Последовательный канал информационного обмена по стандарту ARINC-429

Лабораторная работа №405ADC

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.

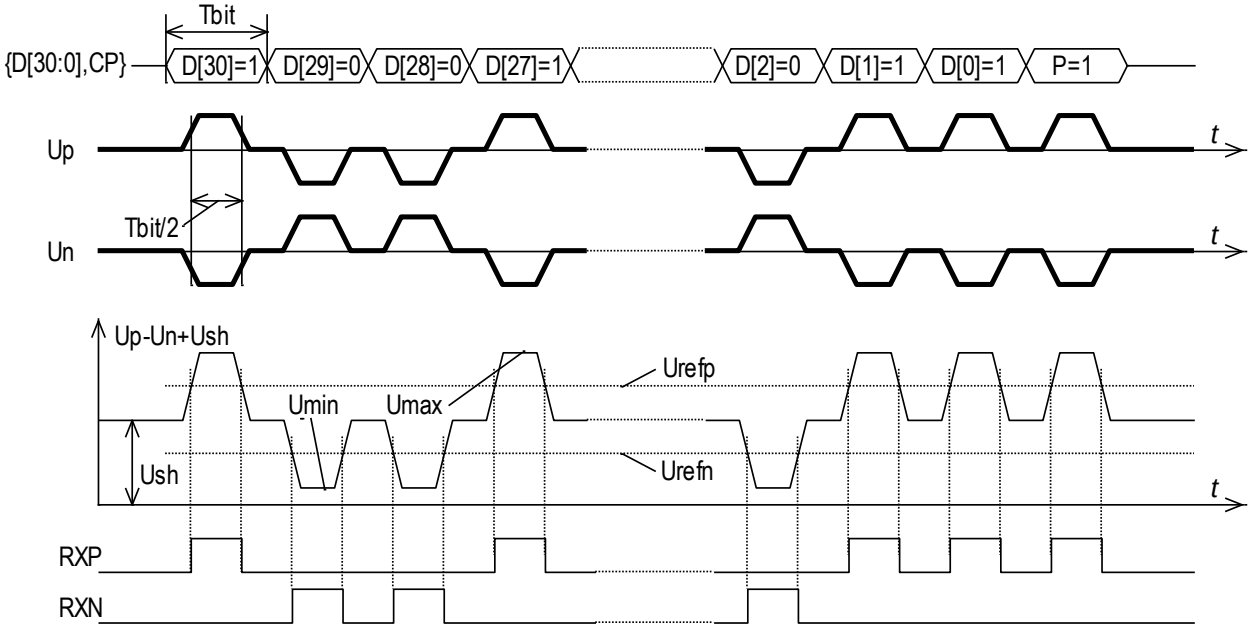

Рис.1. Пример диаграмм сигналов стандарта ARINC 429

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.

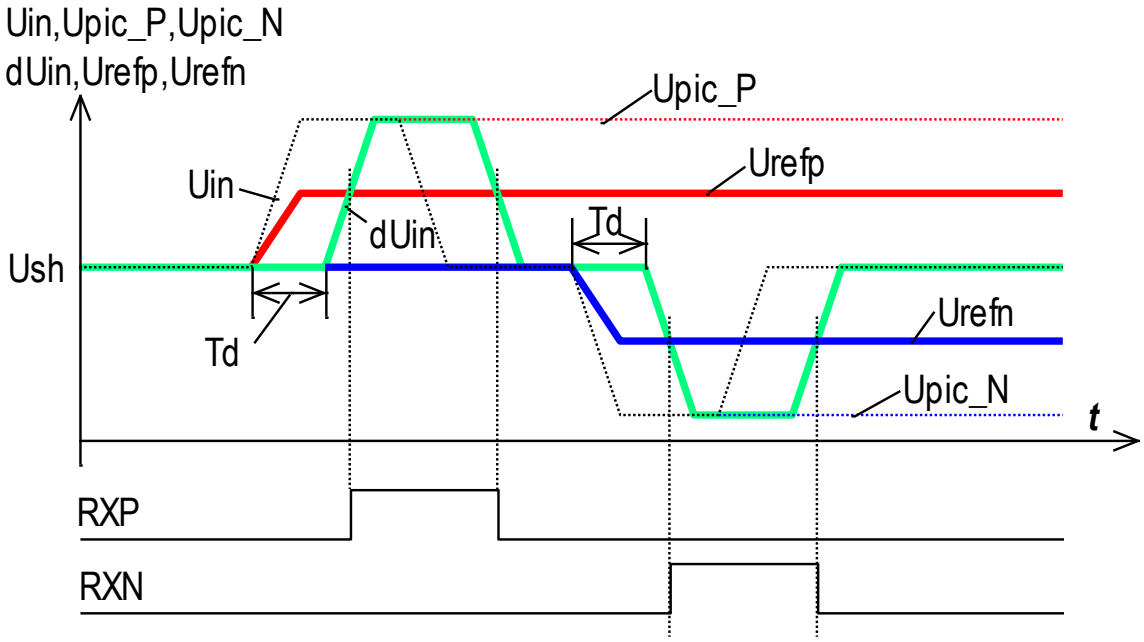

Рис.2. Пример временных диаграмм автоматической установки порогов компараторов на уровне половины амплитуды

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 Del, Adr_rd=Adr_wr – Del. Задержка выходных данных памяти относительно входных при этом равна Del*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 wire [11:0]MTRP,//Умноженная, смещенная трапеция
                input clk,      output reg [6:0]DTRP=0,//Трапеция
                input st,       output reg [8:0]cb_tact=0,
                input [5:0]M,   output reg T_TRP=0,
                input S,        output wire T_front,
                                output wire T_spad,
                                output wire ce);
        //---Генератор ce----------
reg [5:0]cb_ce=0;
assign ce=(cb_ce==`Fclk/(`BR*`NT)) ;
always @ (posedge clk) begin
cb_ce <= (ce | st)? 1 : cb_ce+1 ;
end
//--M-множитель амплитуды
//--AMP=M*DTRPmax/64=M*128*Xmax/64=M*128*15/64=M*30 (по 15 точек на фронт и спад)
assign T_front = (cb_tact>=0) & (cb_tact<15) & T_TRP ;
assign T_spad = (cb_tact>=48) & T_TRP ;
wire [11:0]MDTRP=DTRP*(M<<1) ;
assign MTRP = S? `NS0+MDTRP : `NS0-MDTRP ;

always @ (posedge clk) begin //
cb_tact <= st? 0 : (T_TRP & ce)? cb_tact+1 : cb_tact ;
T_TRP <= st? 1 : ((cb_tact==63) & ce)? 0 : T_TRP ;
DTRP <= st? 1 : (T_front & ce)? DTRP+1 : (T_spad & ce)? DTRP-1 : T_TRP? DTRP : 0 ;
        end
        endmodule
```

В этом модуле данные регистра DTRP образуют трапецеидальный импульс с длительностями фронта и спада по 15 тактов (15*Tce). Длительность вершины 20 тактов (20*Tce). Длительность Tbit 100 тактов (100*Tce). На интервалах фронта T_front и спада T_spad данные меняются с шагом 1. За 15 тактов на интервале фронта данные увеличиваются от 1 до 16, а на интервале спада уменьшаются от 16 до 1.

Для получения импульса с регулируемой амплитудой данные регистра DTRP[6:0] умножаются на M[5:0]*2 (0<=M<=63). Таким образом, амплитуда импульса MTRP[11:0] равна M*2*16=M*32. Положительные импульсы MTRP[11:0] получаются при S=1 суммированием со смещением `NS0, а отрицательные при S=0 – вычитанием из смещения `NS0 (см. модуль параметров “CONST.v”).

Для получения аналогового сигнала ARINC-419 при выполнении работы используется цифроаналоговый преобразователь (ЦАП) R-2R, который имеет 8 разрядов, поэтому из 12 разрядов MTRP[11:0] используются только 8 старших разрядов (MTRP[11:4]=MTRP[11:0]/16).

Напряжение на выходе ЦАП Udac= 3.3V/256*(MTRP[11:0]/16).

Максимальная амплитуда (M=63) Umax=3.3V/256*(63*32/16)=1.624V.

Минимальная амплитуда (M=1) Umin = 3.3V/256*(1*32/16)~=26mV.

Напряжение смещения Ush=3.3V*(`NS0/16)/256=3.3V*(2048/16)/256=3.3V*128/256=1.65V.

1.2 Модуль параметров **CONST**.v

```
`define  Fclk            50000000       //50 MHz
```

```
//`define  BR          12500          //BAUDRATE - скорость 12,5 kBOD
//`define  BR          50000          //BAUDRATE - скорость 500 kBOD
`define  BR          100000         //BAUDRATE - скорость 100 kBOD
`define  NBR         `Fclk/`BR      //50000000/100000=500
`define  NT          100            //Число точек на Tbit
`define  NS0         2048           //Смещение нуля
`define  Azer        100            //Минимальная амплитуда импульса
`define  my_DAT      31'h50000000 //Мои данные
```

1.3 Модуль генератора периодической последовательности слов ARINC-429

```
`include "CONST.v"
module AR_TXA(
                input clk,      output wire[11:0] TXA,
                input [5:0]M,   output wire TXD,
                                output wire en_tx,
                                output wire T_cp);
wire ce_bit, st,ce ;
reg [31:1]sr_dat=0 ;
reg FT_cp=1 ;
wire T_dat = en_tx & !T_cp ;
assign TXD= T_dat? sr_dat[31] : T_cp? FT_cp : 0 ;
//-- Вычисление контрольного бита
always @ (posedge clk) begin
FT_cp <= st? 1 : (sr_dat[31] & ce_bit & T_dat)? !FT_cp : FT_cp ;
sr_dat <= st? `my_DAT : (en_tx & 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(st));
//--Генератор трапецеидальных импульсов
wire [11:0]MTRP ;
assign TXA = en_tx? MTRP : `NS0 ;
Gen_MTRP DD2 (    .clk(clk),      .MTRP(MTRP),
                  .st(ce_bit),    .ce(ce),
                  .S(TXD),
                  .M(M));
endmodule
```

Этот модуль формирует периодическую последовательность слов (`my_DAT[31:0], см. "CONST.v") с паузой между словами в 4 бита. В состав модуля входит модуль генератора трапецеидальных импульсов Gen_ATRP и модуль таймера TX_TIMER. Таймер формирует периодическую последовательности импульсов en_tx с длительностью в 32 бита и с периодом 36 бит. T_cp соответствует интервалу последнего контрольного бита “слова”.

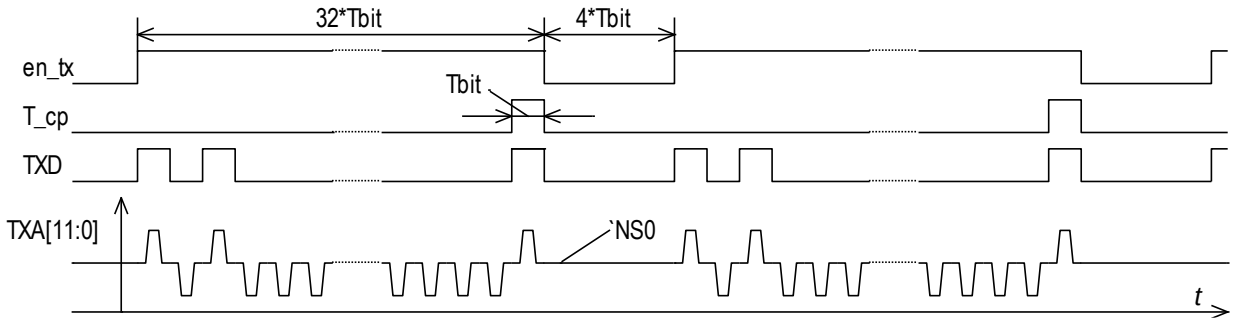

Рис.3 Пример временных диаграмм сигналов en_tx и T_cp TXD и TXA[11:0] модуля AR_TXA

1.4 Модуль формирователя периодической последовательности кадров

```
`include "CONST.v"
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==`NT) & ce & en_start ;

reg [5:0]cb_st=34 ; //Счетчик периода кадров
assign T_cp = (cb_st==31); //Интервал контрольного бита
assign st = (cb_st==35) & 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 RXM_ATRP(
   input [11:0]Inp,
   input clk,    output wire [11:0]MEM_dat,//Задержанные импульсы
   input ce,     output reg [11:0]Xmax=0,//Пиковое значение положительных импульсов
   input res,    output reg [11:0]Xns=`NS0,//Смещение
                 output reg [11:0]Xmin=4095,//Пиковое значение отрицательных импульсов
                 output wire [11:0]REF_P,//Порог компаратора положительных импульсов
                 output wire [11:0]REF_N,//Порог компаратора отрицательных импульсов
```

```
        output wire RXP,//Выход компаратора положительных импульсов
        output wire RXN,//Выход компаратора отрицательных импульсов
        output reg [10:0]AMP=0);// Средняя амплитуда импульсов в кадре

reg [11:0]tInp=`NS0 ;  //reg tRXP=0 ;  reg tRXN=0 ;
reg [6:0]Adr_wr=0;
wire [11:0]Amp_P = Xmax-Xns ;// Амплитуда положительного импульса
wire [11:0]Amp_N = Xns-Xmin ;//Амплитуда отрицательного импульса
assign REF_P = Xns+(Amp_P>>1) ;//Порог компаратора положительного импульса
assign REF_N = Xns-(Amp_N>>1) ;//Порог компаратора отрицательного импульса
assign RXP=(Amp_P>`Azer)? (MEM_dat>REF_P) : 0 ;
assign RXN=(Amp_N>`Azer)? (MEM_dat<REF_N) : 0 ;

always @ (posedge clk)  begin//if (ce)
Adr_wr <= ce? Adr_wr+1 : Adr_wr ;
tInp <= ce? Inp : tInp ;
Xmax <= res? Xns+`Azer : ((tInp>Xmax) & ce)? tInp : Xmax ; //-`dNS | res_P
Xmin <= res? Xns-`Azer : ((tInp<Xmin) & ce)? tInp : Xmin ; //+`dNS | res_N
AMP = res? (Xmax-Xmin)>>1 : AMP ;//
Xns <= res? tInp : Xns ;//
end

wire [6:0]Adr_rd = Adr_wr-`NT/4;//Задержка на 25 тактов (четверть Tbit)
//--Модуль памяти для задержки импульсов
SMEM_12x128 DD1 (       .clk(clk),      .DO(MEM_dat),
                        .we(ce),
                        .DI(tInp),
                        .Adr_wr(Adr_wr),
                        .Adr_rd(Adr_rd));
endmodule
```

Входными данными Inp[11:0] для этого модуля являются данные с выхода 12-и битного цифроаналогового преобразователя напряжения дифференциального усилителя сигналов Up и Un дифференциальной линии связи. В паузе между “словами” по сигналу res в регистре Xns[11:0] фиксируется смещение. Амплитуда оценивается как полу разность пиковых значений Xmax и Xmin и так же фиксируется по сигналу res в регистре Amp[10:0]. Сигнал res вырабатываются в модуле RX_TIMER.

Входной сигнал Inp[11:0] задерживается на Tce/4 (25 тактов) модулем памяти SMEM_12x128 (приложение 6.1). Адрес чтения Adr_rd меньше адреса записи Adr_wr на `NT/4.

2.2 Модуль таймера приемного блока

```
`include "CONST.v"
module RX_TIMER(
        input clk,      output wire ce,//Tce=Tbit/`NP
         input RXP,     output wire res,//Сброс в паузе
        input RXN,      output reg [7:0] N_RXPN=0,//Число импульсов в "слове"
                        output reg err=0); //Ошибка
reg tRXP=0, tRXN=0;
wire dRXP = RXP & !tRXP; //Фронт RXP
```

```
wire dRXN = RXN & !tRXN; //Фонт RXN
reg [5:0]cb_ce=0;
assign ce=(cb_ce==`Fclk/(`BR*`NT));
/*50000000/(100000*100)=5,...50000000/(12500*100)=40*/
reg [7:0] cb_RXPN=0 ;//Счетчик суммы импульсов в "слове"
//--Генератор ce
always @ (posedge clk) begin
cb_ce <= ce? 1 : cb_ce+1 ;
end
//---Генератор сброса
reg [8:0] cb_res=0  ;//Счетчик сброса в паузе между "словами"
assign res = (cb_res==(3*`NT)-1);//
reg tres = 0 ;
reg RCLK = 0 ;// RXP | RXN
always @ (posedge clk) if (ce) begin
tRXP <= RXP ;          tRXN <= RXN ;
tres <= res ;
RCLK <= (dRXP | dRXN) ;//
cb_res <= (RCLK | res)? 0 : cb_res+1 ;//:  ce? cb_res ;
N_RXPN <= res? cb_RXPN : N_RXPN ;
err <= (tres & !(N_RXPN==32))? 1 : 0 ;//Ошибка, если не 32 импульса
end
//---Счет суммы импульсов в "слове"
always @ (posedge tres or posedge RCLK) begin
cb_RXPN <= tres? 0 : cb_RXPN+1 ;
end
endmodule
```

В модуле RX_TIMER в режиме ожидания или в паузе между “словами” вырабатывается сигнал res. Первый импульс res после окончания “слова” задержан на 3*Tbit=300*Tce, а если сигнала нет, то res повторяется с периодом 3*Tbit.

Полное число импульсов в “слове” подсчитывается счетчиком cb_RXPN, и если после окончания “слова” N_RXPN не равно 32, то регистр ошибки err на один такт устанавливается в 1.

2.2 Схема моделирования приемника импульсов ARINC-429

```
module Test_RXM_ATR(
                input clk,              output wire [11:0]MTRP,
                input st,               output wire [11:0]MEM_dat,
                input [5:0] M,          output wire [11:0]Xmax,
                input S,                output wire [11:0]Xns,
                input res,              output wire [11:0]Xmin,
                                        output wire [11:0]REF_P,
                                        output wire [11:0]REF_N,
                                        output wire RXP,
                                        output wire RXN,
                                        output wire [10:0]AMP);

wire ce ;
Gen_MTRP DD2 (     .clk(clk),         .MTRP(MTRP),
```

```
                                .st(st),        .ce(ce),
                                .M(M),
                                .S(S));

RXM_ATRP DD3 (  .Inp(MTRP),   .MEM_dat(MEM_dat),
                .clk(clk),    .RXP(RXP),
                .ce(ce),      .RXN(RXN),
                .res(res),    .Xmax(Xmax),
                              .Xns(Xns),
                              .Xmin(Xmin),
                              .REF_P(REF_P),
                              .REF_N(REF_N),
                              .AMP(AMP));

Endmodule
```

2.2.1 Пример содержательной части задания на моделирование для **Test_RXM_ATR** (`BR=12500, Simulation Run Time=400 us)

```
always begin clk=0; #10; clk=1; #10; end
initial begin
                st = 0;  M = 0;  S = 0;  res = 0;
#3000;          st = 1;  M = 4;  S = 1;  res = 1;
#20;            st = 0;  res = 0;
#80000;         st = 1;  M = 3;  S = 0;  res = 1;
#20;            st = 0;                  res = 0;
#80000;         st = 1;  M = 5;  S = 1;  res = 1;
#20;            st = 0;  res = 0;
#80000;         st = 1;  M = 6;  S = 0;  res = 1;
#20;            st = 0;                  res = 0;
        end
```

Схема моделирования приема сигналов ARINC=429

```
module Test_RX_ATRP_AR_TXA(
                input clk,              output wire [11:0]TXA,
                input [5:0] M,          output wire [11:0]D_RXA,
                                        output wire [11:0]Xns,
                                        output wire RXP,
                                        output wire RXN,
                                        output wire [10:0] AMP,
                                        output wire en_RX,
                                        output wire res,
                                        output wire [7:0]N_RXPN,
                                        output wire en_TX,
                                        output wire err);
wire ce_rx ;
//--Генератор сигналов ARINC-429
```

```
AR_TXA DD1 (      .clk(clk),      .TXA(TXA),
                  .M(M),          .en_tx(en_TX));
//--Таймер приемного блока------------
RX_TIMER DD2 (    .clk(clk),              .ce(ce_rx),
                  .RXP(RXP),              .res(res),
                  .RXN(RXN),              .N_RXPN(N_RXPN),
                                          .err(err));
//--Приемник сигналов ARINC-429
RX_ATRP DD3(.Inp(TXA), .MEM_dat(D_RXA),
                  .clk(clk),              .RXP(RXP),
                  .ce(ce_rx),             .RXN(RXN),
                  .res(res),              .Xns(Xns),
                  .err(err),              .AMP(AMP),
                                          .en_RX(en_RX));
endmodule
```

2.2.1 Пример содержательной части задания на моделирование (`BR=100000, Simulation Run Time=2000 us)

```
always begin clk=0; #10; clk=1; #10; end
        initial begin
                 M=0;
#10000;          M=63; //2016
#420000;         M=32; //1024
#360000;         M=16; //512
#360000;         M=8;   //256
#360000;         M=4;   //128
#360000;         M=3;   //96
        end
```

Таблица 1

№	Скорость `BR[БОД]	Данные `my_DAT	Минимальная амплитуда `Azer
1	50000	31’h100007	40
2	12500	31’h200006	80
3	100000	31’h300005	100
4	12500	31’h400004	130
5	50000	31’h500003	70
6	100000	31’h600002	95
7	12500	31’h700001	200
8	50000	31’h100007	160
9	100000	31’h200006	130
10	12500	31’h300005	75
11	50000	31’h400004	50
12	100000	31’h500003	110
13	12500	31’h600002	170
14	50000	31’h700001	190
15	100000	31’h100007	80
16	12500	31’h200006	95
17	50000	31’h300005	110
18	100000	31’h400004	140

19	12500	31’h500003	220
20	50000	31’h600002	230
21	100000	31’h700001	200

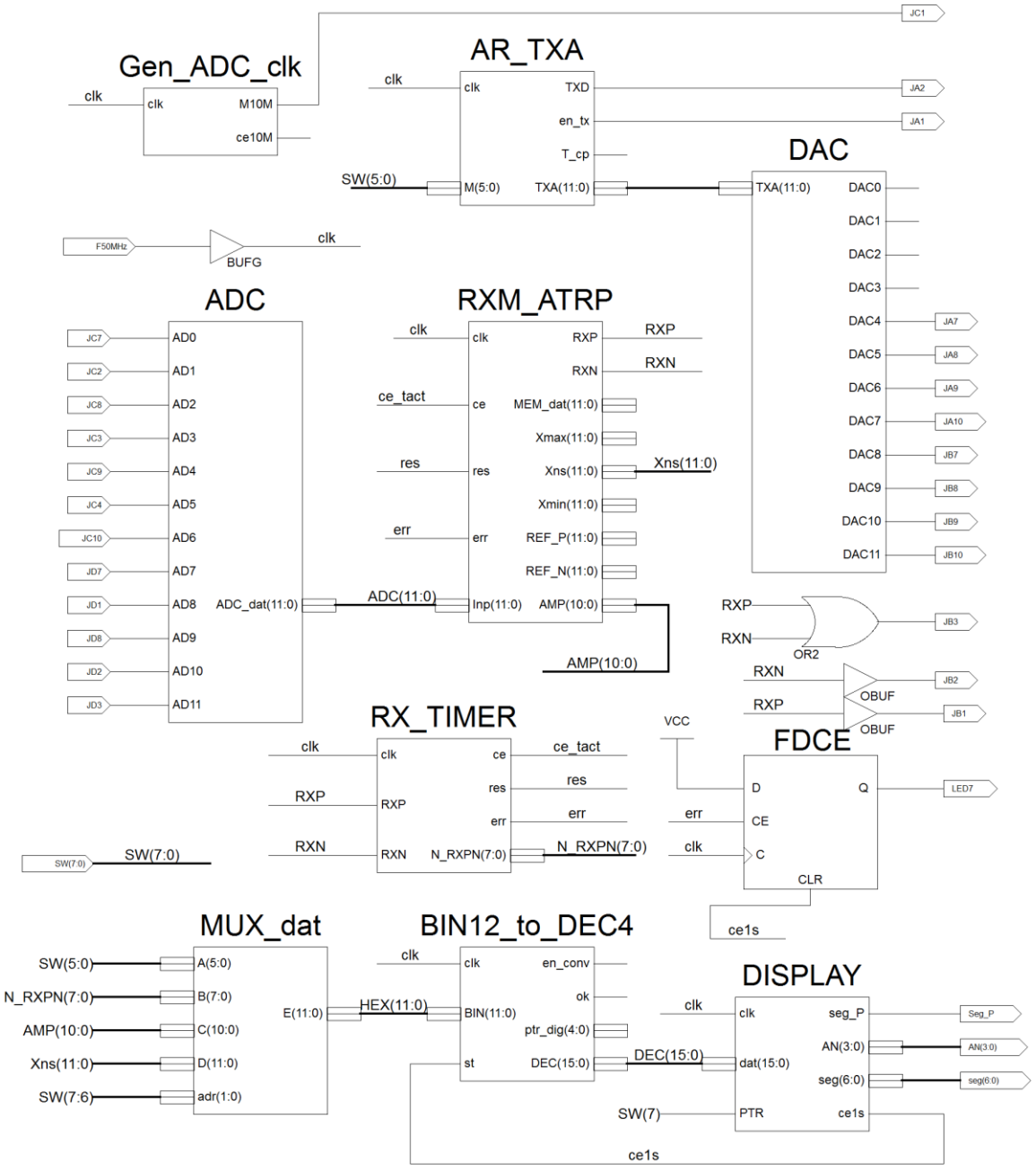

Рис.4 Схема лабораторной работы S_Sch_Lab405ADC

К выводам модуля DAC (нижний ряд выводов разъемов JA и JB макета NEXYS-2) должен подключаться макет цифроаналогового преобразователя DAC_R-2R (приложение 6.5.2), а к выводам модуля ADC (разъемы JC и JB макета NEXYS-2) должен подключаться макет параллельного аналого-цифрового преобразователя PADC (приложение 6.6.3). Выход XN1 ЦАП R-2R должен быть соединен проводной перемычкой с входом XN1 параллельного аналого-цифрового преобразователя.

Модуль Bin12_to_DEC (приложение 6.3) преобразует 12-и битные двоичные данные с выхода мультиплексора MUX_dat (приложение 6.2) в двоично десятичные данные для семи сегментного индикатора DISPLAY (приложение 6.4).

Сигнал синхронизации с частотой 10 MHz для параллельного АЦП вырабатывается модулем Gen_ADC_clk (см. приложение 6.6.2).

Модуль DAC (приложение 6.5.1) выполняет функцию расщепления шины TXA[11:0] в отдельные выводы DAC0,DAC1,…DAC11.

Модуль ADC (приложение 6.6.1) выполняет функцию “сжатия” входов AD0,AD1,…AD11 в шину ADC_dat[11:0].

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 NEXYS2 создать проект с именем Lab405ADC.

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

4.3 Создать Verilog модули и символы всех компонент схемы рис.4.

4.4 Создать Verilog модуль **Test_RXM_ATRP** схемы моделирования работы приема импульсов модулем **RXM_ATRP**. Привести моделирование. Зарисовать эскизы временных диаграмм сигналов: ATRP, MEM_dat, Xmax, Xmin, REF_P, REF_N, RXP, RXN, AMP. Определить минимальное значение M.

4.5 Создать Verilog модуль **Test_RXM_ATRP_AR_TXA** схемы моделирования работы приема “слов” ARINC-429 модулем **RXM_ATRP**. Привести моделирование. Зарисовать эскиз временных диаграмм сигналов: RXP, RXN, en_tx, AMP, M, res и err.

5. Задание к сдаче работы (стоимость 4)

5.1 Создать Schematic S_Sch_Lab405ADC или Verilog модуль V_Sch_Lab405ADC (см. приложение 6.7).

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**.

5.5 Вставить в разъемы JA, JB ЦАП DAC_R-2R, 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 Подобрать в модуле CONST.v dNS так, чтобы положительные и отрицательные импульсы на выходе ЦАП имели одинаковые амплитуды. Определить минимальную амплитуду при которой устойчиво формируются все 32 импульса RXP и RXN в “слове”. Сохранить осциллограммы сигналов с выхода ЦАП, JA1(en_tx), JB1(RXP), JB2(RXN), JB3(RXPN).

6. Приложения

6.1 Модуль памяти для задержки импульсов

```
module SMEM_12x128(
                    input clk,              output wire [11:0] DO,
                    input we,
                    input [11:0] DI,
                    input [6:0] Adr_wr,
                    input [6:0] Adr_rd);

reg [11:0]MEM[127:0] ;
assign DO = MEM[Adr_rd] ;
initial//Инициализация модуля памяти из файла SMEM_12x128.txt
$readmemh ("SMEM_12x128.txt", MEM, 0, 127);
always @ (posedge clk) begin
MEM[Adr_wr] <= we? DI : MEM[Adr_wr] ;
end
endmodule
```

6.2 Мультиплексор данных для индикатора

```
module MUX_dat(
                input [5:0] A,          output wire [11:0] E,
                input [7:0] B,
                input [10:0] C,
                input [11:0] D,
                input [1:0] adr );
assign E =(adr==0)? {10'h000,A} :
          (adr==1)? {4'h0,B} :
          (adr==2)? {1'b0,C} : D ;
endmodule
```

6.3 Преобразователь двоичных чисел в декадные

```
module BIN12_to_DEC4(
      input[11:0]BIN,       output reg[15:0]DEC=0,
      input clk,            output reg [4:0]ptr_dig=0,//Указатель номера декадной цифры
      input st,             output reg en_conv=0,
                            output reg ok=0);
//----Счетчики декадных цифр
reg[3:0]D1dec=0;    reg[3:0]D2dec=0;    reg[3:0]D3dec=0;    reg[3:0]D4dec=0;
```

```
reg q=0 ;
reg [16:0]rest;//Остаток

wire[15:0]Nd =(ptr_dig==4)? 1000 :
              (ptr_dig==3)? 100 :
              (ptr_dig==2)? 10 :
              (ptr_dig==1)? 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 & q & !z ;       //Разрешение инкремента декадной цифры
wire en_dec_ptr  =en_conv & !q & z ;       //Разрешение декремента указателя цифры

always @(posedge clk) begin
q<= st? 0 :en_conv? !q : q ;//q - для исключения необходимости восстановления остатка
en_conv<= st? 1 : (ptr_dig==0)? 0 : en_conv ; //Разрешение преобразования
rest<= st? BIN : en_inc_dig? dx : rest ;//Остаток
ptr_dig<= st? 4: en_dec_ptr?  ptr_dig-1 : ptr_dig ;//Указатель очередной декадной цифры
D4dec <= st? 0 : ((ptr_dig==4) & en_inc_dig)? D4dec+1 : D4dec ;
D3dec <= st? 0 : ((ptr_dig==3) & en_inc_dig)? D3dec+1 : D3dec ;
D2dec <= st? 0 : ((ptr_dig==2) & en_inc_dig)? D2dec+1 : D2dec ;
D1dec <= st? 0 : ((ptr_dig==1) & en_inc_dig)? D1dec+1 : D1dec ;
ok<=(ptr_dig==0) & en_conv;
DEC<= ok? {D4dec,D3dec,D2dec,D1dec}: DEC ;
end
endmodule
```

6.4 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
```

6.5 Цифроаналоговый преобразователь

6.5.1 Модуль DAC

```
module DAC(
        input [11:0] TXA,      output wire DAC0,
                               output wire DAC1,
```

```
                output wire DAC2,
                output wire DAC3,
                output wire DAC4,
                output wire DAC5,
                output wire DAC6,
                output wire DAC7,
                output wire DAC8,
                output wire DAC9,
                output wire DAC10,
                output wire DAC11);
assign DAC0 = TXA[0] ;
assign DAC1 = TXA[1] ;
assign DAC2 = TXA[2] ;
assign DAC3 = TXA[3] ;
assign DAC4 = TXA[4] ;
assign DAC5 = TXA[5] ;
assign DAC6 = TXA[6] ;
assign DAC7 = TXA[7] ;
assign DAC8 = TXA[8] ;
assign DAC9 = TXA[9] ;
assign DAC10 = TXA[10] ;
assign DAC11 = TXA[11] ;
endmodule
```

6.5.2 Макет цифроаналогового преобразователя DAC_R-2R

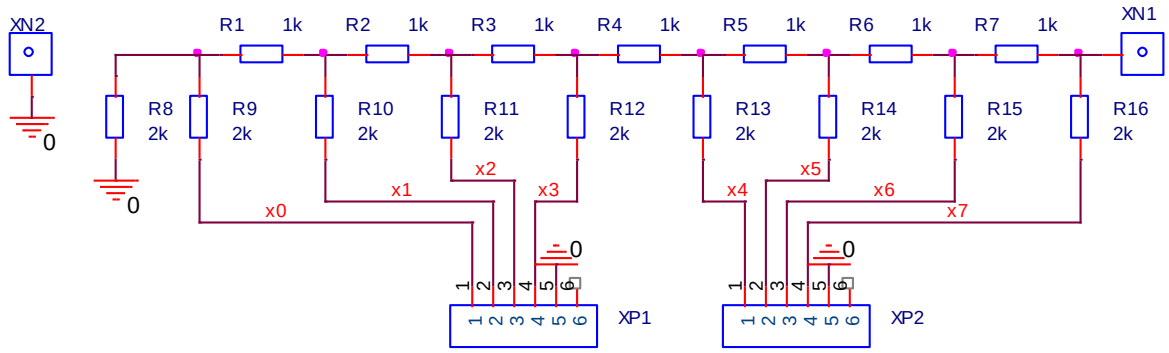

Рис. 5 Схема цифроаналогового преобразователя R-2R

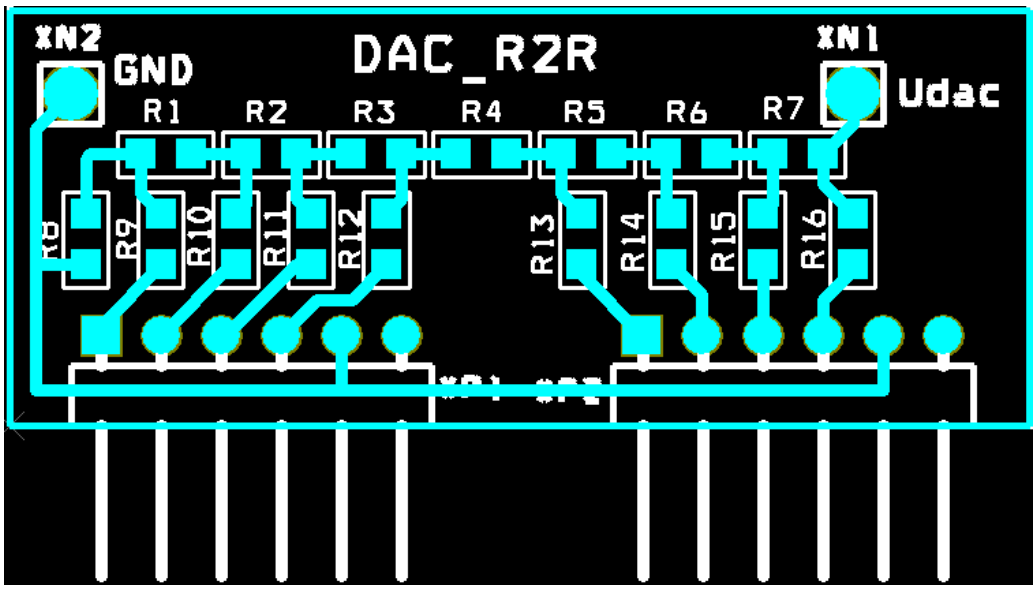

Рис. 6 Лицевая сторона платы цифроаналогового преобразователя DAC_R-2R

6.6 Параллельный аналого-цифровой преобразователь

6.6.1 Модуль ADC

```
module ADC(
        input AD0,              output wire[11:0] ADC_dat,
        input AD1,
        input AD2,
        input AD3,
        input AD4,
        input AD5,
        input AD6,
        input AD7,
        input AD8,
        input AD9,
        input AD10,
        input AD11    );
              //-----JD3,  JD2,   JD8,  JD1, JD7, JC10, JC4, JC9,  JC3, JC8,  JC2,  JC7
assign ADC_dat={AD11,AD10,AD9,AD8,AD7,AD6, AD5,AD4,AD3,AD2,AD1,AD0 };
endmodule
```

6.6.2 Генератор сигнала синхронизации для параллельного АЦП

```
module Gen_ADC_clk( input clk,    output wire M10M,
                                  output wire ce10M);
reg [2:0] cb_10M=0 ;
assign ce10M = (cb_10M==5) ;
assign M10M = !cb_10M[2] ;
always @ (posedge clk) begin
cb_10M  <= ce10M? 1 : cb_10M+1 ;
end
endmodule
```

6.6.3 Макет параллельного АЦП

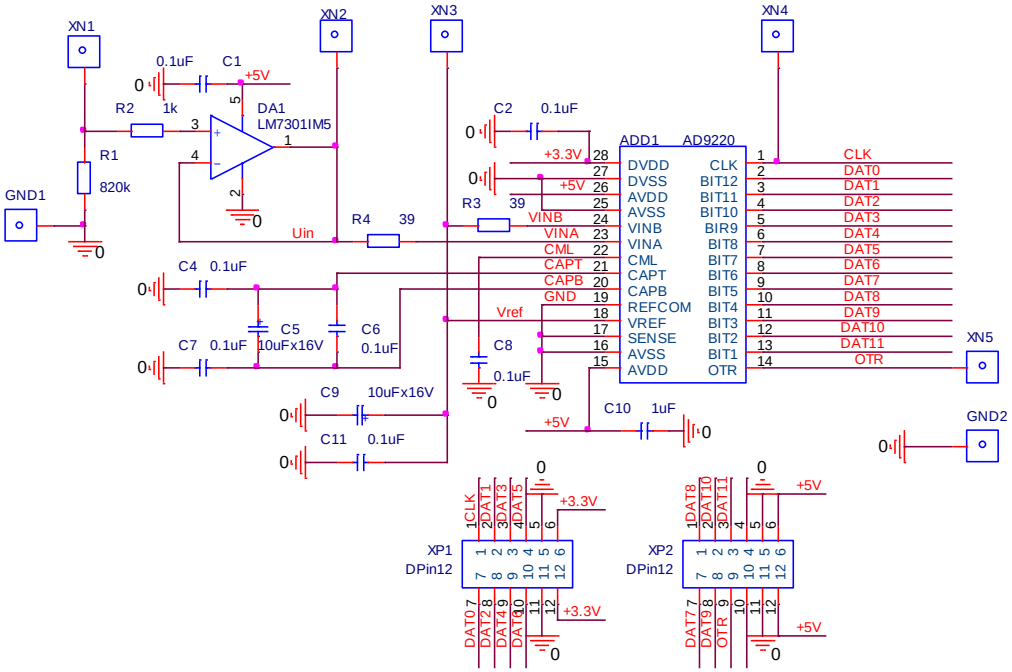

Рис. 7 Схема параллельного аналого-цифрового преобразователя

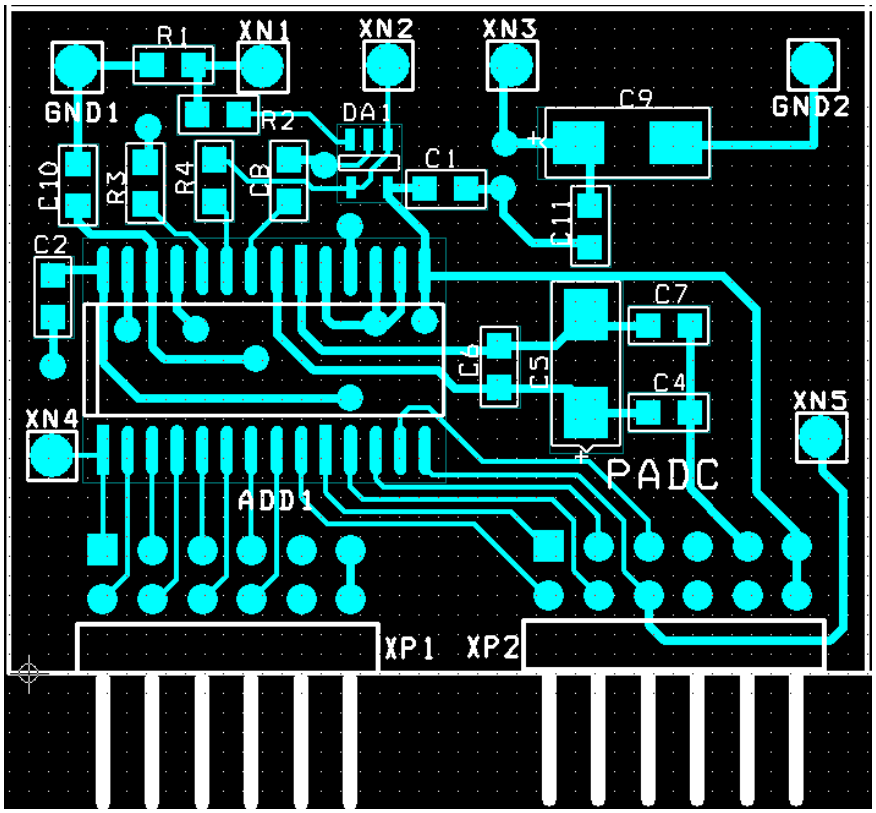

Рис. 8 Лицевая сторона печатной платы параллельного аналого цифрового преобразователя

6.7 Verilog модуль лабораторной работы

```
`include "CONST.v"
module V_Sch_Lab405ADC(
                input F50MHz,
                input [7:0]SW,
//---Выводы передающего модуля для ЦАП R-2R
                                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,//TXD,
//---Выводы приемного модуля
                                output wire JB1,//RXP
                                output wire JB2,//RXN
                                output wire JB3,//(RXP | RXN)
                                output reg LED7=0,   //Индикатор ошибки приема 32 импульсов
//---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 семи сегментного индикатора
                                output wire [3:0] AN,           //Аноды
                                output wire [6:0] seg,          //Сегменты
                                output wire seg_P);             //Точка

//---Данные парлельного АЦП ADC920
wire [11:0]ADC_dat = {JD3,JD2,JD8,JD1,JD7,JC10,JC4,JC9,JC3,JC8,JC2,JC7};//
//----------------------------------------------------------------------
wire clk, ce_tact, ce1s ;

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

```
//--Генератор сигнала синхронизации для АЦП ADC920
wire ADC_clk ;
Gen_ADC_clk DD2(.clk(clk),          .M10M(ADC_clk));
assign JC1 = ADC_clk ;
//----Генератор импульсов формата ARINC-429
wire [11:0]TXA ; wire AR_en_tx, AR_TXD ;
AR_TXA DD3 (        .clk(clk),        .TXA(TXA),
                    .M(SW[5:0]),  .en_tx(AR_en_tx),
                                      .TXD(AR_TXD));

assign JA1 = AR_en_tx ;       //Интервал кадра ARINC-429
assign JA2 = AR_TXD ;         //Последовательные данные

//-- Выходы генератора AR_TXA для ЦАП R-2R
assign JA7 = TXA[4] ;
assign JA8 = TXA[5] ;
assign JA9 = TXA[6] ;
assign JA10= TXA[7] ;
assign JB7 = TXA[8] ;
assign JB8 = TXA[9] ;
assign JB9 = TXA[10] ;
assign JB10= TXA[11] ;

//--Таймер приемного блока------------
wire RXP, RXN, res, err ;
wire [7:0]N_RXPN ;
RX_TIMER DD4 (    .clk(clk),        .ce(ce_tact),
                  .RXP(RXP),    .res(res),
                  .RXN(RXN),    .N_RXPN(N_RXPN),
                                .err(err));

//--Прием ый блок ----------------------
wire [11:0] Xns ; wire [10:0] AMP ;
RXM_ATRP DD5(
                .Inp(ADC_dat),        .RXP(RXP),
                .clk(clk),            .RXN(RXN),
                .ce(ce_tact),         .Xns(Xns),
                .res(res),            .AMP(AMP),
                .err(err));
assign JB1= RXP ; //Выход компаратора положительных импульсов
assign JB2= RXN ; //Выход компаратора отрицательных импульсов
assign JB3= (RXP | RXN) ;//

//--Мультиплексор данных для индикатора
wire [11:0]HEX_dat ;
MUX_dat DD6 (       .A(SW[5:0]),  .E(HEX_dat),
                    .B(N_RXPN),
                    .C(AMP),
                    .D(Xns),
                    .adr(SW[7:6]));
```

```
//--Преобразователь двоичных чисел в декадные
wire [15:0]DEC_dat ;
BIN12_to_DEC4 DD7 (
                .BIN(HEX_dat),        .DEC(DEC_dat),
                .clk(clk),
                .st(ce1s));

//--Семи сегментный светодиодный индикатор
DISPLAY DD8 (.clk(clk),             .AN(AN),              //Аноды
                .dat(DEC_dat),      .seg(seg),            //Сегменты
                 .PTR(SW[7]),       .seg_P(seg_P),        //Точка
                                    .ce1s(ce1s));   //Секунда

always @ (posedge clk or posedge ce1s) begin
LED7 <= ce1s? 0 : err? 1 : LED7 ;
end
endmodule
```

6.8 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" ; # M[0]
NET "SW<1>" LOC = "H18" ; # M[1]
NET "SW<2>" LOC = "K18" ; # M[2]
NET "SW<3>" LOC = "K17" ; # M[3]
NET "SW<4>" LOC = "L14" ; # M[4]
NET "SW<5>" LOC = "L13" ; # M[5]
NET "SW<6>" LOC = "N17" ; # adr[0]
NET "SW<7>" LOC = "R17" ; # adr[1], PTR
```

```
#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" ; # AR_en_tx
NET "JA2" LOC = "K12" ; # AR_TXD
#NET "JA3" LOC = "L17" ; #
#NET "JA4" LOC = "M15" ; #
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" ; #
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" ;#
```

6.9 Файл инициализации модуля памяти SMEM_12x128.txt

```
800
800
```

800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800

800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800

800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800
800