Лабораторная работа №407ND

Последовательный интерфейс SPI

SPI - популярный интерфейс для последовательного обмена данными между микросхемами. Интерфейс SPI изначально он был придуман компанией Motorola, а в настоящее время используется в продукции многих производителей. Его наименование является аббревиатурой от 'Serial Peripheral Bus', что отражает его предназначение - шина для подключения внешних устройств. Шина SPI организована по принципу 'ведущий-ведомый' (MASTER/SLAVE). В качестве ведущего шины обычно выступает микроконтроллер, но им также может быть программируемая логика, DSP-контроллер или специализированная ИС. В роли ведомых выступают различного рода микросхемы, в т.ч. запоминающие устройства (EEPROM, Flash-память, SRAM), часы реального времени (RTC), АЦП/ЦАП, цифровые потенциометры, специализированные контроллеры и др.

Главным компонентом ведущего и ведомого модулей SPI является обычный сдвиговый регистр, в котором, как правило, по фронту сигнала синхронизации SCLK принимают данные, а по спаду меняют передаваемые данные. Поэтому на каждой стороне используется два регистра сдвига. Приемный регистр - сдвигает входные данные по фронту, а передающий выдвигает передаваемые данные по спаду сигнала синхронизации (Рис.1). Сигнал синхронизации генерирует MASTER и от этого сигнала полностью зависит работа SLAVE. Передаваемые ведущим последовательные данные называются MOSI (Master Output Slave Input), а принимаемые от ведомого, последовательные данные называются MISO (Master Input Slave Output). Активным уровнем сигнала разрешения работы или загрузки принятых данных LOAD является логический ноль. Этот сигнал не имеет устоявшегося названия (LOAD, CS, SS, FS…).

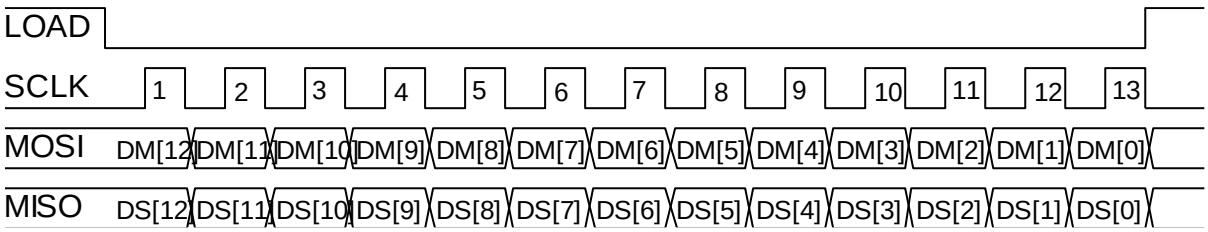

Рис.1 Пример временных диаграмм SPI интерфейса

Схема соединения модулей MASTER и SLAVE показана на рис.2. В некоторых случаях линии MISO не используется, если в модуле SLAVE не предусматривается ответная передача данных или в ней нет потребности. Одностороннюю передачу данных можно встретить у таких микросхем как ЦАП, цифровые потенциометры, программируемые усилители и драйверы. Обязательным условием SPI интерфейса является низкий уровень линии LOAD (CS,SS) при передаче и приеме данных.

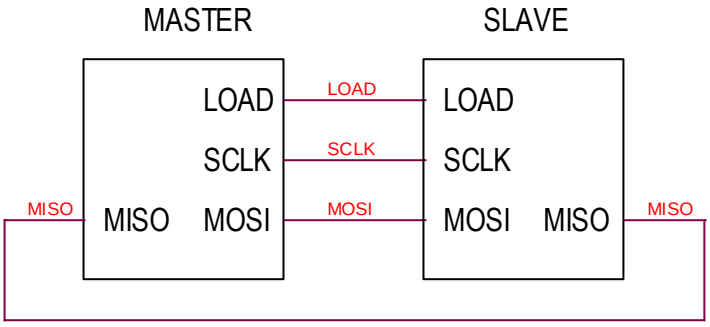

Рис. 2. Схема соединения модулей MASTER и SLAVE

Пример структурной схемы модуля SPI MASTER приведен на рис.3

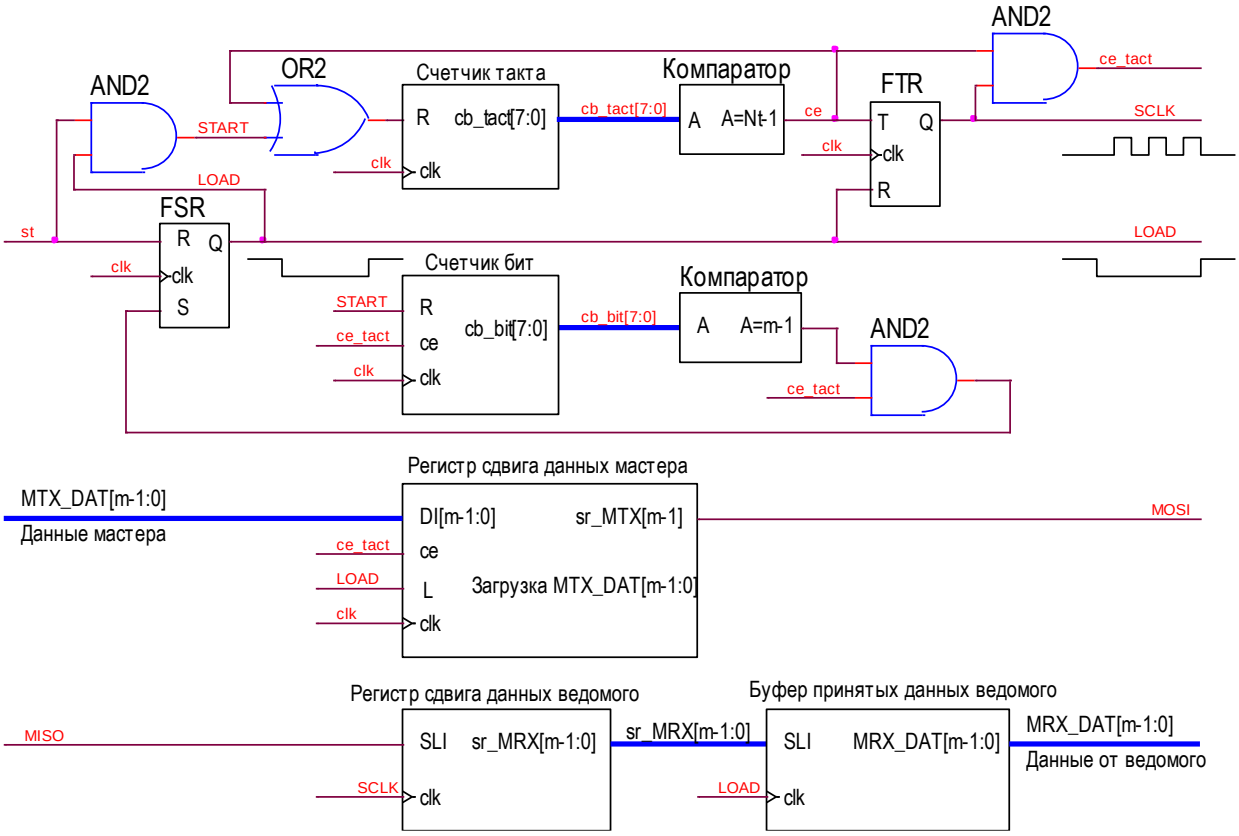

Рис.3 Пример структурной схемы модуля MASTER SPI

В состав схемы модуля входят:
cb_tact - счетчик такта,
cb_bit - счетчик бит,
sr_MTX - регистр сдвига передаваемых данных ведущего,
sr_MRX - регистр сдвига принимаемых данных ведомого,
FSR – SR-триггер формирователь сигнала LOAD,
FTR – T- триггер формирователь сигнала SCLK,
MRX_DAT – регистр принятых данных от SLAVE.

Передаваемые данные MTX_DAT[m-1:0] загружаются в регистр сдвига sr_MTX каждым фронтом сигнала синхронизации при высоком уровне LOAD.

Длительность импульса st должна быть равна одному периоду Tclk сигнала синхронизации clk, а период следования должен быть больше длительности последовательной передачи всех бит данных. Счетчик такта и счетчик бит “сбрасываются” не импульсом st, а импульсом start=st & LOAD, что обеспечивает защиту передачи от преждевременного появления импульса st.

Регистр сдвига sr_MRX принимаемых данных ведомого сдвигает MISO фронтами SCLK, а в конце сеанса его данные записываются фронтом LOAD в выходной регистр MRX_DAT.

Период Tce сигнала ce должен быть равен половине длительности Tbit передачи одного бита данных (Tce=Tbit/2), а период сигнала ce_tact должен быть равен Tbit.

Напомним, что на модуль SLAVE от ведущего поступает только 3 сигнала: LOAD, SCLK и MOSI. По каждому фронту SCLK последовательные данные MOSI вдвигаются в приемный регистр сдвига, и только по фронту сигнала LOAD, данные из приемного регистра сдвига, должны переписываются в регистр результата.

Возвращаемые ведомым данные MISO выдвигаются по спадам сигнала SCLK, которые должны в паузе между передачами загружаться высоким уровнем сигнала LOAD в передающий регистр ведомого. Если в модуле SLAVE нет собственного сигнала синхронизации, то возможна только асинхронная загрузка передающего регистра ведомого. Среди библиотечных компонент ПЛИС нет регистра сдвига с асинхронной загрузкой. Необходим D-триггер с двумя D входами: первый D обычный с загрузкой по фронту сигнала синхронизации clk, а второй Da с асинхронной загрузкой по уровню L. Пример схемы такого триггера приведен на рис.4, а ниже (V.1) схема этого триггера на VERILOG-e.

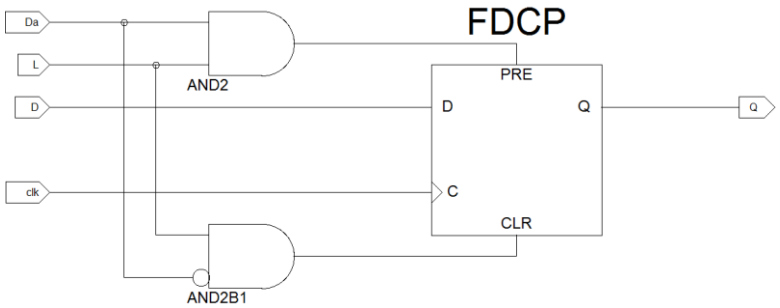

Рис.4 Схема триггера с двумя D входами

V.1 Схема модуля триггера с двумя D входами на VERILOG -e

```
module VF2D(input L,        output reg Q=0,
            input Da,       //Вход для асинхронной загрузки по уровню L
            input D,        // Вход для синхронной загрузки по фронту clk
            input clk);
always @ (posedge L or posedge clk) begin
Q <= L? Da : D ;
end
endmodule
```

1. Задание к допуску (стоимость 3)

1.1 Для заданного числа разрядов (Таблица 1) составить на VERILOG-e схему модуля SPI_MASTER.

1.2 Написать в тетради составленную схему модуля SPI_MASTER.

1.3 Для заданного варианта параметров начертить или написать в тетради схему модуля SPI_SLAVE.

1.4 Начертить временные диаграммы сигналов LOAD, MOSI, SCLK и MISO, соответствующих заданным MTX_DAT[m-1:0] и STX_DAT[m-1:0] (Таблица 1).

Таблица 1

№	Период Повторения Trep	Длительность такта Tbit	m	MASTER DAT MTX_DAT[m-1:0]	SLAVE DAT STX_DAT[m-1:0]
1	1 мс	1 мкс	12	0111 0000 0010	1110 0110 1100
2	100 мкс	0.2 мкс	11	011 0010 0100	101 1000 1101
3	10 мс	10 мкс	10	01 0010 0101	10 0110 1010
4	200 мкс	2 мкс	9	1 0111 1010	1 1101 1011
5	Кнопка	100 мкс	8	1011 0100	1010 1100
6	50 мкс	0.5 мкс	14	00 1111 0000 1110	00 1001 1111 1110
7	250 мкс	5 мкс	13	1 0010 0100 1100	0 1010 0101 1001
8	800 мкс	4 мкс	15	111 1100 0011 0011	101 0110 1010 1010
9	40 мкс	0.8 мкс	14	10 1001 0110 1100	11 1111 1100 0101
10	600 мкс	20 мкс	16	0110 1001 0011 1100	1110 1101 1010 1010

11	800 мкс	8 мкс	14	11 1001 0110 0101	11 1000 0111 1101
12	200 мкс	10 мкс	15	010 1100 0011 0110	110 1011 0011 0110
13	300 мкс	20 мкс	13	1 1100 0011 0010	1 1011 0111 1010
14	20 мс	100 мкс	10	10 0110 0100	11 1110 0101
15	100 мс	2 мс	11	010 1100 1100	110 1101 1001
16	50 мс	1 мкс	12	0011 0010 0101	1010 1010 1100
17	500 мкс	10 мкс	13	1 0010 0100 1001	0 1010 0101 1001
18	800 мкс	4 мкс	15	101 1100 1010 0011	101 0110 1010 1010
19	40 мкс	1 мкс	14	10 1011 0100 1101	11 1011 1100 0111
20	Кнопка	10 мкс	8	1001 0110	0011 1100

2. Задание к выполнению (стоимость 4)

В системе проектирования ISE Для ПЛИС используемой в макете Nexys2 создать проект с именем Lab407ND.

2.1 Создать модуль SPI_MASTER. При заданном значении MTX_DAT[m-1:0] провести моделирование его работы. Подкорректировать, если необходимо, схему и временные диаграммы сигналов. Зарисовать в тетради, полученные временные диаграммы сигналов.

2.2 Создать модуль и символ SPI_SLAVE. При заданном значении STX_DAT[m-1:0] провести моделирование его работы. Подкорректировать, если необходимо, схему и временные диаграммы сигналов. Зарисовать в тетради, полученные временные диаграммы сигналов.

Для моделирования модуля SPI_SLAVE необходимо создать схему Sch_test_SLAVE, в состав которую, в качестве источника сигналов входит модуль SPI_MASTER. Содержательную часть задания tf_Test_SPI на моделирование этой схемы можно составить на основе tf_MASTER.

V.2 Модуль схемы Sch_test_SLAVE

```
`define m 16 //m из Таблицы 1
module Sch_test_SLAVE(
//--------------Выводы SPI_MASTER----------------------
        input st,                       output wire LOAD,
        input clk,                      output wire SCLK,
        input[`m-1:0]MTX_DAT,           output wire MOSI,
        input RESET,                    output wire [`m-1:0]MRX_DAT,
                                        output wire [`m-1:0]sr_MTX,
                                        output wire [`m-1:0]sr_MRX,
                                        output wire [7:0]cb_bit,
                                        output wire ce_tact,
//-------------Выводы SPI_SLAVE----------------------
        input [`m-1:0]STX_DAT,          output wire MISO,
                                        output wire[`m-1:0]sr_STX,
                                        output wire[`m-1:0]sr_SRX,
                                        output wire[`m-1:0]SRX_DAT);

SPI_MASTER DD1 (.st(st),                .LOAD(LOAD),
                .clk(clk),              .SCLK(SCLK),
                .DI(MTX_DAT),           .MOSI(MOSI),

                .clr(RESET),            .DO(MRX_DAT),
```

```
                .MISO(MISO),        .sr_MTX(sr_MTX),
                                    .sr_MRX(sr_MRX),
                                    .cb_bit(cb_bit),
                                    .ce_tact(ce_tact));

SPI_SLAVE DD2 (  .load(LOAD),       .MISO(MISO),
                 .sclk(SCLK),       .sr_STX(sr_STX),
                 .MOSI(MOSI),       .sr_SRX(sr_SRX),
                 .clr(RESET),       .DO(SRX_DAT),
                 .DI(STX_DAT));
endmodule
```

3. Задание к сдаче (стоимость 3)

3.1. Составить схему Sch_Lab407ND состоящую из модулей:

- MASTER - ведущий,
- SLAVE - ведомый,
- DISPLAY – индикатор данных (приложение 4.1),
- MUX64_16 – мультиплексор данных,
- SOURCES_DAT – источник данных,
- Gen_st – генератор импульсов запуска st,
- M2_1 – переключателя сигнала LOAD.

Пример такой схемы приведен на рис.5.

Схемы модулей: MUX64_16, SORCE_DAT и Gen_st составить самостоятельно.

Модуль DISPLAY должен отображать на семи сегментном индикаторе макета передаваемые (MTX_DAT[m-1:0], STX_DAT[m-1:0]) и принимаемые данные (MRX_DAT[m-1:0], SRX_DAT[m-1:0]).

Сигналы MOSI, SCLK, LOAD и MISO модуля MASTER вывести на выводы JB1, JB2, JV3, JB4 макета Nexys2.

Создать Implementation Constraints File (*.ucf, см. приложение 4.2).

3.2 Создать файл конфигурации Sch_LAB407ND.bit (Generate Programming File) или *.mcs (Generate PROM, ACE, or JTAG File), загрузить его в макет.

Проверить соответствие показаний семи сегментного индикатора заданным параметрам.

Продемонстрировать при помощи осциллографа работу модуля MASTER. Проверить соответствие осциллограмм сигналов LOAD, MOSI, SCLK, MISO временным диаграммам пункта 1.4 допуска к работе. Сохранить осциллограммы сигналов LOAD, MOSI, SCLK, MISO.

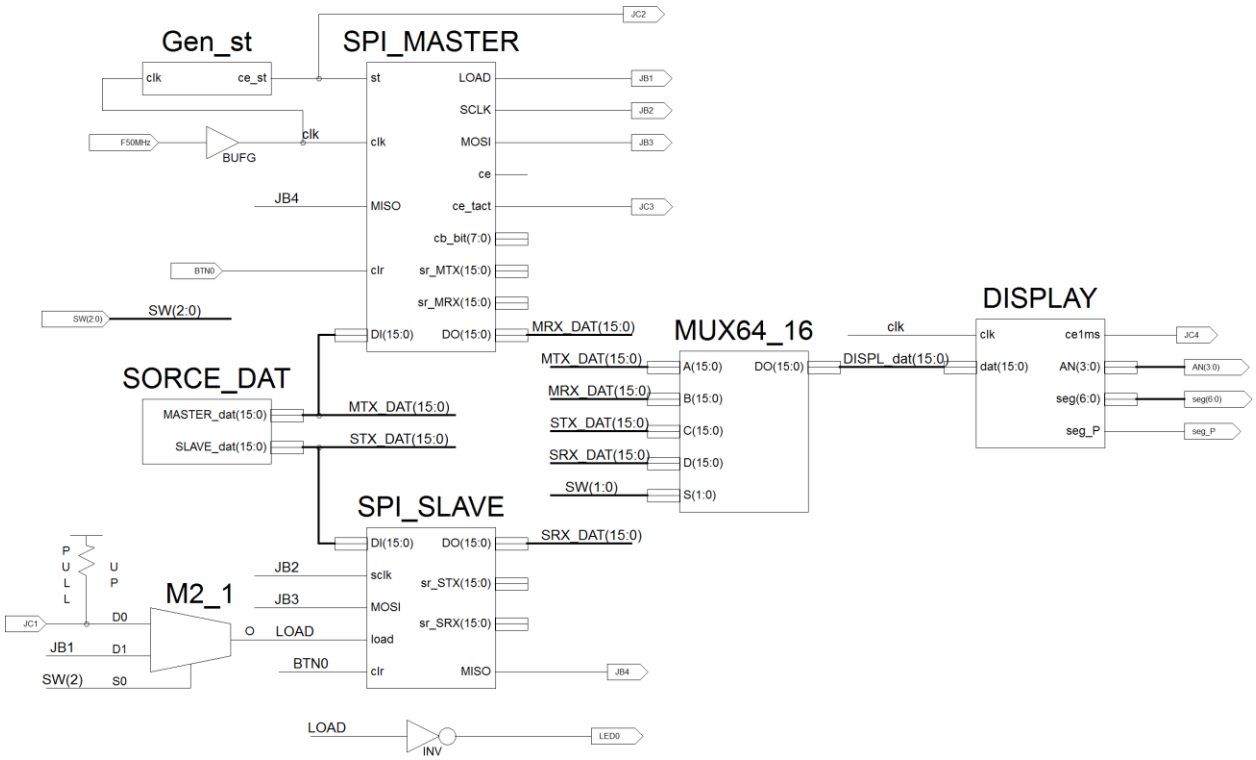

Рис.5 Пример схемы для сдачи лабораторной работы

4. Приложения

4.1 Модуль семи сегментного индикатора

```
module DISPLAY(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
reg [15:0] cb_1ms = 0 ;
wire ce = (cb_1ms==Fclk/F1kHz) ;
wire [1:0]ptr_P=2'b00 ; // Точка справа
//-------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
```

4.2 **Implementation Constraints File**

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

NET "LED0"  LOC = "J14" ; #!LOAD from M2_1
#NET " LED1"  LOC = "J15" ; #LD1
#NET " LED2"  LOC = "K15" ; #LD2
#NET " LED3"  LOC = "K14" ; #LD3
#NET " LED4"  LOC = "E17" ; #LD4
#NET " LED5"  LOC = "P15" ; #LD5
#NET " LED6"  LOC = "F4"  ; #LD6
#NET " LED7"  LOC = "R4"  ; #LD7

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] MUX
NET "SW<1>"  LOC = "H18"  ; # Adr[1] MUX
NET "SW<2>"  LOC = "K18"  ; #SO (M2_1)
#NET "SW<3>"  LOC = "K17"  ; #PTR[0]
#NET "SW<4>"  LOC = "L14"  ; #PTR[1]
#NET "SW<5>"  LOC = "L13"  ; #SWT5
#NET "SW<6>"  LOC = "N17"  ; #SWT6
#NET "SW<7>"  LOC = "R17"  ; #SWT7

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

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"  ; #

NET "JB1"  LOC = "M13"  ; # MOSI (out MASTER)
NET "JB2"  LOC = "R18"  ; # SCLK (out MASTER)
NET "JB3"  LOC = "R15"  ; # LOAD (out MASTER)
NET "JB4"  LOC = "T17"  ; # MISO (input MASTER)
#NET "JB7"  LOC = "P17"  ; #
#NET "JB8"   LOC = "R16"  ; #
#NET "JB9"  LOC = "T18"  ; #
#NET " JB10" LOC = "U18"  ; #

NET "JC1" LOC = "G15"   ; # input LOAD to M2_1
NET "JC2" LOC = "J16"   ; # st
NET "JC3"  LOC = "G13"  ; # ce_tact
NET "JC4"  LOC = "H16"  ; # ce1ms
#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"  ; # en_tx (SYNHR OSC)
#NET "JD7"   LOC = "K14"  ; #
#NET "JD8"   LOC = "K15"  ; #
#NET "JD9"  LOC = "J15"  ; #
#NET "JD10" LOC = "J14"  ; #
```

5. Контрольные вопросы

5.1 Можно ли при реализации модуля SLAVE на ПЛИС не использовать сигнал синхронизации clk (50 МГц)?

5.2 Влияет ли не симметрия сигнала SCLK на качество работы модулей MASTER, SLAVE?

5.3 Влияет ли не стабильность периода сигнала SCLK на качество работы модулей MASTER, SLAVE?

5.4 Влияет ли длительность фронтов и спадов сигнала SCLK на качество работы модулей MASTER, SLAVE?

5.5 Влияет ли длительность фронта и спада сигнала LOAD на качество работы модулей MASTER, SLAVE?

5.6 Влияет ли длительность фронтов и спадов сигналов MOSI и MISO на качество работы модулей MASTER, SLAVE?

5.7 Можно ли использовать SPI интерфейс на больших расстояниях?

5.8 Как правильнее формировать сигналы SCLK и LOAD модуля MASTER: при помощи логического элемента (wire) или при помощи триггера (reg)?