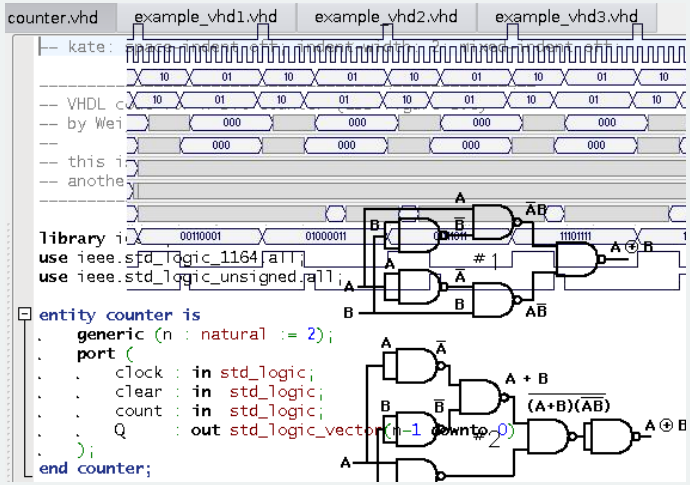
МИЭТ
Национальный исследовательский университет «МИЭТ»
Кафедра ПКИМС

Лингвистические средства проектирования



Лекция 6

Сравнение HDL языков

Что мы будем сравнивать?

VHDL – язык описания цифровой аппаратуры, разработан по заказу министерства обороны США.
1987 – первый стандарт языка
1993 – последний стандарт, полностью поддерживаемый различными САПР

Verilog HDL – язык описания цифровой аппаратуры, разработан компанией Gateway Design Automation, в 1990 году они были куплены Cadence Design Systems.
1995 – первый стандарт языка Verilog HDL
2001 – самый распространённый стандарт

SystemC – библиотека классов и макросов на языке C++
2005 – принят стандарт языка (библиотеки)

Описание устройств

Пример кода для VHDL:

```
-- This is an
-- inverter module

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

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

Пример кода для Verilog HDL:

```
/* This is an
inverter module */

module inv(x, y);
  input x;
  output y;

  // Assignment
  assign y = ~x;

endmodule
```

Как может описываться поведение устройства?

```
module inv(x, y);
  input x;
  output y;

  assign y = ~x;
endmodule
```

```
module inv(x, y);
  input x;
  output y;

  not(y, x);

endmodule
```

```
module inv(x, y);
  input x;
  output y;

  always @(x)
    if(x == 1)
      y = 0;
    else
      y = 1;

endmodule
```

Описание секции начальной инициализации

```
module tb_device_inv();

  reg  x;
  wire y;

  inv u1(x, y);

  initial
  begin
    #0  x = 0;

    #10 x = 1;
    #10 x = 0;
  end

endmodule
```

Встроенные типы данных

Пример типов данных для VHDL:

Скалярные:

```
NATURAL, POSITIVE,
INTEGER, REAL
```

Перечислимые:

```
BIT, CHARACTER, BOOLEAN
```

Векторные:

```
BIT_VECTOR, STRING
```

Физические:

```
TIME
```

Пример типов данных для Verilog HDL:

Скалярные:

```
integer, time, real
```

Значения:

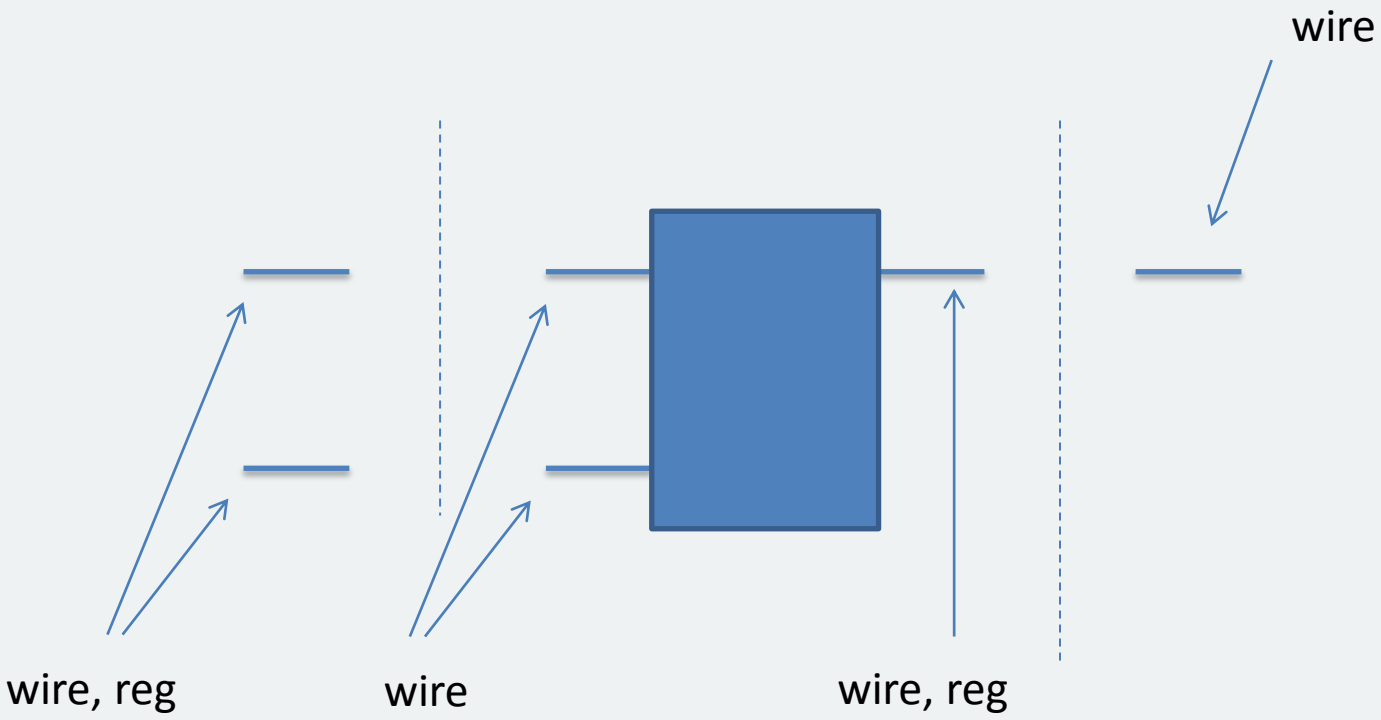
```
0
1
x
z
```

Данные в языке Verilog HDL

В Verilog HDL данные - либо цепи, либо переменные (аналогично сигналам и переменным в VHDL).

Цепи: **`wire`** – только комбинационные схемы

Переменные: **`reg`** – комбинационные и последовательностные схемы



Объявление и запись данных в языке Verilog HDL

Способы организации данных:

1. одно значение (value);
2. массив (array);
3. вектор (vector), шина (bus);
4. память (memory).

```
wire A;
integer data[3:0];
reg [0:7] byte;
reg [7:0] memory_512 [0:511];
```

```
reg [0:7] byte;
reg data [0:7];

byte = 8'b11_00_11_01
data = 8'b11_00_11_01
```

Способы задания значений:

```
0

1'b0

3'b10x

7'h7F

8'b11_00_11_01

32'd120

8'o2Z
```

Операторы для описания типа RTL

NOT : ~

x	y
0	1
1	0

AND : &

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

OR |

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

XOR : ^

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

NAND: ~&

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

NOR : ~|

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

Описание типа RTL

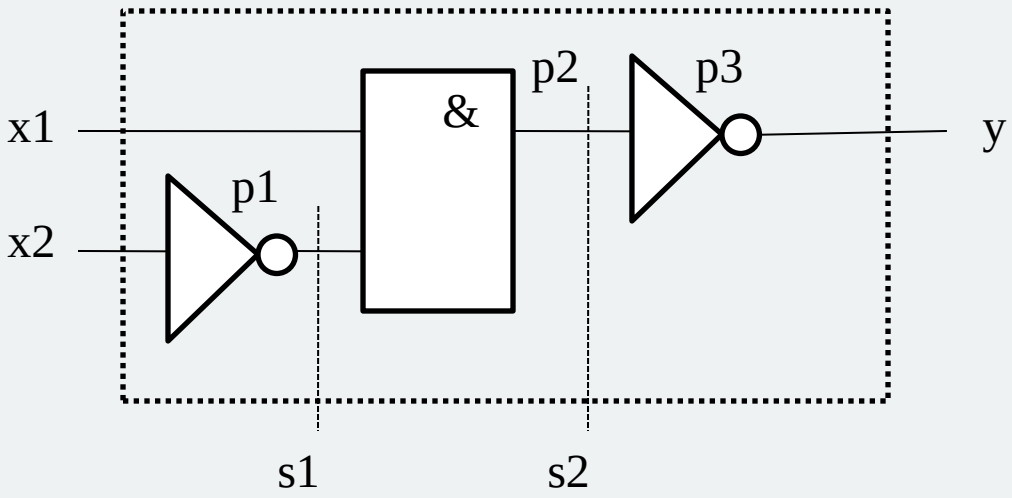
```
module inv(x, y);
  input x;
  output y;

  assign y = ~x;
endmodule
```

```
module inv(x, y);
  input x;
  output y;

  not(y, x);

endmodule
```

```
module DC(i1, i2, o1, o2, o3, o4);
  input  i1, i2;
  output o1, o2, o3, o4;

  assign o1 = ~i1 & ~i2;
  assign o2 = ~i1 &  i2;
  assign o3 =  i1 & ~i2;
  assign o4 =  i1 &  i2;
endmodule
```

Описание библиотеки элементов

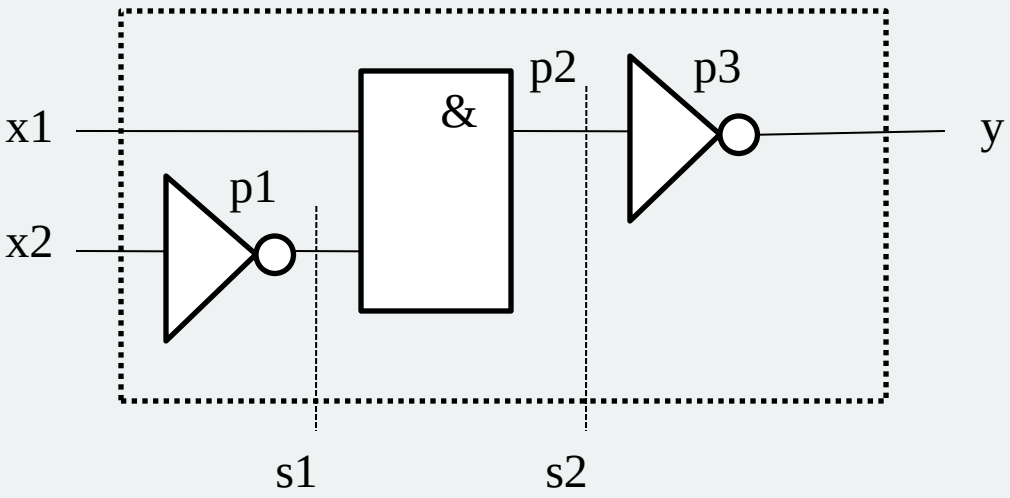
```
module inv(x, y);
  input  x;
  output y;

  assign y = ~x;
endmodule
```




```
module and2(x1, x2, y);
  input  x1, x2;
  output y;

  assign y = x1 & x2;
endmodule
```

Структурное описание схемы (1)

```
module device(x1, x2, y);
  input  x1, x2;
  output y;

  wire s1, s2;

  inv  i1(x2, s1);
  and2 a1(x1, s1, s2);
  inv  i2(s2, y);

endmodule
```

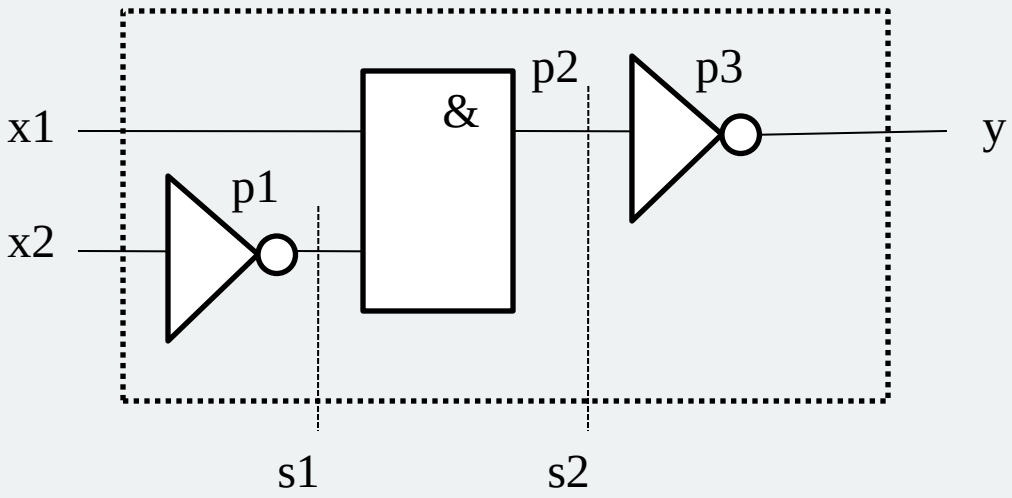



```
module device(x1, x2, y);
  input  x1, x2;
  output y;

  wire s1, s2;

  not i1(s1, x2);
  and a1(s2, x1, s1);
  not i2(y, s2);

endmodule
```

```
not(s1, x2);
and(s2, x1, s1);
not(y, s2);
```

Структурное описание схемы (2)

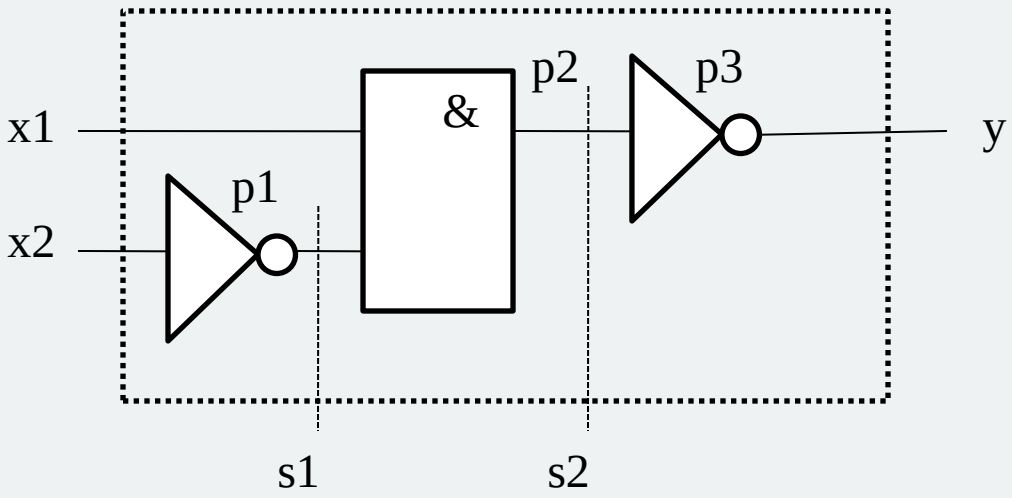
```
module device(x1, x2, y);
  input x1, x2;
  output y;

  wire s1, s2;

  inv  i1(x2, s1);
  and2 a1(x1, s1, s2);
  inv  i2(s2, y);

endmodule
```




```
module device(x1, x2, y);
  input x1, x2;
  output y;

  wire s1, s2;

  inv  i1(.x(x2), .y(s1));
  and2 a1(.x1(x1), .x2(s1), .y(s2));
  inv  i2(.x(s2), .y(y));

endmodule
```

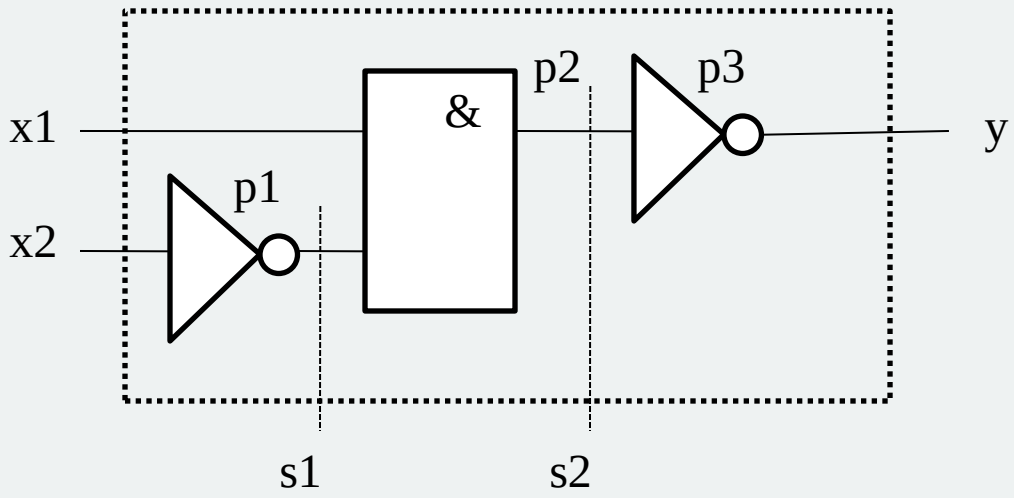
Регистровое описание схемы

```
module device(x1, x2, y);
  input x1, x2;
  output y;

  wire s1, s2;

  not(s1, x2);
  and(s2, x1, s1);
  not(y, s2);

endmodule
```




```
module device(x1, x2, y);
  input x1, x2;
  output y;

  assign y = ~(x1 & ~x2);

endmodule
```

Поведенческое описание схемы

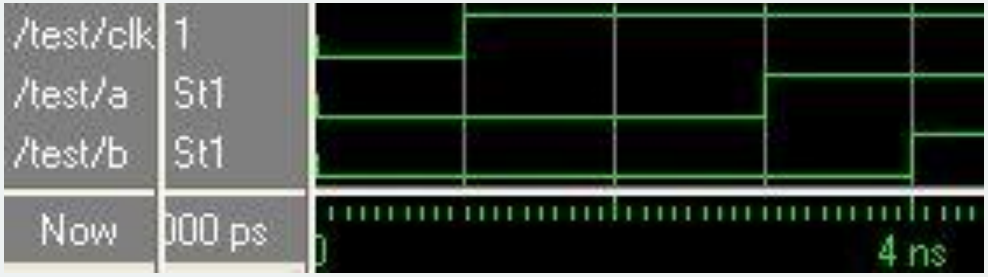



```
module device(x1, x2, y);
  input x1, x2;
  output y;

  wire s1, s2;

  always @ (x1 or x2)
  begin
    if(x1 == 1'b1 && x2 == 1'b0)
      y = 1'b0;
    else
      y = 1'b1;
  end

endmodule
```

Пример на fork и join

```
initial
begin
   a = 0;
   b = 0;
end
```


```
always @(posedge clk)
begin
   #2 a = 1;
   #1 b = a;
end
```




```
initial
begin
   a = 0;
   b = 0;
end
```

```
always @(posedge clk)
fork
   #2 a = 1;
   #1 b = a;
join
```

Функциональное и временное моделирование

Пример кода для VHDL:

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

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

Пример кода для Verilog HDL:

```
module inv(x, y);
  input x;
  output y;

  assign #5 y = ~x;

endmodule
```

Блокирующее и неблокирующее присваивание (1)

```
module block_nonblock();
  reg a, b, c, d , e, f ;

  // Блокирующие присваивания
  initial
  begin
    a = #10 1'b1;                    →   10
    b = #20 1'b0;                    →   30
    c = #40 1'b1;                    →   70
  end

  // Неблокирующие присваивания
  initial
  begin
    d <= #10 1'b1;                   →   10
    e <= #20 1'b0;                   →   20
    f <= #40 1'b1;                   →   40
  end

endmodule
```

Задержки и времена переключений: инерциальные

Способы описания инерциальных задержек

```
and (y_out, x1, x2);

and #3 (y_out, x1, x2);

and #(2, 3) G1(y_out, x1, x2);

and #(2, 3) G1 (y_out, x1, x2), G2 (y_out2, x3, x4);

and #(2:3:4, 4:5:6) G1(y_out, x1, x2);
```

Задержки и времена переключений: транспортные

Описание транспортных задержек

```
wire y_out;

assign #3 y_out = a & b;
```

Спецификация задержки для транспорта по межсоединению

```
wire #3 y_out;

wire y_out = a & b;
```

Короткая форма записи

```
wire #3 y_out = a & b;
```

Задержка от входа к выходу

```
module A( q, a, b, c, d );
  input a, b, c, d;
  output q;
  wire e, f;
  specify
    ( a => q ) = 6;
    ( b => q ) = 7;
    ( c => q ) = 7;
    ( d => q ) = 6;
  endspecify

  or  o1( e, a, b );
  or  o2( f, c, d );
  xor x1( q, e, f );

endmodule
```

Совмещение кода на различных языках

```
module inv(x, y);
  input x;
  output y;

  assign y = ~x;

endmodule

entity tb_inv is
end tb_inv;

architecture test of tb_inv is
  component inv
    port (x: in bit; y: out bit);
  end component;

  signal x_in, y_out : bit;

begin
  p1 : inv port map (x=>x_in, y=>y_out);
end test;
```

Написание тестового окружения (1)

```
module main;
  reg d, clk, rst;
  wire q;

  dflipflop dff (d, clk, rst, q);

  always
    #5 clk = ~clk;

  initial
  begin
    clk = 0;
  end
```

```
initial
begin
  #0  d=0; rst=1;
  #4  d=1; rst=0;
  #50 d=1; rst=1;
  #20 d=0; rst=0;
end

endmodule
```

Verilog PLI (Programming Language Interface)

```
#include <stdio.h>
#include <vpi_user.h>

void hello() {
  printf("HELLO");
}

void register_hello() {
  s_vpi_systf_data data;
  data.type      = vpiSysTask;
  data.tfname    = "$hello";
  data.calltf    = hello;
  data.compiletf = 0;
  data.sizetf    = 0;
  data.user_data = 0;
  vpi_register_systf(&data);
}
```

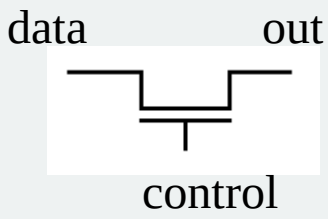
```
module test();

  initial
  begin
    $hello;
    #10 $finish;
  end
endmodule
```

```
gcc
  hello_vpi.c
  -fPIC
  -shared
  -o hello_vpi.so
```

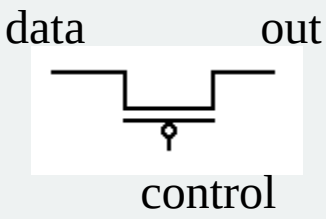
```
ncverilog
  test.v +access+r
  -loadvpi ./hello_vpi.so:register_hello
```

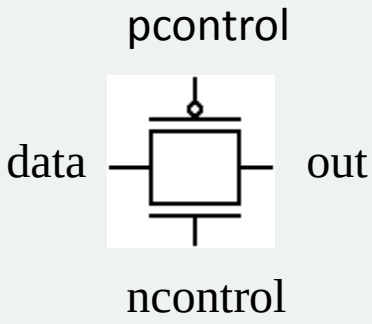
Ключевые модели

nmos:



pmos:






```
nmos(out, data, control);

pmos(out, data, control);

cmos(out, data, ncontrol, pcontrol);
```

```
module inv(x, y);
  input x;
  output y;

  supply1 pwr_bus;
  supply0 gnd_bus;

  pmos(y, pwr_bus, x);
  nmos(y, gnd_bus, x);

endmodule
```