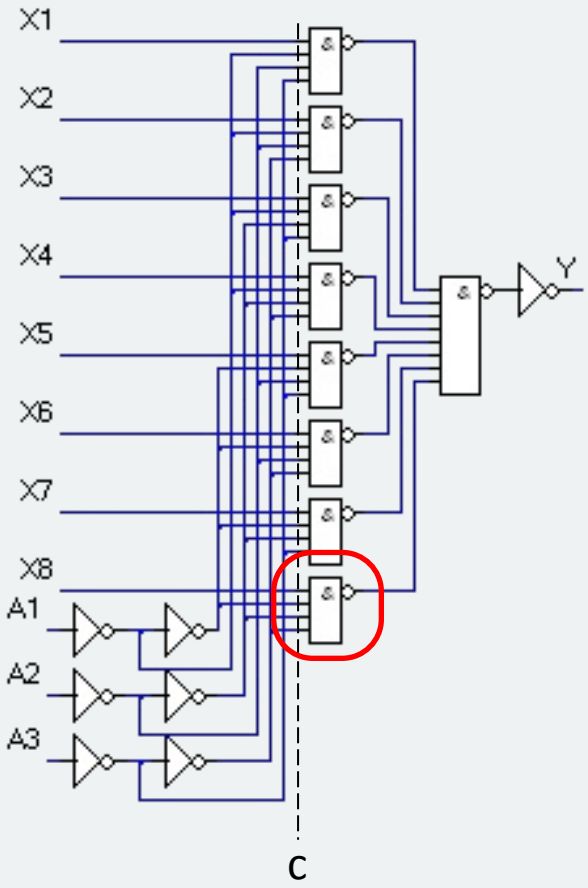
Векторные типы данных в тестовом окружении

```
architecture TEST of tb_NAND8 is

  component NAND8
    port(x: in std_logic_vector(1 to 8);
         y: out std_logic);
  end component;

  signal x: std_logic_vector(1 to 8) := "00000000";
  signal y: std_logic;
begin
  p0 : NAND8 port map(x, y);
  p1 : process
       begin
         x <= std_logic_vector(unsigned(x) + 1);
         wait for 10 ns;
       end process;
end TEST;
```

Векторные типы данных в тестовом окружении (3)

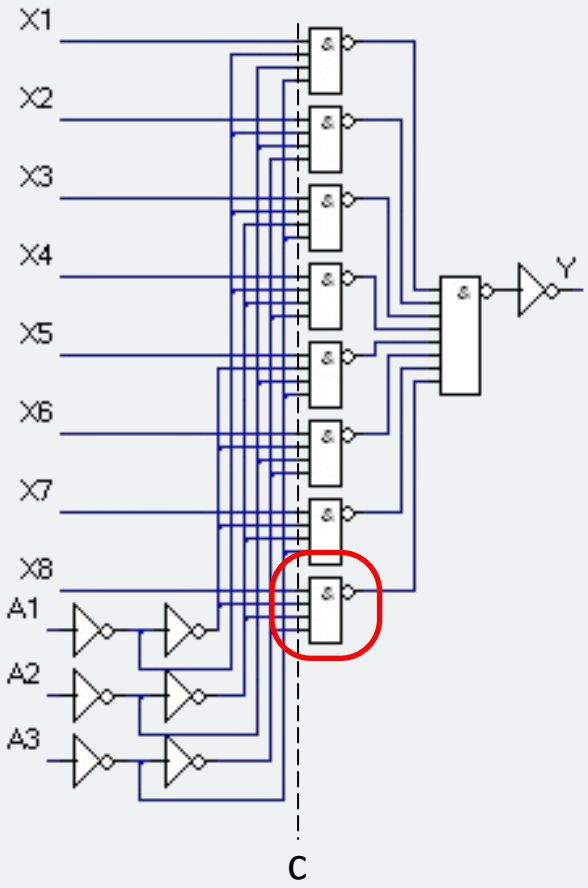


Комбинация вектора из дискретных значений (1)



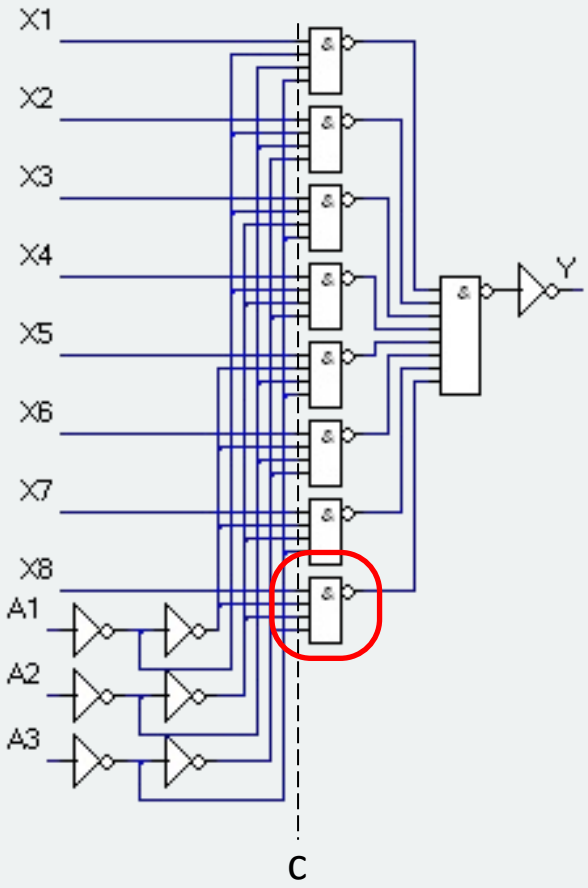

```
signal c2 : std_logic_vector(1 to 4);

begin

   c2(1) <= x8;
   c2(2) <= nnA1;
   c2(3) <= nnA2;
   c2(4) <= nnA3;
   ...

   nand4 port map(c2, d(8));
```

Комбинация вектора из дискретных значений (2)



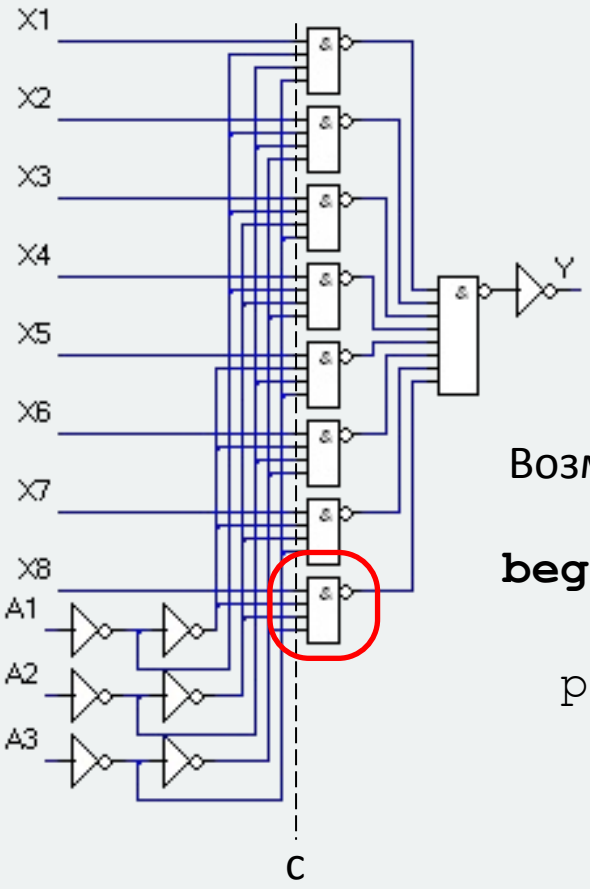

```
signal c2 : std_logic_vector(1 to 4);
begin
  c2 <= x8 & nnA1 & nnA2 & nnA3;
  ...
  nand4 port map(c2, d(8));
```

Комбинация вектора из дискретных значений (3)




```
    signal c2 : std_logic_vector(1 to 4);
begin
    c2 <= (x8, nnA1, nnA2, nnA3);
    ...
    nand4 port map(c2, d(8));
```

Комбинация вектора из дискретных значений (3)



Возможно в вариантах всех стандартов:

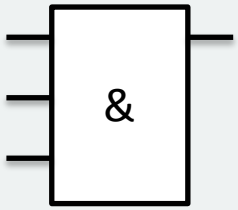
```
begin

  p1 : nand4 port map( x(1) => x8,
                       x(2) => nnA1,
                       x(3) => nnA2,
                       x(4) => nnA3),
                       y    => d(8)
                     );
```

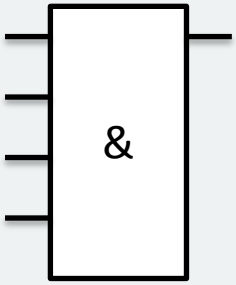
Циклы в языке VHDL

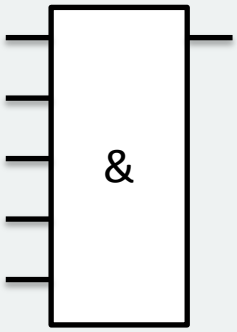
```
entity nand8 is
  port(x: in std_logic_vector(1 to 8);
       y: out std_logic);
end nand8;
```

```
architecture BEH of NAND8 is
begin
  process(x)
    variable c: Integer;
  begin
    c := 0;
    for i in 1 to 8 loop
      if(x(i) = '1') then
        c := c + 1;
      end if;
    end loop;
    if (c = 8) then
      y <= '1';
    else
      y <= '0';
    end if;
  end process;
end BEH;
```

Использование атрибутов массивов и циклов





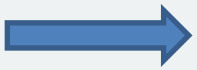



```
entity andN is
  generic(N: Integer := 2);
  port(x: in std_logic_vector(1 to N);
       y: out std_logic);
end andN;
```

Циклы в языке VHDL

```
entity andN is
  generic(N: Integer := 2);
  port(x: in std_logic_vector(1 to N);
       y: out std_logic);
end andN;
```

```
for i in 1 to 8 loop
  if(x(i) = '1') then
    c := c + 1;
  end if;
end loop;
```



```
for i in 1 to N loop
  if(x(i) = '1') then
    c := c + 1;
  end if;
end loop;
```

Атрибуты массивов

Имя атрибута	Назначение атрибута
`A'left`	левая граница массива
`A'right`	правая граница массива
`S'high`	максимальное значение диапазона
`S'low`	минимальное значение диапазона
`S'range`	диапазон массива
`S'length`	длина массива

Циклы в языке VHDL

```
entity andN is
  generic(N: Integer := 2);
  port(x: in std_logic_vector(1 to N);
       y: out std_logic);
end andN;
```

```
for i in 1 to N loop
  if(x(i) = '1') then
    c := c + 1;
  end if;
end loop;
```

```
for i in x'range loop
  if(x(i) = '1') then
    c := c + 1;
  end if;
end loop;
```

Циклы в языке VHDL

```
[<метка>:] loop
             <последовательные операторы>
           end loop [<метка>];
```

```
[<метка>:] for <переменная> in <диапазон> loop
             <последовательные операторы>
           end loop [<метка>];
```

```
[<метка>:] while <условие> loop
             <последовательные операторы>
           end loop [<метка>];
```

Циклы в языке VHDL: exit и next

```
exit [<метка>] [ when <условие> ];
next [<метка>] [ when <условие> ];
```

```
loop
  if x(I) = '1' then
    C := C + 1;
  end if;
  I := I + 1;
  if (I > N) then
    exit;
  end if;
end loop;
```

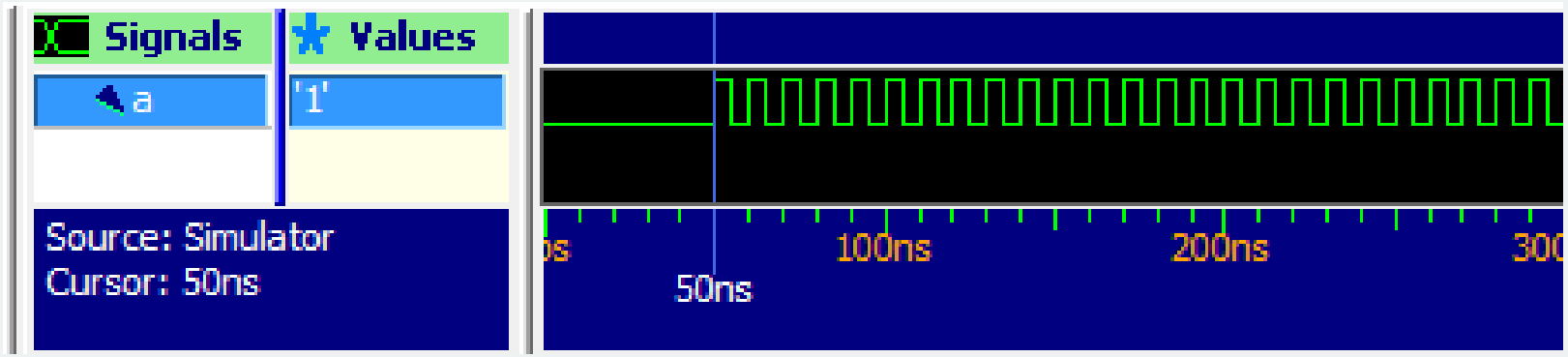
```
loop
  if x(I) = '1' then
    C := C + 1;
  end if;
  I := I + 1;
  exit when I > N;
end loop;
```

```
L1: loop
    ...
L2:   loop
        ...
        exit when C < D and E /= F;
        exit L1 when A > B;
      end loop;
    end loop;
```

Циклы в языке VHDL: пример использования loop (1)

Реализация генерации периодического сигнала
начиная с некоторого момента времени

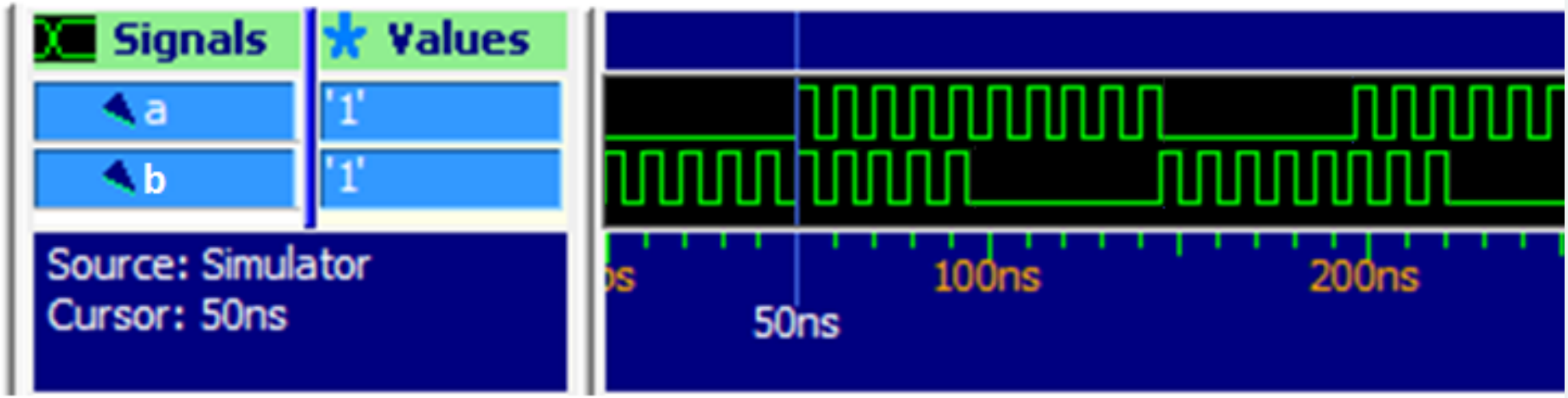
```
p1 : process
     begin
       wait for 50 ns;
       loop
         a <= not a;
         wait for 5 ns;
       end loop;
     end process;
```



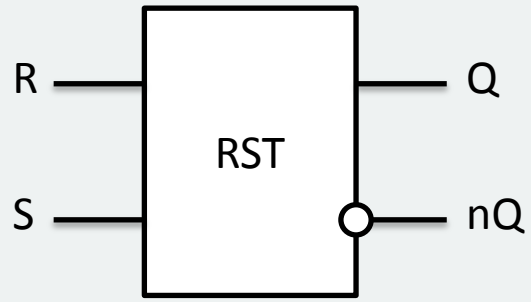
Циклы в языке VHDL: пример использования loop (2)

```
p1 : process
     begin
       wait for 50 ns;
       for i in 1 to 20 loop
         a <= not a;
         wait for 5 ns;
       end loop;
     end process;
```

```
p2 : process
     begin
       for i in 1 to 20 loop
         b <= not b;
         wait for 5 ns;
       end loop;
       wait for 50 ns;
     end process;
```



RS триггер на операторе CASE



R	S	Q	nQ
1	0	0	1
0	1	1	0
0	0	хранение	
1	1	запрещено	

```
process(R, S)
  variable RS: std_logic_vector(1 to 2);
begin
  RS := R & S;
  case RS is
    when "10"   =>  Q <= '0';
                   nQ <= '1';
    when "01"   =>  Q <= '1';
                   nQ <= '0';
    when "11"   =>  Q <= 'X';
                   nQ <= 'X';
    when others => NULL;
  end case;
end process;
```

Оператор REPORT

```
process
begin
   x <= not x;
   report "X-value changed!";
   wait for 10 ns;
end process;
```

```
REPORT: NOTE at 0 ps+0: X-value changed!
    At test.vhdl: (line 55)
        Instance = :tb_test(test):
REPORT: NOTE at 10 ns+0: X-value changed!
    At test.vhdl: (line 55)
        Instance = :tb_test(test):
REPORT: NOTE at 20 ns+0: X-value changed!
    At test.vhdl: (line 55)
        Instance = :tb_test(test):
```

Оператор REPORT: конкатенация строк

```
process
begin
  x <= not x;
  report "X changed at " & time'image(now) &
         " new value is " & std_logic'image(x);
  wait for 10 ns;
end process;
```

```
REPORT: NOTE at 0 ps+0: X changed at 0 ps new value is '0'
    At test.vhdl: (line 55)
        Instance = :tb_test(test):
REPORT: NOTE at 10 ns+0: X changed at 10000 ps new value is '1'
    At test.vhdl: (line 55)
        Instance = :tb_test(test):
REPORT: NOTE at 20 ns+0: X changed at 20000 ps new value is '0'
    At test.vhdl: (line 55)
        Instance = :tb_test(test):
```

Оператор REPORT с указанием степени важности (1)

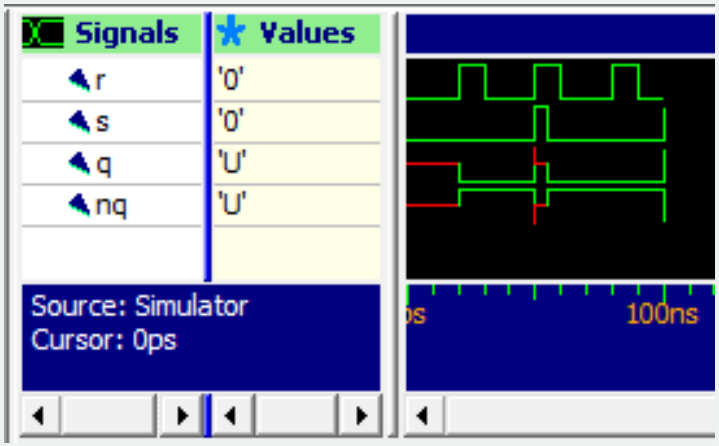
```
report <messagestring> [severity <severity_level>];
```

```
TYPE severity_level IS (note,     -- выводит сообщение
                        warning,  -- выводит предупреждение
                        error,    -- выводит ошибку
                        failure); -- выводит ошибку и
                                  -- завершает моделирования
```

Оператор REPORT с указанием степени важности (2)

```
process (R, S)
begin
  ...
  elsif(R = '1' and S = '1') then
    report "Bad combination at " & time'image(now)
    severity error;
  ...
end process;
```

```
REPORT: ERROR at 50 ns+1: Bad combination at 50000 ps
   At rst.vhdl: (line 14)
      Instance = :tb_rst(test):p0@rst(beh):
```



Оператор ASSERT

```
process(R, S)
begin
  if(R = '1' and S = '1') then
    report "Bad!" severity error;
  end if;
  ...
end process;
```

```
process (R, S)
begin
  assert (R /= '1' or S /= '1') report "Bad!" severity error;

  ...
end process;
```

Вывод текстовой информации

```
REPORT: ERROR at 50 ns+1: Bad combination at 50000 ps
   At rst.vhdl: (line 14)
      Instance = :tb_rst(test):p0@rst(beh):
```

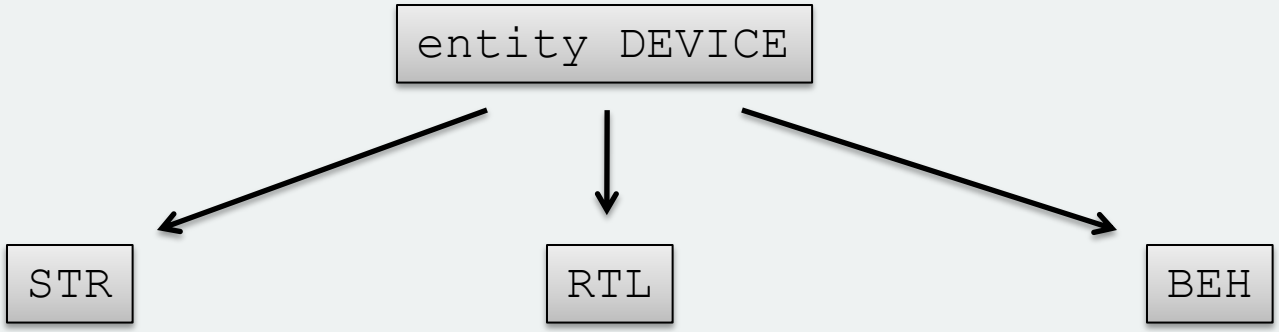
```
use std.textio.all;
...

process(R, S)
  variable ln: line;
begin
  if(R = '1' and S = '1') then
    write(ln, "Bad combination on RS trigger! ");
    write(ln, "Time=");
    write(ln, now);
    writeline(output, ln);
  end if;
  ...
end process;
```

```
Bad combination on RS trigger! Time=50 ns
```

Файловый вывод текстовой информации

```
use std.textio.all;
...

process(R, S)
  variable ln: line;
  file     tf: TEXT open write_mode is "report.log";
begin
  if(R = '1' and S = '1') then
    write(ln, "Bad combination on RS trigger! ");
    write(ln, "Time=");
    write(ln, now);
    writeline(output, ln);
    writeline(tf, ln);
  end if;
  ...
end process;
```

Выбор архитектурного тела для моделирования




```
entity tb_device is
end tb_device;
```

```
architecture TEST of tb_device is
  component DEVICE
  ...
begin

  p1 : DEVICE port map ...

end TEST;
```

Конфигурация (1)

```
entity tb_device is
end tb_device;
```

```
architecture TEST of tb_device is
  component DEVICE
  ...
begin

  p1 : DEVICE port map ...

end TEST;
```

```
configuration config1 of tb_device is
  for test
    for p1 : DEVICE use entity work.DEVICE(STR);
    end for;
  end for;
end configuration;
```

Конфигурация (2)

```
entity tb_device is
end tb_device;
```

```
architecture TEST of tb_device is
  component DEVICE
  ...
begin
  p1 : DEVICE port map ...
  p2 : DEVICE port map ...
  p3 : DEVICE port map ...
  ...
end TEST;
```

```
configuration config1 of tb_device is
  for test
    for p1 : DEVICE use entity work.DEVICE(STR); end for;
    for p2 : DEVICE use entity work.DEVICE(RTL); end for;
    for p3 : DEVICE use entity work.DEVICE(BEH); end for;
   end for;
end configuration;
```