Язык описания аппаратуры Verilog

Базовые типы источников сигналов

1. Проводник – wire

```
wire a;
wire b;
assign a = b;
wire c = b;
wire [3:0] d;
wire [12:0] s = 12; /* 32-х битное десятичное число, которое будет “обрезано” до 13 бит */
wire [12:0] z = 13'd12; //13-ти битное десятичное число
wire [3:0] t = 4'b0101; //4-х битное двоичное число
wire [7:0] q = 8'hA5; // 8-ми битное шестнадцатеричное число A5
```

2. Регистр – reg

```
reg [3:0] r = 0;
reg [7:0] q [0:15]; //память из 16 слов, каждое по 8 бит

always @(a or b or posedge clk) begin
        r <= r + 1;
end
```

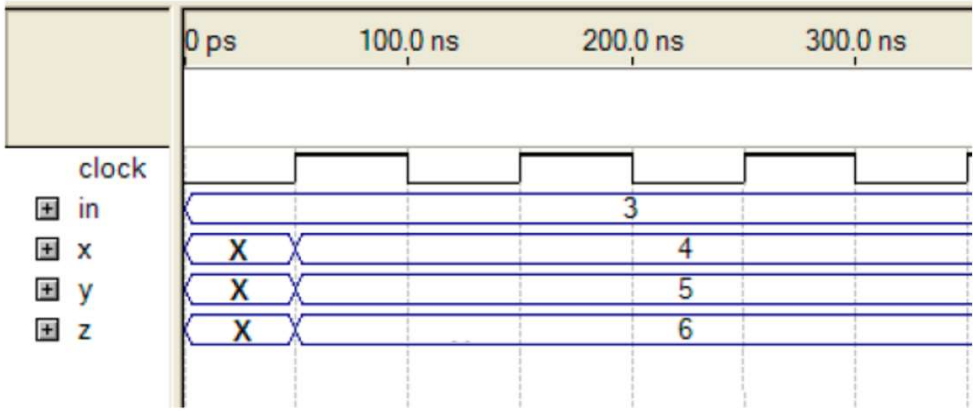
3. Глобальный параметр: `define m 10;

```
wire [`m:0] a;
```

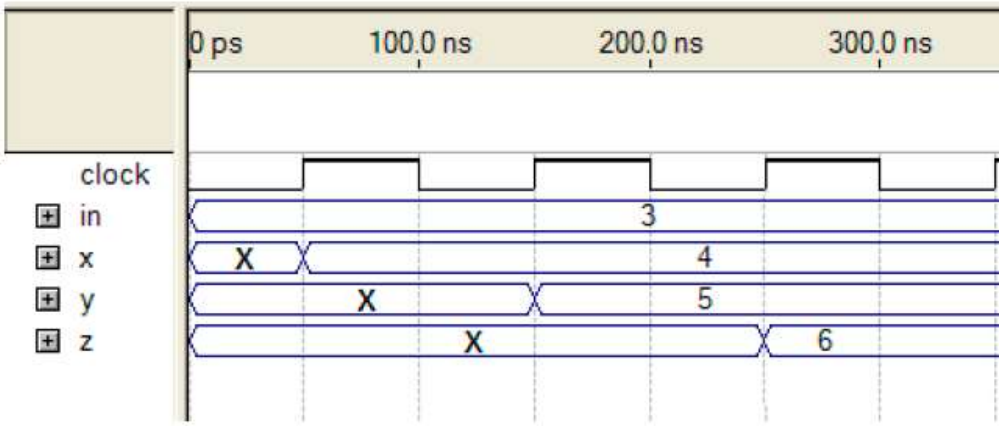
Язык описания аппаратуры Verilog

Блокирующее и синхронное присваивание, «always» блоки

```
wire [3:0]in = 3;
reg[3:0] x, y, z;
always @(posedge clock) begin
        x = in + 1;
        y = x + 1;
        z = y + 1;
end
```




```
wire [3:0]in = 3;
reg[3:0] x, y, z;
always @(posedge clock) begin
        x <= in + 1;
        y <= x + 1;
        z <= y + 1;
end
```

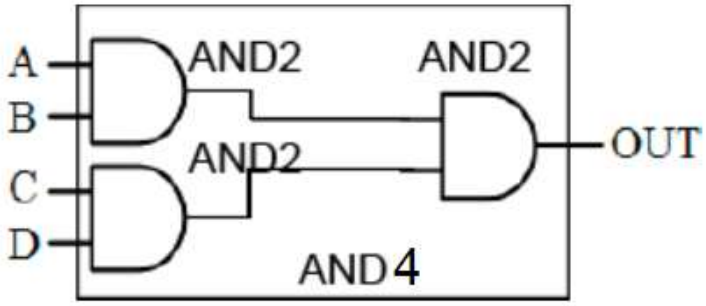


Язык описания аппаратуры Verilog

Иерархия проекта

```
module AND2(input IN1, input IN2, output OUT);
        assign OUT = IN1 & IN2;
endmodule

module AND4(output OUT, input A, input B, input C, input D)
        wire out_1, out_2;
        AND2 DD1( .OUT (out_1), .IN1 (A), .IN2 (B) );
        AND2 DD2( .OUT (out_2), .IN1 (C), .IN2 (D) );
        AND2 DD3( .OUT (OUT), .IN1 (out_1), .IN2 (out_2)
);
endmodule
```



Язык описания аппаратуры Verilog

Арифметические и логические функции

```
wire [7:0] operandA, operandB; //два wire 8-ми битных
wire [8:0] out_sum = operandA + operandB;
wire out_shl = operandA << operandB; /*логический сдвиг влево operandA на количество бит, указанное в operandB*/

/*Логические операции. Каждый бит результата вычисляется отдельно соответственно битам операндов*/
wire [7:0] out_bit_and, out_bit_or, out_bit_xor, out_bit_not;
assign out_bit_and = operandA & operandB; //И
assign out_bit_or = operandA | operandB; //ИЛИ
assign out_bit_xor = operandA ^ operandB; //исключающее ИЛИ
assign out_bit_not = ~operandA; //НЕ
```

Язык описания аппаратуры Verilog

Булевые логические операции

Здесь значение всей шины рассматривается как ИСТИНА если хотя бы один бит в шине не ноль или ЛОЖЬ, если все биты шины – ноль.
Результат получается всегда однобитный (независимо от разрядности операндов) и его значение "1" (ИСТИНА) или "0" (ЛОЖЬ).

```
wire [7:0] operandA, operandB; //два wire 8-ми битных
wire out_bool_and, out_bool_or, out_bool_not;

assign out_bool_and = operandA && operandB; //И
assign out_bool_or = operandA || operandB; //ИЛИ
assign out_bool_not = !operandA; //НЕ
```

Язык описания аппаратуры Verilog

Операции редукции

Позволяют выполнять операции между битами внутри одной шины.

```
wire [7:0] operandA; //два wire 8-ми битных
wire out_reduction_or, out_reduction_and, out_reduction_xor, out_reduction_no1, out_reduction_0;

assign out_reduction_or = |operandA; //есть ли в шине хотя бы одна единица
assign out_reduction_and = &operandA; //все ли биты в шине равны единице
assign out_reduction_xor = ^operandA; //xor всех бит шины
assign out_reduction_no1 = ~|operandA; //обозначает, что в шине нет единиц
assign out_reduction_0 = ~&operandA; //обозначает, что некоторые биты в шине равны нулю
```

Язык описания аппаратуры Verilog

Оператор условного выбора и сравнения

В Verilog есть оператор **‘? :’**. Он фактически реализует мультиплексор.
В данном примере на выходе мультиплексора окажется значение operandA если сигнал sel_in единица. И наоборот, если входной сигнал sel_in равен нулю, то на выходе мультиплексора будет значение operandB.

```
assign out_mux = sel_in ? operandA : operandB;
```

В Verilog можно сравнивать "числа" (а точнее значения регистров или шин). Ниже приведен пример (все сравнения происходят с беззнаковыми числами):

```
wire [7:0] operandA=1, operandB=2;
wire out_eq = operandA == operandB; //равно
wire out_ne = operandA != operandB; //не равно
wire out_ge = operandA >= operandB; //больше или равно
wire out_le = operandA <= operandB; //меньше или равно
wire out_gt = operandA > operandB; //больше
wire out_lt = operandA < operandB; //меньше
```

Логическое моделирование устройств

Для тестирования, отладки своего разрабатываемого устройства используется симулятор, с помощью которого можно посмотреть временные диаграммы написанного модуля, т.е. посмотреть как меняется состояние выходов модуля при изменении состояния входов. Для моделирования устройства нужно:

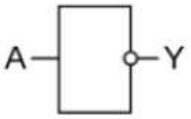
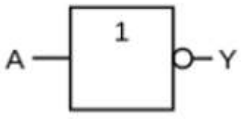
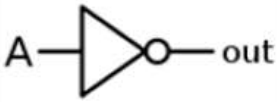
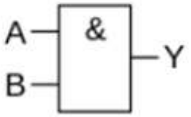
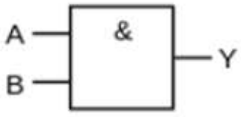
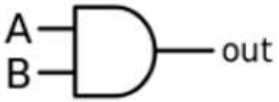
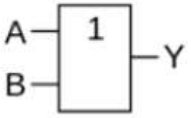
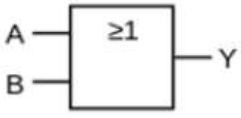
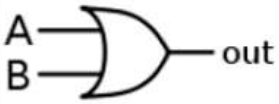
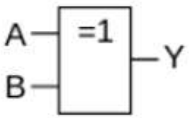
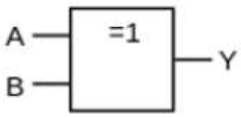
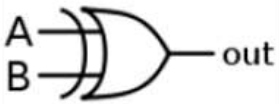
1. Создать задание на моделирование

```
wire [2:0] Q; reg clk, clr;
always begin
        clk=0; #10; clk=1; #10; //Генератор периодического сигнала синхронизации clk
end
initial begin
        clr = 0; //Исходное состояние входа
        #92; clr = 1; //Через 92 нс 1
        #5; clr = 0; // Через 5 нс 0
end
My_module module1 (.Q(Q), .clk(clk), .clr(clr));
```

2. Настроить параметры моделирования (н-р, время наблюдения).
3. Запустить симулятор, получить и проанализировать полученные временные диаграммы.

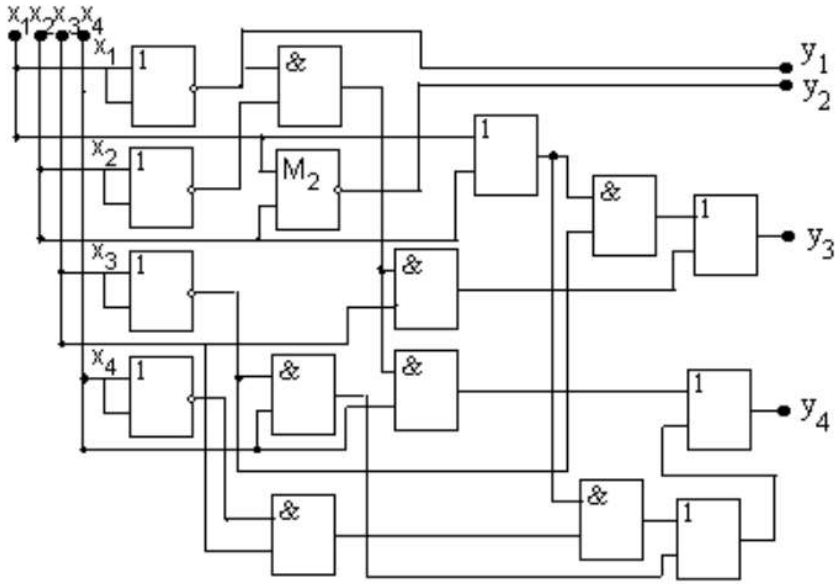
Пример временной диаграммы счетчика

Таблица истинности логических элементов

Логический вентиль	Условные графические обозначения			Функция	Таблица истинности
	ГОСТ 2.743-91	IEC 60617-12 : 1997	US ANSI 91-1984		
НЕ (англ. *NOT gate*)				Отрицание $Y = \overline{A}$ $Y = \neg A$ $Y = \tilde{A}$	A Y 0 1 1 0
И (англ. *AND gate*)				Конъюнкция $Y = A \wedge B$ $Y = A \cdot B$ $Y = A \,\&\, B$ $Y = AB$	A B Y 0 0 0 0 1 0 1 0 0 1 1 1
ИЛИ (англ. *OR gate*)				Дизъюнкция $Y = A \vee B$ $Y = A + B$	A B Y 0 0 0 0 1 1 1 0 1 1 1 1
Исключающее ИЛИ (англ. *XOR gate*) сложение по модулю 2				Строгая дизъюнкция $Y = A \veebar B$ $Y = A \oplus B$	A B Y 0 0 0 0 1 1 1 0 1 1 1 0

Комбинационная логика

У комбинационных устройств состояние выхода однозначно определяется набором входных сигналов (т.е. значения выходных сигналов никак не зависят от своего предыдущего состояния)

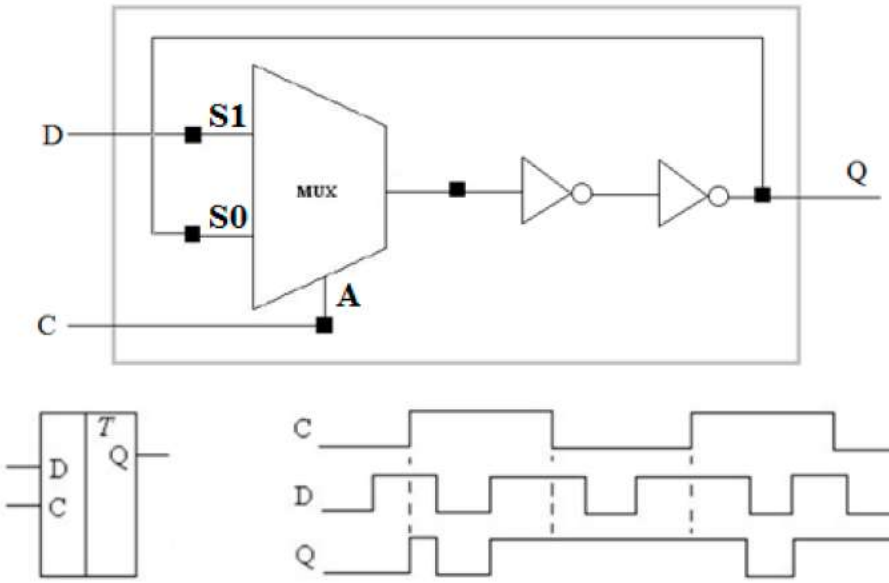


Пример реализации комбинационного устройства на логических элементах

Последовательностная логика

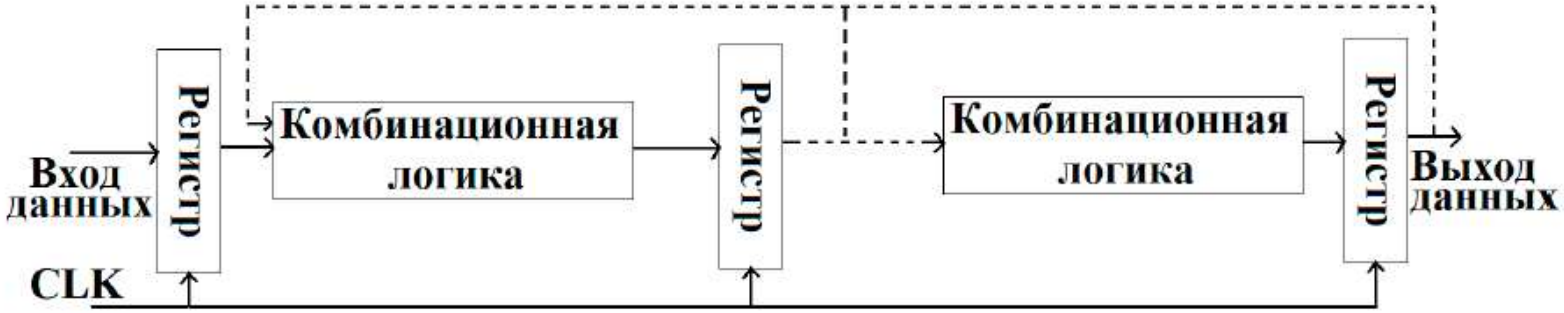
Логика памяти цифровых устройств. Отличается от комбинационной логики тем, что состояния выходов зависят не только от состояния входов, но и от предыстории функционирования устройства. Изменение состояния Выходов возможно только по сигналу синхронизации.

- Триггеры
- Регистры
- Счетчики



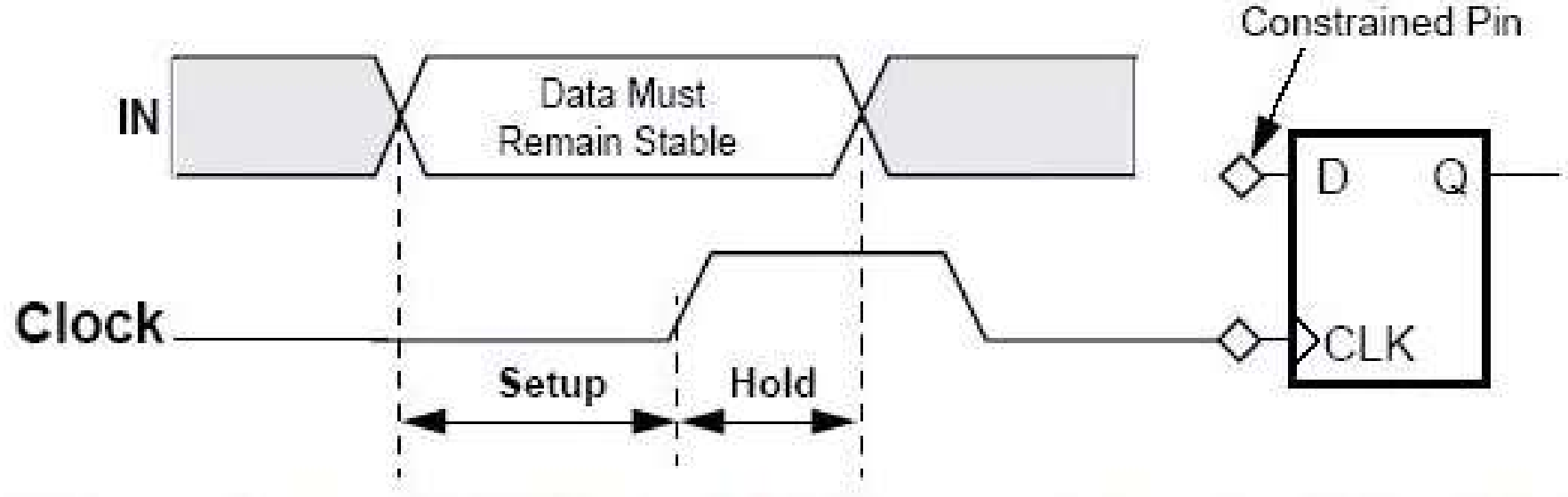
Чередование комбинационной и последовательностной логики

Практически любое современное цифровое устройство представляет собой набор последовательно (или циклично) соединенных схем комбинационной и последовательностной логики. Новые выходные данные формируются по каждому импульсу сигнала синхронизации CLK.



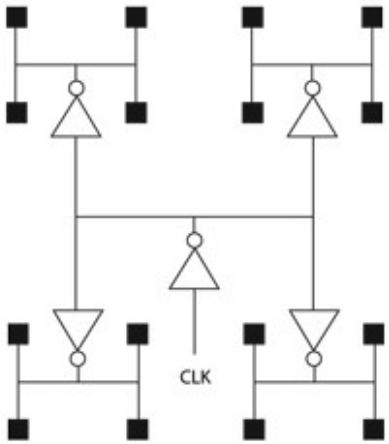
Статический временной анализ

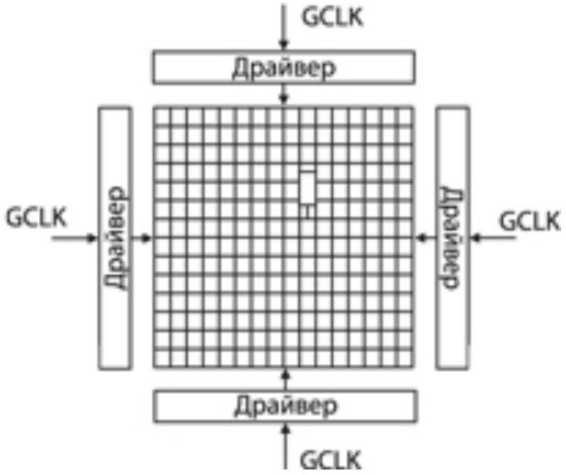
Время установки T_{Setup} – минимальное время от момента завершения переходных процессов на входе данных до момента прихода фронта клока на тактовый вход регистра.
Время удержания T_{Hold} – минимальное время от момента прихода фронта клока на тактовый вход до начала новых переходных процессов на входе данных регистра.

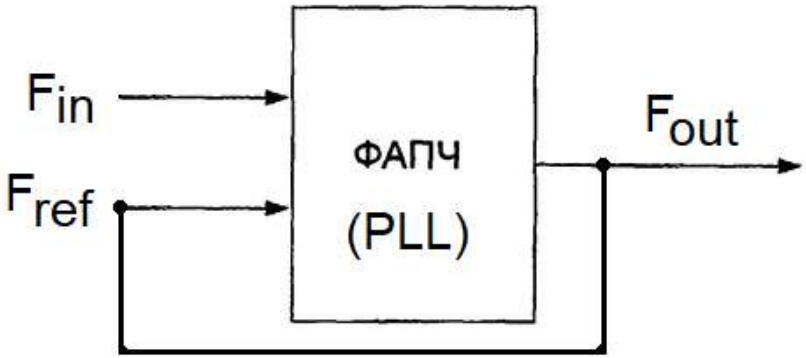


Синхронизация в кристалле

Расфазировка тактовых импульсов в цифровых схемах — серьезная проблема, способная ограничить быстродействие цифровой системы.
Задержка распространения от центрального источника тактовых импульсов несущественна, значение имеет лишь разность фаз между синхронизируемыми точками.
Некоторые варианты решения:

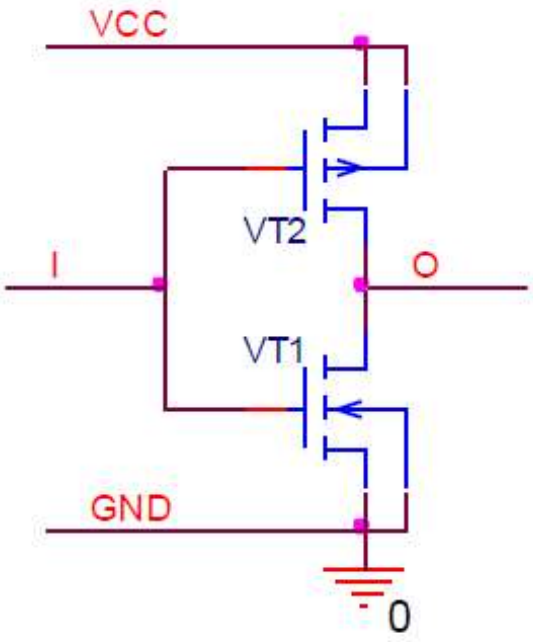


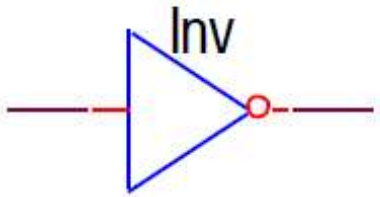


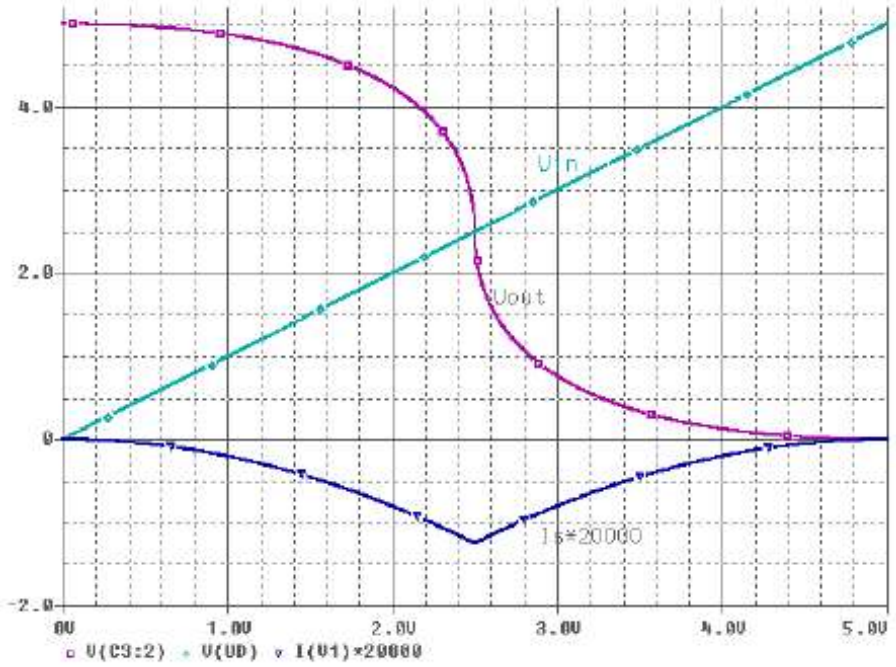


Физическая реализация логических элементов

Инвертор (логический элемент **НЕ**)

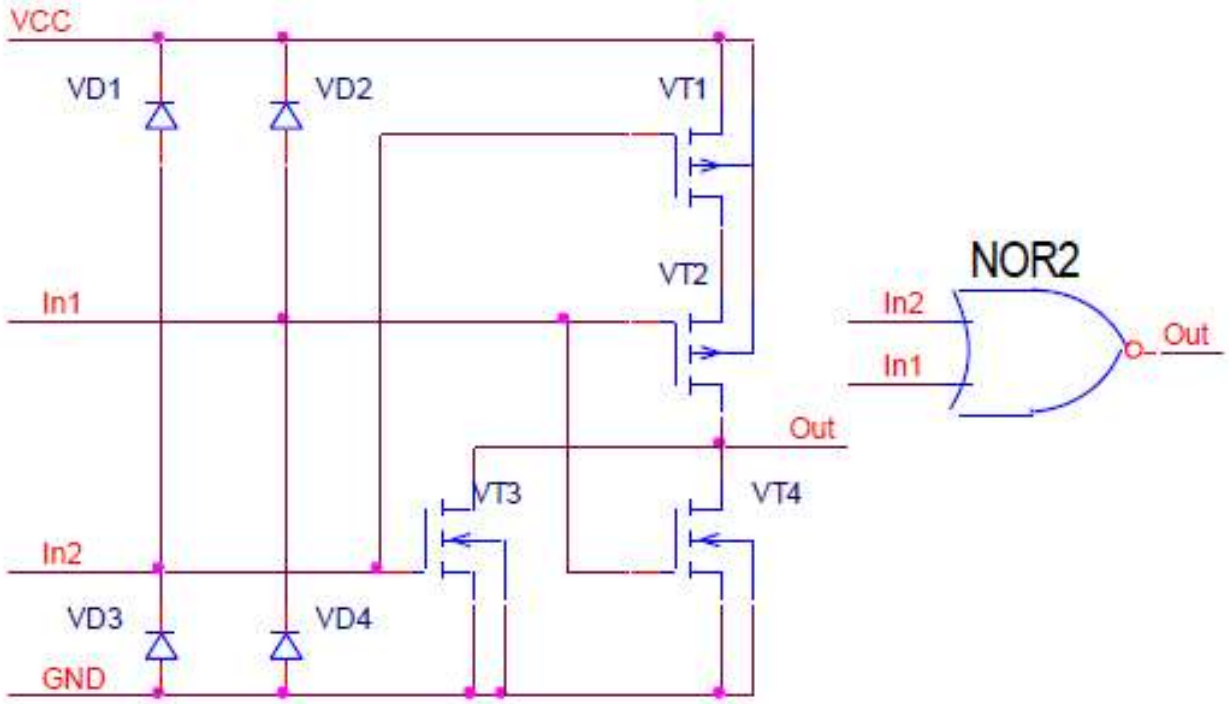






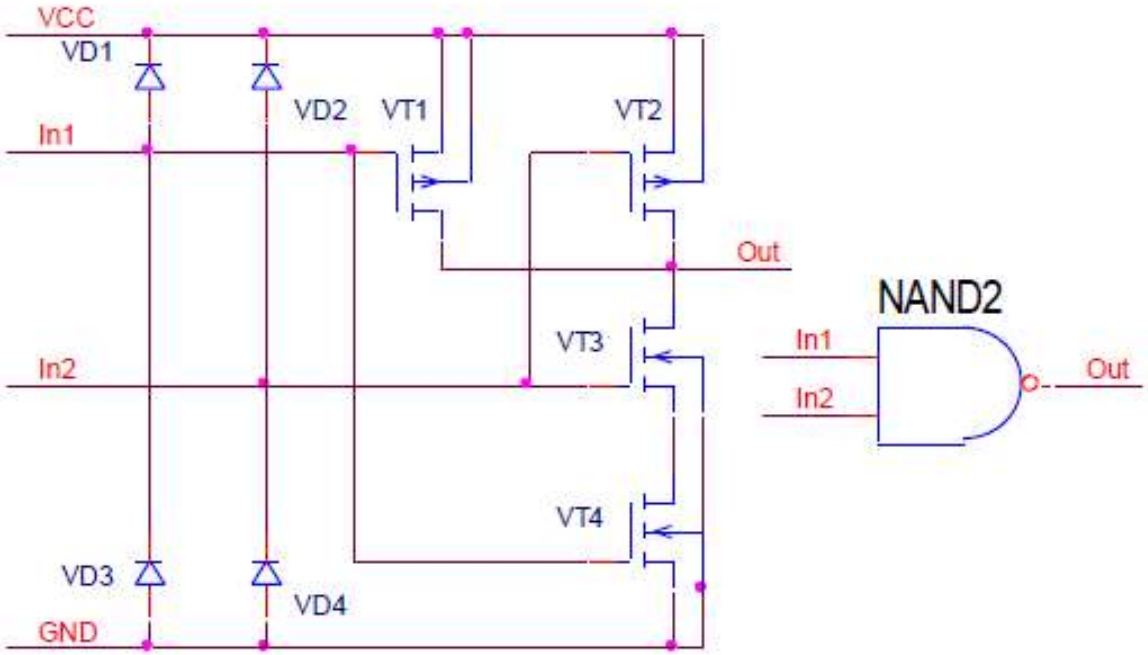
Физическая реализация логических элементов

Двухвходовый логический элемент **ИЛИ-НЕ**



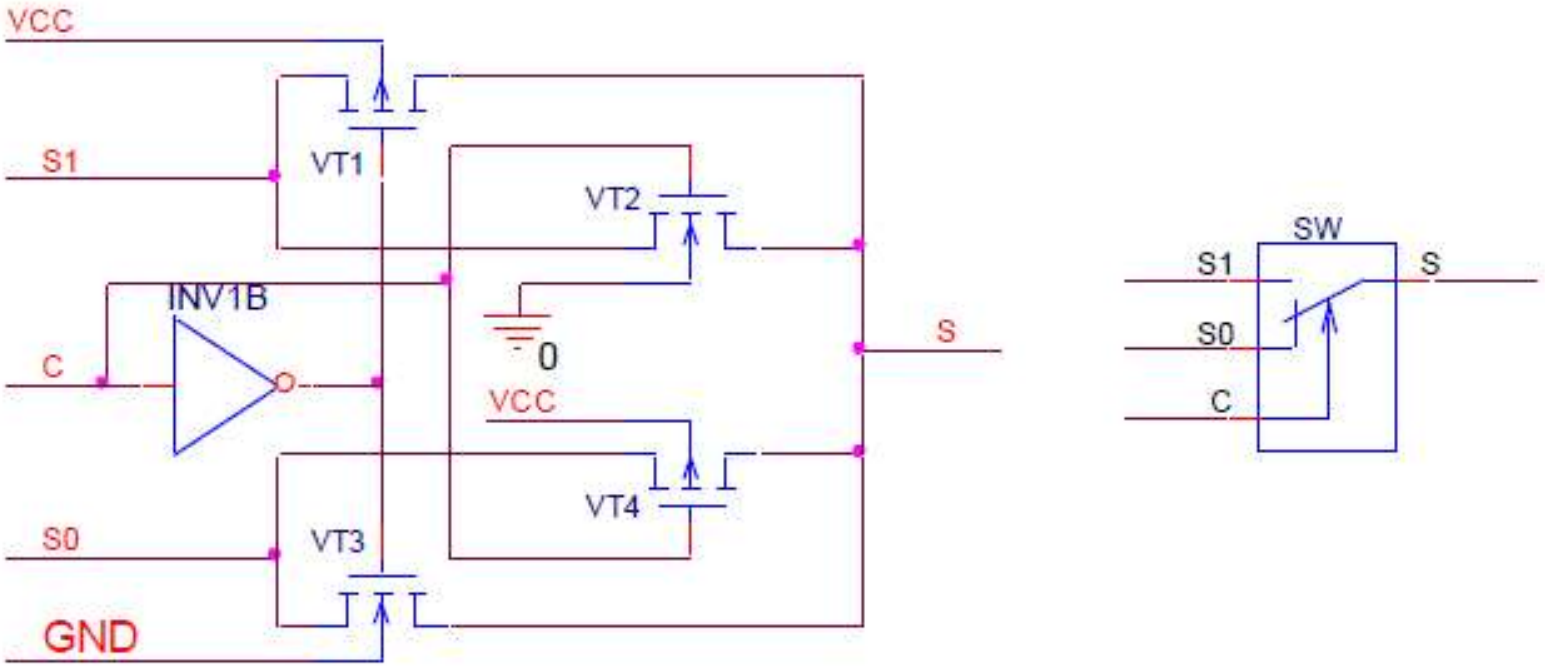
Физическая реализация логических элементов

Двухвходовый логический элемент **И-НЕ**



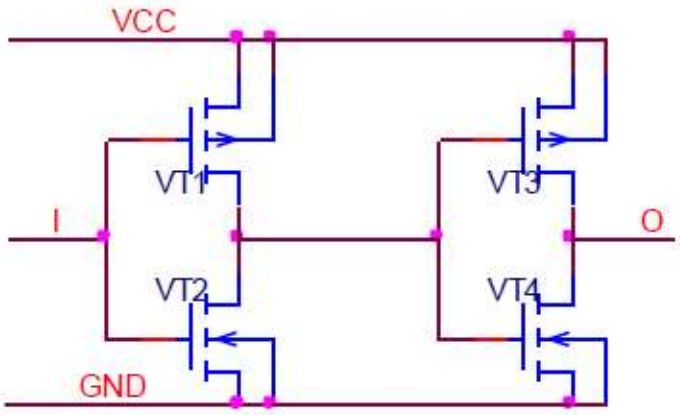
Физическая реализация логических элементов

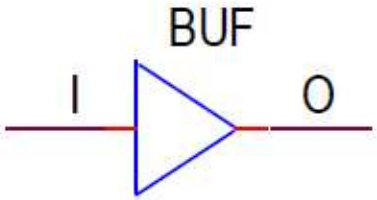
Аналоговый переключатель

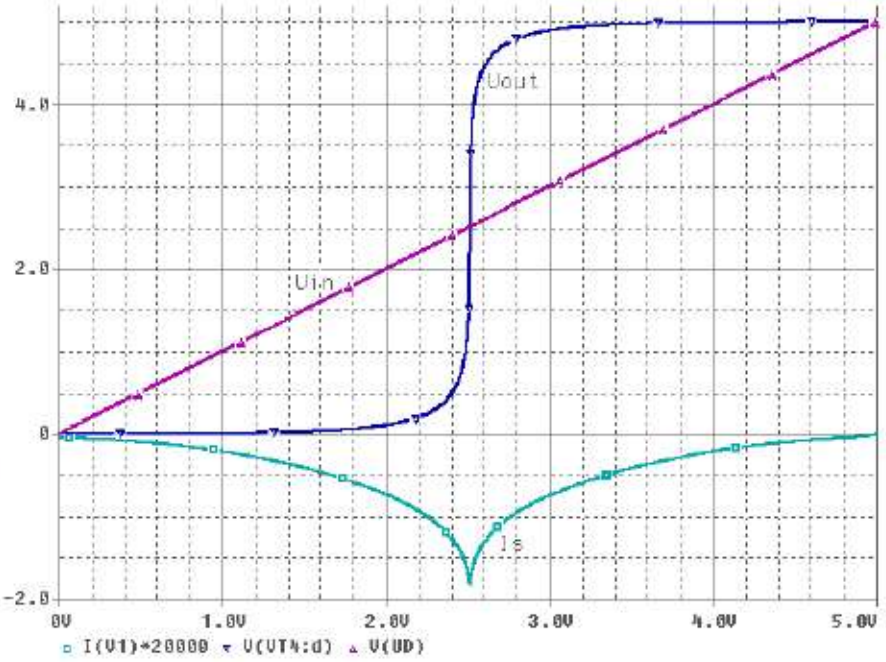


Физическая реализация логических элементов

Буфер - два последовательно соединенных инвертора

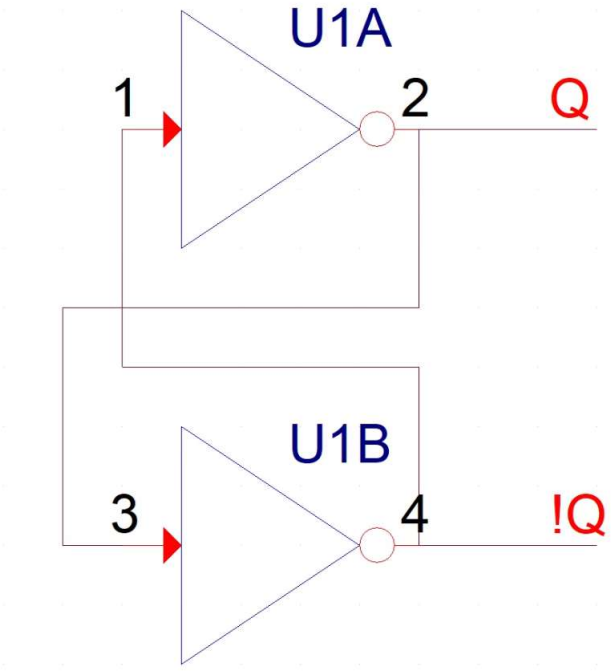






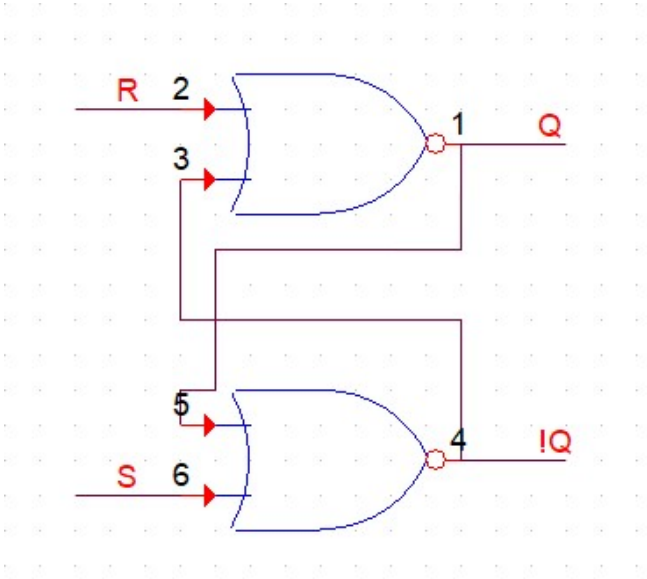
Реализация логических элементов

Триггер



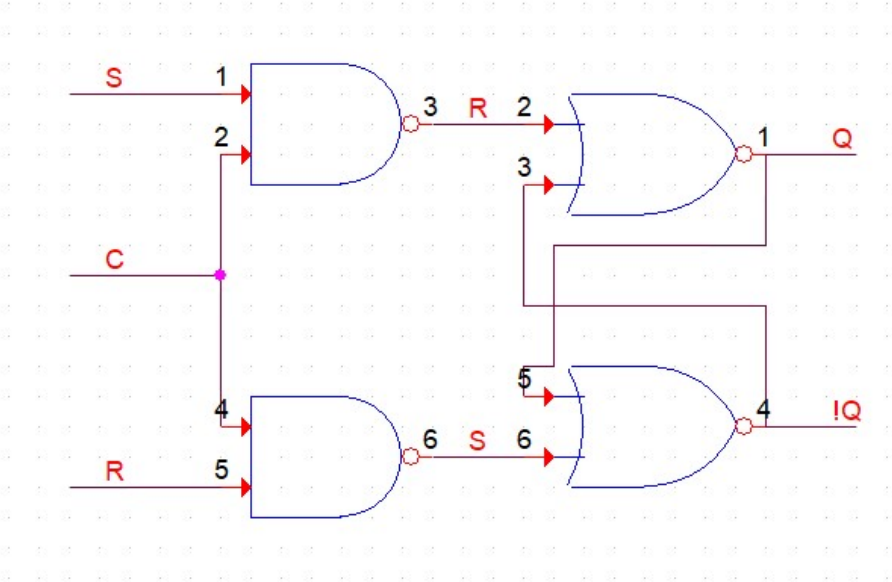
Реализация логических элементов

RS-триггер



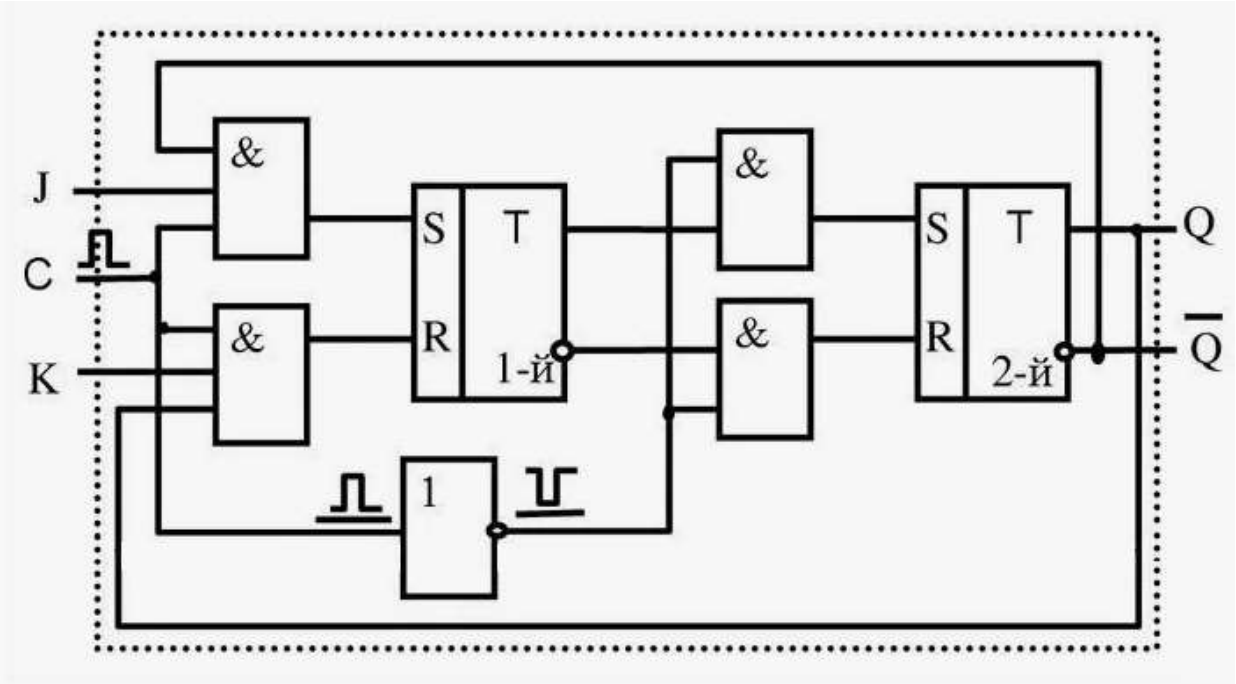
Реализация логических элементов

RS-триггер с синхронизацией



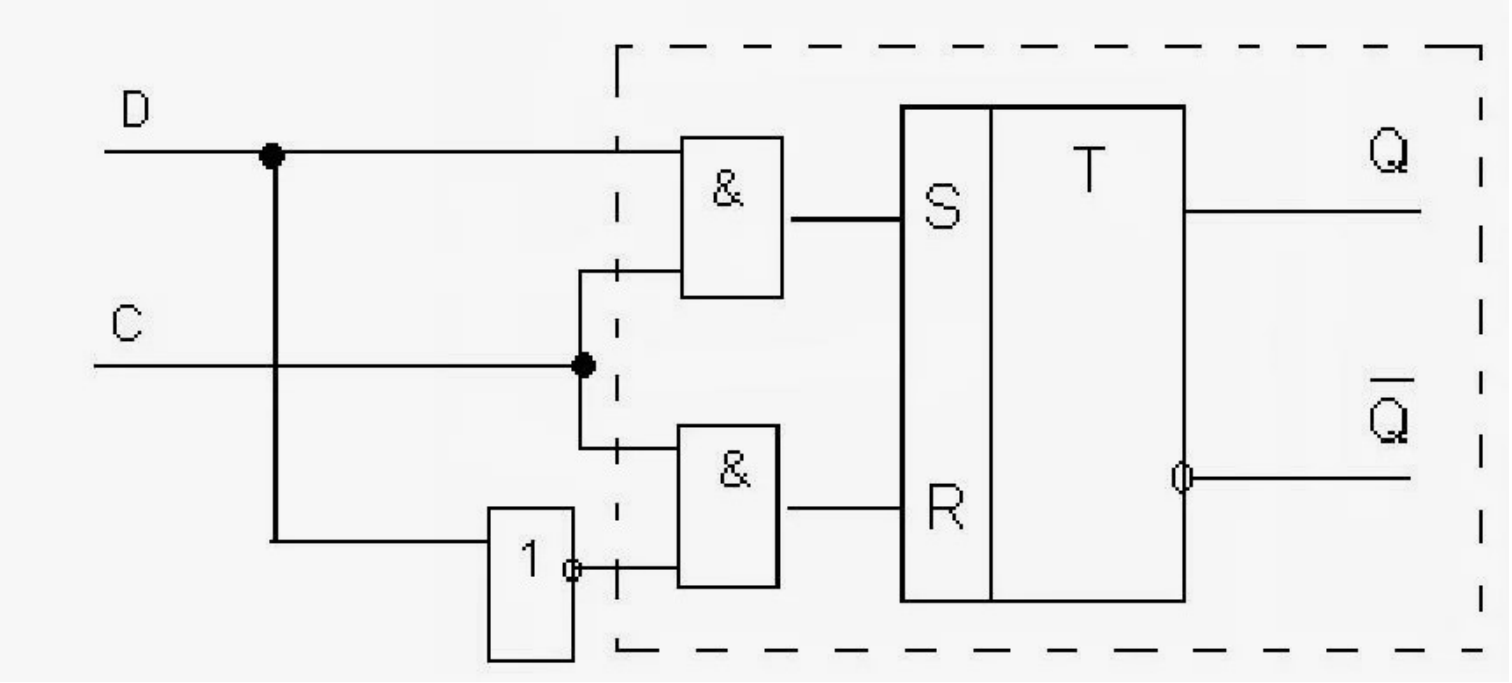
Реализация логических элементов

JK-триггер



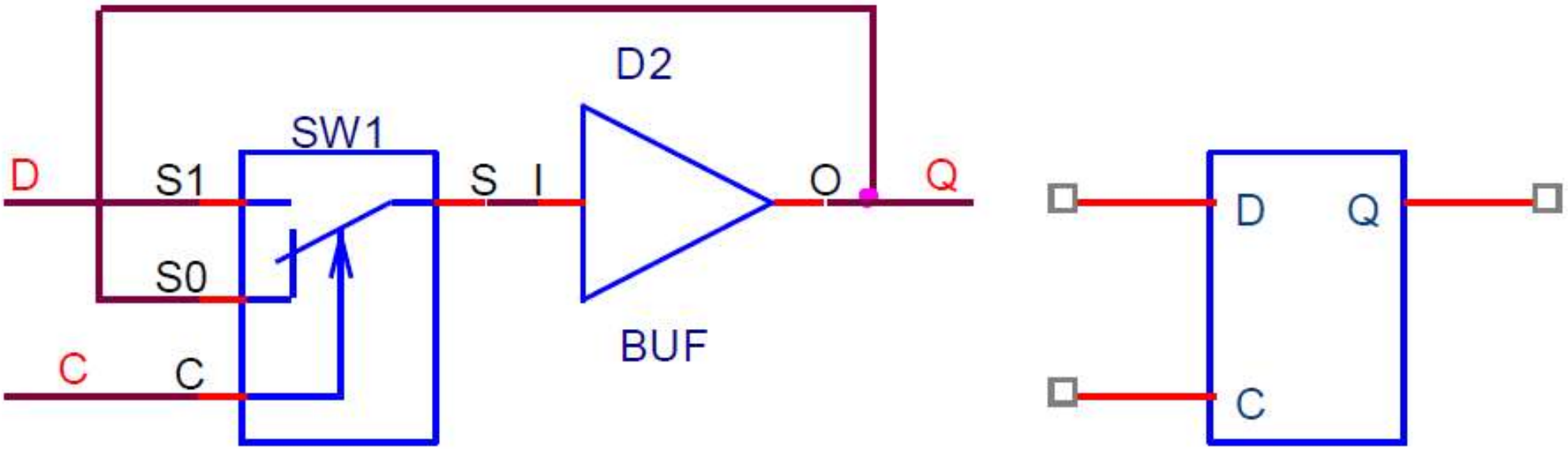
Реализация логических элементов

D-триггер



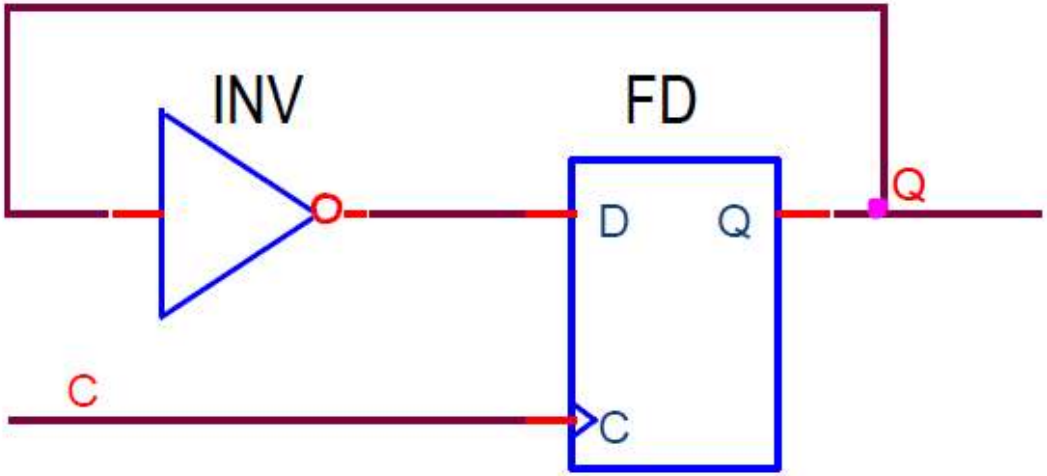
Реализация логических элементов

D-триггер



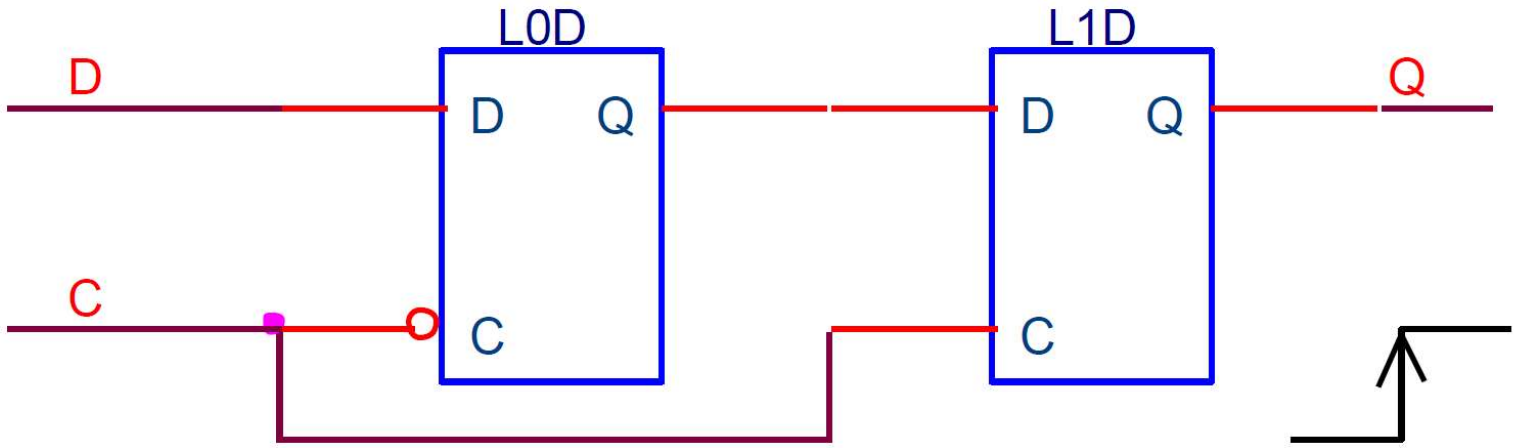
Реализация логических элементов

Т-триггер



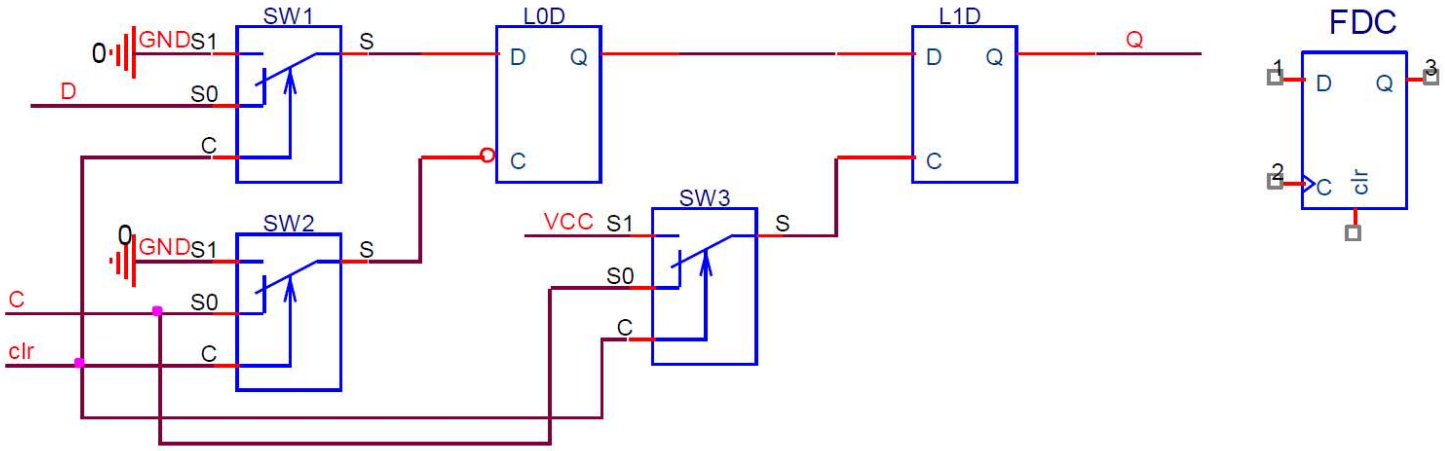
Реализация логических элементов

D-триггер с динамическим входом синхронизации



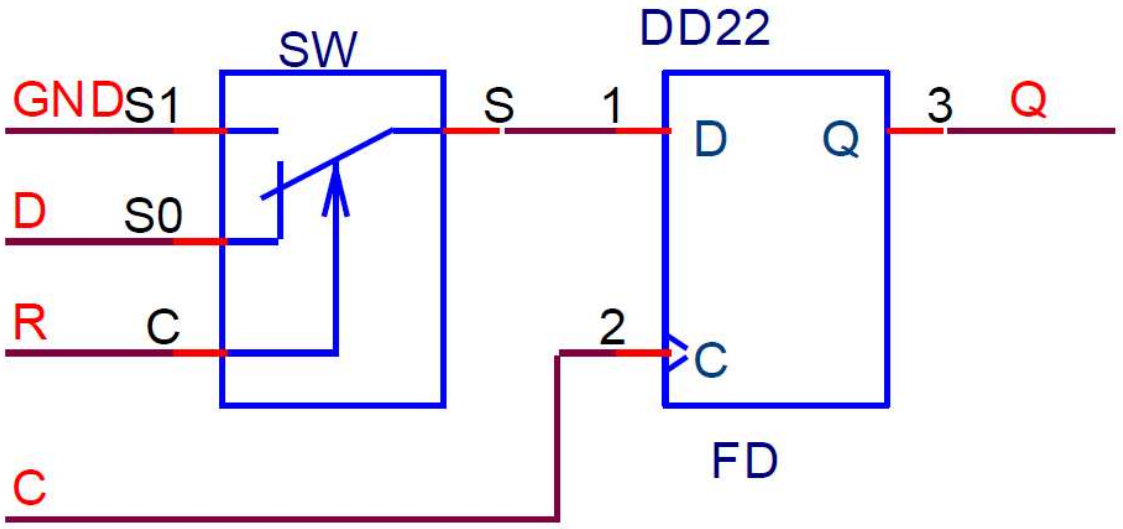
Реализация логических элементов

FDC и FDP триггеры
асинхронный сброс



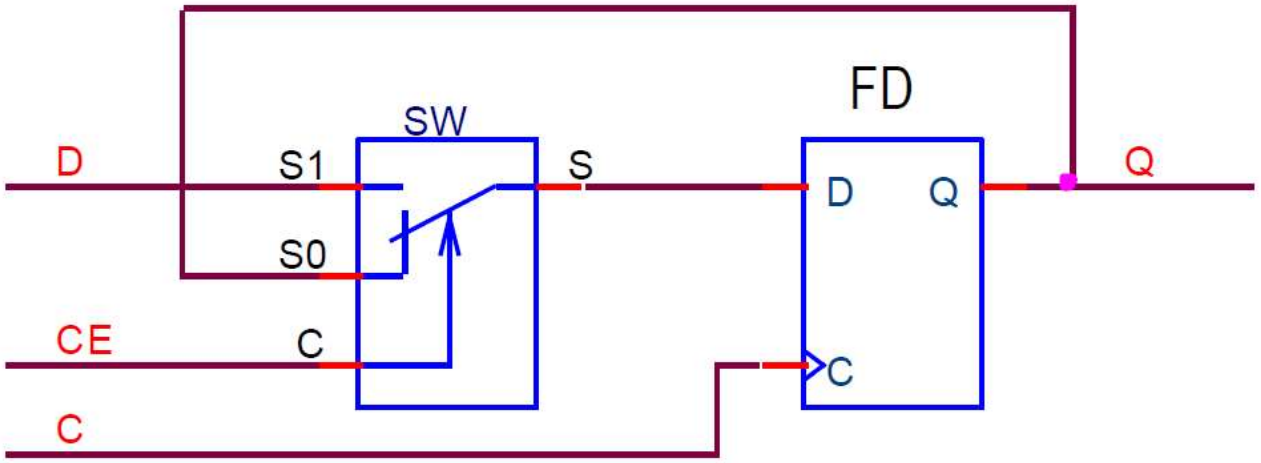
Реализация логических элементов

FDR и FDS триггеры
синхронный сброс



Реализация логических элементов

FDE триггеры
разрешающий вход



Реализация логических элементов

Один из вариантов именования триггеров

Реализация логических элементов

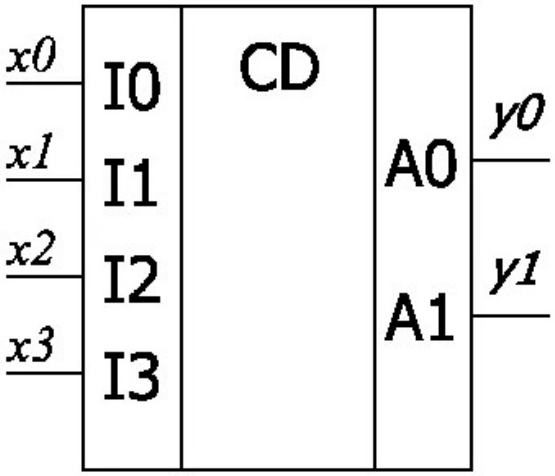
Комбинирование символов в именовании триггеров

FDRSE — D-триггер с синхронной установкой в 0 и 1 и с входом разрешения синхросигнала.
FJKCE — JK-триггер с входом разрешения синхросигнала и асинхронной установкой в 0.
FJKSRE — JK-триггер с синхронной установкой в 1 и 0 и с входом разрешения синхросигнала.
FTCLEX — T-триггер с функцией синхронной загрузки, с входом разрешения синхросигнала и с асинхронной установкой в 0.
FTSRLE — T-триггер с функцией синхронной загрузки, с входом разрешения синхросигнала и с синхронной установкой в 1 и 0.

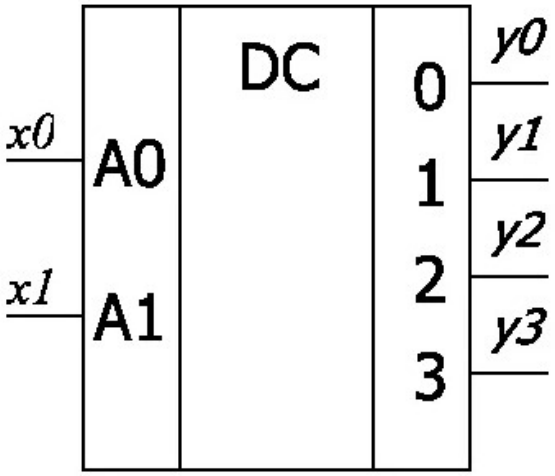
Реализация логических элементов

Шифраторы и дешифраторы

шифратор

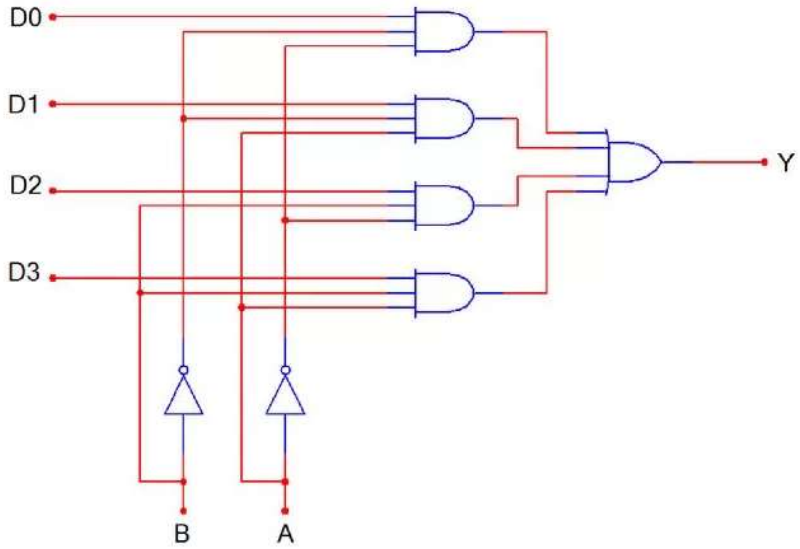


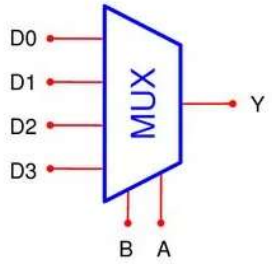
дешифратор



Реализация логических элементов

Мультиплексор

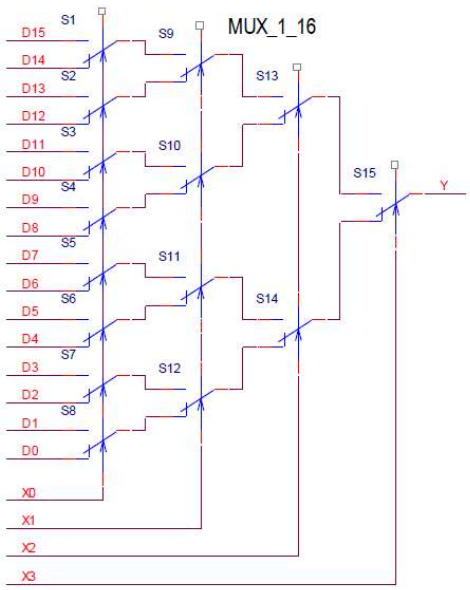




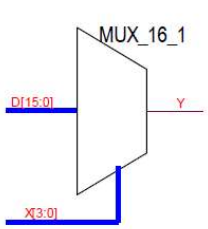
B	A	Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

Реализация логических элементов

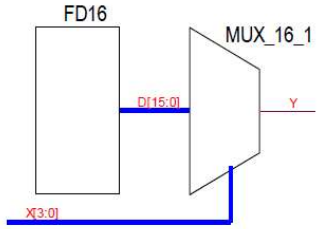
Look Up Table (LUT)



а



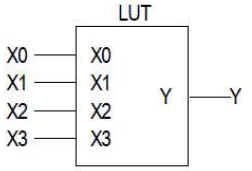
б



в

X3,X2 \ X1,X0	0,0	0,1	1,0	1,1
0,0	D0	D1	D2	D3
0,1	D4	D5	D6	D7
1,0	D8	D9	D10	D11
1,1	D12	D13	D14	D15

г

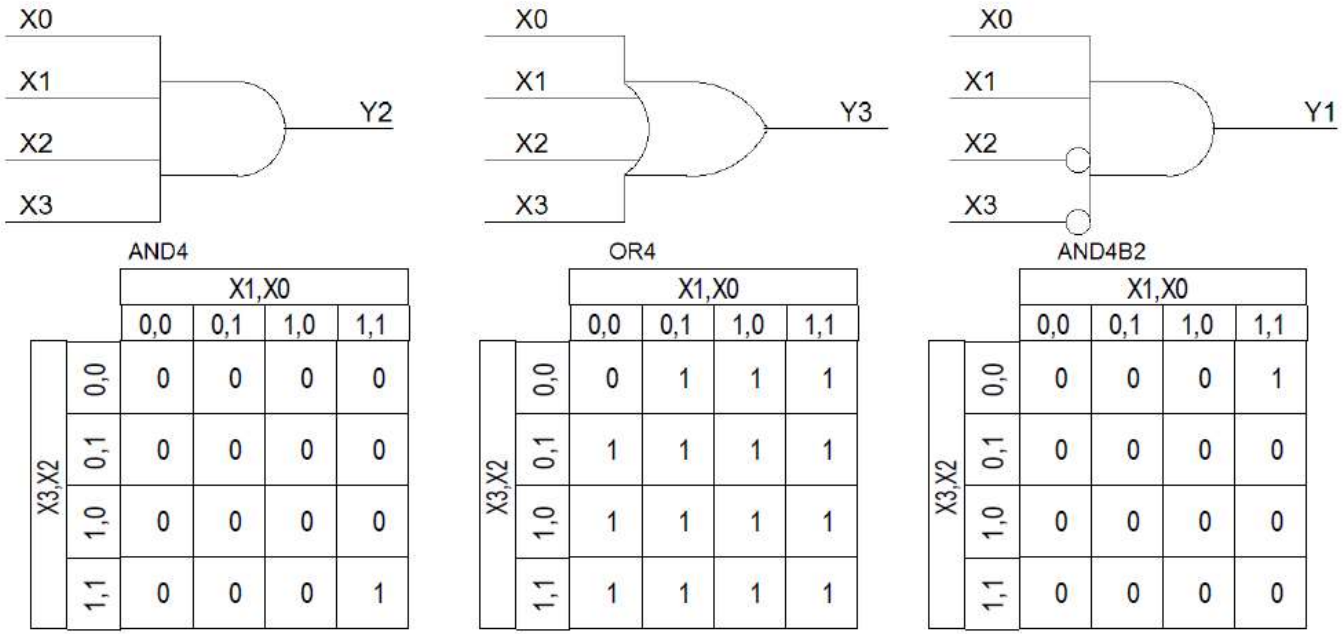


д

Генератор произвольной логической функции: а- схема мультиплексора б- условное обозначение мультиплексора, в- блок схема, г- таблица значений функции, д- условное обозначение

Реализация логических элементов

Look Up Table (LUT)



X3,X2 \ X1,X0	0,0	0,1	1,0	1,1
0,0	0	0	0	0
0,1	0	0	0	0
1,0	0	0	0	0
1,1	0	0	0	1

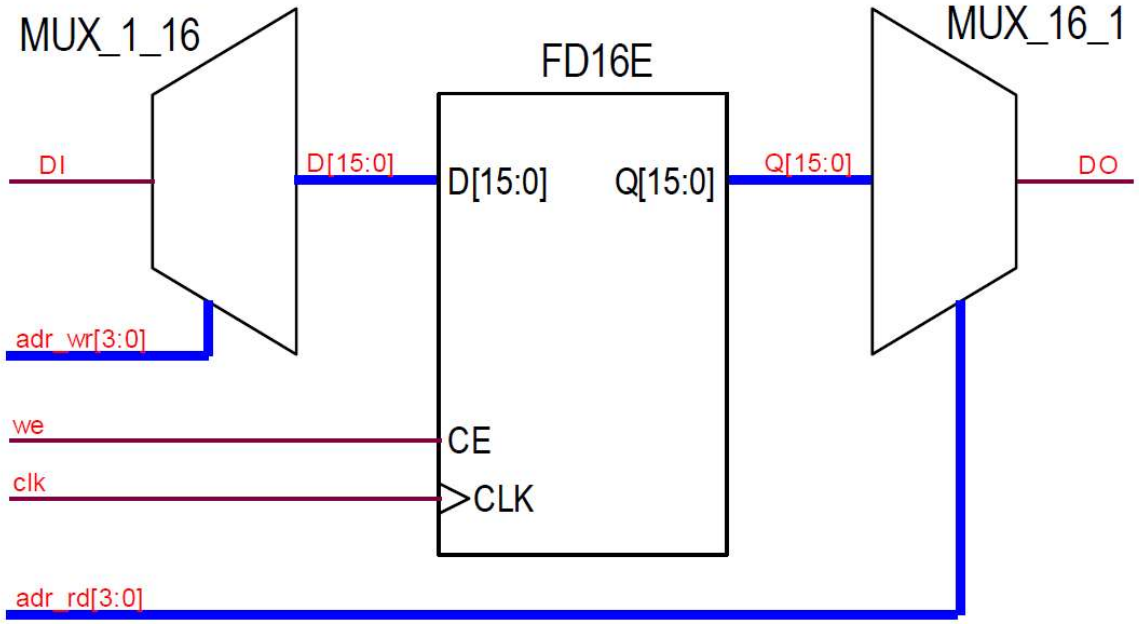
X3,X2 \ X1,X0	0,0	0,1	1,0	1,1
0,0	0	1	1	1
0,1	1	1	1	1
1,0	1	1	1	1
1,1	1	1	1	1

X3,X2 \ X1,X0	0,0	0,1	1,0	1,1
0,0	0	0	0	1
0,1	0	0	0	0
1,0	0	0	0	0
1,1	0	0	0	0

Примеры таблиц логических функций

Реализация логических элементов

Ячейка памяти 16-бит (Slice)



Реализация логических элементов

Асинхронный счетчик на D-триггерах

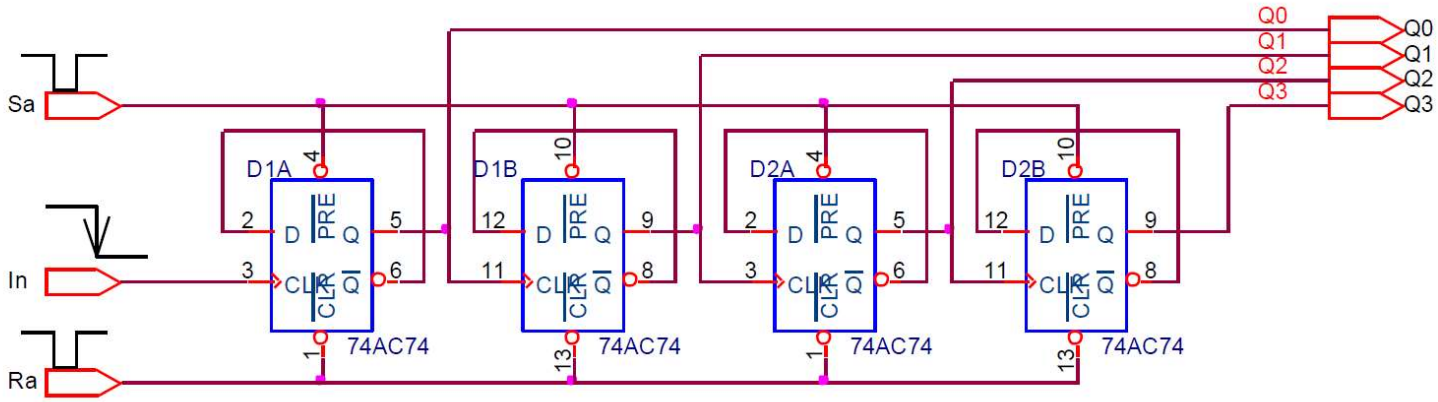


Схема асинхронного 4-х разрядного счетчика на D – триггерах интегральных схем

Реализация логических элементов

Асинхронный счетчик на D-триггерах

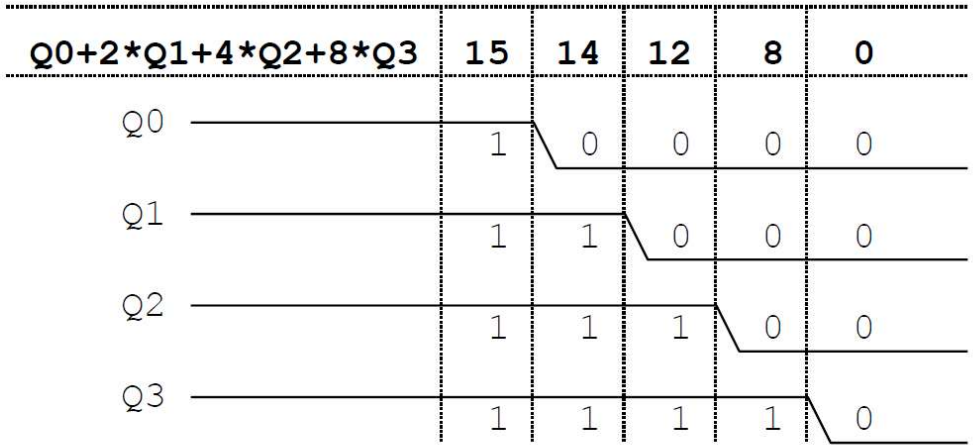
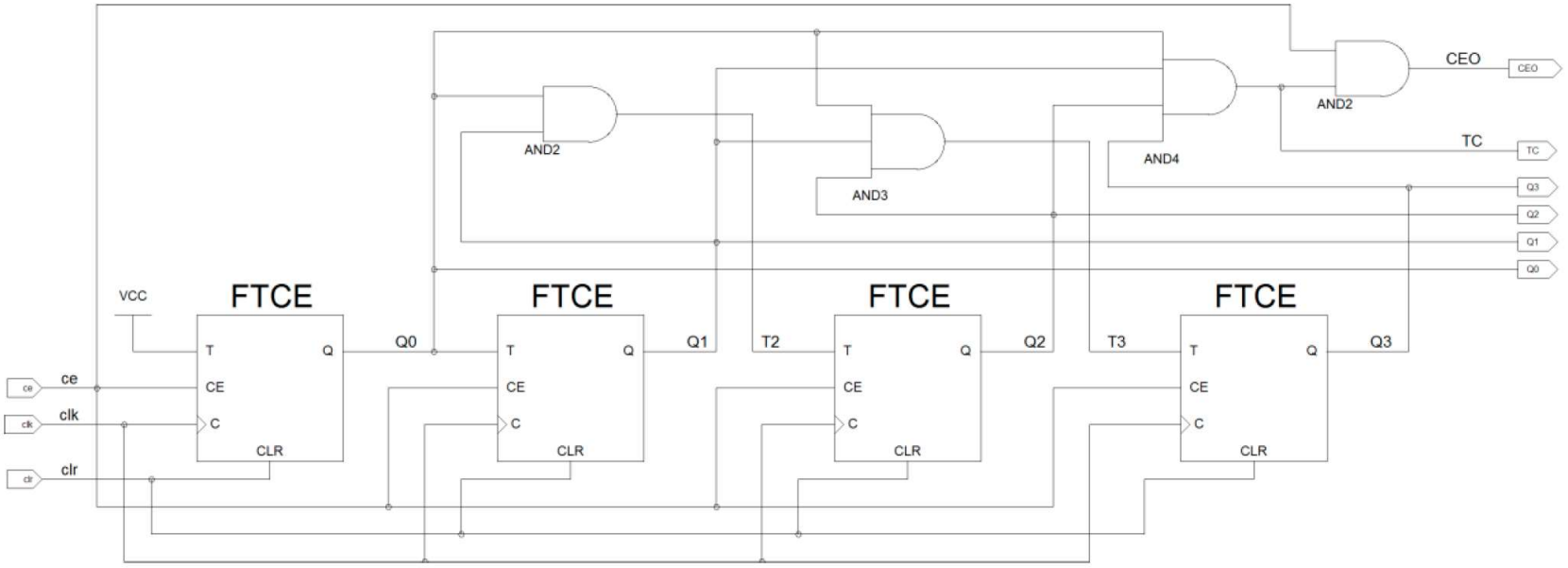


Диаграмма последовательного переключения триггеров асинхронного суммирующего счетчика из 15 в ноль

Реализация логических элементов

Синхронный счетчик



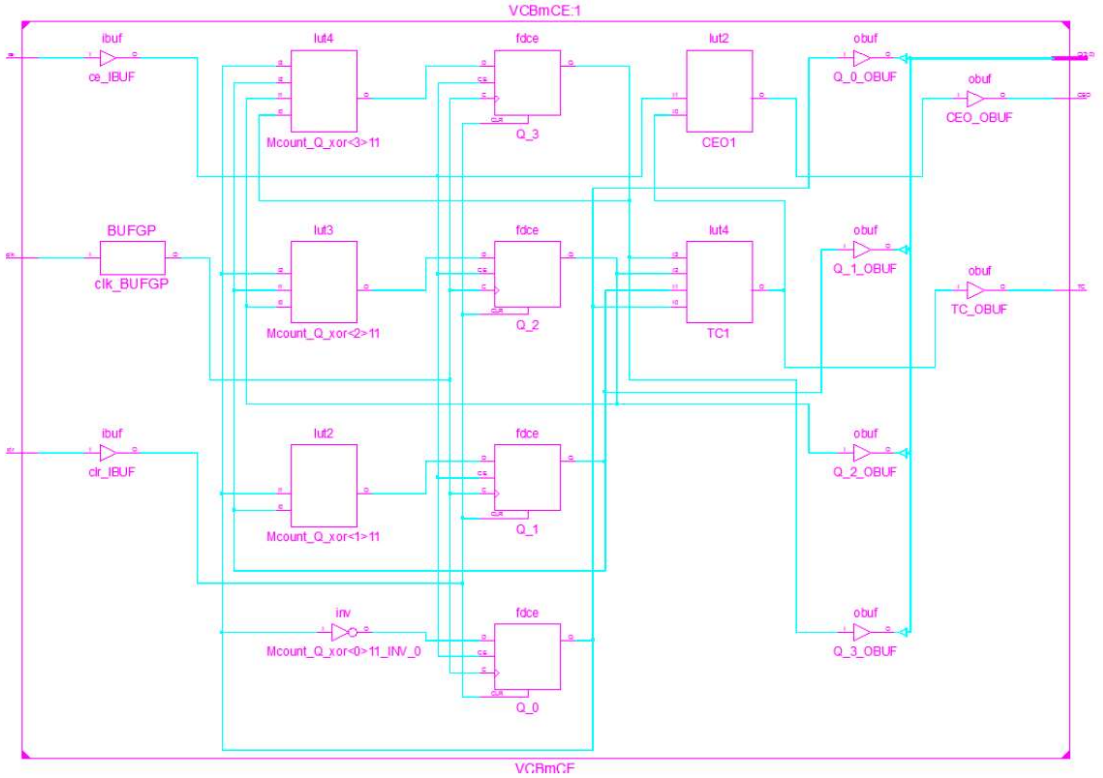
Реализация логических элементов

Verilog модуль суммирующего m- разрядного синхронного счетчика

```
`define m 4 //Число разрядов
module VCBmCE( // Объявление портов, их назначение и параметры
        input clk,      output wire TC,          //"Терминал каунт"
        input ce,       output wire CEO,         //Выход сигнала "переноса"
        input clr,      output reg [`m-1:0] Q=0); //Q=0 это инициализация счетчика
assign TC=(Q==(1<<`m)-1); //Присвоить объявленной цепи TC 1 при Q=2`m-1
assign CEO=TC & ce ;      //Присвоить объявленной цепи CEO (TC & ce)
always @ (posedge clk or posedge clr) begin
Q <= clr? 0 : ce? Q+1 : Q ; //Если clr=1,то Q<=0, иначе если ce=1, то Q<=Q+1 иначе Q<=Q
end
endmodule
```

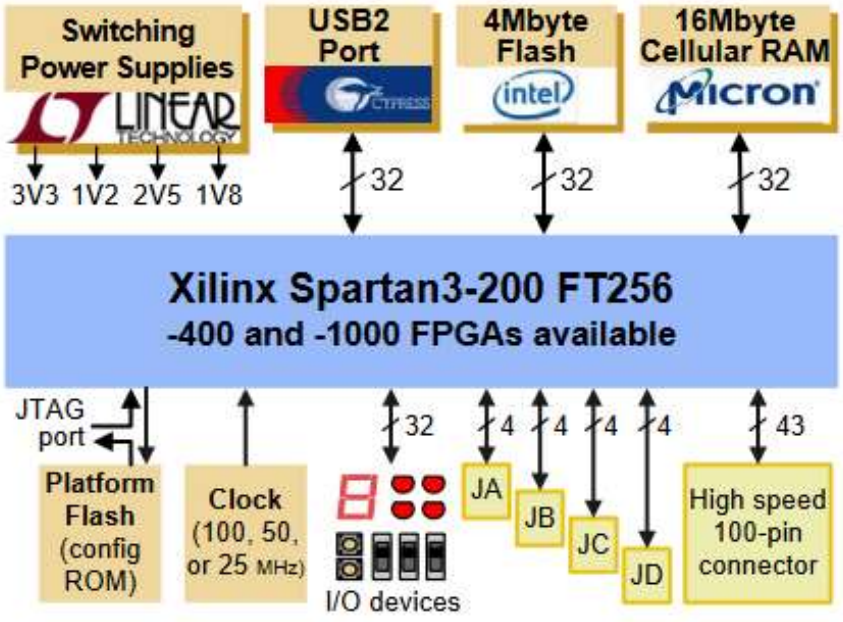
Реализация логических элементов

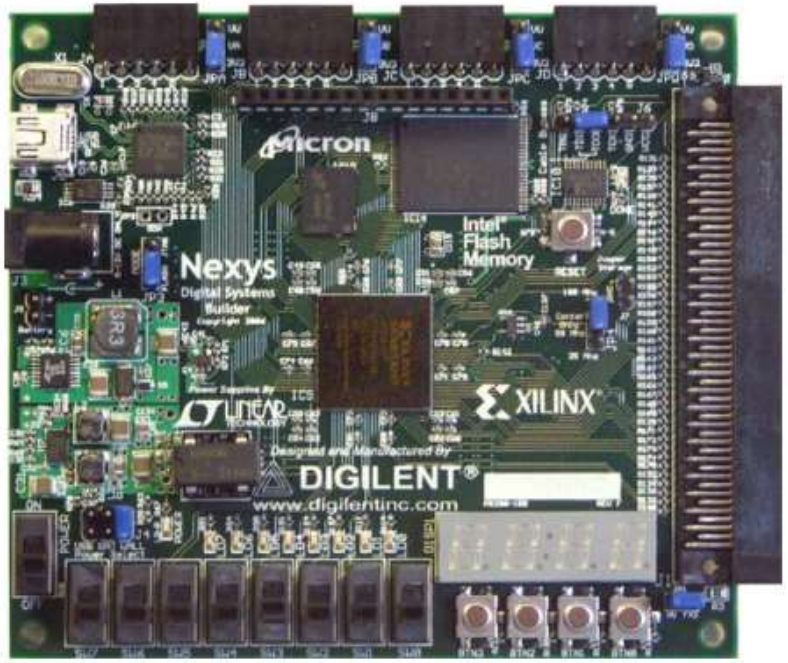
Синхронный счетчик



Макеты на ПЛИС

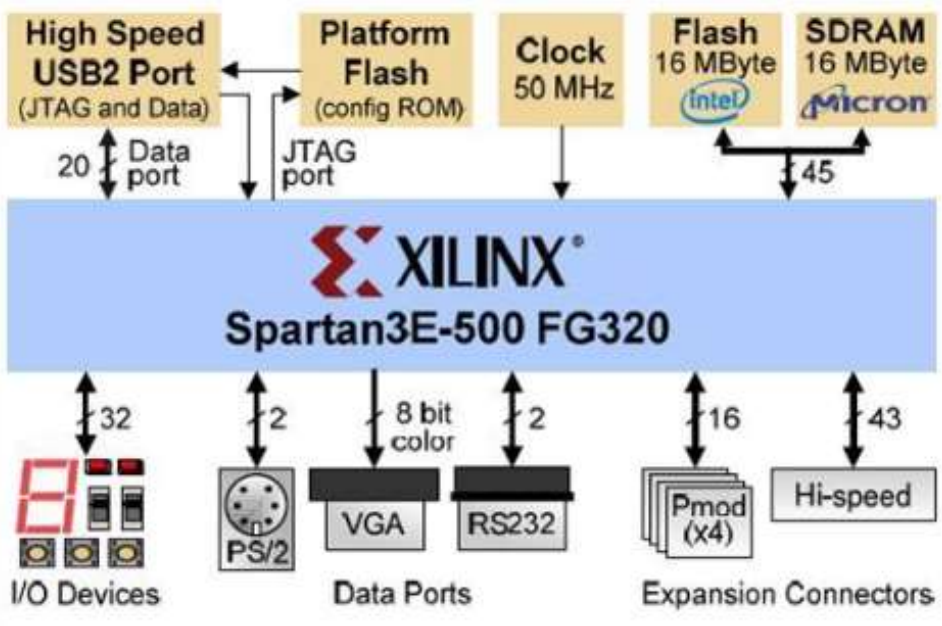
Digilent NEXUS

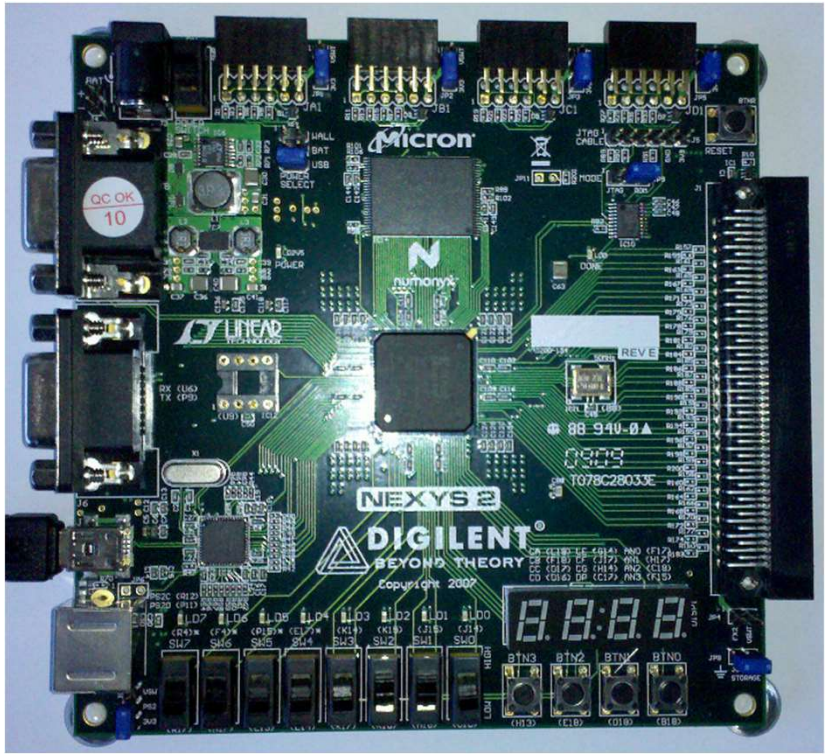




Макеты на ПЛИС

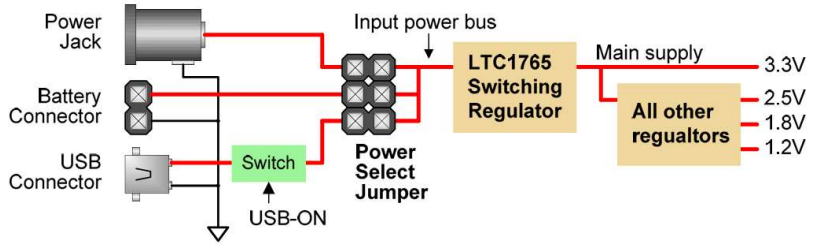
Digilent NEXUS 2

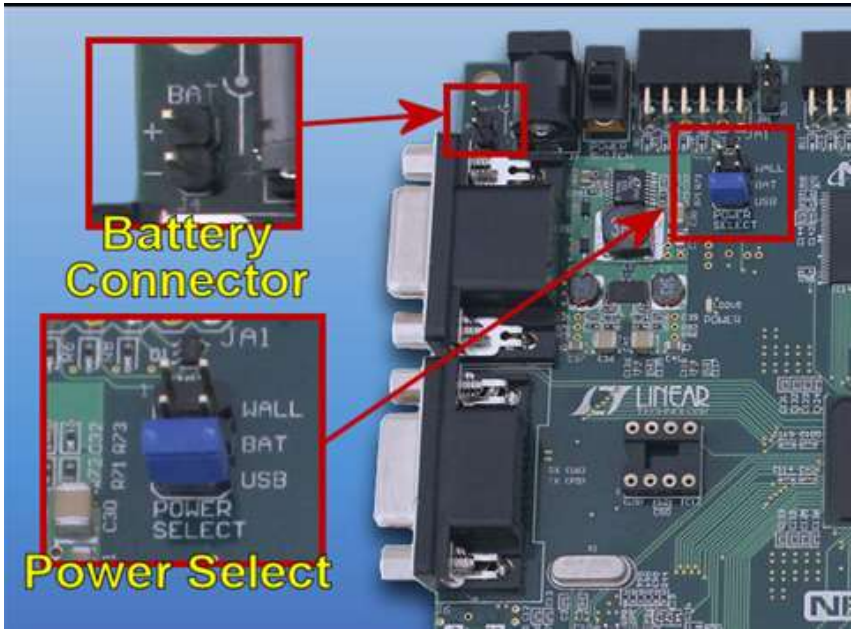




Макеты на ПЛИС

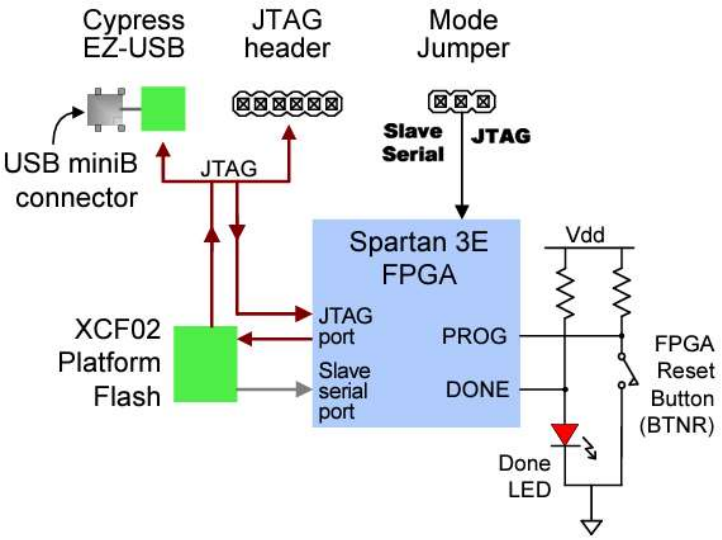
Digilent NEXUS 2 - питание

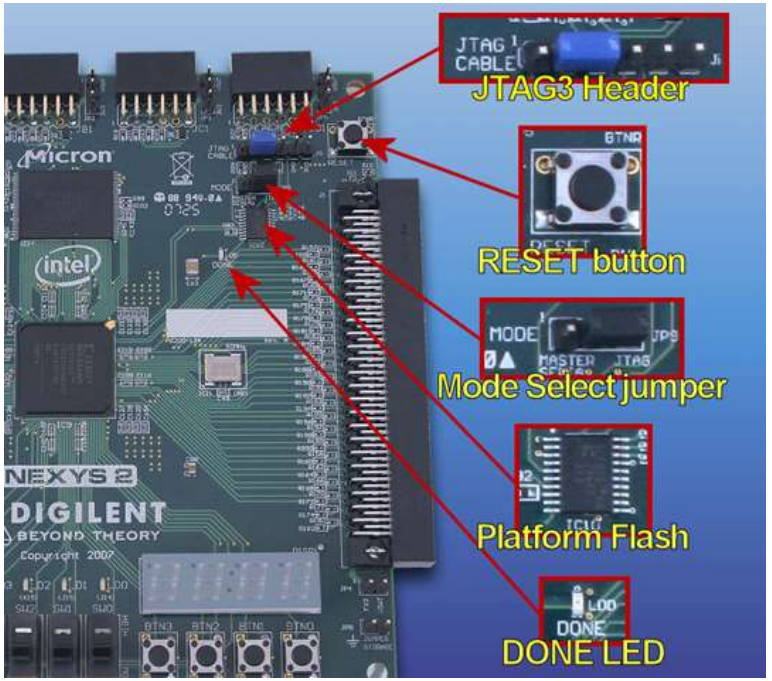




Макеты на ПЛИС

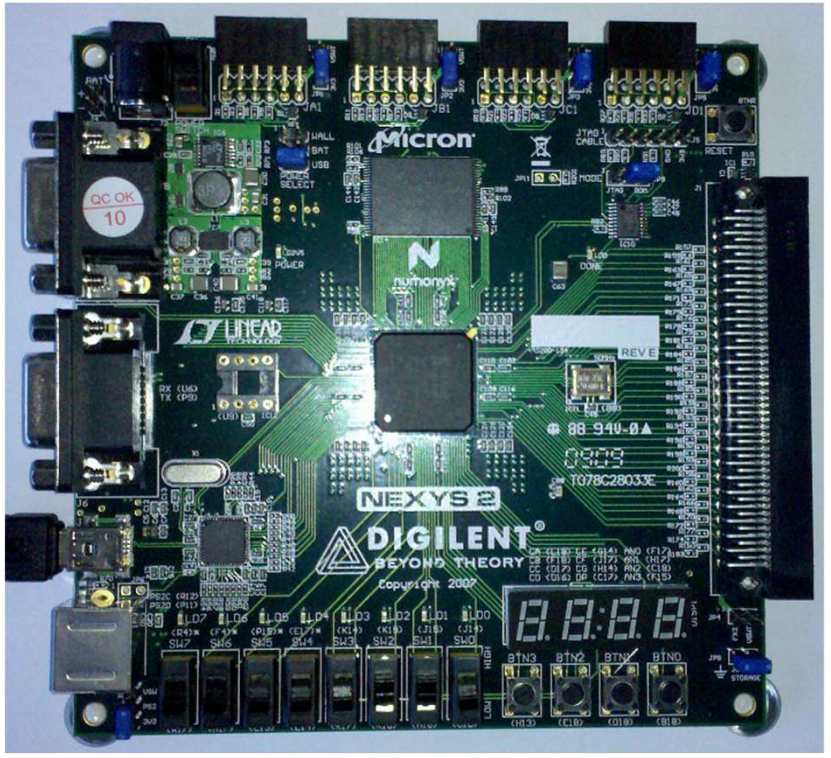
Digilent NEXUS 2 - прошивка





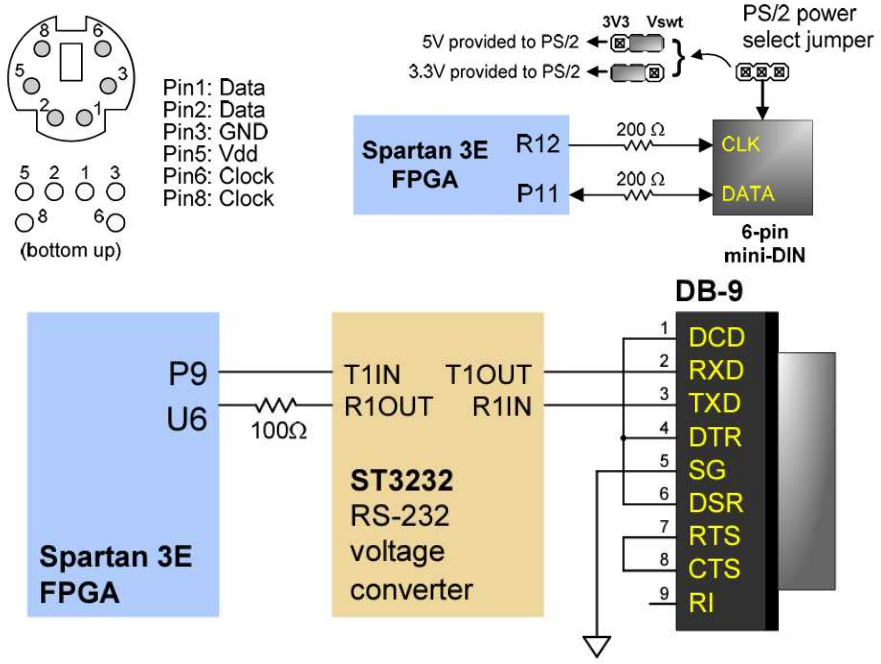
Макеты на ПЛИС

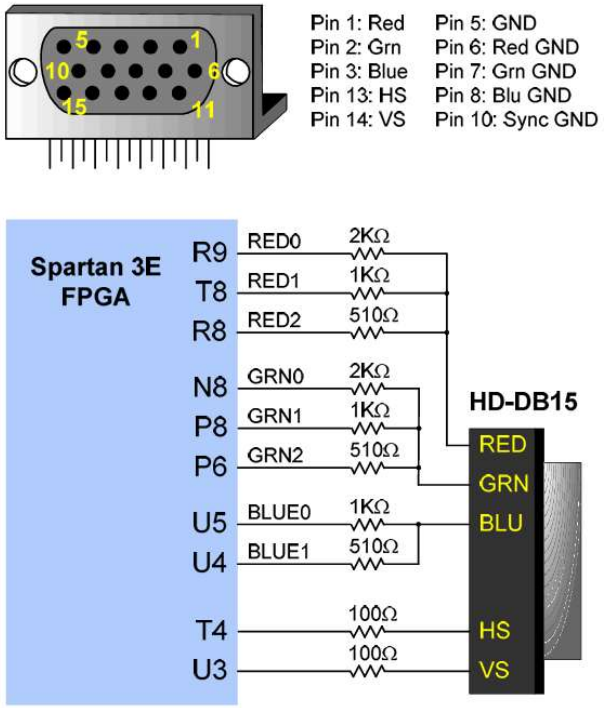
Digilent NEXUS 2 - синхронизация



Макеты на ПЛИС

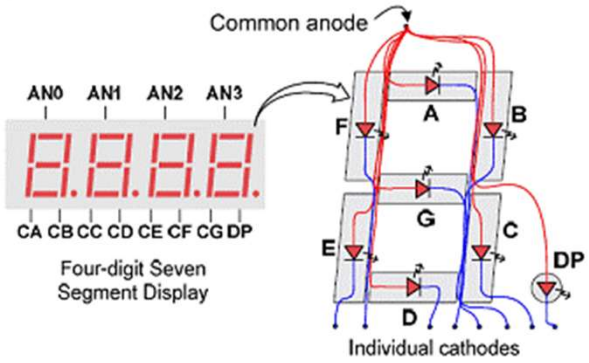
Digilent NEXUS 2 – ввод/вывод

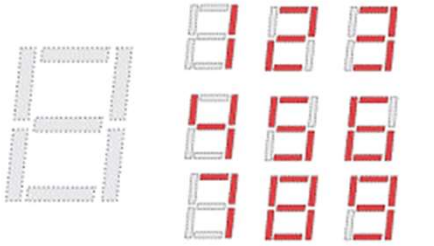




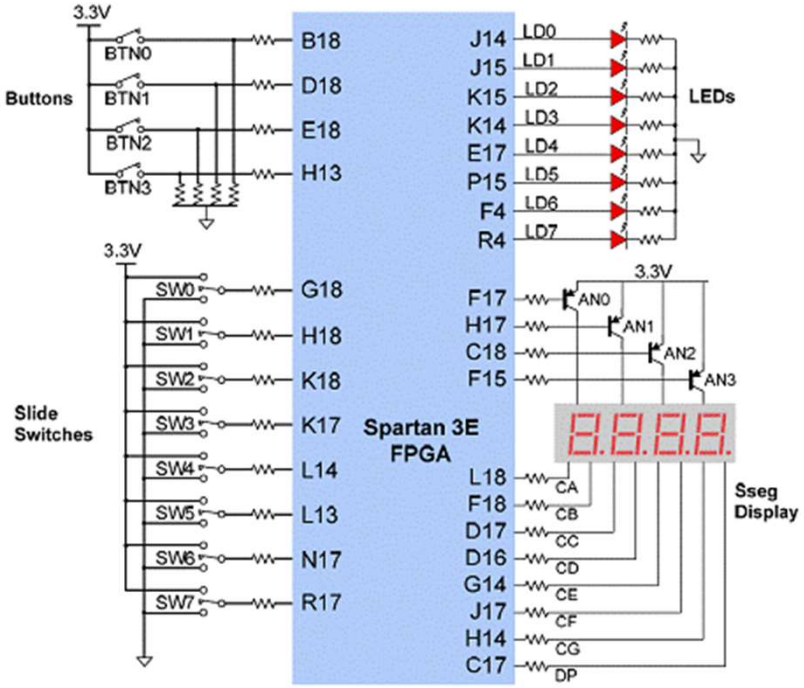
Макеты на ПЛИС

Digilent NEXUS 2 – ввод/вывод

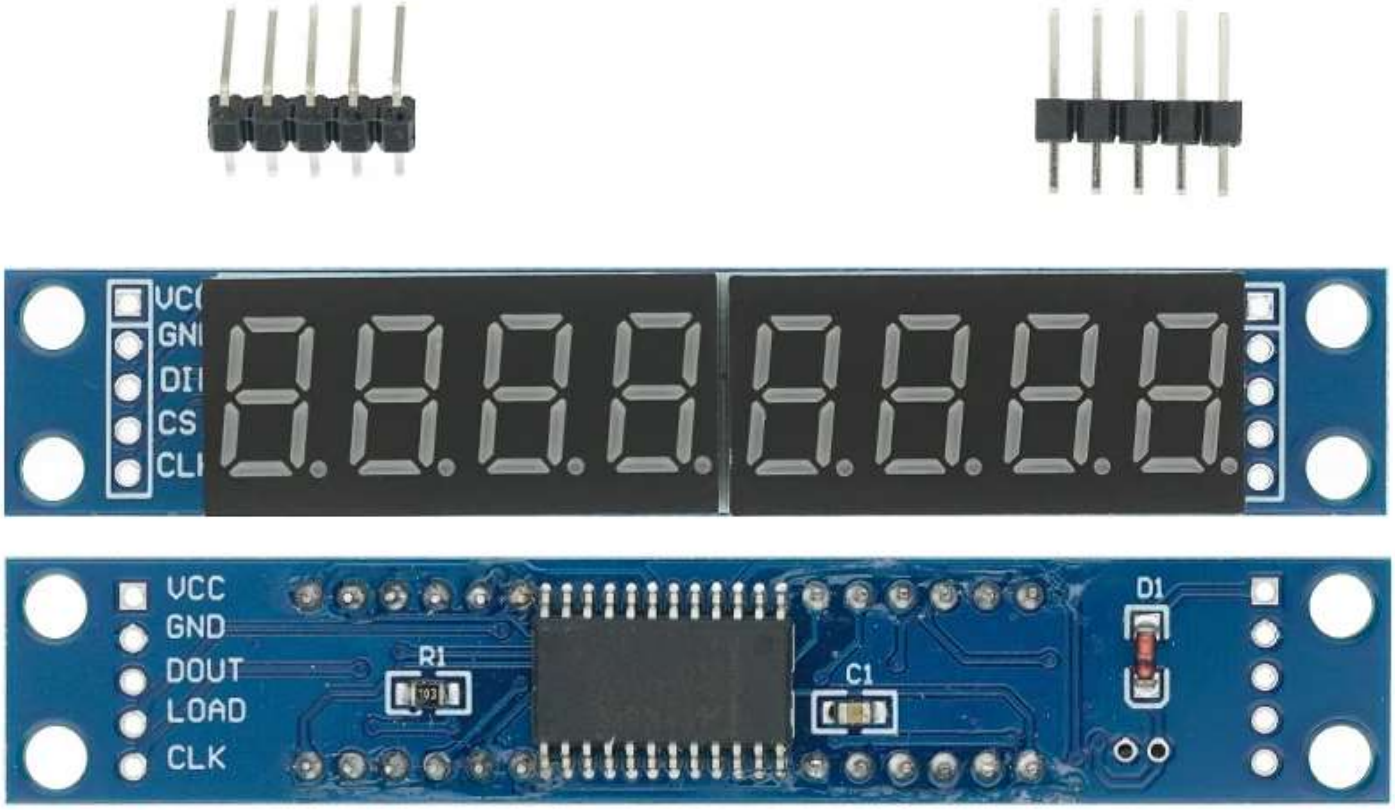




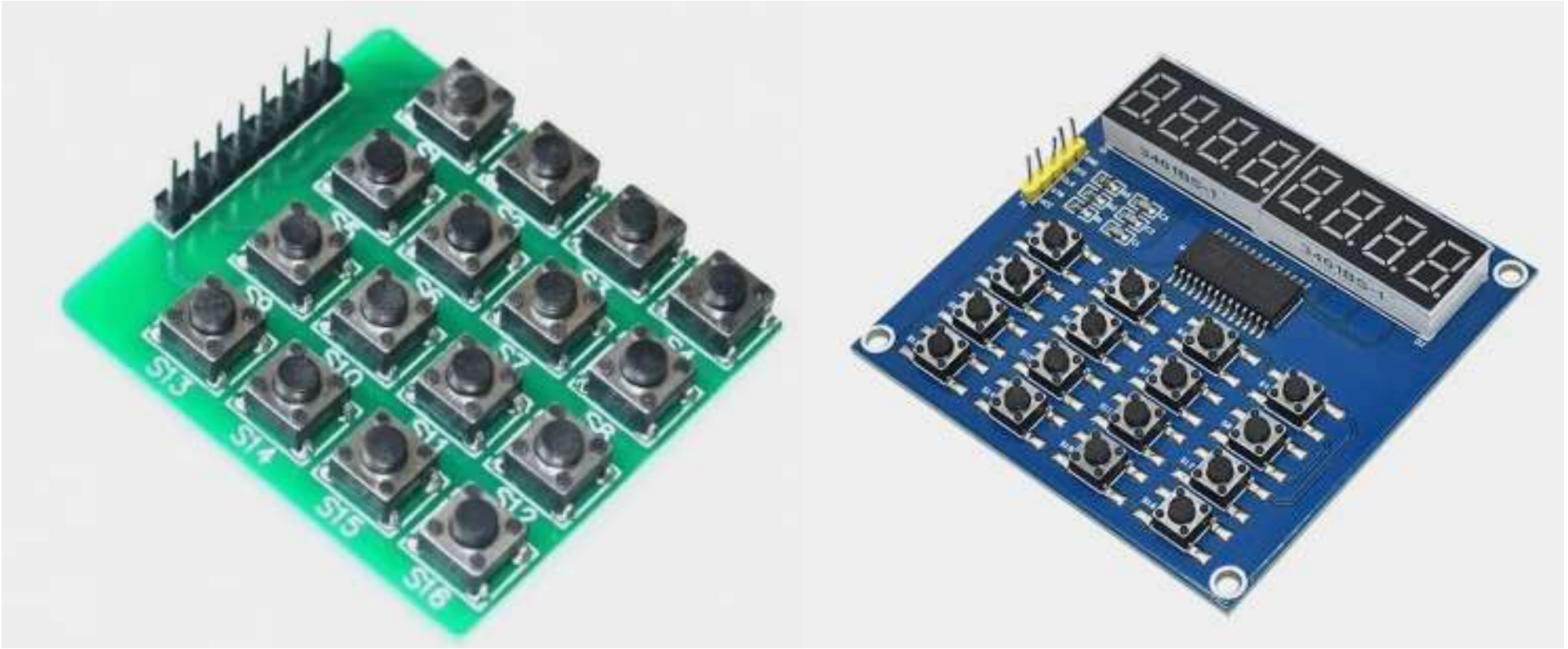
An un-illuminated seven-segment display, and nine illumination patterns corresponding to decimal digits



Макеты на ПЛИС



Макеты на ПЛИС



Макеты на ПЛИС

Digilent NEXUS 2 – ввод/вывод

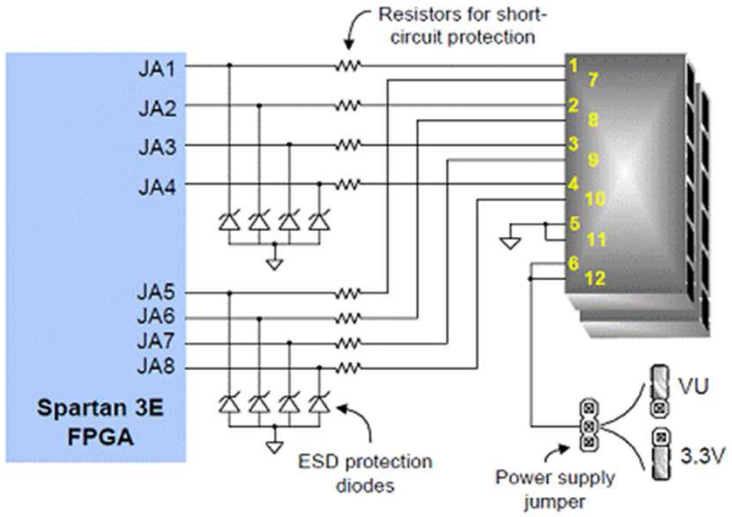
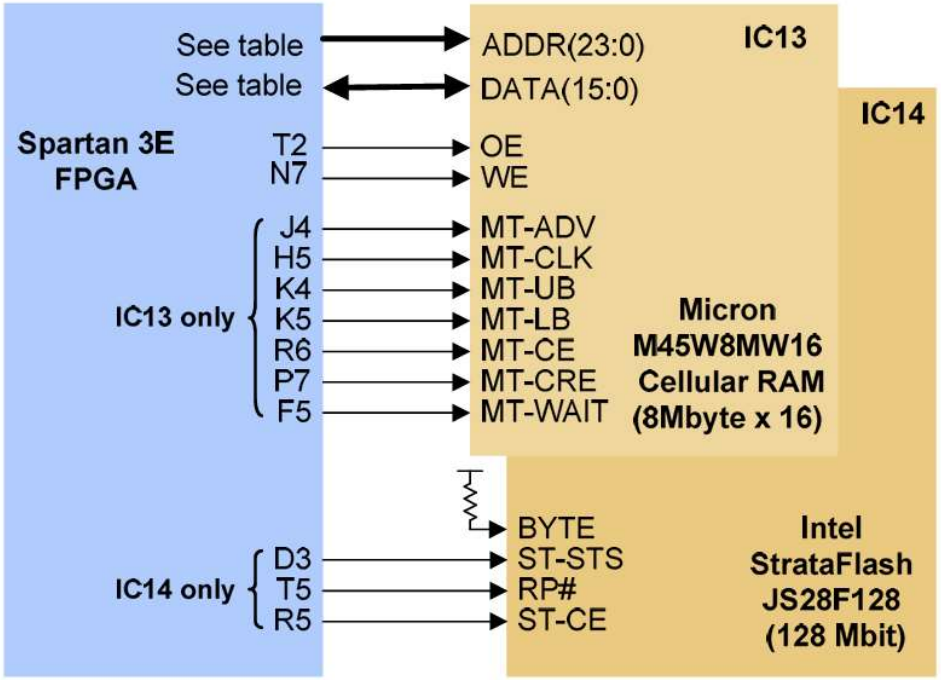


Table 3: Nexys2 Pmod Connector Pin Assignments							
Pmod JA		Pmod JB		Pmod JC		Pmod JD	
JA1: L15	JA7: K13	JB1: M13	JB7: P17	JC1: G15	JC7: H15	JD1: J13	JD7: K14[1]
JA2: K12	JA8: L16	JB2: R18	JB8: R16	JC2: J16	JC8: F14	JD2: M18	JD8: K15[2]
JA3: L17	JA9: M14	JB3: R15	JB9: T18	JC3: G13	JC9: G16	JD3: N18	JD9: J15[3]
JA4: M15	JA10: M16	JB4: T17	JB10: U18	JC4: H16	JC10: J12	JD4:; P18	JD10: J14[4]

Макеты на ПЛИС

Digilent NEXUS 2 – память



Memory Address and Data Bus Pin Assignments				
Address signals			Data signals	
ADDR0: NA	ADDR8: H6	ADDR16: M5	DATA0: L1	DATA8: L3
ADDR1: J1	ADDR9: F1	ADDR17: E2	DATA1: L4	DATA9: L5
ADDR2: J2	ADDR10: G3	ADDR18: C2	DATA2: L6	DATA10: M3
ADDR3: H4	ADDR11: G6	ADDR19: C1	DATA3: M4	DATA11: M6
ADDR4: H1	ADDR12: G5	ADDR20: D2	DATA4: N5	DATA12: L2
ADDR5: H2	ADDR13: G4	ADDR21: K3	DATA5: P1	DATA13: N4
ADDR6: J5	ADDR14: F2	ADDR22: D1	DATA6: P2	DATA14: R3
ADDR7: H3	ADDR15: E1	ADDR23: K6	DATA7: R2	DATA15: T1

Макеты на ПЛИС

Digilent NEXUS 2 примеры известных ошибок документации и модуля

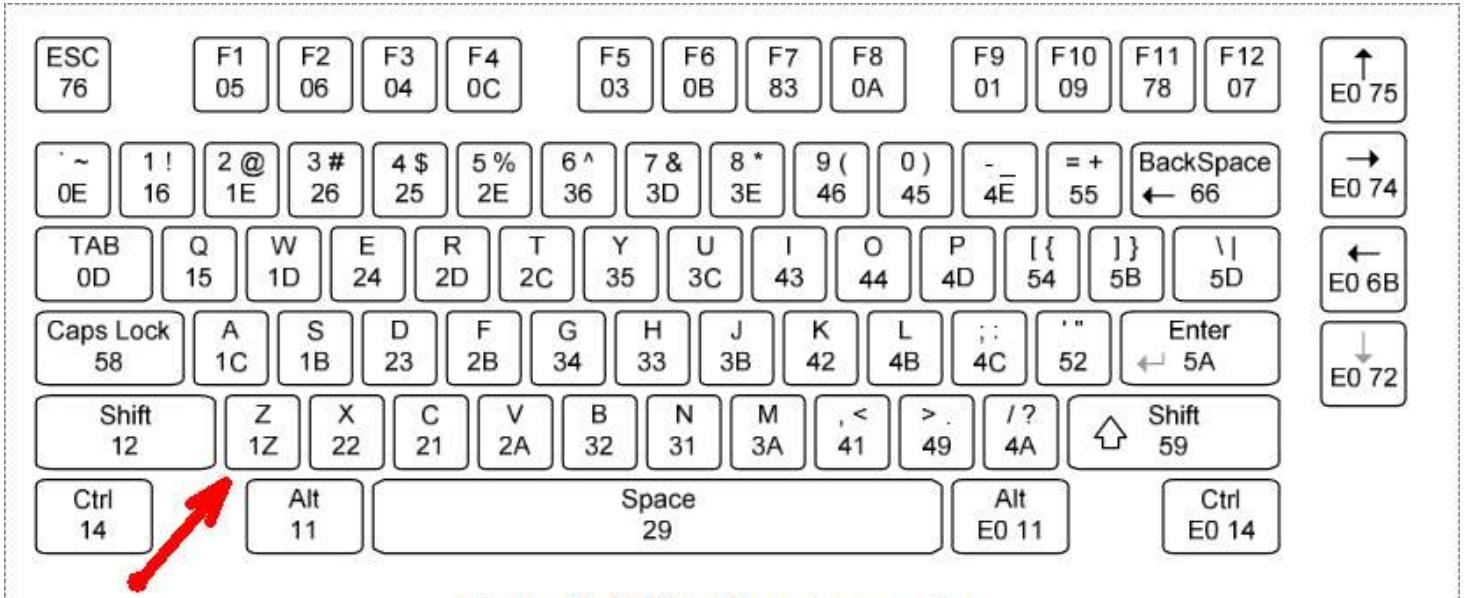
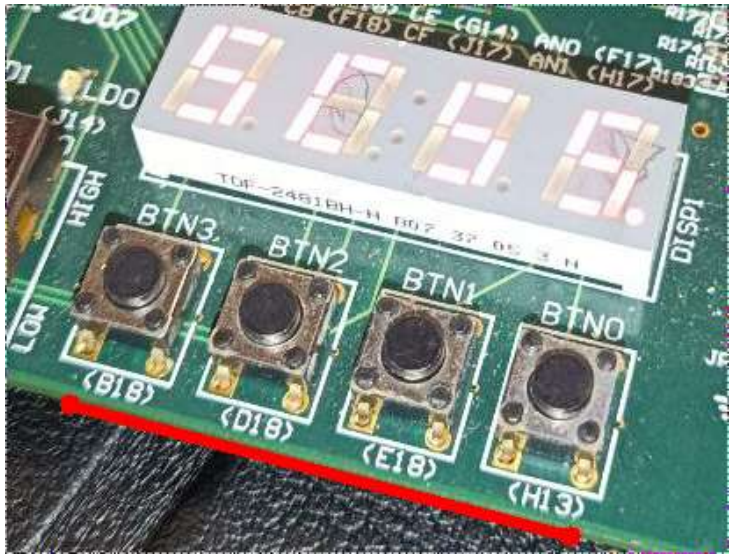
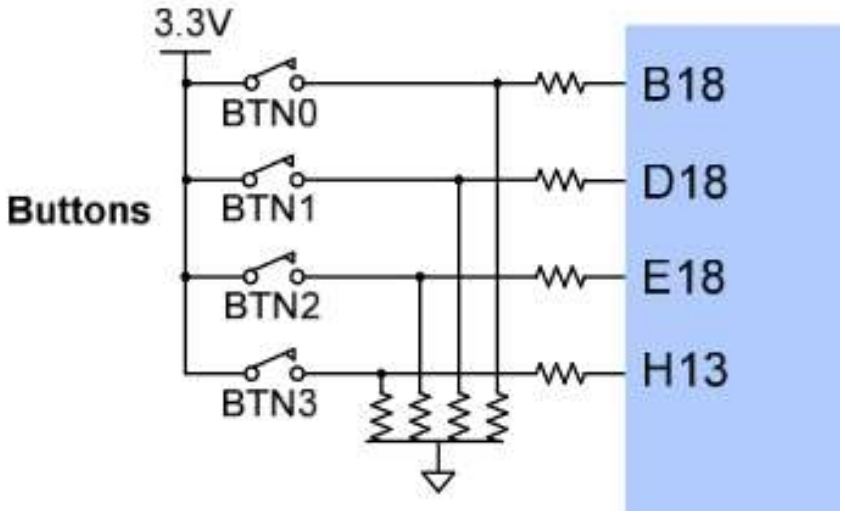


Figure 14: PS/2 keyboard scan codes

Макеты на ПЛИС

Digilent NEXUS 2 примеры известных ошибок документации и модуля





Макеты на ПЛИС

Digilent NEXUS 4

FPGA Part #	XC7A100T-1CSG324C
Logic Slices	15,850 (4 6-input LUTs & 8 flip-flops each)
Block RAM	4,860 Kbits
Clock Tiles	6 (each with PLL)
DSP Slices	240
Internal clock	450MHz+
Cellular RAM	16MB
Quad-SPI Flash	16 MB
Ethernet	10/100 PHY