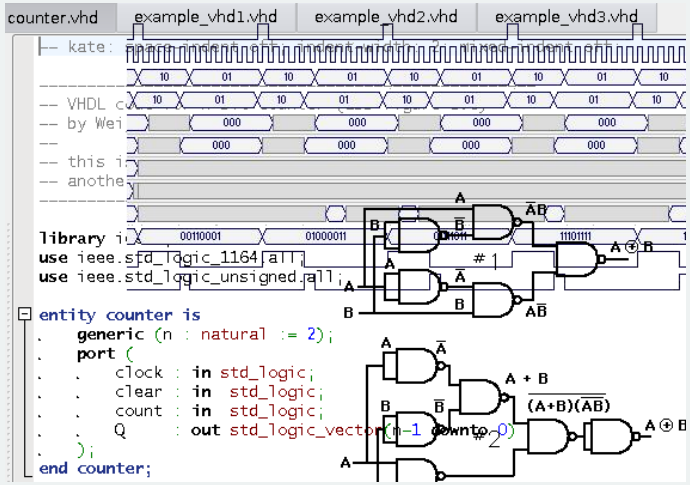
МИЭТ
Национальный исследовательский университет «МИЭТ»
Кафедра ПКИМС

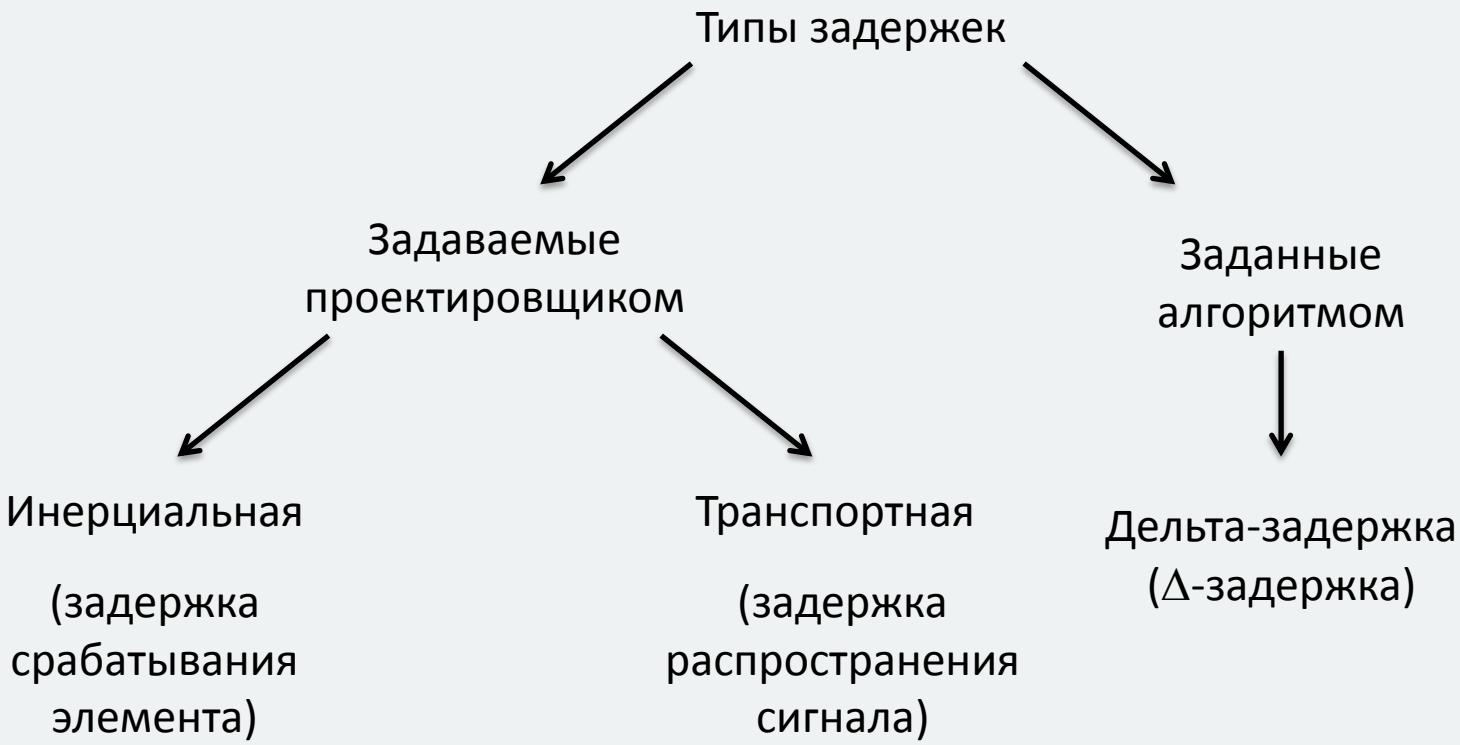
Лингвистические средства проектирования



Лекция 5

Параллельные операторы языка VHDL

Задержки в языке VHDL



Полная форма записи оператора назначения сигнала

```
<имя_сигнала> <= [ reject <время> ]
                 [ transport | inertial ] <значение>
                 [ after <время>, ... ];
```

```
architecture RTL of inv is
begin
  y <= not x;
end RTL;
```

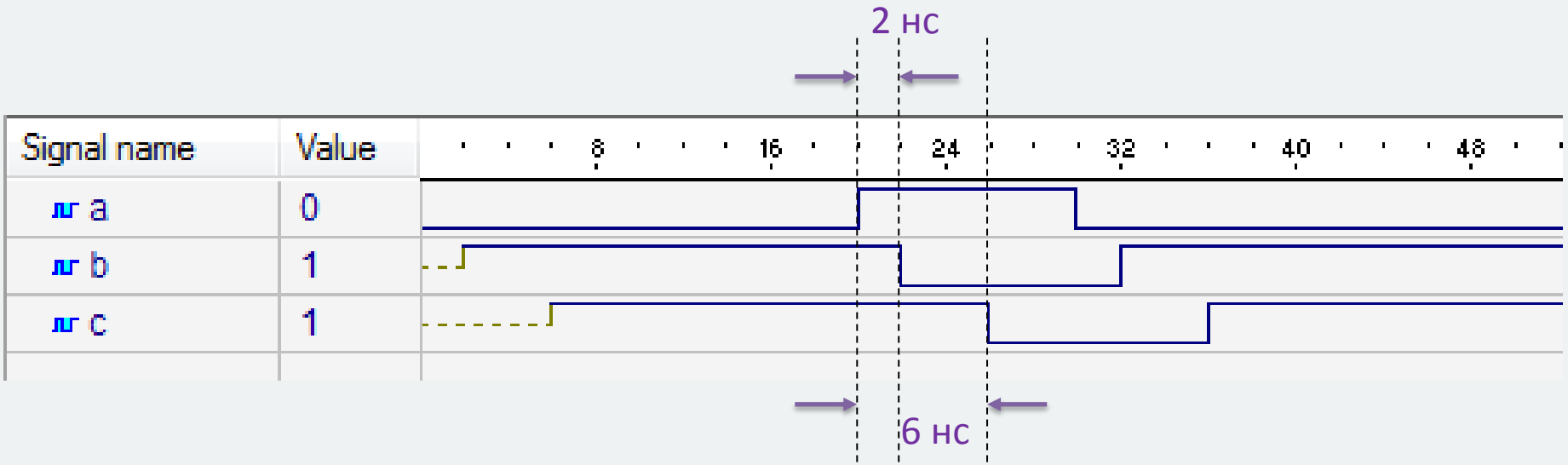
```
architecture RTL of inv is
begin
  y <= transport not x after 5 ns;
end RTL;
```

```
architecture RTL of inv is
begin
  y <= inertial not x after 5 ns;
end RTL;
```

```
architecture RTL of big_inv is
begin
  y <= reject 5 ns inertial not x after 10 ns;
end RTL;
```

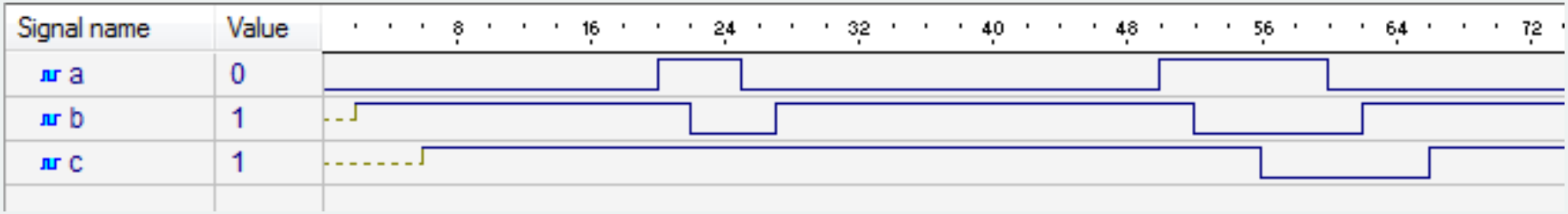
Транспортная задержка в VHDL

```
entity inv is
  generic(t: time := 2 ns);
  port(x: in bit; y: out bit);
end inv;

architecture RTL of inv is
begin
  y <= transport not x after t;
end RTL;

...

p1 : inv generic map (2 ns) port map(a, b);
p2 : inv generic map (6 ns) port map(a, c);
```

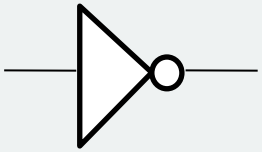


Инерциальная задержка в VHDL

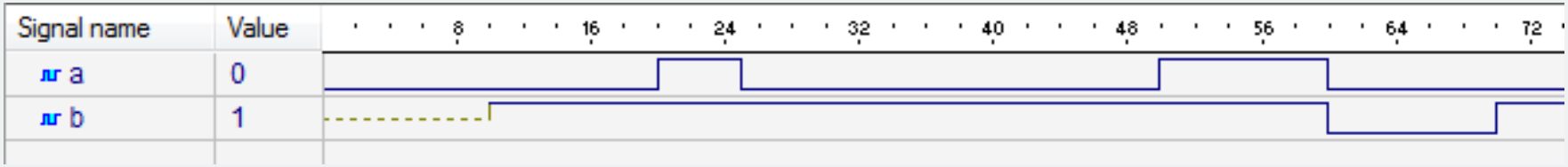
```
entity inv is
  generic(t: time := 2 ns);
  port(x: in bit; y: out bit);
end inv;

architecture RTL of inv is
begin
  y <= inertial not x after t;
end RTL;

...

p1 : inv generic map (2 ns) port map(a, b);
p2 : inv generic map (6 ns) port map(a, c);
p3 : a <= '0' after 0 ns,
          '1' after 20 ns,
          '0' after 25 ns,
          '1' after 50 ns,
          '0' after 60 ns;
```



Инерциальная задержка с reject в VHDL



```
entity inv is
  port(x: in bit; y: out bit);
end inv;

architecture RTL of inv is
begin
  y <= reject 5 ns inertial not x after 10 ns;
end RTL;
```



Оператор условного назначения сигнала (1)

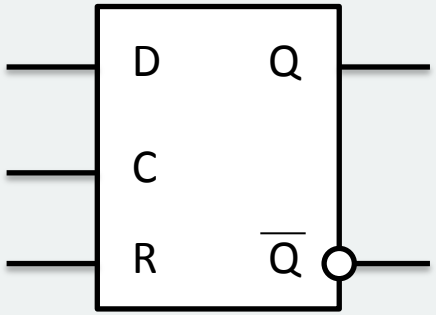
```
<имя_сигнала> <= <значение_1> when <условие_1> else
                 <значение_2> when <условие_2> else
                 ...
                 <значение_N-1> when <условие_N-1> else
                 <значение_N>;
```

```
architecture BEH of XOR2 is
begin
  process(x1, x2)
  begin
    if(x1 = '0' and x2 = '0') then
      y <= '0';
    elsif(x1 = '0' and x2 = '1') then
      y <= '1';
    elsif(x1 = '1' and x2 = '0') then
      y <= '1';
    else
      y <= '0';
    end if;
  end process;
end BEH;
```

```
architecture BEH of XOR2 is
begin
  y <= '0' when (x1 = '0' and x2 = '0') else
       '1' when (x1 = '1' and x2 = '0') else
       '1' when (x1 = '0' and x2 = '1') else
       '0';
end BEH;
```

Оператор условного назначения сигнала (2)

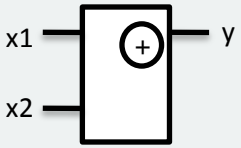
```
process(C, R)
  variable temp : std_logic;
begin
  if(R = '1') then
    temp := '0';
  elsif(C = '0' and not C'stable) then
    temp := D;
  end if;
  Q <= temp;
  nQ <= not temp;
end process;
```




```
architecture BEH of DCRTT is
begin
  Q <= '0' when (R = '1') else
       D when (C = '0' and not C'stable) else
       unaffected;

  nQ <= '1' when (R = '1') else
        not D when (C = '0' and not C'stable) else
        unaffected;
end BEH;
```

Оператор выборочного назначения сигнала

```
with <выражение> select
  <имя_сигнала> <= <значение_1> when <вариант_1>,
                   <значение_2> when <вариант_2>,
                   ...
                   <значение_N-1> when <вариант_N-1>,
                   <значение_N>;
```



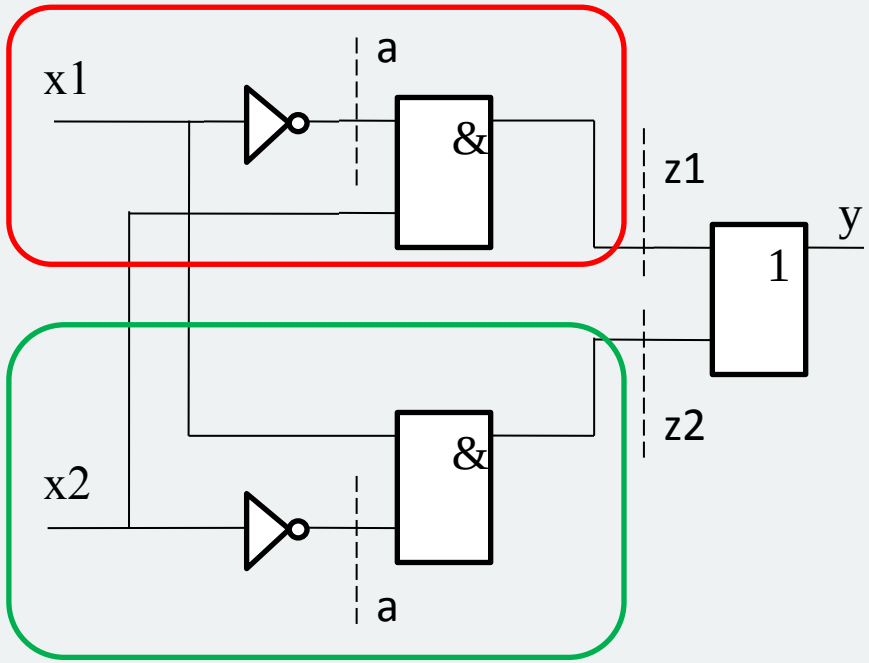

```
process(x)
begin
  case x is
    when "00" y <= '0';
    when "01" y <= '1';
    when "10" y <= '1';
    when "11" y <= '0';
  end case;
end process;
```

```
architecture BEH of xor2 is
begin
  with X select
    y <= '0' when "00",
         '1' when "01",
         '1' when "10",
         '0' when "11";
end BEH;
```

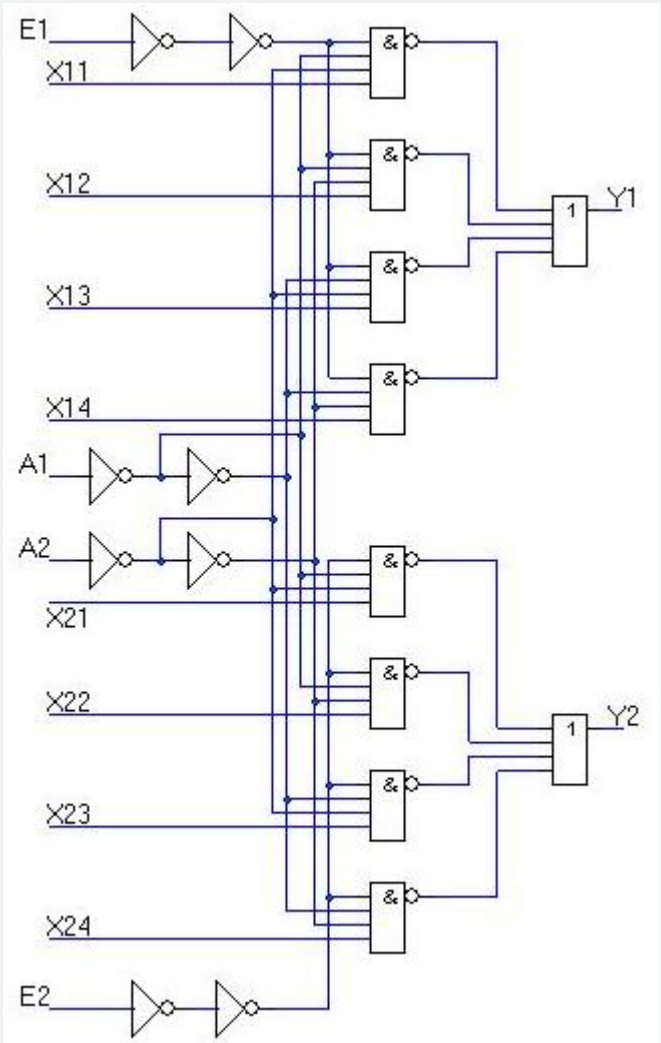
Оператор объявления блока параллельных конструкций (2)

```
architecture RTL of XOR2 is
  signal z1, z2: std_logic;
begin

  B1 : block
         signal a: std_logic;
       begin
         a <= not X1;
         z1 <= a and X2;
       end block B1;

  B2 : block
         signal a: std_logic;
       begin
         a <= not X2;
         z2 <= X1 and a;
       end block B2;

  Y <= z1 or z2;

end BEH;
```

```
<метка>: block
          [ объявления для блока ]
         begin
           <параллельные операторы>
         end block [ <метка> ];
```

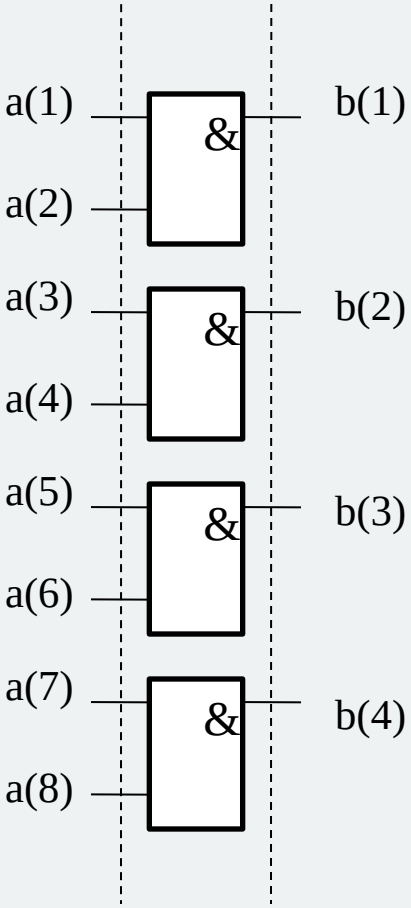


Защищённые блоки параллельных конструкций




```
architecture RTL of DEVICE is
  signal ...
begin
  BLOCK1 : block(E1 = '1')
           signal ...
           begin
             ...
             Y1 <= guarded ...;
           end block BLOCK1;

  BLOCK2 : block(E1 = '0')
           begin
             Y1 <= guarded '1';
           end block BLOCK2;

  BLOCK3 : block(E2 = '1')
           signal ...
           begin
             ...
             Y2 <= guarded ...;
           end block BLOCK3;

  BLOCK4 : block(E2 = '0')
           Y2 <= guarded '1';
           end block BLOCK2;
end STR;
```

Оператор генерации подстановки компонента (1)




```
architecture STR of DEVICE is
  component and2
    port(x1, x2: in bit;
                 y: out bit);
  end component;
  ...
  signal a: bit_vector(1 to 8);
  signal b: bit_vector(1 to 4);

begin

g1: for i in 1 to 4 generate

      p1: and2 port map(a(i*2-1), a(i*2), b(i));

  end generate;

end STR;
```