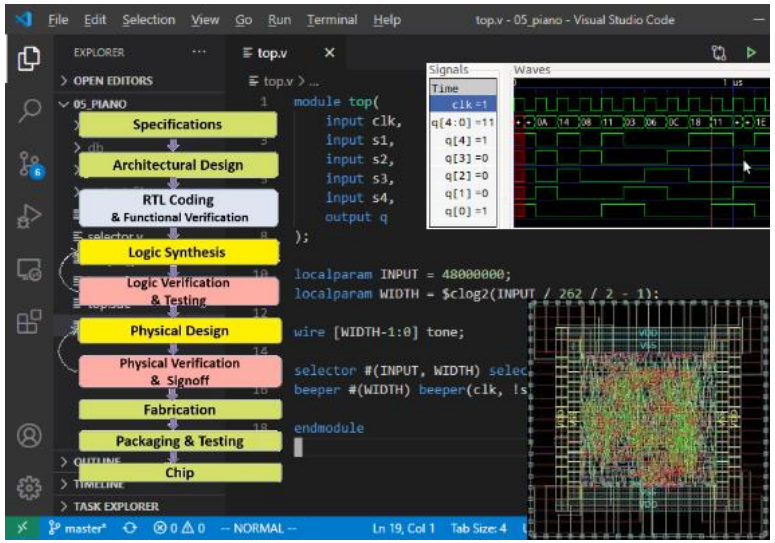
МИЭТ
Национальный исследовательский университет «МИЭТ»
Институт интегральной электроники

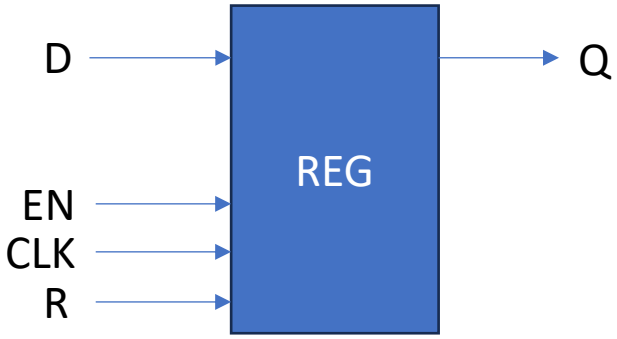
Лингвистические средства проектирования



Лекция 5

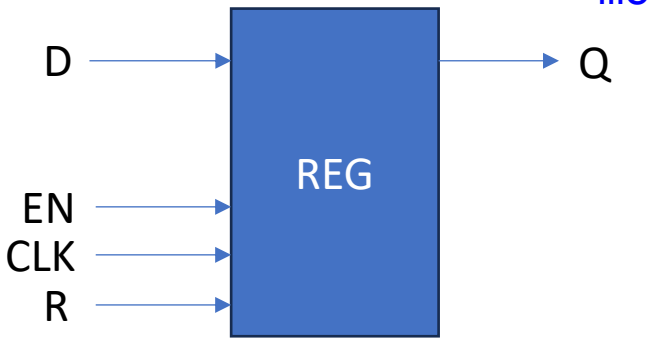
Параметризованное описание модулей.
Алгоритмы моделирования.
VPI и библиотека SystemC.

Параметризованное описание модулей (1)

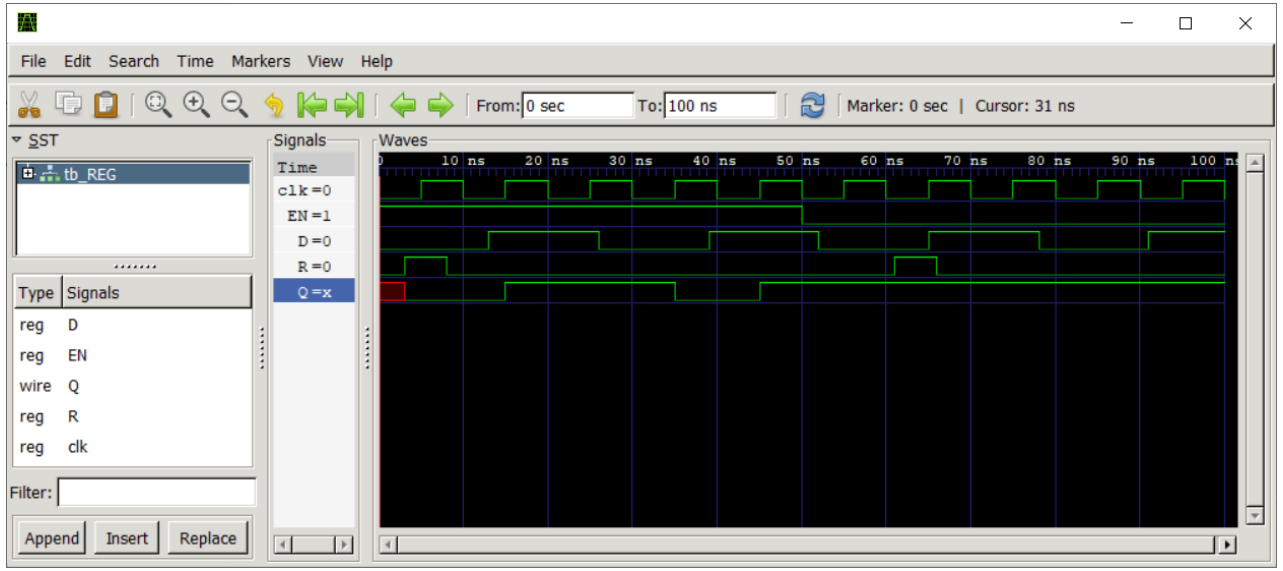



```
module REG(D, EN, clk, R, Q);
    input      D, EN, clk, R;
    output reg Q;

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

Параметризованное описание модулей (2)

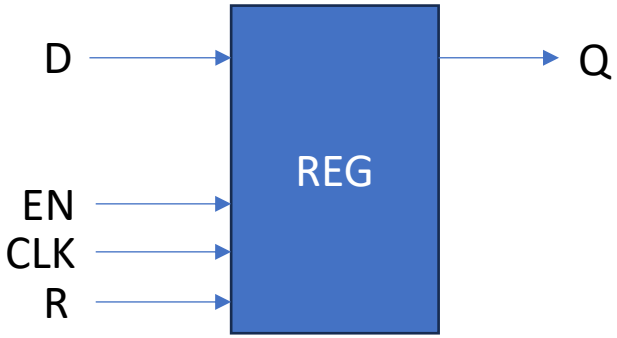



```
module tb_REG;
    reg  D;
    reg  EN, clk, R;
    wire Q;

    REG dut(.*);

    initial begin
        {D, clk, R} = 0;
        EN = 1;
        $dumpfile("REG.vcd");
        $dumpvars(0, tb_REG);
        #100
        $finish;
    end

    always #5  clk = ~clk;
    always #13 D   = ~D;
    ...
endmodule
```



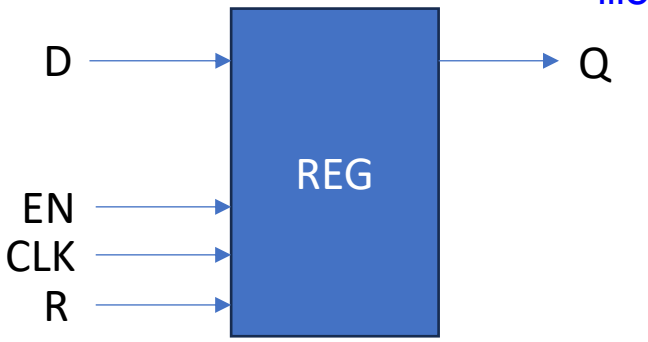

```
initial begin
    #3      R = 1'b1;
    #5      R = 1'b0;
    #53     R = 1'b1;
    #5      R = 1'b0;
end

initial begin
    #50     EN = 1'b0;
end
```

Параметризованное описание модулей (3)

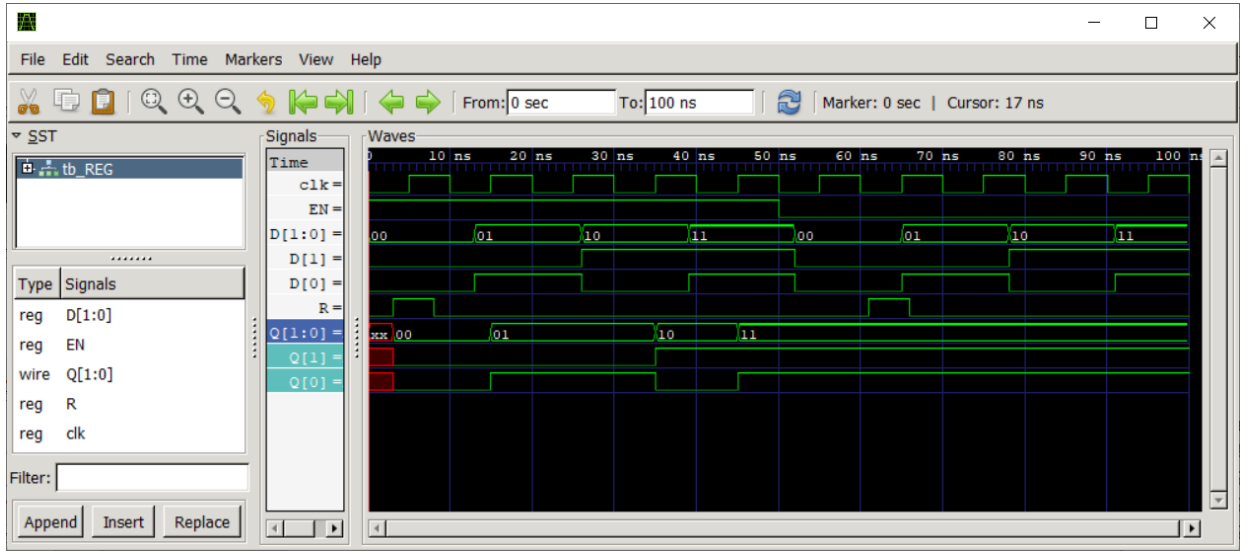



```
module REG(D, EN, clk, R, Q);
    input      [1:0] D;
    input            EN, clk, R;
    output reg [1:0] Q;

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

Параметризованное описание модулей (4)

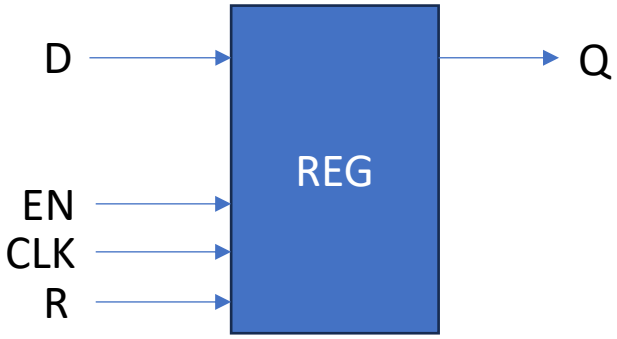



```
module tb_REG;
    reg  [1:0] D;
    reg  EN, clk, R;
    wire [1:0] Q;

    REG dut(.*);

    initial begin
        {D, clk, R} = 0;
        EN = 1;
        $dumpfile("REG.vcd");
        $dumpvars(0, tb_REG);
        #100
        $finish;
    end

    always #5  clk = ~clk;
    always #13 D   = D + 1;
    ...
endmodule
```



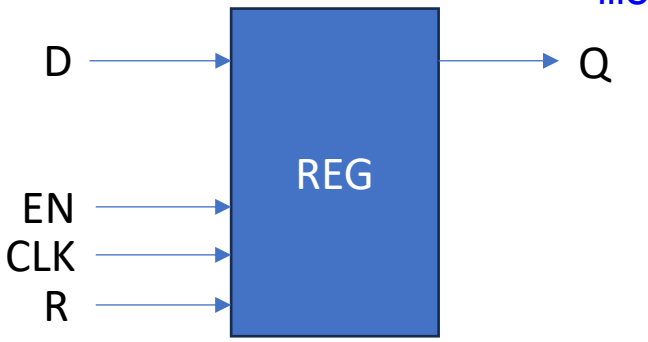

```
initial begin
    #3      R = 1'b1;
    #5      R = 1'b0;
    #53     R = 1'b1;
    #5      R = 1'b0;
end

initial begin
    #50     EN = 1'b0;
end
```

Параметризованное описание модулей (5)

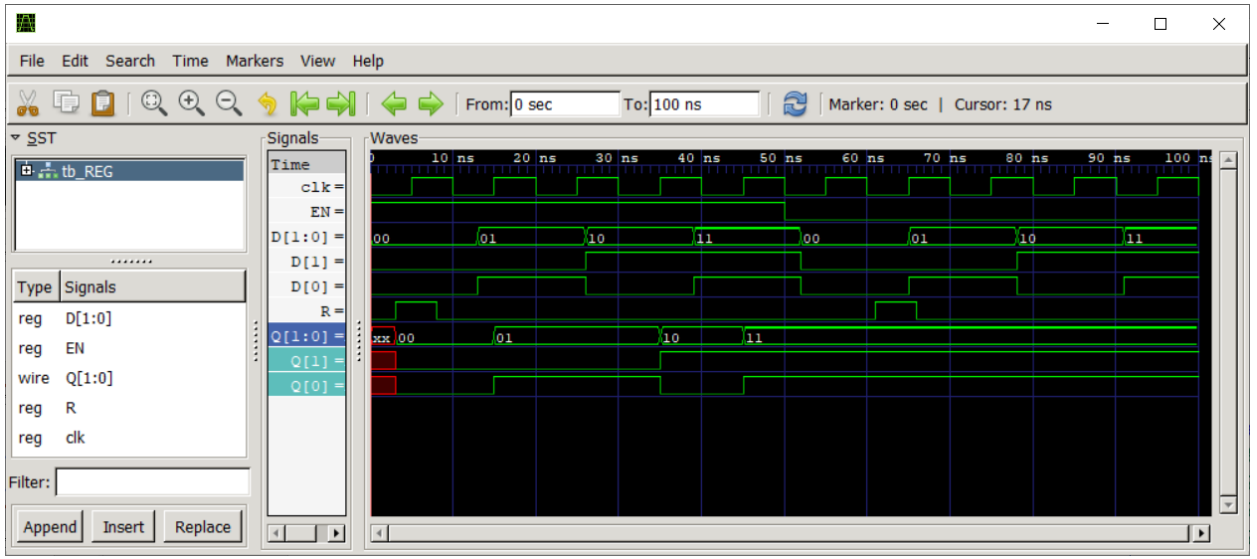



```
module REG #(parameter WIDTH = 2) (D, EN, clk, R, Q);
    input       [WIDTH-1:0] D;
    input                   EN, clk, R;
    output reg  [WIDTH-1:0] Q;

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

Параметризованное описание модулей (6)

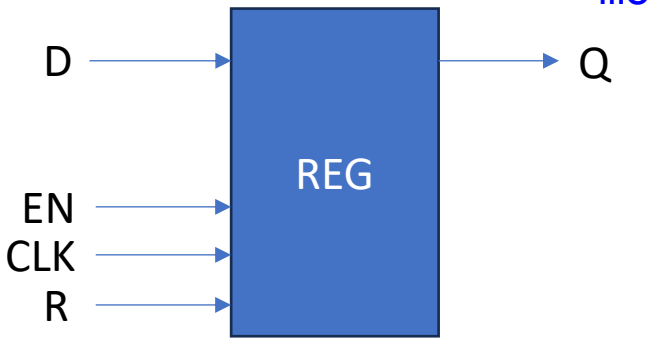



```
module tb_REG;
    reg  [1:0] D;
    reg  EN, clk, R;
    wire [1:0] Q;

    REG dut(.*);

    initial begin
        {D, clk, R} = 0;
        EN = 1;
        $dumpfile("REG.vcd");
        $dumpvars(0, tb_REG);
        #100
        $finish;
    end

    always #5  clk = ~clk;
    always #13 D   = D + 1;
    ...
endmodule
```

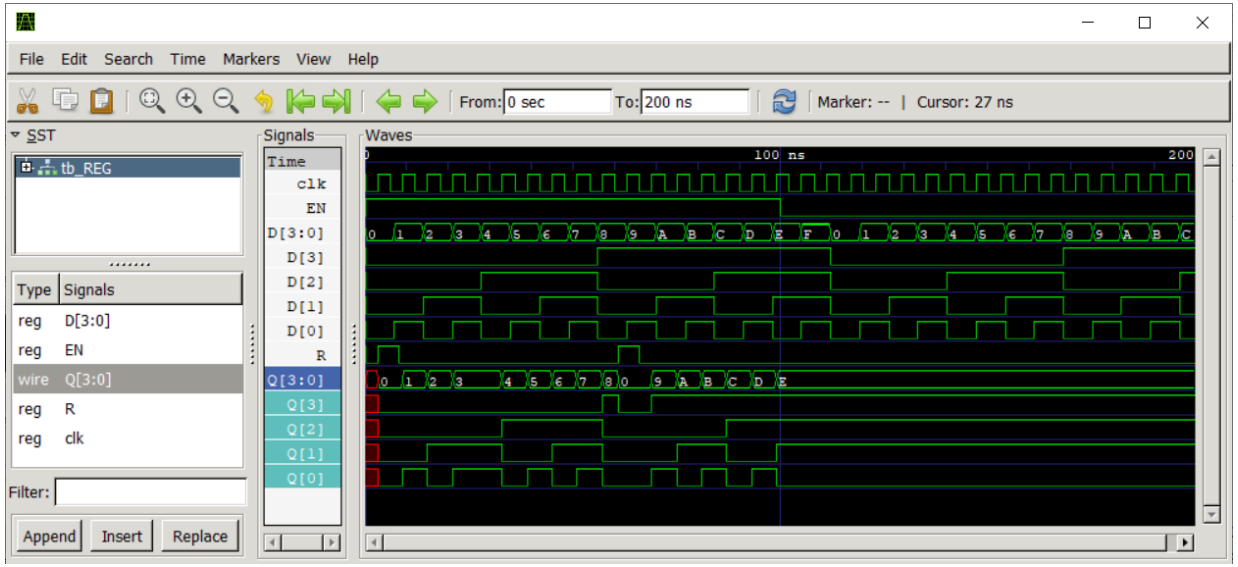



```
initial begin
    #3      R = 1'b1;
    #5      R = 1'b0;
    #53     R = 1'b1;
    #5      R = 1'b0;
end

initial begin
    #50     EN = 1'b0;
end
```

Параметризованное описание модулей (7)

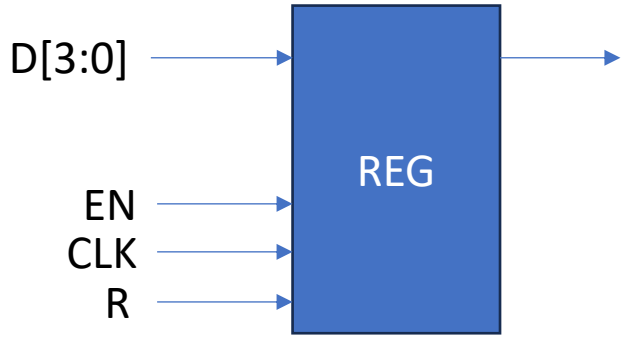



```
module tb_REG;
    reg  [3:0] D;
    reg  EN, clk, R;
    wire [3:0] Q;

    REG #(4) dut(.*);

    initial begin
        {D, clk, R} = 0;
        EN = 1;
        $dumpfile("REG.vcd");
        $dumpvars(0, tb_REG);
        #200
        $finish;
    end

    always #3  clk = ~clk;
    always #7 D  = D + 1;
    ...
endmodule
```




```
initial begin
    #3      R = 1'b1;
    #5      R = 1'b0;
    #53     R = 1'b1;
    #5      R = 1'b0;
end

initial begin
    #100    EN = 1'b0;
end
```

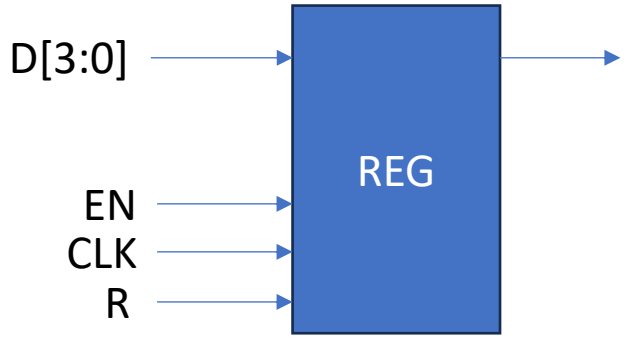
Параметризованное описание модулей (8)




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

    REG #(4) dut(.*);

    ...
```

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

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

    ...
```

Параметризованное описание модулей (9)

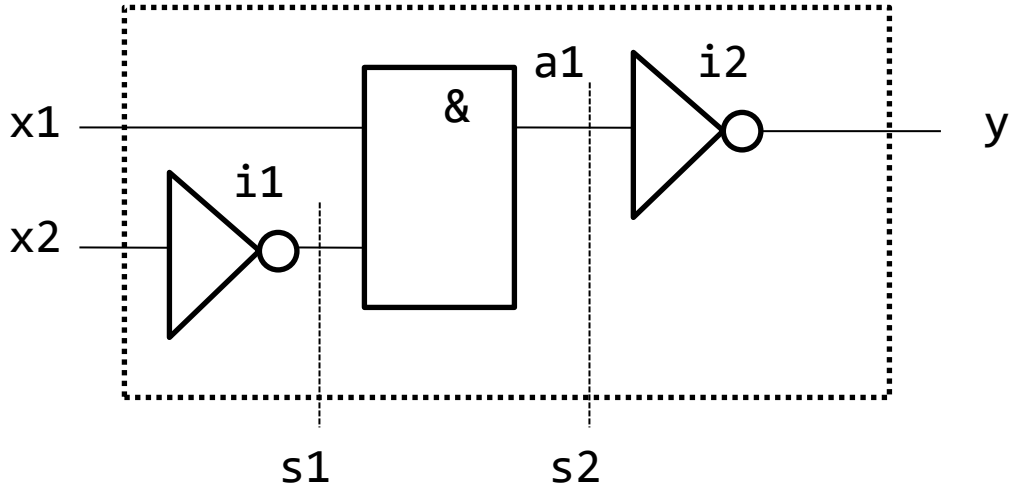



```
module REG #(parameter WIDTH = 2) (D, EN, clk, R, Q);
    input       [WIDTH-1:0] D;
    input                   EN, clk, R;
    output reg  [WIDTH-1:0] Q;

    ...
endmodule
```

```
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;

    ...
endmodule
```

Описание вентилей на основе определяемых пользователем примитивов (UDP) (1)

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

  table
    0  ?    : 1;
    ?  1    : 1;
    1  0    : 0;
  endtable

endprimitive
```



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

Описание вентилей на основе определяемых пользователем примитивов (UDP) (2)

Символ	Значение	Пояснение
0	0	Логический «0»
1	1	Логическая «1»
x	x	Неизвестное значение
?	0, 1, X	Любое перечисленное значение
ab	a-b	Переключение из a в b для 0, 1, x
f	(1-0)	Срез сигнала
r	(0-1)	Фронт сигнала
p	(0-1), (0-x), (x-1), (1-z), (z-1)	Фронт, включая x и z
n	(1-0), (1-x), (x-0), (0-z), (z-0)	Срез, включая x и z
*	(??)	Любое переключение сигнала
-	(0-0), (1-1), (x-x), (z-z)	Сигнал не меняется

Описание комбинационных вентилей через UDP

Синтаксис описания таблицы:

```
<input1> <input2> <input3> ... <inputN> : <output>;
```

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

  table
    0  ?    : 1;
    ?  1    : 1;
    1  0    : 0;
  endtable

endprimitive
```

```
primitive mux(out, a, x1, x2);
  output out;
  input  a, x1, x2;

  table
  // a x1 x2   out
     0  0  ?  : 0;
     0  1  ?  : 1;
     1  ?  0  : 0;
     1  ?  1  : 1;
  endtable

endprimitive
```

Описание последовательностных элементов через UDP

Синтаксис описания таблицы:

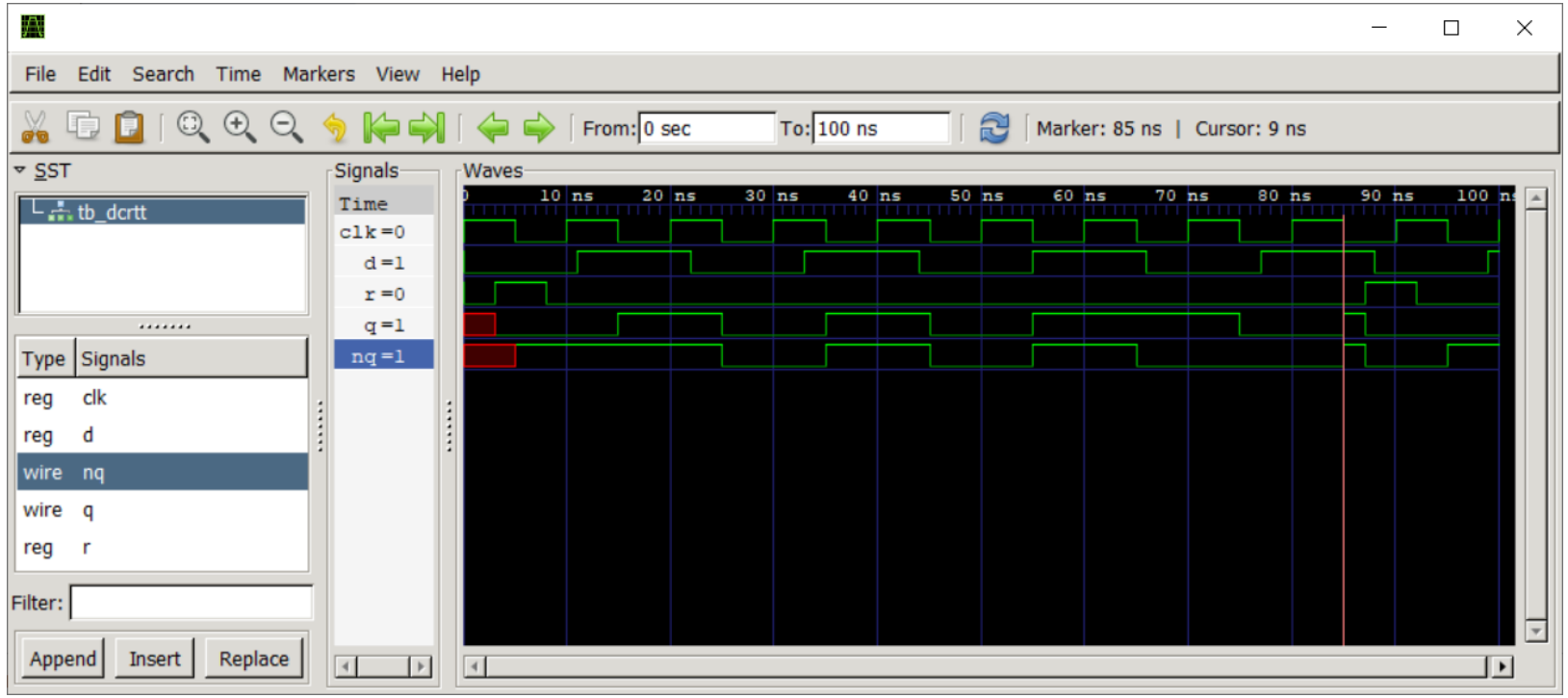
```
<input1> <input2> <input3> ... <inputN> : <current_state> : <next_state>;
```

```
primitive dff(q, clk, d);
  output q;
  input  clk, d;

  table
  // clk   d  q_now q_next
     (10)  0   : ?   : 0; // q=d
     (10)  1   : ?   : 1; // q=d
     (x0)  0   : ?   : 0; // q=d
     (x0)  1   : ?   : 1; // q=d
  endtable

endprimitive
```

```
primitive dff(q, clk, d, r);
  output q;
  input  clk, d, r;

  table
  // clk   d  r  q_now q_next
      ?    ?  1   : ?   : 0; // reset
      ?    ? (01) : ?   : 0; // reset
     (10)  0  0   : ?   : 0; // q=d
     (10)  1  0   : ?   : 1; // q=d
     (x0)  0  0   : ?   : 0; // q=d
     (x0)  1  0   : ?   : 1; // q=d
  endtable

endprimitive
```

Особенности алгоритма моделирования

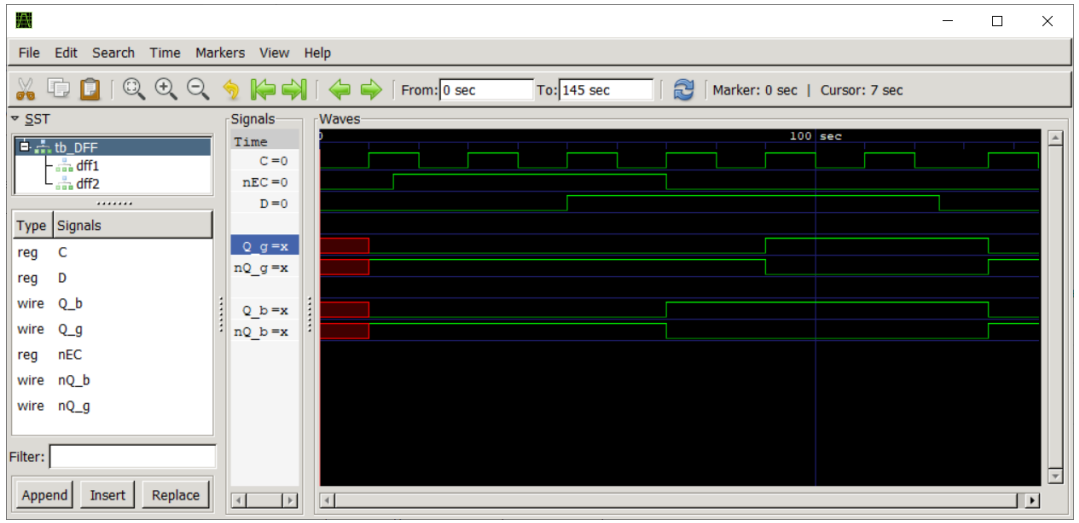
```
module dcrtt(clk, d, r, q, nq);
    input       clk, d, r;
    output reg  q, nq;

    always@(negedge clk, posedge r) begin
        if(r == 1'b1)
            q <= 1'b0;
        else
            q <= d;
        nq <= ~q;
    end

endmodule
```

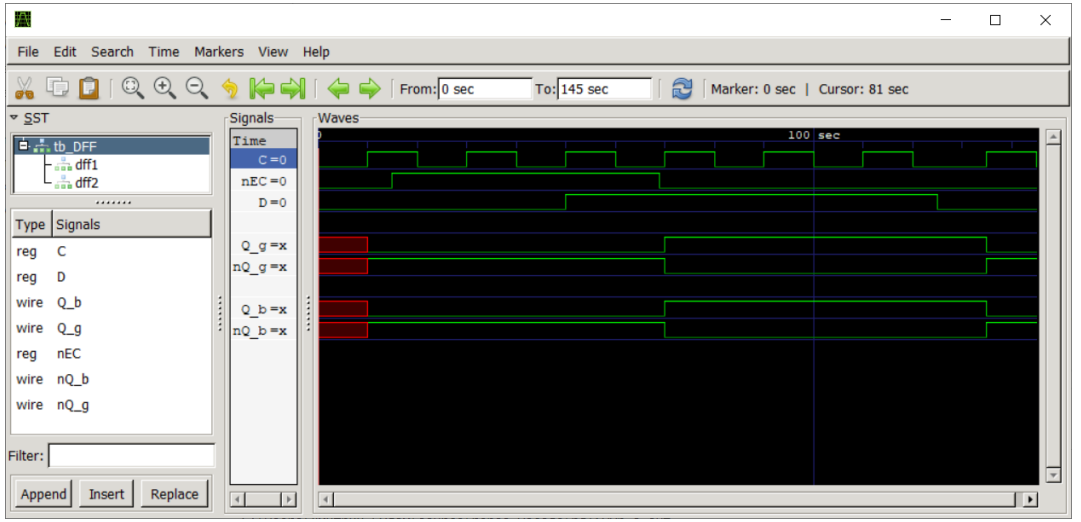


Особенности алгоритма моделирования

```
module DFF_gate(D, C, nEC, Q, nQ);
  input  D, C, nEC;
  output Q, nQ;

  or    or0(net1, D, Cdff);
  not   not1(net0, D);
  or    or1(Cdff, C, nEC);
  or    or2(net2, Cdff, net0);
  nand  nand1(net3, net1, net4);
  nand  nand2(net4, net2, net3);
  nor   nor1(Q, nQ, net5);
  nor   nor2(nQ, Q, net6);
  and   and1(net5, Cdff, net3);
  and   and2(net6, Cdff, net4);
endmodule
```

```
module DFF_beh(D, C, nEC, Q, nQ);
  input D, C, nEC;
  output reg Q, nQ;

  always@(posedge C)
    if(nEC == 1'b0) begin
        Q <= D;
        nQ <= ~D;
    end

endmodule
```





Поддержка VPI (Verilog Programmer Interface) симулятором Icarus Verilog (1)

* https://iverilog.fandom.com/wiki/Using_VPI

```
# include <vpi_user.h>

static int hello_calltf(char *user_data) {
  vpi_printf("Hello, World!\n");
  return 0;
}

          void hello_register() {
            s_vpi_systf_data data;

            data.type      = vpiSysTask;
            data.tfname    = "$hello";
            data.calltf    = hello_calltf;
            data.compiletf = 0;
            data.sizetf    = 0;
            data.user_data = 0;

            vpi_register_systf(&data);
          }
```

```
void (*vlog_startup_routines[])() = {
  hello_register,
  0
};
```

Поддержка VPI (Verilog Programmer Interface) симулятором Icarus Verilog (2)

* https://iverilog.fandom.com/wiki/Using_VPI

```
module main;

  initial begin
    $hello;
    $finish;
  end

endmodule
```

```
% iverilog-vpi hello.c
% iverilog -ohello.vvp hello.v
% vvp -M. -mhello hello.vvp
```

```
student@localhost:~/vpi> iverilog-vpi hello.c
Compiling hello.c...
hello.c: In function 'my_hello_calltf':
hello.c:33:40: warning: unused parameter 'xx' [-Wunused-parameter]
 static PLI_INT32 my_hello_calltf(char *xx)
                                       ^~
hello.c: At top level:
hello.c:39:13: warning: function declaration isn't a prototype [-Wstrict-prototypes]
 static void my_hello_register()
             ^~~~~~~~~~~~~~~~~
hello.c:56:1: warning: function declaration isn't a prototype [-Wstrict-prototypes]
 void (*vlog_startup_routines[])() = {
 ^~~~
Making hello.vpi from  hello.o...
/usr/lib64/gcc/x86_64-suse-linux/7/../../../../x86_64-suse-linux/bin/ld: cannot find -lveriuser: No such file or directory
/usr/lib64/gcc/x86_64-suse-linux/7/../../../../x86_64-suse-linux/bin/ld: cannot find -lvpi: No such file or directory
collect2: error: ld returned 1 exit status
student@localhost:~/vpi>
```

Поддержка VPI (Verilog Programmer Interface) симулятором Icarus Verilog (2)

* https://iverilog.fandom.com/wiki/Using_VPI

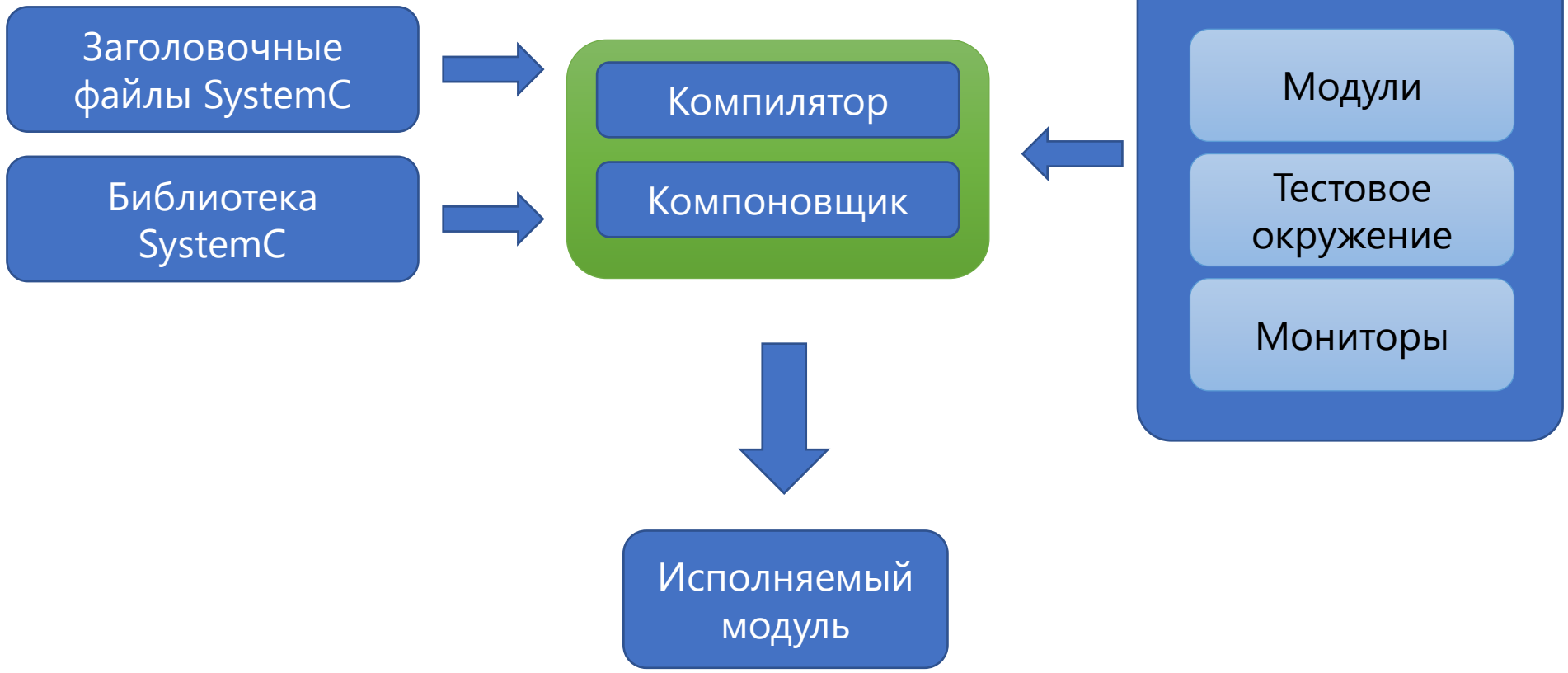
```
module main;

  initial begin
    $hello;
    $finish;
  end

endmodule
```

```
% gcc -I/usr/include/iverilog -c -fpic hello.c
% gcc -shared -o hello.vpi hello.o
% iverilog -o hello.vvp hello.v
% vvp -M -mhello hello.vvp
```

```
student@localhost:~/vpi> gcc -I/usr/include/iverilog -c -fpic hello.c
student@localhost:~/vpi> gcc -shared -o hello.vpi hello.o
student@localhost:~/vpi> iverilog -o hello.vvp hello.v
student@localhost:~/vpi> vvp -M. -mhello hello.vvp
Hello, World!
student@localhost:~/vpi>
```

Библиотека SystemC

https://systemc.org



SystemC: написание кода модулей (1)

```
SC_MODULE(inverter) {
    sc_in   <bool>  x;
    sc_out  <bool>  y;

    void    doOperate() {
      if(true == x)
        y = false;
      else
        y = true;
    }

    SC_CTOR(inverter) {
        SC_METHOD(doOperate);
        sensitive << x;
    }
};
```

```
int sc_main(int, char *[]) {
  sc_signal <bool> y;
  sc_signal <bool> x;

  inv i1("i1");
  i1.x(x);
  i1.y(y);

  x = false;
  sc_start(10, SC_NS);
  x = true;
  sc_start(10, SC_NS);
  x = false;
  sc_start(40, SC_NS);
  x = true;
  sc_start(20, SC_NS);

  return 0;
}
```