|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА - Российский технологический университет»**  **РТУ МИРЭА** |

Институт Информационных Технологий

Кафедра Вычислительной Техники (ВТ)

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3**

«Построение синхронного цифрового автомата»

по дисциплине

«Схемотехника устройств компьютерных систем»

|  |  |
| --- | --- |
| Выполнил студент группы  ИВБО-12-23 | Туктаров Т.А. |
| Принял ассистент кафедры ВТ | Дуксина И.И. |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_2025 г. |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2025 г. |

Москва 2025

АННОТАЦИЯ

Данная работа включает в себя 13 рисунков, 13 листингов. Количество страниц в работе — 34.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc198177667)

[1 ХОД РАБОТЫ 6](#_Toc198177668)

[1.1 Постановка задачи 6](#_Toc198177669)

[1.2 Системная модель 6](#_Toc198177670)

[1.2.1 Создание алгоритма 6](#_Toc198177671)

[1.2.2 Блок-схема алгоритма 6](#_Toc198177672)

[1.2.3 Программная реализация системной модели 7](#_Toc198177673)

[1.2.4 Тестовое покрытие системной модели 9](#_Toc198177674)

[1.3 RTL-модель 10](#_Toc198177675)

[1.3.1 Блок-схема работы цифрового автомата 10](#_Toc198177676)

[1.3.2 Описание автомата на Verilog HDL 11](#_Toc198177677)

[1.3.3 Этап тестирования 13](#_Toc198177678)

[1.4 Реализация основных модулей 14](#_Toc198177679)

[1.4.1 Реализация синхронизатора 14](#_Toc198177680)

[1.4.2 Реализация счётчика 14](#_Toc198177681)

[1.4.3 Реализация триггера 15](#_Toc198177682)

[1.4.4 Реализация делителя частоты 15](#_Toc198177683)

[1.4.5 Реализация модуля управления семисегментными индикаторами 16](#_Toc198177684)

[1.4.6 Реализация модуля фильтра дребезга контактов 16](#_Toc198177685)

[1.4.7 Реализация модуля верхнего уровня 18](#_Toc198177686)

[1.5 Создание и верификация тестового модуля верхнего уровня 21](#_Toc198177687)

[1.6 Создание файла проектных ограничений и загрузка проекта на отладочную плату nexys a7 28](#_Toc198177688)

[ЗАКЛЮЧЕНИЕ 33](#_Toc198177689)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 34](#_Toc198177690)

ВВЕДЕНИЕ

Все комбинационные схемы обладают следующей особенностью: они не имеют эффекта запоминания, т.е. в каждый момент времени значение на выходе меняется в зависимости от значений на входах. Обозначив множество значений логической функции, которая реализована посредством комбинационной схемы, как B и A – множество значений параметров функции [1], можно сказать, что b(t) = f(a(t)), где t – некоторый фиксированный дискретный момент времени. Иными словами, вне зависимости от предыдущего значения функции, новое значение формируется исключительно на основе входных данных.

При переходе от комбинационных схем к цифровым автоматам [2], встаёт вопрос о возможности запоминания информации при изменении входных значений. Пусть будут введены следующие обозначения:

1. А – множество входных символов.
2. B – множество выходных символов.

Для описания цифрового автомата необходимо добавить ещё одно конечное множество, называемое множеством состояний. Под состоянием будет пониматься некоторый «снимок» стабильных значений, составляющих некого объекта, в данном случае - цифрового автомата.

Пусть такое множество будет обозначено как Q. Введение этого множества позволит сохранять «историю» изменения значений, иными словами, разницу в поведении устройства (схемы) при, возможно, одинаковых входных данных можно будет обозначить, используя множество состояний совместно с вышеописанными множествами.

1 ХОД РАБОТЫ

## 1.1 Постановка задачи

Разработать верификационное окружение для цифрового устройства, реализованного на языке описания аппаратуры Verilog HDL. Провести моделирование и верификацию работы устройства согласно выданному варианту. Реализовать тестовый модуль, позволяющий проверить корректность работы проектируемого устройства на различных входных данных. Проанализировать роль верификации на этапах проектирования цифровых систем. Вариант: Решение СЛАУ методом Крамера.[3]

## 1.2 Системная модель

**1.2.1 Создание алгоритма**

Для данной задачи алгоритм с точки зрения достаточно высокого уровня абстракции будет представлен ниже:

1. Ввести элементы матрицы 3х3 и вектора В.
2. Вычислить определитель матрицы 3х3.
3. Если определитель = 0, то вывести код ошибки.
4. Вычислить вектор решения.
5. Вывести результат.

**1.2.2 Блок-схема алгоритма**

На данном этапе, согласно алгоритму, требуется разработать блок-схему. Блок-схема алгоритма представлена на Рисунке 1.1.

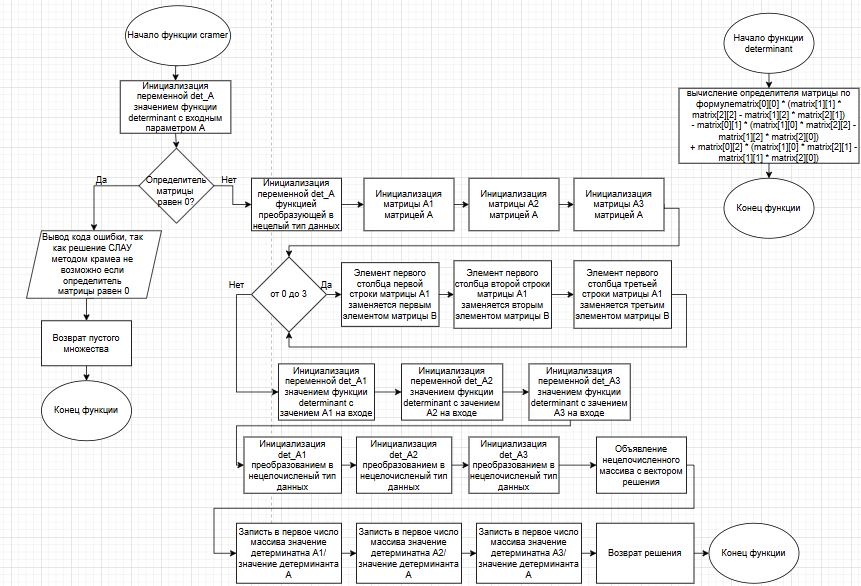


Рисунок 1.1 — Блок-схема алгоритма

**1.2.3 Программная реализация системной модели**

Программная реализация для поставленной задачи согласно системной модели представлена в Листинге 1.1.

Листинг 1.1 — Программная реализация на языке C++

#include <iostream>

#include <vector>

#include <iomanip>

using namespace std;

long long determinant(const vector<vector<long long>>& matrix) {

return matrix[0][0] \* (matrix[1][1] \* matrix[2][2] - matrix[1][2] \* matrix[2][1])

- matrix[0][1] \* (matrix[1][0] \* matrix[2][2] - matrix[1][2] \* matrix[2][0])

+ matrix[0][2] \* (matrix[1][0] \* matrix[2][1] - matrix[1][1] \* matrix[2][0]);

}

vector<double> cramer(const vector<vector<long long>>& A, const vector<long long>& B) {

long long det\_A\_long = determinant(A);

if (det\_A\_long == 0) {

cout << "Код ошибки 01, определитель равен 0." << endl;

return {}; }

double det\_A = static\_cast<double>(det\_A\_long);

vector<vector<long long>> A1 = A;

vector<vector<long long>> A2 = A;

Продолжение листинга 1.1

vector<vector<long long>> A3 = A;

for (int i = 0; i < 3; ++i) {

A1[i][0] = B[i];

A2[i][1] = B[i];

A3[i][2] = B[i];

}

long long det\_A1\_long = determinant(A1);

long long det\_A2\_long = determinant(A2);

long long det\_A3\_long = determinant(A3);

double det\_A1 = static\_cast<double>(det\_A1\_long);

double det\_A2 = static\_cast<double>(det\_A2\_long);

double det\_A3 = static\_cast<double>(det\_A3\_long);

vector<double> reshenie(3);

reshenie[0] = det\_A1 / det\_A;

reshenie[1] = det\_A2 / det\_A;

reshenie[2] = det\_A3 / det\_A;

return reshenie;

}

int main() {

setlocale(LC\_ALL, "RU");

vector<vector<long long>> A(3, vector<long long>(3));

vector<long long> B(3);

cout << "Введите коэффициенты матрицы A (3x3):" << endl;

for (int i = 0; i < 3; ++i) {

for (int j = 0; j < 3; ++j) {

cout << "A[" << i << "][" << j << "]: ";

cin >> A[i][j];

}

}

cout << "Введите свободные члены (вектор B):" << endl;

for (int i = 0; i < 3; ++i) {

cout << "B[" << i << "]: ";

cin >> B[i];

}

vector<double> reshenie = cramer(A, B);

if (!reshenie.empty()) {

cout << "Решение системы:" << endl;

cout << "x = " << fixed << setprecision(2) << reshenie[0] << endl;

cout << "y = " << fixed << setprecision(2) << reshenie[1] << endl;

cout << "z = " << fixed << setprecision(2) << reshenie[2] << endl;

cout << "\nВектор решения: (" << fixed << setprecision(2) << reshenie[0] << ", "

<< fixed << setprecision(2) << reshenie[1] << ", "

<< fixed << setprecision(2) << reshenie[2] << ")" << endl;

}

return 0;

}

**1.2.4 Тестовое покрытие системной модели**

Результаты тестирования представлены в листинге 1.2. Таким образом, были сформированы эталонные наборы значений.

Листинг 1.2 — Результаты работы программной реализации

Введите коэффициенты матрицы A (3x3):

A[0][0]: 1

A[0][1]: -1

A[0][2]: 1

A[1][0]: -1

A[1][1]: 3

A[1][2]: -1

A[2][0]: 1

A[2][1]: -1

A[2][2]: 2

Введите свободные члены (вектор B):

B[0]: 1

B[1]: 1

B[2]: 2

Решение системы:

x = 1

y = 1

z = 1

Вектор решения: (1, 1, 1)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Введите коэффициенты матрицы A (3x3):

A[0][0]: 2

A[0][1]: 2

A[0][2]: 1

A[1][0]: 4

A[1][1]: 1

A[1][2]: 0

A[2][0]: 0

A[2][1]: 2

A[2][2]: 1

Введите свободные члены (вектор B):

B[0]: 1

B[1]: 0

B[2]: 1

Решение системы:

x = 0

y = 0

z = 1

Вектор решения: (0, 0, 1)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Введите коэффициенты матрицы A (3x3):

A[0][0]: 1

A[0][1]: 2

A[0][2]: 3

A[1][0]: 2

A[1][1]: 4

A[1][2]: 6

A[2][0]: 3

Продолжение листинга 1.1

A[2][1]: 6

A[2][2]: 9

Введите свободные члены (вектор B):

B[0]: 1

B[1]: 1

B[2]: 1

Код ошибки 00, определитель равен 0.

## 1.3 RTL-модель

**1.3.1 Блок-схема работы цифрового автомата**

Исходя из рассмотренного выше алгоритма требуется сформировать необходимый набор узлов. Составим блок-схему работы устройства. Реализация блок-схемы представлена на Рисунке 1.2.

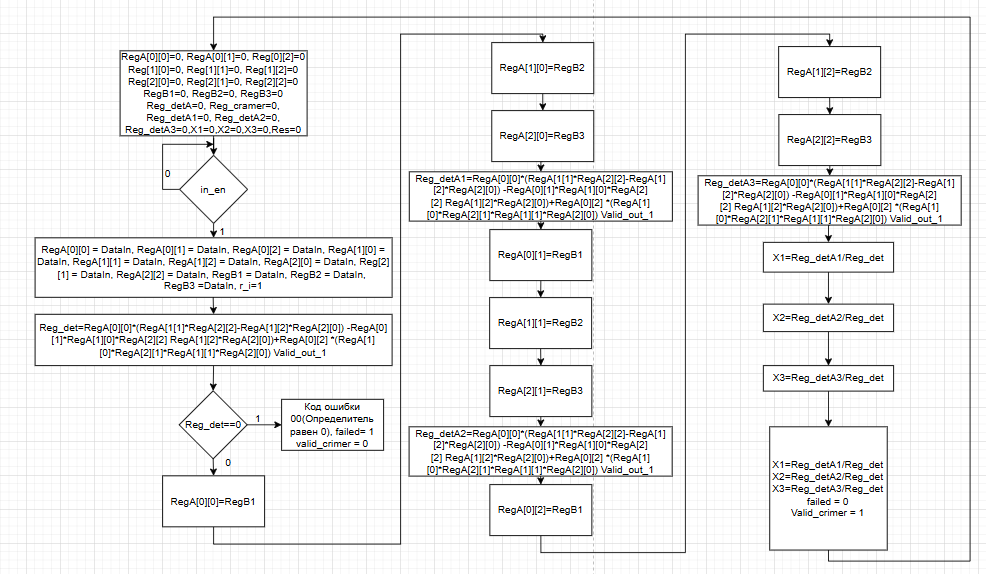


Рисунок 1.2 — Блок-схема работы цифрового автомата

**1.3.2 Описание автомата на Verilog HDL**

Исходный код представлен в Листинге 1.3.

Листинг 1.3 — Программная реализация автомата на Verilog HDL

`timescale 1ns / 1ps

module crimer(

input clk,

input reset,

input input\_en,

input signed [3:0] DataIn,

output reg signed [3:0] X1, X2, X3,

output reg valid\_cramer,

output reg failed,valid\_in,

output reg [2:0] state

);

reg signed [3:0] A[0:2][0:2];

reg signed [3:0] b[0:2];

reg [3:0] counter = 0;

reg signed [3:0] detA, detA1, detA2, detA3;

parameter wait\_start = 0;

parameter read\_data = 1;

parameter calculate = 2;

parameter data\_out = 3 initial

begin

state = wait\_start;

failed<=0;

end

always #5 if(valid\_in) valid\_in = ~valid\_in;

always @(posedge clk) begin

if(reset)

begin

failed <= 0;

valid\_cramer <= 0;

X1 <= 0;

X2 <= 0;

X3 <= 0;

b[0] <= 0;

b[1] <= 0;

b[2] <= 0;

A[0][0] <= 0;

A[0][1] <= 0;

A[0][2] <= 0;

A[1][0] <= 0;

A[1][1] <= 0;

A[1][2] <= 0;

A[2][0] <= 0;

A[2][1] <= 0;

A[2][2] <= 0;

detA<=0;

detA1<=0;

detA2<=0;

detA3<=0;

Продолжение листинга 1.3

counter <= 0;

end

if (input\_en) begin

if (counter < 9) begin

A[counter % 3][counter / 3] <= DataIn;

end

else if (counter < 12) begin

b[counter - 9] <= DataIn;

end

counter <= counter + 1;

valid\_in =1'd1;

end

if (counter == 12) begin

detA = (A[0][0] \* (A[1][1] \* A[2][2] - A[1][2] \* A[2][1]))

- (A[0][1] \* (A[1][0] \* A[2][2] - A[1][2] \* A[2][0]))

+ (A[0][2] \* (A[1][0] \* A[2][1] - A[1][1] \* A[2][0]));

detA1 = (b[0] \* (A[1][1] \* A[2][2] - A[1][2] \* A[2][1]))

- (A[0][1] \* (b[1] \* A[2][2] - A[1][2] \* b[2]))

+ (A[0][2] \* (b[1] \* A[2][1] - A[1][1] \* b[2]));

detA2 = (A[0][0] \* (b[1] \* A[2][2] - A[1][2] \* b[2]))

- (b[0] \* (A[1][0] \* A[2][2] - A[1][2] \* A[2][0]))

+ (A[0][2] \* (A[1][0] \* b[2] - b[1] \* A[2][0]));

detA3 = (A[0][0] \* (A[1][1] \* b[2] - b[1] \* A[2][1]))

- (A[0][1] \* (A[1][0] \* b[2] - b[1] \* A[2][0]))

+ (b[0] \* (A[1][0] \* A[2][1] - A[1][1] \* A[2][0]));for (i = 0; if (detA == 0) begin

failed <= 1;

valid\_cramer <= 0;

end

else begin

X1 <= detA1 / detA;

X2 <= detA2 / detA;

X3 <= detA3 / detA;

valid\_cramer <= 1;

failed <= 0;

end

end

case(state)

wait\_start:

if(input\_en)

state<=read\_data;

else

state<=state;

read\_data:

if(counter == 12)

state <= calculate;

else

state<=state;

calculate:

if(valid\_cramer)

state <= data\_out;

else

state <= state;

Продолжение листинга 1.3

data\_out:

if(reset)

state <= wait\_start;

else

state <= state;

default:

state <= wait\_start;

endcase

end

edmodule

**1.3.3 Этап тестирования**

На данном этапе основной целью является получение значений, идентичных эталонным, а также проверка различных особенностей работы самой схемы. Пример тестового модуля представлен в Листинге 1.4. Результат симуляции представлен на Рисунке 1.3. В ходе тестирования были получены ответы, идентичные эталонным, а сброс автомата в действительности привёл автомат в начальное положение. [4]

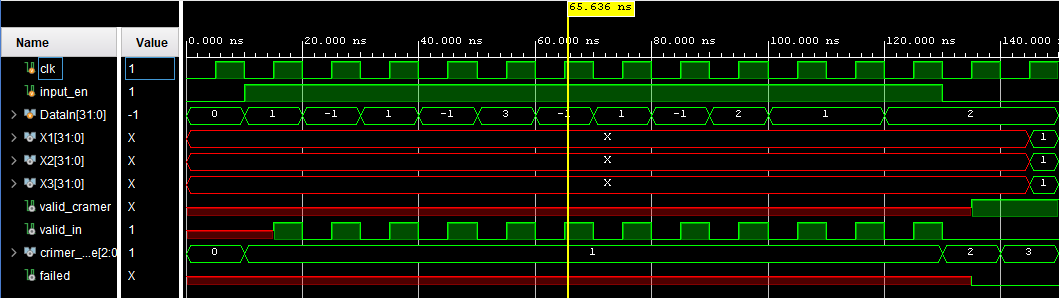


Рисунок 1.3 — Временная диаграмма работы устройства

## 1.4 Реализация основных модулей

**1.4.1 Реализация синхронизатора**

Опишем синхронизатор в модулеs synchro. Реализация представлена в Листинге 1.5.

Листинг 1.5 — Модуль synchro.v

module synchro(

input in\_signal,clk,

output q

);

reg new\_signal = 1'bx;reg last\_signal = 1'bx;wire lq;

dtrigger tr1(.C(clk),.D(new\_signal),

.en(1'b1),.q(lq));

dtrigger tr2(.C(clk),.D(last\_signal),.en(1'b1),.q(q));

always @(posedge clk) begin new\_signal=in\_signal;

last\_signal=lq;end

endmodule

**1.4.2 Реализация счётчика**

Опишем счётчик в модулеs count. Реализация представлена в Листинге 1.6.

Листинг 1.6— Модуль count.v

`timescale 1ns / 1ps

module count# (step = 1, mod = 16) (

input clk,

input dir,

input RE,

input CE,

output reg [$clog2(mod) -1:0] out

);

initial out = 0;

always@(posedge clk or posedge RE)

begin

if (RE)

out <= 0;

else if (CE) begin

if (dir == 0)

out <= (out + step) % mod;

else

out <= (out - step) % mod;

end

end

endmodule

**1.4.3 Реализация триггера**

Опишем d-trigger в модулеs dtrigger. Реализация представлена в Листинге 1.7.

Листинг 1.7 — Модуль dtrigger.v

`timescale 1ns / 1ps

module dtrigger(

input wire C,

input wire D,

input wire en,

output reg q

);

initial begin

q<=0;

end

always @(posedge C) begin

if(en) begin

q <= D;

end

end

endmodule

**1.4.4 Реализация делителя частоты**

Опишем параметризированный делитель частоты в модулеs clk\_divider. Реализация представлена в Листинге 1.8.

Листинг 1.8 — Модуль clk\_divider.v

`timescale 1ns / 1ps

module clk\_divider#(div = 2) (

input clk,

output reg clk\_div

);

reg l;

wire [8:0] out;

count #(.step(1), .mod(div/2)) cntr(

.clk(clk),

.RE(1'b0),

.CE(1'b1),

.dir(1'b0),

.out(out)

);

initial clk\_div = 0;

always@(posedge clk)begin

if (out ==0 && l==1)

clk\_div = ~clk\_div;

l<=1;

end

endmodule

**1.4.5 Реализация модуля управления семисегментными индикаторами**

Опишем модуль управления семисегментными индикаторами в модуле SevenSegmentLED. Реализация представлена в Листинге 1.9.

Листинг 1.9 — Модуль SevenSegmentLED.v

module SevenSegmentLED(

input [7:0] AN\_MASK,

input [31:0] NUMBER,

input clk,

output [7:0] AN,

output reg[7:0] SEG);

wire[2:0] counter\_res;

count #(.mod(8), .step(1)) cntr(

.clk(clk),

.RE(1'b0),

.CE(1'b1),

.dir(1'b0),

.out(counter\_res)

);

reg [7:0] AN\_REG = 0;

assign AN = AN\_REG | AN\_MASK;

wire [3:0] NUMBER\_SPLITTER[0:7]; genvar i;

generate

for (i = 0; i < 8; i = i + 1)

begin

assign NUMBER\_SPLITTER[i] = NUMBER[((i+1)\*4-1)-:4];

end

endgenerate

always @(posedge clk)

begin

case (NUMBER\_SPLITTER[counter\_res])

4'h0: SEG <= 8'b11000000;

4'h1: SEG <= 8'b11111001;

4'h2: SEG <= 8'b10100100;

4'h3: SEG <= 8'b10110000;

4'h4: SEG <= 8'b10011001;

4'h5: SEG <= 8'b10010010;

4'h6: SEG <= 8'b10000010;

4'h7: SEG <= 8'b11111000;

4'h8: SEG <= 8'b10000000;

4'h9: SEG <= 8'b10010000;

4'ha: SEG <= 8'b10001000;

4'hb: SEG <= 8'b10000011;

4'hc: SEG <= 8'b11000110;

4'hd: SEG <= 8'b10100001;

4'he: SEG <= 8'b10000110;

4'hf: SEG <= 8'b10001110;

default: SEG <= 8'b11111111;

endcase

AN\_REG = ~(8'b1 << counter\_res);

end

endmodule

**1.4.6 Реализация модуля фильтра дребезга контактов**

Опишем фильтр дребезга контактов в модулеs filtercon. Реализация представлена в Листинге 1.10.

Листинг 1.10 — Модуль filtercon.v

`timescale 1ns / 1ps

module filtercon #(mode = 2)(

input in\_signal,

input clock\_enable,

input clk,

output wire out\_signal,

output out\_signal\_enable,

output wire [1:0] q\_count

);

wire out\_sync;

wire out\_first\_and;

wire out\_second\_and;

wire out\_third\_and;

synchro syn(

.in\_signal(in\_signal),

.clk(clk),

.q(out\_sync)

);

count cs(

.clk(clk),

.RE(out\_sync~^out\_signal),

.dir(0),

.CE(clock\_enable),

.out(q\_count)

);

dtrigger dt1(

.C(clk),

.D(out\_sync),

.en(out\_second\_and),

.q(out\_signal)

);

dtrigger dt2(

.C(clk),

.D(out\_third\_and),

.en(1'b1),

.q(out\_signal\_enable)

);

assign out\_first\_and = &q\_count;

assign out\_second\_and = out\_first\_and & clock\_enable;

assign out\_third\_and = out\_second\_and & out\_sync;

endmodule

**1.4.7 Реализация модуля верхнего уровня**

Опишем модуль верхнего уровня в модуле main. Реализация представлена в Листинге 1.11.

Листинг 1.11 — Модуль main.v

`timescale 1ns / 1ps

module main(

input [3:0] SWITCHES,

input button\_in, clk, button\_reset\_in,button\_minus,

output [7:0] AN,

output [6:0] SEG,

output reg is\_minus\_last,

output wire is\_failed

output wire reset\_signal,

output reg signed [3:0] Data\_in,

output wire button\_signal\_en,

output wire reset\_signal\_en,

output wire button\_minus\_en,

output wire minus\_signal

output reg [31:0] NUMBER,

output wire button\_signal

);

wire button\_signal, button\_signal\_en, reset\_signal\_en, reset\_signal;

reg[7:0] AN\_MASK = 8'b11111111;

reg [31:0] NUMBER;

wire minus\_signal;

wire button\_minus\_en;

reg signed [3:0] Data\_in;

reg is\_data\_last;

reg is\_minus = 0;

wire [3:0] X1, X2, X3;

wire valid\_cramer;

parameter minus = -1;

reg input\_en = 0;

reg [2:0] n = 0;

reg [3:0] n1 = 0;

reg [3:0] data\_last = 0;

initial begin

NUMBER <= 0;

end

filtercon #(128) dbnc\_minus(

.clk(clk),

.in\_signal(button\_minus),

.clock\_enable(1'b1),

.out\_signal(minus\_singal),

.out\_signal\_enable(button\_minus\_en));

filtercon #(128) dbnc(

.clk(clk),

.in\_signal(button\_in),

.clock\_enable(1'b1),

.out\_signal(button\_signal),

.out\_signal\_enable(button\_signal\_en)

);

filtercon #(128) dbnc\_reset(

Продолжение листинга 1.11

.clk(clk),

.in\_signal(button\_reset\_in),

.clock\_enable(1'b1),

.out\_signal(reset\_signal),

.out\_signal\_enable(reset\_signal\_en)

);

clk\_divider #(1024) div(

.clk(clk),

.clk\_div(clk\_div)

);

SevenSegmentLED led(

.AN\_MASK(AN\_MASK),

.NUMBER(NUMBER),

.clk(clk\_div),

.RESET(reset\_signal),

.AN(AN),

.SEG(SEG)

);

crimer aut(

.clk(clk),

.reset(reset\_signal),

.input\_en(input\_en),

.DataIn(Data\_in),

.X1(X1),

.X2(X2),

.X3(X3),

.valid\_cramer(valid\_cramer),

.failed(is\_failed),

.valid\_in(),

.state()

);

always@(posedge clk)

begin

if(is\_failed)

begin

NUMBER = 0;

AN\_MASK <= 8'b00000000;

end

else

begin

if(button\_minus\_en)

is\_minus <= ~is\_minus;

if (reset\_signal)

begin

Data\_in <= 0;

NUMBER <= 0;

n<=0;

is\_data\_last <= 0;

AN\_MASK <= 8'b11111111;

input\_en<=0;

end

else if (button\_signal\_en)

begin

if(is\_minus) begin

Data\_in <= minus \* SWITCHES;

is\_minus\_last <= 1;

is\_data\_last <= 1;

data\_last<=SWITCHES;

end

else begin

Data\_in <= SWITCHES;

Продолжение листинга 1.11

is\_minus\_last <= 0;

is\_data\_last <= 1;

data\_last<=SWITCHES;

end

input\_en<=1;

if(valid\_cramer) begin

is\_data\_last <= 0;

n <= NEXT(n);

end

end

if(input\_en)

input\_en<=~input\_en;

if(is\_data\_last == 1) begin

AN\_MASK<=~8'b00000001;

NUMBER <= data\_last;

End

else if(valid\_cramer)

begin

case(n)

0: begin NUMBER<=0; NUMBER<=0; AN\_MASK<=~8'b00000000;end

1: begin NUMBER<=0; NUMBER<=X1; AN\_MASK<=~8'b00000001;n1=0;end

2: begin NUMBER<=0; NUMBER<=X2; AN\_MASK<=~8'b00000001;end

3: begin NUMBER<=0; NUMBER<=X3; AN\_MASK<=~8'b00000001;end

4: begin

if(n1==0) begin

NUMBER<=0;

n1 <= n1+1; end

else if(n1<6)

n1 <= n1+1;

else if(n1)

AN\_MASK<=~8'b00010101;

case(n1)

1: NUMBER <= {NUMBER[27:0], X1};

2: NUMBER <= {NUMBER[27:0], 4'b0000};

3: NUMBER <= {NUMBER[27:0], X2};

4: NUMBER <= {NUMBER[27:0], 4'b0000};

5: NUMBER <= {NUMBER[27:0], X3};

endcase

end

endcase

end

end

end

function [2:0] NEXT ( input [2:0] num);

begin

if(num == 4)

NEXT = 1;

else

NEXT = num + 1;

end

endfunction

endmodule

## 1.5 Создание и верификация тестового модуля верхнего уровня

Для тестирования был выбран набор тестов аналогичный тестам конечного автомата. Код тестового модуля представлен в Листинге 1.12.

Листинг 1.12 — Модуль testbench.v

`timescale 1ns / 1ps

module testbench;

reg[3:0] SWITCHES = 0;

reg clk = 0;

reg button = 0;

reg button\_reset = 0;

reg button\_minus = 0;

wire[7:0] AN;

wire[6:0] SEG;

wire[31:0] NUMBER;

wire button\_signal;

wire button\_signal\_en;

wire button\_minus\_en;

wire minus\_signal;

wire reset\_signal\_en;

wire reset\_signalmain cntlr(

.SWITCHES(SWITCHES),

.button\_in(button),

.clk(clk),

.button\_reset\_in(button\_reset),

.AN(AN),

.NUMBER(NUMBER),

.SEG(SEG),

.button\_signal(button\_signal),

.button\_signal\_en(button\_signal\_en),

.reset\_signal\_en(reset\_signal\_en),

.reset\_signal(reset\_signal),

.button\_minus(button\_minus)

.button\_minus\_en(button\_minus\_en)

.minus\_signal(minus\_signal)

);

always #5 clk = ~clk;

initial

begin

#4000

//2

$srandom(35000);

SWITCHES = 4'b0001;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 1;

#1000;

repeat($urandom\_range(40,0))

begin

Продолжение листинга 1.12

button = $random;

#3;

end

button = 0;

#5000;

$srandom(35000);

SWITCHES = 4'b0010;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 1;

#1000;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 0;

#5000;

$srandom(35000);

SWITCHES = 4'b0011;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 1;

#1000;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 0;

#5000;

$srandom(35000);

SWITCHES = 4'b0010;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 1;

#1000;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 0;

Продолжение листинга 1.12

#5000;

$srandom(35000);

SWITCHES = 4'b0100;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 1;

#1000;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 0;

#5000;

$srandom(35000);

SWITCHES = 4'b0110;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 1;

#1000;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 0;

#5000;

$srandom(35000);

SWITCHES = 4'b0011;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 1;

#1000;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 0;

#5000;

$srandom(35000);

SWITCHES = 4'b0110;

repeat($urandom\_range(40,0))

begin

button = $random;

Продолжение листинга 1.12

#3;

end

button = 0;

#5000;

$srandom(35000);

SWITCHES = 4'b0011;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 1;

#1000;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 0;

#5000;

$srandom(35000);

SWITCHES = 4'b0110;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 1;

#1000;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 0;

#5000;

$srandom(35000);

SWITCHES = 4'b1001;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 1;

#1000;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 0;

#5000;

$srandom(35000);

SWITCHES = 4'b0001;

repeat($urandom\_range(40,0))

begin

Продолжение листинга 1.12

button = $random;

#3;

end

button = 1;

#1000;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 0;

#5000;

$srandom(35000);

SWITCHES = 4'b0001;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 1;

#1000;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 0;

#5000;

$srandom(35000);

SWITCHES = 4'b0001;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

End

button = 1;

#1000;

repeat($urandom\_range(40,0))

begin

button = $random;

#3;

end

button = 0;

#5000;

end

endmodule

Результат каждого теста можно проверить по временной диаграмме, рассматривая значение «NUMBER», так как именно оно будет записано в семисигментные индикаторы. [5]

На Рисунке 1.5 представлена временная диаграмма для первого теста.



Рисунок 1.5 — Временная диаграмма первого и второго тестов

На Рисунке 1.6 представлена временная диаграмма для второго теста.

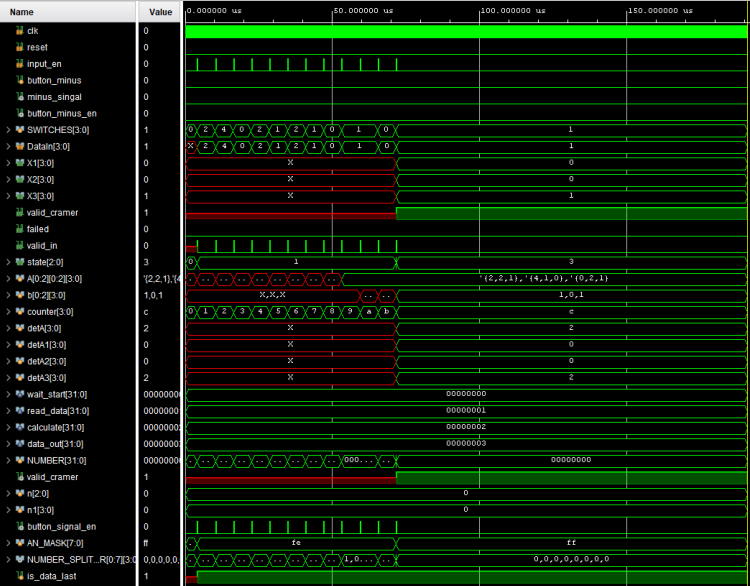


Рисунок 1.6 — Временная диаграмма второго

На Рисунке 1.7 представлена временная диаграмма для третьего теста.



Рисунок 1.7 — Временная диаграмма третьего теста

## 1.6 Создание файла проектных ограничений и загрузка проекта на отладочную плату nexys a7

Содержание файла проектных ограничений представлено в Листинге 1.13.

Листинг 1.13 — Содержимое файла проектных ограничений

create\_clock -period 10.000 -name sys\_clk -waveform {0.000 5.000} -add [get\_ports clk]

set\_property IOSTANDARD LVCMOS33 [get\_ports clk]

set\_property PACKAGE\_PIN E3 [get\_ports clk]

set\_property PACKAGE\_PIN N17 [get\_ports button\_in]

set\_property IOSTANDARD LVCMOS33 [get\_ports button\_in]

set\_property PACKAGE\_PIN M18 [get\_ports button\_reset\_in]

set\_property IOSTANDARD LVCMOS33 [get\_ports button\_reset\_in]

set\_property PACKAGE\_PIN P18 [get\_ports button\_minus]

set\_property IOSTANDARD LVCMOS33 [get\_ports button\_minus]

Продолжение листинга 1.13

set\_property IOSTANDARD LVCMOS33 [get\_ports {SWITCHES[0]}]

set\_property PACKAGE\_PIN J15 [get\_ports {SWITCHES[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {SWITCHES[1]}]

set\_property PACKAGE\_PIN L16 [get\_ports {SWITCHES[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {SWITCHES[2]}]

set\_property PACKAGE\_PIN M13 [get\_ports {SWITCHES[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {SWITCHES[3]}]

set\_property PACKAGE\_PIN R15 [get\_ports {SWITCHES[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {AN[0]}]

set\_property PACKAGE\_PIN J17 [get\_ports {AN[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {AN[1]}]

set\_property PACKAGE\_PIN J18 [get\_ports {AN[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {AN[2]}]

set\_property PACKAGE\_PIN T9 [get\_ports {AN[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {AN[3]}]

set\_property PACKAGE\_PIN J14 [get\_ports {AN[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {AN[4]}]

set\_property PACKAGE\_PIN P14 [get\_ports {AN[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {AN[5]}]

set\_property PACKAGE\_PIN T14 [get\_ports {AN[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {AN[6]}]

set\_property PACKAGE\_PIN K2 [get\_ports {AN[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {AN[7]}]

set\_property PACKAGE\_PIN U13 [get\_ports {AN[7]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {SEG[0]}]

set\_property PACKAGE\_PIN T10 [get\_ports {SEG[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {SEG[1]}]

set\_property PACKAGE\_PIN R10 [get\_ports {SEG[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {SEG[2]}]

set\_property PACKAGE\_PIN K16 [get\_ports {SEG[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {SEG[3]}]

set\_property PACKAGE\_PIN K13 [get\_ports {SEG[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {SEG[4]}]

set\_property PACKAGE\_PIN P15 [get\_ports {SEG[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {SEG[5]}]

set\_property PACKAGE\_PIN T11 [get\_ports {SEG[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {SEG[6]}]

set\_property PACKAGE\_PIN L18 [get\_ports {SEG[6]}]

#LEDS

set\_property IOSTANDARD LVCMOS33 [get\_ports is\_failed]

set\_property PACKAGE\_PIN N15 [get\_ports is\_failed]

set\_property IOSTANDARD LVCMOS33 [get\_ports is\_minus\_last]

set\_property PACKAGE\_PIN G14 [get\_ports is\_minus\_last]

Проект был загружен на отладочную плату NEXYS A7 и протестирован. На Рисунках 1.7–1.13.

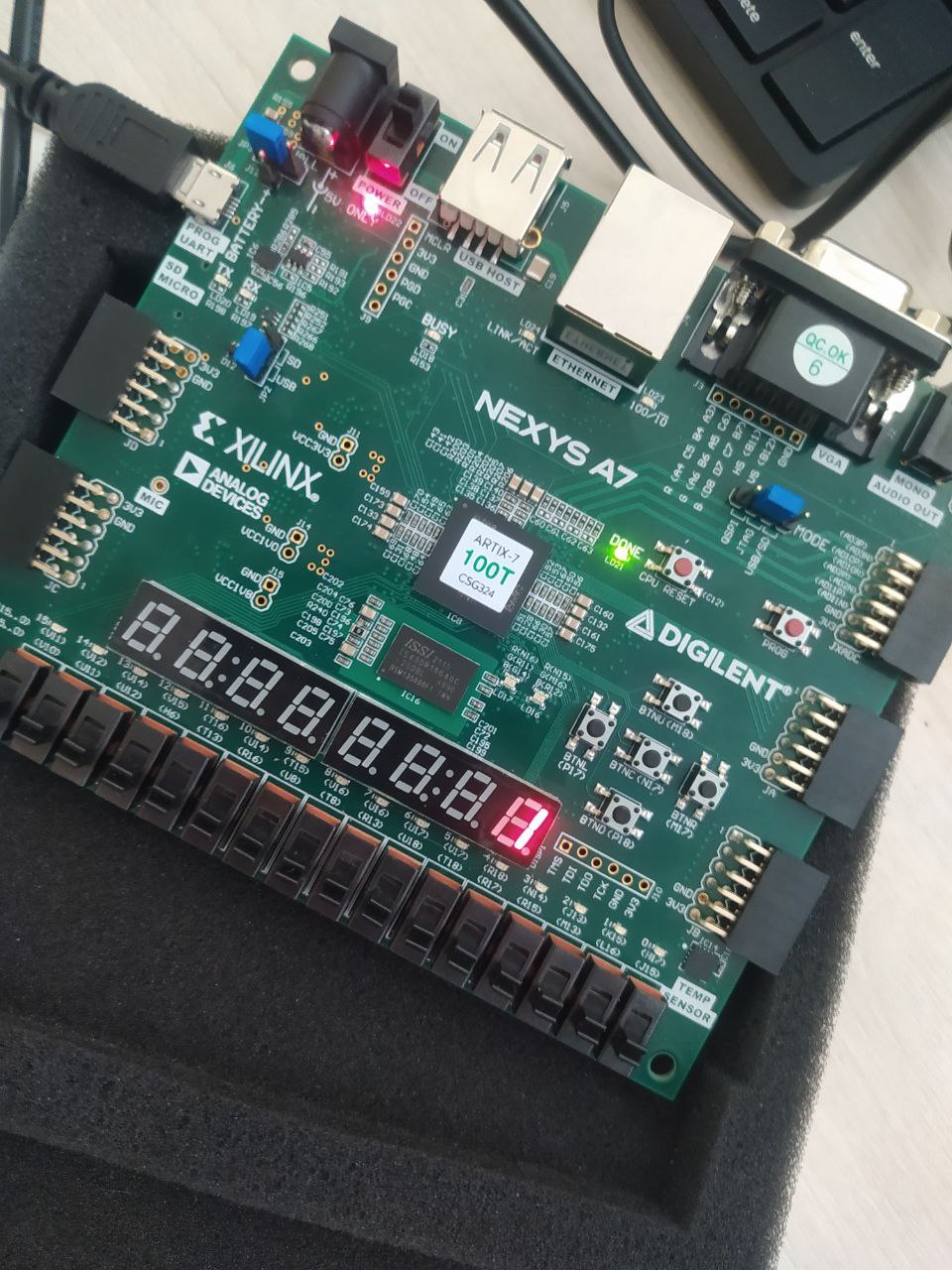


Рисунок 1.7 — Запись значения 1

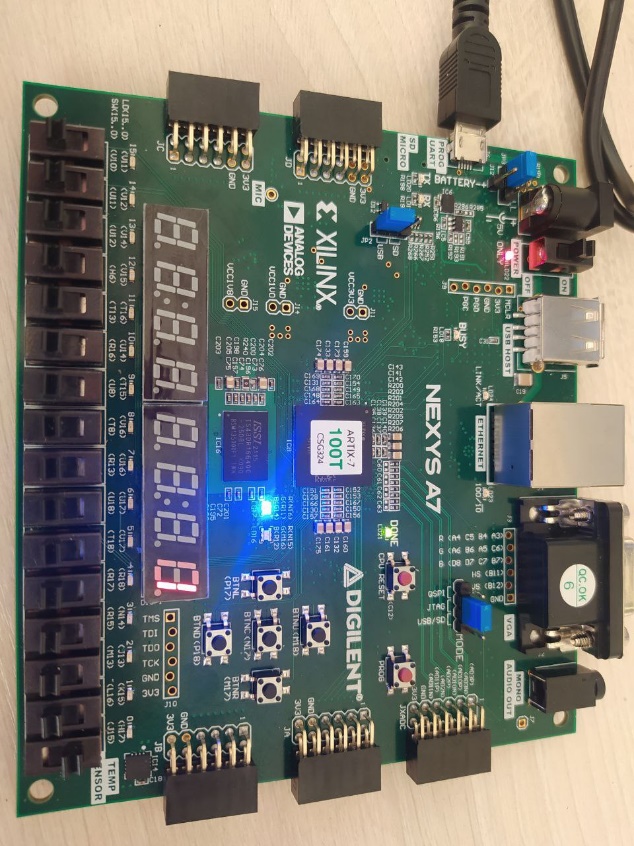


Рисунок 1.8 — Запись значения -1

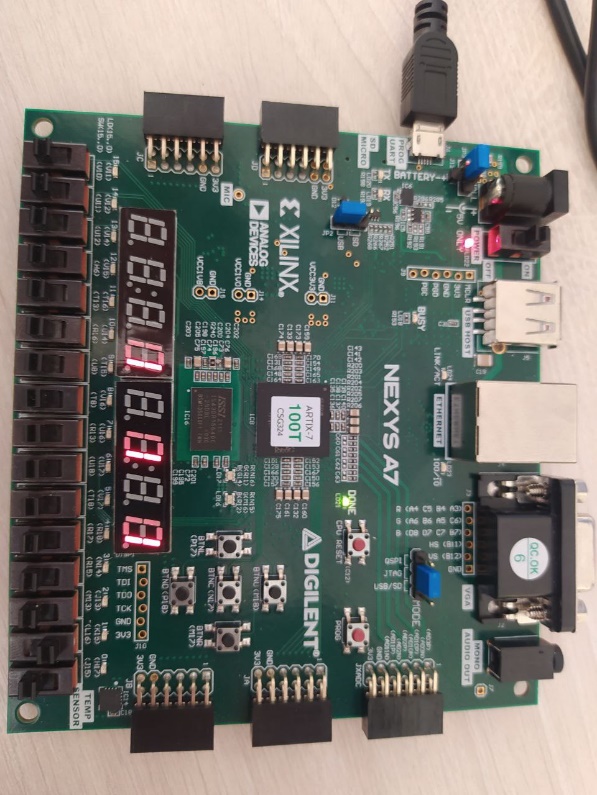


Рисунок 1.9 — Результат теста 1

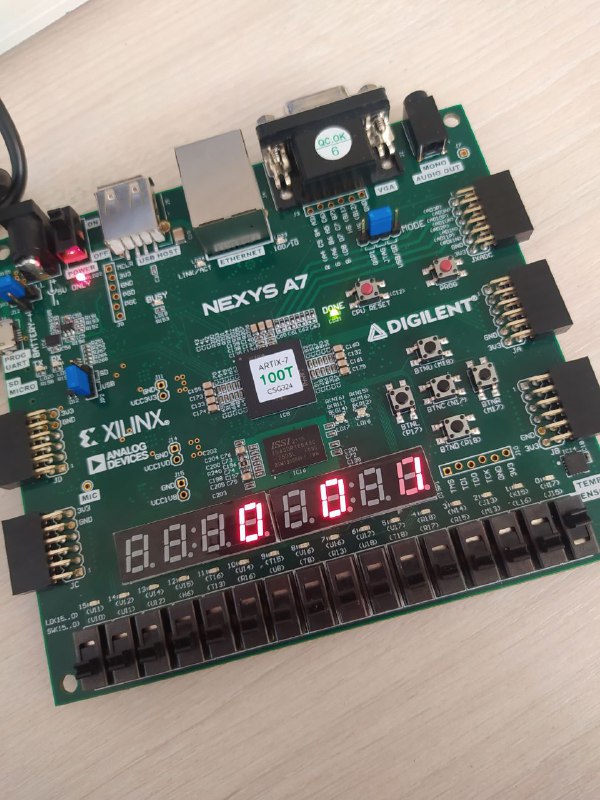


Рисунок 1.10 — Результат теста 2

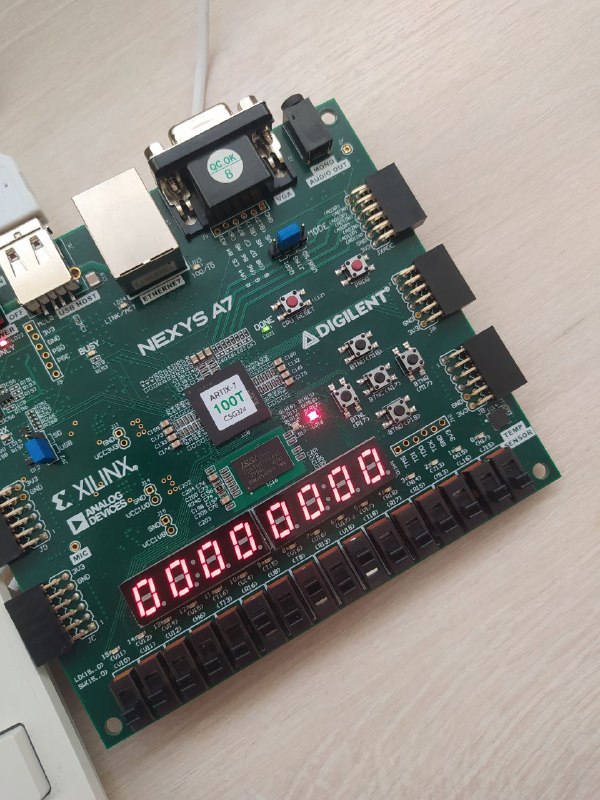


Рисунок 1.13 — Вывод ошибки как результат теста 3

ЗАКЛЮЧЕНИЕ

В результате выполнения лабораторной работы студентами был освоен маршрут проектирования компонентов аппаратного обеспечения, студенты овладели навыком проектирования и реализации конечных автоматов, а также освоили механизм верификации проекта с использованием ПЛИС.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Дуксин, Н. А. Архитектура вычислительных машин и систем. Основы построения вычислительной техники: Практикум : учебное пособие / Н. А. Дуксин, Д. В. Люлява, И. Е. Тарасов. — Москва : РТУ МИРЭА, 2023. — 185 с.

2. Смирнов С.С. Информатика [Электронный ресурс]: Методические указания по выполнению практических и лабораторных работ / С.С. Смирнов — М., МИРЭА — Российский технологический университет, 2018. — 1 электрон. опт. диск (CD-ROM).

3*.* Соловьев В. В. Основы языка проектирования цифровой аппаратуры Verilog. — М.: Горячая линия — Телеком, 2014. — 208 с.

4. Харрис Дэвид М., Харрис Сара Л. Цифровая схемотехника и архитектура компьютера. Издательство: ДМК-Пресс, 2018 г.

5. Максфилд К. Проектирование на ПЛИС. Курс молодого бойца. – М.: Издательский дом «Додэка-XXI», 2007. – 408 с.: илл. (Серия «Программируемые системы»)