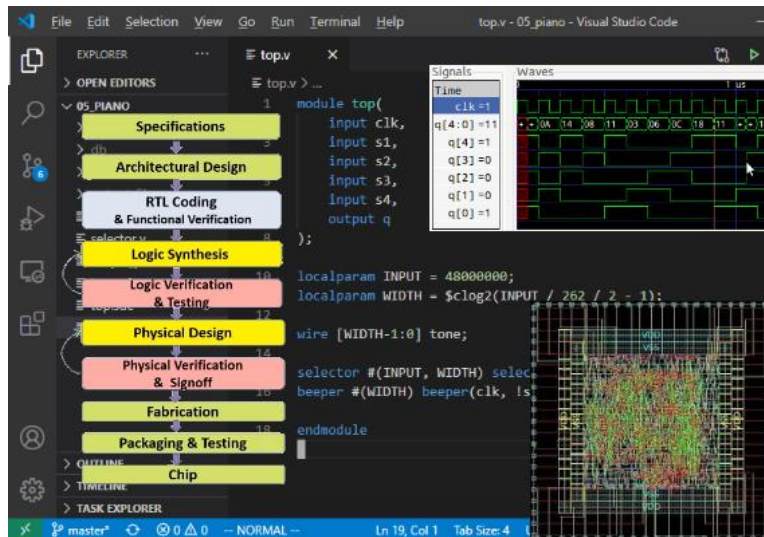




# Лингвистические средства проектирования

Лекция 8



Логический синтез.

## Основные проблемы синтеза

### 1. Проблема синтезируемости конструкций.

Не все операторы языка могут быть представлены в виде схемы.

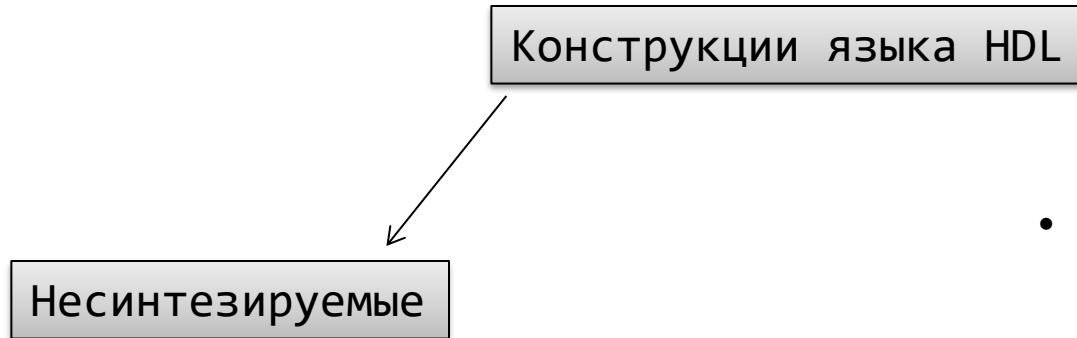
### 2. Проблема корректности синтезированных данных.

Не всегда поведение синтезированной схемы соответствует результатам моделирования. Не всегда поведение схемы, описанной на различных уровнях абстракции, совпадает.

### 3. Проблема эффективности описаний.

Схожие описания в коде могут дать одинаковый результат при моделировании, но совершенно разный результат при синтезе схем.

## Что относится к несинтезируемым конструкциям?



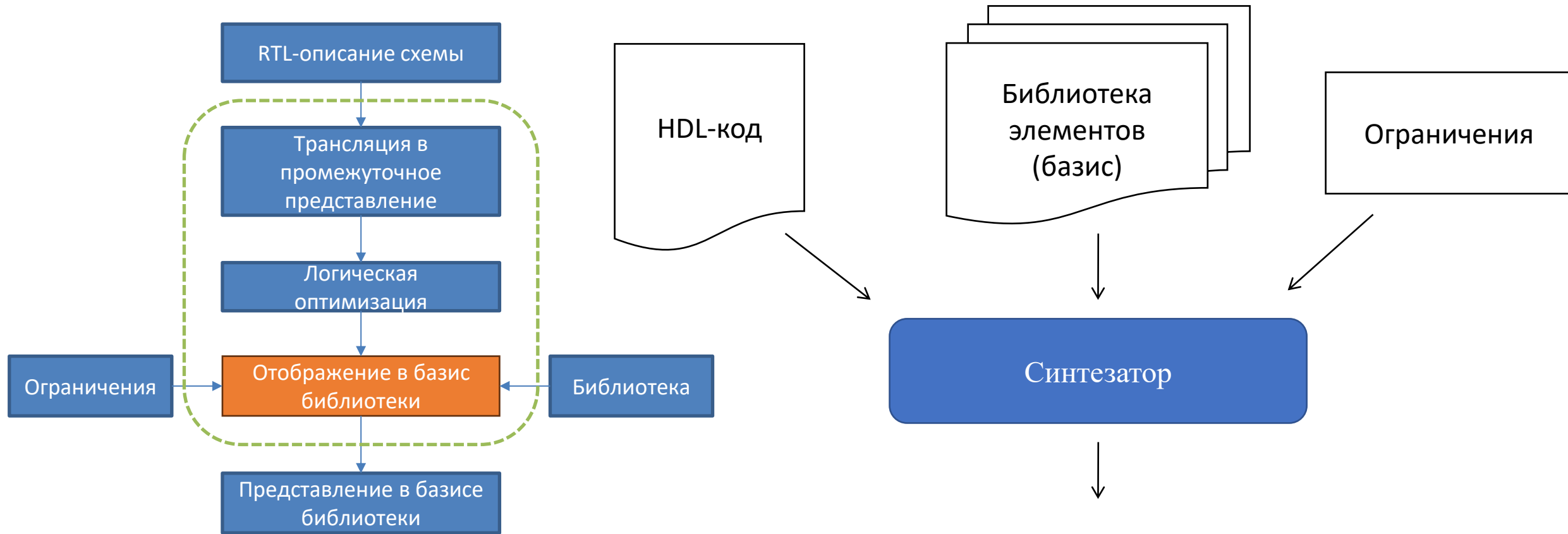
Возможно проведение функциональной и временной верификации.

Для получения топологии нужно:

1. переписывать HDL-код,
2. создавать схемотехническое представление вручную;
3. формировать топологию вручную.

- системные функции (начинаются с \$: \$dumpfile, \$dumpvars, \$monitor, \$finish);
- любые конструкции с нецелыми числами;
- любые конструкции, связанные со временем (задержки и т.д.);
- описания в виде ключевых моделей, UDP, узлы с tri0, tri1;
- события;
- fork, join;
- блоки initial;
- списки чувствительности;

# Синтез цифровых схем



1. gate-level HDL-описание;
2. нетлист на аналоговом языке – spice, spectre;
3. топологическое представление схемы.

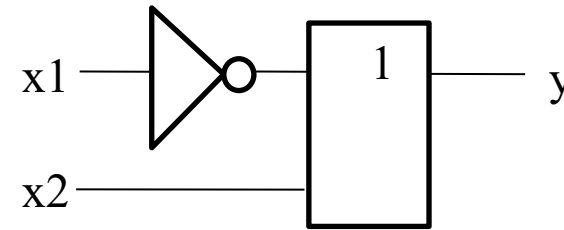
# Синтез комбинационных схем (1)

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



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

	x1	
	0	1
0	1	0
1	1	1

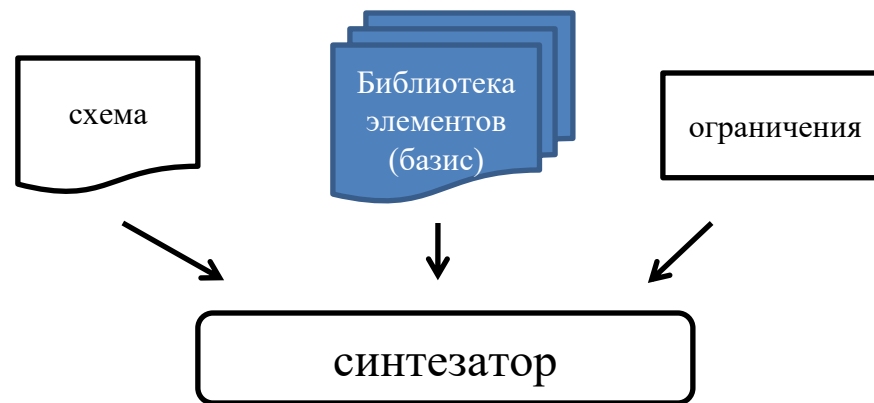


$$\overline{x1} + x2$$

Базис – «ИЛИ» - смотрим по единицам

## Синтез комбинационных схем (2)

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

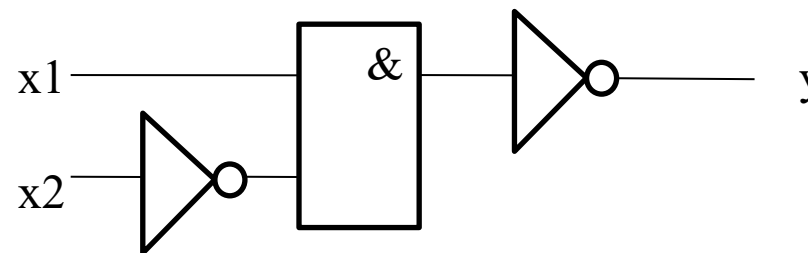


↓

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

→

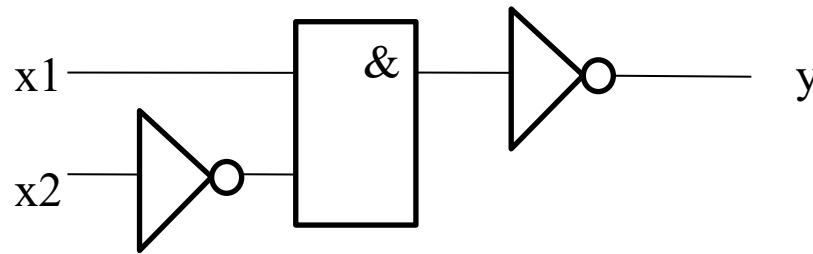
		x1	
		0	1
x2	0	1	0
	1	1	1



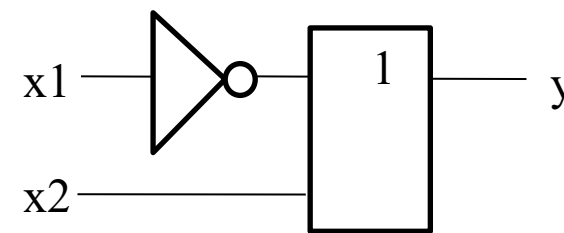
$x1 * x2$

Базис – «И» - смотрим по нулям, выход инвертируем

# Что зависит от выбора библиотеки для синтеза?

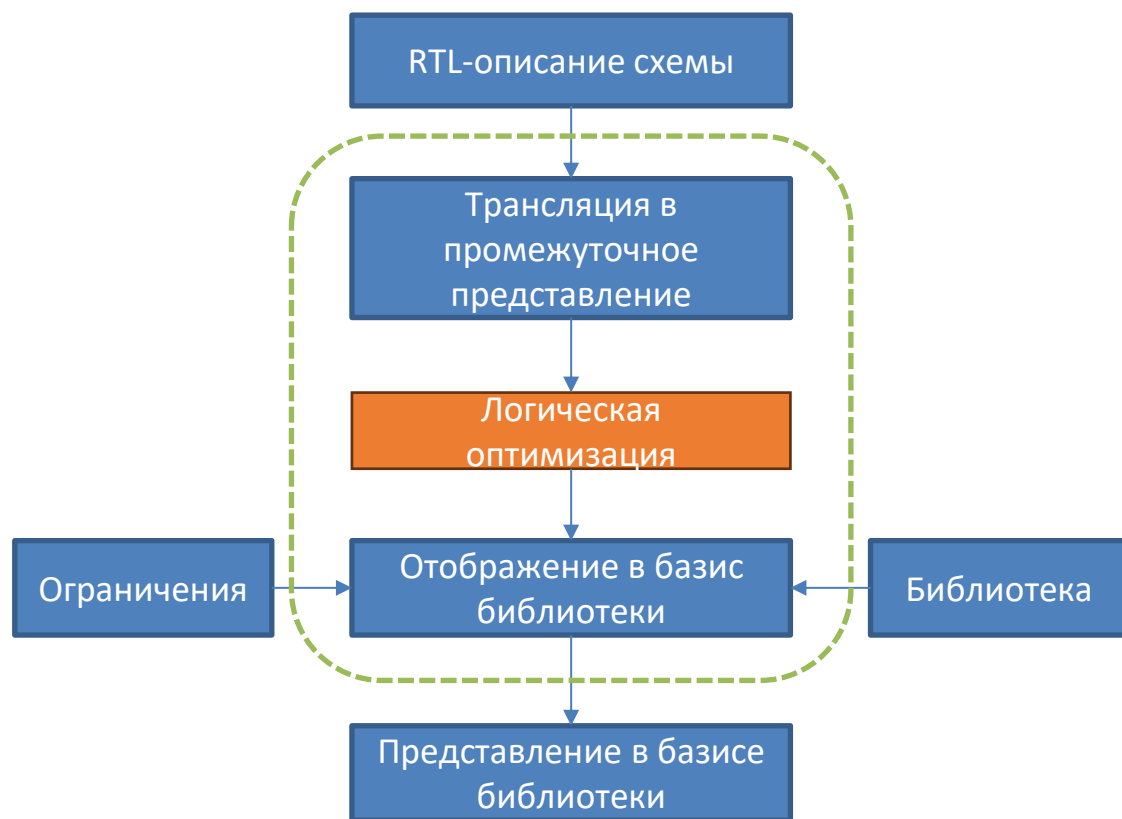


Число транзисторов: 10



Число транзисторов: 8

# Логическая оптимизация



x1	x2	x3	f
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

$$f(x_1, x_2, x_3) = \neg x_1 \wedge \neg x_2 \wedge \neg x_3 + \neg x_1 \wedge \neg x_2 \wedge x_3 + \neg x_1 \wedge x_2 \wedge \neg x_3 + x_1 \wedge x_2 \wedge \neg x_3$$



$$f(x_1, x_2, x_3) = \neg x_1 \wedge \neg x_2 + \neg x_1 \wedge x_2 \wedge \neg x_3 + x_1 \wedge x_2 \wedge \neg x_3$$

## Синтез схем в OpenLane: inverter

<OpenLane path>/designs/inverter/src/inverter.v

```
module inverter (  
    input wire in,  
    output out );  
  
    assign out = ~in;  
  
endmodule
```

<OpenLane path>/designs/inverter/runs/results/synthesis/inverter.v

```
/* Generated by Yosys 0.34  
   (git sha1 4a1b5599258, gcc 8.3.1 -fPIC -Os) */  
  
module inverter(in, out);  
    input in;  
    wire in;  
    output out;  
    wire out;  
    sky130_fd_sc_hd__inv_2 _0_ (  
        .A(in),  
        .Y(out)  
    );  
endmodule
```

## Синтез схемы счётчика в САПР Synopsys

```
module Johnson_count(clk, r, out);
  input clk;
  input r;
  output reg [0:3]out;
  always @ (negedge clk or negedge r)
    if (r == 0)
      out <= 4'b1111;
    else
      out <= out + 1'b1;
endmodule
```

```
module Johnson_count ( clk, r, out );
  output [0:3] out;
  input clk, r;

  wire N2, N3, N4, n1, n3, n4, n5;

  DFFARX1_RVT out_reg3 (n5, clk, n1, out[3], n5);
  DFFARX1_RVT out_reg2 (N2, clk, n1, out[2]);
  DFFARX1_RVT out_reg1 (N3, clk, n1, out[1]);
  DFFARX1_RVT out_reg0 (N4, clk, n1, out[0]);
  INVX0_RVT U3 (r, n1);
  NAND2X0_RVT U8 (out[1], n4, n3);
  AND2X1_RVT U9 (out[2], out[3], n4);
  XNOR2X1_RVT U10 (n3, out[0], N4);
  XOR2X1_RVT U11 (out[1], n4, N3);
  XNOR2X1_RVT U12 (n5, out[2], N2);
endmodule
```

## Синтез схемы счётчика в маршруте OpenLane

```
module counter(clk, r, out);
  wire _00_, _01_, _02_, _03_, _04_, _05_, _06_,
        _07_, _08_, _09_;
  input wire clk, r;
  output wire [0:3] out;
  sky130_fd_sc_hd__inv_2__10_ (.A(out[3]), .Y(_00_));
  sky130_fd_sc_hd__xor2_2__11_ (.A(out[3]), .B(out[2]), .X(_01_));
  sky130_fd_sc_hd__a21oi_2__12_ (.A1(out[3]), .A2(out[2]), .B1(out[1]), .Y(_08_));
  sky130_fd_sc_hd__and3_2__13_ (.A(out[3]), .B(out[2]), .C(out[1]), .X(_09_));
  sky130_fd_sc_hd__nor2_2__14_ (.A(_08_), .B(_09_), .Y(_02_));
  sky130_fd_sc_hd__xor2_2__15_ (.A(out[0]), .B(_09_), .X(_03_));
  sky130_fd_sc_hd__inv_2__16_ (.A(clk), .Y(_04_));
  sky130_fd_sc_hd__inv_2__17_ (.A(clk), .Y(_05_));
  sky130_fd_sc_hd__inv_2__18_ (.A(clk), .Y(_06_));
  sky130_fd_sc_hd__inv_2__19_ (.A(clk), .Y(_07_));
  sky130_fd_sc_hd__dfstp_2__20_ (.CLK(_04_), .D(_00_), .Q(out[3]), .SET_B(r));
  sky130_fd_sc_hd__dfstp_2__21_ (.CLK(_05_), .D(_01_), .Q(out[2]), .SET_B(r));
  sky130_fd_sc_hd__dfstp_2__22_ (.CLK(_06_), .D(_02_), .Q(out[1]), .SET_B(r));
  sky130_fd_sc_hd__dfstp_2__23_ (.CLK(_07_), .D(_03_), .Q(out[0]), .SET_B(r));
endmodule
```

## Основные проблемы синтеза

### 1. Проблема синтезируемости конструкций.

Не все операторы языка могут быть представлены в виде схемы.

### 2. Проблема корректности синтезированных данных.

Не всегда поведение синтезированной схемы соответствует результатам моделирования. Не всегда поведение схемы, описанной на различных уровнях абстракции, совпадает.

### 3. Проблема эффективности описаний.

Схожие описания в коде могут дать одинаковый результат при моделировании, но совершенно разный результат при синтезе схем.

## Подготовка к синтезу: написание кода (1)

Файл counter.v

```
module counter(clk, reset, enable, out);
```

```
    input clk, reset, enable;
```

```
    output reg [2:0] out;
```

```
    always @(posedge reset or negedge clk)
```

```
        if(reset)
```

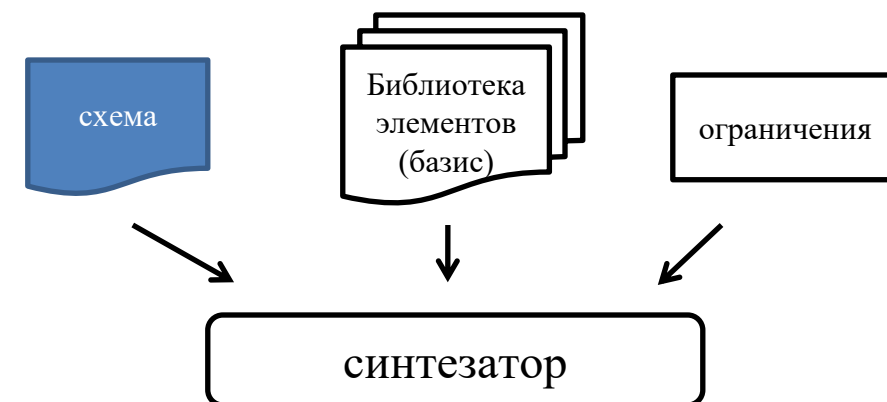
```
            out <= 3'b000;
```

```
        else
```

```
            if(enable)
```

```
                out <= out + 3'b001;
```

```
    endmodule
```



## Подготовка к синтезу: написание кода (2)

```
`timescale 1ns/1ps

module tb_counter;

    reg clock, reset, enable;
    wire [2:0] out;

    initial
    begin
        $dumpfile("tb_counter.vcd");
        $dumpvars(0, tb_counter);

        clock = 0;
        reset = 0;
        enable = 0;

        #500 $finish;
    end
```

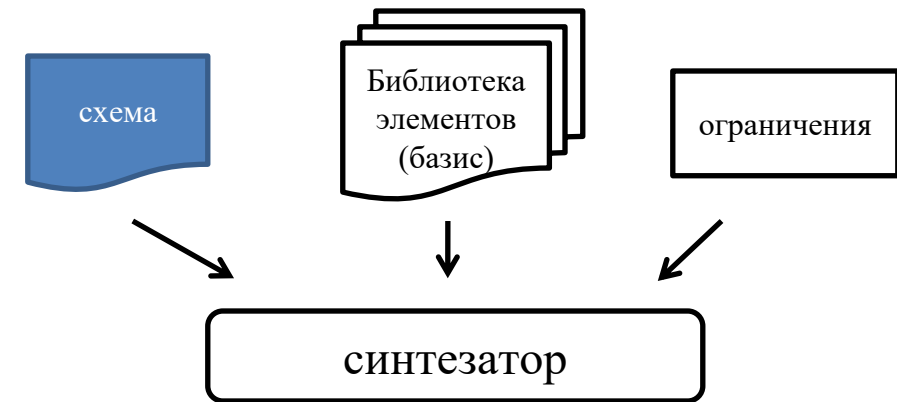
```
initial begin
    #30 reset = 1'b1;
    #5  reset = 1'b0;
end

initial begin
    #100 enable = 1'b1;
    #200 enable = 1'b0;
end

always #5 clock = ~clock;

counter u1 (clock, reset, enable, out);

endmodule
```



## Подготовка к синтезу: создание библиотеки элементов (1)

Файл cells.v (комбинационная часть)

```
module BUF(x, y);  
  input x;  
  output y;  
  
  assign y = x;  
endmodule
```

```
module INV(x, y);  
  input x;  
  output y;  
  
  assign y = ~x;  
endmodule
```

```
module NAND2(x1, x2, y);  
  input x1, x2;  
  output y;  
  
  assign y = ~(x1 & x2);  
endmodule
```

```
module NOR2(x1, x2, y);  
  input x1, x2;  
  output y;  
  
  assign y = ~(x1 | x2);  
endmodule
```

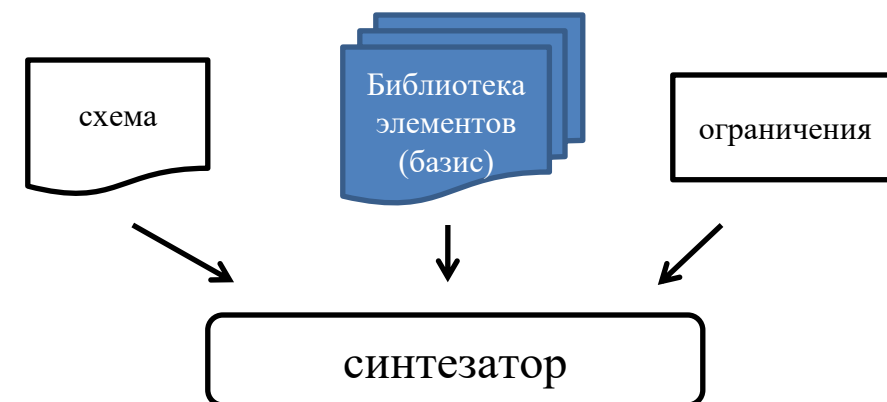


## Подготовка к синтезу: создание библиотеки элементов (2)

Файл cells.v (последовательная часть)

```
module DFF(CLK, D, Q, nQ);  
  input CLK, D;  
  output reg Q, nQ;  
  
  always @(negedge CLK)  
  begin  
    Q <= D;  
    nQ <= ~D;  
  end  
endmodule
```

```
module DFFSR(CLK, D, S, R, Q, nQ);  
  input CLK, D, S, R;  
  output reg Q, nQ;  
  
  always @(posedge R, posedge S, negedge CLK) begin  
    if(S) begin  
      Q <= 1'b1;  
      nQ <= 1'b0;  
    end  
    else  
      if(R) begin  
        Q <= 1'b0;  
        nQ <= 1'b1;  
      end  
      else begin  
        Q <= D;  
        nQ <= ~D;  
      end  
    end  
  end  
endmodule
```



## Подготовка к синтезу: подготовка файлов Liberty (1)

Файл cells.lib

```
library (test) {  
  cell(BUF) {  
    ...  
  }  
  cell(INV) {  
    ...  
  }  
  cell(NAND2) {  
    ...  
  }  
  cell(NOR2) {  
    ...  
  }  
  cell(DFF) {  
    ...  
  }  
  cell(DFFSR) {  
    ...  
  }  
}
```

```
cell(NOR2) {  
  area: 4;  
  pin(x1) {  
    direction: input;  
  }  
  pin(x2) {  
    direction: input;  
  }  
  pin(y) {  
    direction: output;  
    function: "(x1+x2)";  
  }  
}
```



## Подготовка к синтезу: подготовка файлов Liberty (2)

Файл cells.lib

```
library (test) {  
  cell(BUF) {  
    ...  
  }  
  cell(INV) {  
    ...  
  }  
  cell(NAND2) {  
    ...  
  }  
  cell(NOR2) {  
    ...  
  }  
  cell(DFF) {  
    ...  
  }  
  cell(DFFSR) {  
    ...  
  }  
}
```

```
cell(DFF) {  
  area: 18;  
  ff("IQ", "IQN") {  
    clocked_on: CLK;  
    next_state: D;  
  }  
  pin(CLK) {  
    direction: input;  
    clock: true;  
  }  
  pin(D) {  
    direction: input;  
  }  
  pin(Q) {  
    direction: output;  
    function: "IQ";  
  }  
  pin(nQ) {  
    direction: output;  
    function: "IQN";  
  }  
}
```



# Синтез цифровых схем: пакет yosys (1)



```
topgun@localhost:/eda/OpenLane
File Edit View Search Terminal Help
OpenLane Container (4476a58):/openlane/test design$ yosys

-----
| yosys -- Yosys Open SYnthesis Suite
|
| Copyright (C) 2012 - 2020 Claire Xenia Wolf <claire@yosyshq.com>
|
| Permission to use, copy, modify, and/or distribute this software for any
| purpose with or without fee is hereby granted, provided that the above
| copyright notice and this permission notice appear in all copies.
|
| THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
| WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
| MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
| ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
| WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
| ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
| OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
|
-----

Yosys 0.12+45 (git sha1 UNKNOWN, gcc 8.3.1 -fPIC -Os)

yosys>
```

## Синтез цифровых схем: пакет yosys (2)

```
yosys> read_verilog counter.v
```

```
1. Executing Verilog-2005 frontend: counter.v  
Parsing Verilog input from `counter.v' to AST representation.  
Generating RTLIL representation for module `counter'.  
Successfully finished Verilog frontend.
```

```
yosys> read_verilog -lib cells.v
```

```
2. Executing Verilog-2005 frontend: cells.v  
Parsing Verilog input from `cells.v' to AST representation.  
Generating RTLIL representation for module `BUF'.  
Generating RTLIL representation for module `INV'.  
Generating RTLIL representation for module `NAND2'.  
Generating RTLIL representation for module `NOR2'.  
Generating RTLIL representation for module `DFF'.  
Generating RTLIL representation for module `DFFSR'.  
Successfully finished Verilog frontend.
```



## Синтез цифровых схем: пакет yosys (3)

```
yosys> proc; memory; techmap;
```

3. Executing PROC pass (convert processes to netlists).

3.1. Executing PROC\_CLEAN pass (remove empty switches from decision trees).  
Cleaned up 0 empty switches.

3.2. Executing PROC\_RMDEAD pass (remove dead branches from decision trees).  
Marked 1 switch rules as full\_case in process \$proc\$counter.v:6\$1 in module counter.  
Removed a total of 0 dead cases.

3.3. Executing PROC\_PRUNE pass (remove redundant assignments in processes).  
Removed 0 redundant assignments.  
Promoted 0 assignments to connections.

3.4. Executing PROC\_INIT pass (extract init attributes).

3.5. Executing PROC\_ARST pass (detect async resets in processes).  
Found async reset \reset in ``\counter.\$proc\$counter.v:6\$1'.

3.6. Executing PROC\_MUX pass (convert decision trees to multiplexers).  
Creating decoders for process ``\counter.\$proc\$counter.v:6\$1'.  
1/1: \$0\out[2:0]

3.7. Executing PROC\_DLATCH pass (convert process syncs to latches).

3.8. Executing PROC\_DFF pass (convert process syncs to FFs).  
Creating register for signal ``\counter.\out' using process ``\counter.\$proc\$counter.v:6\$1'.  
created \$adff cell ``\$procdff\$5' with negative edge clock and positive level reset.

3.9. Executing PROC\_MEMWR pass (convert process memory writes to cells).

...



## Синтез цифровых схем: пакет yosys (4)

```
yosys> proc; memory; techmap;
```

4. Executing MEMORY pass.

4.1. Executing OPT\_MEM pass (optimize memories).  
Performed a total of 0 transformations.

4.2. Executing OPT\_MEM\_PRIORITY pass (removing unnecessary memory write priority relations).  
Performed a total of 0 transformations.

4.3. Executing OPT\_MEM\_FEEDBACK pass (finding memory read-to-write feedback paths).

4.4. Executing MEMORY\_DFF pass (merging \$dff cells to \$memrd).

4.5. Executing OPT\_CLEAN pass (remove unused cells and wires).  
Finding unused cells or wires in module \counter..  
Removed 0 unused cells and 2 unused wires.  
<suppressed ~1 debug messages>

4.6. Executing MEMORY\_SHARE pass (consolidating \$memrd/\$memwr cells).

4.7. Executing OPT\_MEM\_WIDEN pass (optimize memories where all ports are wide).  
Performed a total of 0 transformations.

4.8. Executing OPT\_CLEAN pass (remove unused cells and wires).  
Finding unused cells or wires in module \counter..

4.9. Executing MEMORY\_COLLECT pass (generating \$mem cells).

4.10. Executing MEMORY\_MAP pass (converting memories to logic and flip-flops).



## Синтез цифровых схем: пакет yosys (5)

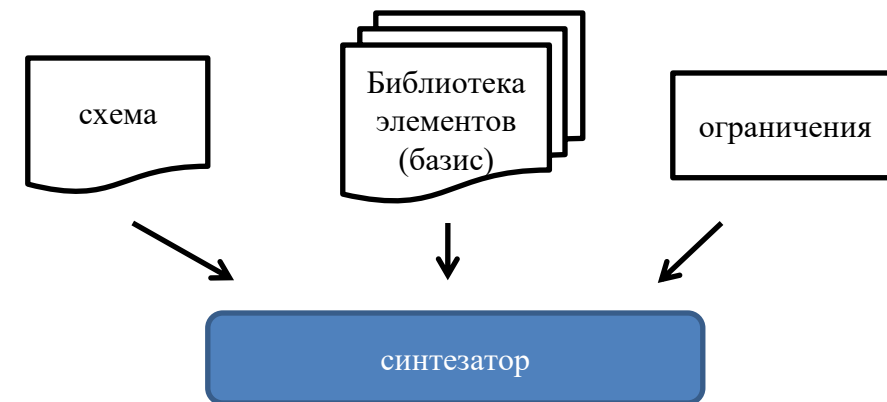
```
yosys> proc; memory; techmap;
```

5. Executing TECHMAP pass (map to technology primitives).

```
5.1. Executing Verilog-2005 frontend: /build/bin/./share/yosys/techmap.v  
Parsing Verilog input from `/build/bin/./share/yosys/techmap.v' to AST representation.  
Generating RTLIL representation for module `_90_simplemap_bool_ops'.  
Generating RTLIL representation for module `_90_simplemap_reduce_ops'.  
Generating RTLIL representation for module `_90_simplemap_logic_ops'.  
...  
Generating RTLIL representation for module `_90_fa'.  
Generating RTLIL representation for module `_90_lcu'.  
Generating RTLIL representation for module `_90_alu'.  
Generating RTLIL representation for module `_90_macc'.  
Generating RTLIL representation for module `_90_alumacc'.  
Generating RTLIL representation for module `$_div_mod_u'.  
Generating RTLIL representation for module `$_div_mod_trunc'.  
Generating RTLIL representation for module `_90_div'.  
Generating RTLIL representation for module `_90_mod'.  
Generating RTLIL representation for module `$_div_mod_floor'.  
Generating RTLIL representation for module `_90_divfloor'.  
Generating RTLIL representation for module `_90_modfloor'.  
Generating RTLIL representation for module `_90_pow'.  
Generating RTLIL representation for module `_90_pmux'.  
Generating RTLIL representation for module `_90_lut'.  
Successfully finished Verilog frontend.
```

5.2. Continuing TECHMAP pass.

```
...  
Using extmapper simplemap for cells of type $or.  
No more expansions possible.  
<suppressed ~224 debug messages>
```



## Синтез цифровых схем: пакет yosys (6)

```
yosys> dfflibmap -liberty cells.lib
```

```
6. Executing DFFLIBMAP pass (mapping DFF cells to sequential cells from liberty file).  
cell DFF (noninv, pins=4, area=18.00) is a direct match for cell type $_DFF_P_.  
cell DFFSR (noninv, pins=6, area=18.00) is a direct match for cell type $_DFFSR_PPP_.  
final dff cell mappings:
```

```
unmapped dff cell: $_DFF_N_  
\DFF _DFF_P_ (.CLK( C), .D( D), .Q( Q), .nQ(~Q));  
unmapped dff cell: $_DFF_NN0_  
unmapped dff cell: $_DFF_NN1_  
unmapped dff cell: $_DFF_NP0_  
unmapped dff cell: $_DFF_NP1_  
unmapped dff cell: $_DFF_PN0_  
unmapped dff cell: $_DFF_PN1_  
unmapped dff cell: $_DFF_PP0_  
unmapped dff cell: $_DFF_PP1_  
unmapped dff cell: $_DFFSR_NNN_  
unmapped dff cell: $_DFFSR_NNP_  
unmapped dff cell: $_DFFSR_NPN_  
unmapped dff cell: $_DFFSR_NPP_  
unmapped dff cell: $_DFFSR_PNN_  
unmapped dff cell: $_DFFSR_PNP_  
unmapped dff cell: $_DFFSR_PPN_  
\DFFSR _DFFSR_PPP_ (.CLK( C), .D( D), .Q( Q), .R( R), .S( S), .nQ(~Q));
```

```
6.1. Executing DFFLEGALIZE pass (convert FFs to types supported by the target).  
Mapping DFF cells in module `counter':  
mapped 3 $_DFFSR_PPP_ cells to \DFFSR cells.
```



## Синтез цифровых схем: пакет yosys (7)

```
yosys> abc -liberty cells.lib
```

7. Executing ABC pass (technology mapping using ABC).

7.1. Extracting gate netlist of module `counter' to ``<abc-temp-dir>/input.blif`'..  
Extracted 28 gates and 35 wires to a netlist network with 5 inputs and 6 outputs.

7.1.1. Executing ABC.

```
Running ABC command: <yosys-exe-dir>/yosys-abc -s -f <abc-temp-dir>/abc.script 2>&1
```

```
ABC: Parsing finished successfully. Parsing time = 0.00 sec
```

```
ABC: Warning: Templates are not defined.
```

```
ABC: Libery parser cannot read "time_unit". Assuming time_unit : "1ns".
```

```
ABC: Libery parser cannot read "capacitive_load_unit". Assuming capacitive_load_unit(1, pf).
```

```
ABC: Scl_LibertyReadGenlib() skipped sequential cell "DFF".
```

```
ABC: Scl_LibertyReadGenlib() skipped sequential cell "DFFSR".
```

```
ABC: Library "test" from "/openlane/test_design/cells.lib" has 4 cells (2 skipped: 2 seq; 0 tri-state;
```

```
0 no func; 0 dont_use). Time = 0.00 sec
```

```
ABC: Memory = 0.00 MB. Time = 0.00 sec
```

```
...
```

```
ABC: + write_blif <abc-temp-dir>/output.blif
```

7.1.2. Re-integrating ABC results.

```
ABC RESULTS: INV cells: 7
```

```
ABC RESULTS: NAND2 cells: 2
```

```
ABC RESULTS: NOR2 cells: 8
```

```
ABC RESULTS: internal signals: 24
```

```
ABC RESULTS: input signals: 5
```

```
ABC RESULTS: output signals: 6
```

```
Removing temp directory.
```



## Синтез цифровых схем: пакет yosys (8)

```
yosys> write_verilog synth.v
```

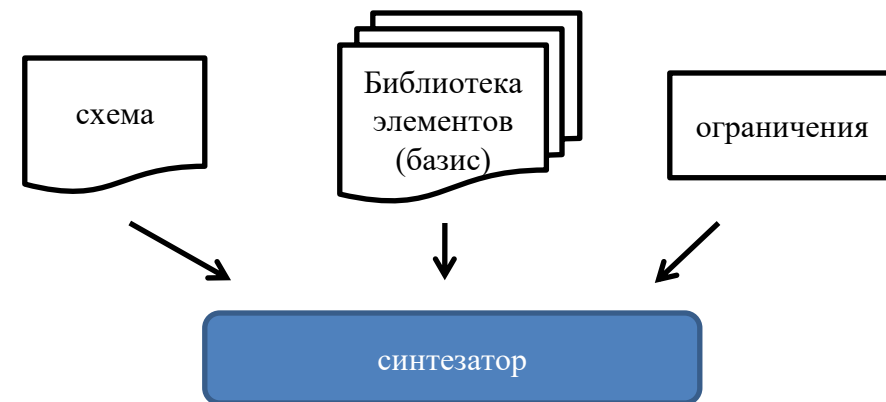
```
8. Executing Verilog backend.  
Dumping module ``counter'.
```



## Результат синтеза

```
input enable;  
(* src = "counter.v:4.19-4.22" *)  
output [2:0] out;  
(* src = "counter.v:3.13-3.18" *)  
input reset;  
INV _63_ (  
  .x(_21_),  
  .y(_09_)  
);  
INV _64_ (  
  .x(_20_),  
  .y(_10_)  
);  
INV _65_ (  
  .x(_07_),  
  .y(_04_)  
);  
INV _66_ (  
  .x(_08_),  
  .y(_11_)  
);  
NOR2 _67_ (  
  .x1(_10_),  
  .x2(_11_),  
  .y(_12_)  
);
```

```
INV _78_ (  
  .x(_07_),  
  .y(_05_)  
);  
INV _79_ (  
  .x(_07_),  
  .y(_06_)  
);  
(* src = "counter.v:6.2-11.25" *)  
DFFSR _80_ (  
  .CLK(_31_),  
  .D(_00_[0]),  
  .Q(out[0]),  
  .R(reset),  
  .S(1'h0),  
  .nQ(_28_)  
);  
(* src = "counter.v:6.2-11.25" *)  
DFFSR _81_ (  
  .CLK(_30_),  
  .D(_00_[1]),  
  .Q(out[1]),  
  .R(reset),  
  .S(1'h0),  
  .nQ(_29_)  
);
```



## Атрибуты в языке Verilog

Общий синтаксис:

```
(* <attribute_name> [= "attribute_value"] *)
```

Черта	Значение для атрибута	Значение для комментария
Синтаксис	(* ... *)	// ... /* ... */
Для кого используется	Средства САПР: <ul style="list-style-type: none"> <li>• синтезаторы</li> <li>• линтеры</li> <li>• симуляторы</li> </ul>	Цифровые разработчики
Назначение	Передача специфических указаний средствам САПР	Объяснение логики кода
Видимость	Является частью AST, разбирается, можно получить доступ через PLI	Полностью игнорируются, вырезаясь препроцессором до непосредственно компиляции
Расположение	Жёстко задано. предваряют конструкции, на которые распространяются	Не принципиально

## Файл заданий для yosys

```
Файл counter.yс:      read_verilog counter.v
                       read_verilog -lib cells.v

                       proc
                       memory
                       techmap

                       dfflibmap -liberty cells.lib
                       abc -liberty cells.lib

                       write_verilog synth.v

Запуск yosys:        yosys counter.yс
```

## Синтез тестовой схемы: зависимость от состава библиотеки (1)

```
module device(x1, x2, y);  
  input x1, x2;  
  output y;  
  wire a, b;  
  not i1(a, x2);  
  and a1(b, x1, a);  
  not i2(y, b);  
endmodule
```

```
module BUF(x, y);  
  ...  
endmodule
```

```
module INV(x, y);  
  ...  
endmodule
```

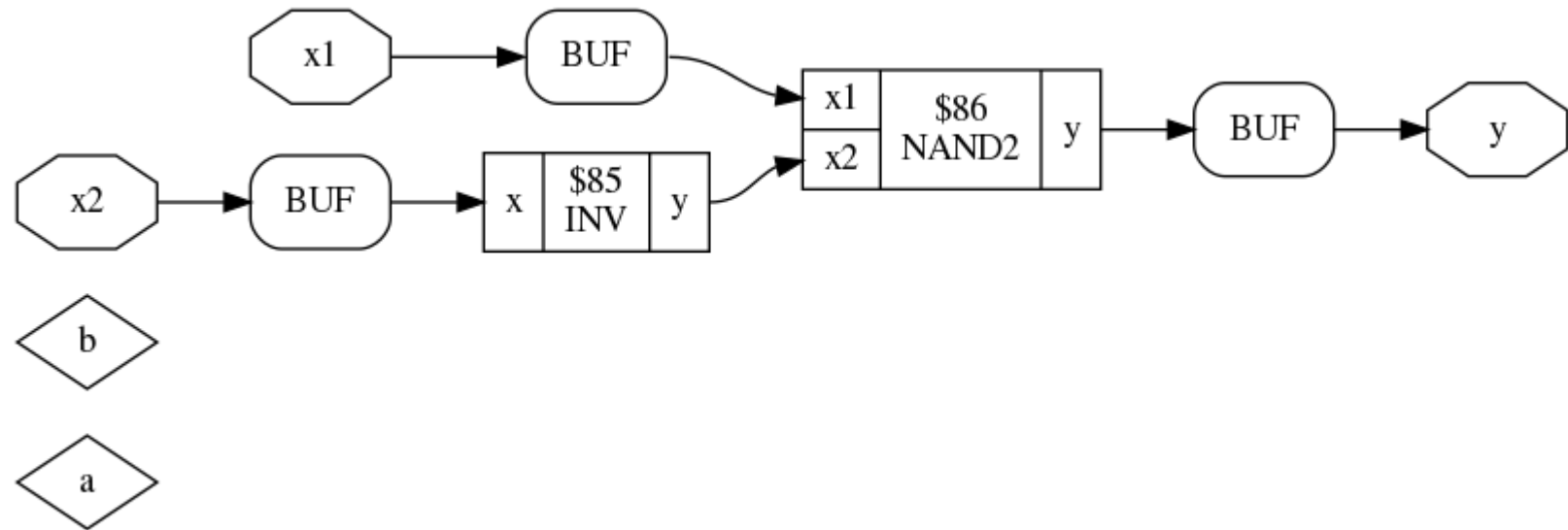
```
module NAND2(x1, x2, y);  
  ...  
endmodule
```

```
module NOR2(x1, x2, y);  
  ...  
endmodule
```

```
module device(x1, x2, y);  
  wire _0_, _1_, _2_, _3_, a, b;  
  input wire x1;  
  input wire x2;  
  output wire y;  
  
  INV _4_ (  
    .x(_2_),  
    .y(_0_)  
  );  
  NAND2 _5_ (  
    .x1(_1_),  
    .x2(_0_),  
    .y(_3_)  
  );  
  assign _1_ = x1;  
  assign _2_ = x2;  
  assign y = _3_;  
endmodule
```

## Синтез тестовой схемы: зависимость от состава библиотеки (2)

```
module device(x1, x2, y);  
  wire _0_, _1_, _2_, _3_, a, b;  
  input wire  x1;  
  input wire  x2;  
  output wire y;  
  
  INV _4_ (  
    .x(_2_),  
    .y(_0_)  
  );  
  NAND2 _5_ (  
    .x1(_1_),  
    .x2(_0_),  
    .y(_3_)  
  );  
  assign _1_ = x1;  
  assign _2_ = x2;  
  assign y = _3_;  
endmodule
```



device

## Синтез тестовой схемы: зависимость от состава библиотеки (3)

Файл cells.lib

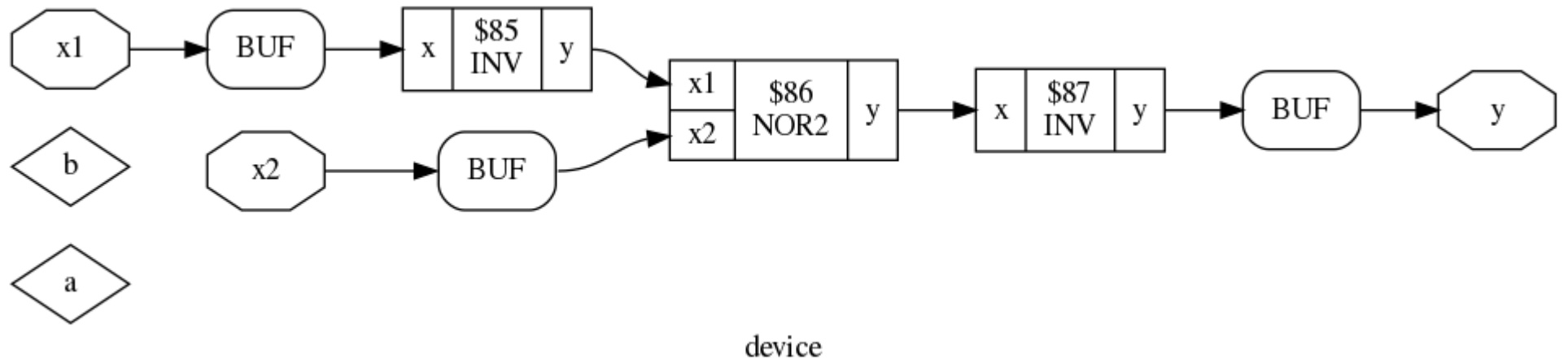
```
library (test) {  
  cell(BUF) {  
    ...  
  }  
  cell(INV) {  
    ...  
  }  
  cell(NAND2) {  
    ...  
  }  
  cell(NOR2) {  
    ...  
  }  
  cell(DFF) {  
    ...  
  }  
  cell(DFFSR) {  
    ...  
  }  
}
```

```
module BUF(x, y);  
  ...  
endmodule  
  
module INV(x, y);  
  ...  
endmodule  
  
module NAND2(x1, x2, y);  
  ...  
endmodule  
  
module NOR2(x1, x2, y);  
  ...  
endmodule
```

```
module device(x1, x2, y);  
  wire _0_, _1_, _2_, _3_, _4_, a, b;  
  input wire x1, x2;  
  output wire y;  
  INV _5_ (  
    .x(_2_),  
    .y(_0_)  
  );  
  NOR2 _6_ (  
    .x1(_0_),  
    .x2(_3_),  
    .y(_1_)  
  );  
  INV _7_ (  
    .x(_1_),  
    .y(_4_)  
  );  
  assign _2_ = x1;  
  assign _3_ = x2;  
  assign y = _4_;  
endmodule
```

## Синтез тестовой схемы: зависимость от состава библиотеки (4)

```
module device(x1, x2, y);  
  wire _0_, _1_, _2_, _3_, _4_, a, b;  
  input wire x1, x2;  
  output wire y;  
  INV _5_ (  
    .x(_2_),  
    .y(_0_)  
  );  
  NOR2 _6_ (  
    .x1(_0_),  
    .x2(_3_),  
    .y(_1_)  
  );  
  INV _7_ (  
    .x(_1_),  
    .y(_4_)  
  );  
  assign _2_ = x1;  
  assign _3_ = x2;  
  assign y = _4_;  
endmodule
```



## Место синтеза в открытом маршруте OpenLane

