Лабораторная работа №401N

Счетчики импульсов на ПЛИС

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.

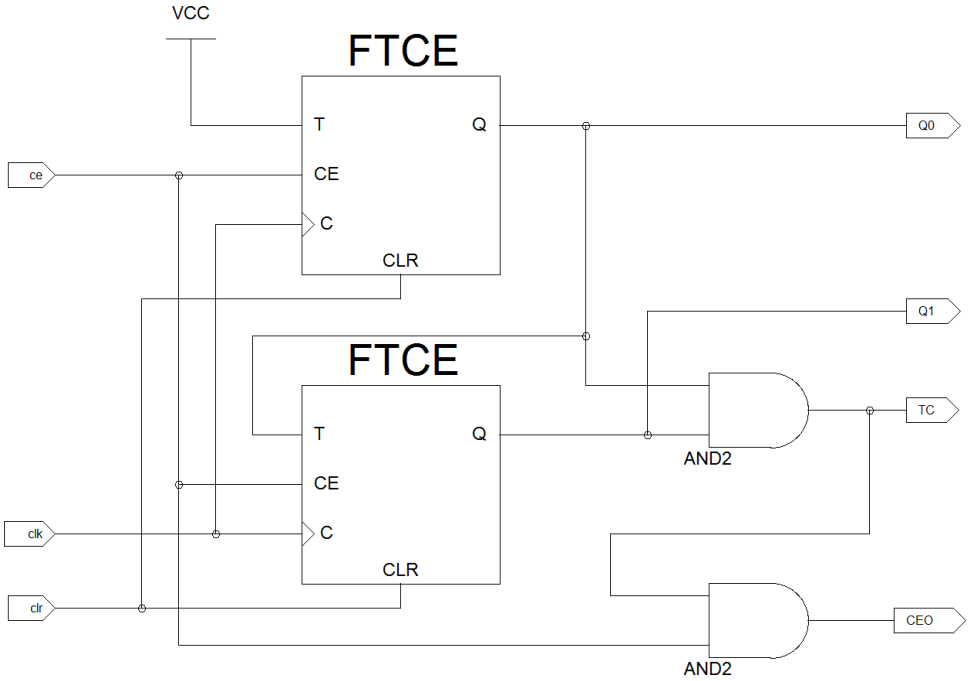

Рис.1.1 Схема двухразрядного суммирующего счетчика CB2CE с входом clr асинхронного сброса и с входом *ce* разрешения счета

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 фронту clk (Q<=Q).

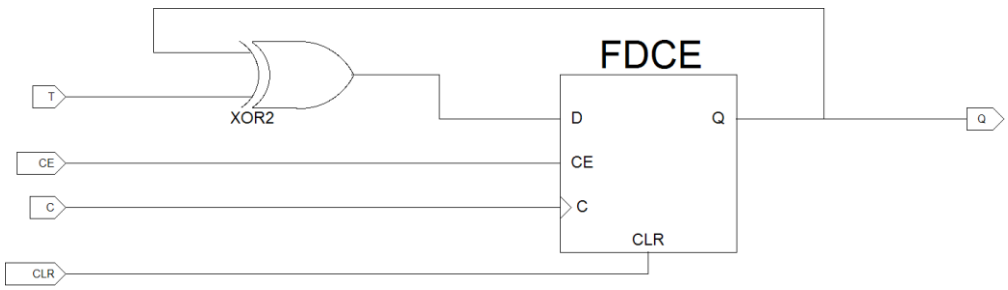

Рис.1.2 Схема FTCE триггера

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.

V1.1 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.

Для синтезатора, реализующего схему из компонент ПЛИС, имя асинхронного входа сброса в 0 может быть любым. Важно чтобы оно было вписано через **or** в перечислении фронтов (posedge clk **or** posedge clr) и было бы первым в последовательности условий (Q1<=clr? 0 : ...).

1.2 Двоичный суммирующий счетчик с синхронным входом сброса

V1.2 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

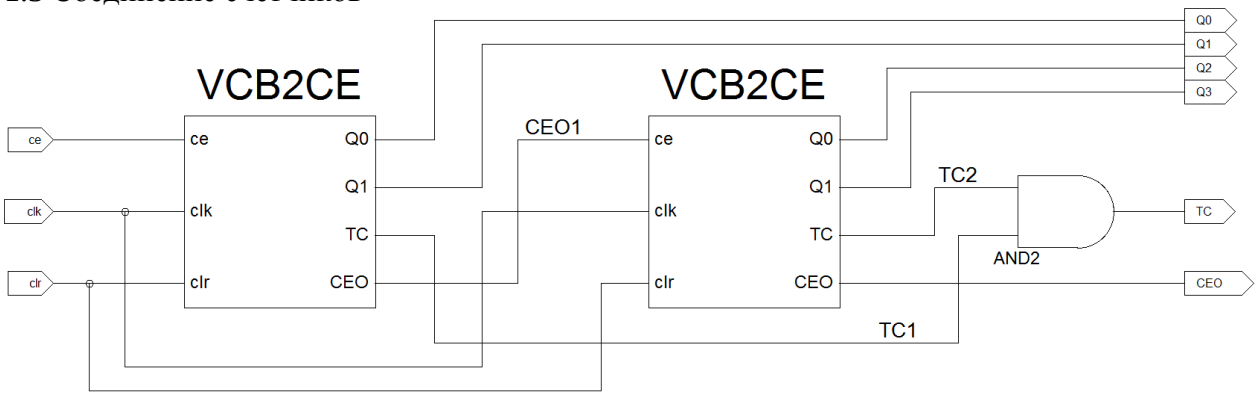

Рис.1.3 Схема четырехразрядного суммирующего счетчика SCB4CE состоящего из двух последовательно соединенных двухразрядных суммирующих счетчиков VCB2CE

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).

V1.3 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, выше описанных 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.

V1.4 Схема суммирующего *m*-разрядного счетчика с асинхронным сбросом в ноль и с входом разрешения счета написанная на языке VERILOG

```
`define m 3
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, реализующего схему из компонент ПЛИС, «знает», что для этой функции можно использовать схему счетчика.

1.5 Логическое моделирование

Для проведения логического моделирования необходимо создать модуль задания Verilog Test Fixture, например, для VCBmCE 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 80 нс аналогичным образом:

```
parameter Tce=80;
always begin ce=1; #(Tclk/4); ce=0; #(3*Tclk/4); end
```

1.5.1 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=0; #(Tclk/2); clk=1; #(Tclk/2); end

// Генератор периодического сигнала ce
parameter Tce=80; //Период сигнала ce 80 нс
always begin ce=0; #(3*Tce/4); ce=1; #(1*Tce/4); end

initial begin           // Initialize Inputs
                clr = 0; //Исходное состояние входов
#92;            clr = 1; //Через 92 нс 1
#5;             clr = 0; // Через 5 нс  0
#111;           clr = 1; //Через 111 нс 1
#70;            clr = 0;  // Через 70 нс 0
end
endmodule
```

В этом модуле задания на моделирование сигнал clr через 92 нс после начала на 5 нс, а затем еще раз через 111 нс на 70 нс устанавливается равным 1.

Сигналы clk и ce задаются генераторами периодических сигналов.

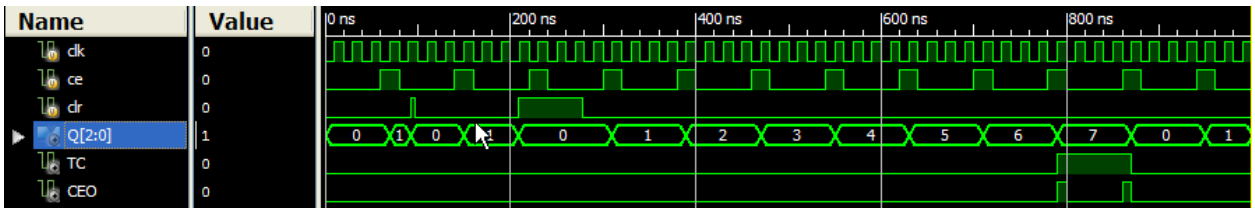

Рис.1.4 Пример временных диаграмм логического моделирования модуля VCBmCE (m=3)

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.

1.6 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.

V1.6 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 [`m-1:0] Q=0 );
assign TC = (Q==9) ;
```

```
assign CEO = ce & TC ;
always @ (posedge clk) begin
Q <= (r | CEO)? 0 : ce? Q+1 : Q ;
end
endmodule
```

1.7 Вычитающий счетчик

V1.7 Схема вычитающего *m*-разрядного счетчика с синхронной установкой в 2^{m}-1 и с входом разрешения счета написанная на языке VERILOG.

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

```
`define m 4
module VCBDmRE (input ce,          output reg [`m-1:0] Q = 0,
                input clk,         output wire TC,
                input s,       output wire CEO);
assign TC = (Q==0) ;           //Q0,Q1,...Q'm-1 ==0
assign TC = 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
```

1.8 Реверсивный счетчик

Схема реверсивного 2-х разрядного счетчика приведена на 1.4.

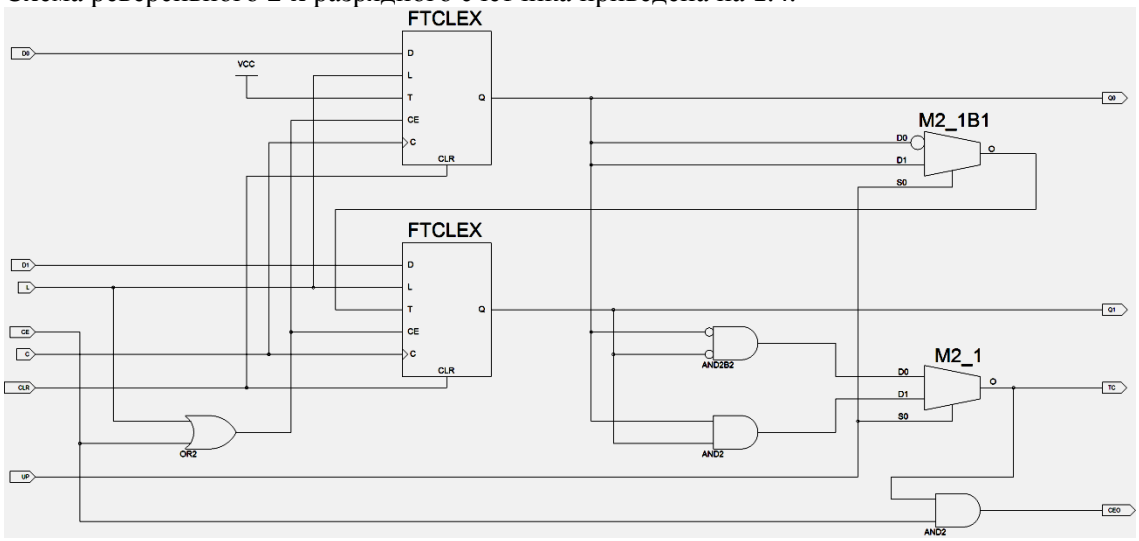

Рис.1.5 Схема библиотечного модуля двухразрядного реверсивного счетчика с входом *CLR* (асинхронного сброса в 0), входом *L* (разрешения синхронной загрузки), входом *UP* (направления счета) и входом *CE* (разрешения счета)

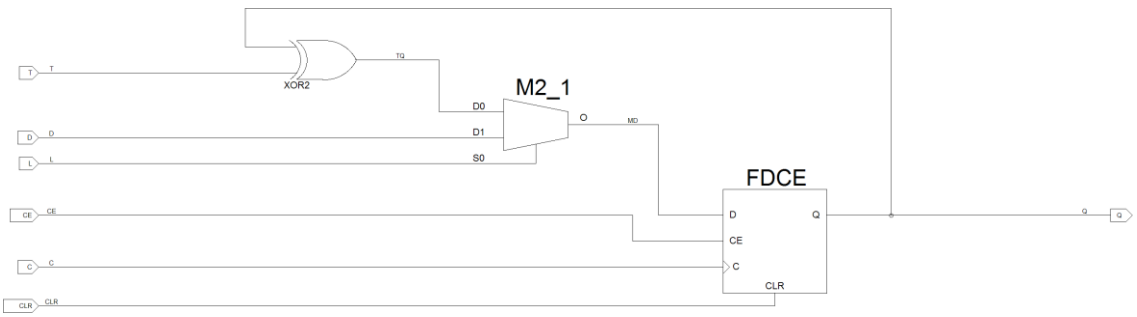

Рис.1.6 Схема загружаемого Т-триггера FTCLEX используемого в реверсивном счетчике

V1.8 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 CE),
                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
```

1.9 Счетчик Джонсона

Схема 4-х разрядного счетчика Джонсона приведена на рис.1.7. 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);

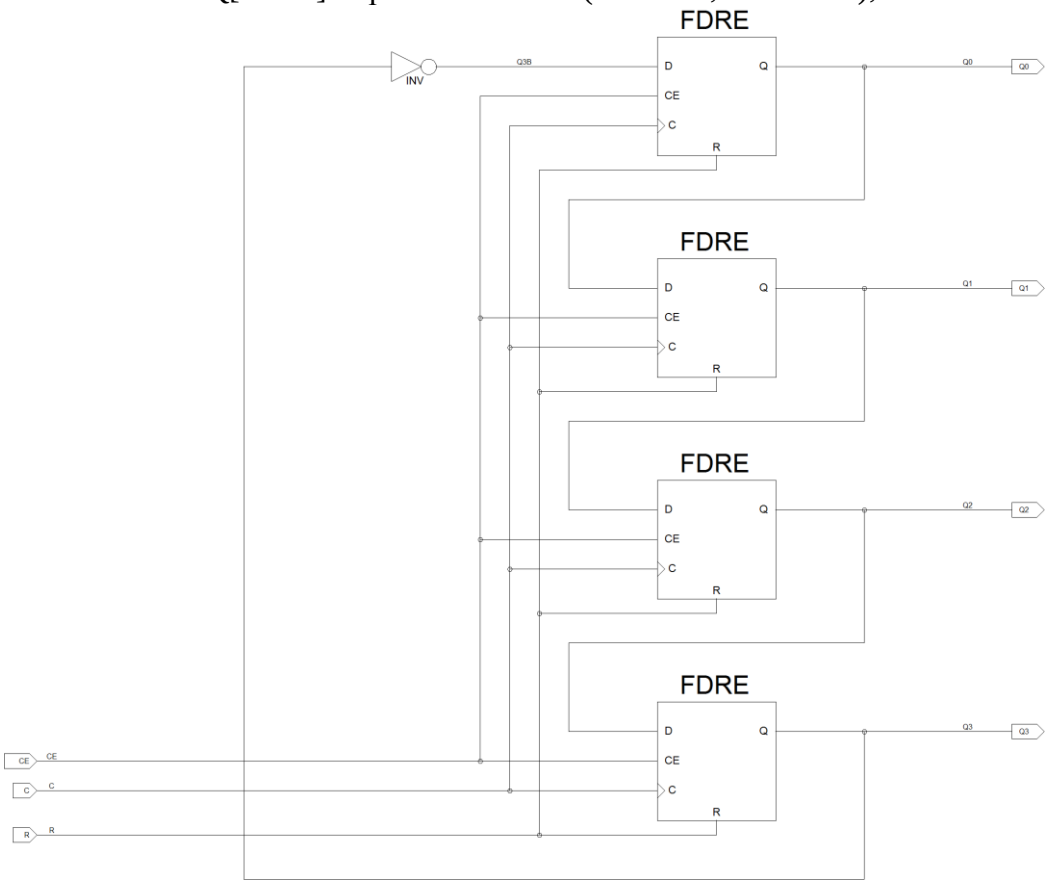

Рис.1.7 Схема библиотечного модуля 4-х разрядного счетчика Джонсона

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.

V1.9 Схема *m*-разрядного счетчика Джонсона на VERILOG-е

```
`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
```

1.10 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:

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

Особенностью кода Грея является то, что в отличие от двоичного кода, где соседние числа могут отличаться во 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 и даже всех 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]).

V1.10 Схема 4-х разрядного счетчика в коде Грея на VERILOG-е

```
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. Описание макета

Лабораторная работа выполняется на макете NEXYS 2. Структурная схема макета NEXYS 2 приведена на рис.2.1. 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.2.2)

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

- 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.

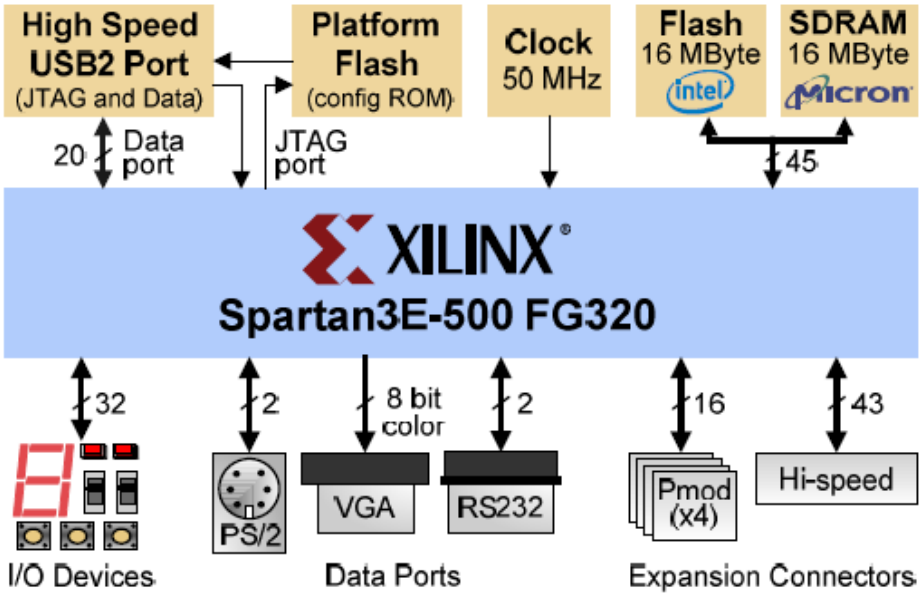

Рис.2.1 Структурная схема макета NEXYS 2

Рис.2.2 Устройства (I/O Devices) отображения и ввода информации макета NEXYS 2

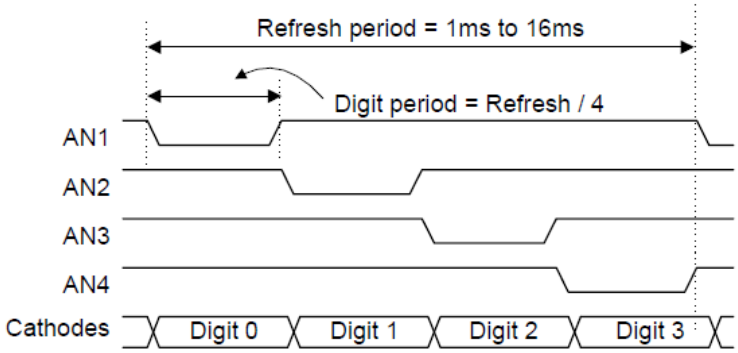

Рис.2.3 Временные диаграммы поочередного включения цифр индикатора

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

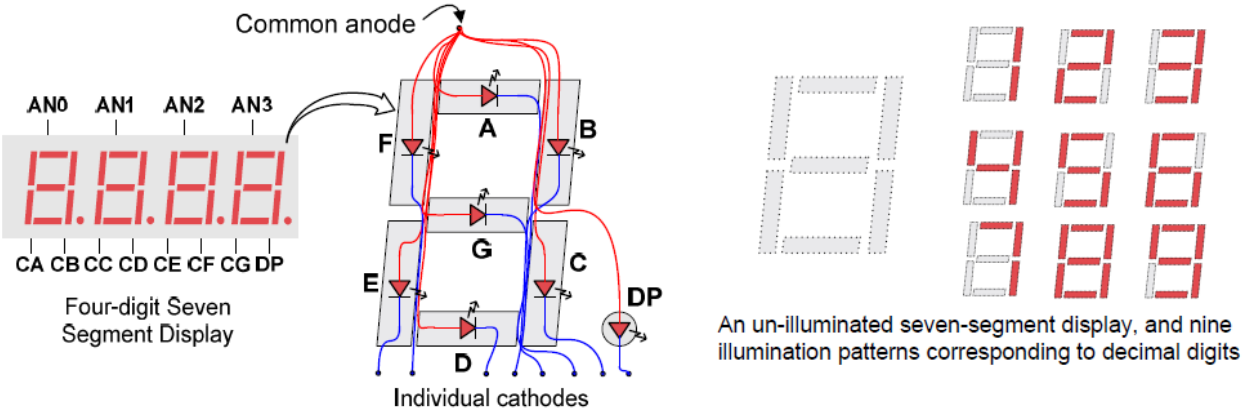

Рис.2.4 Схема соединения светодиодов индикатора

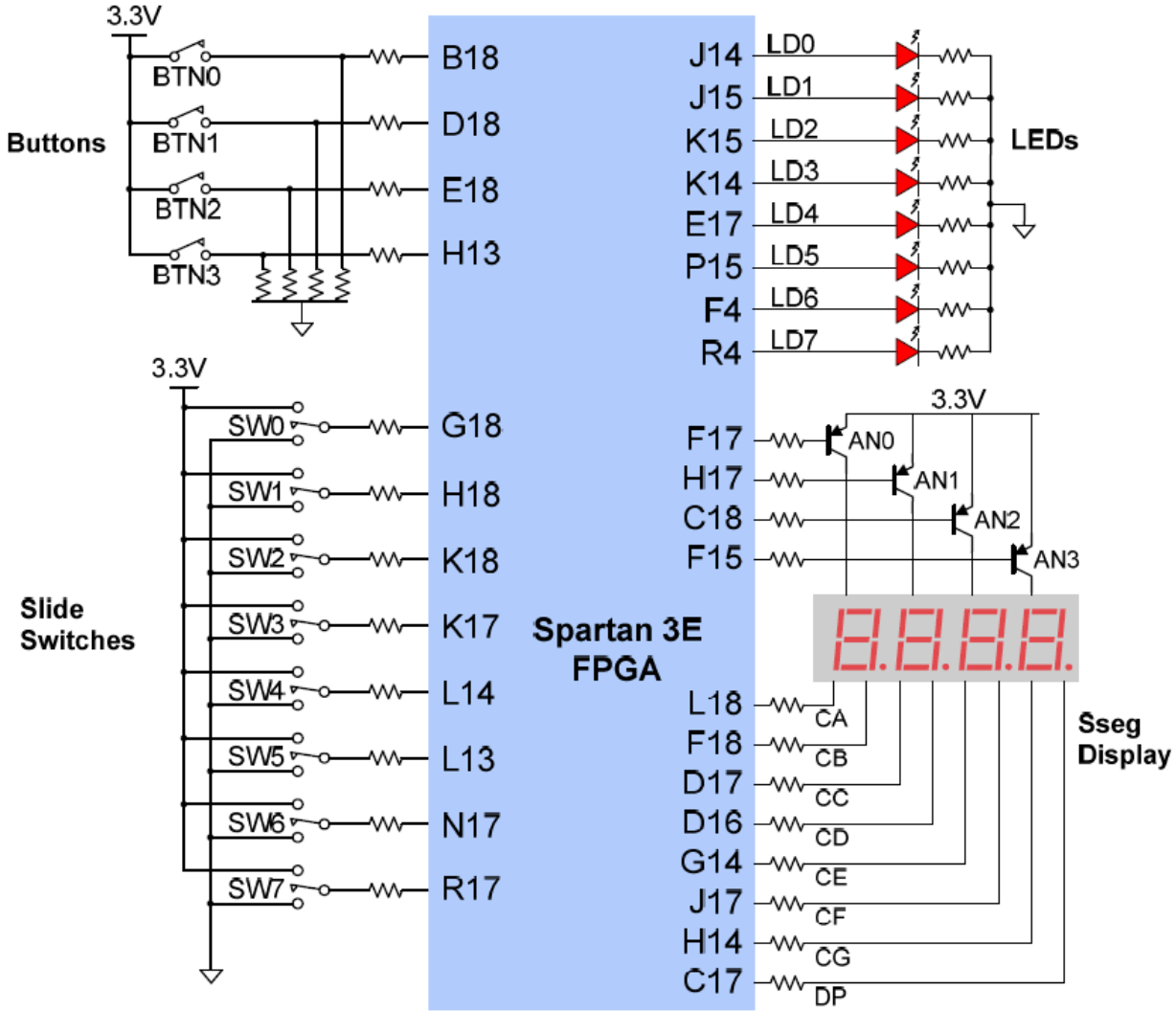

Рис.2.5 Схема соединения органов управления и индикации макета NEXYS-2 с ПЛИС

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

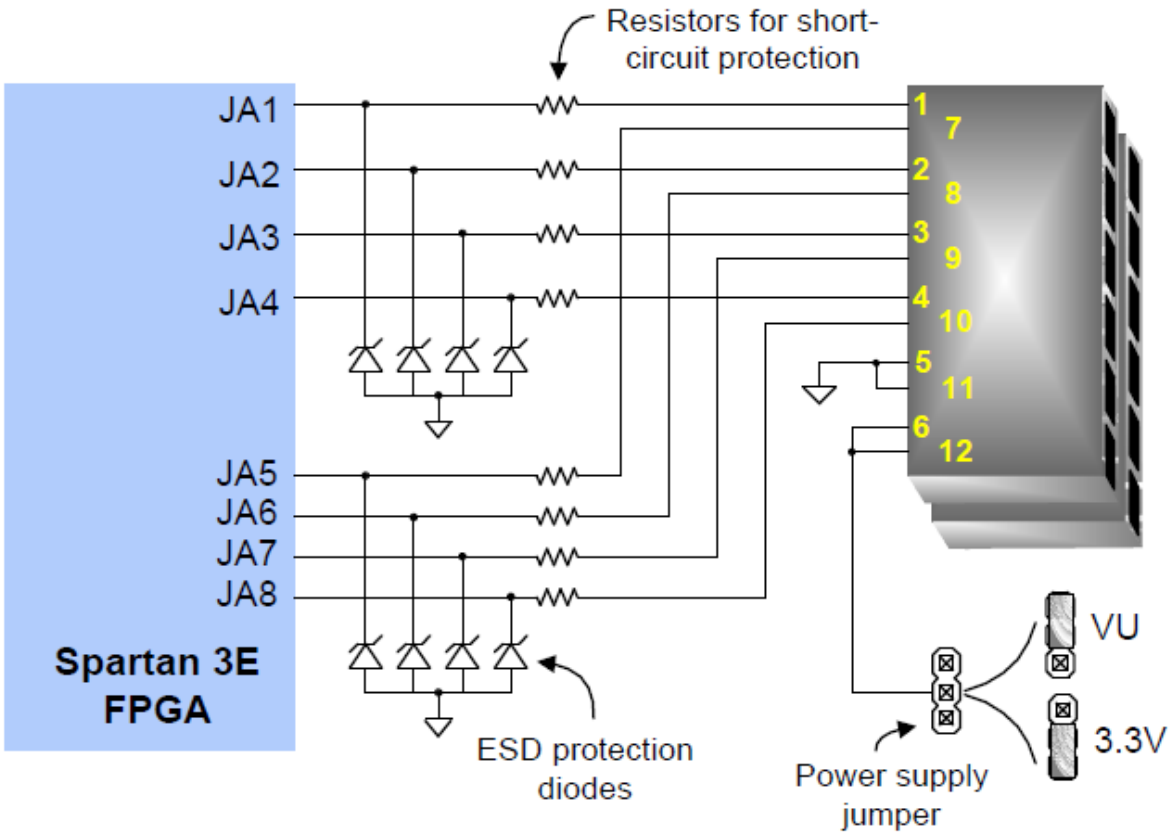

Table 3: Nexys2 Pmod Connector Pin Assignments							
Pmod JA		**Pmod JB**		**Pmod JC**		**Pmod JD**	
JA1: L15	JA7: K13	JB1: M13	JB7: P17	JC1: G15	JC7: H15	JD1: J13	JD7: K14[1]
JA2: K12	JA8: L16	JB2: R18	JB8: R16	JC2: J16	JC8: F14	JD2: M18	JD8: K15[2]
JA3: L17	JA9: M14	JB3: R15	JB9: T18	JC3: G13	JC9: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

Рис.2.6 Порты ввода вывода макета NEXYS 2

3. Задание к допуску

3.1. Начертить в тетради схему модуля двоичного 4-х разрядного счетчика (CB4CE) (см рис.1.1 и текст VCBmCE).
3.2. Используя символическое обозначение модуля двоичного 4-х разрядного счетчика CB4RE, составить и начертить в тетради схему модуля декадного счетчика (CD4RE).
3.3. Написать на VERILOG-е схему вычитающего счетчика (VCBmSED).
3.4. Написать на VERILOG-е схему реверсивного счетчика (VCBmCLED).
3.5. Написать на VERILOG-е схему счетчика Джонсона (VCJmRE).
3.6. Написать на VERILOG-е схему счетчика в коде Грея (VCGrey4RE).

3.7. Написать на VERILOG-е схему генератора последовательного включения цифр семи сегментного индикатора (g4an).
3.8. Написать на VERILOG-е схему дешифратора для семи сегментного индикатора (D7seg) для всех 16 состояний четырех бит входов (0,1,2,..9,A,b,C,d,E,F). Модуль 1.12 предназначен для отображения только декадных цифр (0,1,…,9).
3.9. Написать на VERILOG-е схему мультиплексора 4-х битных цифр (mux16_4).
3.10. Написать на VERILOG-е схему генератора сигналов разрешения счета (Gen1ms).

3.11. Из составленных модулей составить и начертить в тетради схему заданной последовательности счетчиков с отображением их состояния на семи сегментном индикаторе (SCH1).

4. Задание к выполнению

Создать проект с именем Lab401N, для ПЛИС используемой в макете NEXYS2 (Spartan3E, XC3S500E, FG320, XST (VHDL/Verilog), Isim(VHDL/Verilog), Store all values).

В старой системе проектирования ISE7.1i все операции доступны для любого выделенного модуля.

В системе проектирования ISE14 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). Для ISE14 сделать его главным в проекте (Set as Top Module). Ввести на Verilog-е текст схемы модуля. Проверить синтаксис веденного текста схемы (Check Syntax).

- Для ISE14.4 установить режим **Implementation.** В окне процессов (Processes) выполнить синтез каждого модуля (Synthesize XST) . Исправить возможные ошибки, обратить внимание на предупреждения. 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 (Test Bench Waveform (только для ISE7) или Verilog Test Fixture). Для ISE14 предварительно установить режим **Behavioral Simulation.** В окне Initial Length of Test Bench установить необходимое время моделирования. Для ISE7.1i и Test Bench Waveform в окошке GSR(FPGA) поставить «галку». Остальные параметры можно не менять.

  На созданных после нажатия кнопки “OK” временных диаграммах Test Bench Waveform установить необходимые уровни и характерные временные диаграммы входных сигналов. Сохранить модуль задания.
- Для ISE14.4 в окне Properties Simulate Behavioral Model. Установить необходимое время моделирования.
- Выделить модуль задания на моделирование. Провести моделирование работы модуля (Xilinx ISE Simulator/ Simulate Behavioral Model/RUN). Подкорректировать, если необходимо временные диаграммы входных сигналов. Получить содержательные временные диаграммы.

Если на ПК установлен Modelsim и в окне Properties проекта в строке Simulator выбран Modelsim, то моделирование надо проводить в Simulate Post Translate Verilog Model.

4.1. Создать модуль и символ двоичного суммирующего счетчика (VCB4RE). Провести моделирование его работы. Начертить в тетради эскиз полученных временных диаграмм.

4.2. Создать модуль и символ декадного счетчика (VCD4RE). Провести моделирование его работы. Начертить в тетради эскиз полученных временных диаграмм.

4.3. Создать модуль и символ двоичного вычитающего счетчика (VCB4SED). Провести моделирование его работы. Начертить в тетради эскиз полученных временных диаграмм.

4.4. Создать модуль и символ реверсивного счетчика. Провести моделирование его работы. Начертить в тетради эскиз полученных временных диаграмм.
4.5. Создать модуль и символ счетчика Джонсона. Провести моделирование его работы. Начертить в тетради эскиз полученных временных диаграмм.

4.6. Создать модуль и символ счетчика в коде Грея. Провести моделирование его работы. Начертить в тетради эскиз полученных временных диаграмм.

4.7. Создать модуль **Gen4an** генератора последовательного включения цифр семи сегментного индикатора.

4.8. Создать модуль **D7seg** дешифратора для семи сегментного индикатора. Дополнить схему индикацией чисел от 4`hA до 4`hF.

4.9. Создать модуль **MUX16_4** мультиплексора 4-х битных цифр.

4.10. Создать модуль генератора сигналов разрешения счета **Gen1ms**.

4.11 Создать модуль **Gen_P** генератора точки.

4.12 Создать модуль и символ **DISPLAY** индикатора. Провести моделирование его работы. Начертить в тетради эскиз полученных временных диаграмм.

4.13. Составить схему и символ модуля **Gen_Nms_1s** (см. рис.4.1). 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 (mod=0), и с периодом N*1ms - при высоком уровне сигнала на входе mod (mod=1) (N см. в таблице 3),

4.14. Из составленных модулей составить схему (SCH1) заданного варианта последовательности соединения счетчиков (см. Таблицу 3) с отображением их состояния на семи сегментном индикаторе. Пример такой схемы приведен на рис. 4.1.

Для ISE7.1i в редакторе (User Constrains) ввести номера используемых выводов ПЛИС (Приложение 7.7).

Для ISE14 создать (New Source) Implementation Constraints File и в созданный текстовый файл SCH1.ucf ввести список связей портов схемы с контактными площадками ПЛИС, пример которого для макета NEXIS 2 приведен в приложении 7.7. В этом текстовом файле символом # «закомментированы» неиспользуемые выводы ПЛИС.

4.14.1 Провести имплементацию, создать для загрузочный файл конфигурации данной схемы (SCH1.bit для ПЛИС или SCH1.mcs для ROM). Загрузить полученный файл. Продемонстрировать работу каждого счетчика схемы.

4.14.2 Проверить соответствие показаний индикатора каждого счетчика результатам моделирования.

Таблица 3

№	Последовательность соединения счетчиков	Множитель N
1	VCB4RE <-VCD4RE <-VCJ4RE<- VCB4CLED	15
2	VCBD4RE<-VCGrey4RE<- VCB4RE<-VCJ4RE	25
3	VCJ4RE <-VCB4RE<-VCBD4RE <- VCGrey4RE	14
4	VCJ4RE <-VCD4RE <-VCB4CLED<- VCBD4RE	18
5	VCD4RE <- VCBD4RE<- 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	VCBD4RE <-VCJ4RE <-VCD4RE <-VCB4CLED	19
12	VCB4CLED <-VCD4RE <- VCBD4RE<- VCB4RE	11
13	VCD4RE<-VCGrey4RE<- VCB4CLED <-VCJ4RE	31
14	VCJ4RE <- VCB4CLED<- VCD4RE<-VCGrey4RE	17
15	VCB4CLED <-VCB4RE <-VCD4RE <-VCJ4RE	33
16	<-VCGrey4RE<-VCBD4RE <- VCB4RE<-VCJ4RE	55

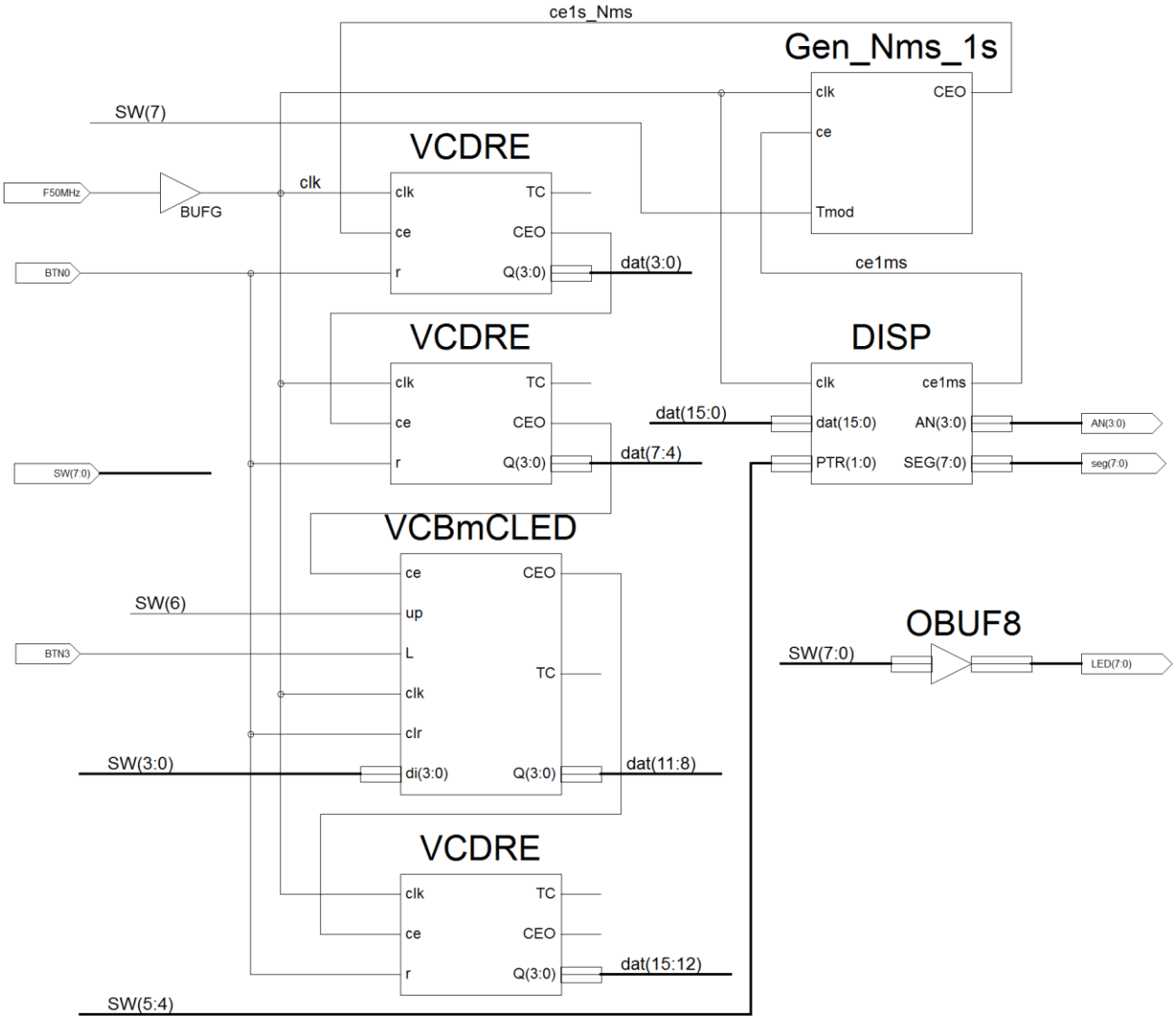

Рис.4.1 Пример схемы соединения счетчиков

5.Задание к сдаче работы

4.1. Создать модуль реверсивного декадного счетчика. Провести моделирование его работы. Продемонстрировать его работу на макете.

4.2. Соединить входы CE используемых счетчиков параллельно. Создать модуль, обеспечивающий изменение состояния счетчиков на 1 от каждого нажатия кнопки. Отладить его работу. Продемонстрировать работу устройства на макете.

4.3. Создать модуль измерения периода сигнала на выходе СЕО генератора **Gen_Nms_1s**. Отладить его работу. Продемонстрировать работу модуля на макете при двух значениях SW[7].

6.Контрольные вопросы

5.1. Сколько триггеров использовано в каждом модуле Вашей схемы?
5.2. Чему равна максимальная частота синхронизации каждого модуля и всей схемы?
5.3. Как зависит максимальная частота синхронизации от числа разрядов счетчика?
5.4. Составьте схему модуля преобразования кода Грея в двоичный код.
5.5. Какие устойчивые циклы 5-и разрядного счётчика Джонсона.
5.6. Как можно получить счётчик, считающий до X (X < N), имея счётчик, считающий до $N = 2^m$ и имеющий вход синхронного сброса?
5.7. Как можно избежать ложных срабатываний в результате дребезга контактов кнопки или иного механического переключателя?
5.8. Предложите схему измерения скважности периодически повторяющихся импульсов, если известно, что длительность импульсов много больше длительности такта синхронизации (скважностью следования импульсов называется отношения периода следования одной последовательности импульсов к их длительности, т.е. в данном случае отношение периода повторения импульса к его длительности).
5.9. Объясните принцип действия 7-ми сегментного индикатора. Почему частота переключения индикаторов выбрана ниже, чем тактовая частота синхронизации макета?

7. Приложения

7.1 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
```

7.2 Дешифратор для семи сегментного индикатора

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

```
endmodule
```

7.3 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
```

7.4 Генератор сигналов с периодом 1мс

```
module Gen1ms (input clk,   //Сигнал синхронизации
                            output wire ce_1ms); //1 милисекунда
parameter Fclk  =50000000 ; //Частота генератора синхронизации 50 МГц
parameter F1kHz =1000 ;     //Частота 1 кГц
reg[16:0]ct_ms = 0  ;  //Счетчик миллисекунд
assign ce_1ms = (ct_ms==0) ;//1 милисекунда
//Делитель частоты
always @(posedge clk) begin
ct_ms <= ce_1ms? ((Fclk/F1kHz)-1) : ct_ms-1 ;//Счет миллисекунд
end
endmodule
```

7.5 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
```

7.6 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
```

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

```
NET "AN<0>"  LOC = "F17"         ;
NET "AN<1>"  LOC = "H17"         ;
NET "AN<2>"  LOC = "C18"         ;
NET "AN<3>"  LOC = "F15"         ;

NET "BTN0"  LOC = "B18"          ;
#NET "BTN1"  LOC = "D18"         ;
#NET "BTN2"  LOC = "E18"         ;
NET "BTN3"  LOC = "H13"          ;

NET "F50MHz"  LOC = "B8"         ;#clk
NET "seg<0>"  LOC = "L18"        ;
NET "seg<1>"  LOC = "F18"        ;
NET "seg<2>"  LOC = "D17"        ;
NET "seg<3>"  LOC = "D16"        ;
NET "seg<4>"  LOC = "G14"        ;
NET "seg<5>"  LOC = "J17"        ;
NET "seg<6>"  LOC = "H14"        ;
NET "seg<7>"  LOC = "C17"        ; #DOT

NET "SW<0>"  LOC = "G18"         ;
NET "SW<1>"  LOC = "H18"         ;
NET "SW<2>"  LOC = "K18"         ;
NET "SW<3>"  LOC = "K17"         ;
NET "SW<4>"  LOC = "L14"         ;
NET "SW<5>"  LOC = "L13"         ;
NET "SW<6>"  LOC = "N17"         ;
NET "SW<7>"  LOC = "R17"         ;

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

#NET "TXD"  LOC = "P9"  ;
#NET "RXD"  LOC = "U6"  ;

#NET "JA1"  LOC = "L15"  ;#Pin1
#NET "JA2"  LOC = "K12"  ;#Pin2
#NET "JA3"  LOC = "L17"  ;#Pin3
#NET "JA4"  LOC = "M15"  ;#Pin4
#NET "JA7"   LOC = "K13"  ;#Pin7
#NET "JA8"   LOC = "L16"  ;#Pin8
#NET "JA9"  LOC = "M14"  ;#Pin9
#NET "JA10" LOC = "M16"  ;#Pin10

#NET "JB1"  LOC = "M13"  ;#Pin1
#NET "JB2"  LOC = "R18"  ;#Pin2
```

```
#NET "JB3"  LOC = "R15"  ;#Pin3
#NET "JB4"  LOC = "T17"  ;#Pin4
#NET "JB7"  LOC = "P17"  ;#Pin7
#NET "JB8"   LOC = "R16"  ;#Pin8
#NET "JB9"  LOC = "T18"  ;#Pin9
#NET "JB10" LOC = "U18"  ;#Pin10

#NET "JC1"  LOC = "G15"  ;#Pin1
#NET "JC2"  LOC = "J16"  ;#Pin2
#NET "JC3"  LOC = "G13"  ;#Pin3
#NET "JC4"  LOC = "H16"  ;#Pin4
#NET "JC7"   LOC = "H15"  ;#Pin7
#NET "JC8"   LOC = "F14"  ;#Pin8
#NET "JC9"  LOC = "G16"  ;#Pin9
#NET "JC10" LOC = "J12"  ;#Pin10

#NET "JD1"  LOC = "J13"  ;#Pin1
#NET "JD2"  LOC = "M18"  ;#Pin2
#NET "JD3"  LOC = "N18"  ;#Pin3
#NET "JD4"  LOC = "P18"  ; #Pin4
#NET "JD7"   LOC = "K14"  ;#LD3
#NET "JD8"   LOC = "K15"  ;#LD3
#NET "JD9"  LOC = "J15"  ;#LD3
#NET "JD10" LOC = "J14"  ;#LD3
```