МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

**«Вятский государственный университет»**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

РАЗРАБОТКА ФУНКЦИОНАЛЬНОЙ СХЕМЫ ОПЕРАЦИОННОГО АВТОМАТА В САПР QUARTUS

Отчет по лабораторной работе дисциплины

«Теория автоматов»

Выполнил студент группы ИВТ-21 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Кудяшев Я.Ю./

Проверил преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Мельцов В.Ю./

Киров 2021

1 Цель работы

Построить функциональную схему операционного автомата в системе автоматического проектирования Quartus, с целью проверки правильности работы операционного автомата.

2 Постановка задачи

Построить функциональную схему операционного автомата, выполняющего алгоритм деления 1 способом в двоичной системе счисления в ПК с ПЗ без ВО и вычитанием в ДК.

3 Разработка функциональной схемы операционного автомата

Функциональная схема операционного автомата представлена на рисунке 1.

4 Разработка содержательной ГСА

Содержательная ГСА представлена на рисунке 2.

5 Разработка микропрограммы

Листинг микропрограммы приведен в приложении А.

6 Результаты работы

Результаты работы программы представлены на рисунках 3-6.

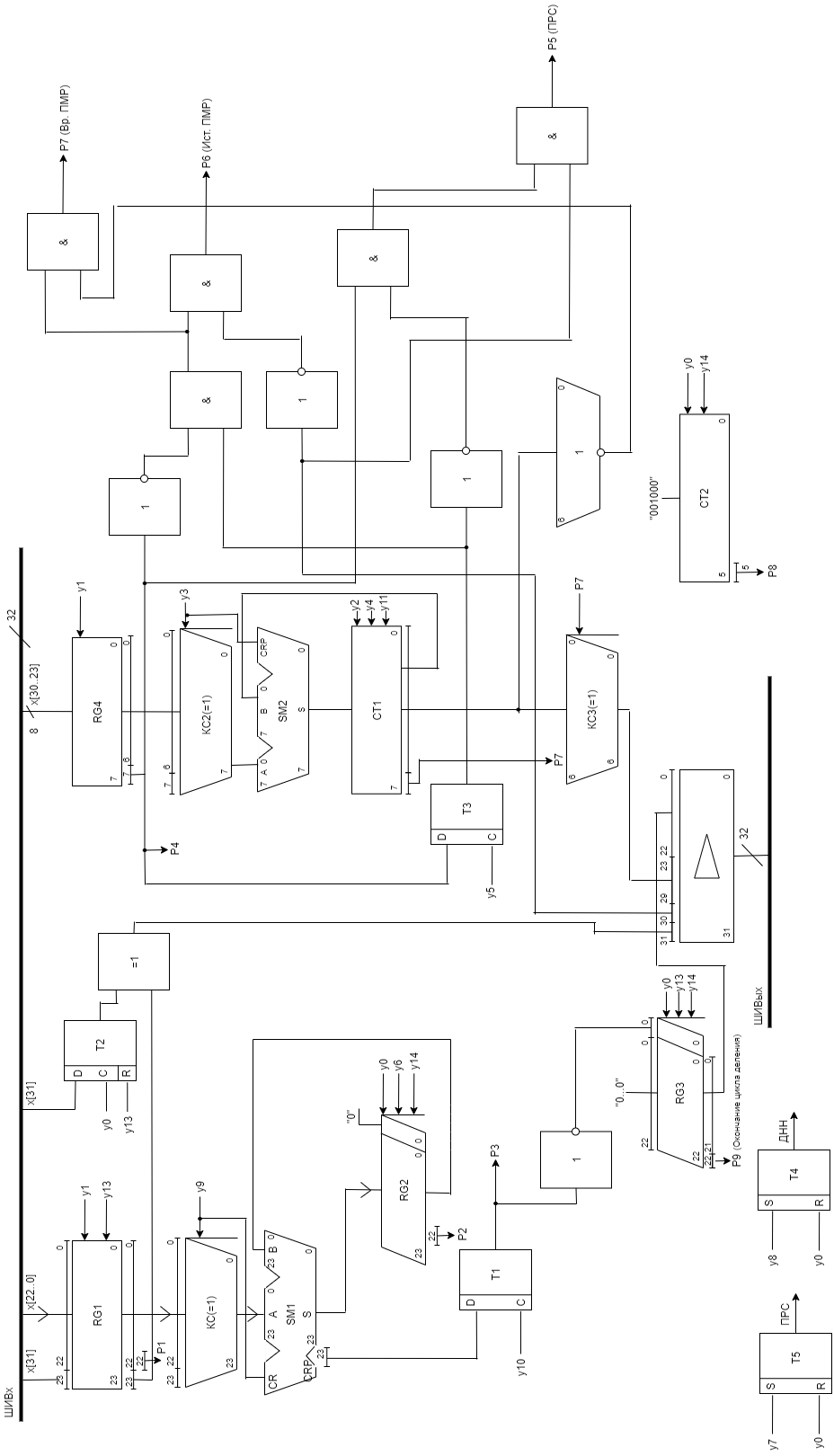


Рисунок 1 – Функциональная схема операционного автомата

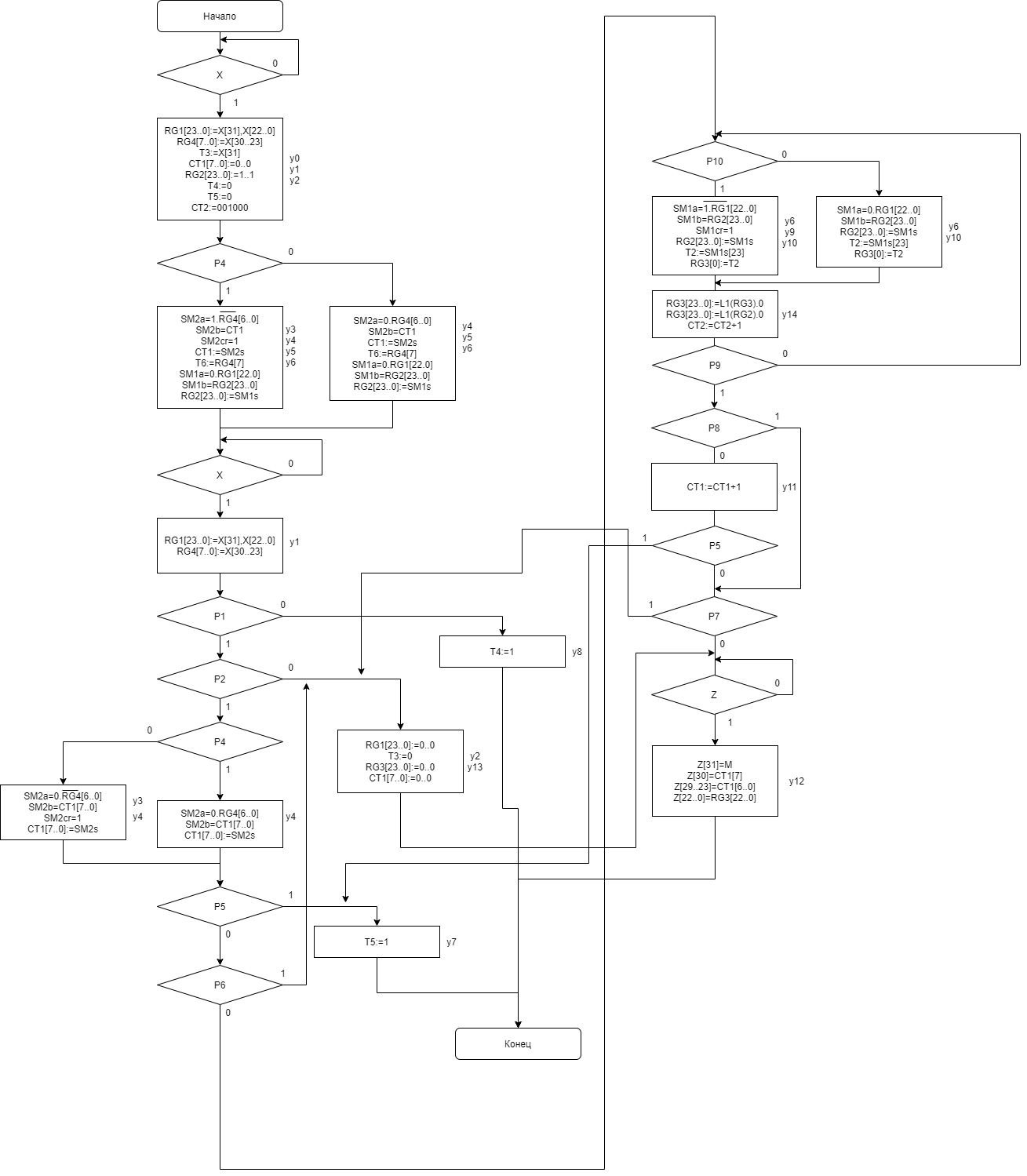


Рисунок 2 – Содержательная ГСА

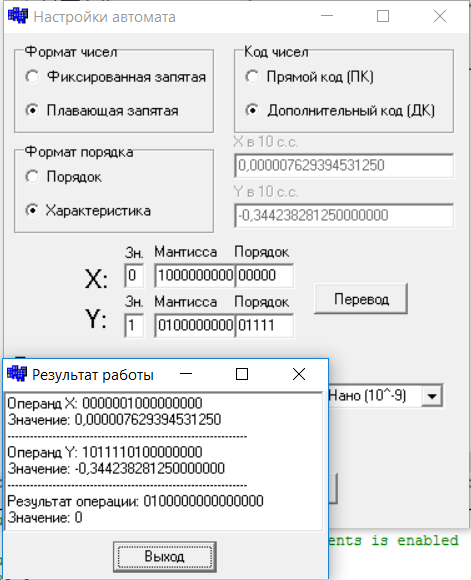


Рисунок 3 – Результат работы программы

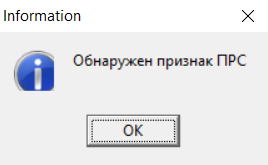


Рисунок 4 – Результат работы программы

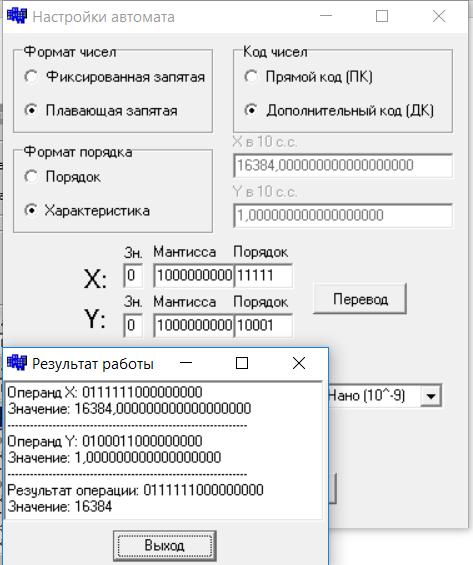


Рисунок 5 – Результат работы программы

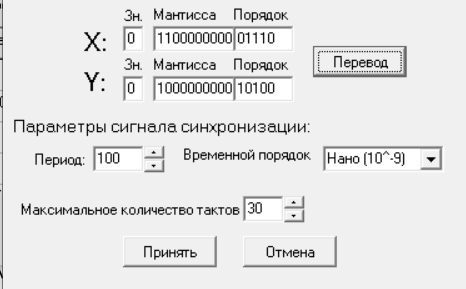
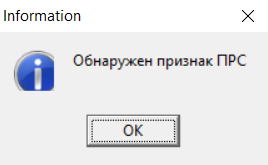


Рисунок 6 – Результат работы программы

Приложение А

(обязательное)

Листинг микропрограммы

// Generated by Quartus II 64-Bit Version 9.0 (Build Build 132 02/25/2009)

// Created on Thu Apr 29 18:08:58 2021

// Module Declaration

module block\_name

(

// {{ALTERA\_ARGS\_BEGIN}} DO NOT REMOVE THIS LINE!

clk, p, clkout, Z, y

// {{ALTERA\_ARGS\_END}} DO NOT REMOVE THIS LINE!

);

// Port Declaration

// {{ALTERA\_IO\_BEGIN}} DO NOT REMOVE THIS LINE!

input clk;

input [8:0] p;

output clkout;

output Z;

output [14:0] y;

// {{ALTERA\_IO\_END}} DO NOT REMOVE THIS LINE!

integer pc=1;//счётчик тактов

reg [14:0] y;

wire clkout;

reg Z=0;

assign clkout=!clk;

always @(posedge clk)

begin

y=15'b000000000000000;

case(pc)

1:begin

y[0]=1; y[1]=1; y[2]=1;

pc=pc+1;

end

2:begin

if (p[3]==1)

begin

y[3]=1; y[4]=1; y[5]=1; y[6]=1;

pc=pc+1;

end else

begin

y[4]=1; y[5]=1; y[6]=1;

pc=pc+1;

end

end

3:begin

y[1]=1;

pc=pc+1;

end

4:begin

if (p[0]==0)

begin

y[8]=1;

pc=50;

end else

if (p[0]==1&p[1]==0)

begin

y[2]=1; y[13]=1;

Z=1;

pc=50;

end else

if (p[0]==1&p[1]==1&p[3]==1)

begin

y[4]=1;

pc=pc+1;

end else

begin

y[3]=1; y[4]=1;

pc=pc+1;

end

end

5:begin

if (p[4]==1)

begin

y[7]=1;

pc=50;

end else

if (p[4]==0&p[5]==1)

begin

y[2]=1; y[13]=1;

Z=1;

pc=50;

end else

if (p[4]==0&p[5]==0&p[2]==0)

begin

y[6]=1; y[10]=1;

pc=pc+1;

end else

begin

y[6]=1; y[10]=1; y[9]=1;

pc=pc+1;

end

end

6:begin

y[14]=1;

pc=pc+1;

end

7:begin

if (p[8]==0)

begin

pc=5;

end else

if (p[8]==1&p[7]==0)

begin

y[11]=1;

pc=pc+1;

end else

if (p[8]==1&p[7]==1&p[6]==1)

begin

y[2]=1; y[13]=1;

Z=1;

pc=50;

end else

if (p[8]==1&p[7]==1&p[6]==0)

begin

Z=1;

y[12]=1;

pc=50;

end

end

8:begin

if (p[4]==1)

begin

y[7]=1;

y=15'b000000000000000;

pc=50;

end else

if (p[4]==0&p[6]==1)

begin

y[2]=1; y[13]=1;

Z=1;

y=15'b000000000000000;

pc=50;

end else

begin

Z=1;

y[12]=1;

y=15'b000000000000000;

pc=50;

end

end

endcase;

end

endmodule