Лабораторная работа №407D

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, сигналы синхронизации и ввода/вывода битового потока которого и образуют интерфейсные сигналы, каждый из которых в принципе может одновременно выполнять и функцию приемника, и функцию передатчика. Однако этого практически никогда не делают, потому что на обеих сторонах, 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 ведущий шины (MASTER) и от этого сигнала полностью зависит работа подчиненного шины (SLAVE). Передаваемые ведущим последовательные данные, как правило, 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,…).

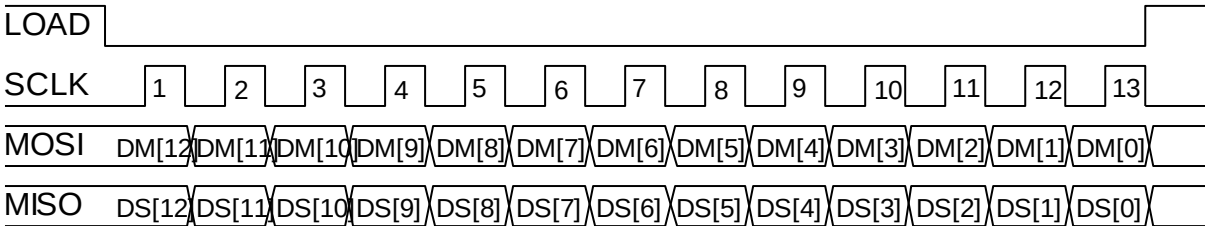


Рис.1 Пример временных диаграмм SPI интерфейса

Существует три типа подключения к шине SPI, в каждом из которых участвуют четыре сигнала. Самое простое подключение, в котором участвуют только две микросхемы, показано на рис.2. Здесь, ведущий шины передает данные по линии MOSI синхронно со сгенерированным им же сигналом SCLK, а подчиненный захватывает переданные биты данных по фронтам принятого сигнала синхронизации. Одновременно с этим подчиненный отправляет свою посылку данных. Представленную схему можно упростить исключением линии MISO, если используемая подчиненная ИС не предусматривает ответную передачу данных или в ней нет потребности. Одностороннюю передачу данных можно встретить у таких микросхем как ЦАП, цифровые потенциометры, программируемые усилители и драйверы. Таким образом, рассматриваемый вариант подключения подчиненной ИС требует 3 или 4 линии связи. Чтобы подчиненная ИС принимала и передавала данные, помимо наличия сигнала синхронизации, необходимо также, чтобы линия SS была переведена в низкое состояние.

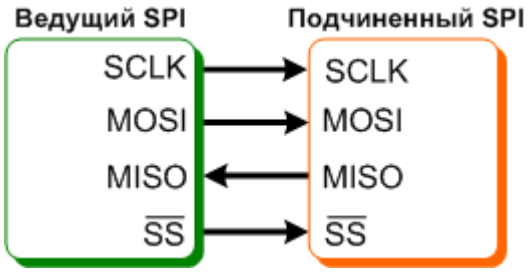


Рис. 2. Простейшее подключение к шине SPI

При необходимости подключения к шине SPI нескольких микросхем используется либо независимое (параллельное) подключение (рис.3), либо каскадное (последовательное) (рис.4). Независимое подключение более распространенное, т.к. достигается при использовании любых SPI-совместимых микросхем. Здесь, все сигналы, кроме выбора микросхем, соединены параллельно, а ведущий шины, переводом того или иного сигнала SS (или LOAD) в низкое состояние, задает, с какой подчиненной ИС он будет обмениваться данными. Главным недостатком такого подключения является необходимость в дополнительных линиях для адресации подчиненных микросхем (общее число линий связи равно 3+n, где n-количество подчиненных микросхем). Каскадное включение избавлено от этого недостатка, т.к. здесь из нескольких микросхем образуется один большой сдвиговый регистр. Для этого выход передачи данных одной ИС соединяется со входом приема данных другой, как показано на рисунке 4. Входы выбора микросхем здесь соединены параллельно и, таким образом, общее число линий связи сохранено равным 4. Однако использование каскадного подключения возможно только в том случае, если его поддержка указана в документации на используемые микросхемы. Чтобы выяснить это, важно знать, что такое подключение по-английски называется 'daisy-chaining'.

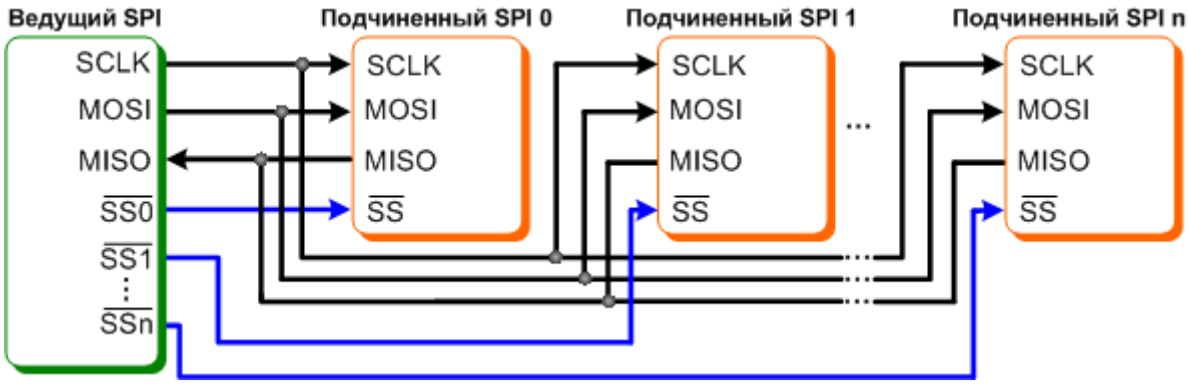


Рис. 3. Независимое подключение к шине SPI

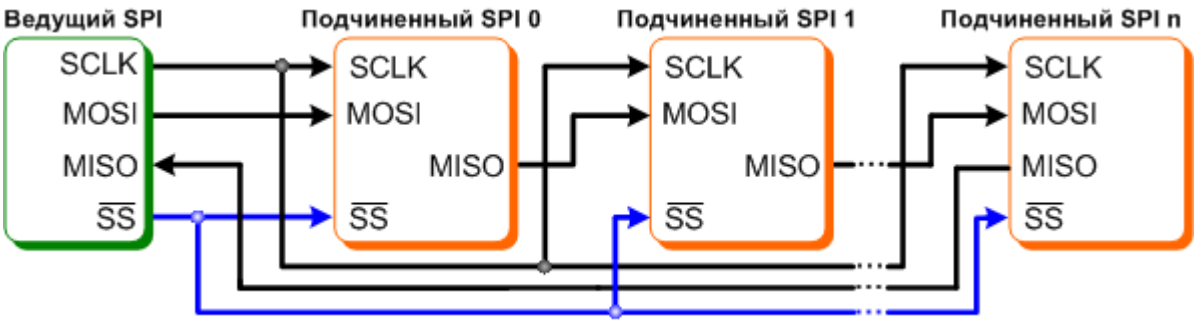


Рис. 4. Каскадное подключение к шине SPI

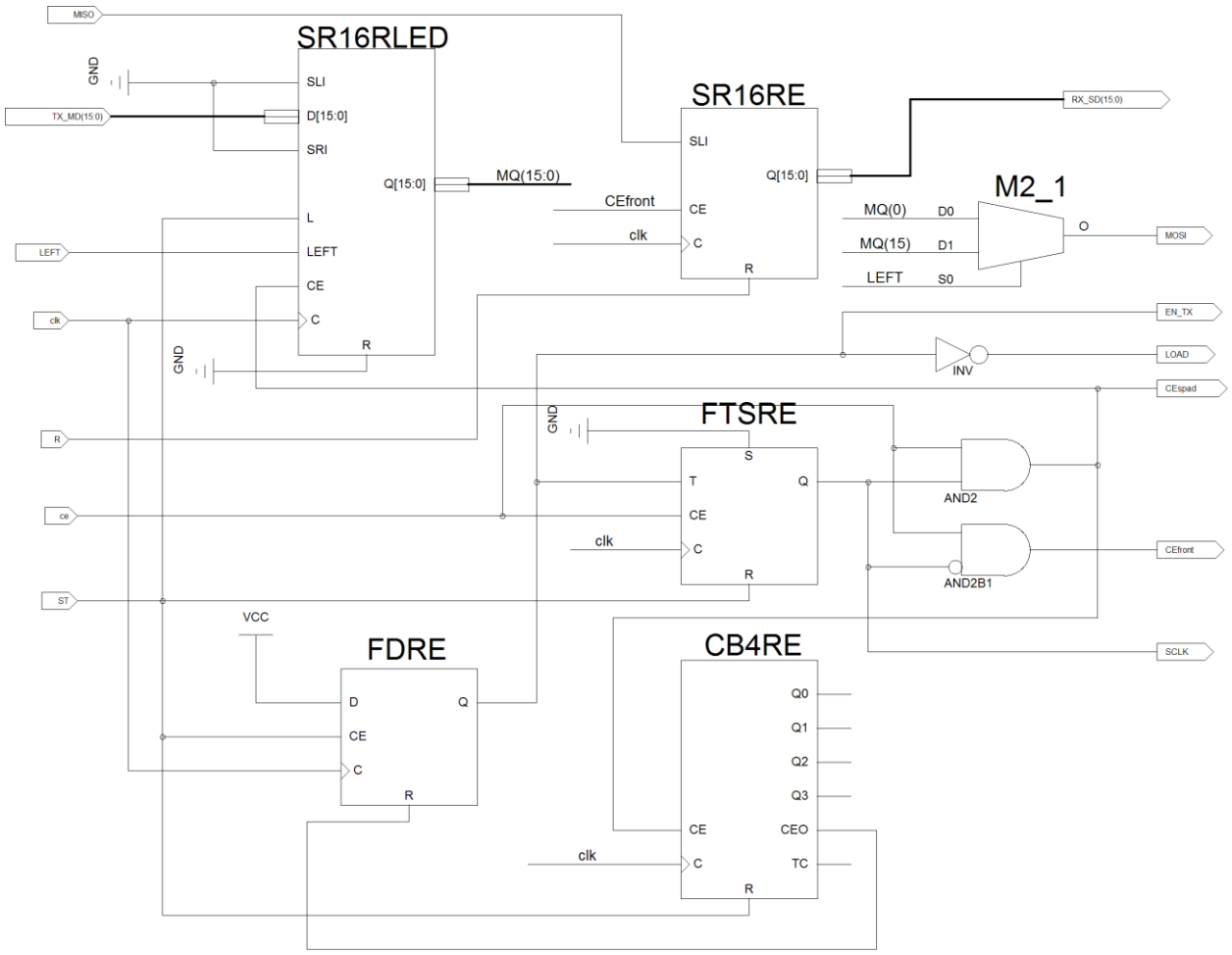


Рис.5 Пример схемы 16-и битного модуля MASTER SPI составленной из библиотечных компонент ПЛИС

В состав схемы модуля входят:

- **SR16RLED** - регистр сдвига передаваемых данных ведущего,
- **SR16RE** – регистр сдвига принимаемых данных ведомого,
- **CB4RE –** счетчик бит,
- **FTSRE –** формирователь сигнала SCLK,
- **FDRE** – формирователь сигнала EN_TX,
- **M2_1 –** переключатель выходного сигнала MOSI.

Регистр сдвига **SR16RLED –** реверсивный и загружаемый. При LEFT=1 – сдвиг влево, т.е. в сторону старших разрядов, а при LEFT=0 – сдвиг вправо, в сторону младших разрядов. Передаваемые данные загружаются импульсом st. Длительность импульса st должна быть равна одному периоду Tclk сигнала синхронизации clk, а период следования должен быть больше длительности последовательной передачи всех бит данных.

Переключатель **M2_1** при LEFT=1 соединяет вывод MOSI со старшим разрядом, а при LEFT=0 – с младшим разрядом регистра **SR16RLED**.

Регистр сдвига **SR16RE** сдвигает данные только влево, поэтому на вход MISO данные должны поступать старшими битами вперед.

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

V.1 Схема модуля MASTER написанная на Verilog-е

```
module MASTER(  input clk,                  output reg EN_TX,
                input ce,                   output wire LOAD,
```

```
                input st,                   output reg SCLK,
                input MISO,                 output wire MOSI,
                input [15:0] TX_MD,         output reg [15:0] RX_SD,
                input LEFT,                 output wire CEfront,
                input R,                    output wire CEspad);

reg [15:0] MQ=0 ; //
reg [3:0]cb_bit ; //
assign MOSI = LEFT? MQ[15] : MQ[0] ;//
assign LOAD = !EN_TX ;//
assign CEfront = !SCLK & ce ; //
assign CEspad = SCLK & ce ; //

always @ (posedge clk) begin
MQ <= st? TX_MD : (CEspad & LEFT)? MQ<<1 : (CEspad & !LEFT)? MQ>>1 : MQ ;
EN_TX <= ((cb_bit==15) & CEspad)? 0 : st? 1 : EN_TX ;
cb_bit <= st? 0 : CEspad? cb_bit+1 : cb_bit ;
SCLK <= st? 0 : (EN_TX & ce)? !SCLK : SCLK ;
RX_SD <= R? 0 : CEfront? RX_SD<<1 | MISO : RX_SD ;
end
endmodule
```

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. Среди библиотечных компонент нет регистра сдвига с асинхронной загрузкой. Для его схемы 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.6, а ниже (V.2) схема этого триггера на Verilog-e.

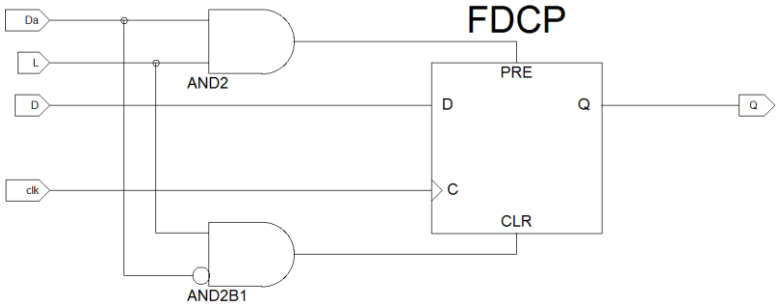


Рис.6 Схема триггера с двумя D входами

V.2 Схема модуля триггера с двумя D входами на Verilog-e

```
module VF2D(input L,        output reg Q=0,
            input Da,
            input D,
            input clk);
always @ (posedge L or posedge clk) begin
Q <= L? Da : D ;
end
endmodule
```

1. Задание к допуску

1.1 Для заданного числа разрядов (Таблица 1) из библиотечных компонент начертить в тетради схему модуля MASTER (рис.5) SPI интерфейса.

1.2 Написать в тетради для заданного числа разрядов схему (V.1) модуля MASTER на Verilog-е.

1.3 Составить схему регистра сдвига заданного числа разрядов с асинхронной загрузкой данных.

1.4 Для заданного варианта параметров начертить или написать в тетради схему модуля SLAVE SPI интерфейса.

1.3 Начертить временные диаграммы сигналов LOAD, MOSI, SCLK и MISO модулей MASTER_BL и SLAVE_BL, соответствующих заданным MD_TX[m-1:0] и SD_TX[m-1:0] (Таблица 1).

Таблица 1

№	Период Повторения Trep	Длительность такта Tbit	m	MASTER DAT MD_TX[m-1:0]	SLAVE DAT SD_TX[m-1:0]
1	1 мс	1 мкс	12	0111 0000 0010	1110 0110 1100
2	100 мкс	0.1 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	0.4 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

2. Задание к выполнению

В системе проектирования ISE Для ПЛИС используемой в макете Nexys2 создать проект с именем Lab407D.

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

При моделировании этого модуля не обязательно устанавливать заданный период Tce=Tbit/2. Ниже приведен пример содержательной части (Verilog Test Fixture) модуля tf_MASTER задания на моделирование в котором Tce=100ns (Tbit=200ns).

```
parameter Tclk=20; //Период сигнала синхронизации 20 нс
always begin clk=1; #(Tclk/2); clk=0; #(Tclk/2); end
parameter Tce=100; //Период сигнала ce 100 нс
always begin ce=0; #(4*Tce/5); ce=1; #(1*Tce/5); end
```

```
// Initialize Inputs
initial begin
        ST = 0; LEFT = 0; TX_MD = 16'h0000; MISO = 0; R = 0;
#180;   ST = 1; LEFT = 1; TX_MD = 16'h1234; MISO = 0; R = 0;
#20;    ST = 0; LEFT = 1; TX_MD = 16'h1234; MISO = 0; R = 0;

#3600;  ST = 1; LEFT = 0; TX_MD = 16'h1234; MISO = 0; R = 0;
#20;    ST = 0; LEFT = 0; TX_MD = 16'h1234; MISO = 0; R = 0;
end
```

При первом запуске (LEFT=1) данные (MOSI) передаются старшими битами вперед, а при втором запуске (LEFT=0) данные передаются младшими битами вперед.

2.2 Создать модуль и символ SLAVE. При заданном значении SD_TX[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 SLAVE необходимо создать схему Test_SPI, в состав которую, в качестве источника сигналов входит модуль MASTER. Пример такой схемы приведен на рис.7. Содержательную часть задания tf_Test_SPI на моделирование этой схемы можно составить на основе tf_MASTER.

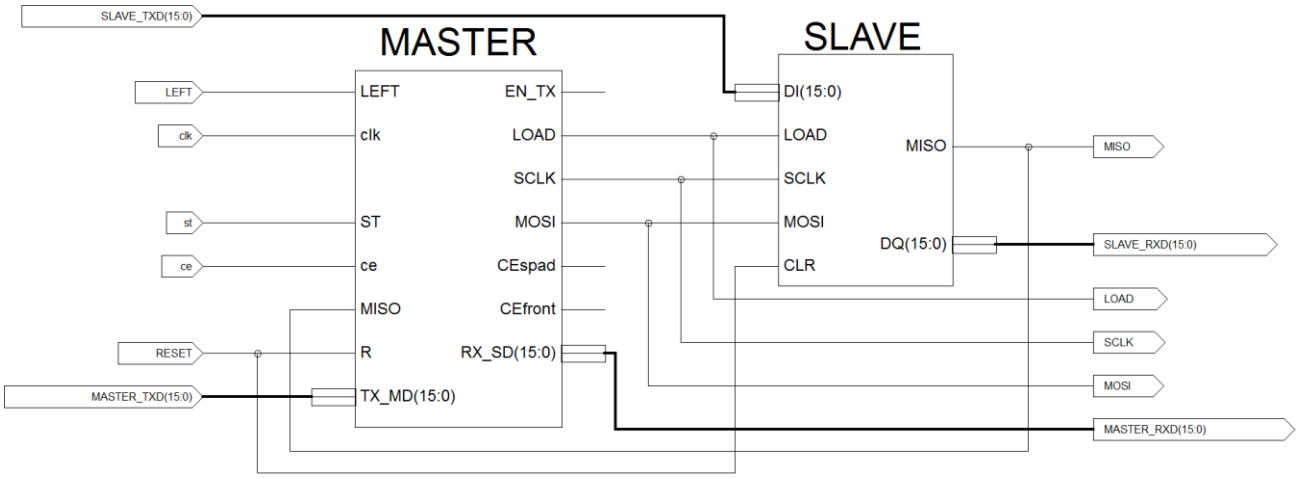


Рис.7 Схема для для совместного моделирования модулей MASTER и SLAVE

V.3 Модуль схемы рис.7

```
module Test_MASTER_SLAVE(
        input ce,                       output wire LOAD,
        input clk,                      output wire SCLK,
        input st,                       output wire MOSI,
        input LEFT,                     output wire MISO,
        input RESET,
        input [15:0]MASTER_TXD,         output wire [15:0]SLAVE_RXD,
        input [15:0]SLAVE_TXD,          output wire [15:0]MASTER_RXD);

MASTER DD1 (    .clk(clk),              .LOAD(LOAD),
                .ce(ce),                .SCLK(SCLK),
                .ST(st),                .MOSI(MOSI),
                .R(RESET),              .RX_SD(MASTER_RXD),
                .LEFT(LEFT),
                .MISO(MISO),
                .TX_MD(MASTER_TXD)   );
```

```
SLAVE DD2 (.LOAD(LOAD),       .MISO(MISO),
           .SCLK(SCLK),       .DQ(SLAVE_RXD),
           .MOSI(MOSI),
           .CLR(RESET),
           .DI(SLAVE_TXD)) ;
endmodule
```

3. Задание к сдаче

3.1. Составить схему Sch_Lab407D, состоящую из модулей:

- MASTER - ведущий,
- SLAVE - ведомый,
- DISPLAY – индикатор данных (приложение 4.1),
- MUX64_16 – мультиплексор данных,
- SOURCES_DAT – источник данных,
- Gen_ce_st – генератор ce и импульсов запуска st,
- FDmRE – регистров принятых данных и
- M2_1 – переключателей входных сигналов.

Пример такой схемы приведен на рис.8.

Схемы модулей: MUX64_16, SOURCES_DAT, Gen_ce_st и FDmRE составить самостоятельно.

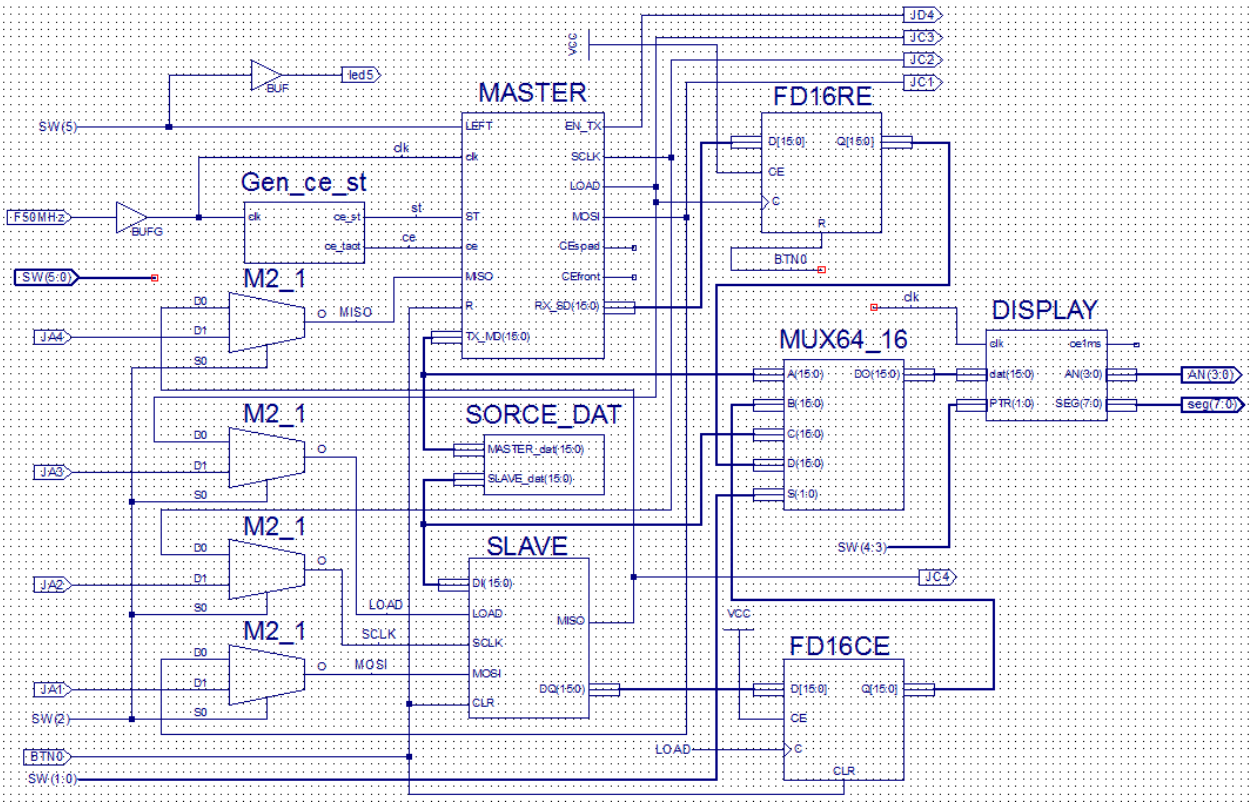
Модуль DISPLAY должен отображать на семи сегментном индикаторе макета передаваемые (MD_TX[m-1:0], SD_TX[m-1:0]) и принимаемые данные (MD_RX[m-1:0], SD_RX[m-1:0]).

Сигналы MOSI, SCLK, LOAD и MISO модуля MASTER вывести на выводы JC1, JC2, JC3, JC4 макета Nexys2.

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

3.2 Создать файл конфигурации Sch_LAB407D.bit (Generate Programming File) или *.mcs (Generate PROM, ACE, or JTAG File), загрузить его в макет. Продемонстрировать при помощи осциллографа работу модуля MASTER. Проверить соответствие осциллограмм сигналов LOAD, MOSI, SCLK и показаний семи сегментного индикатора заданным параметрам. Зарисовать осциллограммы сигналов LOAD, MOSI, SCLK.

3.3 Соединить выводы MASTER со выводами SLAVE (JC1=JA1, JC2=JA2, JC3=JA3, JC4=JA4) . Сопоставить показания индикатора передаваемых и принимаемых данных. Проверить влияние сигнала LEFT на принимаемые данные модулем SLAVE. Провести при помощи осциллографа наблюдение сигнала MISO и зарисовать осциллограмму MISO в тетрадь.



4. Приложения

4.1 Схема модуля **DISPLAY**

```
module DISPLAY      (input clk,              output wire [3:0] AN,
                     input [15:0]DAT,        output wire[7:0] SEG,
                     input [1:0]PTR,         output wire ce1ms );

wire [3:0]Dig; wire [1:0]Adr_dig ;//Проводники связей между модулями
                    //Генератор сигналов анодов
Gen_4An       DD1(  .clk(clk),        .q(Adr_dig),
                    .ce(ce1ms),       .An(AN));
                    //Мультиплексор
MUX16_4       DD2 (  .DI(DAT),        .DO(Dig),
                                      .s(Adr_dig));
                    //Формирователь 7-ми сегментных цифр
D7seg   DD3 (  .dig(Dig),             .seg(SEG[6:0]));
                    //Формирователь точки
Gen_P   DD4 (  .adr_An(Adr_dig),      .seg_P(SEG[7]),
                                      .ptr(PTR) );
                    //Генератор периодического сигнала ce1ms
Gen_ce1ms  DD5 (.clk(clk),      .ce_1ms(ce1ms));
endmodule
```

4.1.1 Схема модуля **Gen_4An**

```
module Gen_4An  (     input clk,        output reg [1:0] q = 0,//Счетчик номера анода
                      input ce,         output wire [3:0] An );
assign    An =    (q==0)? 4'b1110  :    //включение цифры 0 (младшей)
                  (q==1)? 4'b1101  :    //включение цифры 1
                  (q==2)? 4'b1011  :    //включение цифры 2
                          4'b0111 ;     //включение цифры 3 (старшей)
always @ (posedge clk) if (ce) begin
q <= q+1 ;
```

```
end
endmodule
```

4.1.2 Схема модуля **MUX16**

```
module MUX16_4(   input [15:0] DI, output [3:0] DO,
                  input [1:0] s    );

assign  DO =  (s==0)? DI[3:0]:
              (s==1)? DI[7:4]:
              (s==2)? DI[11:8]: DI[15:12];
endmodule
```

4.1.3 Схема модуля **D7seg**

```
module D7seg    (input [3:0] dig,      output wire [6:0] seg);
//                                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    h
                (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
```

4.1.4 Схема модуля **Gen_P**

```
module Gen_P (        input [1:0] ptr,output wire seg_P,
                            input [1:0] adr_An );
assign seg_P = !(ptr==adr_An) ;
endmodule
```

4.1.5 Схема модуля **Gen_ce1ms**

```
module Gen_ce1ms(  input clk,//Сигнал синхронизации
                   output wire ce_1ms); //1 милисекунда

parameter Fclk  =50000000 ; //Частота генератора синхронизации
parameter F1kHz =1000 ;      //Частота 1 кГц

reg[16:0]cb_ms = 0  ;          //Счетчик миллисекунд
assign ce_1ms = (cb_ms==1) ;// 1 милисекунда

//Делитель частоты
always @(posedge clk) begin
cb_ms <= ce_1ms? (Fclk/F1kHz) : cb_ms-1 ; //Счет миллисекунд
```

```
end
endmodule
```

4.2 **Implementation Constraints File**

```
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"  ; #BTN3
#NET "BTN<1>"  LOC = "D18"  ; #BTN2
#NET "BTN<2>"  LOC = "E18"  ; #BTN1
#NET "BTN3"  LOC = "H13"  ; #BTN0

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

#NET "led<0>"  LOC = "J14" ; #LD0
#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 "led<7>"  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<7>"  LOC = "C17"  ; #CP

NET "SW<0>"  LOC = "G18"  ; #SWT0
NET "SW<1>"  LOC = "H18"  ; #SWT1
NET "SW<2>"  LOC = "K18"  ; #SWT2
NET "SW<3>"  LOC = "K17"  ; #SWT3
NET "SW<4>"  LOC = "L14"  ; #SWT4
#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"  ;#Pin1
NET "JA2"  LOC = "K12"  ;#Pin2
NET "JA3"  LOC = "L17"  ;#Pin3
NET "JA4"  LOC = "M15"  ;#Pin4
#NET "JA7"   LOC = "K13"  ;#Pin7
```

```
#NET "JA8"   LOC = "L16"  ;#Pin8
#NET "JA9"  LOC = "M14"  ;#Pin9
#NET "JA10" LOC = "M16"  ;#Pin10

#NET "JB1"  LOC = "M13"  ;#
#NET "JB2"  LOC = "R18"  ;#
#NET "JB3"  LOC = "R15"  ;#
#NET "JB4"  LOC = "T17"  ;#
#NET "JB7"  LOC = "P17"  ;#Pin7
#NET "JB8"   LOC = "R16"  ;#Pin8
#NET "JB9"  LOC = "T18"  ;#Pin9
#NET " JB10" LOC = "U18"  ;#Pin10

NET "JC1" LOC = "G15"   ;#Pin1
NET "JC2" LOC = "J16"   ;#Pin2
NET "JC3"  LOC = "G13"  ;#Pin3
NET "JC4"  LOC = "H16"  ;#Pin4
#NET "JC7" LOC = "H15"   ;#Pin7
#NET "JC8" LOC = "F14"   ;#Pin8
#NET "JC9" LOC = "G16"   ;#Pin9
#NET "JC10" LOC = "J12"  ;#Pin10

#NET "JD1"  LOC = "J13"  ;#Pin1
#NET "JD2"  LOC = "M18"  ;#Pin2
#NET "JD3"  LOC = "N18"  ;#Pin3
NET "JD4"  LOC = "P18"  ; #Pin4
#NET "JD7"   LOC = "K14"  ;#LD3
#NET "JD8"   LOC = "K15"  ;#LD3
#NET "JD9"  LOC = "J15"  ;#LD3
#NET "JD10" LOC = "J14"  ;#LD3
```

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)?