|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«*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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ**

**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.

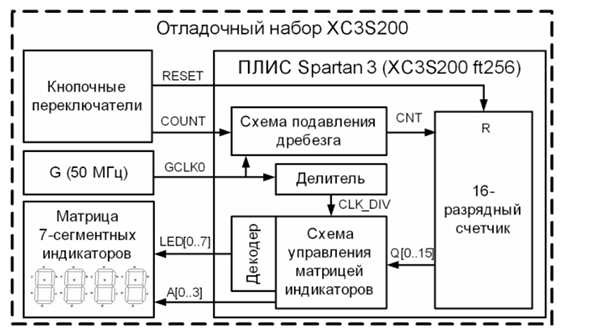


Рисунок 1 – Функциональная схема устройства

На рисунке 2 приведена диаграмма состояний автомата подавления дребезга.

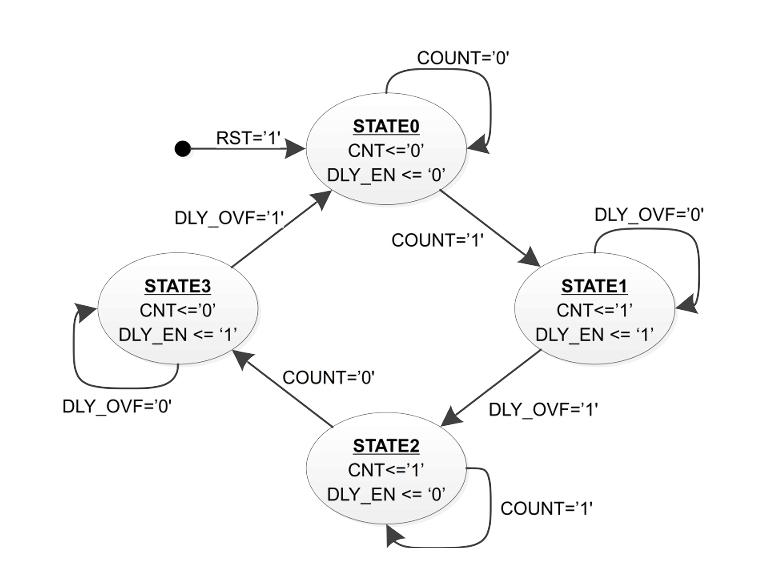


Рисунок 2 - Диаграмма состояния автомата подавления дребезга

**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.

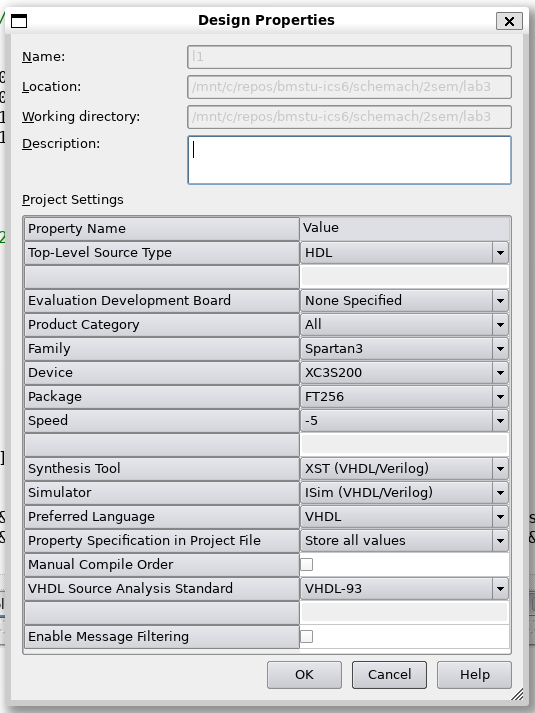


Рисунок 3 – Созданный проект

**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.

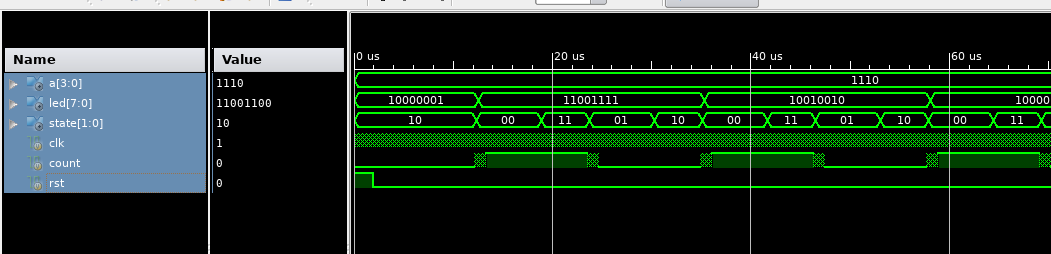


Рисунок 4 – Результат симуляции

**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.

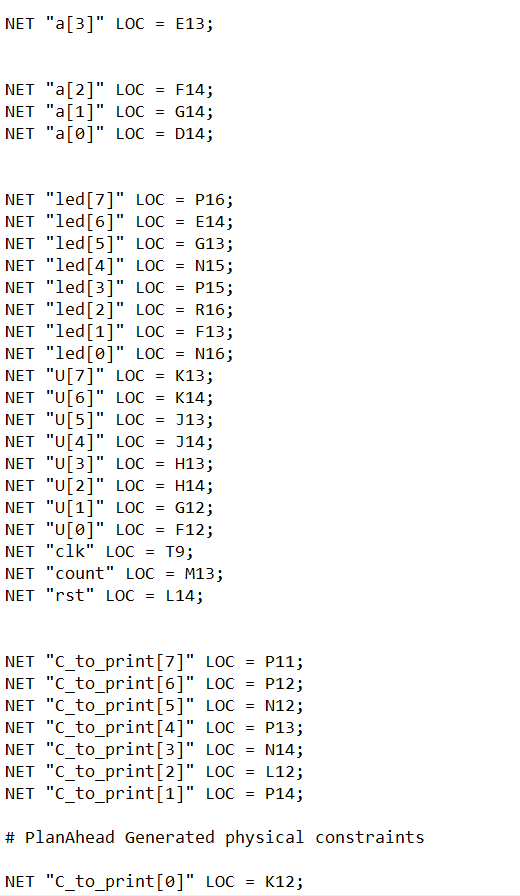


Рисунок 6 – Результат распиновки

**Задание 8**

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

Результат представлен на рисунке 7.

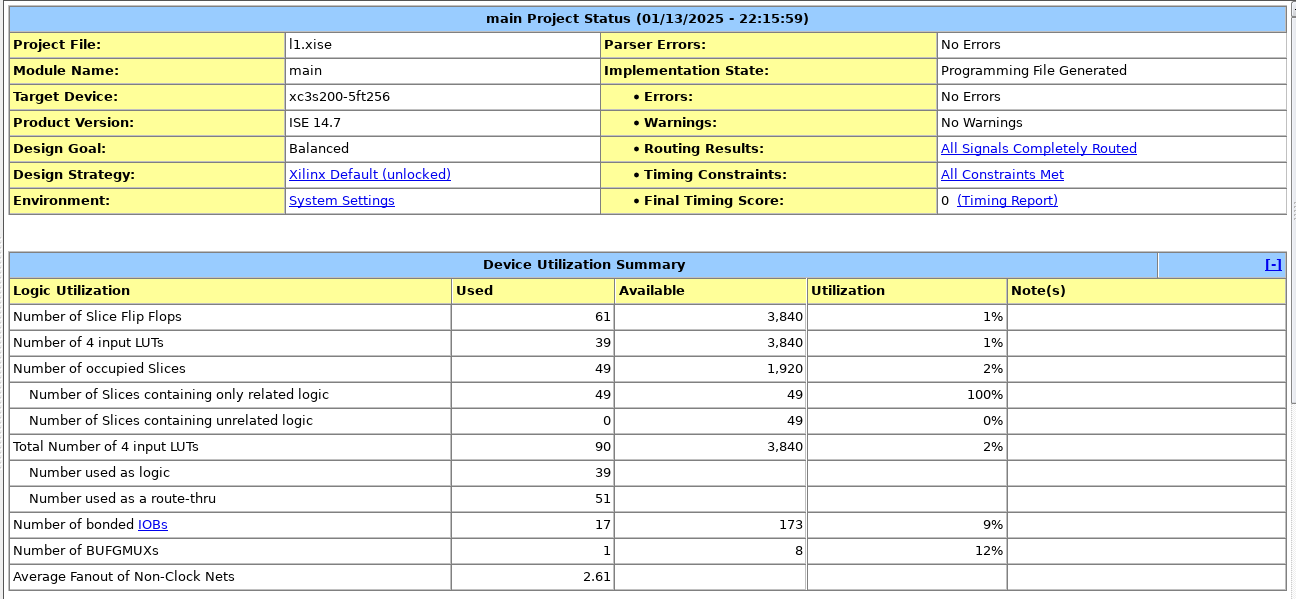


Рисунок 7 – Design Summary

**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.