

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** 3](#_Toc182350060)

[**Задание** 3](#_Toc182350061)

[**Часть 1. LTSpice.** 3](#_Toc182350062)

[Разработка вентиля. 3](#_Toc182350063)

[Моделирование работы схемы и определение задержки вентиля 4](#_Toc182350064)

[Разработка БОЭ. 5](#_Toc182350065)

[Моделирование работы схемы и определение задержки. 9](#_Toc182350066)

[**Часть 2 (Verilog).** 10](#_Toc182350067)

# **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**

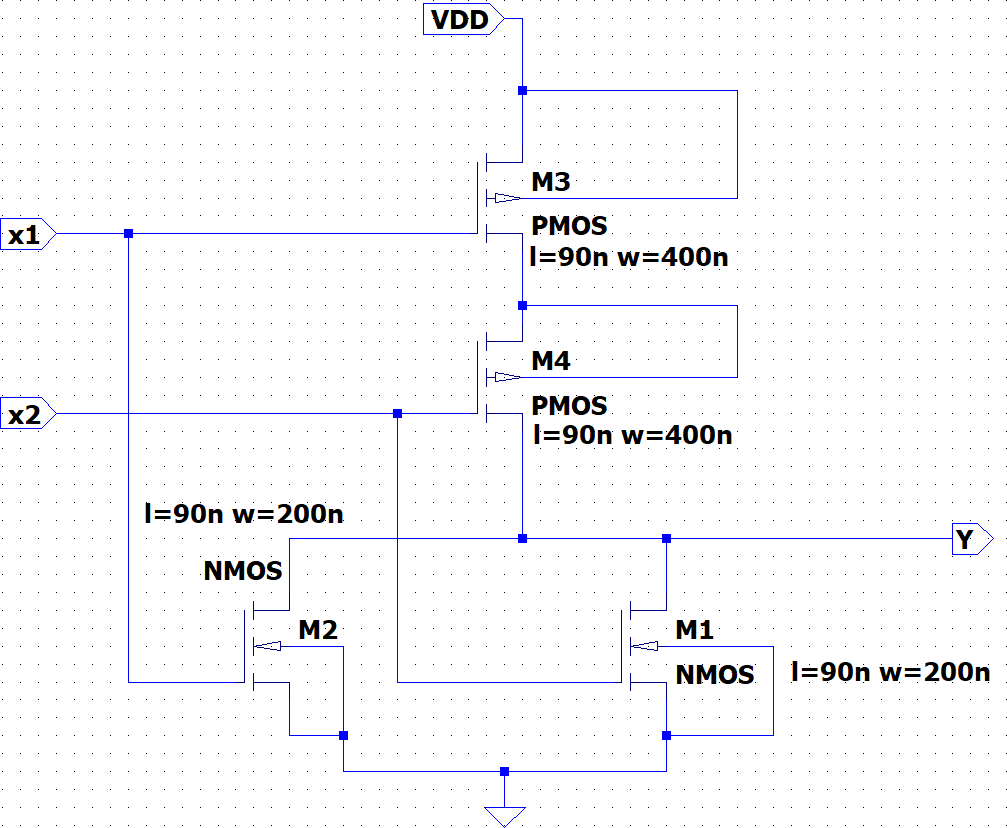
Вариант 7. Логический базис – NOR; БОЭ – Четырехразрядный двоичный сумматор с переносом (полный).

# **Часть 1. LTSpice.**

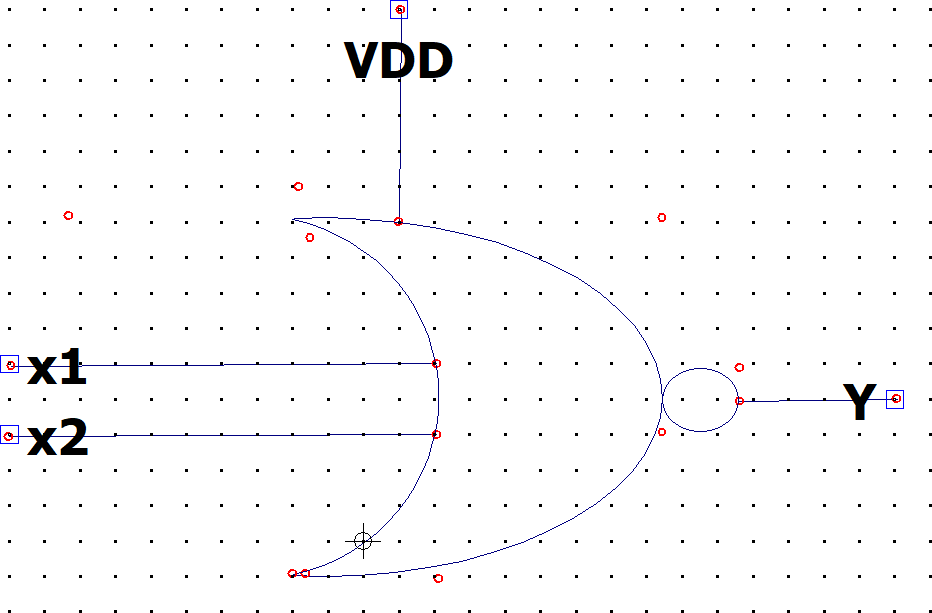
## Разработка вентиля.

Схема разработанного вентиля:

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

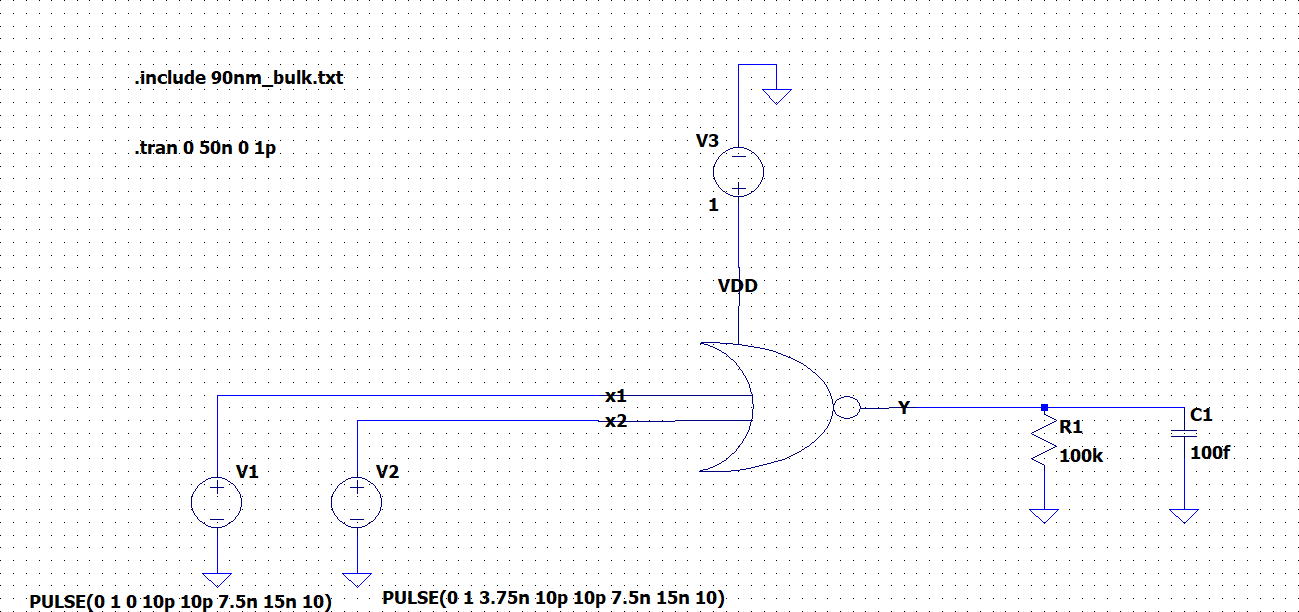


Изображение схема вентиля

**Символ вентиля:**

Изображение Символ вентиля

**Схема тестирования:**



Изображение 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

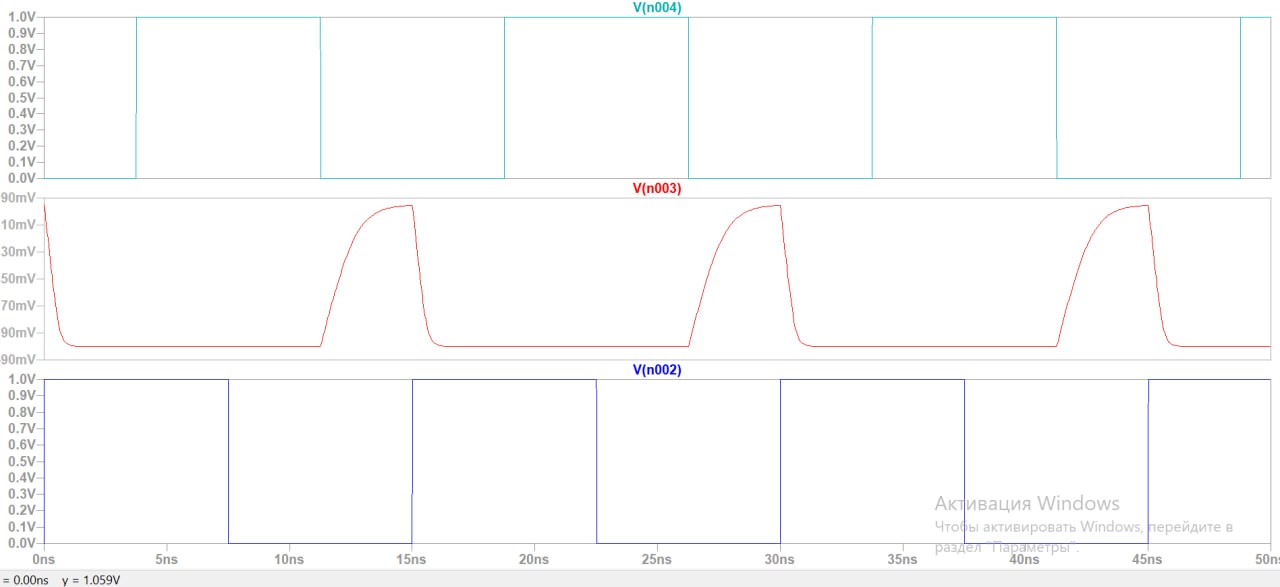
V(n002) - вход 2

**1)**



Изображение временная диаграмма NAND при одинаковой задержке генератора импульса

**2)**



Изображение временная диаграмма NAND

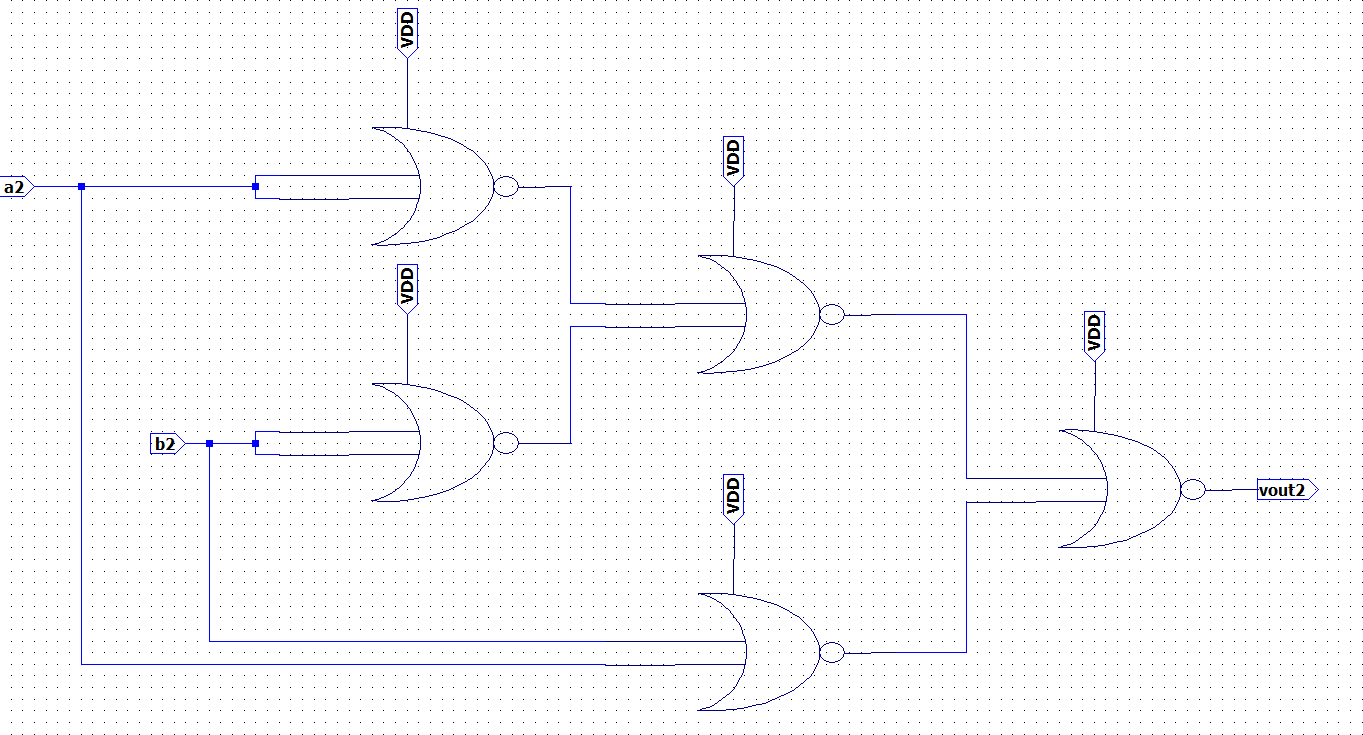
Время задержки подъема = 12.2–11.3 = 0.9нс

Время задержки спада = 15.3–15.0 = 0.3нс

Максимальная частота изменения сигнала: 𝝂𝑴𝒂𝒙 = 1/ (0.9 + 0.3) = 833МГц

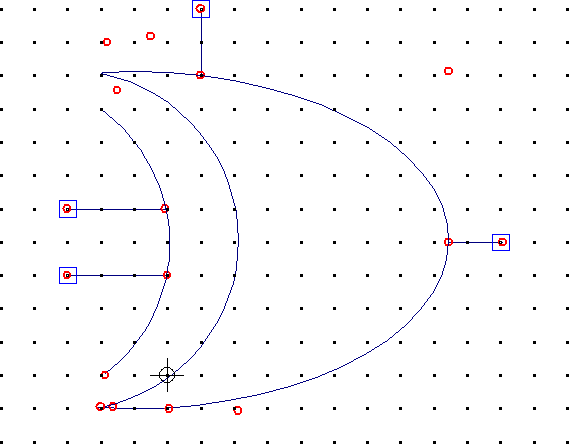
## Разработка БОЭ.

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



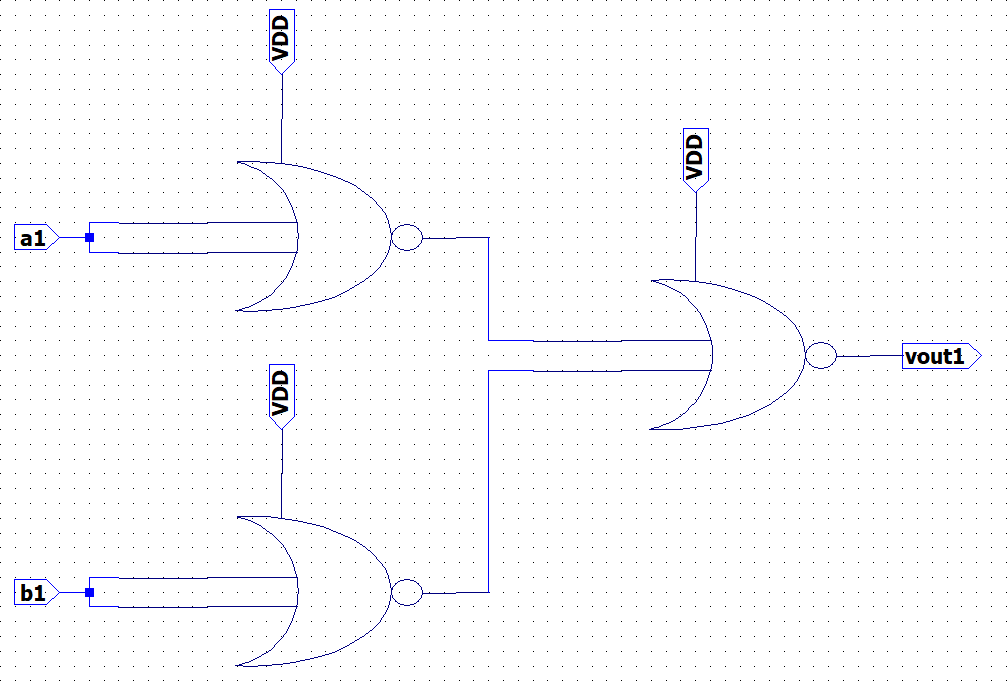
Изображение схема XOR

**Символ XOR:**



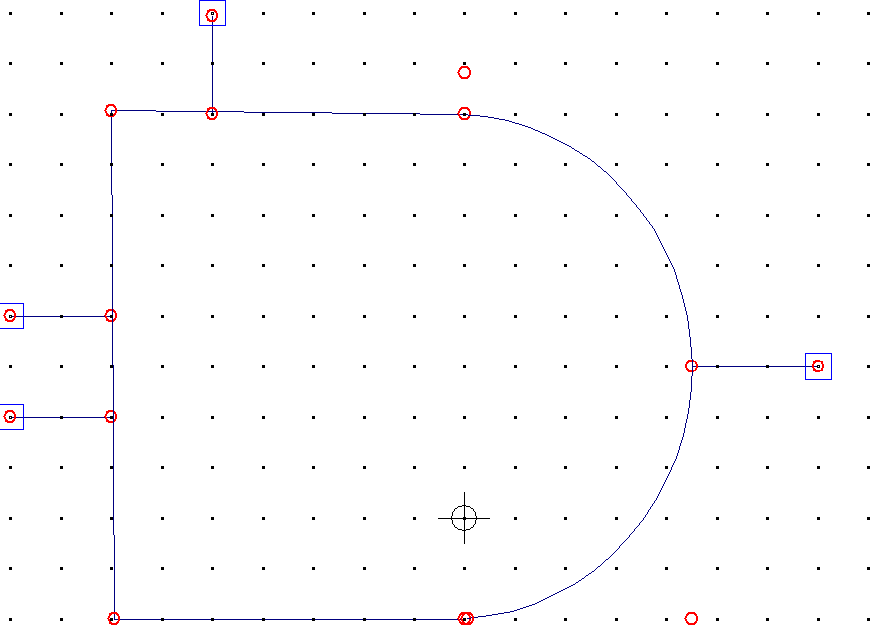
Изображение символ XOR

**Схема AND:**



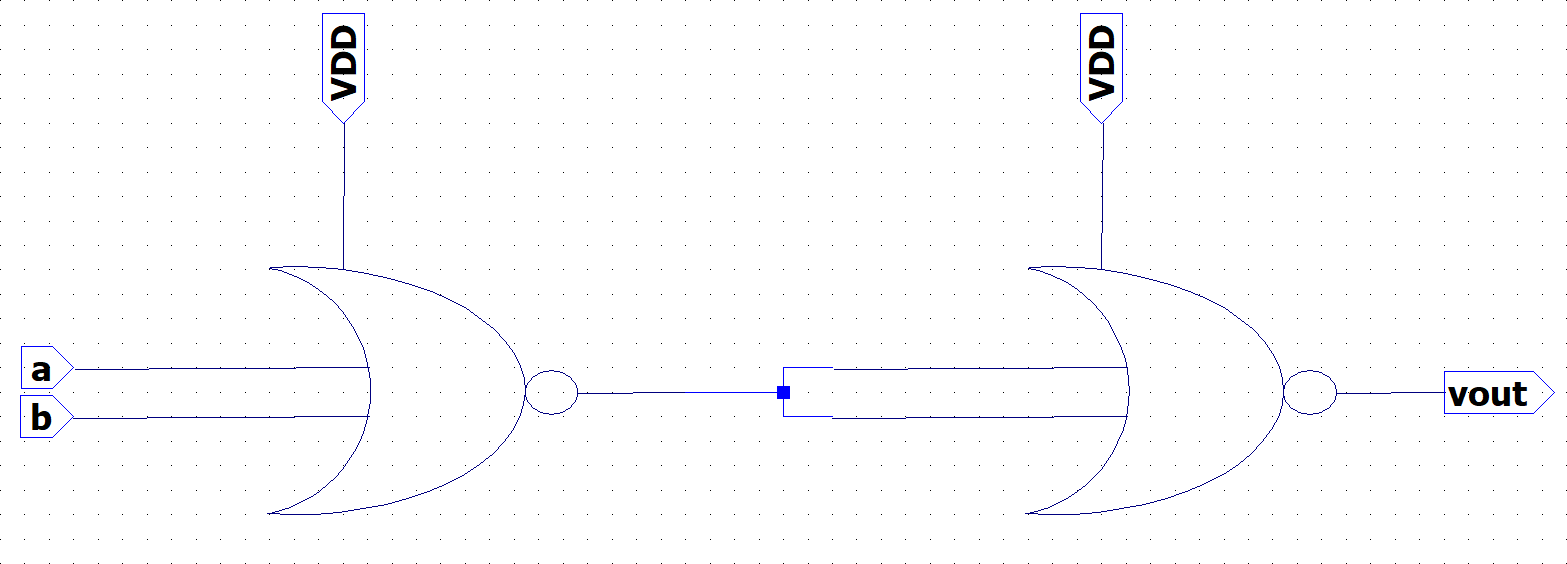
Изображение схема AND

**Символ AND:**



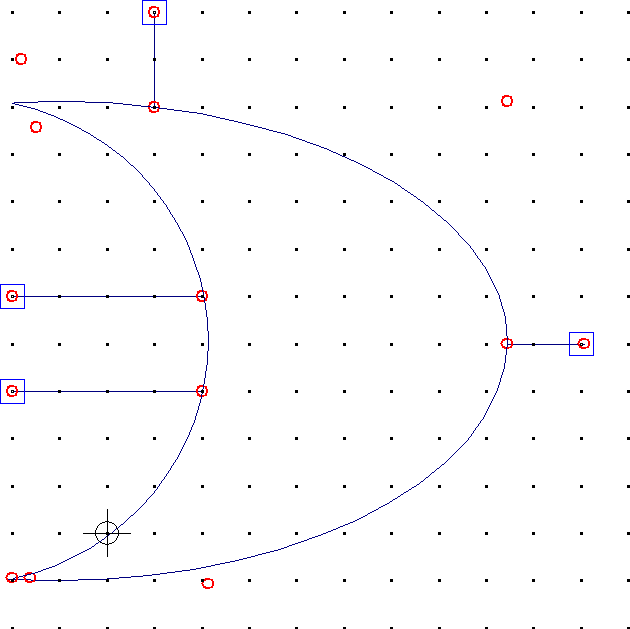
Изображение символ AND

**Схема OR:**



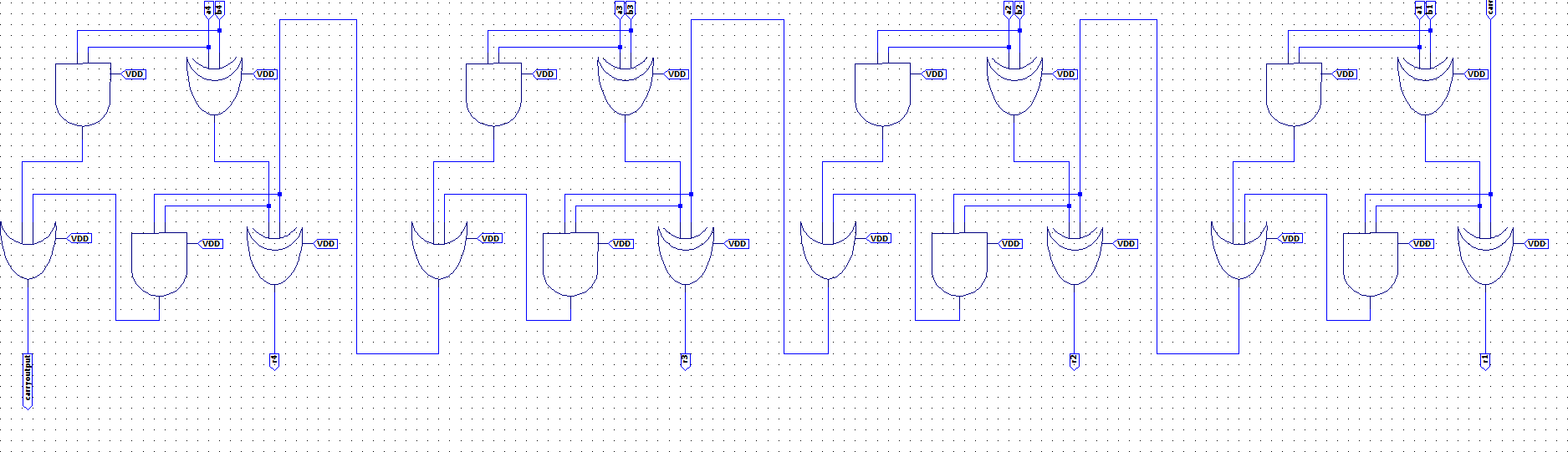
Изображение схема OR

**Символ OR:**

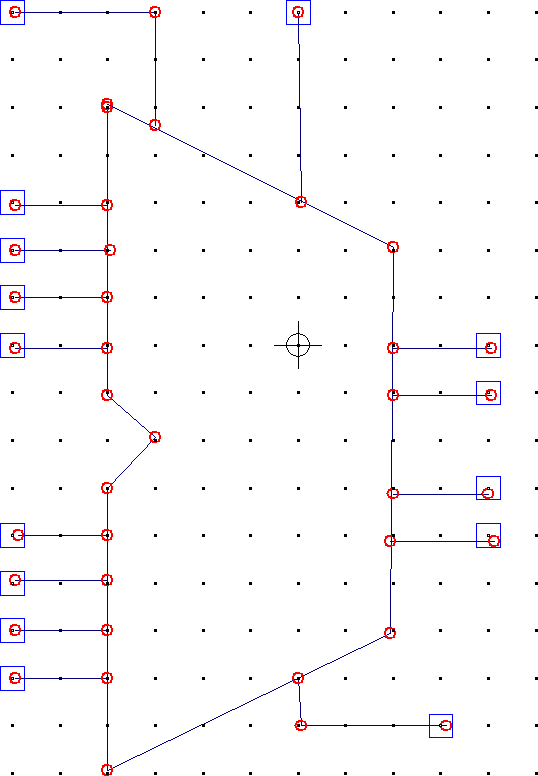


Изображение Символ OR

**Схема сумматора:**

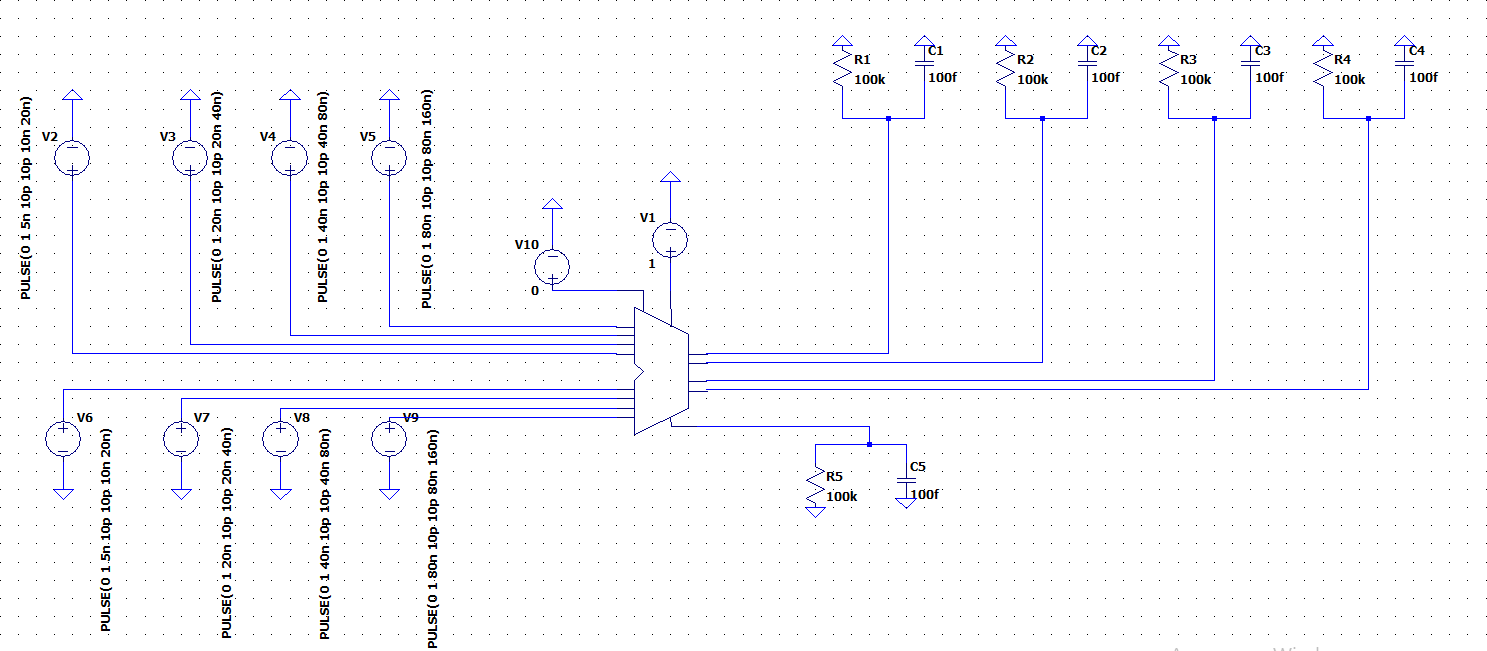


Изображение схема четырехразрядного сумматора

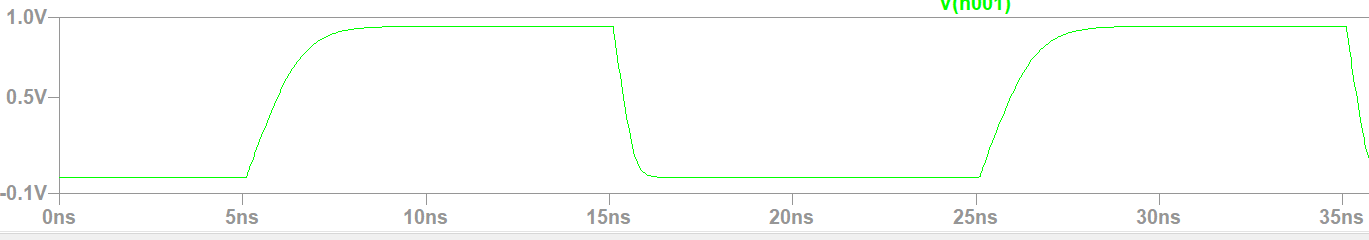


Изображение символ четырехразрядного сумматора

## Моделирование работы схемы и определение задержки.



Изображение схема тестирования БОЭ



Изображение 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:

𝝂𝑴𝒂𝒙 = 1/ (1,3) = 769МГц

# **Часть 2 (Verilog).**

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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | **module** adder(  **input** a,  **input** b,  **input** c\_in,  **output** c\_out,  **output** sum  );  **wire** out\_1, out\_2, out\_3, out\_4, out\_5, out\_6, out\_7;    **nor**(out\_1, a, b);  **nor**(out\_2, out\_1, a);  **nor**(out\_3, out\_1, b);  **nor**(out\_4, out\_2, out\_3);  **nor**(out\_5, out\_4, c\_in);  **nor**(out\_6, out\_5, out\_4);  **nor**(out\_7, out\_5, c\_in);    **nor**(c\_out, out\_1, out\_5);  **nor**(sum, out\_6, out\_7);  **endmodule** |

Листинг 1 Код полного сумматора с использованием nor

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | **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

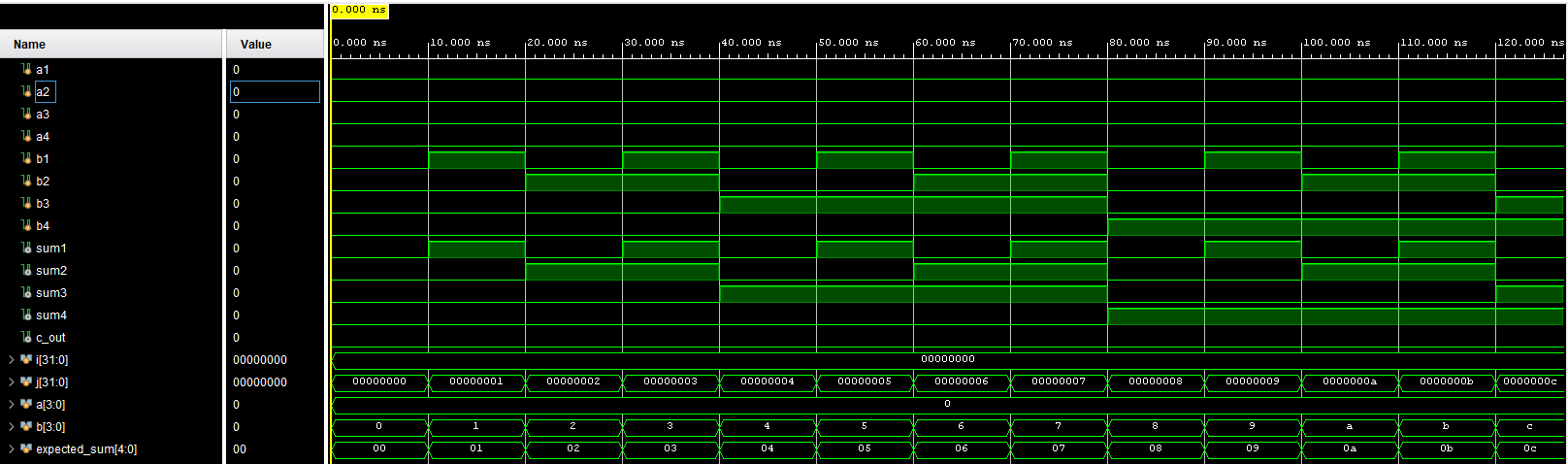
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | **module** multi\_adder\_tb;  **reg** a1, a2, a3, a4;  **reg** b1, b2, b3, b4;    **wire** sum1, sum2, sum3, 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

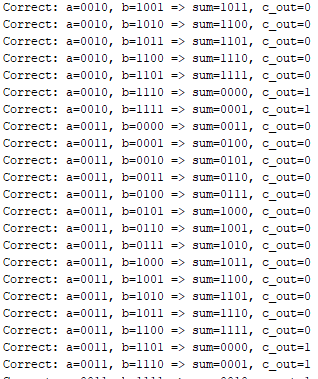
Так же отдельно было разработано тестовое окружение для модуля полного сумматора

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | **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



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.