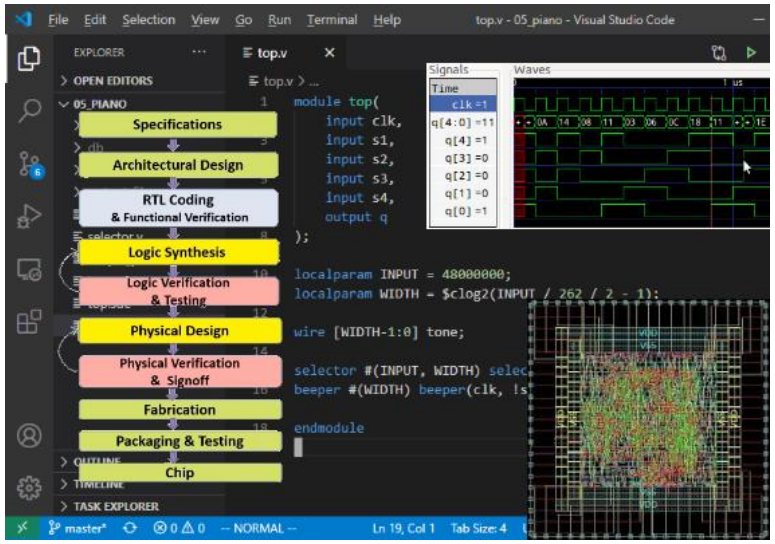
МИЭТ
Национальный исследовательский университет «МИЭТ»
Институт интегральной электроники

Лингвистические средства проектирования



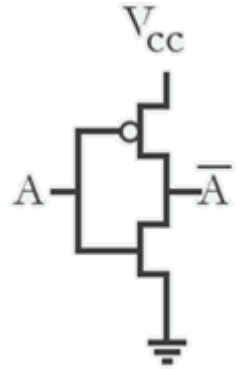
Лекция 1

Маршрут разработки цифровых ИС

Основы синтаксиса языка Verilog HDL

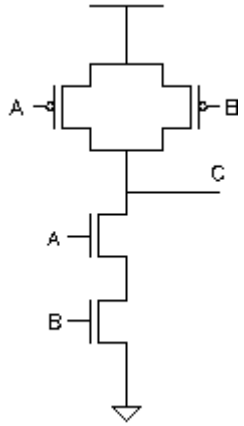
Языки описания схем на транзисторном уровне: SPICE

SPICE - Simulation Program with Integrated Circuit Emphasis
(программа моделирования с акцентом на интегральные схемы)




```
* CMOS Inverter
Vin 1 0 pulse(0v 5v 0ns 5ns 5ns 48ns 120ns)
Vdd 2 0 5v

m1 3 1 0 0 nmos1 L=2.5u W=5u AD=1.0e-11 AS=1.0e-11
m2 3 1 2 2 pmos1 L=2.5u W=15u AD=3.0e-11 AS=3.0e-11
CL 3 0 100fF

.model nmos1 nmos (LEVEL=2, UO=150, VTO=1.4581, GAMMA=1.8658,
+ PHI=0.7974, KP=1.0354e-5 ,LAMBDA=0.02, XJ=0.2u, LD=0.2u,
+ PB=0.9939, NSUB=5e16, NSS=2e10, TOX=50n, TPG=+1)
.model pmos1 pmos (LEVEL=2, UO=316.67, VTO=-1.5488,
+ GAMMA=1.8658, PHI=0.7974, KP=2.1860e-5, LAMBDA=0.02,
+ XJ=0.2u, LD=0.2u, PB=0.9939, NSUB=5e16, NSS=2e10,
+ TOX=50nm, TPG=+1)
.probe v(1) v(3)
.tran 10n 200n
.end
```

Индивидуальные и групповые параметры компонентов ИС



Индивидуальные параметры

Групповые параметры

```
M1 3 2 1 0 NMOS L=1u W=6u

.MODEL NFET NMOS LEVEL=2
+ VTO=-1.44
+ KP=8.64E-6 NSUB=1E17
+ TOX=20n
```

Что значит «SPICE-подобный» симулятор?

Математическая модель схемы:
$F(x, \dot{x}, t) = 0$

- МУП
- Модифицированный МУП

Алгебраизация математической модели схемы:
$F(x) = 0$

- Неявный метод Эйлера
- Метод трапеций

Линеаризация математической модели схемы:
$Yx + I = 0$

Метод Ньютона-Рафсона

Решение математической модели схемы:
$Yx + I = 0$

Метод Гаусса

Что такое «SPICE-подобный» нетлист?

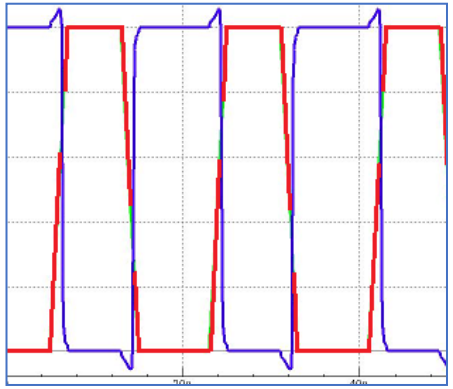
```
* SPICE netlist written by S-Edit Win32 7.00
* Written on Aug 30, 2008 at 11:36:09
* Waveform probing commands
.probe
.options probefilename="Nand2.dat"
+ probetopmodule="Module0"
+ probesdbfile="C:\udo\Nand2.sdb"
* Main circuit: Module0
M1 outp b N2 Gnd Nh L=.15u W=.45u AD=.3375p PD=2.4u AS=.3375p PS=2.4u
M2 N2 a Gnd Gnd Nh L=.15u W=.45u AD=.3375p PD=2.4u AS=.3375p PS=2.4u
M3 outp a Vdd Vdd Ph L=.15u W=.45u AD=66p PD=2.4u AS=.3375p PS=2.4u
M4 outp b Vdd Vdd Ph L=.15u W=.45u AD=.3375p PD=2.4u AS=.3375p PS=2.4u
v5 b Gnd bit({01011} pw=100n on=1.0 off=0.0 rt=.10n ft=.10n delay=0
+ lt=100n ht=100n)
v6 a Gnd bit({01101} pw=100n on=1.0 off=0.0 rt=.10n ft=.10n delay=0
+ lt=100n ht=100n)
vdd vdd gnd 1
.include dual.md
.tran .1n 800n
.power 100n 200n
.print tran a b outp
.END
```

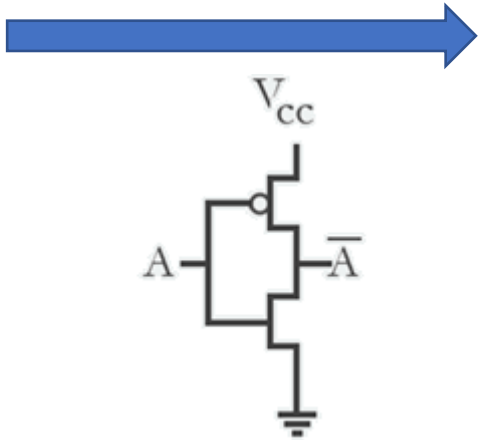
Языки описания схем на транзисторном уровне: spectre

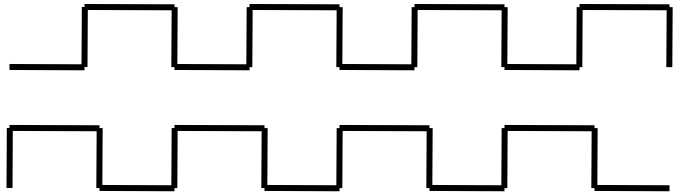
cadence®

```
simulator lang=spectre
global 0

include "./d25.m"

inst0 (op ip vdd vdd) d25P w=1.8e-06 l=2.4e-07
inst1 (op ip vss vss) d25N w=8.4e-07 l=2.4e-07
inst2 (ip vdd) capacitor c=2.51424e-17 m=1

vip (ip 0) vsource val0=0.0 val1=3.0 delay=0n rise=.05n \
  fall=.05n width=5n period=10n type=pulse
vvss (vss 0) vsource dc=0 type=dc
vvdd (vdd 0) vsource dc=3 type=dc

simulatorOptions options reltol=1e-3 vabstol=1e-6 \
  iabstol=1e-12 temp=27 tnom=27 scalem=1.0 scale=1.0 \
  gmin=1e-12 rforce=1 maxnotes=5 maxwarns=5

tran tran stop=40n write="spectre.ic" writefinal="spectre.fc"
annotate=status maxiters=5
```

Появление класса цифровых схем





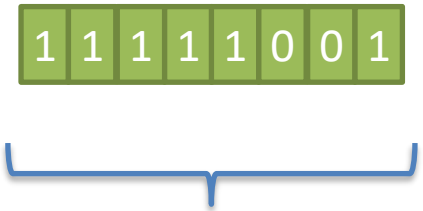


```
* CMOS Inverter

Vin 1 0 pulse(0v 5v 0ns 5ns 5ns 48ns 120ns)
Vdd 2 0 5v

m1 3 1 0 0 nmos1 L=2.5u W=5u AD=1.0e-11 AS=1.0e-11
m2 3 1 2 2 pmos1 L=2.5u W=15u AD=3.0e-11 AS=3.0e-11
CL 3 0 100fF

.model nmos1 nmos (LEVEL=2, UO=150, VTO=1.4581,
+ GAMMA=1.8658, PHI=0.7974, KP=1.0354e-5 ,LAMBDA=0.02,
+ XJ=0.2u, LD=0.2u, PB=0.9939, NSUB=5e16, NSS=2e10,
+ TOX=50n, TPG=+1)
.model pmos1 pmos (LEVEL=2, UO=316.67, VTO=-1.5488,
+ GAMMA=1.8658, PHI=0.7974, KP=2.1860e-5, LAMBDA=0.02,
+ XJ=0.2u, LD=0.2u, PB=0.9939, NSUB=5e16, NSS=2e10,
+ TOX=50nm, TPG=+1)

.probe v(1) v(3)
.tran 10n 200n
.end
```

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

  assign y = ~x;
endmodule
```

Дополненный двоичный код

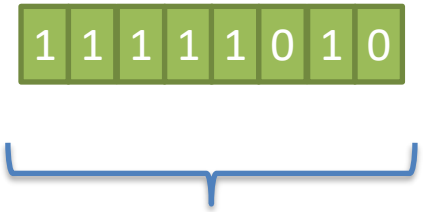
Десятичное представление	Двоичное представление (8 бит)		
	прямой	обратный	дополнительный
127	0111 1111	0111 1111	0111 1111
1	0000 0001	0000 0001	0000 0001
0	0000 0000	0000 0000	0000 0000
-0	1000 0000	1111 1111	---
-1	1000 0001	1111 1110	1111 1111
-2	1000 0010	1111 1101	1111 1110
-3	1000 0011	1111 1100	1111 1101
-4	1000 0100	1111 1011	1111 1100
-5	1000 0101	1111 1010	1111 1011
-6	1000 0110	1111 1001	1111 1010
-7	1000 0111	1111 1000	1111 1001
-8	1000 1000	1111 0111	1111 1000
-9	1000 1001	1111 0110	1111 0111
-10	1000 1010	1111 0101	1111 0110
-11	1000 1011	1111 0100	1111 0101
-127	1111 1111	1000 0000	1000 0001
-128	---	---	1000 0000



+249? -7?

Выполняем операцию «сложение с единицей».

Результат:



Для положительных чисел: +250

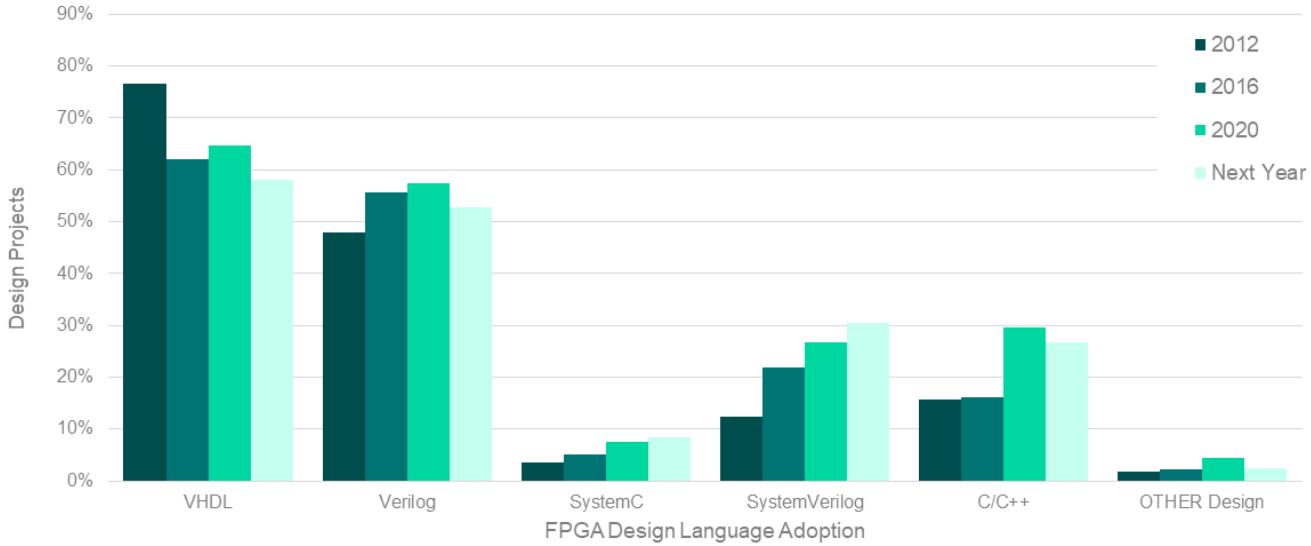
Для отрицательных чисел: -6

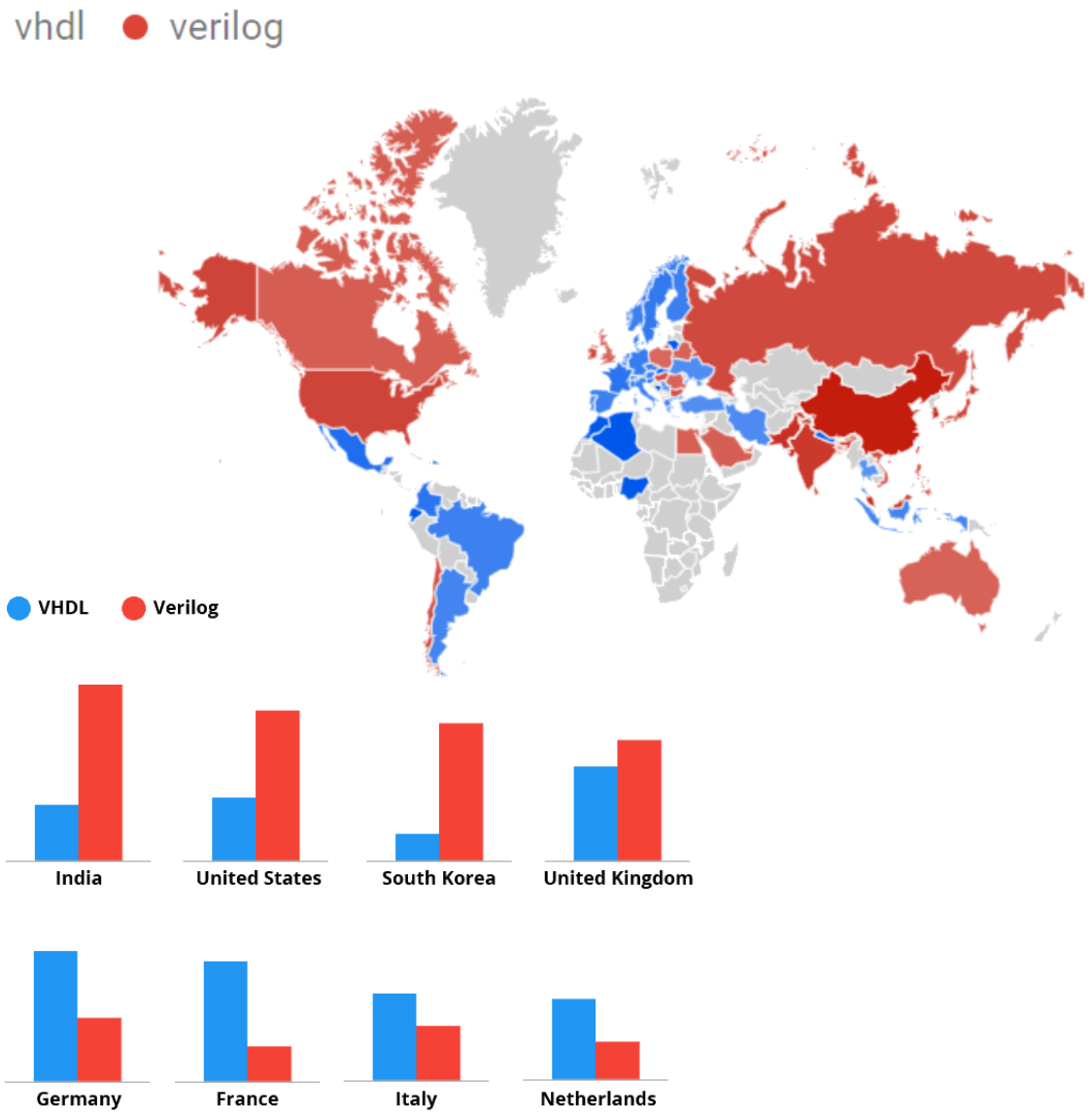
Требования к языкам проектирования

Основными требованиями к цифровым языкам в САПР являются:

1. технологичность разработки методом нисходящего проектирования;
2. получение надёжных систем;
3. мобильность проектов;
4. сопровождаемость проектов;
5. простота написания кода;
6. рациональное разграничение использования средств языка на различных уровнях абстракции.

Verilog vs VHDL

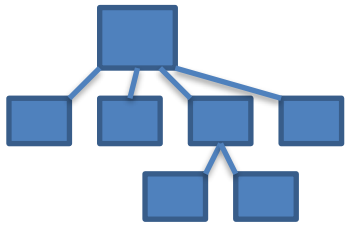




*Wilson Research Group and Mentor, A Siemens Business
https://vhdlwhiz.com/should-i-learn-vhdl-if-verilog-is-becoming-more-popular/

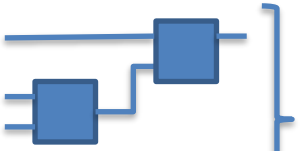
Характеристики языка Verilog HDL

- Поддержка иерархии – проектируемые устройства разбиваются на блоки.
- Описание интерфейса элемента неразрывно связано с описанием его функционирования.
- Функционирование может быть задано либо с помощью алгоритма, либо с помощью указания перечня компонентов и их связи между собой.
- Возможность моделирования параллельно протекающих процессов.
- Возможность проведения и чисто временного, и функционального моделирования.

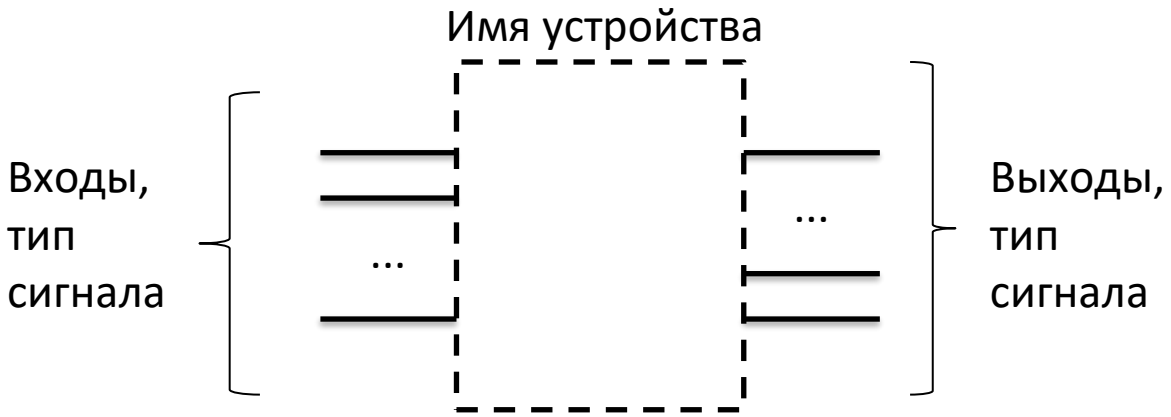


x, y : y = F(x)

```
if (X = 1) then
  Y = 0;
else
  Y <= 1;
end
```



Описание схем на языке Verilog: интерфейсная часть



Имя устройства

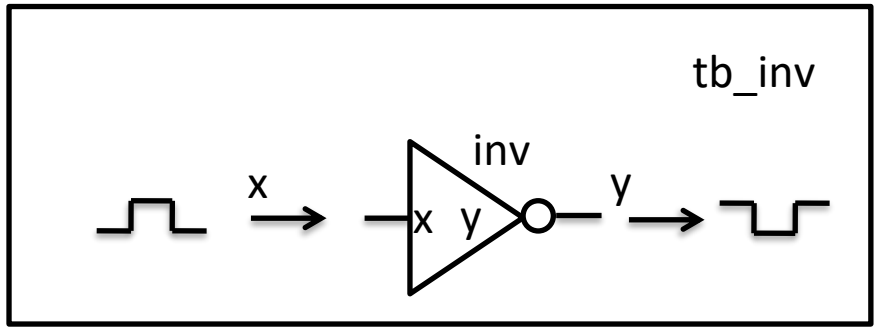
Порты устройства

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


endmodule
```

Указание направлений портов

Написание тестового окружения

```
`timescale 1ns/1ns
module tb_inv;
  reg x;
  wire y;

  inv i1(y, x);

  initial begin
    $dumpfile("inv.vcd");
    $dumpvars(1, tb_inv);
    #0  x = 0;
    #10 x = 1;
    #10 x = 0;
    #10 x = 1;
    #10 x = 0;
    #10 $finish;
end

endmodule
```



Правила именования идентификаторов

Правильные имена:

inverter

nand4

counter_16

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

  assign y = ~x;
endmodule
```

Неправильные имена:

not, **nand** - зарезервированные слова

8_bit_counter - нельзя начинать с цифры

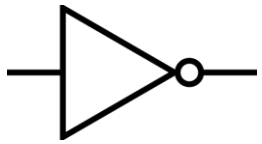
mux? - недопустимый символ в названии

my design - нельзя допускать пробелы в названиях

Можно обойти это ограничение, если отбить недопустимый символ символом «\».
Например:

```
\8inv\?
```

Описание портов устройств (1)

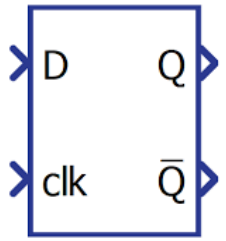


```
module inv (x, y);
  input  x;
  output y;

  ...

endmodule
```

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

  ...

endmodule
```

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

  ...

endmodule
```

Описание портов устройств (2)



D Flip Flop

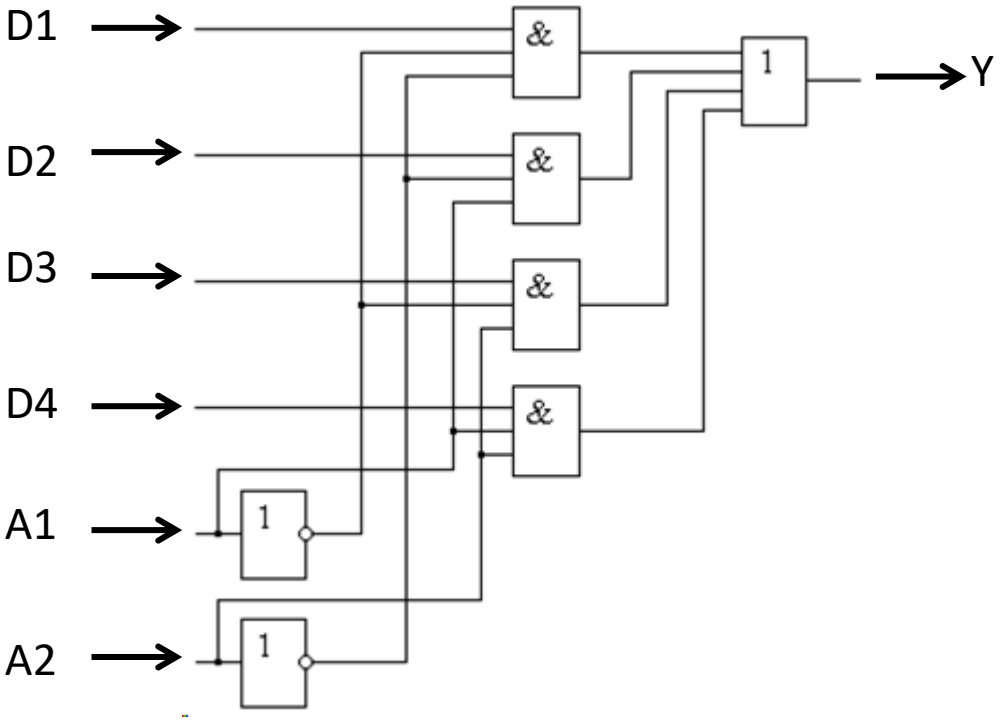
```
module dff (clk, d, q, nq);
  input  clk, d;
  output q, nq;
  reg    q, nq;

  ...

endmodule
```

```
module dff (clk, d, q, nq);
  input clk, d;
  output reg q, nq;

  ...

endmodule
```

```
module dff (
  input clk,
  input d,
  output reg q,
  output reg nq
);

  ...

endmodule
```

Направления портов: входной и выходной

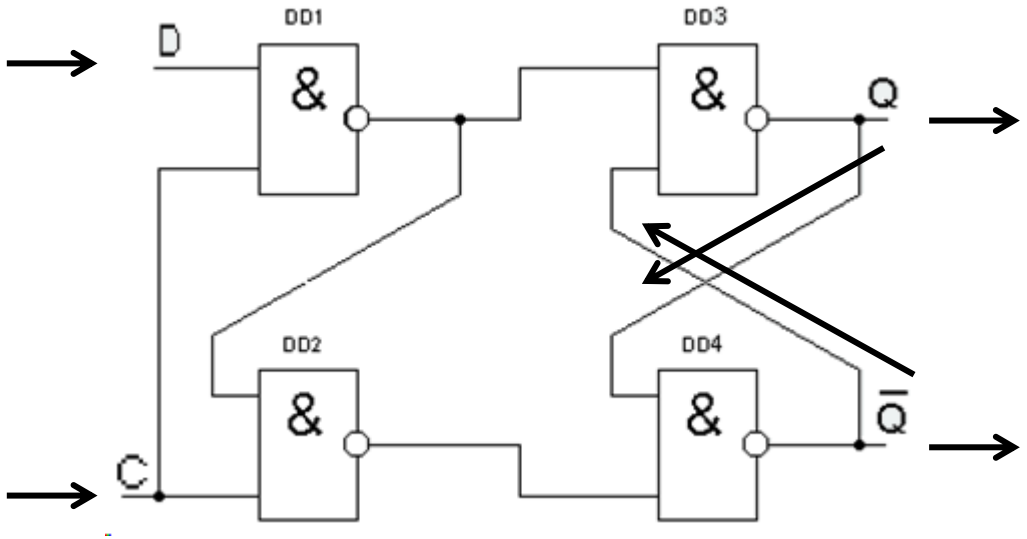
Направление входного порта – **input**
Направление выходного порта - **output**




```
module MUX(D1, D2, D3, D4,
           A1, A2,
           Y);

  input  D1, D2, D3, D4, A1, A2;
  output Y;

endmodule
```

Направления портов: двунаправленный порт

Направление входного порта – **IN**
Направление выходного порта - **INOUT**



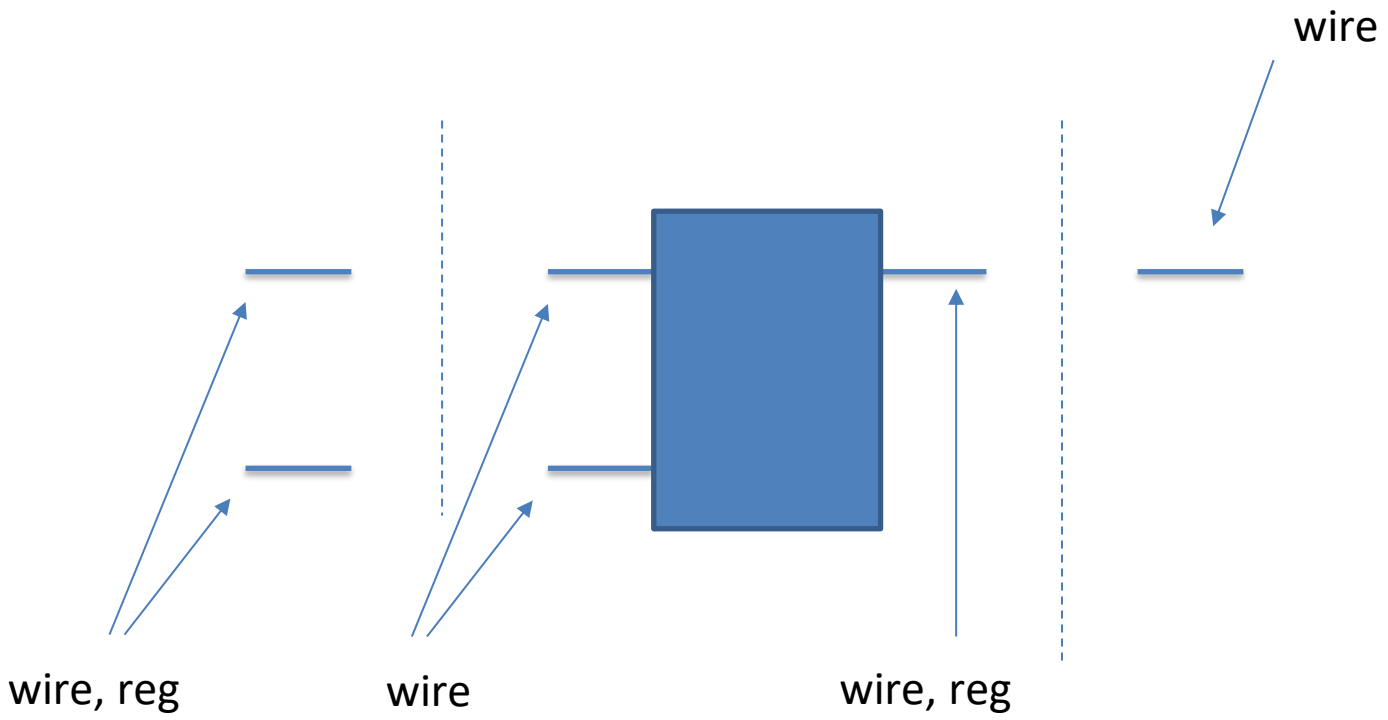

```
module DFF(D, C, Q, nQ);
  input D, C;
  inout Q, nQ;

  ...

end DCTT;
```

Данные в языке Verilog

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



Задание значений сигналов (1)

Сигналы в языке Verilog могут принимать значения:

- 0
- 1
- x
- z

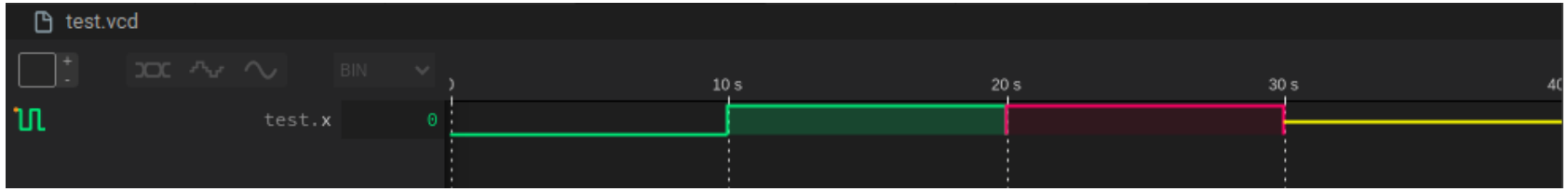
Способы задания константных значений:

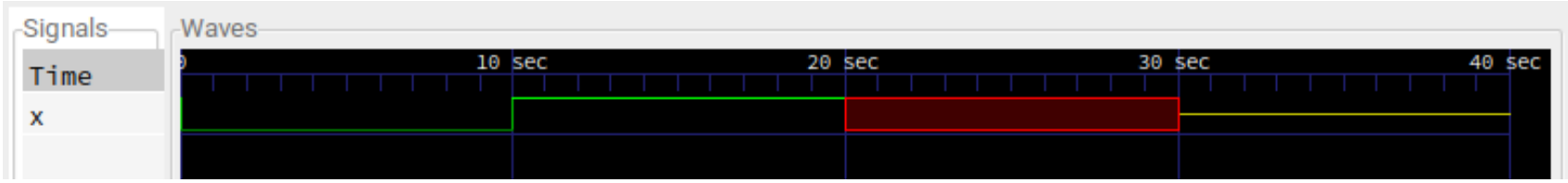
<размер>'<основание><значение>

- 1'b0
- 1'b1
- 1'bx
- 1'bz

```
8'b11001101
8'b11_00_11_01
```

```
module test;
  reg x;
  initial begin
    $dumpfile("test.vcd");
    $dumpvars(1, x);
        x = 1'b0;
    #10 x = 1'b1;
    #10 x = 1'bx;
    #10 x = 1'bz;
    #10 $finish;
  end
endmodule
```





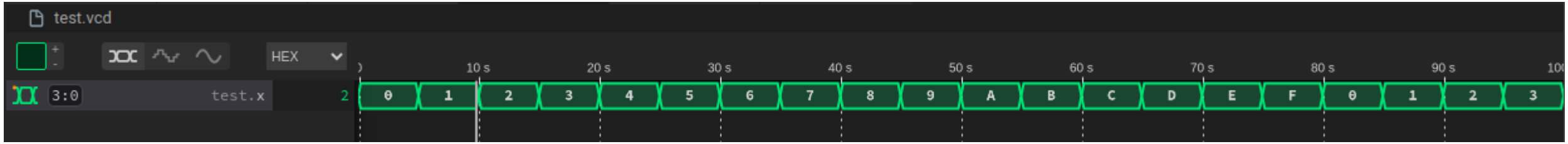
Задание значений сигналов (2)

Способы задания константных значений:
<размер>'<основание><значение>

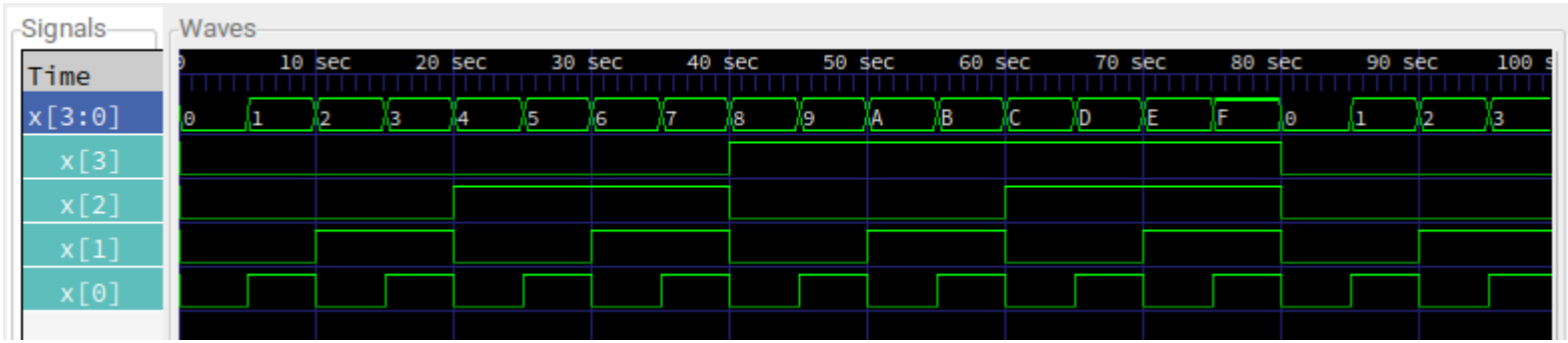
- `4'b1011`
- `2'hFF`
- `3'o671`
- `4'b1x0z`

Данные могут быть организованы в векторы и массивы.
Массивами могут быть объявлены только reg, integer, time

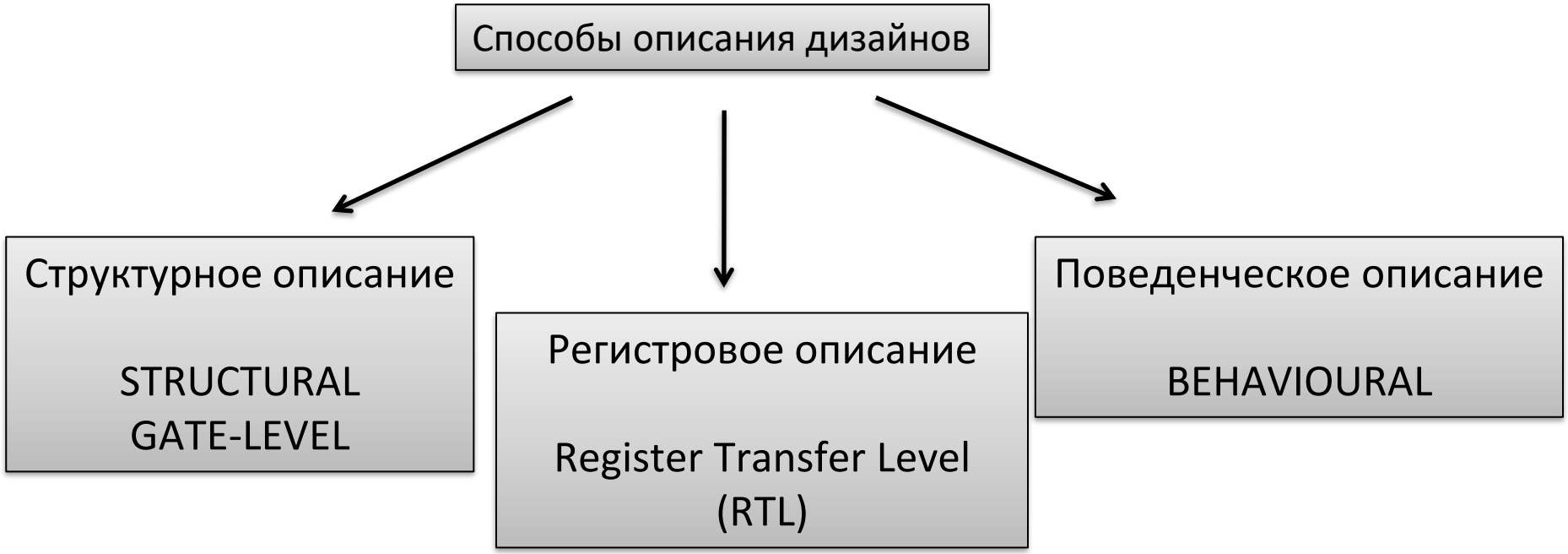
```
wire [3:0] data;      // 4-битный вектор
reg bit [1:8];        // массив из 8 однобитных сигналов
reg [3:0] mem [1:8]; // массив из 8 4-битных векторов
```



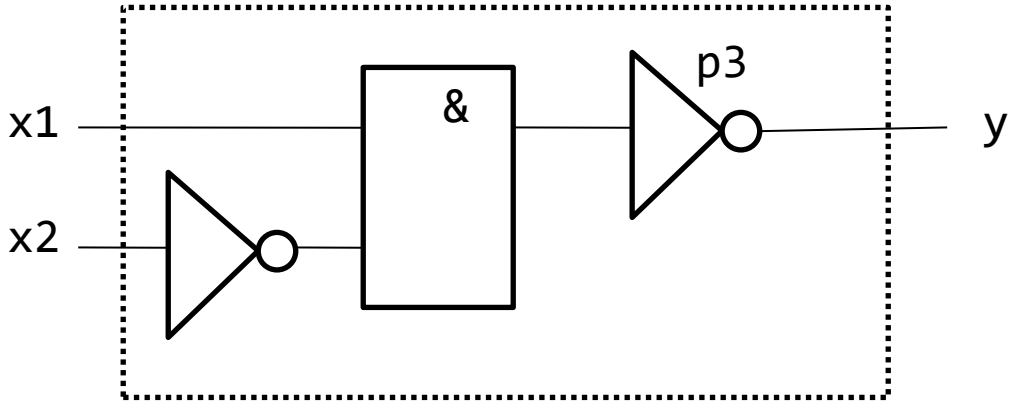

```
reg [0:7] byte;
reg data [0:7];

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



Способы описания поведения элементов

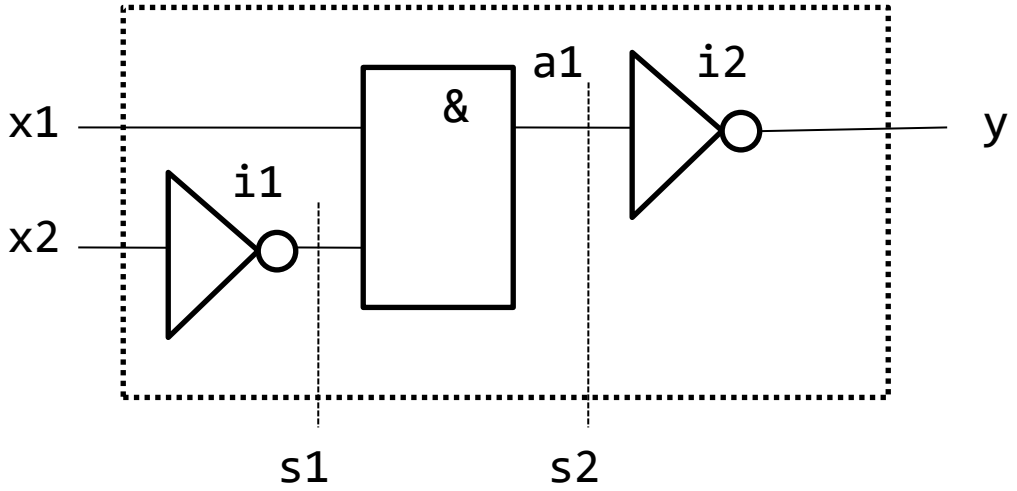


Уровень RTL



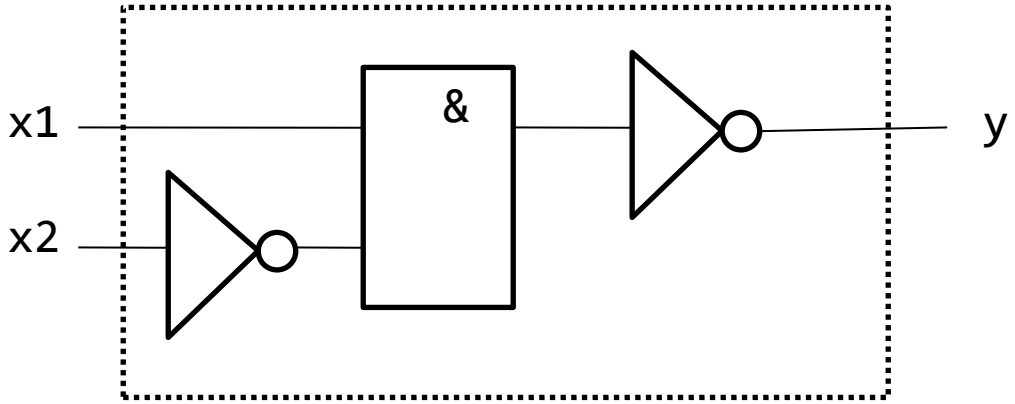

```
module device(x1, x2, y);
  input  x1, x2;
  output y;

  assign y = ~(x1 & ~x2);

endmodule
```

Структурное описание (вентильный уровень)




```
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
```

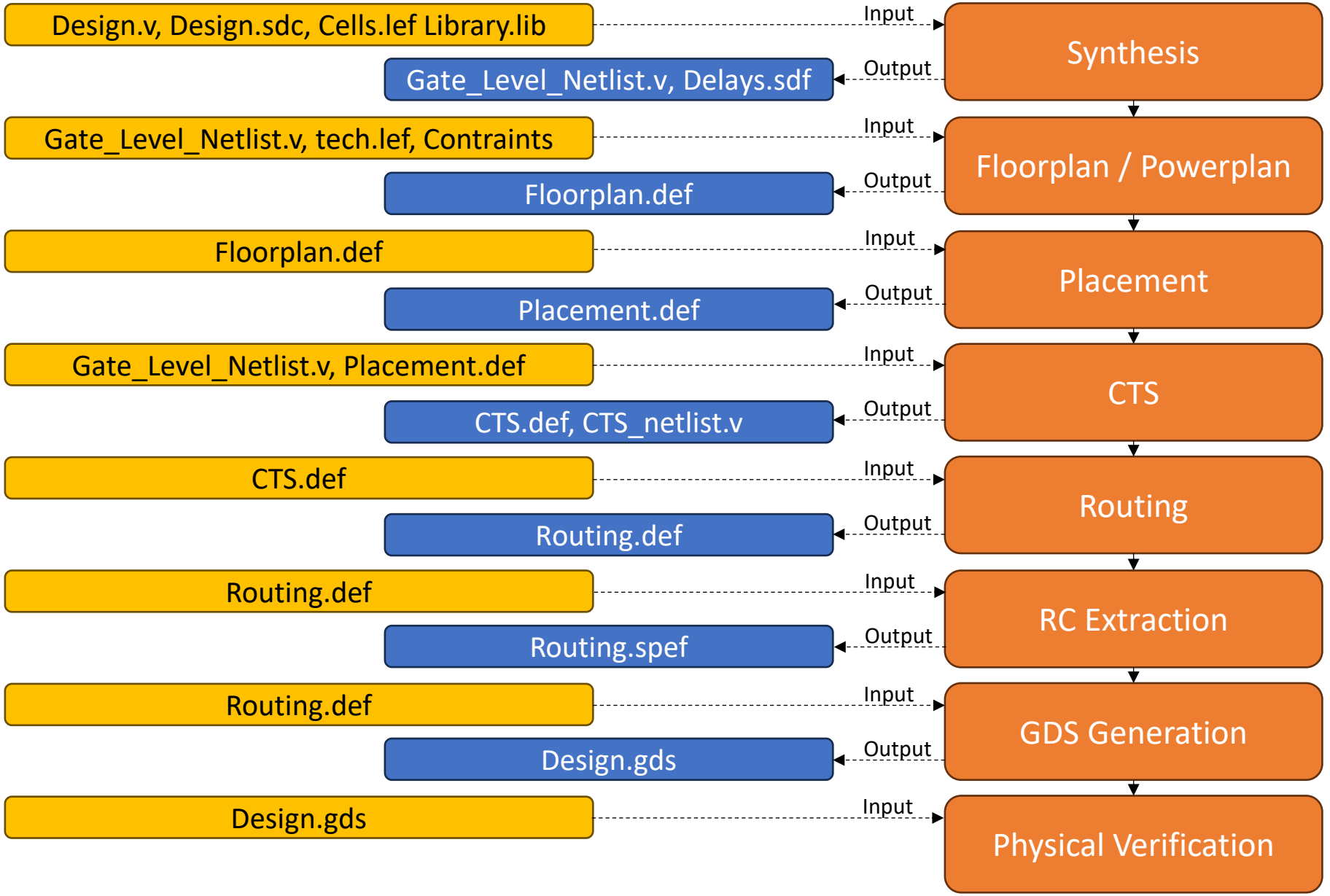
Поведенческое описание




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

  always @ (x1 or x2) begin
    if(x1 == 1 and x2 == 0)
      y <= 1'b0;
    else
      y <= 1'b1;
  end

endmodule
```

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

Открытый маршрут проектирования OpenLANE



Форматы:

- .v
- .lib
- .lef
- .def
- .gds
- .sdf
- .spef
- отчёты STA