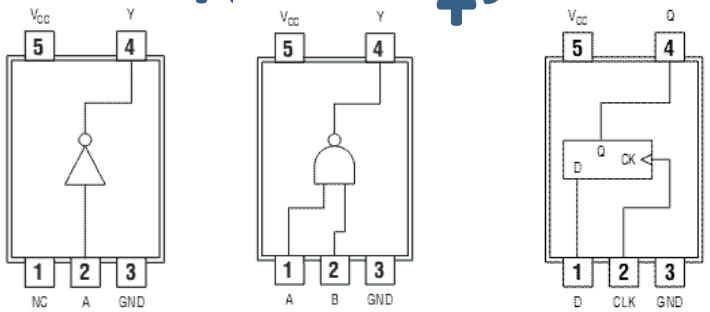
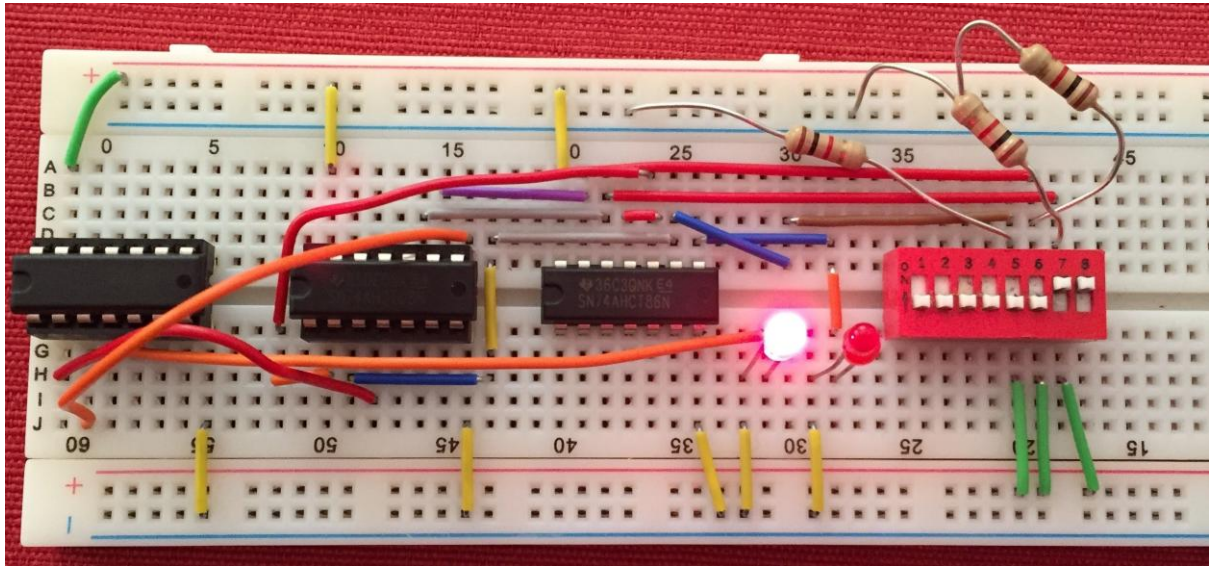
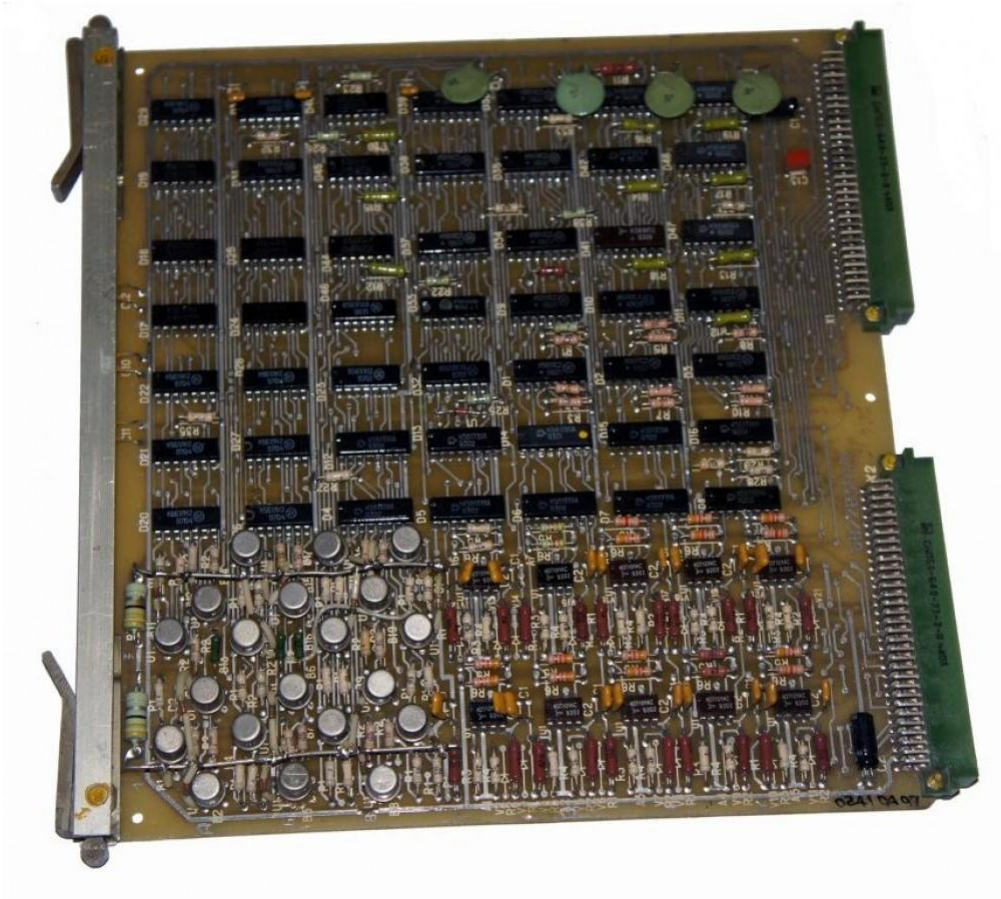
Комбинациялық тізбектерді талдау және синтездеу. Комбинациялық логикалық құрылғыларды жобалау.

Интеграция дәрежесі төмен микросұлбалар.

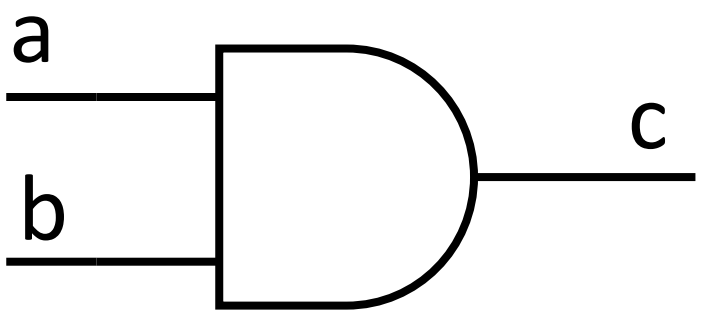




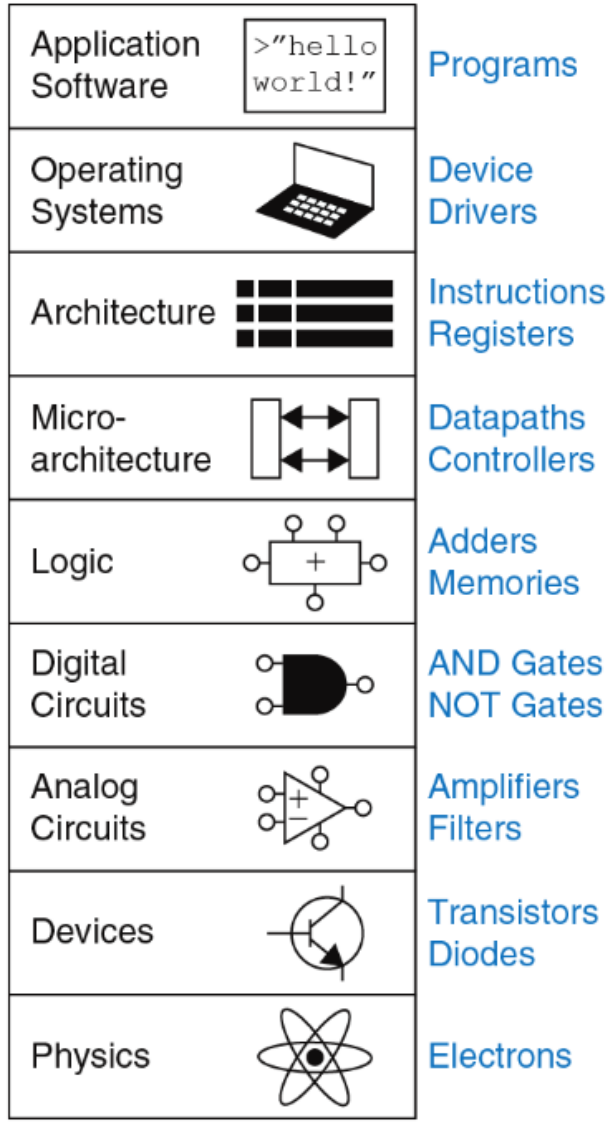


HDL

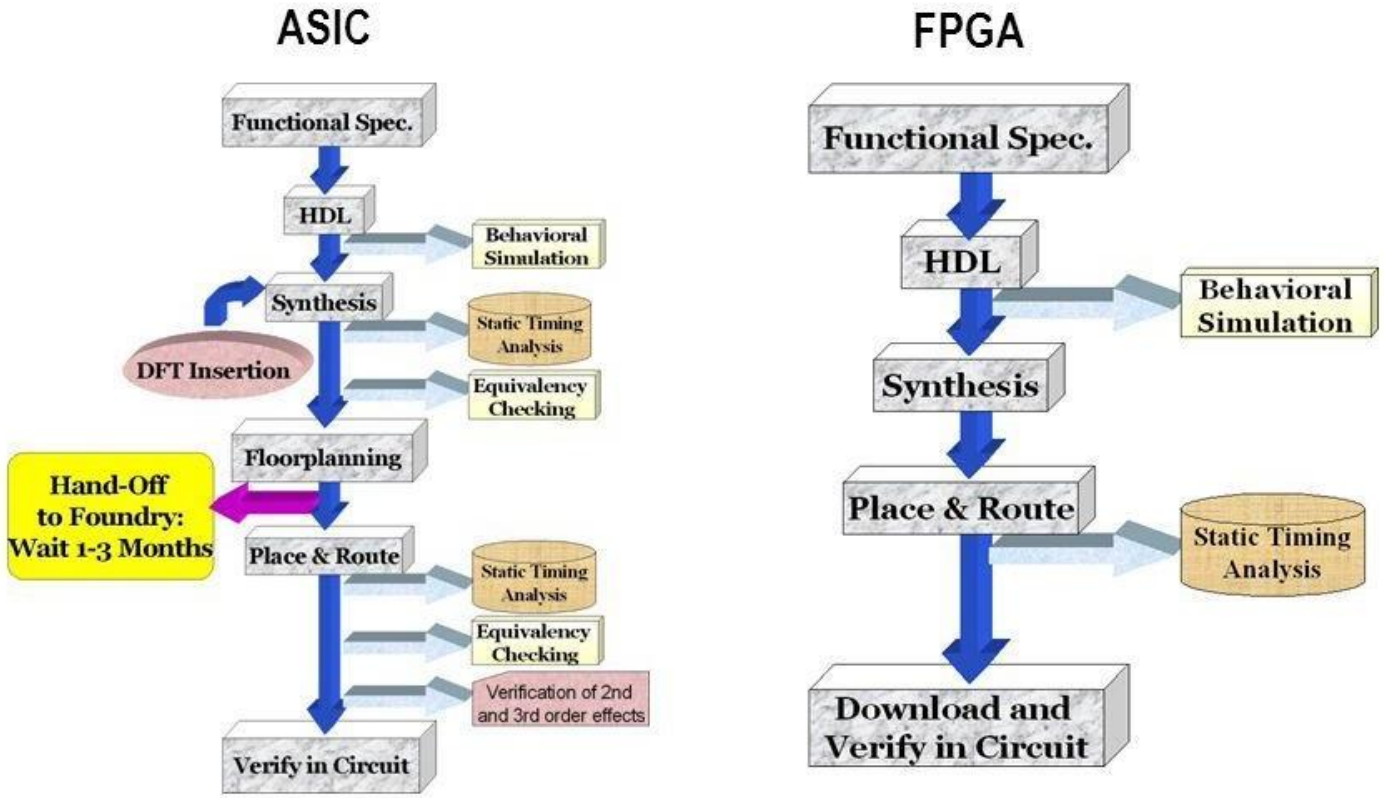
- HDL – Hardware Description Language



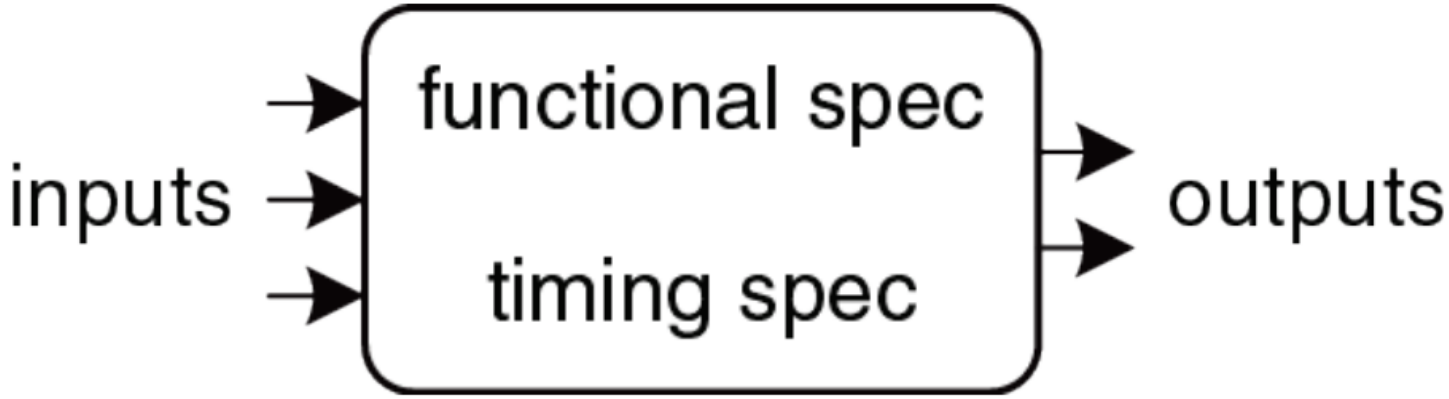
assign c = a & b;



HDL. Жобалау маршруты

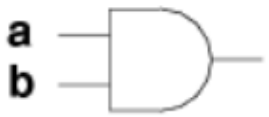


Комбинациялық логика



Комбинациялық логика

AND



a	b	out
0	0	0
0	1	0
1	0	0
1	1	1

OR

a	b	out
0	0	0
0	1	1
1	0	1
1	1	1

XOR

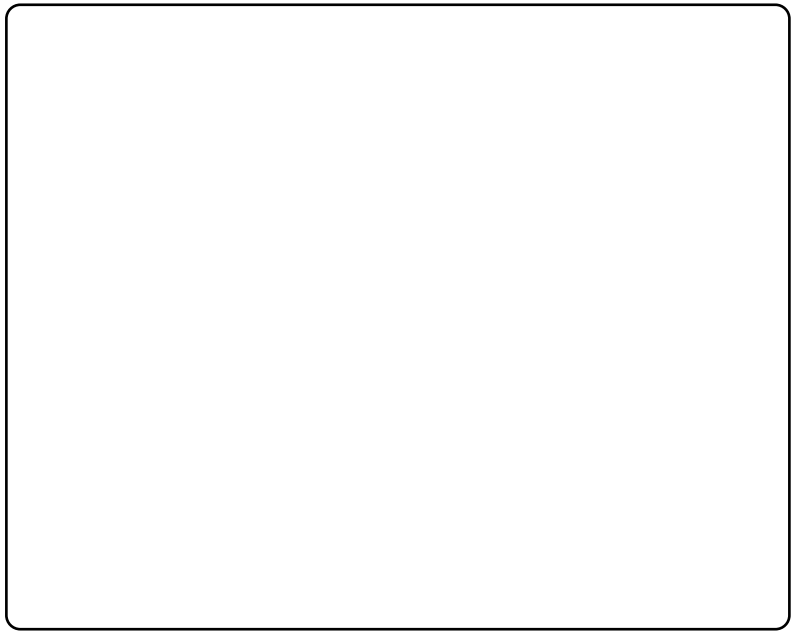
a	b	out
0	0	0
0	1	1
1	0	1
1	1	0

NOT

in	out
0	1
1	0

Verilog HDL

module

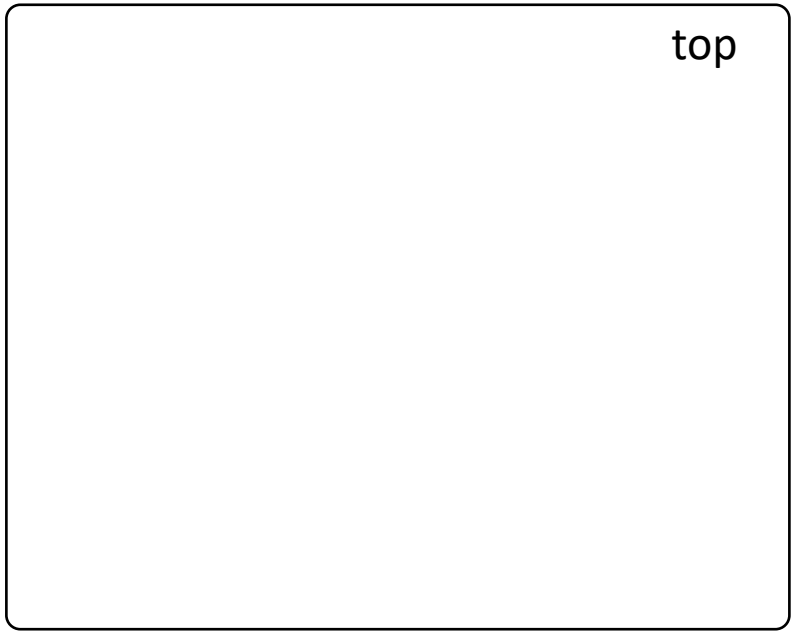


endmodule

Verilog HDL

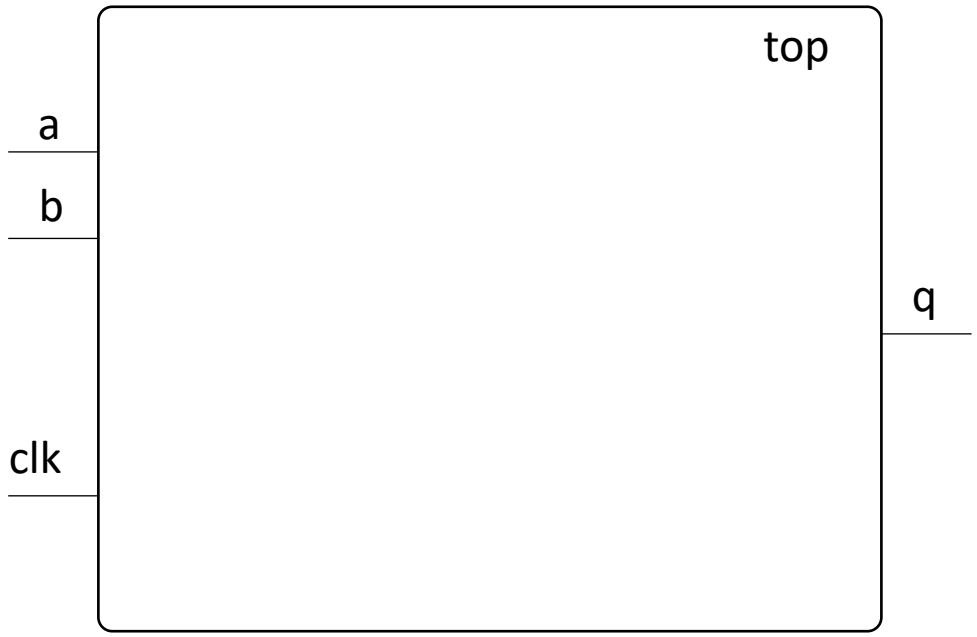
module top

endmodule



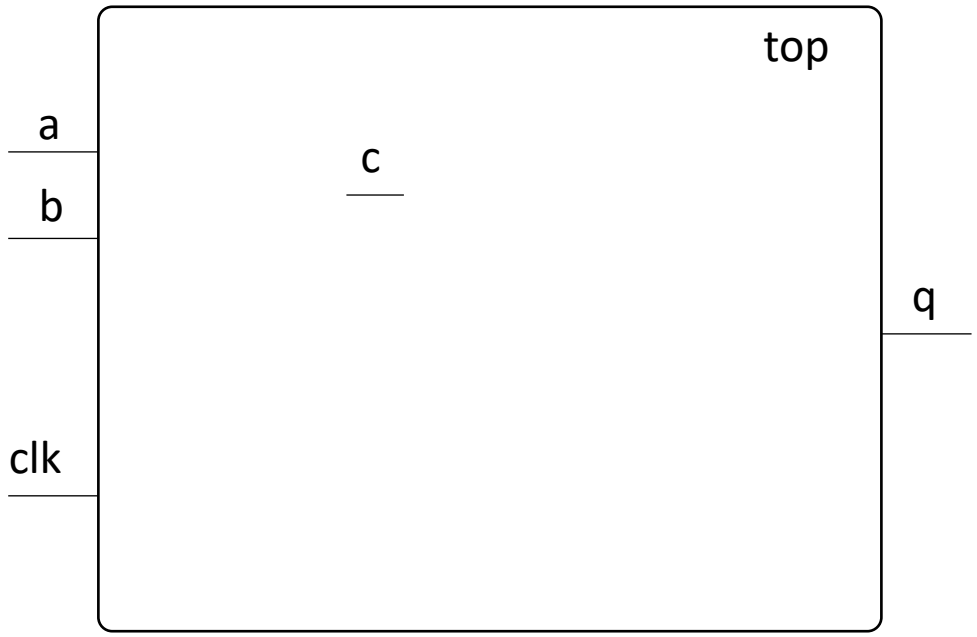
Verilog HDL

```
module top (
        input       clk,
        input       a,
        input       b,
        output      q
);




endmodule
```



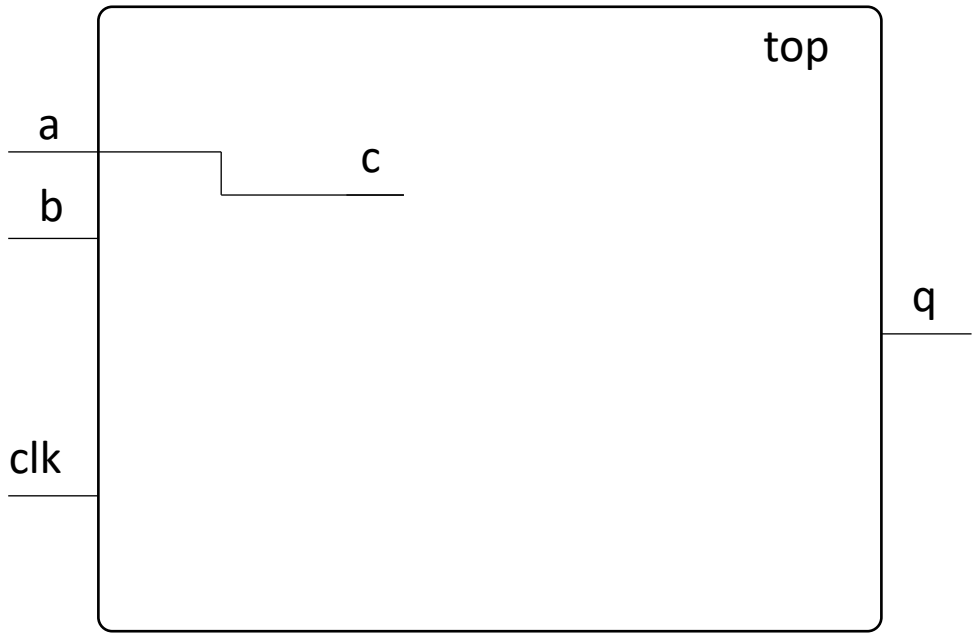
Verilog HDL

```
module top (
        input       clk,
        input       a,
        input       b,
        output      q
);

wire c;



endmodule
```



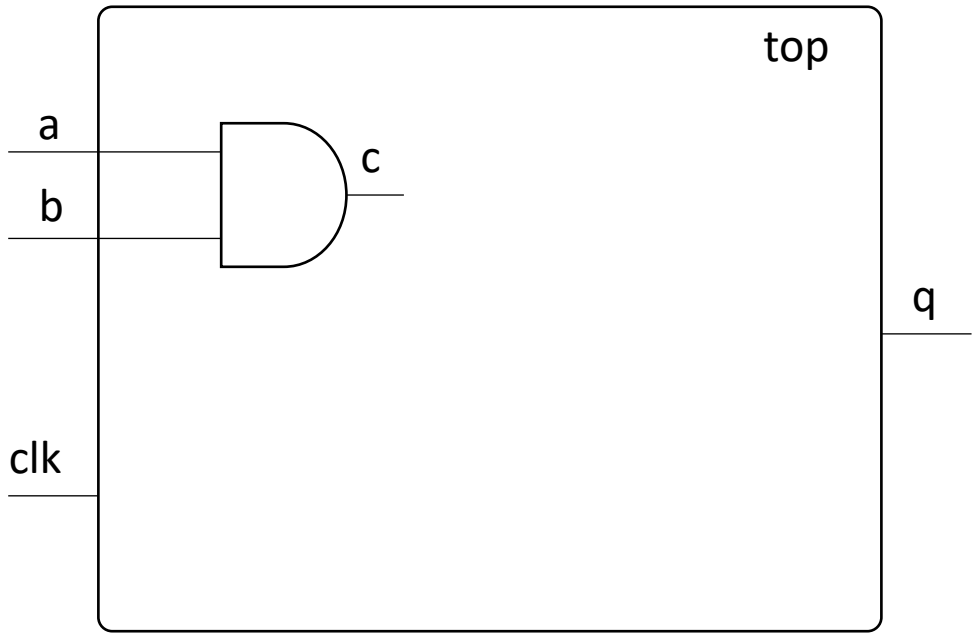
Verilog HDL

```
module top (
        input       a,
        input       b,
        input       clk,
        output      q
);

wire c;


assign c = a;

endmodule
```



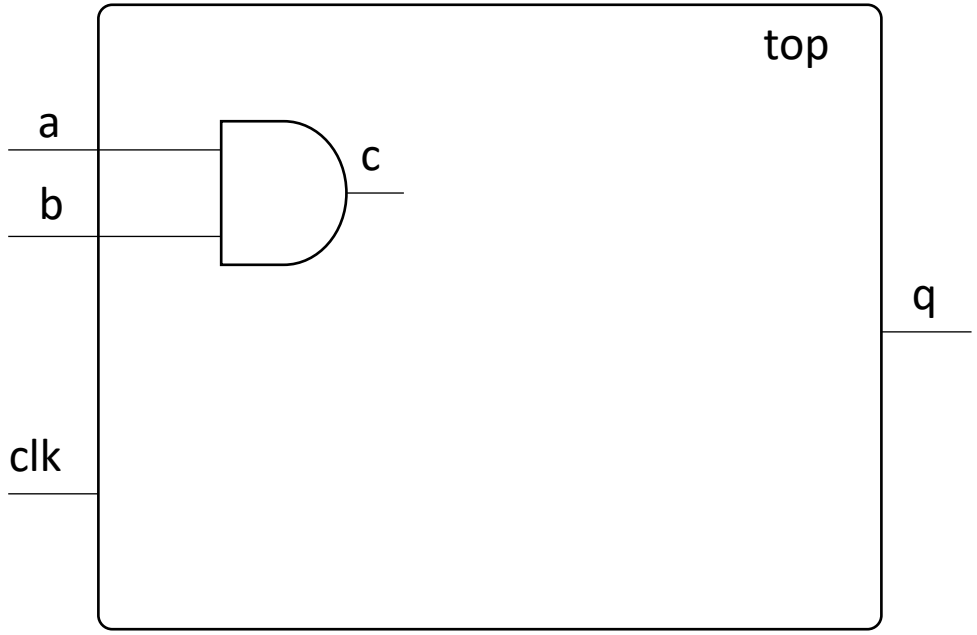
Verilog HDL

```
module top (
        input       clk,
        input       a,
        input       b,
        output      q
);

wire c;


assign c = a & b;

endmodule
```



Verilog HDL

```
module top (
        input       clk,
        input       a,
        input       b,
        output      q
);

wire c;

assign c = a & b;

endmodule
```



Verilog HDL

```
module top (
        input       clk,
        input       a,
        input       b,
        inout       q
);

wire c;

assign c = a & b;

endmodule
```

Verilog HDL

```
module top (
        input       clk,
        input       a,
        input       b,
        output      q
);

wire [3:0] c;
wire [3:0] d;

assign c = d;

endmodule
```

Verilog HDL

```
module top (
        input       clk,
        input       a,
        input       b,
        output      q
);

wire [3:0] c, d;

assign c = d;

endmodule
```

Verilog HDL

```
module top (
        input       clk,
        input       a,
        input       b,
        output      q
);

wire [3:0] c;
wire [8:0] d;

assign q = d[3];
endmodule
```

Verilog HDL

```
module top (
        input       clk,
        input       a,
        input       b,
        output      q
);

wire [3:0] c;
wire [8:0] d;

assign c = d[7:4];
endmodule
```

Verilog HDL

```
module top (
        input       clk,
        input       a,
        input       b,
        output      q
);

wire [3:0] c;
wire [8:0] d;

assign q = d[c];
endmodule
```

Verilog HDL

```
module top (
        input       clk,
        input       a,
        input       b,
        output      q
);

wire [3:0] c;
wire [7:0] d [0:24];

assign c = d[14][7:4];
endmodule
```

Verilog HDL

```
module top (
        input           clk,
        input   [8:0]   a,
        input           b,
        output  [3:0]   q
);

assign q = a[7:4];
endmodule
```

Verilog HDL сандарды сипаттау форматы

```
wire [10:0] a = 7;
wire [10:0] b = 11'd7;
wire [3:0]   c  = 4'b0101;
wire [3:0]   d  = 8'h7B;
wire [47:0] e = 48'hEFCA7ED98F;
```

Негізгі Verilog HDL операциялары

Символ	Мақсат
{}	Біріктіру (concatenation)
+ - * /	Арифметика (arithmetic)
%	Модуль (modulus)
> >= < <=	Қатынас (relational)
!	Логикалық терістеу(logical NOT)
&&	Логикалық ЖӘНЕ (logical AND)
||	Логикалық НЕМЕСЕ(logical OR)

Негізгі Verilog HDL операциялары

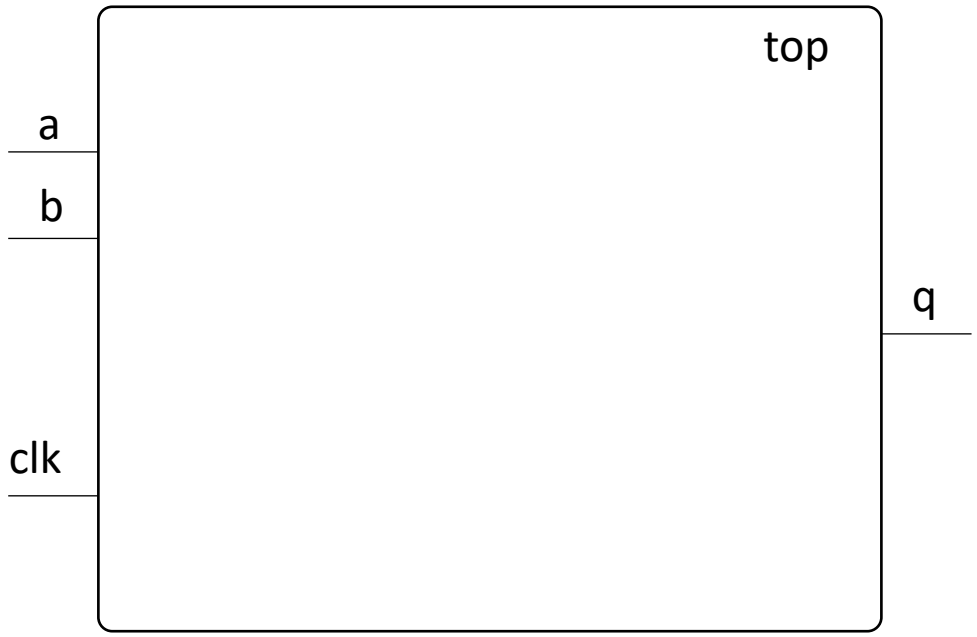
Символ	Мақсат
==	Логикалық теңдік(logical equality)
!=	Логикалық теңсіздік(logical inequality)
===	Сәйкестік (case equality)
!===	Сәйкестік емес(case inecuality)
~	Битті инверсия(bit-wise NOT)
&	Биттік ЖӘНЕ (bit-wise AND)
|	Биттік НЕМЕСЕ(bit-wise OR)

Негізгі Verilog HDL операциялары

Символ	Мақсат
<<	Солға жылжу(left shift)
>>	Оңға жылжу(right shift)
<<<	Циклдік солға жылжу(arithm. left shift)
>>>	Циклдік оңға жылжу(arithm. right shift)
?:	Үштік оператор(ternary)

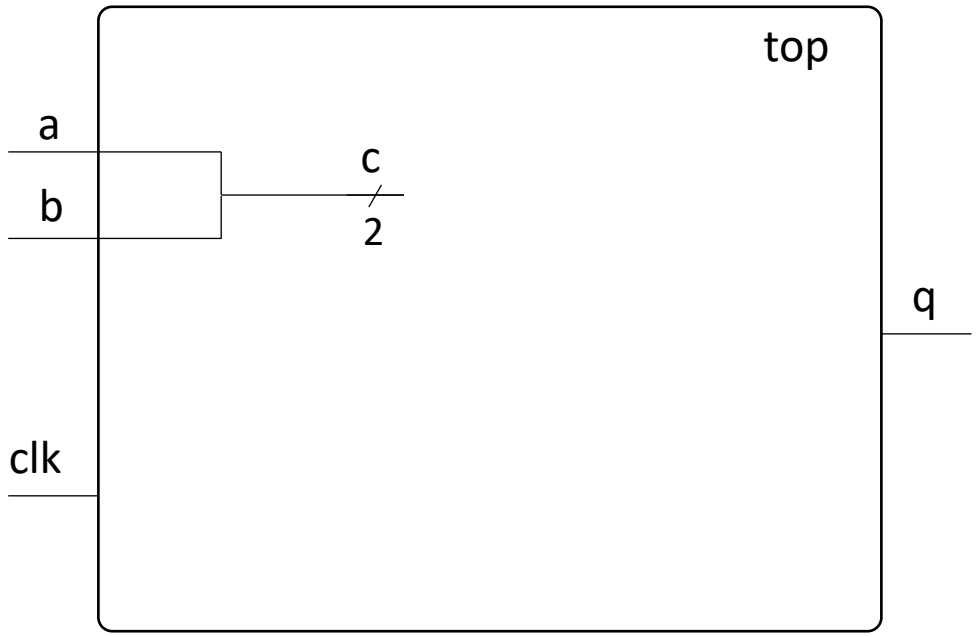
Verilog HDL бит манипуляциясы

```
module top (
        input       clk,
        input       a,
        input       b,
        output      q
);




endmodule
```



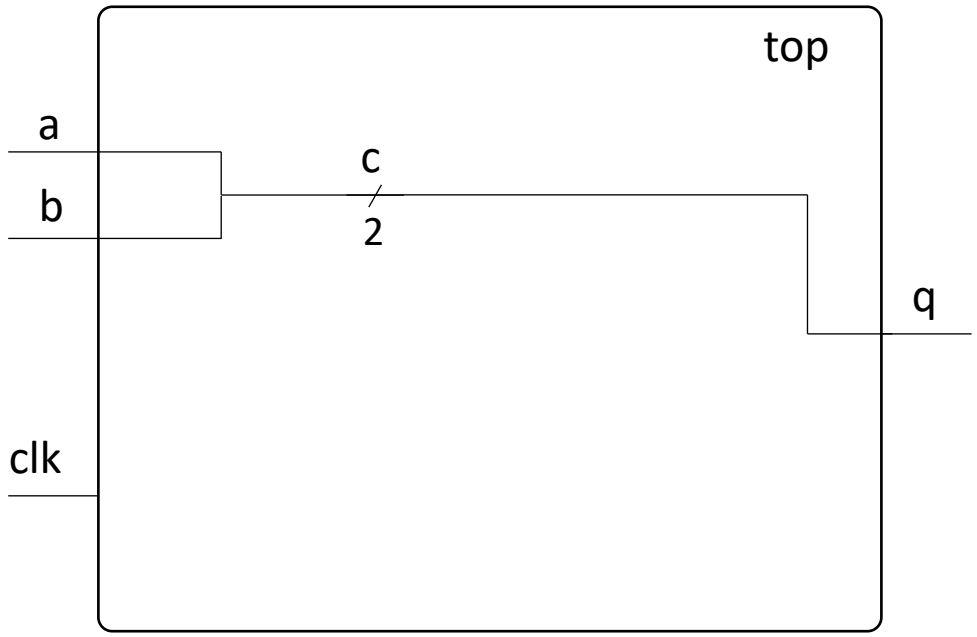
Verilog HDL бит манипуляциясы

```
module top (
        input       clk,
        input       a,
        input       b,
        output      q
);

wire [1:0] c;
assign c = {a,b};



endmodule
```



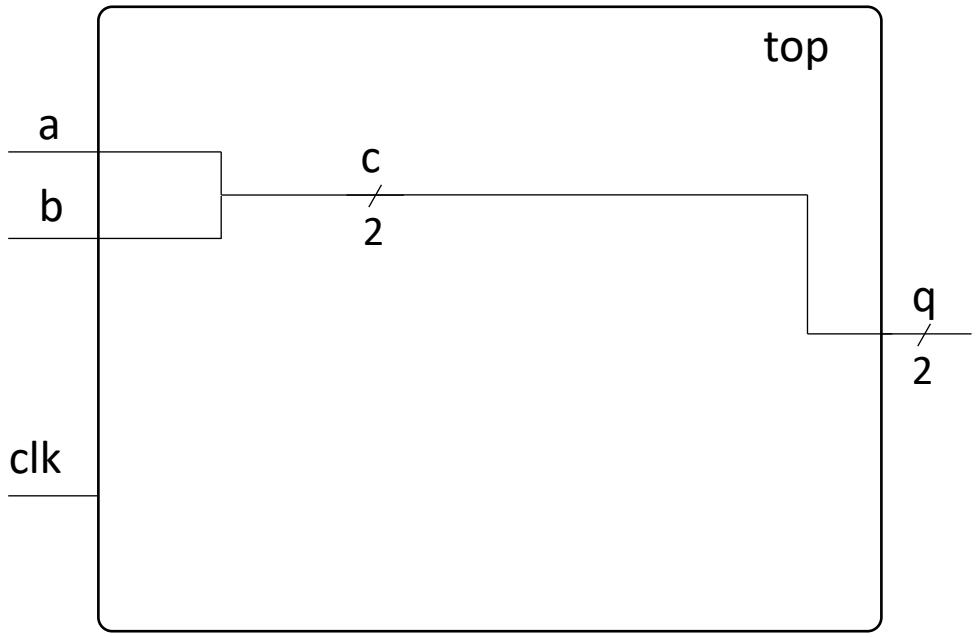
Verilog HDL бит манипуляциясы

```
module top (
        input       clk,
        input       a,
        input       b,
        output      q
);

wire [1:0] c;
assign c = {a,b};

assign q = c[0];

endmodule
```



Verilog HDL бит манипуляциясы

```
module top (
        input       clk,
        input       a,
        input       b,
        output      q
);

wire [1:0] c;
assign c = {a,b};

assign q = c;

endmodule
```



Verilog HDL. Қосу және азайту

```
module simple_add_sub (
        input    [7:0]    operandA, operandB
        output [8:0]    out_sum, out_dif
);

assign out_sum = operandA + operandB;
assign out_dif = operandA - operandB;

endmodule
```

Verilog HDL. Логикалық және арифметикалық ығысулар

```
module simple_add_sub (
        input    [7:0]    operandA, operandB
        output [7:0]    out_shl, out_shr, out_sar
);


assign   out_shl = operandA << operandB;

// мысал: operandA = 8'b1010_1110 operandB = 8'b0000_0011 out_shl = 8'b0001_0101
assign   out_shr = operandA >> operandB[2:0];

//operandA = 8'b1111_1100 out_shl = 8'b1111_1111
assign   out_sar = operandA >>> 3;

endmodule
```

Verilog HDL. Биттік логикалық операциялар

```
module simple_add_sub (
      input   [7:0]   operandA, operandB
      output [7:0]   out_bit_and , out_bit_or, out_bit_or, out_bit_not
);

assign  out_bit_and = operandA & operandB; //8'b0011_1101 & 8'b1010_0110 = 8'b0010_0100
assign  out_bit_or   = operandA | operandB;  //8'b0011_1101 | 8'b1010_0110 = 8'b1011_1111
assign  out_bit_or   = operandA ^ operandB;  //8'b0011_1101 ^ 8'b1010_0110 = 8'b1001_1011
assign  out_bit_not  = ~operandA;                  // ~8'b0011_1101 = 8'b1100_0010

endmodule
```

Verilog HDL. Бульдік логикалық операциялар

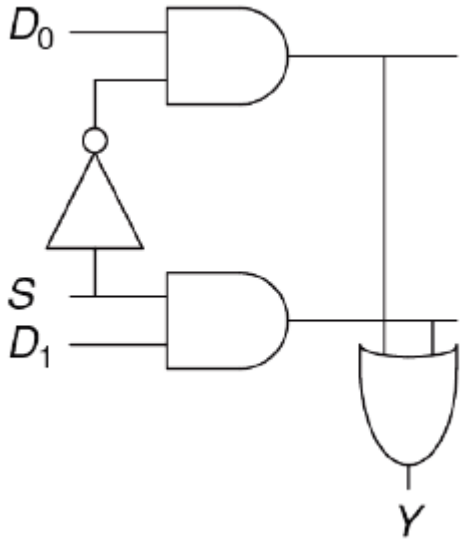
```
module simple_add_sub (
      input   [7:0]   operandA, operandB
      output          out_bool_and , out_bool_or  , out_bool_not
);

assign  out_bool_and = operandA && operandB; //8'b0011_1101 & 8'b1010_0110 = 1'b1
assign  out_bool_or   = operandA || operandB;   //8'b0011_1101 | 8'b1010_0110 = 1'b1
assign  out_bool_not  = !operandA;                  // !8'b0011_1101 = 1'b0

endmodule
```

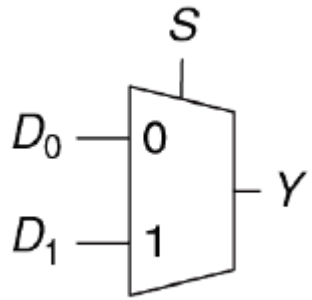
Verilog HDL. Конволюция операциялары

```
module simple_add_sub (
      input   [7:0]   operandA,
      output          out_reduction_and, out_reduction_or, out_redution_xor
);


assign  out_reduction_and = &operandA;  //&8'b0011_1101 = 1'b0
assign  out_reduction_or   = |operandA;  //|8'b0011_1101  = 1'b1
assign  out_redution_xor   = ^operandA;  // ^8'b0011_1101  = 1'b1

endmodule
```

Мультиплексор Verilog HDL

```
module mul (
        input       D0,
        input       D1,
        input       S,
        output      Y
);

assign Y = S ? D1 : D0;

endmodule
```





Verilog HDL. Салыстыру операциялары

```
module simple_add_sub (
      input   [7:0]   operandA, operandB
      output          out_eq, out_ne, out_gt, out_lt, out_ge, out_le
);

assign   out_eq = operandA == operandB;
assign   out_ne = operandA != operandB;
assign   out_ge = operandA >= operandB;
assign   out_le  = operandA <= operandB;
assign   out_gt  = operandA > operandB;
assign   out_lt   = operandA < operandB;


endmodule
```

Verilog HDL. Тип reg

```
module top (
        input           clk,
        input   [8:0]   a,
        input           b,
        output  [3:0]   q
);
reg [3:0] c;

endmodule
```

Verilog HDL. Тип reg

```
module top (
        input               clk,
        input       [8:0]   a,
        input               b,
        output reg [3:0]    q // выходные сигналы и шины можно объявлять как reg
);

endmodule
```

Verilog HDL. Блок always

```
module top (
        input           clk,
        input   [8:0]   a,
        input           b,
        output reg [3:0]  q
);
always @(*) begin
        q = !a[7:4];
end

endmodule
```

Verilog HDL. Блок always

```
wire   [3:0] a, b, c, d, e;
reg    [3:0] f, g, h, j;
always@(*) begin
f = a + b;
g = f & c;
h = g | d;
j = h - e;
end
```

Verilog HDL. Блок always

```
wire   [3:0] a, b, c, d, e;
reg    [3:0] f, g, h, j;
always@(*) begin
j = (((a + b) & c) | d) - e;
end
```

Verilog HDL. If-else

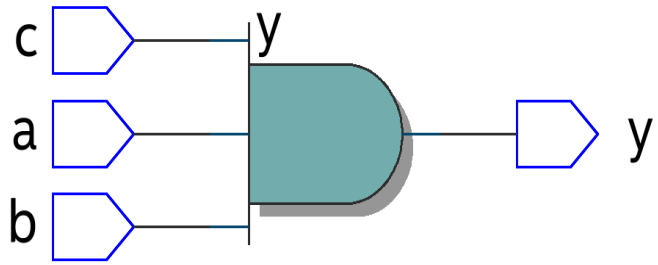
```
reg   [3:0] c;
always @(a or b or d) begin
      if (d)   begin
          c = a & b;
      end   else begin
          c = a + b;
      end
end
```

Verilog HDL. Case

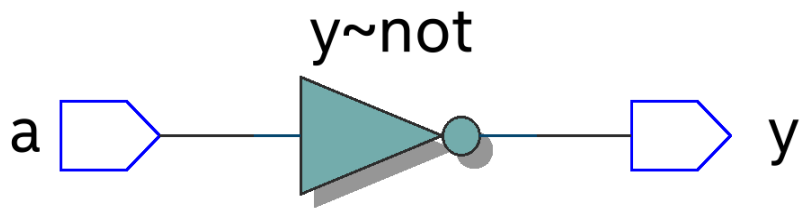
```
reg   [3:0] c;
wire   [1:0] option;
wire   [7:0] a, b, c, d;
reg     [7:0] e;
always @(a or b or c or d or option) begin
case (option)
  0: e = a;
  1: e = b;
  2: e = c;
  3: e = d;
endcase
end
```

Verilog HDL модульдерінің иерархиясы

```
module and_3 (
        input   a, b, c,
        output  y
);

assign y = a & b & c;

endmodule
```

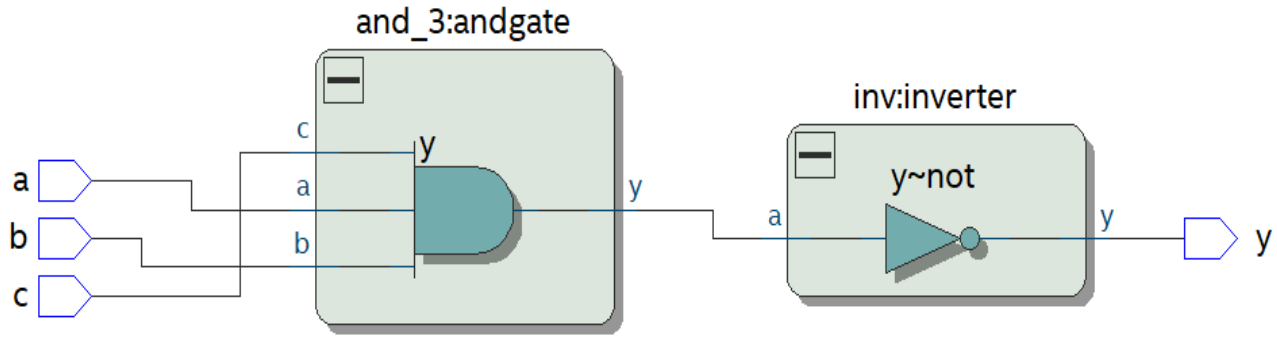



```
module inv (
        input   a
        output  y
);

assign y = ~a;

endmodule
```

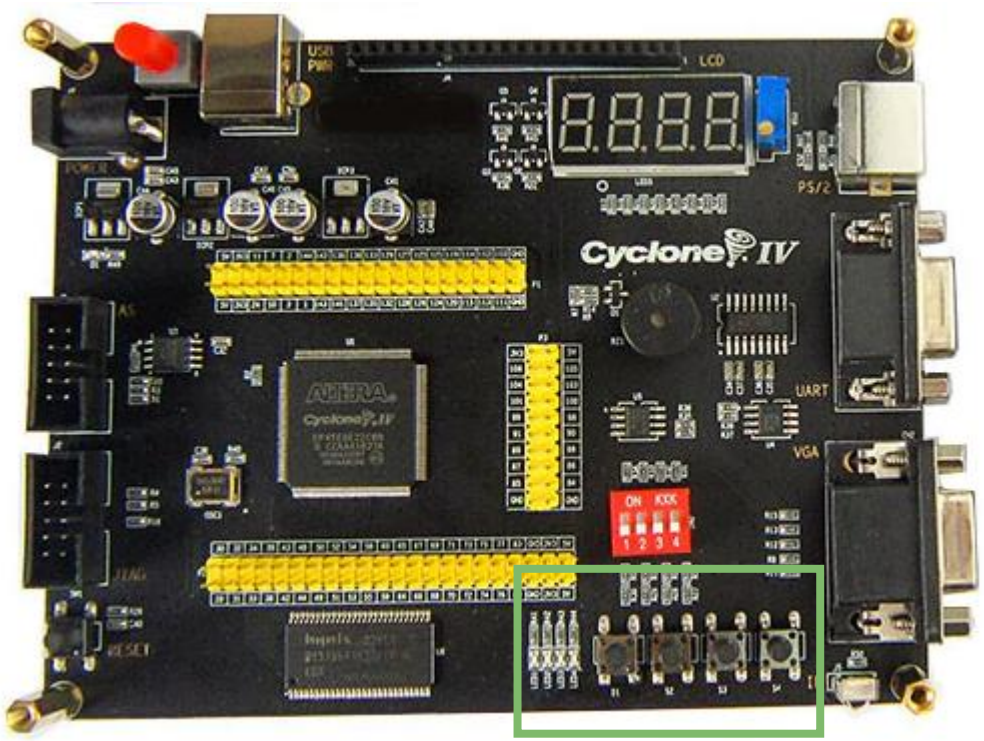


Verilog HDL модульдерінің иерархиясы

```
module top (
            input       a, b, c,
            output      y
);

wire n1;

and_3 andgate (
            .a(a), .b(b),
            .c(c), .y(n1)
);

inv inverter (
            .a(n1), .y(y)
);

endmodule
```



Логикалық қақпалармен жаттығу



Altera Cyclone IV EP4CE6 FPGA

Логикалық қақпалармен жаттығу

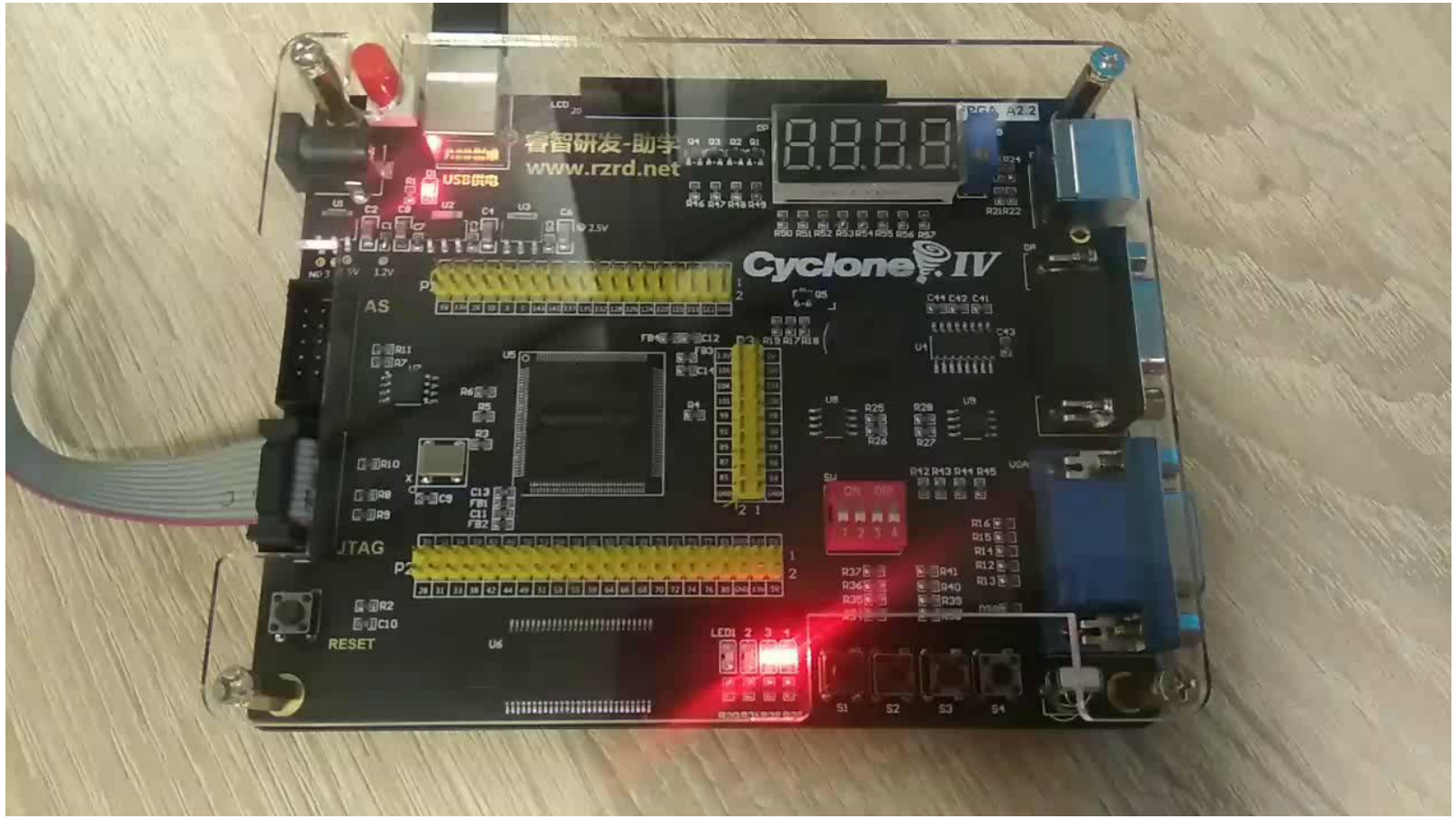
```
module top
(
    …
    input  [3:0] key_sw,
    output [3:0] led,
    …
);
    …

    // Exercise 1: Change the code below.
    // Assign to led [2] the result of AND operation

    assign led [2] = 1'b0;

endmodule
```

```
module top
(
    …
    input  [3:0] key_sw,
    output [3:0] led,
    …
);
    …
    // Exercise 2: Change the code below.
    // Assign to led [3] the result of XOR operation
    // without using "^" operation.
    // Use only operations "&", "|", "~" and parenthesis

    assign led [3] = 1'b0;

endmodule
```

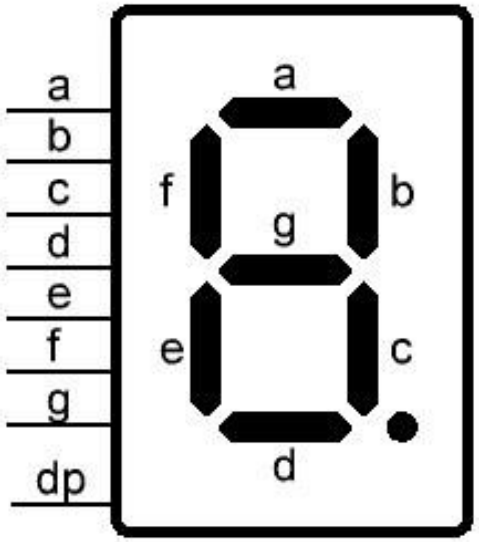
Логикалық қақпалармен жаттығу

1. Жарық диодын [2] ЖӘНЕ операциясының нәтижесіне тағайындаңыз.
2. Операторды пайдаланбай эксклюзивті НЕМЕСЕ (XOR) операциясының нәтижесіне led [3] тағайындаңыз.

Логикалық элементтермен жаттығу.

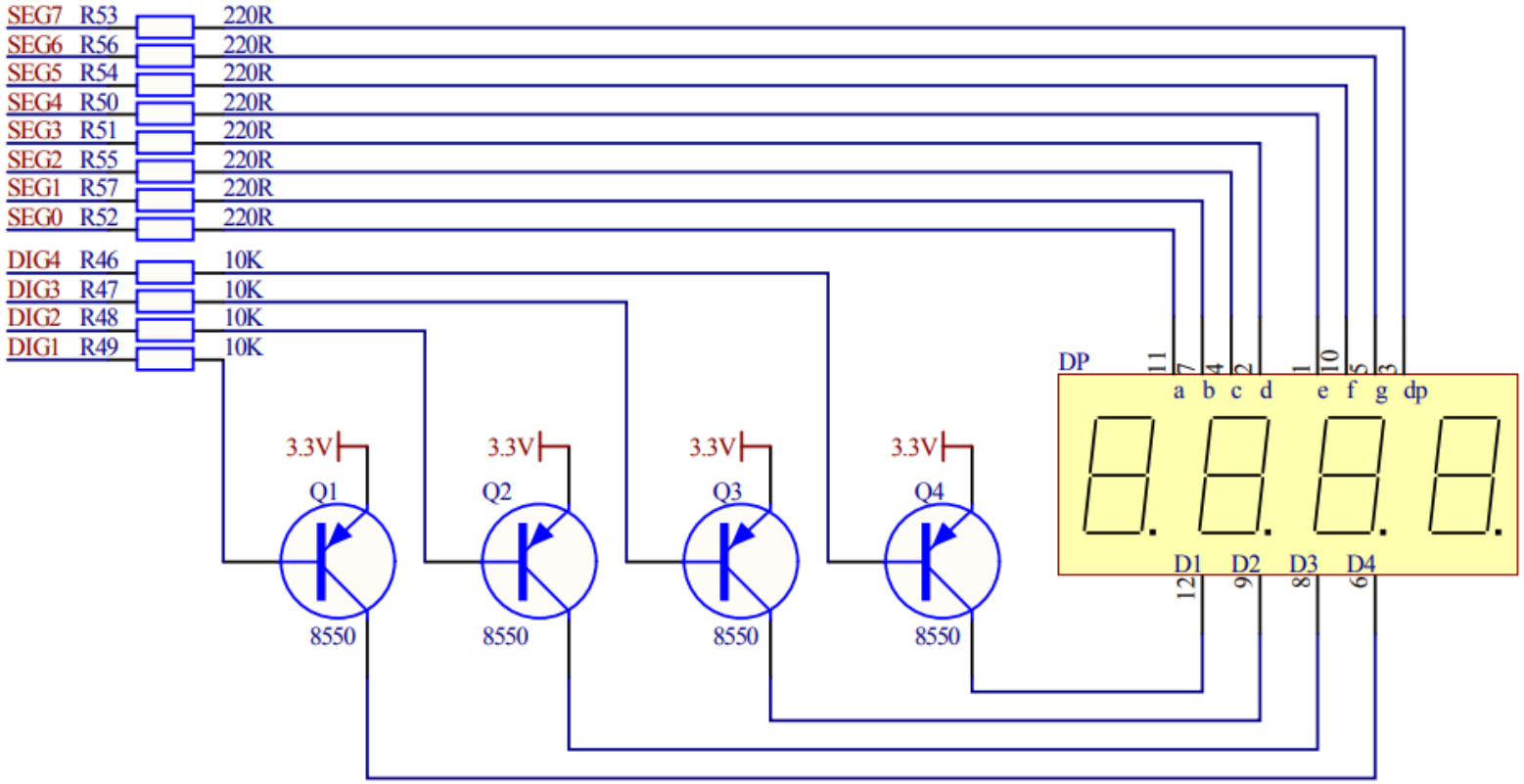


Жеті сегмент көрсеткіші

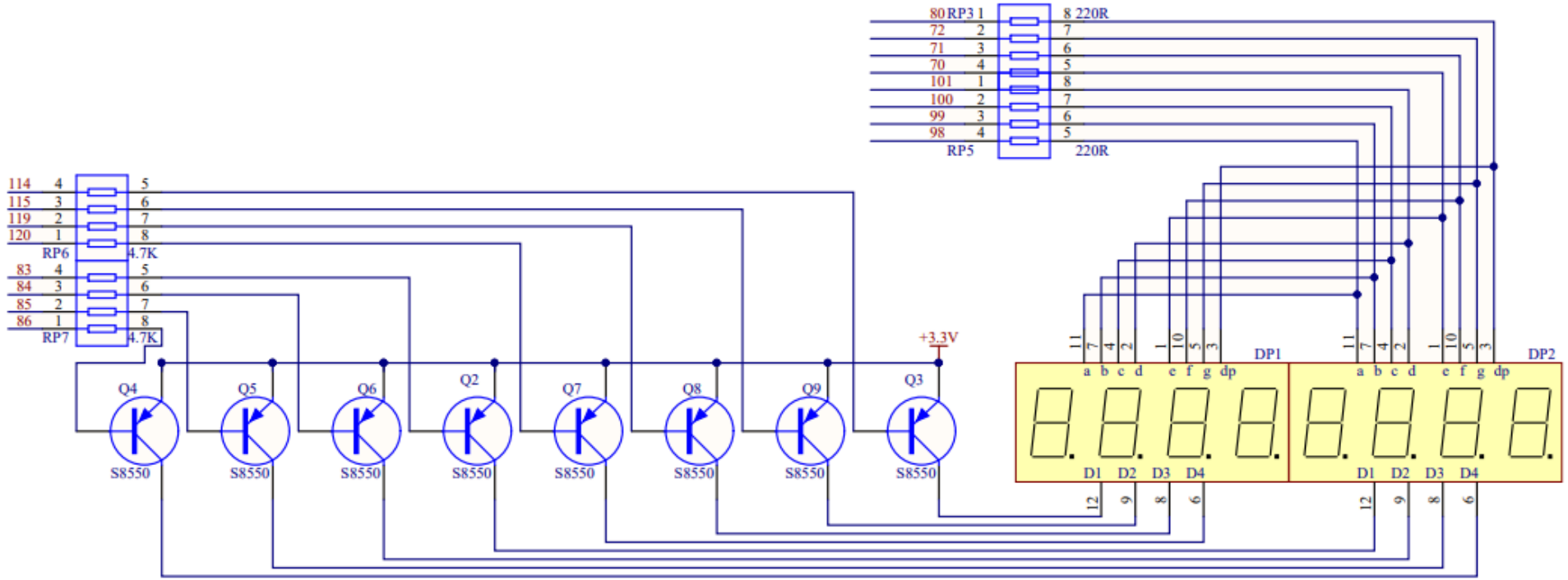


	4-bit шина				7-сегментный индикатор						
Dec	3	2	1	0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

Жеті сегмент көрсеткіші



Жеті сегмент көрсеткіші



Жеті сегментті индикаторда әріптерді көрсету арқылы жаттығу

```
module top
(
   …
   input  [3:0] key_sw,
   output [3:0] led,
   output [7:0] abcdefgh,
   output [3:0] digit,
   …
);
   …
   // Exercise 1: Display the first letters
   // of your first name and last name instead.
   assign abcdefgh =
   assign digit    =
endmodule
```

```
module top
(
   …
);
   …
   // Exercise 2: Display letters of a 4-character word
   // using this code to display letter of ChIP as an example
   reg [7:0] letter;
   always @*
     case (key_sw)
        4'b0111: letter = C;
        …
     endcase

   assign abcdefgh = letter;

endmodule
```

Жеті сегментті индикаторда әріптерді көрсету арқылы жаттығу

1. Жеті сегментті индикаторда атыңыз бен тегіңіздің бірінші әріптерін көрсетіңіз.
2. Жеті сегментті индикаторда CHIP сөзін көрсетіңіз.

Жеті сегментті индикаторда әріптерді көрсету арқылы жаттығу.