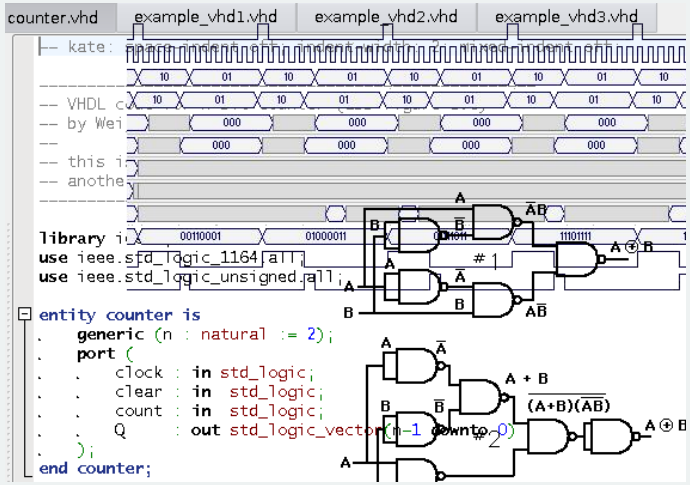
МИЭТ
Национальный исследовательский университет «МИЭТ»
Кафедра ПКИМС

Лингвистические средства проектирования



Лекция 3

Поведенческое архитектурное тело

Процессы – строительные блоки поведенческого описания

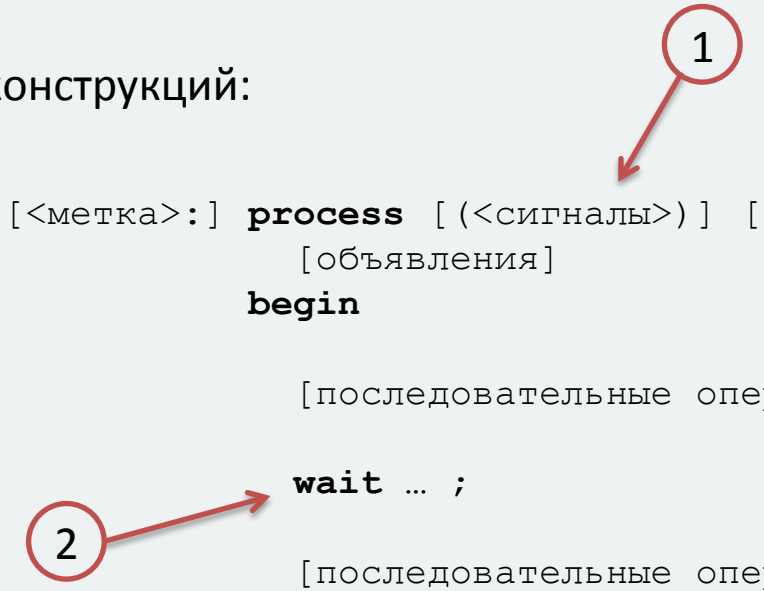
```
[<метка>:] process [(<сигналы>)] [is]
   [объявления]
begin
   [последовательные операторы]
end [process] [<метка>];
```

```
architecture BEH of DEVICE is
…
begin

  p1 : process(x1, a1)
       begin
         …
       end process;

  p2 : process(x2, a2, c)
       begin
         …
       end process;
  …

end BEH;
```

Формы описания процессов

В процессе обязана быть одна из двух конструкций:

1. список чувствительности;
2. оператор ожидания.

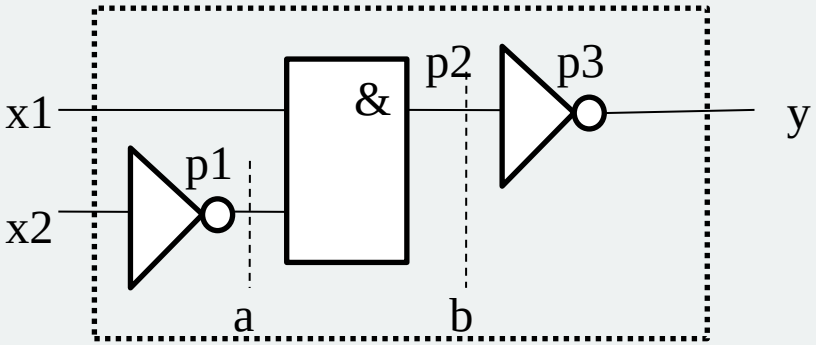


Без списка чувствительности процесс работает **бесконечно** переходя от одной итерации к другой.

Если процесс не содержит ни списка чувствительности, ни оператора ожидания, он **входит в зацикливание**, **моделирование становится невозможным**.

Поведенческое архитектурное тело

x1	x2	y
0	0	1
0	1	1
1	0	0
1	1	1




```
entity DEVICE is
  port(x1: in bit;
       x2: in bit;
        y: out bit);
end DEVICE;
```

```
architecture BEH of DEVICE is
begin

  process(x1, x2)
  begin
    if(x1='1' and x2='0') then
      y <= '0';
    else
      y <= '1';
    end if;
  end process;

end BEH;
```

Формы записи оператора wait

wait – оператор ожидания.

4 формы записи оператора:

1. ожидание в течение определённого времени;

```
wait for <время>;
```

2. ожидание изменения сигнала;

```
wait on <список_сигналов>;
```

3. ожидание выполнения логического условия,

```
wait until <логическое_условие>;
```

4. остановка процесса.

```
wait;
```

Ожидание в течение заданного времени


```
p1: x <= '1' after 0 ns, '0' after 20 ns, '1' after 40 ns,...
```

```
p1 : process
     begin
       x <= '1';
       wait for 20 ns;
       x <= '0';
       wait for 20 ns;
     end process;
```

```
p1 : process
     begin
       x <= not x;
       wait for 20 ns;
     end process;
```

Правильное использование начальных значений сигналов

```
architecture TEST of tb_dctt is
  component dctt
    port(d, c: in std_logic;
         q, nq: out std_logic);
  end component;

  signal d, c: std_logic := '0';
  signal q, nq: std_logic;

begin
  p1 : dctt port map(d, c, q, nq);

  p2 : process
       begin
         c <= not c;
         wait for 7ns;
       end process;
  p3 : process
       begin
         d <= not d;
         wait for 19ns;
       end process;
end BEH;
```

Покрытие набора входных воздействий

AND2

x2	x1	y
0	0	0
0	1	0
1	0	0
1	1	1

```
p1 : process
     begin
       wait for 10 ns;
       x1 <= not x1;
     end process;

p2 : process
     begin
       wait for 20 ns;
       x2 <= not x2;
     end process;
```

```
p1 : process
     begin
       wait for 10 ns;
       x1 <= not x1;
     end process;

p2 : process
     begin
       wait on x1;
       wait on x1;
       x2 <= not x2;
     end process;
```

Ожидание логического условия

AND2

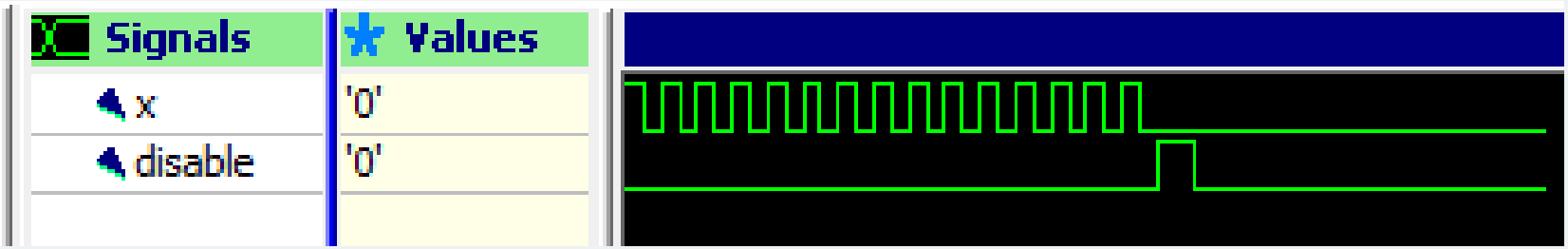
x2	x1	y
0	0	0
0	1	0
1	0	0
1	1	1

```
p1 : process
     begin
       wait for 10 ns;
       x1 <= not x1;
     end process;

p2 : process
     begin
       wait for 20 ns;
       x2 <= not x2;
     end process;
```

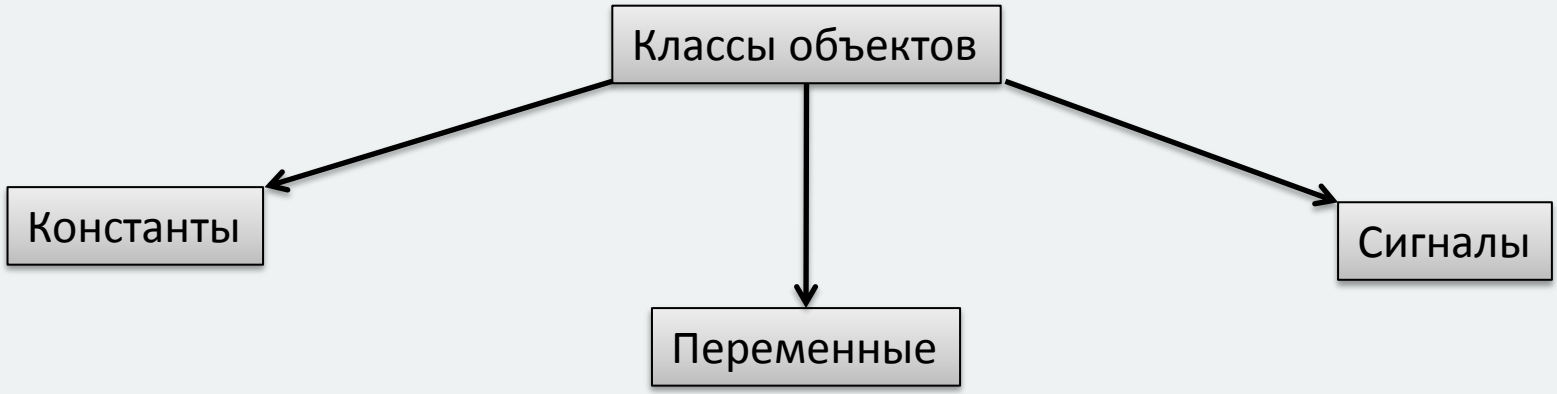
```
p1 : process
     begin
       wait for 10 ns;
       x1 <= not x1;
     end process;

p2 : process
     begin
       wait until x1 = '0';
       x2 <= not x2;
     end process;
```

Остановка процесса

```
p1 : disable <= '0', '1' after 150 ns, '0' after 160 ns;

p2 : process
     begin
       x <= not x;
       wait for 5 ns;
       if(disable = '1') then
         wait;
       end if;
     end process;
```



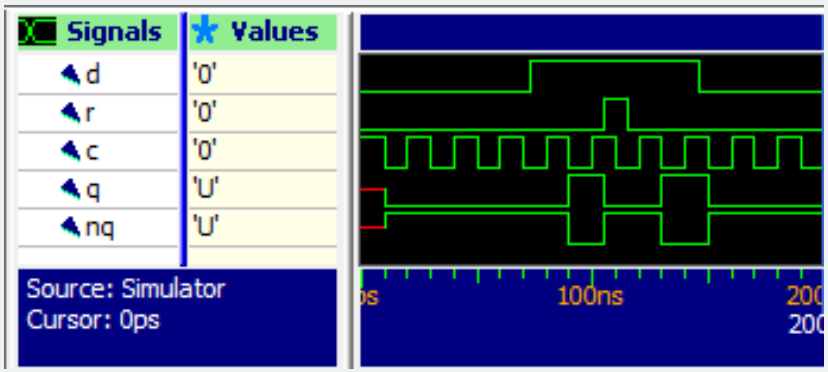
Константы, переменные, сигналы




```
constant LOGIC_ONE : std_logic := '1';

variable TEMP_VAR  : std_logic := '1';

signal   CONNECT   : std_logic := '1';
```

Пример использования переменных

```
entity DCTT is
  port(D, C, R: in std_logic;
       Q, nQ  : out std_logic);
end DCTT;

architecture BEH of DCTT is
begin
 p1: process(C, R)
       variable temp : std_logic;
     begin
       if(R = '1') then
         temp := '0';
       else
         if(C = '0' and not C'stable) then
           temp := D;
         end if;
       end if;
       Q <= temp;
       nQ <= not temp;
     end process;
end BEH;
```



Атрибуты сигналов

Имя атрибута	Назначение атрибута, что возвращает
`S'stable, S'stable(T)`	true, если сигнал не менялся (в течение времени T)
`S'event`	true, если происходит изменение сигнала
`S'active`	true, если была запись, но значение ещё не поменялось
`S'quiet(T)`	true, если не было обращений в течение времени T
`S'last_value`	предыдущее значение сигнала
`S'last_event`	время предыдущего изменения сигнала

Определение фронта сигнала

```
process (C)
begin
  if (not C'stable and C='1') then
    ...
```

```
process (C)
begin
  if (C'event and C='1') then
    ...
```

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

process (C)
begin
  if rising_edge(C) then
    ...
```

Условный оператор if

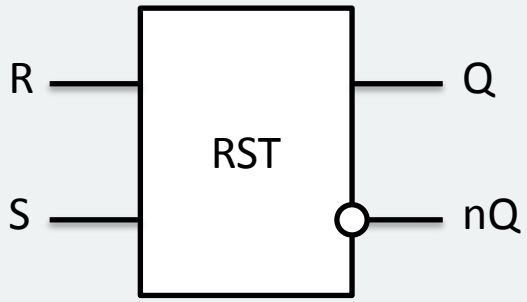
```
process(x1, x2)
begin
  if (x1 = '0' and x2 = '0') then
    y <= '0';
  elsif (x1 = '0' and x2 = '1') then
    y <= '1';
  elsif(...
        ...
  else
        ...
  end if;
end process;
```

XOR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

```
process(x1, x2)
begin
  if (x1 /= x2) then
    y <= '1';
  else
    y <= '0';
  end if;
end process;
```

Оператор множественного выбора case



Тип данных: STD_LOGIC

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';
  end case;
end process;
```

Оператор NULL

Error: CSVHDL0139: test.vhdl: (line 14):
The number of choices for type ''std_logic_vector(1 TO 2)'' must be '81'.
Only '3' choice(s) given. Hint. Use OTHERS to specifiy missing choice(s)

```
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;
```