Лабораторная работа 403N

Измерители частоты

Средняя частота периодического сигнала - это отношение разности фаз сигнала к длительности интервала между точками отсчета фазы. Технически просто оценивать разность фаз по целому числу периодов, тогда частота - это отношение числа периодов к длительности этого числа периодов. Вычисление отношения требует арифметической операции деления, реализация которой до недавнего времени была весьма затруднительна, поэтому в старых частотомерах время измерения (счета) устанавливается декадно-кратным секунде (1 мс, 10 мс, 100 мс, 1 с, 10 с). В этом случае при получении оценки средней частоты $\frac{X}{T_{MN}}$ деление на номинальное время измерения T_{MN} заменяется на приписывание номинальной размерности частоты (1 кГц,…0.1 Гц) числу ***X*** периодов. Интервал измерения в общем случае не равен длительности целого числа ***X*** периодов, а начало интервала измерения не зависит от измеряемого сигнала, поэтому при оценке частоты возникает погрешность дискретности отсчета, относительная величина которой (+/- 1/***X***), становится недопустимо большой при низких частотах.

Для низкочастотных сигналов в старых частотомерах используется режим измерения среднего из ***M*** периодов (***M*** = 1, 10, 100, 1000, 10000). При измерении среднего из ***M*** периодов формируется интервал с длительностью ***M*** измеряемых периодов и подсчитывается на этом интервале число ***Y*** импульсов генератора с эталонным периодом T_{CE} (10 нс,100 нс, 1 мкс, …). И в этом случае при получении оценки среднего периода умножение числа ***Y*** на номинальное значение эталонного периода и деление на ***M*** также сводится к приписыванию ему соответствующей номинальной размерности ($\frac{T_{CEN}}{M}$) . Например, при T_{CEN}=1 мкс и M=1000 единица числа Y соответствует 1 нс.

В обоих режимах в частотомере используется два счетчика, первый счетчик считает число импульсов X входного сигнала на интервале измерения, а второй счетчик формирует интервал измерения с длительностью $M \cdot T_{CE}$ или измеряет Y .

1. Измерение частоты по целому числу периодов (прямоугольная весовая функция)

В современных частотомерах используется метод оценки частоты по целому числу периодов. В этом методе интервал измерения задаётся произвольным генератором импульса необходимой длительности Tm, который синхронизируется при помощи *D*-триггера с измеряемым сигналом и далее первый счетчик cb_X подсчитывает число ***X*** периодов измеряемого сигнала в синхронном интервале Q_Tm, а второй счетчик cb_Y его длительность, т.е. число ***Y*** эталонных интервалов T_{CE}.

Оценка частоты $\hat{F}_X = F_{CEN} \cdot \frac{X}{Y}$, где $F_{CEN} = \frac{1}{T_{CEN}}$ – номинальная частота эталона частоты.

Относительная погрешность δ_D дискретности отсчета в этом случае определяется числом ***Y*** эталонных интервалов в синхронном интервале $X \cdot T_X$ и не зависит от числа ***X*** периодов измеряемого сигнала ($\delta_D \approx \pm\frac{1}{Y}$).

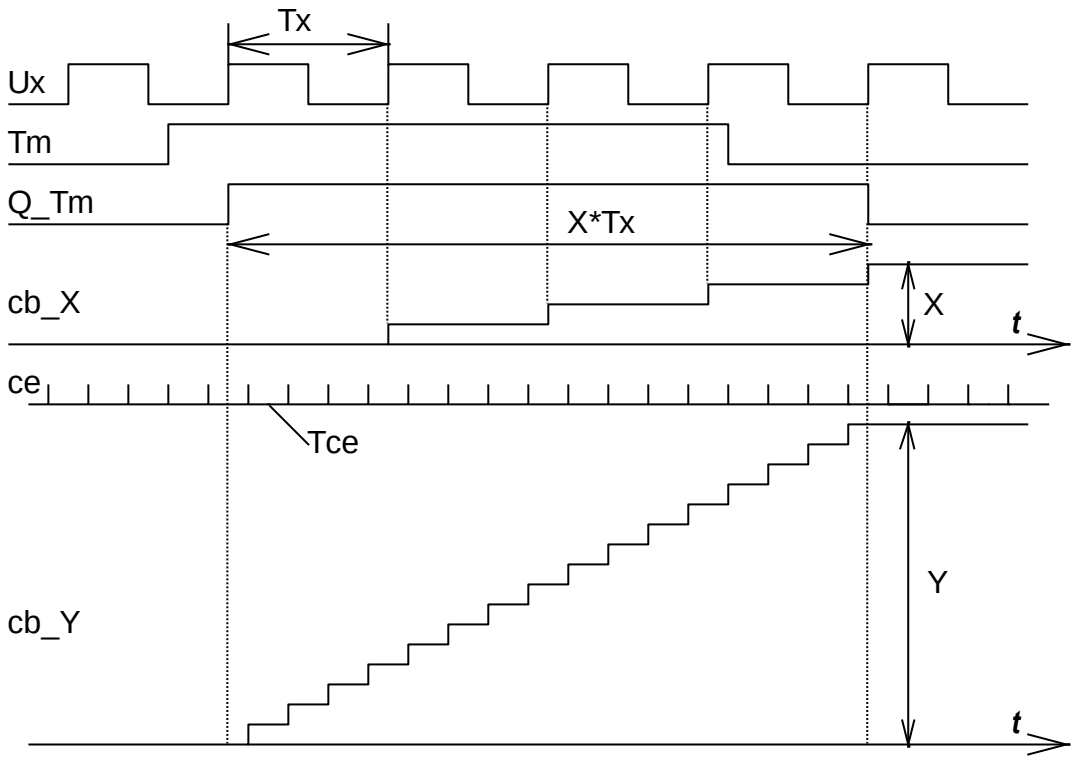


Рис.1 Временные диаграммы измерителя частоты по целому числу периодов

1.1.1 Пример схемы модуля **Fmes_XY** измерения частоты по целому числу периодов

```
`include "CONST_XY.v"
module Fmes_PR_WF(
        input ce,       output wire[`m_M-1:0] Q,    //Целая часть частного
        input Ux,       output wire[`m_S-1:0] F,    //Дробная часть частного
        input clk,      output wire Tm,             //Интервал измерения
        input st,       output reg [`m_X-1:0]X=0,   //Счетчик числа X
                        output reg [`m_Y-1:0]Y=0,   //Счетчик числа Y
                        output wire ce_end,    //Конец синхронного времени измерения
                        output reg Q_Tm=0,  //Синхронный с Tx интервал измерения
                        output wire ok_DIV); //Конец деления

wire clk_Fx ;
reg tQ_Tm=0 ,ttQ_Tm=0 ;
assign ce_end=(!tQ_Tm & ttQ_Tm) ;//Конец синхронного времени измерения
//Выделение целого числа периодов
always @ (posedge clk_Fx) begin
Q_Tm <= Tm ;//Импульс с длительностью целого числа периодов Tx
end
//Счет числа X периодов
always @ (negedge clk_Fx or posedge st) begin
X <= st? 0 : Q_Tm? X+1 : X ;//Счет числа X с асинхронным сбросом
end
//Счет числа Y (число интервалов Tce в целом числе периодов)
always @ (posedge clk) begin
Y <= st? 0 : (Q_Tm & ce)? Y+1 : Y ;//Счет числа Y счетчиком с синхронным сбросом
end
```

```
always @ (posedge clk) begin
tQ_Tm <= Q_Tm ; ttQ_Tm <= tQ_Tm ;
end
// Глобальный буфер входного сигнала
BUFG DD1 (.I(Ux),   .O(clk_Fx));
// Генератор времени измерения
Gen_TM DD2 (.st(st),          .Tm(Tm),
                .clk(clk),
                .ce(ce));
//        Арифметический умножитель
parameter NFce = `Fce ;       // (см. CONST_XY.v)
wire[`m_M-1:0]M=X*NFce;
//        Арифметический делитель
DIV_AB_QF DD3 (  .clk(clk),       .ok_div(ok_DIV),//Конец деления
                 .st(ce_end),     .F(F),//Дробная часть частного
                 .A(M),           .Q(Q),//Целая часть частного
                 .B(Y));
endmodule
```

1.1.2 Пример схемы генератора Tm

```
`include "CONST_XY.v"
module Gen_TM(     input clk,      output reg Tm=0,      //Интервал измерения
                   input st,
                   input ce);     //Tce - Эталон времени

parameter NP=1<<`m  ;             //Число Tce в Tm, NP=2^`m
reg [`m_Y-1:0]cb_NP=0;            //Счетчик числа Tce в Tm
reg Q_st =0 ;                     //Синхронизатор старта
wire T_stop = (cb_NP==1);         //Конец импульса
always @ (posedge clk) begin
Q_st <= st? 1 : ce? 0 : Q_st ;
Tm <= (Q_st & ce)? 1 : (T_stop & ce)? 0 : Tm ;
cb_NP <= (Q_st & ce)? NP : (Tm & ce)? cb_NP-1 : cb_NP ;
end
endmodule
```

1.1.3 Модуль параметров **CONST_XY.v** для **Fmes_PR_WF**

```
`define m 4                 //Для моделирования (см.таблицу 1)
`define Fclk 50000000       //50 MHz (Сигнал синхронизации)
`define Fce   10000         //(см.таблицу 1 и таблицу 2)
`define Nce `Fclk/`Fce      // Для генератора ce модуля Gen_Ux
`define m_NFce    17        //2^`m_NFce > `Fce
//`define m 14              //Для сдачи работы (см.таблицу 2)
`define m_Y `m+1            //Число разрядов счетчика Y
`define m_X `m+1            //Число разрядов счетчика X
`define m_M `m_X+`m_NFce//Число разрядов произведения X*NFce
`define m_S `m_Y            //Число разрядов делителя
```

1.1.4 Пример данных к заданию на моделирование измерителя частоты **Fmes_PR_WF**

```
parameter Tx = 290866;       //Tx=1/Fx=1/3.438 kHz=290.866us=290866ns
parameter Tclk = 20;         //Tclk=1/Fclk=1/50 mHz = 20 ns
parameter Tce = 100000;      //Tce=1/Fce = 1/10000=100000 ns
parameter NTce = Tce/Tclk ; //NTce=5000
always begin Ux=0 ; #(Tx/2) Ux=1; #(Tx/2) ; end          //"Прямой" сигнал
//always begin Ux=1 ; #(Tx/2) Ux=0; #(Tx/2) ; end        //Инверсный сигнал
always begin clk=0 ; #(Tclk/2) clk=1; #(Tclk/2) ; end
always begin ce=0 ; #(Tclk*(NTce-1)) ce=1; #(Tclk) ; end
        initial begin
                st = 0;
#1000;          st = 1;
#20;            st = 0;
```

2. Аккумуляторные частотомеры

В рассмотренном выше частотомере все импульсы суммируются с одинаковым весом 1 (прямоугольная весовая функция). Накапливающий сумматор или, что тоже аккумулятор позволяет построить частотомер, в котором периоды измеряемого сигнала суммируются с разными весами, задаваемыми весовой функцией. Уменьшение весовых коэффициентов на концах интервала измерения снижает влияние случайности начала счета относительно измеряемого сигнала, т.е. снижает погрешность дискретности отсчета.

2.1 Аккумуляторный частотомер с треугольной весовой функцией

Простейшей весовой функцией, удовлетворяющей этому условию, является “треугольная” функция, в которой коэффициенты вначале линейно увеличиваются, достигают к середине интервала измерения максимума и затем линейно уменьшаются до нуля к концу интервала. В общем случае коэффициенты весовой функции должны выбираться из таблицы, т.е. из постоянного запоминающего устройства (ПЗУ). В данном простейшем случае “треугольной” весовой функции дискретные коэффициенты можно формировать при помощи реверсивного счетчика.

Для оценки частоты необходимо вычислить отношение полученной суммы S_A к полной сумме S коэффициентов и умножить это отношение на номинальное значение частоты F_{ceN}, $\hat{F}_X = F_{CEN} \cdot \frac{\sum_{i=1}^{X} A_i}{S}$.

2.1.1 Генератор треугольной весовой функции (A1)

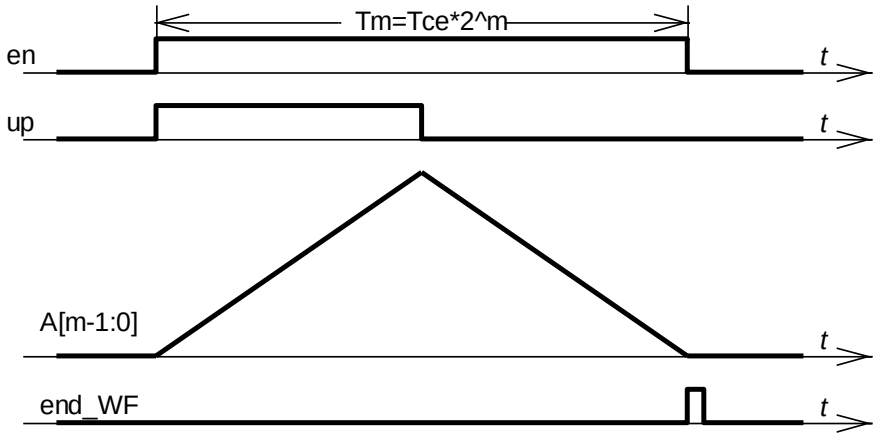


Рис.2 Временные диаграммы генератора треугольной весовой функции

2.1.2 Схема модуля **Gen_TR_WF** генератора треугольной весовой функции

```
`include "CONST_TR.v"
module Gen_TR_WF(input clk,     output reg[`m_WF-1:0] A=0,//Весовые коэффициенты
                 input st,      output reg up=0,          //Направление счета
                 input ce,      output wire Tmes,         //Интервал измерения
                                output wire end_WF,       //Конец функции
                                output reg[`m_S-1:0] S=0); //Сумма коэффициентов
reg en=0, ten=0, Q_st=0;
assign end_WF = !en & ten ;
assign Tmes = en & ten;//
always @ (posedge clk) begin
Q_st <= st? 1 : ce? 0 : Q_st ;
en <=  (Q_st & ce)? 1 : (!up & (A==1) & ce)? 0 : en ;
ten <= ce? en : ten ;
up <= st? 1 : ((A==1<<`m-1) & ce)? 0 : up ;          //Направление счета
A <= st? 0 : (ce & up & en & !(A==1<<`m-1))? A+1 : (ce & !up & en)? A-1 : A ;
S <= st? 0 : (ce & en)? S+A : S ;
end
endmodule
```

2.1.3 Пример данных для задания на моделирования генератора весовой функции

```
parameter Tclk = 20;          //Tclk=1/Fclk=1/50 mHz = 20 ns
parameter NTce = 10 ;         //Fce=5MHz
always begin clk=0 ; #(Tclk/2) clk=1; #(Tclk/2) ; end
always begin ce=0 ; #(Tclk*(NTce-1)) ce=1; #(Tclk) ; end
        initial begin
        st = 0;
#100;   st = 1;
#20;    st = 0;
```

2.1.4 Модуль параметров **CONST_TR.v** для **Gen_TR_WF**

```
`define m 4                  //Для моделирования (см.таблицу 1)
`define Fclk 50000000        //50 Mhz (Сигнал синхронизации макета Nexys2)
`define Fce    100000        //(см.таблицу и таблицы 1,2)
`define Nce `Fclk/`Fce       // Для генератора ce модуля Gen_Ux
`define m_NFce     17        //2^`m_NFce > `Fce
//`define m 14                //для сдачи работы (см.таблицу 2)
`define m_WF `m              //Число разрядов весовой функции
`define m_S 2*`m             //Число разрядов суммы весовых коэффициентов
`define m_M `m_NFce+`m_S  //Число разрядов произведения.
```

Если число разрядов счетчика равно *m*, то интервал измерения равен $T_{CE} \cdot 2^m$, а полная сумма коэффициентов $S = 2^{m-1} \cdot (2^{m-1} + 1)$.

2.1.5 Модуль измерителя частоты **Fmes_TR_WF** с треугольной весовой функцией.

```
`include "CONST_TR.v"
module Fmes_TR_WF(
                input ce,       output wire [`m_WF-1:0]A,  //Весовая функция
                input Ux,       output reg[`m_S-1:0]SA=0,  //Накопленная сумма
                input clk,      output wire[`m_S-1:0]S,    //Полная сумма
```

```
                input st,       output wire Tmes,               //Интервал измерения
                                output wire up,                 //Направления счета
                                output reg [`m:0]X=0, //Число периодов на интервале измерения
                                output wire end_WF,             //Конец функции
                                output wire[`m_M-1:0]Q,         //Целая часть
                                output wire[`m_S-1:0]F,         //Дробная часть
                                output wire ok_DIV);            //Конец делени

wire Fx_clk ;
parameter NFce = `Fce ;             // Множитель делимого (Fx=NFce*SA/S)
wire [`m_M-1:0]M=NFce*SA ;          //Произведение M=NFce*SA

always @ (posedge Fx_clk or posedge st) begin //st - асинхронный сброс
SA <= st? 0 : Tmes? SA+A : SA ;
X <= st? 0 : Tmes? X+1 : X ;
end
BUFG DD1 (.I(Ux),   .O(Fx_clk));//Глобальный буфер
Gen_TR_WF DD2 (  .clk(clk),         .A(A),
                 .ce(ce),           .up(up),
                 .st(st),           .Tmes(Tmes),
                                    .end_WF(end_WF),
                                    .S(S));
DIV_AB_QF DD3 (  .clk(clk),         .Q(Q),          //Целая часть
                 .A(M),             .F(F),          //Дробная часть
                 .B(S),             .ok_div(ok_DIV),
                 .st(end_WF));
endmodule
```

2.1.6 Пример данных к заданию на моделирование измерителя частоты с треугольной весовой функцией

```
parameter Tx = 1684;                //Tx=1/Fx=10^9/593800 = 1684ns
parameter Tclk = 20;                //Tclk=1/Fclk=1/50 mHz = 20 ns
parameter Tce = 1000;               //Tce=1/Fce = 1/1000000=1000 ns
parameter NTce = Tce/Tclk ;         //NTce=50
always begin Ux=0 ; #(Tx/2) Ux=1; #(Tx/2) ; end         //"Прямой" сигнал
//always begin Ux=1 ; #(Tx/2) Ux=0; #(Tx/2) ; end       //Инверсный сигнал
always begin clk=0 ; #(Tclk/2) clk=1; #(Tclk/2) ; end
always begin ce=0 ; #(Tclk*(NTce-1)) ce=1; #(Tclk) ; end
        initial begin
                st = 0;
#1500;          st = 1;
#20;            st = 0;
        End
```

2.2 Аккумуляторный частотомер с параболической весовой функцией

Оптимальной по отношению к аддитивной помехе, приводящей к флюктуациям периода измеряемого сигнала, является весовая параболическая функция вида $x(1 - x)$.

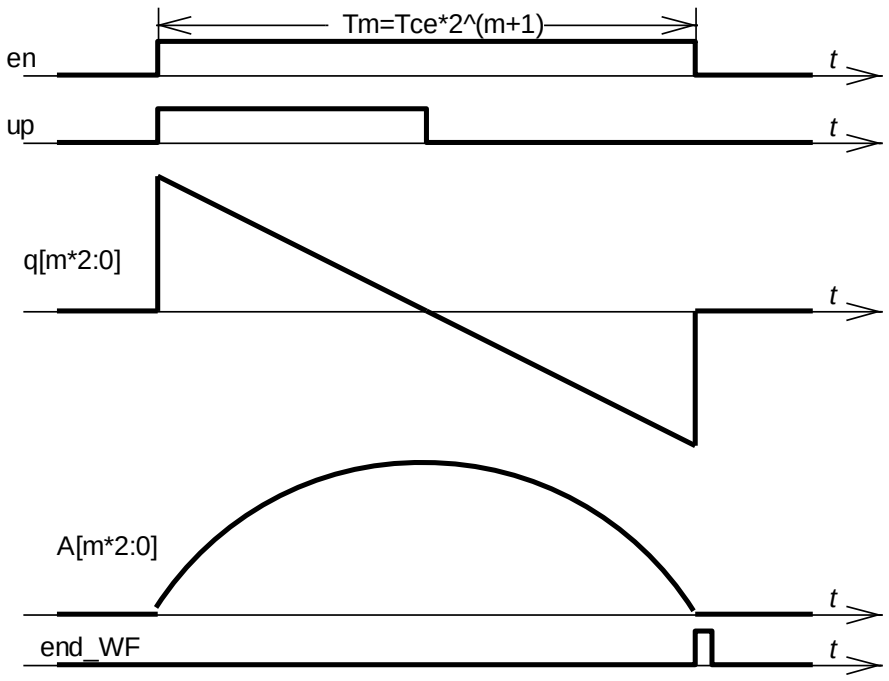


Рис.3 Временные диаграммы генератора параболической весовой функции

В данном случае коэффициенты весовой функции можно вычислять при помощи вычитающего счетчика и аккумулятора. Если на вход аккумулятора подавать число, которое, начиная с $\boldsymbol{D}_0$, с каждым тактом уменьшается на 1, ($\boldsymbol{Q}_i = \boldsymbol{D}_0 - i$) до $-\boldsymbol{D}_0$, то число на выходе аккумулятора, являющееся суммой всех чисел предыдущих тактов, $\boldsymbol{A}_k = k * \boldsymbol{D}_0 - k\ (k+1)/2$, будет вначале увеличиваться, достигать к середине интервала измерения максимума и затем уменьшаться до нуля к концу интервала. Интервал измерения равен $T_{ce}*2^{m+1}$, максимальный коэффициент $\boldsymbol{A}_{max}= 2^{2m}-2^{m-1}(2^m-1)$, а полная сумма коэффициентов $\boldsymbol{S}= 2^m(2^{2m+1}+3*2^m+1)/3$.

2.2.1 Схема модуля **Gen_PAR_WF** генератора параболической весовой функции

```
`include "CONST_PAR.v"
module Gen_PAR_WF(
        input ce,       output wire Tmes,              //Интервал измерения
        input clk,      output reg[`m*2:0] q =0,       //Аргумент весовой функции
        input st,       output reg[`m*2:0]A =0,        //Весовая функция
                        output wire end_WF,            //Конец функции
                        output reg[`m*3-1:0]S =0);     //Сумма всех коэффициентов
reg en=0, ten=0, Q_st=0;
assign end_WF = !en & ten ;
assign Tmes = en ;//& ten
wire end_en = (q==(1<<(`m*2+1))-(1<<`m)) ;
always @ (posedge clk) begin
Q_st <= st? 1 : ce? 0 : Q_st ;
en <= (Q_st & ce)? 1 : (end_en & ce)? 0 : en ;
ten <= ce? en : ten ;
q  <= st? (1<<`m)-1 : (Tmes & ce)?  q-1 : q ;
A  <= st? (1<<`m) : ((q==(1<<(`m*2+1))-(1<<`m)) & ce)? 0 : (Tmes & ce)? A + q : A ;
S  <= st? 0 : (Tmes & ce)?  S+A :  S ;
end
```

```
endmodule
```

2.2.2 Пример данных для задания на моделирования генератора весовой функции

```
parameter Tclk = 20;          //Tclk=1/Fclk=1/50 mHz = 20 ns
parameter NTce = 10 ;         //Fce=5MHz
always begin clk=0 ; #(Tclk/2) clk=1; #(Tclk/2) ; end
always begin ce=0 ; #(Tclk*(NTce-1)) ce=1; #(Tclk) ; end
        initial begin
        st = 0;
#100;   st = 1;
#20;    st = 0;
```

2.2.3 Модуль параметров **CONST_PAR.v** для **Gen_PAR_WF**

```
`define m 3                   //Для моделирования (см.таблицу 1)
`define Fclk 50000000         //50 Mhz (Сигнал синхронизации)
`define Fce   1000000         //1000 kHz
`define Nce `Fclk/`Fce        // Для генератора ce модуля Gen_Ux
`define m_NFce     20         //2^`m_NFce > `Fce
//`define m 10                //Для сдачи работы (см.таблицу 2)
`define m_WF 2*`m+1           //Число разрядов весовой функции
`define m_S 3*`m              //Число разрядов суммы весовых коэффициентов
`define m_M `m_NFce+`m_S  //Число разрядов произведения
```

2.2.4 Модуль измерителя частоты **Fmes_PAR_WF** с параболической весовой функцией.

```
`include "CONST_PAR.v"
module Fmes_PAR_WF(
        input ce,       output wire Tmes,             //Интервал измерения
        input Ux,       output wire [`m_WF-1:0]A,     //Весовая функция
        input clk,      output reg[`m_S-1:0]SA=0,     //Накопленная сумма
        input st,       output wire[`m_S-1:0]S,       //Полная сумма
                        output reg [`m:0]X=0,         //Число периодов на интервале измерения
                        output wire end_WF,           //Конец функции
                        output wire[`m_M-1:0]Q,       //Целая часть
                        output wire[`m_S-1:0]F,       //Дробная часть
                        output wire ok_DIV);          //Конец деления
wire Fx_clk ;
parameter NFce = `Fce ;             //Множитель делимого (Fx=NFce*SA/S)
wire [`m_M-1:0]M=NFce*SA ;          //Произведение M=NFce*SA

always @ (posedge Fx_clk  or posedge st) begin     //st - асинхронный сброс
SA <= st? 0 : Tmes? SA+A : SA ;
X <= st? 0 : Tmes? X+1 : X ;
end
BUFG DD1 (.I(Ux),   .O(Fx_clk));
Gen_PAR_WF  DD2 (.clk(clk),       .A(A),
                 .st(st),         .Tmes(Tmes),
                 .ce(ce),         .end_WF(end_WF),
                                  .S(S));
DIV_AB_QF   DD3 ( .clk(clk),
                 .A(M),           .Q(Q),          //Целая часть
                 .B(S),           .F(F),          //Дробная часть
```

```
                .st(end_WF),   .ok_div(ok_DIV));
endmodule
```

2.2.5 Пример данных к заданию на моделирование измерителя частоты с параболической весовой функцией

```
parameter Tx = 246184;        //Tx=1/Fx=10^9/4062 Hz=246184ns
parameter Tclk = 20;          //Tclk=1/Fclk=1/50 mHz = 20 ns
parameter Tce = 100000; //Tce=1/Fce = 1/1000000=1000 ns
parameter NTce = Tce/Tclk ; //NTce=5000
//always begin Ux=0 ; #(Tx/2) Ux=1; #(Tx/2) ; end
always begin Ux=1 ; #(Tx/2) Ux=0; #(Tx/2) ; end  //Инверсный сигнал
always begin clk=0 ; #(Tclk/2) clk=1; #(Tclk/2) ; end
always begin ce=0 ; #(Tclk*(NTce-1)) ce=1; #(Tclk) ; end
        initial begin
                st = 0;
#100000;        st = 1;
#20;            st = 0;
end
```

2.3 Аккумуляторный частотомер с кусочно-параболической весовой функцией

Ещё больше снижает погрешность дискретности отсчета весовая функция, у которой и первая производная на концах интервала равна нулю, например, $\sin^2 x$. Аппаратными средствами ПЛИС (без использования ПЗУ) вычислять значения квадрата синуса неудобно. Можно аппроксимировать синус отрезками параболы: $a = x^2$ ($0 < x < 1/4$), $a = (1 - x^2)$ ($3/4 < x < 1$), а на интервале ¼ < x < 3/4 “опрокинутой” параболой с вершиной в центре интервала (x = 1/2).

В этой схеме интегрируется “пилообразная” функция, состоящая из положительного и отрицательного треугольников равных площадей.

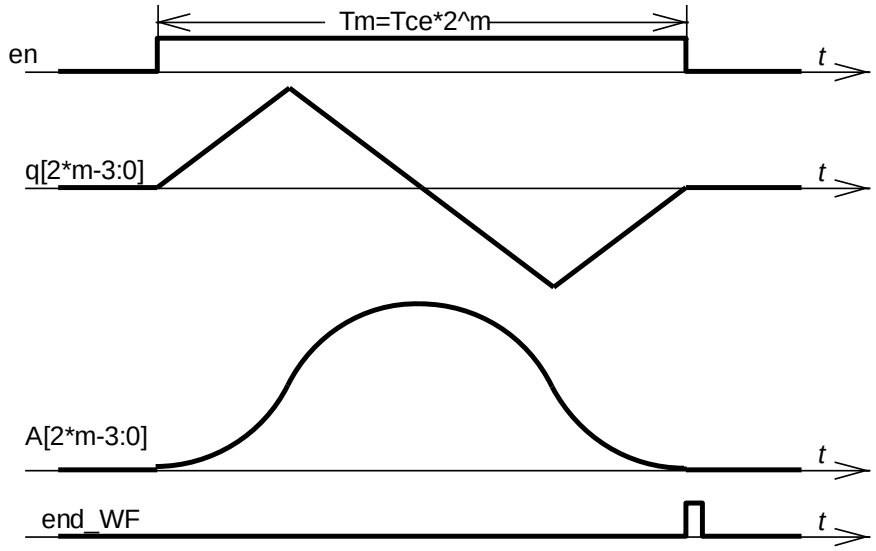


Рис.4 Временные диаграммы генератора кусочно-параболической весовой функции

Интервал измерения *Tm* равен $T_{CE} \cdot 2^m$. Полная сумма коэффициентов

$$S = 2^m \cdot (2^{2m-5} + 1).$$

2.3.1 Схема модуля **Gen_4PAR_WF** генератора кусочно-параболической весовой функции

```
`include "CONST_4PAR.v"
module Gen_4PAR_WF (    output wire Tmes,              //Интервал измерения
                input ce,       output wire end_WF,            //Конец функции
                input clk,      output reg[`m_WF-1:0]A=0, //Весовые коэффициенты
                input st,       output reg [`m_q-1:0]q=0,      //Аргумент весовой функции
                                output reg up=0,               //Направление счета
                                output reg[`m_S-1:0]S=0);      //Сумма коэффициентов
reg en=0 ; reg ten=0 ; reg Q_st=0 ;
assign end_WF = !en & ten ;
assign Tmes = en ;//& ten

always @ (posedge clk) begin
Q_st <= st? 1 : ce? 0 : Q_st ;
en <= (Q_st & ce)? 1 : (ce & (q==0) & up)? 0 : en ;
ten <= ce? en : ten ;
up <= (st | (ce &(!up & q==((1<<`m_q)-(1<<`m-2)+1))))? 1 :
          (ce & ((up & (q==(1<<`m-2)-1)) | (ce & (q==0) & up)))? 0 : up ;
q  <= st? 1 : (ce & en & up)? q+1 : (ce & en & !up)? q-1 : q ;
A  <= st? 1 : (ce & en & (q==0) & up)? 0 : (ce & en)? A + q : A ;
S  <= st? 0 : (en & ce) ? S+A : S ;
end
endmodule
```

2.3.2 Пример данных для задания на моделирования генератора весовой функции

```
parameter Tclk = 20;          //Tclk=1/Fclk=1/50 mHz = 20 ns
parameter NTce = 500 ;        //Fce=100kHz
always begin clk=0 ; #(Tclk/2) clk=1; #(Tclk/2) ; end
always begin ce=0 ; #(Tclk*(NTce-1)) ce=1; #(Tclk) ; end
        initial begin
        st = 0;
#100;   st = 1;
#20;    st = 0;
```

2.3.3 Модуль параметров **CONST_4PAR.v** для **Gen_4PAR_WF**

```
`define m 4                   //Для моделирования (см.таблицу 1)
`define Fclk 50000000         //50 Mhz (Сигнал синхронизации макета Nexys2)
`define Fce   1000000         //1000 kHz (1000000<2^20)
`define Nce `Fclk/`Fce        // Для генератора ce модуля Gen_Ux
`define m_Fce 20              //Число разрядов номинала частоты Fce
//`define m 11                //Число разрядов (см.таблицу 2)
`define m_q 2*`m-3            //Число разрядов аргумента весовой функции
`define m_WF 2*`m-3           //Число разрядов весовой функции
`define m_S 3*`m-4            //Число разрядов суммы весовых коэффициентов
`define m_M `m_Fce+`m_S //Число разрядов произведения
```

2.3.4 Схема модуля **Fmes_4PAR_WF** измерителя частоты с кусочно-параболической весовой функцией.

```
`include "CONST_4PAR.v"
module Fmes_4PAR_WF(output wire Tmes,              //Интервал измерения
        input ce,       output wire end_WF,        //Конец функции
```

```
        input Ux,       output wire up,                 //Направления счета
        input st,       output wire [`m_WF-1:0]A,       //Весовая функция
        input clk,      output reg[`m_S-1:0]SA=0,       //Накопленная сумма
                        output reg [`m:0]X=0, //Число периодов на интервале измерения
                        output wire[`m_M-1:0]Q,         //Целая часть
                        output wire[`m_S-1:0]F,         //Дробная часть
                        output wire ok_DIV);            //Конец деления
wire Fx_clk ;
parameter NFce = `Fce ;                 //Множитель делимого (Fx=NFce*SA/S)
wire [`m_M-1:0]M=NFce*SA ;              //Произведение M=NFce*SA

always @ (posedge Fx_clk or posedge st) begin       //st - асинхронный сброс
SA <= st? 0 : Tmes? SA+A : SA ;     //Накопление A
X <= st? 0 : Tmes? X+1 : X ;        //Счет числа X
end

BUFG DD1 (.I(Ux),   .O(Fx_clk));// Глобальный буфер
// Генератор весовых коэффициентов
Gen_4PAR_WF DD2 (.clk(clk),     .Tmes(Tmes),
                 .st(st),       .A(A),
                 .ce(ce),       .up(up),
                                .end_WF(end_WF),
                                .S(S));
DIV_AB_QF DD3 (.clk(clk),       //Арифметический делитель
                 .A(M),         .Q(Q), //Целая часть
                 .B(S),         .F(F),   //Дробная часть
                 .st(end_WF),   .ok_div(ok_DIV));
endmodule
```

2.3.5 Пример данных к заданию на моделирование измерителя частоты с кусочно-параболической весовой функцией

```
parameter Tx = 290866;      //Tx=1/Fx=1/3.438 kHz=290.866us=290866ns
parameter Tclk = 20;        //Tclk=1/Fclk=1/50 mHz = 20 ns
parameter Tce = 100000;     //Tce=1/Fce = 1/10000=100000 ns
parameter NTce = Tce/Tclk ; //NTce=5000
always begin Ux=0 ; #(Tx/2) Ux=1; #(Tx/2) ; end         //"Прямой" сигнал
//always begin Ux=1 ; #(Tx/2) Ux=0; #(Tx/2) ; end       //Инверсный сигнал
always begin clk=0 ; #(Tclk/2) clk=1; #(Tclk/2) ; end
always begin ce=0 ; #(Tclk*(NTce-1)) ce=1; #(Tclk) ; end
        initial begin
                st = 0;
#1000;          st = 1;
#20;            st = 0;
        end
```

Напомним, что для оценки частоты необходимо вычислить отношение полученной суммы SA к полной сумме S коэффициентов и умножить это отношение на номинальное значение частоты ($F_{CEN} = \frac{1}{T_{CEN}}$) разрешения сигнала синхронизации (ce) генератора весовых коэффициентов.

3. Задание к допуску

Таблица параметров для моделирования

Таблица 1

№	**Вариант измерителя частоты**		S	Fce [Hz]	Fx [kHz]
1	Fmes_XY	m	~16	100000	34.38
2	Fmes_TR_WF	4	72	10000	54.38
3	Fmes_PAR_WF	3	408	1000000	656.2
4	Fmes_4PAR_WF	4	144	1000	0.4688
5	Fmes_XY	4	~16	10000	3.438
6	Fmes_TR_WF	4	72	100000	71.88
7	Fmes_PAR_WF	3	408	1000000	543.8
8	Fmes_4PAR_WF	4	144	100000	65.62
9	Fmes_XY	4	~16	1000000	343.8
10	Fmes_TR_WF	4	72	10000	7.188
11	Fmes_PAR_WF	3	408	100000	54.38
12	Fmes_4PAR_WF	4	144	1000000	593.8
13	Fmes_XY	4	~16	100000	34.38
14	Fmes_TR_WF	4	72	10000	7.188
15	Fmes_PAR_WF	3	408	100000	54.38
16	Fmes_4PAR_WF	4	144	1000000	656.5
17	Fmes_XY	4	~16	100000	46.880
18	Fmes_TR_WF	4	72	1000000	343.80
19	Fmes_PAR_WF	3	408	100000	54.38
20	Fmes_4PAR_WF	4	144	1000000	593.8

3.1 Начертить в тетради эскизы временных диаграмм заданного (в таблицах 1 или 2) варианта генератора весовой функции.

3.2 Для заданного варианта параметров измерителя частоты написать в тетради схему модуля генератора весовой функции .

3.3 Для заданного варианта написать в тетради схему модуля измерителя частоты.

3.4 Начертить и написать в тетради пример структуры регистров и схему модуля арифметического делителя (см. приложение 6.3)

3.5 Написать в тетради схему модуля синтезатора частоты с заданным (в таблице 2) значением NFx. Определить частоту выходного сигнала Ux.

4. Задание к выполнению работы

4.1 Создать модуль заданного варианта CONST_???. Установить в этом модуле значения m и Fce своего варианта из таблицы 1.

4.2 Создать модуль генератора заданного варианта весовой функции. Провести моделирование его работы. Зарисовать в тетради характерные эскизы временных диаграмм.

4.3 Создать модуль **DIV_AB_QF** арифметического делителя (приложение 6.3.2). Провести моделирование его работы. Записать в тетради характерные результаты моделирования.

4.4 Создать модуль заданного варианта измерителя частоты. Провести моделирование его работы для двух вариантов входного сигнала Ux (“прямой” и инверсный). По результатам моделирования оценить частоту Fx и погрешность дискретности отсчета. Зарисовать в тетради характерные эскизы временных диаграмм.

4.5 Создать модуль **HEX27_to_DEC8** преобразователя двоичного числа в декадное двоично-десятичное число (приложение 6.4). Провести моделирование его работы. Записать в тетради характерные результаты моделирования.

4.6 Создать модуль **FB16_to_FD27** преобразователя двоичной дроби в декадную дробь. Провести моделирование его работы (приложение 6.5). Зарисовать в тетради характерные эскизы временных диаграмм.

5. Задание к сдаче работы

Таблица параметров для сдачи работы

Таблица 2

№	Вариант измерителя	Fce [Hz]	NFx	m	Tm [ms]	S или Y
1	Fmes_XY	100000	49	15	327,68	32768
2	Fmes_TR_WF	10000	391	12	409,6	4196352
3	Fmes_PAR_WF	1000000	781	10	2,048	716876800
4	Fmes_4PAR_WF	1000	781	11	2048	268437504
5	Fmes_XY	10000	98	14	1638,4	16384
6	Fmes_TR_WF	100000	98	14	163,84	67117056
7	Fmes_PAR_WF	1000000	391	11	4,096	5730818048
8	Fmes_4PAR_WF	100000	3125	9	5,12	4194816
9	Fmes_XY	1000000	195	13	8,192	8192
10	Fmes_TR_WF	10000	781	11	204,8	1049600
11	Fmes_PAR_WF	100000	781	10	20,48	716876800
12	Fmes_4PAR_WF	1000000	6250	8	0,256	524544
13	Fmes_XY	100000	24	16	655,36	65536
14	Fmes_TR_WF	10000	391	12	409,6	4196352
15	Fmes_PAR_WF	100000	1563	9	10,24	89740800

16	Fmes_4PAR_WF	1000000	1563	10	1,024	33555456
17	Fmes_XY	100000	3125	9	5,12	512
18	Fmes_TR_WF	1000000	98	14	16,384	67117056
19	Fmes_PAR_WF	100000	3125	8	5,12	11250432
20	Fmes_4PAR_WF	1000000	781	11	2,048	268437504

5.1 Установить в модуле CONST_??? значения m и Fce, а в модуле GEN_Ux NFx своего варианта из таблицы 2.

5.2 Создать модули и символы

- **Gen_Ux** - синтезатора сигнала Ux (приложение 6.1),
- **Display** - семи сегментного индикатора (приложение 6.2),
- **BTN_BL** - модуля кнопки (приложение 6.5),
- **MUX_BL** - мультиплексора данных измерителя частоты для модуля **Display** (
  8 16-ти битных входов данных D0i[15:0], ... D7i[15:0],
  3-х битный адрес ADR[2:0] и
  16-ти битный выход Dout[15:0]).

5.3 Создать символы модулей

- **Fmes_???_WF** - своего варианта измерителя частоты,
- **HEX27_to_DEC8** ,
- **FB16_to_FD27**.

5.4 Из созданных модулей и символов создать схему **Sch_Lab403** (см. примеры схем рис.5-рис.8). Согласовать номера разрядов выходных и входных шин модулей. На модуль **HEX27_to_DEC8** надо подавать младшие разряды целой части Q, а на модуль **FB16_to_FD27** старшие разряды F дробной части делителя модуля **Fmes_??_WF**. В схемах рис.6-рис.8 модуль **BUS_BL** необходим, если число разрядов выходной шины SA меньше 32.

Создать для схемы **Sch_Lab403** файл «Sch_Lab403.ucf», согласовать имена сигналов «Sch_Lab403.ucf» файла и схемы.

Создать файл конфигурации *.bit или *.mcs, загрузить в макет. Проверить и отладить при помощи индикатора и осциллографа работу генератора сигнала **Ux.** По сигналам **ce** или **ce1ms** провести калибровку частотомера осциллографа. Определить «дискрет» изменения частоты генератора **Gen_Ux**. После проверки и отладки продемонстрировать преподавателю работу генератора **Gen_Ux.**

5.5 При заданном NFx кнопкой BTN0 запустить измеритель частоты. Проверить соответствие показаний семи сегментного индикатора частоте сигнала Ux. Добиться правильного соответствия. Сделать 10 отсчетов.

5.5.1 По разности между максимальным и минимальным отсчетом определить относительную погрешность дискретности.

5.5.2 По среднему значению из 10 отсчетом определить относительное смещение оценки частоты.

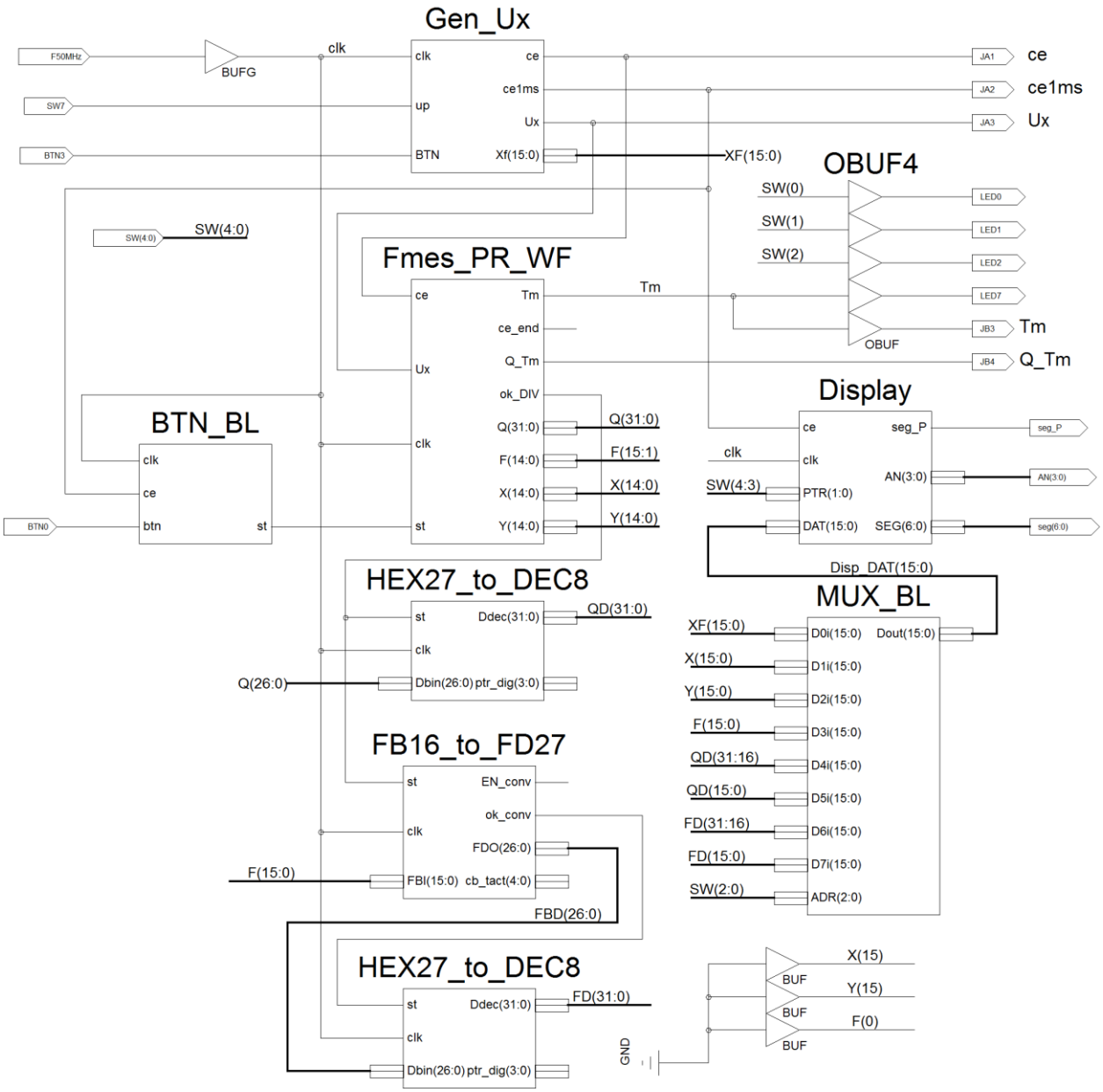


Рис.5 Пример схемы лабораторной работы варианта измерителя частоты Fmes_PR_WF с прямоугольной весовой функцией (m=14)

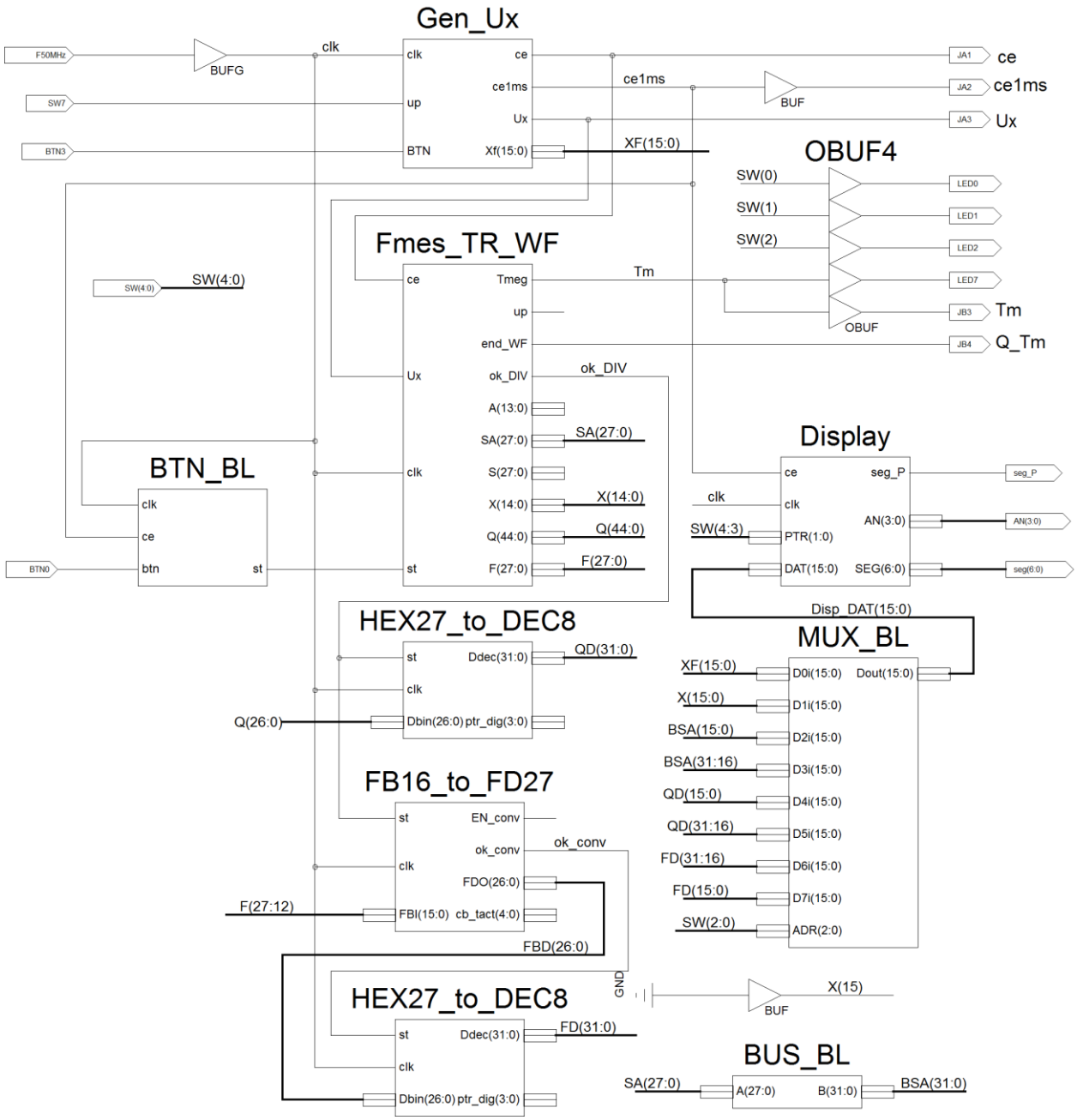


Рис.6 Пример схемы лабораторной работы варианта измерителя частоты Fmes_TR_WF с треугольной весовой функцией (m=14)

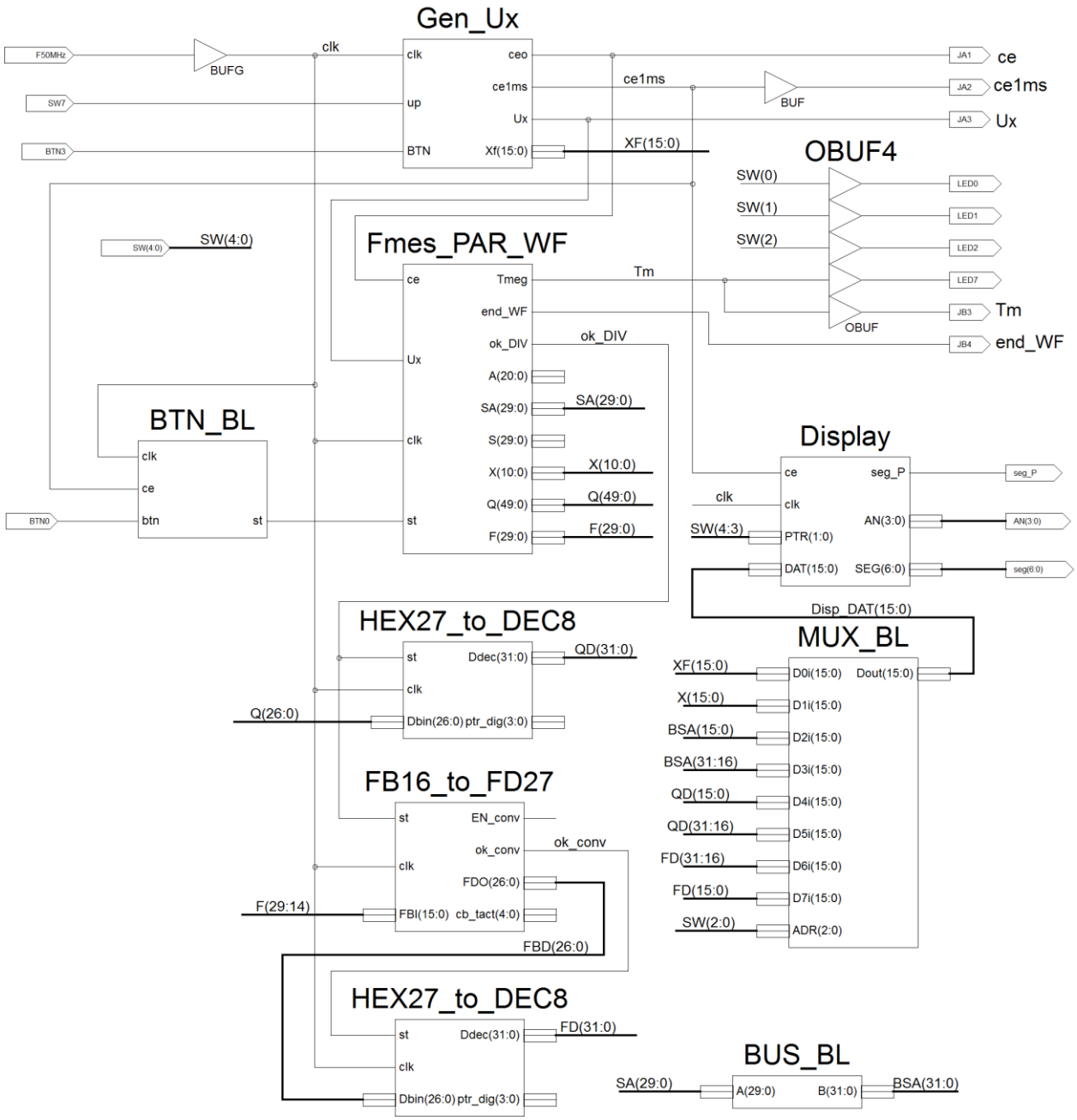


Рис.7 Пример схемы лабораторной работы варианта измерителя частоты Fmes_PAR_WF с параболической весовой функцией (m=10)

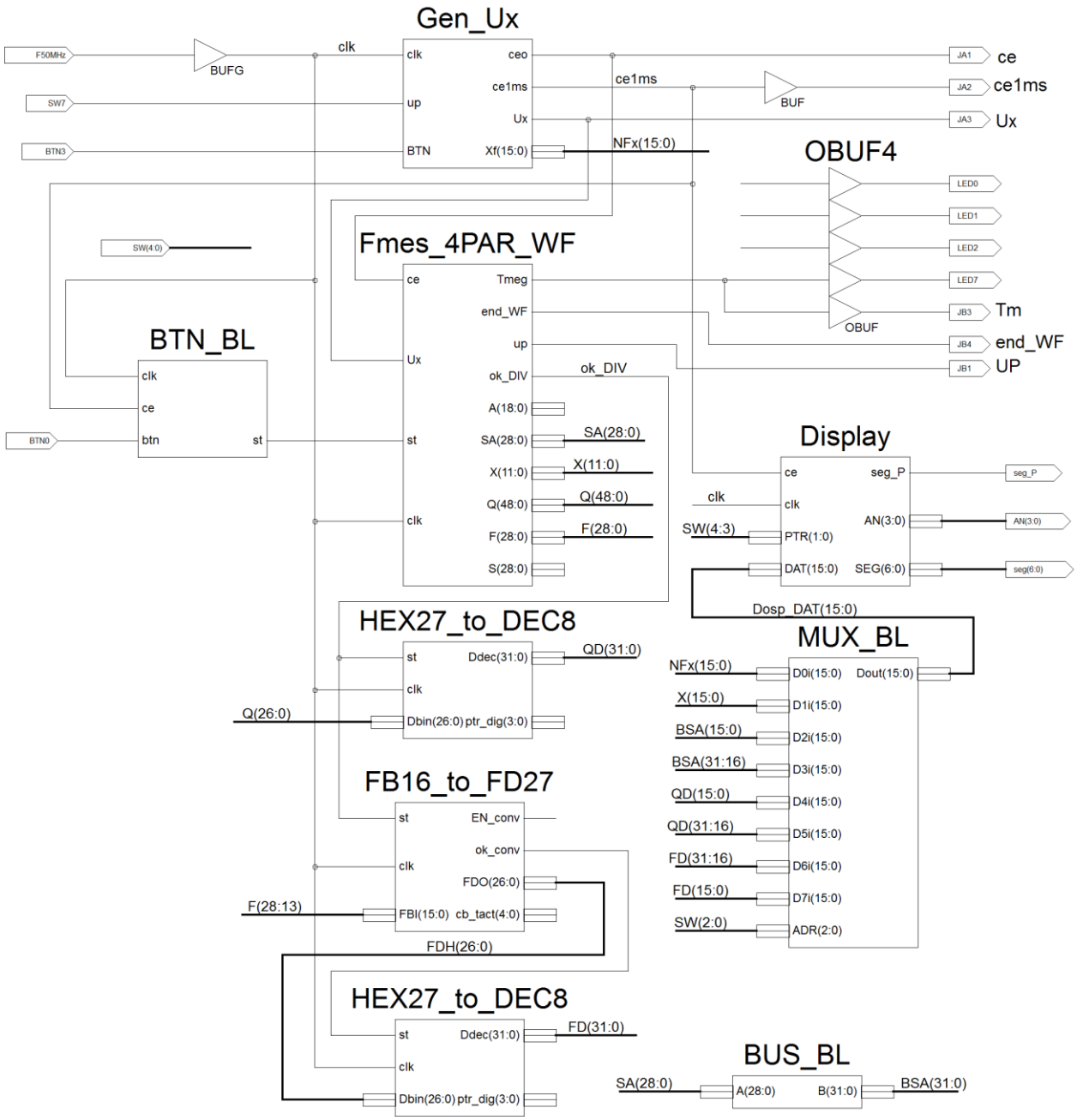


Рис.8 Пример схемы лабораторной работы варианта измерителя частоты Fmes_4PAR_WF с кусочно параболической весовой функцией (m=11)

6. Приложения

6.1 Схема модуля **Gen_Ux** генератора сигнала Ux

```
//`include "CONST_TR.v"
//`include "CONST_XY.v"
//`include "CONST_PAR.v"
`include "CONST_4PAR.v"
module Gen_Ux(      input clk,      output reg ceo=0,
                    input up,       output wire ce1ms,//Для подавления "дребезга" кнопок
                    input BTN,      output reg Ux=0,
```

```
                                        output reg[15:0]Xf=781);//(NFx см. в таблице 2)
parameter F1kHz = 1000 ;
reg [15:0]cb_ce1ms=0 ;        //Счетчик 1 милисекунды
assign ce1ms = (cb_ce1ms==`Fclk/F1kHz) ;
reg [15:0]cb_ce=0 ;           //Счетчик ce синтезатора частоты Ux
wire ce = (cb_ce==`Nce) ;     //
reg tBTN=0, ttBTN=0 ;         //2 D-триггера кнопки
wire dBTN = tBTN & !ttBTN & ce1ms;      //Подавление "дребезга" кнопки

always @ (posedge clk) begin
cb_ce1ms <= ce1ms? 1 : cb_ce1ms+1 ;
cb_ce <= ce? 1 : cb_ce+1 ;
tBTN <= ce1ms? BTN : tBTN ;       ttBTN <= ce1ms? tBTN : ttBTN ;
Xf <= (up & dBTN)? Xf+1 : (!up & dBTN)? Xf-1 : Xf ;//Реверсивный счетчик X
ceo <= ce ;
end
// Синтезатор частоты
parameter M=100000 ;          //Ёмкость аккумулятора
reg[16:0]ACC=0 ;              //Регистр аккумулятора
wire co = (ACC>=M) ;          //Сигнал переноса (переполнения)
always @ (posedge clk) if (ce) begin
ACC <= co? ACC+Xf-M : ACC+Xf ;
Ux <= co ;                    //Очистка от «грязи»
end
endmodule
```

В этом модуле частота выходного сигнала Fx=X*Fce/M, M=100000, Fce - из таблицы 2, X - число с реверсивного счетчика, при up=1 фронт сигнала на входе BTN увеличивает X на 1, а при up=0 - уменьшает X на 1.

6.2 Схема модуля **Display** семи сегментного индикатора

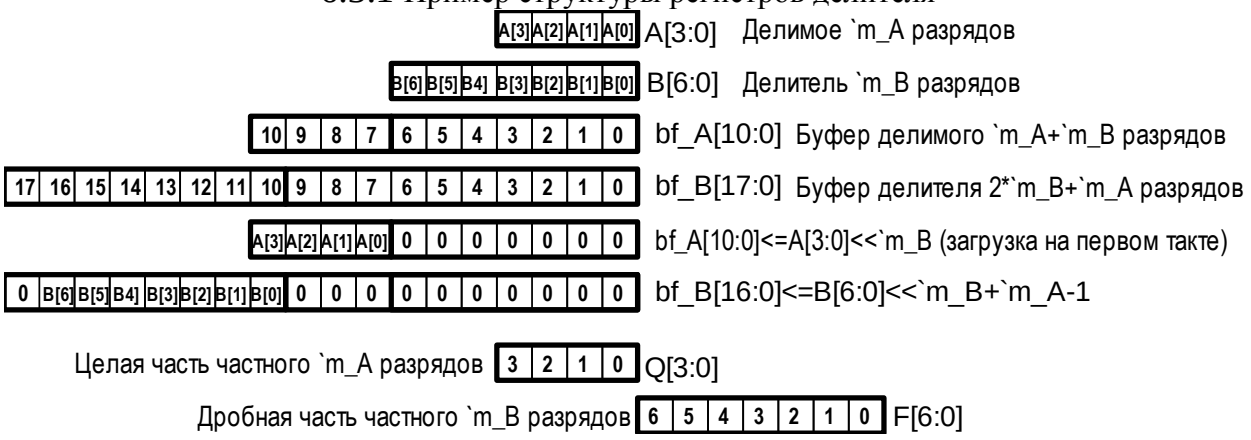
```
module Display        (
                input ce,              output wire [3:0] AN,//Аноды светодиодов
                input clk,             output wire[6:0] SEG,//Сегменты (катоды)
                input [15:0]DAT,       output wire seg_P,//Точка
                input [1:0]PTR);

wire [3:0]dig;
//Генератор "анодов"
reg [1:0] adr_dig = 0;                //Счетчик номера цифры (анода)
assign  AN =(adr_dig==0)? 4'b1110 ://включение цифры 0 (младшей)
            (adr_dig==1)? 4'b1101 ://включение цифры 1
            (adr_dig==2)? 4'b1011 ://включение цифры 2
                          4'b0111 ;//включение цифры 3 (старшей)
always @ (posedge clk) if (ce) begin
adr_dig <= adr_dig+1 ; //Адрес цифры
end
// Мультиплексор цифр
```

```
assign dig =   (adr_dig==0)?      DAT[3:0]: //Очередная цифра
               (adr_dig==1)?      DAT[7:4]:
               (adr_dig==2)?      DAT[11:8]:
                                  DAT[15:12];
// Дешифратор семи сегментных символов цифр
//                                gfedcba        сегменты
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       h
               (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
               (dig==10)? 7'b0001000 ://A
               (dig==11)? 7'b0000011 ://b
               (dig==12)? 7'b1000110 ://C
               (dig==13)? 7'b0100001 ://d
               (dig==14)? 7'b0000110 ://E
                          7'b0001110 ;//F
// Генератор точки
assign seg_P = !(PTR==adr_dig) ;
endmodule
```

6.3 Арифметический делитель

6.3.1 Пример структуры регистров делителя



6.3.2 Схема модуля арифметического делителя

```
`include "CONST_XY.v"
//`include "CONST_TR.v"
//`include "CONST_PAR.v"
//`include "CONST_4PAR.v"
`define m_A `m_M
`define m_B `m_S

module DIV_AB_QF (input clk,
        input [`m_A-1:0] A,   output wire [`m_A-1:0] Q,//Целая часть
```

```
        input [`m_B-1:0] B,   output wire [`m_B-1:0] F,//Дробная часть
        input st,             output reg ok_div=0,
        output reg [`m_A+`m_B-1:0]bf_A=0,//Буфер делимого `m_A+`m_B разрядов
        output reg [`m_A+2*`m_B-1:0]bf_B=0);//Буфер делителя 2*`m_B +`m_A  разрядов

reg [`m_A+`m_B-1:0]sr_QF=0 ;//Регистр сдвига частного `m_A+`m_B разрядов
reg[7:0]cb_tact=0 ;//счетчик тактов деления
reg TQ=0 ;//Интервал деления
wire bitQF = (bf_A>=bf_B) ;//Текущий бит деления
wire T_end = (cb_tact==`m_B+`m_A-1) ;   //Последний такт деления
assign Q=sr_QF[`m_B+`m_A-1:`m_B];       //Целая часть частного
assign F=sr_QF[`m_B-1:0];               //Дробная часть частного

always @ (posedge clk) begin
bf_A <= st? A<<`m_B : (TQ & bitQF)? bf_A-bf_B : bf_A ;
bf_B <= st? B<<`m_B+`m_A-1: TQ? bf_B>>1 : bf_B ;
cb_tact <= st? 0 : TQ? cb_tact+1 : cb_tact ;
TQ <= T_end? 0 : st? 1 : TQ ;
sr_QF <= st? 0: TQ? sr_QF<<1 | bitQF : sr_QF ; //частное
ok_div <= T_end ;
end
endmodule
```

6.4 Преобразователь **HEX27_to_DEC8**

```
module HEX27_to_DEC8 (
        input[26:0]Dbin,      output wire [31:0]Ddec,
        input clk,            output reg [3:0]ptr_dig=0,//Указатель номера декадной цифры
        input st );
reg[3:0]D1dec=0;    reg[3:0]D2dec=0;    reg[3:0]D3dec=0;    reg[3:0]D4dec=0;
reg[3:0]D5dec=0;    reg[3:0]D6dec=0;    reg[3:0]D7dec=0;    reg[3:0]D8dec=0;

reg en_conv=0, q=0 ;
reg [27:0]rest=0;//Остаток
assign Ddec= {D8dec, D7dec, D6dec, D5dec, D4dec, D3dec, D2dec, D1dec} ;

wire d8 = (ptr_dig==8);       wire d7 = (ptr_dig==7);
wire d6 = (ptr_dig==6);       wire d5 = (ptr_dig==5);
wire d4 = (ptr_dig==4);       wire d3 = (ptr_dig==3);
wire d2 = (ptr_dig==2);       wire d1 = (ptr_dig==1);

wire[26:0]Nd=       d8?  10000000 :
                    d7?  1000000 :
                    d6?  100000 :
                    d5?  10000 :
                    d4?  1000 :
                    d3?  100 :
                    d2?  10 :
                    d1?  1 : 0 ;
wire [27:0]dx = rest-Nd ;//Разность между остатком и di (i=8,7,6,5,4,3,2,1)
wire z = dx[27] ;       // Знак разности
wire en_inc_dig  = en_conv &  q & !z ;//Разрешение инкремента декадной цифры
```

```
wire en_dec_ptr  = en_conv & !q &  z ;//Разрешение декремента указателя цифры

always @(posedge clk) begin
q <= st? 0 : en_conv? !q : q ;//q - для исключения необходимости восстановления остатка
en_conv <= st? 1 : (ptr_dig==0)? 0 : en_conv ;       //Разрешение преобразования
rest <= st? Dbin : en_inc_dig? dx : rest ;           //Остаток
ptr_dig <= st? 8: en_dec_ptr?  ptr_dig-1 : ptr_dig ;  //Указатель очередной декадной цифры
D8dec <= st? 0 : (d8 & en_inc_dig)? D8dec+1 : D8dec ;
D7dec <= st? 0 : (d7 & en_inc_dig)? D7dec+1 : D7dec ;
D6dec <= st? 0 : (d6 & en_inc_dig)? D6dec+1 : D6dec ;
D5dec <= st? 0 : (d5 & en_inc_dig)? D5dec+1 : D5dec ;
D4dec <= st? 0 : (d4 & en_inc_dig)? D4dec+1 : D4dec ;
D3dec <= st? 0 : (d3 & en_inc_dig)? D3dec+1 : D3dec ;
D2dec <= st? 0 : (d2 & en_inc_dig)? D2dec+1 : D2dec ;
D1dec <= st? 0 : (d1 & en_inc_dig)? D1dec+1 : D1dec ;
end
endmodule
```

6.5 Схема модуля преобразователя двоичной дроби в декадную дробь

```
module FB16_to_FD27 (              output wire [26:0] FDO,
                  input [15:0] FBI,      output reg[4:0]cb_tact=0,
                  input clk,             output reg EN_conv=0,
                  input st,              output wire ok_conv);
   parameter Dmax=27'h2FAF080 ;//50000000 , 99999999=27'h5f5e0ff
reg [15:0] sr_FBI =0 ; reg [31:0]sr_FBO =0 ;
reg [31:0]ACC = 0 ;
assign FDO = ACC[31:5] ;
assign ok_conv = (cb_tact==16) ;

always @ (posedge clk) begin
sr_FBI <= st? FBI : EN_conv? sr_FBI<<1 : sr_FBI ;
sr_FBO <= st? {Dmax,5'b00000} : EN_conv? sr_FBO>>1 : sr_FBO ;
ACC <= st? 0 : (EN_conv & sr_FBI[15])? ACC + sr_FBO : ACC ;
cb_tact <= st? 0 : EN_conv? cb_tact+1 : cb_tact ;
EN_conv <= st? 1 : ok_conv? 0 : EN_conv ;
end
endmodule
```

6.6 Схема модуль **BTN_BL** кнопки

```
module BTN_BL(     input btn,       output st,//Tst=Tclk
                   input clk,
                   input ce);  // ce1ms
reg q1=0, q2=0 ;
assign st= q1 & !q2 & ce ;
always @ (posedge clk) begin
q1 <= ce? btn : q1 ;    q2 <= ce? q1 : q2 ;
end
endmodule
```

6.7 Распределение портов схемы по контактным площадкам ПЛИС (*.ucf файл)
(не используемые в данной работе выводы ПЛИС закомментированы символом #)

```
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 "LED0"  LOC = "J14" ; #LD0
#NET "led1"  LOC = "J15" ; #LD1
#NET "led2"  LOC = "K15" ; #LD2
#NET "led3"  LOC = "K14" ; #LD3
#NET "led4"  LOC = "E17" ; #LD4
#NET "led5"  LOC = "P15" ; #LD5
#NET "led6"  LOC = "F4"  ; #LD6
#NET "led7"  LOC = "R4"  ; #LD7

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_P"  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 "SW7"  LOC = "R17"  ; #SWT7

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

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"  ;#| PULLUP
#NET "JB2"  LOC = "R18"  ;#| PULLUP
#NET "JB3"  LOC = "R15"  ;#
NET "JB4"  LOC = "T17"  ;#
#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

#NET "DAC<0>" LOC = "N8"  ;#GRN0
#NET "DAC<1>" LOC = "R9"  ;#RED0
#NET "DAC<2>" LOC = "T8"  ;#RED1
#NET "DAC<3>" LOC = "R8"  ;#RED2
```