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_N93

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. Максимальное падение напряжения на входе управляемого устройства (ведомого) не должно превышать ~12 V. Минимальный уровень тока управления не может быть меньше 4 mA, поэтому он часто используется не только для управления, но и для питания электронных модулей ведомого. Например, при напряжении 12 V максимальная мощность потребления ведомого не должна превышать 4 mA* 12 V = 48 mW.

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 генерации и декодирования FSK HART сигнала выпускаются специальные микросхемы, например, DS8500.

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. Длительность старт бита равна длительности одного бита, а длительность стоп бита может составлять одну, полторы или две длительности бита. Данные передаются младшими битами вперед.

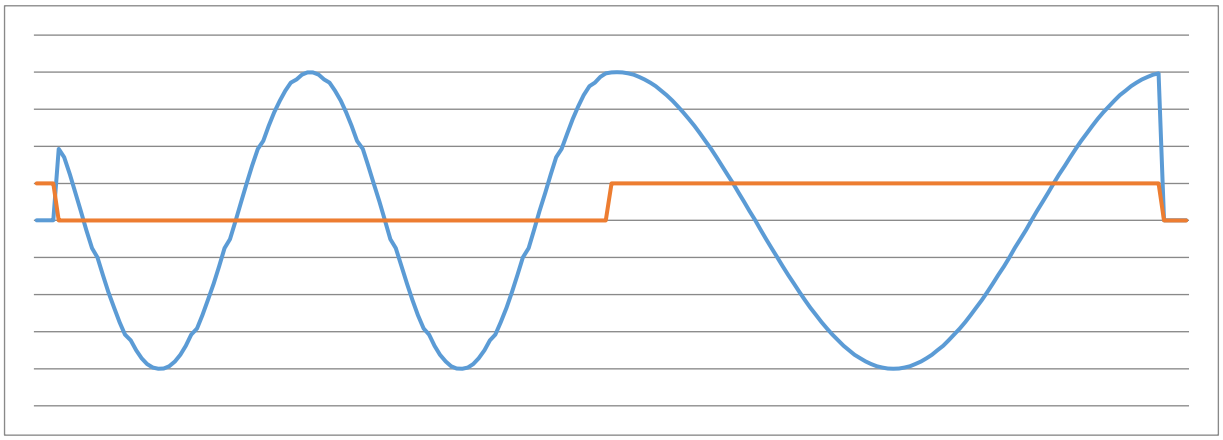


Рис.1 Пример временной диаграммы двух бит (0 и 1) FSK сигнала

HART протокол требует в каждом пакете перед блоком данных передавать «преамбулу» содержащую от 5 до 20 байт 8'hFF. Это, по видимому, необходимо для обеспечения возможности фазировки приемника, т.е. определения границ бит. Например, в непрерывном потоке «1» или «0» границы бит определить вообще невозможно.

Теория оптимального приема частотно модулированных сигналов давно и хорошо развита. Но аппаратная реализация ее достижений в данном случае очень громоздка. Это связано с неизвестностью и непрерывностью фаз синусоид «1» (1200 Hz) и «0» (2200 Hz) на границах бит.

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 (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 ($T_1 = T_{bit} = \frac{1}{1200Hz} = 833.33\mu s$).

Такая относительно небольшая задержка (T_d < 10% длительности бита) позволяет очень просто определять значение бита, например, по числу импульсов OCD (Output Carry Detect) превышения порога REF модулем сигнала FSK. Генератор импульсов границ тактов FSK_tact с периодом T_{bit} можно фазировать фронтом первого импульса OCD. Счетчик импульсов OCD cb_OCD надо считывать и сбрасывать тактовыми импульсами FSK_tact. Число импульсов OCD на интервале бита «0» не может быть меньше 3-х, а на итервале бита «1», если он второй, не может быть больше 2-х. Выходной сигнал RXN_bit = 0, если в конце такта cb_OCD>=3, и rx_bit = 1, если cb_OCD<3.

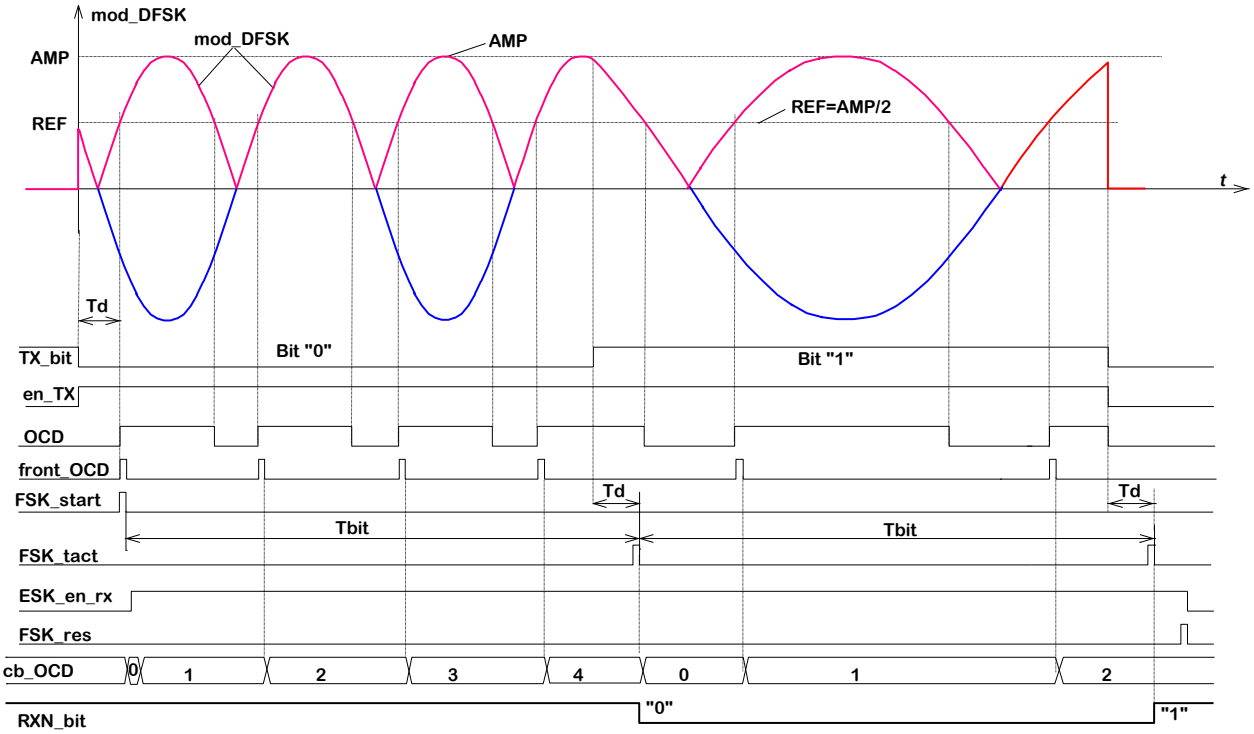


Рис.2 Пример временных диаграмм сигналов декодера бит FSK сигнала

Для автоматической установки порога REF на уровне половины амплитуды необходимо на первом такте пиковым детектором определить амплитуду и сравнивать модуль mod_DFSK задержанного сигала FSK с измеренным порогом. Амплитуда это половина разности, а среднее значение это пололовина суммы пиковых значений сигала FSK, поэтому интервал измерения должен быть не меньше периода. Первый бит это старт бит, т.е. «0», поэтому задержка сигнала FSK должна быть не меньше T_0.

Сигнал FSK_en_rx устанавливается в 1 импульсами OCD, а сбрасывается в 0 импульсом res в паузе после окончания пакета, т.е. если в течении половины такта нет импульсов OCD.

1. Генераторы сигналов FSK

1.1 Модуль генератора одного бита сигнала FSK

```
module Gen_FSK_1bit (
   input clk,          output wire [11:0]FSK_SH,  //Сигнал FSK со смещением
   input st,           output wire S,             //Знак синусоиды
   input D,            output wire T_bit,         //Интервал бита
   input [7:0] M,      output wire ce_SIN); //Сигнал дискретизации синусоиды

parameter SH=2048 ;  // Смещение для ЦАП на середину диапазона
wire[11:0]SIN ; //”Синусоида”
assign FSK_SH = T_bit? SIN+SH : SH ; // Импульс смещенной «синусоиды»
//--Ждущий генератор импульса Tbit=833 us (1/1200 Hz)
Gen_PW_1bit DD1 (.clk(clk),      .PW(T_bit),
                 .st(st));
//---Генератор ce для Gen_SIN (D=1 Fce=120kHz, D=0 Fce=220kHz)
Gen_Tce DD2 (    .clk(clk),      .CO(ce_SIN),
                 .D(D));
//---Генератор "синусоиды"--------------------
Gen_SIN DD3 (    .clk(clk),      .SIN(SIN), //SIN = S? Y : -Y ;
                 .ce(ce_SIN),    .S(S),   //S=1   sin>0
                 .M(M)); // Множитель
endmodule
```

1.1.1 Ждущий генератор импульса с длительностью одного бита

```
module Gen_PW_1bit (
        input clk,      output reg PW=0,
        input st);
parameter Fclk=50000000;
parameter F1 =        1200; // Частота «синусоиды» бита «1»
reg [15:0]cb_PW=0; // Счетчик длительности импульса.
wire  ce_end= (cb_PW==Fclk/F1); // Fclk/F1=41666

always @ (posedge clk) begin
cb_PW <= st? 1 : PW? cb_PW+1 : cb_PW ;
PW <= st? 1 : ce_end? 0 : PW ;
end
endmodule
```

1.1.2 Модуль генератора Tce для «синусоиды»

```
module Gen_Tce (    input clk,      output reg CO=0,
                    input D);
```

```
parameter Fclk=50000000 ;//Fclk=50MHz
parameter NP = 100  ; //Число точек синусоиды
parameter F1 = 1200 ; //Частота синусоиды сигнала "1"
parameter F0 = 2200 ; //Частота синусоиды сигнала "0"

wire [8:0]Nt = D? Fclk/(F1*NP) : Fclk/(F0*NP) ;
reg [8:0]cb_ce=0 ;
wire ce = (cb_ce==Nt) ;

always @ (posedge clk) begin
cb_ce <= ce? 1 : cb_ce+1 ;
CO <= ce ;
end
endmodule
```

1.1.3 Модуль генератора «синусоиды»

```
module Gen_SIN(          output reg S=1,          //S=1   sin>0
               input clk,    output reg [7:0] X=0, //Xmin<=X<=Xmax
               input ce,     output wire [10:0] Y, //Y=NA*sin(2*pi*X/NP)
               input[7:0]M,  output wire [11:0] SIN, //SIN = S? Y : -Y
                             output wire ceo);

parameter NP=100 ;           //Число точек
parameter Xmin=0 ;
parameter Xmax=NP/4 ;        //Xmax=`NP/4=25,
reg up=1 ;                   //Триггер направления счета
assign ceo = ce & (X==Xmin) & !S ;
//wire [11:0]SH=2048 ;       //Смещение
wire [19:0]MY=Y*M ;          //Умножение на M
wire[10:0] AY = MY >> 7 ;    //Деление на 128

always @ (posedge clk) if (ce) 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
//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;
endmodule
```

Частота «синусоиды» Fsin=Fce/100. Для бит «0» Fce=220 kHz, а для бит «1» Fce=120 kHz.

Амплитуда синусоиды AMP=2000*M/128 (128>=M>=1).

1.2 Модуль генератора одного байта сигнала FSK

```
module Gen_FSK_UART_byte (
        input [7:0]TX_dat,      output wire UTXD, //Последовательные данные
        input clk,              output wire [3:0]cb_bit, //Счетчик бит
        input st,               output wire en_tx, //Интервал передачи байта
        input [7:0]M,           output wire S, //Знак синусоиды
                                output wire [11:0] FSK_SH, //Синал FSK со смещением
                                output wire ce_SIN); //Сигнал дискретизации

wire ce_tact, T_stop ;
//---UART передатчик одного байта----------
UTXD1B DD1 (.clk(clk),      .TXD(UTXD), //Выход
            .dat(TX_dat),   .en_tx_byte(en_tx), //Интервал передачи
            .st(st),        .cb_bit(cb_bit),
                            .ce_tact(ce_tact),
                            .T_stop(T_stop));

//--Генератор 1 бита FSK
wire st_bit= st | (ce_tact & en_tx & !T_stop) ; //Импулсы запуска генератора бита FSK
Gen_FSK_1bit DD2 (.clk(clk),    .FSK_SH(FSK_SH),
                  .st(st_bit),  .S(S),
                  .D(UTXD),     .ce_SIN(ce_SIN),
                  .M(M) ); // Множитель
endmodule
```

1.2.1 Модуль UART передатчика одного байта

```
module UTXD1B(  input clk,      output wire TXD, //Выход
                input[7:0]dat,  output wire ce_tact, //Строб такта
                input st,       output reg en_tx_byte=0, //Интервал передачи
                                output reg [3:0] cb_bit=0,
                                output wire T_start, //Старт такт
                                output wire T_dat, //Интервал данных
                                output wire T_stop, //Стоп такт
                                output wire ce_stop, //Конец кадра
```

```
                                output reg [7:0] sr_dat=0 ); //Регистр сдвига данных

parameter Fclk=50000000 ; //50 MHz
parameter VEL = 1200 ; //1.2 kBod (из таблицы 1 вариантов)
parameter Nt = Fclk/VEL ; //434

reg [15:0]cb_tact=0 ;                   //Счетчик такта
assign ce_tact = (cb_tact==Nt) ;        //Строб такта
assign T_start=((cb_bit==0) & en_tx_byte);//Старт такт
assign T_dat = (cb_bit<9) & (cb_bit>0);  //Интервал данных
assign T_stop = (cb_bit==9) ;           //Стоп такт
assign ce_stop = T_stop & ce_tact ;     //Строб стопа
assign TXD = T_start? 0 : T_dat? sr_dat[0] : 1 ; //Последовательные данные
wire start = st & !en_tx_byte ;

always @ (posedge clk) begin
cb_tact <= (start | ce_tact)? 1 : cb_tact+1 ;
en_tx_byte  <= st? 1 : ce_stop? 0 : en_tx_byte ;
cb_bit <= start? 0 : (ce_tact & en_tx_byte)? cb_bit+1 : cb_bit ;
sr_dat <= (T_start & ce_tact)? dat : (T_dat & ce_tact)? sr_dat>>1 : sr_dat ;
end
endmodule
```

2. Приемники сигналов FSK

2.1 **Модуль приемника FSK сигнала одного бита**

```
module DET_FSK_bit_Nocd (
input clk,                  output wire [11:0]mod_DFSK, //Модуль DFSK
input [11:0]FSK_SH,  output reg RXN_bit=1, //Декодированый бит по количеству OCD
                            output reg OCD=0, // (OCD=(mod_DFSK>=AMP/2))
                            output reg [2:0]cb_OCD=0, //Счетчик импульсов OCD
                            output reg [2:0]N_OCD=0, //Буфер счетчика импульсов OCD
                            output wire FSK_tact, //Период FSK_tact Ttact=1/1200=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 wire [11:0] AMP, //Амплитуда
                            output wire [10:0] bf_AMP, //Буфер амплитуды
                            output wire [12:0] SH,//Смещение
                            output wire [11:0] bf_SH, //Буфер смещения
                            output wire [11:0] DFSK_SH, //Задержанный сигнал FSK
                            output wire ce_ADC, //Синал дискретизации
                            output wire FSK_res ); //Импульс "сброса" в паузе

parameter Amin=40; //Минимальная амплитуда
parameter dREF=Amin/4  ; //Гистерезис компаратора
parameter Fclk=50000000 ; //Частота сигнала синхронизации
parameter F1 =          1200 ; //Скорость передачи (частота следования бит)
parameter NP = 100 ; //Число точек синусоиды сигнала FSK
```

```
parameter Fadc= NP*F1 ; //Частота дискретизации

//--Измеритель смещения и амплитуды--------
Mes_AMP_SH  DD1 (
        .FSK_SH(FSK_SH),  .DFSK_SH(DFSK_SH), //Задержанный на такт сигнал FSK_SH
        .clk(clk),        .SH(SH), //Смещение
        .res(FSK_res),    .bf_SH(bf_SH), //Буфер смещения
                          .AMP(AMP), //Амплитуда
                          .bf_AMP(bf_AMP), //Буфер амплитуды
                          .ce_ADC(ce_ADC)); //Синал дискретизации
//-------------------------------------------
wire [11:0]DFSK = DFSK_SH-bf_SH ; //Вычитание измеренного среднего смещения
assign mod_DFSK=DFSK[11]? -DFSK : DFSK ; //Абсолютное значение (модуль) DFSK
wire [11:0]REF = AMP>>1 ; //Деление AMP на 2 (REF=AMP/2)
assign FSK_tact = (cb_tact==NP); //Fadc/F1

reg tOCD=0, tfront_OCD=0 ; //
wire front_OCD = (OCD & !tOCD);
assign FSK_start = front_OCD & !FSK_en_rx ; //

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

always @ (posedge clk) if (ce_ADC) begin
tOCD <= OCD; tfront_OCD <= front_OCD ;
OCD <=  ((mod_DFSK>=REF+dREF) & (mod_DFSK>Amin))? 1 :
          (mod_DFSK<=REF-dREF)? 0 : OCD ;
FSK_en_rx <= front_OCD? 1 : FSK_res? 0 : FSK_en_rx ;
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 ;
cb_OCD <= (FSK_tact | FSK_start)? 0 : tfront_OCD? cb_OCD+1 : cb_OCD ;
N_OCD <= FSK_start? 0 : FSK_tact? cb_OCD : N_OCD ;
RXN_bit <= (FSK_tact & (cb_OCD>=3) & FSK_en_rx)? 0 : FSK_tact? 1 : RXN_bit ; //
end
endmodule
```

2.1.2 **Модуль измерения амплитуды и смещения**

```
module Mes_AMP_SH(
        module Mes_AMP_SH(
        input [11:0] FSK_SH,        output wire [11:0] DFSK_SH, //Задержанный сигнал FSK
        input clk,                  output wire [12:0] SH, //Смещение
        input res,                  output reg [12:0] bf_SH=12'h800, //Среднее смещение
                                    output wire [12:0] AMP,//Амплитуда
                                    output reg [12:0] bf_AMP=2000, //Средняя амплитуда
                                    output wire ce_ADC); //input st,
parameter Fclk=50000000 ; //Fclk=50MHz
parameter F1=1200 ; //F1=1200Hz
parameter ND=100 ; //Число тактов задержки
reg [10:0] cb_ce =0 ;
assign ce_ADC = (cb_ce==Fclk/(F1*ND)) ;
reg [6:0] Adr_wr=0 ;
wire [6:0] Adr_rd = Adr_wr-ND ;//Задержка на Tbit=100*Tce=T1
```

```
always @ (posedge clk) begin
cb_ce <= (ce_ADC)? 1 : cb_ce+1 ; //
Adr_wr <= ce_ADC? Adr_wr+1 : Adr_wr ;
end
//--Модуль памяти для задержки сигнала FSK
MEM12x128 DD2 (.clk(clk),                    .DO(DFSK_SH),
                                             .we(ce_ADC),
                                             .DI(FSK_SH),
                                             .Adr_wr(Adr_wr),
                                             .Adr_rd(Adr_rd));
reg [12:0]PIC_max = 2049; //Регистр пикового детектора максимума сигнала FSK
reg [12:0]PIC_min = 2047; //Регистр пикового детектора минимума сигнала FSK
assign SH = (PIC_max+PIC_min)>>1 ; //Полусумма
assign AMP= (PIC_max-PIC_min)>>1 ; //Полуразность
//--Получение максимума и минимума сигнала FSK
always @ (posedge clk) if (ce_ADC) begin
PIC_max <= res? 12'h801 : (FSK_SH>PIC_max)? FSK_SH : PIC_max ;
PIC_min <= res? 12'h7FF : (FSK_SH<PIC_min)? FSK_SH : PIC_min ;
end
//--Реверсивные следящие измерители смещения и амплитуды
wire [12:0] dAMP = AMP-bf_AMP ; //Разность
wire [12:0] mod_dAMP = dAMP[12]? -dAMP : dAMP ; //Модуль разности
always @ (posedge clk) if (ce_ADC & res) begin
bf_SH <= (SH[11:0]>bf_SH)? bf_SH+1 : (SH[11:0]<bf_SH)? bf_SH-1 : bf_SH ;
bf_AMP <= dAMP[12]? bf_AMP-(mod_dAMP>>3) : bf_AMP+(mod_dAMP>>3) ;
end
endmodule
```

2.1.3 **Модуль памяти для задержки FSK сигнала**

```
module MEM12x128(
        input clk,              output wire [11:0] DO,
        input we,               //output reg [11:0] DO,
        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
```

init_MEM12x128.txt - текстовый файл инициализации модуля памяти надо создать самостоятельно в любом текстовом редакторе. Он содержит 128 строк 800 (12'h800=2048).

2.1.4 **Схема моделирования детектора FSK бита**

```
module Test_DET_FSK_bit(
        input clk,              output wire [11:0]FSK_SH,
```

```
        input st,               output wire S,
        input TX_bit,           output wire T_bit,
        input [7:0] M,          //output wire ce_SIN,
                        //--Сигналы детектора FSK бита ------
                                output wire [11:0]mod_DFSK, //Модуль DFSK
                                output wire [11:0] DFSK_SH, //
                                output wire RXN_bit, //Декодированый бит по количеству OCD
                                output wire OCD, // (OCD=(mod_DFSK>=AMP/2))
                                output wire [2:0]cb_OCD, //Счетчик импульсов OCD
                                output wire [2:0]N_OCD, //Буфер счетчика импульсов OCD
                                output wire FSK_tact, //Период FSK_tact  = 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 [11:0] AMP, //Амплитуда
                                output wire [10:0] bf_AMP, //Буфер амплитуды
                                //output wire [12:0] SH, //Смещение
                                output wire [11:0] bf_SH, //Буфер смещения
                                output wire ce_ADC, //Синал дискретизации
                                output wire FSK_res); //Импульс "сброса" в паузе

//--Генератор сигнала FSK 1-го бита
Gen_FSK_1bit DD1 (.clk(clk),      .FSK_SH(FSK_SH),
                  .st(st),        .S(S),
                  .D(TX_bit),     .T_bit(T_bit),
                  .M(M));

//--Детектор FSK бита по числу импульсов OCD
DET_FSK_bit_Nocd DD2 (
        .FSK_SH(FSK_SH),  .DFSK_SH(DFSK_SH),
         .clk(clk),       .mod_DFSK(mod_DFSK),
                          .OCD(OCD),
                          .cb_OCD(cb_OCD),
                          .N_OCD(N_OCD),
                          .bf_AMP(bf_AMP),
                          .bf_SH(bf_SH),
                          .FSK_res(FSK_res),
                          .FSK_en_rx(FSK_en_rx),
                          .ce_ADC(ce_ADC),
                          .RXN_bit(RXN_bit),
                          .FSK_start(FSK_start),
                          .FSK_tact(FSK_tact));
endmodule
```

2.1.5,6 **Варианты входных сигналов для Verilog Test Fixture моделирования Test_DET_FSK_bit**

```
//Вариант 1. Первый бит «0» второй бит «1»
always begin clk=0; #10; clk=1; #10; end
initial begin
                st = 0;  TX_bit = 0;    M = 0;
# 188000;       st = 1;  TX_bit = 0;    M = 128; //Старт сигнала первого бита
#20;            st = 0;
```

```
#833000;        st = 1;  TX_bit = 1; // Старт сигнала второго бита
#20;            st = 0;
end

// Вариант 2. Первый бит «1» второй бит «0»
always begin clk=0; #10; clk=1; #10; end
initial begin
                st = 0;  TX_bit = 1;    M = 0;
#358500;        st = 1;  TX_bit = 1;    M = 128; //Старт сигнала первого бита
#20;            st = 0;
#833000;        st = 1;  TX_bit = 0; // Старт сигнала второго бита
#20;            st = 0;
end
```

2.2 **Модуль FSK приемника одного байта**

```
module RX_FSK_byte_Nocd(
        input [11:0]FSK_SH, output wire [11:0] DFSK_SH,
        input clk,          output wire [11:0] mod_DFSK,
                            output wire [10:0] bf_AMP,
                            output wire [11:0] bf_SH,
                            output wire OCD,
                            output wire [2:0]cb_OCD,
                            output wire [2:0]N_OCD,
                            output wire ce_ADC,
                            output wire FSK_start,
                            output wire FSK_tact,
                            output wire FSK_en_rx,
                            output wire FSK_res,
                            output wire URXD,
                            output wire en_rx_byte,
                            output wire ok_rx_byte,
                            output wire [7:0] RX_dat,
                            output wire [3:0]UART_cb_bit);
//--Детектор FSK бита по числу импульсов OCD
DET_FSK_bit_Nocd DD1 (
                .FSK_SH(FSK_SH),  .DFSK_SH(DFSK_SH),
                .clk(clk),        .mod_DFSK(mod_DFSK),
                                  .OCD(OCD),
                                  .cb_OCD(cb_OCD),
                                  .N_OCD(N_OCD),
                                  .bf_AMP(bf_AMP),
                                  .bf_SH(bf_SH),
                                  .FSK_res(FSK_res),
                                  .FSK_en_rx(FSK_en_rx),
                                  .ce_ADC(ce_ADC),
                                  .RXN_bit(URXD),
                                  .FSK_start(FSK_start),
                                  .FSK_tact(FSK_tact));

//--Приемник байта
URXD1B DD2 (        .inp(URXD),   .rx_dat(RX_dat),
```

```
                    .clk(clk),      .ok_rx_byte(ok_rx_byte),
                                    .cb_bit(UART_cb_bit),
                                    .en_rx_byte(en_rx_byte));
endmodule
```

2.2.1 **Модуль UART приемника одного байта**

```
module URXD1B (   input inp,      output reg en_rx_byte=0,
                  input clk,      output reg [7:0] rx_dat=0,
                                  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) ; //Границы тактов
reg [7:0] sr_dat=0 ; //Регистр сдвига приема бит данных
wire spad_inp = !tin & ttin ; //Спад входного сигнала inp
wire start_rx = spad_inp & !en_rx_byte ;
assign ce_bit = (cb_tact==Fclk/(2*F1)); //Импульсы середины тактов
assign ok_rx_byte = (ce_bit & (cb_bit==9) & en_rx_byte & tin);
assign T_dat = ((cb_bit>=1) & (cb_bit<9));

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 ;
rx_dat <= ok_rx_byte? sr_dat : rx_dat ;
end
endmodule
```

2.2.2 **Схема моделирования приемника FSK байта**

```
module Test_RX_FSK_byte (
        input [7:0]TX_dat,      output wire UTXD,//Последовательные данные
        input clk,              output wire [3:0]cb_bit,//Счетчик бит
        input st,               output wire en_tx,//Интервал передачи байта
        input [7:0]M,           output wire S,//Знак синусоиды

                                output wire [11:0] FSK_SH,//Синал FSK со смещением
                                output wire ce_SIN, //Сигнал дискретизации
                                output wire [11:0] DFSK_SH,
                                output wire [11:0] mod_DFSK,
                                output wire [10:0] bf_AMP,
                                output wire [11:0] bf_SH,
                                output wire OCD,
                                output wire [2:0]cb_OCD,
                                output wire [2:0]N_OCD,
```

```
                                output wire ce_ADC,
                                output wire FSK_start,
                                output wire FSK_tact,
                                output wire FSK_en_rx,
                                output wire FSK_res,
                                output wire URXD,
                                output wire en_rx_byte,
                                output wire ok_rx_byte,
                                output wire [7:0] RX_dat,
                                output wire [3:0]UART_cb_bit);
//--Генератор FSK сигнала одного байта
Gen_FSK_UART_byte DD1(
        .TX_dat(TX_dat),        .UTXD(UTXD), //Последовательные данные
        .clk(clk),              .cb_bit(cb_bit), //Счетчик бит
        .st(st),                .en_tx(en_tx), //Интервал передачи байта
        .M(M),                  .S(S),//Знак синусоиды
                                .FSK_SH(FSK_SH), //Синал FSK со смещением
                                .ce_SIN(ce_SIN)); //Сигнал дискретизации

//--Приемник FSK сигнала одного байта
RX_FSK_byte_Nocd DD2(
        .FSK_SH(FSK_SH),    .DFSK_SH(DFSK_SH),
        .clk(clk),          .mod_DFSK(mod_DFSK),
                            .bf_AMP(bf_AMP),
                            .bf_SH(bf_SH),
                            .OCD(OCD),
                            .cb_OCD(cb_OCD),
                            .N_OCD(N_OCD),
                            .ce_ADC(ce_ADC),
                            .FSK_start(FSK_start),
                            .FSK_tact(FSK_tact),
                            .FSK_en_rx(FSK_en_rx),
                            .FSK_res(FSK_res),
                            .URXD(URXD),
                            .en_rx_byte(en_rx_byte),
                            .ok_rx_byte(ok_rx_byte),
                            .RX_dat(RX_dat),
                            .UART_cb_bit(UART_cb_bit));
endmodule
```

2.2.3 **Входные сигналы для Verilog Test Fixture моделирования Test_RX_FSK_byte приемника FSK байта**

```
always begin clk=0; #10; clk=1; #10; end
        initial begin
                start = 0;      TX_dat = 0;         M = 0;
#358500;        start = 1;      TX_dat = 8'h5A;     M = 128;
#20;            start = 0;

#10000000;      start = 1;      TX_dat = 8'hD4;     M = 128;
#20;            start = 0;
        end
```

3. Задание к допуску (стоимость 2)

3.1 Начертить в тетради временные диаграммы сигналов рис.2.

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

3.3 Переписать в тетрадь схему модуля **DET_FSK_bit_Nocd** модуля приемника FSK сигнала одного бита.

4. 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_NS, 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"

4.1 Создать Verilog модуль **Gen_FSK_1bit** и, входящие в его состав, модули: **Gen_PW_1bit**, **Gen_Tce** и **Gen_SIN**.

4.2 Создать Verilog модуль **DET_FSK_bit_Nocd** и, входящие в его состав, модули: **Mes_AMP_SH** и **MEM12x128**. Для модуля памяти создать текстовый фай инициализации **init_MEM12x128.txt** (128 строк: 800).

4.3 Создать Verilog модуль **Test_DET_FSK_bit** схемы для моделирования детектора FSK бита.

4.4 Создать Verilog модуль **Gen_FSK_UART_byte** и модуль **UTXD1B** UART передатчика одного байта.

4.5 Создать Verilog модуль **RX_FSK_byte_Nocd** и модуль **URXD1B** UART приемника одного байта.

4.6 Создать Verilog модуль **Test_RX_FSK_byte** схемы для моделирования детектора FSK байта.

4.7 Выполнить синтез каждого созданного модуля. Исправить ошибки (Errors), обратить внимание на предупреждения (Warnings).

4.8 Создать для схемы **Test_DET_FSK_bit** модуль **tf_Test_DET_FSK_bit** задания на моделирование (Verilog Test Fixture). Установить необходимые данные входных сигналов (см. п. 2.1.5,6).

4.9 Создать для схемы **Test_RX_FSK_byte** модуль **tf_Test_RX_FSK_byte** задания на моделирование (Verilog Test Fixture). Установить необходимые данные входных сигналов (см. п. 2.2.3).

4.10 Провести моделирование схемы **Test_DET_FSK_bit** детектора FSK бита для двух вариантов пары бит («0,1») и («1,0») при Simulation Run Time = 5 ms. Определить

минимальную амплитуду FSK сигнала. Проверить соответствие временных диаграмм рис.2, полученным временным диаграммам сигналов варианта пары бит («0,1»).

4.11 Провести моделирование схемы **Test_RX_FSK_byte** приемника FSK байта при Simulation Run Time = 25 ms. Проверить правильность приема первого и второго байта. Зарисовать эскизы временных диаграмм первого байта: UTXD, cb_bit, en_tx, N_OCD, FSK_start, FSK_en_rx, FSK_res, URXD, en_rx_byte.

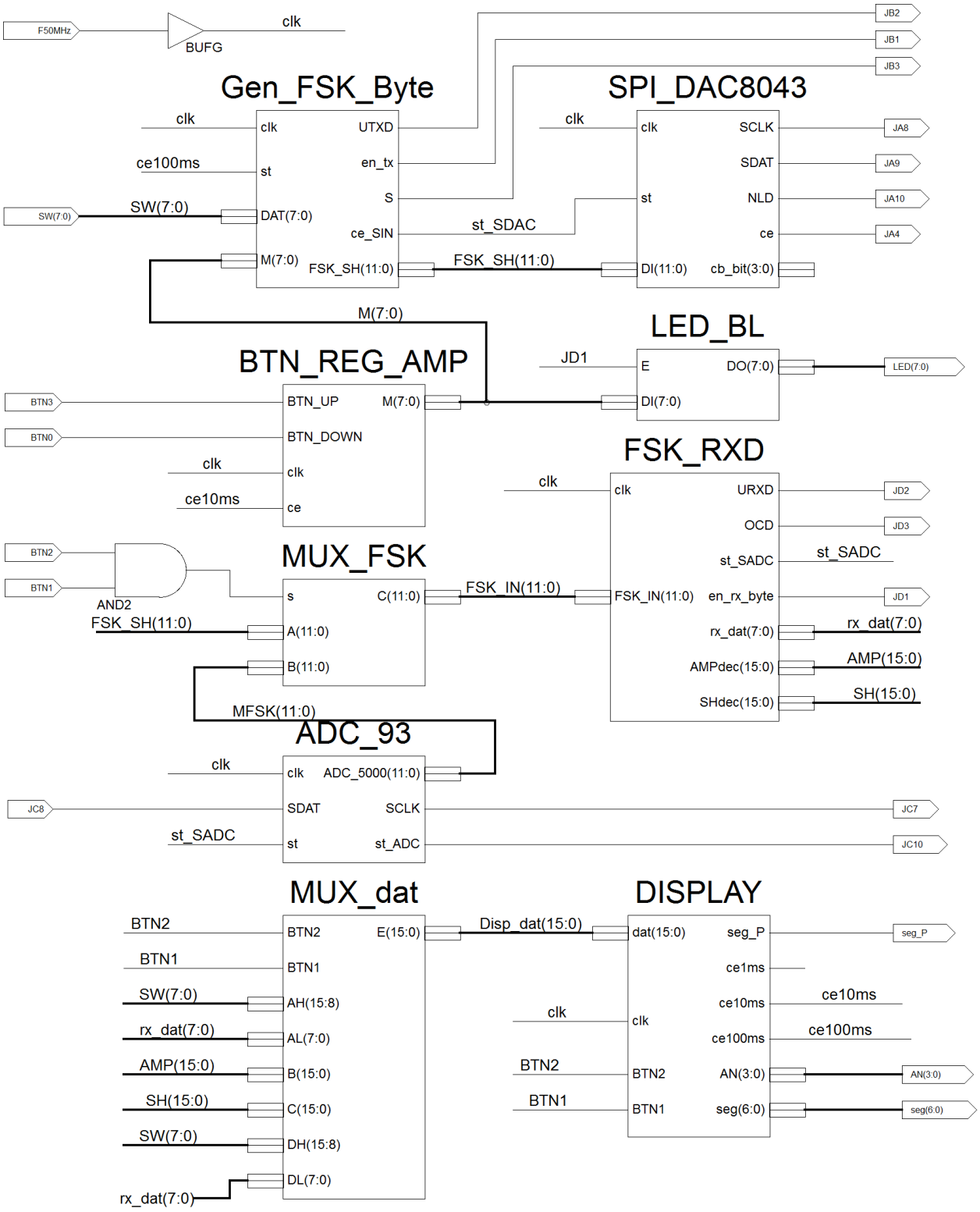


Рис. 3 Схема лабораторной работы S_Sch_Lab414_N93

В этой схеме переключателями SW[7:0] устанавливаются данные передаваемого байта, кнопками BTN3 и BTN1 регулируется амплитуда, кнопками BTN1 и BTN2 устанавливается адрес отображаемых данных.

- {BTN2,BTNT1} = {0,0} - Disp_dat= {tx_dat, rx_dat},
- {BTN2,BTNT1} = {1,0} - Disp_dat = bf_SH (смещение сигнала FSK, X.XXX V),
- {BTN2,BTNT1} = {0,1} - Disp_dat = bf_AMP (амплитуда сигнала FSK, X.XXX V),
- {BTN2,BTNT1} = {1,1} - Disp_dat= {tx_dat, rx_dat}, но в отличие от {0,0} на вход приемника **FSK_RXD** подаются данные не с выхода модуля **ADC_93**, а непосредственно с выхода цифрового генератора **Gen_FSK_byte**.

5. Задание к сдаче (стоимость 4)

Для схемы **S_Sch_Lab414_NS93** (рис.3) дополнительно создать модули и символы:

- **SPI_DAC8043** (приложение 6.1.1),
- **SPI_AD7893** (приложение 6.2.2),
- **MULT_5000_DIV_4096** (приложение 6.2.3),
- **ADC_93** (приложение 6.2.4),
- **FSK_RXD** (приложение 6.3),
- **BIN12_to_DEC4** (приложение 6.3.1),
- **BTN_REG_AMP** (приложение 6.4),
- **MUX_dat** (приложение 6.5),
- **LED_BL** (приложение 6.6),
- **DISPLAY** (приложение 6.7).

Схему модуля MUX_FSK составить самостоятельно (C=A, если S=1, иначе C=B)

5.1 Создать лист схемы **S_Sch_Lab414_N93** (New Source - Schematic). Из созданных символов составить схему рис.3. Выполнить синтез (Synthesize - XST), исправить ошибки.
5.2 Для текстового варианта схемы **V_Sch_Lab414_N93** (приложение 6.9) символы создавать не надо, кроме того можно не создавать и модули **FSK_RXD** и **MUX_dat**.

5.3 Для выбранного варианта **S_Sch_Lab414_N93** или **V_Sch_Lab414_N93** создать *.ucf (New Source - Implementation Constraints File) (приложение 6.8).

5.4 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 AD7893. В JP3 вставить перемычку на +5V.

Соединить проводной перемычкой выход XN3 MDAC с входом XN1 SADC (см. рис.15)..

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

5.7 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 M (BTN3,BTN0) и переключателей SW[7:0] на амплитуду и форму FSK сигнала. Определить минимальную амплитуду FSK сигнала при

которой еще нет ошибок приема байта. Сохранить осциллограммы сигналов с выходов XN2 SADC (Uadc) и JD2 (URXD) при максимальной и минимальной амплитуде.

6. Приложения

6.1 Множительный цифроаналоговый преобразователь DAC8043

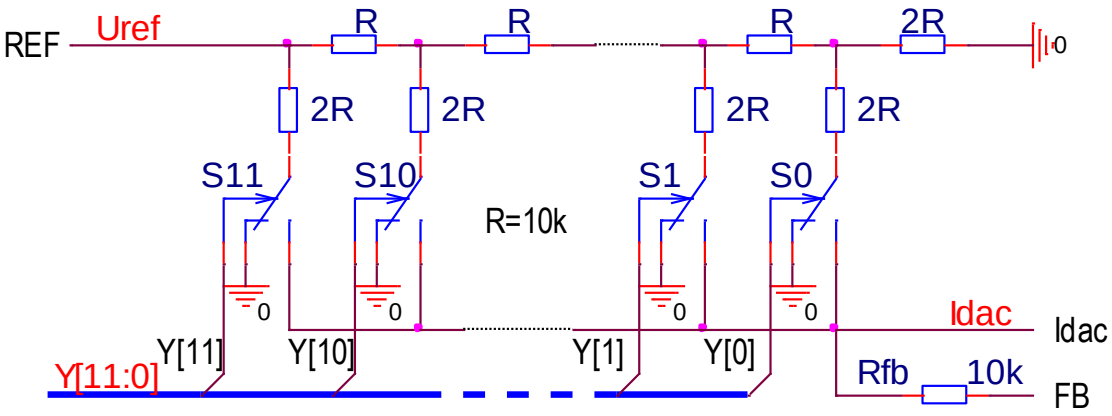


Рис.4 Схема цифроаналогового преобразователя (ЦАП) DAC8043 с переключателями тока S11,S10,…S1,S0

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

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

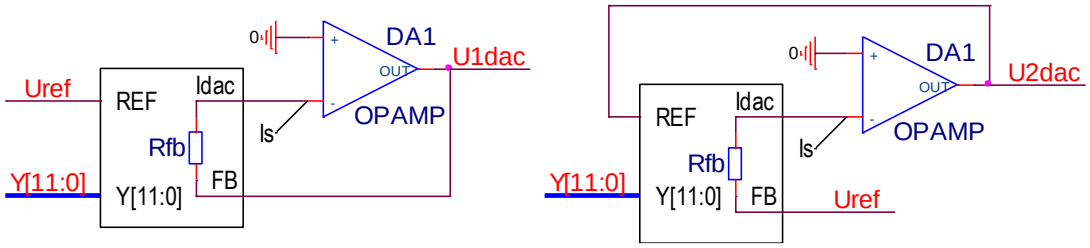
или $I_{dac} = \frac{U_{ref}}{R}\frac{Y}{2^{12}}$.

Для схемы рис.5.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.5.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. 5 Цифроаналоговые преобразователи: а – с прямой и б –обратной зависимостью выходного напряжения от числа Y

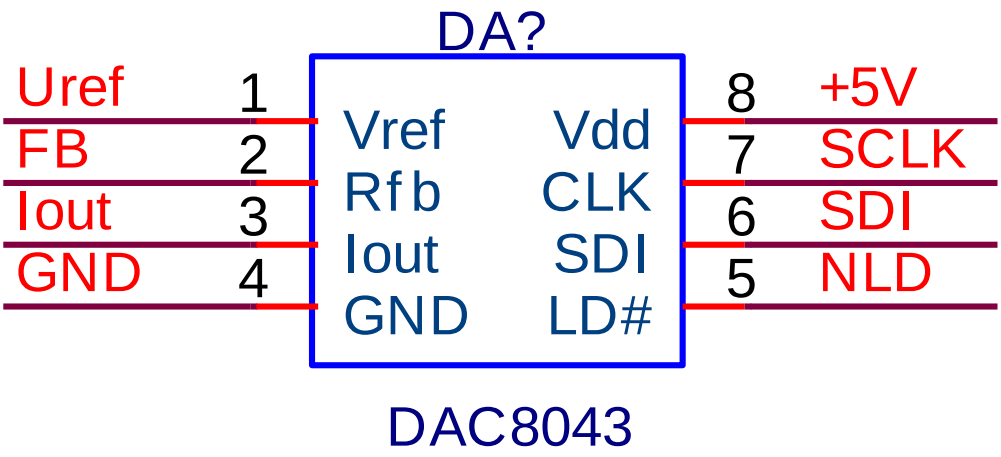


Рис.6 Выводы множительного цифроаналогового преобразователя DAC8043

2.1 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).

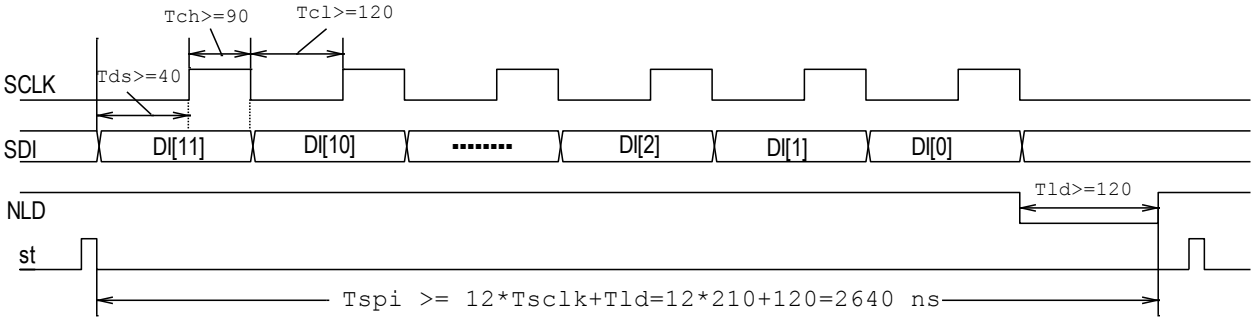


Рис.7 Временные диаграммы SPI интерфейса цифроаналогового преобразователя DAC8043 (размерность интервалов времени – нс)

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.

6.1.1 **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 ;
assign ce=(cb_ce==Tsclk/Tclk);
reg en_sh=0 ;
reg [11:0]sr_dat=0 ;
assign SDAT = sr_dat[11];

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
```

6.1.2 Макет цифроаналогового преобразователя MDAC

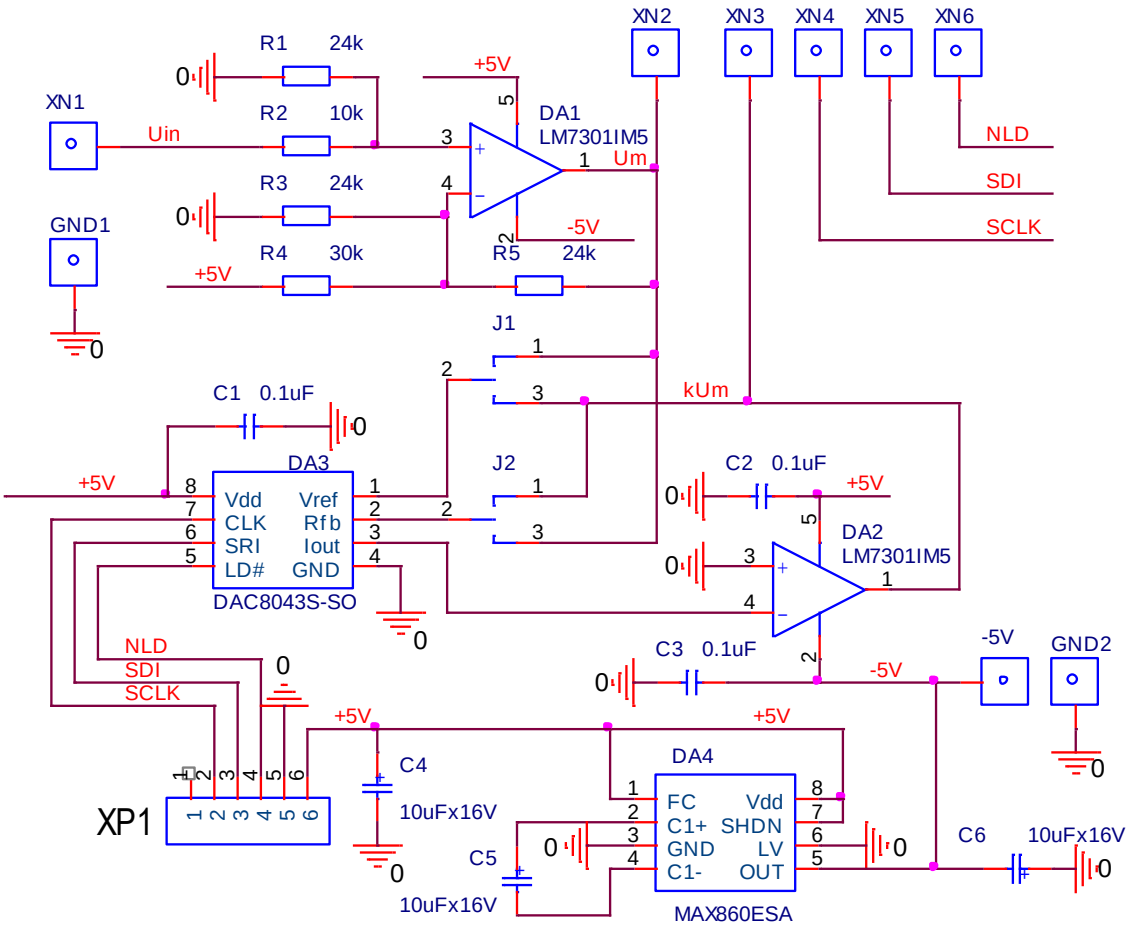


Рис.8 Схема макета MDAC цифроаналогового преобразователя DAC8043

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)

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

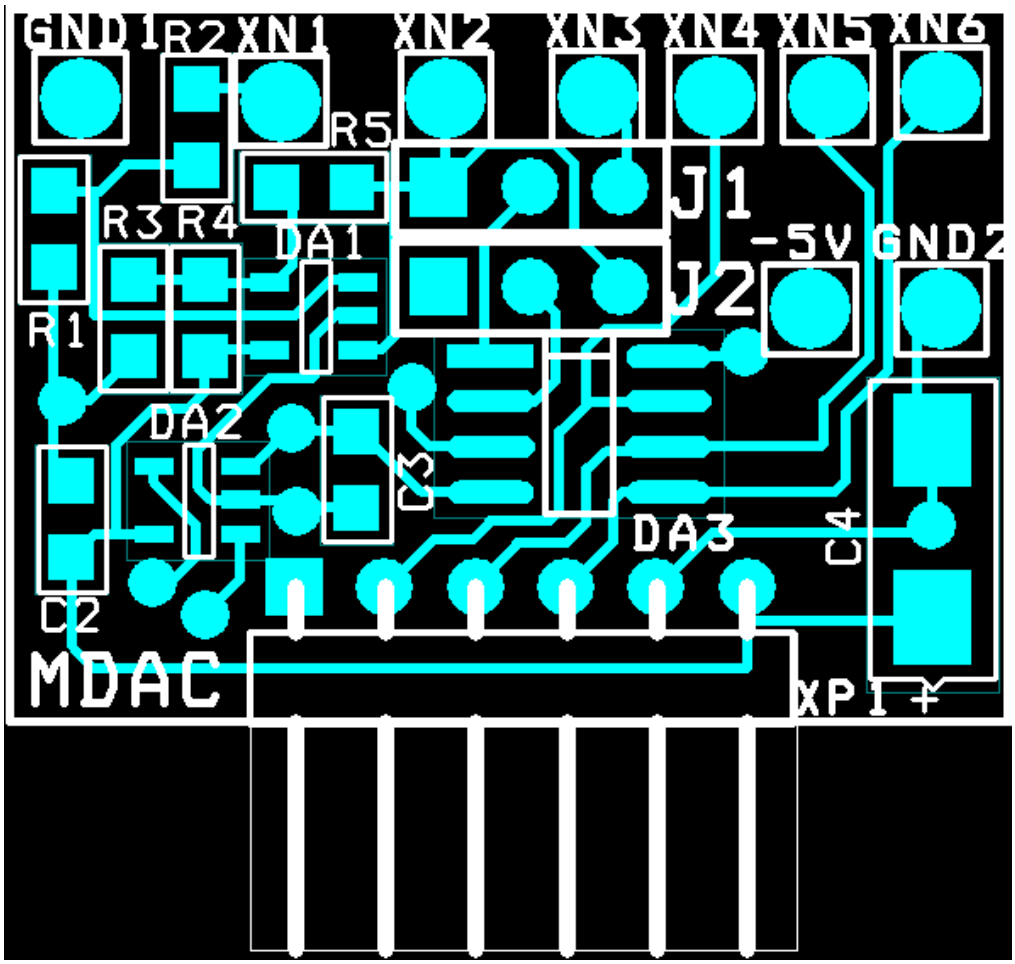


Рис.9 Печатная плата макета множительного цифроаналогового преобразователя DAC8043 (контактные площадки J1.1 и J2.1 – квадратные)

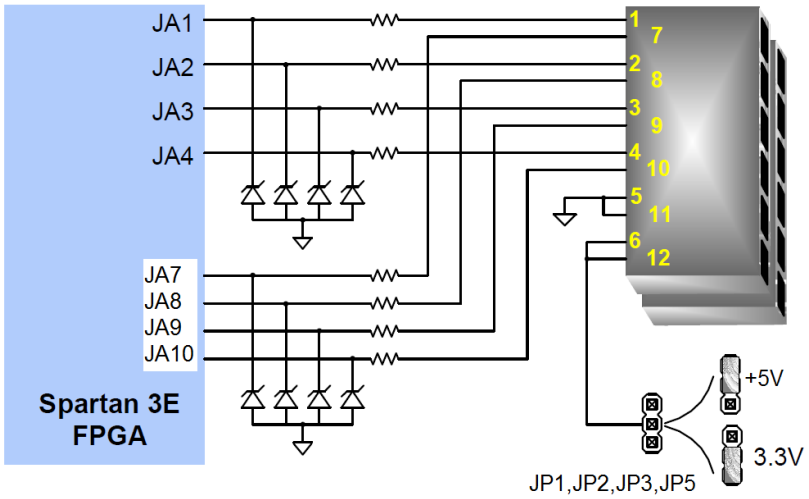


Рис.10 Порты макета NEXYS2

6.2 Аналого-цифровой преобразователь (АЦП) AD7893-5

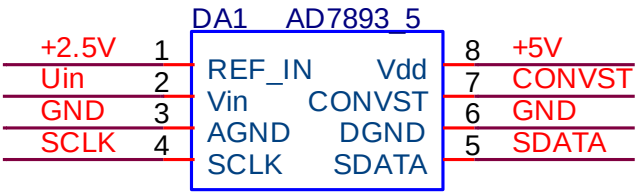


Рис.11 Микросхема аналого-цифрового преобразователя AD7893-5

6.2.1 Назначение выводов АЦП DAC7893-5

[см. Data sheet AD7893.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 AD7893 is +2.5 V.

Pin2 Vin - Analog Input Channel. The analog input range is 0 V to +5 V (AD7893-5).

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 AD7893. 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 AD7893 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 AD7893-5.

Pin6 DGND - Digital Ground. Ground reference for digital circuitry.

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 AD7893-5 при U_{REF} = +2.5V от 0 до +5 V (FSR=5V).
LSB - вес младшего разряда для AD7893-5, LSB=FSR/4096=0.122 mV.

Входное напряжение Vin	Выходные данные (HEX) ADC_dat[11:0]
+FSR – 1 LSB	от 12'hFFE до 12'hFFF
+FSR – 2 LSB	от 12'hFFD до 12'hFFE
+FSR – 3 LSB	от 12'hFFC до 12'hFFD
GND + 3 LSB	от 12'h002 до 12'h003
GND + 2 LSB	от 12'h001 до 12'h002
GND + 1 LSB	от 12'h000 до 12'h001

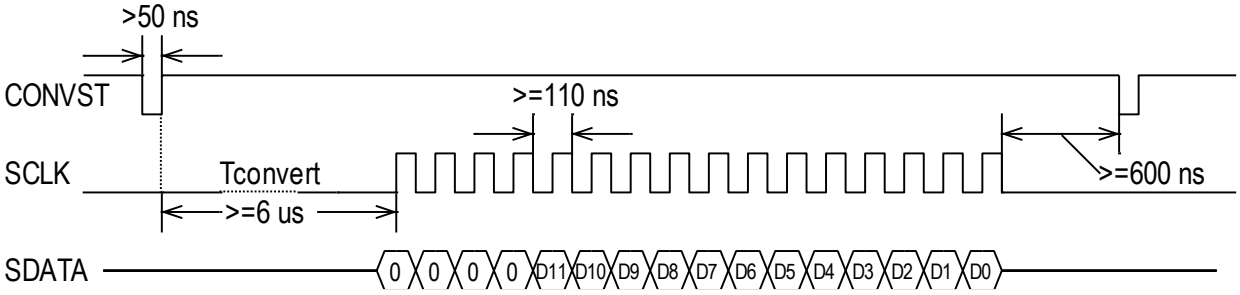


Рис.12 Временные диаграммы сигналов АЦП AD7893

По спаду импульса запуска CONVST начинается процесс последовательного приближения, который продолжается в течение 6 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.

6.2.2 SPI модуль считывания данных АЦП AD7893

```
module SPI_AD7893(
        input clk,          output reg SCLK=0,//Импульсы синхронизации
        input SDAT,         output reg st_ADC=1,//
        input st,           output reg [11:0]ADC_dat=0,
                            output reg ok_adc=0,
                            output reg [11:0]sr_DAT=0);//

reg [4:0]cb_bit=0;
reg T_CONV=0;
reg [2:0]cb_ce=0 ;
wire ce=(cb_ce==5);//Tce=8*20=160 ns
reg [7:0]cb_T_CONV=0 ;
wire end_T_CONV = (cb_T_CONV==50) & ce ;//160*36=5760ns (было 40)
wire stop_SCLK = (cb_bit==15) & ce;
reg en_SCLK=0 ;
```

```
wire start = st & !(T_CONV | en_SCLK) ;

always @ (posedge clk) begin
cb_ce <= (start | ce)? 0 : cb_ce+1 ;
st_ADC <= start? 0 : (cb_ce==2)? 1 : st_ADC ;
cb_T_CONV <= start? 1 : (T_CONV & ce)? cb_T_CONV+1 : cb_T_CONV ;
T_CONV <= start? 1 : end_T_CONV? 0 : T_CONV ;
en_SCLK <= end_T_CONV? 1 : stop_SCLK? 0 : en_SCLK ;
cb_bit <= start? 0 : (en_SCLK & ce)? cb_bit+1 : cb_bit ;
SCLK <= (start | (en_SCLK & ce))? 0  : (en_SCLK & (cb_ce==2))? 1 : SCLK ;// (cb_ce==2)
ok_adc <= (stop_SCLK & ce)? 1 : 0 ;
ADC_dat <= ok_adc? sr_DAT[9:0]<<2 : ADC_dat ; /*На MDAC к этой работе AD7893 10-и битные*/
end

always @ (negedge SCLK) begin
sr_DAT <= start? 0 : sr_DAT<<1 | SDAT ;
end
endmodule
```

В этом модуле после импульса старта CONVST начало процесса считывания данных задерживается на 6.4 мкс счетчиком cb_BUSY и регистром BUSY. Минимальный период запуска 9.56 мкс.

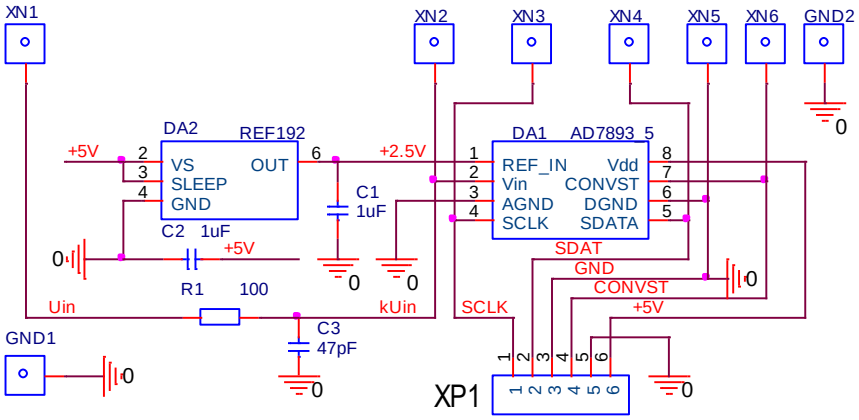


Рис.13 Схема макета SADC АЦП AD7893-5

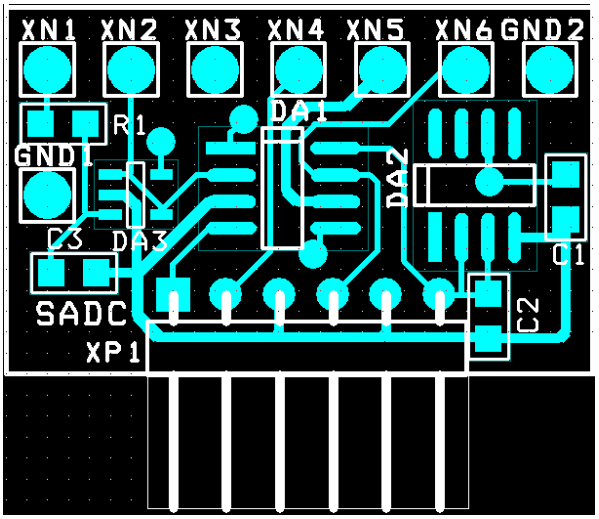


Рис.14 Лицевая сторона печатной платы макета SADC АЦП AD7893-5

Штыри печатной платы макета AD7893 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.8). JC5 и JC11 соединены с GND.

Данные ADC_{dat} линейно связаны с входным напряжением U_{in}, $ADC_{dat} = 4096\frac{U_{in}}{5V}$. При соединении выхода (XN3) макета ЦАП с входом (XN1) макета АЦП

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

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

6.2.3 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
```

6.2.4 Модуль ADC со встроенным умножителем

```
module ADC_93(
        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

wire [11:0]ADC_4096 ; //ADC_4096=(Uin/4.096V)*4096
SPI_AD7893 DD1 (  .clk(clk),            .SCLK(SCLK),
                  .st(st),              .st_ADC(st_ADC),
                  .SDAT(SDAT),          .ADC_dat(ADC_4096));

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

endmodule
```

6.3 Модуль приемника FSK байта со встроенными преобразователями DIN12_to_DEC4

```
module FSK_RXD(         output wire [7:0] rx_dat,
        input [11:0]FSK_IN, output wire [15:0] AMPdec, //
        input clk,          output wire [15:0] SHdec, //
                            output wire URXD, //JD2
                            output wire OCD, //JD3
                            output wire st_SADC,
                            output wire en_rx_byte);
wire ok_rx_byte ;
wire [11:0]AMP ; wire [12:0]SH ;

RX_FSK_byte_Nocd DD1 (
                .clk(clk),          .RX_dat(rx_dat),
```

```
                .FSK_SH(FSK_IN),  .en_rx_byte(en_rx_byte),
                                  .URXD(URXD),
                                  .OCD(OCD),
                                  .bf_AMP(AMP),
                                  .bf_SH(SH),
                                  .ok_rx_byte(ok_rx_byte),
                                  .ce_ADC(st_SADC));

BIN12_to_DEC4 DD2 (     .BIN(AMP),          .DEC(AMPdec),
                        .st(ok_rx_byte),
                        .clk(clk));

BIN12_to_DEC4 DD3 (     .BIN(SH),           .DEC(SHdec),
                        .st(ok_rx_byte),
                        .clk(clk));
endmodule
```

6.3.1 Двоично десятичный преобразователь

```
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 ok=0,      //Конец преобразования
                                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
```

6.4 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 ce10ms

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
```

6.5 Модуль мультиплексора данных для Display

```
module MUX_dat(
        input BTN2,         output wire [15:0] E,
        input BTN1,
        input [15:8] AH,
        input [ 7:0] AL,
        input [15:0] B,
        input [15:0] C,
        input [15:8] DH,
        input [ 7:0] DL     );

wire [1:0]adr= {BTN2,BTN1} ;
assign E =      (adr==2'b00)? {AH,AL} :
                (adr==2'b01)? B :
                (adr==2'b10)? C :
                              {DH,DL} ;
endmodule
```

6.6 Модуль светодиодов

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

6.7 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 BTN2,             output wire seg_P,     //Точка
                input BTN1,             output reg ce1ms,      //1 милисекунда
                                        output wire ce10ms,   //10 милисекунд
                                        output reg ce100ms=0); //0.1 секунда

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

wire [1:0]PTR = {BTN2,BTN1} ;
wire [1:0]ptr_P =       (PTR==0)? 2'b10 : //Точка в центре (XX.XX)
                        (PTR==1)? 2'b11 : //Точка   слева  (X.XXX)
                        (PTR==2)? 2'b11 : //Точка   слева  (X.XXX)
                                  2'b10 ; //Точка в центре (XX.XX)
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 ;

//--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 ;
ce1ms <= ce ;
ce100ms <= (cb_100ms==10) & ce10ms ;
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
```

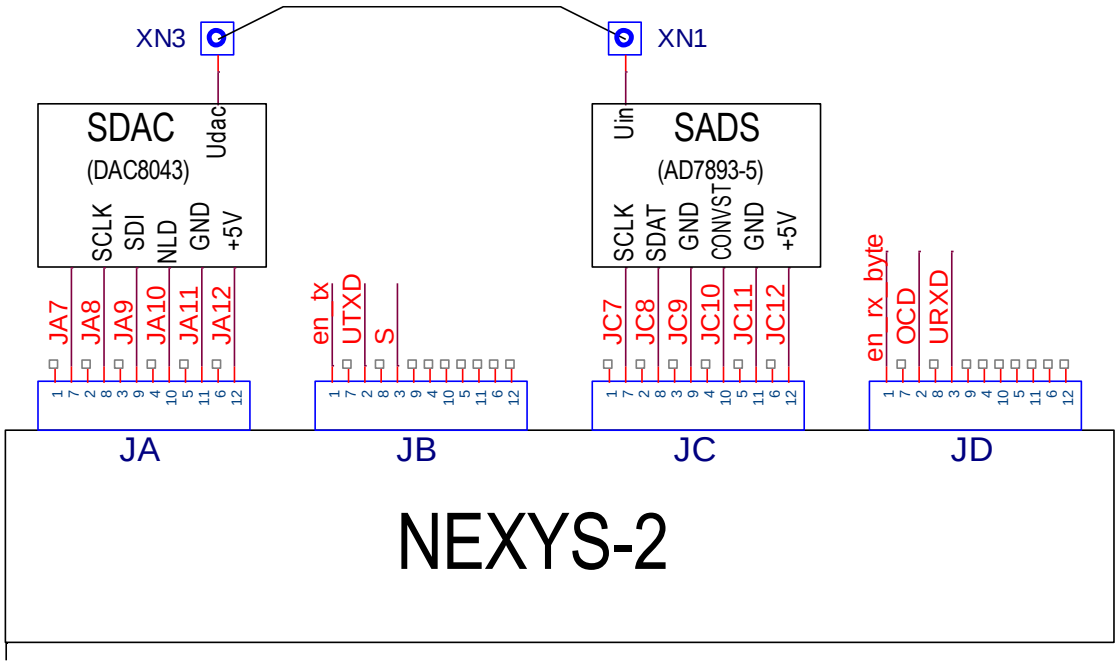


Рис.15 Схема соединения макетов MDAC и SADC с макетом NEXYS-2

6.8 Распределение сигналов по выводам ПЛИС (S_Sch_Lab414_NS93.ucf файл)

```
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" ; # DOWN(/2)
NET "BTN1" LOC = "D18" ; # bf_AMP
NET "BTN2" LOC = "E18" ; # bf_SH
NET "BTN3" LOC = "H13" ; # UP (*2)
```

```
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" ; #dat[0]
NET "SW<1>" LOC = "H18" ; #dat[1]
NET "SW<2>" LOC = "K18" ; #dat[2]
NET "SW<3>" LOC = "K17" ; #dat[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" ; #
#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" ; #
#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 AD7893
NET "JC8" LOC = "F14" ; #SDAT AD7893
#NET "JC9" LOC = "G16" ; # GND
NET "JC10" LOC = "J12" ; #CONVST AD7893

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

6.9 Текстовый вариант схемы лабораторной работы

```
module V_Sch_Lab414_N93(
                input F50MHz,          output wire [3:0] AN, //Аноды
                input [7:0]SW,         output wire [6:0] seg, //Сегменты
                input BTN0,            output wire seg_P,     //Точка
                input BTN3,            output wire [7:0]LED,
                input BTN2,
                input BTN1,
                        //--Выводы к ЦАП DAC8043----
                                        output wire JA8,       //SCLK DAC8043
                                        output wire JA9,       //SDAT DAC8043
                                        output wire JA10,      //NLD DAC8043

                                        output wire JB1, //en_tx
                                        output wire JB2, //UTXD
                                        output wire JB3, //S
                                        output wire JB4, //st_ADC

                                        output wire JD1,       //en_rx_byte
                                        output wire JD2,       //URXD
                                        output wire JD3,       //OCD
                                        output wire JD4,       //ce_ADC
                        //--Выводы к АЦП AD7893----
                                        output wire JC7,       //SCLK AD7893
                input JC8,                                     //SDAT AD7893
                                        output wire JC10,//CONVST AD7893
                                        output wire JC1,//SCLK
                                        output wire JC2);//SDAT
wire clk, ce1ms, ce100ms, st_SDAC  ;
assign JB4=st_SDAC ;
assign JC1=JC7 ;//SCLK AD7893
assign JC2=JC8 ;//SDAT AD7893
//--Глобальный буфер сигнала синхронизации
BUFG DD0 (.I(F50MHz),    .O(clk));

//--Генератор FSK байта------------------------------
wire [11:0]FSK_OUT ;
```

```
wire [7:0]M ; //Множитель M от кнопочного регулятора амплитуды
Gen_FSK_UART_byte DD1 (
                .M(M),                  .en_tx(JB1),
                .TX_dat(SW),            .UTXD(JB2),
                .clk(clk),              .S(JB3),
                .st(ce100ms),           .FSK_SH(FSK_OUT),
                                        .ce_SIN(st_SDAC));

//--Цифроаналоговый преобразователь----------------------
SPI_DAC8043 DD2 (               .SDAT(JA9), //Последовательные данные
                .st(st_SDAC),           .SCLK(JA8), //Сигнал синхронизации
                .clk(clk),              .NLD(JA10), //Асинхронная загрузка
                .DI(FSK_OUT));

//---Приемник FSK байта-------------------------------
wire [11:0]FSK_RX ;
wire [11:0]MFSK_RX ;
wire st_SADC, ok_rx_byte, en_rx_byte ;
assign JD4=st_SADC ;
wire[7:0] rx_dat ;
wire [11:0]AMP ;
wire [11:0] SH ;
assign JD1 = en_rx_byte ;

//Мультиплексор сигнала для FSK пиемника байта
wire [11:0]FSK_IN = (BTN2 & BTN1)? FSK_OUT : MFSK_RX ;
RX_FSK_byte_Nocd DD3 (          .RX_dat(rx_dat),
                .clk(clk),              .en_rx_byte(en_rx_byte),
                .FSK_SH(FSK_IN),        .URXD(JD2),
                                        .OCD(JD3),//
                                        .bf_AMP(AMP),
                                        .bf_SH(SH),
                                        .ok_rx_byte(ok_rx_byte),
                                        .ce_ADC(st_SADC));

//---Аналого цифровой преобразователь-------------------
SPI_AD7893 DD4 (
                .clk(clk),              .SCLK(JC7),             //Импульсы синхронизации
                .SDAT(JC8),             .st_ADC(JC10),          //
                .st(st_SADC),           .ADC_dat(FSK_RX));

//---Умножитель на 5000/4096----------------------
MULT_5000_DIV_4096 DD5 (.A(FSK_RX), .B(MFSK_RX));
//--Кнопочный регулятор  амплитуды FSK------------
BTN_REG_AMP DD6 (
                .clk(clk),              .M(M),//Множитель
                .BTN_UP(BTN3),                  //Кнопка сдвига влево (умножение на 2)
                .BTN_DOWN(BTN0),        //Кнопка сдвига вправо (деление на 2)
                .ce(ce1ms));
//--Преобразователь двоичного числа AMPbin в двоично десятичное AMPdec
wire [15:0]AMPdec ;
BIN12_to_DEC4 DD7 (     .BIN(AMP),   .DEC(AMPdec),
```

```
                        .clk(clk),
                        .st(ok_rx_byte));
//--Преобразователь двоичного числа SHbin в двоично десятичное SHdec
wire [15:0]SHdec ;
BIN12_to_DEC4 DD8 (     .BIN(SH),     .DEC(SHdec),
                        .clk(clk),
                        .st(ok_rx_byte));
//--Мултиплексор данных для индикатора
wire [1:0]BTN = {BTN2,BTN1} ;

wire [15:0]disp_dat = (BTN==0)? {SW,rx_dat} :
                      (BTN==1)?   AMPdec :
                       (BTN==2)?  SHdec : {SW,rx_dat} ;

//--Семи сегментный светодиодный индикатор
DISPLAY DD9 (   .clk(clk),              .AN(AN),      //Аноды
                .dat(disp_dat),         .seg(seg),    //Сегменты
                .BTN2(BTN2),            .seg_P(seg_P), //Точка
                .BTN1(BTN1),            .ce1ms(ce1ms),//1 милисекунда
                                        //.ce10ms(ce10ms),    //10 милисекунд
                                        .ce100ms(ce100ms)); //0.1 секунды

//--Модуль включения светодиодов
LED_BL DD10 (   .DI(M),         .DO(LED), //M - включает только один светодиод
                .E(en_rx_byte)); //en_rx_byte - включает все светодиоды
endmodule
```