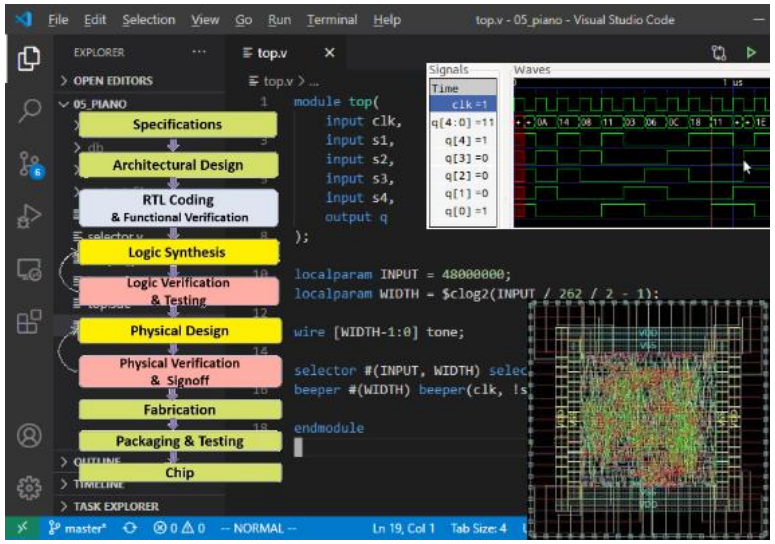
Национальный исследовательский университет «МИЭТ»

Институт интегральной электроники

Лингвистические средства проектирования



Лекция 7

Модели задержек.

Временное моделирование.

Логический синтез.

Использование параметров: глобальные (1)

```
module REG (D, EN, clk, R, Q);

    parameter WIDTH = 2;

    input      [WIDTH-1:0] D;
    input                  EN, clk, R;
    output reg [WIDTH-1:0] Q;

    always@(posedge clk, posedge R)
        if(EN == 1'b1)
            if(R == 1'b1)
                Q <= 0;
            else
                Q <= D;
endmodule
```

```
module tb_REG;
    reg  [3:0] D;
    reg  EN, clk, R;
    wire [3:0] Q;

    REG #(.WIDTH(4)) dut(.*);

    ...
```

Использование параметров: глобальные (2)

```
module REG (D, EN, clk, R, Q);

    parameter WIDTH = 2;

    input      [WIDTH-1:0] D;
    input                  EN, clk, R;
    output reg [WIDTH-1:0] Q;

    always@(posedge clk, posedge R)
        if(EN == 1'b1)
            if(R == 1'b1)
                Q <= 0;
            else
                Q <= D;
endmodule
```

```
module tb_REG;
    reg  [3:0] D;
    reg  EN, clk, R;
    wire [3:0] Q;

    REG dut(.*);

    defparam dut.WIDTH=4;

    ...
```

Опасность defparam (1)

```
module REG (D, EN, clk, R, Q);

    parameter WIDTH = 2;

    input      [WIDTH-1:0] D;
    input                  EN, clk, R;
    output reg [WIDTH-1:0] Q;

    always@(posedge clk, posedge R)
        if(EN == 1'b1)
            if(R == 1'b1)
                Q <= 0;
            else
                Q <= D;
endmodule
```

```
module tb_REG;
    reg  [3:0] D;
    reg  EN, clk, R;
    wire [3:0] Q;

    defparam dut.WIDTH=4;

    REG dut(.*);

    defparam dut.WIDTH=8;

    ...
```

Опасность defparam (2)

```
module REG (D, EN, clk, R, Q);

    parameter WIDTH = 2;

    input      [WIDTH-1:0] D;
    input                  EN, clk, R;
    output reg [WIDTH-1:0] Q;

    always@(posedge clk, posedge R)
        if(EN == 1'b1)
            if(R == 1'b1)
                Q <= 0;
            else
                Q <= D;
endmodule
```

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

  defparam REG.WIDTH=8;

endmodule
```

```
module tb_REG;
    reg  [3:0] D;
    reg  EN, clk, R;
    wire [3:0] Q;

    defparam dut.WIDTH=4;

    REG dut(.*);

    ...
```

Использование параметров: локальные

```
module tb_REG;
    reg  [3:0] D;
    reg  EN, clk, R;
    wire [3:0] Q;

    REG #(.WIDTH(4)) dut(.*);

    ...
```

```
module tb_REG;
    reg  [7:0] D;
    reg  EN, clk, R;
    wire [7:0] Q;

    REG #(.WIDTH(8)) dut(.*);

    ...
```

```
module tb_REG;

    localparam SIZE = 4;

    reg  [SIZE-1:0] D;
    reg  EN, clk, R;
    wire [SIZE-1:0] Q;

    REG #(.WIDTH(SIZE)) dut(.*);

    ...
```

Использование макроопределений

```
`define PARAM_VALUE 4

module REG (D, EN, clk, R, Q);

    input      [`PARAM_VALUE-1:0] D;
    input                         EN, clk, R;
    output reg [`PARAM_VALUE-1:0] Q;

    always@(posedge clk, posedge R)
        if(EN == 1'b1)
            if(R == 1'b1)
                Q <= 0;
            else
                Q <= D;
endmodule
```

```
module tb_REG;
    reg  [`PARAM_VALUE-1:0] D;
    reg  EN, clk, R;
    wire [`PARAM_VALUE-1:0] Q;

    REG dut(.*);

    ...
```

Использование параметров через аргументы командной строки (1)

```
module REG (D, EN, clk, R, Q);

    input      [`PARAM_VALUE-1:0] D;
    input                         EN, clk, R;
    output reg [`PARAM_VALUE-1:0] Q;

    always@(posedge clk, posedge R)
        if(EN == 1'b1)
            if(R == 1'b1)
                Q <= 0;
            else
                Q <= D;
endmodule
```

```
module tb_REG;
    reg  [`PARAM_VALUE-1:0] D;
    reg  EN, clk, R;
    wire [`PARAM_VALUE-1:0] Q;

    REG dut(.*);

    ...
```

```
> iverilog -DPARAM_VALUE=8 parametrized.v
```

Использование параметров через аргументы командной строки (2)

```
module REG (D, EN, clk, R, Q);

    parameter WIDTH = 2;

    input      [WIDTH-1:0] D;
    input                  EN, clk, R;
    output reg [WIDTH-1:0] Q;

    always@(posedge clk, posedge R)
        if(EN == 1'b1)
            if(R == 1'b1)
                Q <= 0;
            else
                Q <= D;
endmodule
```

```
module tb_REG;

    parameter SIZE = 2;

    reg  [SIZE-1:0] D;
    reg  EN, clk, R;
    wire [SIZE-1:0] Q;

    REG #(SIZE) dut(.*);
    ...
```

```
> iverilog -P tb_REG.SIZE=8 parametrized.v
```

Выбор моделируемого тестового окружения через аргументы командной строки

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

`timescale 1ns/1ns
module tb_inv;
  reg x = 0;
  wire y;
  inv dut(x, y);
  initial begin
    $dumpfile("inv.vcd");
    $dumpvars(1, tb_inv);
    #100 $finish;
  end
  always #10 x = ~x;
endmodule

> iverilog -s tb_inv test.v
```

```
module and2 (x1, x2, y);
  input x1, x2;
  output y;
  assign y = x1 & x2;
endmodule

`timescale 1ns/1ns
module tb_and2;
  reg x1 = 0, x2 = 0;
  wire y;
  and2 dut(.*);
  initial begin
    $dumpfile("and2.vcd");
    $dumpvars(1, tb_and2);
    #100 $finish;
  end
  always #10 x1 = ~x1;
  always #20 x2 = ~x2;
endmodule
```

Аргументы командной строки: вывод результатов препроцессора

Файл inv.v

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

Файл and2.v

```
module and2 (x1, x2, y);
  input x1, x2;
  output y;
  assign y = x1 & x2;
endmodule
```

```
iverilog -E -oall.v tb.v
```

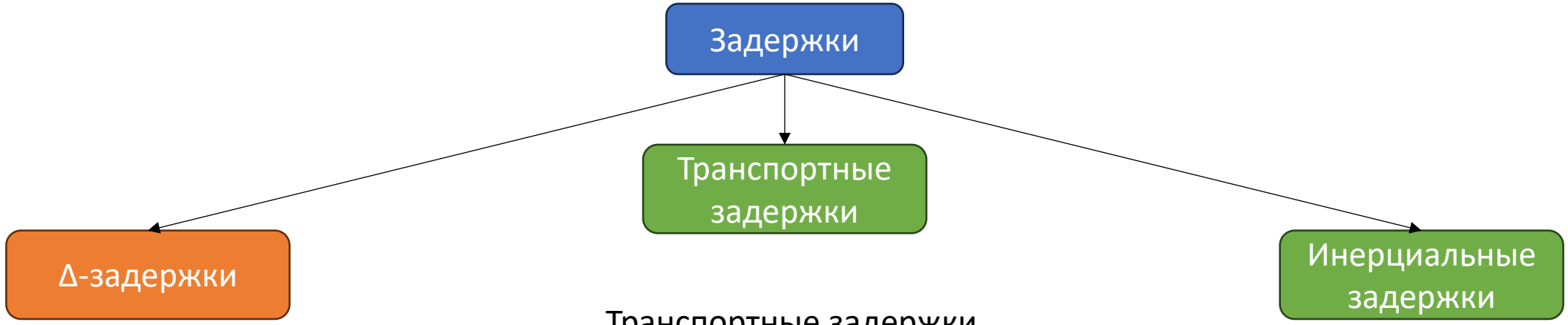
Файл design.v

```
`include "inv.v"
`include "and2.v"

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(y));
  inv i2(.x(s2), .y(y));
endmodule
```

Файл tb.v

```
`include "design.v"

`timescale 1ns/1ns
module tb_device;
  reg x1, x2;
  wire y;

  device i1(.*);

  always #5  x1 = ~x1;
  always #10 x2 = ~x2;

  initial begin
    $dumpfile("device.vcd");
    $dumpvars(0, tb_device);
    {x1, x2} = 0;
    #100 $finish;
end

endmodule
```

Работа со временем: задержки



Обусловлены алгоритмом моделирования, мы никак не можем на них повлиять

Транспортные задержки моделируют задержку, связанную с распространением сигнала в межсоединениях.

Транспортные задержки обеспечивают сдвиг сигнала вне зависимости от длительности переключения.

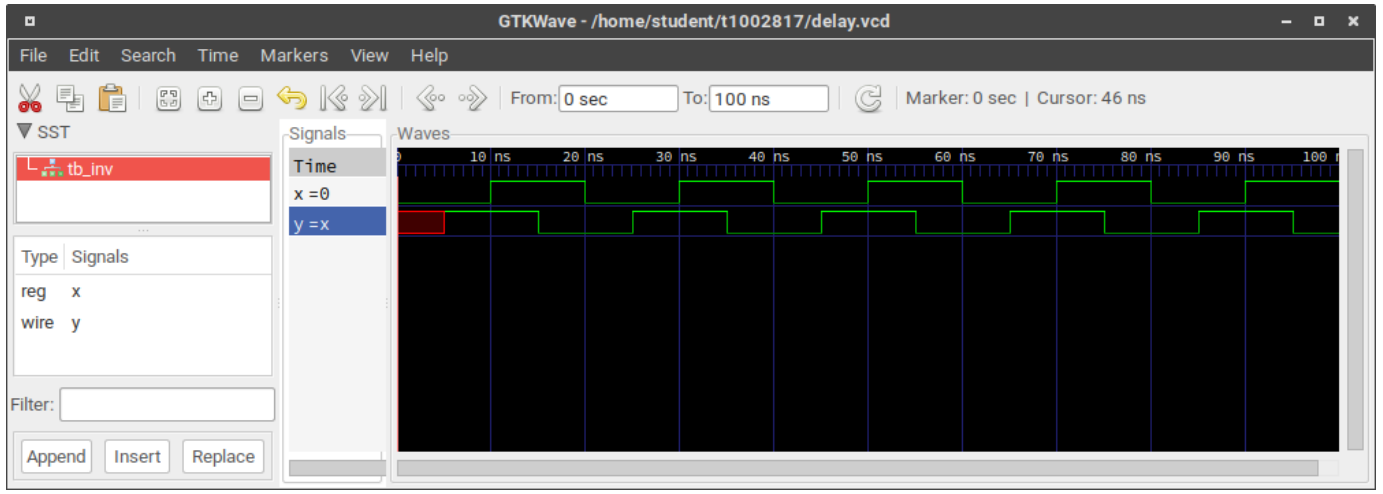
Инерционные задержки моделируют внутреннюю задержку срабатывания логических вентилей.

Элемент не срабатывает, если время меньше указанной инерциальной задержки.

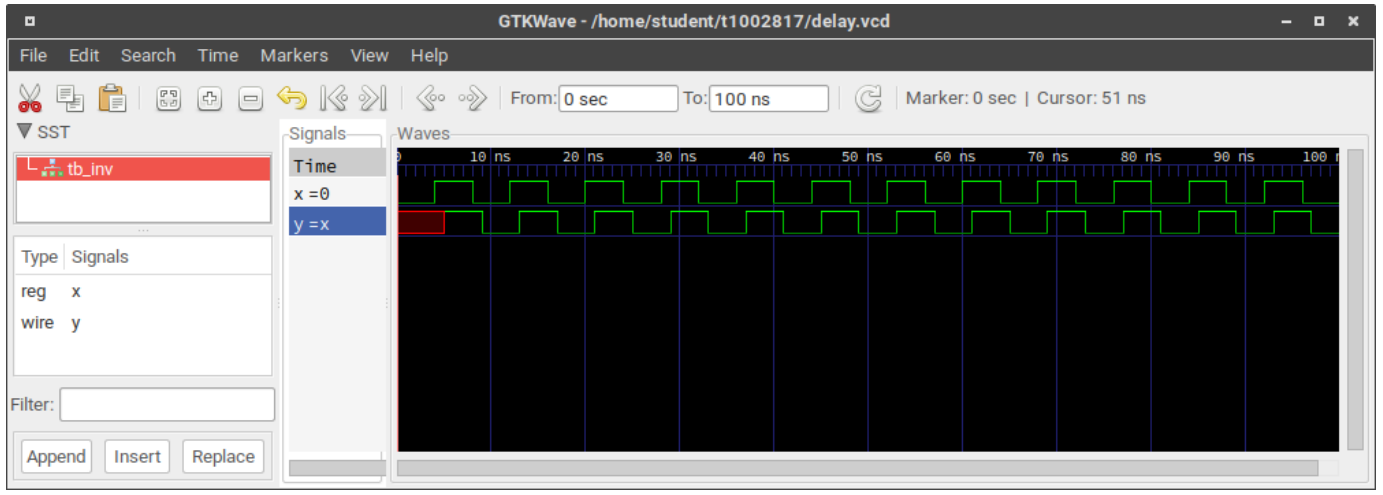
Транспортные задержки

```
`timescale 1ns/1ns
module inv (x, y);
  input x;
  output reg y;

  always @*
    y <= #5 ~x;

endmodule
```

Переключение через каждые 10нс (больше указанной задержки)



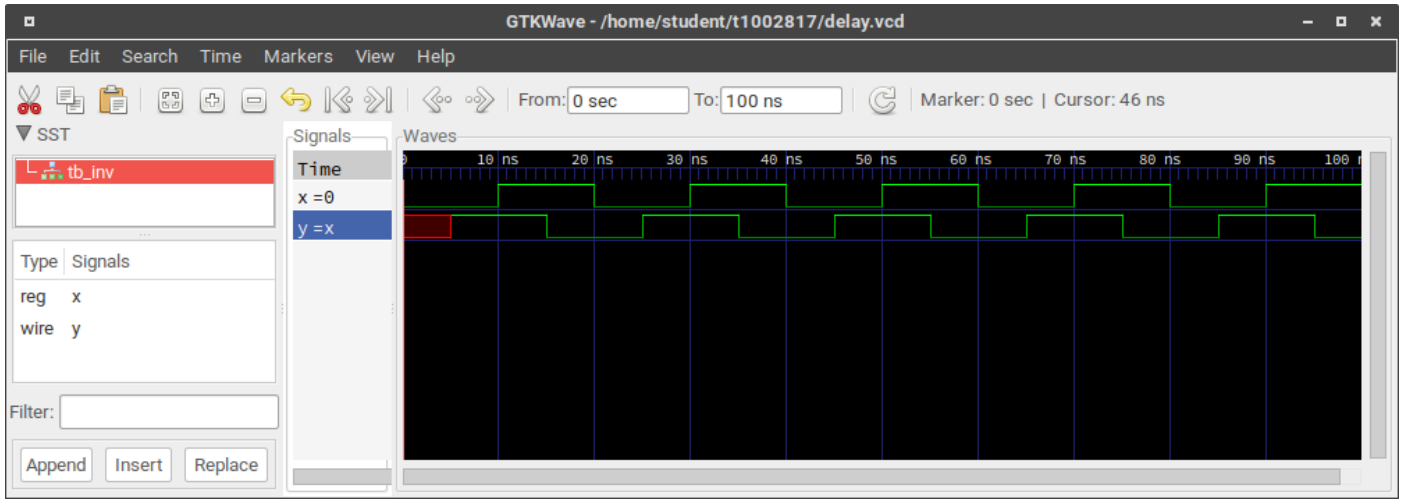
Переключение через каждые 4нс (меньше указанной задержки)



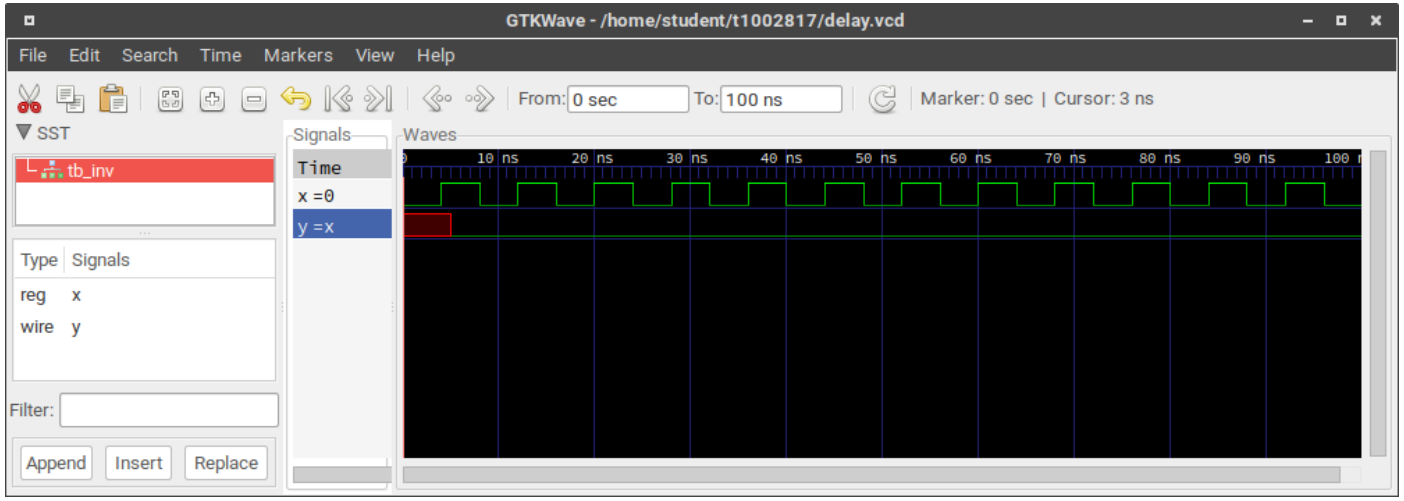
Инерциальные задержки внутри вентиля (1)

```
`timescale 1ns/1ns
module inv (x, y);
  input x;
  output reg y;

  always @*
    #5 y <= ~x;

endmodule
```

Переключение через каждые 10нс (больше указанной задержки)

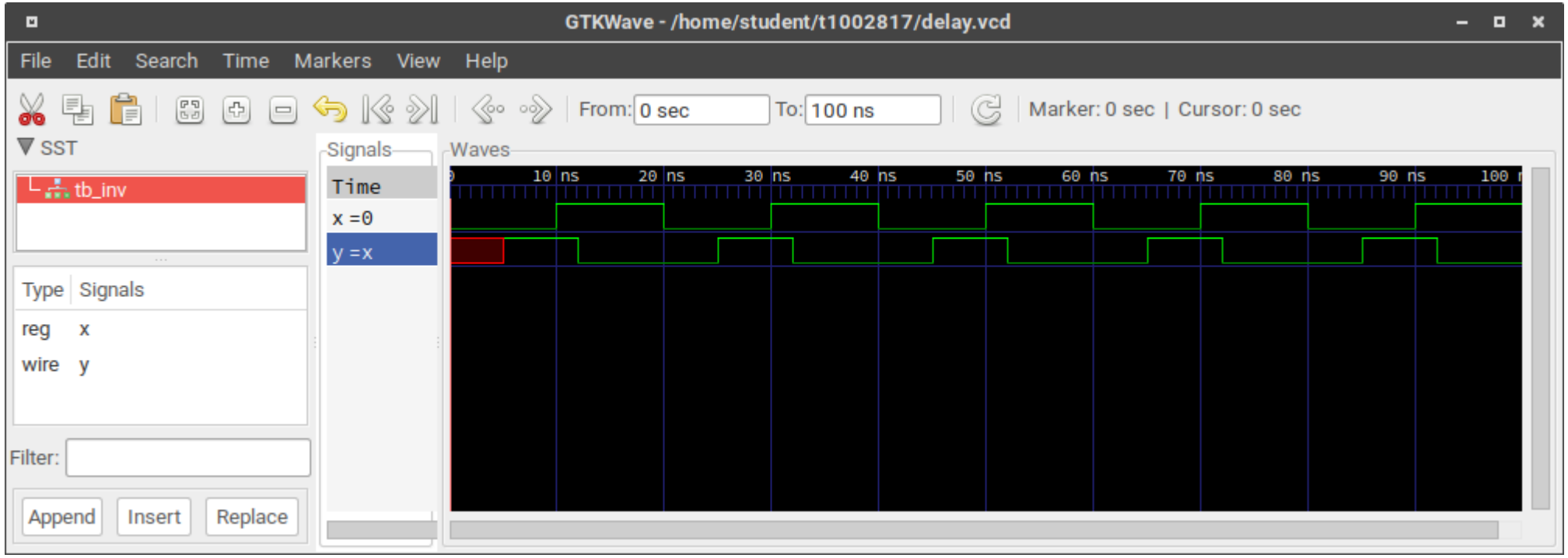


Переключение через каждые 4нс (меньше указанной задержки)



Инерциальные задержки внутри примитива

```
not #(5) dut(y, x);

not #(5, 2) dut(y, x);
```



Минимальное, типичное, максимальное значения задержек

```
`timescale 1ns/1ns
module inv (x, y);
  input x;
  output reg y;

  always @*
    y <= #5 ~x;

endmodule
```

```
y <= #(4:5:6) ~x;
```

```
student@localhost:~/t1002817> iverilog -s tb_inv delay.v
delay.v:7: warning: choosing typ expression.
student@localhost:~/t1002817>
```

```
> iverilog -Tmin test.v
```

```
student@localhost:~/t1002817> iverilog -s tb_inv -Tmin delay.v
student@localhost:~/t1002817>
```

Инерциальные задержки внутри многовходового вентиля

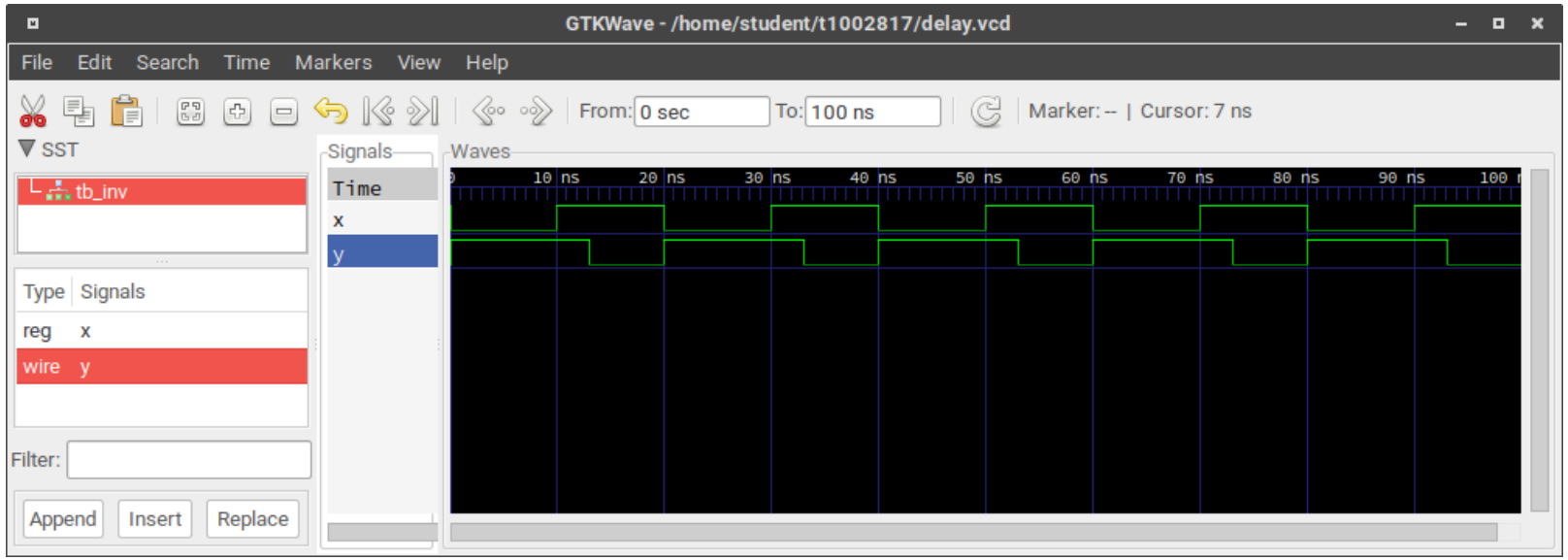
```
module device(x1, x2, y);
  input x1, x2;
  output y;

  wire s1, s2;

  specify
    if  (x1) (x1 => y) = 2;
    if ~(x1) (x1 => y) = 3;
    if  (x2) (x2 => y) = 4;
    if ~(x2) (x2 => y) = 5;
  endspecify

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

endmodule
```

icarus does not yet support specify blocks in general

Инерциальные задержки внутри многовходового вентиля (3)

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

  specify
    if(x) (x => y) = 3;
  endspecify

  not (y, x);

endmodule
```

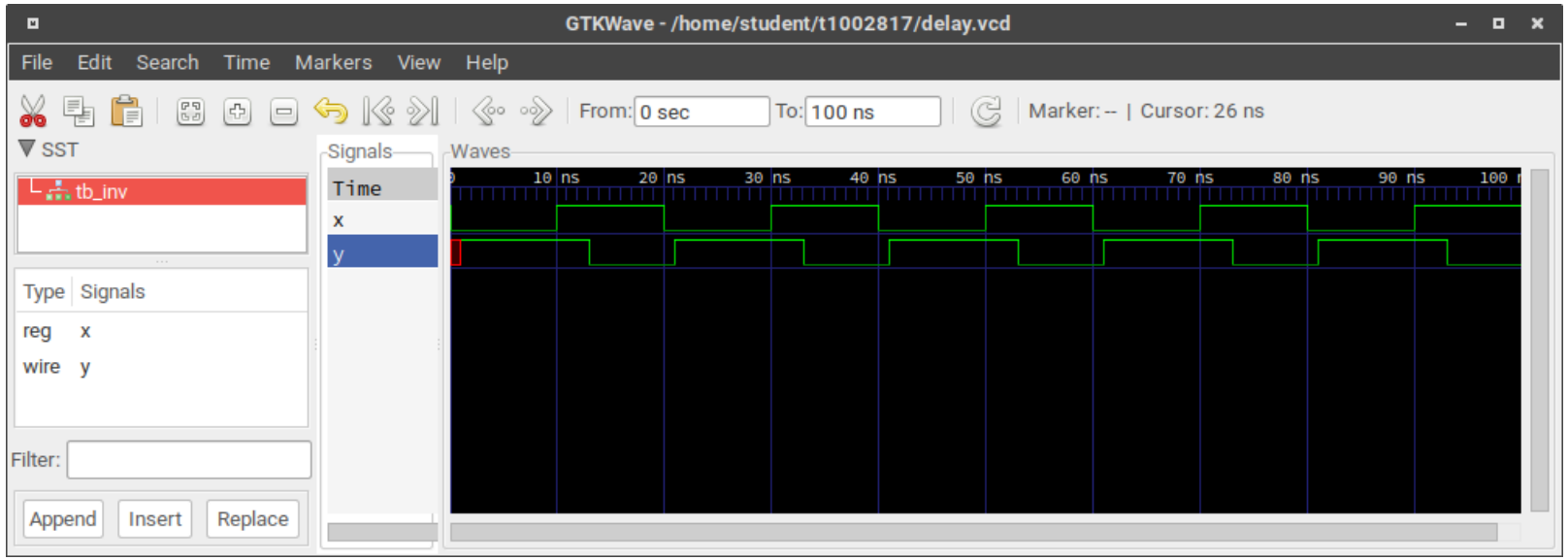
```
iverilog -gspecify test.v
```



Инерциальные задержки внутри многовходового вентиля (4)

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

  specify
    if  (x) (x => y) = 3;
    if (~x) (x => y) = 1;
  endspecify

  not (y, x);

endmodule
```

```
iverilog -gspecify test.v
```



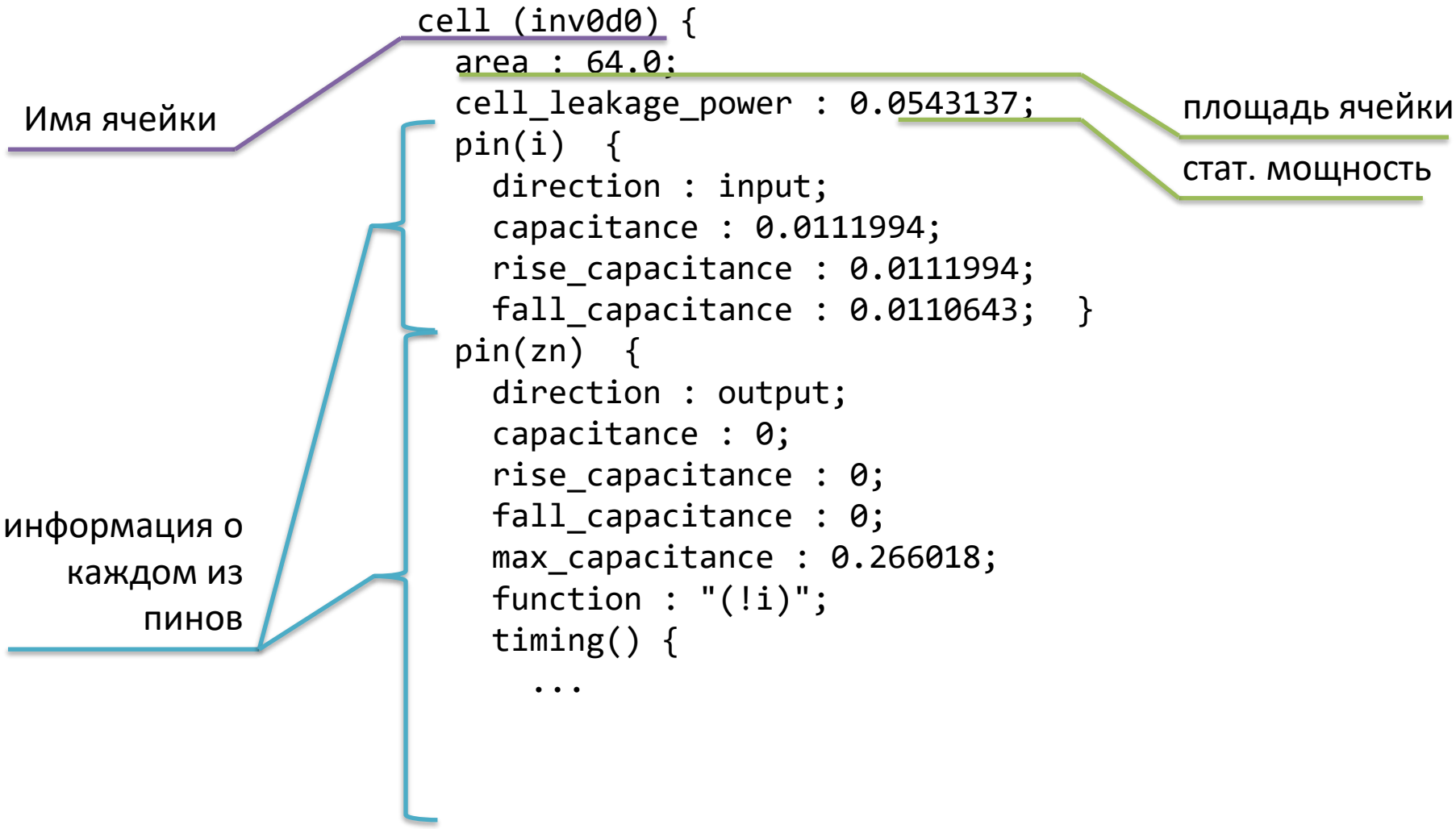
Стандартный формат описания задержек (SDF)

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

  assign y = ~x;
endmodule
```

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

  inv dut(x, y);

  initial begin
    $dumpfile("delay.vcd");
    $dumpvars(1, tb_inv);
    $sdf_annotate("inv_sdf.sdf", tb_inv);
    x = 0;
    #100 $finish;
  end

  always #10 x = ~x;
endmodule
```

```
(DELAYFILE
    (SDFVERSION "3.0")
    (DESIGN "inv_sdf")
    (DATE "Mon Mar 18 16:11:43 2024")
    (PROGRAM "STA")
    (VERSION "2.4.0")
    (DIVIDER .)
    (TIMESCALE 1ns)
    (CELL
        (CELLTYPE "inv")
        (INSTANCE dut)
        (DELAY
            (ABSOLUTE
                (IOPATH x y (3:4:5) (2:3:4))
            )
        )
    )
)
```

Результат характеризации: формат Synopsys Liberty (1)




```
cell (inv0d0) {
  area : 64.0;
  cell_leakage_power : 0.0543137;
  pin(i)  {
    direction : input;
    capacitance : 0.0111994;
    rise_capacitance : 0.0111994;
    fall_capacitance : 0.0110643;  }
  pin(zn)  {
    direction : output;
    capacitance : 0;
    rise_capacitance : 0;
    fall_capacitance : 0;
    max_capacitance : 0.266018;
    function : "(!i)";
    timing() {
      ...
```

Результат характеризации: формат Synopsys Liberty (2)

От какого пина смотрим зависимость

«Форма» реакции

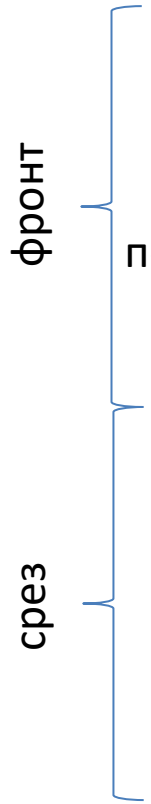
Время нарастания фронта

Выходная нагрузка

значения задержки

```
timing() {
     related_pin : "i";
     timing_sense : negative_unate;
     cell_rise(delay_template_6x6) {
       index_1 ("0.016, 0.032, 0.064, 0.128, 0.256, 0.512");
       index_2 ("0.08, 0.16, 0.32, 0.48, 0.64, 0.8");
       values ( \
         "0.055434, 0.069821, 0.082992, 0.096737, 0.108544, 0.114477", \
         "0.089193, 0.096673, 0.131445, 0.153615, 0.170875, 0.184564", \
         "0.162878, 0.178567, 0.197923, 0.243968, 0.275384, 0.301804", \
         "0.358053, 0.362199, 0.386151, 0.415653, 0.429929, 0.490506", \
         "0.811713, 0.819202, 0.832069, 0.867784, 0.894318, 0.91146", \
         "1.81723, 1.82331, 1.83447, 1.86957, 1.89758, 1.90694");
     }
```

Результат характеризации: формат Synopsys Liberty (3)

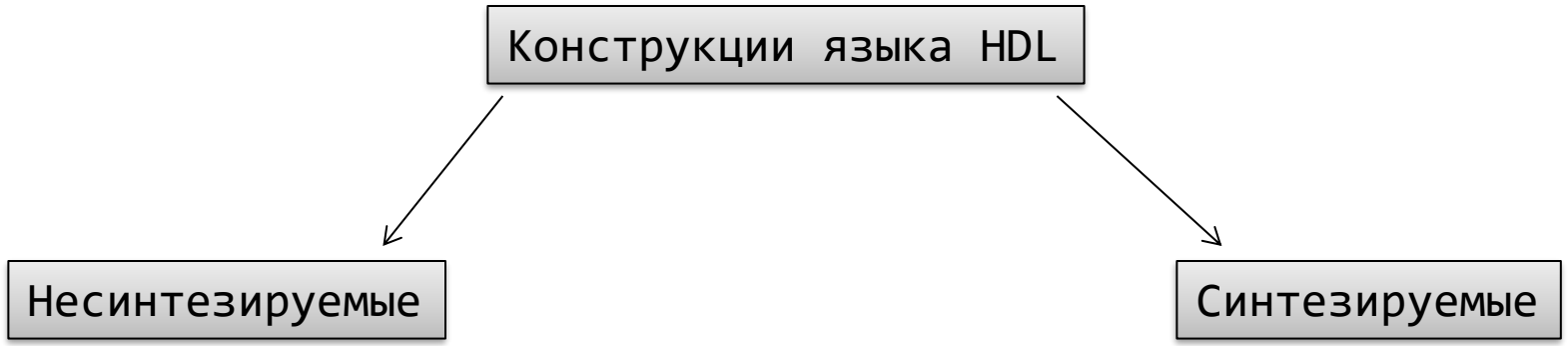


задержка 50%-50%

время переключения выхода 20%-80%

```
timing() {
     related_pin : "i";
     timing_sense : negative_unate;
     cell_rise(delay_template_6x6) {
       index_1 ("0.016, 0.032, 0.064, 0.128, 0.256, 0.512");
       ...
     }
     rise_transition(delay_template_6x6) {
       index_1 ("0.016, 0.032, 0.064, 0.128, 0.256, 0.512");
       ...
     }
     cell_fall(delay_template_6x6) {
       index_1 ("0.016, 0.032, 0.064, 0.128, 0.256, 0.512");
       ...
     }
     fall_transition(delay_template_6x6) {
       index_1 ("0.016, 0.032, 0.064, 0.128, 0.256, 0.512");
       ...
     }
```

Цифровой синтез



Возможно проведение функциональной и временной верификации.

Для получения топологии нужно:

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

Возможно проведение функциональной и временной верификации, автоматического синтеза схемотехнического представления и топологии .

Задача цифрового синтеза

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

  always @(x1 or x2)
    if (x1 == 1 && x2 == 0)
      y <= 0;
    else
      y <= 1;

endmodule
```

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

  wire a, b;

  LIB_NOT i1(x2, a);
  LIB_AND a1(x1, a, b);
  LIB_NOT i2(b, y);

endmodule
```

Синтез схем в OpenLane: spm (1)

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

```
module spm #(parameter bits=32) ( input clk, input rst, input x, input[bits-1: 0] a, output y);
  wire[bits: 0] y_chain;
  assign y_chain[0] = 0;
  assign y = y_chain[bits];

  wire[bits-1:0] a_flip;
  generate
    for (genvar i = 0; i < bits; i = i + 1) begin : flip_block
      assign a_flip[i] = a[bits - i - 1];
    end
  endgenerate

  delayed_serial_adder dsa[bits-1:0](.clk(clk), .rst(rst), .x(x), .a(a_flip),
                                     .y_in(y_chain[bits-1:0]), .y_out(y_chain[bits:1]));

endmodule
```

Синтез схем в OpenLane: spm (2)

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

```
/* Generated by Yosys 0.34
(git sha1 4a1b5599258, gcc 8.3.1 -fPIC -Os) */

module spm(clk, rst, x, a, y);
  wire _000_;
  wire _001_;
  wire _002_;
  wire _003_;
  wire _004_;
  wire _005_;
  wire _006_;
  wire _007_;
  wire _008_;
  wire _009_;
  wire _010_;
  wire _011_;
  wire _012_;
  wire _013_;
  wire _014_;
  ...
```

```
sky130_fd_sc_hd__buf_1 _102_ (
  .A(_000_),
  .X(_030_)
);
sky130_fd_sc_hd__a21o_2 _103_ (
  .A1(_030_),
  .A2(a[21]),
  .B1(\dsa[10].last_carry ),
  .X(_031_)
);
sky130_fd_sc_hd__nand3_2 _104_ (
  .A(_009_),
  .B(a[21]),
  .C(\dsa[10].last_carry ),
  .Y(_032_)
);
sky130_fd_sc_hd__nand2_2 _105_ (
  .A(_031_),
  .B(_032_),
  .Y(_033_)
);
```

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.