Первый шаг в цифровой схемотехнике и языке описания аппаратуры SystemVerilog

Дмитрий Смехов

ЛяЛямбда 2003

DigitalDesignSchool

- Школа синтеза цифровых схем
  https://youtube.com/@user-nh3su7wd6e
- https://github.com/yuri-panchul/basics-graphics-music
- HDL, RTL and FPGA
  https://bit.ly/2022-08-01-verilog-1-bishkek-yuri-panchul

Модель OSI

Прикладной
Представления
Сеансовый
Транспортный
Сетевой
Канальный
Физический

Упрощённая модель

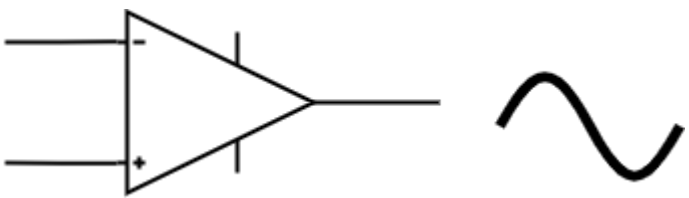
Структура
Устройство
Печатная плата
Микроархитектура
Физическая микроэлектроника

Поведение
Application
Протоколы взаимодействия
Микроархитектура
Физическая микроэлектроника

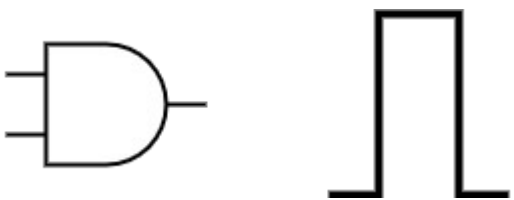
Цифровая или Аналоговая схема

Первый уровень разделения
- цифровая схемотехника
- аналоговая схемотехника

Аналоговая схема



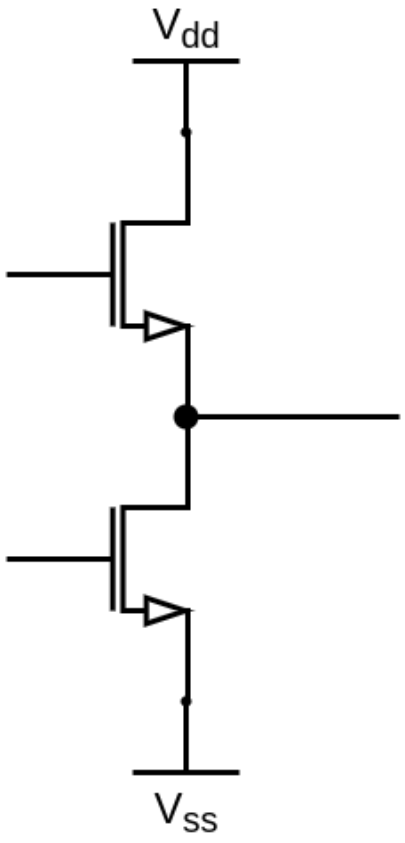
Цифровая схема



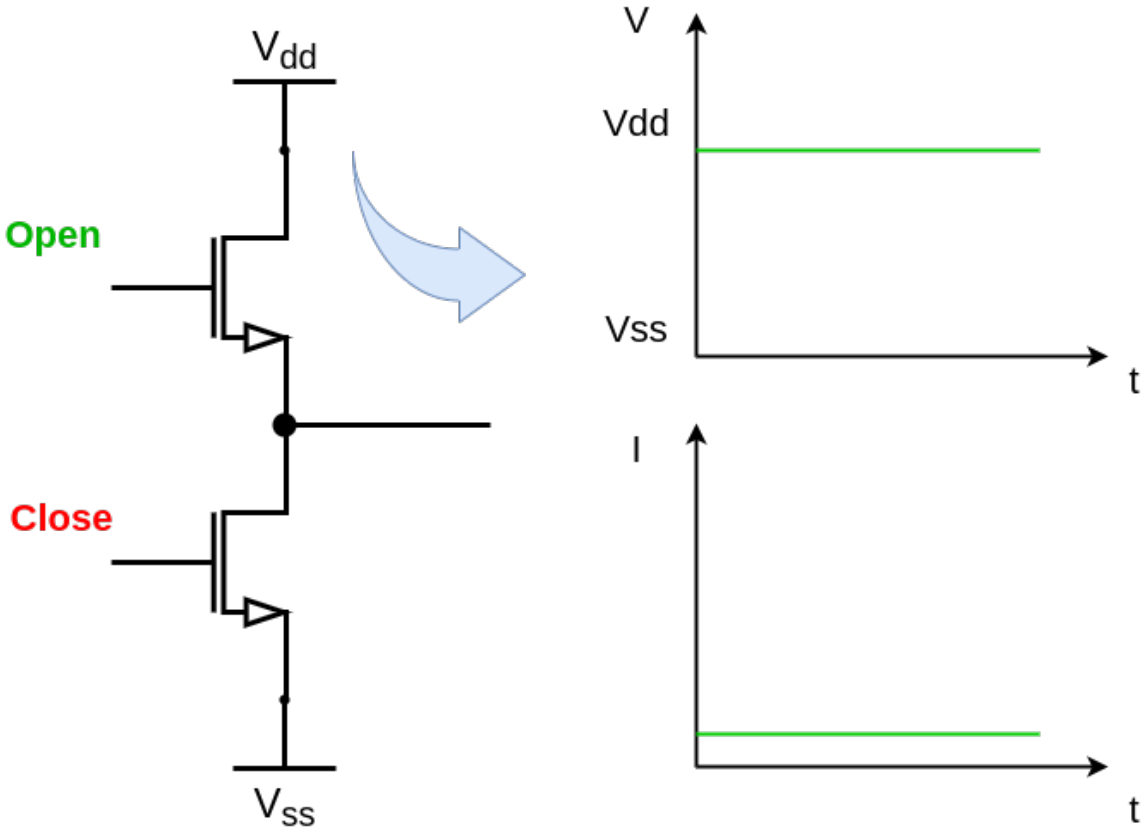
Двоичный код: 0 или 1

Троичный код: -1, 0, +1

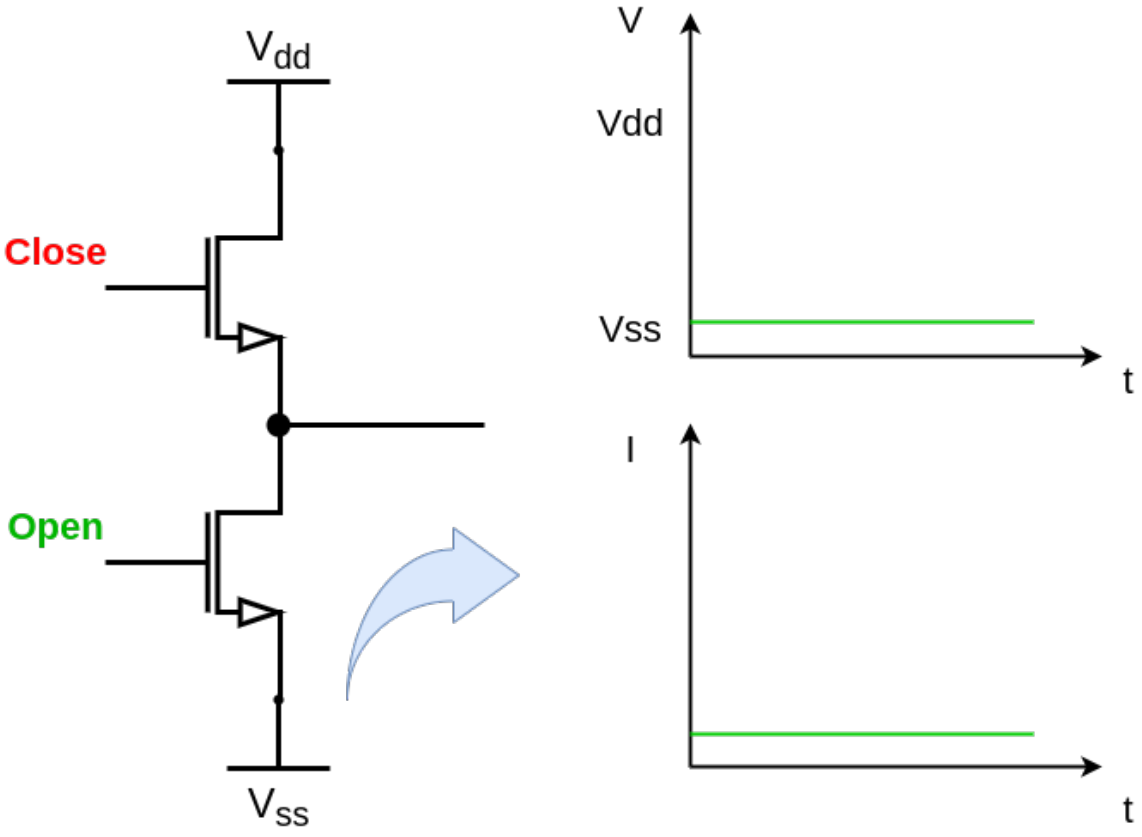
Выходной каскад цифровой схемы



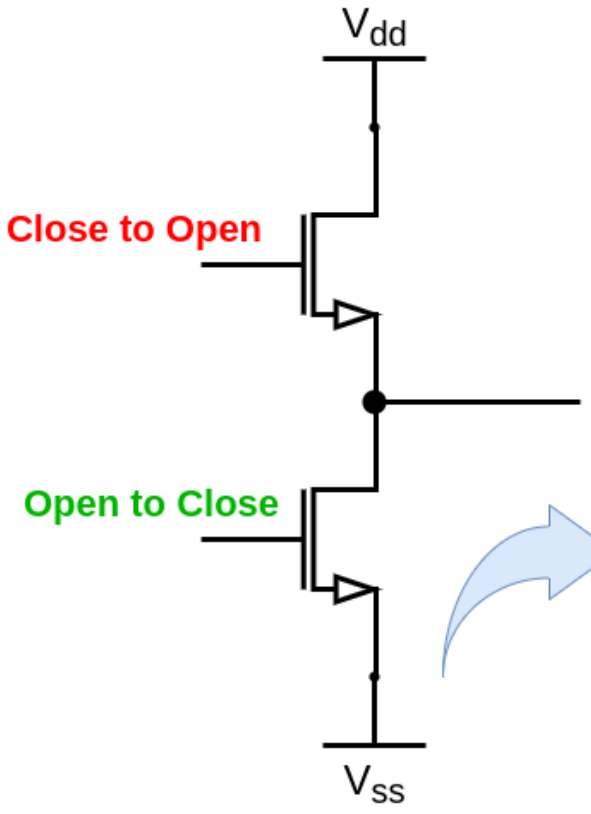
Формирование 1

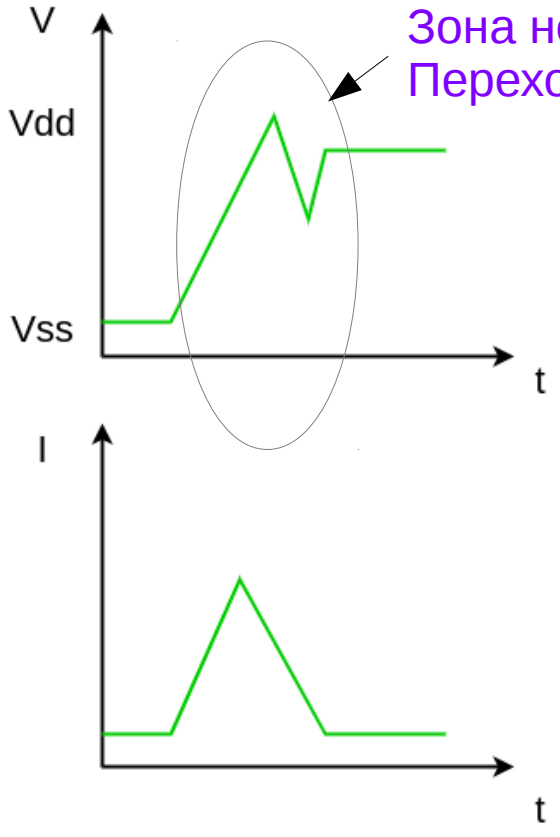


Формирование 0



Переход между 0 и 1





Зона неопределённости.
Переходный процесс

2023

Полезная книга

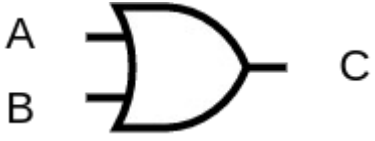
Базис И, ИЛИ, НЕ

Логическое И

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

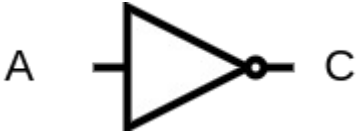
Логическое ИЛИ

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

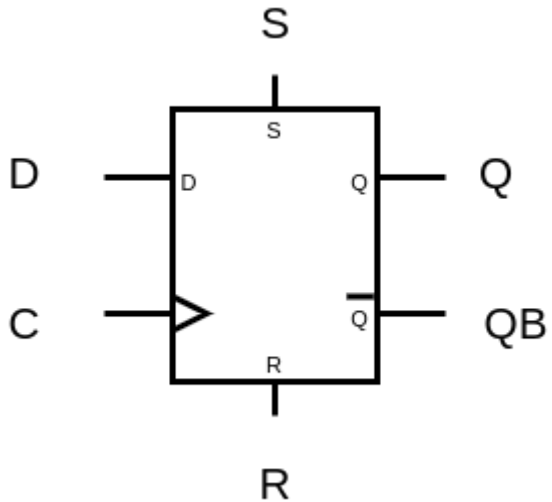


Логическое НЕ

A	C
0	1
1	0



D-триггер

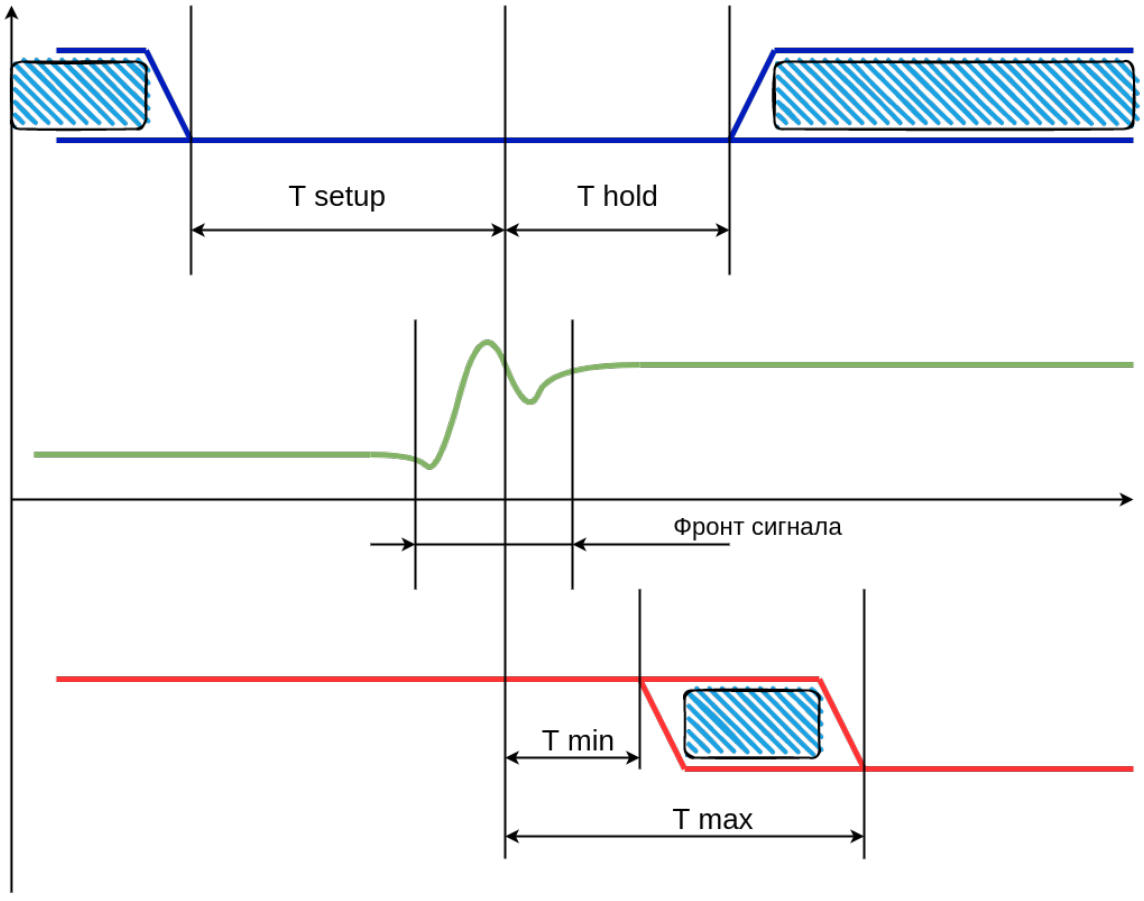


Логическое И

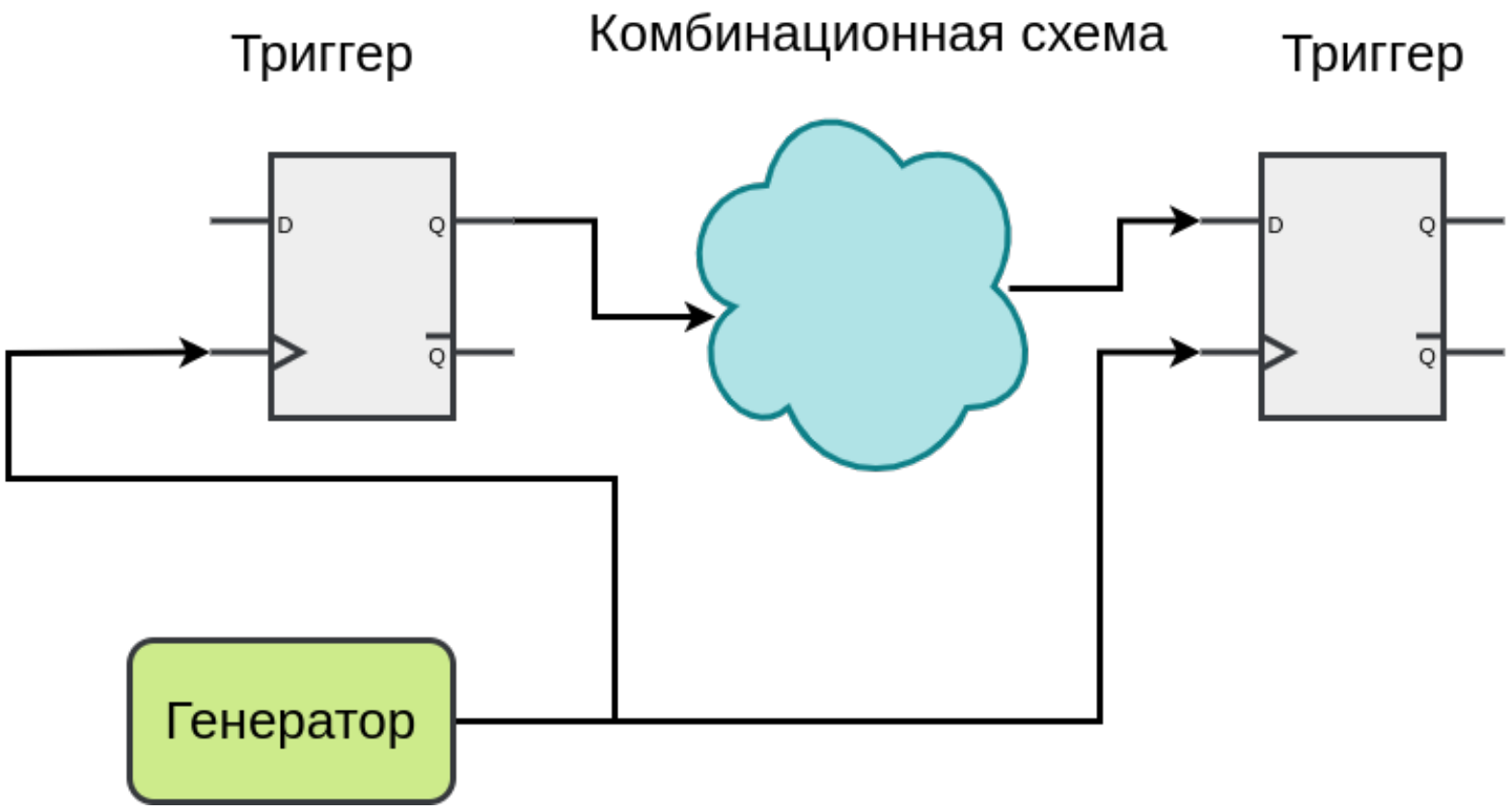
D	S	R	C	Q	QB
X	X	1	X	0	1
X	1	0	X	1	0
0	0	0	R	0	1
1	0	0	R	1	0

Фронт сигнала.
Переход из 0 в 1.

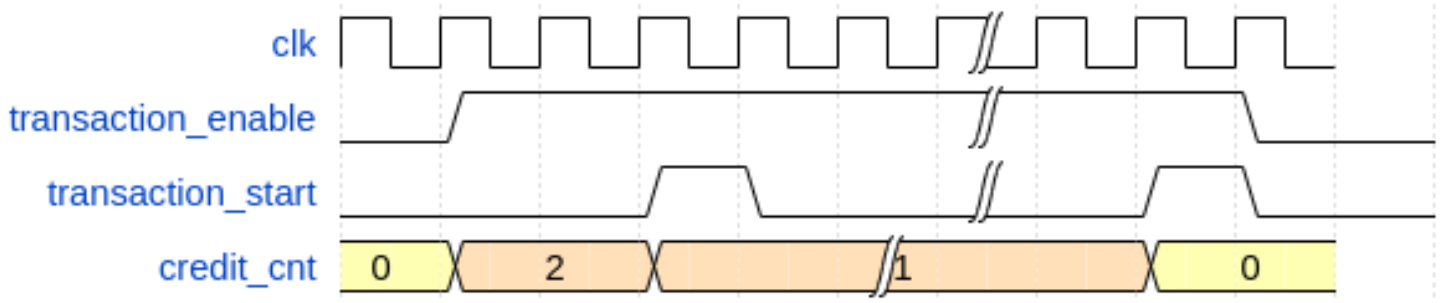
D-триггер — фиксация данных



Синхронные схемы



Временная диаграмма



Языки описания аппаратуры

- VHDL - основан на языке ADA
- Verilog
- SystemVerilog - основан на Verilog,
- А также другие
  - Chisel
  - C++ (Catapult, Vivado HLS)
  - C# (Quokka)
  - MATLAB HDL-Coder
  - Labview

SystemVerilog

Создан на основе следующих языков:

- Verilog
- Superlog - сложные типы данных
- HVL — объектно-ориентированность
- PSL - SystemVerilog Assertions

Где можно посмотреть:

- https://www.chipverify.com/systemverilog/systemverilog-tutorial
- ChatGPT

Module

```
module my_and2 (
   input    wire  a,   //  входной сигнал
   input    wire  b,
   output  wire  c    // выходной сигнал
);

assign  c = a & b;

endmodule
```

Иерархия модулей

```
module my_and3 (
   input    wire [2:0]  a,   //  шина из трёх сигналов (это не массив)
   output  wire          c    // выходной сигнал
);

logic      tmp;

my_and2   inst_0
(
        .a      ( a[0] ),
        .b      ( a[1] )
        .c      ( tmp )
);

assign  c = tmp & a[2];

endmodule
```

Опечатка в имени сигнала

```
module my_and3 (
   input    wire [2:0]  a,   //  шина из трёх сигналов (это не массив)
   output   wire        c    // выходной сигнал
);

logic      tmp;

my_and2   inst_0
(
      .a      ( a[0] ),
      .b      ( a[1] )
      .c      ( tmp_with_error )
);

assign  c = tmp & a[2];

endmodule
```

Присваивания

```
assign  c2 = c1 & a3;
assign  c1 = a1 & a2;

always_comb begin

        c3 = a1 & a2;
        c4 = c3 & a3
end

always_ff @(posedge clk) begin

        c5 <= #1 a1 & a2
        c6 <= #1 c5 & a3

        c7 <= #1 a1 & a2 & a3

end
```

Если поменять порядок строк, то результат измениться

Особенности:

#1 - это задержка на одну условную единицу

<= НЕ блокирующее присваивание

= Блокирующее присваивание

Значения c2 и c4 совпадают

Значение c7 отстаёт на такт от c2

Значение c6 НЕ совпадает с c2 и c7

Использование составных сигналов

```
logic           a, b, c, d;
logic [3:0]     gr_0, gr_1;

 assign gr_0 = { a, b, c, d };

 assign { a, b, c, d } = gr_1;
```

2023

Рекомендация

2023

Лабораторная работа 1

- Расположение

  ~/projects/basics-graphics-music/01_and_or_not_xor_de_morgan
- Подготовка
  - открыть терминал
  - перейти в 01_and_or_not_xor_de_morgan/scripts
  - выполнить ./prepare_public_package.bash
- Выполнение
  - Запустить Visual Studio Code, выбрать каталог
  -
  - Определить значения при котором загораются светодиоды
  - Определить значения которые возвращают кнопку
  - Выполнить упражнения