ИТМО

Функциональная схемотехника

Отчет по лабораторной работе №1

Группа P3332
Вариант 7

Выполнили:

Студенты группы P3332:

Батаргин Егор Александрович
Терновский Илья Евгеньевич

г. Санкт-Петербург

2024 г.

Оглавление

Цель работы

1. Получить базовые знания о принципах построения цифровых интегральных схем с использованием технологии КМОП.
2. Познакомится с технологией SPICE-моделирования схем на транзисторах.
3. Получить навыки описания схем базовых операционных элементов (БОЭ) комбинационного типа на вентильном уровне с использованием языка описания аппаратуры Verilog HDL.

Задание

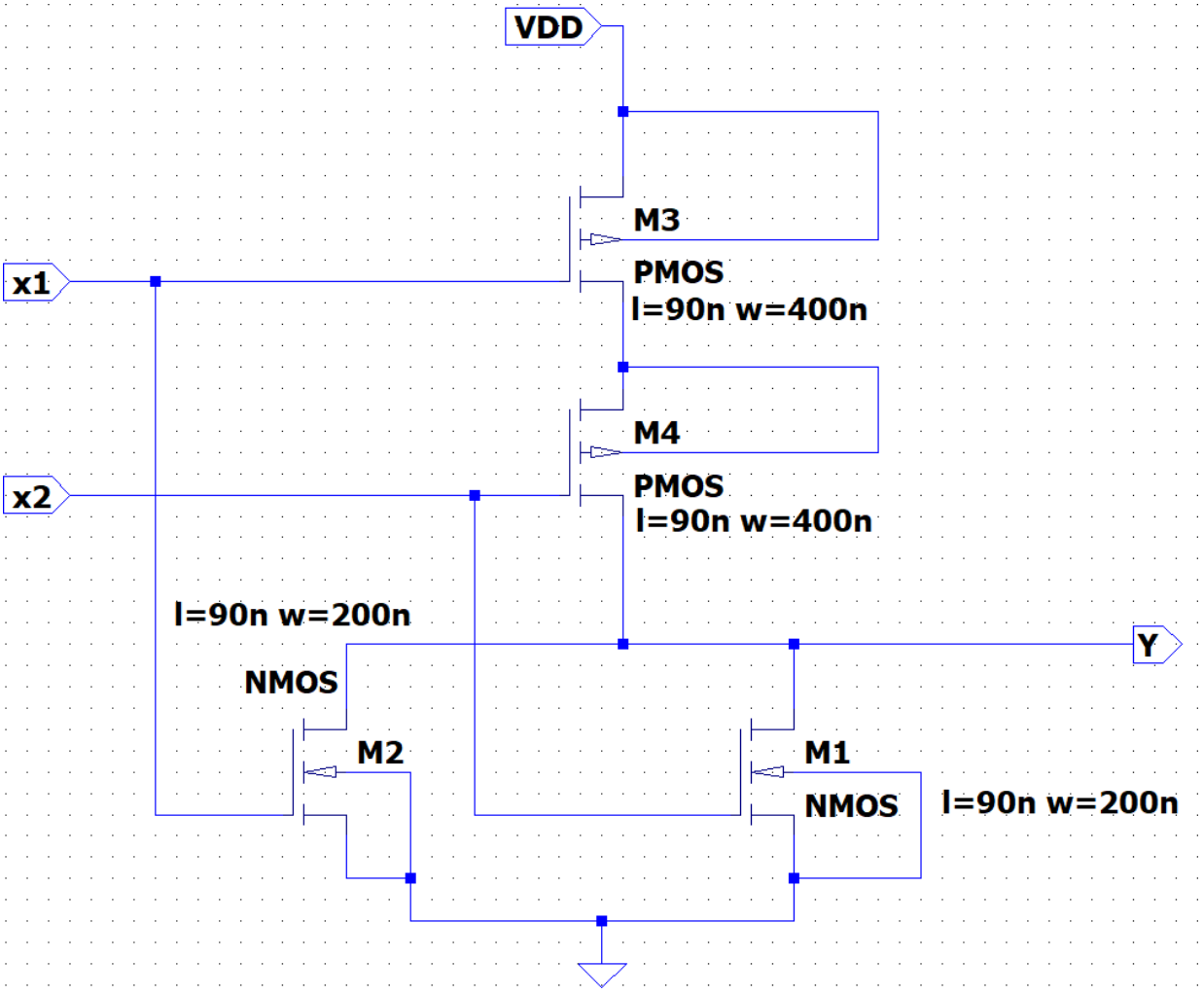
Вариант 7. Логический базис – NOR; БОЭ – Четырехразрядный двоичный сумматор с переносом (полный).

Часть 1. LTSpice.

Разработка вентиля.

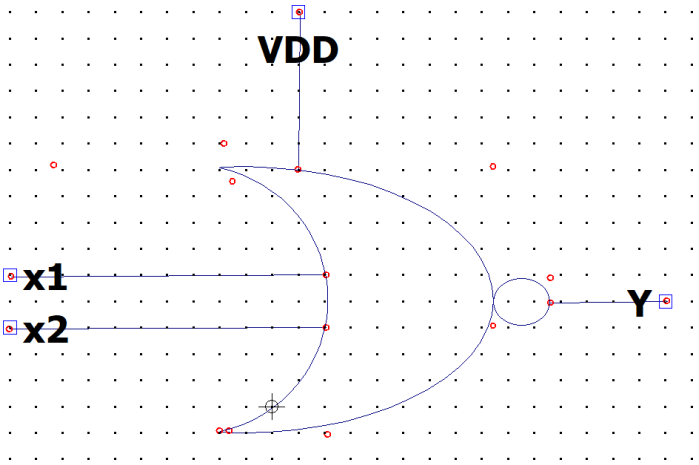
Схема разработанного вентиля:

X1, X2 – входы, Y – выход, VDD – напряжение питания; использовано по 2 транзистора PMOS и NMOS.



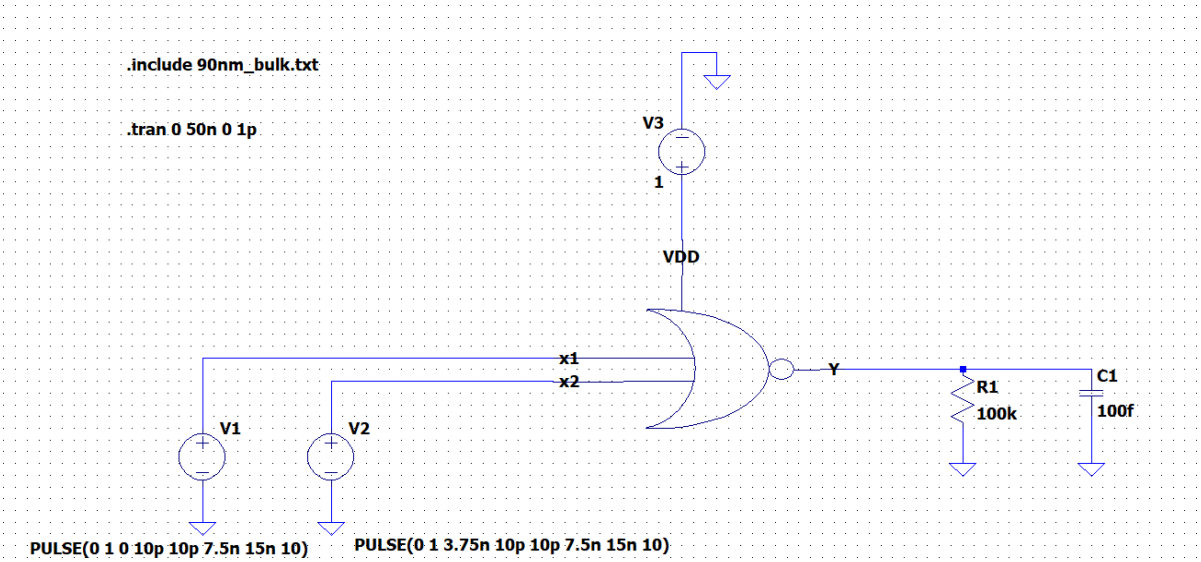
Изображение 1 схема вентиля

Символ вентиля:



Изображение 2 Символ вентиля

Схема тестирования:



Изображение 3 схема тестирования

Моделирование работы схемы и определение задержки вентиля

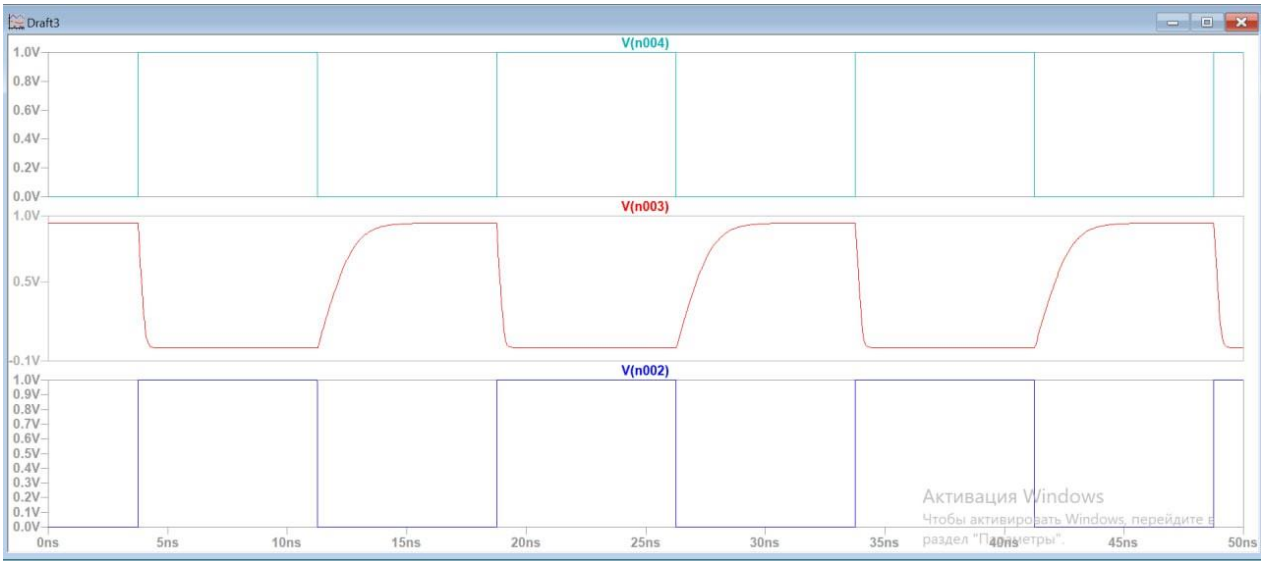
Моделирование при одинаковой задержке генераторов импульса. На первом графике генерируются пары аргументов (1, 1) и (0, 0), с ожидаемым значением вентиля 0 и 1 соответственно.

V(n004) - вход 1

V(n003) - выход

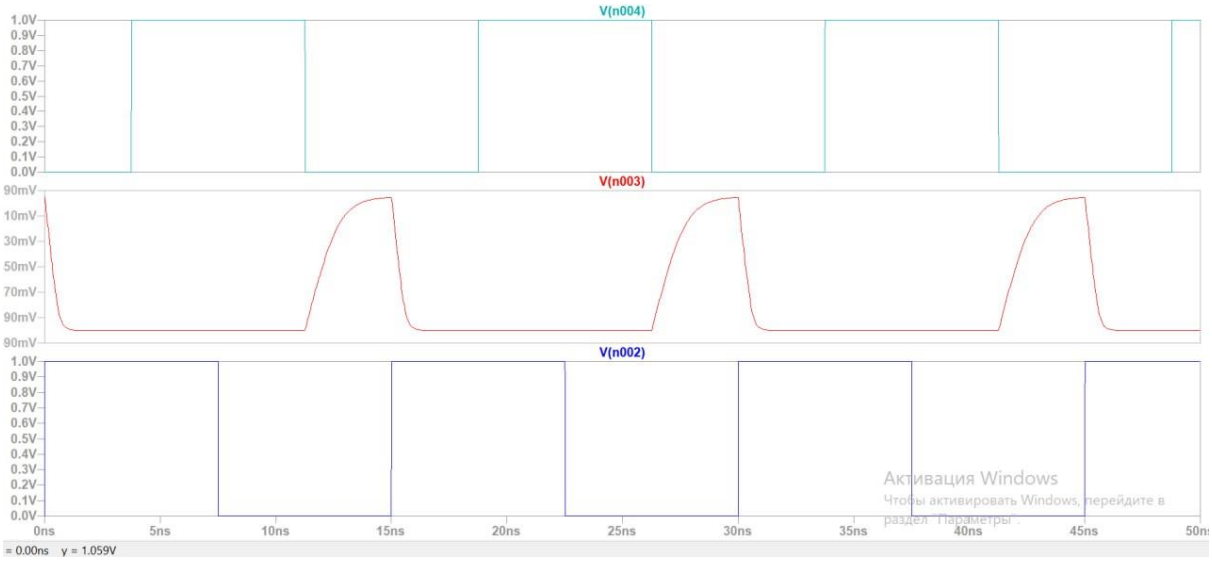
V(n002) - вход 2

1)



Изображение 4 временная диаграмма NOR при одинаковой задержке генератора импульса

2)



Изображение 5 временная диаграмма NOR

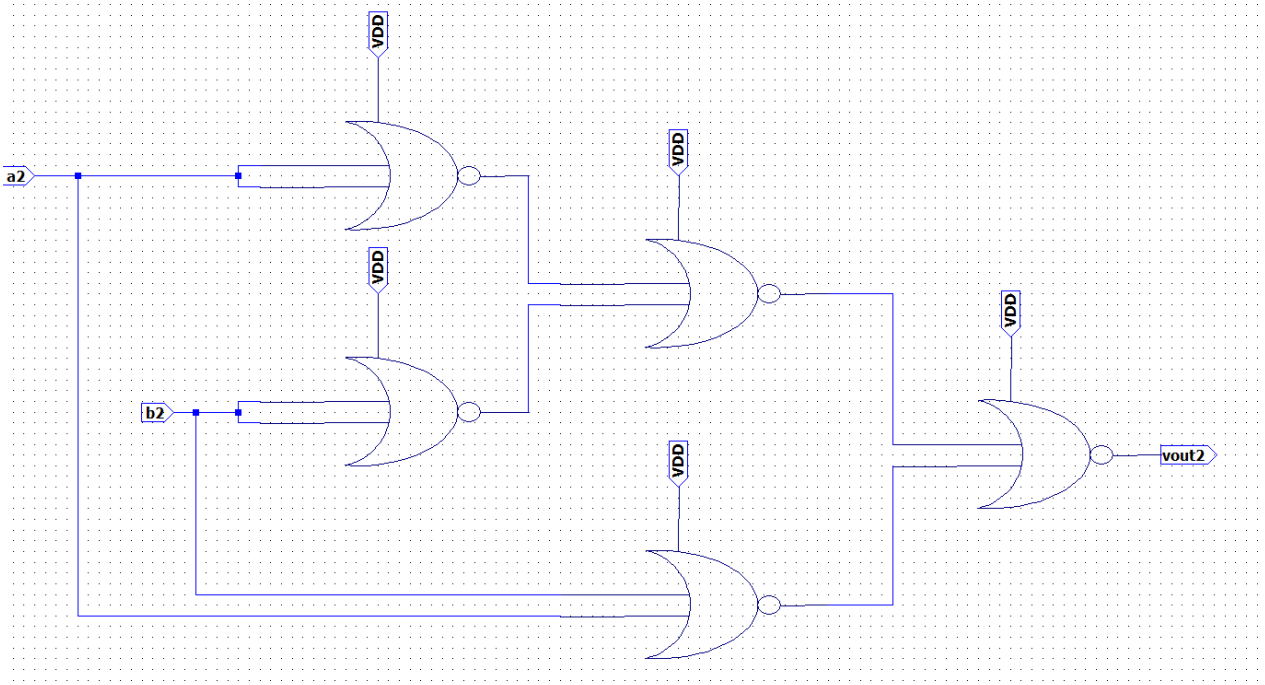
Время задержки подъема = 12.2–11.3 = 0.9нс

Время задержки спада = 15.3–15.0 = 0.3нс

Максимальная частота изменения сигнала: $\boldsymbol{\nu Max}$ = 1/ (0.9 + 0.3) = 833МГц

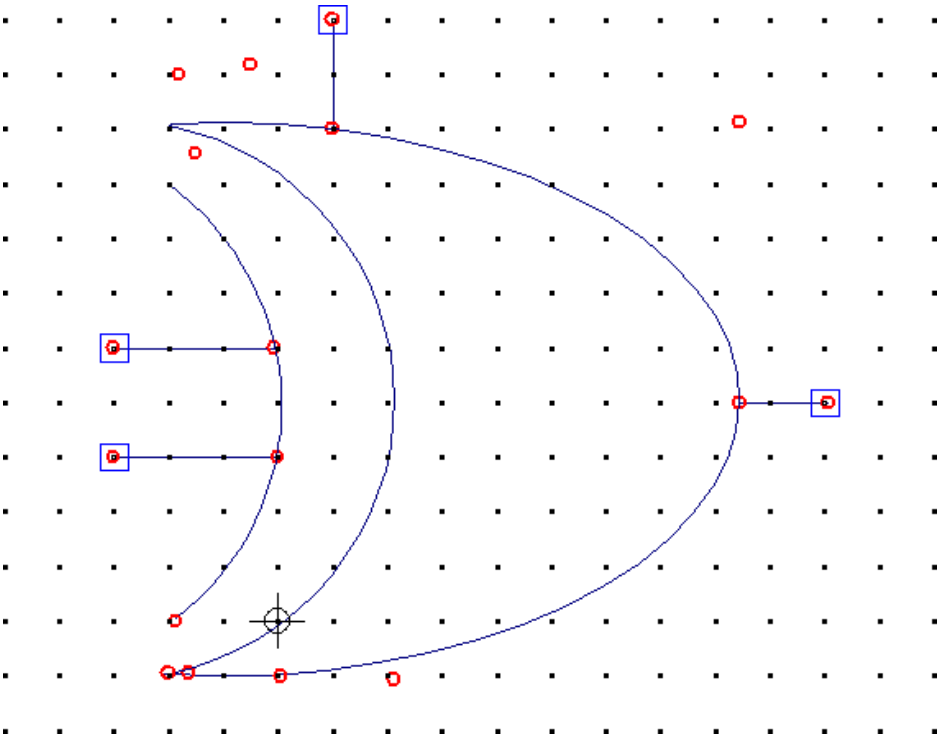
Разработка БОЭ.

Были реализованы элементы XOR, OR, AND на основе NOR. На основе этих элементов был построен сумматор.



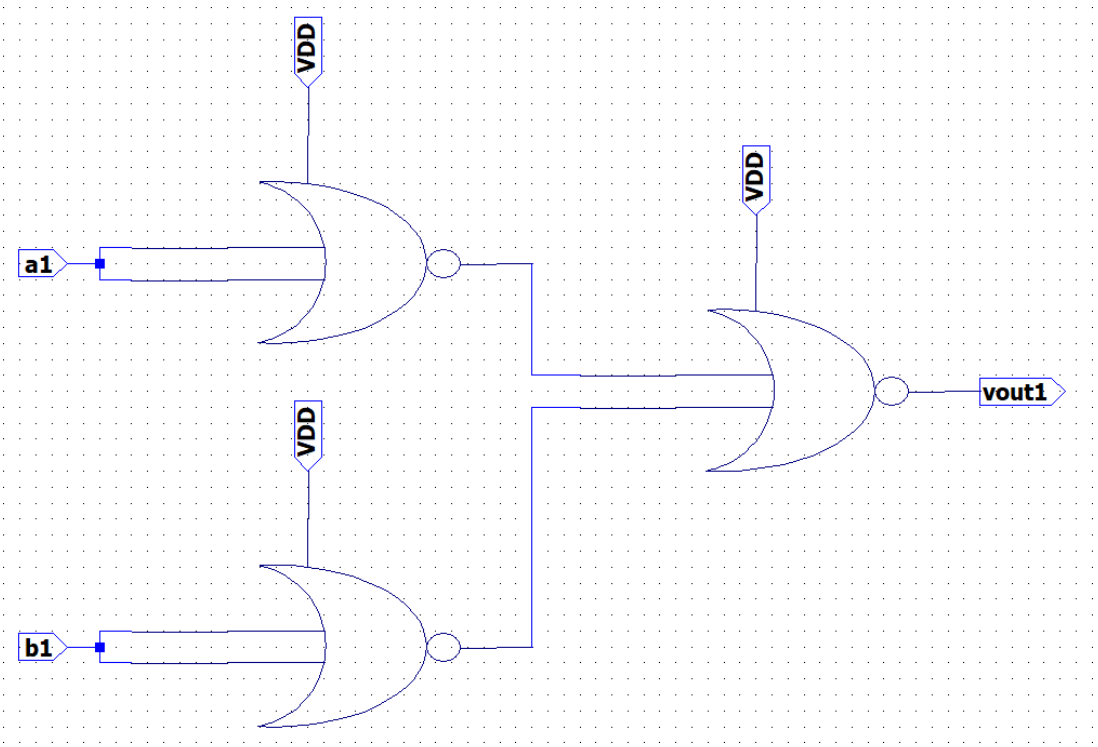
Изображение 6 схема XOR

Символ XOR:



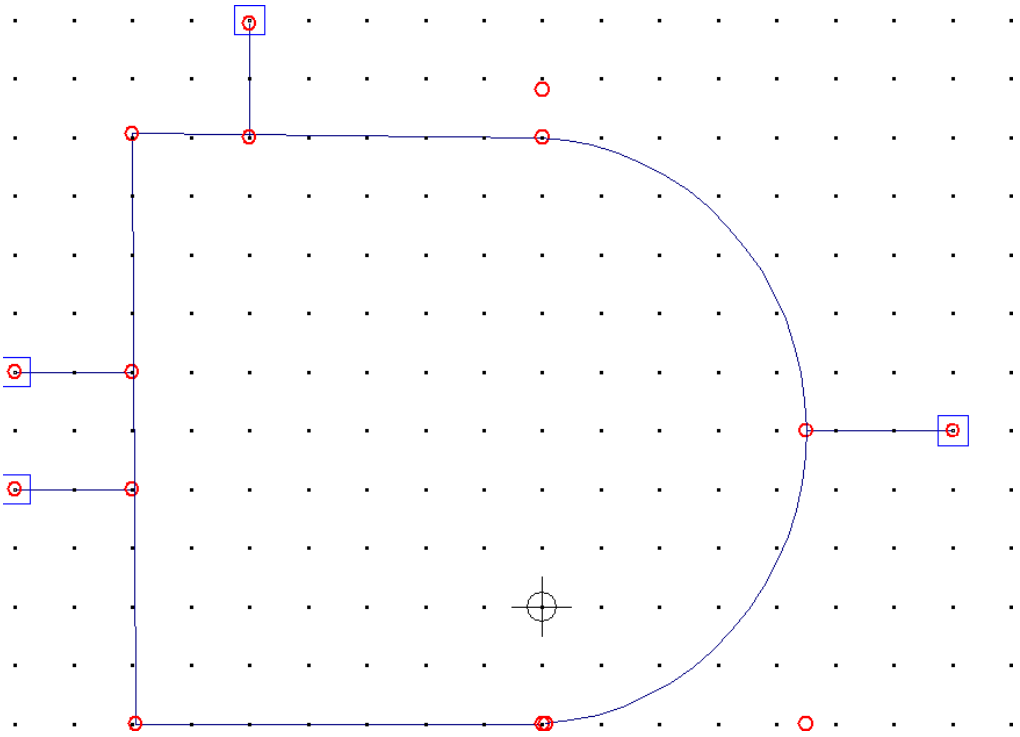
Изображение 7 символ XOR

Схема AND:



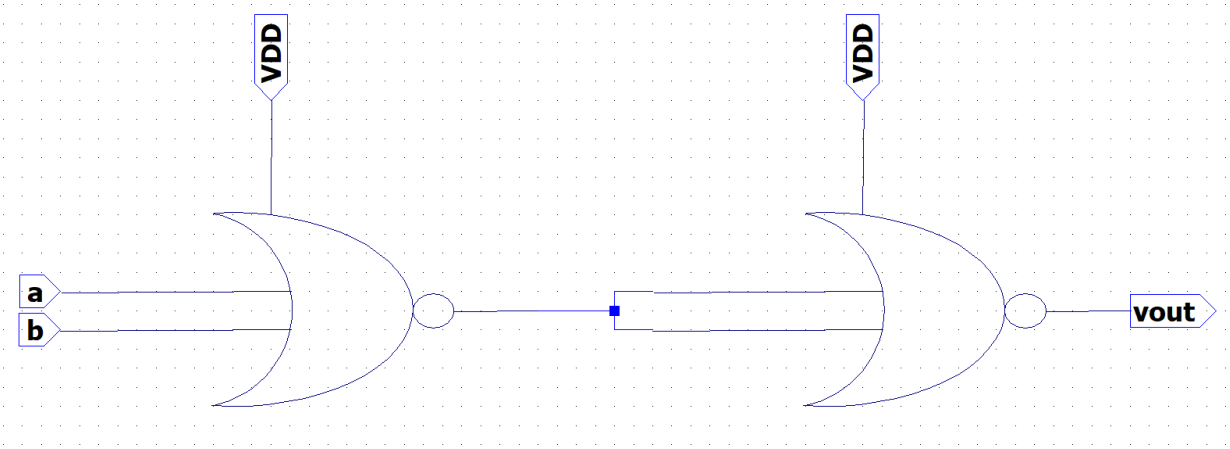
Изображение 8 схема AND

Символ AND:



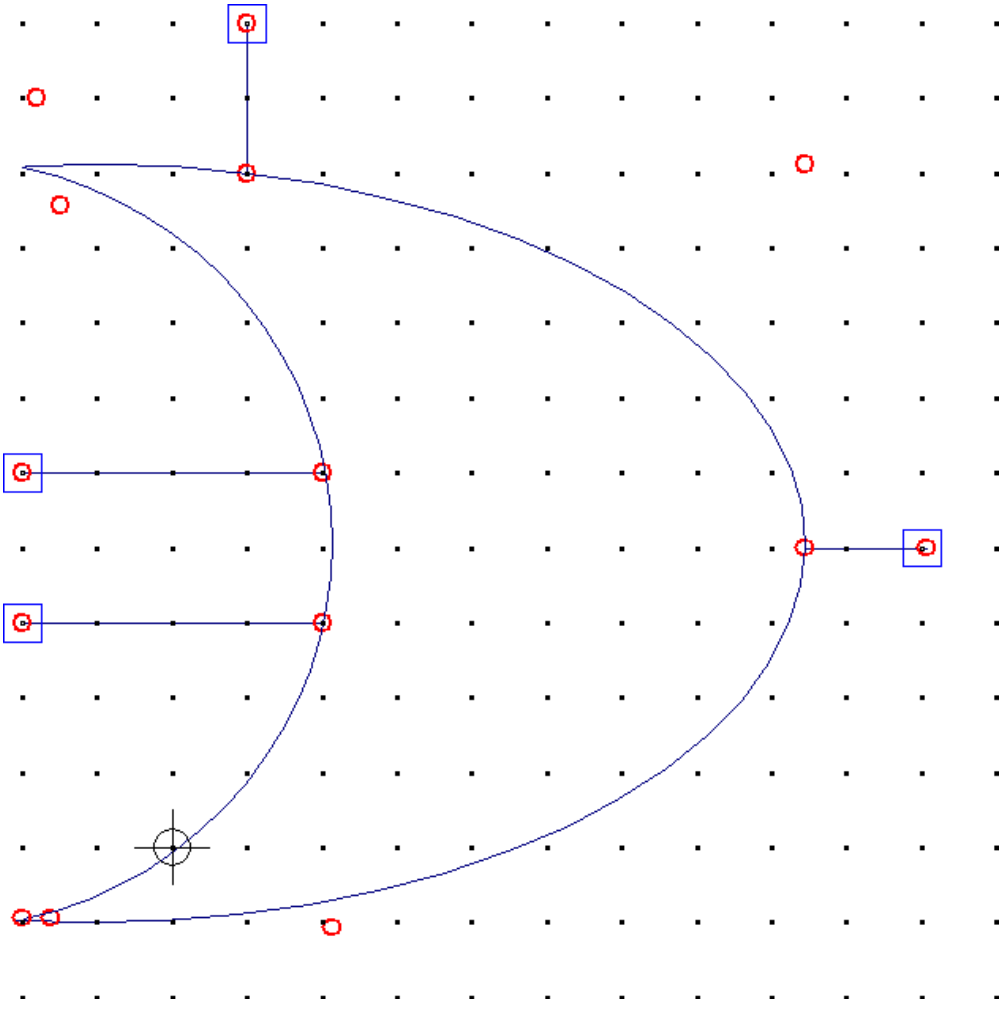
Изображение 9 символ AND

Схема OR:



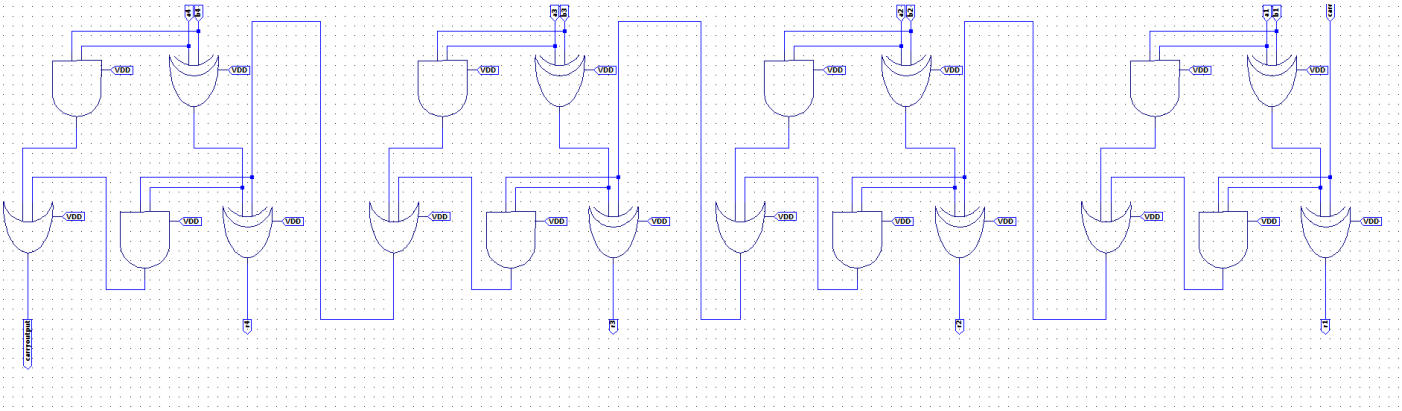
Изображение 10 схема OR

Символ OR:

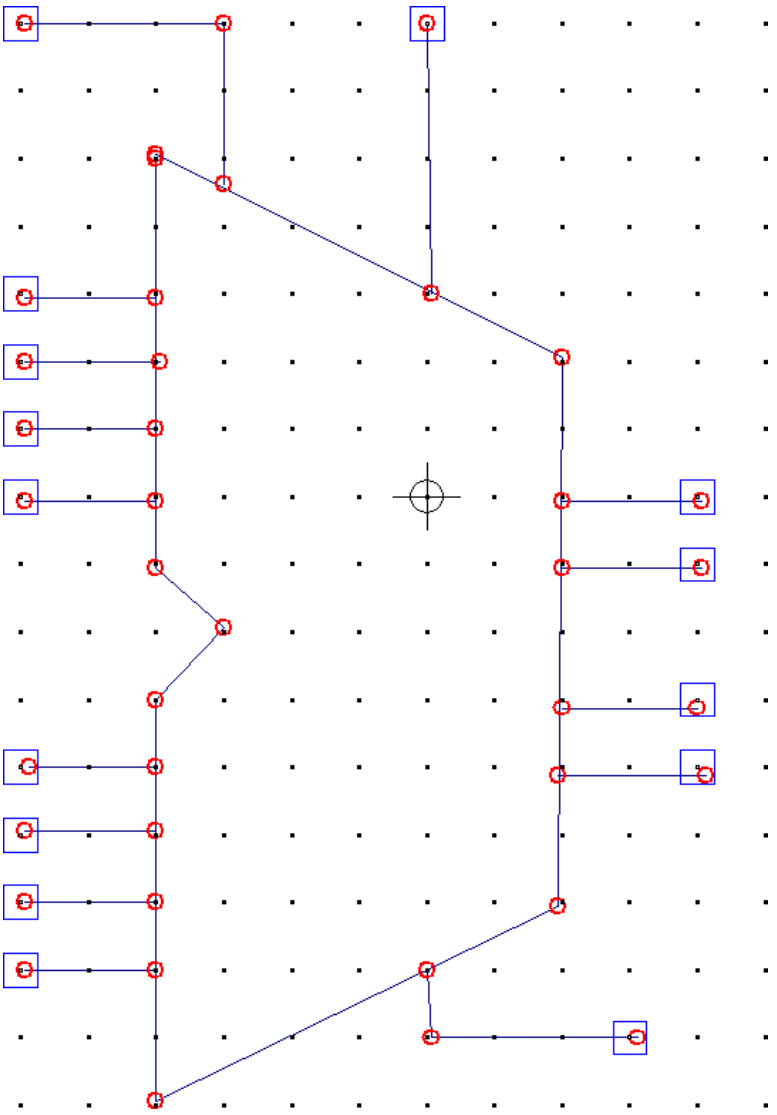


Изображение 11 Символ OR

Схема сумматора:

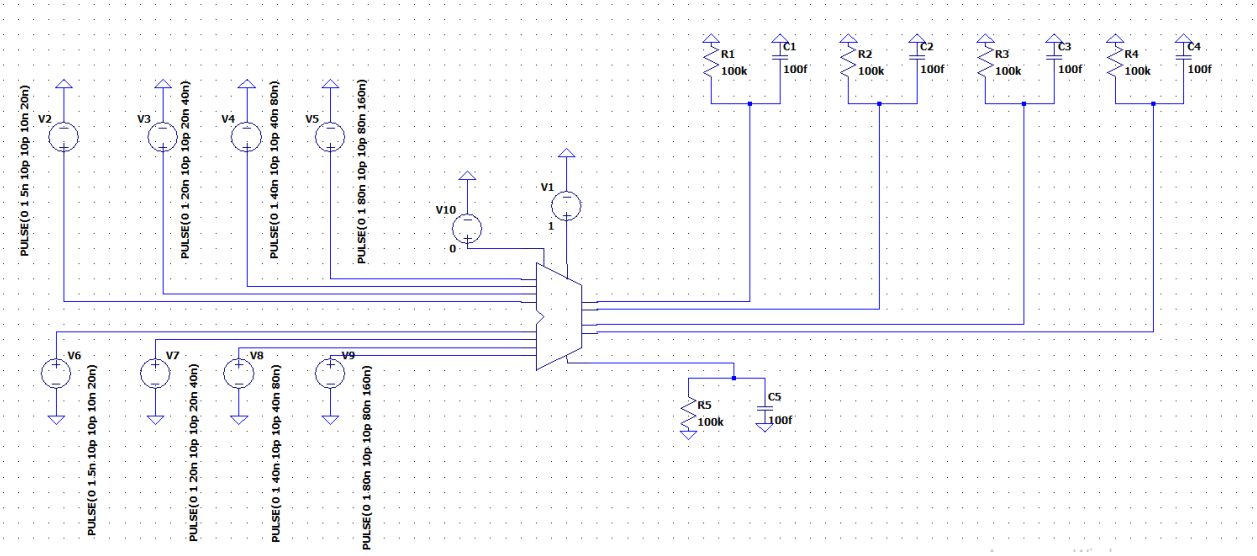


Изображение 12 схема четырехразрядного сумматора

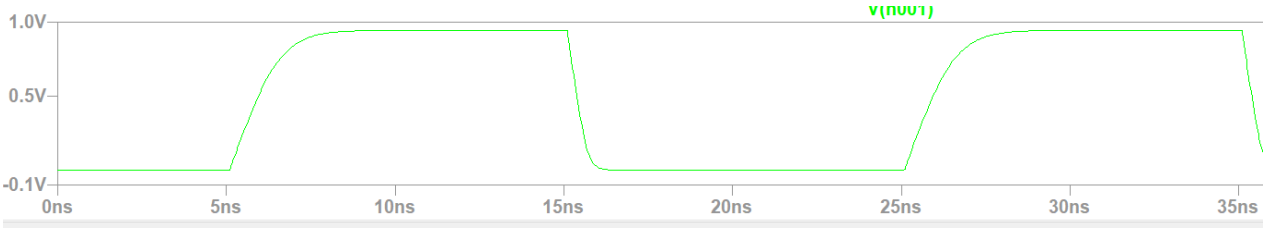


Изображение 13 символ четырехразрядного сумматора

Моделирование работы схемы и определение задержки.



Изображение 14 схема тестирования БОЭ



Изображение 15 временная диаграмма процесса тестирования БОЭ (для одного из выходов)

Время задержки подъема = 6,0 – 5,0 = 1нс

Время задержки спада = 15,3 –15.0 = 0.3нс

Максимальная частота изменения сигнала = 1+0.3 = 1,3нс

Максимальная частота изменения сигнала вентиля:

$\boldsymbol{vMax}$ = 1/ (1,3) = 769МГц

Часть 2 (Verilog).

Для упрощения работы 4-х битный сумматор был разделен на отдельные полные сумматоры.

```
1  module adder(
2      input a,
3      input b,
4      input c_in,
5      output c_out,
6      output sum
7  );
8
9      wire out_1, out_2, out_3, out_4, out_5, out_6, out_7;
10
11     nor(out_1, a, b);
12     nor(out_2, out_1, a);
13     nor(out_3, out_1, b);
14     nor(out_4, out_2, out_3);
15     nor(out_5, out_4, c_in);
16     nor(out_6, out_5, out_4);
17     nor(out_7, out_5, c_in);
18
19     nor(c_out, out_1, out_5);
20     nor(sum, out_6, out_7);
21
```

```
endmodule
```

Листинг 1 Код полного сумматора с использованием nor

```
module multi_adder(
    input a1, a2, a3, a4,
    input b1, b2, b3, b4,
    output sum1, sum2, sum3, sum4,
    output c_out
);

    wire c_out1, c_out2, c_out3;

    adder adder1 (.c_out(c_out1), .sum(sum1), .a(a1), .b(b1), .c_in(0));
    adder adder2 (.c_out(c_out2), .sum(sum2), .a(a2), .b(b2), .c_in(c_out1));
    adder adder3 (.c_out(c_out3), .sum(sum3), .a(a3), .b(b3), .c_in(c_out2));
    adder adder4 (.c_out(c_out), .sum(sum4), .a(a4), .b(b4), .c_in(c_out3));
endmodule
```

Листинг 2 Код разработанного модуля БОЭ с использованием полного сумматора

```

module multi_adder_tb;
    reg a1, a2, a3, a4;
    reg b1, b2, b3, b4;

    wire sum1, sum2, sum3, sum4;
    wire c_out;

    multi_adder uut (
        .a1(a1), .a2(a2), .a3(a3), .a4(a4),
        .b1(b1), .b2(b2), .b3(b3), .b4(b4),
        .sum1(sum1), .sum2(sum2), .sum3(sum3), .sum4(sum4),
        .c_out(c_out)
    );

    integer i, j;
    reg [3:0] a;   // 4-битная переменная для хранения значений a1-a4
    reg [3:0] b;   // 4-битная переменная для хранения значений b1-b4
    reg [4:0] expected_sum; // Ожидаемая сумма

    initial begin
        // Цикл для перебора всех возможных комбинаций значений 4-битных чисел a и
b
        for (i = 0; i < 16; i = i + 1) begin
            for (j = 0; j < 16; j = j + 1) begin
                a = i;
                b = j;

                // Присваиваем значения для каждого из битов a и b
                {a4, a3, a2, a1} = a;
                {b4, b3, b2, b1} = b;

                expected_sum = a + b;

                #10

                // Проверяем правильность результата
                if ({c_out, sum4, sum3, sum2, sum1} == expected_sum) begin
                    $display("Correct: a=%b, b=%b => sum=%b, c_out=%b", a, b,
{sum4, sum3, sum2, sum1}, c_out);
                end else begin
                    $display("Error: a=%b, b=%b => sum=%b (expected %b), c_out=%b
(expected %b)",
```

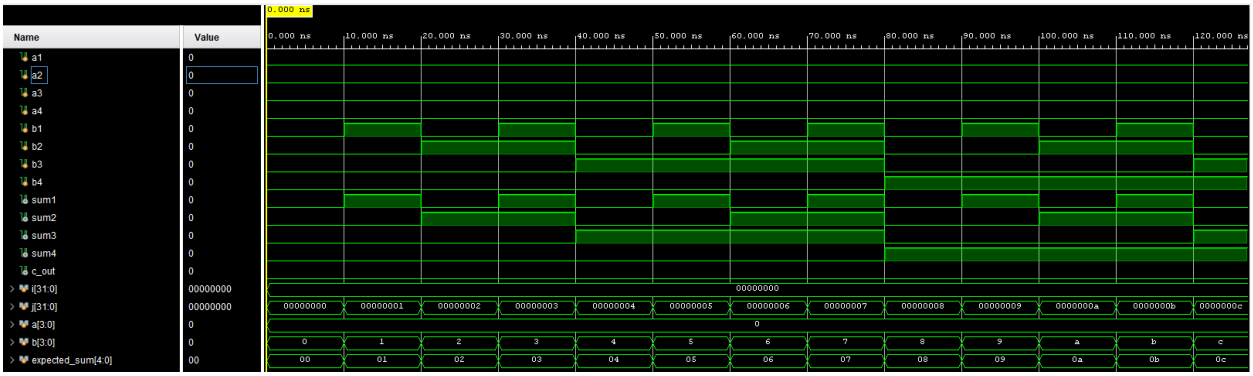
```
                                a, b, {sum4, sum3, sum2, sum1}, expected_sum[3:0],
c_out, expected_sum[4]);
                end
            end
        end
    end
endmodule
```

Листинг 3 Код разработанного тестового окружения БОЭ

Так же отдельно было разработано тестовое окружение для модуля полного сумматора

```
module adder_tb;
    reg a_in, b_in, c_in;
    wire sum, c_out;

    adder adder_1 (
        .a(a_in),
        .b(b_in),
        .c_in(c_in),
        .sum(sum),
        .c_out(c_out)
    );

    integer i;
    reg [2:0] test_val;
    reg expected_sum;
    reg expected_c_out;

    initial begin
        for (i = 0; i < 8; i = i + 1) begin
            test_val = i;
            a_in = test_val[0];
            b_in = test_val[1];
            c_in = test_val[2];

            // Вычисляем ожидаемые значения суммы и бита переноса
            expected_sum = (a_in ^ b_in) ^ c_in;
            expected_c_out = (a_in & b_in) | (c_in & (a_in ^ b_in));

            #10 // ждем

            // Проверяем соответствие реальных и ожидаемых значений
            if (sum == expected_sum && c_out == expected_c_out) begin
                $display("Correct: a_in=%b, b_in=%b, c_in=%b => sum=%b, c_out=%b",
a_in, b_in, c_in, sum, c_out);
            end else begin
                $display("Error: a_in=%b, b_in=%b, c_in=%b => sum=%b (expected %b),
c_out=%b (expected %b)",
                         a_in, b_in, c_in, sum, expected_sum, c_out,
expected_c_out);
            end
        end
    end
endmodule
```

Листинг 4 Код разработанного тестового окружения полного сумматора



Изображение 166 Временная диаграмма процесса тестирования БОЭ

```
Correct: a=0010, b=1001 => sum=1011, c_out=0
Correct: a=0010, b=1010 => sum=1100, c_out=0
Correct: a=0010, b=1011 => sum=1101, c_out=0
Correct: a=0010, b=1100 => sum=1110, c_out=0
Correct: a=0010, b=1101 => sum=1111, c_out=0
Correct: a=0010, b=1110 => sum=0000, c_out=1
Correct: a=0010, b=1111 => sum=0001, c_out=1
Correct: a=0011, b=0000 => sum=0011, c_out=0
Correct: a=0011, b=0001 => sum=0100, c_out=0
Correct: a=0011, b=0010 => sum=0101, c_out=0
Correct: a=0011, b=0011 => sum=0110, c_out=0
Correct: a=0011, b=0100 => sum=0111, c_out=0
Correct: a=0011, b=0101 => sum=1000, c_out=0
Correct: a=0011, b=0110 => sum=1001, c_out=0
Correct: a=0011, b=0111 => sum=1010, c_out=0
Correct: a=0011, b=1000 => sum=1011, c_out=0
Correct: a=0011, b=1001 => sum=1100, c_out=0
Correct: a=0011, b=1010 => sum=1101, c_out=0
Correct: a=0011, b=1011 => sum=1110, c_out=0
Correct: a=0011, b=1100 => sum=1111, c_out=0
Correct: a=0011, b=1101 => sum=0000, c_out=1
Correct: a=0011, b=1110 => sum=0001, c_out=1
```

Изображение 17 Пример отчета тестирования БОЭ

Вывод.

В процессе выполнения данной работы мы познакомились со средой моделирования LTspice и языком описания аппаратуры Verilog. В качестве опытного образца мы создал собственный вентиль NOR и на его основе создали и протестировали Четырехразрядный двоичный сумматор с переносом.

Так же узнали, что такое задержка реакции и задержка распространения, а именно то, что задержка распространения — это задержка, с которой сигнал пройдет через всю схему и на выходе мы получим правильный результат, в то время как задержка реакции — это задержка, с которой выход элемента начнет изменятся после изменения входа этого элемента. Проще говоря, задержка распространения — это максимальная задержка, в то время как задержка реакции минимальная. Измеряются во временных единицах, нс, мс, с и т.д.

Так же проанализировав работу мы пришли к выводу, что задержка подъёма не очень сильно увеличилась в связи с тем, что несмотря на увеличение количества элементов, так как мы смотрим на задержку реакции, а она зависит по большей части от самого вентиля задержка увеличиваться будет не линейно увеличению количества этих вентилей, так как изменения происходят, если можно так выразиться, во всей схеме почти одновременно.

К тому же мы углубились в то, что такое дешифратор - дешифратор является комбинационным БОЭ,

на вход которого приходит набор сигналов, а на выходе получается уже другой набор сигналов при этом количество таких сигналов либо равно изначальному, либо больше. Так же стоит заметить, что часто используется еще один сигнал управления! Часто дешифратор\шифратор используют для перехода из двоичной СС в десятичную, дабы затем использовать этот сигнал для обращения к определенным устройствам, например при помощи 3 управляющих сигналов мы можем обратиться к 8 разным устройствам (дешифратор 3в8). К тому же их удобно использовать для контроля дисплеев и различных схожих устройств. Обычно строятся в базисе И с использованием НЕ, но могут иметь и уникальную логику с использованием других вентилей.