

Университет ИТМО
Факультет программной инженерии и компьютерной техники
Кафедра вычислительной техники

ЛАБОРАТОРНАЯ РАБОТА № 2 ПО ДИСЦИПЛИНЕ
"СХЕМОТЕХНИКА ЭВМ"

ВАРИАНТ: 2

Выполнили: Айтуганов Д. А.
Чебыкин И. Б.

Группа: Р3301

Проверяющий: Баевских А. Н.

Содержание

1	Цели работы	2
2	Задача	2
3	Структура модулей	2
4	Структурная схема RTL-модели	4
5	Структура тестового окружения	6
6	Временные диаграммы	7
7	Вывод	7

1 Цели работы

1. Знакомство с принципами работы последовательных интерфейсов ввода/вывода: I2C, SPI.
2. Изучение основ разработки аппаратных контроллеров периферийных устройств.
3. Изучение основ работы с цифровыми датчиками.

2 Задача

Разработать контроллер датчика освещенности PmodALS.

Контроллер должен реализовывать следующие функции:

- Поддерживать обмен данными с датчиком освещенности посредством интерфейса SPI
- При значении переключателя $SW[0] = 0$ показывать факт наличия света в аудитории с помощью светодиодов: $LED[15:0] = 0xFFFF$ – свет выключен, $LED[15:0] = 0x0000$ – свет включен. Таким образом, все светодиоды должны загораться, когда свет в аудитории выключается, и выключаться – когда свет в аудитории есть.
- При значении переключателя $SW[0] = 1$ показывать на светодиодах текущее значение освещенности, считанное с датчика

3 Структура модулей

Листинг 1: src/lab2.v

```
'timescale 1ns / 1ps

module lab2(
    clk,
    sdo,
    switch,
    comp_data,
    sck,
    led,
    cs
);

    input clk;
    input sdo;
    input switch;
    input [7:0] comp_data;

    output sck;
    output [15:0] led;
    output cs;

    wire [7:0] data;
    wire read_flag;

    read_reader (
        .clk(clk),
        .sdo(sdo),
        .data(data),
        .cs(cs),
        .sck(sck),
        .read_flag(read_flag)
    );

    write_led writer (
```

```

        .clk(read_flag),
        .data(data),
        .comp_data(comp_data),
        .switch(switch),
        .led(led)
    );

endmodule

```

Листинг 2: src/read.v

```

`timescale 1ns / 1ps
`define START 4
`define END 12
module read (
    clk,
    sdo,
    data,
    cs,
    sck,
    read_flag
);

    input clk;
    input sdo; //for PmodALS

    output data;
    output cs; //for PmodALS
    output sck; //for PmodALS
    output read_flag;

    reg cs = 0;
    reg read_flag = 0;
    reg[7:0] data = 0;
    reg[3:0] counter = 15;
    reg[5:0] divider = 0;

    assign sck = divider[5];

    always @ (posedge clk)
        divider = divider + 1;

    always @ (sck) begin
        if(counter == 15) begin
            cs = 1;
        end else begin
            cs = 0;
        end
    end;

    always @ (posedge sck) begin
        if(counter != 15)
            counter = counter + 1;
        if(cs)
            counter = 0;

        if(counter == `START)
            read_flag <= 0;
        else if(counter == `END)
            read_flag <= 1;
        if(!read_flag) begin
            data = data << 1;
            data[0] = sdo;
        end
    end;

end;

endmodule

```

Листинг 3: src/write_led.v

```

`timescale 1ns / 1ps

module write_led (

```

```

clk,
data,
comp_data,
switch,
led
);

input clk;
input data;
input comp_data;
input switch;

output led;

wire[7:0] data;
wire[7:0] comp_data;

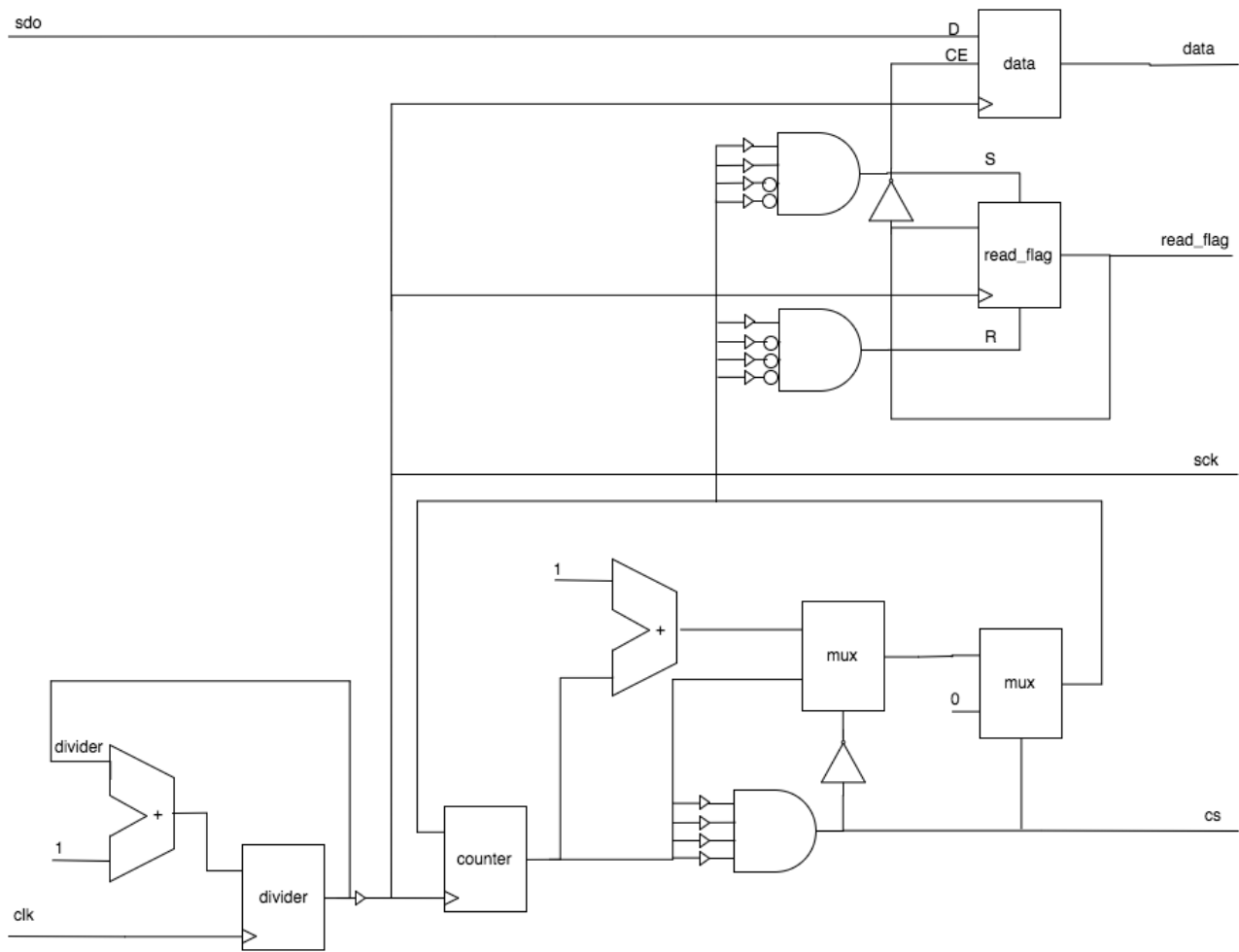
reg[15:0] led = 16'hFFFF;

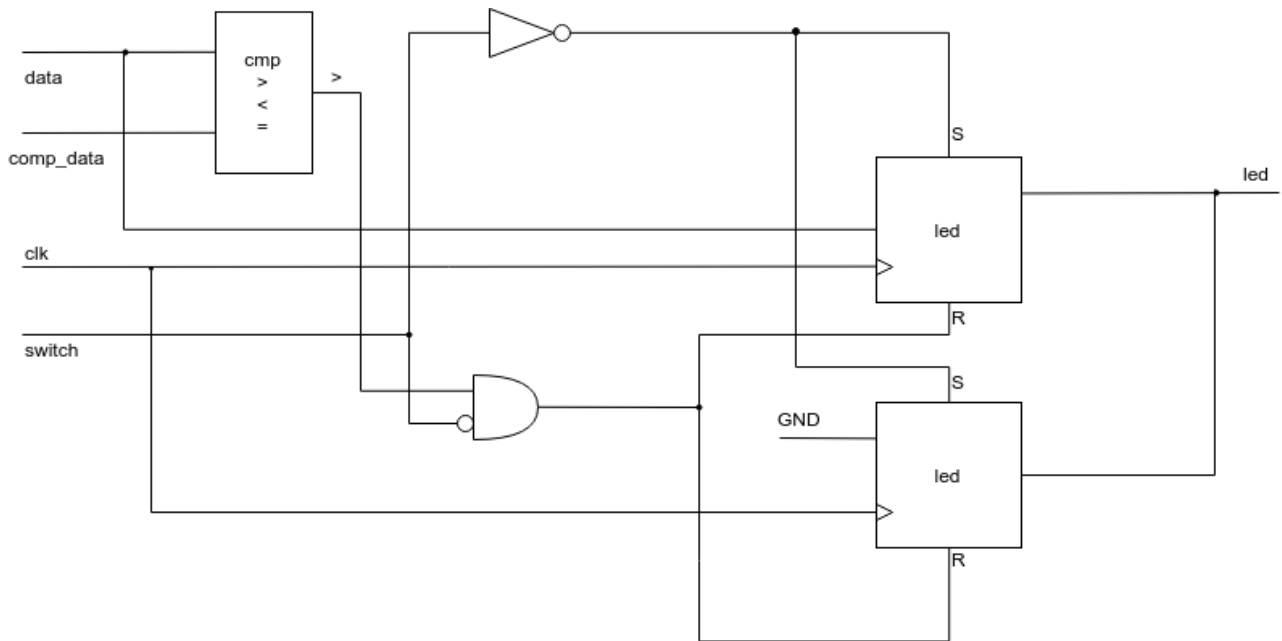
always @(posedge clk) begin
    if( switch == 0 ) begin
        if( data > comp_data )
            led = 16'h0000;
        else
            led = 16'hFFFF;
    end else begin
        led = data;
    end;
end;

endmodule

```

4 Структурная схема RTL-модели





5 Структура тестового окружения

ЛИСТИНГ 4: src/test.v

```

`timescale 1ns / 1ps

module test;

    // Inputs
    reg clk;
    reg sdo;
    reg switch = 0;
    reg pf;
    reg [7:0] comp_data = 8'h0F;
    reg [15:0] test_input = 16'b0001111011100000;
    reg [3:0] i = 15;

    wire [15:0] led;
    wire sck;

    // Instantiate the Unit Under Test (UUT)
    lab2 uut (
        .clk(clk),
        .sdo(sdo),
        .switch(switch),
        .comp_data(comp_data),
        .sck(sck),
        .led(led),
        .cs(cs)
    );

    event sw_trigger;

    initial begin
        forever begin
            @ (sw_trigger);
            @ (posedge clk);
            switch = ~switch;
        end
    end

    initial begin
        // Initialize Inputs
        pf = 0;
        clk = 0;
    end
endmodule

```

```

    switch = 0;
    sdo = 0;
end

always begin
    #0.01 clk = ~clk;
end

always begin
    #100 switch <= ~switch;
end

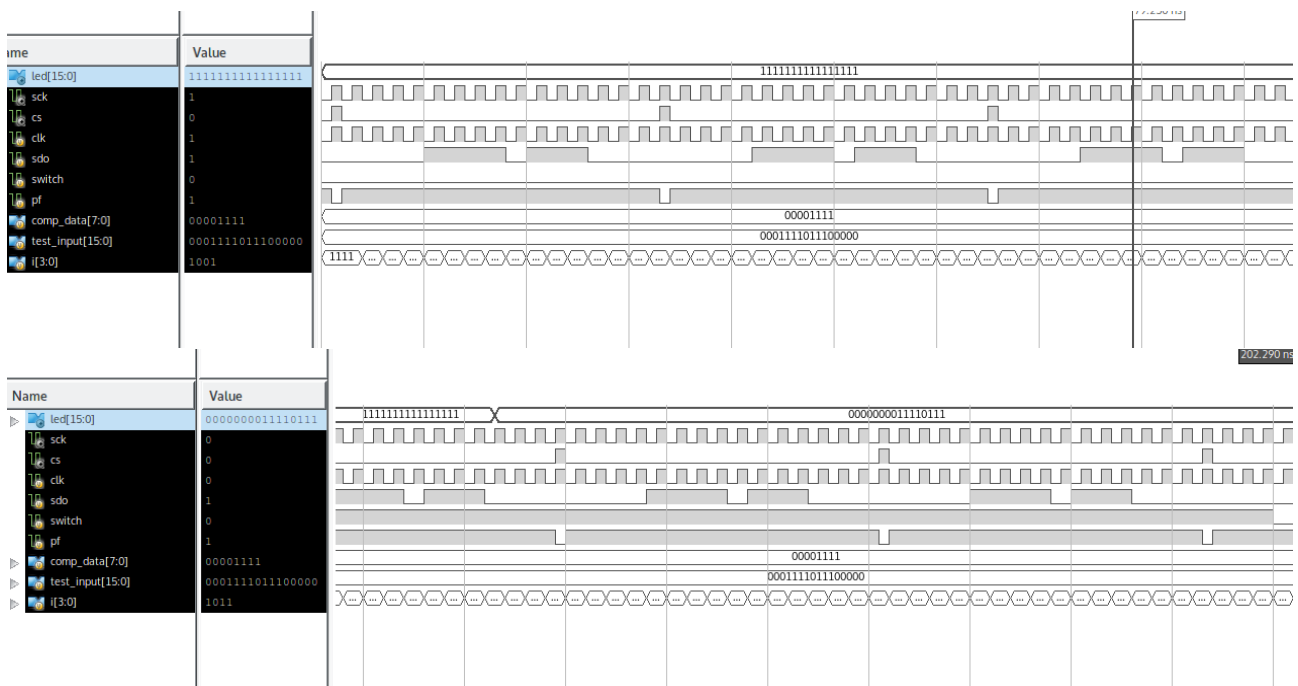
always @(posedge cs)
    pf = 0;

always @(negedge cs)
    pf = 1;

always @(negedge sck) begin
    if(pf && i > 0) begin
        sdo = test_input[i];
        i = i - 1;
    end else
        i = 15;
end
endmodule

```

6 Временные диаграммы



7 Вывод

В ходе данной лабораторной работы мы ознакомились с принципами работы последовательных интерфейсов ввода/вывода на примере SPI, а также мы изучили основы работы с цифровыми датчиками и научились разрабатывать аппаратные контроллеры периферийных устройств.