Лабораторная работа Lab401

(версия 01.09.2022)

Счетчики импульсов на ПЛИС

Курс лабораторных работ «Импульсные цифровые устройства»

Лабораторная работа выполняется на макете **Digilent NEXYS2 Board** в среде разработки **Xilinx ISE Design Suite 14.4**

Содержание

Цель работы

В лабораторной работе предлагается на ПЛИС XC3S-500E-FG320 фирмы XILINX лабораторного макета NEXYS2 реализовать и изучить различные счетчики импульсов по предлагаемым схемам на VERILOG-е. Дополнительно для сдачи работы необходимо самостоятельно составить схему и отладить на макете схему подавления «дребезга» кнопки, а также схему измерения периода генератора импульсов.

1. Схемы счетчиков

1.1. Двоичный суммирующий счетчик с асинхронным входом сброса

Схема двухразрядного суммирующего счетчика с асинхронным входом clr сброса в ноль и с входом ce (Clock Enable) разрешения счета приведена на рис.1.1. В состав схемы входят два Т-триггера FTCE и два логических элемента AND2. Как правило, кроме выходов триггеров Q0, Q1 выходами счетчиков являются еще сигнал переполнения TC (Terminal Count) и сигнал переноса CEO (Clock Enable Output). В суммирующем счетчике TC=1, когда число на счетчике равно максимальному значению (Q0=Q1=1). Выход CEO предназначен для соединения с входом *ce* следующего счетчика.

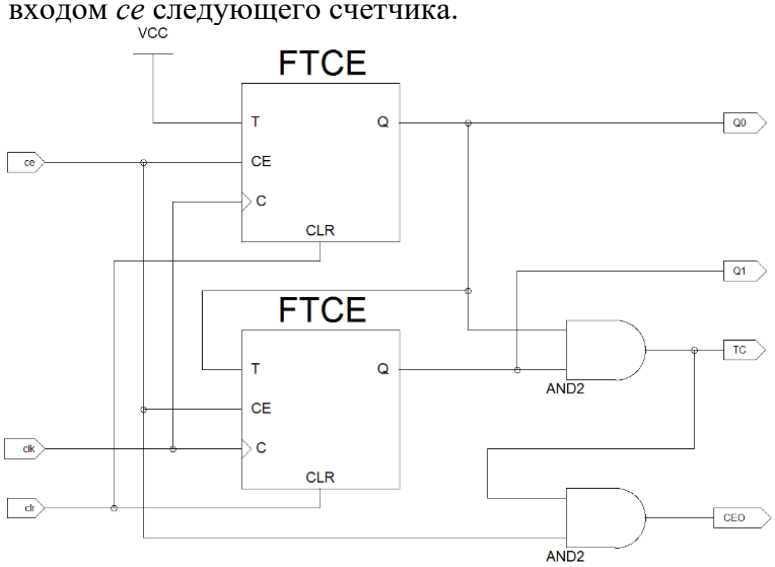


Рис.1.1 Схема двухразрядного суммирующего счетчика CB2CE с входом clr асинхронного сброса и с входом *ce* разрешения счета

В этом счетчике используются Т-триггеры FTCE с входом ce (Clock Enable) и входом clr асинхронного сброса в 0.

Особенность асинхронного входа clr состоит в том, что сброс в 0 происходит не по фронту, а по уровню clr=1 независимо от наличия сигнала синхронизации clk и уровней сигналов на других входах ce и T.

T - это вход разрешения переключения. Если ce=1, clr=0 и T=1, то по фронту сигнала синхронизации clk триггер переключается в противоположное состояние (Q<=!Q). При T=0 триггер не переключается (Q<=Q).

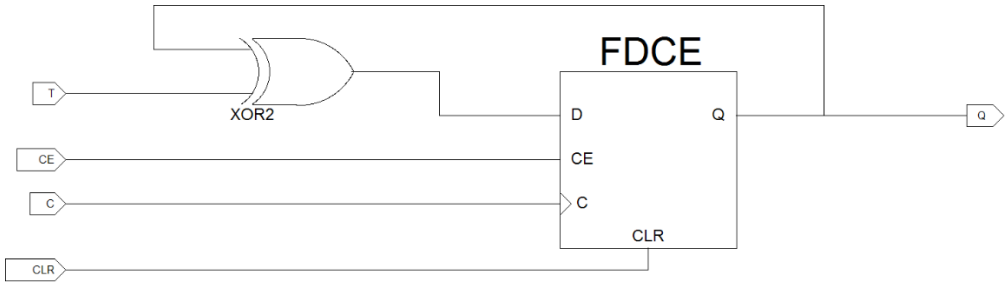


Рис.1.2 Схема FTCE триггера

FTCE триггер состоит из D-триггера FDCE и логического элемента XOR2 (исключающее ИЛИ), который выполняет функцию управляемого инвертора. При T=0 сигнал на его выходе повторяет Q, а при T=1 – инвертирует Q. Таким образом, T-триггер при clr=0 и ce=1 по фронту сигнала синхронизации переключается всегда, но при T=1 в противоположное состояние, а при T=0 в предыдущее состояние, т.е. не переключается.

Схема модуля суммирующего 2-х разрядного счетчика с асинхронным сбросом в ноль и с входом разрешения счета написанная на языке VERILOG:

```
module VCB2CE (   input ce,       output reg Q0 = 0,
                  input clk,      output reg Q1 = 0,
                  input clr,      output wire TC,
                                  output wire CEO);

      assign TC = (Q1 & Q0) ;      //Q0 & Q1=1
      assign CEO = ce & TC ;       //Сигнал переноса
      always @ (posedge clk or posedge clr) begin
            Q0 <= clr? 0 : ce? !Q0 : Q0 ; /* Если clr=1, то сброс в 0, иначе если ce=1, то
            "переключаться", иначе "стоять"*/
            Q1 <= clr? 0 : (ce & Q0)? !Q1 : Q1 ; /* Если clr=1, то сброс в 0, иначе если (ce
            & Q0)=1, то "переключаться", иначе "стоять"*/
      end
endmodule
```

После имени модуля в скобках перечисляются все порты модуля и указываются их функциональное назначение. Для наглядной совместимости с символом модуля желательно располагать входы слева, а выходы справа.

В этом модуле T-триггеры (регистры) Q0=0 и Q1=0 инициализируются в стартовое состояние 0. Инициализация (не обязательно в 0) необходима для обеспечения возможности логического моделирования (Simulate Behavior Model). Эта инициализация также обеспечивает заданное состояние регистров модуля при включении питания ПЛИС.

В любом триггере может быть только один вход синхронизации. И если в скобках always блока имеется два posedge (posedge name1 **or** posedge name2), то при синтезе схемы один из них — это вход синхронизации, а другой это вход асинхронного сброса. Для синтезатора, реализующего схему из компонент ПЛИС, имя асинхронного входа может быть любым. Важно чтобы оно было бы первым в последовательности условий (Q1<=name2? 0 : ...).

1.2. Двоичный суммирующий счетчик с синхронным входом сброса

Схема модуля суммирующего 2-х разрядного счетчика с синхронным сбросом в ноль и с входом разрешения счета, написанная на языке VERILOG:

```
module VCB2RE (   input ce,       output reg Q0 = 0,
                  input clk,      output reg Q1 = 0,
                  input r,        output wire TC,
                                  output wire CEO);
    assign TC = (Q1 & Q0) ;       //(Q0,Q1)=1
    assign CEO = ce & TC ;        //Сигнал переноса
    always @ (posedge clk) begin
        Q0 <= r? 0 : ce? !Q0 : Q0 ; /* Если r=1, то сброс в 0, иначе если ce=1, то "переключаться", иначе "стоять"*/
        Q1 <= r? 0 : (ce & Q0)? !Q1 : Q1 ; /* Если r=1, то сброс в 0, иначе если (ce & Q0)=1, то "переключаться", иначе "стоять"*/
    end
endmodule
```

В этом модуле сброс в 0 происходит также при r=1 независимо от сигналов на других входах, но только по фронту (posedge) сигнала синхронизации clk, т.е. без сигнала синхронизации сброс невозможен.

1.3. Соединение счетчиков

Для построения 4-х разрядного счетчика SCB4CE из двух 2-х разрядных счетчиков VCB2CE необходимо их входы clk соединить в общий вход clk, а входы clr в общий вход clr. Выход CEO первого (младшего) счетчика соединить с входом ce второго (старшего) счетчика. Проводникам выводов Q0 и Q1 младшего счетчика можно присвоить такие же имена, а проводникам выводов Q0 и Q1 старшего счетчика тогда необходимо присвоить имена Q2 и Q3. Выходной сигнал переноса TC есть логическое произведение сигналов TC1 и TC2 обоих счетчиков (TC=TC1 & TC2).

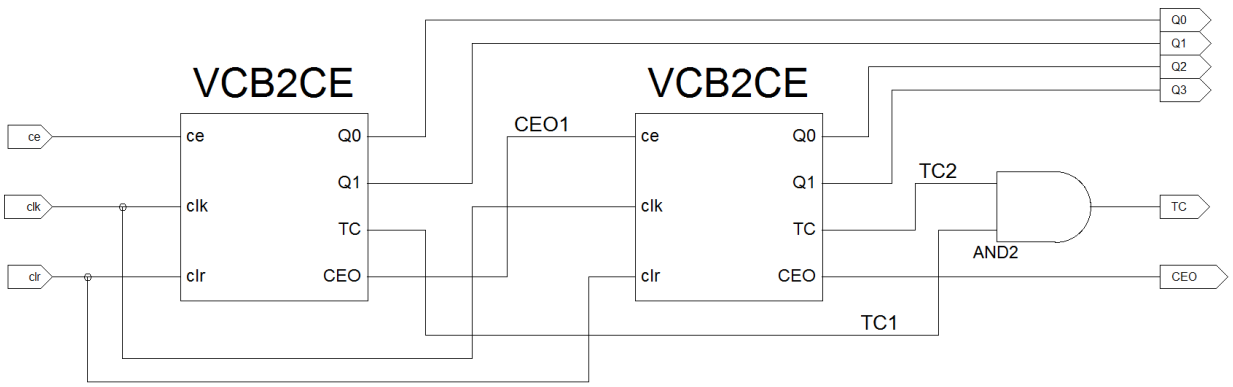


Рис.1.3 Схема четырехразрядного суммирующего счетчика SCB4CE состоящего из двух последовательно соединенных двухразрядных суммирующих счетчиков VCB2CE

Схема модуля четырехразрядного суммирующего счетчика, состоящего из двух последовательно соединенных двухразрядных суммирующих счетчиков, написанная на языке VERILOG:

```
module VCB4CE (   input ce,       output wire Q0,
                  input clk,      output wire Q1,
                  input clr,      output wire Q2,
                                  output wire Q3,
                                  output wire TC,
```

```
                                output wire CEO);
        wire CEO1, TC1, TC2;
        assign TC = TC1 & TC2;
        VCB2CE DD1 (.ce(ce),        .Q0(Q0),
                    .clk(clk),      .Q1(Q1),
                    .clr(clr),      .TC(TC1),
                                    .CEO(CEO1));

        VCB2CE DD2 (.ce(CEO1),  .Q0(Q2),
                    .clk(clk),      .Q1(Q3),
                    .clr(clr),      .TC(TC2),
                                    .CEO(CEO));
endmodule
```

Этот модуль является примером того, как составлять схему из «самодельных» VERILOG модулей. В данном случае в состав схемы входят два, вышеописанных модуля **VCB2CE**, позиционные обозначения которых DD1 и DD2 (могут быть произвольными). После позиционного обозначения дается список используемых портов модуля. Здесь также, ради наглядности порты ввода желательно располагать слева, а порты вывода справа. Перед именем порта модуля ставится точка, а в скобках имя проводника. Проводники связей между модулями, которых нет в портах ввода вывода, должны быть вначале объявлены отдельно (wire CEO1, TC1, TC2;). Проводнику “output wire TC” присваивается значение соответствующей логической функции (assign TC = TC1 & TC2).

1.4. Параметрическое задание числа разрядов

При создании модулей с произвольным числом разрядов удобно пользоваться параметрическим заданием числа разрядов. Пример такого модуля приведен ниже.

Схема суммирующего *m*-разрядного счетчика с асинхронным сбросом в ноль и с входом разрешения счета написанная на языке VERILOG:

```
`define m 4
module VCBmCE (input ce,        output reg [`m-1:0] Q = 0,
               input clk,       output wire TC,
               input clr,       output wire CEO);
        assign TC = (Q==(1<<`m)-1) ;        //Q0 & Q1 &...& Q'm-1 ==1
        assign CEO = ce & TC ;              //Сигнал переноса
        always @ (posedge clk or posedge clr ) begin
                Q <= clr? 0 : ce? Q+1 : Q ; /* Если clr=1, то сброс в 0 независимо от clk, иначе
                если ce=1, то "суммировать", иначе "стоять"*/
        end
endmodule
```

В этом счетчике число разрядов Q[`m-1:0] (триггеров) задано через `define параметром `m. Выражение 1<<`m означает сдвиг 1 на `m разрядов влево, т.е. 1<<`m =$2^{`m}$, а TC=(Q[`m-1:0]==($2^{`m}$ -1) означает, что TC=1, когда все Q[i] в $Q = \sum_{i=0}^{`m-1} Q[i] \cdot 2^i$ равны 1.

Для инкремента Q[`m-1:0] на 1 должны использоваться `m-разрядные регистр и сумматор, но синтезатор, реализующий схему из компонент ПЛИС, «знает», что для этой функции можно использовать схему счетчика.

Схема суммирующего 4-х разрядного счетчика с синхронным сбросом в ноль и с входом ce разрешения счета написанная на языке VERILOG:

```
module VCB4RE (input ce,          output reg [`m-1:0] Q = 0,
               input clk,         output wire TC,
               input R,           output wire CEO);
    assign TC = (Q==15) ;         //Q0 & Q1 &...& Q'm-1 ==1
    assign CEO = ce & TC ;        //Сигнал переноса
    always @ (posedge clk) begin
        Q <= R? 0 : ce? Q+1 : Q ; /* Если R=1, то сброс в 0, иначе если ce=1, то
        "суммировать", иначе "стоять"*/
    end
endmodule
```

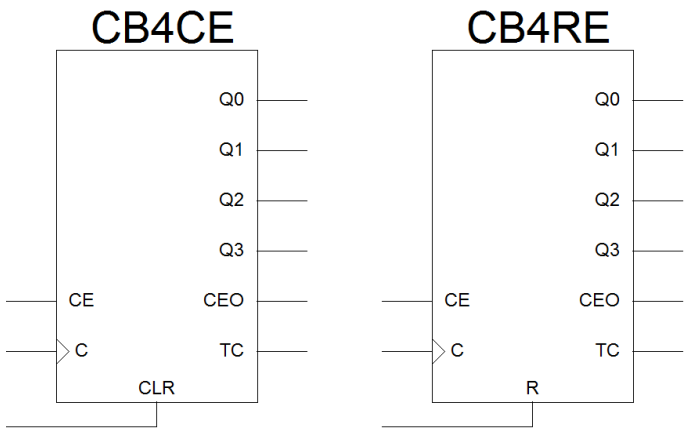


Рис.1.4 Символы суммирующих 4-х разрядных суммирующих счетчиков с асинхронным (CLR) и синхронным (R) сбросом в ноль и входом разрешения счета (CE)

1.5. Декадный счетчик

В декадном счетчике число состояний Q[3:0] равно 10 (0,1,…8,9). TC=1 при Q=9. Сброс в 0 происходит по фронту сигнала синхронизации при Q=9 и ce=1 или при R=1.

Схема суммирующего декадного счетчика с синхронным сбросом в ноль и с входом разрешения счета написанная на языке VERILOG:

```
module VCDRE  (    input clk,     output wire TC,
                   input ce,      output wire CEO,
                   input R,       output reg [3:0] Q=0 );
    assign TC = (Q==9) ;
    assign CEO = ce & TC ;
    always @ (posedge clk) begin
        Q <= (R | CEO)? 0 : ce? Q+1 : Q ;
    end
endmodule
```

1.6. Вычитающий счетчик

Схема вычитающего *m*-разрядного счетчика с синхронной установкой в 2^m-1 и с входом разрешения счета написанная на языке VERILOG:

```
`define m 4
module VCBDmSE (input ce,                output reg [`m-1:0] Q = (1<<`m)-1,
                input clk,               output wire TC,
                input s,                 output wire CEO);
        assign TC = (Q==0) ;             //Q0,Q1,...Q'm-1 ==0
        assign CEO = ce & TC ;           //Сигнал переноса
        always @ (posedge clk) begin
                Q <= s?  ((1<<`m)-1) : ce? Q-1 : Q ; /* Если s=1, то запись 2^m-1, иначе если
                ce=1, то "вычитать", иначе "стоять"*/
        end
endmodule
```

В вычитающем счетчике TC=1, когда число на счетчике равно минимальному значению Q[`m-1:0]=0.

1.7. Реверсивный счетчик

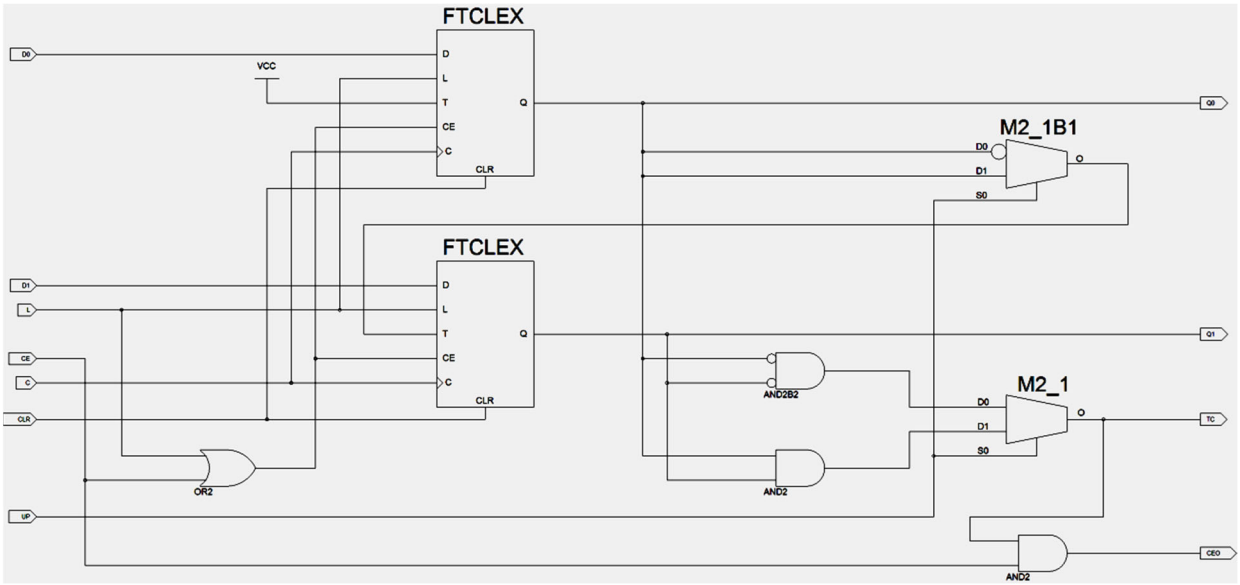


Рис.1.5 Схема библиотечного модуля двухразрядного реверсивного счетчика с входом *CLR* (асинхронного сброса в 0), входом *L* (разрешения синхронной загрузки), входом *UP* (направления счета) и входом *CE* (разрешения счета)

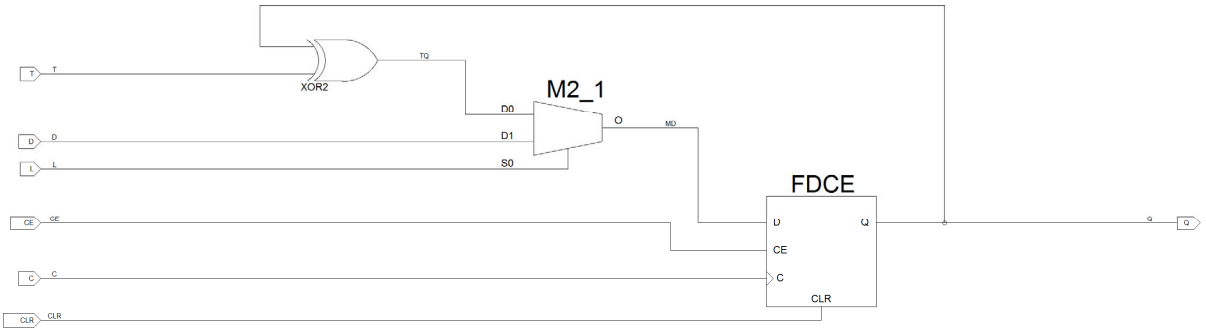


Рис.1.6 Схема загружаемого Т-триггера FTCLEX используемого в реверсивном счетчике

Схема *m*-разрядного реверсивного счетчика, написанная на языке VERILOG:

```
`define m 4
module VCBmCLED(input ce,                output reg [`m-1:0] Q =0,
                input up,                output wire CEO,
                input [`m-1:0] di,       output wire TC,
                input L,
                input clk,
                input clr);
        assign TC = up? (Q==(1<<`m)-1) : (Q==0) ; /* если up=1, то TC=1 при Q=2^m-1, иначе
        TC=1 при Q=0 */
        assign CEO = ce & TC ;
        always @ (posedge clr or posedge clk) begin
                if (clr) Q <= 0;  //асинхронный сброс
                else     Q <= L? di : (up & ce)? Q+1 : (!up & ce)? Q-1 : Q ;
        end
endmodule
```

При clr=L=0, ce=1 и up=1 по каждому фронту сигнала синхронизации clk Q увеличивается на 1 (Q<=Q+1), а при up=0 - Q уменьшается на 1 (Q<=Q-1). При L=1 не зависимо от ce по фронту сигнала синхронизации Q принимает значение di (Q[`m-1:0]<=di[`m-1:0]). Высокий уровень clr=1 удерживает Q[`m-1:0]=0 независимо от уровней сигналов на других входах.

1.8. Счетчик Джонсона

Схема 4-х разрядного счетчика Джонсона приведена на рис.1.8. Счетчик Джонсона состоит из последовательно соединенных D-триггеров (выход Q[i] с входом D[i+1], i=0,1,…m-1). Выход Q[m-1] последнего (старшего) соединяется с входом D[0] первого (младшего) D-триггера через инвертор. В библиотечном модуле счетчика Джонсона нет выходов TC и CEO.

В счетчике Джонсона возможно несколько устойчивых циклов. Период цикла для исходного состояния Q[m-1:0]=0 равен 2*m*Tce (или Tclk, если ce=1);

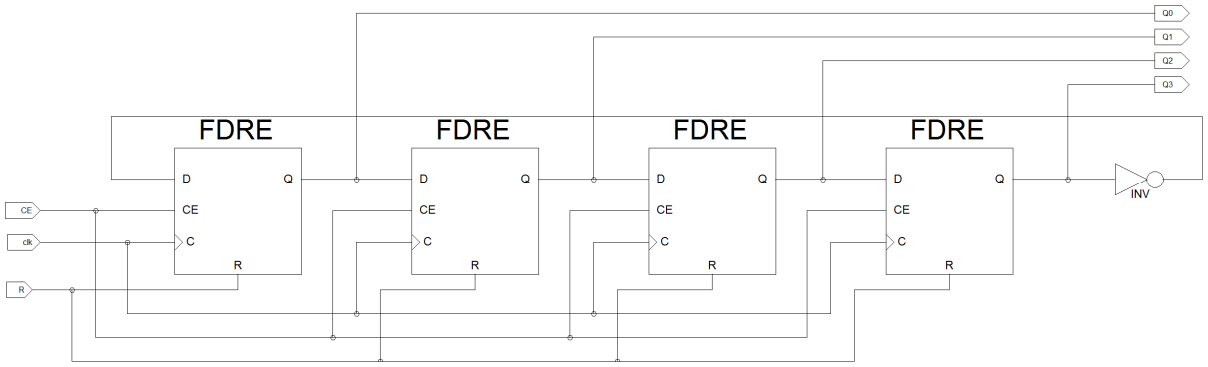


Рис.1.7 Схема библиотечного модуля 4-х разрядного счетчика Джонсона

Таблица 1

Nt	Q3	Q2	Q1	Q0	X		Q3	Q2	Q1	Q0	X
0	0	0	0	0	0		0	0	1	0	2
1	0	0	0	1	1		0	1	0	1	5
2	0	0	1	1	3		1	0	1	1	B
3	0	1	1	1	7		0	1	1	0	6
4	1	1	1	1	F		1	1	0	1	D
5	1	1	1	0	E		1	0	1	0	A
6	1	1	0	0	C		0	1	0	0	4
7	1	0	0	0	8		1	0	0	1	9
8	0	0	0	0	0		0	0	1	0	2

В таблице 1 показаны возможные состояния $X = \sum_{i=0}^{3} Q_i \cdot 2^i$ 4-х разрядного счетчика Джонсона:

- 0,1,3,7,F,E,C,8 - для исходного состояния 0000,
- 2,5,B,6,D,A,4,9 - для исходного состояния 0010.

Схема *m*-разрядного счетчика Джонсона на VERILOG-e:

```
`define m 4
module VCJmRE    (input ce,        output wire TC,
                  input clk,       output wire CEO,
                  input R,         output reg[`m-1:0] Q = 0);
        assign TC = (Q==(1<<`m)-1) ;        //q0,q1,...q'm-1 ==1
        assign CEO = ce & TC ; //Сигнал переноса
        always @ (posedge clk) begin
                Q <= R? 0 : ce? Q<<1  |  !Q[`m-1] : Q ;
        end
endmodule
```

1.9. Счетчик в коде Грея

Таблица 2

X	x_3	x_2	x_1	x_0		*Y*	y_3	y_2	y_1	y_0	x_0
0	0	0	0	0		0	0	0	0	0	0
1	0	0	0	1		1	0	0	0	1	1
2	0	0	1	0		3	0	0	1	1	0
3	0	0	1	1		2	0	0	1	0	1
4	0	1	0	0		6	0	1	1	0	0
5	0	1	0	1		7	0	1	1	1	1

6	0	1	1	0		5	0	1	0	1	0
7	0	1	1	1		4	0	1	0	0	1
8	1	0	0	0		12	1	1	0	0	0
9	1	0	0	1		13	1	1	0	1	1
10	1	0	1	0		15	1	1	1	1	0
11	1	0	1	1		14	1	1	1	0	1
12	1	1	0	0		10	1	0	1	0	0
13	1	1	0	1		11	1	0	1	1	1
14	1	1	1	0		9	1	0	0	1	0
15	1	1	1	1		8	1	0	0	0	1
							q_4	q_3	q_2	q_1	q_0

В таблице 2 приведены значения бит двоичного кода X и кода Грея Y.

Цифры y_i (i=0,1,2,…m-1) *m*-разрядного кода Грея связаны с цифрами x_i двоичного кода следующим образом:

$$y_{m-1} = x_{m-1}, \qquad x_{m-1} = y_{m-1},$$
$$y_{m-2} = x_{m-2} \wedge x_{m-1}, \qquad x_{m-2} = y_{m-2} \wedge y_{m-1},$$
$$y_{m-3} = x_{m-3} \wedge x_{m-2}, \qquad x_{m-3} = y_{m-3} \wedge y_{m-2} \wedge y_{m-1},$$
$$\ldots\ldots\ldots\ldots\ldots \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$y_1 = x_1 \wedge x_2, \qquad x_1 = y_1 \wedge y_2 \wedge \ldots\ y_{m-2} \wedge y_{m-1},$$
$$y_0 = x_0 \wedge x_1, \qquad x_0 = y_0 \wedge y_1 \wedge y_2 \wedge \ldots\ y_{m-2} \wedge y_{m-1}.$$

Особенностью кода Грея является то, что в отличие от двоичного кода, где соседние числа могут отличаться во многих и даже во всех разрядах, в коде Грея любые соседние числа отличаются только в одном разряде. Применительно к счетчику эта особенность кода Грея проявляется в том, что при переходе в соседнее состояние всегда переключается только один триггер. А это означает, что счетчик в коде Грея создает существенно меньше электромагнитных помех, чем двоичный счетчик, в котором возможно одновременное переключение многих триггеров.

Из таблицы 2 видно, что условиями для переключения очередного разряда являются:

$y_0 = q_1 - q_0 = 0,$
$y_1 = q_2 - \{ q_1, q_0\} = 11,$
$y_2 = q_3 - \{ q_2, q_1, q_0\} = 101,$
$y_3 = q_4 - \{ q_3, q_2, q_1, q_0\} = 1001$, где q[4:0] регистр кода Грея (Y[3:0]= q[4:1]).

Схема 4-х разрядного счетчика в коде Грея на VERILOG:

```
module VCGrey4Re ( input clk,     output wire [3:0] Y, //Код Грея
                   input ce,      output wire CEO,
                   input r,       output wire TC);
    reg [4:0]q = 0;
    assign TC = (q[4:0]==((1<<4) | 1)) ;
    assign CEO = ce & TC ;
    assign Y = q[4:1] ;
    always @ (posedge clk) begin
        q[0] <= (r | CEO)? 0 : ce? !q[0]: q[0]      ; // Дополнительный триггер
        q[1] <= (r | CEO)? 0 : ((q[0]==0) & ce)? !q[1] : q[1];
        q[2] <= (r | CEO)? 0 : ((q[1:0]==((1<<1) | 1)) & ce)? !q[2] : q[2] ;
        q[3] <= (r | CEO)? 0 : ((q[2:0]==((1<<2) | 1)) & ce)? !q[3] : q[3] ;
        q[4] <= (r | CEO)? 0 : ((q[3:0]==((1<<3) | 1)) & ce)? !q[4] : q[4] ;
    end
endmodule
```

2. Логическое моделирование

Для проведения логического моделирования (цель моделирования - получение временных диаграмм сигналов на всех входах и выходах модуля) необходимо создать модуль задания моделирования (Verilog Test Fixture), представляющий собой оснастку, внешний модуль, который будет формировать сигналы воздействия на наш моделируемый модуль (синхронизацию clk, сигналы на логических входах моделируемого модуля). Например, для VCBmCE в качестве задания моделирования создадим tf_VCBmCE. Описание портов создается автоматически (в приведенном ниже примере выделено курсивом). Сигналы входов также автоматически объявляются регистрами, на которые в разделе initial begin необходимо задать желаемые временные диаграммы.

Периодические сигналы задаются до initial begin. Например, сигнал синхронизации clk с периодом 20 нс можно задать следующим образом:

```
parameter Tclk=20;
always begin clk=1; #(Tclk/2); clk=0; #(Tclk/2); end,
```

а периодический сигнал ce с периодом 160 нс и длительностью 20 нс аналогичным образом:

```
parameter Tce=160;
always begin ce=1; #(Tclk/8); ce=0; #(7*Tclk/8); end
```

Пример модуля задания на моделирование (Verilog Test Fixture):

```
module tf_VCBmCE;
	// Inputs
	reg ce;
	reg clk;
	reg clr;
	// Outputs
	wire [2:0] Q;
	wire TC;
	wire CEO;
	// Instantiate the Unit Under Test (UUT)
	VCBmCE uut (
		.ce(ce),
		.Q(Q),
		.clk(clk),
		.TC(TC),
		.clr(clr),
	.CEO(CEO) );
	//Генратор периодичеккого сигнала синхронизации clk
	parameter Tclk=20; //Период сигнала синхронизации 20 нс
	always begin clk=1; #(Tclk/2); clk=0; #(Tclk/2); end

	// Генератор периодического сигнала ce
	parameter Tce=160; //Период сигнала ce 160 нс
	always begin ce=0; #(7*Tce/8); ce=1; #(1*Tce/8); end //Только целая часть деления

	initial begin		// Initialize Inputs
				clr = 0; //Исходное состояние входов
```

```
        #380;       clr = 1; //Через 38 нс 1
        #10;        clr = 0; // Через 10 нс        0
        #611;       clr = 1; //Через 611 нс 1
        #270;       clr = 0; // Через 270 нс 0
    end
endmodule
```

В этом модуле задания на моделирование сигнал clr через 380 нс после начала на 10 нс, а затем еще раз через 611 нс на 270 нс устанавливается равным 1.

Сигналы clk и ce задаются генераторами периодических сигналов.

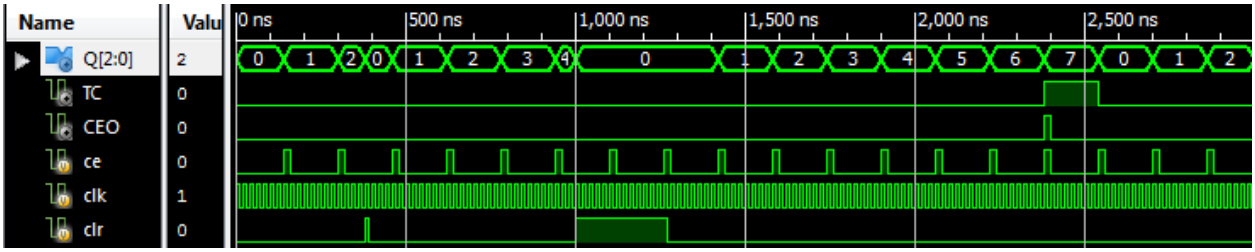


Рис.2.1. Пример временных диаграмм логического моделирования модуля VCBmCE (m=3)

На приведенной временной диаграмме состояние триггеров Q[2:0] счетчика отображается в виде положительного декадного числа (Radix – Unsigned Decimal). Высокий уровень сигнала clr=1 сбрасывает в 0 триггеры и удерживает их в 0, независимо от сигналов на других входах (ce и clk) счетчика.

3. Описание макета

Лабораторная работа выполняется на макете NEXYS 2. Структурная схема макета NEXYS 2 приведена на рис.3.1. Из всех устройств макета, кроме ПЛИС (Spartan3E-500 FG320), для выполнения работы достаточно использовать:

- Clock генератор 50 MHz,
- USB порт, который обеспечивает питание макета и загрузку конфигурации (связей между компонентами),
- I/O Devices - устройства ввода вывода.

В состав I/O Devices входят:

- 8 переключателей, которыми можно задавать 8 бит данных,
- 4 кнопки,
- 8 светодиодов и
- 4-х разрядный семи сегментный светодиодный индикатор (рис.3.2).

ПЛИС Spartan3E-500 FG320 имеет достаточно большой объем компонент и для выполнения существенно более сложных работ:

- 9312 триггеров (FD),
- 9312 табличных 4-х входных генераторов логических функций (4 inputs LUTs),
- 232 доступных для использования блоков ввода вывода (bonded IOBs),
- 73 кбит распределенной памяти (Slice RAM),
- 16 аппаратных модулей блочной памяти с суммарной емкостью 360 кбит
- 20 аппаратных умножителей 18x18 бит.

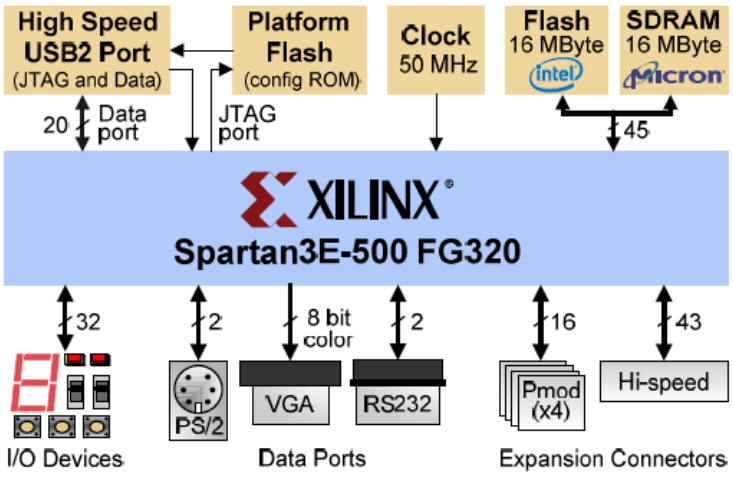


Рис.3.1 Структурная схема макета NEXYS 2

Рис.3.2 Устройства (I/O Devices) отображения и ввода информации макета NEXYS 2

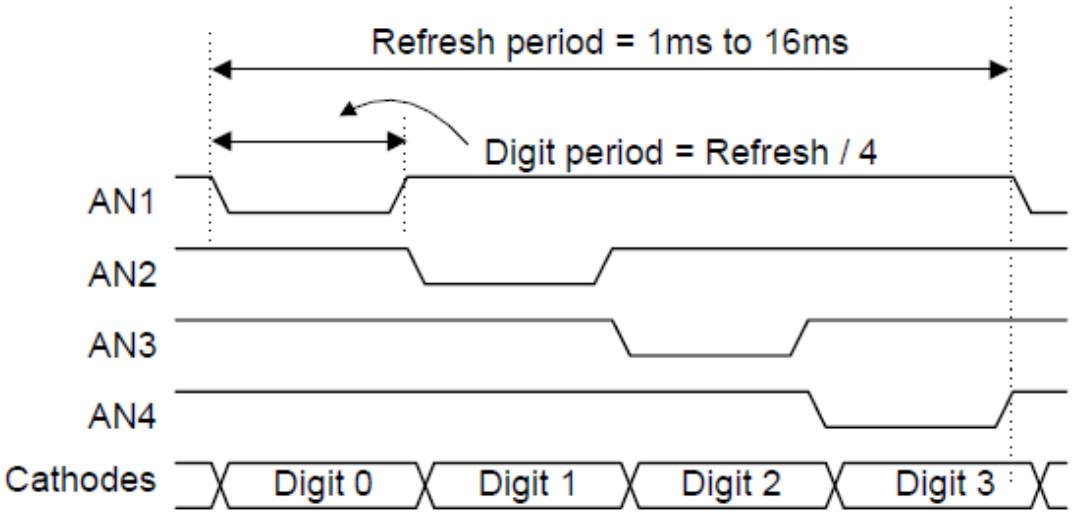


Рис.3.3 Временные диаграммы поочередного включения цифр индикатора

Светодиодный семи сегментный динамический индикатор имеет 4 цифры. Аноды светодиодов каждой цифры соединены вместе и напряжение (+3.3V) подается на них через транзисторные ключи. Одноименные сегменты (катоды светодиодов) всех четырех цифр также соединены вместе, поэтому цифры необходимо включать поочередно.

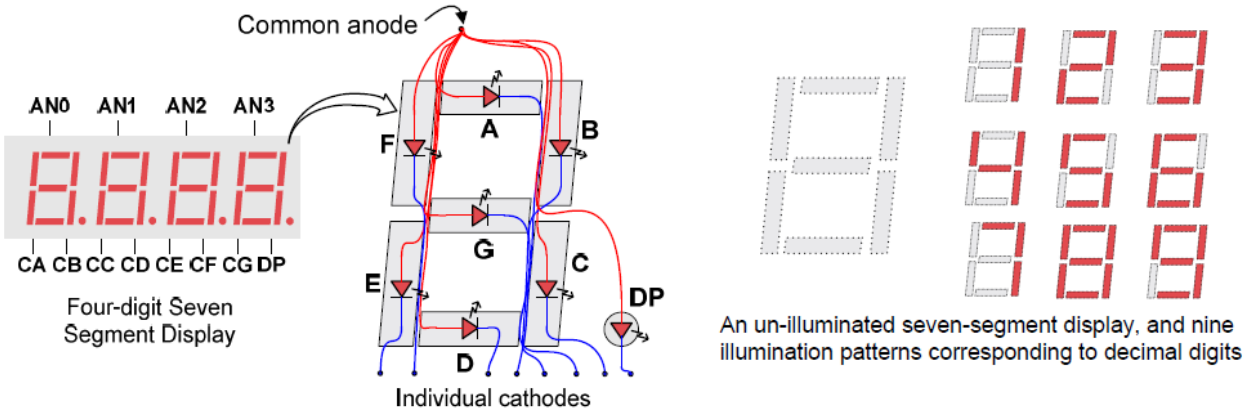


Рис.3.4 Схема соединения светодиодов индикатора

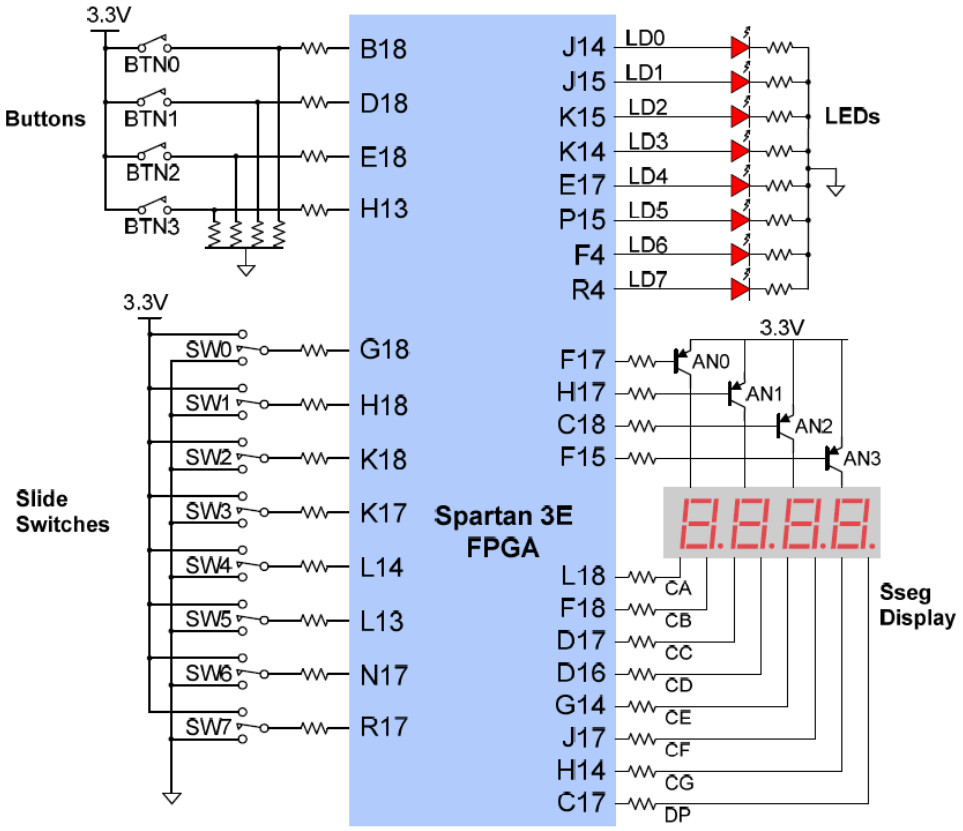


Рис.3.5 Схема соединения органов управления и индикации макета NEXYS-2 с ПЛИС

В прямоугольнике Spartan 3E FPGA указаны номера контактных площадок ПЛИС макета органов управления и индикации.

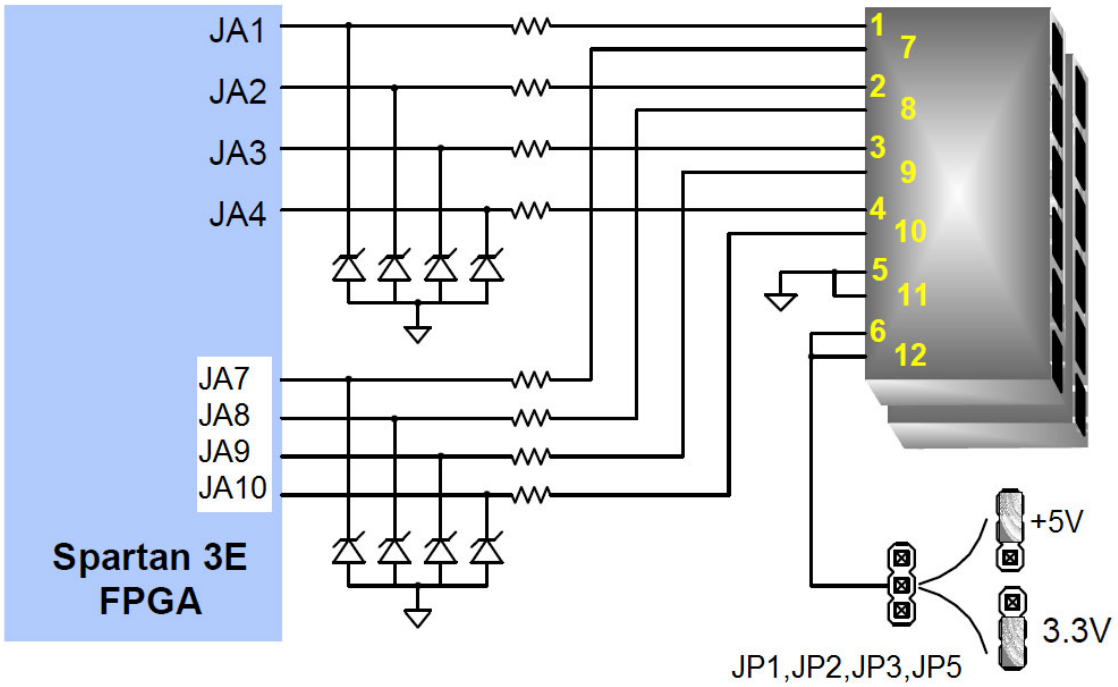


Рис.3.6 Варианты подключения источников +5V и +3.3V к выводам 6,12 разъемов JA,JB,JC,JD макета NEXYS-2

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]

Рис.3.7 Порты ввода вывода макета NEXYS 2

4. Задание к допуску

За выполнение «задания к допуску» можно получить 2 балла.

4.1. На рисунке 1.1 показана схема двухразрядного суммирующего счетчика CB2CE с входом clr асинхронного сброса и с входом *ce* разрешения счета, выполненная с использованием триггеров FTCE. Начертить в тетради схему модуля аналогичного четырехразрядного разрядного счетчика (CB4CE).

4.2. Используя символическое обозначение модуля двоичного 4-х разрядного счетчика CB4RE (рис.1.4), составить и начертить в тетради схему модуля декадного счетчика (CD4RE). Декадный счетчик должен переходить в ноль {Q3,Q2,Q1,Q0}={0,0,0,0} не из {Q3,Q2,Q1,Q0}={1,1,1,1}, а из {Q3,Q2,Q1,Q0}={1,0,0,1}.

4.3. Объяснить принцип работы всех счетчиков, описанных в разделе 1.

4.4. Объяснить принцип работы семи сегментного индикатора в лабораторном макете и назначение модулей **Gen1ms, Gen4an, D7seg, mux16_4**, описанных в разделе **Приложение**. Модуль дешифратора **D7seg** предназначен для отображения только декадных цифр (0,1,…,9), необходимо дополнить его индикацией HEX цифр: A, b, C, d, E, F.

4.5. Из составленных модулей и созданных к ним символов составить и начертить в тетради схему заданной последовательности счетчиков с отображением их состояния на семи сегментном индикаторе (SCH1 см.рис.4.1) с набором счетчиков для Вашего варианта задания (см. таблицу 3).

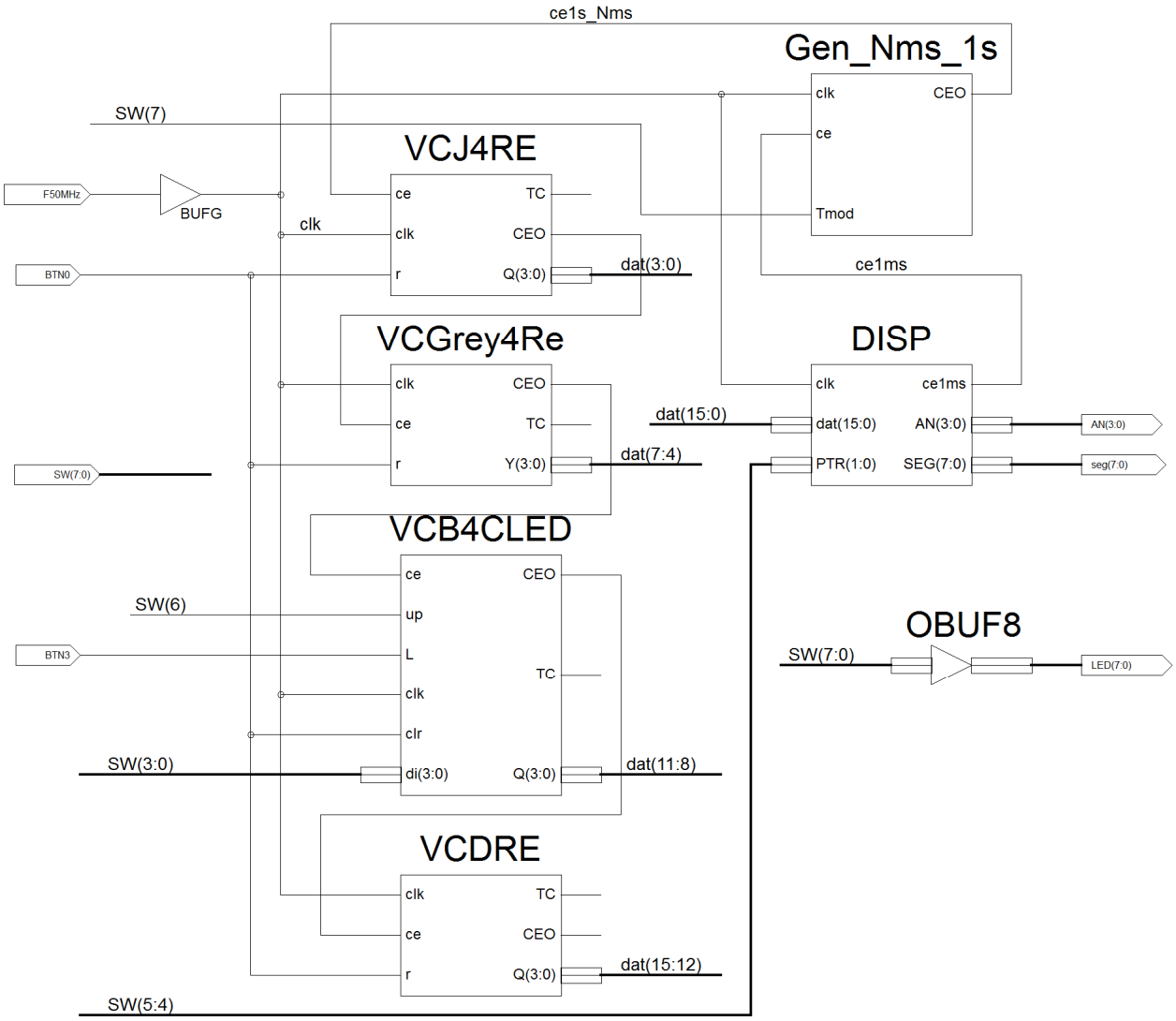


Рис.4.1 Пример схемы лабораторной работы Sch_lab401

В этой схеме счетчики соединены в соответствии 19-м вариантом задания (таблица 3).

Таблица 3

№	Последовательность соединения счетчиков	Множитель N
1	VCB4RE <-VCD4RE <-VCJ4RE<- VCB4CLED	15
2	VCBD4SE<-VCGrey4RE<- VCB4RE<-VCJ4RE	25
3	VCJ4RE <-VCB4RE<-VCBD4SE <- VCGrey4RE	14
4	VCJ4RE <-VCD4RE <-VCB4CLED<- VCBD4SE	18
5	VCD4RE <- VCBD4SE<- VCB4RE < VCB4CLED	20
6	VCJ4RE <-VCGrey4RE<- VCB4CLED<- VCD4RE	11

7	VCB4RE<-VCD4RE <- VCJ4RE<- VCB4CLED	41
8	VCB4CLED <-VCB4RE<-VCD4RE <- VCJ4RE	21
9	VCB4RE<-VCD4RE <- VCJ4RE<- VCB4CLED	16
10	VCJ4RE<- VCB4CLED<- VCB4RE<-VCD4RE	13
11	VCBD4SE <-VCJ4RE <-VCD4RE <-VCB4CLED	19
12	VCB4CLED <-VCD4RE <- VCBD4SE<- VCB4RE	11
13	VCD4RE<-VCGrey4RE<- VCB4CLED <-VCJ4RE	31
14	VCJ4RE <- VCB4CLED<- VCD4RE<-VCGrey4RE	17
15	VCB4CLED <-VCB4RE <-VCD4RE <-VCJ4RE	33
16	<-VCGrey4RE<-VCBD4SE <- VCB4RE<-VCJ4RE	55
17	VCB4CLED <-VCD4RE <- VCBD4SE<- VCB4RE	25
18	VCD4RE <- VCBD4SE<- VCB4RE < VCB4CLED	31
19	VCD4RE <- VCB4CLED <- VCGrey4RE <- VCJ4RE	27
20	VCB4RE<-VCD4RE <- VCJ4RE<- VCB4CLED	19

5. Задание к выполнению

За выполнение «задания к выполнению» можно получить 4 балла.

В папке FRTK создать папку со своим именем (только латинские символы). Далее в этой папке в системе ISE Design Suite 14.4 создать проект с именем Lab401N, для ПЛИС используемой в макете NEXYS2 (Spartan3E, XC3S500E, FG320, XST (VHDL/Verilog), Isim(VHDL/Verilog), Store all values).

В системе проектирования ISE14.4 все операции доступны только для модуля помещенного на вершину проекта (Set as Top Module).

В окне источников (Sources) создать (New Source) заданный модуль (Verilog Module). Сделать его главным в проекте (Set as Top Module). Ввести на Verilog-е текст схемы модуля. Проверить синтаксис веденного текста схемы (Check Syntax).

- Установить режим **Implementation.** В окне процессов (Processes) выполнить синтез каждого модуля (Synthesize XST). Исправить возможные ошибки, обратить внимание на предупреждения (Warnings). Проверить соответствует ли Вашим представлениям синтезированная логическая схема этого модуля (View RTL Schematic). Посмотреть технологическую (View Technology Schematic) схему синтезированного модуля. Из отчета о синтезе (View Synthesis Report) выписать число слайсов (Slices), триггеров (Flip Flops) и таблиц логических функций (LUTs) необходимых для реализации модуля.
- Создать схемотехнический символ модуля (Design Utilites/Create Schematic Symbol).
- Создать для этого модуля задание на моделирование (Verilog Test Fixture). В окне Simulation Behavioral Model/ISim Properties установить необходимое время моделирования. Остальные параметры можно не менять.

  Выделить модуль задания на моделирование. Провести моделирование работы модуля (Simulate Behavioral Model/RUN). Подкорректировать, если необходимо временные диаграммы входных сигналов (Verilog Test Fixture). Получить содержательные временные диаграммы.

5.1. Создать модуль и символ двоичного суммирующего счетчика (**VCB4RE**). Провести моделирование его работы. Начертить в тетради эскизы содержательных фрагментов полученных временных диаграмм.

5.2. Создать модуль и символ декадного счетчика (**VCD4RE**). Провести моделирование его работы. Начертить в тетради эскизы содержательных фрагментов полученных временных диаграмм.

5.3. Создать модуль и символ двоичного вычитающего счетчика (**VCBDmSE**). Провести моделирование его работы. Начертить в тетради эскизы содержательных фрагментов полученных временных диаграмм.

5.4. Создать модуль и символ реверсивного счетчика (**VCBmCLED**). Провести моделирование его работы. Начертить в тетради эскизы содержательных фрагментов полученных временных диаграмм.

5.5. Создать модуль и символ счетчика Джонсона (**VCJ4RE**). Провести моделирование его работы. Начертить в тетради эскизы содержательных фрагментов полученных временных диаграмм.

5.6. Создать модуль и символ счетчика в коде Грея (**VCGrey4RE**). Провести моделирование его работы. Начертить в тетради эскизы содержательных фрагментов полученных временных диаграмм.

5.7. Создать модуль генератора последовательного включения цифр семи сегментного индикатора (**Gen4an**), мультиплексора 4-х битных цифр (**MUX16_4**), генератора сигналов разрешения счета (**Gen1ms**), генератора точки (**Gen_P**), дешифратора для семи сегментного индикатора (**D7seg**). Необходимо дополнить дешифратор индикацией HEX цифр: A, b, C, d, E, F. Создать модуль и символ **DISPLAY** индикатора. Verilog код данных модулей приведен в разделе **Приложение**.

5.8. Создать модуль и символ **Gen_Nms_1s** (см. в разделе **Приложение**) в соответствии с Вашим вариантом задания (N см. в таблице 3). Этот генератор должен давать на выходе периодическую последовательность импульсов с длительностью Tclk (20 ns) и с периодом 1s - при низком уровне сигнала на входе mod (Tmod=0), и с периодом N*1ms - при высоком уровне сигнала на входе mod (Tmod=1).

5.9. Из составленных модулей составить схему (Sch1_lab401) заданного варианта последовательности соединения счетчиков (свой вариант см. в Таблице 3) с отображением их состояния на семи сегментном индикаторе. Пример такой схемы приведен на рис. 4.1.

Для ISE14 создать (New Source) Implementation Constraints File и в созданный текстовый файл Sch1_lab401.ucf ввести список связей портов схемы с контактными площадками ПЛИС, пример которого для макета NEXIS 2 приведен в разделе **Приложение**. В этом текстовом файле символом # «закомментированы» неиспользуемые выводы ПЛИС.

5.10. Провести имплементацию, создать загрузочный файл конфигурации для данной схемы Sch1_lab401.bit (Generate Programming File) для загрузки в ПЛИС или *.mcs (Configure Target Device) для загрузки в ПЗУ (XCF04S). В папке проекта найти EXEL файл Sch1_lab401_pad.xls, проверить правильность использования выводов ПЛИС, например, на выводе B8 должен быть сигнал F50MHz.

Программой ADEPT загрузить полученный файл Sch1_lab401.mcs в ПЗУ конфигурации или Sch1_lab401.bit непосредственно в ПЛИС. После загрузки в ПЗУ

необходимо нажать кнопку RESET на макете. Если загрузка выполнена в ПЛИС, то нажимать RESET не нужно. Продемонстрировать работу каждого счетчика схемы. Проверить соответствие показаний индикатора каждого счетчика результатам моделирования.

6. Задание к сдаче работы

За выполнение «задания к сдаче» можно получить 4 балла (2 балл за каждый пункт).

6.1. Создать модуль подавления дребезга кнопки BTN2, обеспечивающий изменение состояния счетчиков на 1 от каждого нажатия кнопки (пример временных диаграмм модуля подавления дребезга кнопки см. п.8.9 в разделе **Приложение**).

Соединить в схеме рис.4.1 входы CE используемых счетчиков параллельно и соединить с выходом модуля подавления кнопки. Продемонстрировать работу устройства на макете.

6.2. Создать модуль измерения периода сигнала на выходе CEO генератора **Gen_Nms_1s** (пример временных диаграмм модуля измерения периода сигнала см. п.8.10 в разделе **Приложение**). Модуль должен обеспечить отображение на семи сегментном индикаторе макета период сигнала на выходе CEO генератора **Gen_Nms_1s** в десятичном виде. Отладить работу модуля и продемонстрировать на макете при двух значениях SW[7].

7. Контрольные вопросы

7.1. Сколько триггеров использовано в каждом модуле Вашей схемы?
7.2. Чему равна максимальная частота синхронизации каждого модуля и всей схемы?
7.3. Как зависит максимальная частота синхронизации от числа разрядов счетчика?
7.4. Как можно получить счётчик, считающий до X (X < N), имея счётчик, считающий до N = 2^m и имеющий вход синхронного сброса?
7.5. Как можно избежать ложных срабатываний в результате дребезга контактов кнопки или иного механического переключателя?
7.6. Объясните принцип действия 7-ми сегментного индикатора. Почему частота переключения индикаторов выбрана ниже, чем тактовая частота синхронизации макета?

8. Приложение

8.1 Генератор последовательного включения цифр семи сегментного индикатора.

```
module Gen4an (     input clk,      output reg [1:0] q = 0, //Счетчик номера анода
                    input ce,       output wire [3:0] an );
    assign   an = (q==0)? 4'b1110: //включение цифры 0 (младшей)
                  (q==1)? 4'b1101: //включение цифры 1
```

```
                    (q==2)? 4'b1011: //включение цифры 2
                                4'b0111; //включение цифры 3 (старшей)
        always @ (posedge clk) if (ce) begin
                q <= q+1;
        end
endmodule
```

8.2 Дешифратор декадных цифр для семи сегментного индикатора.

```
module D7seg(input [3:0] dig,          output wire [6:0] seg);
                        //gfedcba
        assign   seg  = (dig==0)? 7'b1000000: //0      a
                        (dig==1)? 7'b1111001: //1 f|       |b
                        (dig==2)? 7'b0100100: //2     g
                        (dig==3)? 7'b0110000: //3 e|       |c
                        (dig==4)? 7'b0011001: //4      d
                        (dig==5)? 7'b0010010: //5
                        (dig==6)? 7'b0000010: //6
                        (dig==7)? 7'b1111000: //7
                        (dig==8)? 7'b0000000: //8
                        (dig==9)? 7'b0010000: //9
                                  7'b1111111; //
endmodule
```

8.3 Мультиплексор 4-х битных цифр.

```
module MUX16_4 (  input [15:0] dat,        output wire [3:0] do,
                  input [1:0] adr);
        assign  do =    (adr==0)? dat[3:0]:
                        (adr==1)? dat[7:4]:
                        (adr==2)? dat[11:8]: dat[15:12];
endmodule
```

8.4 Генератор сигналов с периодом 1 мс.

```
module Gen1ms (input clk,   //Сигнал синхронизации
                                output wire ce1ms=0); //1 миллисекунда
        parameter Fclk =50000000; // Частота генератора синхронизации 50 МГц
        parameter F1kHz =1000; // Частота 1 кГц
        reg[15:0]cb_ms = 0; // Счетчик миллисекунды
        assign ce1ms = (cb_ms==1); //1 милисекунда
        //Делитель частоты
        always @(posedge clk) begin
                cb_ms <= ce1ms? ((Fclk/F1kHz)) : cb_ms-1; // Счет миллисекунд
        end
endmodule
```

8.5 Модуль генератора точки.

```
module Gen_P (input [1:0] ptr, output wire seg_P,
```

```
                input [1:0] adr_An );
        assign seg_P = !(ptr==adr_An);
endmodule
```

8.6 Модуль семи сегментного индикатора.

```
module DISPLAY(  input clk,          output wire [3:0] AN,
                 input [15:0] dat,   output wire [7:0] SEG,
                 input [1:0] PTR,    output wire ce1ms);
        wire [3:0] Dig;
        wire [1:0] Adr_dig;
        //Генератор "анодов"
        Gen4an DD1( .clk(clk), .q(Adr_dig), .ce(ce1ms), .an(AN));
        // Мультиплексор цифр
        MUX16_4 DD2 (.dat(dat), .do(Dig), .adr(Adr_dig));
        // Дешифратор семи сегментных символов цифр
        D7seg DD3 (.dig(Dig), .seg(SEG[6:0]));
        // Генератор точки
        Gen_P DD4 (.adr_An(Adr_dig), .seg_P(SEG[7]), .ptr(PTR));
        // Генератор ce1ms
        Gen1ms DD5 (.clk(clk), .ce1ms(ce1ms));
endmodule
```

8.7 Схема модуля генератора Gen_Nms_1s.

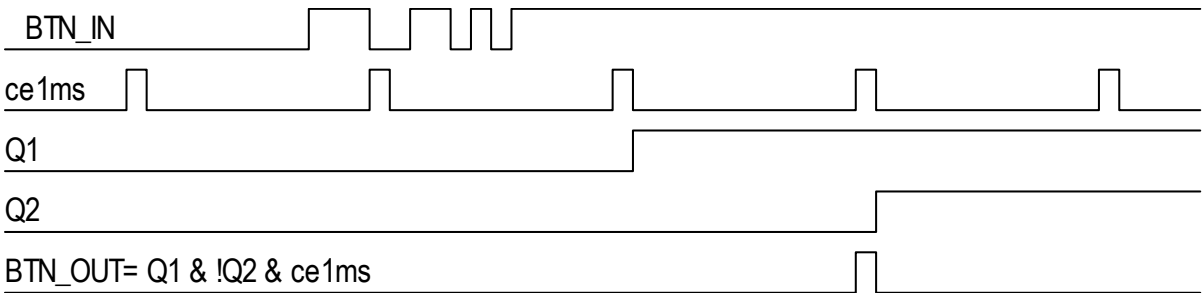
```
`define N 27 //Заданный вариант N из таблицы 3
module Gen_Nms_1s (input clk,    output wire CEO,
                   input ce,     //ce1ms
                   input Tmod);
        parameter F1kHz=1000;    //Частота 1 кГц
        parameter F1Hz=1;        //Частота 1 Гц

        reg[9:0]cb_Nms = 0;  //Счетчик N миллисекунд
        wire[9:0]Nms = Tmod? `N-1 : ((F1kHz/F1Hz)-1);  //Число для делителя
        частоты
        assign CEO =  ce & (cb_Nms==0);  //1 секунда или Nms
        always @(posedge clk)  if (ce) begin
                cb_Nms  <= (cb_Nms==0)? Nms :  cb_Nms-1; //Счет N миллисекунд
        end
endmodule
```

8.8 Связь портов схемы с контактными площадками ПЛИС (файл *.ucf).

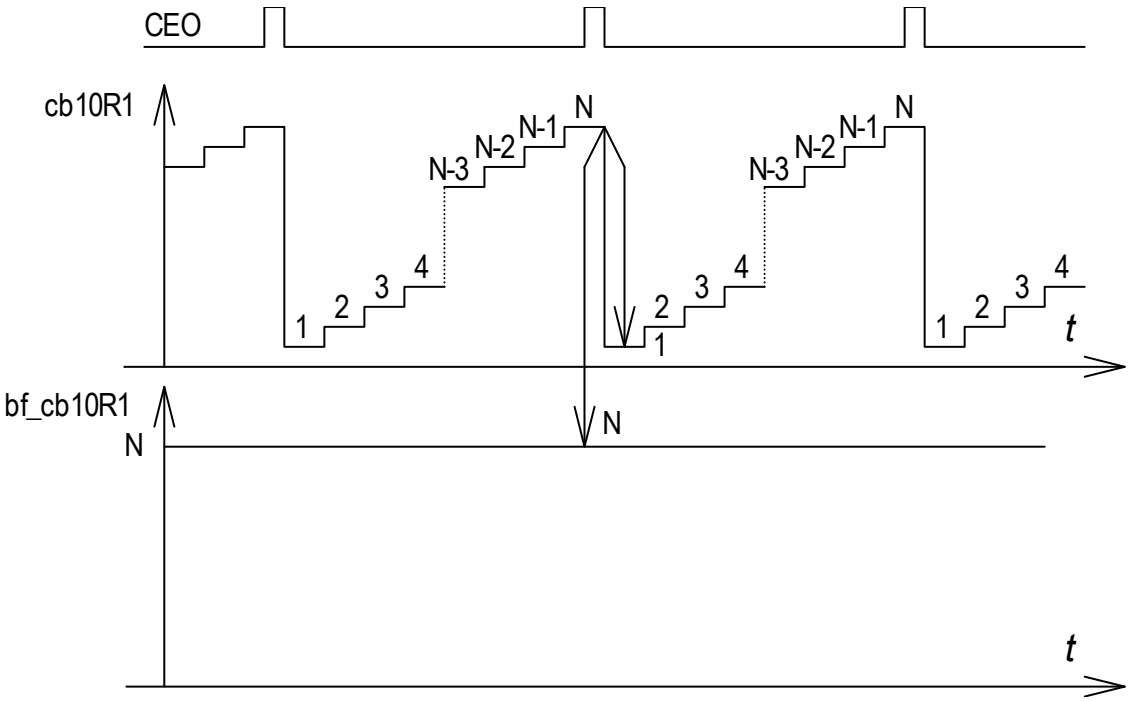
```
NET "AN<0>"  LOC = "F17"  ; #AN0
NET "AN<1>"  LOC = "H17"  ; #AN1
NET "AN<2>"  LOC = "C18"  ; #AN2
NET "AN<3>"  LOC = "F15"  ; #AN3

NET "BTN0"  LOC = "B18"  ; #BTN3
```

```
#NET "BTN1"  LOC = "D18"  ; #BTN2
#NET "BTN2"  LOC = "E18"  ; #BTN1
NET "BTN3"  LOC = "H13"  ; #BTN0

NET "F50MHz"  LOC = "B8"  ; #F50MHz

NET "seg<0>"  LOC = "L18"  ; #CA
NET "seg<1>"  LOC = "F18"  ; #CB
NET "seg<2>"  LOC = "D17"  ; #CC
NET "seg<3>"  LOC = "D16"  ; #CD
NET "seg<4>"  LOC = "G14"  ; #CE
NET "seg<5>"  LOC = "J17"  ; #CF
NET "seg<6>"  LOC = "H14"  ; #CG
NET "seg<7>"   LOC = "C17"  ; #CP

NET "SW<0>"  LOC = "G18"  ; #SWT0
NET "SW<1>"  LOC = "H18"  ; #SWT1
NET "SW<2>"  LOC = "K18"  ; #SWT2
NET "SW<3>"  LOC = "K17"  ; #SWT3
NET "SW<4>"  LOC = "L14"  ; #SWT4
NET "SW<5>"  LOC = "L13"  ; #SWT5
NET "SW<6>"  LOC = "N17"  ; #SWT6
NET "SW<7>"  LOC = "R17"  ; #SWT7

NET "LED<0>"  LOC = "J14"  ; #LD0
NET "LED<1>"  LOC = "J15"  ; #LD1
NET "LED<2>"  LOC = "K15"  ; #LD2
NET "LED<3>"  LOC = "K14"  ; #LD3
NET "LED<4>"  LOC = "E17"  ; #LD4
NET "LED<5>"  LOC = "P15"  ; #LD5
NET "LED<6>"  LOC = "F4"  ; #LD6
NET "LED<7>"  LOC = "R4"  ; #LD7
```

8.9 Пример временных диаграмм модуля подавления дребезга кнопки.



8.10 Пример временных диаграмм измерения периода сигнала CEO модуля Gen_Nms_1s.



В этом примере cb10R1[9:0] это счетчик (со сбросом не в 0, а в 1). Bf_cb_10R1[9:0] это регистр (буфер результата), в который записывается значение счетчика в момент сброса и далее выводится на дисплей. По сигналу CEO модуля Gen_Nms_1s текущее состояние счетчика N переписывается в буфер результата, а счетчик сбрасывается в 1.