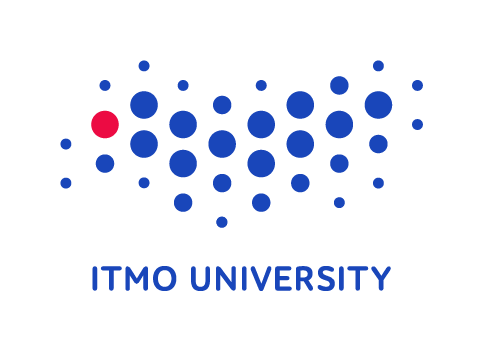
Санкт-Петербургский Национальный Исследовательский

Университет ИТМО

Факультет программной инженерии и компьютерной техники



Вариант № 6

Лабораторная работа № 1

«Введение в проектирование цифровых интегральных схем»

по дисциплине

*«Функциональная схемотехника»*

Выполнил: Антонов Всеволод Владимирович

Студент группы: P33301

Преподаватель:

Салонина Екатерина Александровна

Санкт-Петербург, 2023 г.

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:

Логический базис – NAND; БОЭ – позиционный дешифратор «3 в 8».

# Часть №1. LTSpice.

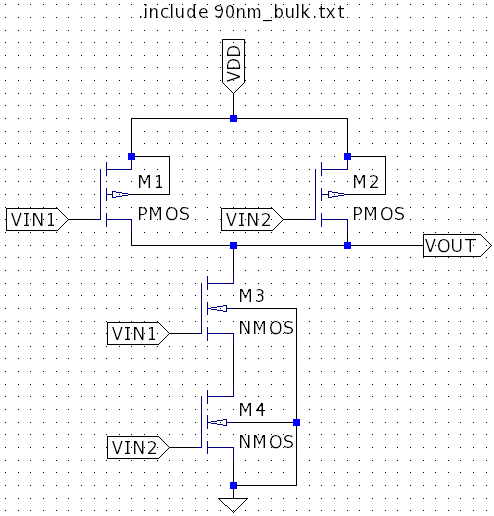
## Разработка вентиля.

Таблица истинности вентиля:

| **A** | **B** | **R** |
| --- | --- | --- |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Схема разработанного вентиля:

VIN{1, 2} – входы, VOUT - выход, VDD – напряжение питания; использовано по 2 транзистора PMOS и NMOS.



Символ вентиля:

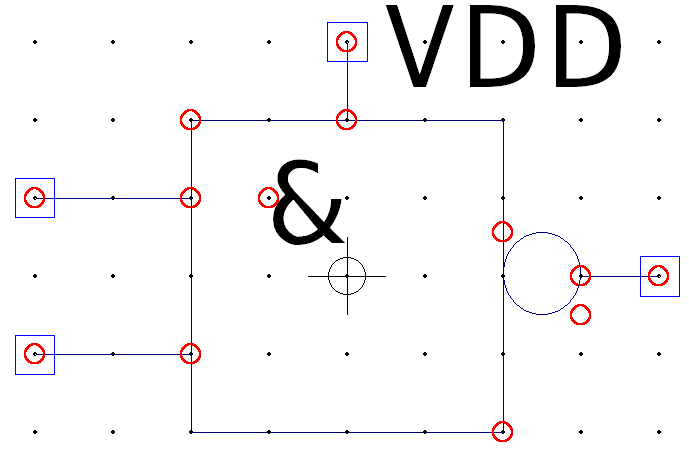
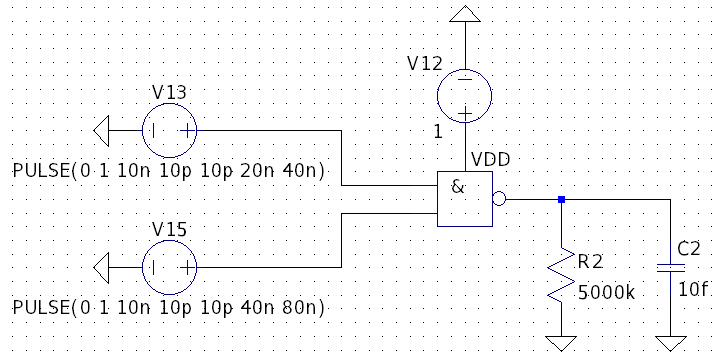
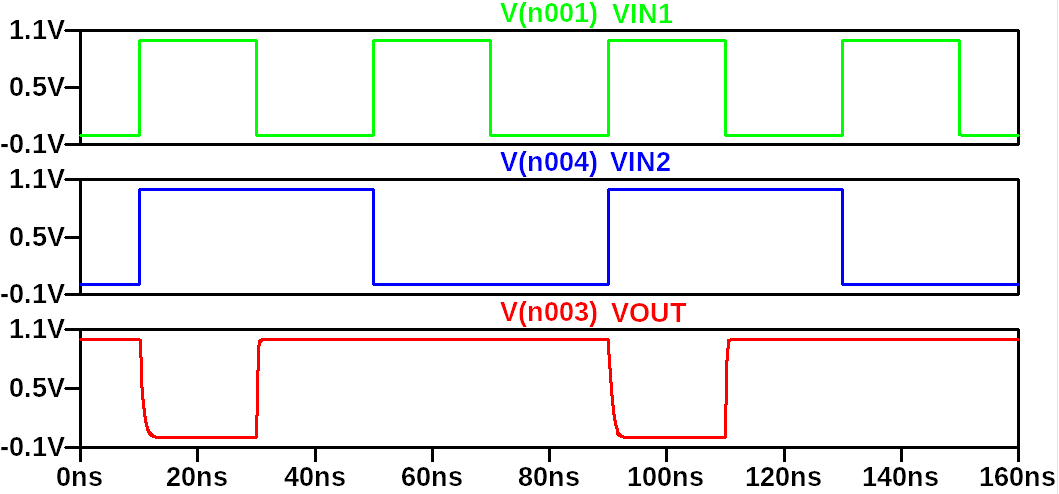


Схема тестирования:



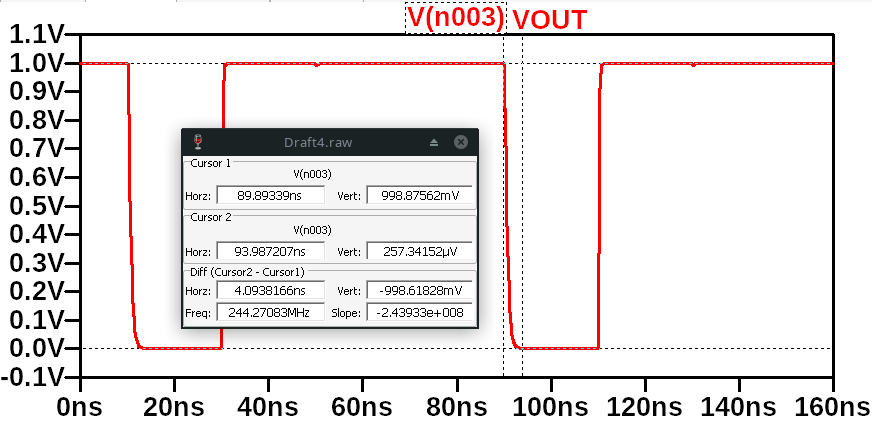
VDD = VIN = 1 В; начальное напряжение - 0 В, активное напряжение - 1 В, задержка запуска - 10 нс, время фронта и спада - 10 пс, активное время первого источника напряжения - 20 нс, период - 40 нс, для следующего последние две характеристики в два раза больше, а частоты, соответственно, меньше; резистор и конденсатор отвечают за имитацию задержки, при этом чем сопротивление прямо пропорционально высоте сигнала, а ёмкость – величине задержки.

Временная диаграмма процесса тестирования вентиля:



1 В – логическая единица, 0 В – логический ноль; NAND подаёт на выход единицу во всех случаях, кроме равенства единице обоих входных сигналов. Это отображено на рисунке – на 10 и 90 нс выходной сигнал падает в ноль и остаётся таким в течение 10 нс, после чего выходит обратно в единицу.

Результат измерения задержки распространения сигнала через вентиль:



Два курсора на верхней (~ 1 В) и нижней (~ 0 В) границах заднего фронта.

Задержка равна T ~ 4,1 нс

Тогда максимальная частота работы вентиля равна f = 1/T = 244 МГц

# Разработка БОЭ.

На базе данного вентиля для удобства разработки БОЭ (дешифратора 3 в 8) я создал также инвертор и NAND с 4 входами.

Схема и символ инвертора.

Логическое выражение:

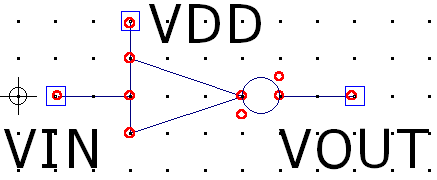
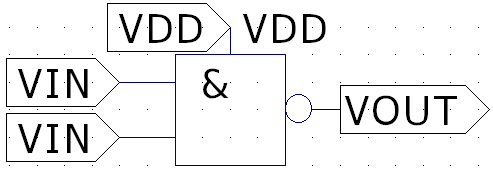
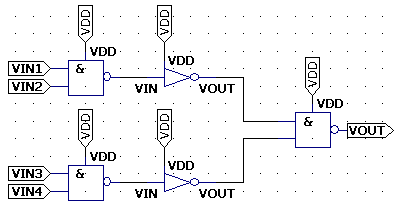


Схема и символ NAND с 4 входами.

Логическое выражение:



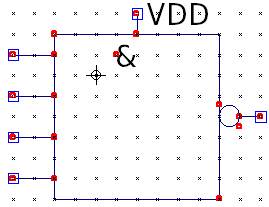
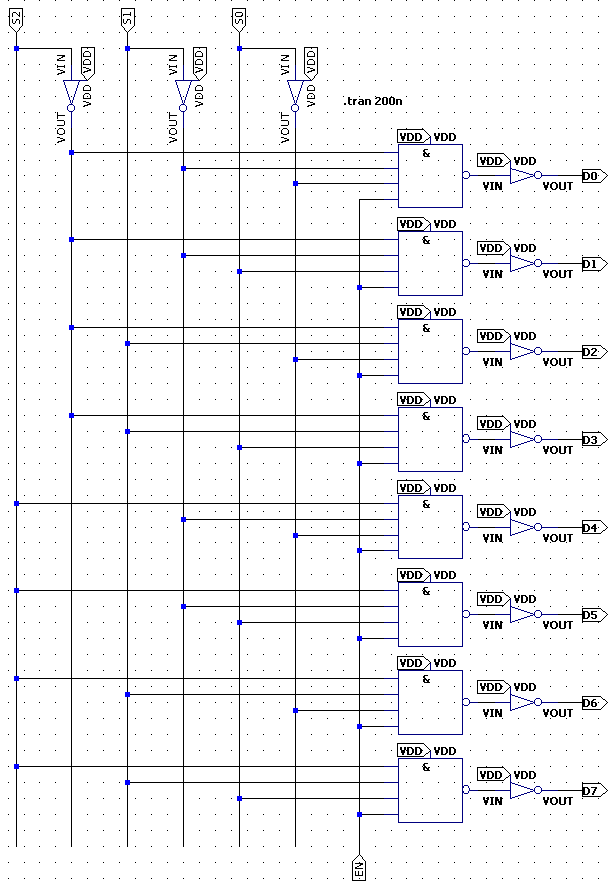


Схема разработанного БОЭ:



Модуль преобразует трёхразрядное двоичное число на входе в десятичное на выходе (вернее, в его унитарный код). На вход модулю подаётся двоичное число от 0 до 7 (входы S{0-2} соответствуют его разрядам, S2 - старший, S0 - младший), на выходе находятся 8 бит (D{0-7}), один из которых, порядок которого соответствует входу, становится равным 1, если сигнал разрешения EN (Enable), отвечающий за активность дешифратора, равен 1 (при равном 0 весь выход тоже будет нулевой). При изменении входа единице станет равен уже другой бит, а предыдущий обнулится. Таким образом, дешифратор реализует модель one-hot (без инверторов перед выходами тенденция была бы обратной (1 – неактивное, 0 – активное), что называется one-cold).

Символ разработанного БОЭ:

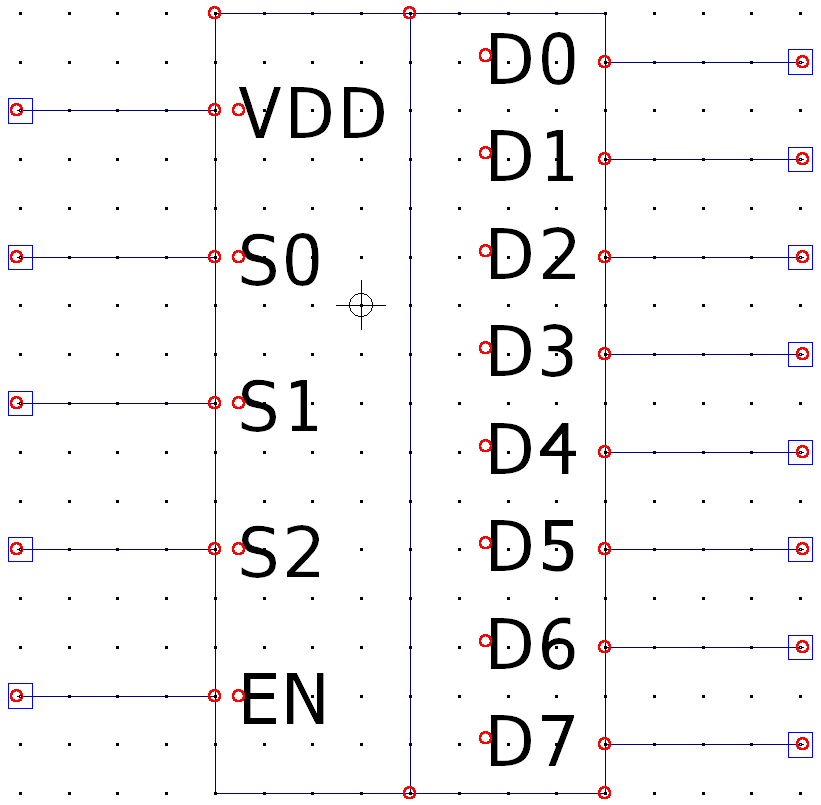
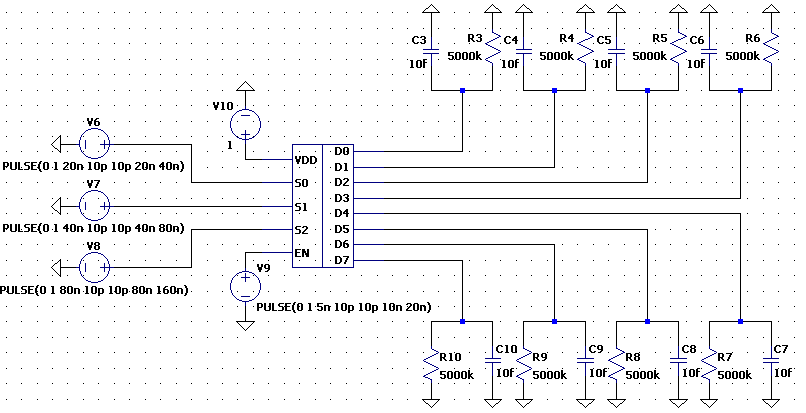
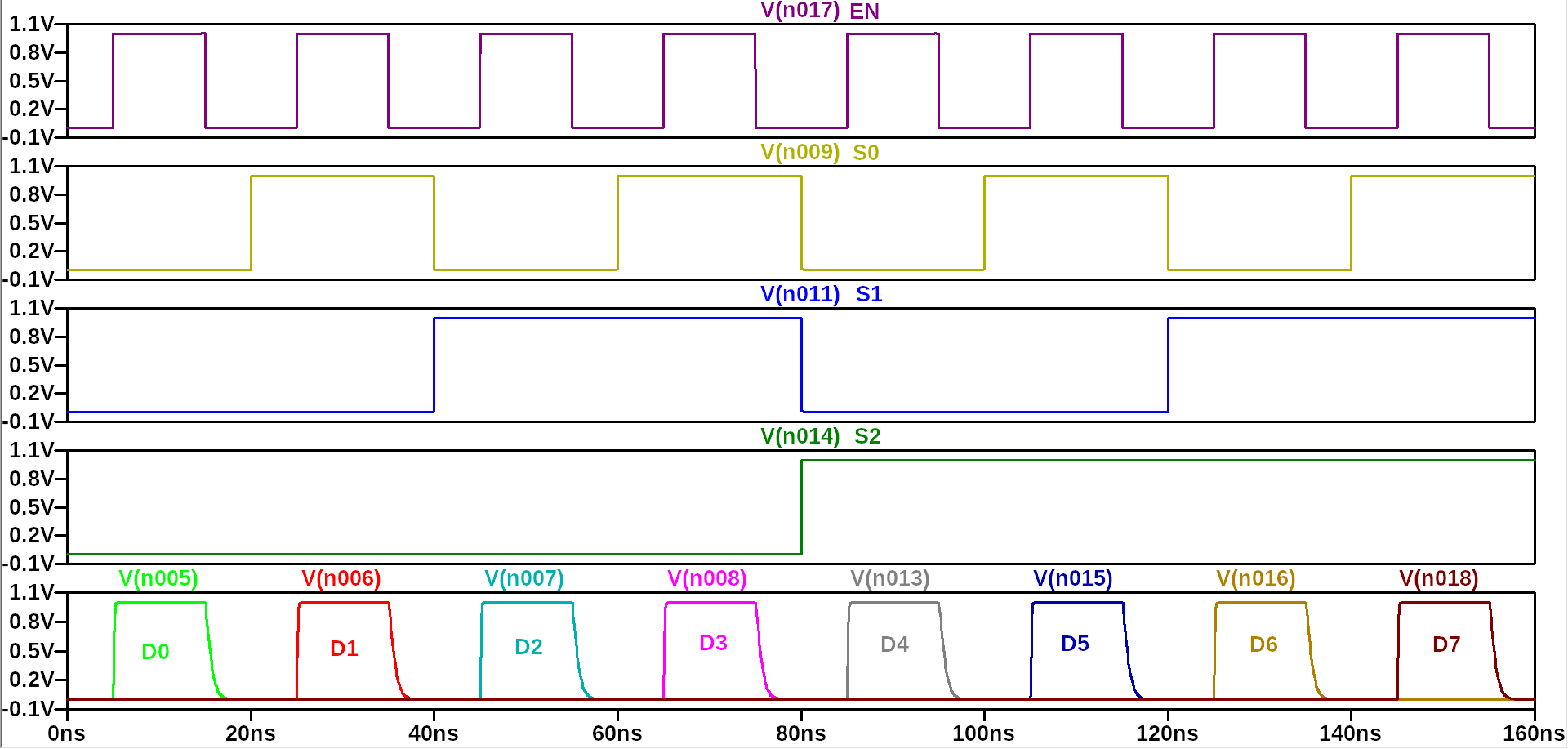


Схема тестирования:

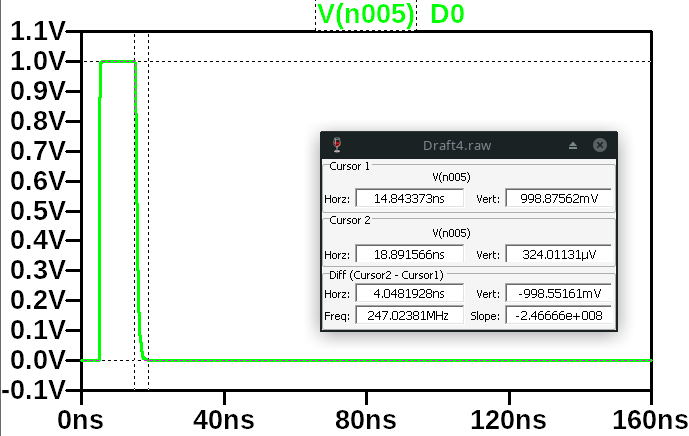


Как и в случае с тестированием вентиля, к каждому выходу я присоединил по резистору и конденсатору. Частота сигнала на младшем, среднем и старшем входе больше частоты сигнала на EN в 2, 4 и 8 раз соответственно; так сделано в целях задания последовательной подачи чисел от 0 до 7, переключения активных выходов и EN таким образом, чтобы при изменении сигналов выходы всегда были неактивны и не происходило выбросов.

Временная диаграмма процесса тестирования БОЭ:



Результат измерения задержки распространения сигнала через БОЭ:



Два курсора на верхней (~ 1 В) и нижней (~ 0 В) границах заднего фронта.

Задержка равна T ~ 4,05 нс

Тогда максимальная частота работы БОЭ равна f = 1/T = 247 МГц

## Часть № 2. Verilog.

Код разработанного модуля БОЭ:

**`timescale** 1ns / 1ps

module decoder(

input [2:0] s,

input en,

output [7:0] d

);

**wire** [2:0] not\_s;

**wire** [7:0] not\_d;

**wire** [7:0] s\_2\_1;

**wire** [7:0] not\_s\_2\_1;

**wire** [7:0] s\_0\_en;

**wire** [7:0] not\_s\_0\_en;

// nand(not\_s, s, s);

nand(not\_s[2], s[2], s[2]);

nand(not\_s[1], s[1], s[1]);

nand(not\_s[0], s[0], s[0]);

nand(not\_s\_2\_1[0], not\_s[2], not\_s[1]);

nand(not\_s\_0\_en[0], not\_s[0], en);

nand(s\_2\_1[0], not\_s\_2\_1[0],not\_s\_2\_1[0]);

nand(s\_0\_en[0], not\_s\_0\_en[0], not\_s\_0\_en[0]);

nand(not\_d[0], s\_2\_1[0], s\_0\_en[0]);

nand(not\_s\_2\_1[1], not\_s[2], not\_s[1]);

nand(not\_s\_0\_en[1], s[0], en);

nand(s\_2\_1[1], not\_s\_2\_1[1],not\_s\_2\_1[1]);

nand(s\_0\_en[1], not\_s\_0\_en[1], not\_s\_0\_en[1]);

nand(not\_d[1], s\_2\_1[1], s\_0\_en[1]);

nand(not\_s\_2\_1[2], not\_s[2], s[1]);

nand(not\_s\_0\_en[2], not\_s[0], en);

nand(s\_2\_1[2], not\_s\_2\_1[2],not\_s\_2\_1[2]);

nand(s\_0\_en[2], not\_s\_0\_en[2], not\_s\_0\_en[2]);

nand(not\_d[2], s\_2\_1[2], s\_0\_en[2]);

nand(not\_s\_2\_1[3], not\_s[2], s[1]);

nand(not\_s\_0\_en[3], s[0], en);

nand(s\_2\_1[3], not\_s\_2\_1[3],not\_s\_2\_1[3]);

nand(s\_0\_en[3], not\_s\_0\_en[3], not\_s\_0\_en[3]);

nand(not\_d[3], s\_2\_1[3], s\_0\_en[3]);

nand(not\_s\_2\_1[4], s[2], not\_s[1]);

nand(not\_s\_0\_en[4], not\_s[0], en);

nand(s\_2\_1[4], not\_s\_2\_1[4],not\_s\_2\_1[4]);

nand(s\_0\_en[4], not\_s\_0\_en[4], not\_s\_0\_en[4]);

nand(not\_d[4], s\_2\_1[4], s\_0\_en[4]);

nand(not\_s\_2\_1[5], s[2], not\_s[1]);

nand(not\_s\_0\_en[5], s[0], en);

nand(s\_2\_1[5], not\_s\_2\_1[5],not\_s\_2\_1[5]);

nand(s\_0\_en[5], not\_s\_0\_en[5], not\_s\_0\_en[5]);

nand(not\_d[5], s\_2\_1[5], s\_0\_en[5]);

nand(not\_s\_2\_1[6], s[2], s[1]);

nand(not\_s\_0\_en[6], not\_s[0], en);

nand(s\_2\_1[6], not\_s\_2\_1[6],not\_s\_2\_1[6]);

nand(s\_0\_en[6], not\_s\_0\_en[6], not\_s\_0\_en[6]);

nand(not\_d[6], s\_2\_1[6], s\_0\_en[6]);

nand(not\_s\_2\_1[7], s[2], s[1]);

nand(not\_s\_0\_en[7], s[0], en);

nand(s\_2\_1[7], not\_s\_2\_1[7],not\_s\_2\_1[7]);

nand(s\_0\_en[7], not\_s\_0\_en[7], not\_s\_0\_en[7]);

nand(not\_d[7], s\_2\_1[7], s\_0\_en[7]);

// nand(not\_d[0], not\_s[2], not\_s[1], not\_s[0], en);

// nand(not\_d[1], not\_s[2], not\_s[1], s[0], en);

// nand(not\_d[2], not\_s[2], s[1], not\_s[0], en);

// nand(not\_d[3], not\_s[2], s[1], s[0], en);

// nand(not\_d[4], s[2], not\_s[1], not\_s[0], en);

// nand(not\_d[5], s[2], not\_s[1], s[0], en);

// nand(not\_d[6], s[2], s[1], not\_s[0], en);

// nand(not\_d[7], s[2], s[1], s[0], en);

// nand(d, not\_d, not\_d);

nand(d[0], not\_d[0], not\_d[0]);

nand(d[1], not\_d[1], not\_d[1]);

nand(d[2], not\_d[2], not\_d[2]);

nand(d[3], not\_d[3], not\_d[3]);

nand(d[4], not\_d[4], not\_d[4]);

nand(d[5], not\_d[5], not\_d[5]);

nand(d[6], not\_d[6], not\_d[6]);

nand(d[7], not\_d[7], not\_d[7]);

endmodule

Вход: 3-битная шина s с декодируемым числом и сигнал разрешения en.

Выход: 8-битная шина d, где порядок равного 1 бита соответствует декодированному числу.

Используются 6 вспомогательных (5 6-битных и 1 3-битная) шин для передачи результатов операции NAND.

not\_s, not\_d – инвертированные биты шин s и d.

(not\_)s\_2\_1[n] - (не)инвертированные биты результата (not\_)s[2] NAND (not\_)s[1] (т.е. выполнение NAND для соответствующих числу n по инверсии старшего и среднего битов).

(not\_)s\_0\_en[n] - (не)инвертированные биты результата (not\_)s[0] NAND (not\_)en (т.е. выполнение NAND для соответствующего числу n по инверсии младшего бита и сигнала разрешения).

Можно заметить, что в коде 8 раз повторяются разделённые переносом 5 строчек, которые отличаются только номером бита и присутствием/отсутствием приставки «not\_» для нужной для этого числа инверсии.

Сначала вычисляются инвертированные биты s, потом по очереди инвертированные биты not\_d (с указанными выше промежуточными вычислениями NAND), которые инвертируются для получения итогового вывода d.

Код разработанного тестового окружения БОЭ:

**`timescale** 1ns / 1ps

module decoder\_tb;

**reg** [2:0] s;

**wire**[7:0] d;

**reg** en;

**integer** i;

decoder decoder\_1(

.s(s),

.d(d),

.en(en)

);

initial begin

for(i = 0; i < 8; i = i+1) begin

s = i;

en = 1;

#10

if (d == 2\*\*i) begin

**$display**(**"Correct! s=%b, d=%b, en=%b, i=%0d"**, s, d, en, i);

end else begin

**$display**(**"Incorrect! s=%b, d=%b, en=%b, i=%0d"**, s, d, en, i);

end

en = 0;

#10

if (d == 0) begin

**$display**(**"Correct! s=%b, d=%b, en=%b, i=%0d"**, s, d, en, i);

end else begin

**$display**(**"Incorrect! s=%b, d=%b, en=%b, i=%0d"**, s, d, en, i);

end

end

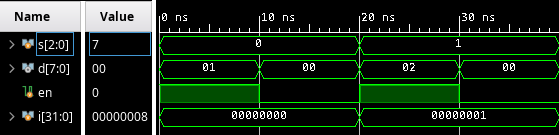
#10 **$stop**;

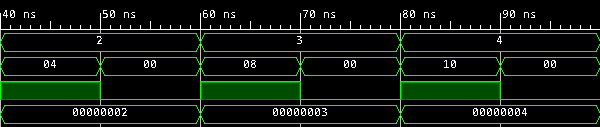
end

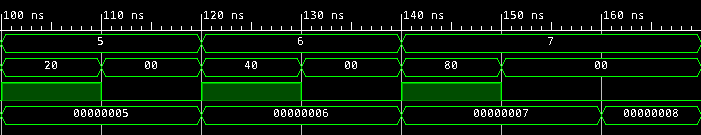
endmodule

В тестовом окружении задаются переменные входа (s, en) и выхода (d), а также целочисленная переменная i, которая будет изменяться в цикле от 0 до 8; её значение (кроме финального 8) будет присваиваться шине s. Затем проверяется работоспособность дешифратора при включенном и выключенном сигнале разрешения; так как число на входе соответствует порядку равного 1 бита, можно проверять это как равенство результата степени двойки этого числа.

Временная диаграмма процесса тестирования БОЭ:







Как можно видеть, во время процесса тестирования переменная i изменялась от 0 до 8 каждые 20 нс, переменная s – от 0 до 7 (в последний раз присваивания не происходило), сигнал en каждые 10 нс инвертировался, вследствие чего переменная d в течение этого промежутка времени менялась от степени двойки числа s к 0 и наоборот.

Вывод в консоль:

Correct! s=000, d=00000001, en=1, i=0

Correct! s=000, d=00000000, en=0, i=0

Correct! s=001, d=00000010, en=1, i=1

Correct! s=001, d=00000000, en=0, i=1

Correct! s=010, d=00000100, en=1, i=2

Correct! s=010, d=00000000, en=0, i=2

Correct! s=011, d=00001000, en=1, i=3

Correct! s=011, d=00000000, en=0, i=3

Correct! s=100, d=00010000, en=1, i=4

Correct! s=100, d=00000000, en=0, i=4

Correct! s=101, d=00100000, en=1, i=5

Correct! s=101, d=00000000, en=0, i=5

Correct! s=110, d=01000000, en=1, i=6

Correct! s=110, d=00000000, en=0, i=6

Correct! s=111, d=10000000, en=1, i=7

Correct! s=111, d=00000000, en=0, i=7

Выводы по работе:

Я освежил свои навыки работы с LTSpice (с которым сталкивался ещё на робототехнике до перевода с потерей года), узнал о нескольких дополнительных функциях (например, о создании символов, задании портов, изменении толщины текста и линий на графиках и их размеров), познакомился со средой разработки интегральных микросхем Vivado, языком описания аппаратуры Verilog и его базовым синтаксисом, научился писать на нём модули и тестовые окружения, разобрался в принципе работы вентиля NAND и дешифратора 3 в 8, создав их схемы (дешифратор на основе NAND), и узнал о one-hot и one-cold кодировании, несколько раз переделывая их с нуля.