МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА 44

КУРСОВОЙ ПРОЕКТ
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

старший преподаватель	_____	А.Н. Долидзе
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОМУ ПРОЕКТУ

по дисциплине: СХЕМОТЕХНИКА

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № _____ _____ _____
подпись, дата инициалы, фамилия

Санкт-Петербург 2025

Проектное задание

Разработать электронное устройство в соответствии с предложенной схемой и исходными данными, которое обеспечило бы заданную точность и качество работы. Работоспособность устройства обязательно подтверждается моделированием с применением САПР Quartus.

Разработка устройства включает в себя следующие этапы:

- составление и обоснование выбора функциональной схемы;
- разработку проектируемой схемы в САПР Quartus с использованием стандартных блоков, предоставляемых САПР (И, НЕ, дешифратор, триггер и т.д.) и описание ее работы с использованием временных диаграмм;
- разработку проектируемой схемы в САПР Quartus с использованием языка описания аппаратуры SystemVerilog и описание ее работы с использованием временных диаграмм;
- анализ двух подходов к разработке проектируемой схемы;
- обоснование и выбор FPGA на базе которого будет синтезироваться разрабатываемая схема;
- моделированием с применением САПР Quartus.
- назначение входов и выходов проектируемой схемы на выбранном FPGA.

Согласно варианту (схема 3, вариант 2), устройство является блоком для работы памяти с периферийными устройствами. Предложенная схема устройства представлена на рисунке 1.

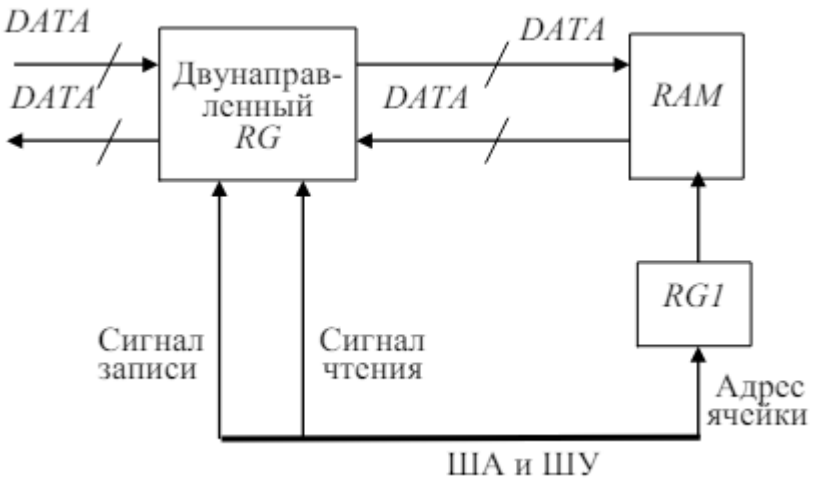

Рисунок 1 – Схема устройства

В таблице 1 представлен вариант работ с указанием заданной разрядности, адресности и дополнительного задания.

Таблица 1

Вариант	Разрядность	Адресность	Дополнительно
2	8	16	Без доп. задания

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Целью курсового проектирования является освоение методов расчета, схемотехнического проектирования и конструирования элементов и блоков ЦВМ на базе программируемых логических интегральных схема (ПЛИС, далее FPGA).

В ходе выполнения курсового проекта должны быть получены схема устройства и программный код на языке описания аппаратуры System Verilog, соответствующие заданному в проектном задании функционалу. Устройство представляет собой преобразователь кодов, включающее в себя схемы контроля выдачи входного и выходного кодов.

1. РАЗРАБОТКА ФУНКЦИОНАЛЬНОЙ СХЕМЫ

1.1. ВЫБОР ФУНКЦИОНАЛЬНОЙ СХЕМЫ

Реализовать заданный функционал можно различными способами. В ходе выполнения работы было принято решение использовать по возможности готовые модули среды Quartus. Например, в среде присутствует нужный нам модуль памяти, который может работать с двунаправленными шинами. Также в его параметрах можно указать нужную нам адресность и разрядность. Для корректной работы шины памяти после двунаправленного регистра поставлен элемент с тремя состояниями, который работает как буфер. К нему подключён провод от сигнала на разрешение записи в память.

Что же касается двухстороннего регистра, то такого модуля нет в среде Quartus. Поэтому мы создадим нужный нам пользовательский модуль на основе D-триггеров. Кол-во триггеров равно заданной разрядности. На входе использовалась комбинация элементов И, ИЛИ и НЕ, позволяющие принять сигнал либо с одной шины данных, либо с другой, со сбросом. Если mode=1, то осуществляется запись, если mode=0 – то чтение.

Последний модуль в схеме – регистр для подачи адреса в память. Все эти модули управляются одним частотным сигналом.

1.2. СОСТАВЛЕНИЕ СХЕМЫ УСТРОЙСТВА С ИСПОЛЬЗОВАНИЕМ СТАНДАРТНЫХ БЛОКОВ

В ходе разработки схемы устройства были использован стандартный модуль *LPM_RAM_IO*, *lpm_dff* а также *D-триггеры*, логические элементы *or*, *and*, *not*. Также для упрощения итоговой схемы, был создан пользовательский блок. На рисунках 2- 3 представлен результат моделирования схемы в графическом редакторе среды Quartus.

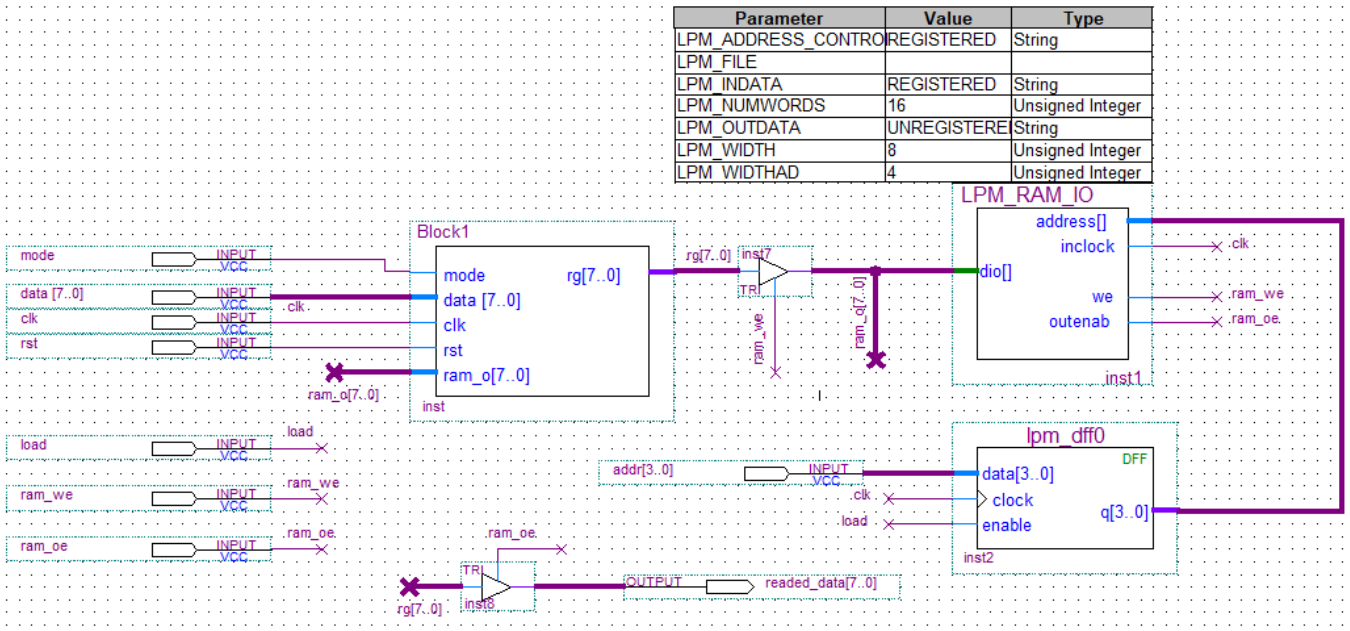

Рисунок 2 – Основная схема блока

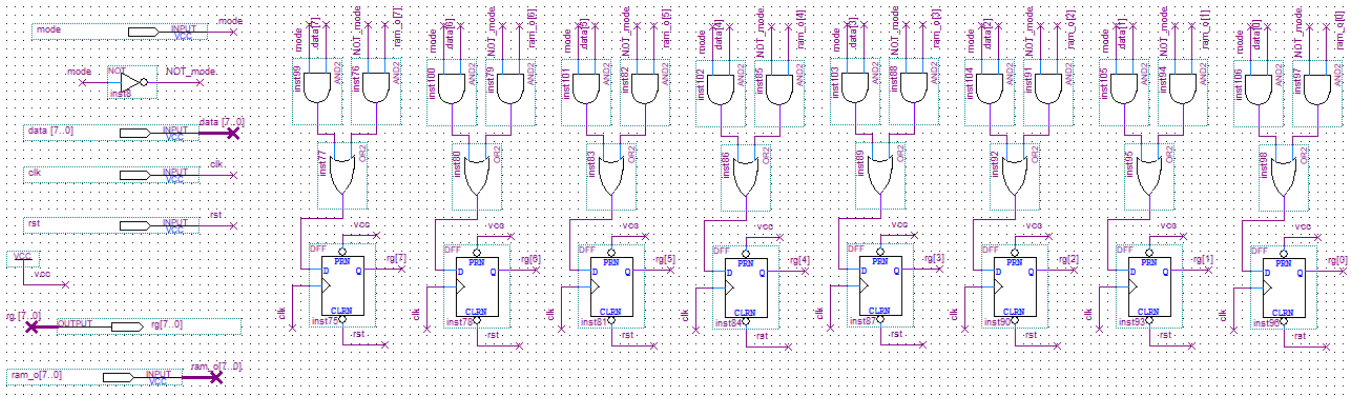

Рисунок 3 – Схема двунаправленного регистра

На рисунке 4 представлен результат симулирования схемы в виде временной диаграммы.

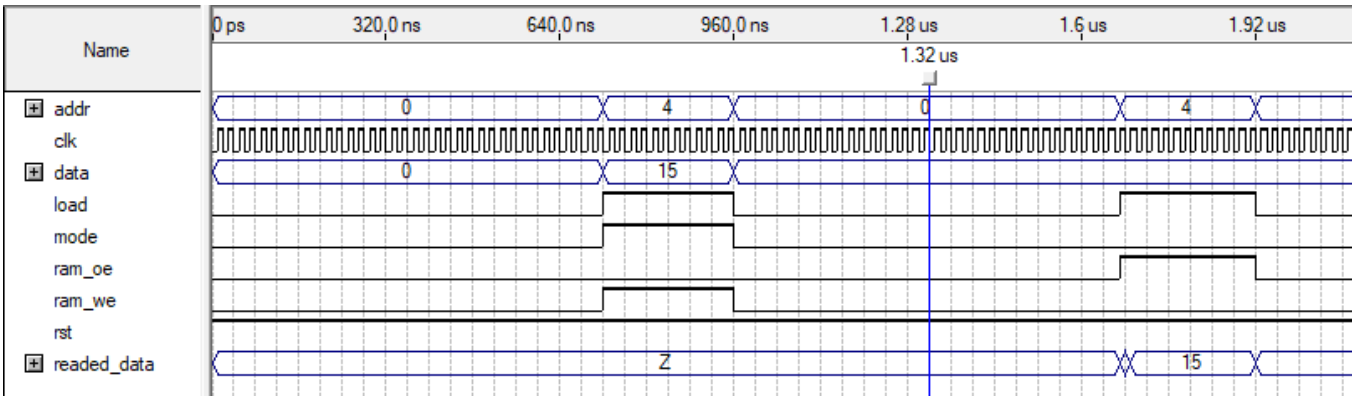

Рисунок 4 – Временная диаграмма

1.3. СОСТАВЛЕНИЕ СХЕМЫ УСТРОЙСТВА С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА ОПИСАНИЯ АППАРАТУРЫ SYSTEM VERILOG

Программа состоит из двух файлов: для пользовательского блока двустороннего регистра и основной программы. Код программы представлен в листингах 1-2.

Листинг 1

```
module block1(
    mode,
    clk,
    rst,
    data,
    ram_o,
    rg
);


input     mode;
input     clk;
input     rst;
input     [7:0] data;
input     [7:0] ram_o;
output    [7:0] rg;

wire      NOT_mode;
reg       [7:0] rg_ALTERA_SYNTHESIZED;
wire      vcc;
wire      SYNTHESIZED_WIRE_0;
wire      SYNTHESIZED_WIRE_1;
wire      SYNTHESIZED_WIRE_2;
```

```
wire	SYNTHESIZED_WIRE_3;
wire	SYNTHESIZED_WIRE_4;
wire	SYNTHESIZED_WIRE_5;
wire	SYNTHESIZED_WIRE_6;
wire	SYNTHESIZED_WIRE_7;
wire	SYNTHESIZED_WIRE_8;
wire	SYNTHESIZED_WIRE_9;
wire	SYNTHESIZED_WIRE_10;
wire	SYNTHESIZED_WIRE_11;
wire	SYNTHESIZED_WIRE_12;
wire	SYNTHESIZED_WIRE_13;
wire	SYNTHESIZED_WIRE_14;
wire	SYNTHESIZED_WIRE_15;
wire	SYNTHESIZED_WIRE_16;
wire	SYNTHESIZED_WIRE_17;
wire	SYNTHESIZED_WIRE_18;
wire	SYNTHESIZED_WIRE_19;
wire	SYNTHESIZED_WIRE_20;
wire	SYNTHESIZED_WIRE_21;
wire	SYNTHESIZED_WIRE_22;
wire	SYNTHESIZED_WIRE_23;



assign	SYNTHESIZED_WIRE_4 = data[6] & mode;

assign	SYNTHESIZED_WIRE_7 = data[5] & mode;

assign	SYNTHESIZED_WIRE_10 = data[4] & mode;
```

```
assign     SYNTHESIZED_WIRE_13 = data[3] & mode;

assign     SYNTHESIZED_WIRE_16 = data[2] & mode;

assign     SYNTHESIZED_WIRE_19 = data[1] & mode;

assign     SYNTHESIZED_WIRE_22 = data[0] & mode;



always@(posedge clk or negedge rst or negedge vcc)
begin
if (!rst)
    begin
    rg_ALTERA_SYNTHESIZED[7] = 0;
    end
else
if (!vcc)
    begin
    rg_ALTERA_SYNTHESIZED[7] = 1;
    end
else
    begin
    rg_ALTERA_SYNTHESIZED[7] = SYNTHESIZED_WIRE_0;
    end
end

assign     SYNTHESIZED_WIRE_2 = ram_o[7] & NOT_mode;
```

```
	assign	SYNTHESIZED_WIRE_0	=	SYNTHESIZED_WIRE_1	|
SYNTHESIZED_WIRE_2;


	always@(posedge clk or negedge rst or negedge vcc)
	begin
	if (!rst)
	   begin
	   rg_ALTERA_SYNTHESIZED[6] = 0;
	   end
	else
	if (!vcc)
	   begin
	   rg_ALTERA_SYNTHESIZED[6] = 1;
	   end
	else
	   begin
	   rg_ALTERA_SYNTHESIZED[6] = SYNTHESIZED_WIRE_3;
	   end
	end

	assign	SYNTHESIZED_WIRE_5 = ram_o[6] & NOT_mode;

	assign	NOT_mode =  ~mode;

	assign	SYNTHESIZED_WIRE_3	=	SYNTHESIZED_WIRE_4	|
SYNTHESIZED_WIRE_5;


	always@(posedge clk or negedge rst or negedge vcc)
```

```
begin
if (!rst)
    begin
    rg_ALTERA_SYNTHESIZED[5] = 0;
    end
else
if (!vcc)
    begin
    rg_ALTERA_SYNTHESIZED[5] = 1;
    end
else
    begin
    rg_ALTERA_SYNTHESIZED[5] = SYNTHESIZED_WIRE_6;
    end
end

assign     SYNTHESIZED_WIRE_8 = ram_o[5] & NOT_mode;

assign     SYNTHESIZED_WIRE_6     =     SYNTHESIZED_WIRE_7     |
SYNTHESIZED_WIRE_8;


always@(posedge clk or negedge rst or negedge vcc)
begin
if (!rst)
    begin
    rg_ALTERA_SYNTHESIZED[4] = 0;
    end
else
if (!vcc)
```

```
        begin
        rg_ALTERA_SYNTHESIZED[4] = 1;
        end
    else
        begin
        rg_ALTERA_SYNTHESIZED[4] = SYNTHESIZED_WIRE_9;
        end
    end

    assign  SYNTHESIZED_WIRE_11 = ram_o[4] & NOT_mode;

    assign  SYNTHESIZED_WIRE_9  =  SYNTHESIZED_WIRE_10  |
SYNTHESIZED_WIRE_11;


    always@(posedge clk or negedge rst or negedge vcc)
    begin
    if (!rst)
        begin
        rg_ALTERA_SYNTHESIZED[3] = 0;
        end
    else
    if (!vcc)
        begin
        rg_ALTERA_SYNTHESIZED[3] = 1;
        end
    else
        begin
        rg_ALTERA_SYNTHESIZED[3] = SYNTHESIZED_WIRE_12;
        end
```

```
        end

        assign     SYNTHESIZED_WIRE_14 = ram_o[3] & NOT_mode;

        assign     SYNTHESIZED_WIRE_12     =     SYNTHESIZED_WIRE_13     |
SYNTHESIZED_WIRE_14;


        always@(posedge clk or negedge rst or negedge vcc)
        begin
        if (!rst)
            begin
            rg_ALTERA_SYNTHESIZED[2] = 0;
            end
        else
        if (!vcc)
            begin
            rg_ALTERA_SYNTHESIZED[2] = 1;
            end
        else
            begin
            rg_ALTERA_SYNTHESIZED[2] = SYNTHESIZED_WIRE_15;
            end
        end

        assign     SYNTHESIZED_WIRE_17 = ram_o[2] & NOT_mode;

        assign     SYNTHESIZED_WIRE_15     =     SYNTHESIZED_WIRE_16     |
SYNTHESIZED_WIRE_17;
```

```
always@(posedge clk or negedge rst or negedge vcc)
begin
if (!rst)
   begin
   rg_ALTERA_SYNTHESIZED[1] = 0;
   end
else
if (!vcc)
   begin
   rg_ALTERA_SYNTHESIZED[1] = 1;
   end
else
   begin
   rg_ALTERA_SYNTHESIZED[1] = SYNTHESIZED_WIRE_18;
   end
end

assign     SYNTHESIZED_WIRE_20 = ram_o[1] & NOT_mode;

assign     SYNTHESIZED_WIRE_18     =     SYNTHESIZED_WIRE_19     |
SYNTHESIZED_WIRE_20;


always@(posedge clk or negedge rst or negedge vcc)
begin
if (!rst)
   begin
   rg_ALTERA_SYNTHESIZED[0] = 0;
   end
```

```
else
if (!vcc)
    begin
    rg_ALTERA_SYNTHESIZED[0] = 1;
    end
else
    begin
    rg_ALTERA_SYNTHESIZED[0] = SYNTHESIZED_WIRE_21;
    end
end

assign    SYNTHESIZED_WIRE_23 = ram_o[0] & NOT_mode;

assign    SYNTHESIZED_WIRE_21    =    SYNTHESIZED_WIRE_22    |
SYNTHESIZED_WIRE_23;

assign    SYNTHESIZED_WIRE_1 = data[7] & mode;

assign    rg = rg_ALTERA_SYNTHESIZED;
assign    vcc = 1;

endmodule
```

Листинг 2

```
 module block2(
    mode,
    clk,
    rst,
    load,
```

```
    ram_we,
    ram_oe,
    addr,
    data,
    readed_data
);


input     mode;
input     clk;
input     rst;
input     load;
input     ram_we;
input     ram_oe;
input     [3:0] addr;
input     [7:0] data;
output    [7:0] readed_data;

wire      [7:0] ram_o;
wire      [7:0] rg;
wire      [3:0] SYNTHESIZED_WIRE_0;




Block1    b2v_inst(
    .mode(mode),
    .clk(clk),
    .rst(rst),
```

```
    .data(data),
    .ram_o(ram_o),
    .rg(rg));


lpm_ram_io_0  b2v_inst1(
    .inclock(clk),
    .we(ram_we),
    .outenab(ram_oe),
    .address(SYNTHESIZED_WIRE_0),
    .dio(ram_o)
    );


lpm_dff0 b2v_inst2(
    .clock(clk),
    .enable(load),
    .data(addr),
    .q(SYNTHESIZED_WIRE_0));

assign    ram_o[7] = ram_we ? rg[7] : 1'bz;
assign    ram_o[6] = ram_we ? rg[6] : 1'bz;
assign    ram_o[5] = ram_we ? rg[5] : 1'bz;
assign    ram_o[4] = ram_we ? rg[4] : 1'bz;
assign    ram_o[3] = ram_we ? rg[3] : 1'bz;
assign    ram_o[2] = ram_we ? rg[2] : 1'bz;
assign    ram_o[1] = ram_we ? rg[1] : 1'bz;
assign    ram_o[0] = ram_we ? rg[0] : 1'bz;

assign    readed_data[7] = ram_oe ? rg[7] : 1'bz;
```

```
assign     readed_data[6] = ram_oe ? rg[6] : 1'bz;
assign     readed_data[5] = ram_oe ? rg[5] : 1'bz;
assign     readed_data[4] = ram_oe ? rg[4] : 1'bz;
assign     readed_data[3] = ram_oe ? rg[3] : 1'bz;
assign     readed_data[2] = ram_oe ? rg[2] : 1'bz;
assign     readed_data[1] = ram_oe ? rg[1] : 1'bz;
assign     readed_data[0] = ram_oe ? rg[0] : 1'bz;


endmodule

module lpm_ram_io_0(inclock,we,outenab,address,dio);
/* synthesis black_box */

input inclock;
input we;
input outenab;
input [3:0] address;
inout [7:0] dio;

endmodule
```

На рисунке 5 представлен результат симуляции схемы, реализованной с помощью языка описания аппаратуры System Verilog.

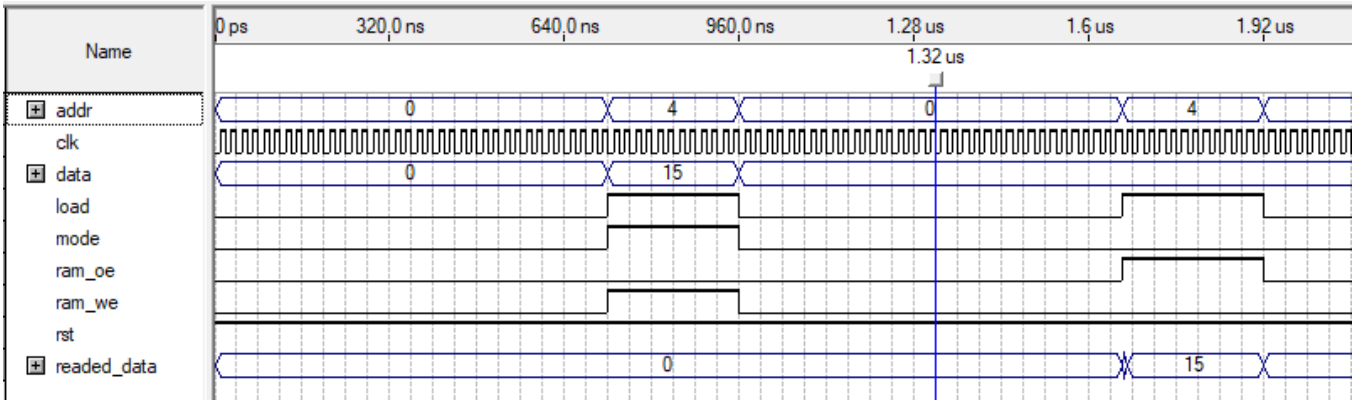

Рисунок 5 – Временная диаграмма

2. ВЫБОР FPGA

В ходе выполнения проекта было принято решение использовать ПЛИС Stratix II – EP2S15F484C3 ввиду того, что память в ПЛИС требует большое количество ресурсов и пинов. На рисунке 6 представлена схема подключения к ПЛИС.

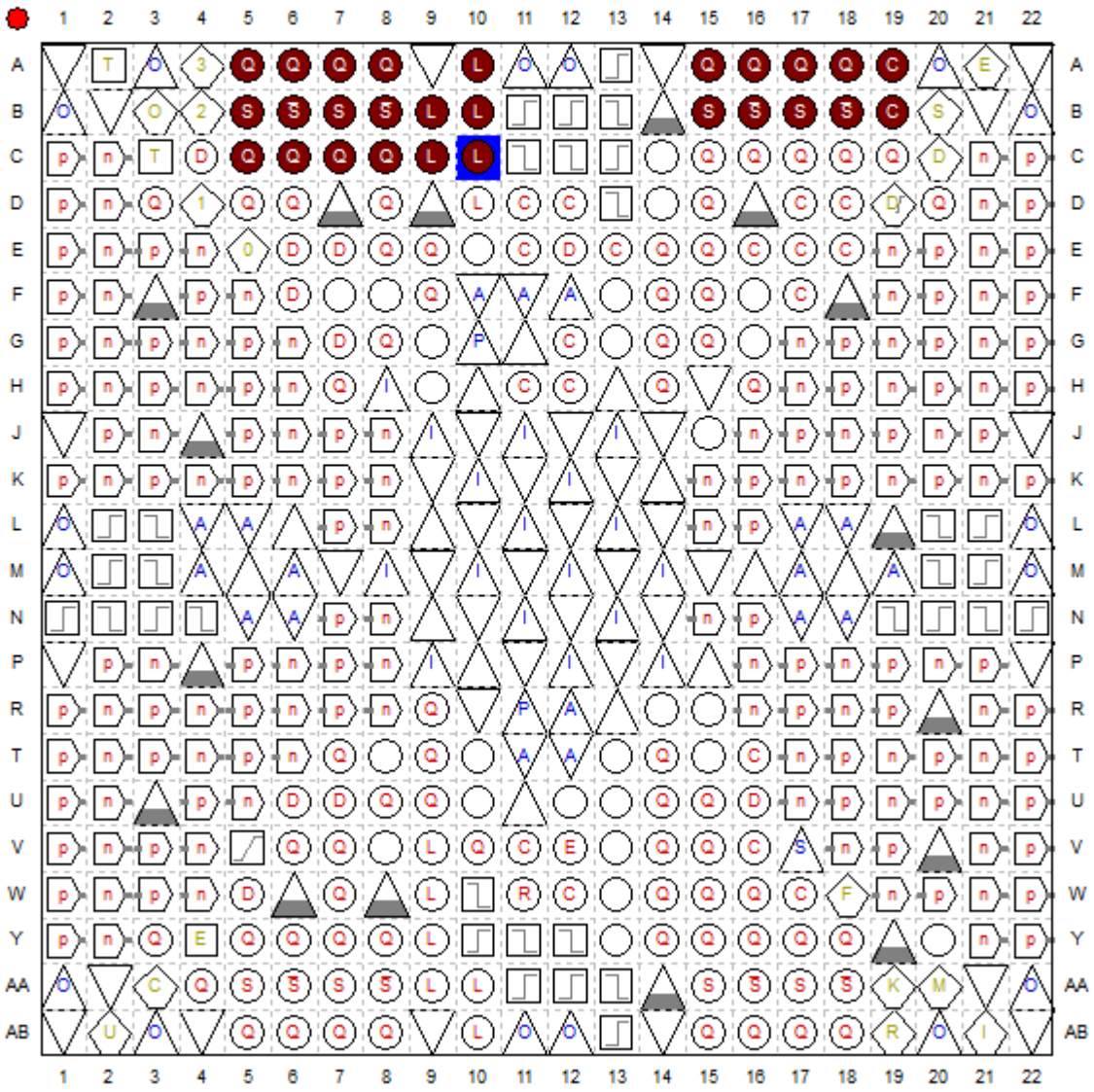

Рисунок 6 – Схема подключения выходов

На рисунке 7 представлена таблица назначения входов и выходов схемы.

	Node Name	Direction	Location	I/O Bank	VREF Group
1	clk	Input	PIN_A5	4	B4_N1
2	data[7]	Input	PIN_A6	4	B4_N1
3	data[6]	Input	PIN_A7	4	B4_N1
4	data[5]	Input	PIN_A8	4	B4_N1
5	data[4]	Input	PIN_A10	9	B4_N1
6	data[3]	Input	PIN_A15	3	B3_N0
7	data[2]	Input	PIN_A16	3	B3_N0
8	data[1]	Input	PIN_A17	3	B3_N0
9	data[0]	Input	PIN_A18	3	B3_N0
10	mode	Input	PIN_A19	3	B3_N1
11	ram_o[7]	Input	PIN_B5	4	B4_N1
12	ram_o[6]	Input	PIN_B6	4	B4_N1
13	ram_o[5]	Input	PIN_B7	4	B4_N1
14	ram_o[4]	Input	PIN_B8	4	B4_N1
15	ram_o[3]	Input	PIN_B9	9	B4_N1
16	ram_o[2]	Input	PIN_B10	9	B4_N1
17	ram_o[1]	Input	PIN_B15	3	B3_N0
18	ram_o[0]	Input	PIN_B16	3	B3_N0
19	rg[7]	Output	PIN_B17	3	B3_N0
20	rg[6]	Output	PIN_B18	3	B3_N0
21	rg[5]	Output	PIN_B19	3	B3_N1
22	rg[4]	Output	PIN_C5	4	B4_N1
23	rg[3]	Output	PIN_C6	4	B4_N1
24	rg[2]	Output	PIN_C7	4	B4_N1
25	rg[1]	Output	PIN_C8	4	B4_N1
26	rg[0]	Output	PIN_C9	9	B4_N1
27	rst	Input	PIN_C10	9	B4_N1

Рисунок 7 – Таблица назначения входов и выходов схемы

ЗАКЛЮЧЕНИЕ

В ходе выполнения проекта был получен блок для работы памяти с периферийными устройствами. Моделирование устройства выполнено с помощью графической среды САПР Quartus и с помощью языка описания аппаратуры System Verilog. Обе реализации дают одинаковый и подобный друг другу результат.

Устройство работает конкретно, соответствует поставленной задачи и функционалу.