«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 домашнему заданию

Дисциплина: Схемотехника

Название лабораторной работы: Проектирование цифровых устройств на основе ПЛИС

Вариант № 68

Студент гр. ИУ6-53Б _____ **В.К. Залыгин**
(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 $U[7:0]$ и выдающее сигналы управления $C[7:0]$ операционному блоку в соответствии с приведенной ниже диаграммой переходов.

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

Домашнее задание по дисциплине
"Основы проектирования устройств ЭВМ"
Вариант 68

Отладочная плата: Spartan 3 Starter Kit
ПЛИС: Xilinx Spartan 3 XC3S200

Разработать устройство управления схемного типа,
принимающее входное командное слово U[7:0]
и выдающее сигналы управления C[7:0] операционному блоку
в соответствии с приведенной ниже диаграммой переходов.
Разработать модуль для тестирования работы устройства,
покрывающий все переходы. Выполнить моделирование устройства.

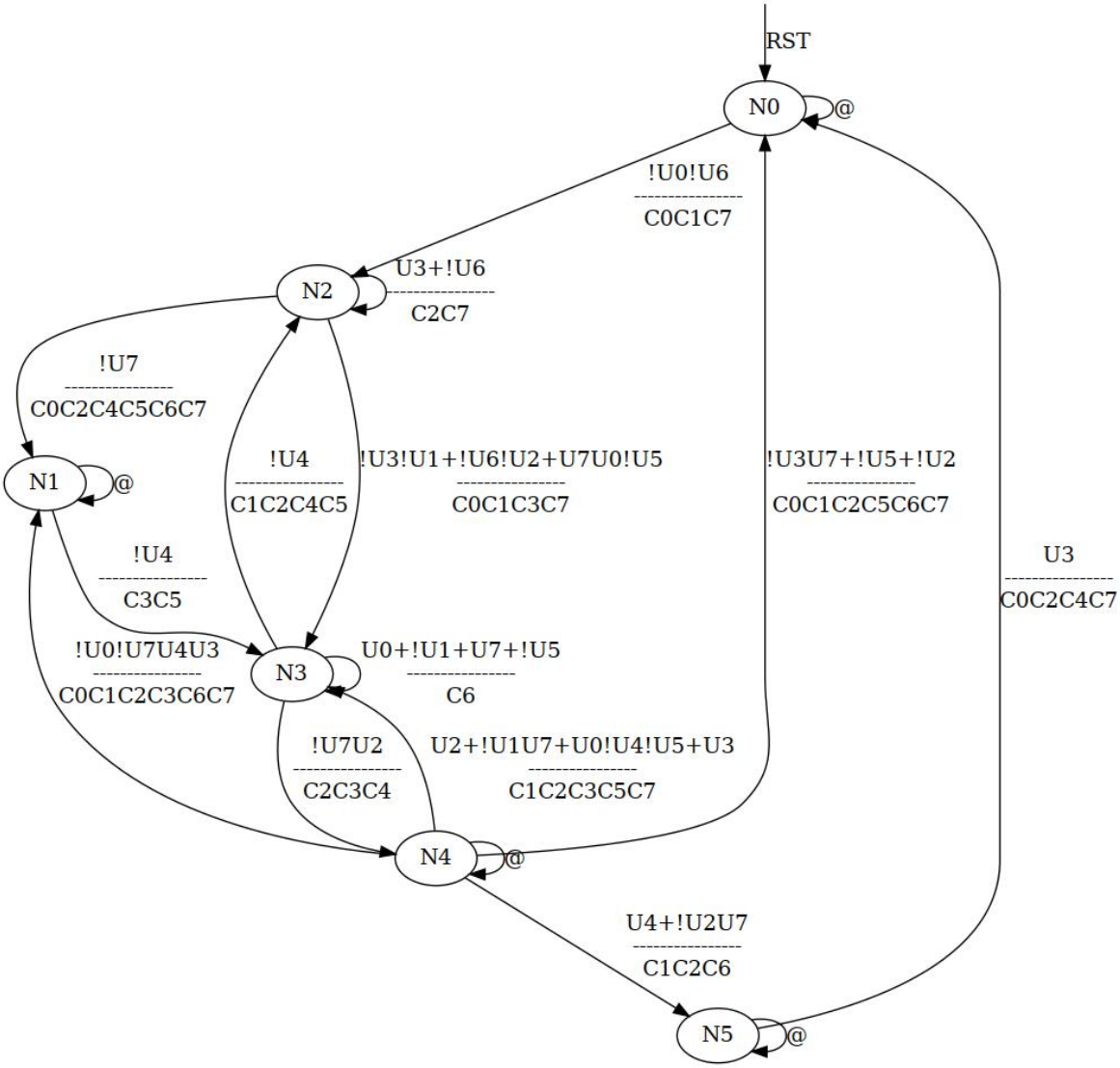

Рисунок 1 – Вариант задания

Выполнение работы

По графу переходов автомата определимо, что данный автомат является автоматом Мили, так как выходные сигналы зависят от состояния автомата и входных сигналов.

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

Листинг 1 – Описание автомата

```
module automata(
          input clk,
          input rst,
          input en,
          input [7:0] U,
          output reg [7:0] C
    );

localparam [2:0] N0 = 3'b000;
localparam [2:0] N1 = 3'b001;
localparam [2:0] N2 = 3'b010;
localparam [2:0] N3 = 3'b011;
localparam [2:0] N4 = 3'b100;
localparam [2:0] N5 = 3'b101;

reg [2:0] state;

always @(posedge clk) begin
     if (rst) begin
          state <= N0;
          C <= 8'b00000000;
     end else
          case (state)
               N0:
               if (~U[0] & ~U[1]) begin
                    state <= N2;
                    C <= 8'b10000011;
               end else begin
                    C <= 8'b00000000;
               end
               N1:
               if (~U[4]) begin
                    state <= N3;
                    C <= 8'b00101000;
               end else begin
                    C <= 8'b00000000;
               end
               N2:
```

```
                if (U[3] | (~U[6])) begin
                     state <= N2;
                     C <= 8'b10000100;
                end else if (~U[7]) begin
                     state <= N1;
                     C <= 8'b11110101;
                end else if ((~U[3] & ~U[1]) | (~U[6] & ~U[2]) |
(U[7] & U[0] & ~U[5])) begin
                     state <= N3;
                     C <= 8'b10001011;
                end
                N3:
                if (~U[4]) begin
                     state <= N2;
                     C <= 8'b00110110;
                end else if (~U[7] & U[2]) begin
                     state <= N4;
                     C <= 8'b00011100;
                end else if (U[0] | ~U[1] | ~U[7] | ~U[5]) begin
                     state <= N3;
                     C <= 8'b01000000;
                end
                N4:
                if (~U[0] & ~U[7] & U[4] & U[3]) begin
                     state <= N1;
                     C <= 8'b11001111;
                end else if (U[4] | (~U[2] & U[7])) begin
                     state <= N5;
                     C <= 8'b01000110;
                end else if (U[2] | (~U[1] & U[7]) | (U[0] & ~U[4] &
~U[5]) | U[3]) begin
                     state <= N3;
                     C <= 8'b10101110;
                end else if ((~U[3] & U[7]) | ~U[5] | ~U[2]) begin
                     state <= N0;
                     C <= 8'b11100111;
                end else begin
                     C <= 8'b00000000;
                end
                N5:
                if (U[3]) begin
                     state <= N0;
                     C <= 8'b10010101;
                end else begin
                     C <= 8'b00000000;
                end
           endcase
end

endmodule
```

Далее написан модуль тестов, в котором покрыты все переходы, существующие в графе переходов автомата.

Для этого построим таблицу переходов 1.

Таблица 1 – Таблица переходов

Дуга	$U_2[7:0]$	U_{10}	$C_2[7:0]$	C_{10}
$N_0 \to N_0$	00000001	1	00000000	0
$N_0 \to N_2$	00000000	0	10000011	131
$N_2 \to N_2$	10001100	140	10000100	132
$N_2 \to N_1$	01000110	6	11110101	245
$N_1 \to N_1$	00010000	16	00000000	0
$N_1 \to N_3$	00000000	0	00101000	40
$N_3 \to N_3$	00010000	16	01000000	64
$N_3 \to N_4$	00110110	54	00011100	28
$N_4 \to N_5$	00010000	16	01000110	70
$N_5 \to N_5$	00000000	0	00000000	0
$N_5 \to N_0$	00001000	8	10010101	149
$N_0 \to N_2$	00000000	0	10000011	131
$N_2 \to N_3$	11000000	192	10001011	139
$N_3 \to N_4$	00010110	22	00011100	28
$N_4 \to N_1$	00011000	24	11001111	207
$N_1 \to N_3$	00000000	0	00101000	40
$N_3 \to N_2$	00100010	34	00110110	54
$N_2 \to N_3$	11000000	192	10001011	139
$N_3 \to N_4$	00110110	54	00011100	28
$N_4 \to N_0$	00000000	0	11100111	231

Код теста приведен в листинге 2.

Листинг 2 – Код теста

```
`timescale 1ns / 1ps
module automata_test;
```

```
reg clk;
reg rst;
reg [7:0] U;
wire [7:0] C;

automata automata1(
     .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
     clk = clk_period;
     rst = 0;
     #100
     rst = 1;
     #100
     rst = 0;
     #100

     // Первая петля
     U = 'b00000001; // N0 -> N0
     #clk_period;
     U = 'b00000000; // N0 -> N2
     #clk_period;
     U = 'b10001100; // N2 -> N2
     #clk_period;
     U = 'b01000110; // N2 -> N1
     #clk_period;
     U = 'b00010000; // N1 -> N1
     #clk_period;
     U = 'b00000000; // N1 -> N3
     #clk_period;
     U = 'b00010000; // N3 -> N3
     #clk_period;
     U = 'b00110110; // N3 -> N4
     #clk_period;
     U = 'b00010000; // N4 -> N5 ??
     #clk_period;
     U = 'b00000000; // N5 -> N5
     #clk_period;
     U = 'b00001000; // N5 -> N0
     #clk_period;

     // Третья петля
     U = 'b00000000; // N0 -> N2
```

```
        #clk_period;
        U = 'b11000000; // N2 -> N3
        #clk_period;
        U = 'b00010110; // N3 -> N4
        #clk_period;
        U = 'b00011000; // N4 -> N1
        #clk_period;
        U = 'b00000000; // N1 -> N3
        #clk_period;
        U = 'b00100010; // N3 -> N2
        #clk_period;
        U = 'b11000000; // N2 -> N3
        #clk_period;
        U = 'b00110110; // N3 -> N4
        #clk_period;
        U = 'b00000000; // N4 -> N0
        #clk_period;

        rst = 1;
end

endmodule
```

Далее, при запуске симуляции была получена следующая картина, представленная на рисунке 2.

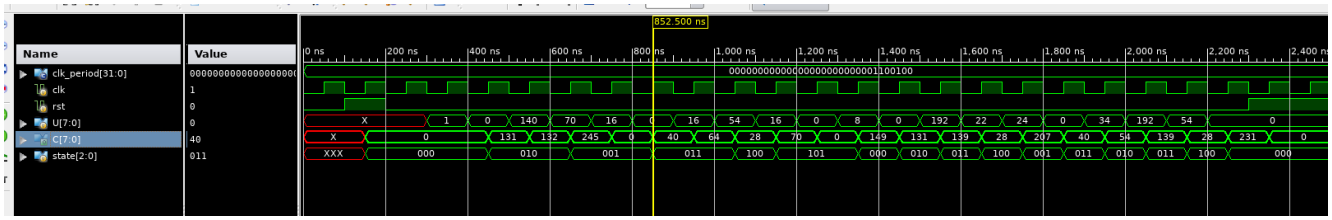

Рисунок 2 – Результат симуляции

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.