Последовательный канал информационного обмена по стандарту **MIL-STD-1553**

(ГОСТ 26765.52-87, ГОСТ Р. 52070-2003)

Лабораторная работа №406_ADC

Стандарт MIL-STD-1553, изначально разрабатывался по заказу МО США для использования в военной бортовой авионике. Впервые опубликован в США как стандарт ВВС в 1973 году, применён на истребителе F-16 . Принят в качестве стандарта НАТО — STANAG 3838 AVS. Позднее спектр его применения существенно расширился, стандарт стал применяться и в гражданских системах.

Данные передаются по витой проводной паре последовательно словами по 16 бит. Длительность каждого слова 20 мкс и состоит из 20 тактов по 1 мкс. В первые три такта передаются 2 импульса синхронизации с длительностью 1.5 мкс каждый. Затем в течение 16 тактов передаются 16 бит данных (D[15:0] - старшими битами вперед) и на последнем 20-м такте передается бит контроля четности (дополнение до нечетности числа единиц в слове). Полярность импульсов синхронизации определяется назначением слова. Например, в командном слове (CW) и в ответном слове (RW) первый импульс синхронизации положительный, а в слове данных (DW) – отрицательный. В качестве кода передачи используется биполярный фазоманипулированный код (Манчестер II). Биты данных передаются не потенциально, а перепадом напряжения в центре такта. Перепад напряжения от “+” к “–“ (⎾⏌) соответствует единице, а перепад от “–“ к “+” (⏌⎾) соответствует нулю. Размах напряжения (Umax-Umin) на линии может быть в интервале от 1.4 В до 20 В (амплитуда от 0.7 до 10 В). Пример временных диаграмм контрольного слова CW[15:0]=16’h4F15 и слова данных DW[15:0]=16’hB0EA приведен на рис. 1.

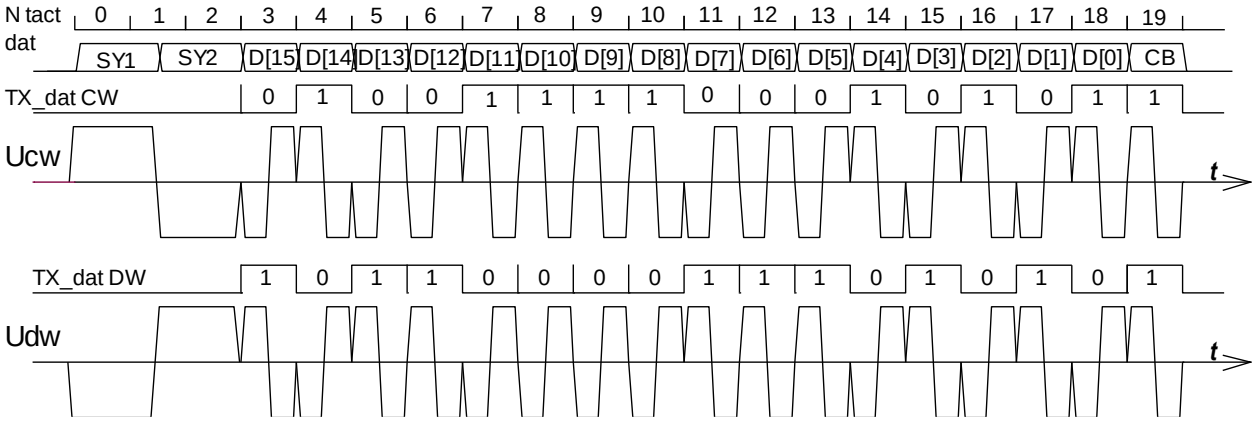

Рис.1 Пример временных диаграмм сигналов контрольного слова и слова данных стандарта MIL-STD-1553

Код Манчестер II является самосинхронизирующимся, т.е. он передает не только данные, но и эталон времени передатчика. В середине каждого такта данных обязательно имеется перепад напряжения, по которому можно принимать данные и синхронизировать эталон времени приёмника.

Слова данных передаются без промежутка (“впритык”) к командному слову или ответному слову так и между собой.

На первом этапе выполнения данной работы в системе проектирования цифровых устройств на ПЛИС моделируется работа модулей:

- Gen_SY_SD – генератора импульсов синхронизации и данных
- MIL_DET_SY – детектора сигналов синхронизации слов CW и DW.
- MIL_TX_DAC – генератора сигналов слов CW и DW.

На втором этапе выполнения на основе предложенных алгоритмов и временных диаграмм дается задание на проектирование и отладку модуля MIL_DET_DAT и модуля MIL_RX_ADC приемника сигналов слов CW и DW.

На этапе сдачи работы для макета NEXYS2 составляется схема, в состав которой входят отлаженные модули, и проверяется их работа.

1.1 Схема модуля ждущего генератора импульсов синхронизации и данных

```
`include "CONST.v"
module Gen_SY_SD(
        input clk,      output reg [7:0] TRP=0,      //Трапецеидальный импульс
        input st,       output wire [7:0] MTRP,      //MPTR=TRP*M/128
        input [7:0] M, output wire ce_stop,          //Конец импульса
        input D,        output wire [7:0] SMTRP_SH,//Смещенный импульс
        input SY_SD, output reg [7:0]cb_tact=0,      //Счетчик тактов
                        output reg QM=0,             //Полярность импульса
                        output reg T_TRP=0,          //Интервал импульса
                        output wire T_front,         //Интервалы фронтов
                        output wire T_spad);         //Интералы спадов

//M- множитель амплитуды, SY_SD - SY/dat, D - CW_DW/bit
wire [7:0]NP = SY_SD? `Nsy : `NT ; //Число точек длительностей импульсов SY и SD
assign T_front =((cb_tact<`Nfr) | ((cb_tact>NP/2-1) & ((cb_tact<NP/2+`Nfr)))) & T_TRP ;
assign T_spad = (((cb_tact>NP/2-(`Nfr+1)) & (cb_tact<NP/2)) |
                    ((cb_tact>NP-(`Nfr+1)) & (cb_tact<NP))) & T_TRP ;
assign ce_stop = (cb_tact==NP-1) ;
wire [15:0]AMTRP= TRP*M ; //Умножение на M
assign MTRP = AMTRP>>7 ; //Деление на 2^7

always @ (posedge clk) begin
T_TRP <= st? 1 : ce_stop? 0 : T_TRP ;
cb_tact <= (st | ce_stop)? 0 : T_TRP ? cb_tact+1 : cb_tact ;
TRP <= (st | ce_stop)? 0 : T_front? TRP+`dTRP : T_spad? TRP-`dTRP : TRP ;//Было 31
QM <= (st | ce_stop)? 0 : (cb_tact==NP/2)? 1 : QM;
end
assign SMTRP_SH = D^QM? `SH0+MTRP : `SH0-MTRP ;
endmodule
```

В этом модуле по сигналу st формируется пара биполярных трапецеидальных импульсов. В паузе между запусками выходной сигнал SMTRP_SH равен смещению SH=128.

Амплитуда AMP=120*M/128 каждого импульса может принимать значения от 9 (M=1) до 120 (M=128). Длительность пары импульсов при SY_SD=1 (синхронизация) равна 150*Tclk=3 us, а при SY_SD=0 (данные) равна 50*Tclk=1 us. Входной сигнал D определяет

назначение выходного сигнала SMTRP_SH: D=1 – синхронизация контрольного слова или бит “1”, D=0 – синхронизация слова данных или бит “0”.

Выходной сигнал SMTRP_SH предназначен для 8-ми битного цифроаналогового преобразователя R2R (ЦАП). Напряжение на выходе этого ЦАП Udac=3.3V*X/256, где X= SMTRP_SH[7:0]. В паузе между импульсами (SMTRP_SH[7:0]=128) напряжение на выходе ЦАП Udac=1.65V. Максимальная амплитуда выходного напряжения ЦАП (M=128, AMP=120) равна 3.3V*120/256=1.55V, а минимальная амплитуда (M=1, AMP=9) ~13mV.

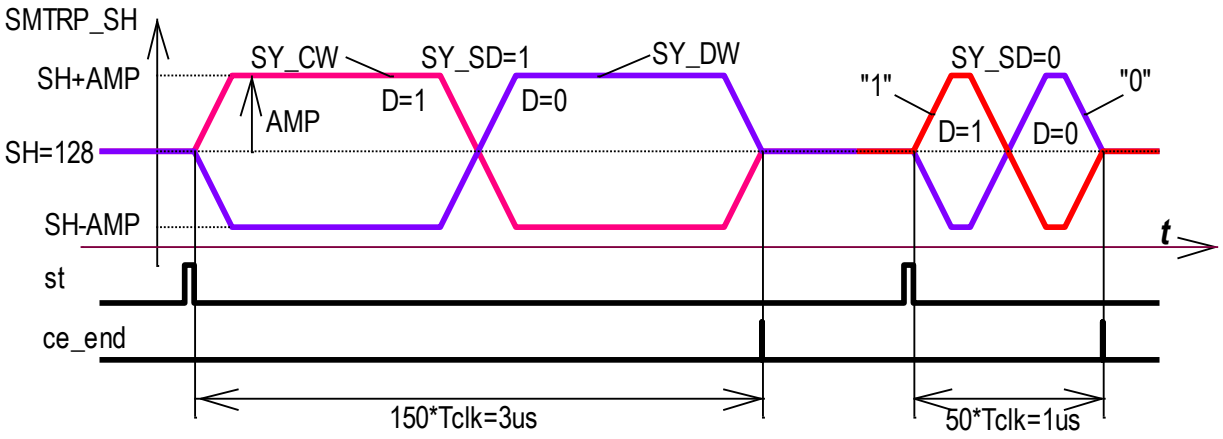

Рис.2 Временные диаграммы сигналов модуля **Gen_SY_SD** ждущего генератора импульсов синхронизации и данных

1.2 Vодуль параметров CONST.v

```
`define Fclk        50000000 //50 MHz
`define F10MHz 10000000 //10 MHz
`define Fce          1000000 //1 MHz
`define F1kHz           1000 //1 kHz
`define F1Hz               1 //1 Hz
`define SH0    8'h80     //Номинал смещения
`define dTRP     14      //Шаг изменения амплитуды TRP
`define Nsy      150     //Число тактов сигнала синхронизации Tsy=`Nsy*Tclk
`define NT       50      //Число тактов длительности бита данных Tbit=`NT*Tclk
`define Nfr       8      //Число тактов длительностей фронтов и спадов
`define N_DW     1       //Число слов данных
`define REF_SY    52     //Порог счетчика SY
`define REF_X     24     //Порог счетчика X
`define Amin       6     //Минимальная амплитуда
`define my_CW   16'h0123 //Контрольное слово
`define my_DW   16'hFEDC //Слово данных
```

1.3 Схема модуля генератора сигналов контрольного слова и слов данных

```
`include "CONST.v"
module MIL_TX_DAC(
module MIL_TX_DAC(output wire [7:0]MIL_DAC,//Данные для ЦАП
        input clk,          output reg T_SY=0,     //Интервал синхронизации
        input[15:0]dat,     output reg en_tx=0,  //Разрешение передачи
        input st,           output reg T_dat=0, //Интервал данных
        input [7:0]M,       output wire T_cp,     //Такт контрольного бита
```

```
                output wire TX_bit, //Последовательные данные
                output reg FT_cp=1,     //Счетчик четности
                output reg [4:0]cb_bit=0,//Счетчик бит слова
                output reg [5:0]cb_DW=0, //Счетчик слов данных
                output wire CW,      //Контрольное слово
                output wire ce_tact);//Границы бит

assign CW = (cb_DW==0) ;  //CW=1-контрольное слово, CW=0-слово данных
assign T_cp = (cb_bit==19) ;  //Интервал контрольного бита
reg [5:0]cb_tact=0 ;              //Счетчик такта
assign ce_tact = (cb_tact==`Fclk/`Fce) ;//Границы бит
wire ce_end = T_cp & ce_tact ;//Конец кадра
reg tce_end=0, tce_tact=0, tst=0;
reg [15:0]sr_dat=0 ;              //Регистр сдвига данных

always @ (posedge clk) begin
tce_end <= ce_end ; tce_tact <= ce_tact ; tst <= st ;
cb_tact <= (st | ce_tact)? 1 : cb_tact+1 ;
cb_bit <= (st | ce_end)? 0 : (ce_tact & en_tx)? cb_bit+1 : cb_bit ;
cb_DW <= st? 0 : (en_tx & ce_end)? cb_DW+1 : cb_DW ;
T_SY <= (st | (en_tx & tce_end))? 1 : ((cb_bit==2) & ce_tact)? 0 : T_SY ;
T_dat <= ((cb_bit==2) & ce_tact & en_tx)? 1 : (ce_tact & (cb_bit==18))? 0 : T_dat ;
en_tx <= st? 1 : ((cb_DW==`N_DW) & ce_end)? 0 : en_tx ;
sr_dat <= (tst | tce_end)? dat : (T_dat & ce_tact)? sr_dat<<1 : sr_dat ;
FT_cp <= (st | ce_end)? 1 : (T_dat & ce_tact & sr_dat[15])? !FT_cp : FT_cp ;
end
assign TX_bit = T_SY? CW : T_dat? sr_dat[15] : T_cp? FT_cp : 0 ;

// Генератор сигналов импульсов синхронизации и данных
wire st_SY_SD = tst | (tce_tact & !T_SY & en_tx) ;//
Gen_SY_SD DD1 (.clk(clk),      .SMTRP_SH(MIL_DAC),//Смещенный импульс
                .M(M),          //.ce10MHz(ce10MHz),
                .st(st_SY_SD),
                .D(TX_bit),
                .SY_SD(T_SY));
endmodule
```

В состав этого модуля входит описанный выше модуль ждущего генератора импульсов синхронизации и данных. При первом запуске в начале слова формируется импульс T_SY с длительностью 3*Tbit=3us, который по входу (SY_SD=T_SY=1) модуля Gen_SY_SD обеспечивает формирование сигнала синхронизации слова, а при следующих 17 запусков (SY_SD=T_SY=0) формируются импульсы данных.

Счетчик слов данных cb_DW определяет не только номер слова данных, но и назначение слова. При cb_DW=0 формируется сигал синхронизации соответствующий контрольному слову.

Триггер en_tx по st устанавливается в 1, а сбрасывается в 0 только в конце последнего слова данных слова (cb_DW == N_DW).

В регистр сдвига данных sr_dat[15:0] по st загружаются данные dat[15:0] контрольного слова, а по импульсам ce_end концов слов данные очередных слов данных. На интервале T_dat 16-и бит данных по сигналу ce_tact с периодом Tbit=1us данные регистра сдвигаются

в лево (sr_dat<<1). Старший разряд sr_dat[15] на каждом такте по входу D модуля Gen_SY_SD определяет значение передаваемого бита данных.

Триггер FT_cp является однобитным счетчиком числа единиц данных. В начале каждого слова он устанавливается в 1 и на интервале T_dat 16-и бит данных переключается столько раз, сколько единиц в слове и на последнем такте T_cp передается его состояние, автоматически дополняющее количество единиц в полном слове до нечетного.

1.4 Схема модуля детектора сигнала синхронизации слов CW и DW

```
`include "CONST.v"
module MIL_DET_SY(output wire [7:0] D1Inp,   //Задержанный на 1.5us сигнал Inp
        input [7:0]Inp,     output wire [7:0] D2Inp,       //Задержанный на 3.0us сигнал Inp
        input clk,          output reg [7:0] SH=`SH0,    //Измеренное смещение
        input res,          output reg [7:0] AMP=0,      //Измеренная амплитуда
                            output wire RXP1,   //D1Inp>REF_P
                            output wire RXN1,  //D1Inp<REF_N
                            output wire RXP2,   //D2Inp>REF_P
                            output wire RXN2,  //D2Inp<REF_N
                            output wire RXPN,  //RXPN <= (RXP1 | RXN1)
                            output wire SY,   //Интервал совпадения сигналов компараторов
                            output reg [7:0] cb_SY=0,
                            output wire ok_SY //Есть сигнал синхронизации CW или DW
                            );
reg [27:0]ACC=0 ;//Аккумулятор смещения
reg [19:0]cb_ACC=0 ;//Счетчик интервала накопления
wire ceACC= (cb_ACC==(1<<20)-1);
reg [7:0] PIC_max = `SH0 ;//Пиковый детектор максимума
reg [7:0] PIC_min = `SH0 ;//Пиковый детектор минимума
wire [7:0] AMP_P = (PIC_max-SH) ; //(PIC_max-SH)/2
wire [7:0] AMP_N = (SH-PIC_min) ; //(SH-PIC_min)/2
wire [7:0]REF_P= SH+(AMP_P>>1) ;
wire [7:0]REF_N= SH-(AMP_N>>1) ;
assign RXP1=(AMP_P>`Amin) & (D1Inp>REF_P);
assign RXN1=(AMP_N>`Amin) & (D1Inp<REF_N);
assign RXPN = (RXP1 | RXN1) ;
assign RXP2=(AMP_P>`Amin) & (D2Inp>REF_P);
assign RXN2=(AMP_N>`Amin) & (D2Inp<REF_N);
assign SY=(RXP1 & RXN2) | (RXP2 & RXN1);
assign ok_SY = (cb_SY > `REF_SY) ;//Есть сигнал синхронизации CW или DW

reg [7:0] cb_adr=0;//Адрс записи входного сигнала Inp
always @ (posedge clk) begin
cb_ACC <= cb_ACC+1 ;
ACC <= ceACC? Inp : ACC+Inp ;//Inp
SH <= ceACC? ACC>>20 : SH ;
cb_adr <=  cb_adr+1 ;//Адес записи Inp в модуль памяти
PIC_max <= (res | ceACC)? SH : ((Inp>PIC_max))? Inp : PIC_max ;//
PIC_min <= (res | ceACC)? SH : ((Inp<PIC_min))? Inp : PIC_min ;//
cb_SY <= SY? cb_SY+1 : 0 ; // Счетчик длительности сигнала синхронизации
AMP <= (res | ceACC)? (PIC_max-PIC_min)>>1 : AMP ;
end
```

```
//Задержка сигнала Inp на 75*Tclk=11.5us и на 150*Tclk=3.0us
wire we=1 ;//Разрешение записи
wire [7:0]Adr1_rd=cb_adr-75 ;  //Адес чтения задержанного сигнала D1Inp
wire [7:0]Adr2_rd=cb_adr-150 ;//Адес чтения задержанного сигнала D2Inp
MEM8x256 DD1 (  .clk(clk),          .D1O(D1Inp),//Задержка на  75*Tclk=1.5us
                .we(we),            .D2O(D2Inp),//Задержка на 150*Tclk=3.0us
                .DI(Inp),
                .Adr_wr(cb_adr),
                .Adr1_rd(Adr1_rd),
                .Adr2_rd(Adr2_rd));
endmodule
```

В этом модуле автоматически определяется смещение SH и амплитуда AMP входного сигнала с целью установки порогов REF_P и REF_N компараторов на относительном уровне половины амплитуды: REF_P=SH+AMP_P/2, REF_N=SH-AMP_N/2. Смещение SH определяется как среднее значение сигнала Inp за время Tclk*2^20~2s. Для оценки амплитуд используются пиковые детекторы (PIC_max) максимальных и (PIC_min) минимальных значений сигнала Inp, Амплитуда и пороги компараторов определяются по входному (не задержанному сигналу), на входы же компараторов подаются задержанные сигналы D1Inp и D2Inp с двух выходов модуля задержки, задержанные на половину (1.5us) и целую (3.0us) длительность сигнала синхронизации (см. рис.3).

Логические произведения (RXN1 & RXP2) или (RXP1 & RXN2) выходных сигналов компараторов имеют длительность около половины длительности сигнала синхронизации слова (1.5us) только на интервале второй половины сигнала синхронизации задержанного сигнала D1Inp, а на остальных интервалах слов эта длительность не превышает половины длительности бита (0.5us). Поэтому сигнал синхронизации однозначно обнаруживается по длительности сигнала SY=(RXP1 & RXN2) | (RXP2 & RXN1) при помощи счетчика cb_SY и компаратора ok_SY = (cb_SY>REF_SY). Максимальная длительность сигнала SY зависит от длительности фронтов и спадов импульсов входного сигнала. Например, при длительности фронта (спада) Tfr=8*Tclk максимальная длительность SY равна 1340ns (1500ns-8*20ns=1340). При такой длительности фронта максимальное значение счетчика cb_SY равно 1340/20=67. Надо учитывать еще и неодинаковость эталонов времени приемника и передатчика. Например, если счетчик cb_SY инкрементируется на интервале SY с периодом Tclk большим номинального значения 20ns, например 21ns, то порог REF_SY должен быть еще меньше REF_SY $\leq$ 1340/21=63. На нижний предел порога REF_SY также влияет неодинаковость эталонов времени. Например, при минимальной длительности фронта Tfr=0 и периоде сигнала синхронизации приемника не 20ns, а 19ns за пределами сигнала SY максимальное значение cb_SY может быть равно 53 (1000/19=52,6). Поэтому, если рассчитывать на максимальную относительную разность эталонов времени передатчика и приемника $\pm 5\%$, то порог REF_SY можно установить в середине допустимого диапазона REF_SY=(63+53)/2=28.

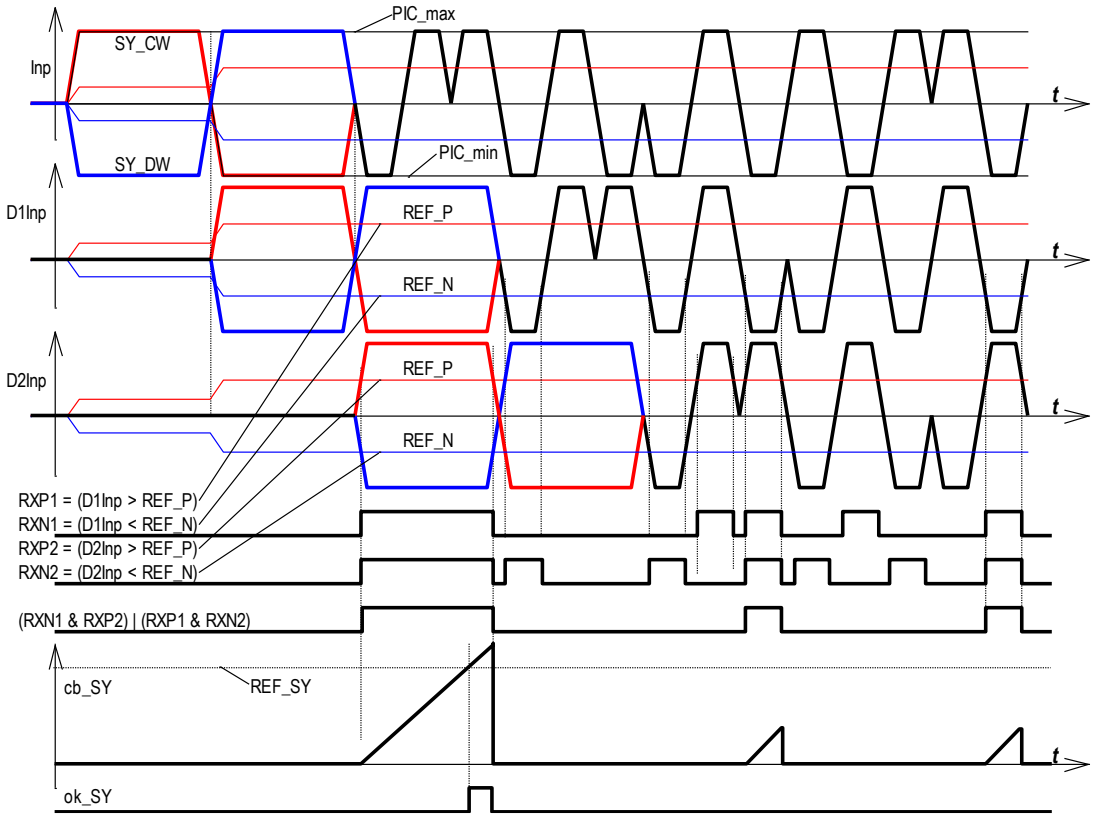

Рис.3 Временные диаграммы модуля MIL_DET_SY детектора сигнала синхронизации

1.4.1 Схема модуля памяти для задержки

```
module MEM8x256(
        input clk,          output reg [7:0] D1O,
        input we,           output reg [7:0] D2O,
        input [7:0] DI,
        input [7:0] Adr_wr,
        input [7:0] Adr1_rd,
        input [7:0] Adr2_rd);

reg [7:0]MEM[255:0] ;
//assign DO = MEM[Adr_rd] ; //Слайсовая память
initial //Инициализация модуля памяти из файла init_MEM8x256.txt
$readmemh ("init_MEM8x256.txt", MEM, 0, 255);
always @ (posedge clk) begin
MEM[Adr_wr] <= we? DI : MEM[Adr_wr] ;
D1O <= MEM[Adr1_rd] ; //Блочная память
D2O <= MEM[Adr2_rd] ; //Блочная память
end
endmodule
```

Для задержки сигнала используется модуль памяти MEM8x256.

Этот модуль памяти (8 бит на 256 ячеек) инициализируется текстовым файлом init_MEM8x256.txt, в котором должно быть 256 строк данных ячеек в HEX формате. В данном случае это 80 - номинальное значение смещения (8'h80=128).

1.5 Детектор бит данных слов CW и DW MIL-1553

Схему модуля MIL_DET_DAT детектора бит данных слов составить самостоятельно.

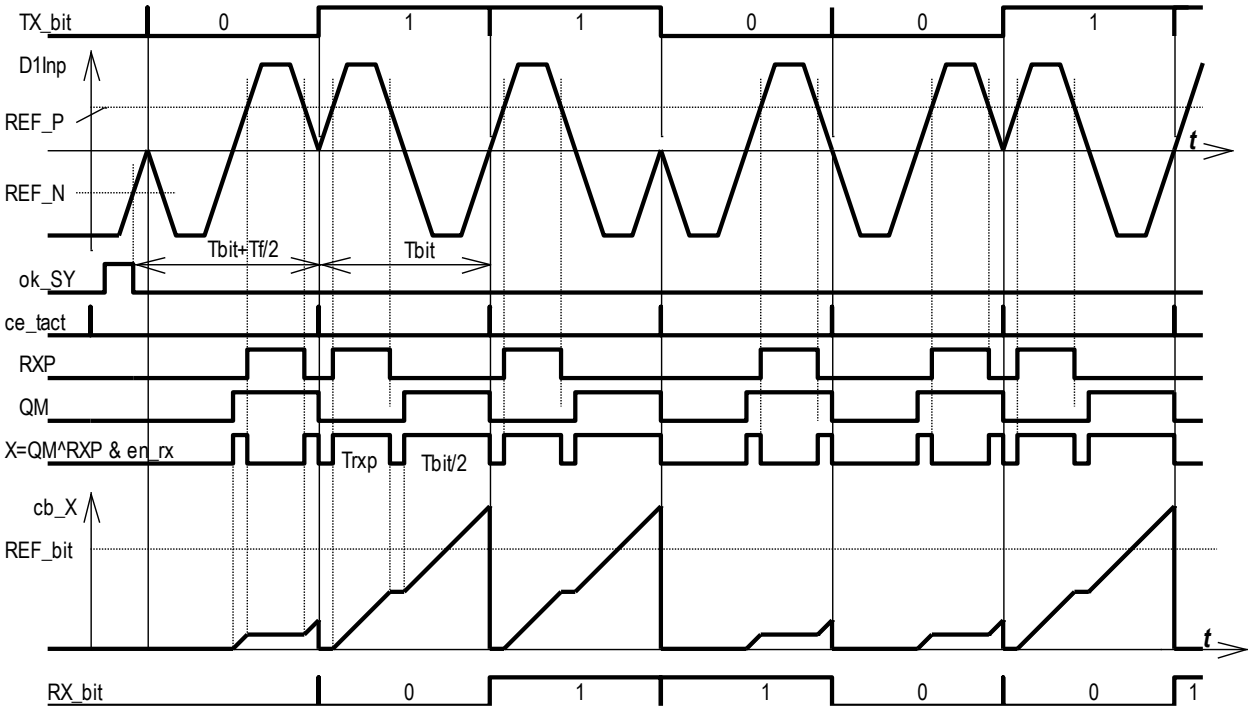

Рис.4 Временные диаграммы модуля MIL_DET_DAT детектора данных слов CW и DW

При составлении схемы детектора данных предполагается, что известны номинальные значения длительностей фронтов и периодов импульсов данных:

Tbit=Ntact*Tclk=50*20ns=1000nx=1us (Ntact-Nd=50),

Tfr=Nfr*Tclk=8*20ns=160ns

Для декодирования бит удобно использовать опорный сигнал QM (см. рис.4), эквивалентный сигналу RXN слова, в котором все биты 0. Сигнал X (“исключающее или” RXP и QM) X=RXP^QM равен 0 при равенстве RXP=QM и равен 1 при не равенстве RXP и QM. Например, при нулевой длительности фронтов X совпадает с TX_bit передаваемых данных. При Tfr не равном 0 X не остается постоянным на интервале T_bit, поэтому для исключения необходимости определения точки считывания X надежнее считывать не сам сигнал X, а в конце такта сигналом ce_tact “интеграл” на интервале T_bit от X. Интегратор X это счетчик cb_X, который при X=1 инкрементируется, например, с периодом Tclk и в конце каждого такта сигналом ce_tact “сбрасывается” в 0:

cb_X <= (ok_SY | ce_tact)? 0 : (X & en_rx)? cb_X+1 : cb_X ;

en_rx <= ok_SY? 1 : (ce_tact & (cb_bit==16))? 0 : en_rx ;

cb_bit <= ok_SY? 0 : (en_rx & ce_tact)? cb_bit+1 : cb_bit ;

При Tclk=20ns и Tbit=1us максимальное значение cb_Xmax = 48-8=40, а минимальное значение cb_Xmin = 8. Если в конце такта cb_X>REF_bit, то триггер RX_bit (приемник бит) надо устанавливать в 1, а при cb_X<=REF_bit надо устанавливать в 0, где REF_bit, например, равен (cb_Xmax + cb_Xmin)/2=24.

Опорный сигнал QM можно формировать при помощи счетчика cb_QM[5:0], который сигналом ok_SY устанавливается в -Nfr/2, а по ce_tact сбрасывается в 1, и на интервале T_bit инкрементируется с периодом Tclk:

cb_QM <= ok_SY? -Nfr/2 : ce_tact? 1 : cb_QM+1 ;

Триггер QM по ok_SY | ce_tact сбрасывается в 0, а по cb_QM== Ntact/2 устанавливается в 1:

QM <= (ok_SY | ce_tact)? 0 : ((cb_QM==Ntact/2) & en_rx)? 1 : QM ;

Для контроля правильности принятого слова удобно использовать однобитный счетчик FT_cp (Т-триггер), которым подсчитывается количество единиц в слове, включая контрольный бит. Если в начале каждого слова сигналом ok_SY триггер FT_cp устанавливать в 0, то при нечетном числе 1 в слове после окончания приема слова FT_cp переключившись нечетное число раз будет равен 1, что и будет означать нечетное число единиц в принятом слове т.е. отсутствие ошибки контроля четности:

FT_cp <= ok_SY? 0 : (ce_tact & en_rx & (cb_X>REF_X))? !FT_cp : FT_cp ;

Сигнал ok_rx разрешения считывания слова ok_rx = FT_cp & tce_end, где tce_end задержанный на Tclk ce_end (ce_end = ce_tact & (cb_bit==16)).

RS-триггер CW_DW – указатель принятого слова должен устанавливаться в 1 сигналом ok_SY & RXN, а сбрасываться в ноль после приема контрольного слов сигналом ok_SY & RXN. Если в приемнике есть счетчик cb_DW принятых слов, то признаком контрольного слова может служить CW=(cb_DW==0).

Эталоны времени (периоды сигналов синхронизации clk_tx и clk_rx) передатчика и приемника не могут быть точно одинаковые. Например, если периоды сигналов синхронизации приемника и передатчика отличается на ±3%, то к 17-му такту (контрольному биту) смещение Tsh центра контрольного бита приемника будет больше половины длительности бита передатчика Tsh=±0.51*Tbit, что исключит возможность правильного приема контрольного бита.

Для подстройки эталона времени приемника к эталону времени передатчика необходимо в центре каждого такта проводить принятыми сигналами коррекцию счетчика такта cb_tact приемника. Положение центра такта передатчика можно оценить только по паре сигналов RXP и RXN. Сигнал логической суммы RXPN=RXP | RXN равен нулю в паузах между импульсами RXP и RXN (см. рис.5). Длительности пауз в середине такта и на границах такта равны длительности фронта Tfr=Nfr*Tclk. Если счетчик такта cb_tact в середине такта, например, в интервале коррекции en_corr=(cb_QM>=14) & (cb_QM<32), сигналом T_corr = en_corr & !RXPN принудительно удерживать в ожидаемом значении (Ntact-Nfr)/2=21, а сбрасывать в 1 не при Ntact=50, а при Ntact-Nfr=42 (ce_tact = (cb_tact==NT-Nfr)), то при одинаковых периодах Tclk передатчика и приемника фронт сигнала QM будет точно в середине такта принятого сигнала передатчика:

cb_tact <= ok_SY? -Nfr/2 : ce_tact? 1 : T_corr? (NT-Nfr)/2 : cb_tact+1 ;

Принятый бит данных RX_bit устанавливается в 1 импульсом ce_tact при cb_X>REF_X,

RX_bit <= ok_SY? 0 : (ce_tact & en_rx)? (cb_X>REF_X) : RX_bit ;

На вход младшего разряда приемного регистр сдвига sr_dat подается RX_bit и сдвигаются в сторону старших разрядов по сигналу tce_tact (tce <= ce_tact):

sr_dat <= ok_SY? 0 : (tce_tact & en_rx)? sr_dat<<1 | RX_bit : sr_dat ;

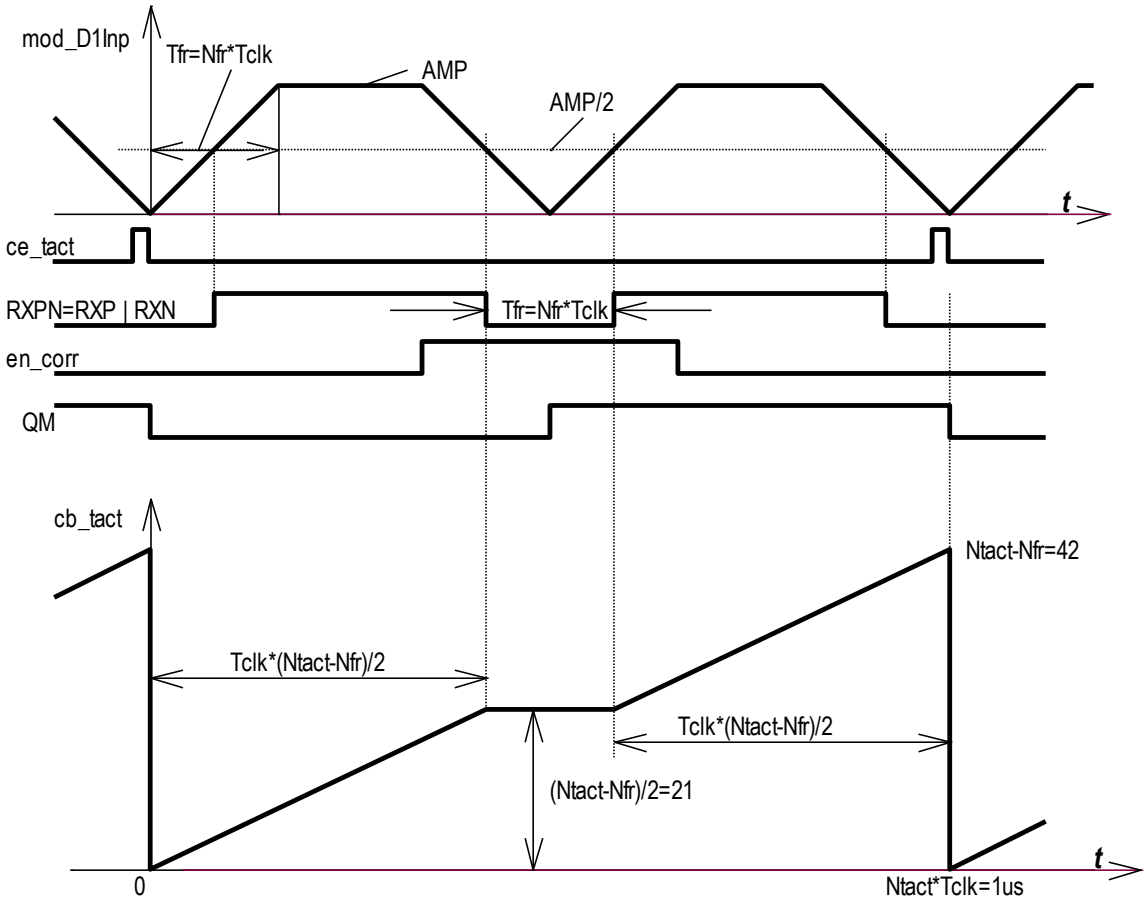

Рис.5 Временные диаграммы коррекции счетчика cb_tact модуля MIL_DET_DAT

1.5 Приемник сигналов слов интерфейса MIL-1553

```
module MIL_RX_ADC (output wire [7:0] DInp,    //Задержанный на 1.5us сигнал
      input clk,          output wire [7:0] SH,     // Смещение
      input [7:0]Inp,     output wire [7:0] AMP,    //Амплитуда
      input R,            output wire RXP,          //"Положительные" импульсы
                          output wire RXN,          //"Отрицательные" импульсы
                          output wire ok_SY,        //Обнаружен сигнал синхронизации CW или DW
                          output wire en_rx,        //Интервал приема слова
                          output wire [5:0] cb_tact,//Счетчик такта приемника
                          output wire ce_tact,      //Границы тактов приемника
                          output wire [4:0]cb_bit,  //Счетчик бит приемника
                          output wire QM,           //Имитатор RXN нулевых бит

                          output wire X,            //QM^RXP
                          output wire [7:0]cb_X,    //Счетчик длительности X
                          output wire RX_bit,       //Принятый бит
                          output wire FT_cp,        //Триггер контроля четности
                          output wire [15:0] sr_dat,//Приемный регистр сдвига данных
                          output wire ok_rx,        //Есть прием данных
                          output wire en_corr,      //Интервал коррекции

                          output wire T_cp,         //Интервал контрольного бита
                          output wire res,          //Сброс пиковых детекторов
                          output reg [5:0] cb_DW=0,//Счетчик слов
```

```
                        output reg [15:0] bf_CW=0,//Буфер контрольного слова
                        output reg [15:0] bf_DW=0 //Буфер слова данных
                        );
//Детектор сигнала синхронизации, измеритель смещения и амплитуды
MIL_DET_SY DD1 ( .D1Inp(DInp),  // Задержанный на 1.5us сигнал
            .Inp(Inp),    .SH(SH),        // Смещение
            .clk(clk),    .AMP(AMP),   //   Амплитуда
            .res(res),    .RXP1(RXP),   //"Положительные" импульсы
                          .RXN1(RXN),  //"Отрицательные" импульсы
                          .ok_SY(ok_SY));

//Созданный самостоятельно детектор бит и приемник данных
MIL_DET_DAT DD2 (       .en_rx(en_rx),       //Интервал приема слова
            .clk(clk),          .cb_tact(cb_tact),   //Счетчик такта приемника
            .ok_SY(ok_SY),      .ce_tact(ce_tact),   //Границы тактов приемника
            .RXP(RXP),          .cb_bit(cb_bit),     //Счетчик бит приемника
            .RXN(RXN),          .QM(QM),             //Имитатор RXN нулевых бит
                                .X(X),               //QM^RXP
                                .cb_X(cb_X),         //Счетчик длительности X
                                .RX_bit(RX_bit),  //Принятый бит
                                .sr_dat(sr_dat),     //Приемный регистр сдвига данных
                                .FT_cp(FT_cp),    //Триггер контроля четности
                                .ok_rx(ok_rx),       //Есть прием данных
                                .en_corr(en_corr));//Интервал коррекции
//-------------------------------------------------------------------------------
assign T_cp = (cb_bit==16);
reg [7:0]cb_res=0;
assign res = (cb_res==3) & ce_tact ;
always @ (posedge clk)  begin
cb_res <= ((RXP | RXN) & en_rx)? 0 : (!en_rx & ce_tact)? cb_res+1 : cb_res ;
cb_DW <= res? 0 : ok_rx? cb_DW+1 : cb_DW ;
bf_CW <= R? 0 : ( (cb_DW==0) & ok_rx)? sr_dat : bf_CW ;
bf_DW <= R? 0 : (!(cb_DW==0) & ok_rx)? sr_dat : bf_DW ;
end
endmodule
```

В состав этого модуля входит модуль MIL_DET_DAT детектора бит данных слов CW и DW, схема которого должна быть составлена самостоятельно.

Для автоматического сброса пиковых детекторов модуля MIL_DET_SY используется сигнал res, который вырабатывается в паузах между словами.

Счетчик cb_DW количества принятых слов сбрасывается сигналом res, а инкрементируется сигналом ok_rx.

Регистры буферов bf_CW и bf_DW принятых слов CW и DW загружаются по сигналу ok_rx, а сбрасываются внешним сигналом R.

2. Задание к допуску (стоимость 2)

2.1 Получить от преподавателя номер набора параметров (из таблицы 1), в который входят: контрольное слово CW и слово данных DW .

Таблица 1

№	CW (HEX)	DW (HEX)
1	1234	5678
2	5678	789A

3	9ABC	6523
4	DEF0	2233
5	FEDC	55AA
6	BA98	8811
7	7654	1188
8	3210	6699
9	1122	7711
10	3344	BCDE
11	5566	C3A5
12	7788	A587
13	6699	2D0F
14	AA55	E178
15	CC33	3C5A
16	00FF	4D20
17	FF00	55AA
18	F0F0	CC33
19	0FF0	4DD4
20	F00F	8181

2.2 Начертить в тетради временные диаграммы сигналов, заданных CW и DW (см. рис.1 и таблицу 1).

2.3 Начертить в тетради временные диаграммы модуля MIL_DET_SY детектора сигнала синхронизации (рис.3).

2.4 Начертить в тетради временные диаграммы модуля MIL_DET_DAT детектора данных (рис.4).

2.5 Начертить в тетради временные диаграммы коррекции счетчика cb_tact модуля MIL_DET_DAT (рис.5).

3. Задание к выполнению (стоимость 5)

Создать проект с именем Lab406_ADC, для ПЛИС XC3S500E-4FG320, используемой в макете NEXYS2. В окне Properties проекта в строке Simulator - выбрать ISim в дальнейшем моделирование проводить в Simulate Behavioral Model.

3.1. Создать Verilog модули Gen_SY_SD и MIL_DET_SY. Выполнить синтез (Synthesize-XST), исправить ошибки (Errors), обратить внимание на предупреждения (Warnings). Составить схему и модуль Test_MIL_DET_SY (см. приложение 6.1). Создать (Verilog Test Fixture) tf_Test_MIL_DET_SY задание на моделирование этой схемы. Провести моделирование при SY_SD=1 и различных значениях D и M. Сопоставить результаты моделирования с временными диаграммами рис.3.

3.2 Создать Verilog Module **MIL_TX_DAC.** Выполнить синтез. Создать tf_MIL_TX_DAC задание на моделирования этого модуля. Для своего варианта слов CW и DW провести моделирование. Сопоставить результаты моделирования с временными диаграммами рис.1.

3.3 Составить схему модуля **MIL_DET_DAT** детектора данных слов CW и DW. Создать модуль **MIL_DET_DAT,** выполнить синтез, исправить ошибки, проверить важность предупреждений.

3.4 Создать модуль **MIL_RX_ADC** приемника слов CW и DW. Выполнить синтез, исправить ошибки. Для моделирования **MIL_RX_ADC** составить схему и модуль Test_MIL_RX_ADC (см. приложение 6.2).

Провести моделирование при заданных значениях CW и DW для трех вариантов периода сигнала clk_rx синхронизации приемника (см. приложение 6.3).

Определить диапазон допустимой относительной разности периодов сигналов синхронизации clk_tx и clk_rx модулей передатчика MIL_TX_DAC и приемника MIL_RX_ADC.

Для сдачи лабораторной работы кроме NEXYS2, необходимы: макет цифроаналогового преобразователя (ЦАП) DAC_R2R и макет аналого-цифрового преобразователя (АЦП) PADC, которые соединяются с NEXYS2 в соответствии с рис.6.

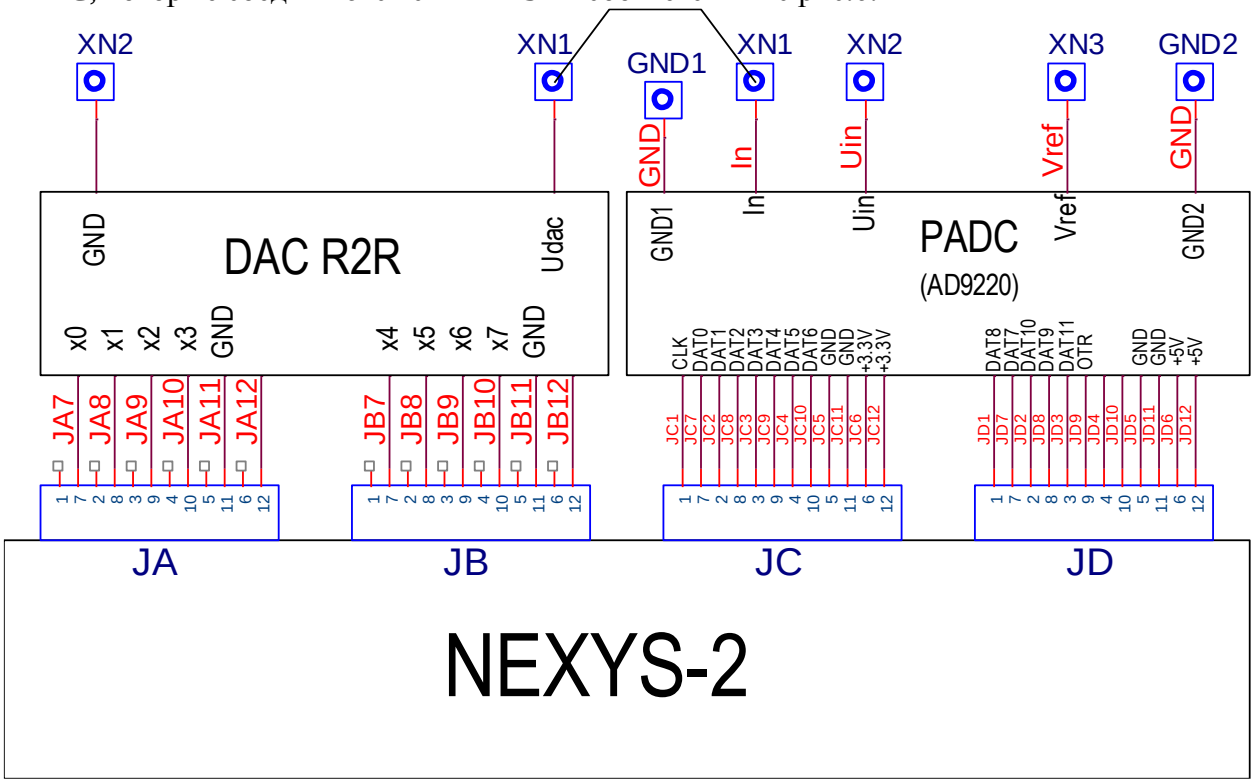

Рис.6 Схема соединения ЦАП DAC_R2R и АЦП с NEXYS2

Схемы и описание макетов ЦАП DAC_R2R и АЦП PADC приведены в приложениях 6.3 и 6.4.

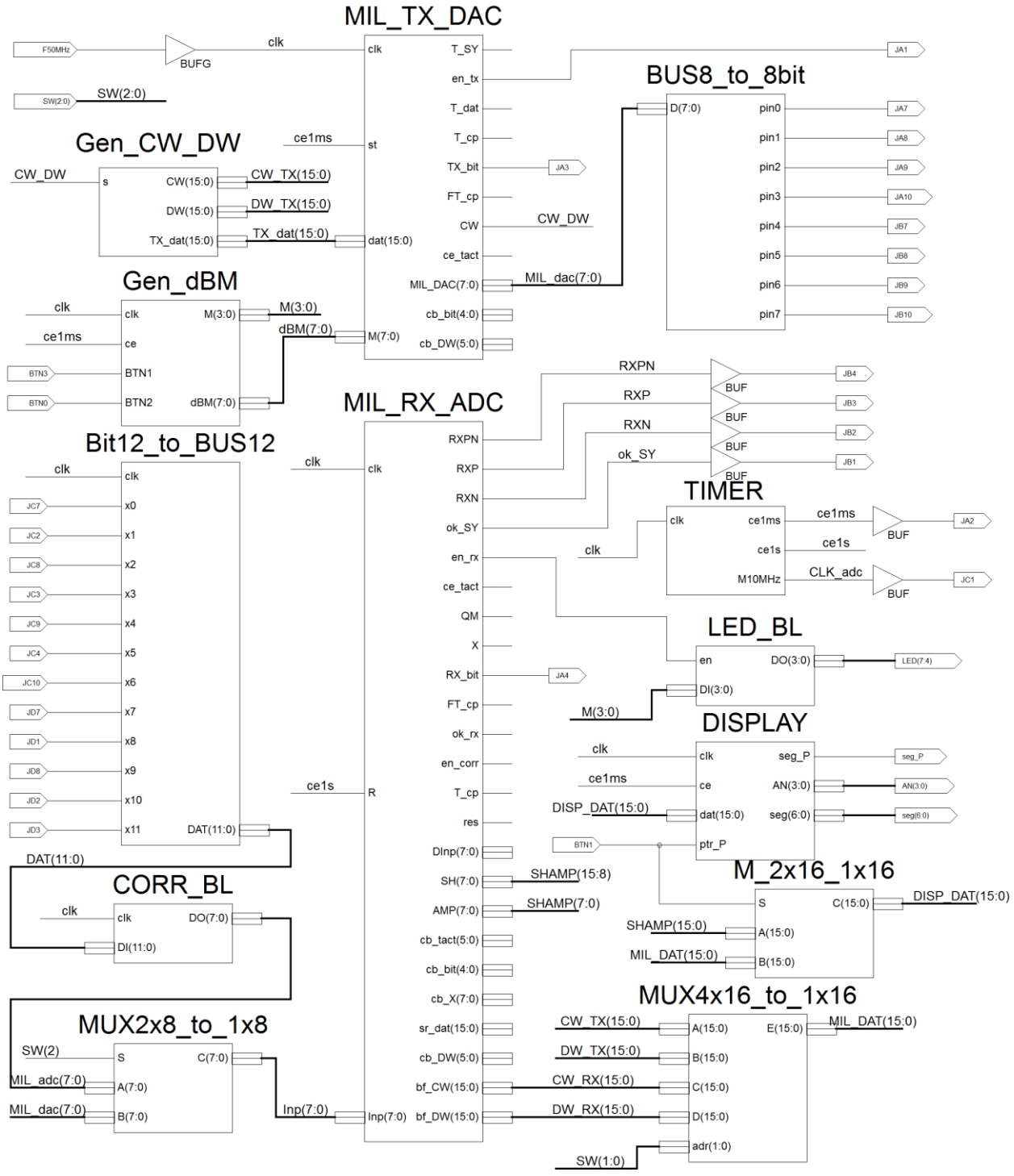

Рис.7 Пример схемы S_Sch_Lab406_ADC для сдачи работы

В этой схеме модуль **Gen_CW_DW** (приложение 6.5) должен выдавать на выходе TX_dat[15:0] при s=1 CW, а при s=0 – DW. Два выхода CW[15:0] и DW[15:0] предназначены для индикации передаваемых слов.

Модуль **Gen_dBM** (приложение 6.6) 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).

Модуль **LED_BL** (приложение 6.7) предназначен для индикации сигнала en_rx приема слов и M номера множителя dBM. Этот модуль вынуждено использует только 4 светодиода LED[7:0] 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.

Модуль **TIMER** (приложение 6.8) кроме сигналов ce1ms и ce1s вырабатывает сигнал синхронизации CLK_adc с частотой 10MHz для макета PADC параллельного АЦП. Максимально допустимая частота преобразования (синхронизации) используемого в макете PADC АЦП AD9220 равна 10MHz.

Сигнал ce1ms используется для подавления “дребезга” кнопок и для запуска передатчика MIL_TX_DAC.

Сигнал ce1s с периодом 1 сек используется для периодического сброса буферов принятых слов.

Модуль **MUX2x8_to_1x8** (приложение 6.9) при SW[2]=0 подает на вход приемника данные MIL_dac передатчика, а при SW[2]=1 корректированные данные MIL_adc приемника.

Модуль **MUX4x16_to_1x16** (приложение 6.10) подает на вход модуля семи сегментного индикатора **DISPLAY** (приложение 6.11) 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’b11 – DW_RX[15:0] (принятое слово данных).

Модуль **M_2x16_to_1x16** (создать самостоятельно) дополнительно переключает входные данные индикатора. 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.

4. Задание к сдаче работы (стоимость 3+(2 за 4.7)

4.1 Создать символы модулей: MIL_TX_DAC, MIL_RX_ADC.

4.2 Создать остальные модули и символы, входящие в схему лабораторной работы (рис.7). Создать (New Sources, Schematic) лист схемы S_Sch_LAB406_ADC. Разместить на листе схемы созданные символы. Соединить выводы символов в соответствии со схемой.

4.4 Для составленной схемы Sch_LAB406_ADC создать Implementation Constraints File (см. Приложение 6.12).

4.5 Создать файл конфигурации Sch_LAB406_ADC.bit (Generate Programming File) или *.mcs (Configure Target Device), загрузить его в макет. Подключить к NEXYS2 макеты DAC_R2R и PADC (см. рис.6). Продемонстрировать при помощи осциллографа напряжение Udac сигнала на выходе макета ЦАП DAC_R2R (“ждущую” развертку осциллографа запускать сигналом T_SY=JA1). Проверить влияние множителя dBM на осциллограмму Udac. Сопоставить осциллограмму Udac с рис.1. Сохранить осциллограмму Udac при максимальном значении M.

4.6 Соединить выход ЦАП DAC_R2R передатчика со входом АЦП PADC приемника. Провести при помощи осциллографа наблюдение сигналов ok_SY (JB1), RXN (JB2), RXP (JB3) и RXPN (JB4) приемника. При необходимости отладить схему приемника.

Сопоставить показания индикатора передаваемых и принимаемых данных. Определить минимальные dBM для SW[2]=0 и SW[2]=1, при которых происходит безошибочный прием данных. После достижения правильной работы приемника сохранить осциллограммы его сигналов.

4.7 Выяснить причины не совпадения минимальных уровней цифрового (MIL_dac, SW[2]=0) и “аналогового” (MIL_adc, SW[2]=1) сигналов, при которых происходит безошибочный прием данных.

5. Контрольные вопросы

5.1 Оценить предельно допустимую относительную разность периодов эталонов времени передатчика и приемника без коррекции счетчика cb_tact декодера данных приемника.
5.2 Влияет ли небольшая не симметрия сигналов RXP и RXN на качество работы дешифратора?
5.3 Влияет ли наличие промежутков между соответствующими фронтами и спадами сигналов RXP и RXN на качество работы приемника?
5.4 Можно ли в принципе без линии задержки декодировать сигнал синхронизации слов MIL-1553?
5.5 Как определить, чему соответствуют принятые данные: контрольному слову (CW) или слову данных (DW)?

6. Приложения

6.1 Схема тестирования детектора сигнала синхронизации

```
module Test_MIL_DET_SY(
module Test_MIL_DET_SY(
        input clk,          output wire [7:0] SMTRP_SH,//Смещенный импульс
        input st,           output wire T_TRP,          //Интервал импульса
        input [7:0] M,
        input D,
        input SY_SD,
                            output wire [7:0] D1Inp,   //Задержанный на 1.5us сигнал Inp
        input res,          output wire [7:0] D2Inp,   //Задержанный на 3.0us сигнал Inp
                            output wire [7:0] SH,      //Измеренное смещение
                            output wire [7:0] AMP,     //Измеренная амплитуда
                            output wire RXP1,  //D1Inp>REF_P
                            output wire RXN1,  //D1Inp<REF_N
                            output wire RXP2,  //D2Inp>REF_P
                            output wire RXN2,  //D2Inp<REF_N
                            output wire RXPN,
                            output wire SY,   //Интервал совпадения сигналов компараторов
                            output wire [7:0] cb_SY,
                            output wire ok_SY);
// Генератор пары биполярных трапецеидальных импульсов
Gen_SY_SD DD1 (
            .clk(clk),     .SMTRP_SH(SMTRP_SH),
            .st(st),       .T_TRP(T_TRP),
            .M(M),
            .D(D),
```

```
            .SY_SD(SY_SD));
// Детектор сигнала синхронизации
MIL_DET_SY DD2 (
        .Inp(SMTRP_SH),   .D1Inp(D1Inp),
        .clk(clk),        .D2Inp(D2Inp),
        .res(res),        .SH(SH),
                          .AMP(AMP),
                          .RXP1(RXP1),
                          .RXN1(RXN1),
                          .RXP2(RXP2),
                          .RXN2(RXN2),
                          .SY(SY),
                          .RXPN(RXPN),
                          .cb_SY(cb_SY),
                          .ok_SY(ok_SY));
endmodule
```

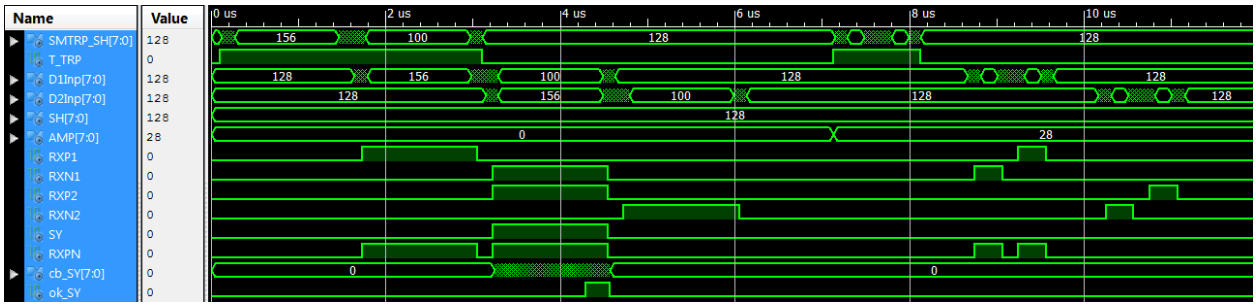

Рис.7 Пример временных диаграмм моделирования модуля детектора сигнала синхронизации

6.2 Схема тестирования приемника сигналов слов

```
module Test_MIL_RX_ADC(  output wire [7:0]MIL_DAC,
        input clk_tx,            output wire T_SY, // Интервал синхронизации
        input[15:0]TX_dat,       output wire en_tx, //Разрешение передачи
        input st,          output wire TX_bit, //Последовательные данные передатчика
        input [7:0]M,      output wire [4:0]cb_bit_tx,//Счетчик бит слов передатчика
        input R,
        input clk_rx,
                           output wire [7:0] DInp,
                           output wire [7:0] SH,
                           output wire [7:0] AMP,
                           output wire RXP,
                           output wire RXN,
                           output wire ok_SY,
                           output wire [4:0] cb_bit_rx,
                           output wire [5:0] cb_tact,
                           output wire ce_tact,
                           output wire en_rx,
                           output wire QM,
                           output wire RXPN,
                           output wire X,
                           output wire en_corr,
                           output wire [7:0]cb_X,
```

```
                        output wire RX_bit,
                        output wire FT_cp,
                        output wire T_cp_rx,
                        output wire [15:0] sr_dat,
                        output wire ok_rx,
                        output wire res);

assign RXPN = RXP | RXN ;
//-Передатчик сигналов слов CW и DW
MIL_TX_DAC DD1 (      .MIL_DAC(MIL_DAC),
        .clk(clk_tx),    .T_SY(T_SY),        // Интервал синхронизации
        .dat(TX_dat),  .en_tx(en_tx),        //Разрешение передачи
        .st(st),          .TX_bit(TX_bit),    //Последовательные данные
        .M(M),           .cb_bit(cb_bit_tx) ); //Счетчик бит слова

//-Приемник сигналов слов CW и DW
MIL_RX_ADC DD2 (      .DInp(DInp),
        .clk(clk_rx),          .SH(SH),
        .Inp(MIL_DAC),     .AMP(AMP),
        .R(R),                 .RXP(RXP),
                               .RXN(RXN),
                               .ok_SY(ok_SY),
                               .cb_bit(cb_bit_rx),
                               .cb_tact(cb_tact),
                               .ce_tact(ce_tact),
                               .en_rx(en_rx),
                               .QM(QM),
                               .X(X),
                               .en_corr(en_corr),
                               .cb_X(cb_X),
                               .RX_bit(RX_bit),
                               .FT_cp(FT_cp),
                               .T_cp(T_cp_rx),
                               .sr_dat(sr_dat),
                               .ok_rx(ok_rx),
                               .res(res) );
endmodule
```

6.2.1 Содержательная часть задания на моделирование схемы Test_MIL_RX_ADC

```
//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 = 21.0 ;  // Период clk_rx = 21.0
//parameter PRX = 19.0 ;  // Период clk_rx = 19.0
always begin clk_rx = 1'b0; #(PRX/2); clk_rx = 1'b1; #(PRX/2);  end
    initial begin
```

```
        st = 0;   M = 0;      TX_dat = 0;    R=0;
#4000;  st = 1;   M = 128;    TX_dat = 16'h9234; //my CW
#20;    st = 0;
#18000;           M = 128;    TX_dat = 16'hA678;//my DW
   end
```

6.3 Макет цифроаналогового преобразователя DAC_R2R

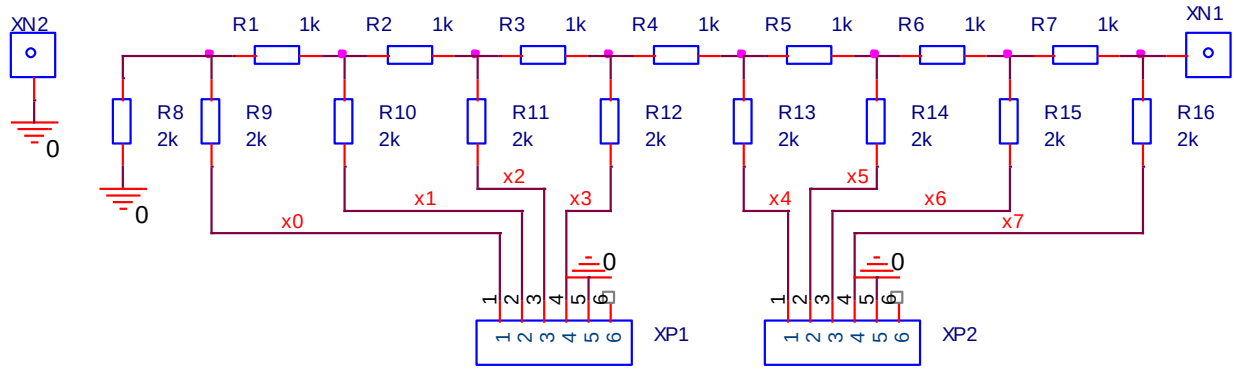

Рис.8 Схема цифроаналогового преобразователя DAC_R2R

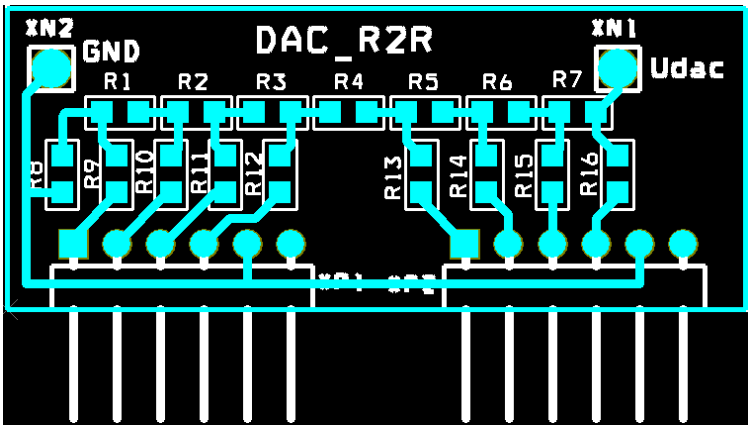

Рис.9 Лицевая сторона платы цифроаналогового преобразователя DAC_R-2R

Штыри печатной платы макета DAC_R2R предлагается вставлять в нижний ряд гнезд разъемов JA и JB макета NEXYS2 (JA7,JA8,JA9,JA10,JA11,JA12, JB7,JB8,JB9,JB10,JB11,JB12).

6.3.1 Модуль “расщепления” шины MIL_dac на отдельные входы ЦАП

```
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
```

6.4 Параллельный аналого-цифровой преобразователь

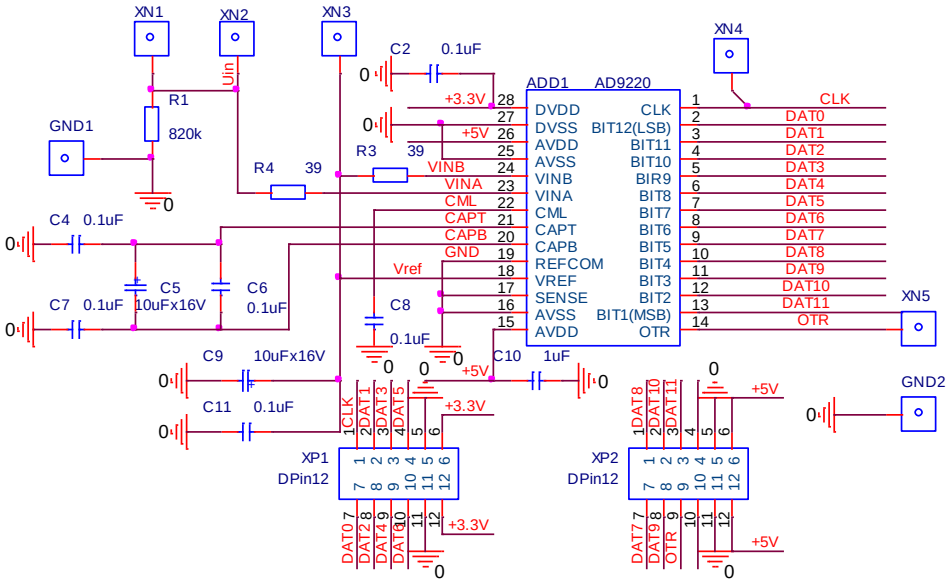

Рис. 10 Схема макета аналого-цифрового преобразователя PADC

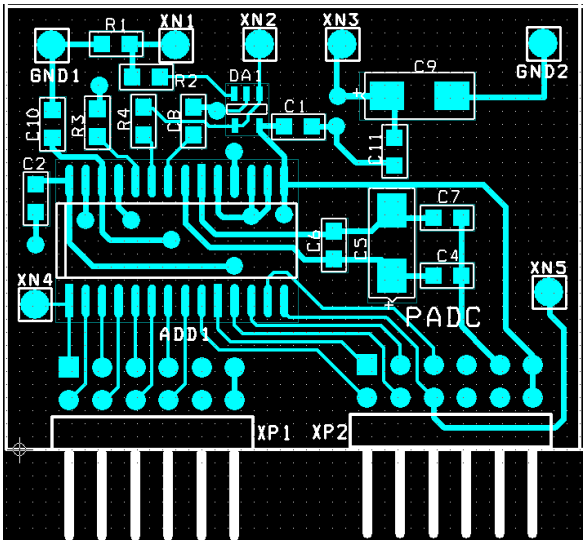

Рис.11 Лицевая сторона печатной платы параллельного аналого-цифрового преобразователя

Штыри печатной платы макета параллельного АЦП предлагается вставлять в гнезда разъемов JC и JD макета NEXYS2.

Напряжение питания +5V подается на XP2.6 платы макета AD9220 от JD12 при наличии перемычки между средним выводом и верхним выводом VSWT JP5 NEXYS2. JD5 и JD11 соединены с GND.

Напряжение питания +3.3V подается на XP1.6 платы макета AD9220 от JC12 при наличии перемычки между средним выводом и верхним выводом 3V3 JP3 NEXYS2.

6.4.1 Модуль “сборки” выходов АЦП в шину DAT

```
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
```

В этом АЦП значение выходной 12-и битной шины DAT[11:0] пропорционально входному напряжению Uin, DAT[11:0] =(Uin/Uref)*4096, где Uref=5V. Вход PADC должен соединяться с выходом DAC_R2R. Для согласования цифровых данных 11-и битной выходной шины DAT АЦП с 8-и битной шиной MIL_dac ЦАП DAC_R2R необходимо выполнить коррекцию. Например, напряжение Udac на выходе ЦАП DAC_R2R при MIL_dac=SH=128, Udac=(128/256)*Vcc. При Vcc=3.2V Udac==1.60V. А такому напряжению входа АЦП соответствует значение шины DAC=(1.60/5)*4096= 1310.72, поэтому для получения 8-ми битной шины MIL_adc[7:0] необходимо умножить DAT[11:0] на 128/1310.72=0.09765625= 6400/65536. Эта операция выполняется модулем CORR_BL.

6.4.2 Модуль коррекции данных АЦП

```
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=6400 ;// Mk=k*2^16=6206 при Vcc=3,3V
wire [27:0]MDI=DI*Mk ;
always @ (posedge clk) begin
DO <= MDI>>16 ;
end
```

6.5 Модуль генератора слов

```
`include "CONST.v"
module Gen_CW_DW(input s,  output wire [15:0]CW,
                           output wire [15:0]DW,
                           output wire [15:0]TX_dat);
assign CW=`my_CW ;//"Моё" контрольное слово
```

```
assign DW=`my_DW ;//"Моё" слово данных
assign TX_dat = s? CW : DW ;//Передаваемое слово

endmodule
```

6.6 Модуль генератора множителя dBM

```
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
```

6.7 Модуль **LED_BL** индикации приема и множителя M

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

6.8 Модуль TIMER

```
`include "CONST.v"
module TIMER(input clk,    output wire M10MHz,
                           output wire ce1ms,
                           output wire ce1s);
reg [2:0]cb_ce=0 ;
wire ce= (cb_ce==`Fclk/`F10MHz);//50MHz/10MHz=5
```

```
assign M10MHz = cb_ce[1] ;
reg [13:0] cb_1ms=0;
assign ce1ms = (cb_1ms==`F10MHz/`F1kHz) & ce ;//10MHz/1kHz=10000
reg [9:0]cb_1s=0 ;
assign ce1s = (cb_1s==`F1kHz/`F1Hz) & ce1ms ;//1kHz/1Hz=1000

always @ (posedge clk) begin
cb_ce <= ce? 1 : cb_ce+1 ;
cb_1ms <= ce1ms? 1 : ce? cb_1ms+1 : cb_1ms ;
cb_1s <= ce1s? 1 : ce1ms? cb_1s+1 : cb_1s ;
end
endmodule
```

6.9 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
```

6.10 Модуль мультиплексора **MUX4x16_to_1x16**

```
module MUX4x16_to_1x16(input [15:0] A,        output wire [15:0] E,
                       input [15:0] B,
                       input [15:0] C,
                       input [15:0] D,
                       input [1:0] adr );
assign E=   (adr==0)? A :
            (adr==1)? B :
            (adr==2)? C : D ;
endmodule
```

6.11 Модуль семи сегментного индикатора данных

```
module DISPLAY( input clk,          output wire[3:0] AN,    //Аноды
                input[15:0]dat,     output wire[6:0] seg,//Сегменты
                input ce,           output wire seg_P,   //Точка
                input ptr_P);
reg [1:0]cb_an=0 ;//Счетчик анодов
//-------Указатель точки-------
wire [1:0]PTR = ptr_P? 2'b10 : 2'b00 ;//XX.XX или XXXX.
assign seg_P = !(PTR == cb_an) ;

always @ (posedge clk) if (ce) begin
cb_an<= cb_an+1 ;
end

//-------Переключатель анодов-------------
assign  AN =   (cb_an==0)? 4'b1110 ://включение цифры 0 (младшей)
               (cb_an==1)? 4'b1101 ://включение цифры 1
               (cb_an==2)? 4'b1011 ://включение цифры 2
```

```
                                4'b0111 ;//включение цифры 3 (старшей)
//-------Переключатель тетрад (HEX цифр)-------------
wire[3:0] dig =(cb_an==0)? dat[3:0]:
                (cb_an==1)? dat[7:4]:
                (cb_an==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
endmodule
```

6.12 Распределение сигналов по контактным площадкам 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" ; # dec_M
NET "BTN1" LOC = "D18" ; #
#NET "BTN2" LOC = "E18" ; #
NET "BTN3" LOC = "H13" ; # inc_M

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" ; #TX_CW/TX_DW
NET "SW<1>" LOC = "H18" ; #RX_CW/RX_DW
NET "SW<2>" LOC = "K18" ; #DAC_dat/ADC_dat
```

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

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

NET "JA1" LOC = "L15" ; # en_tx
NET "JA2" LOC = "K12" ; # ce1ms
#NET "JA3" LOC = "L17" ; #
#NET "JA4" LOC = "M15" ; #
NET "JA7" LOC = "K13" ; # x0
NET "JA8" LOC = "L16" ; # x1
NET "JA9" LOC = "M14" ; # x2
NET "JA10" LOC = "M16"; # x3

NET "JB1" LOC = "M13" ; # ok_SY
NET "JB2" LOC = "R18" ; # RXN
NET "JB3" LOC = "R15" ; # RXP
NET "JB4" LOC = "T17" ; # RXPN
NET "JB7" LOC = "P17" ; # x4
NET "JB8" LOC = "R16" ; # x5
NET "JB9" LOC = "T18" ; # x6
NET "JB10" LOC = "U18"; # x7

NET "JC1" LOC = "G15" ; # CLK_adc
NET "JC2" LOC = "J16" ; # X1
NET "JC3" LOC = "G13" ; # X3
NET "JC4" LOC = "H16" ; # X5
NET "JC7" LOC = "H15" ; # X0
NET "JC8" LOC = "F14" ; # X2
NET "JC9" LOC = "G16" ; # X4
NET "JC10" LOC = "J12" ;# X6

NET "JD1" LOC = "J13" ;# X8
NET "JD2" LOC = "M18" ;# X10
NET "JD3" LOC = "N18" ;# X11
#NET "JD4" LOC = "P18" ;#
NET "JD7" LOC = "K14" ;   #LED<3>, X7
```

```
NET "JD8" LOC = "K15" ;   #LED<2>, X9
#NET "JD9" LOC = "J15" ; #LED<1>
#NET "JD10" LOC = "J14"; #LED<0>
```