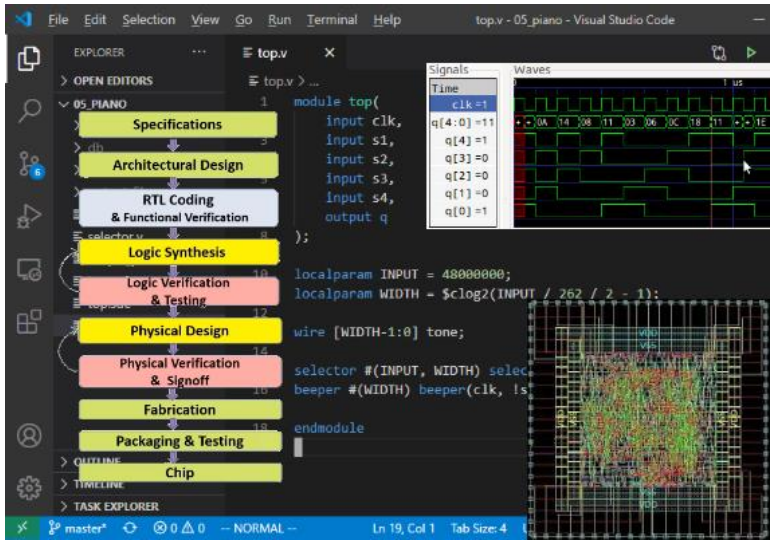
МИЭТ
Национальный исследовательский университет «МИЭТ»
Институт интегральной электроники

Лингвистические средства проектирования

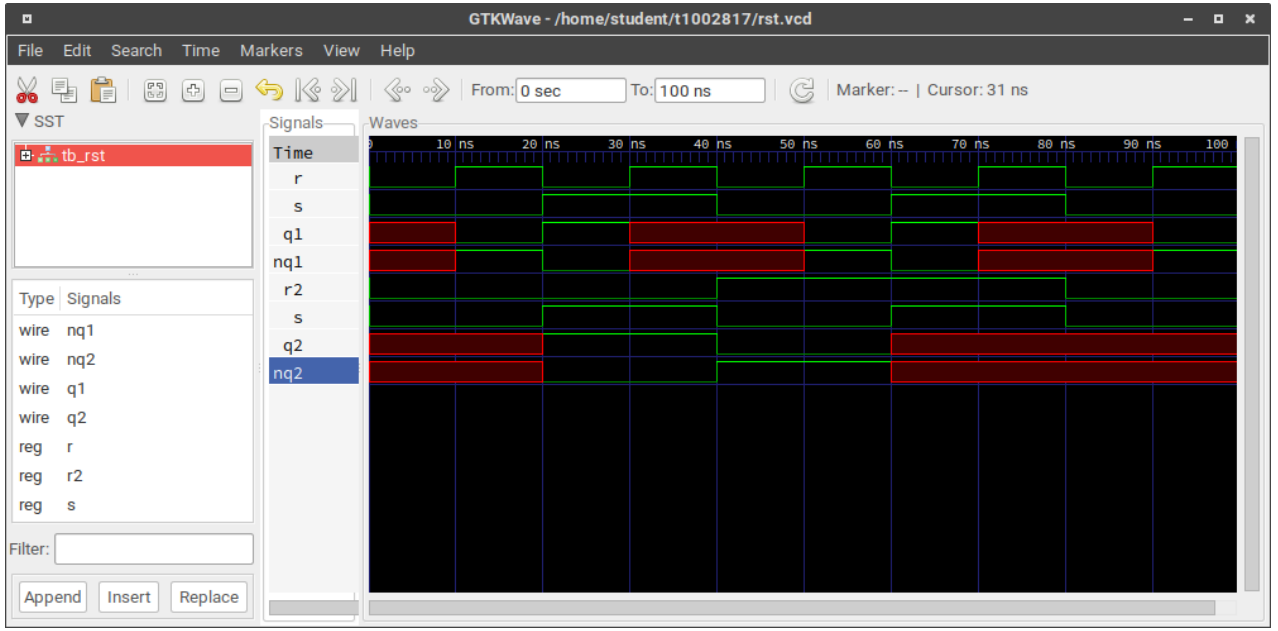


Лекция 4

Последовательные и параллельные операторы

Вывод данных и управление процессом моделирования (1)

```
module rstrig(r, s, q, nq);
  input r, s;
  output reg q, nq;

  always @(r, s) begin
    case ({r, s})
      1: {q, nq} <= 2;
      2: {q, nq} <= 1;
      3: begin
           {q, nq} <= 2'bxx;
           $display("Error [%m]! Illegal seq at %0t!", $realtime);
         end
    endcase
  end
endmodule
```



Вывод данных и управление процессом моделирования (2)

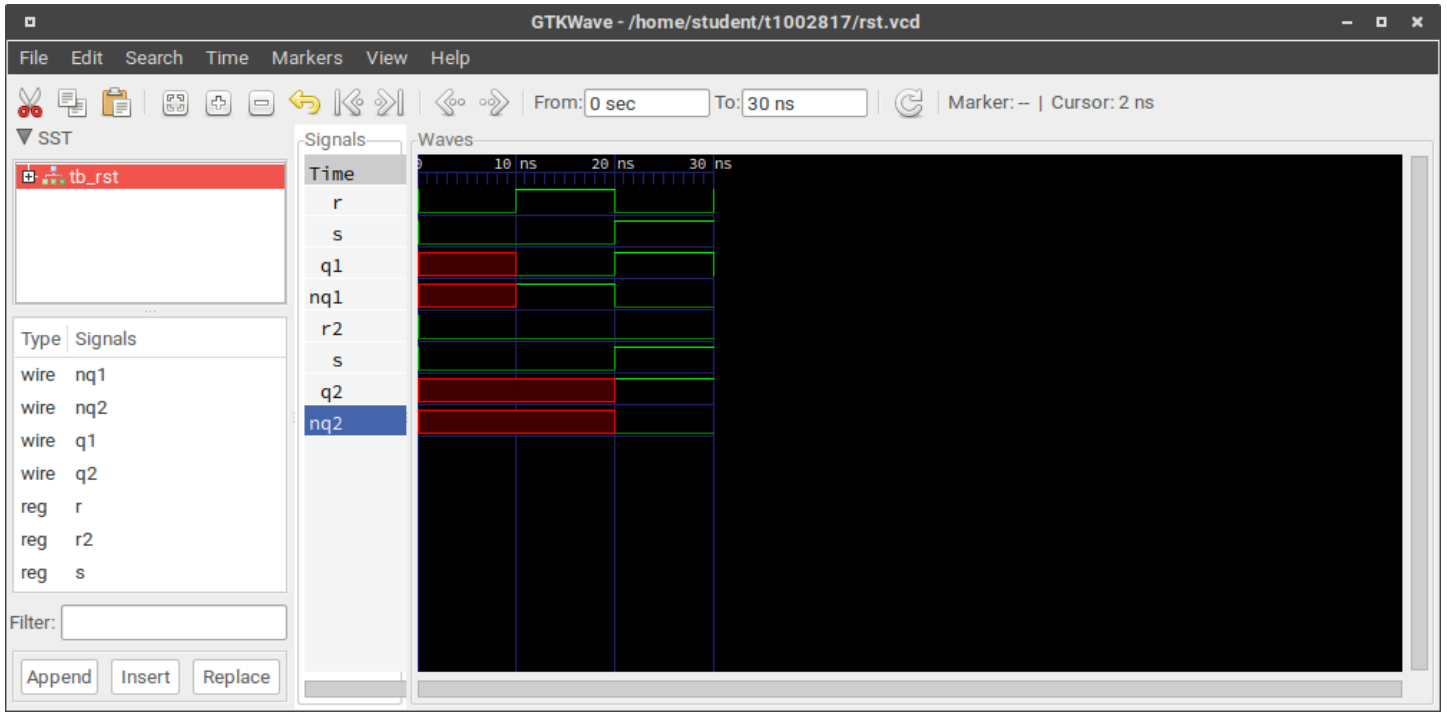
```
module rstrig(r, s, q, nq);
  input r, s;
  output reg q, nq;

  reg state = 0;

  always @(r, s) begin
    case ({r, s})
      0: if (state == 1)
           {q, nq} <= 2'bxx;
      1: {q, nq} <= 2;
      2: {q, nq} <= 1;
      3: begin
           $display("Error [%m]! Illegal seq at %0t!", $realtime);
           state <= 1;
           {q, nq} <= 0;
         end
    endcase
  end
endmodule
```

```
1: {state, q, nq} <= 2;
2: {state, q, nq} <= 1;
```

```
1: {q, nq, state} <= 2 << 1;
2: {q, nq, state} <= 1 << 1;
```

Вывод данных и управление процессом моделирования (3)

```
module rstrig(r, s, q, nq);
  input r, s;
  output reg q, nq;

  reg state = 0;

  always @(r, s) begin
    case ({r, s})
      0: if (state == 1)
           {q, nq} <= 2'bxx;
      1: {q, nq} <= 2;
      2: {q, nq} <= 1;
      3: begin
           $display("Error [%m]! Illegal seq at %0t!", $realtime);
           state <= 1;
           {q, nq} <= 0;
           $finish;
         end
...
```

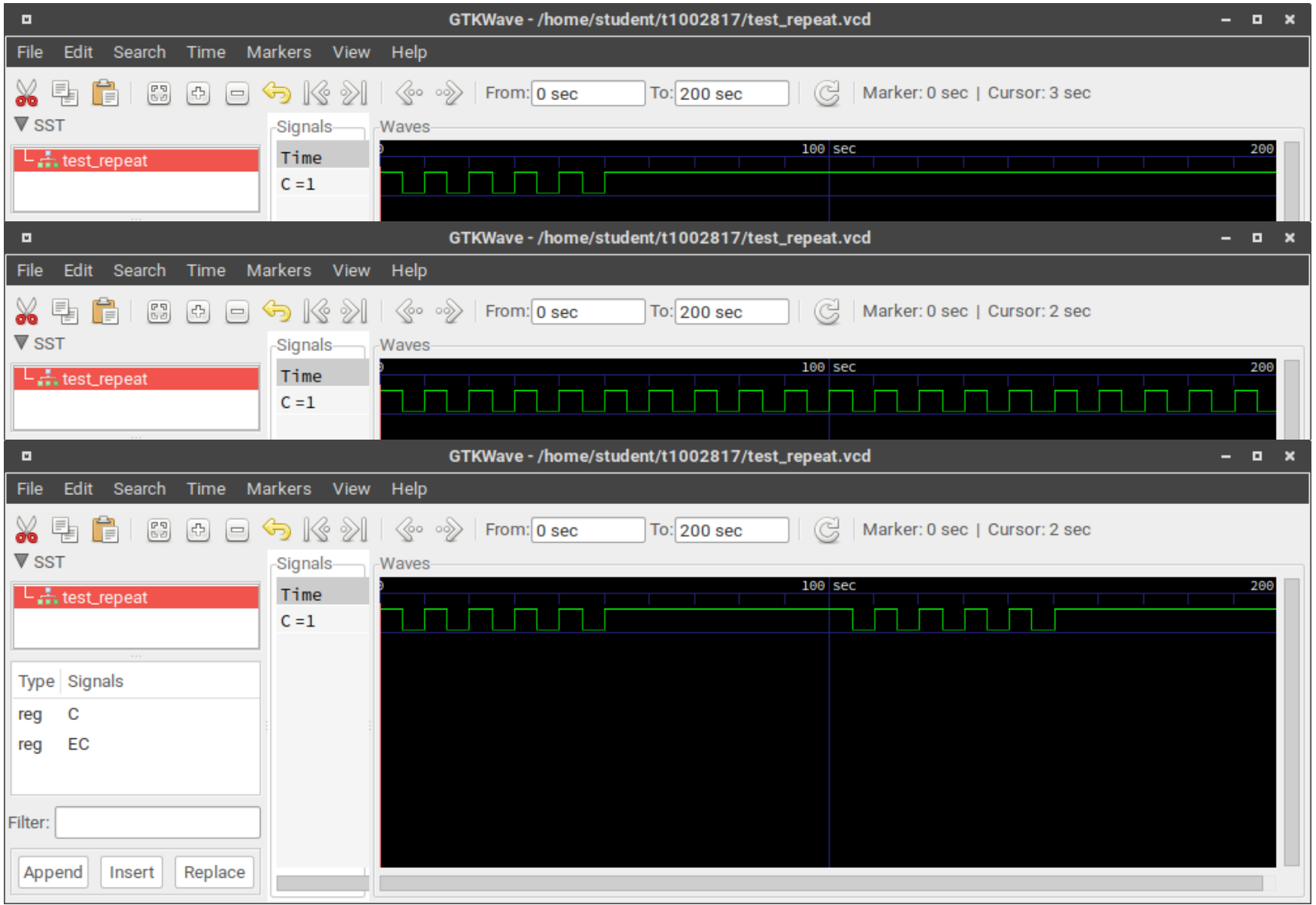


Особенности написания кода

```
`define STATUS_OK    0
`define STATUS_ERROR 1

module rstrig(r, s, q, nq);
  input r, s;
  output reg q, nq;

  reg state = `STATUS_OK;

  always @(r, s) begin
    case ({r, s})
      0: if (state == `STATUS_ERROR)
           {q, nq} <= 2'bxx;
      1: {state, q, nq} <= 2;
      2: {state, q, nq} <= 1;
      3: begin
           $display("Error [%m]! Illegal seq at %0t!", $realtime);
           state <= `STATUS_ERROR;
           {q, nq} <= 0;
         end
...
```

Последовательные операторы: циклы forever и repeat (1)

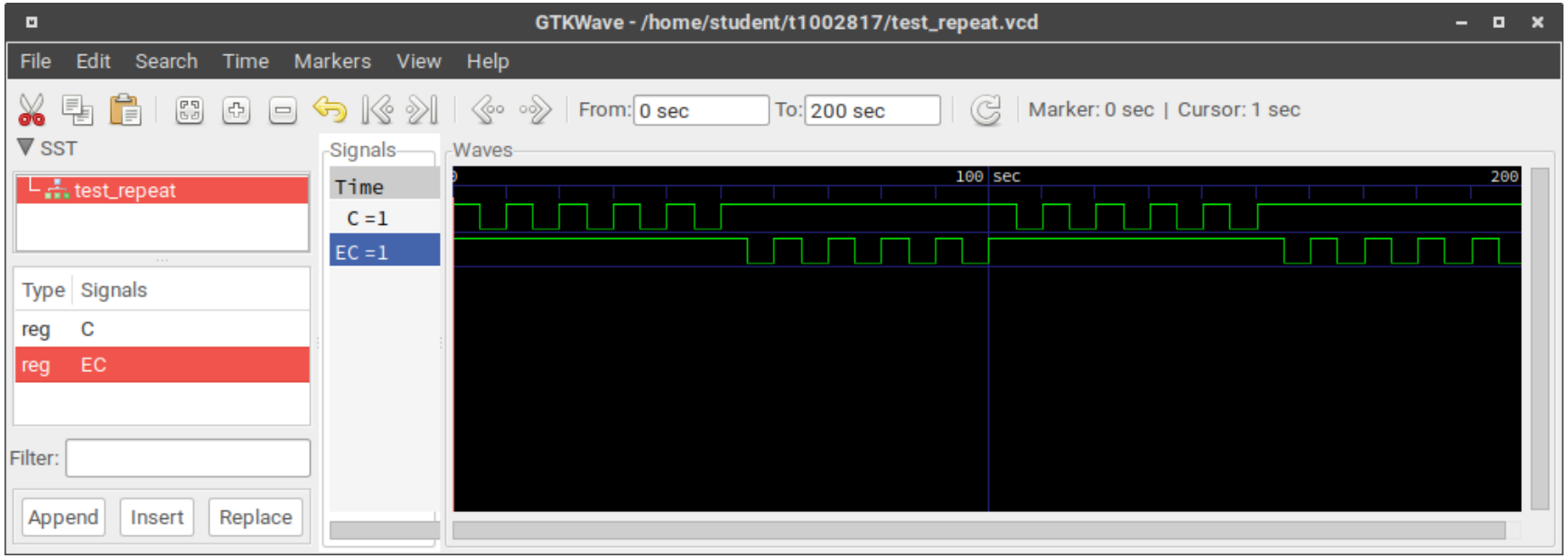
```
initial begin
  repeat (10)
    #5 C = ~C;
end

initial begin
  forever
    repeat (10)
      #5 C = ~C;
end

initial begin
  forever begin
    repeat (10)
      #5 C = ~C;
    #50;
  end
end
```



Последовательные операторы: циклы forever и repeat (2)

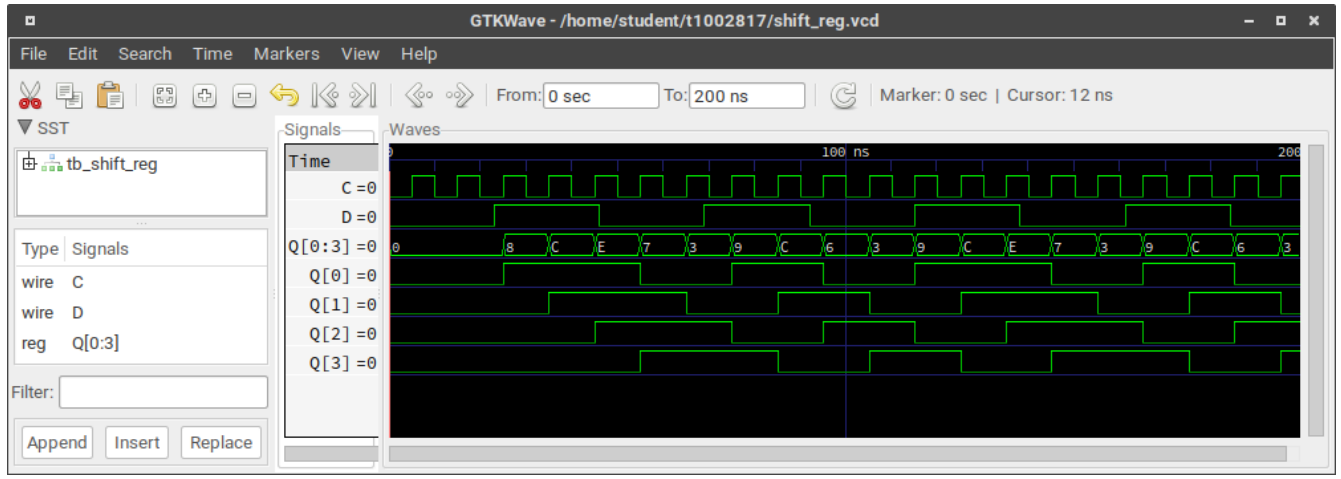
```
initial begin
  forever begin
    repeat (10)
      #5 C = ~C;
    #50;
  end
end
```

```
initial begin
  forever begin
    #50;
    repeat (10)
      #5 EC = ~EC;
  end
end
```



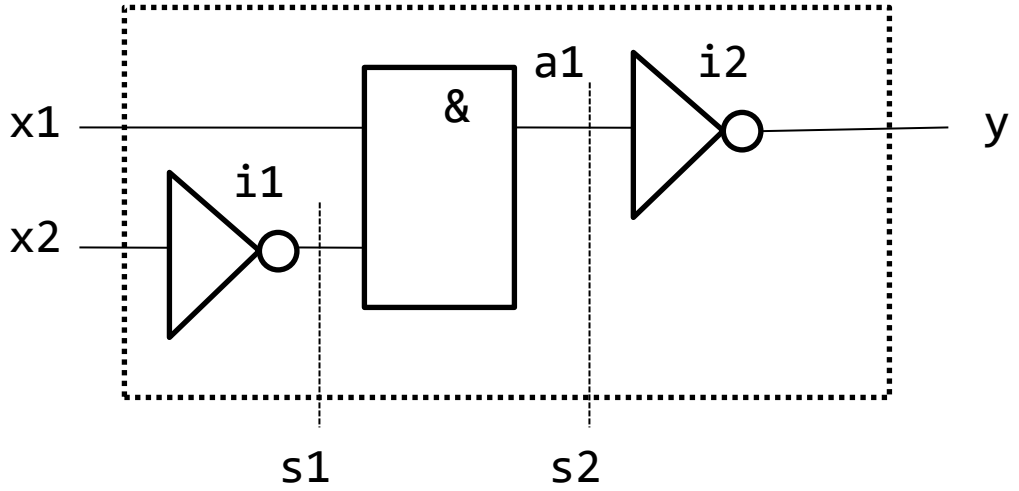
Последовательные операторы: цикл for

```
output reg [3:0] Q;
...
always @(posedge C) begin
  Q[3] <= Q[2];
  Q[2] <= Q[1];
  Q[1] <= Q[0];
  Q[0] <= D;
end
```




```
integer i;
always @(posedge C) begin

  for(i = 3; i > 0; i = i - 1)
    Q[i] <= Q[i - 1];

  Q[0] <= D;

end
```

Параллельные операторы: позиционное и ассоциативное назначение портов

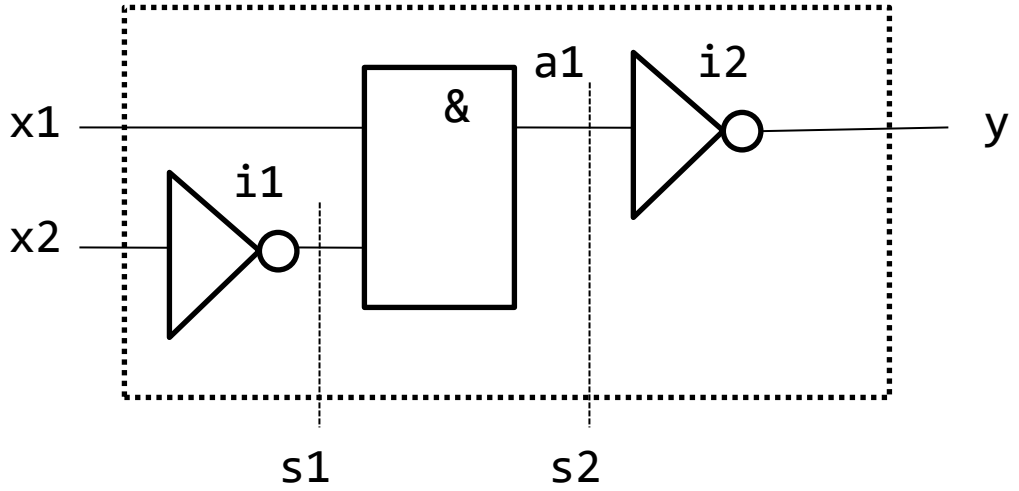



```
module device(x1, x2, y);
  input  x1, x2;
  output y;

  wire s1, s2;

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

endmodule
```

```
and2 a1(.x2(s1), .y(s2), .x1(x1));
```

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

Параллельные операторы: инстанцирование модулей (1)




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

  wire s1, s2;

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

endmodule
```

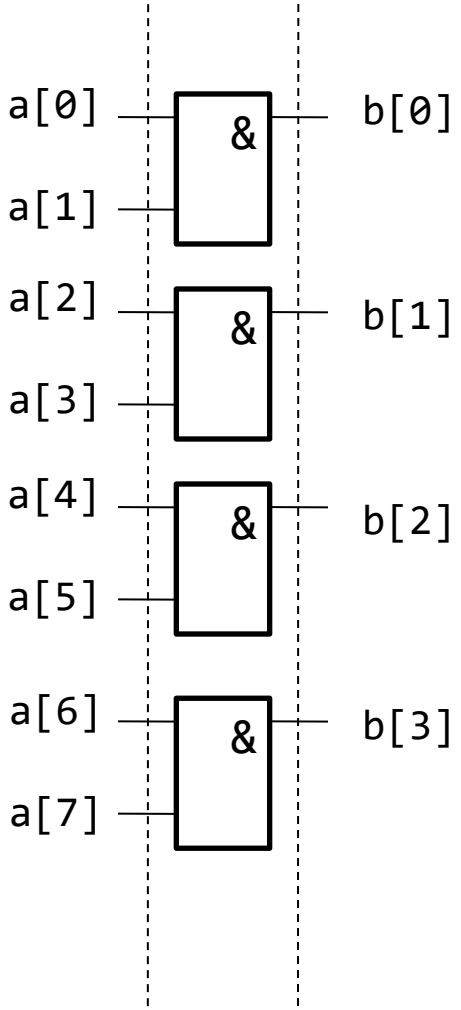
```
inv  i1(x2, s1),
     i2(s2, y);
and2 a1(x1, s1, s2);
```

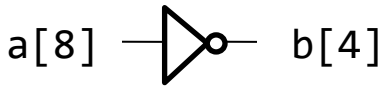
Параллельные операторы: инстанцирование модулей (2)

```
module tb_device;
  reg x1, x2;
  wire y;

  device i1(.x1(x1), .x2(x2), .y(y));
  always #5  x1 = ~x1;
  always #10 x2 = ~x2;

  initial begin
    $dumpfile("device.vcd");
    $dumpvars(1, tb_device);
    x1 = 0;
    x2 = 0;
    #100
    $finish;
  end

endmodule
```

```
device i1(.*);
```

Параллельные операторы: оператор generate (1)

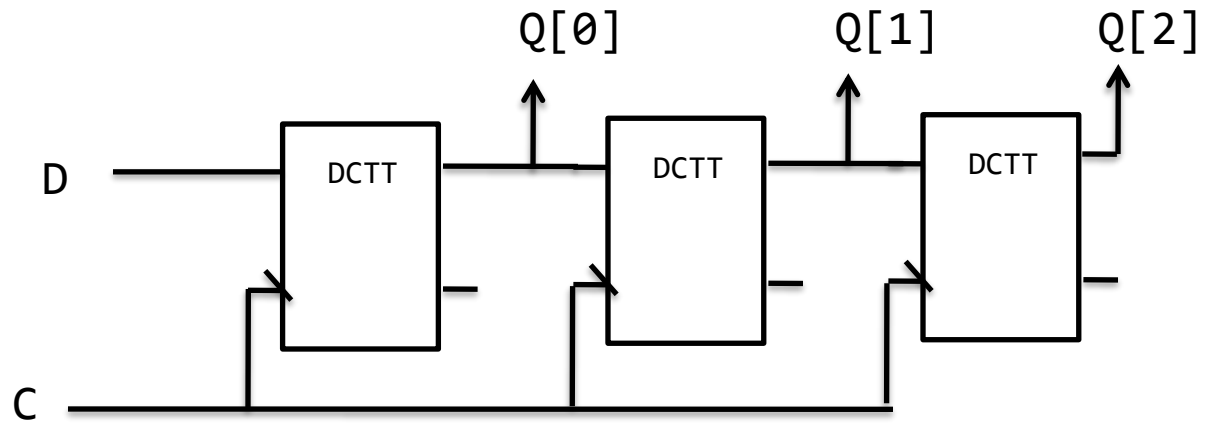




Структурное описание:

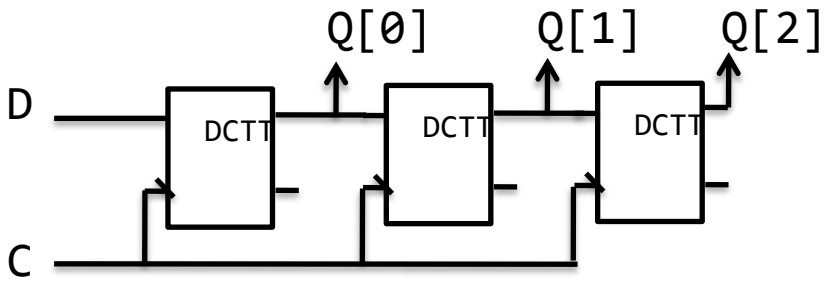
```
input  [8:0] a;
wire   [4:0] b;


genvar g;

generate

  for (g = 0; g < 4; g = g + 1)
    and2 i(a[g*2], a[g*2 + 1], b[g]);

endgenerate

inv i1(a[8], b[4]);
```

Параллельные операторы: оператор generate (2)




```
genvar g;

dctt i1(.d(D), .c(C), .q(Q[0]), .nq());

generate

  for (g = 1; g < 3; g = g + 1)
    dctt i(.d(Q[g - 1]), .c(C), .q(Q[g]), .nq());

endgenerate
```

Параллельные операторы: оператор generate (3)




```
genvar g;

generate

  for (g = 0; g < 3; g = g + 1)

    if (g == 0)
      dctt i1(.d(D), .c(C), .q(Q[g]), .nq());
    else
      dctt i(.d(Q[g - 1]), .c(C), .q(Q[g]), .nq());

endgenerate
```

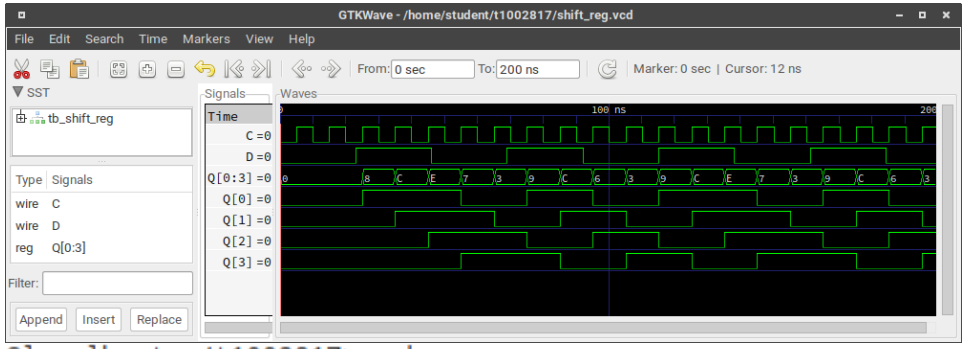
Автоматизация запуска: скрипты bash

```
#!/usr/bin/bash

if [ -f ./a.out ]; then
  rm ./a.out
fi

iverilog -c ./config.cmd
if [ -f ./a.out ]; then
  ./vvp a.out
  gtkwave ./inv.vcd
else
  echo -e "\e[31m Verilog code compilation error!\e[0m"
fi
```

```
student@localhost:~/t1002817> ./run.sh
tb_inv.v:5: error: Unknown module type: in
2 error(s) during elaboration.
*** These modules were missing:
        in referenced 1 times.
***
Verilog code compilation error!
student@localhost:~/t1002817>
```

Автоматизация запуска: синтаксис Makefile

```
all: analyze run view

analyze: ./shift_reg.v ./tb_shift_reg.v
        @echo "Erasing simulation results"
        rm -f ./shift_reg.out
        rm -f ./shift_reg.vcd
        @echo "Analyzing design..."
        iverilog -o ./shift_reg.out \
                    ./shift_reg.v \
                    ./tb_shift_reg.v

run: ./shift_reg.out
        vvp ./shift_reg.out

view: ./shift_reg.vcd
        gtkwave ./shift_reg.vcd
```




```
student@localhost:~/t1002817> make
Erasing simulation results
rm -f ./shift_reg.out
rm -f ./shift_reg.vcd
Analyzing design...
iverilog -o ./shift_reg.out ./shift_reg.v ./tb_shift_reg.v
./tb_shift_reg.v:7: error: Unknown module type: shft_reg
2 error(s) during elaboration.
*** These modules were missing:
        shft_reg referenced 1 times.
***
make: *** [Makefile:8: analyze] Error 2
student@localhost:~/t1002817>
```

```
student@localhost:~/t1002817> make
Erasing simulation results
rm -f ./shift_reg.out
rm -f ./shift_reg.vcd
Analyzing design...
iverilog -o ./shift_reg.out ./shift_reg.v ./tb_shift_reg.v
vvp ./shift_reg.out
VCD info: dumpfile shift_reg.vcd opened for output.
gtkwave ./shift_reg.vcd

GTKWave Analyzer v3.3.104 (w)1999-2020 BSI

[0] start time.
[200] end time.
```