Лабораторная работа 404_1В

1.1 Интерфейс RS-232 (COM-порт)

Интерфейс RS-232, официально называемый "EIA/TIA–232–E", но более известный как интерфейс "COM-порта", ранее был одним из самых распространенных интерфейсов в компьютерной технике. Он и до сих пор встречается в настольных компьютерах, несмотря на появление более скоростных и "интеллектуальных" интерфейсов, таких как USB и FireWare. К его достоинствам с точки зрения разработчиков аппаратуры можно отнести простоту реализации протокола. Физический интерфейс RS-232 реализуется разъемом типа DB-9M

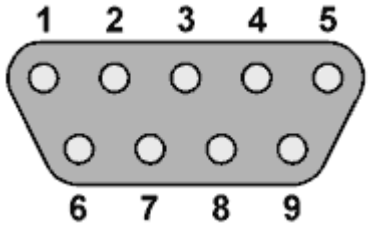


Рис.1 9-контактная вилка типа DB-9M COM порта компьютера

Таблица 1

№	Сигнал	Направление	Назначение
1	DCD	Вход	Обнаружена несущая
2	RXD	Вход	Принимаемые данные
3	TXD	Выход	Передаваемые данные
4	DTR	Выход	Компьютер готов
5	GND	-	Общий провод
6	DSR	Вход	Устройство готово
7	RTS	Выход	Компьютер готов к передаче
8	CTS	Вход	Устройство готово к приему
9	RI	Вход	Обнаружен вызов

1.2 Электрические характеристики

Физические уровни передатчика: "0" – от +5 до +15 В, "1" – от -5 до -15 В.

Физические уровни приемника: "0" – выше +3 В, "1" – ниже -3 В.

Максимальная нагрузка на передатчик: входное сопротивление приемника не менее 3 кОм.

Для преобразования физических сигналов TXD и RXD RS-232 COM-порта в логические уровни URXD и UTXD протокола UART и обратно используются специальные микросхемы. Например, ADM202, MAX3232 или ST3232.

В данной работе изучается наиболее распространенный, простейший асинхронный режим передачи и приема, в котором используются только два сигнала TXD и RXD.

Каждый, передаваемый блок данных, дополняется нулевым старт битом и заканчивается единичным стоп битом. Длительность старт бита равна длительности одного бита, а длительность стоп бита может составлять одну, полторы или две длительности бита. Данные передаются младшими битами вперед. Данные могут дополняться битом контроля четности. Количество бит данных может быть от 4 до 8.

Контроль четности может быть четным ("even", когда общее количество единичных битов в принятых данных, включая сам бит четности, должно быть четным), нечетным ("odd", когда общее количество единичных битов в принятых данных, включая сам бит четности, должно быть нечетным) или вообще отсутствовать.

Скорость обмена данными (BAUD RATE) задается в битах в секунду и выбирается из ряда стандартных значений (300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 430800, 921600), но может быть и нестандартной, если поддерживаются обеими сторонами.

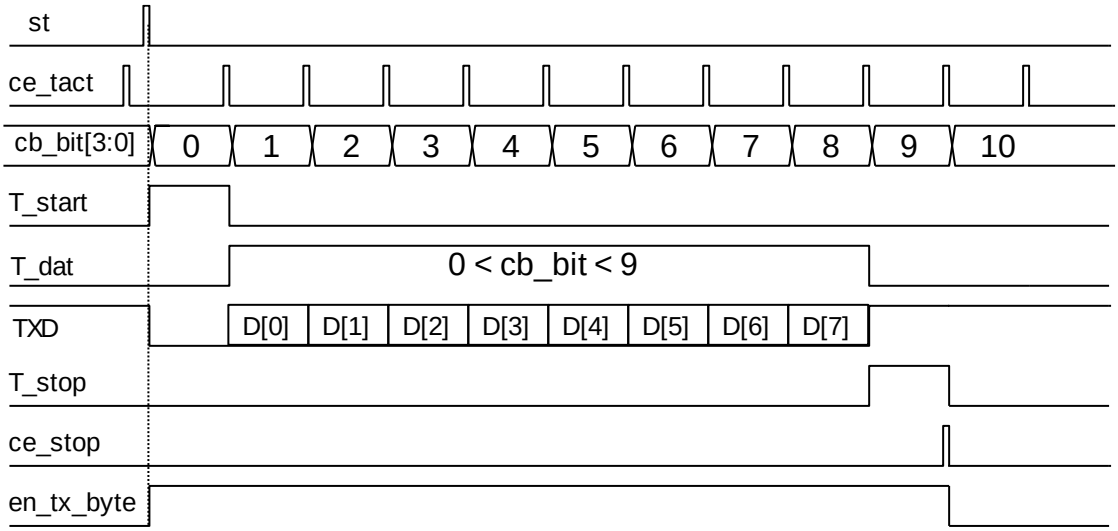


Рис.2 Пример временных диаграмм модуля UTXD1B, передатчика одного байта (8 бит данных без бита контроля четности)

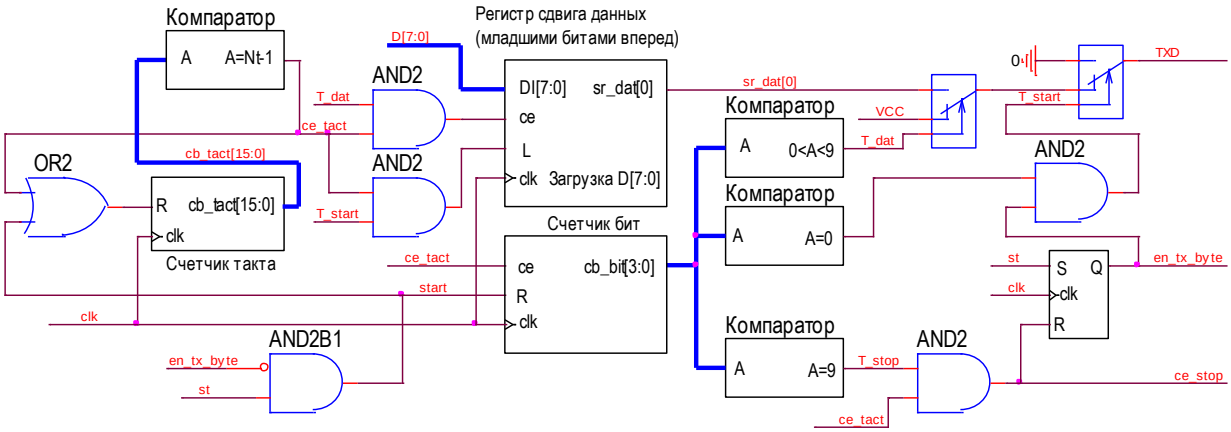


Рис.3 Структурная схема модуля UTXD1B, передатчика одного байта

В этой схеме всего 4 регистра:

cb_tact[15:0] - счетчика такта,

cb_bit[3:0] – счетчик бит,

en_tx_byte – RS-триггер интервала передачи байта,

sr_dat[7:0] – регистр сдвига передаваемых данных.

Число разрядов счетчика такта cb_tact должно соответствовать заданной скорости VEL (BAUD RATE). Например, при VEL=115200 бод (Nt=50000000/115200=434) счетчик

должен иметь 9 разрядов, а при скорости 300 бод -18 разрядов (Nt=50000000/300=166667). При моделировании, для наглядности можно увеличить VEL до 6500000 (Nt=8).

1.3 Схема модуля UTXD1B передатчика одного байта на языке VERILOG

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

parameter Fclk=50000000 ; //50 MHz
parameter VEL = 115200 ; //115.2 kBod (из таблицы 1 вариантов)
parameter Nt = Fclk/VEL ; //~434
reg [15:0]cb_tact=0 ;                          //Счетчик такта
assign ce_tact = (cb_tact==`Nt) ;              //Компаратор cb_tact с Nt
assign T_start = ((cb_bit==0) & en_tx) ;       // Компаратор старт такта
assign T_dat = (cb_bit<9) & (cb_bit>0);        // Компаратор интервала данных
assign T_stop = ((cb_bit==9) & en_tx);         // Компаратор стоп такта
assign ce_stop = T_stop & ce_tact ;            //Строб стопа
assign TXD = T_start? 0 : T_dat? sr_dat[0] : 1 ; //Последовательные данные sr_dat[0]
wire start = st & !en_tx_byte ;
//-------------
always @ (posedge clk) begin
cb_tact <= (start | ce_tact)? 1 : cb_tact+1 ; //"Сброс" в 1 при cb_tact==`Nt
en_tx_byte  <= st? 1 : ce_stop? 0 : en_tx_byte ;
cb_bit <= start? 0 : (ce_tact & en_tx)? cb_bit+1 : cb_bit ;
sr_dat <= (T_start & ce_tact)? dat : (T_dat & ce_tact)? sr_dat>>1 : sr_dat ; /* Сдвиг sr_dat
младшими битами вперед*/
end
endmodule
```

1.4 Алгоритм работы приемника:

Приемник байта можно строить по классическому алгоритму, как это делается, например, в UART приемниках микроконтроллеров, т.е. принимать решение о значении очередного бита в середине такта.

После первого спада сигнала RXD с задержкой в половину длительности бита проверить RXD, если RXD =0 то с задержкой в полную длительность бита т.е. в середине первого бита данных записать текущий уровень RXD в регистр младшего бита данных (D[0]). Повторить в серединах следующих битов считывание и запись состояния RXD для остальных бит данных (D[1],D[2,...D[7]]). После этого также в середине следующего (последнего стоп) бита проверить, что RXD=1. Если это так, то считать, что прием данных успешно завершен.

В качестве регистра данных удобно использовать регистр сдвига. Тогда принимаемые данные не надо записывать в соответствующие разряды регистра данных, а достаточно вдвигать младшими битами вперед в регистр сдвига. Первый бит сдвинется 8 раз и окажется в младшем разряде, а последний бит в старшем разряде регистра сдвига.

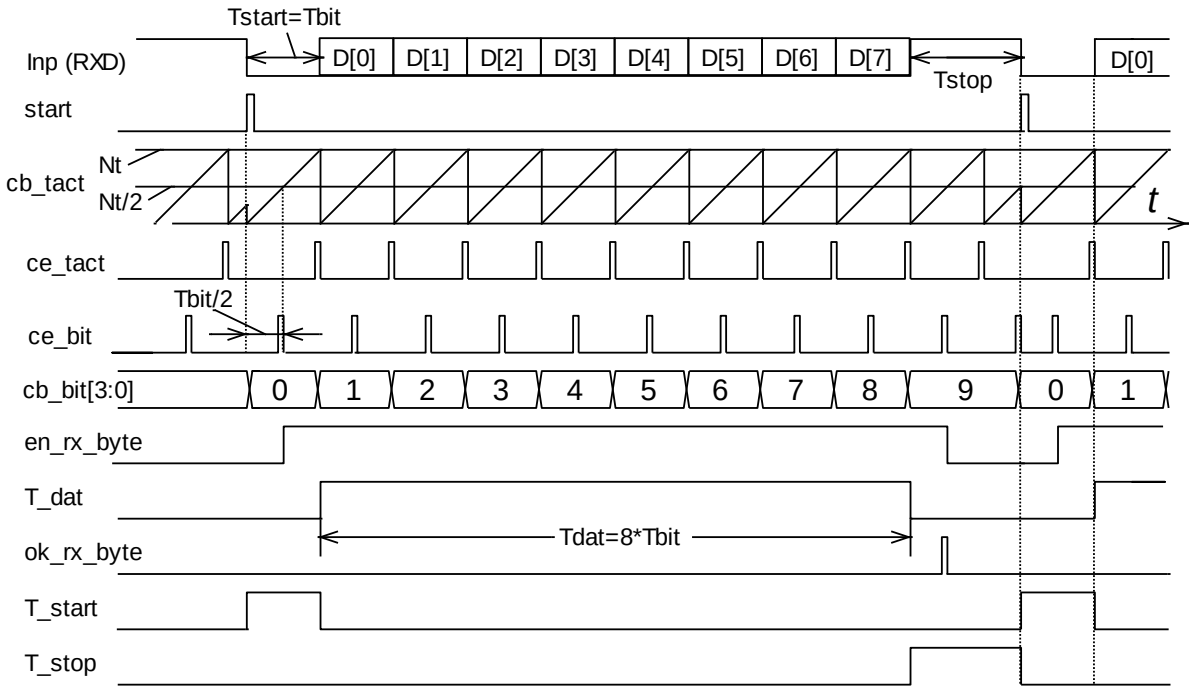


Рис.4 Пример временных диаграмм приемника одного байта (8 бит данных) URXD1B (без контроля четности)

В модуле URXD1B число разрядов счетчика такта cb_tact также, как и модуле передатчика UTXD1B должно быть достаточным для заданной скорости VEL.

Сигнал ce_tact соответствует границам бит, а сигнал ce_bit – серединам бит. Периоды сигналов ce_tact и ce_bit должны быть равны ожидаемой длительности бита принимаемых данных. Генератор сигналов ce_tact и ce_bit фазируется импульсом старта приема байта.

Импульс старта приема байта start формируется по спаду входного сигнала при условии en_rx_byte=0, т.е., что прием предыдущего байта уже закончен, start=dRXD & !en_rx_byte, где dRXD это импульсы соответствующие спадам входного сигнала RXD. Подобная операция делается и в передатчике одного байта (см. рис.3) для исключения сброса счетчиков при попытке повторного старта до окончания передачи байта.

Сигнал разрешения приема байта en_rx_byte устанавливается в 1 в середине такта T_start при условии, что RXD=0.

Решение о конце приема байта принимается на девятом такте по сигналу ce_bit. Данные D[0],…D[7] задвигаются в регистр сдвига sr_dat[7:0] младшими битами вперед по сигналу ce_bit на интервале данных T_dat.

Сигнал подтверждения правильности приема байта ok_rx_byte формируется на девятом такте T_stop при условии, что RXD=1.

Модуль приемника одного байта URXD1B необходимо спроектировать и отладить самостоятельно.

1.5 Схема моделирования (URXD1B) приёмника одного байта

```
module Sch_test_URXD1B( input tx_clk,          output wire TXD,
                        input st,              output wire[3:0]cb_bit_tx,
                        input[7:0]tx_dat,      output wire en_rx_byte,
                        input rx_clk,          output wire [7:0]sr_dat,
                                               output wire [3:0]cb_bit_rx,
                                               output wire ok_rx_byte,
```

```
                                output wire start_rx,
                                output wire T_start,
                                output wire T_dat,
                                output wire T_stop,
                                output wire ce_tact,
                                output wire ce_bit,
                                output wire RXD);

UTXD1B DD1 (        .clk(tx_clk),    .TXD(TXD),
                    .dat(tx_dat),    .cb_bit(cb_bit_tx),
                    .st(st)) ;

URXD1B DD2 (        .Inp(TXD),       .en_rx_byte(en_rx_byte),
                    .clk(rx_clk),    .sr_dat(sr_dat),
                                     .cb_bit(cb_bit_rx),
                                     .ok_rx_byte(ok_rx_byte),
                                     .start(start_rx),
                                     .T_start(T_start),
                                     .T_dat(T_dat),
                                     .T_stop(T_stop),
                                     .ce_tact(ce_tact),
                                     .ce_bit(ce_bit),
                                     .RXD(RXD));

endmodule
```

В этой схеме используются два модуля: модуль передатчика одного байта UTXD1B и модуль спроектированного приемника одного байта URXD1B. Сигналы синхронизации у модулей передатчика и приемника разные: tx_clk и rx_clk. Это необходимо для проверки устойчивости спроектированного модуля приемника к неодинаковости эталонов времени (периодов Tclk) передатчика и приемника. Выходной сигнал TXD передатчика UTXD1B подается на вход **Inp** приемника, а также для контроля и на выходной порт TXD схемы Sch_test_URXD1B. Выходной сигнал RXD приемника это задержанный D триггером на Trx_clk входной сигнал Inp.

Для моделирования отдельного модуля или всей схемы удобно создавать Verilog Test Fixture. При этом система проектирования автоматически создает текстовое описание всех портов модулей и схемы, а также связей между ними. Остается только дописать генераторы периодических сигналов синхронизации и параметры входных сигналов, т.е. заполнить блок initial begin …end.

1.6 К заданию на моделирование схемы Sch_test_URXD1B

```
always begin tx_clk = 1'b0; #10   tx_clk = 1'b1; #10;   end //        PERIOD = 20
always begin rx_clk = 1'b0; #10   rx_clk = 1'b1; #10;  end //        PERIOD = 20
//always begin rx_clk = 1'b0; #10.4 rx_clk = 1'b1; #10.4;  end //    PERIOD = 20.8
//always begin rx_clk = 1'b0; #9.7 rx_clk = 1'b1; #9.7;  end //      PERIOD = 19.4
initial begin
        st  = 0 ;        tx_dat = 0 ;
#1000;  st = 1;          tx_dat = 8'bXXXXXXXX;// my tx_dat (из таблицы 1)
#20;    st = 0;          tx_dat = 8'bXXXXXXXX;// my tx_dat (из таблицы 1)
end
```

В этом примере по старту моделирования tx_dat=0 и st=0. Далее через 1000ns: st=1, tx_dat= 8’bXXXXXXXX; (из таблицы 1), затем через 20ns st=0 (длительность импульса st 20ns).

В Process Properties логического симулятора Simulate Behavioral Model надо установить подходящее (Simulation Run Time) время моделирования, например, 15*Tbit (Tbit=1/VEL).

Таблица 1

№	VEL[bit/s]	tx_dat [bin]
1	115200	8’b10000001
2	57600	8’b10000010
3	19200	8’b10000011
4	38400	8’b10000100
5	921600	8’b10000101
6	115200	8’b10000110
7	460800	8’b10000111
8	19200	8’b10001000
9	115200	8’b10001001
10	57600	8’b10001010
11	115200	8’b10001011
12	460800	8’b10001100
13	57600	8’b10001101
14	115200	8’b10001110
15	19200	8’b10001111
16	230400	8’b10010000
17	19200	8’b10010001
18	57600	8’b10010010
19	9600	8’b10010011
20	115200	8’b10010100

2.Задание к допуску

2.1 Начертить в тетради временные диаграммы сигналов UTXD1B (рис.2).

2.2 Начертить в тетради схему модуля UTXD1B передатчика одного байта (рис.3).

2.3 Начертить в тетради временные диаграммы сигналов URXD1B (рис.4).

2.4 Составить и начертить (или написать на VRILOG-e) схему модуля URXD1B.

3. Задание к выполнению работы

Создать проект с именем Lab404 для ПЛИС, используемой в макете NEXYS или NEXYS-2.

3.1 Создать модуль UTXD1B передатчика одного байта. Провести моделирование его работы при VEL= 6250000. Для заданного значения dat[7:0] зарисовать временные диаграммы сигналов модуля.

3.2 Создать модуль URXD1B приемника одного байта.

3.3 Создать модуль схемы Sch_test_URXD1B.

3.4 Создать модуль tf_Sch_test_URXD1B (Verilog Test Fixture) для схемы Sch_test_URXD1B.

3.5 Провести моделирование модуля приемника одного байта URXD1B при различных значениях периода сигнала синхронизации rx_clk приемников. Определить допустимый диапазон периода rx_clk.

Зарисовать эскизы временных диаграмм сигналов: RXD, en_rx_byte, sr_dat, cb_bit_rx, ce_tact, ce_bit_rx, T_start, T_dat, T_stop, ok_rx_byte.

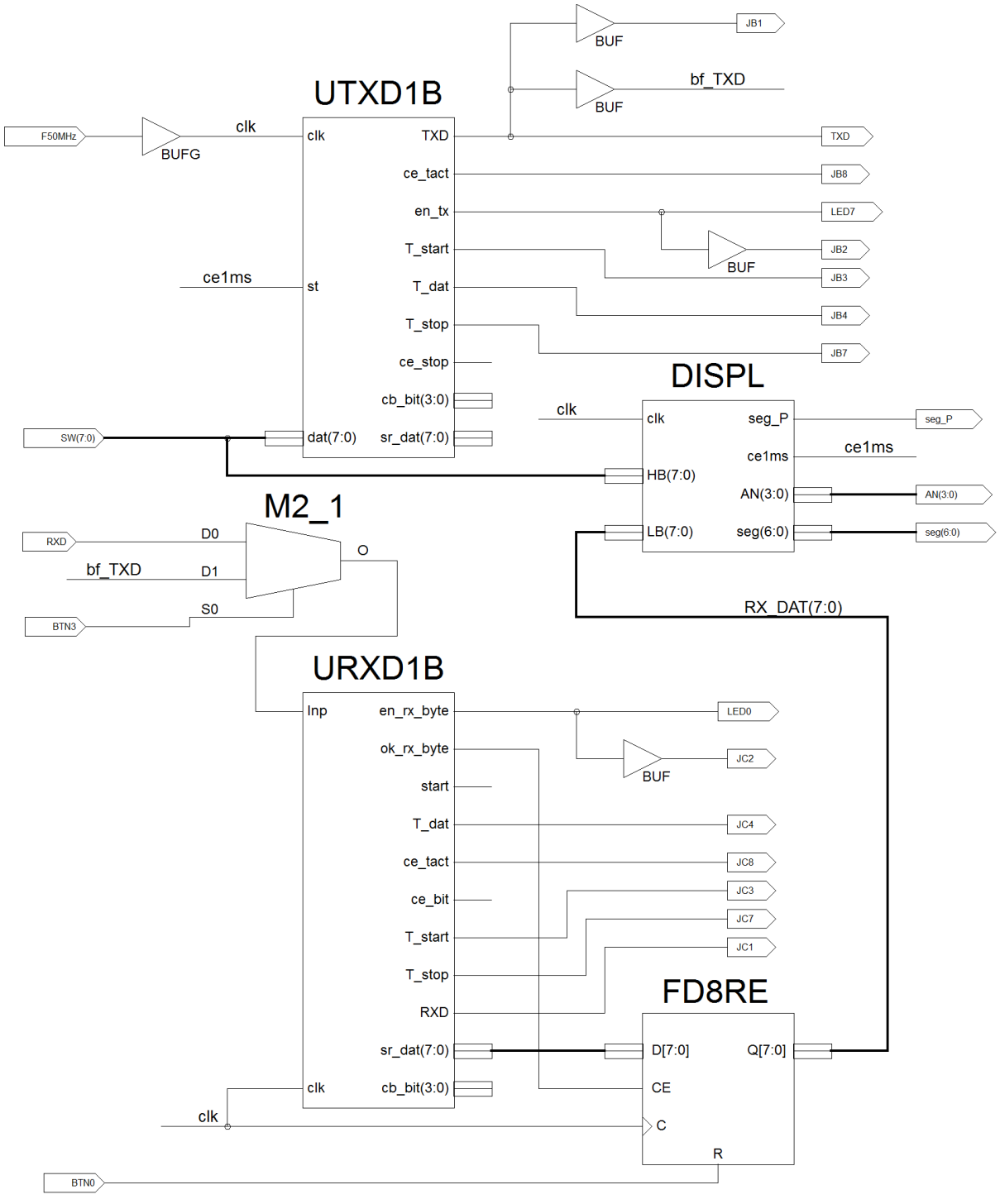


Рис.5 Пример схемы для сдачи работы

Модуль DISPL предназначен для отображения передаваемых и принимаемых данных на семи сегментном индикаторе макета (см. приложение 5.1). Старшие (левые) две цифры отображают tx_dat[7:0], а младшие (правые) две цифры отображают RX_DAT[7:0].

Байт данных для передатчика tx_dat[7:0] задается переключателями SW[7:0] макета (см. приложение 5.2).

Светодиоды LED7 и LED0 отображают состояния en_tx и en_rx_byte.

Кнопка BTN0 предназначена для сброса буфера FD8RE принятых данных. Кнопка BTN3 и мультиплексор M2-1 предназначены для обеспечения внутренней связи выхода передатчика со входом приемника (при BTN3=1) без внешней проводной связи TXD -> RXD на разъеме DB-9 COM порта макета (см. приложение 5.3).

Байт данных для передатчика задается переключателями SW[7:0] макета (см. приложение 5.3).

4. Задание к сдаче работы

4.1 В модулях UTXD1B и URXD1B установить заданную скорость (Таблица 1). Создать символы этих модулей

4.2 Создать модуль и символ семи сегментного индикатора DISPL (приложение 5.1)

4.3 Создать Schematic модуль Sch_LAB404 (см. рис.5).

4.4 Создать для Sch_LAB404 Implementation Constraints File (*.ucf см.приложение 5.6). Создать конфигурацию составленной схемы загрузить в макет (Generate Target PROM/ACE File). Отладить при необходимости работу схемы макета. Продемонстрировать работу макета.

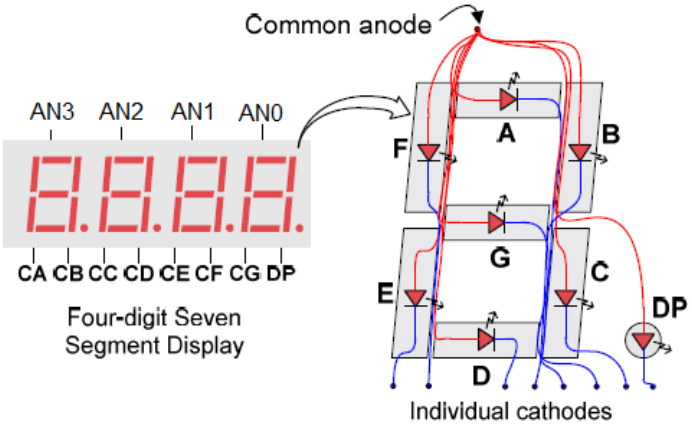
4.5 Получить и сохранить осциллограммы сигналов модуля UTXD1B: TXD (JB1), en_tx_byte (JB2), T_start (JB3), T_dat (JB4), T_stop (JB7).

4.6 Получить и сохранить осциллограммы сигналов модуля URXD1B: RXD (JC1), en_rx_byte (JC2), T_start (JC3), T_dat (JC4), T_stop (JC7).
Для всех осциллограмм ждущую развертку осциллографа запускать фронтом сигнала en_tx_byte (JB2).

4.7 Получить осциллограмму сигнала на выходе COM порта макета (вывод 2 DB-9, см. приложение 5.2)

5. Приложения

5.1 Семи сегментный индикатор

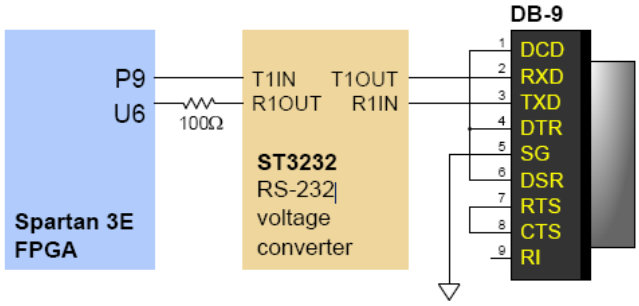


```
module DISPL (      input clk,              output wire[3:0] AN, //Аноды
                    input [15:0]dat,       output wire [6:0] seg, //Сегменты
                                           output wire seg_P,     //Точка
                                           output reg ce1ms=0);

parameter Fclk=50000 ;      //50000 kHz
parameter F1kHz=1 ;         //1 kHz
wire [1:0]ptr_P = 2'b10;    //Точка в центре
reg [15:0] cb_1ms = 0 ;
wire ce = (cb_1ms==Fclk/F1kHz) ;

always @ (posedge clk) begin
cb_1ms <= ce? 1 : cb_1ms+1 ;
ce1ms <= ce ;
end
//-----------------------------------------------
reg [1:0]cb_an=0 ; //Счетчик анодов
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
//-------Указатель точки-------
assign seg_P = !(ptr_P == cb_an) ;
endmodule
```

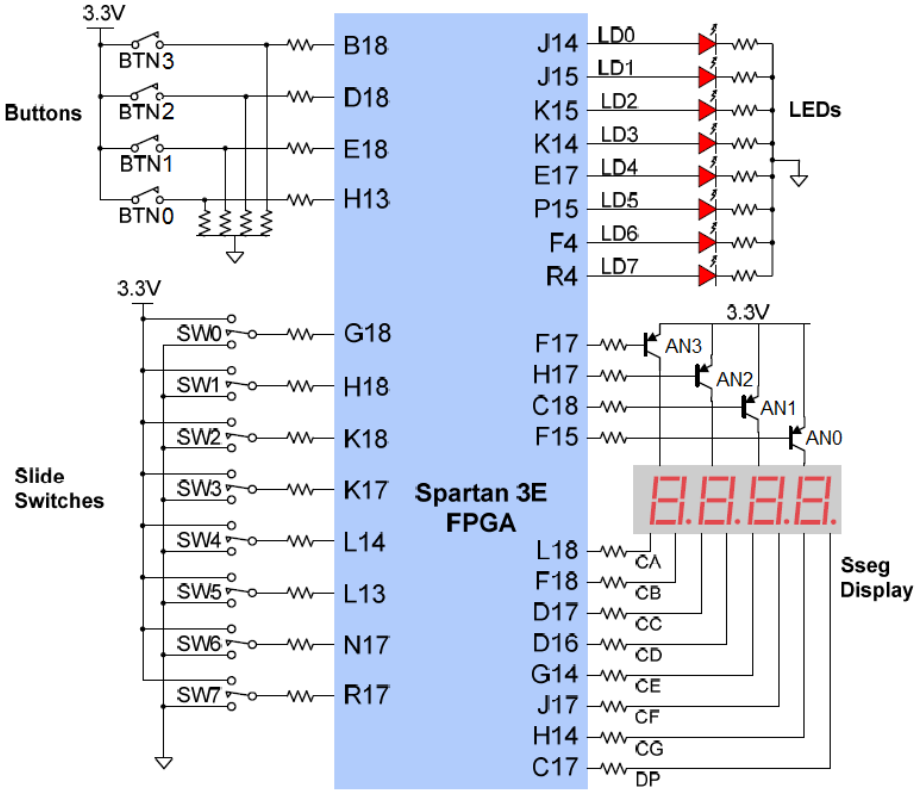
5.2 Расположение светодиодов (LED0-LED7), переключателей (SW0-SW7), кнопок (BTN0-BTN3) и индикатора (DISPL) на макете NEXYS2

5.3 COM порт макета NEXYS-2

На макете NEXYS-2 через преобразователь уровней ST3232 выведены только два сигнала: RXD и TXD. Остальные выводы соединены перемычками так, чтобы имитировать готовность к приему и передаче обеих сторон.



5.4 Схема соединений выводов ПЛИС с компонентами макета



5.5 Схема портов ввода/вывода макета NEXYS-2

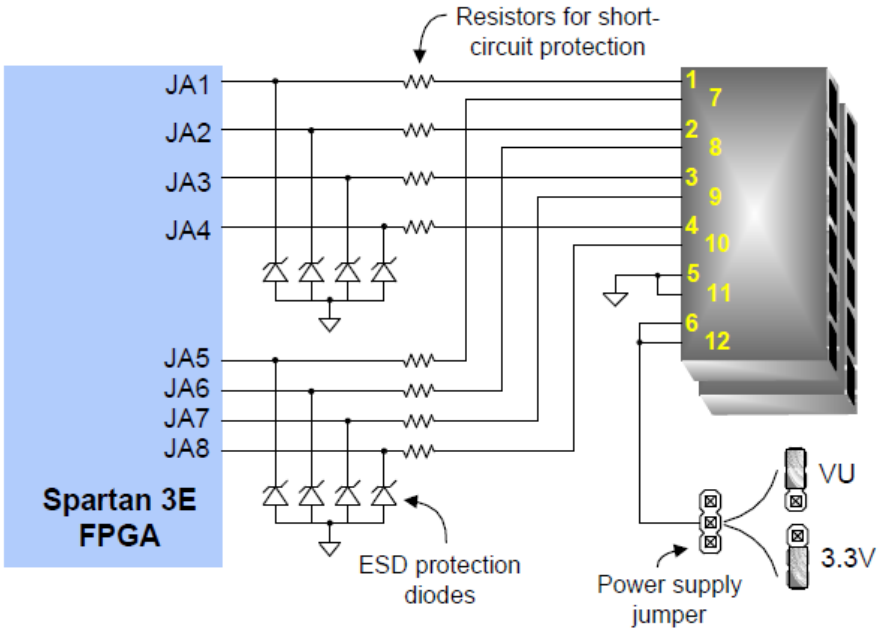


Table 3: Nexys2 Pmod Connector Pin Assignments

Pmod JA		Pmod JB		Pmod JC		Pmod JD	
JA1: L15	JA7: K13	JB1: M13	JB7: P17	JC1: G15	JC7: H15	JD1: J13	JD7: K14[1]
JA2: K12	JA8: L16	JB2: R18	JB8: R16	JC2: J16	JC8: F14	JD2: M18	JD8: K15[2]
JA3: L17	JA9: M14	JB3: R15	JB9: T18	JC3: G13	JC9: G16	JD3: N18	JD9: J15[3]
JA4: M15	JA10: M16	JB4: T17	JB10: U18	JC4: H16	JC10: J12	JD4:; P18	JD10: J14[4]

Notes: [1] shared with LD3 [2] shared with LD3 [3] shared with LD3 [4] shared with LD3

5.6 Список выводов ПЛИС макета Nexys2 (*.ucf файл)

```
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_P7"  LOC = "C17"  ; #CP

NET "LED0" LOC = "J14" ; #en_rx_byte
#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"  ; # en_tx

NET "BTN0"  LOC = "B18"  ; # RESET
#NET "BTN1"  LOC = "D18"  ; #BTN2
#NET "BTN2"  LOC = "E18"  ; #BTN1
NET "BTN3"  LOC = "H13"  ; Inp <- TXD

NET "TXD"  LOC = "P9"  ; #TXD(NESYX2)->RXD(COM port PC)
NET "RXD"  LOC = "U6" ; #RXD(NEXYS2)<-TXD(COM port PC)

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

#NET "JA1"  LOC = "L15"  ;#
#NET "JA2"  LOC = "K12"  ;#
#NET "JA3"  LOC = "L17"  ;#
#NET "JA4"  LOC = "M15"  ;#
#NET "JA7"   LOC = "K13"  ;#
#NET "JA8"   LOC = "L16"  ;#
#NET "JA9"  LOC = "M14"  ;#
#NET "JA10" LOC = "M16"  ;#
#         Выводы передатчика
NET "JB1"  LOC = "M13"  ; # TXD
NET "JB2"  LOC = "R18"  ; # en_tx
NET "JB3"  LOC = "R15"  ; # T_start
NET "JB4"  LOC = "T17"  ; # T_dat
NET "JB7"  LOC = "P17"  ; # T_stop
NET "JB8"     LOC = "R16" ;# ce_tact
#NET "JB9"  LOC = "T18" ;#
#NET "JB10" LOC = "U18" ;#
#         Выводы приемника
NET "JC1" LOC = "G15"   ; # RXD
NET "JC2" LOC = "J16"   ; # en_rx_byte
NET "JC3"  LOC = "G13"  ; # T_start
NET "JC4"  LOC = "H16"  ; # T_dat
NET "JC7" LOC  = "H15"  ; # T_stop
NET "JC8" LOC  = "F14"  ;# ce_tact
#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
```