«Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ

О Т Ч Е Т

по лабораторной работе № 2

Дисциплина: Схемотехника

Название лабораторной работы: Проектирование цифровых устройств на основе ПЛИС

Вариант № 14

Студент гр. ИУ6-53Б _____ **В.К. Залыгин**
(Подпись, дата) (И.О. Фамилия)

Преподаватель _____ **М. Гейне**
(Подпись, дата) (И.О. Фамилия)

Москва, 2024

Цель работы

Закрепление на практике теоретических сведений, полученных при изучении методики проектирования цифровых устройств на основе программируемых логических интегральных схем (ПЛИС), получение необходимых навыков работы с системой автоматизированного проектирования ISE WebPack устройств на основе ПЛИС фирмы Xilinx, изучение аппаратных и программных средств моделирования, макетирования и отладки устройств на основе ПЛИС.

Выполнение работы

На рисунке 1 приведена функциональная схема устройства.

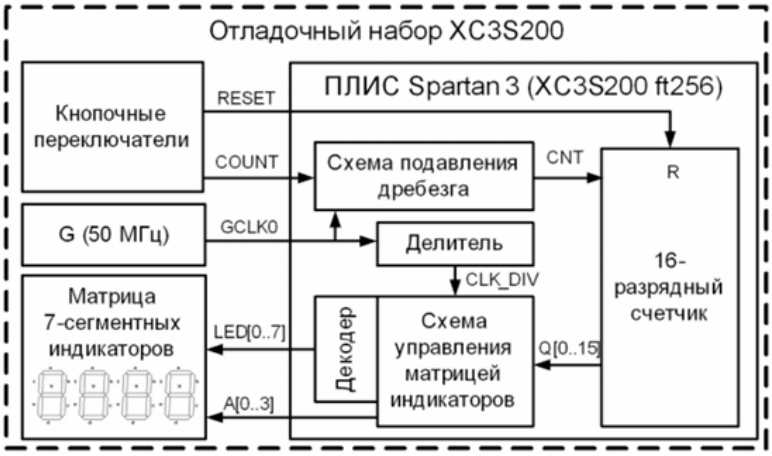

Рисунок 1 – Функциональная схема устройства

На рисунке 2 приведена диаграмма состояний автомата подавления дребезга.

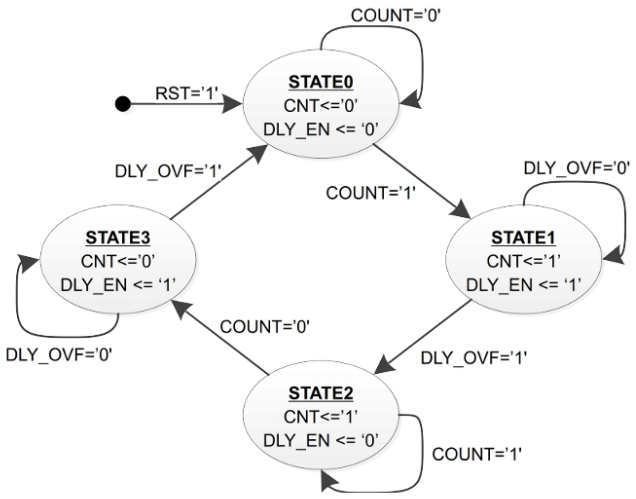

Рисунок 2 - Диаграмма состояния автомата подавления дребезга

Задание 1

Задание: выполнить кодирование состояний автомата, представленного на рисунке 3, в соответствии с индивидуальным вариантом из таблицы 1.

Таблица 1 – Вариант 14

Вариант	Набор	Двоичный код состояния S(1), S(0)			
		State0	State1	State2	State3
14	XC3S200	10	00	11	01

В таблице выходов 2 определены функции сигналов управления в соответствии с заданным вариантом.

Таблица 2 – Таблица выходов

Состояние	State0	State1	State2	State3
Двоичный код состояния	1 0	0 0	1 1	0 1
CNT	0	1	1	0
DLY_EN	0	1	0	1

Заполненная таблица 3 иллюстрирует значение сигналов SN0 и SN1 в соответствии с различными этапами работы устройства.

Таблица 3 – Сигналы SN(*) и D*

COUNT	DLY_OVF	S1(t)	S0(t)	S1(t+1)	S0(t+1)	SN(1)	SN(0)	Описание события
0	X	1	0	1	0	1	0	Ожидание нажатия кнопки
1	X	1	0	0	0	0	0	Нажатие кнопки
X	0	0	0	0	0	0	0	Ожидание окончания счета
X	1	0	0	1	1	1	1	Конец счета
1	X	1	1	1	1	1	1	Ожидание отпускания
0	X	1	1	0	1	0	1	Отпускание кнопки
X	0	0	1	0	1	0	1	Ожидание окончания счета
X	1	0	1	1	0	1	0	Конец счета

Таблицы 4 и 5 демонстрируют карты Карно для минимизации функций SN0 и SN1.

Таблица 4 – Минимизация SN0

S1, S0 / CNT, DLY_OVF	00	01	11	10
00		1	1	
01	1		1	
11	1		1	
10		1	1	

Тогда SN0 = (DLY_OVF && ~ S1 && ~S0) || (~DLY_OVF && ~S1 && S0) || (S1 && S0).

Таблица 5 – Минимизация SN1

S1, S0 / CNT, DLY_OVF	00	01	11	10
00				1
01	1	1		1
11	1	1	1	
10			1	

Тогда SN1 = (DLY_OVF && ~S1) || (COUNT && S1 && S0) || (~COUNT && S1 && ~S0).

Задание 2

Задание: разработайте текстовое описание модуля в соответствии с полученными функциями DLY_EN, CNT, SN[0], SN[1] на основе шаблона.

Выполнение данного задание приведено в листинге 1.

Листинг 1 – Описание модуля подавления дребезга

```
//     10 .
module lab2_example (
     input rst, //
     input clk, //
     input count, //
     output wire cnt, // ,
     output wire[1:0] s_out //
     );
     //
     localparam STATE0 = 2'b10;
     localparam STATE1 = 2'b00;
     localparam STATE2 = 2'b11;
     localparam STATE3 = 2'b01;
     //     t
```

```
     reg[1:0]  s;
     //      t+1
     wire[1:0]  sn;
     reg [20:0] counter; //  2^20
     wire dly_ovf; // " "
     wire dly_en;//
     assign s_out = s;
     //
     always @(posedge clk)
          if(rst)
               s <= STATE0;
          else
               s <= sn;
          //      CNT  DLY_EN (  )
     assign cnt = s[0] == s[1];
     assign dly_en = ~s[1];
          //      (  )
     assign sn[1] = (dly_ovf & ~s[1]) | (count & s[1] & s[0]) |
(~count & s[1] & ~s[0]);
     assign sn[0] = (dly_ovf & ~s[1] & ~s[0]) | (~dly_ovf & ~s[1] &
s[0]) | (s[1] & s[0]);
          //
     always @(posedge clk)
          if(rst || (dly_en == 1'b0))
               counter <= 1'b0;
          else
               counter <= counter + 1;

     assign dly_ovf = (counter == 2**20); //
endmodule
```

Также был создан проект для ПЛИС заданной вариантом, что показано на рисунке 3.

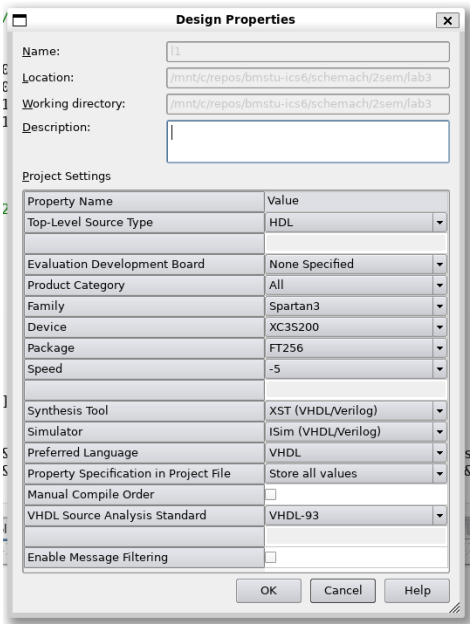

Рисунок 3 – Созданный проект

Задание 3

Задание: в интегрированном редакторе тестов САПР Xilinx ISE разработать тест для полученного устройства и выполнить моделирование его работы в симуляторе Isim.

Код теста приведен в листинге 2.

Листинг 2 – Код теста

```
`timescale 10ns/1ns
module main_tb();
    reg clk=1'b0, count=1'b0, rst=1'b1;
    wire [3:0] a;
    wire [7:0] led;
    wire [1:0] state;
    main uut (
         .clk,
         .count,
         .rst,
         .a,
         .led,
         .state
    );

    task click();
        begin
            @(posedge clk) count = #1 1'b1;
            @(posedge clk) count = #1 1'b0;
            @(posedge clk) count = #1 1'b1;
            @(posedge clk) count = #1 1'b0;
            @(posedge clk) count = #1 1'b1;
            @(posedge clk) count = #1 1'b0;
            @(posedge clk) count = #1 1'b1;
            #1000
            @(posedge clk) count = #1 1'b0;
            @(posedge clk) count = #1 1'b1;
            @(posedge clk) count = #1 1'b0;
            @(posedge clk) count = #1 1'b1;
            @(posedge clk) count = #1 1'b0;
            @(posedge clk) count = #1 1'b1;
            @(posedge clk) count = #1 1'b0;
        end
    endtask
    always #10 clk = ~clk;

    initial begin
        #200 rst = 1'b0;
        #1000;
        click;
        #1000;
        click;
        #1000;
```

```
        click;
        #1000;
        click;
        #1000;
        $finish;
    end
endmodule
```

При запуске симуляции была получена картина, представленная на рисунке 4. Состояния изменяются в заданной вариантом последовательности.

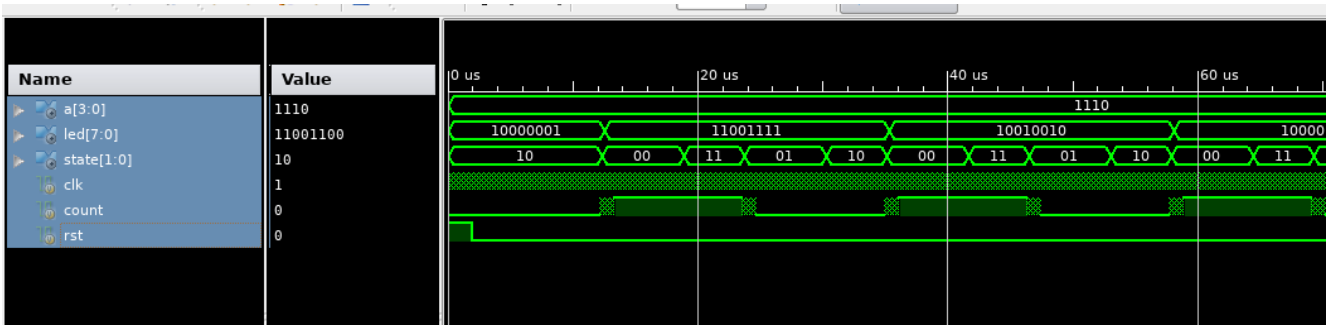

Рисунок 4 – Результат симуляции

Задание 4

Разработать устройство управления, принимающее 16-разрядное информационное слово Q[0..15] и управляющее их последовательной выдачей по шине D[0..3] на декодер 7-сегментных индикаторов в соответствии с показанной на рисунке 6 диаграммой.

Код модуля приведен в листинге 3.

Листинг 3 – Код модуля

```
module seven_segment_driver (
     input clk,
     input clk_div,
     input [15:0] q,
     input rst,
     output [3:0] d,
     output reg [3:0] a
);

     always @(posedge clk)
          if (rst)
               a<=4'b1110;
          else
               if (clk_div)
                    a <= {a[2:0], a[3]};
```

```
assign d[0] = q[0] & ~a[0] | q[4] & ~a[1] | q[8] & ~a[2]  | q[12] &
~a[3];
assign d[1] = q[1] & ~a[0] | q[5] & ~a[1] | q[9] & ~a[2]  | q[13] &
~a[3];
assign d[2] = q[2] & ~a[0] | q[6] & ~a[1] | q[10] & ~a[2] | q[14] &
~a[3];
assign d[3] = q[3] & ~a[0] | q[7] & ~a[1] | q[11] & ~a[2] | q[15] &
~a[3];

endmodule
```

Код модуля с тестом приведен в листинге 4.

Листинг 4 – Код модуля тестирования

```
module test_seven_seg_driver;
     // Inputs
     reg clk;
     reg clk_div;
     reg [15:0] q;
     reg rst;
     // Outputs
     wire [3:0] d;
     wire [3:0] a;
     // Instantiate the Unit Under Test (UUT)
     seven_seg_driver uut (
          .clk(clk),
          .clk_div(clk_div),
          .q(q),
          .rst(rst),
          .d(d),
          .a(a)
     );
  parameter clk_period = 20;
  initial begin
    clk = 0;
    forever #(clk_period/2) clk = ~clk;
  end
  // clk_div generation (более реалистичный пример)
  initial begin
    clk_div = 0;
    forever #(clk_period*2) clk_div = ~clk_div; // Меняется каждые 2
такта clk
  end
  initial begin
    rst = 1;
    #(clk_period*2);
    rst = 0;
  end
  initial begin
  #(clk_period*2);
    q = 16'h0000;
    repeat (16) begin
        q = q + 1;
```

```
        #(clk_period*4); // wait
    end
  end
endmodule
```

Результат моделирования приведен на рисунке 5.

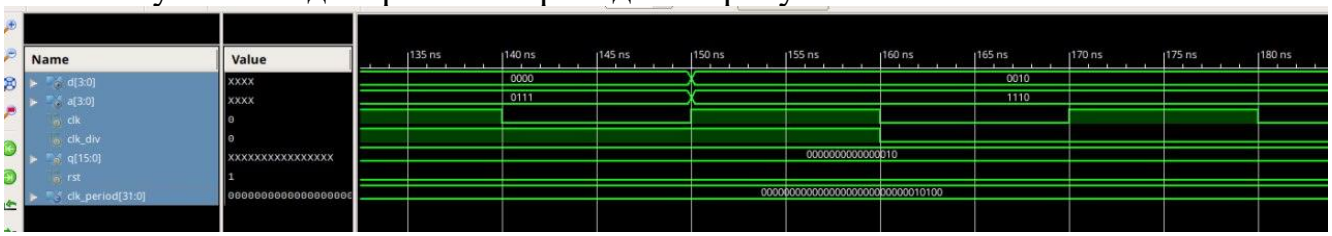

Рисунок 5 – Результат моделирования

Задание 5

Разработать поведенческое Verilog описание схемы преобразования четырехразрядного информационного кода D[0..3] в код активизации 7 - сегментного индикатора LED[0..7] в соответствии с таблицей 4.

Код для данного задания приведен в листинге 5.

Листинг 5 – Код модуля преобразования

```
module led_decode (
input [3:0] dh,
output reg [7:0] seg_data
);
always @(dh)
     case (dh)
          4'b0000: seg_data = 8'b10000001;
          4'b0001: seg_data = 8'b11001111;
          4'b0010: seg_data = 8'b10010010;
          4'b0011: seg_data = 8'b10000110;
          4'b0100: seg_data = 8'b11001100;
          4'b0101: seg_data = 8'b10100100;
          4'b0110: seg_data = 8'b10100000;
          4'b0111: seg_data = 8'b10001111;
          4'b1000: seg_data = 8'b10000000;
          4'b1001: seg_data = 8'b10000100;
          4'b1010: seg_data = 8'b10001000;
          4'b1011: seg_data = 8'b11100000;
          4'b1100: seg_data = 8'b10110001;
          4'b1101: seg_data = 8'b11000010;
          4'b1110: seg_data = 8'b10110000;
          4'b1111: seg_data = 8'b10111000;
          default: seg_data = 8'bx;
     endcase
endmodule
```

Задание 6

В редакторе схем САПР ISE добавить исходное описание, указав путь к файлу main.v.

Код файла main.v приведен в листинге 6.

Листинг 6 – Код main.v

```
module main (
     input clk,
     input count,
     input rst,
     output [3:0] a,
     output [7:0] led,
     output [1:0] state
);
    wire cnt;
    lab2_example lab2_example_inst (
        .clk(clk),
        .rst(rst),
        .count(count),
        .cnt(cnt),
        .s_out(state)
    );

     //
     reg [16:0] counter;
     wire counter_ovf = (counter == 2**16);
     always @(posedge clk)
          if(rst || counter_ovf)
               counter <= 16'b0;
          else
               counter <= counter + 1;

     //   CNT
    reg cnt_ff;
    wire cnt_rise = (cnt==1'b1) && (cnt_ff==1'b0); /*     */
    always @(posedge clk)
        if(rst)
            cnt_ff <= 1'b0;
        else
            cnt_ff <= cnt; /*    */

    //
    reg [15:0] main_counter;
    always @(posedge clk)
        if(rst)
            main_counter <= 1'b0;
        else
            if(cnt_rise)
                main_counter <= main_counter + 1;

    wire [3:0] driver_decoder_bus;
    seven_segment_driver ssd_inst (
        .clk(clk),
```

```
        .rst(rst),
        .q(main_counter),
        .clk_div(counter_ovf), /*    */
        .d(driver_decoder_bus), /*   */
        .a(a)
    );

    led_decode led_decode_inst (
        .dh(driver_decoder_bus), /*      */
        .seg_data(led) /*    */
    );

endmodule
```

Задание 7

В программе Xilinx PACE создать файл ограничений *.ucf или добавьте в проект имеющийся main_xc3s200.ucf. В редакторе необходимо назначить внешние выводы сигналам разрабатываемого устройства в соответствии с таблицей 5.

Результат распиновки приведен на рисунке 6.

```
NET "a[3]" LOC = E13;


NET "a[2]" LOC = F14;
NET "a[1]" LOC = G14;
NET "a[0]" LOC = D14;


NET "led[7]" LOC = P16;
NET "led[6]" LOC = E14;
NET "led[5]" LOC = G13;
NET "led[4]" LOC = N15;
NET "led[3]" LOC = P15;
NET "led[2]" LOC = R16;
NET "led[1]" LOC = F13;
NET "led[0]" LOC = N16;
NET "U[7]" LOC = K13;
NET "U[6]" LOC = K14;
NET "U[5]" LOC = J13;
NET "U[4]" LOC = J14;
NET "U[3]" LOC = H13;
NET "U[2]" LOC = H14;
NET "U[1]" LOC = G12;
NET "U[0]" LOC = F12;
NET "clk" LOC = T9;
NET "count" LOC = M13;
NET "rst" LOC = L14;


NET "C_to_print[7]" LOC = P11;
NET "C_to_print[6]" LOC = P12;
NET "C_to_print[5]" LOC = N12;
NET "C_to_print[4]" LOC = P13;
NET "C_to_print[3]" LOC = N14;
NET "C_to_print[2]" LOC = L12;
NET "C_to_print[1]" LOC = P14;

# PlanAhead Generated physical constraints

NET "C_to_print[0]" LOC = K12;
```

Рисунок 6 – Результат распиновки

Задание 8

В САПР ISE выполнить автоматический синтез технологической схемы, размещение и трассировку полученного устройства на кристалле.

Результат представлен на рисунке 7.

main Project Status (01/13/2025 - 22:15:59)			
Project File:	l1.xise	**Parser Errors:**	No Errors
Module Name:	main	**Implementation State:**	Programming File Generated
Target Device:	xc3s200-5ft256	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	No Warnings
Design Goal:	Balanced	**• Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	All Constraints Met
Environment:	System Settings	**• Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	61	3,840	1%	
Number of 4 input LUTs	39	3,840	1%	
Number of occupied Slices	49	1,920	2%	
Number of Slices containing only related logic	49	49	100%	
Number of Slices containing unrelated logic	0	49	0%	
Total Number of 4 input LUTs	90	3,840	2%	
Number used as logic	39			
Number used as a route-thru	51			
Number of bonded IOBs	17	173	9%	
Number of BUFGMUXs	1	8	12%	
Average Fanout of Non-Clock Nets	2.61			

Рисунок 7 – Design Summary

Задание 9

Выполнить программирование макетной ПЛИС Spartan3 отладочного набора XC3S200 или Nexys2.

Результаты тестирования в виде таблицы приведены в таблице 5.

Таблица 5 – Результаты тестирования

Номер теста	Ожидаемый результат	Полученный результат
1	При нажатии кнопки происходит увеличение значения на 1 на семисегментом индикаторе	При нажатии кнопки происходит увеличение значения на 1 на семисегментом индикаторе
2	При нажатии и удерживании кнопки в течении 10 секунда происходит увеличение значения на 7-сегментном индикаторе на 1	При нажатии и удерживании кнопки в течении 10 секунда происходит увеличение значения на 7-сегментном индикаторе на 1
3	При нажатии кнопки 22 раза на 7-сегментном индикаторе отображается значение 22	При нажатии кнопки 22 раза на 7-сегментном индикаторе отображается значение 22

Вывод

В ходе лабораторной работы были закреплены на практике теоретические сведения, полученных при изучении методики проектирования цифровых устройств на основе программируемых логических интегральных схем (ПЛИС) и получены необходимые навыки работы с системой автоматизированного проектирования ISE WebPack устройств на основе ПЛИС фирмы Xilinx. А также были изучены аппаратные и программные средства моделирования, макетирования и отладки устройств на основе ПЛИС.