Частотно-манипулированный сигнал FSK (Frequency Shift Key) с непрерывной фазой CPFSK (Continuous Phase FSK) используемый для передачи данных в автоматизированных системах управления технологическими процессами (АСУ ТП) по **HART** протоколу (Highway Addressable Remote Transducer)

Лабораторная работа №414_F95

Передача данных по HART протоколу используется в аналоговых АСУ ТП с токовой петлей 4-20 mA. Минимальному уровню управляющего сигнала соответствует ток 4 mA, а максимальному – ток 20 mA. Для передачи и приема цифровых данных HART протокол использует частотно-манипулированный сигнал с непрерывной фазой (Continuous Phase Frequency Shift Key) **CPFSK**. В дальнейшем будем подразумевать непрерывность фазы и использовать укороченную мнемонику - **FSK**.

Скорость передачи данных 1200 бит в секунду (1200 Bod). Логической «1» соответствует один полный период синусоиды с частотой 1200 Hz, а логическому «0» - два неполных периода синусоиды с частотой 2200 Hz. Пример временной диаграммы FSK сигала для двух бит (0 и 1) приведен на рис.1.

Цифровой сигнал ведущего с амплитудой 0.5 mA накладывается на ток управления 4-20 mA. Среднее значение FSK сигнала равно 0, а частота (1200 Hz) существенно выше ширины полосы частот (25 Hz) управляющего сигнала, поэтому FSK сигнал не влияет на аналоговое управление.

Ведомый отвечает «напряжением», т.е. модулирует падание напряжение на линии управления FSK сигналом с амплитудой 0.5 V.

Для передачи данных используется асинхронный UART протокол (Universal Asynchronous Receiver-Transmitter), где каждый, передаваемый байт (блок из 8-ми бит данных), дополняется нулевым старт битом и заканчивается единичным стоп битом. Длительности старт и стоп битов равны длительности одного бита. Данные передаются младшими битами вперед. Этот же протокол используется в COM порте персональных компьютеров.

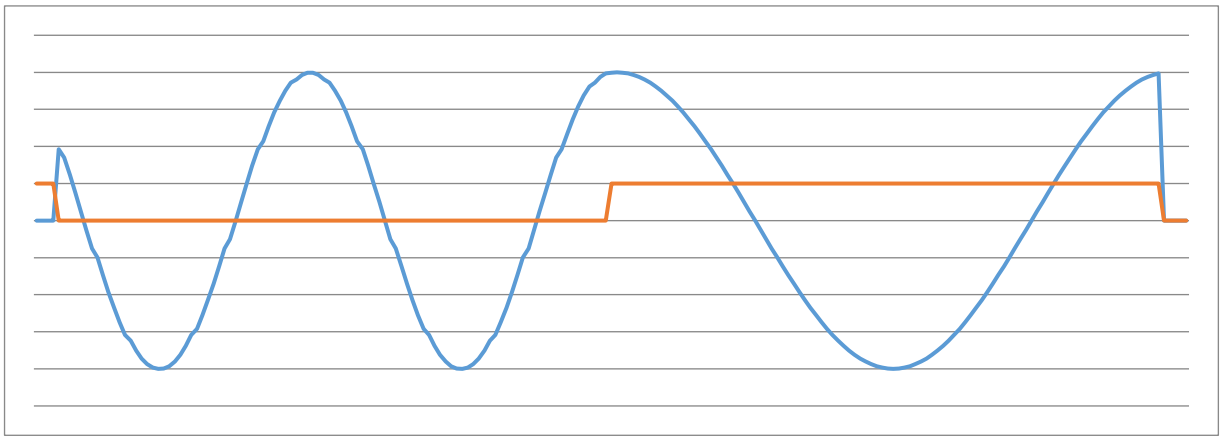


Рис.1 Пример временной диаграммы двух бит (0 и 1) FSK сигнала

В данном случае задача определения границ бит существенно упрощается благодаря тому, что в паузе между передачами сигала FSK нет (равен 0), а также тому, что передача начинается с нулевого старт бита. Если начинать прием с момента превышения абсолютным значением (модулем) FSK сигнала половины его амплитуды AMP (см. рис.2), то максимальная задержка T_d относительно начала бита

$$T_d = \frac{T_0}{6} = \frac{1}{6F_0} = \frac{1}{6 \cdot 2200\ Hz} \approx \frac{454.5}{6}\ \mu s \approx 75.75\ \mu s \approx 0.09 \cdot T_1 = 0.09 \cdot T_{bit},$$

где: T_0 и F_0 - период и частота сигнала бита «0»,

T_1 , T_{bit}– период сигнала «1» и длительность бита ($T_1 = T_{bit} = \frac{1}{1200Hz} = 833.33\mu s$).

Генератор импульсов границ тактов FSK_tact с периодом T_{bit} можно фазировать фронтом первого импульса OCD (Output Carry Detect) превышения порога REF=AMP/2 модулем сигнала FSK.

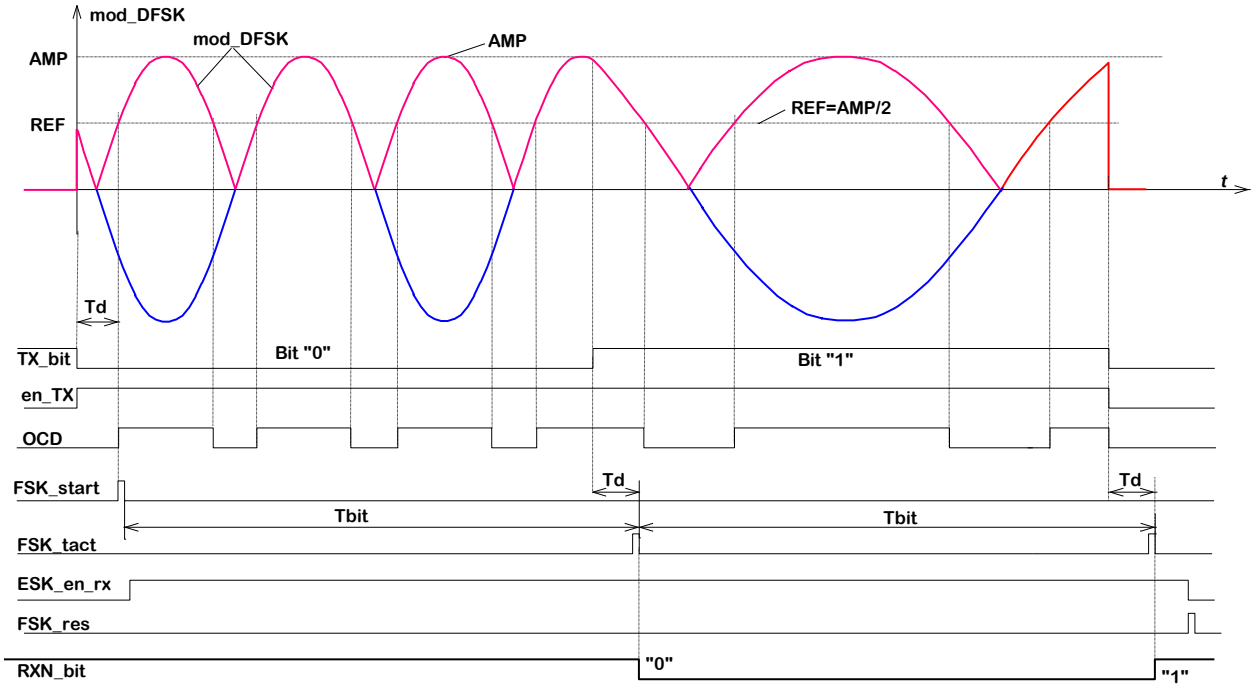


Рис.2 Пример временных диаграмм сигналов декодера бит FSK сигнала

Для автоматической установки порога REF на уровне половины амплитуды необходимо на интервале первого бита пиковыми детекторами определить амплитуду и сравнивать модуль mod_DFSK задержанного на длительность одного бита сигала DFSK с измеренным порогом. Среднее значение это половина суммы, амплитуда это половина разности пиковых значений сигала FSK, поэтому интервал измерения должен быть не меньше максимального периода синусоиды. Если первый бит это старт бит, т.е. «0», то задержка сигнала FSK должна быть не меньше T_0 .

Аналоговыми средствами реализовывать этот метод очень неудобно, прежде всего, из-за трудностей реализации неискажающей линии задержки сигнала. Неудобно также реализовывать пиковый детектор.

Если «оцифровать» входной сигнал при помощи аналого-цифрового преобразователя, то с цифровыми данными все необходимые операции решаются очень просто. Например, цифровая линия задержки - это модуль памяти, в который непрерывно записываются данные по линейно растущему с тактом Tce адресу Adr_wr, а считываются также по аналогично растущему адресу, но смещенному на Del, Adr_rd=Adr_wr – Del. Задержка выходных данных памяти относительно входных при этом равна Del*Tce. Здесь Tce – период дискретизации.

Пиковый детектор максимума, например, - это регистр в который с каждым тактом записываются данные, только при условии, что они превышают текущие данные регистра, в противном случае данные регистра сохраняются до “команды” сброса.

Сигнал FSK_en_rx устанавливается в 1 импульсами OCD (Output Carry Detect) превышения порога, а сбрасывается в 0 импульсом FSK_res в паузе после окончания пакета, т.е. если в течении половины T_{bit} нет импульсов OCD.

Относительно небольшая задержка определения границы бита (T_d < 10% длительности бита) позволяет очень просто определять значения бит, например, по количеству импульсов или по максимальной длительности импульсов OCD.

Более естественно и более надежно декодировать FSK сигнал по амплитудам первой и второй гармоник. При точном определении границ бит амплитуда первой гармоники сигнала bit1 равна амплитуде FSK, а амплитуда второй гармоники равна нулю. Частота сиусоиды bit0 2200 Hz примерно в 2 рада больше частоты сиусоиды bit1 1200 Hz, поэтому аплитуда второй гармоники сигнала bit0 немного меньше, а амплитуда первой гармоники существенно меньше амплитуды FSK сигнала. Если на каждом интервале Tbit вычислять амплитуды F1amp перовой и F2amp второй гармоник и сравивать их в конце Tbit, то при F1amp > F2amp bit=1, а при F1amp < F2amp bit=0. Небольшое смещение на $T_d \leq 0.09 \cdot T_{bit}$ интервала вычисления гармоник относительно интервала бита слабо влияет на отношение амплитуд гармоник.

Реальная часть амплитуды n-й гармоники

$$Y_{re}(n) = \frac{1}{N}\sum_{k=0}^{N-1} FSK(k)\cos(2\pi nk/N).$$

Мнимая часть амплитуды n-й гармоники

$$Y_{im}(n) = \frac{1}{N}\sum_{k=0}^{N-1} FSK(k)\sin(2\pi nk/N).$$

Амплитуда n-й гармоники

$$modY(n) = \sqrt{Y_{re}(n)^2 + Y_{im}(n)^2}.$$

В данном случае, вычисления амплитуд только двух гарионик, нет необходимости использовать двоично кратное число точек *N* и метод быстрого преобразования Фурье.

1. Модуль измерения амплитуды n-й гармоники

```
`include "CONST.v"
module DFTn (
    input[11:0]X, output wire [10:0]modY,    //Амплитуда гармоники
    input ce,     output wire [17:0]Wre,     //Реальная часть поворотного множителя
    input clk,    output wire [17:0]Wim,     //Мнимая часть поворотного множителя
    input st,     output wire [11:0]WXre,    //Реальная часть произведения X(k) на W(k)
    input ce_bit, output wire [11:0]WXim,    //Мнимая часть произведения X(k) на W(k)
    input [6:0]n, output reg [18:0]Yre=0,    //Аккумулятор реальных частей произведений
                  output reg [18:0]Yim=0,    //Аккумулятор мнимых частей произведений
                  output wire [35:0]SQreim,  //Сумма квадратов Yre и Yim
                  output wire [6:0]k,        //Адрес поворотного множителя k=|k*n|modNP
                  output reg [7:0]ACC=0);    //Аккумулятор адреса поворотных множителей

assign k =(ACC>=`NP)? ACC-`NP : ACC[6:0] ; //k=ACC по модулю NP
//--Умножение на поворотный множитель Wre=cos(2*pi*n*k/N)
MULT18x12 DD1 ( .X(X),     .WX(WXre),
                .W(Wre),
                .clk(clk));
//--Умножение на поворотный множитель Wim=sin(2*pi*n*k/N)
```

```
MULT18x12 DD2 (  .X(X),       .WX(WXim),
                 .W(Wim),
                 .clk(clk));
//--Таблица реальной части Wre поворотных множителей
ROM_18x100_Wre DD3 (   .k(k), .Wre(Wre));

//--Таблица мнимой части Wim поворотных множителей
ROM_18x100_Wim DD4 ( .k(k), .Wim(Wim));

reg st_SQRT=0 ; //Старт извлечения корня
reg tce_bit=0 , tst=0;
reg [18:0] bfYre=0 ; reg [18:0] bfYim=0 ; // Суммы Yre и Yim в конце Tbit
wire CO = (ACC>=`NP) & ce ;  //Сигнал переполнеия аккумулятора

always @ (posedge clk) begin
tce_bit <= ce_bit ;     tst <= st ; //Задежка на Tclk=20ns=1/50Mhz
ACC <= (ce_bit)? 0 : CO? (ACC+n)-`NP : ce? ACC+n : ACC;
Yre <= (tst | tce_bit)? {{7{WXre[11]}},WXre} : ce? Yre+{{7{WXre[11]}},WXre} : Yre ;
Yim <= (tst | tce_bit)? {{7{WXim[11]}},WXim} : ce? Yim+{{7{WXim[11]}},WXim} : Yim ;
bfYre <= ce_bit? Yre : bfYre ;
bfYim <= ce_bit? Yim : bfYim ;
st_SQRT <= tce_bit ;
end

wire S_Yre=bfYre[18] ; wire S_Yim=bfYim[18] ;    //Знаки сумм
wire [17:0]modYre = S_Yre? -bfYre : bfYre ;      //Модуль суммы bfYre
wire [17:0]modYim = S_Yim? -bfYim : bfYim ;      //Модуль суммы bfYim
wire [35:0]SQre = modYre[17:0]*modYre[17:0] ;    //Квадрат модуля суммы bfYre
wire [35:0]SQim = modYim[17:0]*modYim[17:0] ; //Квадрат модуля суммы bfYim
assign SQreim = SQre+SQim ;                      //Сумма квадратов сумм Yre и Yim
wire [17:0]SQRT ;
assign modY = (SQRT*1310)>>16 ;   //Деление на 50, а не на 100
//--Модуль извлечения квадратного корня
SQRT_BL DD5 (.X(SQreim),       .Q(SQRT),
             .st(st_SQRT),
             .clk(clk));
endmodule
```

1.1 Модуль параметров CONST.V

```
`define Fclk  50000000  //Частота сигнала синхронизации
`define Fbit       1200  //Частота передачи бит
`define NP          100  //Число точек синусоиды сигнала FSK
`define Fd    `Fbit*`NP  //Частота дискретизации
`define F1ce        120  //Частота дискретизации F1, bit=1
`define F0ce        220  //Частота дискретизации F0, bit=0
`define SH         2048  //Смещение сигнала FSK
`define Amin         50  //Минимальная амплитуда
`define ND          `NP  //Число тактов задержки сигнала FSK
`define Nbit          8  //Число бит в UART кадре
```

1.2 Модуль умножения X на поворотный множитель W

```
module MULT18x12 (
        input [11:0] X,         output wire[11:0] WX, //WX=X*W/2^16
        input [17:0] W,
        input clk);
reg[27:0] P=0 ;         //Регистр произведения P=X*W
wire SX=X[11] ;         //Знак X
wire SW=W[17] ;         //Знак поворотного множителя W
wire SP= SX^SW ;        //Знак произведения
wire [10:0]mod_X = SX? -X : X ;         //Модуль X
wire [16:0]mod_W = SW? -W : W ;         //Модуль поворотного множителя W
wire [27:0]mod_P = mod_X*mod_W; //Произведение модулей X и W
assign WX = P>>16 ;                     //Деление P[27:0] на 2^16

always @ (posedge clk) begin
P =  SP? -mod_P : mod_P ;       //Восстановление знака произведения
end
endmodule
```

1.3 Модуль памяти реальной части поворотных множителей

```
module ROM_18x100_Wre (
            input [6:0] k,          output wire[17:0] Wre);
reg [17:0]ROM[100:0] ;
assign Wre = ROM[k] ; //Таблица Wre (слайсовая память)
initial  //Инициализация модуля памяти из файла Wr100.txt
$readmemh ("Wr100.txt", ROM, 0, 100);
endmodule
```

1.4 Модуль памяти мнимой части поворотных множителей

```
module ROM_18x100_Wim (
            input [6:0] k,          output wire[17:0] Wim);
reg [17:0]ROM[100:0] ;
assign Wre = ROM[k] ; //Таблица Wim (слайсовая память)
initial  //Инициализация модуля памяти из файла Wi100.txt
$readmemh ("Wi100.txt", ROM, 0, 100);
endmodule
```

Модули памяти поворотных множителей инициализируются текстовыми файлами Wr100.txt и Wi100.txt (приложения 6.8,9), которые должны быть папке проекта ISE14 лабраторной работы.

1.5 Модуль извлечения корня квадратного

```
`define m 18 //Число разрядов результата
module SQRT_BL (    input [2*`m-1:0] X,  output reg[`m-1:0]Q=0,
                    input st,            output reg en=0,
                    input clk);

wire [2*`m-1:0]M=Q*Q ;
wire DI =(M<=X);
//---Регистр последовательного приближения---
reg[`m:0] T=0 ;   //Регистр сдвига импульсов T
integer i ;         //Индекс цикла for
```

```
always @ (posedge clk)  begin
T <= st? 1<<`m :en? T>>1 : T ; //Сдвиг импульсов T вправо
Q[`m-1] <= st? 1 : T[`m]? DI : Q[`m-1] ; //Загрузка старшего бита выходного регистра Q
for (i=`m-2 ; i>= 0; i=i-1)                    // Цикл for
Q[i] <= st? 0 : T[i+2]? 1 : T[i+1]? DI : Q[i] ;  //Загрузка очередного бита регистра Q
en<= st? 1 : (T[0] &en)? 0 :en ;      //Интервал  последовательного приближения
end
endmodule
```

1.6 Схема моделирования DFTn измерителя амплитуды n-й гармоники

```
module Test_DFTn(    output wire S,          //Знак синусоиды
        input clk,          output wire[11:0] SIN,//Синусоида
        input st,           output wire ce_tact,  //Период дискретизации DFT
        input[7:0]M,        output wire ce_sd,    //Период дискретизации синусоиды
                            output wire ce_SIN,  //Период синусоиды
                            output wire ce_bit,   //Период бит
                            output wire [10:0]modY,//Амплитуда гармоники
                            output wire [6:0]k,   //Адрес поворотного множителя, k=|k*n|modNP
                            output wire D,        //Бит данных
                            output wire [6:0]n);  //Номер гармоники,

//--Таймер (генератор тактов)
TIMER DD1 (.clk(clk),  .ce(ce_tact), //(Частота ce_tact 100*Fbit=100*2200Hz=220kHz)
                .st(st),      .ce_bit(ce_bit)); //(Частота ce_bit Fbit=2200Hz)
//--Генераторсигнала синусоиды
//--M - множитель амплитуды (при M=128 Amp=2000)
//--D - управление частотой (при D=0 Fsin=2200Hz, при D=1 Fsin=1200Hz)
Gen_FSK_SIN DD2 (
        .clk(clk),      .S(S),//S=1    sin>0
        .D(D),          .SIN(SIN),
        .Mamp(M),       .ce_SIN(ce_SIN), //
                        .ce_sd(ce_sd)); //
//--Вычислитель амплитуды n-й гармоники
DFTn DD3 (
        .X(SIN),            .modY(modY),
        .ce(ce_tact),       .k(k),
        .clk(clk),
        .st(st),
        .ce_bit(ce_bit),
        .n(n));
//--Генератор данных D=0,1 и номера гармоники  n=1,2,...,99
Gen_Dn DD4 (.st(ce_bit),   .D(D),
                .clk(clk),      .n(n));

endmodule
```

1.7 Модуль таймера

```
`include "CONST.v"
module TIMER(
        input clk,  output wire ce,
```

```
        input st,    output wire ce_bit);

parameter M=50000 ; //Емкость аккумулятора синтезатора частоты
reg [15:0]ACC_ce=0 ;
assign ce = (ACC_ce+`F1ce>=M) ;
reg [6:0] cb_bit=0 ;
assign ce_bit = (cb_bit==`NP) & ce ;

always @ (posedge clk) begin
ACC_ce <= st? 0 : ce? (ACC_ce+`F1ce)-M : ACC_ce+`F1ce ;
cb_bit <= (st | ce_bit)? 1 : ce? cb_bit+1 : cb_bit ;
end

endmodule
```

1.8 Модуль генератора «синусоиды»

```
`include "CONST.v"
module Gen_FSK_SIN (output reg S=1,       //S=1   sin>0, знак SIN
        input clk,          output wire[11:0]SIN, //SIN = S? Y : -Y
        input D,            output wire ce_SIN,  //Период SIN, Tce_SIN=NP*Tce_sd
        input[7:0]Mamp, output wire ce_sd);   //Сигнал дискретизации SIN

// D - бит управления частотой FSK (D=0 Fsin=2200Hz, D=1 Fsin=1200Hz)
// Mamp[7:0] - Регулятор амплитуды SIN, AMPsin=NA*Mamp/128 (NA=2000)
parameter Macc = 50000 ;   //Емкость аккумулятора
parameter Xmin=0 ;         //Минимальное значение "пилы"
parameter Xmax=`NP/4 ;     //Xmax=`NP/4=250,
wire [10:0] Y ;            //Модуль SIN
reg [7:0] X=0 ;            //Регистр "пилы"
reg up=1 ;                 //Триггер направления счета
assign ce_SIN = (X==Xmin) & !S ; //Период SIN, Tce_SIN=NP*Tce_sd
wire [19:0]MY=Y*Mamp ; //Умножение на Mamp
wire[10:0] AY = MY >> 7 ; //Деление на 128

//--Синтезатор частоты дискретизации SIN
reg [15:0]ACC=0  ; //Аккумулятор синтезатора частоты
wire [15:0]Xf= D? `F1ce : `F0ce ; //Накапливаемое число Xf=120 или Xf=220
assign ce_sd = (ACC+Xf>=Macc) ;     /*Сигнал переполнения аккумулятора или
сигнал дискретизации SIN. Частота Fce_sd=100*Fsin=Xf*50MHz/Macc */
always @ (posedge clk) begin
ACC <= ce_sd? (ACC+Xf)-Macc : (ACC+Xf) ; //
end
//--Таблица значений четверти периода синусоиды
//Y=NA*sin(2*pi*X/NP), NA=2000-амплитуда, NP=100 - число точек
assign Y =     (X==0)? 0  :
               (X==1)? 126 :
               (X==2)? 251 :
               (X==3)? 375 :
               (X==4)? 497 :
               (X==5)? 618 :
               (X==6)? 736 :
```

```
            (X==7)? 852 :
            (X==8)? 964 :
            (X==9)? 1072 :
            (X==10)? 1176 :
            (X==11)? 1275 :
            (X==12)? 1369 :
            (X==13)? 1458 :
            (X==14)? 1541 :
            (X==15)? 1618 :
            (X==16)? 1689 :
            (X==17)? 1753 :
            (X==18)? 1810 :
            (X==19)? 1860 :
            (X==20)? 1902 :
            (X==21)? 1937 :
            (X==22)? 1965 :
            (X==23)? 1984 :
            (X==24)? 1996 :
            (X==25)? 2000 : 0 ;

assign SIN = S? AY: -AY;

always @ (posedge clk) if (ce_sd) begin
X<= up? X+1 : X-1  ;     //Генератор "пилы"
up <= (X==Xmin+1)? 1 : (X==Xmax-1)? 0 : up ;
S <= ((X==Xmin+1) & !up)? !S : S ;
end
endmodule
```

Частота «синусоиды» Fsin=Fce/100. Для бит=0 Fce=220 kHz, а для бит=1 Fce=120 kHz. Частоты дискретизации Fce не кратны частоте синхронизации Fclk=50MHz (50000000/120000=416,6(6), 50000000/220000=227,27(27)), поэтому нельзя воспользоваться целочисленным делителем частоты.

Синтезатор частоты позволяет получить точное значение средней частоты Fce 220 kHz или 220 kHz при Fclk=50000 kHz. Синтезатор частоты это аккумулятор (накапливающий сумматор), а средняя частота сигнала ce_sd, переполнения аккумулятора, пропорциональна накапливаемому числу Xf и обратно пропорциональна емкости аккумулятора Macc, Fce=Fclk*Xf/Macc. Если емкость аккумулятора Macc=50000, то при Xf=120 Fce=120kHz, а при Xf=220 Fce=220kHz.

Если число Xf не кратно емкости Macc то сигнал переполнения аккумулятора не периодический. Интервалы времени между соседними переполнениями принимают два значения [Macc/Xf] и [Macc/Xf]+1, где [….] – целая часть числа. В данном случае даже для Xf=220 получается относительно небольшие флюктуации (1/227~0,44%) периода сигнала дискретизации «синусоиды», которые практически не сказываются на качестве сигнала FSK.

Непрерывность фазы получается потому, что для «синусоиды» бита «1» и бита «0» используется один и тот же генератор, в котором на границах разных бит скачком меняется частота, т.е. крутизна изменения фазы а не фаза.

Амплитуда синусоиды AMP=2000*Mamp/128 (128>=Mamp>=1) пропорциональна множителю Mamp.

Амплитуда Uamp напряжения на выходе цифроаналогового преобразователя Uamp=2000*Mamp/128 mV

1.9 Модуль генератора данных D и номера гармоники n

```
module Gen_Dn(
   input st,         output wire D,
   input clk,        output wire [6:0] n);

reg [2:0]cb_tact = 0 ;
assign D = cb_tact[0] ;
assign n =  (cb_tact==0)? 1 : //99
            (cb_tact==1)? 2 : //98
            (cb_tact==2)? 1 : //99
            (cb_tact==3)? 2 : //98
            (cb_tact==4)? 2 : //98
            (cb_tact==5)? 1 : //99
            (cb_tact==6)? 2 : //98
                          1 ; //99
always @ (posedge clk) begin
cb_tact <= st? cb_tact+1 : cb_tact ;
end
endmodule
```

1.10 Данные сигналов задания на моделирования tf_Test_DFTn

```
always begin clk=0; #10; clk=1; #10; end
    initial begin
                st = 0; Mamp = 128; //
#1000000;       st = 1; Mamp = 128; //Множитель амплитуды сигнала FSK
#20;            st = 0;
    end
```

2. Приемник FSK сигнала

2.1 Детектор FSK сигнала

```
`include "CONST.v"
module DET_FSK(             output wire [11:0] DFSK_SH,//Задержанный сигнал FSK_SH
        input [11:0]FSK_SH, output wire OCD, // (OCD=(mod_DFSK>=AMP/2))
        input clk,          output wire [11:0] AMP,     // Текущая амплитуда
                            output wire [12:0] bf_SH,   // Среднее  смещение
                            output wire FSK_tact,       // Период FSK_tact=833us
                            output reg [6:0]cb_tact=0,  // Счетчик такта
                            output wire FSK_start,      // Старт приема бит
                            output reg FSK_en_rx=0,     // Интервал приема бит
                            output reg[3:0]cb_rx_bit=0, //Счетчик принятых бит
                            output wire ce_Fd,          //Сигнал дискретизации
                            output wire FSK_res,        //Импульс "сброса" в паузе
                            output wire[10:0]F1_AMP,    //Амплитуда первой гармоники
                            output wire[10:0]F2_AMP,    //Амплитуда второй гармоники
                            output reg RX_bit=1,        //Декодированный бит по DFT
                            output wire ok_rx_bit);
//--Измеритель смещения и амплитуды--------
Mes_AMP_SH DD1(
```

```
            .FSK_SH(FSK_SH),     .DFSK_SH(DFSK_SH),//Задержанный FSK_SH
            .clk(clk),           .AMP(AMP),        // Текущая амплитуда
            .res(FSK_res),       .bf_SH(bf_SH),    // Среднее  смещение
                                 .ce_Fd(ce_Fd),    // Сигнал дискретизации
                                 .OCD(OCD));       // Сигнал превышения порога
//-----------------------------------------
wire [11:0]DFSK = DFSK_SH-bf_SH ; //Вычитание среднего  смещения
assign FSK_tact = (cb_tact==`NP) & ce_Fd ; //
reg tFSK_tact=0 ;
reg tOCD=0 ; //
wire front_OCD = (OCD & !tOCD);
assign FSK_start = front_OCD & !FSK_en_rx & ce_Fd ;//

reg [6:0]cb_FSK_res=0; //Счетчик паузы
assign FSK_res = ((cb_FSK_res==`NP/2) & FSK_en_rx) & ce_Fd ;

always @ (posedge clk) if (ce_Fd) begin
tOCD <= OCD ;
FSK_en_rx <= front_OCD? 1 : FSK_res? 0 : FSK_en_rx ;
cb_rx_bit <= FSK_start? 0 : (FSK_en_rx & FSK_tact)? cb_rx_bit+1 : cb_rx_bit ;
cb_FSK_res <= OCD? 0 : FSK_en_rx? cb_FSK_res+1 : cb_FSK_res ;
cb_tact <= (FSK_tact | FSK_start)? 1 : cb_tact+1 ;
tFSK_tact <= FSK_tact ;
end
wire st_DFT = tFSK_start | (FSK_en_rx & FSK_tact) ;
assign ok_rx_bit = tFSK_tact & FSK_en_rx & ce_Fd;

//--Измеритель амплитуды первой гармоники
wire [6:0]n1= 1 ; //99
DFTn DD2 (  .X(DFSK),          .modY(F1_AMP),
            .ce(ce_Fd),
            .clk(clk),
            .st(st_DFT),
            .ce_bit(FSK_tact),
            .n(n1));
//--Измеритель амплитуды второй гармоники
wire [6:0]n0= 2 ;//98
DFTn DD3 (  .X(DFSK),          .modY(F2_AMP),
            .ce(ce_Fd),
            .clk(clk),
            .st(st_DFT),
            .ce_bit(FSK_tact),
            .n(n0));

always @ (posedge clk) begin
RX_bit <= FSK_res? 1 : ok_rx_bit? (F1_AMP>F2_AMP) : RX_bit ;
end
endmodule
```

2.2 Измеритель смещения и амплитуды сигнала FSK_SH

```
`include "CONST.v"
```

```
module Mes_AMP_SH(
        input [11:0]FSK_SH,     output wire [11:0] DFSK_SH, //Задержанный сигнал FSK
        input clk,              output reg [12:0]bf_SH=`SH,   //Реверсивный счетчик
        input res,              output wire [11:0] AMP,       //Текущая амплитуда
                                output reg OCD=0,             //Превышение порога
                                output wire ce_Fd);           //Сигнал дискретизации

parameter dREF=`Amin/4 ; //Гистерезис компаратора
wire [11:0]DFSK = DFSK_SH-bf_SH ; //Вычитание среднего смещения
wire [11:0]mod_DFSK = DFSK[11]? -DFSK : DFSK; //Абсолютное значение (модуль) DFSK
wire [11:0]REF = AMP>>1 ; //Деление AMP на 2 (REF=AMP/2)
reg [10:0] cb_ce =0 ; //Tce=1/Fce
assign ce_Fd = (cb_ce==`Fclk/(`Fbit*`NP)) ; //Сигнал дискретизации
reg [6:0] Adr_wr=0 ;
wire [6:0] Adr_rd = Adr_wr-`ND ; //Задержка на Tbit=`ND*Tce=T1

always @ (posedge clk) begin
cb_ce <= (ce_Fd)? 1 : cb_ce+1 ; //
Adr_wr <= ce_Fd? Adr_wr+1 : Adr_wr ;
end
//--Модуль памяти для задержки сигнала FSK
MEM12x128 DD2 ( .clk(clk),              .DO(DFSK_SH),
                .we(ce_Fd),
                .DI(FSK_SH),
                .Adr_wr(Adr_wr),
                .Adr_rd(Adr_rd));
reg [11:0]PIC_max = 0; //Регистр пикового детектора максимума сигнала FSK
reg [11:0]PIC_min = 4095; //Регистр пикового детектора минимума сигнала FSK
wire [12:0]SH = (PIC_max+PIC_min)>>1 ; //Полусумма
assign AMP= (PIC_max-PIC_min)>>1 ; //Полуразность
//--Получение максимума и минимума сигнала FSK
always @ (posedge clk)  begin //
PIC_max <= res? 12'h000 : ((FSK_SH>PIC_max) & ce_Fd)? FSK_SH : PIC_max ;
PIC_min <= res? 12'hFFF : ((FSK_SH<PIC_min) & ce_Fd)? FSK_SH : PIC_min ;
end

always @ (posedge clk) if (ce_Fd) begin
//--Реверсивный следящий измеритель смещения
bf_SH <= (SH>bf_SH)? bf_SH+1 : (SH<bf_SH)? bf_SH-1 : bf_SH ;
//--Компаратор с гистерезисом
OCD <=  ((mod_DFSK>=REF+dREF) & (mod_DFSK>`Amin))? 1 :
          (mod_DFSK<=REF-dREF)? 0 : OCD ;
end
endmodule
```

2.3 Модуль памяти для задержки сигнала FSK

```
module MEM12x128(
          input clk,          output reg [11:0] DO,
          input we,
          input [11:0] DI,
          input [6:0] Adr_wr,
```

```
        input [6:0] Adr_rd);

reg [11:0]MEM[127:0] ;
//assign DO = MEM[Adr_rd] ; //Слайсовая память
initial //Инициализация модуля памяти из файла init_MEM12x64.txt
$readmemh ("init_MEM12x128.txt", MEM, 0, 127);
always @ (posedge clk) begin
MEM[Adr_wr] <= we? DI : MEM[Adr_wr] ;
DO <= MEM[Adr_rd] ;//Блочная память
end
endmodule
```

Модуль памяти **MEM12x128** инициализируется текстовым файлом init_MEM12x64.txt. В этом текстовом файле 128 строк 800 (смещение FSK сигнала в HEX формате).

2.4 Модуль генератора FSK байта

```
`include "CONST.v"
module Gen_FSK_byte (
        input clk,          output wire ce_bit,       //Границы бит
        input st,           output reg en_tx=0,       //Интервал передачи
        input [7:0]dat,     output wire TXD,          //Биты данных
        input [7:0]Mamp, output wire [11:0]FSK_SH, //Пакет смещенных  синусоид
                            output wire S,            //Знак синусоиды
                            output wire ce_SIN,       // Период синусоиды
                            output wire ce_sd );      //Период дискретизации синусоиды

//--Mamp[7:0] - Регулятор амплитуды SIN, AMPsin=NA*Mamp/128 (NA=2000)
wire [11:0]SIN; //Синусоида
assign FSK_SH = en_tx? SIN+`SH : `SH ; //Пакет смещенных синусоид
parameter Macc=50000 ; //Емкость аккумулятора синтезатора частоты
reg [15:0]ACC_ce=0 ; //Аккумулятор синтезатора частоты сигнала ce
wire ce = (ACC_ce+`Fce>=Macc) ; //Сигнал переполнения аккумулятора

reg [6:0] cb_tact=0 ; //Счетчик такта
assign ce_bit = (cb_tact==`NP) & ce ; // Период ce_bit 1/1200Hz
reg [3:0] cb_bit=0;         //Счетчик бит
wire ce_end = (cb_bit==`Nbit+1) & ce_bit ; // Конец кадра
wire T_dat = (cb_bit>0) & (cb_bit<(`Nbit+1)) ; // Интервал данных
reg [7:0] sr_dat =0 ;  //Регистр сдвига данных
assign TXD = ((cb_bit==0) & en_tx)? 0 : T_dat? sr_dat[0] : 1 ; //Биты данных

always @ (posedge clk) begin
ACC_ce <= st? 0 : ce? (ACC_ce+`Fce)-Macc : ACC_ce+`Fce ;
cb_tact <= (st | ce_bit)? 1 : ce? cb_tact+1 : cb_tact ;
cb_bit <= st? 0 : (ce_bit & en_tx)? cb_bit+1 : cb_bit ;
en_tx <= st? 1 : ce_end? 0 : en_tx ;
sr_dat <= st? dat : (T_dat & ce_bit)? sr_dat>>1 : sr_dat ;
end
//--Генератор синусоиды: TXD=0 Fsin=2200Hz, TXD=1 Fsin=1200Hz
Gen_FSK_SIN DD1 (
        .clk(clk),          .S(S),                  //S=1   sin>0
```

```
        .D(TXD),        .SIN(SIN),          //Синусоида
        .Mamp(Mamp),    .ce_SIN(ce_SIN),    //Период синусоиды
                        .ce_sd(ce_sd));     //Период дискретизации синусоиды
```

2.5 Схема моделирования модуля измерителя амплитуды и смещения

```
module Test_Mes_AMP_SH(
//--Сигналы   генератора Gen_FSK_byte
        input clk,              output wire [11:0]FSK_SH,//Сигнал FSK
        input st,               output wire S,              //Знак SIN
        input [7:0]dat,         output wire ce_SIN,         //Границы периода SIN
        input [7:0]Mamp,        output wire ce_tx_bit,      //Границы бит
                                output wire en_tx,          //Интервал передачи
                                output wire TXD,            //Передаваемые данные
//--Сигналы измерителя смещения и амплитуды
                                output wire [11:0] DFSK_SH, //Задержанный сигнал FSK
                                output wire [12:0] SH,      //Смещение
                                output wire [11:0] AMP,     //Амплитуда
                                output wire OCD             //Сигнал превышения порога
                                );
//--Генератор  Gen_FSK_byte
Gen_FSK_byte DD1 (
        .clk(clk),          .FSK_SH(FSK_SH),
        .st(st),            .S(S),
        .dat(dat),          .ce_SIN(ce_SIN),
        .Mamp(Mamp),        .ce_bit(ce_tx_bit),
                            .en_tx(en_tx),
                            .TXD(TXD),
                            .ce_sd(ce_sd));
//--Измеритель смещения и амплитуды--------
Mes_AMP_SH DD2(
            .FSK_SH(FSK_SH), .DFSK_SH(DFSK_SH),//Задержанный сигнал FSK_SH
            .clk(clk),      .bf_SH(bf_SH),      //Смещение
            .res(st),       .AMP(AMP),          //Амплитуда
                            .ce_Fd(ce_Fd),      //Синал дискретизации
                            .OCD(OCD));
endmodule
```

2.6 Данные для моделирования измерителя амплитуды и смещения

```
always begin clk=0; #10; clk=1; #10; end
    initial begin
                st = 0; Mamp = 0;     dat = 0;
#1000000;       st = 1; Mamp = 128;
#20;            st = 0;
#3000000;       st = 1; Mamp = 64;
#20;            st = 0;
#3000000;       st = 1; Mamp = 32;
#20;            st = 0;
#3000000;       st = 1; Mamp = 16;
#20;            st = 0;
#3000000;       st = 1; Mamp = 8;
```

```
#20;            st = 0;
    end
endmodule
```

2.7 Схема моделирования детектора FSK сигнала

```
module Test_DET_FSK(
//-----Сигналы Gen_FSK_byte-------------------
        input clk,              output wire [11:0]FSK_SH,//FSK сигнал
        input st,               output wire S,          //Знак SIN
        input [7:0]dat,         output wire ce_SIN,  //Период SIN
        input [7:0]Mamp,        output wire en_tx,     //Интервал передачи
                                output wire TX_bit,  //Бит передатчика
                                output wire ce_tx_bit,//Границы бит передатчика
//-----Сигналы детектора FSK-----------------
                output wire [11:0] DFSK_SH,//Задержанный сигнал FSK_SH
                output wire OCD,            // OCD=(mod_DFSK>=AMP/2)
                output wire [11:0] AMP,     //Амплитуда
                output wire [12:0] bf_SH,   //Смещение
                output wire FSK_tact,       //Период FSK_tact Ttact=1/F1=833us
                output wire [6:0]cb_tact,   //Счетчик такта
                output wire FSK_start,      //Старт приема бит
                output wire FSK_en_rx,      //Интервал приема бит
                output wire[3:0]cb_rx_bit,  //Счетчик принятых бит
                output wire FSK_res,        //Импульс "сброса" в паузе
                output wire[10:0]F1_AMP, //Амплитуда первой гармоники
                output wire[10:0]F2_AMP, //Амплитуда второй гармоники
                output wire RX_bit,         //Декодированный бит по DFT
                output wire ok_rx_bit);

//--Генератор  Gen_FSK_byte
Gen_FSK_byte DD1 (
        .clk(clk),          .FSK_SH(FSK_SH),
        .st(st),            .S(S),
        .dat(dat),          .ce_SIN(ce_SIN),
        .Mamp(Mamp),        .ce_bit(ce_tx_bit),
                            .en_tx(en_tx),
                            .TXD(TX_bit));
//--Детектор FSK
DET_FSK DD2 (.DFSK_SH(DFSK_SH),     //Задержанный сигнал FSK_SH
        .FSK_SH(FSK_SH),    .OCD(OCD), // OCD=(mod_DFSK>=AMP/2)
        .clk(clk),                  .AMP(AMP),          //Амплитуда
                                    .bf_SH(bf_SH),      //Буфер смещения
                                    .FSK_tact(FSK_tact),    //Период FSK_tact= 833us
                                    .cb_tact(cb_tact),      //Счетчик такта
                                    .FSK_start(FSK_start),  //Старт приема бит
                                    .FSK_en_rx(FSK_en_rx),//Интервал приема бит
                                    .cb_rx_bit(cb_rx_bit),  //Счетчик принятых бит
                                    .FSK_res(FSK_res),      //Импульс "сброса" в паузе
                                    .F1_AMP(F1_AMP),    //Амплитуда первой гармоники
                                    .F2_AMP(F2_AMP),    //Амплитуда второй гармоники
                                    .RX_bit(RX_bit),        //Декодированый бит по DFT
```

```
.ok_rx_bit(ok_rx_bit)  );

endmodule
```

2.8 Данные задания на моделирования tf_Test_DET_FSK

```
always begin clk=0; #10; clk=1; #10; end
    initial begin
                st = 0; dat = 0;          Mamp = 0; //
#1000000;       st = 1; dat = 8'h0F;      Mamp = 16; //Mamp=1,2,…128
#20;            st = 0;
    end
```

4. Задание к допуску (стоимость 2)

4.1 Начертить в тетради временные диаграммы сигналов рис.2.

4.2 Переписать в тетрадь схему модуля **DFTn** измерителя амплитуды n-й гармоники.

4.3 Переписать в тетрадь схему модуля **Mes_AMP_SH** измерения смещения и амплитуды FSK сигнала.

Таблица вариантов параметров

№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Amin	51	55	65	60	45	59	64	75	90	100	60	68	98	120	35	40	50	70	80	130
DAT	F1	F2	F3	F4	F5	F6	F7	F8	F9	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA

5. Задание к выполнению (стоимость 4)

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 Lab414_F95, для ПЛИС используемой в макете NEXYS2 (Spartan3E, XC3S500E, FG320, XST (VHDL/Verilog), Isim(VHDL/Verilog), Store all values).

Verilog модули создаются и синтезируются в режиме Implementation. New Source\Verilog Module\”name”. Все операции доступны только для модуля, помещенного на вершину проекта (Set as Top Module).

Моделируются модули в режиме Simulation. Для моделируемого модуля необходимо создавать задание на моделирование New Source\Verilog Test Fixture\”name”

Для всех созданных модулей выполнить синтез (Synthesize - XST), исправить ошибки (Errors), обратить внимание на предупреждения (Warnings).

5.1 Создать Verilog модуль **DFTn** и, входящие в его состав, модули: комплексного умножителя **MULT18x12**, модулей памяти **ROM_18x100_Wre**, **ROM_18x100_Wim** поворотных множителей Wre, Wim и модуля извлечения квадратного корня **SQRT_BL**.

Для модулей памяти поворотных множителей создать текстовые файлы Wr100.txt и Wi100.txt (приложения 7.1,2), которые должны быть папке проекта Lab414_F95 лабраторной работы.

5.2 Создать Verilog модуль схемы **Test_DFTn** и входящие в ее состав модули: **TIMER**, **Gen_FSK_SIN** – генератора синусоиды и **Gen_Dn** - генератора **D** и **n**.

5.3 Создать для схемы **Test_DFTn** Verilog Test Fixture **tf_Test_DFTn** задания на моделирование. Установить необходимые данные входных сигналов (см. п. 1.10).

5.4 Переключиться в режим Simulation. При Simulation Run Time=12ms провести моделирование **tf_Test_DFTn**. Зарисовать временные диаграммы **modY**, **D** и **n**.

Найти и записать максимальное значение первой гармоники (n=1) для bit0 (D=0) и для bit1 (D=1).

Найти и записать максимальное значение второй гармоники (n=2) для bit0 (D=0) и для bit1 (D=1).

5.5.1 Создать Verilog модуль **Gen_FSK_byte** и, входящий в его состав, модуль **Gen_FSK_SIN**.

5.5.2 Создать Verilog модуль **Mes_AMP_SH** входящий в его состав, модуль **MEM12x128** памяти задержки сигнала. Для модуля памяти создать текстовый файл init_MEM12x64.txt. В этом текстовом файле 128 строк 800 (смещение FSK сигнала в HEX формате).

5.6 Создать Verilog модуль **Test_MES_AMP_SH** схемы для моделирования измерителя амплитуды и смещения генератора **Gen_FSK_byte** .

5.7 Создать для схемы **Test_MES_AMP_SH** Verilog Test Fixture **tf_Test_MES_AMP_SH** задания на моделирование. Установить необходимые данные входных сигналов (см. п. 2.6).

5.8 При Simulation Run Time=20ms провести моделирование **tf_Test_MES_AMP_SH** для `Nbit=0 и своем варианте `Amin (см. CONXT.v). Зарисовать эскизы временных диаграмм сигналов TXD, OCD, Mamp, bf_SH и AMP.

5.9 Создать Verilog модули **DET_FSK**, **Test_DET_FSK** и Verilog Test Fixture **tf_Test_DET_FSK**.

5.10 При Nbit=8 провести моделирование схемы **tf_Test_DET_FSK** приемника FSK байта (Simulation Run Time=12ms). Проверить соответствие сигнала Rx_bit сигналу TX_bit. Зарисовать эскизы временных диаграмм сигналов: en_tx, TX_bit, FSK_en_rx, cb_rx_bit, F1_AMP, F2_AMP, RX_bit. Найти и записать минимальное отношение амплитуд гармоник на интервале FSK_en_rx.

6. Задание к сдаче (стоимость 4)

Для схемы **S_Sch_Lab414_F95** (рис.4) дополнительно создать модули и символы:

- **SPI_DAC8043** (приложение 7.3.2),
- **SPI_AD7895** (приложение 7.4.2),
- **ADC_95** (приложение 7.4.3),
- **MULT_5000_DIV_4096** (приложение 7.4.4)
- **FSK_FRXD** (приложение 7.5),
- **URXD_FSK_1byte** (приложение 7.5.1),
- **URXD1B** (приложение 7.5.2),
- **PIC_DET** (приложение 7. 5.3)
- **BIN12_to_DEC4** (приложение 7.5.4),
- **BTN_REG_DAT** (приложение 7.6),
- **BTN_REG_AMP** (приложение 7.7),
- **MUX_dat** (приложение 7.8),
- **MUX_FSK** (приложение 7.9),
- **LED_BL** (приложение 7.10),
- **DISPLAY** (приложение 7.11).

6.1 Создать лист схемы **S_Sch_Lab414_F95** (New Source - Schematic). Из созданных символов составить схему рис.3. Выполнить синтез (Synthesize - XST), исправить ошибки.

6.2 Для текстового варианта схемы **V_Sch_Lab414_F95** (приложение 7.12) символы создавать не надо, кроме того можно не создавать и модули **MUX_FSK** и **MUX_dat**.

6.3 Для выбранного варианта **S_Sch_Lab414_F95** или **V_Sch_Lab414_F95** создать *.ucf (New Source - Implementation Constraints File) (приложение 7.11).

6.4 В нижний ряд гнезд разъема JA макета NEXYS2 (JA7,JA8,JA9,JA10,JA11,JA12) вставить штыри печатной платы MDAC макета DAC8043. В JP1 вставить перемычку на +5V.

В нижний ряд гнезд разъема JC макета NEXYS2 (JC7,JC8,JC9,JC10,JC11,JC12) вставить штыри печатной платы SADC макета AD7895_3. В JP3 вставить перемычку на +5V.

Соединить проводной перемычкой выход XN3 MDAC с входом XN1 SADC (см. рис.3).

6.6 Создать *.bit (Generate Programming File) для загрузки в ПЛИС или *.mcs (Configure Target Device) для загрузки в ПЗУ (XCF04S). Загрузить в макет.

6.7 Провести при помощи осциллографа наблюдение осциллограмм напряжений на выходе XN2 макета SADC и на выводах JB2 (UTXD) и JD2 (URXD) макета NEXYS2. Синхронизировать ждущую развертку осциллографа необходимо фронтом сигнала en_tx (JB1). Проверить влияние множителя Mamp (BTN3, BTN0) и TX_DAT (BTN2, BTN1) на амплитуду и форму FSK сигнала. Определить минимальную амплитуду FSK сигнала при которой нет ошибок приема. Сравнить с минимальной амплитудой при SW[3]=1. Сохранить осциллограммы сигналов с выходов XN2 SADC (Uadc) и JD2 (URXD) при максимальной и минимальной амплитуде.

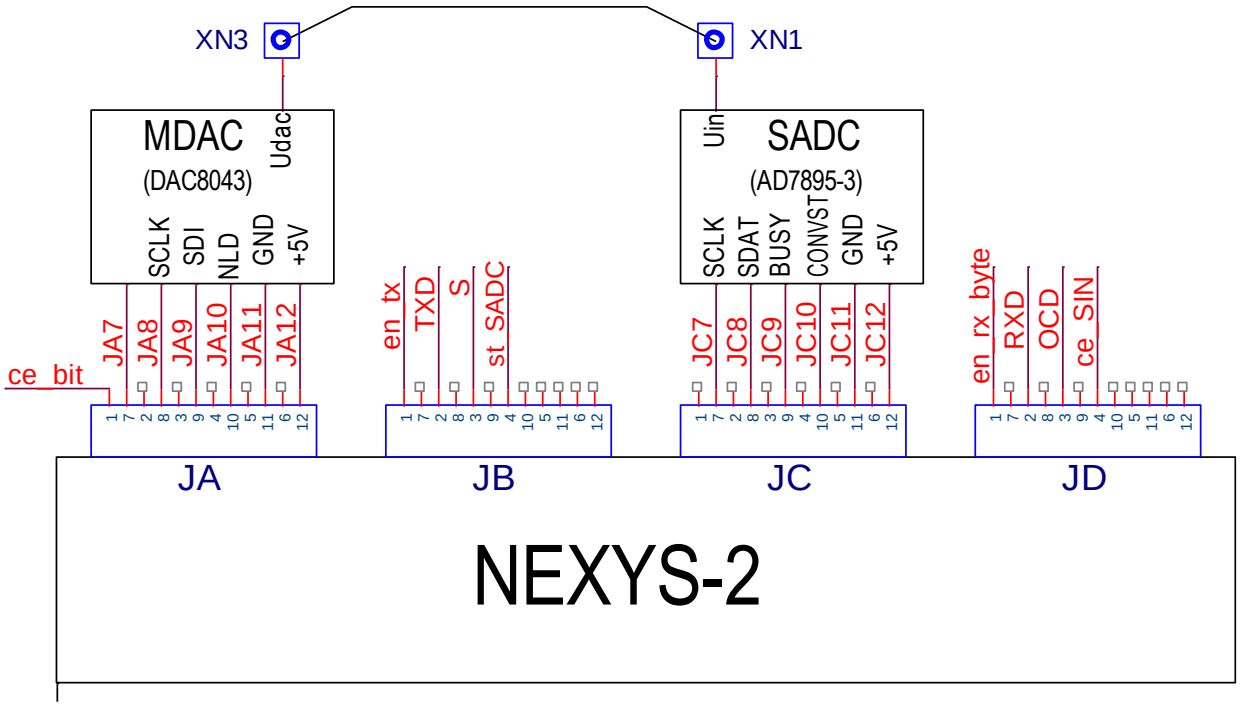


Рис.3 Схема соединения макетов MDAC и SADC с макетом NEXYS-2

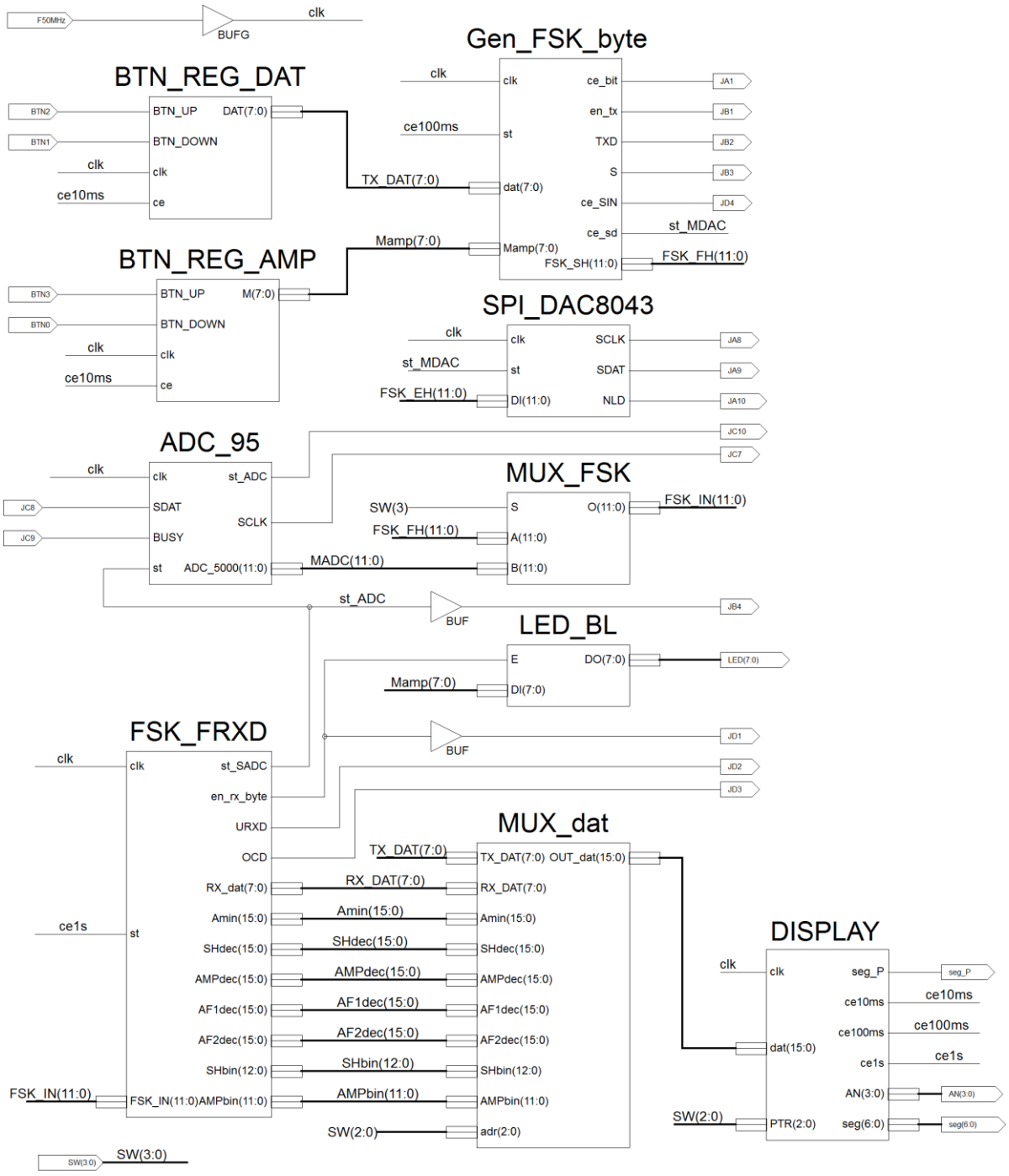


Рис. 4 Схема лабораторной работы S_Sch_Lab414_F95

В этой схеме:

- SW[2:0] – адрес отображаемых дисплеем данных,
- SW[3] – при SW[3]=0 на вход FSK_IN приемника FSK_RXD подаются данные с выхода модуля АЦП ADC_95, а при SW[3]=1 с выхода модуля генератора Gen_FSK_byte (FSK_IN=FSK_SH),
- кнопками BTN3 и BTN1 - регулируется амплитуда,
- кнопками BTN2 и BTN1 - регулируются данные TX_DAT.

Описание модулей: BTN_REG_DAT, BTN_REG_AMP, ADC_95, SPI_DAC8043, FSK_RXD, MUX_dat, DISPLAY, LED_BL и MUX_FSK приведены в приложениях 7.3-7.10

7. Приложения

7.1 Файл Wi100.txt инициализации модуля памяти поворотных множителей Wim=2^16*sin(2*pi*k/NP)

00000
01013
02015
02FF8
03FAA
04F1B
05E3D
06CFF
07B54
0892B
09679
0A32E
0AF3E
0BA9D
0C540
0CF1B
0D825
0E055
0E7A2
0EE05
0F378
0F7F5
0FB77
0FDFB
0FF7E
10000
0FF7E
0FDFB
0FB77
0F7F5
0F378
0EE05
0E7A2
0E055
0D825
0CF1B
0C540
0BA9D
0AF3E
0A32E
09679
0892B
07B54
06CFF
05E3D
04F1B
03FAA
02FF8

02015
01013
00000
3EFED
3DFEB
3D008
3C056
3B0E5
3A1C3
39301
384AC
376D5
36987
35CD2
350C2
34563
33AC0
330E5
327DB
31FAB
3185E
311FB
30C88
3080B
30489
30205
30082
30000
30082
30205
30489
3080B
30C88
311FB
3185E
31FAB
327DB
330E5
33AC0
34563
350C2
35CD2
36987
376D5
384AC
39301
3A1C3
3B0E5
3C056
3D008
3DFEB
3EFED

00000

7.2 Файл Wr100.txt инициализации модуля памяти поворотных множителей Wrm=2^16*cos(2*pi*k/NP)

10000
0FF7E
0FDFB
0FB77
0F7F5
0F378
0EE05
0E7A2
0E055
0D825
0CF1B
0C540
0BA9D
0AF3E
0A32E
09679
0892B
07B54
06CFF
05E3D
04F1B
03FAA
02FF8
02015
01013
00000
3EFED
3DFEB
3D008
3C056
3B0E5
3A1C3
39301
384AC
376D5
36987
35CD2
350C2
34563
33AC0
330E5
327DB
31FAB
3185E
311FB
30C88
3080B
30489

30205
30082
30000
30082
30205
30489
3080B
30C88
311FB
3185E
31FAB
327DB
330E5
33AC0
34563
350C2
35CD2
36987
376D5
384AC
39301
3A1C3
3B0E5
3C056
3D008
3DFEB
3EFED
00000
01013
02015
02FF8
03FAA
04F1B
05E3D
06CFF
07B54
0892B
09679
0A32E
0AF3E
0BA9D
0C540
0CF1B
0D825
0E055
0E7A2
0EE05
0F378
0F7F5
0FB77
0FDFB
0FF7E

00000

7.3 Множительный цифроаналоговый преобразователь DAC8043

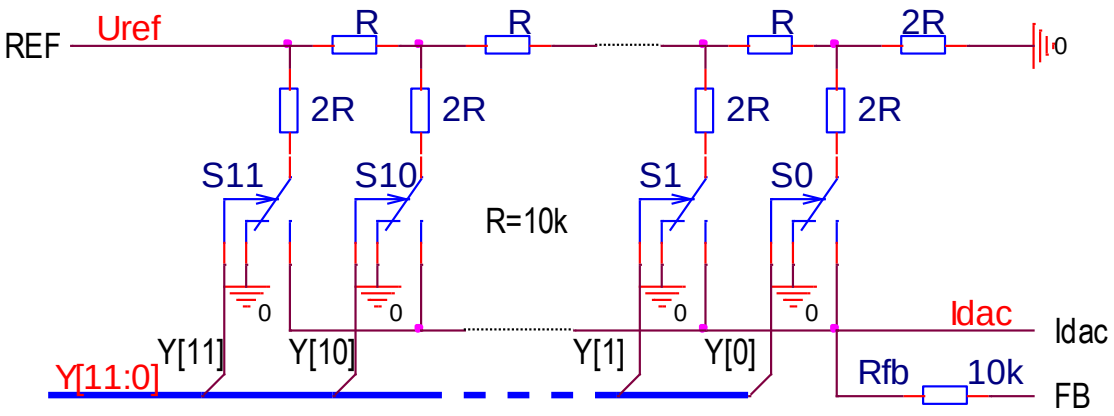


Рис.5 Схема цифроаналогового преобразователя DAC8043 с переключателями тока S11,S10,…S1,S0

ЦАП DAC8043 состоит из резисторного делителя R-2R, 12-ти переключателей тока (S11, S10,…S0, S0) и резистора обратной связи $R_{fb}=R$. Напряжение U_{ref} может быть как положительным так и отрицательным в пределах от -18Vдо +18V. Напряжение на выходе I_{dac} должно быть ровно 0. Тогда

$$I_{dac} = \frac{U_{ref}}{2R} y_{11} + \frac{1}{2}\frac{U_{ref}}{2R} y_{10} + \frac{1}{2^2}\frac{U_{ref}}{2R} y_9 + \cdots \frac{1}{2^{10}}\frac{U_{ref}}{2R} y_1 + \frac{1}{2^{11}}\frac{U_{ref}}{2R} y_0 = \\ = \frac{U_{ref}}{2R \cdot 2^{11}} \sum_{i=0}^{11} y_i \, 2^i$$

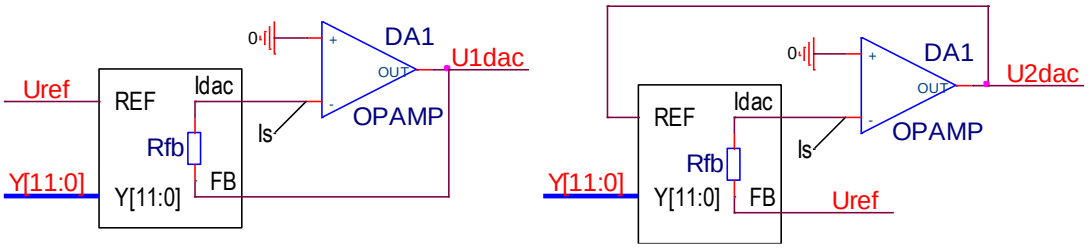
или $I_{dac} = \frac{U_{ref}}{R}\frac{Y}{2^{12}}$.

Для схемы рис.6.a, если напряжение смещения на входе операционного усилителя (ОУ) $U_s = 0$ и ток смещения инверсного входа ОУ $I_s=0$ при $R_{fb}=R$

$U1_{dac} = -U_{ref}\frac{Y}{2^{12}}$ - пропорционально числу *Y*.

Аналогично для схемы рис.6.b, при $R_{fb}=R$ и $U_s=0$, $I_s=0$

$U2_{dac} = -U_{ref} \cdot \frac{2^{12}}{Y}$ – обратно пропорционально числу *Y*.



а б

Рис. 6 Цифроаналоговые преобразователи: а – с прямой и б –обратной зависимостью выходного напряжения от числа Y

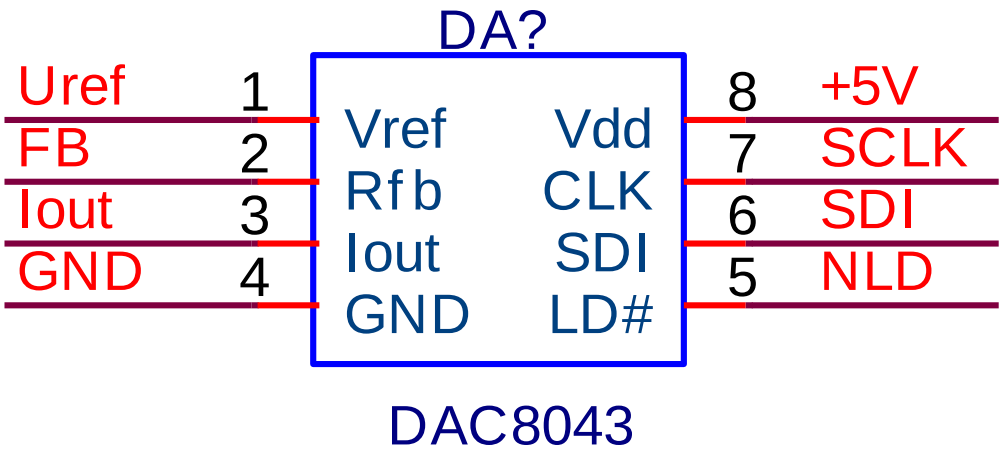


Рис.7 Выводы множительного цифроаналогового преобразователя DAC8043

7.3.1 Назначение выводов цифроаналогового преобразователя DAC8043
[см. Datasheet DAC8043.pdf]

Pin1 REF–DAC Reference Voltage, Input Pin (±18 V).
Pin2 FB - DAC Feedback Resistor Pin. This pin establishes voltage output for the DAC by connecting to an external amplifier output(±18 V).
Pin3 Iout -DAC Current Output.
Pin4 GND – Ground Pin.
Pin5 LD# - Load Strobe, Level-Sensitive Digital Input. Transfers shift register data to DAC register while active low (Tld>12ns).
Pin6 SDI - 12-Bit Serial Register Input. Data loads directly into the shift register MSB first. Extra leading bits are ignored.
Pin7 SCLK- Serial Clock Input. Positive edge clocks data into shift register (Tsclk>210ns).
Pin8 Vdd- Positive Power Supply Input (4.75V <Vdd< 5.25V, Is<500 uA).

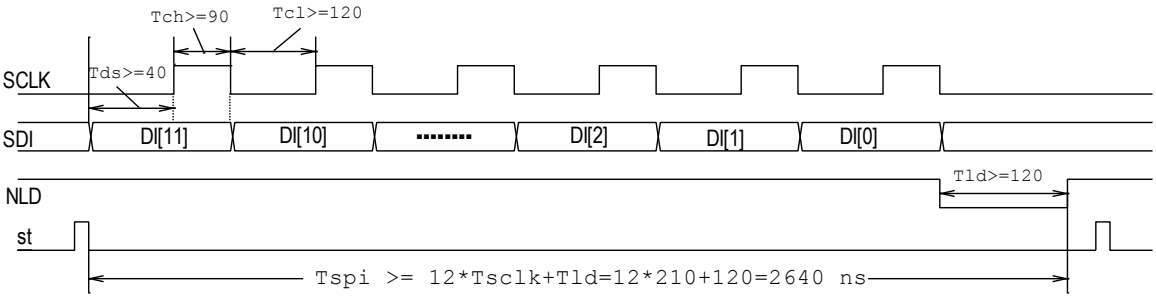


Рис.8 Временные диаграммы SPI интерфейса цифроаналогового преобразователя DAC8043 (размерность интервалов времени – нс)

При создании схемы модуля SPI_DAC8043 для макета NEXYS2 удобно установить Tch=Tcl=120 нс, тогда период сигнала SCLK Tsclk = Tch+Tcl = 240 нс. Сигал асинхронной загрузки NLD должен иметь длительность не менее 120 нс и может совпадать с последним импульсом SCLK. Если , T_{NLD} = SCLK/2=120 нс, то минимальный период сигнала запуска st равен 12*Tclk=12*240 нс = 2880 нс.

7.3.2 Модуль последовательной загрузки данных в ЦАП DAC8043

```
module SPI_DAC8043 (
        input clk,          output reg SCLK=0,
        input st,           output wire SDAT,
```

```
                input [11:0] DI,   output reg NLD=1,
                                   output reg [3:0]cb_bit=0,
                                   output wire ce);

parameter Tsclk = 240 ; //Tsclk=Tcl+Tch=120+120
parameter Tclk = 20 ; //Tclk=1/50 MHz

reg [3:0]cb_ce = 0 ;              //Счетчик тактов
wire ce=(cb_ce==Tsclk/Tclk);      //Границы тактов
reg en_sh=0 ;                     //Интервал сдвига данных
reg [11:0]sr_dat=0 ;              //Регистр сдвига данных
assign SDAT = sr_dat[11];         //Последовательные данные
reg [3:0]cb_bit=0;                //Счетчик бит

always @ (posedge clk) begin
cb_ce <= (st | ce)? 1 : cb_ce+1 ;
SCLK <= (st | ce)? 0 : (en_sh & (cb_ce==5))? 1 : SCLK ;
en_sh <= st? 1 : ((cb_bit==11) & ce)? 0 : en_sh ;
cb_bit <= st? 0 : (en_sh & ce)? cb_bit+1 : cb_bit ;
sr_dat <= st? DI : (en_sh & ce)? sr_dat<<1 : sr_dat ;
NLD <= st? 1 : ce? !(cb_bit==11) : NLD ;
end
endmodule
```

7.3.3 Макет цифроаналогового преобразователя MDAC

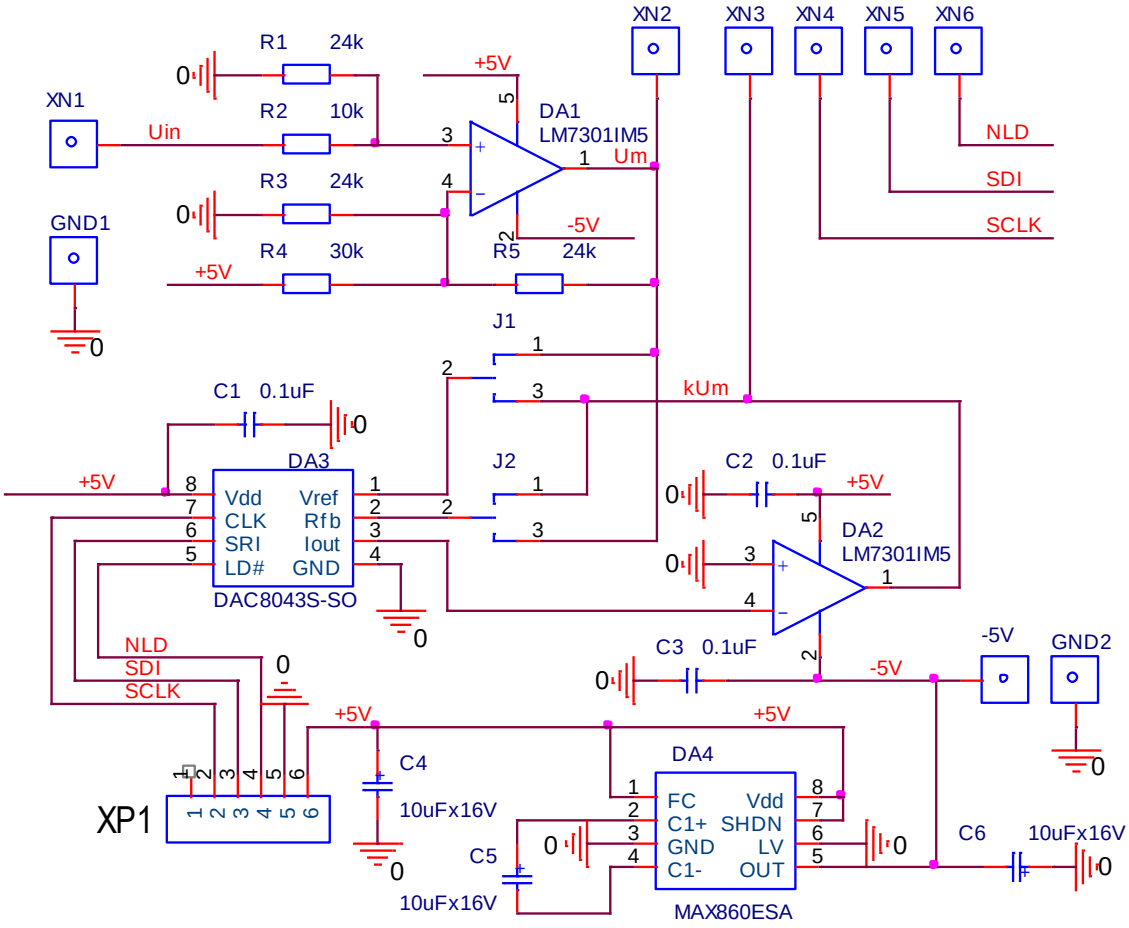


Рис.9 Схема макета MDAC цифроаналогового преобразователя DAC8043

Функциональность схемы макета определяется переключателями (джамперами) J1 и J2. При замыкании выводов J1.1 и J1.2, а также J2.1 и J2.2 напряжение *kUm* на выходе DA2, $kUm = -U_{ref}\frac{Y}{4096}$ пропорционально *Y*, а при замыкании выводов J1.2 и J1.3, а также J2.2 и J2.3 напряжение *kUm* на выходе DA2, $kUm = -U_{ref}\frac{4096}{Y}$, обратно пропорционально *Y*, где U_{ref}=U_m, а *Y* (0<=*Y*<=4095) - число загружаемое в регистр DAC8043 по SPI интерфейсу.

Напряжение U_m на выходе операционного усилителя DA1,

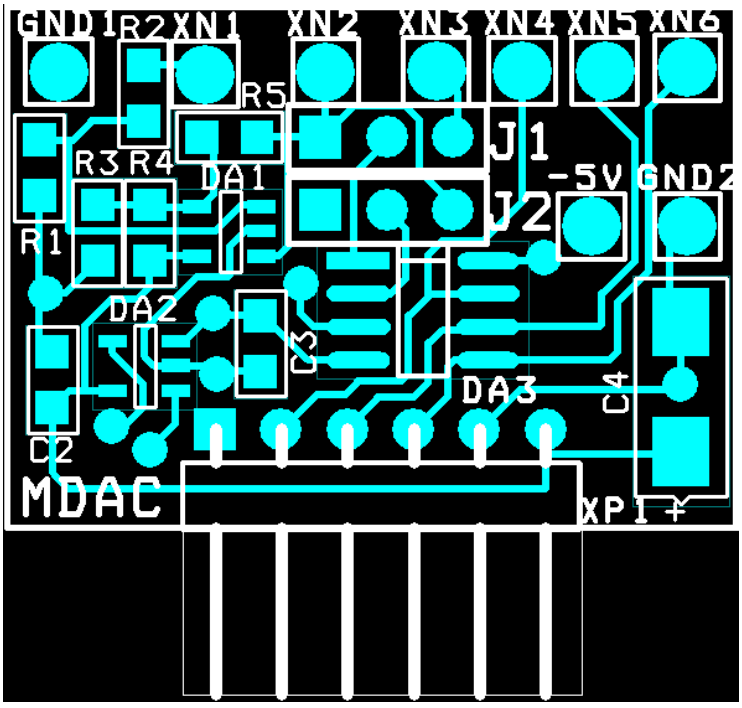
$$U_m = U_{in}\frac{R_1(R_5(R_3+R_4)+R_3\cdot R_4)}{R_3\cdot R_4(R_1+R_2)} - V_{dd}\frac{R_5}{R_4}, \qquad V_{dd} = +5V$$

При $U_{in} = 0V$, $U_m = -4V$.
При $U_{in} = 2.0238V$, $U_m = 0V$.
При $U_{in} = +4.095V$, $U_m = +4.094V$.
Приближенно, $U_m \approx 2(U_{in} - 2V)$.

На макетах MDAC к данной работе перемычки на джамперах J1 и J2 запаяны режим прямой зависимости U_{dac} от числа FSK_SH.

Напряжение *Uin* должно быть равно 0 (вход XN1 свободен или соединен GND). В этом случае напряжение U_{dac} на выходе ОУ DA2 (XN3)

$$U_{dac} = 4V\frac{FSK\_SH}{4096}.$$



7.

Рис.10 Печатная плата макета множительного цифроаналогового преобразователя DAC8043

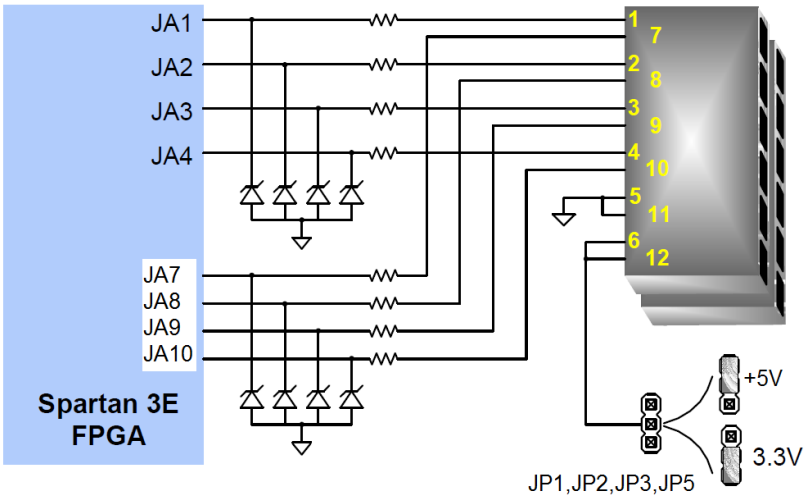


Рис.11 Порты макета NEXYS2

7.4 Аналого-цифровой преобразователь (АЦП) AD7895-3

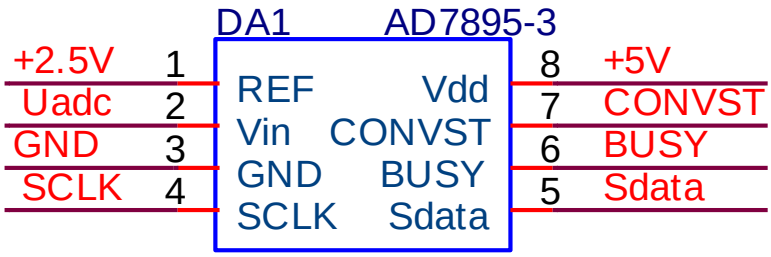


Рис.12 Микросхема аналого-цифрового преобразователя AD7893-5

7.4.1 Назначение выводов аналого-цифрового преобразователя DAC7895-3

[см. Data sheet AD7895.pdf]

Pin1 REF_IN - Voltage Reference Input. An external reference source should be connected to this pin to provide the reference voltage for the AD7893's conversion process. The REF IN input is buffered on chip. The nominal reference voltage for correct operation of the AD7895 is +2.5 V.

Pin2 Vin - Analog Input Channel. The analog input range is 0 V to +2.5 V (AD7895-3).

Pin3 AGND - Analog Ground. Ground reference for track/hold, comparator, digital circuitry and DAC.

Pin4 SCLK - Serial Clock Input. An external serial clock is applied to this input to obtain serial data from the AD7895. A new serial data bit is clocked out on the rising edge of this serial clock, and data is valid on the falling edge. The serial clock input should be taken low at the end of the serial data transmission.

Pin5 SDATA- Serial Data Output. Serial data from the AD7895 is provided at this output. The serial data is clocked out by the rising edge of SCLK and is valid on the falling edge of SCLK. Sixteen bits of serial data are provided with four leading zeros followed by the 12 bits of conversion data. On the sixteenth falling edge of SCLK, the SDATA line is disabled (three-stated). Output data coding straight binary for the AD7895-3.

Pin6 BUSY - used to indicate when the part is doing a conversion. The BUSY pin will go high on the falling edge of CONVST and will return low when the conversion is complete.

Pin7 CONVST - Convert Start. Edge-triggered logic input. On the falling edge of this input, the serial clock counter is reset to zero. On the rising edge of this input, the track/hold goes into its hold mode and conversion is initiated.
Pin8 Vdd - Positive supply voltage, +5 V +/-5%.

FSR – диапазон входного напряжения для AD7895-3 при U_{REF} = +2.5V от -2.5V до +2.5 V (FSR=5V).
LSB - вес младшего разряда для AD7895-3, LSB=FSR/4096=0.122 mV.

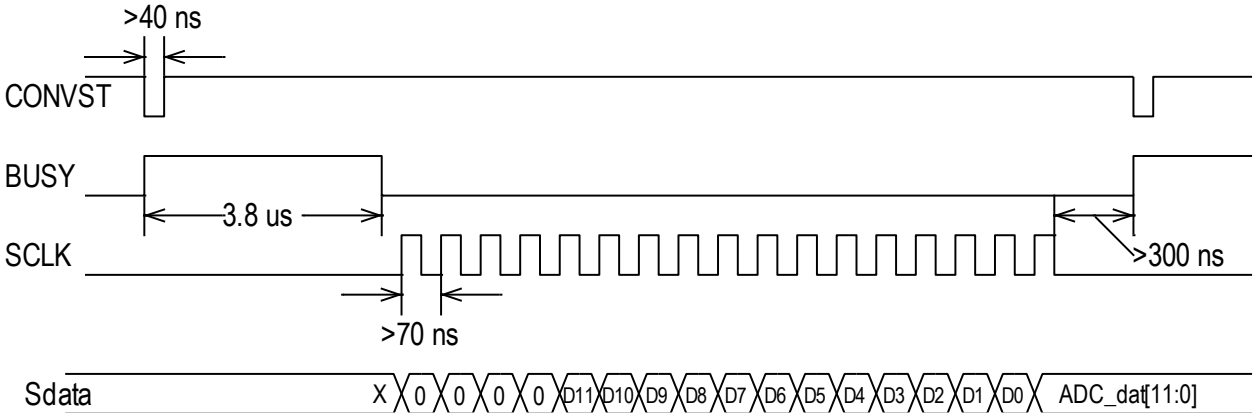


Рис.13 Временные диаграммы сигналов АЦП AD7895

По спаду импульса запуска CONVST начинается процесс последовательного приближения, который продолжается в течение 3.4 мкс. После его окончания разрешается последовательное считывание данных аналого-цифрового преобразования. Номинально данные должны считываться по фронтам импульсов синхронизации, хотя допускается считывание и по спадам импульсов синхронизации.

Функционально система выгрузки данных АЦП рассчитана на 16 разрядов, поэтому для первых 4-х импульсов синхронизации SDATA=0. После 16-го импульса SCLK SDATA переходит в третье состояние.

7.4.2 SPI модуль считывания данных АЦП AD7895

```
module SPI_AD7895 (
        input clk,      output wire SCLK,   //Импульсы синхронизации
        input BUSY,     output reg st_ADC=1,//CONVST
        input SDAT,     output reg [11:0]ADC_dat=0, //Буфер данных
        input st,       output reg ok_adc=0, //Конец преобразования
                        output reg [11:0]sr_DAT=0, //Регистр сдвига данных
                        output reg [4:0]cb_bit=0, //Счетчик бит
                        output wire ce_tact); //Границы тактов
reg tBUSY=0, ttBUSY=0, Q=0 ;
reg [1:0]cb_ce=0 ;              //Счетчик такта
wire ce=(cb_ce==3);             //Период ce Tce=80 ns
wire ce_bit = ce & !Q ;         //Период Q=160 ns
assign ce_tact = ce & Q ;       //Период Q=160 ns
wire start_SCLK = !tBUSY & ttBUSY & ce;
wire stop_SCLK = (cb_bit==16) & ce_tact;
reg en_SCLK=0 ;                 //Интервал считывания данных
assign SCLK = Q & en_SCLK ; //Период SCLK=160 ns

always @ (posedge clk) begin
cb_ce <= st? 0 : cb_ce+1 ;
```

```
Q <= st? 0 : ce? !Q : Q ;
st_ADC <= st? 0 : ce? 1 : st_ADC ;
tBUSY <= ce? BUSY : tBUSY ;    ttBUSY <= ce? tBUSY : ttBUSY;
en_SCLK <= start_SCLK? 1 : stop_SCLK? 0 : en_SCLK ;
cb_bit <= st? 0 : (en_SCLK & ce_tact)? cb_bit+1 : cb_bit ;
sr_DAT <= st? 0 : (ce_bit & en_SCLK)? sr_DAT<<1 | SDAT : sr_DAT ;
ok_adc <= stop_SCLK? 1 : ce? 0 : ok_adc ;
ADC_dat <= ok_adc? sr_DAT : ADC_dat ;
end
endmodule
```

В этом модуле после импульса старта st_ADC (CONVST) начало процесса считывания данных задерживается на 3.8 мкс и индицируется сигналом BUSY. Минимальный период запуска 3800+16*160+300 = 6660 нс.

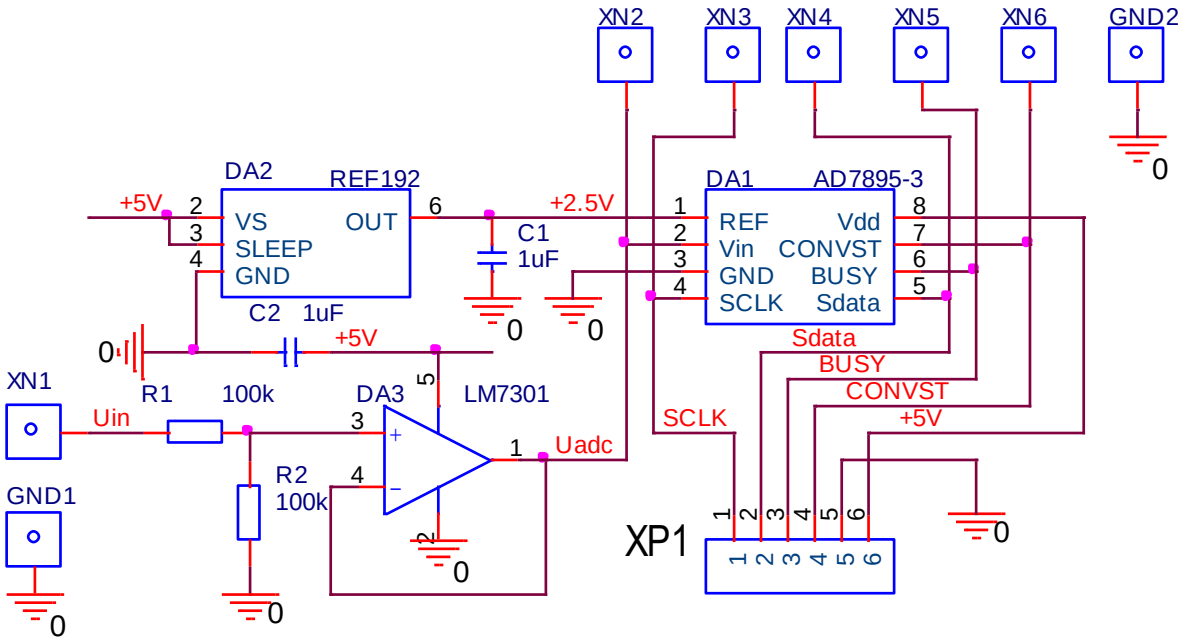


Рис.14 Схема макета SADC АЦП AD7895-3

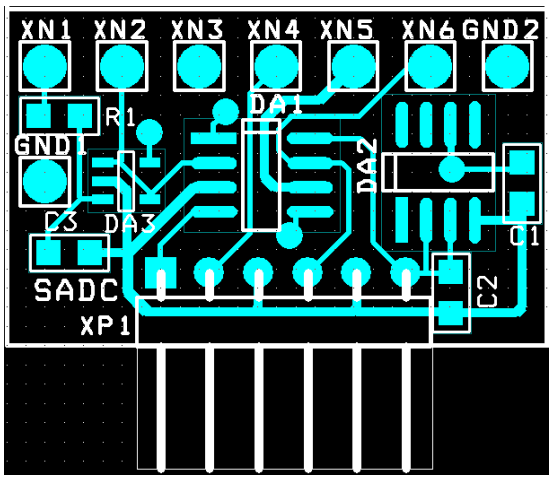


Рис.15 Лицевая сторона печатной платы макета SADC АЦП AD7895-3

Штыри печатной платы макета AD7895 предлагается вставлять в нижний ряд гнезд разъема JC макета NEXYS2 (JC7, JC8, JC9, JC10, JC11, JC12). Напряжение питания +5V

подается на XP1.6 платы макета AD7895 от JC12 при наличии перемычки между средним выводом и верхним выводом (VSWT) JP3 NEXYS2 (см. рис.11). JC5 и JC11 соединены с GND.

Данные ADC_{dat} линейно связаны с входным напряжением *Uadc* и *Uin* (*Uadc*=*Uin*/2),

$$ADC_{dat} = 4096\frac{U_{adc}}{2.5V} = 4096\frac{U_{in}}{2\cdot 2.5V} = 4096\frac{U_{in}}{5V}.$$

При соединении выхода (XN1) макета ЦАП с входом (XN1) макета АЦП

$$U_{in} = U_{dac} = 4.096V\frac{FSK}{4096}, \text{ или } ADC_{dat} = 4096\frac{4.096V}{5V}\cdot\frac{FSK}{4096} = \frac{4.096}{5}FSK.$$

Для приравнивания весов разрядов FSK и выхода АЦП необходимо умножить ADC_{dat} на 5000 и разделить на 4096, тогда $MADC_{dat} = ADC_{dat}\cdot\frac{5000}{4096} = FSK$.

7.4.3 Модуль ADC со встроенным умножителем

```
module ADC_95(
    input clk,      output wire [11:0]ADC_5000, //ADC_5000=(Uin/5.000V)*5000
    input SDAT,     output wire st_ADC, //Convst
    input st,       output wire SCLK,   //Импульсы синхронизации
    input BUSY);

wire [11:0]ADC_4096 ;//ADC_4096=(Uin/4.096V)*4096
SPI_AD7895 DD1 ( .clk(clk),       .SCLK(SCLK),
                 .st(st),         .st_ADC(st_ADC),
                 .SDAT(SDAT),     .ADC_dat(ADC_4096),
                 .BUSY(BUSY));

MULT_5000_DIV_4096 DD2 (.A(ADC_4096), .B(ADC_5000));//B=(A*5000)/4096
endmodule
```

7.4.4 Модуль умножения на 5000 и деления на 4096

```
module MULT_5000_DIV_4096( input [11:0]A, output wire[11:0] B );
wire [24:0]MA = A*5000 ;  //Умножение на 5000
assign B = MA>>12 ;     //Деление на 2^12=4096
endmodule
```

7.5 Модуль приемника FSK байта со встроенными преобразователями DIN12_to_DEC4

```
`include "CONST.v"
module FSK_FRXD (
        input [11:0]FSK_IN,
        input clk,
        input st,
                        output wire st_SADC,        //Старт АЦП AD7895
                        output wire en_rx_byte,     //JD1
                        output wire URXD,           //JD2
                        output wire OCD,            //JD3
                        output wire [7:0] RX_dat,   //Принятый байт
                        output wire [15:0] Amin,    //Минимальная амплитуда
                        output wire [15:0] SHdec,   //Смещение (DEC)
                        output wire [15:0] AMPdec,//Амплитуда (DEC)
                        output wire [15:0] AF1dec,  //Первая гармоника
                        output wire [15:0] AF2dec,  //Вторая гармоника
                        output wire [12:0] bf_SH,   //Смещение (HEX)
```

```
                                output wire [11:0]bf_AMP);//Амплитуда (HEX)
wire  ok_rx_bit ;
wire [10:0] F2_AMP ; wire [10:0] F1_AMP ;
URXD_FSK_1byte DD1 (        .RXD(URXD),
            .clk(clk),              .RX_dat(RX_dat),
            .FSK_SH(FSK_IN), .en_rx_byte(en_rx_byte),
                                .OCD(OCD),
                                .bf_AMP(bf_AMP),
                                .bf_SH(bf_SH),
                                .ce_Fd(st_SADC),
                                .F2_AMP(F2_AMP),
                                .F1_AMP(F1_AMP),
                                .ok_rx_bit(ok_rx_bit));

//--Пиковый детектор амплитуды первой гармоники F1 AMP
wire [11:0]PIC_F1 ;
PIC_DET DD2 (
        .A(F1_AMP),     .PIC(PIC_F1),
        .ce(ok_rx_bit),
        .clk(clk),
        .st(st));
//--Пиковый детектор амплитуды второй гармоники F2 AMP
wire [11:0]PIC_F2 ;
PIC_DET DD3 (
        .A(F2_AMP),     .PIC(PIC_F2),
        .ce(ok_rx_bit),
        .clk(clk),
        .st(st));
//--Преобразователь двоичного числа PIC_AMP в двоично-десятичное AMPdec
BIN12_to_DEC4 DD4 ( .BIN(bf_AMP),   .DEC(AMPdec),
                    .st(st),
                    .clk(clk));
//--Преобразователь двоичного числа PIC_SH в двоично-десятичное SHdec
BIN12_to_DEC4 DD5 ( .BIN(bf_SH),      .DEC(SHdec),
                    .st(st),
                    .clk(clk));
//--Преобразователь двоичного числа PIC_F1 в двоично десятичное AF1dec
BIN12_to_DEC4 DD6 ( .BIN(PIC_F1),    .DEC(AF1dec),
                    .clk(clk),
                    .st(st));
//--Преобразователь двоичного числа PIC_F2 в двоично десятичное AF2dec
BIN12_to_DEC4 DD7 ( .BIN(PIC_F2),    .DEC(AF2dec),
                    .clk(clk),
                    .st(st));
//--Преобразователь двоичного числа `Amin в двоично десятичное Amin
wire [11:0]bin_Amin=`Amin ;
BIN12_to_DEC4 DD8 ( .BIN(bin_Amin),.DEC(Amin),
                    .clk(clk),
                    .st(st));
endmodule
```

7.5.1 Модуль приемника FSK байта

```
module URXD_FSK_1byte(output wire [11:0] DFSK_SH,//Задержаный сигнал FSK_SH
    input [11:0]FSK_SH,     output wire OCD,          //Превышение порога
    input clk,              output wire [12:0] bf_SH,   //Буфер смещения SIN_FSK
                            output wire [10:0] bf_AMP,//Буфер амплитуды SIN_FSK
                            output wire RXD,          //Принятый бит
                            output wire ce_Fd,        //Сигнал дискретизации
                            output wire ok_rx_bit,    //Сигнал приема бита
                            output wire en_rx_byte,   //Интервал приема
                            output wire [3:0] cb_rx_bit,//Счетчик бит
                            output wire [7:0] RX_dat,   //Принятый байт
                            output wire[10:0]F2_AMP, //DFT Амплитуда 2-й гармоники FSK
                            output wire[10:0]F1_AMP);//DFT Амплитуда 1-й гармоники FSK

//--Детектор FSK бита
DET_FSK DD1 (.FSK_SH(FSK_SH), .DFSK_SH(DFSK_SH),
             .clk(clk),       .OCD(OCD),
                              .bf_SH(bf_SH),
                              .bf_AMP(bf_AMP),
                              .RX_bit(RXD),
                              .ce_Fd(ce_Fd),
                              .ok_rx_bit(ok_rx_bit),
                              .F2_AMP(F2_AMP),     //Амплитуда второй гармоники
                              .F1_AMP(F1_AMP)); //Амплитуда первой гармоники
//--Приемник одного байта
wire start_rx ;
URXD1B DD2 ( .inp(RXD),    .en_rx_byte(en_rx_byte),
             .clk(clk),    .RX_dat(RX_dat),
                           .cb_bit(cb_rx_bit),
                           .start_rx(start_rx));
always @ (posedge clk) if (start_rx) begin
bf_AMP <= AMP ;
end
endmodule
```

7.5.2 Модуль UART приемника 1-го байта

```
module URXD1B(   input inp,   output reg en_rx_byte=0,
                 input clk,   output reg [7:0] RX_dat=0,
                              output wire start_rx,
                             output wire ok_rx_byte,
                              output reg [3:0]cb_bit=0,
                              output wire T_dat,
                              output wire ce_tact,
                              output wire ce_bit  );

parameter Fclk=50000000 ; //Fclk=50MHz
parameter F1=1200 ;      //1200 Bod
reg tin=0, ttin=0;//
reg [15:0]cb_tact=0 ;
assign ce_tact = (cb_tact==Fclk/F1) ;
reg [7:0] sr_dat=0 ;
```

```
wire spad_inp = !tin & ttin ;//spad_inp
assign start_rx = spad_inp & !en_rx_byte ;
assign ce_bit = (cb_tact==Fclk/(2*F1));
assign ok_rx_byte = (ce_bit & (cb_bit==9) & en_rx_byte & tin);
assign T_dat = ((cb_bit>=1) & (cb_bit<=8));

always @ (posedge clk) begin
tin <= inp ; ttin <= tin ;
cb_tact <= (start_rx | ce_tact)? 1 : cb_tact+1;
en_rx_byte <= ((cb_bit==9) & ce_bit)? 0 : (ce_bit & !tin)? 1: en_rx_byte ;
cb_bit <=  (start_rx | ((cb_bit==9) & ce_tact))? 0 : (ce_tact & en_rx_byte)? cb_bit+1 : cb_bit ;
sr_dat <= start_rx? 0 : (ce_bit & T_dat)? sr_dat >>1 | tin<<7 : sr_dat ;//in
RX_dat <= ok_rx_byte? sr_dat : RX_dat ;
end
endmodule
```

7.5.3 Пиковый детектор

```
module PIC_DET(
   input [10:0] A,      output reg [10:0] PIC,
   input ce,
   input clk,
   input st);
always @ (posedge clk) begin
PIC <= st? 0 : (ce & (A>PIC))? A : PIC ;
end
endmodule
```

7.5.4 Преобразователь двоичного числа в двоично-десятичное

```
module BIN12_to_DEC4(
        input [11:0] BIN, output wire [15:0] DEC,
        input st,             output reg [2:0]ptr_dig=0,   //Указатель номера декадной цифры
        input clk,            output reg en_conv=0);       //Разрешение преобразования

reg[3:0]D1dec=0;     reg[3:0]D2dec=0;
reg[3:0]D3dec=0;     reg[3:0]D4dec=0;

reg [16:0]rest=0; //Остаток
assign DEC= {D4dec,D3dec,D2dec,D1dec} ;

wire d4 = (ptr_dig==4); wire d3 = (ptr_dig==3);
wire d2 = (ptr_dig==2);  wire d1 = (ptr_dig==1);

wire[15:0]Nd =  d4?  1000 :
                      d3?   100 :
                      d2?    10 :
                      d1?     1 : 0 ;
wire [16:0]dx = rest-Nd ; //Разность между остатком и di (i=,4,3,2,1)
wire z = dx[16] ; // Знак разности
wire en_inc_dig  = en_conv & !z ; //Разрешение инкремента декадной цифры
wire en_dec_ptr  = en_conv &  z ; //Разрешение декремента указателя цифры
```

```
always @(posedge clk) begin
en_conv <= st? 1 : (ptr_dig==0)? 0 : en_conv ;    //Разрешение преобразования
rest <= st? BIN : en_inc_dig? dx : rest ;              //Текущий остаток
ptr_dig <= st? 4: en_dec_ptr?  ptr_dig-1 : ptr_dig ;    //Указатель очередной декадной цифры
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 ;
//ok <=(ptr_dig==0) & en_conv;
end
endmodule
```

7.6 Модуль регулятора данных

```
module BTN_REG_DAT(
   input BTN_UP,              output reg [7:0] DAT=8'h80, //Данные
   input BTN_DOWN,
   input clk,
   input ce); //ce1ms или ct10ms

reg [1:0]Q_UP ; //
reg [1:0]Q_DOWN ; //
wire st_UP= Q_UP[0] & !Q_UP[1] & ce ; //Для сдвига вправо (делить на 2)
wire st_DOWN= Q_DOWN[0] & !Q_DOWN[1] & ce ; //Для сдвига влево (умножить на 2)

wire Mmax=(DAT==8'hFF); //Максимальное значение M=128
wire Mmin=(DAT==8'h01); //Минимальное значение M=1

always @ (posedge clk) if (ce) begin
Q_UP <= Q_UP<<1 | BTN_UP ; //Сдвиг BTN_UP
Q_DOWN <= Q_DOWN<<1 | BTN_DOWN ; //Сдвиг BTN_DOWN
DAT <= (!Mmax & st_UP)? DAT+1 : (!Mmin & st_DOWN)? DAT-1 : DAT ;
end

endmodule
```

7.7 Модуль регулятора амплитуды

```
module BTN_REG_AMP(
     input BTN_UP,  output reg [7:0] M=8'h80, //Множитель
     input BTN_DOWN,
     input clk,
     input ce);//ce1ms или ct10ms

reg [1:0]Q_UP ;//
reg [1:0]Q_DOWN ;//
wire st_UP= Q_UP[0] & !Q_UP[1] & ce ;//Для сдвига вправо (делить на 2)
wire st_DOWN= Q_DOWN[0] & !Q_DOWN[1] & ce ;//Для сдвига влево (умножить на 2)

wire Mmax=(M==8'h80); //Максимальное значение M=128
wire Mmin=(M==8'h01); //Минимальное значение M=1
```

```
always @ (posedge clk) if (ce) begin
Q_UP <= Q_UP<<1 | BTN_UP ;//Сдвиг BTN_UP
Q_DOWN <= Q_DOWN<<1 | BTN_DOWN ;//Сдвиг BTN_DOWN
M <= (!Mmax & st_UP)? M<<1 : (!Mmin & st_DOWN)? M>>1 : M ;
end
endmodule
```

7.8 Модуль мультиплексора данных для Display

```
module MUX_dat(
    input [7:0] TX_DAT,     output wire [15:0] OUT_dat,
    input [7:0] RX_DAT,
    input [15:0] Amin,
    input [15:0] SHdec,
    input [15:0] AMPdec,
    input [15:0] AF1dec,
    input [15:0] AF2dec,
    input [12:0] SHbin,
    input [11:0] AMPbin,
    input [2:0]adr);

assign OUT_dat = (adr[2:0]==0)? {TX_DAT,RX_DAT} :
                 (adr[2:0]==1)? Amin :
                 (adr[2:0]==2)? SHdec :
                 (adr[2:0]==3)? AMPdec :
                 (adr[2:0]==4)? AF1dec :
                 (adr[2:0]==5)? AF2dec :
                 (adr[2:0]==6)? {3'b000,SHbin} :
                                {4'b0000,AMPbin};
endmodule
```

7.9 Модуль мультиплексора входного сигнала для модуля FSK_RXD

```
module MUX_FSK(
   input [11:0] A,output wire[11:0] O,
   input [11:0] B,
   input S);

assign O = S? A : B ;
endmodule
```

7.10 Модуль светодиодов

```
module LED_BL(    input [7:0] DI,        output wire [7:0] DO,
                  input E );
assign DO= DI | {8{E}} ;
endmodule
```

7.11 Модуль семи сегментного светодиодного индикатора

```
module DISPLAY(input clk,          output wire[3:0] AN,    //Аноды
               input [15:0]dat,    output wire [6:0]seg,   //Сегменты
               input [2:0]PTR,     output wire seg_P,      //Точка
```

```
                                    output wire ce10ms,      //10 милисекунд
                                    output wire ce100ms,     //100 милисекунд
                                    output reg ce1s=0);      //1 секунда

parameter Fclk=50000000 ;      //50000000 Hz
parameter F1kHz=1000 ;         //1 kHz
parameter F100Hz=100 ;         //100 Hz

wire [1:0]ptr_P = (PTR==0)? 2'b10 :   //Точка в центре      (XX.XX)
                  (PTR==7)? 2'b00 :   //Точка справа    (XXXX.)
                            2'b11 ;   //Точка     слева  (X.XXX)

reg [15:0] cb_1ms = 0 ;
wire ce = (cb_1ms==Fclk/F1kHz) ;
reg [3:0]cb_10ms=0 ;
assign ce10ms = (cb_10ms==10) & ce ;
reg [3:0]cb_100ms=0 ;
assign ce100ms = (cb_100ms==10) & ce10ms ;
reg [3:0]cb_1s=0 ;

//--Генератор сигнала ce (период 1 мс, длительность Tclk=20 нс)--
always @ (posedge clk) begin
cb_1ms <= ce? 1 : cb_1ms+1 ;
cb_10ms <= ce10ms? 1 : ce? cb_10ms+1 : cb_10ms ;
cb_100ms <= ce100ms? 1 : ce10ms? cb_100ms+1 : cb_100ms ;
cb_1s <= ce1s? 1 : ce100ms? cb_1s+1 : cb_1s ;
ce1s <= (cb_1s==10) & ce100ms ;
end
//------ Счетчик цифр ----------------------------------------
reg [1:0]cb_dig=0 ;
always @ (posedge clk) if (ce) begin
cb_dig <= cb_dig+1 ;
end
//-------Переключатель «анодов»-------------
assign AN =   (cb_dig==0)? 4'b1110 : //включение цифры 0 (младшей)
              (cb_dig==1)? 4'b1101 : //включение цифры 1
              (cb_dig==2)? 4'b1011 : //включение цифры 2
                           4'b0111 ; //включение цифры 3 (старшей)
//-------Переключатель тетрад (HEX цифр)-------------
wire[3:0] dig =(cb_dig==0)? dat[3:0]:
                (cb_dig==1)? dat[7:4]:
                (cb_dig==2)? dat[11:8]: dat[15:12];
//-------Семи сегментный дешифратор----------
                          //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
                (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
//-------Указатель точки-------
assign seg_P = !(ptr_P == cb_dig) ;
endmodule
```

7.12 Распределение сигналов по выводам ПЛИС (файл Sch_Lab414_F95.ucf)

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

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

NET "BTN0" LOC = "B18" ; # dec Mamp
NET "BTN1" LOC = "D18" ; # dec TX_DAT
NET "BTN2" LOC = "E18" ; # inc TX_DAT
NET "BTN3" LOC = "H13" ; # inc Mamp

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_P"  LOC = "C17" ; #DOT

NET "SW<0>" LOC = "G18" ; #adr[0]
NET "SW<1>" LOC = "H18" ; #adrt[1]
NET "SW<2>" LOC = "K18" ; #adr[2]
NET "SW<3>" LOC = "K17" ; #sw[3]
#NET "SW<4>" LOC = "L14" ; #dat[4]
#NET "SW<5>" LOC = "L13" ; #dat[5]
#NET "SW<6>" LOC = "N17" ; #dat[6]
#NET "SW<7>" LOC = "R17" ; #dat[7]

NET "LED<0>" LOC = "J14" ; #
NET "LED<1>" LOC = "J15" ; #
NET "LED<2>" LOC = "K15" ; #
NET "LED<3>" LOC = "K14" ; #
NET "LED<4>" LOC = "E17" ; #
NET "LED<5>" LOC = "P15" ; #
NET "LED<6>" LOC = "F4"  ; #
NET "LED<7>" LOC = "R4"  ; #
```

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

NET "JA1" LOC = "L15" ; #ce_SIN
#NET "JA2" LOC = "K12" ; #
#NET "JA3" LOC = "L17" ; #
#NET "JA4" LOC = "M15" ; #
#NET "JA7" LOC = "K13" ; #
NET "JA8" LOC = "L16" ; #SCLK DAC8043
NET "JA9" LOC = "M14" ; #SDAT DAC8043
NET "JA10" LOC = "M16"; #NLD DAC8043

NET "JB1" LOC = "M13" ; #en_tx
NET "JB2" LOC = "R18" ; #UTXD
NET "JB3" LOC = "R15" ; #S
NET "JB4" LOC = "T17" ; #st_SADC
#NET "JB7" LOC = "P17" ; #
#NET "JB8" LOC = "R16" ; #
#NET "JB9" LOC = "T18" ; #
#NET "JB10" LOC = "U18"; #

#NET "JC1" LOC = "G15" ; #
#NET "JC2" LOC = "J16" ; #
#NET "JC3" LOC = "G13" ; #
#NET "JC4" LOC = "H16" ; #
NET "JC7" LOC = "H15" ; #SCLK AD7895
NET "JC8" LOC = "F14" | PULLDOWN ; #SDAT AD7895
NET "JC9" LOC = "G16" ; #BUSY_AD7895
NET "JC10" LOC = "J12" ; #CONVST AD7895

NET "JD1" LOC = "J13" ;# en_rx_byte
NET "JD2" LOC = "M18" ;# URXD
NET "JD3" LOC = "N18" ;# OCD
NET "JD4" LOC = "P18" ;#st_SDAC
#NET "JD7" LOC = "K14" ;#
#NET "JD8" LOC = "K15" ;#
#NET "JD9" LOC = "J15" ;#
#NET "JD10" LOC = "J14" ;#
```

7.13 Текстовый вариант схемы лабораторной работы

```
module V_Sch_Lab414_F95(
        input F50MHz,              //Сигнал генератора F50MHz NEXYS-2
//--Реверсивный источник данных FSK----------
        input BTN2, //inc DAT
        input BTN1, //dec DAT
//--Регулятор умножителя амплитуды FSK----------
        input BTN3,/*inc Mamp*/   output wire [7:0] LED, //Светодиоды
        input BTN0,/*dec Mamp*/
//--Генератор сигнала FSK-------------------------
                output wire JB1,//en_tx
```

```
                output wire JB2,//UTXD
                output wire JB3,//S знак FSK SIN
                output wire JB4,//st_SADC
//--SPI_DAC8043-----------------------------------
                output wire JA1,//ce_bit
                output wire JA8,//SCLK_DAC_8043
                output wire JA9,//SDAT_DAC_8043
                output wire JA10,//NLD_DAC_8043
//--SPI_AD7895------------------------------------
        input JC8,/*SDAT*/      output wire JC7,//SCLK_AD7895
        input JC9,/*BUSY*/      output wire JC10,//St_ADC
//--Мультиплексор сигнала FSK и данных DISPLAY------
        input [3:0]SW, //Управление режимом работы и отображением данных
//--Приемник FSK байта-----------------------------
                output wire JD1, // en_rx_byte
                output wire JD2, //URXD - декодированные данные FSK
                output wire JD3, //OCD - сигнал компаратора
                output wire JD4, //st_SDAC
//--Семи сегментнй светодиодный индикатор DISPLAY---
                output wire [4:0]AN, //Аноды светодиодов сегментов
                output wire [6:0]seg, //Сегменты
                output wire seg_P);   //Точка

wire clk, ce10ms, ce100ms, ce1s, en_rx_byte, st_SDAC, st_SADC ;
assign JB4 = st_SADC ;
assign JD1 = en_rx_byte ;
wire [7:0]TX_DAT ; //Данные генератора сигнала FSK
wire [7:0] Mamp ;     //Множитель амплитуды сигнала FSK
wire [11:0]FSK_SH; //Выход генератора FSK_byte
wire [11:0]MFSK ; //Выход модуля АЦП ADC_95
wire [11:0]FSK_IN = SW[3]? FSK_SH : MFSK ;//Мультиплексор
//---Глобальный буфер сигнала синхронизации---
BUFG DD1 (.I(F50MHz),   .O(clk));
//--Генерарор сигнала FSK_byte----------------
Gen_FSK_byte DD2 (
        .clk(clk),          .S(JB3),
        .st(ce100ms),       .FSK_SH(FSK_SH),
        .dat(TX_DAT),  .ce_SIN(JD4),
        .Mamp(Mamp),   .TXD(JB2),
                       .en_tx(JB1),
                       .ce_bit(JA1),
                       .ce_sd(st_SDAC));
//--Регулятор амплитуды сигнала FSK----------------
BTN_REG_AMP DD3 (
            .clk(clk),          .M(Mamp),
            .BTN_UP(BTN3),
            .BTN_DOWN(BTN0),
            .ce(ce10ms));
//--Реверсивный регулятор данных сигнала FSK-------
BTN_REG_DAT DD4 (
            .clk(clk),              .DAT(TX_DAT),
            .BTN_UP(BTN2),
```

```
            .BTN_DOWN(BTN1),
            .ce(ce10ms));
//--SPI_DAC8043-----------------------------------
SPI_DAC8043 DD5 (
        .clk(clk),          .SCLK(JA8),
        .st(st_SDAC),       .SDAT(JA9),
        .DI(FSK_SH),        .NLD(JA10));
//--АЦП FSK сигнала
ADC_95 DD6 (
    .clk(clk),          .ADC_5000(MFSK),//ADC_5000=(Uin/5.000V)*5000
    .SDAT(JC8),         .st_ADC(JC10),      //Convst
    .st(st_SADC),       .SCLK(JC7),         //Импульсы синхронизации
    .BUSY(JC9));
//--Приемник FSK байта-----------------------------
wire [7:0]RX_DAT ; //Данные FSK_RXD - приемника сигнала FSK
wire [15:0]Amin ;    //Минимальная амплитуда FSK
wire[15:0]TX_RX_DAT={TX_DAT,RX_DAT} ;
wire[11:0]bf_AMP ;
wire[15:0]AMPdec ;
wire[12:0]bf_SH ;
wire[15:0]SHdec ;
wire[15:0]AF1dec ;
wire[15:0]AF2dec ;

FSK_RXD DD7 (       .RX_dat(RX_DAT), //Принятый байт
    .FSK_IN(FSK_IN),.st_SADC(st_SADC), //Старт АЦП AD7895
    .clk(clk),      .en_rx_byte(en_rx_byte), //JD1
    .st(ce1s),      .URXD(JD2),         //JD2
                    .OCD(JD3),          //JD3
                    .Amin(Amin),        //Минимальная амплитуда FSK
                    .SHdec(SHdec),      //Смещение (DEC)
                    .AMPdec(AMPdec), //Амплитуда (DEC)
                    .AF1dec(AF1dec),    //Первая гармоника
                    .AF2dec(AF2dec),    //Вторая гармоника
                    .bf_SH (bf_SH),     //Смещение (HEX)
                    .bf_AMP (bf_AMP));//Амплитуда(HEX)

//--Мултиплексор данных для DISPLAY
wire[15:0]disp_dat=  (SW[2:0]==0)? TX_RX_DAT :
                     (SW[2:0]==1)? Amin :
                     (SW[2:0]==2)? SHdec :
                     (SW[2:0]==3)? AMPdec :
                     (SW[2:0]==4)? AF1dec :
                     (SW[2:0]==5)? AF2dec :
                     (SW[2:0]==6)? {3'b000,bf_SH} :
                                   {4'b0000,bf_AMP};
//--Семи сегментный светодиодный индикатор
DISPLAY DD8 (.clk(clk),         .AN(AN),            //Аноды
             .dat(disp_dat),    .seg(seg),          //Сегменты
             .PTR(SW[2:0]),     .seg_P(seg_P),      //Точка
                                .ce10ms(ce10ms),    //10 милисекунд
                                .ce100ms(ce100ms),//100 милисекунд
```

```
                                    .ce1s(ce1s));           //1 секунда
//--Модуль включения светодиодов
LED_BL DD9 (.DI(Mamp),      .DO(LED), //Mamp-включает только один светодиод
                .E(en_rx_byte));  //en_rx_byte - включает все светодиоды

endmodule
```