Последовательный канал информационного обмена по стандарту ARINC-429

Лабораторная работа Lab405_ADC

Стандарт ARINC-429, разработанный фирмой ARINC, предназначен для межсистемного обмена информацией в коммерческих и транспортных самолётах (в России это ГОСТ-18977-79). Скорость передачи 12.5, 50 или 100 кбит/сек. Соединительные провода — экранированные витые пары. На одной шине (витой паре) может быть только один передатчик и не более 20 приемников. Передатчик всегда активен, он либо передаёт слова данных или выдаёт "пустой" уровень (0 В). Размер “слова” составляет 32 бита. Бит 32 – контроль четности (дополнение до нечетного числа единиц). Код – биполярный самосинхронизирующийся, с возвратом к нулю (RZ). Логической единице соответствует положительный импульс, а логическому нулю – отрицательный импульс. Длительность импульса равна половине интервала следования (длительности бита). Импульсы должны иметь пологие фронты примерно 0.1-0.3 от длительности импульса. Пример фрагмента временных диаграмм сигналов шины стандарта ARINC 429 приведен на рис.1.

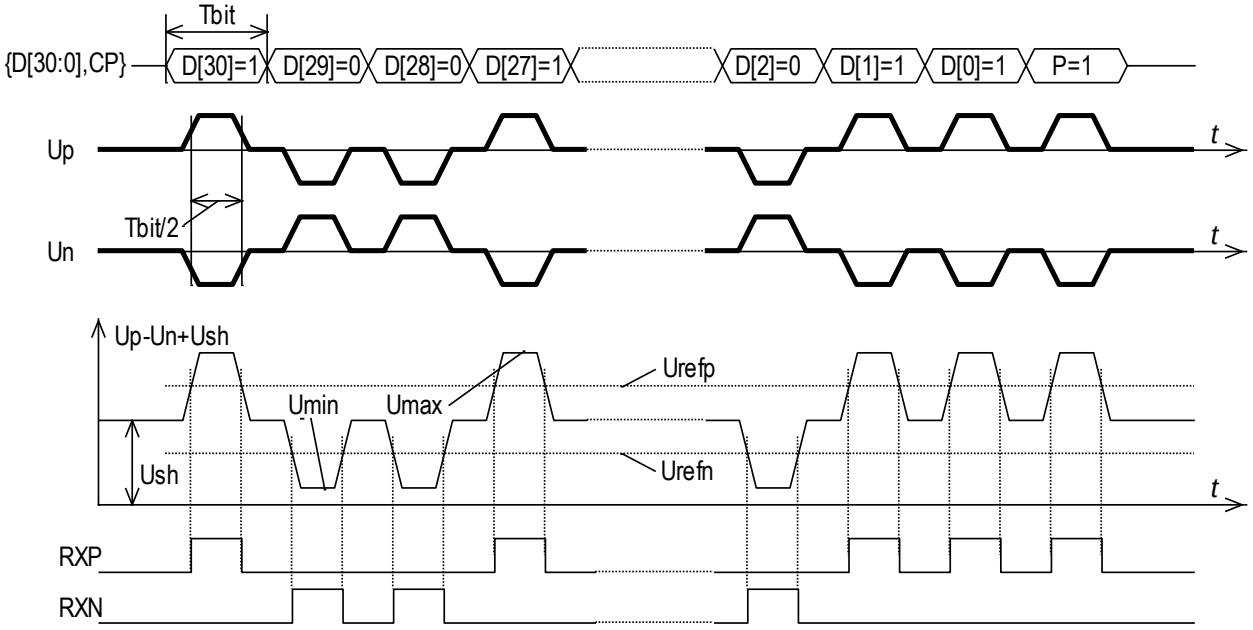

Рис.1. Пример диаграмм сигналов стандарта ARINC 429

Для преобразования аналоговых сигналов Up и Un в логические сигналы RXP и RXN рекомендуется использовать микросхемы фирмы HOLT, например, HI-8591.

В тех случаях, когда нет возможности использовать профессиональные микросхемы, то приходится применять дифференциальный усилитель и два компаратора с порогами на уровне половины амплитуды положительных и отрицательных импульсов. При однополярном напряжении питания напряжение на выходе дифференциального усилителя смещено на Ush (примерно половина напряжения питания). Для автоматической установки порогов Urefp и Urefn на необходимых уровнях: Urefp=Ush+(Umax-Ush)/2, Urefn=Ush-(Ush-Umin)/2 надо знать амплитуду каждого импульса.

Если все импульсы имеют одинаковую амплитуду, то для всех импульсов, кроме первого положительного и первого отрицательного, Umax и Umin можно определить при помощи пиковых детекторов. При этом первые импульсы будут пропущены. Если недопустим пропуск первых импульсов и, более того, если все импульсы имеют разные амплитуды, то задачу установки порога на уровне половины его амплитуды можно решить для задержанного импульса. Задержка импульса должна быть больше длительности фронта. Если после спада задержанного импульса сбрасывать напряжение пикового детекторов до исходного уровня Ush, то для каждого импульса будет устанавливаться свой порог. Напряжение смещения Ush можно измерить в паузе между передачами пакетов.

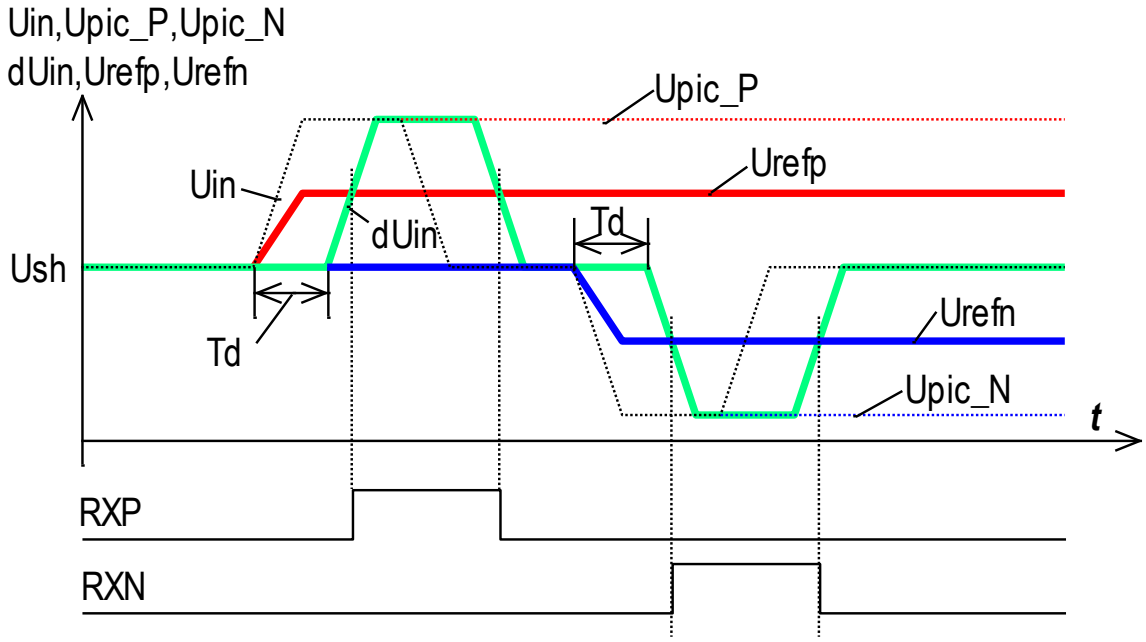

Рис.2. Пример временных диаграмм автоматической установки порогов компараторов на уровне половины амплитуды

Аналоговыми средствами реализовывать этот метод очень неудобно, прежде всего, из-за трудностей реализации неискажающей линии задержки импульсов. Неудобно также реализовывать пиковые детекторы и устройство выборки и запоминания напряжения смещения.

Если «оцифровать» сигнал с выхода дифференциального усилителя 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 Ndel, Adr_rd=Adr_wr – Ndel. Задержка выходных данных памяти относительно входных при этом равна Ndel*Tce.

Пиковый детектор максимума, например, - это регистр в который с каждым тактом записываются данные, только если они превышают текущие данные регистра. Аналогично, в регистр пикового детектора минимума записываются данные только при условии, что они меньше текущих данных регистра.

1. Цифровой имитатор сигналов ARINC-429

1.1 Модуль генератора трапецеидального импульса со смещением и регулируемой амплитудой

```
`include "CONST.v"
module Gen_MTRP(  output reg [7:0]MTRP_SH,   //Умноженная, смещенная трапеция
        input clk,        output reg [7:0]TRP=0,       //Трапеция
        input st,         output reg [6:0]cb_tact=0,   //Счетчик такта
        input [7:0]M,     output reg T_TRP=0,          //Интервал такта
        input S);  //Полярность импульса

//---Генератор ce----------
reg [5:0]cb_ce=0;
wire ce=(cb_ce==`Fclk/(`BR*`NP));//Пример,50000000/(100000*100)=5, Tce=Tclk*5=100ns
always @ (posedge clk) begin
cb_ce <= (ce | st)? 1 : cb_ce+1  ;
end

wire T_front = (cb_tact>=0) & (cb_tact<`Nfr) & T_TRP ;  // Интервал фронта
wire T_spad = (cb_tact>=`NP/2) & T_TRP ;                // Интервал спада
wire [15:0]AMTRP=TRP*M ;              //Умножение на M
wire [7:0]MTRP = AMTRP>>7 ;           //Деление на 128

always @ (posedge clk) begin
MTRP_SH <= (S & ce)? `SH0+MTRP : ce? `SH0-MTRP : MTRP_SH;
cb_tact <= st? 0 : (T_TRP & ce)? cb_tact+1 : cb_tact ;
T_TRP <= st? 1 : ((cb_tact==`NP/2+`Nfr) & ce)? 0 : T_TRP ;
TRP <= st? 0 : (T_front & ce)? TRP+7 : (T_spad & ce)? TRP-7 : T_TRP? TRP : 0 ;
end
endmodule
```

В соответствии с данными модуля параметров CONSY.v этом модуле данные регистра TRP образуют трапецеидальный импульс с длительностями фронта и спада по `Nfr=15 тактов, 15*Tce. Длительность вершины 20 тактов (`NP/2-2*`Nfr). Длительность Tbit 100 тактов (`NP). На интервалах фронта T_front и спада T_spad данные меняются с шагом 7. За 15 тактов на интервале фронта данные увеличиваются от 7 до 15*7=105, а на интервале спада уменьшаются от 105 до 7.

Для получения импульса с регулируемой амплитудой данные регистра TRP[7:0] умножаются на M[7:0] (AMTRP[15:0]=TRP*M ,0<=M<=128). Таким образом, амплитуда импульса AMTRP[15:0] равна M*105, а амплитуда импульса MTRP[7:0] равна M*105/128. Положительные импульсы сигнала MTRP_SH получаются при S=1 суммированием со смещением `NS0, а отрицательные при S=0 – вычитанием из смещения `NS0.

Для получения аналогового сигнала ARINC-429 при выполнении работы используется цифроаналоговый преобразователь (ЦАП) DAC_R2R.

Напряжение на выходе ЦАП Udac= 3.3V*MTRP_SH/256.

Максимальная амплитуда (M=128) Umax=3.3V*105 /256= 1.35V.

Минимальная амплитуда (M=1) Umin = 3.3V*1/256~=13mV.

Напряжение смещения Ush=3.3V*`NS0/256=3.3V*128/256=1.65V.

1.2 Модуль параметров **CONST**.v

```
`define  Fclk       50000000   //50 MHz – частота сигнала синхронизации NEXYS-2
`define  F10MHz 10000000   //10 MHz для PADC – частота дискретизации АЦП
```

```
`define  BR       12500       //BAUDRATE - скорость  12,5 kBOD
//`define  BR     50000       //BAUDRATE - скорость  50 kBOD
//`define  BR     100000      //BAUDRATE - скорость 100 kBOD
`define  NBR      `Fclk/`BR   //50000000/12500=4000
`define   NP      100         //Число точек на Tbit
`define   Nfr     15          //Число точек фронта/спада
`define   Nbit    32          //Число бит в слове
`define   Npauz   10          //Число бит паузы (для передатчика) 4<= Npauz<=200
`define  SH0      8'h80       //Смещение (инициализация модуля памяти 8'h80=128)
`define  Amin     16          //Минимальная амплитуда импульса A=M*105/128
`define  Mmin     22          //Минимальный множитель M
`define  my_DAT       31'h12345678        //Мои данные (см. Таблицу 1)
`define  Sim_DAT  31'h00000000    //Для проверки контроля четности при моделировании
```

1.3 Модуль генератора периодической последовательности слов ARINC-429

```
`include "CONST.v"
module AR_TXA(
        input clk,          output wire[7:0] TXA,
        input [7:0]M,       output wire TX_bit,
        input [30:0]DAT,output wire en_tx,
        input ext_st,       output wire T_cp,
        input SW,           output wire ce_bit,
                            output wire [4:0]NW
                            );//SW=0- внешний запуск, SW=1-запуск от TX_TIMER

wire my_st, ce ;
wire st = SW? my_st : ext_st ;
reg [30:0]sr_dat=0 ;
reg FT_cp=1 ;
wire T_dat = en_tx & !T_cp ;
assign TX_bit= T_dat? sr_dat[30] : T_cp? FT_cp : 0 ;
//-- Вычисление контрольного бита
always @ (posedge clk) begin
FT_cp <= st? 1 : (sr_dat[30] & ce_bit & T_dat)? !FT_cp : FT_cp ;
sr_dat <= st? DAT : (T_dat & ce_bit)? sr_dat<<1 : sr_dat ;
end
//--Формирователь периодической последовательности кадров
TX_TIMER DD1 (  .clk(clk),    .ce_bit(ce_bit),
                .ce(ce),      .en_tx(en_tx),
                              .T_cp(T_cp),
                              .st(my_st),
                              .NW(NW));
//--Генератор трапецеидальных импульсов
wire [7:0]MTRP_SH ;
assign TXA = en_tx? MTRP_SH : `SH0 ;

Gen_MTRP DD2 (  .clk(clk),        .MTRP_SH(MTRP_SH),
                .st(ce_bit),  .ce(ce),
                .S(TX_bit),
```

.M(M));
endmodule Этот модуль формирует периодическую последовательность слов (`my_DAT[31:0], см. "CONST.v" ) с паузой между словами в `Npauz бит. В состав модуля входит модуль генератора трапецеидальных импульсов Gen_MTRP и модуль таймера TX_TIMER. Таймер формирует периодическую последовательности импульсов en_tx с длительностью в 32 бита и с периодом (32+ Npauz) бит. T_cp соответствует интервалу последнего контрольного бита “слова”.

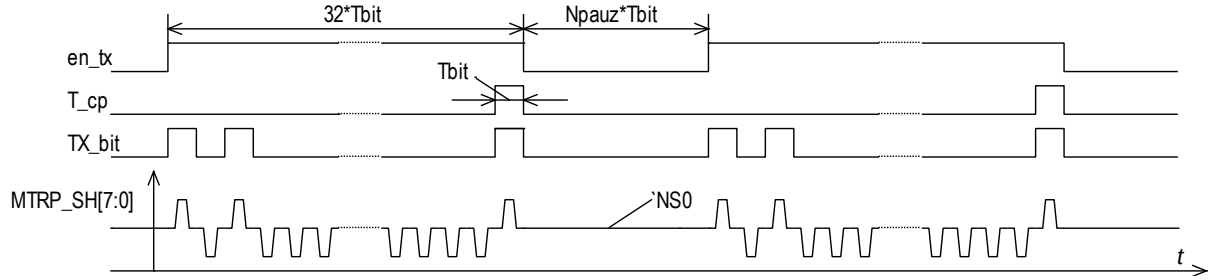

Рис.3 Пример временных диаграмм сигналов en_tx и T_cp TX_bit и MTRP_SH[7:0] модуля AR_TXA

1.4 Модуль формирователя периодической последовательности слов

```
`include "CONST.v"
module TX_TIMER(
        module TX_TIMER(
        input clk,    output wire ce_bit,//
        input ce,     output reg en_tx=0,  //Интервал передачи кадра
                      output wire st,          //Начало кадра
                      output wire T_cp);   //Интервал контрольного бита

reg [7:0]cb_start=0;              //Счетчик задержки старта
wire en_start = cb_start[7];    //Задержка на 128 Tce
reg [8:0]cb_bit=0 ;               //Счетчик бит
assign ce_bit = (cb_bit==`NP) & ce & en_start ;
reg [8:0]cb_st=`Nbit+`Npauz-5 ;    //Счетчик периода кадров
assign T_cp = (cb_st==`Nbit-1) ;    //Интервал контрольного бита
assign st = (cb_st==`Nbit+`Npauz-1) & ce_bit ;    //Начало кадра

always @ (posedge clk) begin
cb_start <= (!en_start & ce)? cb_start+1 : cb_start ;
cb_bit <= ce_bit? 1 : ce? cb_bit+1 : cb_bit ;
cb_st <= st? 0 : ce_bit? cb_st+1 : cb_st ;
en_tx <= st? 1 : (T_cp & ce_bit)? 0 : en_tx ;
end
endmodule
```

2. Приемник сигналов ARINC429

2.1 Модуль приемника сигналов ARINC-429 с автоматической установкой порога на уровне половины амплитуды

```
`include "CONST.v"
module AR_RXA(
```

```
    input [7:0]Inp,     output wire [7:0]SH,      //Смещение
    input clk,          output wire [7:0]DInp,    //Задерженные импульсы
                        output wire [7:0] AMP,    //Амплитуда импульсов
                        output wire [7:0]bf_AMP,  //Амплитуда импульсов в кадре
                        output wire RXP,  //Выход компаратора положительных импульсов
                        output wire RXN, //Выход компаратора отрицательных импульсов
                        output wire RXPN,         //RXPN=(RXP | RXN)
                        output reg en_rx=0,       //Интервал приема слова
                        output reg FT_cp=0,       //Триггер контроля четности
                        output reg RX_bit=0,      //Бит слова
                        output reg [5:0]cb_bit=0, //Счетчик бит "слова"
                        output wire ok_rx,        //Есть прием "слова"
                        output reg [30:0]bf_dat=0, //Принятые данные "слова"
                        output reg [30:0]sr_dat=0, //Регистр сдвига данных
                        output wire res,          //Сброс после приема "слова"
                        output wire M10M          //Сигнал дискретизации АЦП PADC
                        );
reg [2:0] cb_10M=0 ;//Счетчик сигнала дискретизации АЦП
wire ce10M = (cb_10M==5) ;
assign M10M = !cb_10M[1] ;//Сигнал дискретизации АЦП PADC

always @ (posedge clk) begin
cb_10M  <= ce10M? 1 : cb_10M+1 ;
end
//----------------------
assign RXPN = (RXP | RXN) ;
wire ce ;
//--Измеритель смещения и амплитуды (создать самостоятельно)
Mes_SH_AMP DD1 (.Inp(Inp),   .SH(SH),       //Смещение в паузе
                .clk(clk),   .DInp(DInp),   //Задержанные импульсы
                .res(res),   .AMP(AMP),     //Амплитуда импульсов в кадре
                             .bf_AMP(bf_AMP),//Буфер амплитуды
                             .RXP(RXP),     //Выход компаратора положительных импульсов
                             .RXN(RXN),     //Выход компаратора отрицательных импульсов
                             .ce(ce));      //Сигнал дискретизации приемника
reg tRXPN=0 ;
wire front_RXPN = RXPN & !tRXPN; //Фронт RXPN
wire spad_RXPN  = !RXPN & tRXPN; //Спад RXPN
//---Генератор сброса
reg [19:0] cb_res=0  ;    //Счетчик сброса в паузе между "словами"
assign res = (cb_res==4*`NP-1);
wire T_cp = (cb_bit==31) ;
wire ce_end = T_cp & spad_RXPN ;
assign ok_rx = (FT_cp & ce_end) ;

always @ (posedge clk) if (ce) begin
tRXPN <= RXPN;    //
en_rx <= RXPN? 1 : res? 0 : en_rx ;
cb_res <= RXPN? 0 : cb_res+1 ;
cb_bit <= res? 0 : spad_RXPN? cb_bit+1 : cb_bit ;
FT_cp  <= res? 0 : (front_RXPN & RXP)? !FT_cp : FT_cp ;
RX_bit <= res? 0 : front_RXPN? RXP : RX_bit;
```

```
sr_dat <= (front_RXPN & !T_cp)? sr_dat<<1 | RXP : sr_dat ;//
bf_dat <= ok_rx? sr_dat : bf_dat ;//
end
endmodule
```

Схему модуля Mes_SH_AMP предстоит составить самостоятельно (см. временные диаграммы рис.2).

Приемник AR_RXA должен работать со своим сигналом синхронизации поэтому в модуле Mes_SH_AMP должен быть свой генератор сигнала дискретизации се точно такой же как и в модуле генератора трапецеидального импульса Gen_MTRP.

В модуле **AR_RXA** импульс res вырабатывается после последнего импульса RXPN с задержкой 4*Tbit=4*`NT*Tce=400*Tce.

Для контроля четности используется FT триггер FT_cp (однобитный счетчик), который по сигналу res сбрасывается в 0, а импульсами “единиц” слова переключается в противоположное состояние. При нечетном числе “единиц” в конце слова FT_cp=1, что является признаком нечетного числа единиц в слове.

Полное число импульсов в “слове” подсчитывается счетчиком cb_bit, и если после окончания “слова” cb_bit не равно 32 или FT_cp не равно 1, то сигнал ok_rx не вырабатывается и данные из регистра сдвига sr_dat[30:0] не переписываются в выходной регистр bf_dat[30:0].

В состав модуля **AR_RXA** входит модуль Mes_SH_AMP компараторов с автоматическим измерением смещения и амплитуды. В состав, которого должен входить модуль памяти MEM8x32 для задержки входного сигнала Inp[7:0].

Выходные данные DO[7:0] = DInp[7:0] памяти должны быть задержаны относительно входных DI[7:0] = Inp[7:0] не менее, чем на длительность фронта сигнала (Tfr=Tce*Nfr=Tce*15). Например, адрес чтения данных памяти:
wire [4:0]Adr_rd = Adr_wr-`NP/4; Задержка на 25 тактов (четверть Tbit).

Все регистры и модуль памяти должны работать по условию if(ce).

2.1.1 Модуль памяти

```
module MEM8x32(
   input clk,      output wire [7:0] DO,
   input we,
   input [7:0] DI,
   input [4:0] Adr_wr,
   input [4:0] Adr_rd);

reg [7:0]MEM[31:0] ;
assign DO = MEM[Adr_rd] ;
initial//Инициализация модуля памяти из файла init_MEM8x32.txt
$readmemh ("init_MEM8x32.txt", MEM, 0, 31);
always @ (posedge clk) begin
MEM[Adr_wr] <= we? DI : MEM[Adr_wr] ;
end
endmodule
```

Текстовый файл инициализации модуля памяти init_MEM8x32.txt должен иметь 32 строки числа 80 – номинального смещения SH0[7:0] в HEX формате: `SH0=8’h80 = 128.

Регистры пиковых детекторов PICmax и PICmin должны сбрасываться в исходные состояния `SH сигналом res.

```
PICmax <= res? `SH0 : (Inp>PICmax)? Inp : PICmax ;
PICmin <= res? `SH0 : (Inp<PICmin)? Inp : PICmin ;
```

Амплитуды положительных и отрицательных импульсов:

```
wire [7:0]AMP_P = PICmax-SH ;
wire [7:0]AMP_N = SH-PICmin ;
```

Пороги компараторов REF_P и REF_N:

```
wire [7:0]REF_P = SH+(AMP_P>>1) ; // Деление амплитуды на 2
wire [7:0]REF_N = SH-(AMP_N>>1) ; // Деление амплитуды на 2
```

При формировании выходных импульсов компараторов RXP и RXN надо учитывать минимальную амплитуду `Amin:

```
assign RXP =(AMP_P>`Amin) & (DInp>REF_P) ;
assign RXN =(AMP_N>`Amin) & (DInp
```

2.2 Схема моделирования модуля измерения амплитуды и смещения

```
module Test_Mes_SH_AMP (
   input clk,      output wire [7:0]MTRP_SH, //Умноженная, смещенная трапеция
   input st,       output wire [6:0]cb_tact,     //Счетчик такта
   input [7:0]M,   output wire T_TRP,            //Интервал такта
   input S,        output wire ce,               //Сигнал дискретизации
   input res,
            //Сигналы модуля измерения смещения и амплитуды
                  output wire [7:0]SH,    //Смещение
                  output wire [7:0]DInp,  //Задержанные импульсы
                  output wire [7:0]AMP, //Текущая амплитуда
                  output wire [7:0]bf_AMP,//Конечная  амплитуда
                  output wire RXP,//Выход компаратора положительных импульсов
                  output wire RXN);//Выход компаратора отрицательных импульсов
//Генератор трапецеидальных импульсов
Gen_MTRP DD1 (
      .clk(clk),       .MTRP_SH(MTRP_SH),//Умноженная, смещенная трапеция
      .st(st),         .cb_tact(cb_tact), //Счетчик тактов
      .S(S),           .T_TRP(T_TRP),//Интервалы импульсов
      .M(M),           .ce(ce));
   //Измеритель смещения и амплитуды
Mes_SH_AMP DD2 (
      .Inp(MTRP_SH), .DInp(DInp),
      .clk(clk),          .SH(SH),
      .ce(ce),            .AMP(AMP),
      .res(res),          .bf_AMP(bf_AMP),
                          .RXP(RXP),
                          .RXN(RXN) );
endmodule
```

2.2.1 Пример содержательной части задания на моделирование модуля **Mes_SH_AMP**
(`Amin=19, скорость 12,5 kBOD, Simulation Run Time 1500 us)

```
always begin clk=0; #10; clk=1; #10; end
   initial begin
```

```
        st = 0;    S = 0;    M = 0; res = 0;
// Первый запуск. Положительный импульс
#10000; st = 1;    S = 1;    M = 27; res = 0;
#20;       st = 0;
//-- Отрицательный импульс
#80000; st=1; S = 0 ;
#20;       st=0;
//--Чтение амплитуды
#100000;   res=1;
#800;      res=0;
// Второй запуск. Положительный импульс
#80000; st = 1;    S = 1;    M = 26; res = 0;
#20;       st = 0;
//-- Отрицательный импульс
#80000; st=1; S = 0 ;
#20;       st=0;
//--Чтение амплитуды
#100000;   res=1;
#800;      res=0;
// Третий запуск. Положительный импульс
#80000; st = 1;    S = 1;    M = 25; res = 0;
#20;       st = 0;
//-- Отрицательный импульс
#80000; st=1; S = 0 ;
#20;       st=0;
//--Чтение амплитуды
#100000;   res=1;
#800;      res=0;
// Четвертый запуск. Положительный импульс
#80000; st = 1;    S = 1;    M = 24; res = 0;
#20;       st = 0;
//-- Отрицательный импульс
#80000; st=1; S = 0 ;
#20;       st=0;
//--Чтение амплитуды
#100000;   res=1;
#800;      res=0;
// Пятый запуск. Положительный импульс
#80000; st = 1;    S = 1;    M = 26; res = 0;
#20;       st = 0;
//-- Отрицательный импульс
#80000; st=1; S = 0 ;
#20;       st=0;
//--Чтение амплитуды
#100000;   res=1;
#800;      res=0;
end
endmodule
```

2.3 Схема моделирования приемника AR_RXA

```
`include "CONST.v"
module Test_AR_RXA(
    input clk_tx,       output wire[7:0] TXA,
    input SW,           output wire TX_bit,
    input ext_st,       output wire en_tx,
                        output wire T_cp,
                        output wire [4:0]NW,
                        output wire [7:0]M,

    input clk_rx,       output wire M10M,      //Сигнал дискретизации АЦП
                        output wire [7:0]DInp, //Задерженные импульсы
                        output wire [7:0]SH,   //Смещение
                        output wire [7:0]AMP, //Амплитуда импульсов в кадре
                        output wire [7:0]bf_AMP,//Амплитуда импульсов в конце кадра
                        output wire RXP,       //Выход компаратора положительных импульсов
                        output wire RXN,       //Выход компаратора отрицательных импульсов
                        output wire RXPN,      //RXPN=RXP | RXN
                        output wire [30:0]RX_dat,//Принятое слово
                        output wire [30:0]sr_dat,//Регистр сдвига
                        output wire RX_bit,    //Принятый бит
                        output wire [5:0]cb_bit,//Число импульсов в кадре
                        output wire FT_cp,     //Триггер контроля четности
                        output wire en_rx,     //Интервал приема слова
                        output wire ok_rx,     //Успешный прием слова
                        output wire res,       //Авто сброс в паузе
                        output wire [30:0]TX_dat
                        );

assign TX_dat=  (NW==0)? `Sim_DAT+0 ://  0 единиц
                (NW==1)? ` Sim _DAT+1 ://1 единица
                (NW==2)? ` Sim _DAT+3 ://     2 единицы
                (NW==3)? ` Sim _DAT+7 ://3 единицы
                (NW==4)? ` Sim _DAT+15 :// 4 единицы
                         ` Sim _DAT+31 ;// 5 единиц

assign M = (NW==0)? `Mmin+3:
           (NW==1)? `Mmin+2 :
           (NW==2)? `Mmin+1 :
           (NW==3)? `Mmin+0 :
           (NW==4)? `Mmin-1 : `Mmin-2 ;

AR_TXA DD1 (.clk(clk_tx),     .TXA(TXA),
           .M(M),             .TX_bit(TX_bit),
           .DAT(TX_dat),      .en_tx(en_tx),
           .SW(SW),           .T_cp(T_cp),
           .ext_st(ext_st),   .NW(NW));

AR_RXA DD2 (.Inp(TXA),      .DInp(DInp),
           .clk(clk_rx),    .SH(SH),
                            .AMP(AMP),
```

```
.bf_AMP(bf_AMP),
.RXP(RXP),
.RXN(RXN),
.RXPN(RXPN),
.bf_dat(RX_dat),
.sr_dat(sr_dat),
.RX_bit(RX_bit),
.cb_bit(cb_bit),
.FT_cp(FT_cp),
.en_rx(en_rx),
.ok_rx(ok_rx),
.res(res),
.M10M(M10M));
endmodule
```

В созданный модуль tf_Test_AR_RXA задания на моделирование модуля Test_AR_RXA надо включить модуль параметров CONST.v.

2.3.1 Пример содержательной части задания на моделирование для схемы **Test_AR_RXA** (`Amin=19, `BR=12500, Simulation Run Time=20 ms)

```
//Сигналы синхронизации clk_tx и clk_rx модулей передатчика и приемника должны быть разные.
// Сигнал синхронизации передатчика
parameter PTX = 20.0 ;          //Период tx_clk
always begin clk_tx = 1'b0; #(PTX/2); clk_tx = 1'b1; #(PTX/2);  end
// Варианты  сигнала синхронизации приемника
//parameter PRX = 20.0 ;  // Период clk_rx = 20.0
parameter PRX = 10.0 ;     // Период clk_rx = 10.0
//parameter PRX = 40.0 ;  // Период clk_rx = 40.0
always begin clk_rx = 1'b0; #(PRX/2); clk_rx = 1'b1; #(PRX/2);  end
    initial begin
            SW=0; ext_st=0;
#10000; SW=1; ext_st=0; //SW=1 - периодический запуск от таймера
    end
endmodule
```

Таблица 1

№	Скорость `BR[БОД]	Данные `my_DAT	Минимальная амплитуда `Amin
1	50000	31’h40800038	12
2	12500	31’h41000030	13
3	100000	31’h41800028	14
4	12500	31’h42000020	15
5	50000	31’h42800018	16
6	100000	31’h43000010	17
7	12500	31’h43800008	18
8	50000	31’h44000038	17
9	100000	31’h44800030	16
10	12500	31’h45000028	15

11	50000	31'h45800020	14
12	100000	31'h46000018	13
13	12500	31'h46800010	12
14	50000	31'h47000008	13
15	100000	31'h47800038	14
16	12500	31'h48000030	15
17	50000	31'h48800028	16
18	100000	31'h49000020	17
19	12500	31'h49800018	18
20	50000	31'h4A000010	19
21	100000	31'h4A800008	20

3. Задание к допуску (стоимость 2)

3.1 Начертить в тетради временные диаграммы рис.2.

3.2 Для своего варианта данных my_DAT[31:0] (см. таблицу 1) определить значение контрольного бита и начертить в тетради эскиз временных диаграмм рис.3.

3.3 Начертить в тетради схему лабораторной работы рис.4.

4. Задание к выполнению работы (стоимость 4)

4.1 В папке E:\FRTK\my_NAME\ для ПЛИС XC3S500E-5fg320 макета NEXYS-2 создать проект с именем Lab405ADC.

4.2 Для заданного варианта параметров создать Verilog module **CONST**.

4.3 Составить схему модуля **Mes_SH_AMP** измерения амплитуды и смещения сигнала генератора Gen_MTRP трапецеидальных импульсов. Выполнить синтез созданного модуля **Mes_SH_AMP**. Исправить ошибки (Errors), проверить важность предупреждений (Warnings).

4.4 Создать Verilog модуль **Test_Mes_SH_AMP** схемы моделирования работы модуля **Mes_ SH_AMP** измерения смещения и амплитуды. Провести моделирование. Сравнить полученные временные диаграммы с рис.2. Определить минимальное значение AMP.

4.5 Создать Verilog модуль **Test_AR_RXA** схемы моделирования работы приема "слов" ARINC-429 модулем **AR_RXA**. Привести моделирование. Сравнить полученные временные диаграммы с рис.3.

Для сдачи лабораторной работы кроме NEXYS-2, необходимы: макет цифроаналогового преобразователя (ЦАП) DAC_R2R и макет аналого-цифрового преобразователя (АЦП) PADC, которые соединяются с NEXYS2 в соответствии с рис.5.

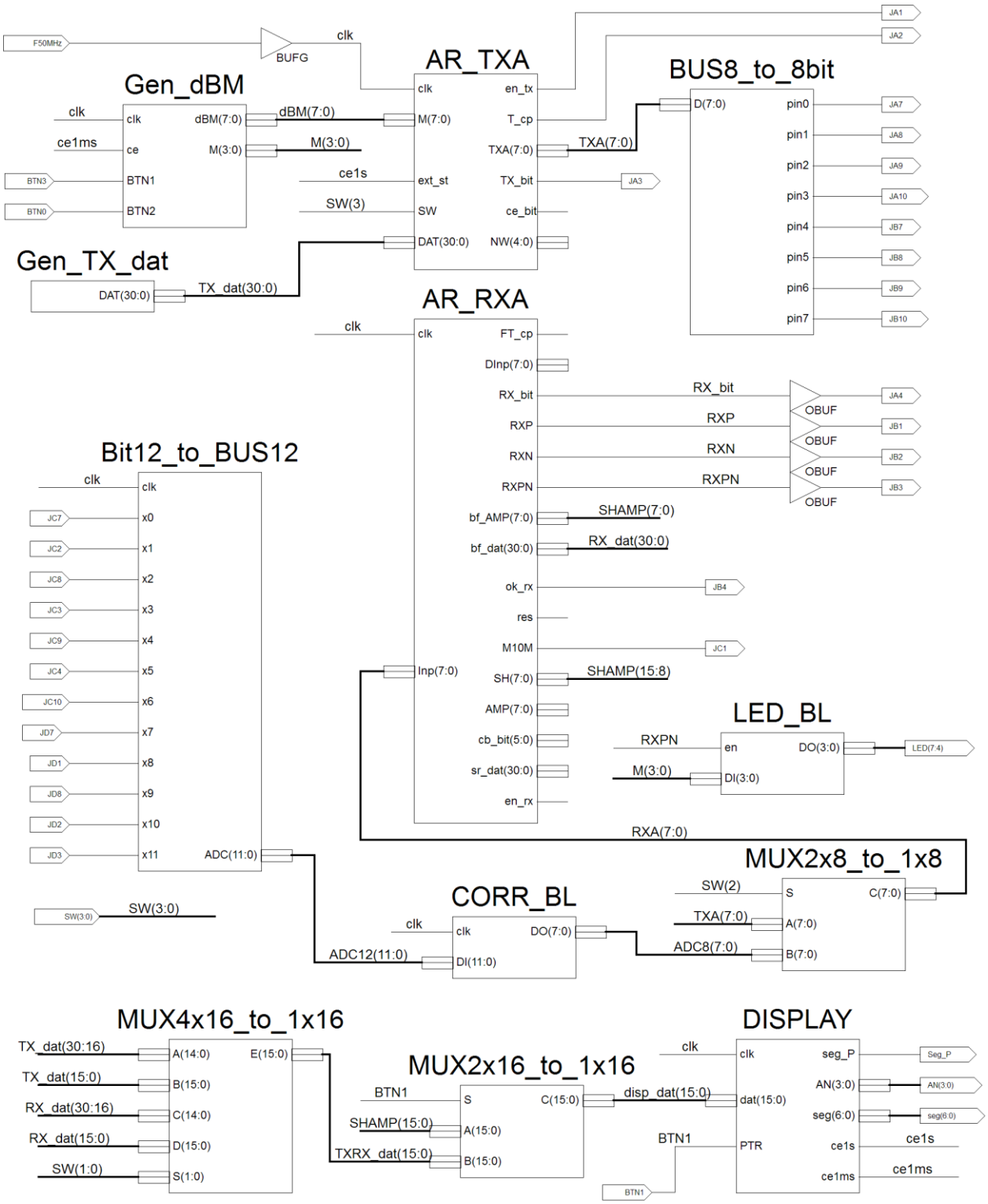

Рис.4 Схема лабораторной работы S_Sch_Lab405ADC

В этой схеме модуль **Gen_TX_dat** должен выдавать на выходе DAT[30:0] слово своего варианта (assign DAT =`my_DAT).

Модуль **Gen_dBM** (приложение 6.1) предназначен для выдачи dBM[7:0] - множителя амплитуды, который может принимать 10 значений с шагом 3dB: (dBM=128, 91, 64, 45, 32, 23, 16, 11, 8, 6). Управление множителем осуществляется кнопками BTN3 и BTN0 макета NEXYS2 (BTN3 увеличивает, а BTN0 уменьшает dBM).

Номинально (BTN1=0) на индикаторе отображаются передаваемые или принимаемые слова, а при нажатии кнопки BTN1 отображаются измеренные смещение и амплитуда цифрового сигнала Inp[7:0] в HEX формате.

Модуль **LED_BL** (приложение 6.2) предназначен для индикации сигнала en_rx приема слов и M номера множителя dBM. Этот модуль вынуждено использует только 4 светодиода LED[7:4] потому, что в макете NEXYS2 выводы ПЛИС K14, K15, J15 и J14 используются как для светодиодов LED[3:0], так и для выводов разъема JD: JD7, JD8, JD9 и JD10, которые в этой работе “заняты” макетом PADC.

Сигнал синхронизации CLK_adc с частотой 10MHz для макета PADC параллельного АЦП вырабатывается модулем приемника AR_RXA. Максимально допустимая частота преобразования (синхронизации) используемого в макете PADC АЦП AD9220 равна 10MHz.

Модуль **MUX2x8_to_1x8** (приложение 6.3) при SW[2]=0 подает на вход приемника данные MIL_dac передатчика, а при SW[2]=1 корректированные данные MIL_adc приемника.

Модуль **CORR_BL** (приложение 6.4) согласует 12-и битную шину DAC[11:0] данных параллельного АЦП с 8-и битной шиной Inp[7:0] приемника AR_RXA.

Модуль **MUX4x16_to_1x16** (приложение 6.5) подает на вход модуля семи сегментного индикатора **DISPLAY** (приложение 6.6) при:

- SW[1:0]=2’b00 – DW_TX[15:0] (переданное слово данных),
- SW[1:0]=2’b01 – CW_TX[15:0] (переданное контрольное слово),
- SW[1:0]=2’b10 – CW_RX[15:0] (принятое контрольное слово),
- SW[1:0]=2’b00 – DW_RX[15:0] (принятое слово данных).

Выходной сигнал ce1ms модуля **DISPLAY** используется для подавления “дребезга” кнопок.

Сигнал ce1s с периодом 1 сек используется при SW[3]=1 для периодического запуска передатчика MIL_TX_DAC. Такой редкий запуск вызван тем, при связи передатчика с приемником через канал ЦАП АЦП флюктуации при большей частоте затрудняют считывание данных амплитуды и смещения с семи сегментного индикатора.

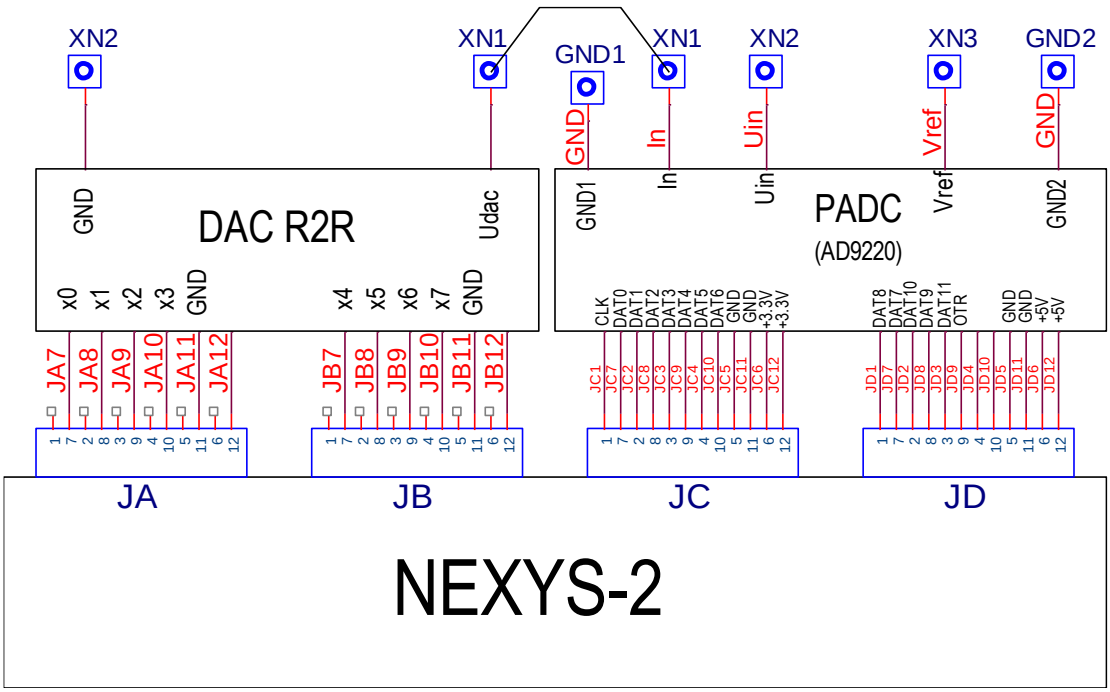

Рис.5 Схема соединения макетов ЦАП DAC_R2R и АЦП PADC с макетом NEXYS-2

Схемы и описание макетов ЦАП DAC_R2R и АЦП PADC и модули **BUS8_to_8bit Bit12_to_BUS12** приведены в приложениях 6.7 и 6.8.

5. Задание к сдаче работы (стоимость 4)

5.1 Создать Schematic S_Sch_Lab405ADC или Verilog модуль V_Sch_Lab405ADC (см. приложение 6.9).

5.2 При работе в Schematic на листе схемы S_Sch_Lab405ADC разместить все необходимые символы и соединить их в соответствии с рис.4.

5.3 Провести Synthesize-XST созданной схемы. Исправить ошибки (**Errors**), проверить существенность замечаний (**Warnings**).

5.4 Создать Implementation Constraints File (*.ucf). Выполнить **Generate Programming File** или **Configure Target Device (**приложение 6.10**).**

5.5 Вставить в разъемы JA, JB ЦАП DAC_R2R, а в разъемы JC, JD параллельный АЦП PADC. Соединить выход ЦАП с входом АЦП.

5.6 Загрузить, созданную конфигурацию схемы в ПЛИС (*.bit) или в ПЗУ (*.mcs) макета NEXYS-2. Проверить при помощи осциллографа и индикатора работу макета.

5.7 Определить минимальную амплитуду при которой устойчиво формируются все 32 импульса RXP и RXN в “слове” и правильно принимаются данные. Сохранить осциллограммы сигналов с выхода ЦАП, JA1(en_tx), JB1(RXP), JB2(RXN), JB3(RXPN).

6. Приложения

6.1 Модуль регулятора амплитуды импульсов

```
module Gen_dBM(
   input clk,        output wire[7:0] dBM,
   input ce,     output reg [3:0] M=9,
   input BTN1,
   input BTN2 );
reg [1:0]Q1 ;
reg [1:0]Q2 ;
wire st1= Q1[0] & !Q1[1] ;//Фронт BTN1
wire st2= Q2[0] & !Q2[1] ;//Фронт BTN2
wire Mmax=(M==4'h9);//Максимальное значение M=9
wire Mmin=(M==4'h0);//Минимальное значение M=0

always @ (posedge clk) if (ce) begin
Q1 <= Q1<<1 | BTN1 ;//
Q2 <= Q2<<1 | BTN2 ;//
M <= (!Mmax & st1)? M+1 : (!Mmin & st2)? M-1 : M ;//
end
assign dBM=  (M==9)?128 ://128              = 0dB
             (M==8)? 91 ://128/SQRT(2)  =-3dB
             (M==7)? 64 ://128/2        =-6dB
             (M==6)? 45 ://64/SQRT(2)   =-9dB
```

```
            (M==5)? 32 ://64/2          =-12dB
            (M==4)? 23 ://32/SQRT(2)    =-15dB
            (M==3)? 16 ://32/2          =-18dB
            (M==2)? 11 ://16/SQRT(2)    =-21dB
            (M==1)?  8 ://16/2          =-24dB
            (M==0)?  6 :// 8/SQRT(2) =-27dB
                     0 ;
endmodule
```

6.2 Модуль светодиодов

```
module LED_BL(
   input en,         output [3:0] DO,
   input [3:0] DI);
assign DO=en? 4'hF : DI ;
endmodule
```

6.3 Модуль мультиплексора **MUX2x8_to_1x8**

```
module MUX2x8_to_1x8(
   input [7:0] A,  output wire [7:0] C,
   input [7:0] B,
   input S);
assign C=S? A : B ;
endmodule
```

6.4 Модуль **CORR_BL** согласования данных АЦП ЦАП

```
module CORR_BL(
   input [11:0] DI,    output reg[7:0] DO,
   input clk);
//SHadc=4096*1.65V/5.00V=1352; SH=128; k=SH/Shadc=128/1352=0.0947
parameter Mk=6206 ;   //(Mk=k*2^16=6206)
wire [24:0]MDI=DI*Mk ;
always @ (posedge clk) begin
DO <= MDI>>16 ;
end
endmodule
```

В модулях PADC (АЦП) и DAC_R2R (ЦАП) используются разные опорные напряжения REF_{ADC} PADC это 5V, а REF_{DAC} DAC_R2R это 3.3V. Поэтому простым отбрасыванием 4-х младших разрядов ADC, т.е. делением на 16 не обойтись. Надо учитывать и неодинаковость опорных напряжений.

Напряжение U_{out} на выходе DAC_R2R $U_{out} = \frac{DAC}{256} REF_{DAC} = \frac{DAC}{256} 3.3V$ при *DAC*=SH=128 равно 1.65V.

При таком напряжении на входе АЦП число $ADC = \frac{U_{in}}{REF_{ADC}} 4096 = \frac{1.65V}{5V} 4096 = 1352$ а для приемника такому напряжению смещения должно соответствовать число 128.

Поэтому число ADC надо умножить не на 1/16, а на k=128/1352=0.0947. Эта операция и выполняется в модуле **CORR_BL**

6.5 Модуль **MUX4x16_to_1x16**

```
module MUX4x16_to_1x16(
        input [14:0] A,     output wire [15:0] E,
        input [15:0] B,
        input [14:0] C,
        input [15:0] D,
        input [1:0] S );
assign E=   (S==2'b00)? {1'b0,A} :
            (S==2'b01)? B :
            (S==2'b10)? {1'b0,C} : D ;
endmodule
```

6.6 Модуль семи сегментного индикатора

```
module DISPLAY( input clk,          output wire[3:0] AN, //Аноды
                input [15:0]dat,    output wire [6:0] seg, //Сегменты
                input PTR,          output wire seg_P,   //Точка
                                    output wire ce1s);   //1 секунда
parameter Fclk=50000 ;    //50000 kHz
parameter F1kHz=1 ;       //1 kHz
wire [1:0]ptr_P = !PTR? 2'b10 : //Точка в центре
                          2'b11 ; //Точка слева
reg [15:0] cb_1ms = 0 ;
wire ce = (cb_1ms==Fclk/F1kHz) ;
reg [9:0]cb_1s=0 ;
assign ce1s = (cb_1s==1000) & ce ;

//--Генератор сигнала ce (период 1 мс, длительность Tclk=20 нс)--
always @ (posedge clk) begin
cb_1ms <= ce? 1 : cb_1ms+1 ;
cb_1s <= ce1s? 1 : ce? cb_1s+1 : cb_1s ;
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];
//-------Семисегментный 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
```

6.7 Цифроаналоговый преобразователь

6.7.1 Модуль «расщепления» шины DAC[7:0] цифроаналогового преобразователя

```
module BUS8_to_8bit(
   input [7:0]D,
                    output wire pin0,
                    output wire pin1,
                    output wire pin2,
                    output wire pin3,
                    output wire pin4,
                    output wire pin5,
                    output wire pin6,
                    output wire pin7);

assign pin0=D[0], pin1=D[1], pin2=D[2], pin3=D[3];
assign pin4=D[4], pin5=D[5], pin6=D[6], pin7=D[7];

endmodule
```

6.7.2 Макет цифроаналогового преобразователя DAC_R-2R

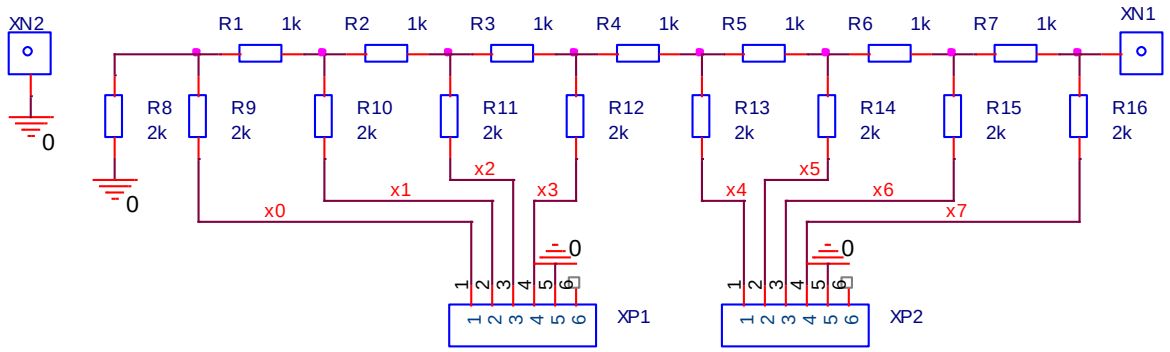

Рис. 5 Схема цифроаналогового преобразователя R-2R

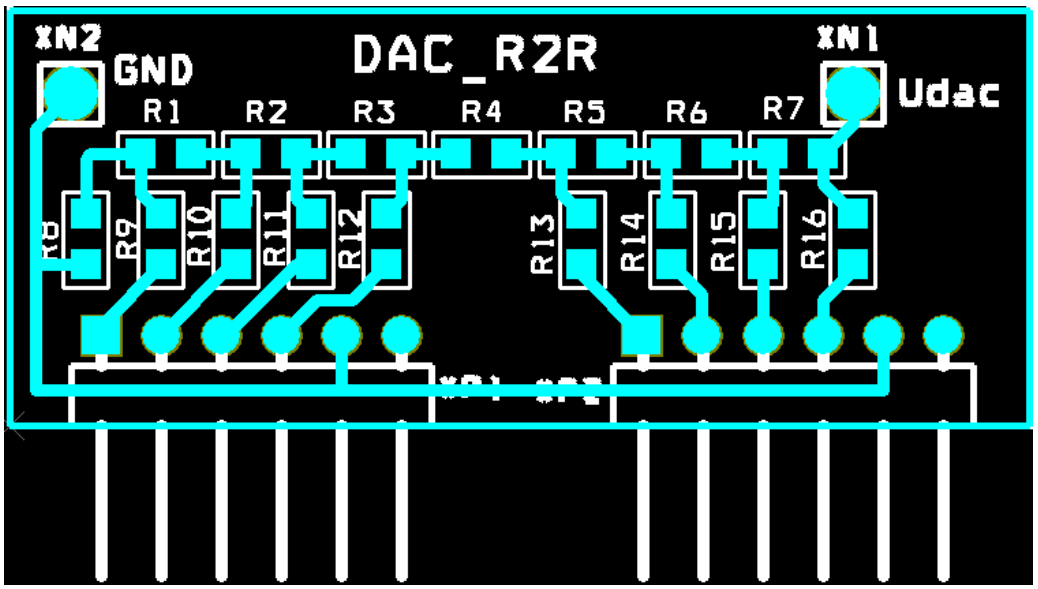

Рис. 6 Лицевая сторона платы цифроаналогового преобразователя DAC_R-2R

6.8 Параллельный аналого-цифровой преобразователь

6.8.1 Модуль «упаковки» сигналов AD0,…AD11 параллельного АЦП в шину DAT[11:0]

```
module Bit12_to_BUS12(
    input clk,    output reg [11:0] DAT=0,
    input x0,
    input x1,
    input x2,
    input x3,
    input x4,
    input x5,
    input x6,
    input x7,
    input x8,
    input x9,
    input x10,
```

```
    input x11);
always @ (posedge clk) begin
DAT= {x11,x10,x9,x8,x7,x6,x5,x4,x3,x2,x1,x0} ;
end
endmodule
```

6.8.2 Макет PADC параллельного АЦП

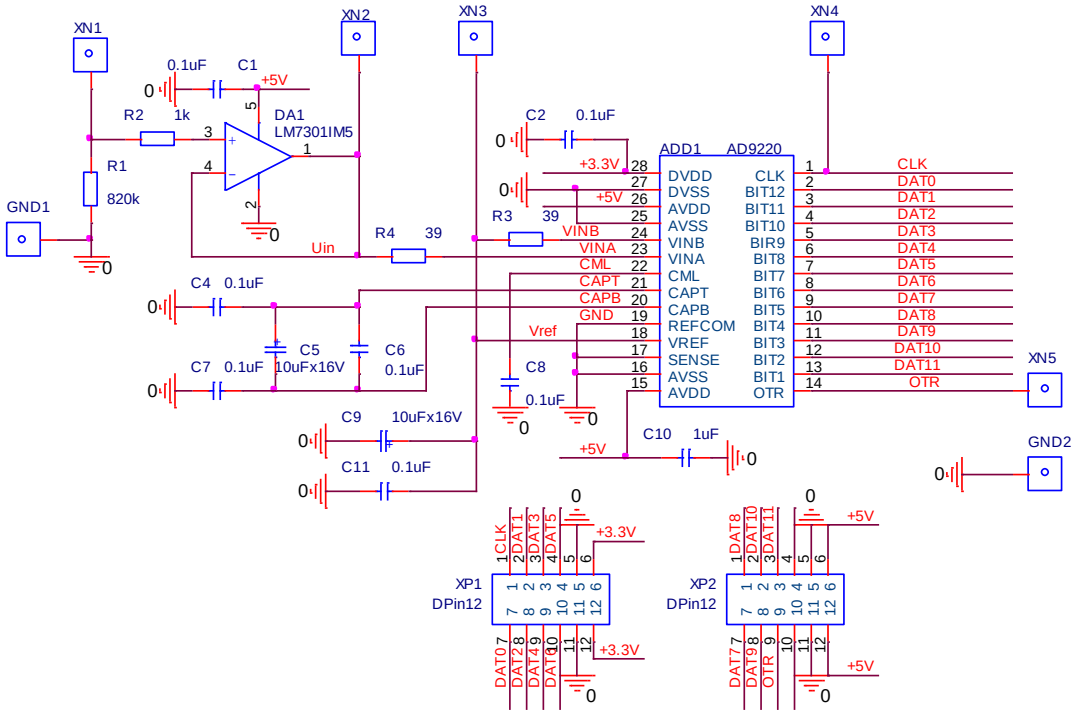

Рис. 7 Схема параллельного аналого-цифрового преобразователя

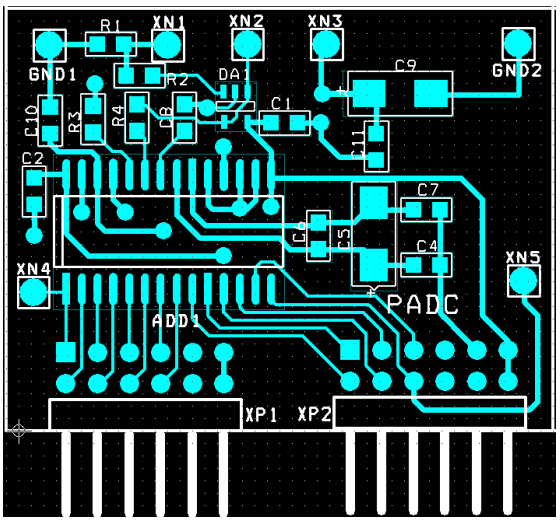

Рис. 8 Лицевая сторона печатной платы параллельного аналого цифрового преобразователя

Модуль согласования 12-и битной шины DAT[11:0] данных параллельного АЦП с 8-и битной шиной Inp[7:0] приемника AR_RXA.

6.9 Verilog модуль лабораторной работы

```
`include "CONST.v"
module V_Sch_Lab405ADC(
        input F50MHz,
        input [3:0]SW,
        input BTN3, //Увеличение M
        input BTN1, //Переключение индикации данных
        input BTN0, //Уменьшение M
//---Выводы передающего модуля для ЦАП R2R
                        output wire JA7,//DAC[0]
                        output wire JA8,//DAC[1]
                        output wire JA9,//DAC[2]
                        output wire JA10,//DAC[30]
                        output wire JB7,//DAC[4]
                        output wire JB8,//DAC[5]
                        output wire JB9,//DAC[6]
                        output wire JB10,//DAC[7]
//---Контрольные выводы передающего модуля
                        output wire JA1,//en_tx,
                        output wire JA2,//T_cp,
                        output wire JA3,//TX_bit
                        output wire JA4,//RX_bit
//---Выводы приемного модуля
                        output wire JB1,//RXP
                        output wire JB2,//RXN
                        output wire JB3,//RXPN
                        output wire JB4,//ok_rx
                        output wire [7:4]LED,//Индикатор M
//---Выводы параллельного АЦП
                        output wire JC1,//ADC_clk
        input JD3,  //ADC_dat[11]
        input JD2,  //ADC_dat[10]
        input JD8,  //ADC_dat[9]
        input JD1,  //ADC_dat[8]
        input JD7,  //ADC_dat[7]
        input JC10, //ADC_dat[6]
        input JC4,  //ADC_dat[5]
        input JC9,  //ADC_dat[4]
        input JC3,  //ADC_dat[3]
        input JC8,  //ADC_dat[2]
        input JC2,  //ADC_dat[1]
        input JC7,  //ADC_dat[0]

//-----Выводы семисегментного индикатора
                        output wire [3:0] AN,   //Аноды
                        output wire [6:0] seg,  //Сегменты
                        output wire seg_P);     //Точка

//---Данные параллельного АЦП ADC920
```

```
wire [11:0]ADC_dat = {JD3,JD2,JD8,JD1,JD7,JC10,JC4,JC9,JC3,JC8,JC2,JC7};//
//----------------------------------------------------------------------
wire clk, ce1s, ce1ms ;

//--Глобальный буфер сигнала синхронизации
BUFG DD1 (.I(F50MHz), .O(clk));

//--Регулятор амплитуды
wire [7:0]dBM ; wire [3:0]M ;
Gen_dBM DD2 (.clk(clk),        .dBM(dBM),
             .ce(ce1ms),     .M(M),
             .BTN1(BTN0),
             .BTN2(BTN3));

//--Светодиодный блок
LED_BL DD3 (.en(RXPN), .DO(LED),
            .DI(M));

//----Генератор импульсов формата ARINC-429
wire [7:0]TXA ; wire [30:0]TX_dat=`my_DAT ;
AR_TXA DD4 (.clk(clk),     .TXA(TXA),
            .M(dBM),       .TX_bit(JA3),
            .DAT(TX_dat),  .en_tx(JA1),
            .ext_st(ce1s), .T_cp(JA2),
            .SW(SW[3]));

//-- Выходы генератора AR_TXA для ЦАП R-2R
BUS8_to_8bit DD5 (.D(TXA), .pin0(JA7),
                           .pin1(JA8),
                           .pin2(JA9),
                           .pin3(JA10),
                           .pin4(JB7),
                           .pin5(JB8),
                           .pin6(JB9),
                           .pin7(JB10));

//--Модуль согласования данных АЦП ЦАП
wire [7:0]ADC8 ;
CORR_BL DD6 (   .DI(ADC_dat),   .DO(ADC8),
                .clk(clk));

//--Приеный блок ----------------------
wire [7:0]RXA = SW[2]? ADC8 : TXA ;
wire [30:0]RX_dat ; wire [15:0]SHAMP ;
AR_RXA DD7 (
        .Inp(RXA),     .SH(SHAMP[15:8]),        //Смещение
        .clk(clk),     .bf_AMP(SHAMP[7:0]),//Амплитуда импульсов в кадре
                       .RXP(RXP),   //Выход компаратора положительных импульсов
                       .RXN(RXN),  //Выход компаратора отрицательных импульсов
                       .RXPN(RXPN), //RXPN=(RXP | RXN)
```

```
                        .RX_bit(JA4),//Бит слова
                        .ok_rx(JB4),//Есть прием "слова"
                        .bf_dat(RX_dat),//Принятые данные "слова"
                        .M10M(JC1));//Сигнал дискретизации АЦП PADC);

assign JB1= RXP ;//Выход компаратора положительных импульсов
assign JB2= RXN ;//Выход компаратора отрицательных импульсов
assign JB3= RXPN ;//

//--Мультиплексор данных для индикатора
wire [15:0]TXRX_dat= (SW[1:0]==0)? {1'b0,TX_dat[30:16]} :
                     (SW[1:0]==1)? TX_dat[15:0] :
                     (SW[1:0]==3)? {1'b0,RX_dat[30:16]} :
                                   RX_dat[15:0] ;

wire [15:0]DISP_dat= BTN1? SHAMP[15:0] : TXRX_dat ;
//--Семи сегментный светодиодный индикатор
DISPLAY DD8 (.clk(clk),          .AN(AN),      //Аноды
             .dat(DISP_dat),     .seg(seg),    //Сегменты
             .PTR(BTN1),         .seg_P(seg_P),  //Точка
                                 .ce1ms(ce1ms),  //1ms
                                 .ce1s(ce1s));   //Секунда
endmodule
```

6.10 Связь портов схемы с контактными площадками ПЛИС (файл *.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" ;
NET "BTN1" LOC = "D18" ;
#NET "BTN2" LOC = "E18" ;
NET "BTN3" LOC = "H13" ;

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" ; #
NET "SW<1>" LOC = "H18" ; #
NET "SW<2>" LOC = "K18" ; #
NET "SW<3>" LOC = "K17" ; #
```

```
#NET "SW<4>" LOC = "L14" ; #
#NET "SW<5>" LOC = "L13" ; #
#NET "SW<6>" LOC = "N17" ; #
#NET "SW<7>" LOC = "R17" ; #


#NET "LED0" LOC = "J14" ; #
#NET "LED1" LOC = "J15" ; #
#NET "LED2" LOC = "K15" ; #
#NET "LED3" LOC = "K14" ; #
NET "LED4" LOC = "E17" ; #
NET "LED5" LOC = "P15" ; #
NET "LED6" LOC = "F4" ; #
NET "LED7" LOC = "R4" ; #

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

NET "JA1" LOC = "L15" ; # en_tx
NET "JA2" LOC = "K12" ; # T_cp
NET "JA3" LOC = "L17" ; #TX_bit
NET "JA4" LOC = "M15" ; #RX_bit
NET "JA7" LOC = "K13" ; # DAC[0]
NET "JA8" LOC = "L16" ;# DAC[1]
NET "JA9" LOC = "M14" ;# DAC[2]
NET "JA10" LOC = "M16" ;# DAC[3]

NET "JB1" LOC = "M13" ; # RXP
NET "JB2" LOC = "R18" ; # RXN
NET "JB3" LOC = "R15" ; # RXP | RXN
NET "JB4" LOC = "T17" ; # ok_rx
NET "JB7" LOC = "P17" ; # DAC[4]
NET "JB8" LOC = "R16" ; # DAC[5]
NET "JB9" LOC = "T18" ; # DAC[6]
NET "JB10" LOC = "U18" ; # DAC[7]

NET "JC1" LOC = "G15" ; #ADC_clk
NET "JC2" LOC = "J16" ; #ADC_dat[1]
NET "JC3" LOC = "G13" ; # ADC_dat[3]
NET "JC4" LOC = "H16" ; # ADC_dat[5]
NET "JC7" LOC = "H15" ; # ADC_dat[0]
NET "JC8" LOC = "F14" ; # ADC_dat[2]
NET "JC9" LOC = "G16" ;# ADC_dat[4]
NET "JC10" LOC = "J12" ;# ADC_dat[6]

NET "JD1" LOC = "J13" ;# ADC_dat[8]
NET "JD2" LOC = "M18" ;# ADC_dat[10]
NET "JD3" LOC = "N18" ;# ADC_dat[11]
#NET "JD4" LOC = "P18" ;#
NET "JD7" LOC = "K14" ;# ADC_dat[7]
NET "JD8" LOC = "K15" ;# ADC_dat[9]
#NET "JD9" LOC = "J15" ;#
#NET "JD10" LOC = "J14" ;#
```