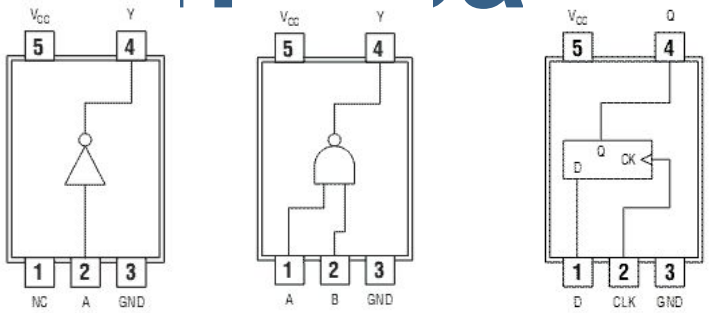
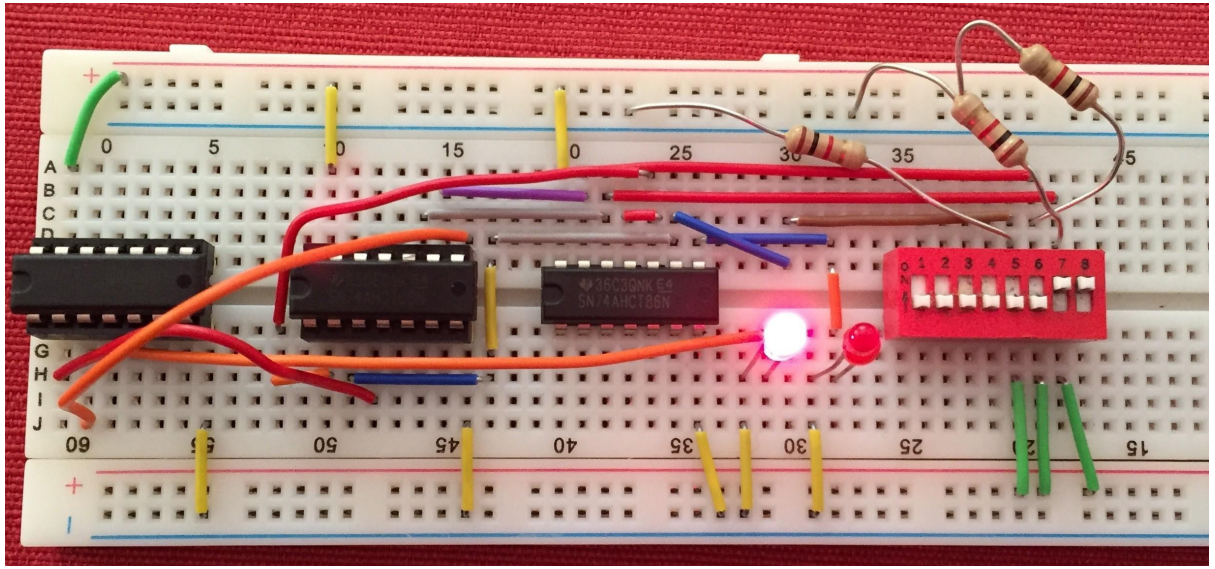
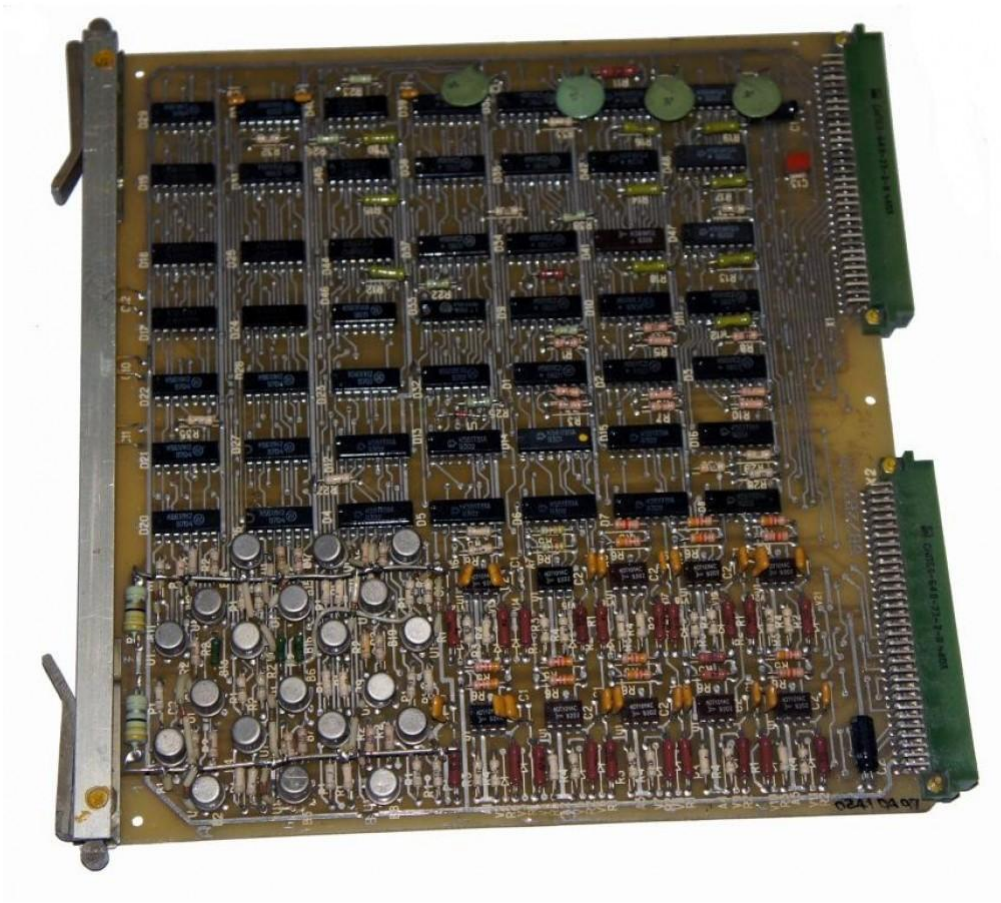
Комбинационная логика и ее описание на языке Verilog

Микросхемы малой степени интеграции





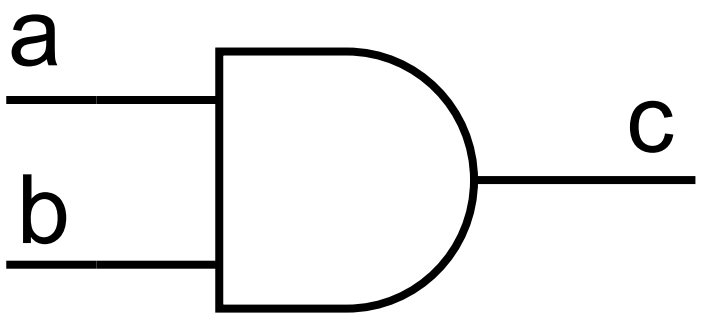


MIET NATIONAL RESEARCH UNIVERSITY OF ELECTRONIC TECHNOLOGY

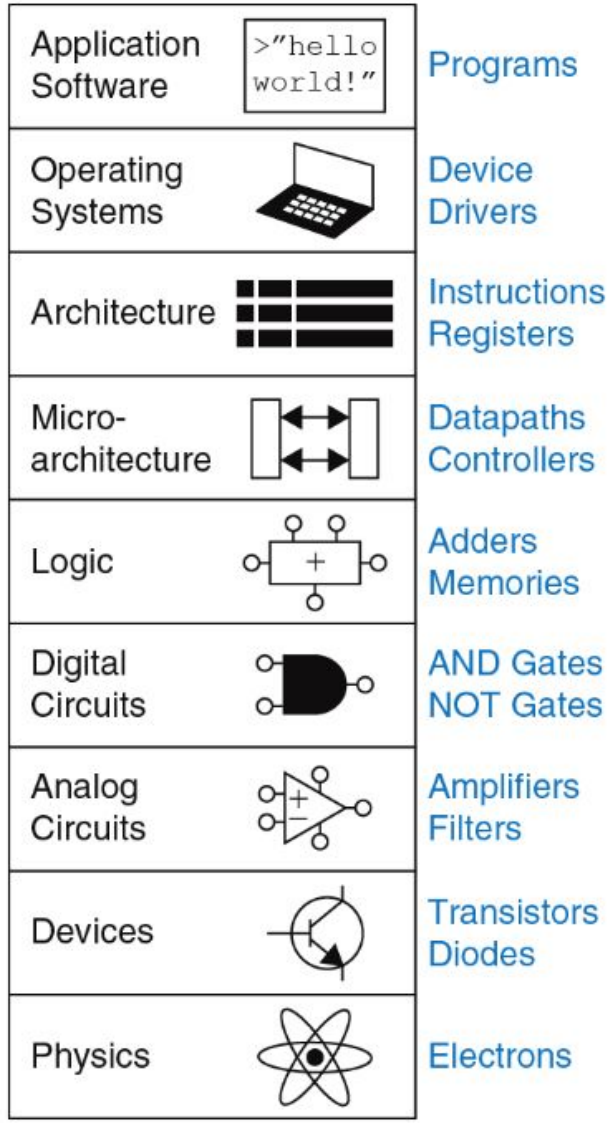
Управление сложностью в микроэлектронике

HDL

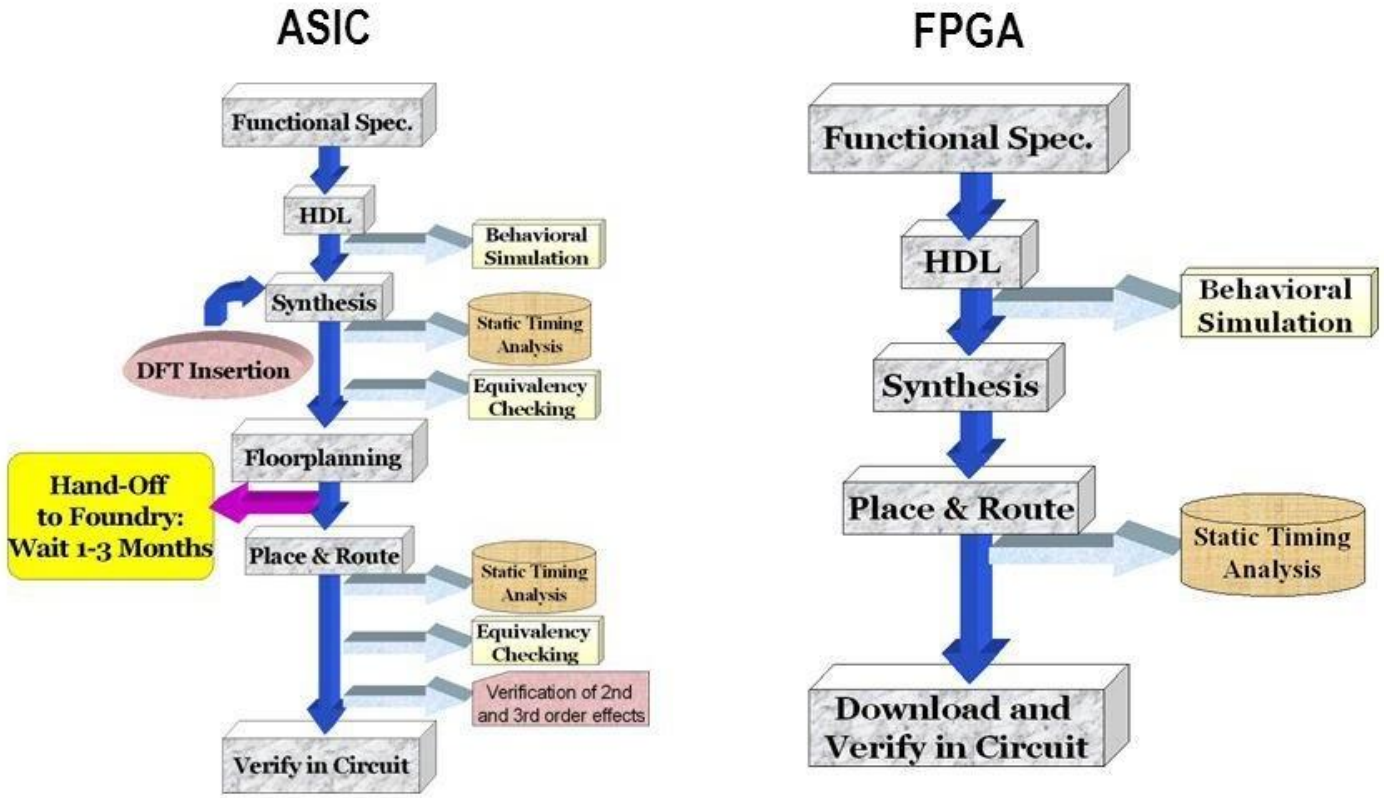
- HDL – Hardware Description Language




```
assign c = a &
b;
```



HDL. Маршрут проектирования



HDL. Выбор

Verilog VHDL

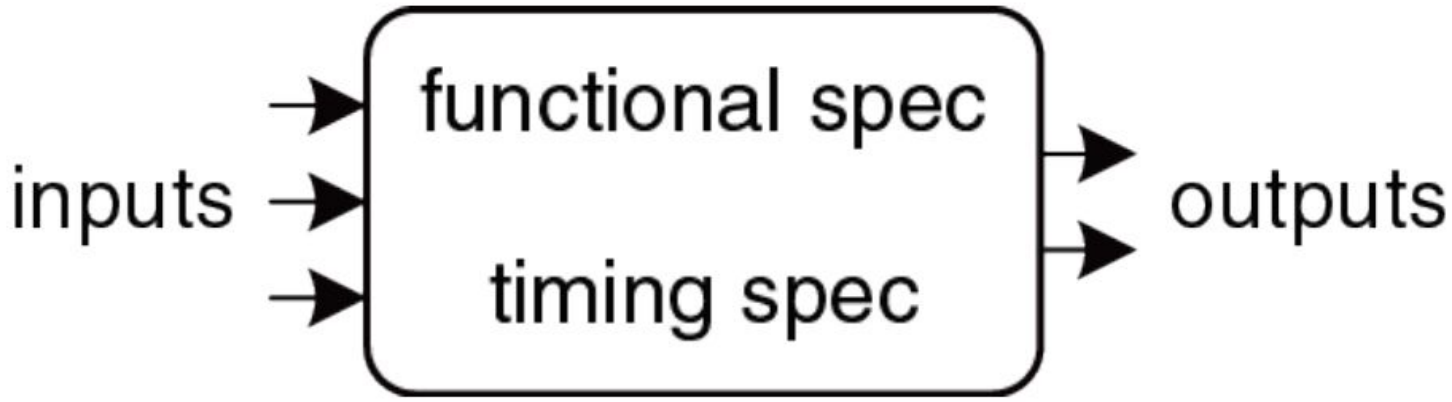
Verilog HDL. История

- Verilog был разработан компанией Gateway Design Automation в 1984 году как фирменный язык для симуляции логических схем.
- В 1989 году Gateway приобрела компания Cadence, и Verilog стал открытым стандартом в 1990 году под управлением сообщества Open Verilog International.

Verilog HDL. История

- Язык стал стандартом IEEE в 1995 году.
- В 2005 году язык был расширен для упорядочивания и лучшей поддержки моделирования и верификации систем.
- Эти расширения были объединены в единый стандарт, который сейчас называется SystemVerilog(стандарт IEEE 1800-2009).

Комбинационная логика

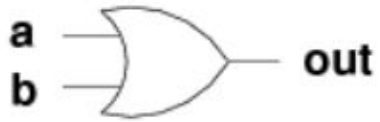


Комбинационная логика

AND

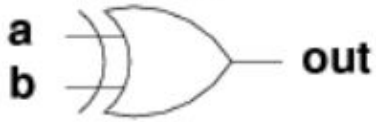
a	b	out
0	0	0
0	1	0
1	0	0
1	1	1

OR



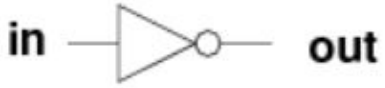
a	b	out
0	0	0
0	1	1
1	0	1
1	1	1

XOR



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

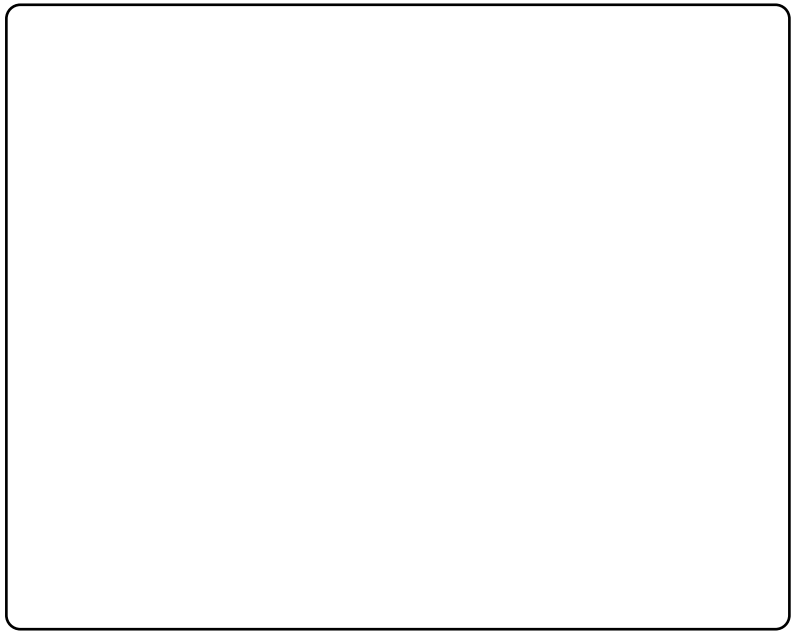
NOT



in	out
0	1
1	0

Verilog HDL

module

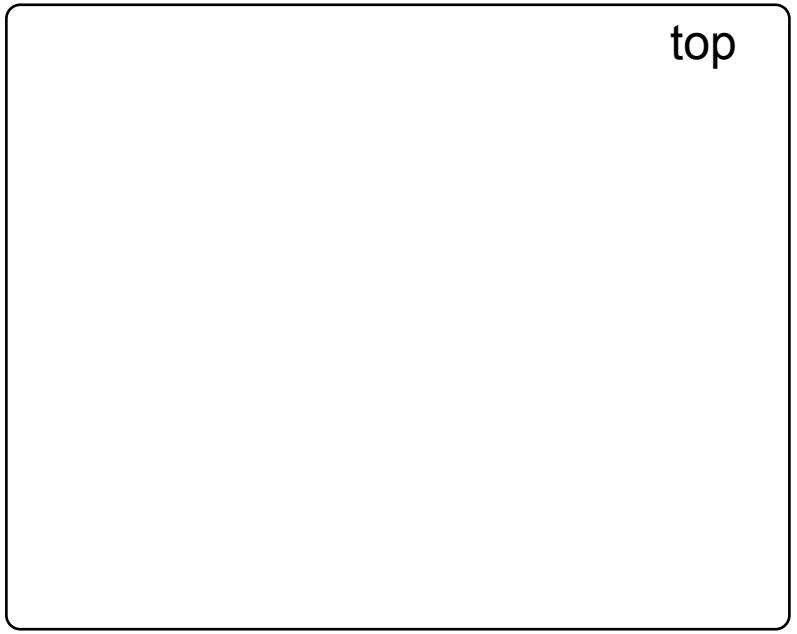


endmodule

Verilog HDL

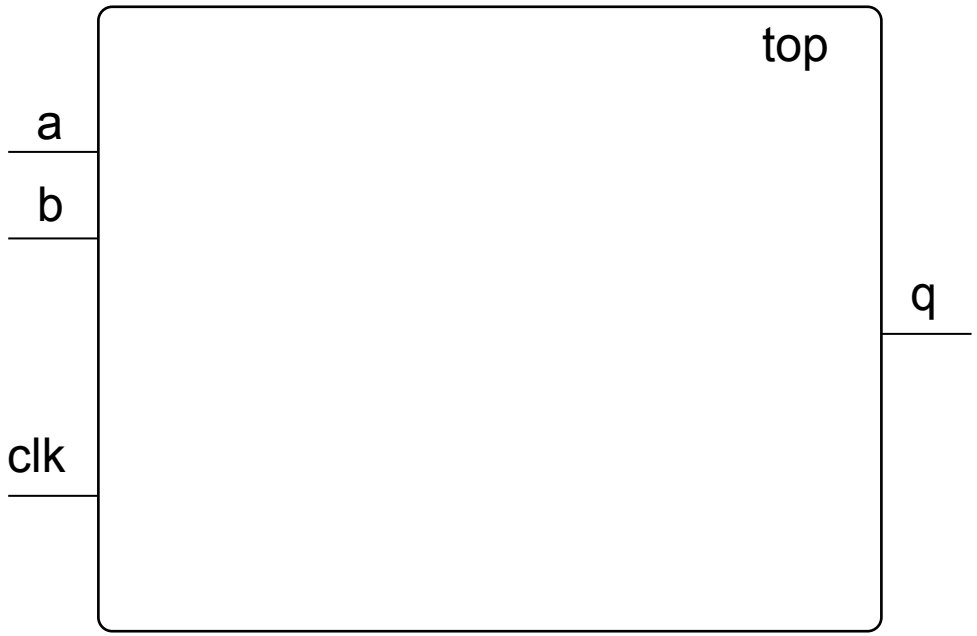
module top

endmodule



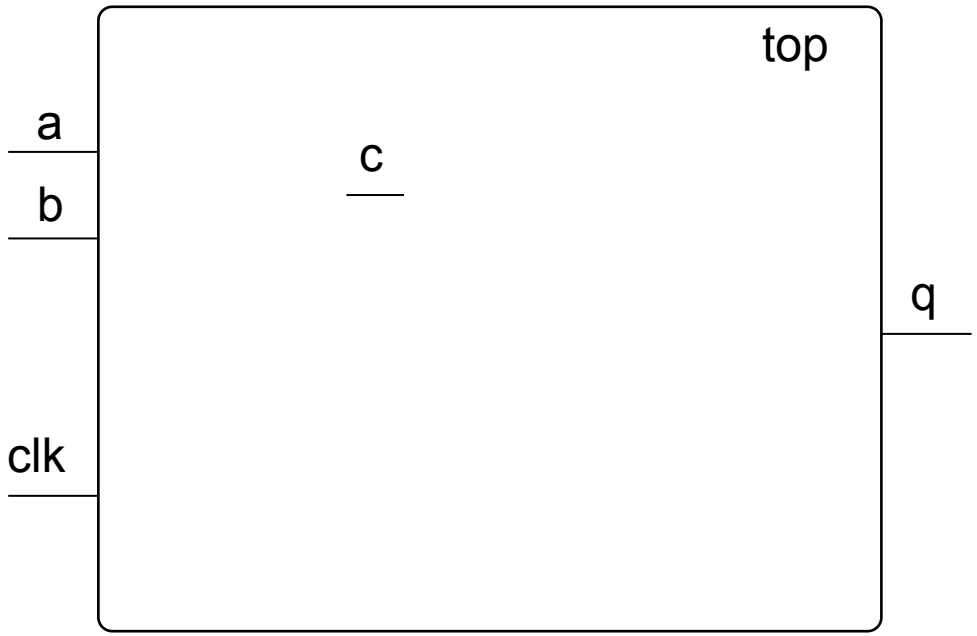
Verilog HDL

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




endmodule
```



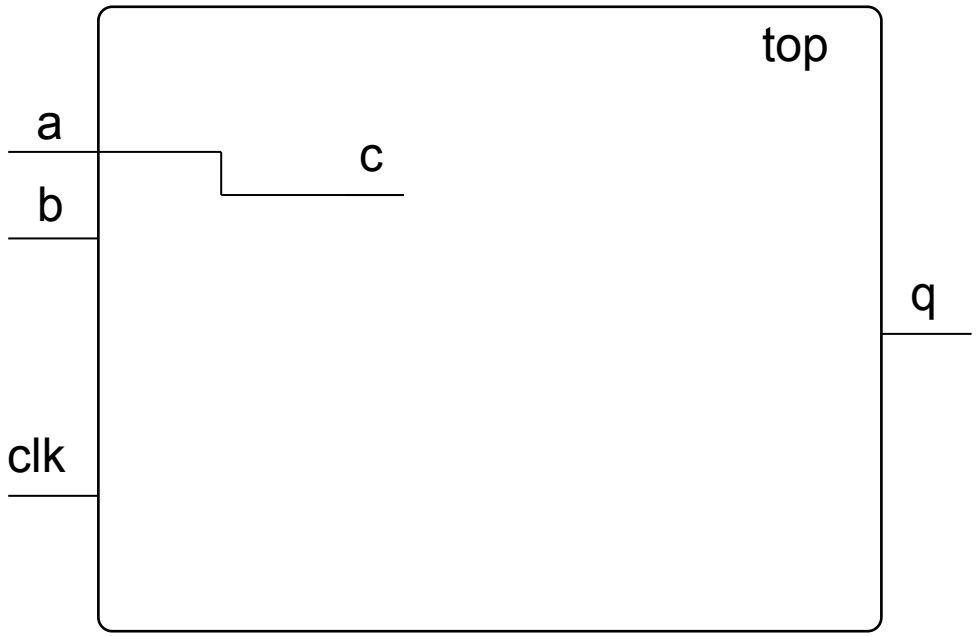
Verilog HDL

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

wire c;



endmodule
```



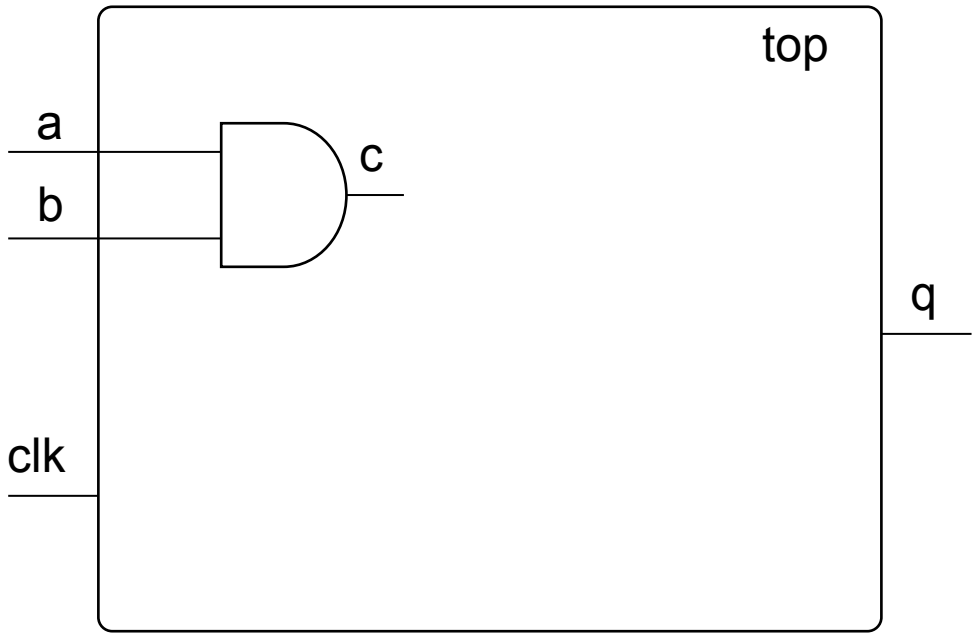
Verilog HDL

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

wire c;


assign c = a;

endmodule
```



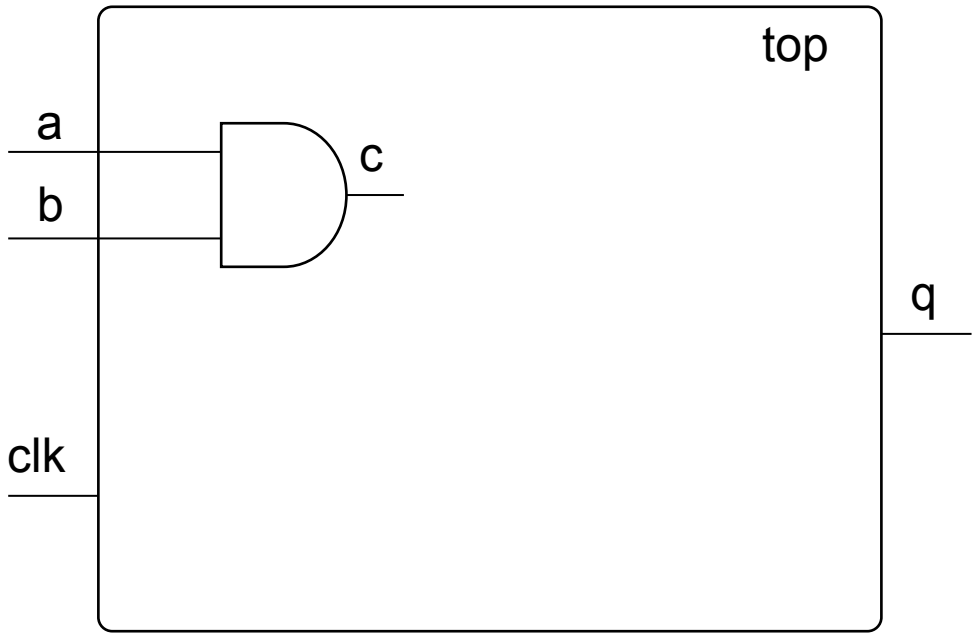
Verilog HDL

```
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;

// Это комментарий, он ничего не изменяет
assign c = a & b;

endmodule
```



MIET NATIONAL RESEARCH UNIVERSITY OF ELECTRONIC TECHNOLOGY

Verilog HDL

```
module top (
        input           clk,
        input           a,
        input           b,
        inout           q     // двунаправленный сигнал, используется для внешних контактов микросхем
);

wire c;

// Это комментарий, он ничего не изменяет
assign c = a & b;

endmodule
```

Verilog HDL

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

wire [3:0] c; // 4-х проводная шина
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; // объявление нескольких проводов одинаковой разрядности

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]; // можно присвоить нужный бит.

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]; // можно назначить нужные биты другому проводу.

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]; // можно назначить номер бита определяемой шиной.

endmodule
```

Verilog HDL

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

wire [3:0] c;
wire [7:0] d [0:24]; // массив из двадцатипяти 8-битных шин

assign c = d[14][7:4]; // подключение 4 бит из 14 шины массива шин.

endmodule
```

Verilog HDL

```
module top (
        input           clk,
        input     [8:0]a, // входные и выходные порты модуля могут быть многоразрядными шинами.
        input           b,
        output    [3:0]q
);

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

Формат описания чисел Verilog HDL

```
wire [10:0] a = 7; //32-х битное десятичное число, которое будет “обрезано” до 11 бит
wire [10:0] b = 11’d7; //11-ти битное десятичное число
wire [3:0]   c  = 4'b0101; //4-х битное двоичное число
wire [3:0]   d  = 8’h7B;   //8-ми битное шестнадцатеричное число 7B
wire [47:0] e = 48’hEFCA7ED98F;  //48-ми битное шестнадцатеричное число
```

Основные операции Verilog HDL

Символ	Назначение
{}	Конкатенация (concatenation)
+ - * /	Арифметические (arithmetic)
%	Модуль (modulus)
> >= < <=	Отношения (relational)
!	Логическое отрицание (logical NOT)
&&	Логическое И (logical AND)
||	Логическое ИЛИ (logical OR)

Основные операции Verilog HDL

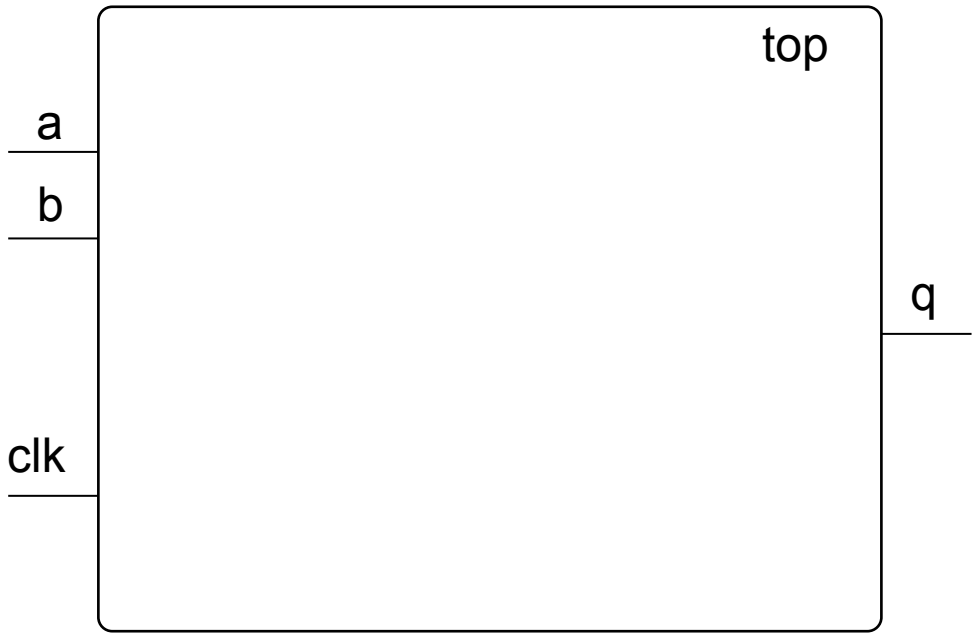
Символ	Назначение
==	Логическое равенство (logical equality)
!=	Логическое неравенство (logical inequality)
===	Идентичность (case equality)
!===	Не идентичность (case inecuality)
~	Побитовая инверсия (bit-wise NOT)
&	Побитовое И (bit-wise AND)
|	Побитовое ИЛИ (bit-wise OR)

Основные операции Verilog HDL

Символ	Назначение
<<	Сдвиг влево (left shift)
>>	Сдвиг вправо (right shift)
<<<	Циклический сдвиг влево (arithm. left shift)
>>>	Циклический сдвиг вправо (arithm. right shift)
?:	Тернарный оператор (ternary)

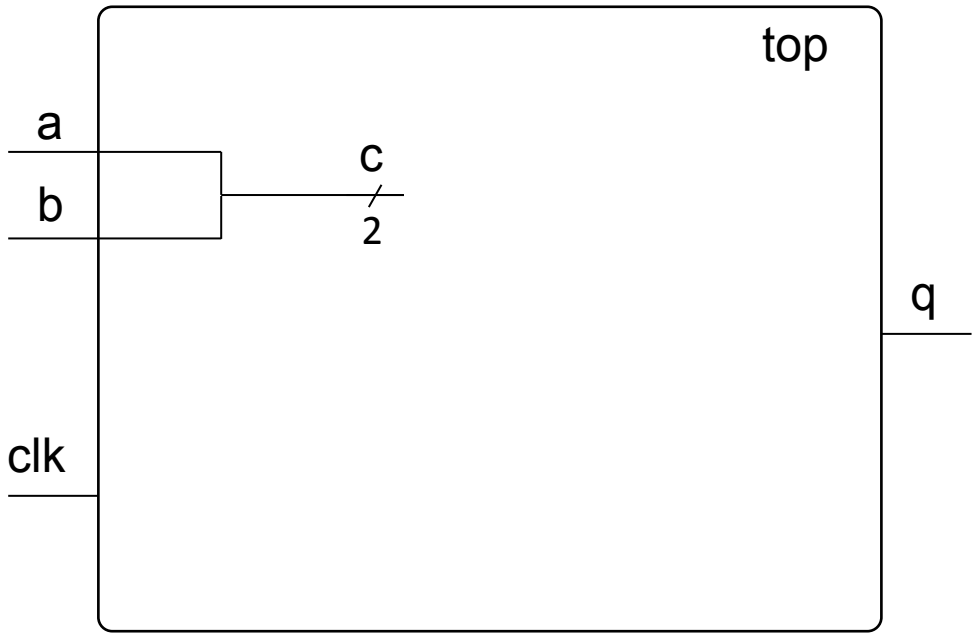
Манипуляции с битами Verilog HDL

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




endmodule
```



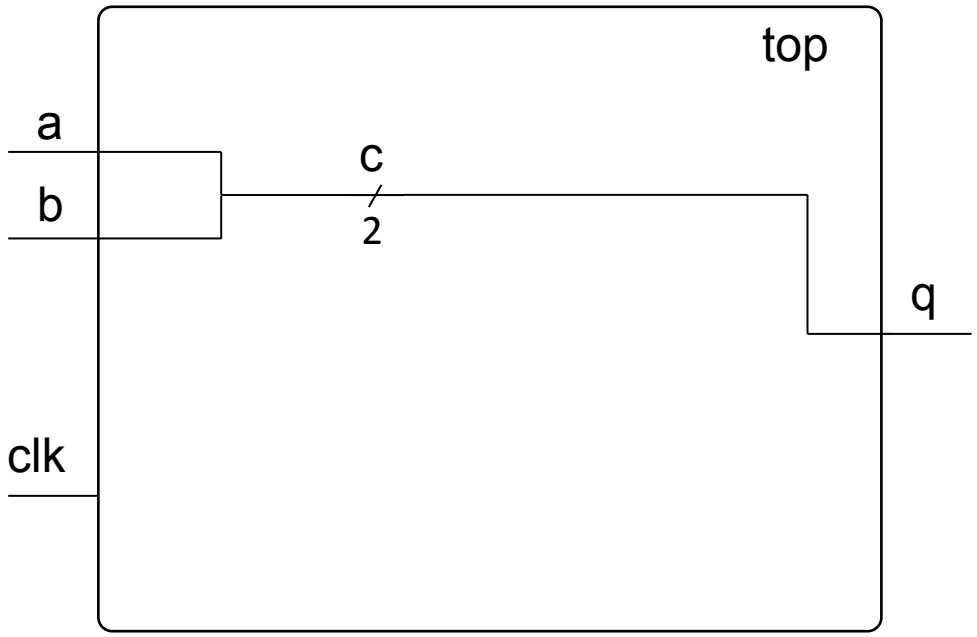
Манипуляции с битами Verilog HDL

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

wire [1:0] c; // Многобитный сигнал (шина)

assign c = {a,b}; // Конкатенация



endmodule
```



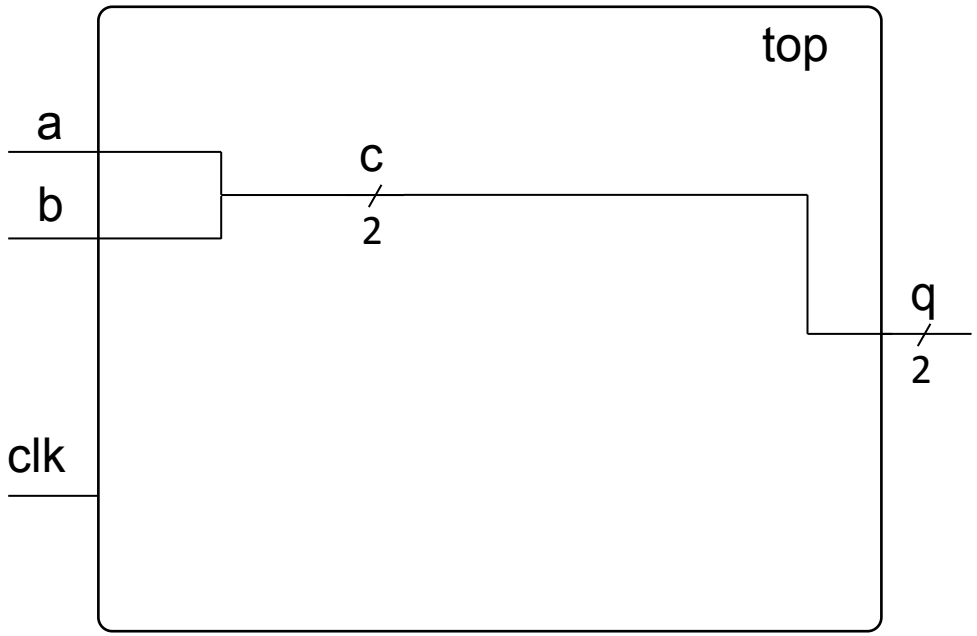
Манипуляции с битами Verilog HDL

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

wire [1:0] c; // Многобитный сигнал (шина)

assign c = {a,b}; // Конкатенация

assign q = c[0]; // Обращение к биту

endmodule
```



Манипуляции с битами Verilog HDL

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

wire [1:0] c; // Многобитный сигнал (шина)

assign c = {a,b}; // Конкатенация

assign q = c;

endmodule
```



Verilog HDL. Сложение и вычитание

```
module simple_add_sub (
        input    [7:0]    operandA, operandB //два входных 8-ми битных операнда
        output [8:0]    out_sum, out_dif        // Выходы для арифметических операций имеют дополнительный 9-й бит переполнения
);
//Максимальное 8-ми битное безнаковое число 255. При сложении 255 + 255 получится 510 которое можно представить минимум 9-ю битами.
assign out_sum = operandA + operandB;
assign out_dif = operandA - operandB;
//Если сделать выход сумматора 8-ми битным то случится переполнение и результат сложения 255 + 3 будет равен 2.
endmodule
```

Verilog HDL. Логические и арифметические сдвиги

```
module simple_add_sub (
        input    [7:0]    operandA, operandB //два входных 8-ми битных операнда
        output [7:0]    out_shl, out_shr, out_sar //Выходы для операций сдвига
);

//логический сдвиг влево на значение в  operandB.
assign  out_shl = operandA << operandB;

// пример: на сколько сдвигать определяется 3-мя битами второго операнда. Например operandA = 8'b1010_1110 operandB = 8'b0000_0011 тогда out_shl = 8'b0001_0101
assign  out_shr = operandA >> operandB[2:0];
//арифметический сдвиг вправо (сохранение знака числа) Например operandA = 8'b1111_1100 тогда out_shl = 8'b1111_1111
assign  out_sar = operandA >>> 3;

endmodule
```

MIET NATIONAL RESEARCH UNIVERSITY OF ELECTRONIC TECHNOLOGY

Verilog HDL. Битовые логические операции

```
module simple_add_sub (
        input    [7:0]    operandA, operandB //два входных 8-ми битных операнда
        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. Булевые логические операции

```
module simple_add_sub (
        input    [7:0]    operandA, operandB //два входных 8-ми битных операнда
        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
```

MIET NATIONAL RESEARCH UNIVERSITY OF ELECTRONIC TECHNOLOGY

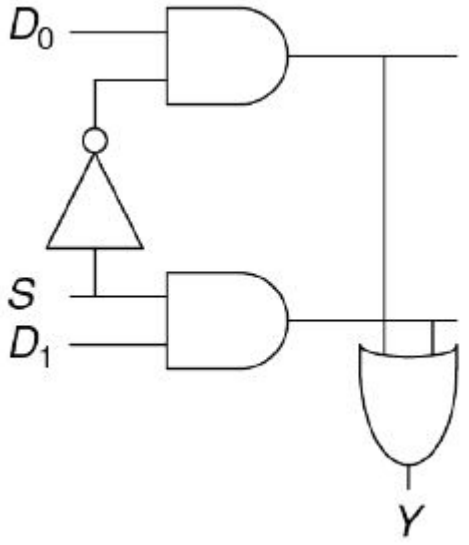
Verilog HDL. Операции свертки

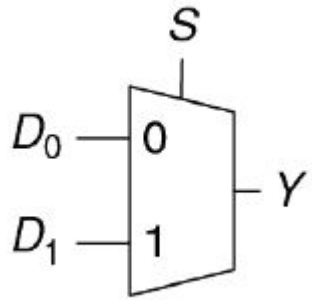
```
module simple_add_sub (
        input    [7:0]    operandA,
        output             out_reduction_and, out_reduction_or, out_redution_xor
);

//Операции выполняются между битами внутри одной шины
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
```

MIET NATIONAL RESEARCH UNIVERSITY OF ELECTRONIC TECHNOLOGY

Мультиплексор Verilog HDL

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

assign Y = S ? D1 : D0;

endmodule
```





Verilog HDL. Операции сравнения

```
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 используют при поведенческом (behavioral) описании схемы. Если регистру постоянно присваивается значение комбинаторной (логической) функции, то он ведет себя точно как провод (wire). Если же регистру присваивается значение в синхронной логике, например по фронту сигнала тактовой частоты, то ему, в конечном счете, будет соответствовать физический D-триггер или группа D-триггеров. D-триггер – это логический элемент способный запоминать один бит информации.

```
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
);
//Присвоение в блоке always возможно только для сигналов типа reg!!!!
always @(*) begin
        q = !a[7:4];
end

endmodule
```

Verilog HDL. Блок always

```
//Связанные между собой выражения превратятся в общую цепь комбинационной логики
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

```
//Связанные между собой выражения превратятся в общую цепь комбинационной логики
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

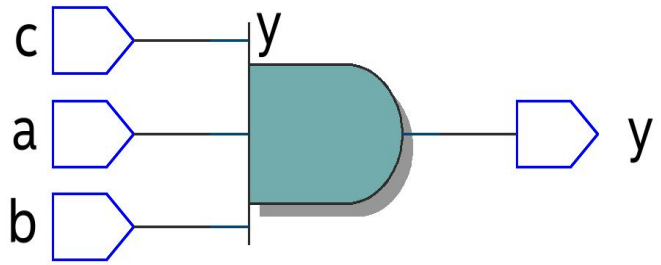
```
//можно описывать мультиплексоры с помощью if-else а не тернарного оператора
reg  [3:0] c;
always @(a or b or d) begin
      if (d)  begin
          c = a & b;
      end  else begin
          c = a + b;
      end
end
```

Verilog HDL. Case

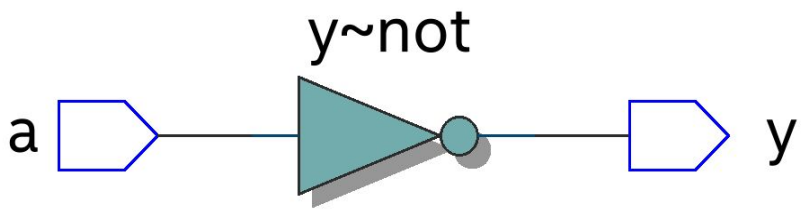
```
//многовходовые мультиплексоры и дешифраторы можно описывать через case
reg  [3:0] c;
wire  [1:0] option;
wire  [7:0] a, b, c, d;
reg    [7:0] e;
always @(a or b or c or d or option) begin
case (option)
  0: e = a;
  1: e = b;
  2: e = c;
  3: e = d;
endcase
end
```

Иерархия модулей Verilog HDL

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

assign y = a & b & c;

endmodule
```

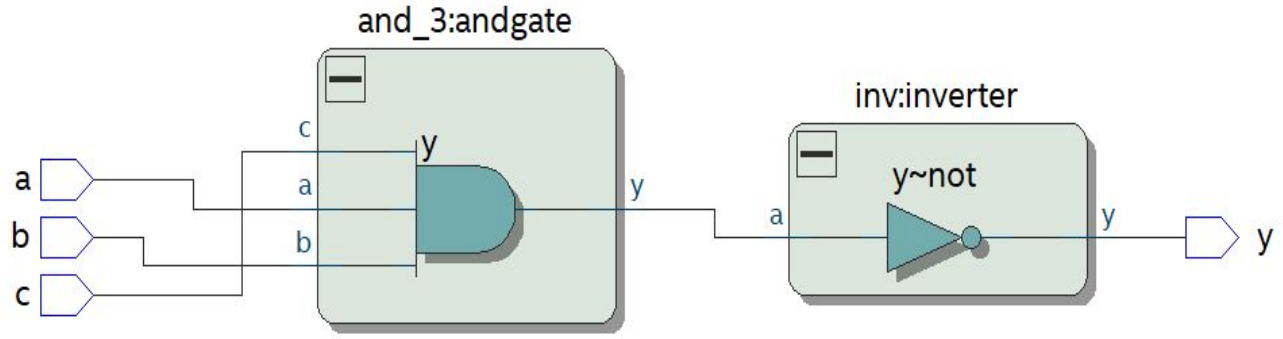



```
module inv (
        input     a
        output    y
);

assign y = ~a;

endmodule
```



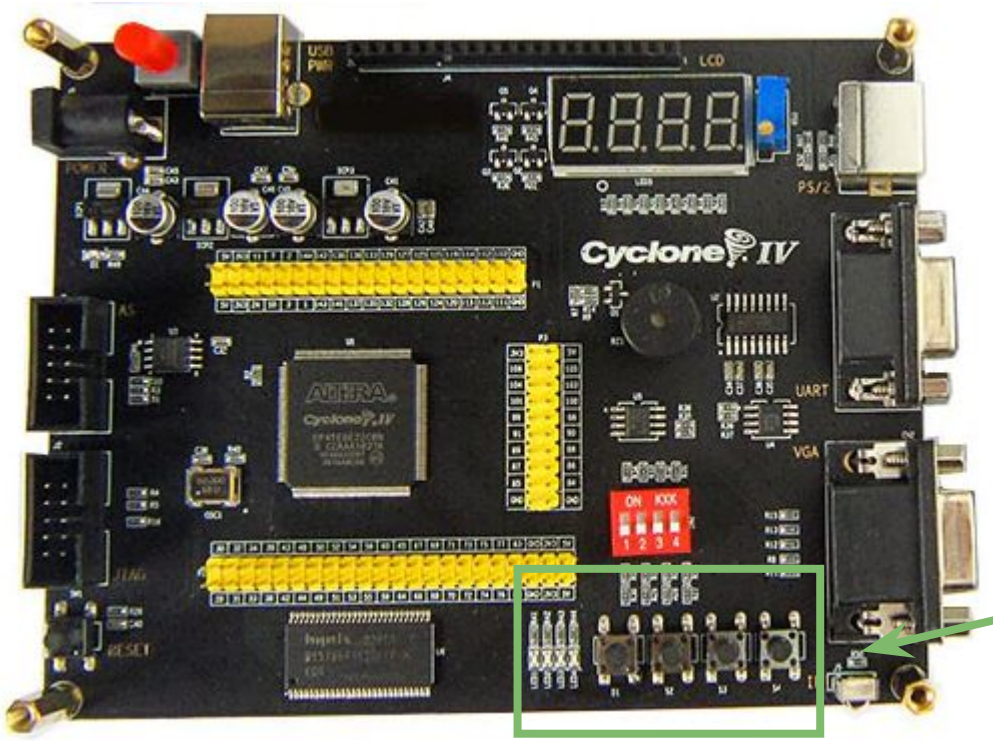
Иерархия модулей Verilog HDL

```
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
```



Имя подключаемого **модуля** (and_3, inv)
Название примитива. Например, нам может понадобиться 3 копии модуля and_3. Тогда мы сможем подключить 3 экземпляра модуля **and_3**, используя различные наименования для прототипов (andgate_1, andgate_2 …)
Символ **точка**, перед наименованием порта отсылает к реальному порту подключаемого модуля. **В скобках** обозначается , куда будут подключаться сигналы в top-модуле

Упражнение с логическими элементами



Altera Cyclone IV EP4CE6 FPGA

Вывод результатов логических операций над входными воздействиями с кнопок на светодиоды.

MIET NATIONAL RESEARCH UNIVERSITY OF ELECTRONIC TECHNOLOGY

Упражнение с логическими элементами

```
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
```

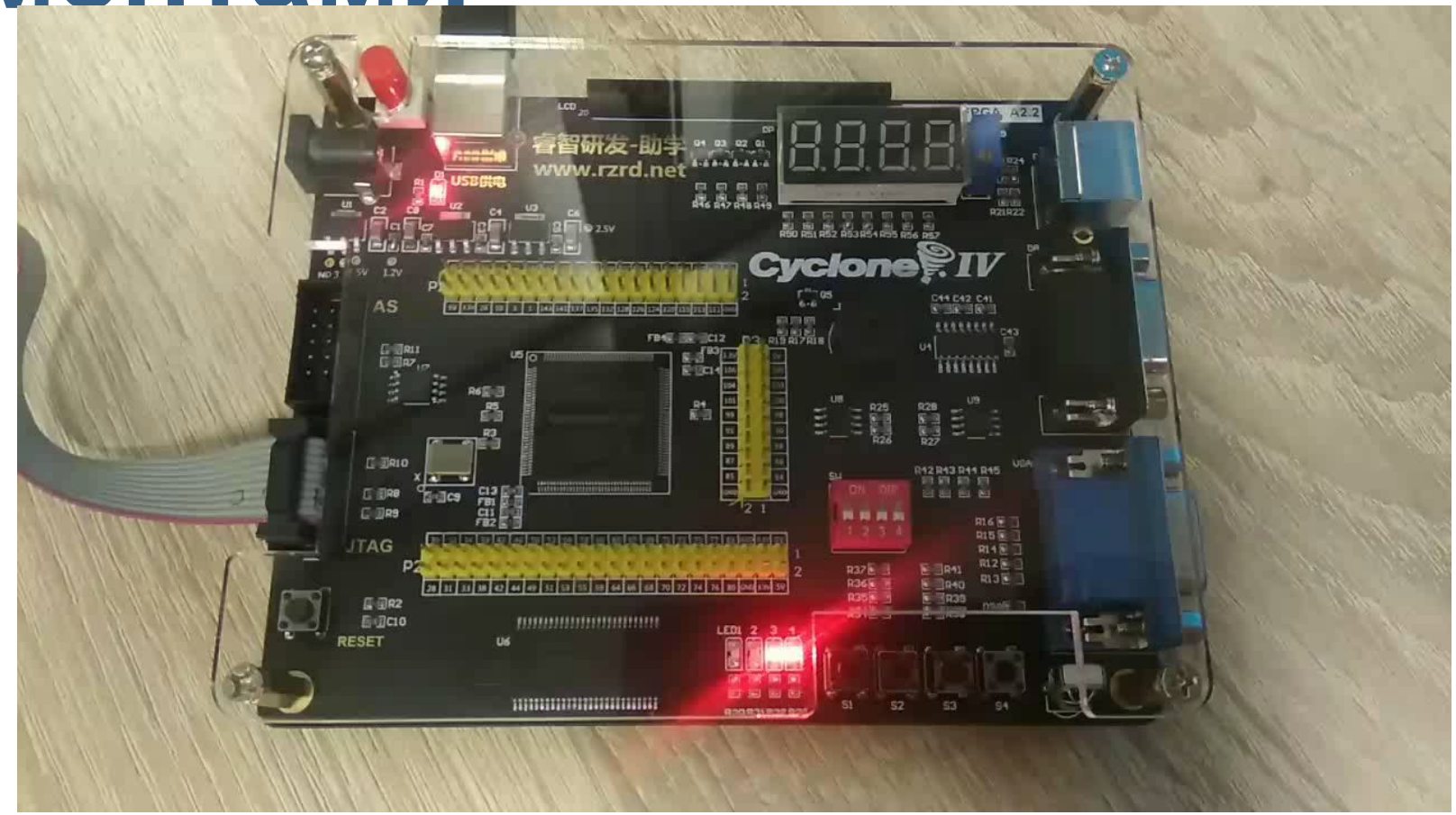
```
module top
(
    …
    input  [3:0] key_sw,
    output [3:0] led,
    …
);
    …
    // Exercise 2: Change the code below.
    // Assign to led [3] the result of XOR operation
    // without using "^" operation.
    // Use only operations "&", "|", "~" and parenthesis

    assign led [3] = 1'b0;

endmodule
```

MIET NATIONAL RESEARCH UNIVERSITY OF ELECTRONIC TECHNOLOGY

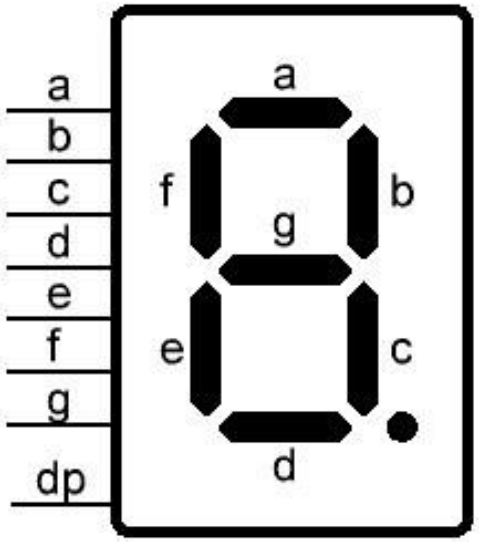
Упражнение с логическими элементами

1. Присвойте led[2] результат операции И (AND).
2. Присвойте led[3] результат операции исключающего ИЛИ (XOR) без использования ее оператора.

Упражнение с логическими элементами

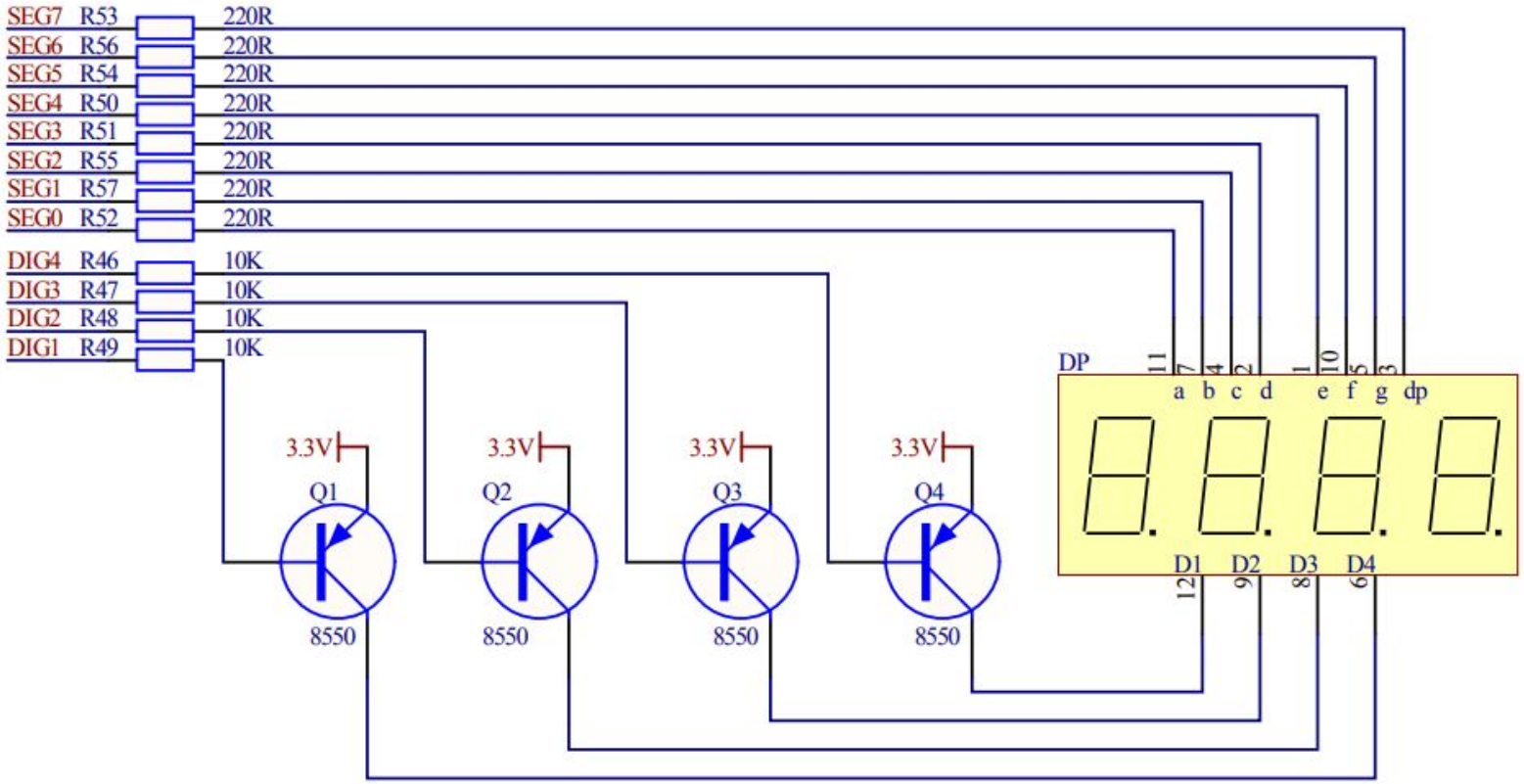


Семисегметный индикатор

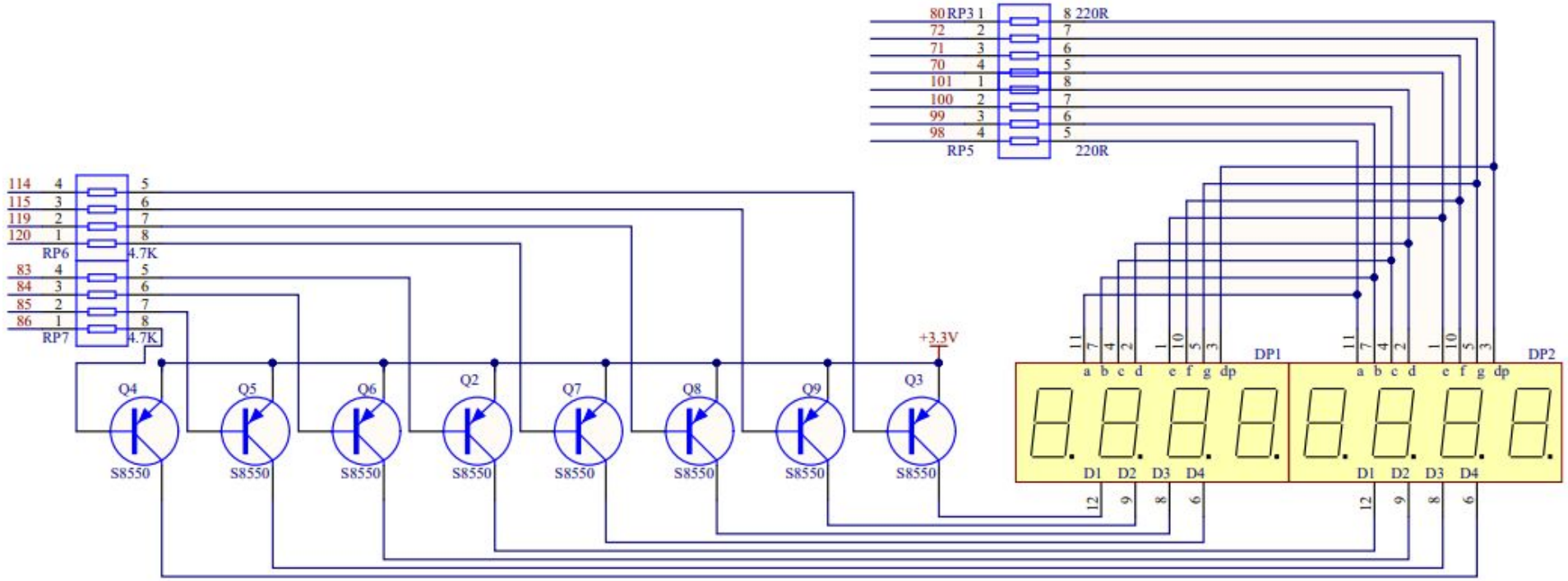


	4-bit шина				7-сегментный индикатор						
Dec	3	2	1	0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1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

Семисегметный индикатор



Семисегметный индикатор



Упражнение с выводом буквы на семисегментный индикатор

```
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
```

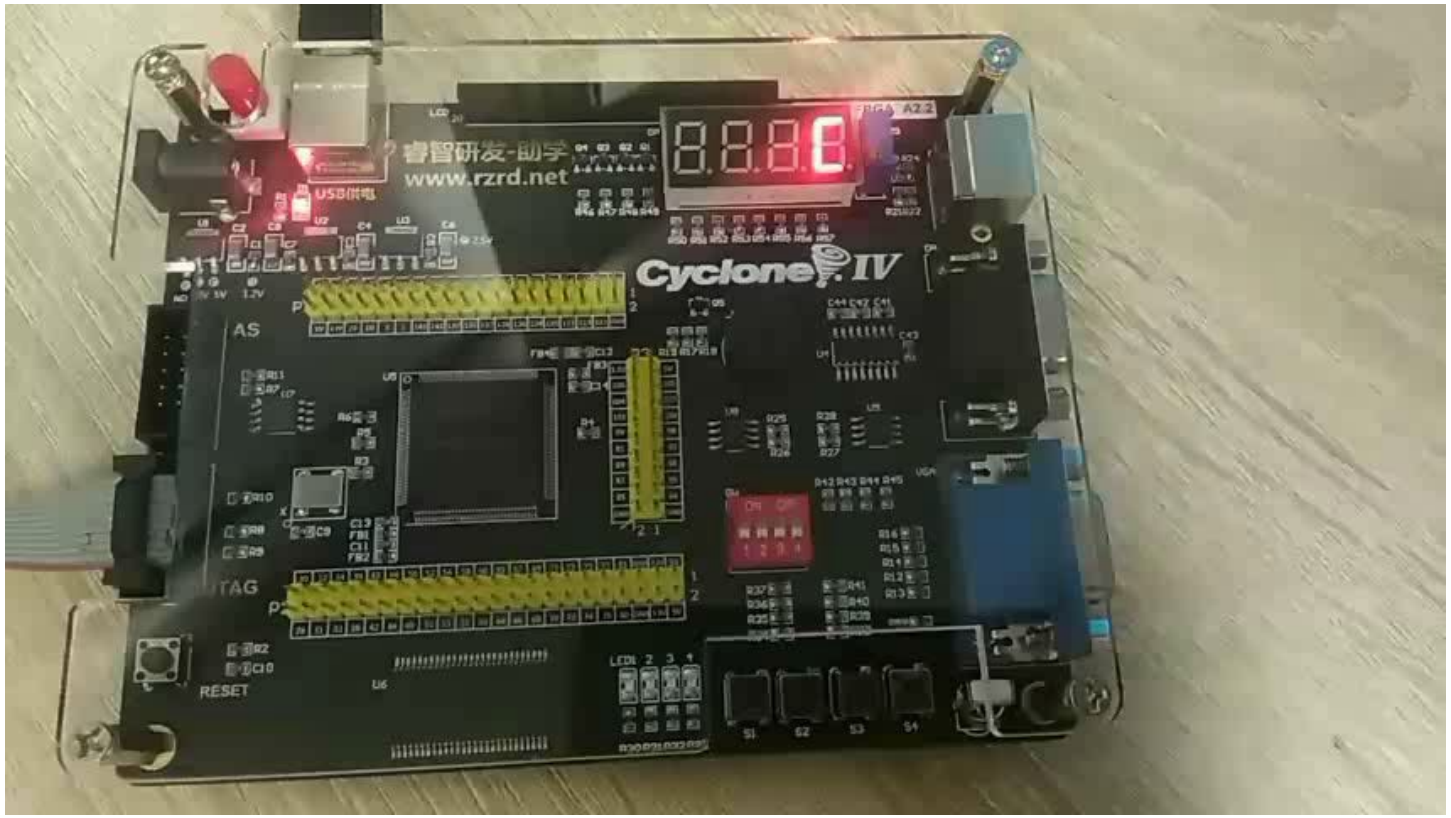
```
module top
(
    …
);
    …
    // Exercise 2: Display letters of a 4-character word
    // using this code to display letter of ChIP as an exa
mple
    reg [7:0] letter;
    always @*
      case (key_sw)
          4'b0111: letter = C;
          …
      endcase

    assign abcdefgh = letter;

endmodule
```

MIET NATIONAL RESEARCH UNIVERSITY OF ELECTRONIC TECHNOLOGY

Упражнение с выводом буквы на семисегментный индикатор

1. Выведите первые буквы своего имени и фамилии на семисегментный индикатор.
2. Выведите слово CHIP на семисегментный индикатор.

Упражнение с выводом буквы на семисегментный индикатор



MIET NATIONAL RESEARCH UNIVERSITY OF ELECTRONIC TECHNOLOGY

Спасибо за внимание.