Университет ИТМО

Факультет программной инженерии и компьютерной техники
Кафедра вычислительной техники

Лабораторная работа № 3 по дисциплине "Схемотехника ЭВМ"

Вариант: 2

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

Группа: P3301

Проверяющий: Баевских А. Н.

СПб, 2016

Содержание

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

1 Структура модулей

```
module ioctrl_wb #(parameter BASE_ADDR = 32'h00000800) (

    // system signals
    input clk_i,
    input rst_i,

    // wb signals
    input      [31:0] dat_i,
    output reg [31:0] dat_o,
    input      [31:0] adr_i,
    input      we_i,
    input      [3:0] sel_i,
    input      cyc_i,
    input      stb_i,
    output reg ack_o,

   // spi signals
   input sdo_i,
   output sck_o,
   output cs_o
);

localparam IDLE = 0;
localparam ACK  = 1;
wire read_flag;

// Request to read from slave
wire read  = cyc_i & stb_i & !we_i;

// Request to write to slave
wire write = cyc_i & stb_i & we_i;

reg state_r;
wire[7:0] data_r;

spi_read spi (
   .clk(clk_i),
  .sdo(sdo_i),
  .reset(rst_i),
  .data(data_r),
  .cs(cs_o),
  .sck(sck_o),
  .read_flag(read_flag)
);

always@(posedge clk_i, posedge rst_i)
    if(rst_i) begin
        state_r <= 0;
        ack_o   <= 1'b0;
        dat_o   <= 0;
    end else begin

        ack_o <= 1'b0;
```

```
        case(state_r)
            IDLE:
                begin
                    if(read && read_flag) begin
                        dat_o <= (adr_i == BASE_ADDR)? data_r: 0;
                        state_r <= ACK;
                    end
                end
            ACK:
                begin
                    ack_o   <= 1'b1;
                    state_r <= IDLE;
                end
        endcase
    end

endmodule
```

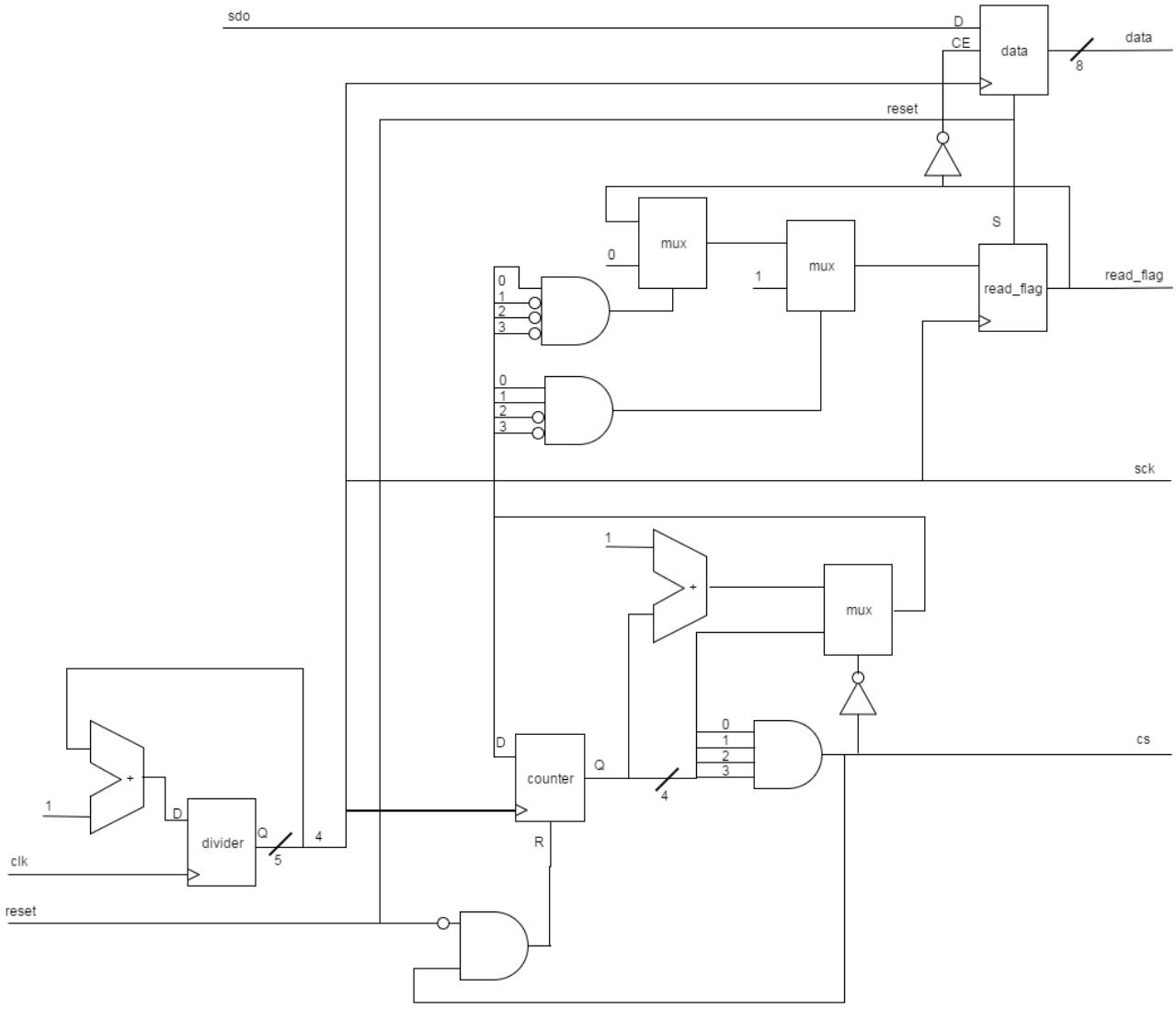
Листинг 1: src/hdl/ioctrl_wb.v

```
`timescale 1ns / 1ps
`define START 4
`define END 12

module spi_read (
   clk,
  sdo,
  reset,
  data,
  cs,
  sck,
  read_flag
    );

  input clk;
  input sdo; //for PmodALS
  input reset;

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

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

  assign sck = divider[5];
  assign cs = counter == 15? 1: 0;

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

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

    if(reset) begin
      data = 0;
      read_flag = 1;
    end else begin
      if(counter == `START)
        read_flag = 0;
      if(counter == `END)
        read_flag = 1;

      if(!read_flag) begin
        data = data << 1;
        data[0] = sdo;
      end

      if(cs)
        counter = 0;
    end
```

```
  end;

endmodule
```

Листинг 2: src/hdl/spi_read.v

2 Структурная схема RTL-модели



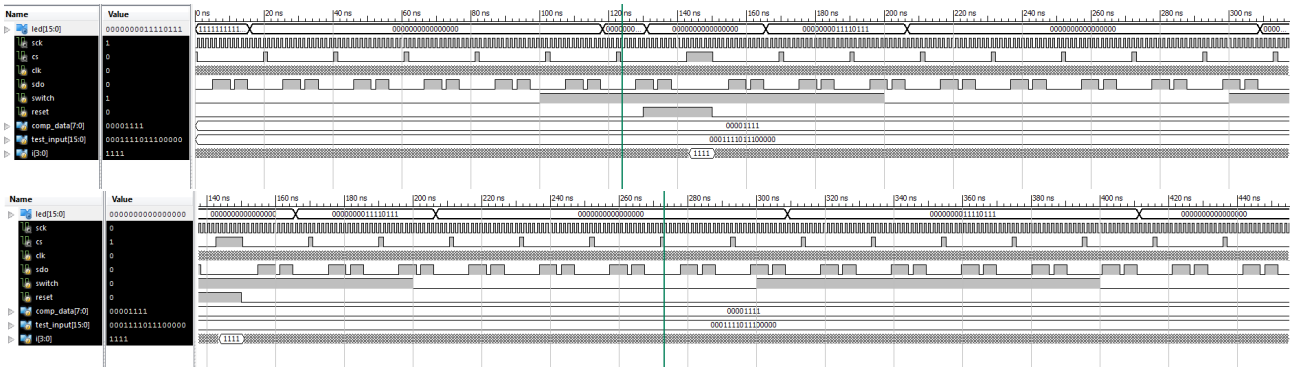
3 Структура тестового окружения

```
‘timescale 1ns / 1ps


module testbench;

      //Inputs
      reg mips_clk;
      reg mips_rst;
      reg [15:0] sw;


      //Outputs
      wire [15:0] led;

     // SPI ports
```

```
    reg    sdo;
    wire  sck;
    wire  cs;

    // SPI Test
    reg [15:0] test_input = 16'b0001111011100000;
    reg [3:0] spi_i = 15;

     //Instantiate the Unit Under Test (UUT)
     mips_system uut (
          .clk(mips_clk),
          .rst(mips_rst),

          .sw(sw),
          .led(led),

       .sdo_i(sdo),
       .sck_o(sck),
       .cs_o(cs)
     );

     initial begin
          mips_rst = 1;

          // Wait 100 ns for global reset to finish
          #100;
          mips_rst = 0;


     end // initial begin

     initial begin
          mips_clk = 0;
       sdo = 0;
          sw = 7;
          forever
               #0.1 mips_clk = !mips_clk;
     end

  integer i;

     initial begin
          for (i = 0; i < 1000000; i=i+1)
               @(posedge mips_clk);

          $stop();
     end

     initial begin
          $display("Trace register $t0");

          @(negedge mips_rst);

          forever begin
               @(posedge mips_clk);

               $display("%d ns: $t0 (REG8) = %x", $time, uut.pipeline_inst.idecode_inst.
    regfile_inst.rf[8]);
               $display("%d ns: $t1 (REG9) = %x", $time, uut.pipeline_inst.idecode_inst.
    regfile_inst.rf[9]);
          end
     end

   always @(negedge sck) begin
     if(!cs && spi_i > 0) begin
       sdo = test_input[spi_i];
       spi_i = spi_i - 1;
     end else
       spi_i = 15;
  end
endmodule
```

Листинг 3: src/hdl/testbench.v

4 Временные диаграммы



5 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.