ИТМО

Лабораторная работа №1

по дисциплине: Функциональная схемотехника

Вариант: 4

Выполнил: Неграш Андрей, P33301

Преподаватель: Салонина Екатерина Александровна

Санкт-Петербург, 2023

Цели работы

1. Получить базовые знания о принципах построения цифровых интегральных схем с использованием технологии КМОП.
2. Познакомиться 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 в соответствие с вариантом

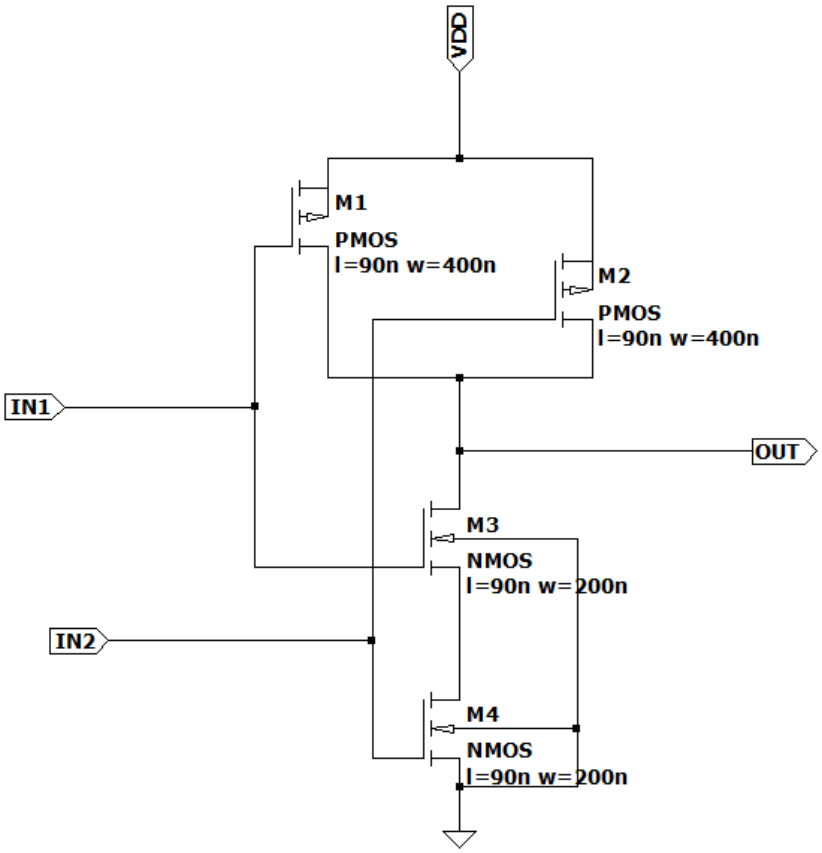
Лабораторная работа состоит из двух частей.

Первая часть посвящена проектированию цифровых вентилей на полевых транзисторах, построению схем на базе вентилей и знакомству с технологией SPICE-моделирования. Первая часть работы выполняется в программном пакете LTspice. При построении схем вентилей необходимо использовать КМОП-транзисторы с параметрами из файла, предоставленного преподавателем.

Вторая часть посвящена знакомству с языком описания аппаратуры Verilog HDL, изучению особенностей его использования для описания схем на вентильном уровне и приобретению навыков тестирования таких схем. Вторая часть работы выполняется с использованием Vivado Simulator, входящего в пакет Vivado Design Suite.

Выполнение части 1

Согласно варианту, требуется создать схему вентиля в логическом базисе NAND. Схема вентиля находится на GitHub, скриншот схемы вентиля выглядит следующим образом:



Символ вентиля согласно стандарту ANSI выглядит следующим образом:

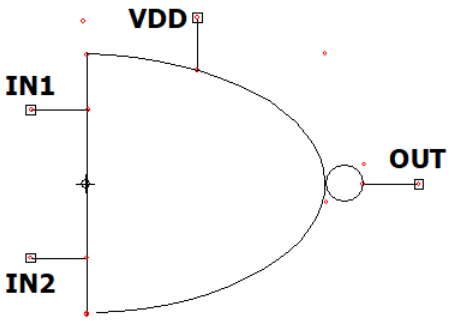
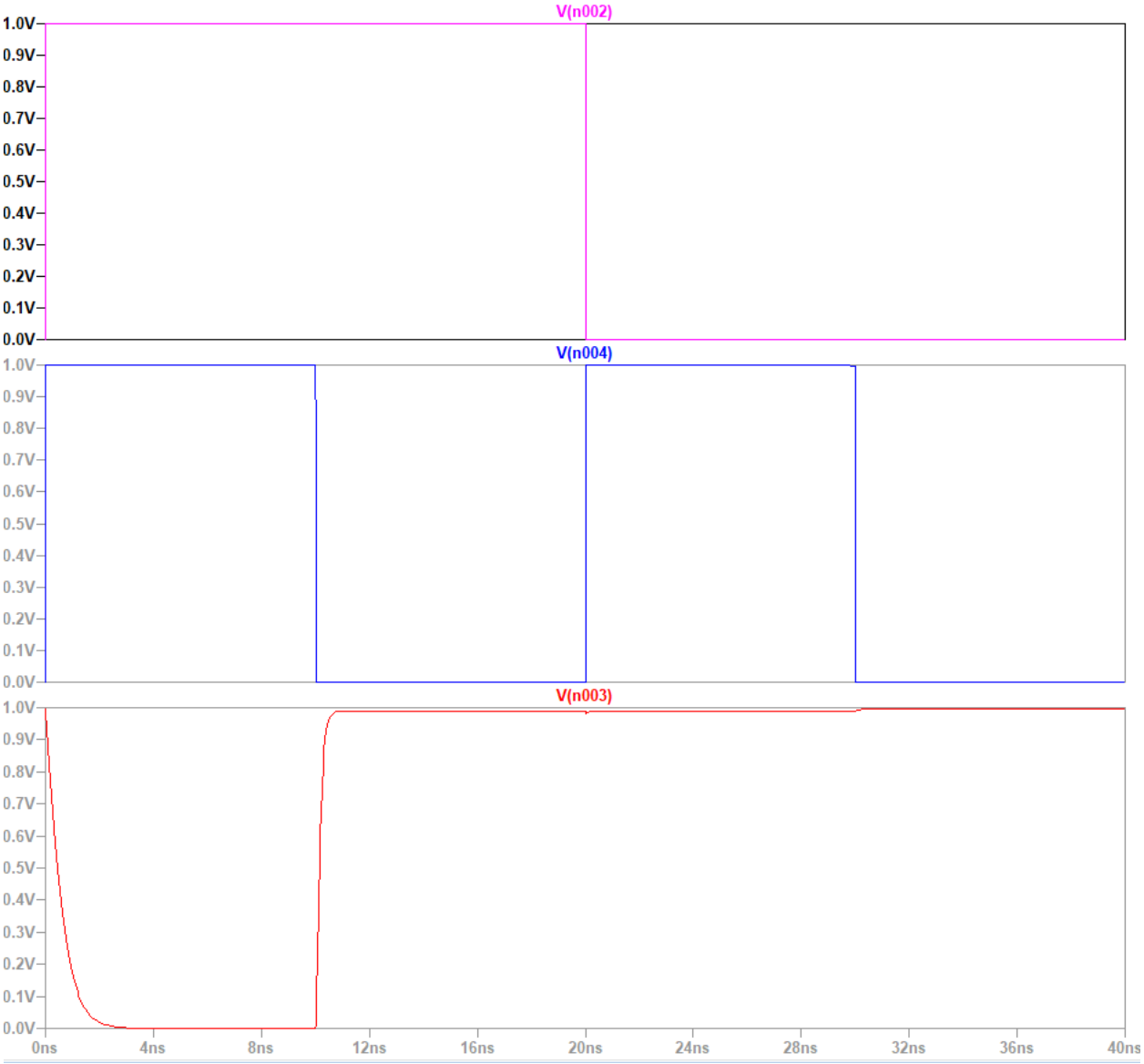


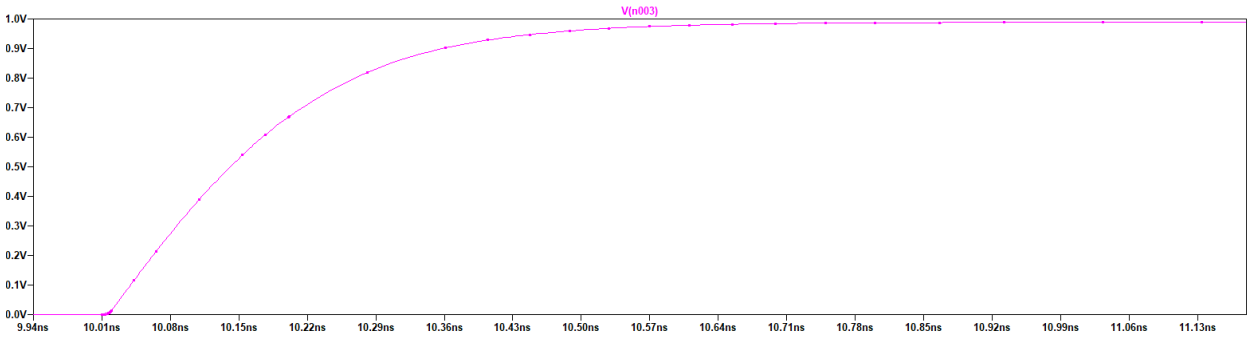
График тестирования выглядит следующим образом (специально заданы входные параметры пульсации с разной амплитудой):



Верхний график соответствует пульсации входного напряжения, подаваемого на IN1, средний – подаваемого на IN2. Стоит заметить, что частота пульсации во втором случае в 2 раза больше – это сделано для того, чтобы протестировать все возможные комбинации (в данном случае их 4). Нижний график соответствует выходному напряжению, и мы видим, что значение логического нуля достигается только при поданных на оба входа единицах, а во всех остальных случаях выходное напряжение соответствует значению логической единицы.

Этот тест демонстрирует корректность работы NAND относительно входных параметров.

Результат измерения задержки распространения сигнала через вентиль представлен на следующем скриншоте:



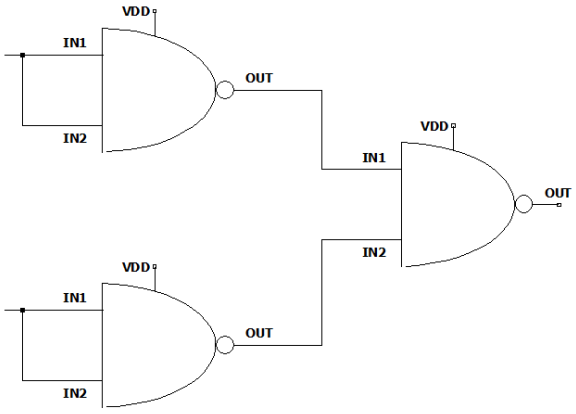
Согласно этим данным, задержка составляет около 880 пс

Перейдём к построению БОЭ согласно варианту (Позиционный шифратор «8 в 3»)

В классическом исполнении этот шифратор состоит из элементов, работающих как логическое OR (ИЛИ). Однако моя задача сделать его на основе созданного ранее NAND (И-НЕ). Для этого воспользуемся преобразованием:

$$\overline{\bar{A} * \bar{B}} = A + B$$

В среде LTspice данное преобразование реализовывается следующей схемой:

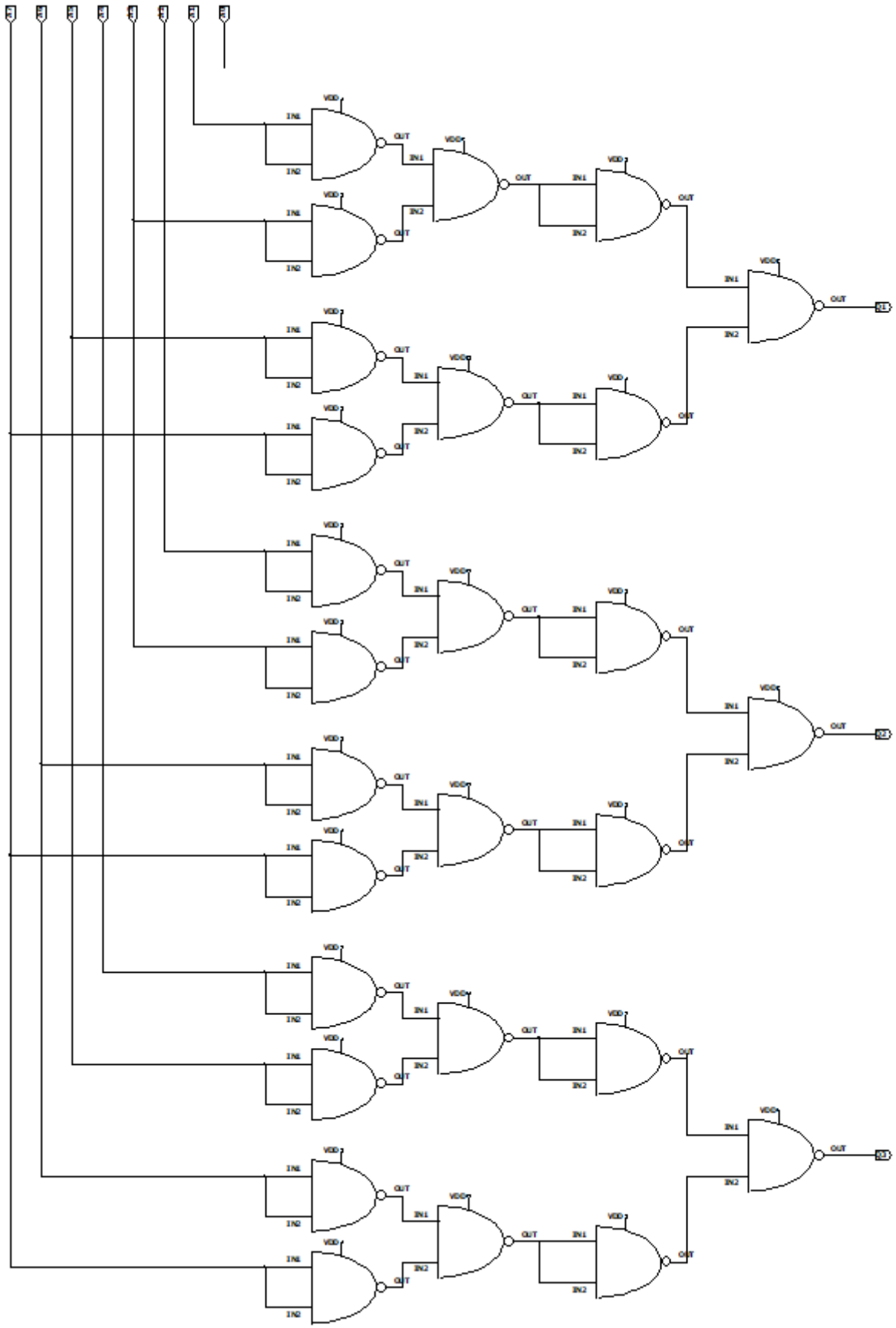


Шифратор «8 в 3» подразумевает наличие 8 входов (a_0, a_1, … a_7) и 3 выходов (Q_1, Q_2 и Q_4). Логические формулы для сигналов представлены ниже:

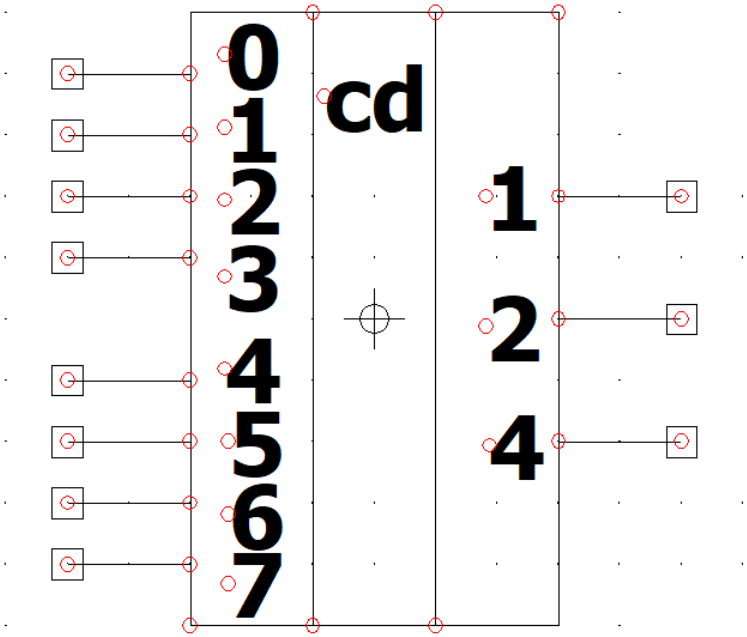
$$Q_1 = a_7 + a_5 + a_3 + a_1;$$
$$Q_2 = a_7 + a_6 + a_3 + a_2;$$
$$Q_4 = a_7 + a_6 + a_5 + a_4.$$

Мы видим, что сигнал a_0 не влияет на результаты выходов, а все остальные так или иначе участвуют в формировании конечного сигнала.

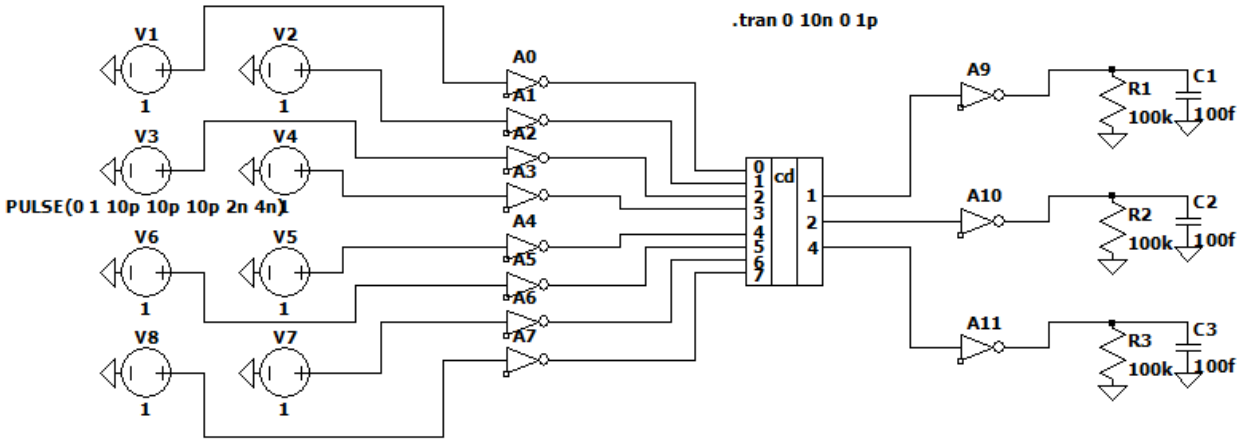
Схема разработанного шифратора «8 в 3» представлена на скриншоте:



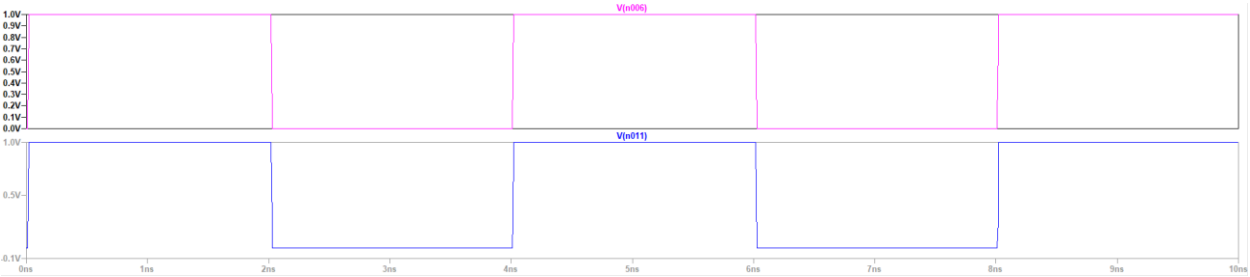
Символ данного БОЭ будет изображён так:



Для тестирования шифратора создадим такую схему:



В ней все сигналы на входе в шифратор будут содержать 0, кроме a_2. Взглянув на формулы, заметим, что на выходе должен синхронно со входом пульсировать выход Q_2. Проверим это с помощью симуляции. Верхний график соответствует входной пульсации на шифратор до инвертора, а нижний – выходному, снятому после инвертора.



Если не считать сигнал a_0, все возможные сигналы имеют одинаковую длину маршрута, а значит и максимальное время задержки у любого пути одинаково.

Выполнение части 2

Для упрощения работы я создал отдельный модуль для логического ИЛИ на основе И-НЕ, который принимает на вход 4 сигнала, и уже с его помощью описал шифратор.

Модуль логического ИЛИ:

```
`timescale 1ns / 1ps

    module or_by_nand(
        input a0,
        input a1,
        input a2,
        input a3,
        output res
    );

    wire not_a0, not_a1, not_a2, not_a3, or_1, or_2, not_or1, not_or2;

    nand(not_a0, a0, a0);
    nand(not_a1, a1, a1);
    nand(or_1, not_a0, not_a1);
    nand(not_or1, or_1, or_1);

    nand(not_a2, a2, a2);
    nand(not_a3, a3, a3);
    nand(or_2, not_a2, not_a3);
    nand(not_or2, or_2, or_2);

    nand(res, not_or1, not_or2);

endmodule
```

Его тестирующий модуль:

```
`timescale 1ns / 1ps

    module or_tb;
    reg a_0, a_1, a_2, a_3;
    wire c_out;

    or_by_nand or_by_nand_1 (
        .a0(a_0),
        .a1(a_1),
        .a2(a_2),
        .a3(a_3),
        .res(c_out)
    );

    integer i;
    reg [3:0] test_val;
    reg expected_val;

    initial begin
        for (i = 0; i < 16; i = i+1) begin
            test_val = i;
            a_0 = test_val[0];
            a_1 = test_val[1];
            a_2 = test_val[2];
            a_3 = test_val[3];
            expected_val = |test_val;
            #10

            if (c_out == expected_val) begin
                $display ("Correct!!! a_0=%b, a_1=%b, a_2=%b, a_3=%b, res=%b", a_0, a_1, a_2, a_3, c_out);
            end else begin
                $display ("Wrong!!! a_0=%b, a_1=%b, a_2=%b, a_3=%b, res=%b, expected=%b", a_0, a_1, a_2, a_3, c_out, expected_val);
            end
        end
        #10 $stop;
    end

endmodule
```

Результат тестирования (вывод в консоль):

```
# run 1000ns
Correct!!! a_0=0, a_1=0, a_2=0, a_3=0, res=0
Correct!!! a_0=1, a_1=0, a_2=0, a_3=0, res=1
Correct!!! a_0=0, a_1=1, a_2=0, a_3=0, res=1
Correct!!! a_0=1, a_1=1, a_2=0, a_3=0, res=1
Correct!!! a_0=0, a_1=0, a_2=1, a_3=0, res=1
Correct!!! a_0=1, a_1=0, a_2=1, a_3=0, res=1
Correct!!! a_0=0, a_1=1, a_2=1, a_3=0, res=1
Correct!!! a_0=1, a_1=1, a_2=1, a_3=0, res=1
Correct!!! a_0=0, a_1=0, a_2=0, a_3=1, res=1
Correct!!! a_0=1, a_1=0, a_2=0, a_3=1, res=1
Correct!!! a_0=0, a_1=1, a_2=0, a_3=1, res=1
Correct!!! a_0=1, a_1=1, a_2=0, a_3=1, res=1
Correct!!! a_0=0, a_1=0, a_2=1, a_3=1, res=1
Correct!!! a_0=1, a_1=0, a_2=1, a_3=1, res=1
Correct!!! a_0=0, a_1=1, a_2=1, a_3=1, res=1
Correct!!! a_0=1, a_1=1, a_2=1, a_3=1, res=1
$stop called at time : 170 ns : File "E:/vivado_11/shifrator/shifrator.srcs/sim_1/new/or_tb.v" Line 35
```

Сам модуль шифратора:

```
`timescale 1ns / 1ps

    module shifrator(
        input [7:0] a,
        output q1,
        output q2,
        output q4
    );

    or_by_nand quit1(.a0(a[1]), .a1(a[3]), .a2(a[5]), .a3(a[7]), .res(q1));
    or_by_nand quit2(.a0(a[2]), .a1(a[3]), .a2(a[6]), .a3(a[7]), .res(q2));
    or_by_nand quit3(.a0(a[4]), .a1(a[5]), .a2(a[6]), .a3(a[7]), .res(q4));

endmodule
```

Тесты для шифратора:

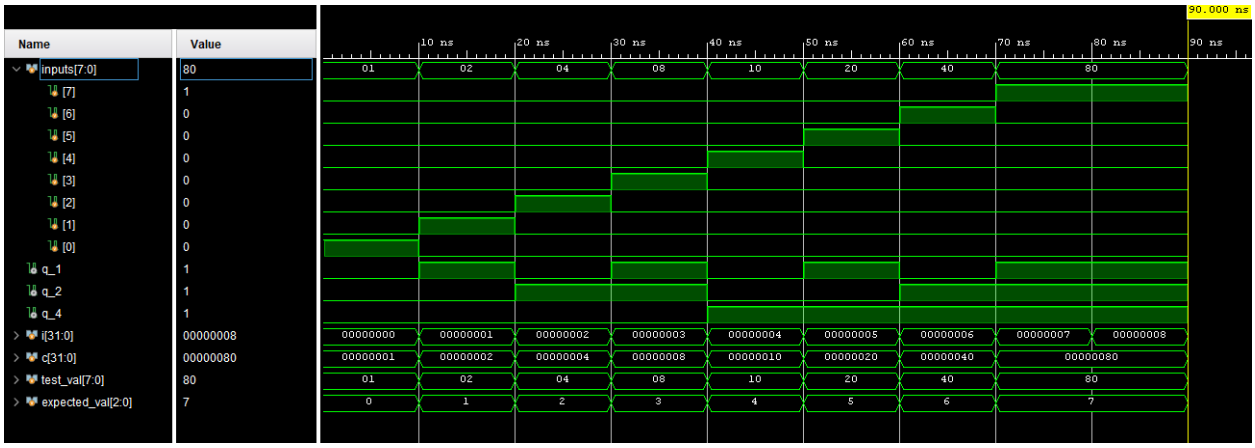
```
 `timescale 1ns / 1ps

module shifrator_tb;
    reg [7:0] inputs;
    wire q_1, q_2, q_4;

    shifrator shifrator_1(
        .a(inputs),
        .q1(q_1),
        .q2(q_2),
        .q4(q_4)
    );

    integer i, c;
    reg [7:0] test_val;
    reg [2:0] expected_val;

    initial begin
        for (i = 0; i < 8; i = i+1) begin
            c = $pow(2, i);
            test_val = c;
            inputs = test_val;
            expected_val = i;
            #10
            if (q_1 == expected_val[0] && q_2 == expected_val[1] && q_4 == expected_val[2]) begin
                $display ("Correct for %b: q4=%b, q2=%b, q1=%b", c, q_4, q_2, q_1);
            end else begin
                $display ("Wrong for %b: q4=%b, q2=%b, q1=%b", c, q_4, q_2, q_1);
            end
        end
        #10 $stop;
    end

endmodule
```

Вывод в консоль:

```
# run 1000ns
Correct for 00000000000000000000000000000001: q4=0, q2=0, q1=0
Correct for 00000000000000000000000000000010: q4=0, q2=0, q1=1
Correct for 00000000000000000000000000000100: q4=0, q2=1, q1=0
Correct for 00000000000000000000000000001000: q4=0, q2=1, q1=1
Correct for 00000000000000000000000000010000: q4=1, q2=0, q1=0
Correct for 00000000000000000000000000100000: q4=1, q2=0, q1=1
Correct for 00000000000000000000000001000000: q4=1, q2=1, q1=0
Correct for 00000000000000000000000010000000: q4=1, q2=1, q1=1
$stop called at time : 90 ns : File "E:/vivado_ll/shifrator/shifrator.srcs/sim_1/new/shifrator_tb.v" Line 31
```

В данном случае тестируются только случаи, при которых шифратор должен работать (только один из сигналов активен), а не перебор всех 256 вариантов, поскольку задача шифратора состоит именно в преобразовании одного сигнала из 8 в более укороченном виде, а не отсечение возможных одновременно приходящих сигналов.

Временная диаграмма тестирования выглядит следующим образом:



На ней видно, что входные сигналы идут «лесенкой», что демонстрирует полное тестирование, а в качестве выходных сигналов складывается номер входного порта в двоичном виде.

Вывод

Итак, в процессе выполнения данной лабораторной работы я познакомился со средой LTspice и с языком описания аппаратуры Verilog HDL. В качестве опытного образца я создал собственный вентиль NAND и на его основе создал и протестировал позиционный шифратор «8 в 3».

Все файлы, связанные с данной лабораторной работой, доступны на GitHub: https://github.com/ANegrash/Functional_circuitry