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

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

Лабораторная работа №406AD

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 на линии может быть в интервале от 1.4 В до 20 В. Пример временных диаграмм слов приведен на рис. 1.

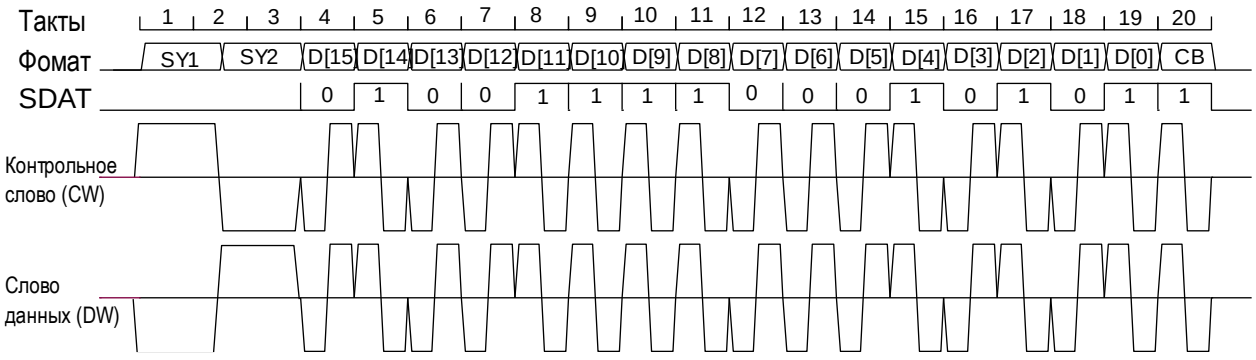

Рис.1 Пример временных диаграмм контрольного слова и слова данных стандарта MIL-STD-1553

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 в данной работе в системе проектирования цифровых устройств на ПЛИС моделируется работа модуля MIL_TXD логических сигналов TXP и TXN контрольного слова (рис.2) и слова данных (рис.3). TXP соответствует положительным импульсам, а TXN – отрицательным импульсам линии связи.

На втором этапе выполнения на основе предложенных алгоритмов, временных диаграмм и структурных схем дается задание на проектирование и отладку модуля MIL_RXD приемника сигналов TXP и TXN.

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

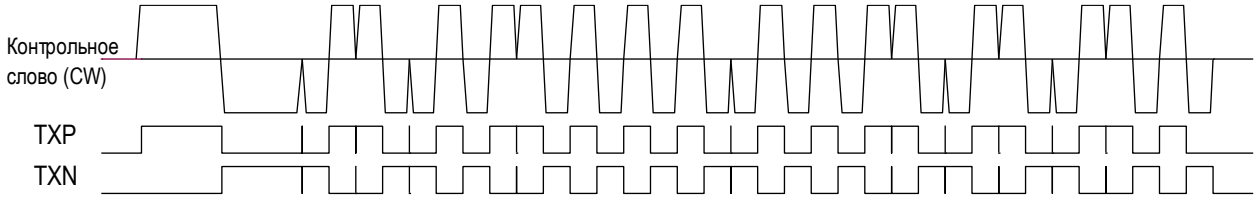

Рис. 2 Логические сигналы передатчика контрольного слова

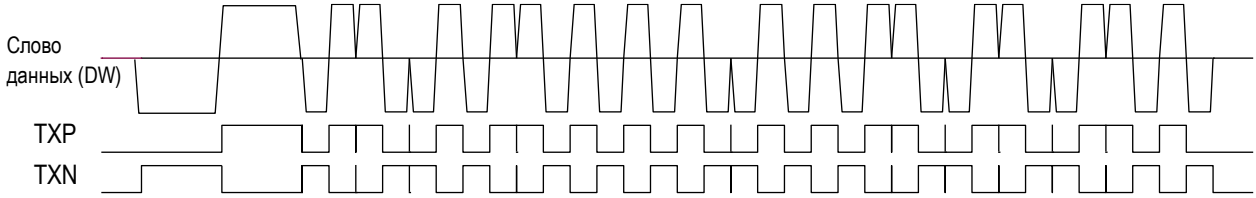

Рис. 3 Логические сигналы передатчика слова данных

1.1 Схема модуля передатчика MIL-1553

```
module MIL_TXD
        (  input clk,                output wire TXP, //”Положительные” импульсы
           input[15:0]dat,           output wire TXN, //”Отрицательные” импульсы
           input txen,               output wire SY1, // Первый импульс синхронизации
                                     output wire SY2, // Второй импульс синхронизации
                                     output reg en_tx=0, //Разрешение передачи
                                     output reg T_dat=0, //Интервал данных
                                     output wire T_end, //Такт конца слова
                                     output wire SDAT, //Последовательные данные
                                     output reg FT_cp=0, //Счетчик четности
                                     output reg [4:0]cb_bit =0, //Счетчик бит слова
                                     output wire ce_tact);

parameter TXvel = 1000000 ; // 1MHz
parameter Fclk  = 50000000 ; // 50 MHz
reg [5:0]cb_ce =0; //Счетчик полутакта
wire ce = (cb_ce == (Fclk /(2*TXvel)));
reg bf_SY1=0, bf_SY2=0 ; // Буферы сигналов синхронизации
```

```
reg CW_DW=0 ; /*Регистр режима: CW_DW=1 – контрольное слово, CW_DW=0 – слово данных*/
reg ttxen=0; //Задержаный на Tce_tact сигнал запуска
reg QM =0; //Модулятор
assign ce_tact = ce & QM ;              //Tce_tact=Tbit=1us
wire start = ttxen & !en_tx ;           // Импульс старта передачи
assign SY1 = en_tx & bf_SY1 ;           // Первый импульс синхронизации
assign SY2 = en_tx & bf_SY2 ;           // Второй импульс синхронизации
reg [15:0]sr_dat=0 ;                    // Регистр сдвига данных
wire st1 = (cb_bit==1) & en_tx ;        // Импульс старта 1-го импульса синхронизации
wire st2 = (cb_bit==2) & en_tx ;        // Импульс старта 2-го импульса синхронизации
wire st18 = (cb_bit==18) & en_tx;       //Конец интервала данных
assign T_end = (cb_bit==19) & en_tx; //Конец кадра, бит контроля четности
assign TXP = (en_tx & (( CW_DW & SY1) | //"Положительные" импульсы
                      (!CW_DW & SY2)              |
                       (T_dat & (sr_dat[15]^QM))   |
                       (T_end & (FT_cp^QM)))) ^ ((T_dat | T_end) & ce_tact) ;

assign TXN = (en_tx & ((!CW_DW & SY1) | //"Отрицательные" импульсы
                       (CW_DW & SY2)               |
                       (T_dat & (sr_dat[15]^!QM))   |
                       (T_end & (FT_cp^!QM)))) ^ ((T_dat | T_end) & ce) ;
assign SDAT = sr_dat[15] & T_dat ;          // Последовательные данные

always @ (posedge clk) begin
cb_ce <= ce? 1 : cb_ce+1 ;    //Tcet=Tbit/2=0.5us
QM <= (ce | start)? !QM : QM ; // Триггер меандра модулятора последовательных данных
end

always @ (posedge clk) if (ce) begin
bf_SY1 <= (st1 & !QM)? 0 : ((T_end & QM) | (txen & !en_tx))? 1 : bf_SY1 ;
bf_SY2 <= (st1 & !QM)? 1 : (st2 & QM)? 0 : bf_SY2 ;
end

always @ (posedge clk) if (ce_tact) begin
ttxen <= txen ; //Задержка на Tce_tact
en_tx <= ttxen? 1 : (!txen & T_end)? 0 : en_tx ;
cb_bit <= (!en_tx | T_end)? 0 : en_tx? cb_bit+1 : cb_bit ;
T_dat <= st2? 1 : st18? 0 : T_dat ;
sr_dat <= st2? dat : T_dat? sr_dat<<1 : sr_dat ;
FT_cp <= st2? 1 : (T_dat & sr_dat[15])? !FT_cp : FT_cp ;
CW_DW <= start? 1 : T_end? 0 : CW_DW ; //
end
endmodule
```

Триггер формата слова CW_DW, определяющий порядок импульсов синхронизации, по старту устанавливается в 1, что обеспечивает передачу первым

контрольного слова, а в конце слова сбрасывается в 0 и тем самым обеспечивает передачу всех остальных слов в формате слова данных.

Триггер en_tx по старту устанавливается в 1, а сбрасывается в 0 только в конце слова, если входной сигнал txen=0. Например, для передачи двух слов достаточно чтобы длительность импульса запуска txen была больше 20 мкс, но меньше 40 мкс.

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

1. Задание к допуску (стоимость 2)

1.1 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

1.2 Начертить в тетради временные диаграммы сигналов SY1, SY2, SDAT, FT_cp, TXP и TXN модуля MIL_TXD для заданных CW и DW (см. таблицу 1).

1.3 Переписать в тетрадь схему модуля MIL_TXD MIL-1553.

2. Задание к выполнению (стоимость 5)

Создать проект с именем Lab406AD, 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.

2.1. В окне источников (Sources) создать (New Source) модуль MIL_TXD генератора сигналов TXP и TXN стандарта MIL-1553 и далее на Verilog-е или в «схематике» составить схему этого модуля.

Создать для этого модуля задание на моделирование (Verilog Test Fixture). Период сигнала синхронизации clk в Verilog Test Fixture можно задать через параметр.

```
parameter PERIOD = 20;
always begin clk = 1'b0; #(PERIOD/2) clk = 1'b1;  #(PERIOD/2);  end
```

или непосредственно, через длительности 1'b0 и 1'b1 (в наносекундах):

```
always begin  clk = 1'b0; #10  clk = 1'b1; #10; end
```

В блоке “initial begin…end” вначале (#100) установить заданное значение контрольного слова и через ~10 мкс (#10000) слова данных. Суммарная длительность сигнала txen должна быть больше длительности контрольного слова. Например:

```
initial begin
                dat = 0;              txen = 0; //
#100;           dat = 16'h1234;       txen = 1; // my CW
#10000;         dat = 16'h5678;       txen = 1; // my WD
#24000;         dat = 16'h0000;       txen = 0;
   end
```

Установить заданные значения контрольного слова и слова данных.

2.2. Провести моделирование работы модуля MIL_TXD для заданных слов (таблица 1). При этом все регистры модулей должны быть инициализированы (например, приравнены к нулю). Подкорректировать, если необходимо, схему и временные диаграммы входных сигналов. Получить содержательные временные диаграммы. Проверить соответствие полученных временных диаграмм пункта 1.2 задания к допуску.

2.3 Используя приведенные ниже временные диаграммы и фрагменты схем составить схему модуля MIL_RXD приемника сигналов TXP и TXN с выходов модуля MIL_TXD.

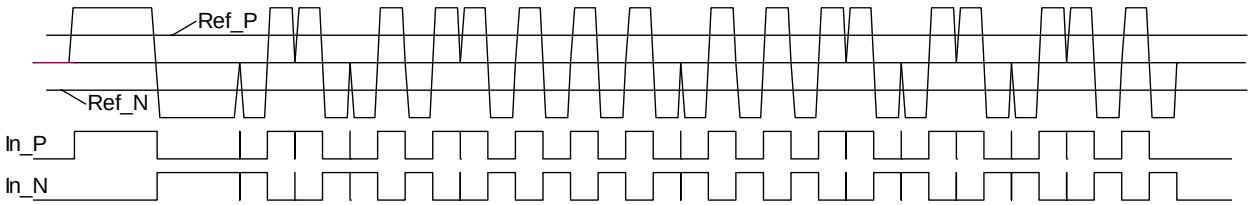

Рис. 4 Логические сигналы компараторов контрольного слова CW на входе приемника

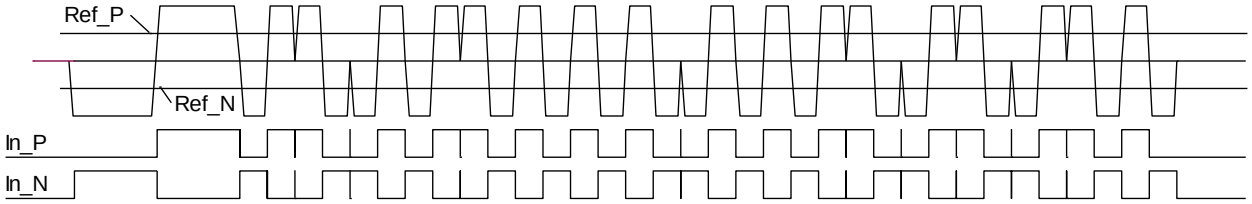

Рис. 5 Логические сигналы компараторов слова данных DW на входе приемника

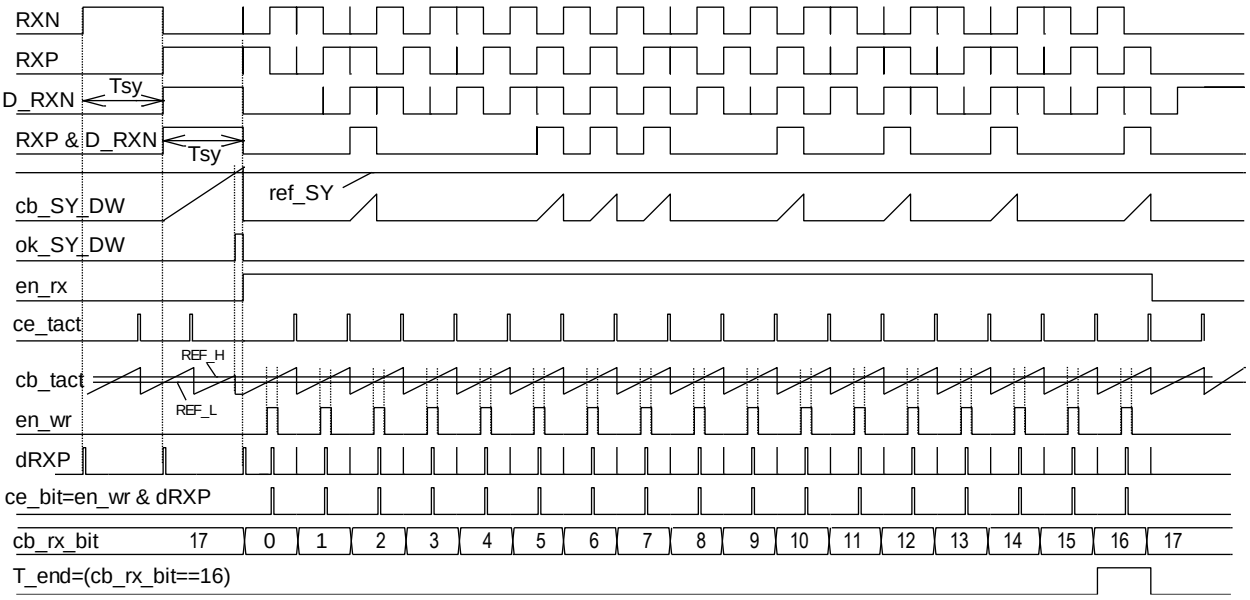

Рис.6 Пример сигналов приемника слова данных DW

RXP и RXN это задержанные на один такт Tclk сигналы компараторов In_P и In_N. Для обнаружения сигнала синхронизации слова данных предлагается сформировать сигнал D_RXN задержанный относительно RXN на Tsy=1.5 мкс. Импульсы логического произведения сигналов (RXP & D_RXN) только на интервале второго импульса синхронизации имеют длительность 1.5 мкс, остальные импульсы имеют длительность не более 0.5 мкс. Поэтому для обнаружения сигнала синхронизации достаточно при помощи счетчика cb_SY_DW измерять, например, в единицах Tclk, длительности этих импульсов, а в паузах между ними удерживать в 0. Максимальное значение числа на счетчике на интервале второго импульса синхронизации равно 75 (1500nc/20nc=75). Если это число, например, превышает ref_SY=70, то можно считать, что сигнал синхронизации слова данных ok_SY_DW обнаружен ok1_SY_DW=(cb_SY_DW>=ref_SY).

Для обнаружения же сигнала синхронизации контрольного слова предлагается сформировать D_RXP т.е. задержать на Tsy=1.5 мкс сигнал RXP, и аналогично при помощи другого счетчика cb_SY_CW и компаратора обнаруживать длинный импульс логического произведения сигналов (RXN & D_RXP) по превышению порога, ok_SY_CW=(cb_SY_CW>=ref_SY).

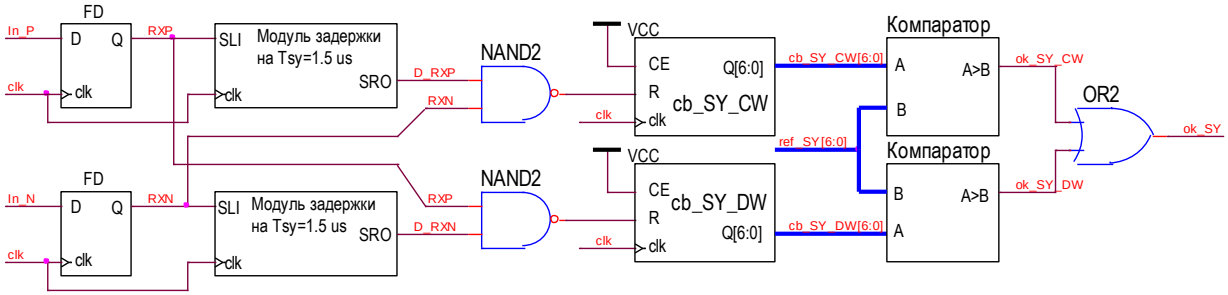

Рис. 7 Структурная схема обнаружителя сигнала синхронизации контрольного слова и слова данных

Для задержки сигналов RXN и RXP вполне допустимо использовать регистры сдвига sr_RXN[74:0] и sr_RXP[74:0] по 75 разрядов каждый (75*Tclk=75*20ns=1.5us).

Более экономично использовать модуль двухразрядной “слайсовой” памяти SMEM_2x128.

```
module SMEM_2x128(input [1:0] DI,        output wire[1:0] DO,
                  input [6:0] adr_wr,
                  input [6:0] adr_rd,
                  input clk,
                  input we);
reg [1:0]MEM [127:0] ;
assign DO=MEM[adr_rd] ;
initial  //Инициализация модуля памяти из файла SMEM_2x128.txt
$readmemh ("SMEM_2x128.txt", MEM, 0, 127);
always  @ (posedge clk) begin
MEM[adr_wr] <= we? DI : MEM[adr_wr] ;
end
endmodule
```

Адрес записи adr_wr[6:0] это семиразрядный счетчик тактов Tclk, а адрес чтения adr_rd = adr_wr - 75. При этом выходные данные DO будут задержаны относительно входных, DI на 75*Tclk=1.5us. Запись должна производиться непрерывно, поэтому разрешение записи we=1. На вход DI надо подавать {RXP,RXN}, а выход DO использовать как D_RXP=DO[1], D_RXN=DO[0]. Для моделирования желательно инициализировать память из текстового файла, который содержит 128 строк нулей.

Эталоны времени (периоды сигналов синхронизации clk_tx и clk_rx) передатчика и приемника не одинаковые. Для подстройки эталона времени приемника к эталону времени передатчика необходимо в центре каждого такта импульсами перепадов RXP или RXN проводить коррекцию счетчика такта приемника cb_tact т.е. записывать в него среднее значение set_M. A=25, а для исключения импульсов, соответствующих перепадам RXP и RXN между тактами это надо делать в выделенном интервале en_wr = (cb_tact>=ref_L) & (cb_tact<=ref_H) (см. рис.8).

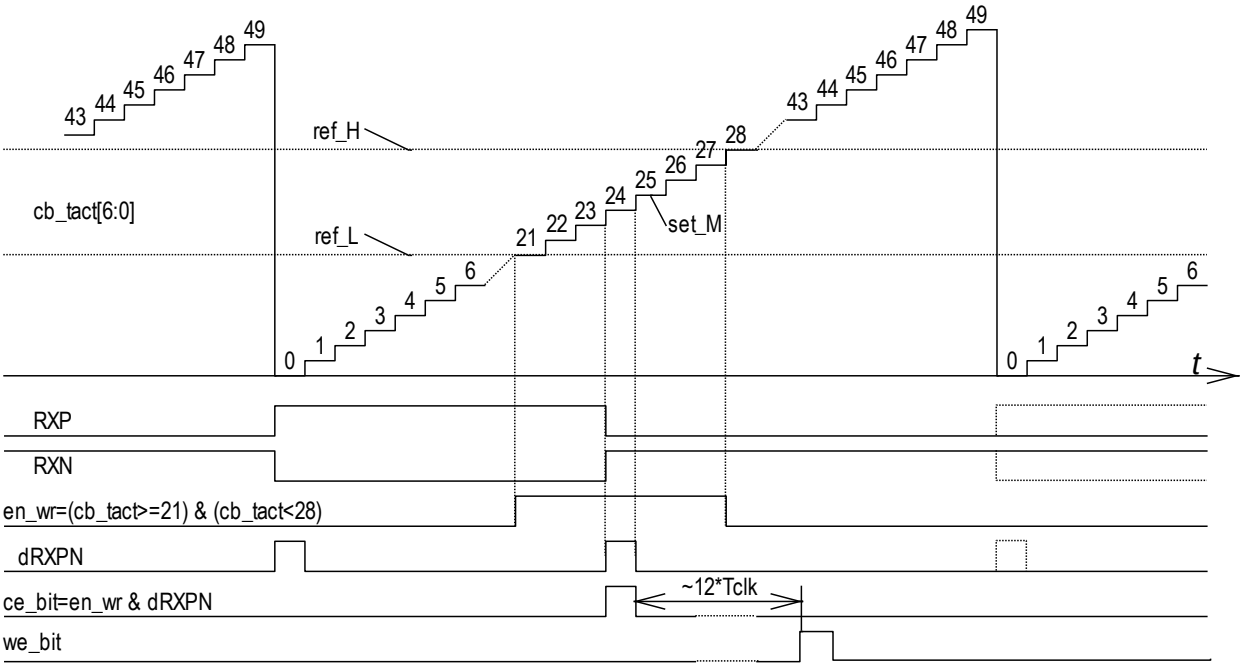

Рис. 8 Временные диаграммы синхронизации счетчика такта **cb_tact** импульсами перепадов RXP (или RXN) в центре такта

Для уменьшения влияния помех и параметров компараторов желательно считывать данные (RXN) не по сигналу ce_bit, а с задержкой на четверть такта (1мкс/4~=12*Tclk) по сигналу we_bit.

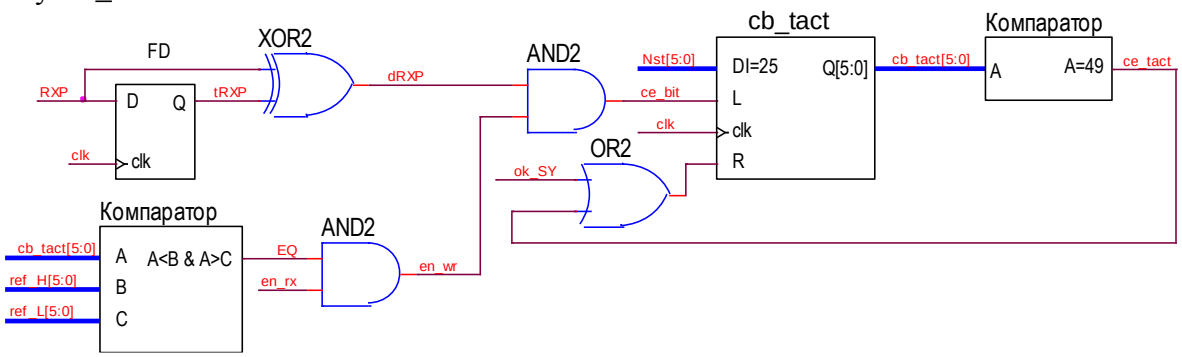

Рис. 8 Структурная схема синхронизации счетчика такта **cb_tact** импульсами перепадов RXP (или RXN) в центре такта

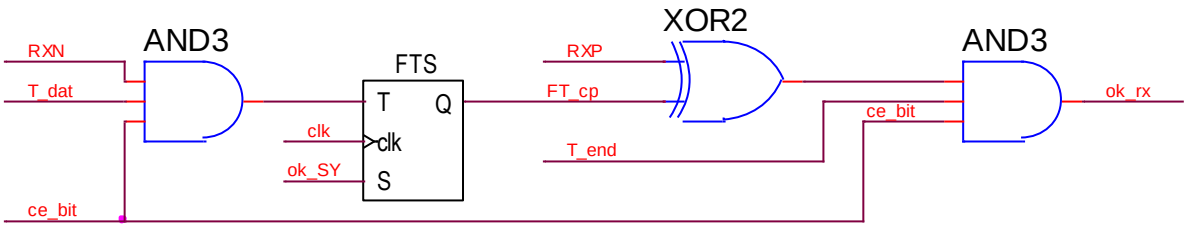

Рис. 9 Схема формирователя сигнала ok_rx

Для контроля правильности принятого слова однобитным счетчиком (Т триггером) FT_cp подсчитывается количество единиц на интервале данных T_dat. В начале каждого слова сигналом ok_SY триггер FT_cp устанавливается в 1 и на интервале 16 бит данных переключается столько раз, сколько единиц в слове, поэтому к последнему такту его состояние, как и в передатчике, дополняет количество единиц в полном слове до нечетного. На последнем такте T_end сравнивается принятый бит с FT_cp и если они совпадают, то формируется ok_rx.

2.4 Для отладки составленного модуля MIL_RXD необходимо составить схему, в которую входят только два модуля: MIL_TXD и MIL_RXD.

```
module Sch_test_MIL_RXD (
        //-----Вход и выходы MIL_TXD----------------------
        input clk_tx,           output wire TXP, //"Положительные" импульсы
        input[15:0]dat,         output wire TXN, //"Отрицательные" импульсы
        input txen,             output wire SY1, // Первый импульс синхронизации
                                output wire SY2, // Второй импульс синхронизации
        //-----Вход и выходы MIL_RXD----------------------
        input clk_rx,           output wire ok_SY, //Есть синхроимпульс
                                output wire dRXP, //Импульсы перепадов RXP
                                output wire[5:0] cb_tact, //Счетчик такта
                                output wire[4:0] cb_bit, //Счетчик бит
                                output wire ce_tact, //Границы такта
                                output wire en_rx, //Интервал приема слова
                                output wire en_wr, //Интервал разрешения коррекции
                                output wire ce_bit, //Центр такта
                                output wire T_dat, //Интервал данных
```

```
                                output wire T_end, //Интервал контрольного бита
                                output wire FT_cp, //Однобитный счетчик единиц
                                output wire [15:0]sr_dat , //Регистр сдвига данных
                                output wire ok_rx,  //Подтверждение верного приема
                                output wire CW_DW //Назначение принятых данных
                                );
//------Модуль передатчика MIL_TXD ------------------------
MIL_TXD DD1 (       .clk(clk_tx),    .TXP(TXP), //"Положительные" импульсы
                    .dat(dat),       .TXN(TXN), //"Отрицательные" импульсы
                    .txen(txen),     .SY1(SY1), // Первый импульс синхронизации
                                     .SY2(SY2)); // Второй импульс синхронизации
//------Модуль приемника MIL_RXD --------------------------
MIL_RXD DD2 (       .In_P(TXP),    .ok_SY(ok_SY),
                    .In_N(TXN),    .dRXP(dRXP),
                    .clk(clk_rx),  .cb_tact(cb_tact),
                                   .cb_bit(cb_bit),
                                   .ce_tact(ce_tact),
                                   .en_rx(en_rx),
                                   .en_wr(en_wr),
                                   .ce_bit(ce_bit),
                                   .T_dat(T_dat),
                                   .T_end(T_end),
                                   .FT_cp(FT_cp),
                                   .sr_dat(sr_dat),
                                   .ok_rx(ok_rx),
                                   .CW_DW(CW_DW) );
endmodule
```

Сигналы синхронизации clk_tx и clk_rx модулей передатчика и приемника должны быть разные.

```
// Сигнал синхронизации передатчика
always begin clk_tx = 1'b0; #10 clk_tx = 1'b1; #10;   end  //  PERIOD = 20
// Варианты  сигнала синхронизации приемника
always begin clk_rx = 1'b0; #10 clk_rx = 1'b1; #10;  end  //  PERIOD = 20
// always begin clk_rx = 1'b0; #10.3 clk_rx = 1'b1; #10.3;  end  //  PERIOD = 20.6 (+3%)
// always begin clk_rx = 1'b0; #9.7 clk_rx = 1'b1; #9.7;  end  //  PERIOD = 19.4 (-3%)
// Временные параметры и данные входных сигналов
initial begin               // Initialize Inputs
                txen = 0; dat = 0;//
#100;           txen = 1; dat = 16'h1234; // my CW (см. Таблицу1)
#10000;         txen = 1; dat = 16'h5678; // my DW (см. Таблицу1)
#24000;         txen = 0; dat = 16'h0000;
end
endmodule
```

2.5 Провести моделирование при заданных значениях CW и DW для трех вариантов сигнала clk_rx синхронизации приемника.

2.6 Определить диапазон допустимой относительной разности периодов сигналов синхронизации модулей MIL_TXD и MIL_RXD.

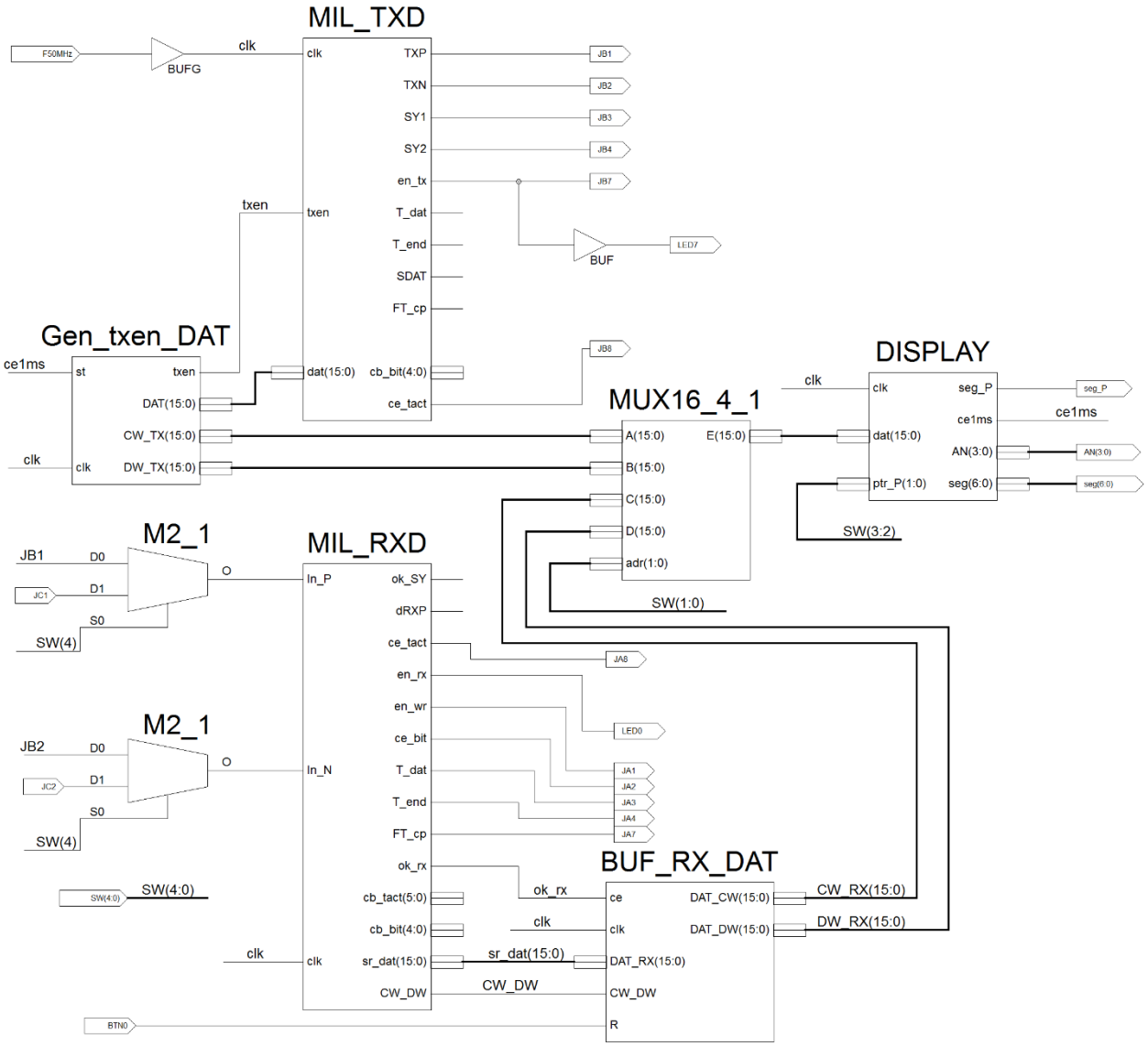

Рис.10 Пример схемы для сдачи работы

3. Задание к сдаче работы (стоимость 3)

3.1 Создать модули и символы: Gen_txen_DAT, BUF_RX_DAT, DISPLAY и MUX16_4_1 (см. Приложения 5.1-5.4).

3.2 Для загрузки в макет Nexys2 составить схему Sch_LAB406AD (см., например, рис.10). В состав схемы, кроме MIL_TXD и MIL_RXD должен входить и модуль отображения передаваемых и принимаемых данных (Display), модуль импульса запуска и заданных вариантов слов (Gen_txen_DAT), буфер принятых данных (BUF_RX_DAT) и мультиплексор данных для индикатора (MUX16_4_1).

3.3 Для схемы Sch_LAB406AD создать Implementation Constraints File (см. Приложение 5.5).

3.8 Создать файл конфигурации Sch_LAB406AD.bit (Generate Programming File) или *.mcs (Generate PROM, ACE, or JTAG File), загрузить его в макет. Продемонстрировать при помощи осциллографа работу передатчика. Проверить соответствие осциллограмм

сигналов sdat, TXP и TXN показаниям семи сегментного индикатора. Сохранить осциллограммы сигналов sdat, TXP и TXN.

3.9 Соединить выходы передатчика со входами приемника (можно проводные связи заменить внутренними т.е. SW[4]=0). Сопоставить показания индикатора передаваемых и принимаемых данных. Провести при помощи осциллографа наблюдение выведенных сигналов приемника. При необходимости отладить схему приемника. После достижения правильной работы приемника сохранить осциллограммы его сигналов.

4. Контрольные вопросы

4.1 Оценить предельно допустимую относительную разность периодов эталонов времени передатчика и приемника.
4.2 Влияет ли небольшая не симметрия сигналов RXP и RXN на качество работы дешифратора?
4.3 Влияет ли наличие промежутков между соответствующими фронтами и спадами сигналов RXP и RXN на качество работы приемника?
4.4 Можно ли в принципе без линии задержки декодировать сигнал синхронизации кадра MIL-1553?
4.5 Как определить, чему соответствуют принятые данные: контрольному слову (CW) или слову данных (DW)?

5. Приложения

5.1 Модуль импульса запуска, контрольного слова и слова данных

```
module Gen_txen_DAT(      output reg txen=0,
                  input st,       output wire[15:0] DAT,
                  input clk,      output wire[15:0] CW_TX,
                                  output wire[15:0] DW_TX);

assign CW_TX = 16'h1234 ; // my CW (Таблица 1)
assign DW_TX = 16'h5678 ; // my BW (Таблица 1)
assign DAT = txen? CW_TX : DW_TX ;
reg [10:0]cb_txen=0 ;
wire ce_end = (cb_txen==1100) ; //20ns*1100=22 000ns=22us>20us

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

5.2 Буфер принятых данных

```
module BUF_RX_DAT(
                  input ce,                    output reg[15:0] DAT_CW=0,
                  input clk,                   output reg[15:0] DAT_DW=0,
                  input [15:0] DAT_RX,
                  input CW_DW,
                  input R);
always @ (posedge clk or posedge R) begin
```

```
DAT_CW <= R? 0 : ( CW_DW & ce)? DAT_RX : DAT_CW ;
DAT_DW <= R? 0 : (!CW_DW & ce)? DAT_RX : DAT_DW ;
end
endmodule
```

5.3 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 [1:0]ptr_P,       output wire seg_P,    //Точка
                                          output reg ce1ms=0);
parameter Fclk=50000 ;       //50000 kHz
parameter F1kHz=1 ;          //1 kHz
reg [15:0] cb_1ms = 0 ;
wire ce = (cb_1ms==Fclk/F1kHz) ;
//-------Генератор сигнала ce1ms (период 1 мс, длительность Tclk=20 нс) --------
always @ (posedge clk) begin
cb_1ms <= ce? 1 : cb_1ms+1 ;
ce1ms <= ce ;
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
```

5.4 Мультиплексор данных

```
module MUX16_4_1(
        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
```

5.5 Распределение сигналов по контактным площадкам ПЛИС (файл *.ucf)

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

NET "AN<0>"  LOC = "F17"  ; #AN0
NET "AN<1>"  LOC = "H17"  ; #AN1
NET "AN<2>"  LOC = "C18"  ; #AN2
NET "AN<3>"  LOC = "F15"  ; #AN3

NET "seg<0>"  LOC = "L18"  ; #CA
NET "seg<1>"  LOC = "F18"  ; #CB
NET "seg<2>"  LOC = "D17"  ; #CC
NET "seg<3>"  LOC = "D16"  ; #CD
NET "seg<4>"  LOC = "G14"  ; #CE
NET "seg<5>"  LOC = "J17"  ; #CF
NET "seg<6>"  LOC = "H14"  ; #CG
NET "seg_P"  LOC = "C17"  ; #CP

NET "SW<0>"  LOC = "G18"  ; #adr[0]
NET "SW<1>"  LOC = "H18"  ; #adr[1]
NET "SW<2>"  LOC = "K18"  ; #PTR[0]
NET "SW<3>"  LOC = "K17"  ; #PTR[1]
NET "SW<4>"  LOC = "L14"  ; #S_M2_1
#NET "SW<5>"  LOC = "L13"  ; #
#NET "SW<6>"  LOC = "N17"  ; #
#NET "SW<7>"  LOC = "R17"  ; #

NET "BTN0"  LOC = "B18"  ; #RESET
#NET "BTN1"  LOC = "D18"  ; #
#NET "BTN2"  LOC = "E18"  ; #
#NET "BTN3"  LOC = "H13"  ; #
```

```
NET "LED0"  LOC = "J14"  ; #en_rx
#NET "LED<1>"  LOC = "J15"  ; #LD1
#NET "LED<2>"  LOC = "K15"  ; #LD2
#NET "LED<3>"  LOC = "K14"  ; #LD3
#NET "LED<4>"  LOC = "E17"  ; #LD4
#NET "LED<5>"  LOC = "P15"  ; #LD5
#NET "LED<6>"  LOC = "F4"  ; #LD6
NET "LED7"  LOC = "R4"  ; #en_tx

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

NET "JA1"  LOC = "L15"  ; #en_wr
NET "JA2"  LOC = "K12"  ; #ce_bit
NET "JA3"  LOC = "L17"  ; #T_dat
NET "JA4"  LOC = "M15" ; #T_end
NET "JA7" LOC = "K13" ; #FT_cp
NET "JA8" LOC = "L16"  ; #ce_tact
#NET "JA9"  LOC = "M14"  ;#
#NET "JA10" LOC = "M16"  ;#

NET "JB1"  LOC = "M13"  ; #TXP
NET "JB2"  LOC = "R18"  ; #TXN
NET "JB3"  LOC = "R15"  ; #SY1
NET "JB4"  LOC = "T17"  ; #SY2
NET "JB7"  LOC = "P17"  ; #en_tx
NET "JB8" LOC = "R16" ; #ce_tact
#NET "JB9"  LOC = "T18" ;#
#NET "JB10" LOC = "U18" ;#

NET "JC1"  LOC = "G15"  ; # to In_P
NET "JC2"  LOC = "J16"   ; # to In_N
#NET "JC3"  LOC = "G13"  ; #
#NET "JC4"  LOC = "H16"  ; #
#NET "JC7" LOC = "H15"  ; #
#NET "JC8" LOC = "F14"  ; #
#NET "JC9"  LOC = "G16"  ; #
#NET "JC10" LOC = "J12"  ; #

#NET "JD1"  LOC = "J13"  ;#
#NET "JD2"  LOC = "M18"  ;#
#NET "JD3"  LOC = "N18"  ;#
#NET "JD4"  LOC = "P18"  ;#
#NET "JD7"   LOC = "K14"  ; #LD3
#NET "JD8"   LOC = "K15"  ; #LD3
#NET "JD9"  LOC = "J15"  ; #LD3
#NET "JD10" LOC = "J14"  ; #LD3
```