МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра САПР

ОТЧЕТ

по лабораторной работе №5
по дисциплине «Электроника и схемотехника»
Тема: «Коммуникация по интерфейсу SPI»

Студент гр. 3363 _____ Минко Д.А.

Студент гр. 3363 _____ Овсейчик Н.И.

Студент гр. 3363 _____ Гончаренко О.Д.

Преподаватель _____ Рыбин В.Г.

Санкт-Петербург
2024

Цель работы

Создать два проекта для ведущего и ведомого устройств (для удобства последующей прошивки). В каждом из них описать отдельно модуль SPI и поместить его в оболочку, которая будет передавать сигналы в модуль и выводить передаваемые данные на светодиоды.

Ход работы

1. В проекте для ведущей платы был написан модуль SPI, который отвечает за состояния без передачи и с передачей, а также отслеживает, были ли переданы пакеты. Затем он был помещен в оболочку, где описано инкрементирование и передача при помощи кнопок.

2. В отдельном проекте был написан SPI модуль для ведомой платы, затем помещена в оболочку, которая передает значение регистра на светодиоды.

3. Отдельно была сделана разводка для ведомого и ведущего устройства (рис. 1 и рис. 2). Разводка идентична за исключением того, что на ведомом устройстве учитываются кнопки: одна для увеличения двоичного значения числа, вторая для передачи. При этом на ведущем устройстве значение не обнуляется, делать это нужно вручную.

1	MISO	output		39	2	False	***LVCMOS18***	***8***	***UP***
2	MOSI	input		37	2	False	***LVCMOS18***	N/A	***UP***
3	SCLK	input		36	2	False	***LVCMOS18***	N/A	***UP***
4	SS	input		38	2	False	***LVCMOS18***	N/A	***UP***
5	led[0]	output		10	3	False	***LVCMOS18***	***8***	***UP***
6	led[1]	output		11	3	False	***LVCMOS18***	***8***	***UP***
7	led[2]	output		13	3	False	***LVCMOS18***	***8***	***UP***
8	led[3]	output		14	3	False	***LVCMOS18***	***8***	***UP***
9	led[4]	output		15	3	False	***LVCMOS18***	***8***	***UP***
10	led[5]	output		16	3	False	***LVCMOS18***	***8***	***UP***

Рис. 1

	Port	Direction	Diff Pair	Location	Bank	Exclusive	IO Type	Drive	Pull Mode	PCI Clamp
1	MISO	input		39	2	False	LVCMOS18	N/A	UP	ON
2	MOSI	output		37	2	False	LVCMOS18	8	UP	N/A
3	SCLK	output		36	2	False	LVCMOS18	8	UP	N/A
4	SS	output		38	2	False	LVCMOS18	8	UP	N/A
5	clk	input		52	1	False	LVCMOS18	N/A	UP	ON
6	increment	input		4	3	False	LVCMOS18	N/A	UP	ON
7	led[0]	output		10	3	False	LVCMOS18	8	UP	N/A
8	led[1]	output		11	3	False	LVCMOS18	8	UP	N/A
9	led[2]	output		13	3	False	LVCMOS18	8	UP	N/A
10	led[3]	output		14	3	False	LVCMOS18	8	UP	N/A
11	led[4]	output		15	3	False	LVCMOS18	8	UP	N/A

Рис. 2

4. При настройке важно выставить на обеих платах режим «Embedded Flash Mode», чтобы прошивка сохранялась. Дополнительно платы соединены между собой по gnd и пину на 5V.

Выводы

В результате выполнения лабораторной работы было написано два SPI-модуля для Master’a и для Slave. Они были помещены в оболочку, с помощью которой и осуществляется передача бинарного значения с ведущего на ведомое.

ПРИЛОЖЕНИЕ 1 – ИСХОДНЫЙ КОД ПРОГРАММЫ

1. Модуль Master_Control

```
module master_control
(
     input clk,
     input increment,
     input transmit,
     output [5:0] led,

     output SCLK,
     output MOSI,
     input MISO,
     output SS
);

reg [7:0] data_out;
wire [7:0] data_in;

reg [19:0] start_counter;
wire start;

reg [19:0] inc_counter;

spi_master m (clk, start, data_out, data_in, SCLK, MOSI, MISO,
SS);

always @(posedge clk)
begin
     if (!increment)
     begin
          if (!inc_counter[19])
          begin
               inc_counter = inc_counter + 1;
               if (inc_counter[19])
```

```
                    data_out = data_out + 1;
          end
     end
     else
     begin
          inc_counter <= 0;
     end
end

always @(posedge clk)
begin
     if (!transmit)
     begin
          if (!start_counter[19])
               start_counter <= start_counter + 1;
     end
     else
     begin
          start_counter <= 0;
     end
end

assign led = ~data_out[5:0];
assign start = start_counter[19];

endmodule
```

2. Модуль SPI_Slave

```
module spi_master
(
     input clk,
     input start,
     input [7:0] data_out,
     output reg [7:0] data_in,
```

```
     output reg SCLK,
     output reg MOSI,
     input MISO,
     output reg SS
);

reg [1:0] state;
reg [2:0] data_counter;

initial
begin
     data_in <= 0;
     state <= 0;
     SS <= 1;
     MOSI <= 0;
     SCLK <= 0;
     data_counter <= 0;
end

always @(posedge clk)
begin
     case (state)
          0:
          begin
               if (start) // If not transmitting but commanded to
start
               begin
                    state <= 1;

                    SS <= 0;
                    SCLK <= 0;
                    data_counter <= 7;
                    MOSI <= data_out[7];
               end
          end
```

```
          1: // If transmitting
          begin
               if (!SCLK) // If making a posedge
               begin
                    data_in[data_counter] <= MISO;

                    if (!data_counter)
                         state <= 2;
                    else
                         data_counter <= data_counter - 1'b1;
               end
               else
               begin
                    MOSI <= data_out[data_counter];
               end

               SCLK <= ~SCLK;
          end
          2: // If packets sent
          begin
               SCLK <= 0;
               state <= 3;
               MOSI <= 0;
          end
          3: // If ending transmission
          begin
               SS <= 1;
               state <= 0;
          end
     endcase
end

endmodule
```

3. Модуль SPI_Slave

```
module spi_slave
(
     input [7:0] data_out,
     output reg [7:0] data_in,

     input SCLK,
     input MOSI,
     output MISO,
     input SS
);

reg state;
reg [2:0] data_counter;

initial
begin
     data_in <= 0;
     state <= 0;
     data_counter <= 0;
end

always @(posedge SCLK)
begin
     if (!SS)
     begin
          if (!state)
          begin
               state = 1;
               data_counter = 7;
          end

          data_in = { data_in[6:0], MOSI };
          if (data_counter == 0) state = 0;
          data_counter = data_counter - 1;
```

```
     end
end

assign MISO = state;

endmodule

module slave_control
(
     output [5:0] led,

     input SCLK,
     input MOSI,
     output MISO,
     input SS
);

reg [7:0] data_out;
wire [7:0] data_in;

spi_slave s (data_out, data_in, SCLK, MOSI, MISO, SS);

assign led = ~data_in[5:0];

endmodule
```