МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Национальный исследовательский университет «МЭИ»

Институт ИРЭ

Кафедра ЭиН

ТВОРЧЕСКИЙ КОНКУРС

Направление 11.03.04 Электроника и наноэлектроника
(код и наименование)

Образовательная программа Микроэлектроника и твердотельная электроника

Форма обучения очная
(очная/очно-заочная/заочная)

Тема: Коммутатор для соединения любого из 3 приёмников с любым из 8 передатчиков

Студент ЭР-05-18 _____ Буслаев М.Р.
группа подпись фамилия и инициалы

Москва, 2022

СОДЕРЖАНИЕ

ПРОЕКТИРОВАНИЕ

1.1. Принцип работы устройства

Задача состоит в передаче сигналов с 8 входов на 3 выхода, при этом на один выход не может быть подключено более одного входа, поскольку это вызовет конфликт у входов устройства – короткое замыкание входов. В то же время на один вход может быть подключено несколько выходов. Рисунок 1 показывает, как мы не можем соединять входы с выходами, рисунок 2 показывает, как можем.

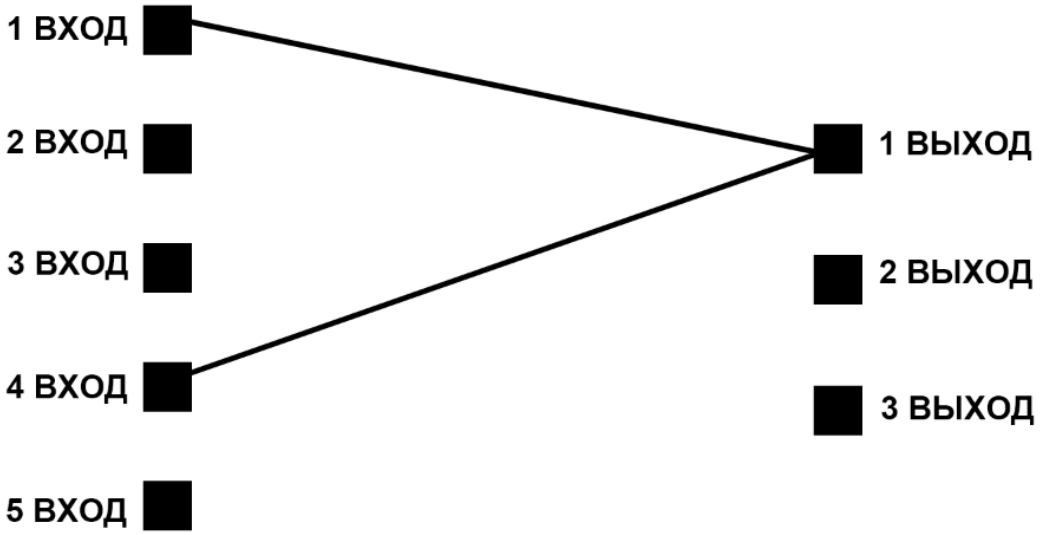

Рисунок 1 – Неправильный способ коммутации

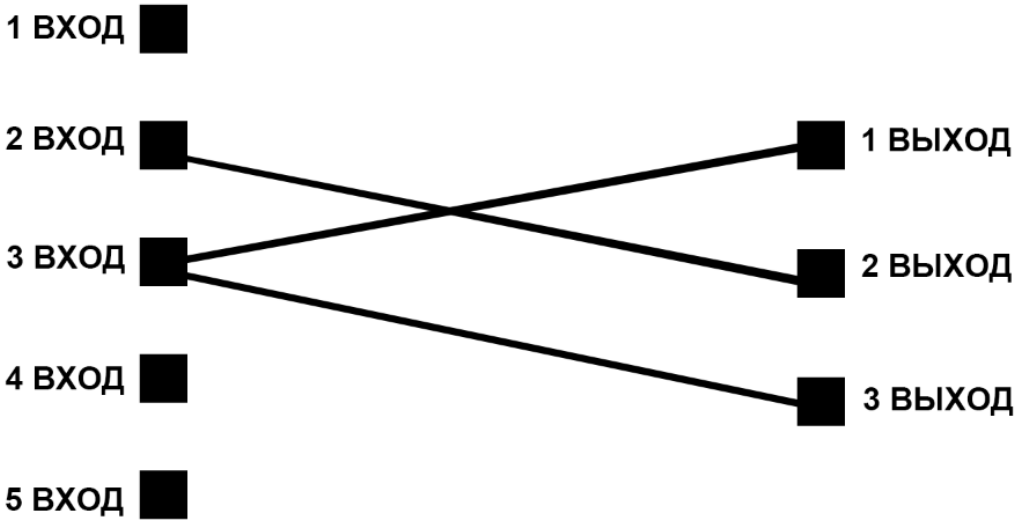

Рисунок 2 – Корректный способ коммутации

То есть помимо соединения входов и выход нам также необходимо отсеять неправильные способы коммутации. Сделать это можно за счет отбора 8 входов для каждого отдельного выхода, то есть мы должны под каждый выход отобрать лишь один вход. Такую функцию в сфере цифровых устройств выполняет мультиплексор.

Поскольку у нас имеется 8 входов, мультиплексор должен из этих входов выбрать тот, который нужно подать на выход. Если же мы захотим не подключать ни один из этих входов на выход, нам нужно предусмотреть подключение логического нуля на выход. Для этого мы просто можем сделать подать логический ноль на один из входов мультиплексора и соответствующий ему управляющий трехразрядный сигнал, например подадим управляющий сигнал 000 и 0 на первый вход(обозначен как D0).

Получается, что основным устройством нашего коммутатора будет мультиплексор 8 к 1, а управление мультиплексором будет осуществляться через 3-ех битный сигнал.

Помимо этого, одним из условий задания является возможность запоминания состояния коммутатора, чтобы мы могли управляющие сигналы на мультиплексоры подавать не на протяжении всего времени работы коммутатора. То есть один раз подаем на мультиплексор управляющий сигнал и забываем про него до тех пор, пока не надо поменять его. В этом нам поможет триггер – он будет запоминать состояние управляющих входов.

1.2. Блок-схема устройства

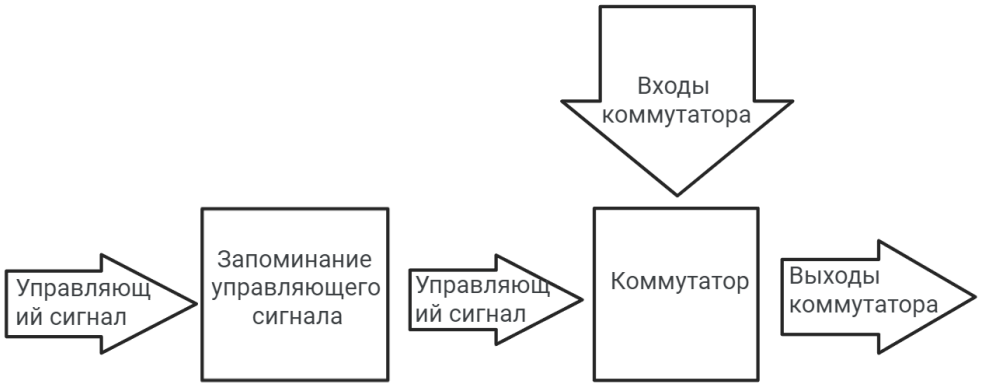

Рисунок 3 – Логическая блок-схема устройства

Управляющий сигнал будет управлять 3-мя мультиплексорами, по одному мультиплексору на каждый выход коммутатора. У нас будет 3 параллельных 3-ех битных шины на каждый мультиплексор, эти входы будут управляющими. Итого у нас будет 12 управляющих сигналов.

Коммутатор представляет из себя 3 мультиплексора, ко входам которых будут подведены выходы с запоминающего блока.

Входы коммутатора – 8 параллельных входов, принимающих цифровой сигнал, 3 выхода также цифровые.

Помимо этого запоминающий регистр будет состоять из 3-ех разрядного синхронизирующего сигнала и 3-ех разрядного управляющего сигнала. Если нам нужно поменять состояние на первом мультиплексоре, мы подадим синхронизирующий сигнал 100, если на всех мультиплексорах, подадим 111, регистр будет запоминать управляющий сигнал ABC.

C	B	A	W	C	B	A	W
L	L	L	$\overline{D_0}$	H	L	L	$\overline{D_4}$
L	L	H	$\overline{D_1}$	H	L	H	$\overline{D_5}$
L	H	L	$\overline{D_2}$	H	H	L	$\overline{D_6}$
L	H	H	$\overline{D_3}$	H	H	H	$\overline{D_7}$

Таблица 1. Логическая таблица мультиплексора 8к1, Dx – x-вый вход, A,B,C – управляющие сигналы

Алгоритм

1. Подаем управляющий сигнал
2. Запоминаем управляющий сигнал, далее его течение можно прервать
3. С выхода запоминающего устройства управляющий сигнал идет к коммутатору
4. Управляющий сигнал соединяет определенные вход с определенным выходом.

1.3. Принципиальные схемы блоков устройства

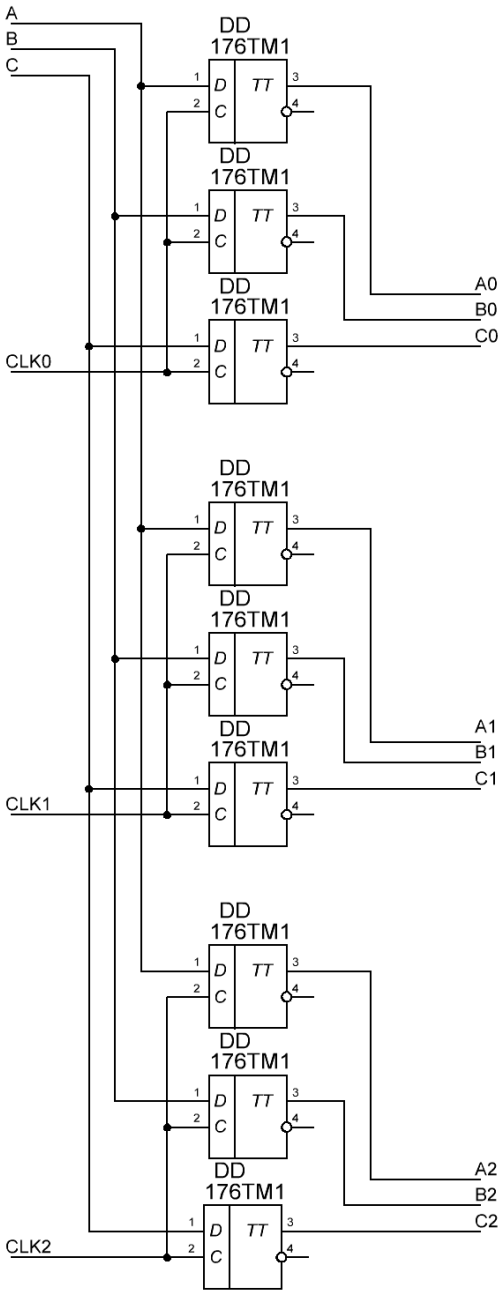

Рисунок 4 – Схема запоминающего блока

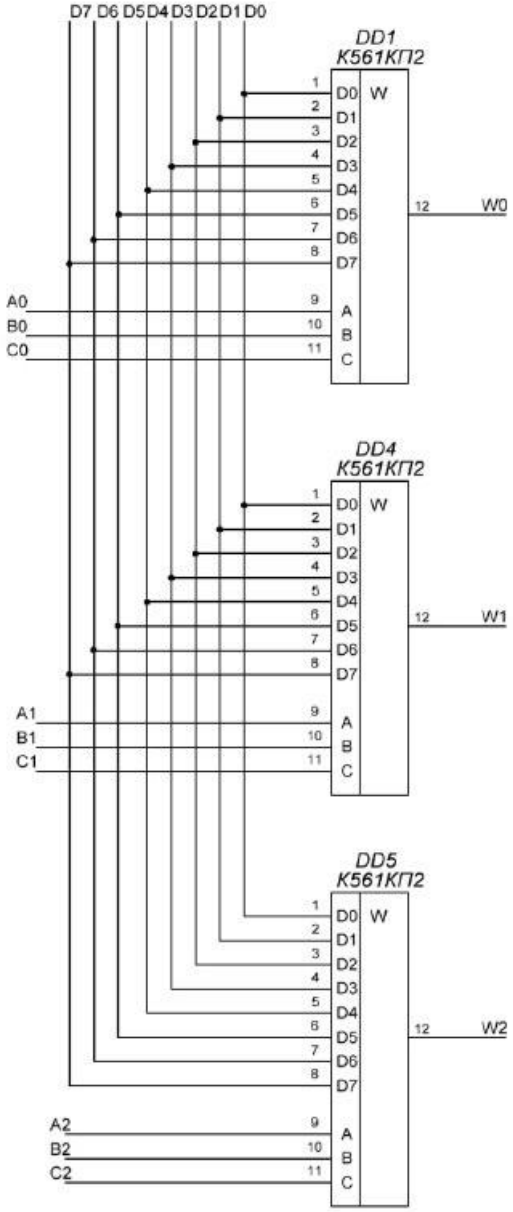

Рисунок 5 – Схема коммутирующего блока

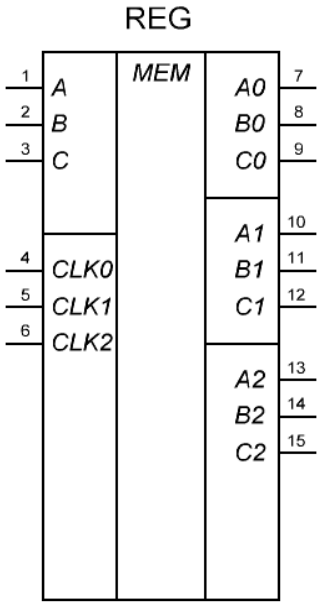

Рисунок 6 – Схема запоминающего блока как цельного устройства

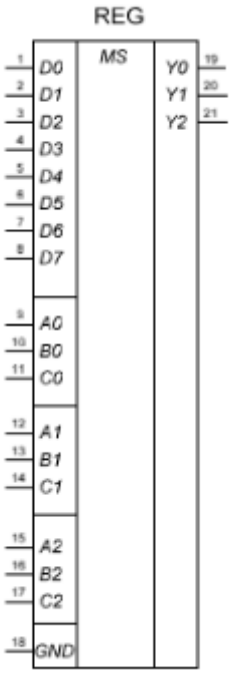

Рисунок 7 – Схема коммутатора как цельного устройства

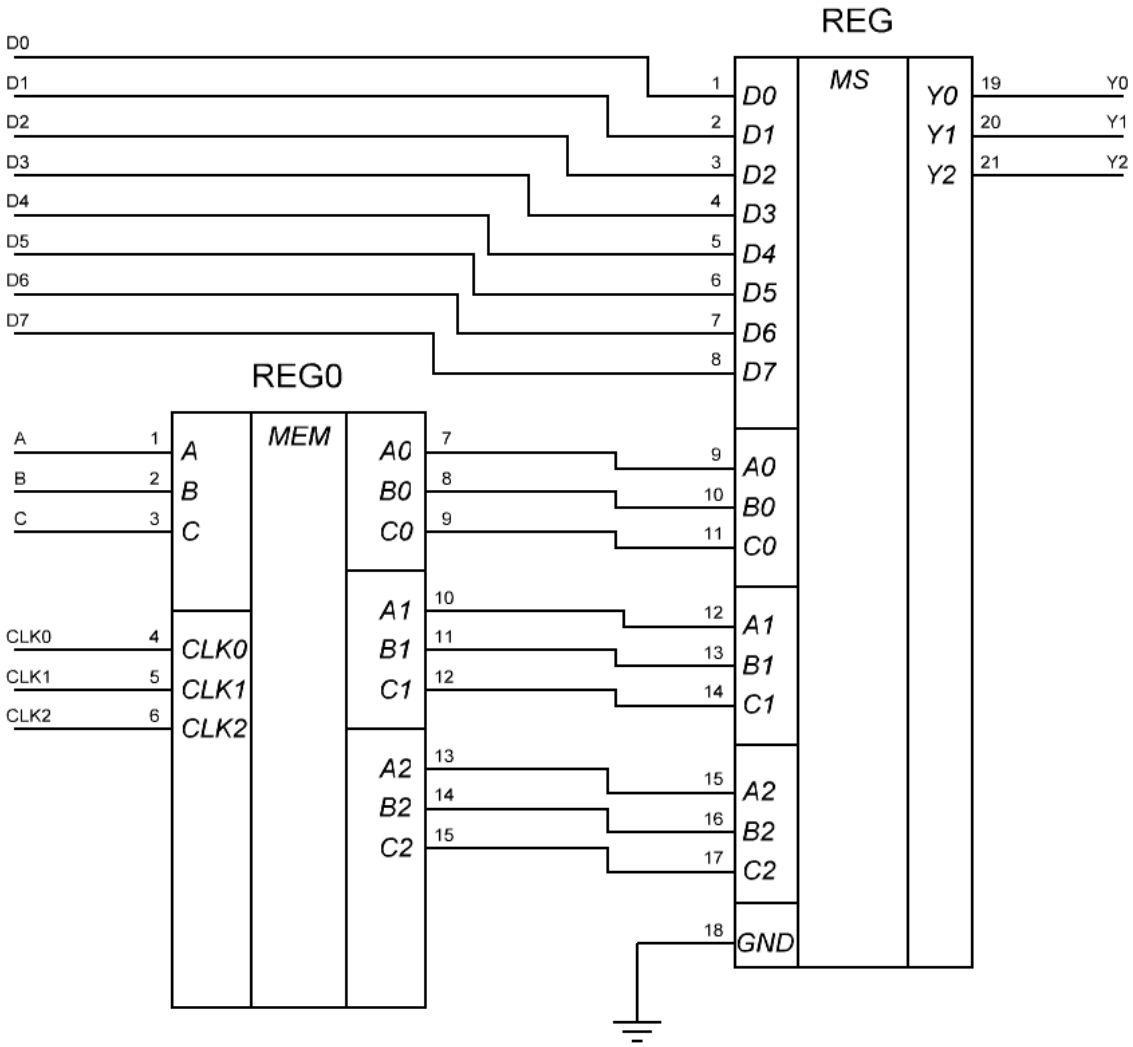

Рисунок 8 – Схема соединения логических блоков

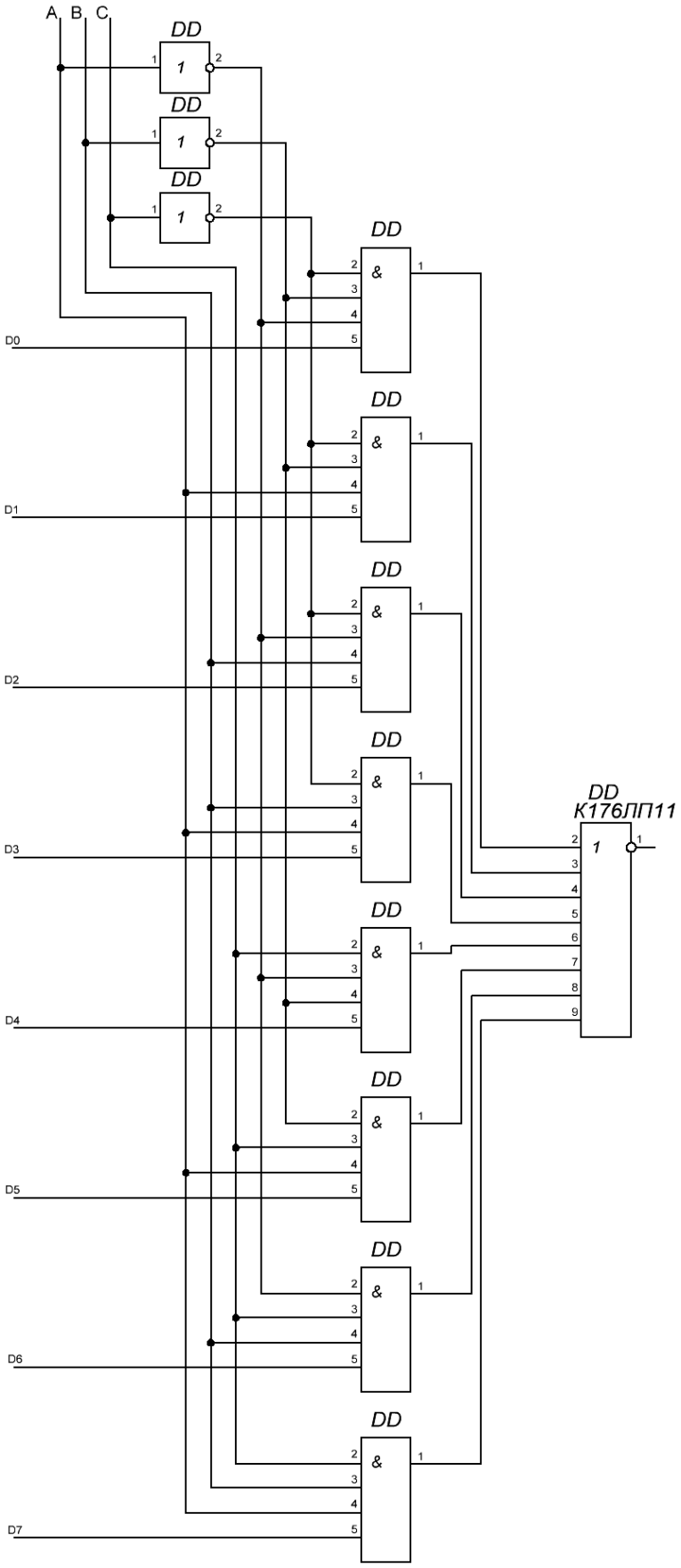

Рисунок 9 – Схема мультиплексора

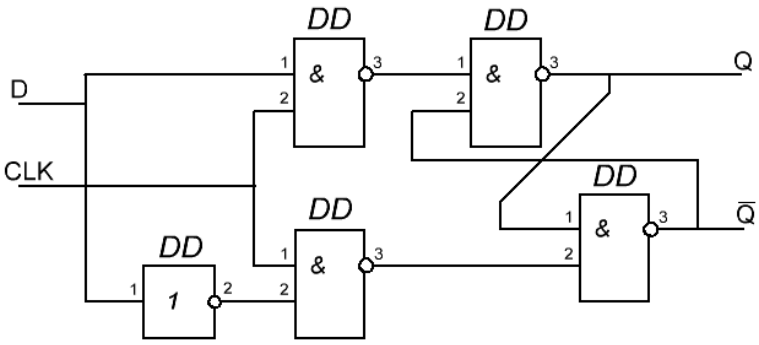

Рисунок 10 – Схема триггера

1.4 Проектирование и моделирование в программе virtuoso

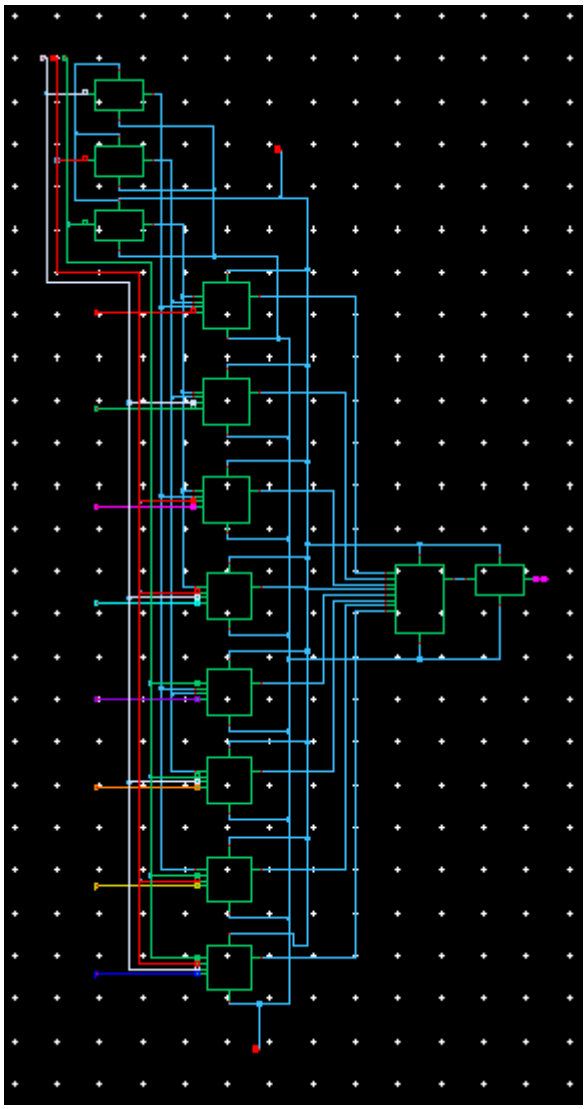

Рисунок 11 – Schematic представление мультиплексора в программе virtuoso

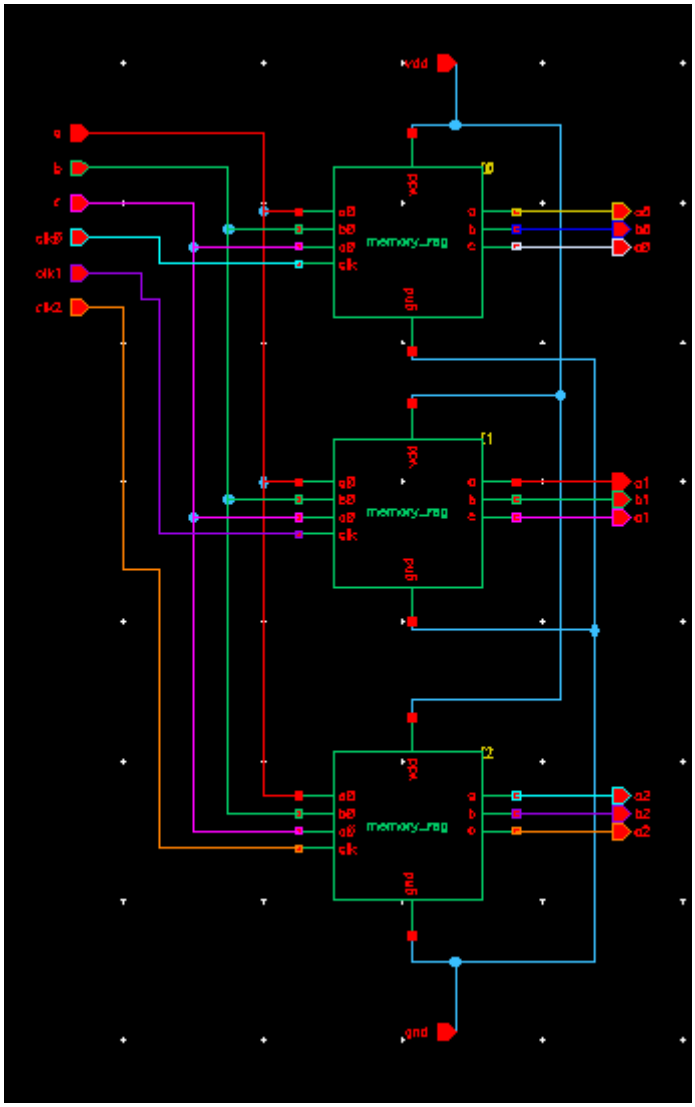

Рисунок 12 – Schematic представление запоминающего регистра в программе virtuoso

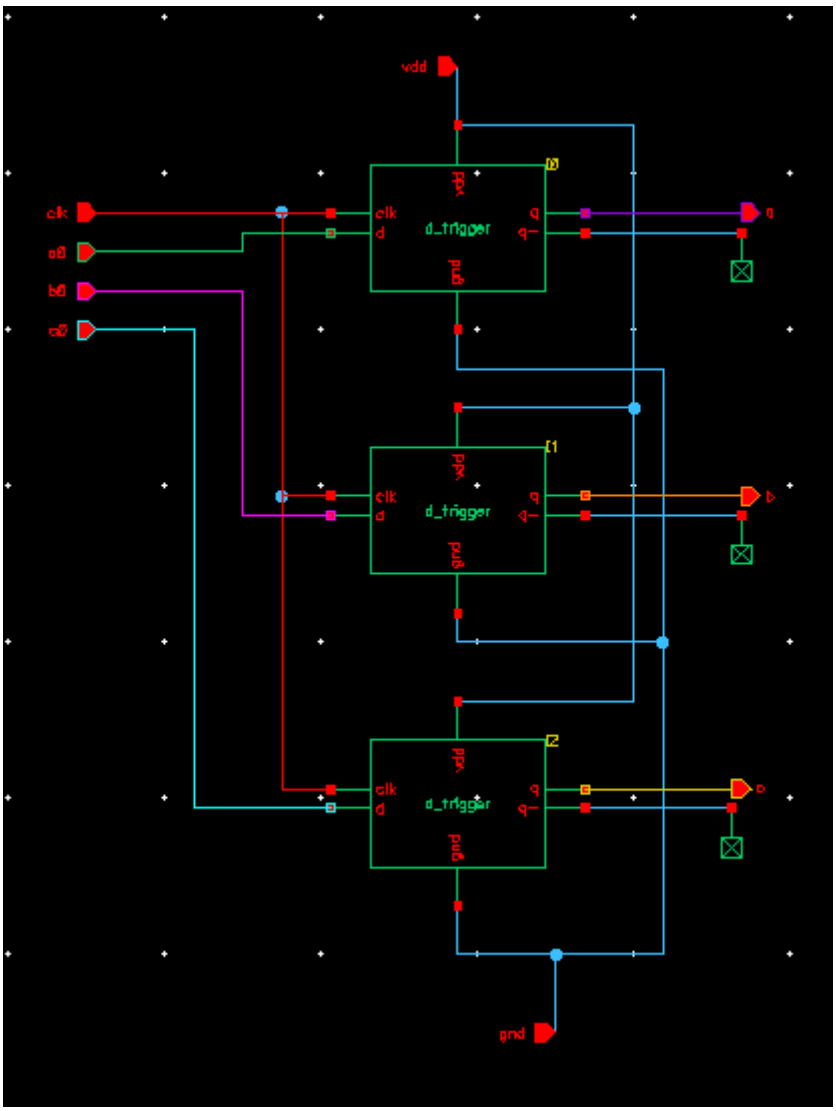

Рисунок 13 – Schematic представление ячейки регистра в программе virtuoso

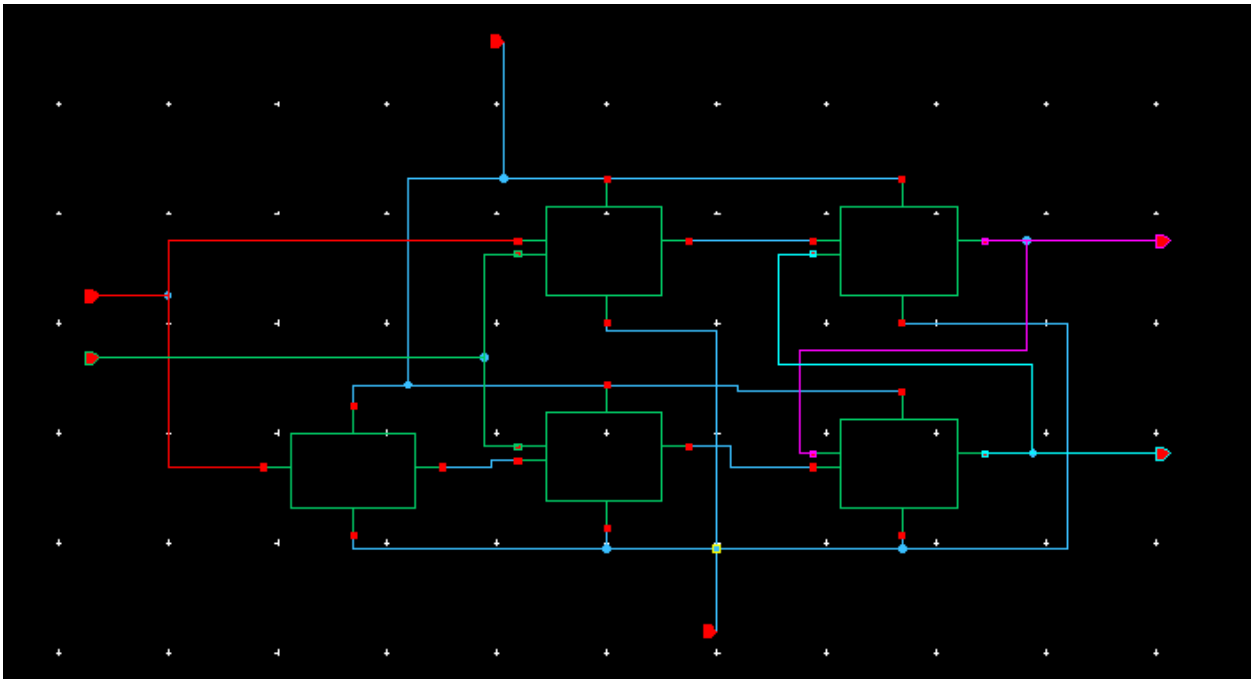

Рисунок 14 – Schematic представление d-триггера в программе virtuoso

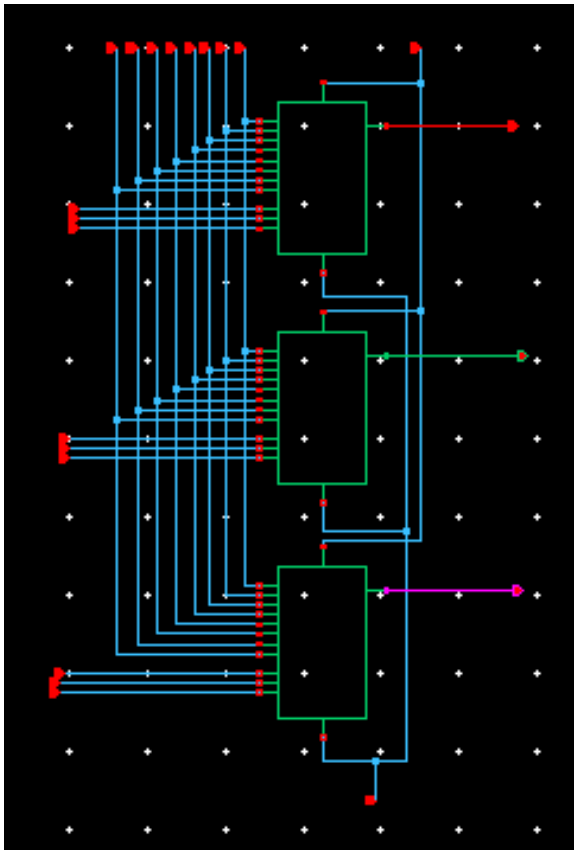

Рисунок 15 – Schematic представление регистра мультиплексоров в программе virtuoso

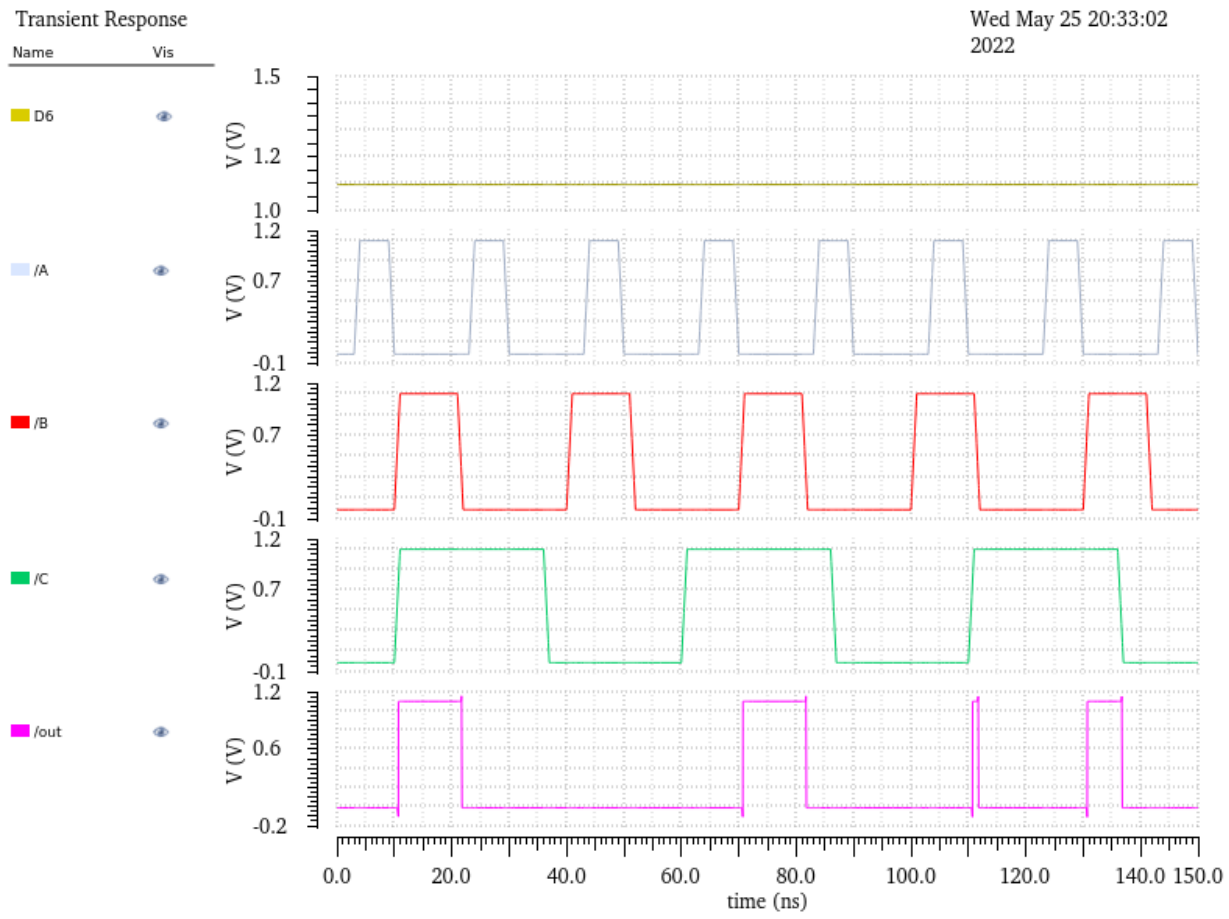

Рисунок 16 – Моделирование мультиплексора в программе virtuoso, для наглядности на вход D6 подаем напряжение логической единицы, остальные входы оставим равными нулю

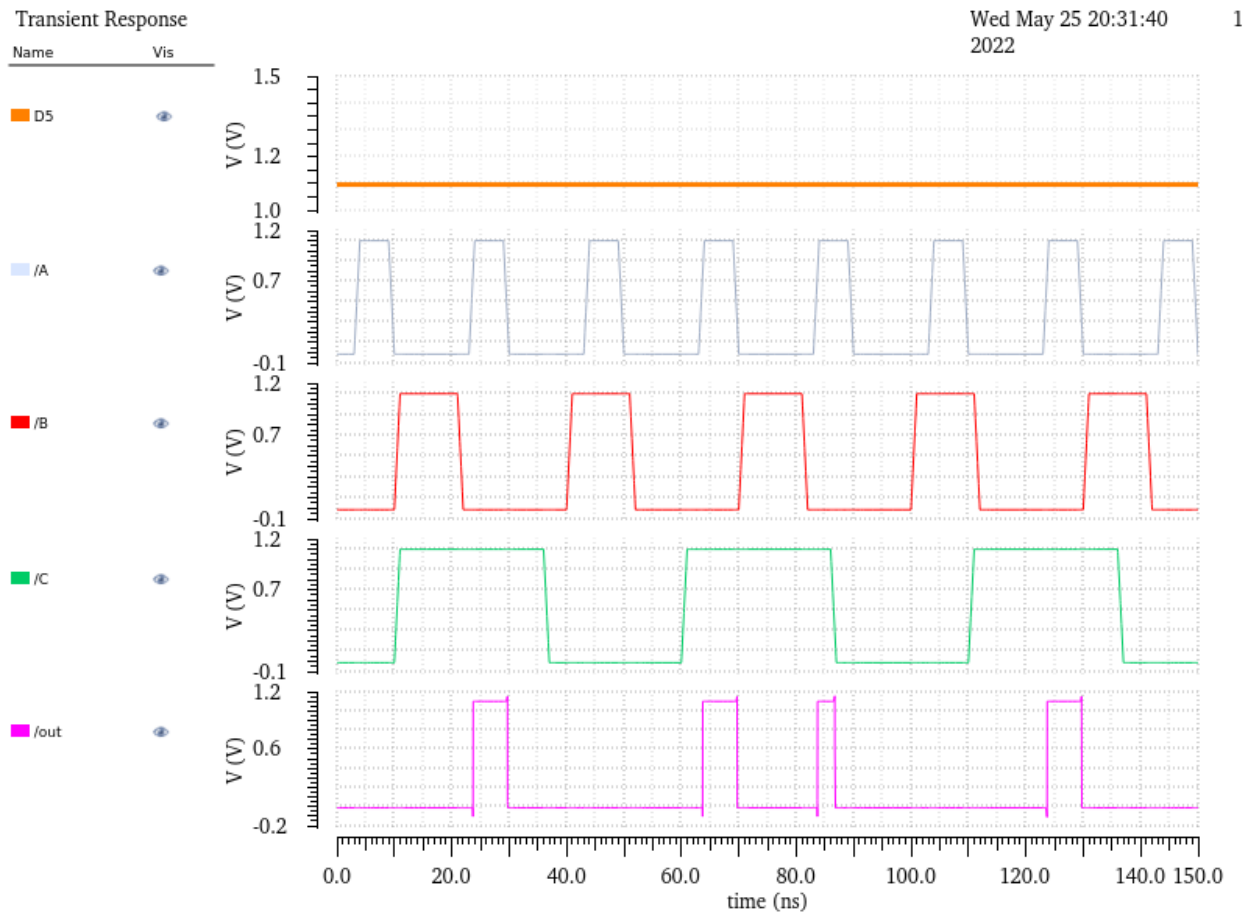

Рисунок 17 – Моделирование мультиплексора в программе virtuoso, для наглядности на вход D5 подаем напряжение логической единицы, остальные входы оставим равными нулю

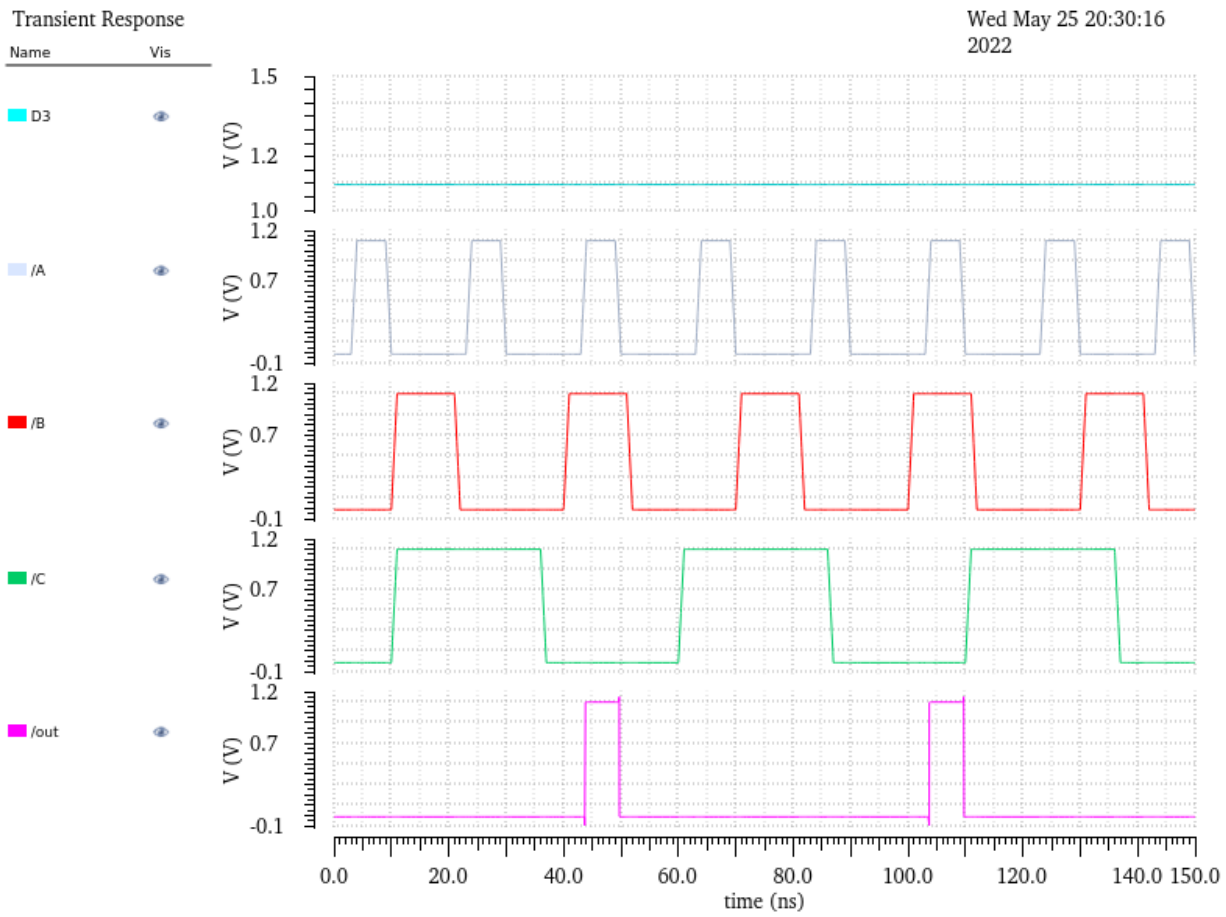

Рисунок 18 – Моделирование мультиплексора в программе virtuoso, для наглядности на вход D3 подаем напряжение логической единицы, остальные входы оставим равными нулю

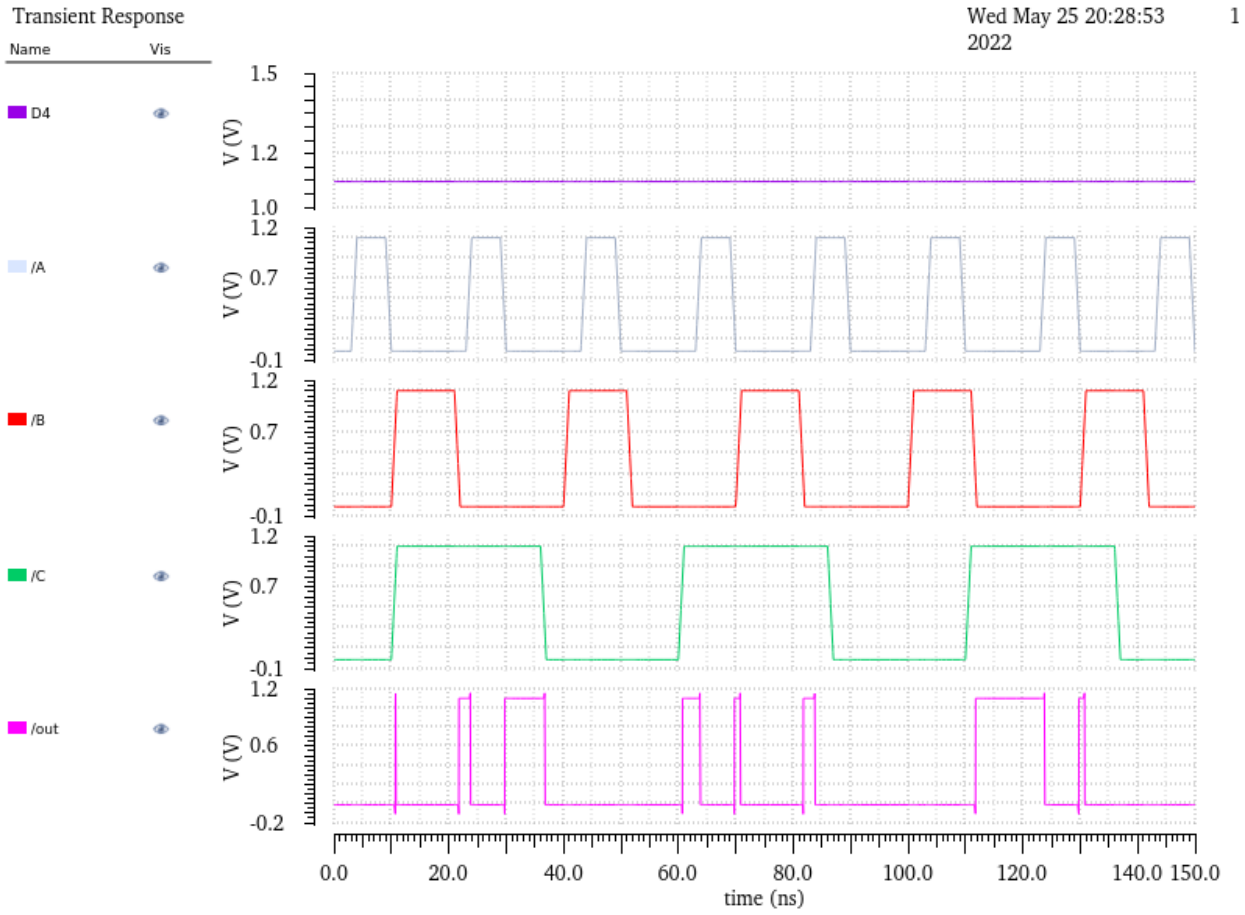

Рисунок 19 – Моделирование мультиплексора в программе virtuoso, для наглядности на вход D4 подаем напряжение логической единицы, остальные входы оставим равными нулю

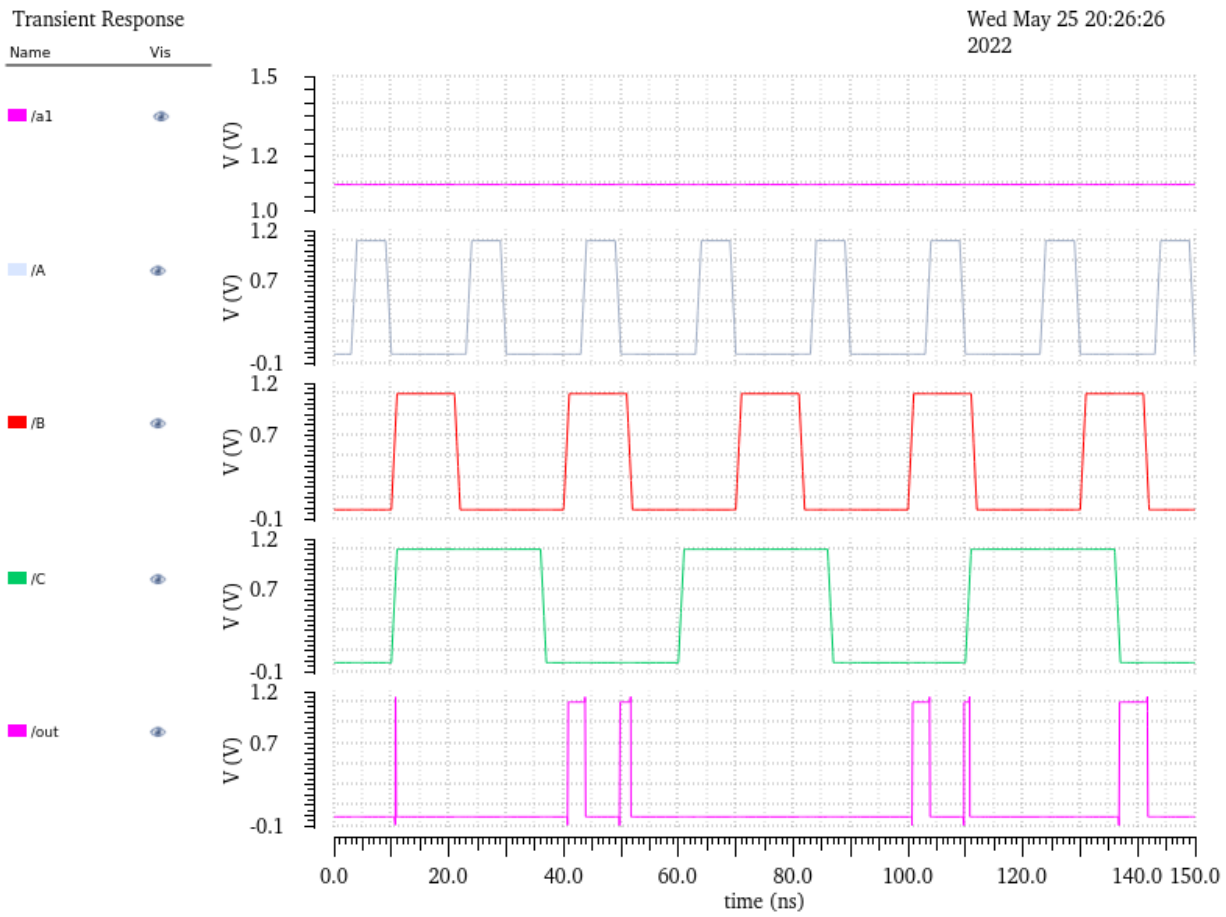

Рисунок 20 – Моделирование мультиплексора в программе virtuoso, для наглядности на вход D2(a1 на графике) подаем напряжение логической единицы, остальные входы оставим равными нулю

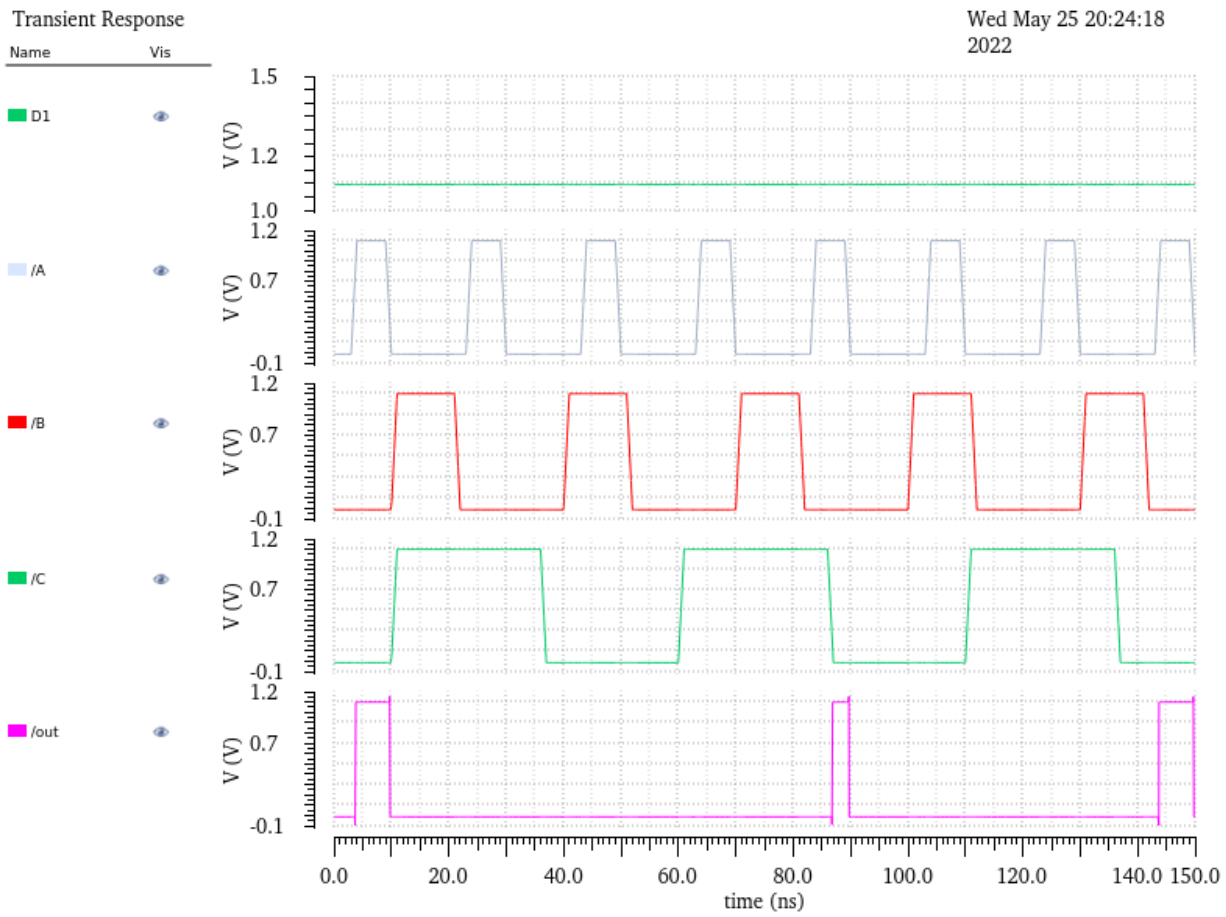

Рисунок 21 – Моделирование мультиплексора в программе virtuoso, для наглядности на вход D1 подаем напряжение логической единицы, остальные входы оставим равными нулю

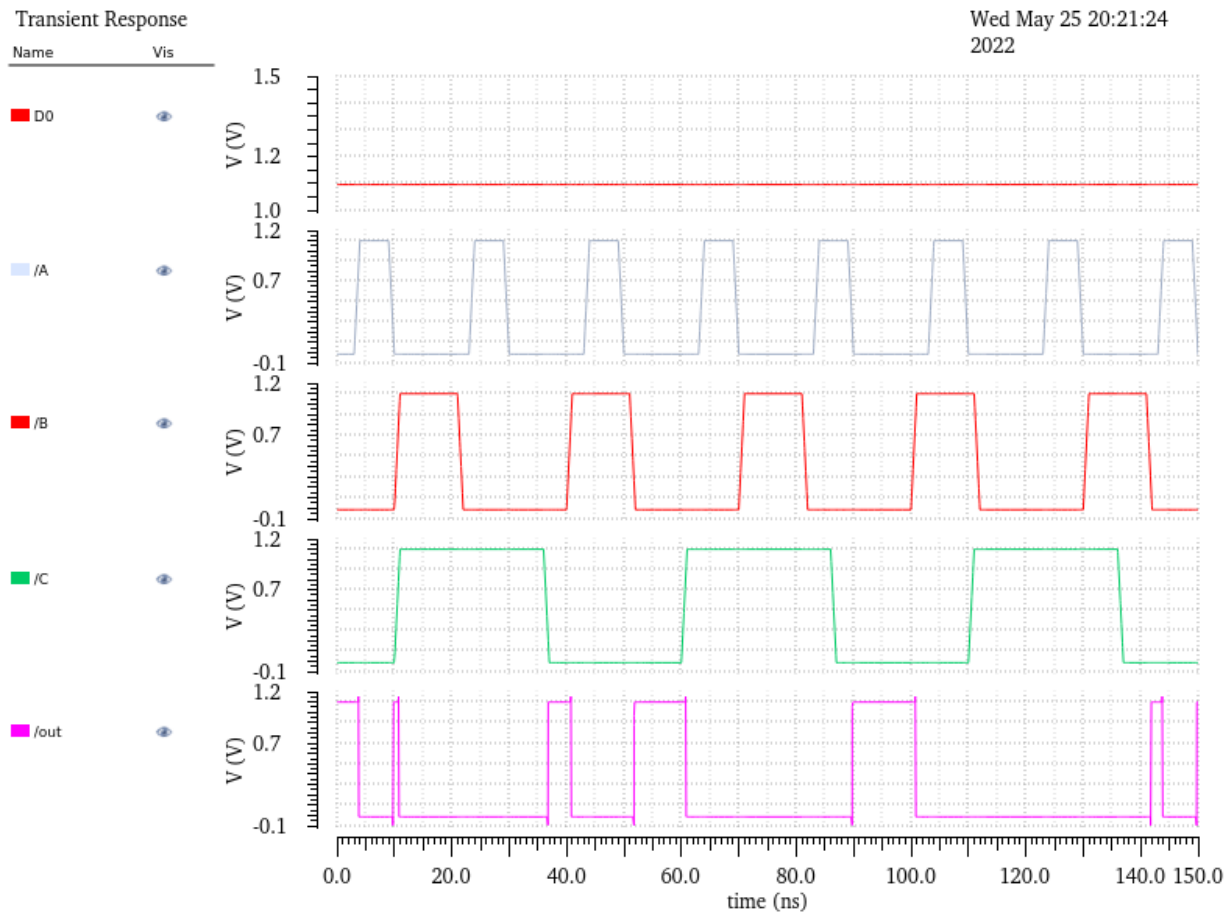

Рисунок 22 – Моделирование мультиплексора в программе virtuoso, для наглядности на вход D0 подаем напряжение логической единицы, остальные входы оставим равными нулю

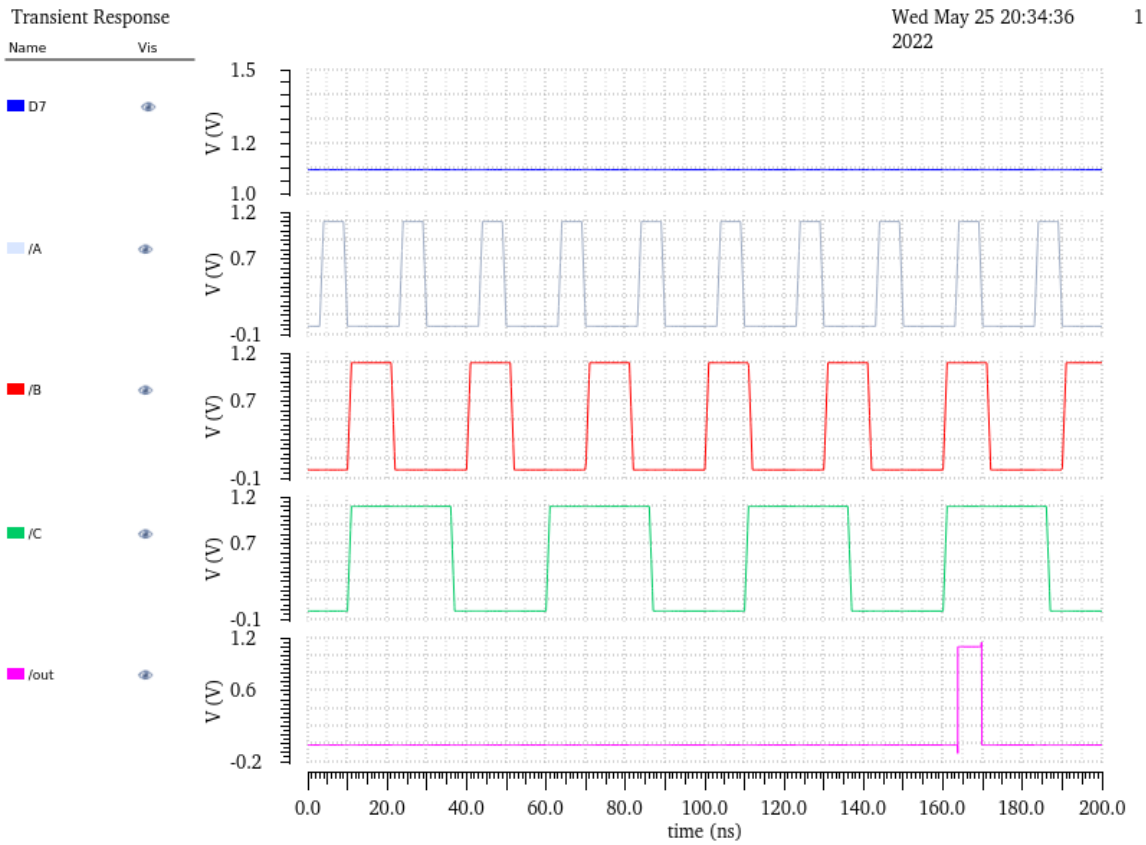

Рисунок 23 – Моделирование мультиплексора в программе virtuoso, для наглядности на вход D7 подаем напряжение логической единицы, остальные входы оставим равными нулю

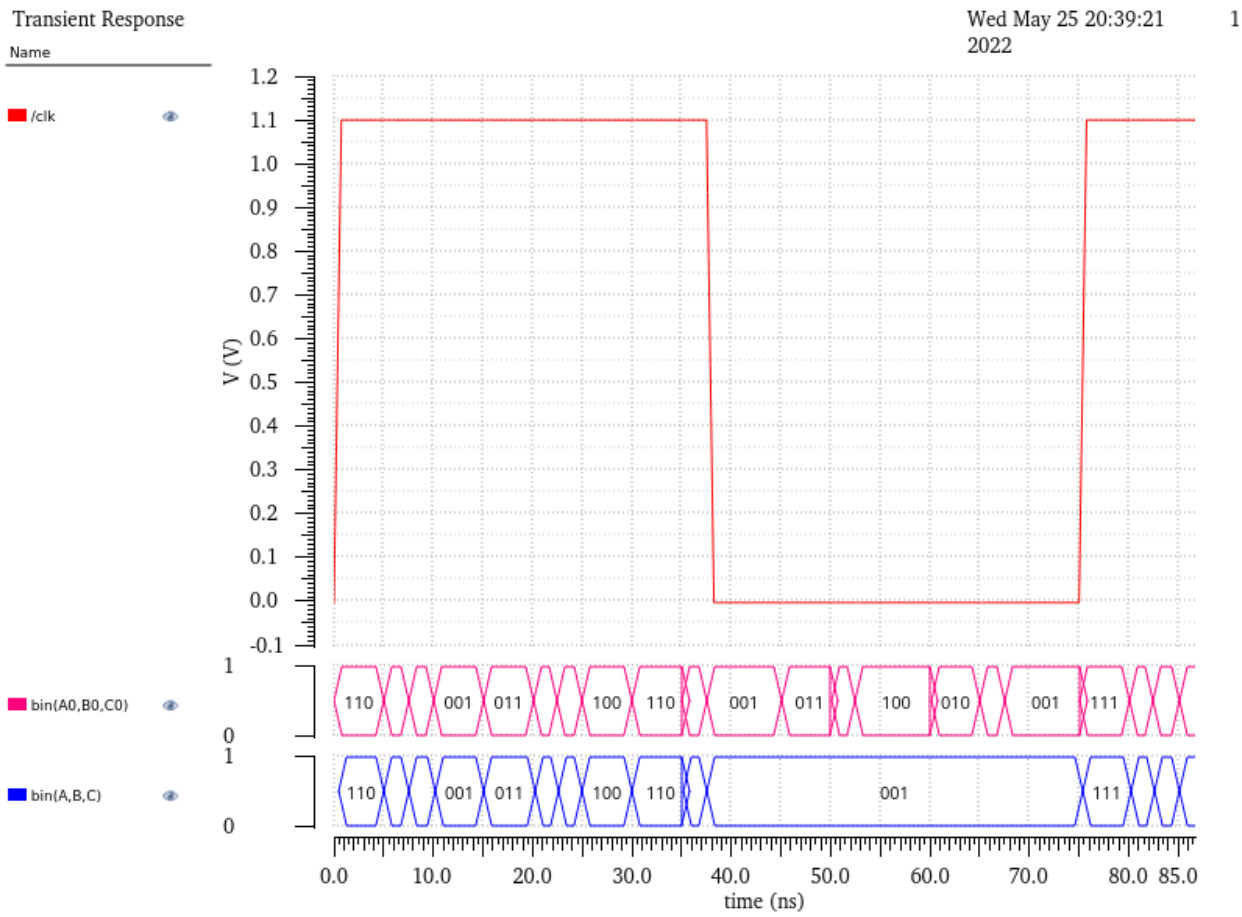

Рисунок 24 – Моделирование работы ячейки запоминающего регистра в программе virtuoso

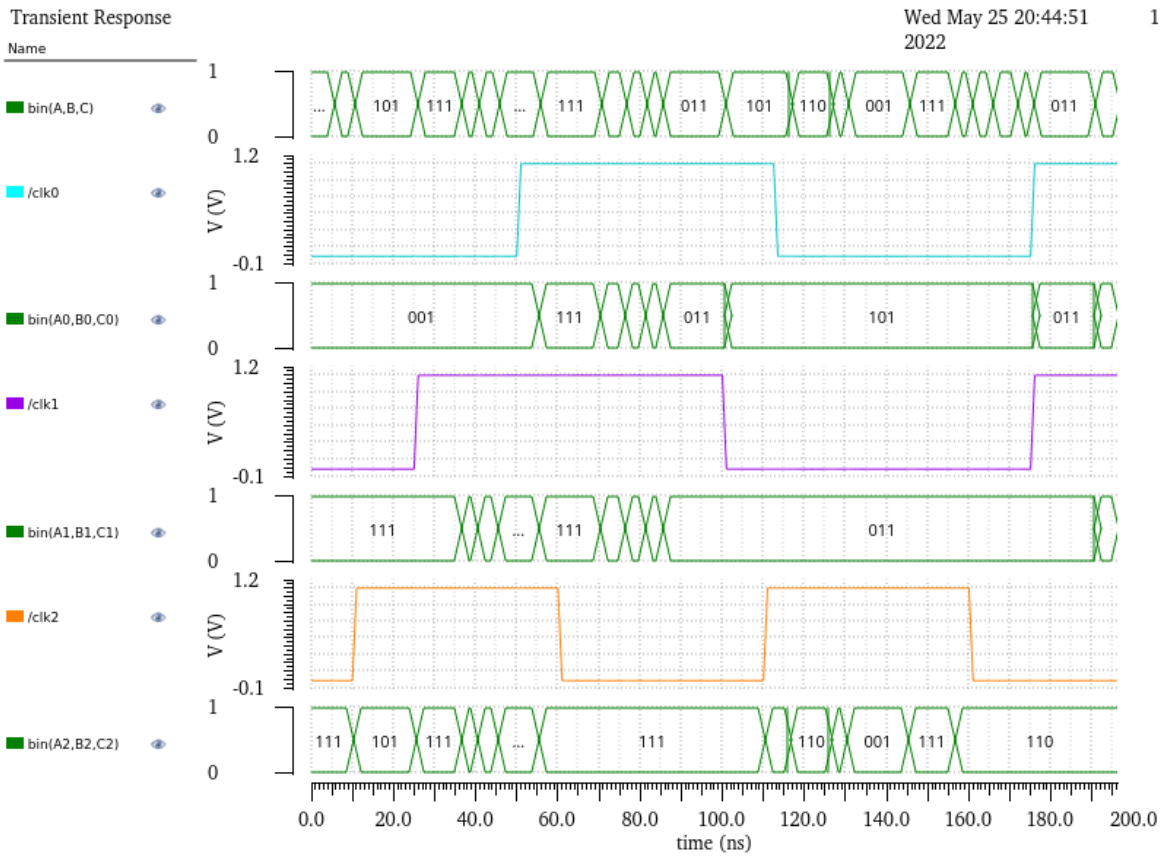

Рисунок 25 – Моделирование работы запоминающего регистра в программе virtuoso

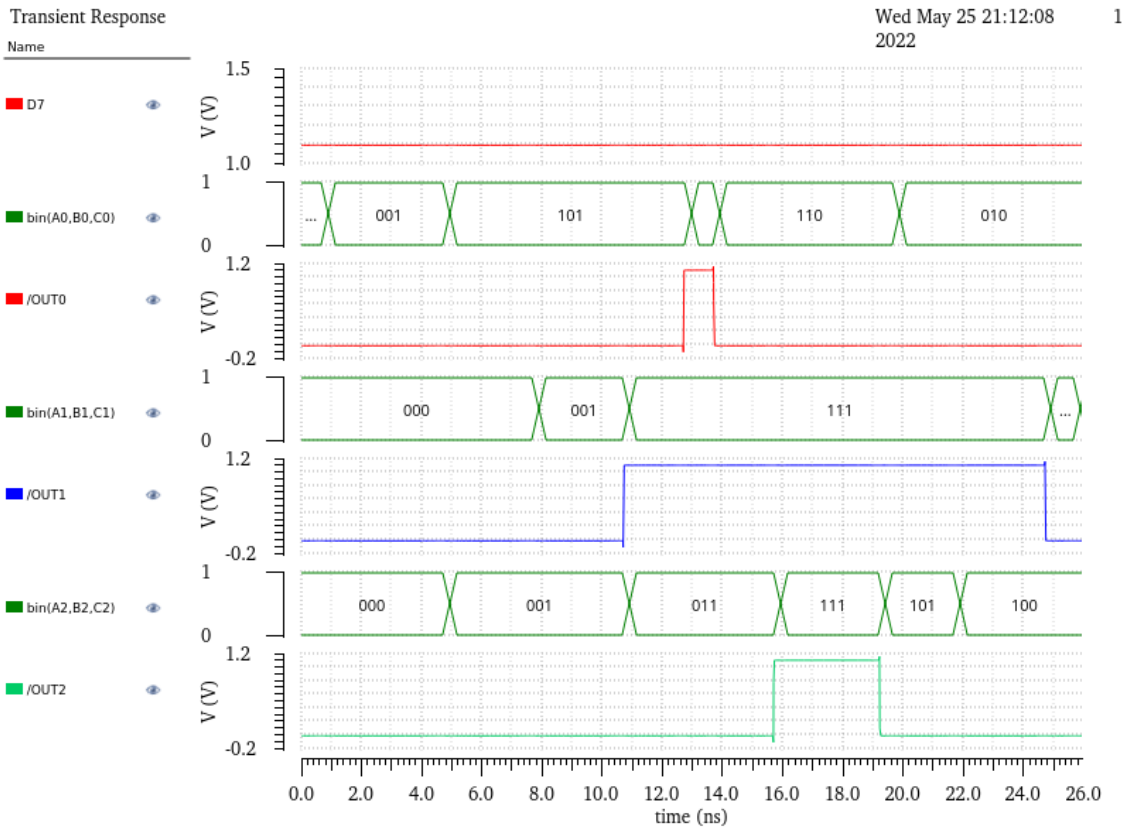

Рисунок 26 – Моделирование работы блока мультиплексоров в программе virtuoso, на вход D7 подаем 1, на остальные входы 0

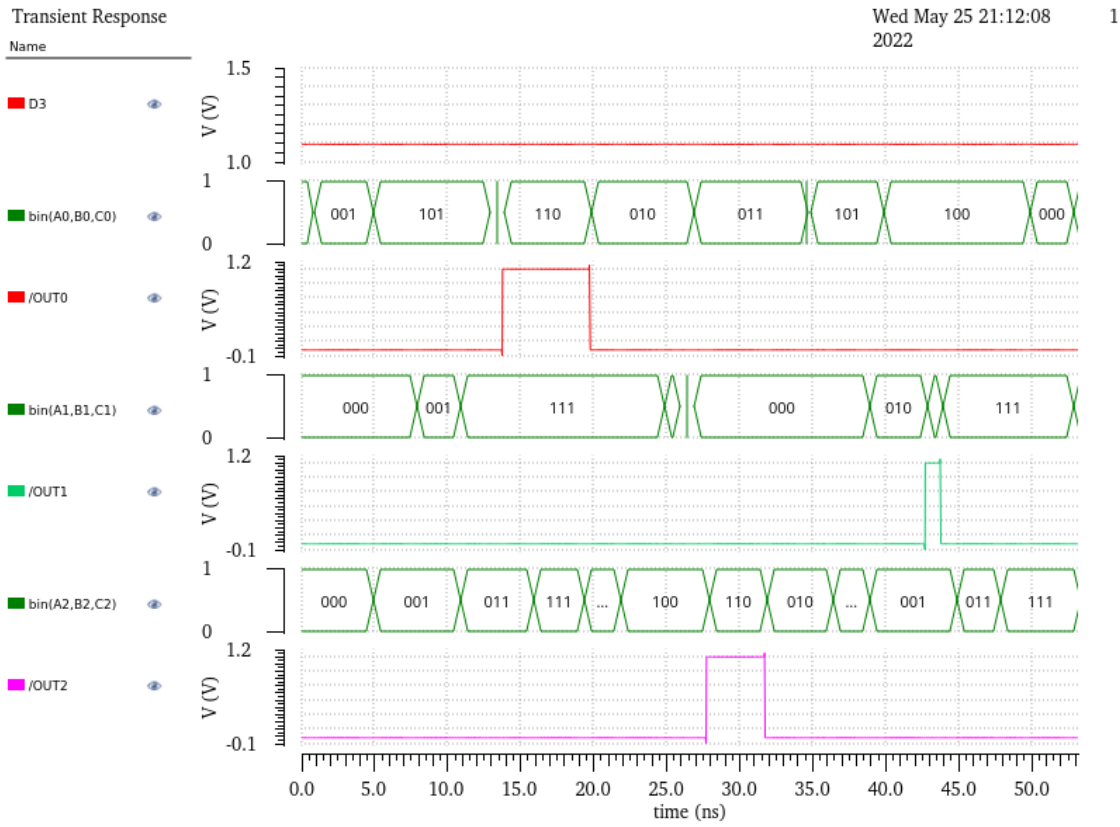

Рисунок 27 – Моделирование работы блока мультиплексоров в программе virtuoso, на вход D3 подаем 1, на остальные входы 0

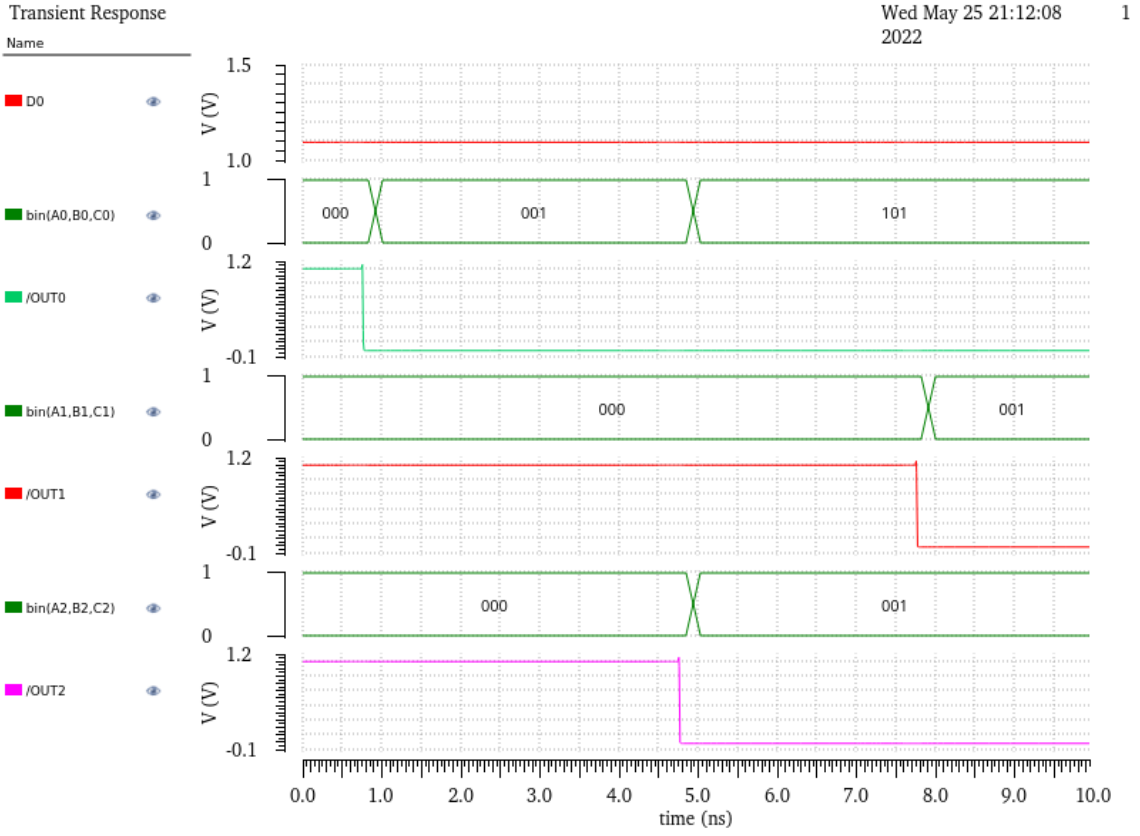

Рисунок 28 – Моделирование работы блока мультиплексоров в программе virtuoso, на вход D0 подаем 1, на остальные входы 0

1.5 Работа с verilog-описанием, его синтез и моделирование

Поведенческое verilog – описание устройства commutator.v:

```
`timescale 1ns/1ns

module multiplexer_8_to_1(d,control,out);
input [2:0]control;
input [7:0]d;
output reg out;
always @ (d[0] or d[1] or d[2] or d[3] or d[4] or d[5] or d[6] or d[7] or control[0] or control[1]) begin
case(control)
3'b000 : out <= d[0];
3'b100 : out <= d[1];
3'b010 : out <= d[2];
3'b110 : out <= d[3];
3'b001 : out <= d[4];
3'b101 : out <= d[5];
3'b011 : out <= d[6];
3'b111 : out <= d[7];
default : out <= 0;
endcase
end
endmodule

module mx_reg(d,control,out);
input [7:0] d;
input [2:0][2:0] control;
output [2:0] out;
multiplexer_8_to_1 dd0(.d(d), .control(control[0]), .out(out[0]));
multiplexer_8_to_1 dd1(.d(d), .control(control[1]), .out(out[1]));
multiplexer_8_to_1 dd2(.d(d), .control(control[2]), .out(out[2]));
endmodule

module d_trigger(d,clk,q,qbar);
output reg q;
output qbar;
input d, clk;
assign qbar = ~q;
always @ (d or clk) begin
if (clk)
#1 q = d;
end
endmodule

module mem_reg(control,clk,out);
input clk;
input [2:0] control;
output [2:0] out;
d_trigger dd0(.d(control[0]), .clk(clk), .q(out[0]), .qbar());
d_trigger dd1(.d(control[1]), .clk(clk), .q(out[1]), .qbar());
d_trigger dd2(.d(control[2]), .clk(clk), .q(out[2]), .qbar());
endmodule

module mem3_reg(control,clk,out);
input [2:0]clk;
input [2:0] control;
output [2:0][2:0] out;
mem_reg dd0(.control(control), .clk(clk[0]), .out(out[0]));
mem_reg dd1(.control(control), .clk(clk[1]), .out(out[1]));
mem_reg dd2(.control(control), .clk(clk[2]), .out(out[2]));
endmodule

module commutator(inputs,control,clk,outputs);
input [2:0]clk;
input [2:0] control;
input [7:0] inputs;
```

```
output [2:0] outputs;
wire [2:0][2:0] mem_out;
mem3_reg dd0(.control(control),.clk(clk),.out(mem_out));
mx_reg dd1(.d(inputs),.control(mem_out),.out(outputs));
endmodule
```

Описание testbench – файла для коммутатора:

```
`include "commutator.v"
`timescale 1ns/1ns

module commutator_test;
reg [2:0] control;
reg [2:0]clk;
reg [7:0]inputs;
wire [2:0]outputs;
initial begin
clk <= 3'b000;
control <= 3'b000;
inputs <= 8'b00000000;
end
commutator dd0(.inputs(inputs),.control(control),.clk(clk),.outputs(outputs));

always #50 clk[0] = ~clk[0];
always #35 clk[1] = ~clk[1];
always #43 clk[2] = ~clk[2];

always #30 control[0] = ~control[0];
always #27 control[1] = ~control[1];
always #25 control[2] = ~control[2];

always #12 inputs[0] = ~inputs[0];
always #11 inputs[1] = ~inputs[1];
always #10 inputs[2] = ~inputs[2];
always #9 inputs[3] = ~inputs[3];
always #8 inputs[4] = ~inputs[4];
always #7 inputs[5] = ~inputs[5];
always #4 inputs[6] = ~inputs[6];
always #2 inputs[7] = ~inputs[7];
initial begin
$dumpfile("commutator_test.vcd");
$dumpvars;
$display("test complete");
#400;
$finish;
end
endmodule
```

Синтезированное в программе genus verilog – описание коммутатора:

```
// Generated by Cadence Genus(TM) Synthesis Solution 19.10-p002_1
// Generated on: May 25 2022 20:10:28 MSK (May 25 2022 17:10:28 UTC)

// Verification Directory fv/commutator

module commutator(inputs, control, clk, outputs);
input [7:0] inputs;
input [2:0] control, clk;
output [2:0] outputs;
wire [7:0] inputs;
wire [2:0] control, clk;
wire [2:0] outputs;
wire [8:0] mem_out;
wire UNCONNECTED, UNCONNECTED0, UNCONNECTED1, UNCONNECTED2,
UNCONNECTED3, UNCONNECTED4, UNCONNECTED5, UNCONNECTED6;
wire UNCONNECTED7, n_0, n_1, n_2, n_3, n_4, n_5, n_6;
wire n_7, n_8;
MXI4XL g754(.A (n_0), .B (n_1), .C (n_3), .D (n_4), .S0 (mem_out[4]),
```

```
.S1 (mem_out[5]), .Y (outputs[1]));
MX2XL g752(.A (n_7), .B (n_6), .S0 (mem_out[8]), .Y (outputs[2]));
MX2XL g753(.A (n_8), .B (n_5), .S0 (mem_out[1]), .Y (outputs[0]));
MX4XL g756(.A (inputs[0]), .B (inputs[1]), .C (inputs[4]), .D
(inputs[5]), .S0 (mem_out[2]), .S1 (mem_out[0]), .Y (n_8));
MX4XL g757(.A (inputs[0]), .B (inputs[2]), .C (inputs[4]), .D
(inputs[6]), .S0 (mem_out[7]), .S1 (mem_out[6]), .Y (n_7));
MX4XL g755(.A (inputs[1]), .B (inputs[5]), .C (inputs[3]), .D
(inputs[7]), .S0 (mem_out[6]), .S1 (mem_out[7]), .Y (n_6));
MX4XL g758(.A (inputs[2]), .B (inputs[6]), .C (inputs[3]), .D
(inputs[7]), .S0 (mem_out[0]), .S1 (mem_out[2]), .Y (n_5));
AOI22XL g759(.A0 (inputs[3]), .A1 (mem_out[3]), .B0 (inputs[7]), .B1
(n_2), .Y (n_4));
AOI22XL g760(.A0 (inputs[1]), .A1 (mem_out[3]), .B0 (inputs[5]), .B1
(n_2), .Y (n_3));
AOI22XL g761(.A0 (inputs[2]), .A1 (mem_out[3]), .B0 (inputs[6]), .B1
(n_2), .Y (n_1));
AOI22XL g762(.A0 (inputs[0]), .A1 (mem_out[3]), .B0 (inputs[4]), .B1
(n_2), .Y (n_0));
INVXL g767(.A (mem_out[3]), .Y (n_2));
TLATX1 dd0_dd0_dd1_q_reg(.G (clk[0]), .D (control[1]), .Q
(mem_out[1]), .QN (UNCONNECTED));
TLATX1 dd0_dd2_dd1_q_reg(.G (clk[2]), .D (control[1]), .Q
(mem_out[7]), .QN (UNCONNECTED0));
TLATX1 dd0_dd2_dd0_q_reg(.G (clk[2]), .D (control[0]), .Q
(mem_out[6]), .QN (UNCONNECTED1));
TLATX1 dd0_dd0_dd0_q_reg(.G (clk[0]), .D (control[0]), .Q
(mem_out[0]), .QN (UNCONNECTED2));
TLATX1 dd0_dd1_dd2_q_reg(.G (clk[1]), .D (control[2]), .Q
(mem_out[5]), .QN (UNCONNECTED3));
TLATX1 dd0_dd1_dd1_q_reg(.G (clk[1]), .D (control[1]), .Q
(mem_out[4]), .QN (UNCONNECTED4));
TLATX1 dd0_dd2_dd2_q_reg(.G (clk[2]), .D (control[2]), .Q
(mem_out[8]), .QN (UNCONNECTED5));
TLATX1 dd0_dd0_dd2_q_reg(.G (clk[0]), .D (control[2]), .Q
(mem_out[2]), .QN (UNCONNECTED6));
TLATX1 dd0_dd1_dd0_q_reg(.G (clk[1]), .D (control[0]), .Q
(UNCONNECTED7), .QN (mem_out[3]));
endmodule
```

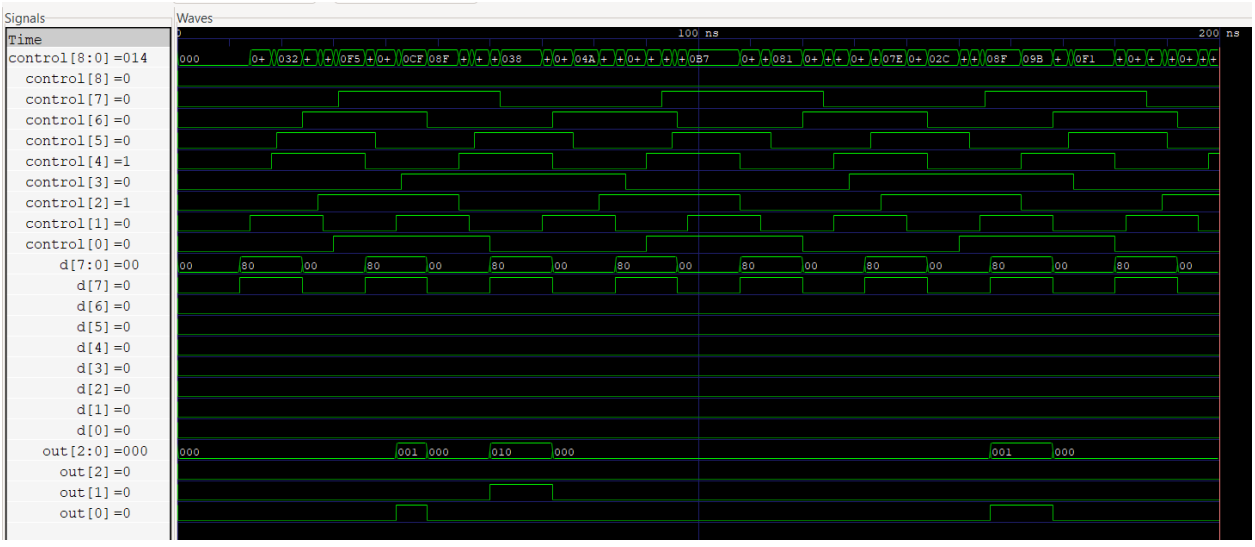

Рисунок 29 – Моделирование работы блока мультплексоров в программе gtkwave, для понятности импульс подаем лишь на вход D7, управляющие и синхронизирующие сигналы подобраны случайно, временные задержки работы устройства не учтены

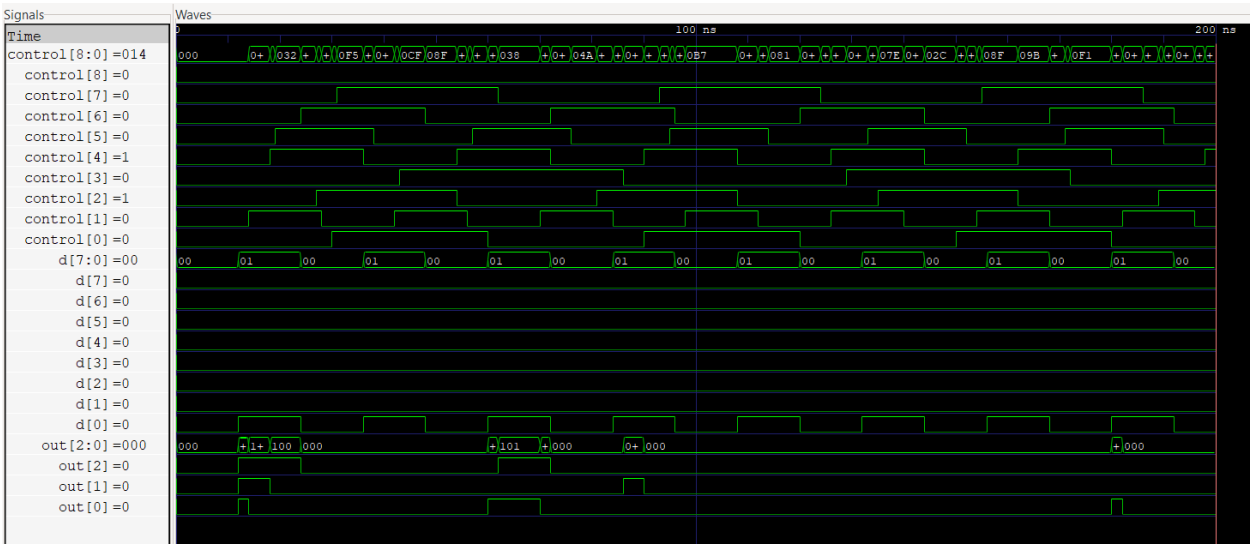

Рисунок 30 – Моделирование работы блока мультиплексоров в программе gtkwave, для понятности импульс подаем лишь на вход D0

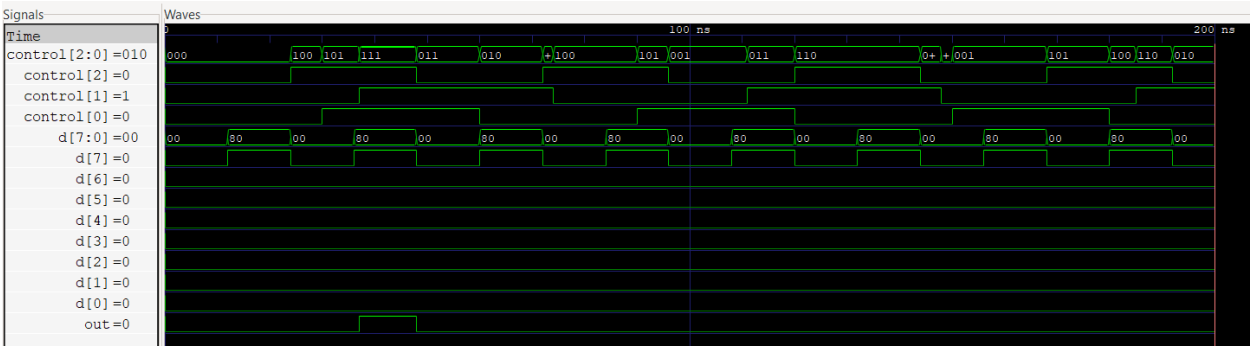

Рисунок 31 – Моделирование работы в программе gtkwave, для понятности импульс подаем лишь на вход D7

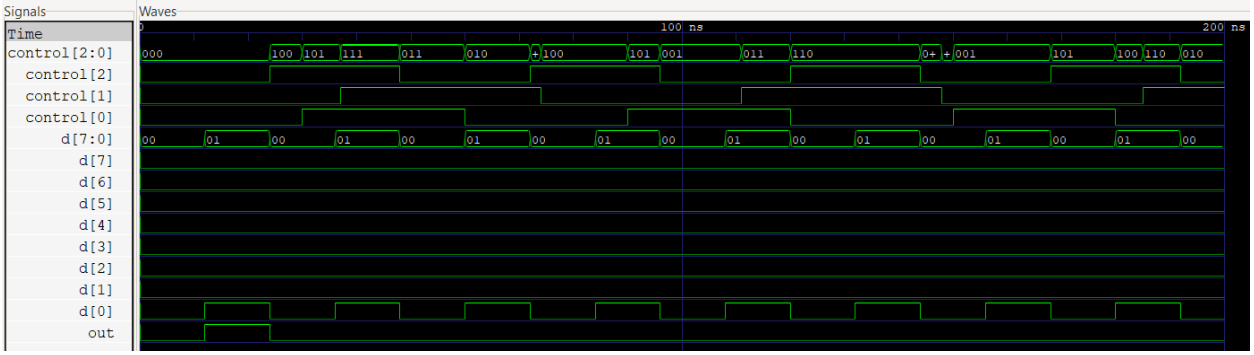

Рисунок 32 – Моделирование работы мультиплексора в программе gtkwave, для понятности импульс подаем лишь на вход D0

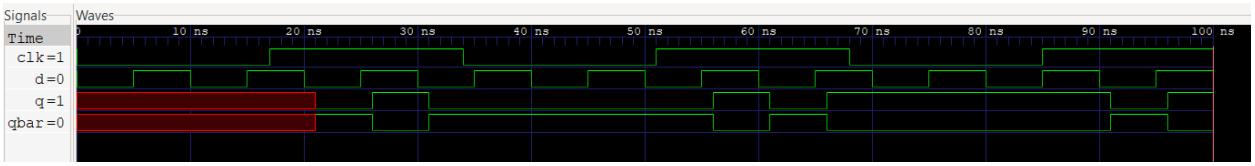

Рисунок 33 – Моделирование работы d-триггера в программе gtkwave

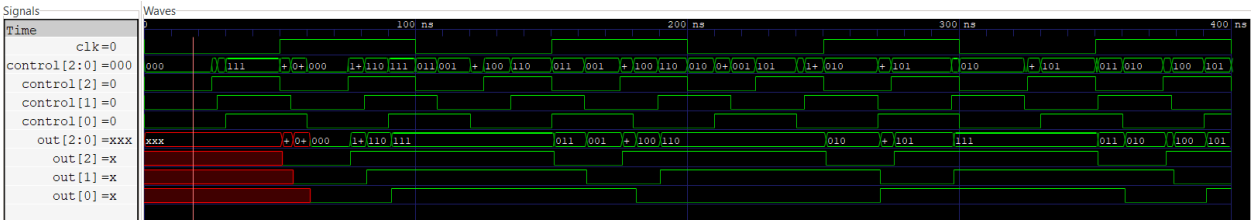

Рисунок 34 – Моделирование работы ячейки запоминающего регистра в программе gtkwave

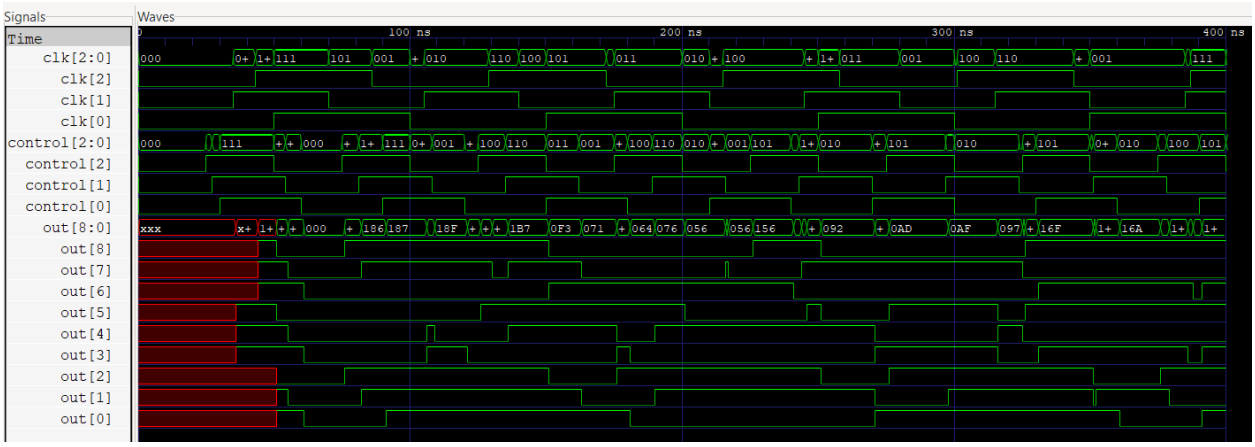

Рисунок 35 – Моделирование работы запоминающего регистра в программе gtkwave

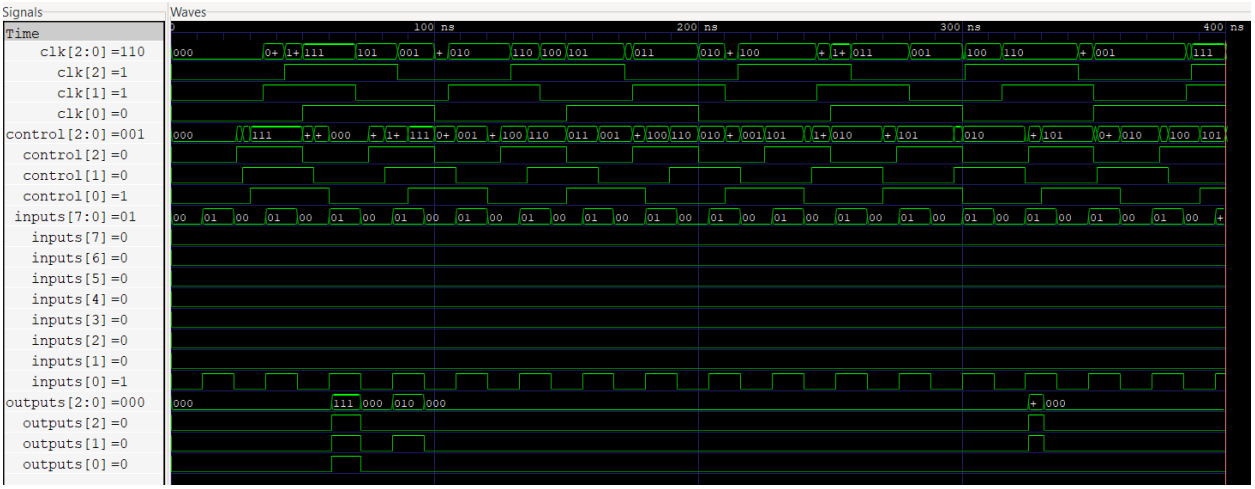

Рисунок 35 – Моделирование работы коммутатора в программе gtkwave, для наглядности импульс подаем лишь на первый вход

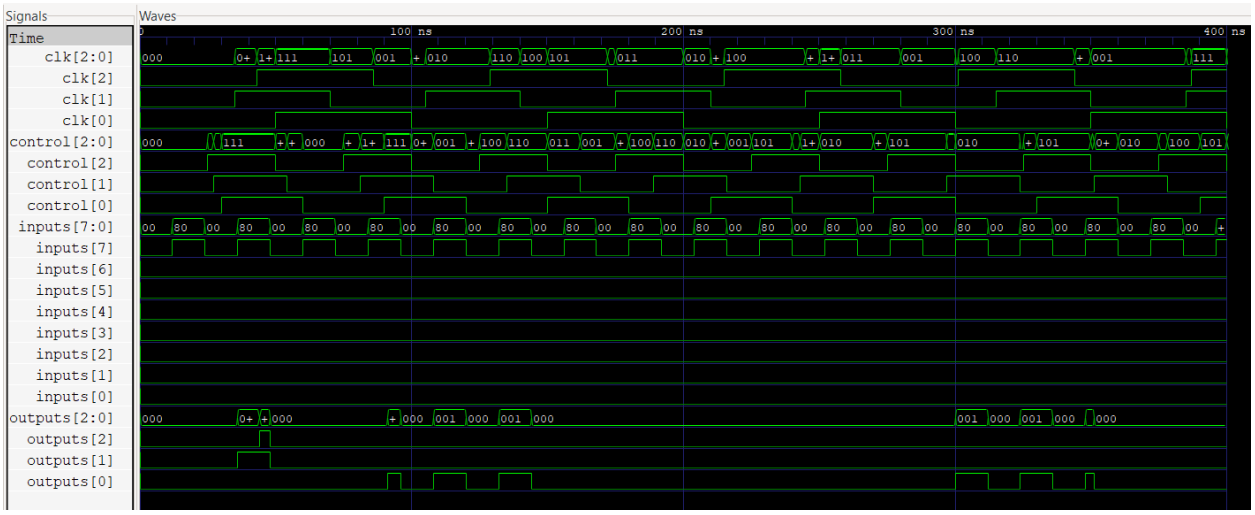

Рисунок 36 – Моделирование работы коммутатора в программе gtkwave, для наглядности импульс подаем лишь на последний вход

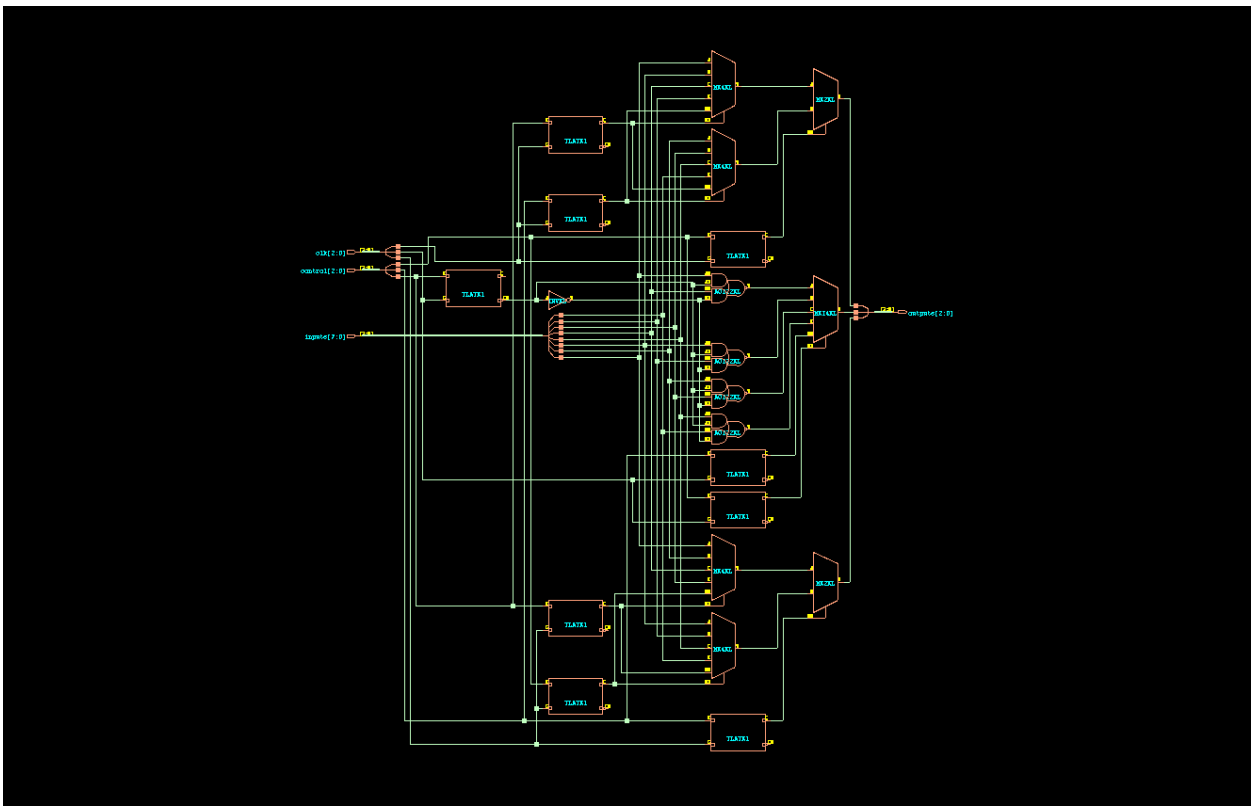

Рисунок 37 – Schematic-представление, сгенерированное в программе genus, как мы видим, базовыми примитивами программы были выбраны не 8к1 мультиплексоры, а 4к1, вероятнее всего по причине отстутствия в базовых библиотеках данного мультиплексора

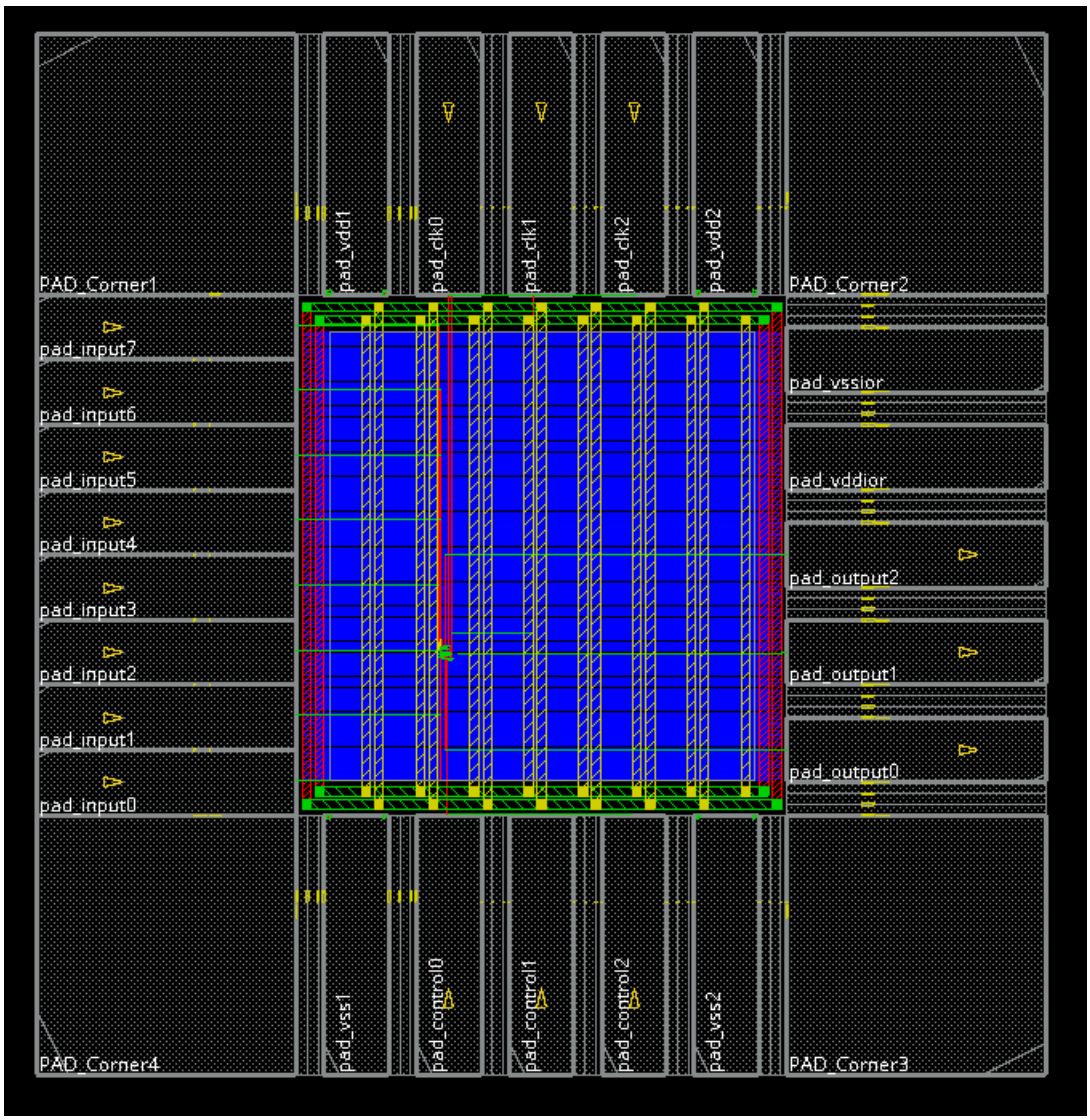

Рисунок 38 – Коммутатор, собранный в программе innovus, угловые площадки не используются, слева у нас 8 входов коммутатора, сверху 2 контакта для питания VDD и 3 контакта CLK, справа 3 выхода коммутатора, контакты VSSIOR и VDDIOR, снизу 3 контакта CONTROL и 2 контакта для земли

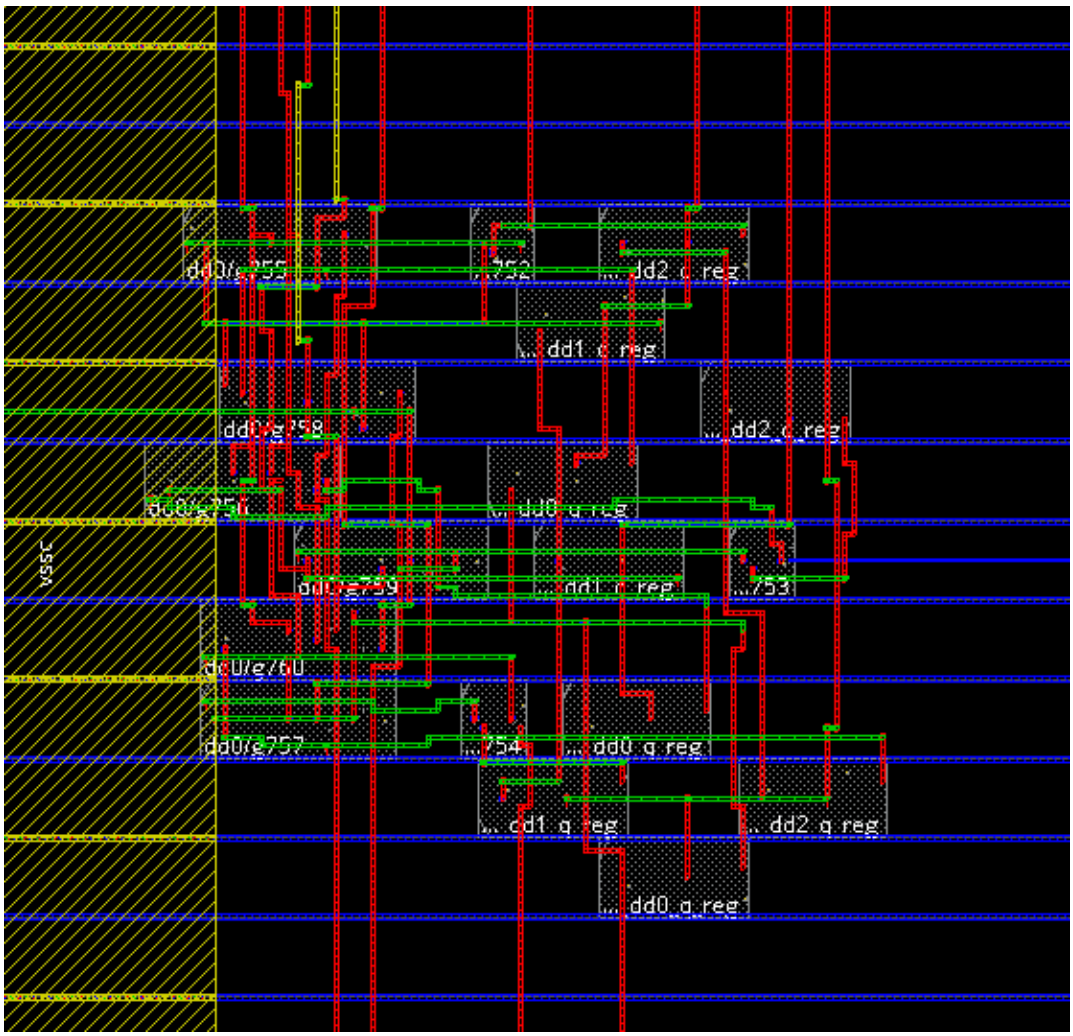

Рисунок 39 – Layout-представление, автоматически собранное в ядре микросхемы, схема прошла проверку DRC-анализа

ЗАКЛЮЧЕНИЕ

Мне понравился процесс синтеза verilog-описания своего устройства, genus это очень удобный интструмент, позволяющий разрабатывать микросхемы любой сложности, обходясь достаточно простым и коротким кодом. В процессе работы меня еще больше привлекла моя специальность и та область, в которой в будущем мне придется работать.

У меня получилась очень простая схема и очень малое использование минимальной площади кристалла, размеры сземы соизмеримы с шириной одной контактной площадки.

Я надеюсь, что в будущем смогу поработать с более нагруженными схемами, помимо этого хотелось бы попробовать VHDL.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. **Черных, А.Г**. Технология изготовления КМОП транзисторов/А.Г Черных, Д.А. Котов -Минск: Изд-во «БГУИР», 2008.
2. **Вальд, А.** Последовательный анализ./ А.Вальд.– Москва: Физматгиз, 1960.
3. **Левин, Б.Р.** Теория надежности радиотехнических систем./ Б.Р Левин. **–** Москва**:** Изд-во «Советское радио», 1972
4. **Райкин, А.Л**. Элементы теории надежности для проектирования технических систем./ А.Л. Райкин. -Москва: Изд-во «Советское радио»,1967
5. **Перроте, А.И.** Вопросы надежности радиоэлектронной аппаратуры./ А.И. Перроте, А.И. Сторчак – Москва: Изд-во «Советское радио», 1976.
6. **Авдеев, В.А**. Периферийные устройства: интерфейсы, схемотехника, программирование / В.А. Авдеев. - М.: ДМК, 2016. - 848 с
7. **Аверченков, О.Е**. Схемотехника: аппаратура и программы / О.Е. Аверченков. - М.: ДМК, 2014. – 588
8. **Амосов, В.В.** Схемотехника и средства проектирования цифровых устройств / В.В. Амосов. - СПб.: БХВ-Петербург, 2012. - 560 с
9. **Ашихмин, А.С.** Цифровая схемотехника. Шаг за шагом / А.С. Ашихмин. - М.: Диалог-МИФИ, 2008. - 304 с.
10. **Блюм, Х.** Схемотехника и применение мощных импульсных устройств / Х. Блюм. - М.: Додэка, 2008. - 352 с.
11. **Бойко, В.И.** Схемотехника электронных систем. Аналоговые и импульсные устройства / В.И. Бойко, А.Н. Гуржий, В.Я. Жуйков [и др.]. - СПб.: БХВ-Петербург, 2004. - 496 с.