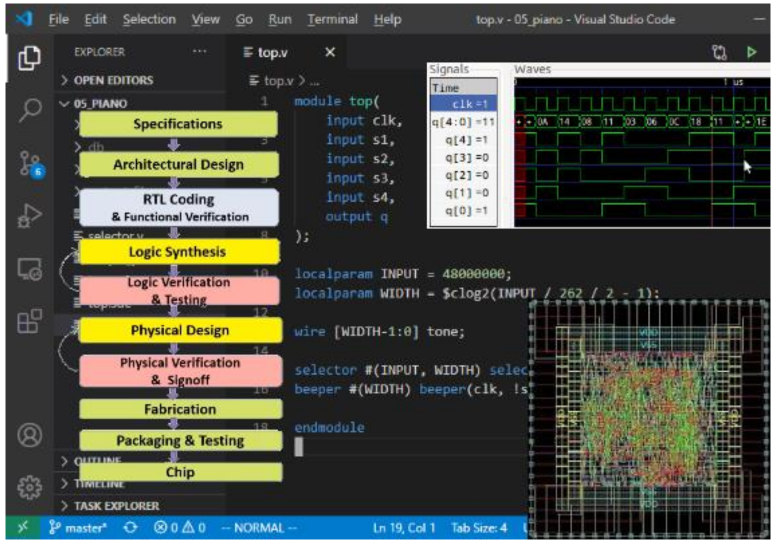
Национальный исследовательский университет «МИЭТ»

Институт интегральной электроники

Лингвистические средства проектирования



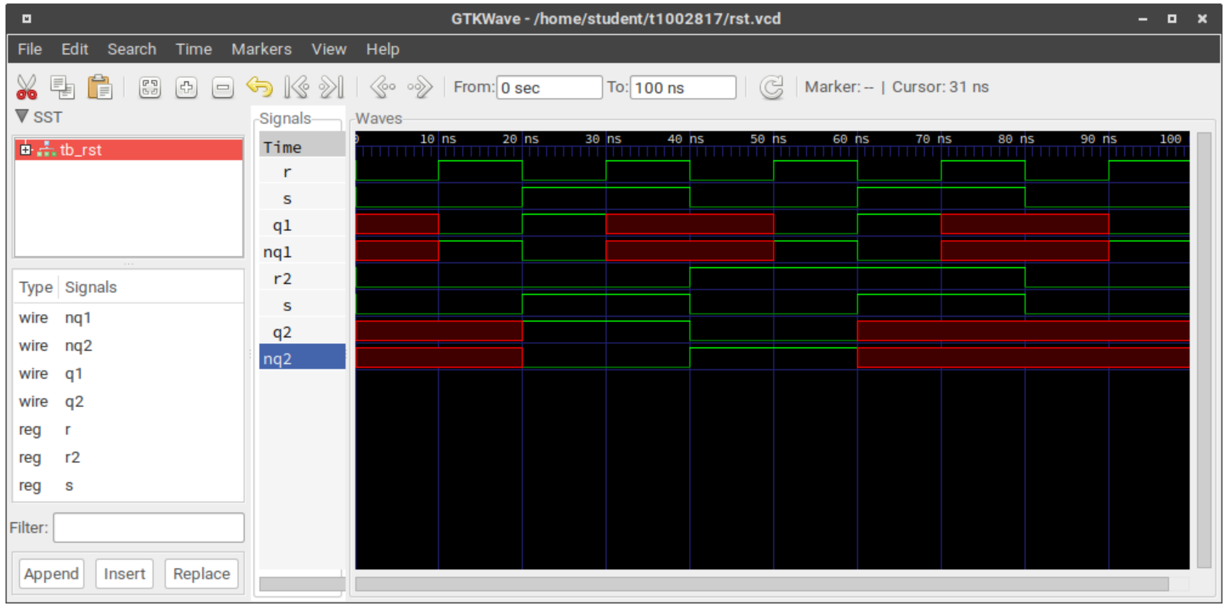
```
module top(
    input clk,
    input s1,
    input s2,
    input s3,
    input s4,
    output q
);

localparam INPUT = 48000000;
localparam WIDTH = $clog2(INPUT / 262 / 2 - 1);

wire [WIDTH-1:0] tone;

selector #(INPUT, WIDTH) selec
beeper #(WIDTH) beeper(clk, !s

endmodule
```

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
```



Type	Signals
wire	nq1
wire	nq2
wire	q1
wire	q2
reg	r
reg	r2
reg	s

Вывод данных и управление процессом моделирования (2)

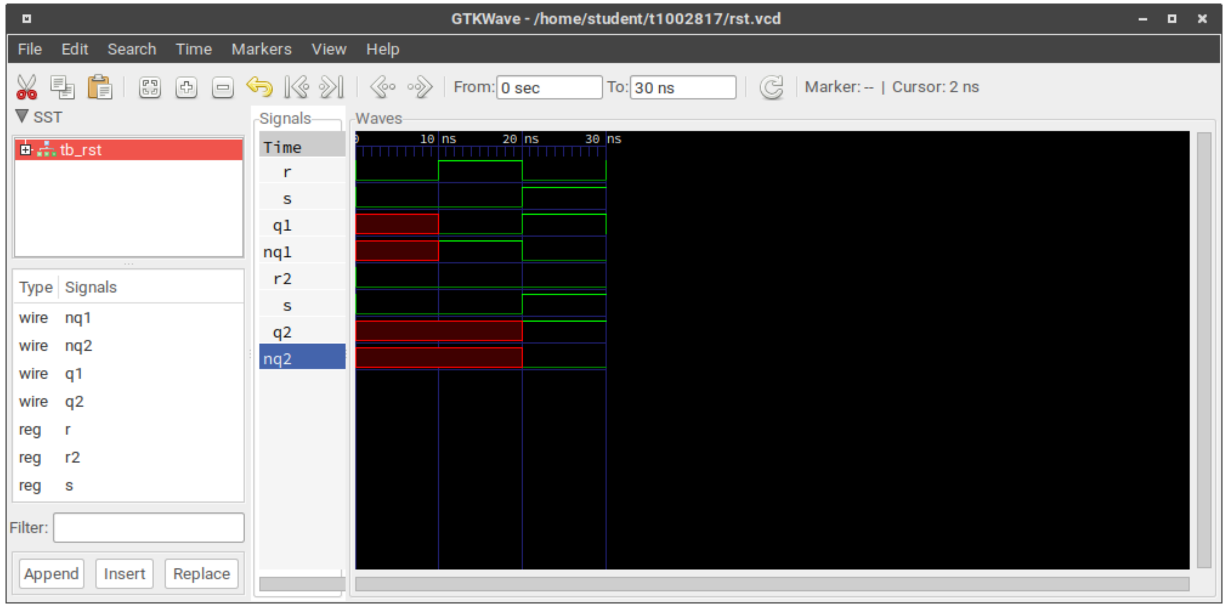
```
module rstrig(r, s, q, nq);
  input r, s;
  output reg q, nq;

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

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



Type	Signals
wire	nq1
wire	nq2
wire	q1
wire	q2
reg	r
reg	r2
reg	s

Использование макроопределний

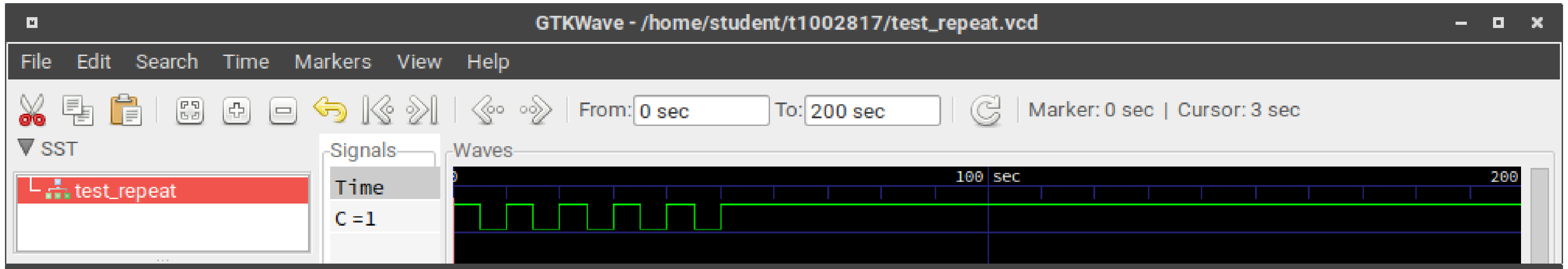
```
`define STATUS_OK    0
`define STATUS_ERROR 1

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
...
```

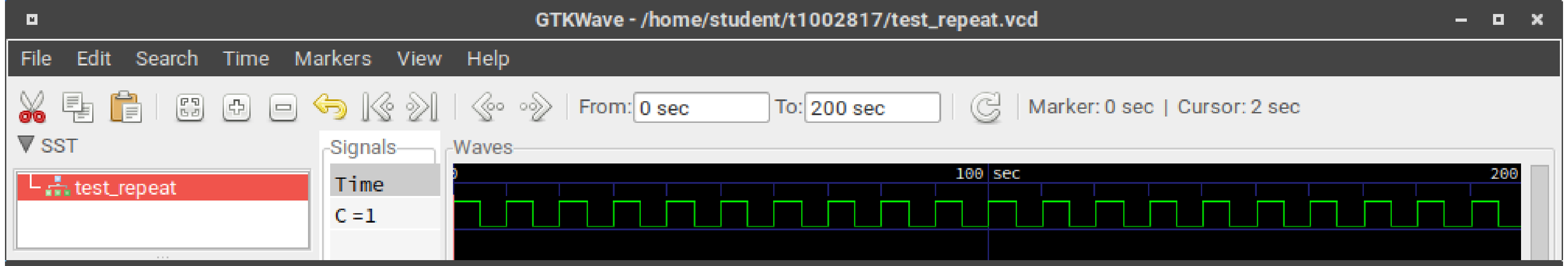
Последовательные операторы: циклы forever и repeat (1)

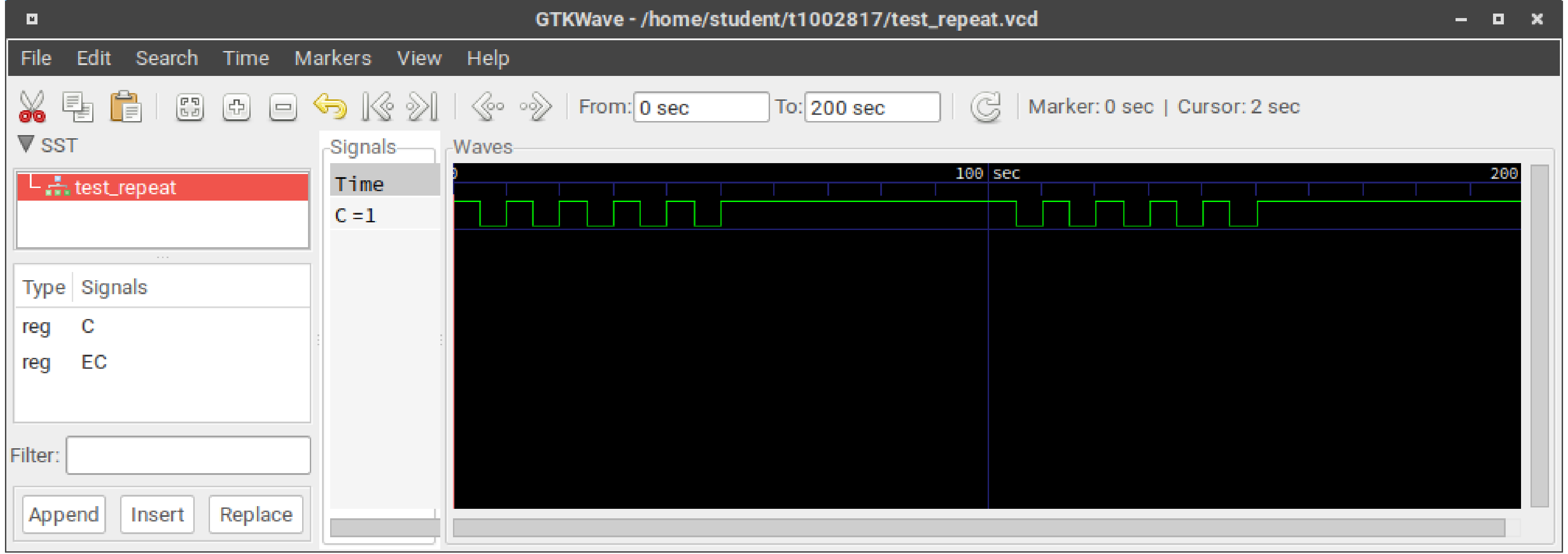
```
initial begin
  repeat (10)
    #5 C = ~C;
end
```

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

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

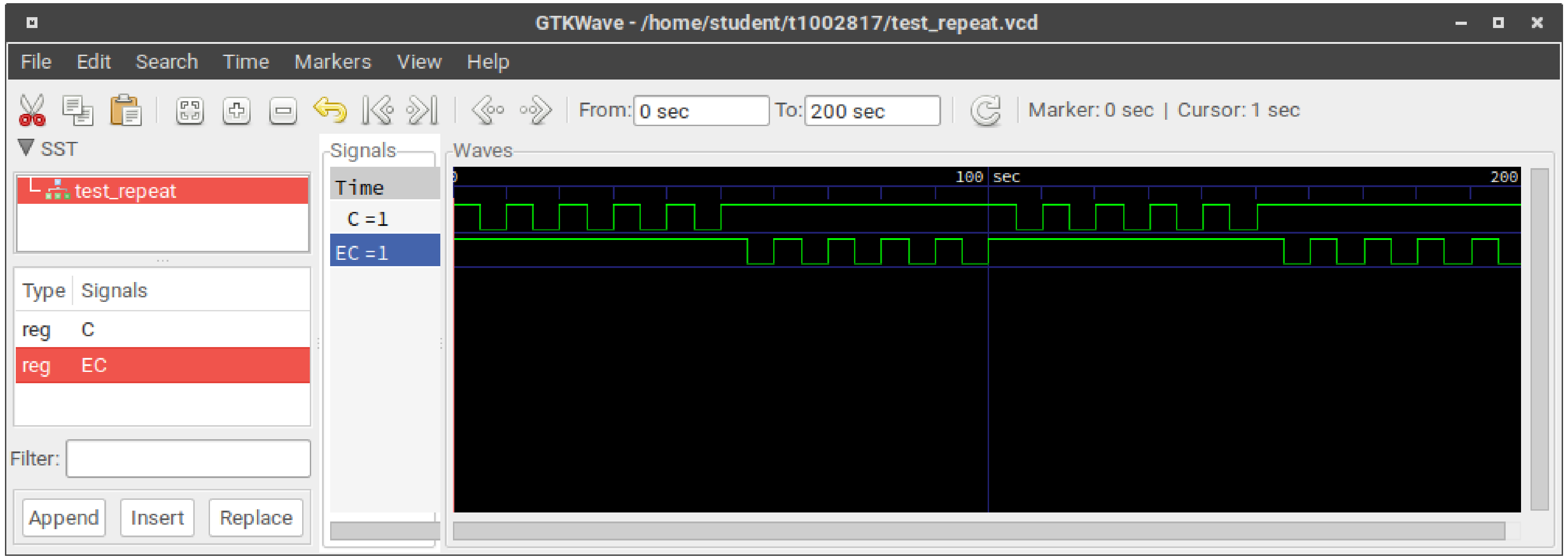

GTKWave - /home/student/t1002817/test_repeat.vcd
File Edit Search Time Markers View Help
From: 0 sec To: 200 sec
Marker: 0 sec | Cursor: 3 sec
SST
test_repeat
Signals
Time
C =1
Waves
100 sec
200


GTKWave - /home/student/t1002817/test_repeat.vcd
File Edit Search Time Markers View Help
From: 0 sec To: 200 sec
Marker: 0 sec | Cursor: 2 sec
SST
test_repeat
Signals
Time
C =1
Waves
100 sec
200



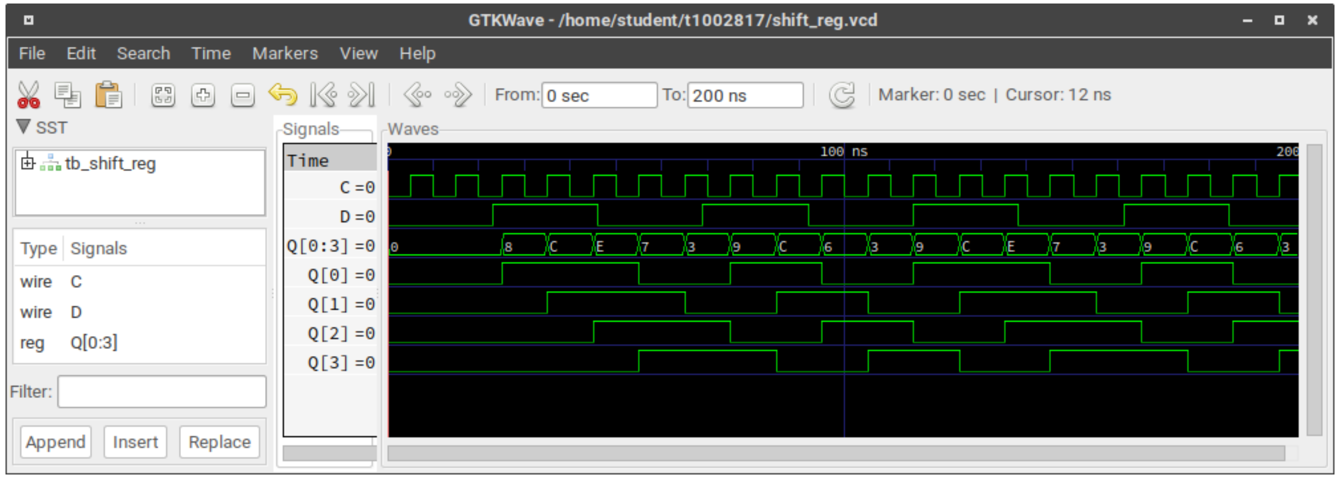
Последовательные операторы: циклы forever и repeat (2)

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


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

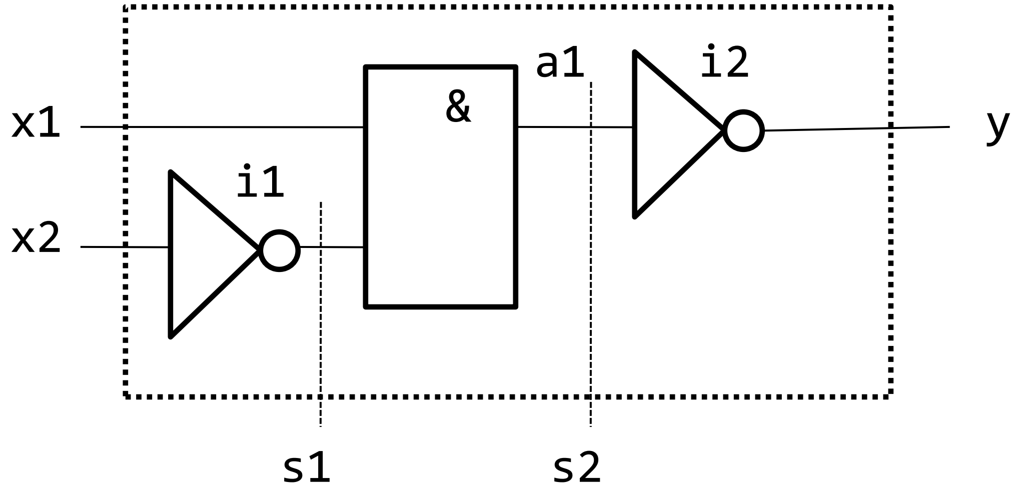


Последовательные операторы: цикл for (2)

```
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

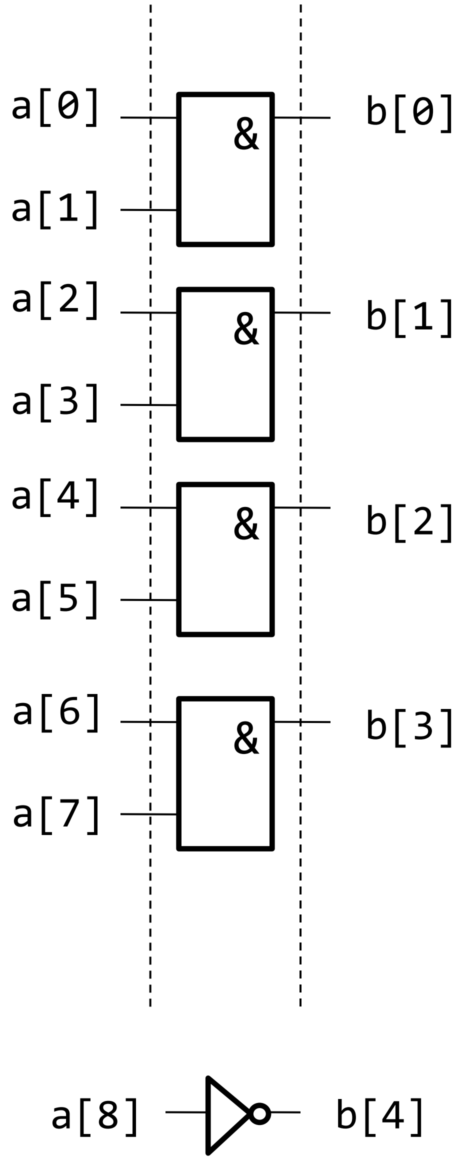


```
module device(x1, x2, y);
  input  x1, x2;
  output y;

  wire s1, s2;

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

endmodule
```

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

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

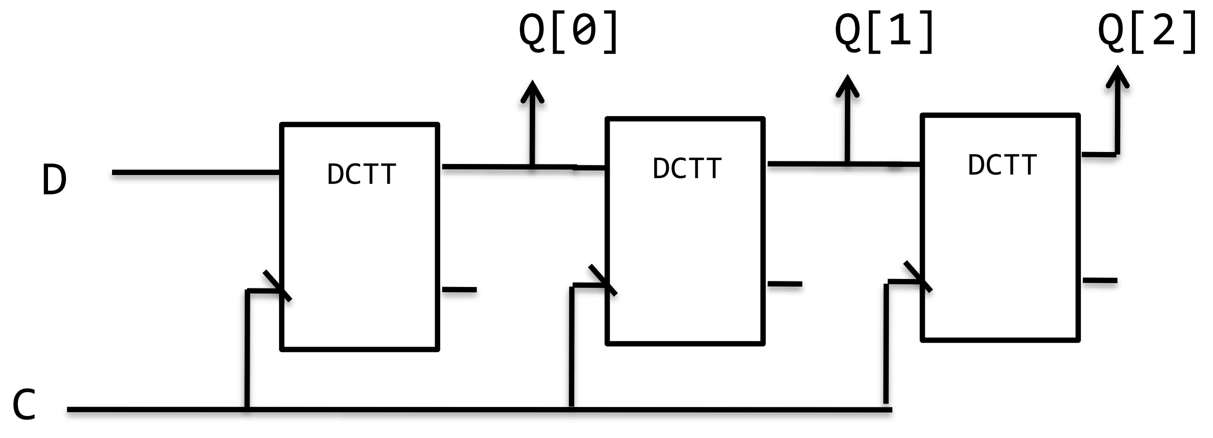
Параллельные операторы: оператор generate (1)



Структурное описание:

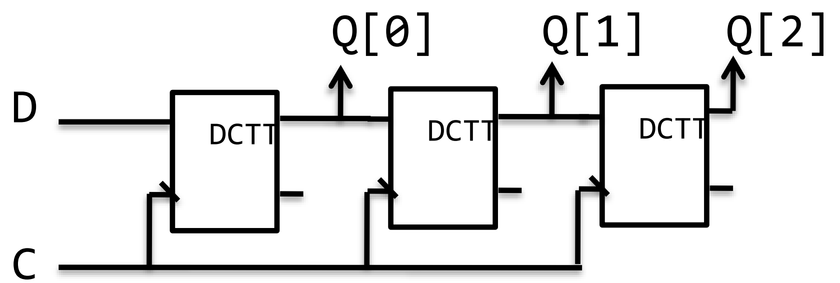
```
input  [8:0] a;
wire   [4:0] b;



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
```