|  |  |
| --- | --- |
| **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 домашнему заданию**

**Дисциплина: \_Схемотехника\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Название лабораторной работы: Проектирование цифровых устройств на основе ПЛИС**

**Вариант № 57**

Студент гр. \_ИУ6-53Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_А.В. Архипов\_**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_М. А. Гейне\_\_\_\_**

(Подпись, дата) (И.О. Фамилия)

Москва, 2024

**Задание**

Разработать устройство управления схемного типа, принимающее входное командное слово и выдающее сигналы управления операционному блоку в соответствии с приведенной ниже диаграммой переходов.

Разработать модуль для тестирования работы устройства, покрывающий все переходы. Выполнить моделирование устройства.

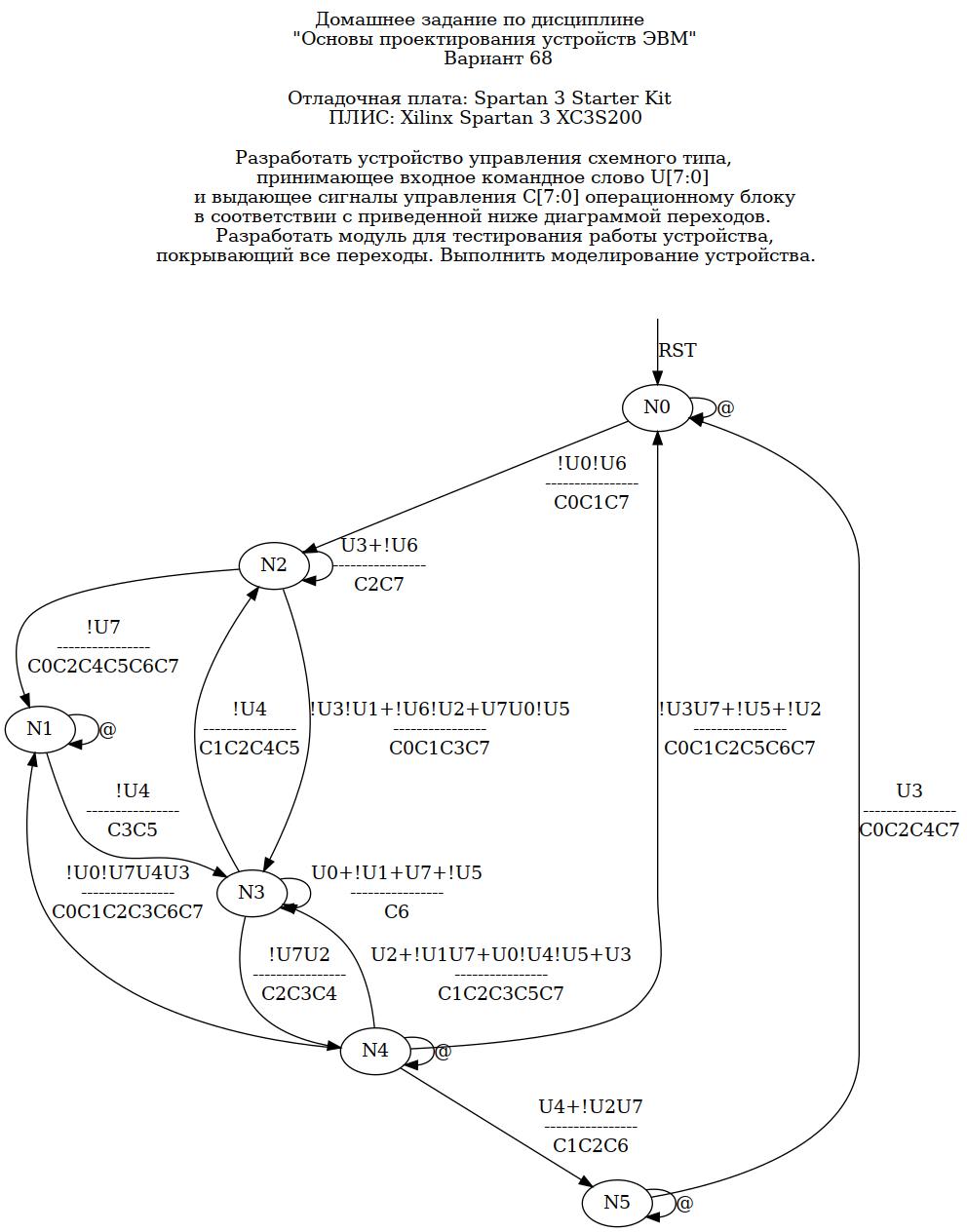


Рисунок 1 – Вариант задания

**Выполнение работы**

По графу переходов автомата определимо, что данный автомат является автоматом Мура, так как выходные сигналы зависят только от текущего состояния автомата.

Ниже в листинге 1 приведен код описания данного автомата на языке Verilog.

Листинг 1 – Описание модуля подавления дребезга

|  |
| --- |
| module automate(  input clk,  input rst,  input [7:0] U,  output reg [7:0] C  );  // Состояния автомата  localparam [2:0] S0 = 3'b000;  localparam [2:0] S1 = 3'b001;  localparam [2:0] S2 = 3'b010;  localparam [2:0] S3 = 3'b011;  localparam [2:0] S4 = 3'b100;  localparam [2:0] S5 = 3'b101;  // Текущее состояние  reg [2:0] state;  // Таблица переходов  always @(posedge clk) begin  if (rst) begin  state = S0;  end else begin  case (state)    S0: begin  if ((U[0] | ~U[4])) begin  state = S1;  end else if (U[5] | U[3] | (U[7] & ~U[2]) | ~U[4]) begin  state = S2;  end else begin  state = S0;  end  end    S1: begin  if (~U[2]) begin  state = S0;  end else begin  state = S1;  end  end    S2: begin  if (U[0] & ~U[2]) begin  state = S0;  end else if (~U[3]) begin  state = S5;  end else begin  state = S2;  end  end    S3: begin  if (~U[7] | U[0] | (~U[2] & ~U[1])) begin  state = S0;  end else begin  state = S3;  end  end    S4: begin  if (~U[0] & U[3] & ~U[4]) begin  state = S0;  end else begin  state = S4;  end  end    S5: begin  if (U[6] | U[7] | (~U[4] & U[0])) begin  state = S2;  end else if (~U[3] | ~U[1] | (U[4] & U[2])) begin  state = S4;  end else if ((U[0] & ~U[4]) | ~U[5] | U[6] | ~U[7] | U[2]) begin  state = S3;  end else begin  state = S5;  end  end  endcase  end  end  // Таблица выходов  always @(state) begin  case (state)  S0: C = 8'b01010111;  S1: C = 8'b10001010;  S2: C = 8'b00101100;  S3: C = 8'b11010111;  S4: C = 8'b10000001;  S5: C = 8'b00000011;  endcase  end  endmodule |

Далее напишем модуль тестов, в котором необходимо покрыть все переходы, существующие в графе переходов автомата.

Для этого построим таблицу переходов.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дуга |  |  |  |  |
|  | 00000001 | 1 | 00000000 | 0 |
|  | 00000000 | 0 | 10000011 | 131 |
|  | 10001100 | 140 | 10000100 | 132 |
|  | 01000110 | 6 | 11110101 | 245 |
|  | 00010000 | 16 | 00000000 | 0 |
|  | 00000000 | 0 | 00101000 | 40 |
|  | 00010000 | 16 | 01000000 | 64 |
|  | 00110110 | 54 | 00011100 | 28 |
|  | 00010000 | 16 | 01000110 | 70 |
|  | 00000000 | 0 | 00000000 | 0 |
|  | 00001000 | 8 | 10010101 | 149 |
|  | 00000000 | 0 | 10000011 | 131 |
|  | 11000000 | 192 | 10001011 | 139 |
|  | 00010110 | 22 | 00011100 | 28 |
|  | 00011000 | 24 | 11001111 | 207 |
|  | 00000000 | 0 | 00101000 | 40 |
|  | 00100010 | 34 | 00110110 | 54 |
|  | 11000000 | 192 | 10001011 | 139 |
|  | 00110110 | 54 | 00011100 | 28 |
|  | 00000000 | 0 | 11100111 | 231 |

Код теста приведен в листинге 2.

Листинг 2 – Код теста

|  |
| --- |
| `timescale 1ns / 1ps  module automate\_test;  // Inputs  reg clk;  reg rst;  reg [7:0] U;  // Outputs  wire [7:0] C;  // Instantiate the Unit Under Test (UUT)  automate uut (  .clk(clk),  .rst(rst),  .U(U),  .C(C)  );  parameter clk\_period = 100;  initial begin  clk = 0;  forever #(clk\_period/2) clk = ~clk;  end  initial begin  // Initialize Inputs  clk = clk\_period;  rst = 0;  // Wait 100 ns for global reset to finish  #100;  // Сбросили ресет, вошли в S0  rst = 1;  #100;  rst = 0;  #100    //Первая петля  U = 'b00010100; //N0 -> N0  #clk\_period;  U = 'b00010001; //N0 -> N1  #clk\_period;  U = 'b00000100; //N1 -> N1  #clk\_period;  U = 'b00000000; //N1 -> N0  #clk\_period;    //Вторая петля  U = 'b00011000; //N0 -> N2  #clk\_period;  U = 'b00000000; //N2 -> N5  #clk\_period;  U = 'b00001110; //N5 -> N3  #clk\_period;  U = 'b00000000; //N3 -> N0  #clk\_period;    //Третья петля  U = 'b00011000; //N0 -> N2  #clk\_period;  U = 'b00000000; //N2 -> N5  #clk\_period;  U = 'b00000000; //N5 -> N4  #clk\_period;  U = 'b00000000; //N4 -> N4  #clk\_period;  U = 'b00001000; //N4 -> N0  #clk\_period;    //Fourth loop  U = 'b00011000; //N0 -> N2  #clk\_period;  U = 'b00000000; //N2 -> N5  #clk\_period;  U = 'b11000000; //N5 -> N2  #clk\_period;  U = 'b00000001; //N2 -> N0  #clk\_period;  end  endmodule |

Далее, запустив симуляцию была получена следующая картина, представленная на рисунке 2.

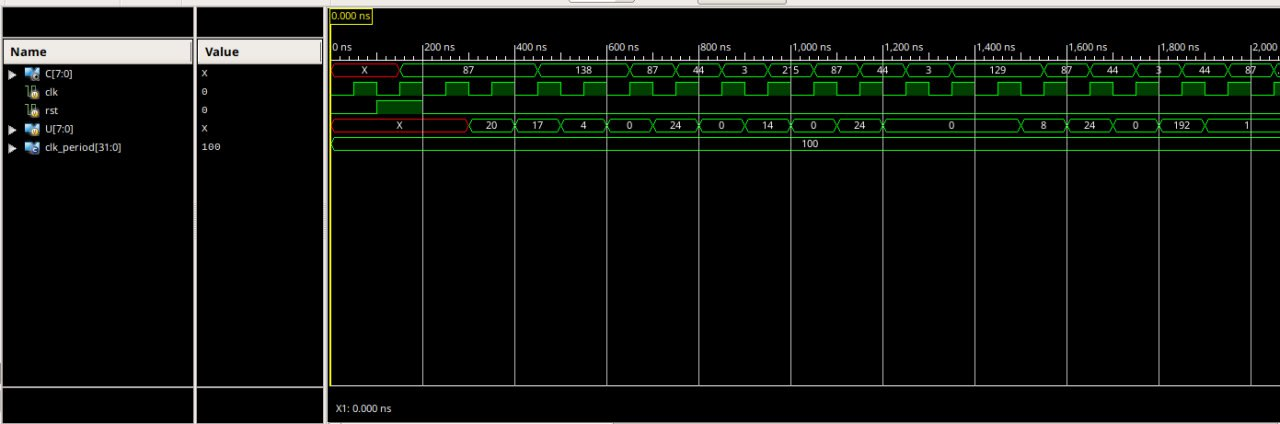


Рисунок 2 – Результат симуляции

**Вывод**

В ходе выполнения домашней работы был написан модуль для реализации автомата по графу переходов автомата, заданному вариантом. Для данного автомата также был написан модуль тестов, покрывающий все переходы между состояниями.