

Министерство цифрового развития, связи и массовых коммуникаций
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Сибирский государственный университет телекоммуникаций и
информатики»
(СибГУТИ)
Кафедра инфокоммуникационных систем и сетей

ЛАБОРАТОРНАЯ РАБОТА 1

по дисциплине «Прототипирование телекоммуникационных систем»

«Разработка модуля комбинационной логики»

Мелентьев О.Г.

Методические указания
к лабораторной работе

Новосибирск, 2025

Цель работы:

Изучение основных этапов проектирования на примере простейшей комбинационной схемы, включая сборку и компиляцию.

Задание:

- 1) Ознакомиться с основными понятиями языка описания аппаратуры System Verilog.
- 2) Составить таблицу истинности для заданных логических выражений:
$$Y0 = \bar{A}; \quad Y1 = A \vee B; \quad Y2 = A \cdot B$$
- 3) Создать схему комбинационной логики в System Verilog.
- 4) Провести симуляцию для проверки корректности созданной схемы при выполнении заданных логических выражений.

Отчет должен содержать:

- цель работы;
- исходную схему комбинационной логики;
- код программы в виде скриншота;
- эпюры, полученные вследствие симуляции;
- сделанные выводы о полученных результатах.

Методические указания

1. Основные понятия языка описания аппаратуры System Verilog, необходимые для выполнения лабораторной работы.

System Verilog - язык описания цифровых схем, предназначенный как для описания проекта, так и для его моделирования. Достаточно большое число операторов языка System Verilog не участвуют непосредственно в синтезе, а предназначены лишь для тестирования создаваемых цифровых схем.

Существуют два базовых типа данных (источников сигнала), используемых в языке: *nets* (сети) и *variables* (переменные):

- *Сетевой тип данных* представляет собой физическое соединение между структурными объектами. Они переносят информацию, не производя над ней никаких вычислений, их назначение нельзя поменять во время работы цифрового устройства.

Наиболее распространенным типом сетевых данных является *wire* (проводник). Значение *wire* – это функция того, что присоединено к нему. Проводники, передающие несколько битов информации, называются “шина”. Количество проводников в шине определяется любыми двумя целыми числами, разделенными двоеточием внутри квадратных скобок.

- *Переменный тип данных* обычно представляет собой часть памяти. Он хранит присвоенное ему значение до следующего присвоения. Наиболее

распространенными типами переменных данных являются *reg* (переменная, которая может сохранять свое значение), *parameter* (константа), *integer* (целое число), *time* (время).

Структура проекта (программы):

- Основной единицей языка System Verilog является модуль, который начинается ключевым словом *module* и заканчивается словом *endmodule*. За ключевым словом *module* следует название проекта, за которым в круглых скобках следует список портов модуля: *input* (входные) и *output* (выходные).

Порты бывают трех типов *input* – входы, *output* – выходы, *inout* – двунаправленные. Входы и двунаправленные порты должны иметь тип *wire*, а выходы могут быть как *wire*, так и *reg*.

- Если для описания проекта требуются дополнительные переменные, они должны быть объявлены перед их первым использованием.

Побитовые операторы	
используются для выполнения операций над битами в двоичном представлении чисел	
&	побитовое И
	побитовое ИЛИ
^	побитовое исключающее ИЛИ
~	побитовое отрицание
<<	побитовый сдвиг влево
>>	побитовый сдвиг вправо

- В языке System Verilog нет привычных операторов присваивания, как в языках программирования. *Присваивание значений основным типам данных* (*wire* и *reg*) имеет различную природу. Присваивание значений переменным *reg* подобно оператору присваивания в языках программирования. Однако *wire* предназначены для соединения элементов проекта и ассоциируются с электрическими цепями, соединяющими выводы элементов.
- Конструкция *assign*:** Переменная, которая определяется с помощью конструкции *assign*, должна быть объявлена *wire*. Значение *wire* вычисляется каждый раз заново, когда меняется хотя бы один из операндов, входящих в правую часть конструкции. В качестве операндов

могут применяться векторные величины. Конструкция *assign* относится к непрерывному назначению, с ней может применяться только одно выражение.

Конструкции *assign* обеспечивает непрерывное присваивание и имеет следующий формат:

assign <имя переменной> = <вычисления, операторы Verilog>;

Например:

assign result = op1 + op2;

assign y = ~data[1:0];

Все операторы непрерывного назначения *assign* выполняются параллельно, *assign* применяется для проектирования комбинационных частей устройства.

2. Выполнение лабораторной работы

1) Ввести программу теста, представленную на рисунке 1.

```
include "easy_logick.sv"
module test_b;

    reg clk;
    reg [2:0] counter;

    wire [1:0] key;
    wire [2:0] out;
    initial begin
        clk = 0;
        counter = 0;
    end

    always
        #10 clk = ~clk;

    always @(posedge clk) begin
        counter=counter+1;
    end

    assign key[0] = (counter == 3)||(counter == 7);
    assign key[1] = (counter == 5)||(counter == 7);

    easy_logick easy_logick (
        .key(key),
        .led(out)
    );
endmodule
```

Рисунок 1 - Программа теста (тестбенч)

Программа теста (рис.1) создана для выполнения данной лабораторной работы в дистанционном формате . Файлы должны иметь расширение (.sv).

2) Создать комбинационную схему (рис.2)

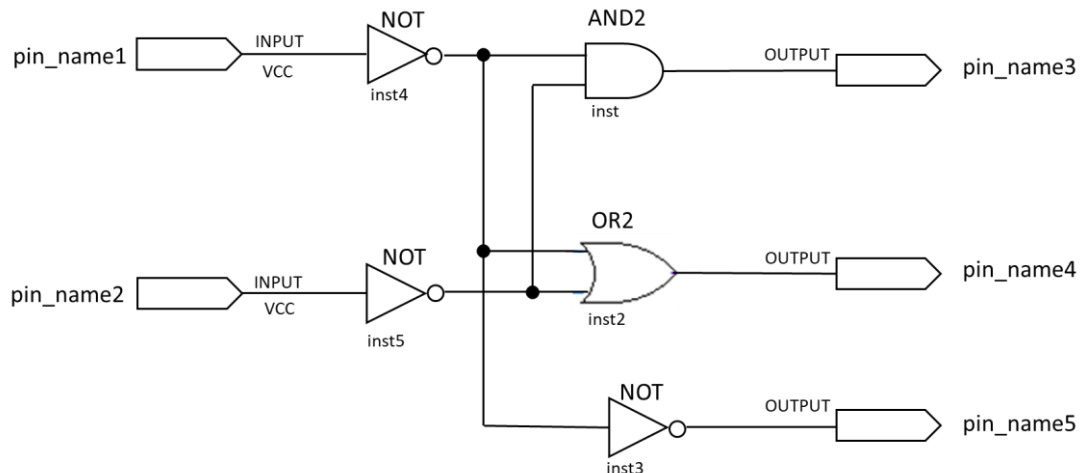


Рисунок 2 – Схема комбинационной логики

3) Создать верхнюю часть модуля. Файл обязательно назвать easy_logick с расширением .sv (рис. 3)

```
module easy_logick
(
    input [1:0]key,
    output [2:0]led
);
// to do
wire a = key [0];
wire b = key [1];

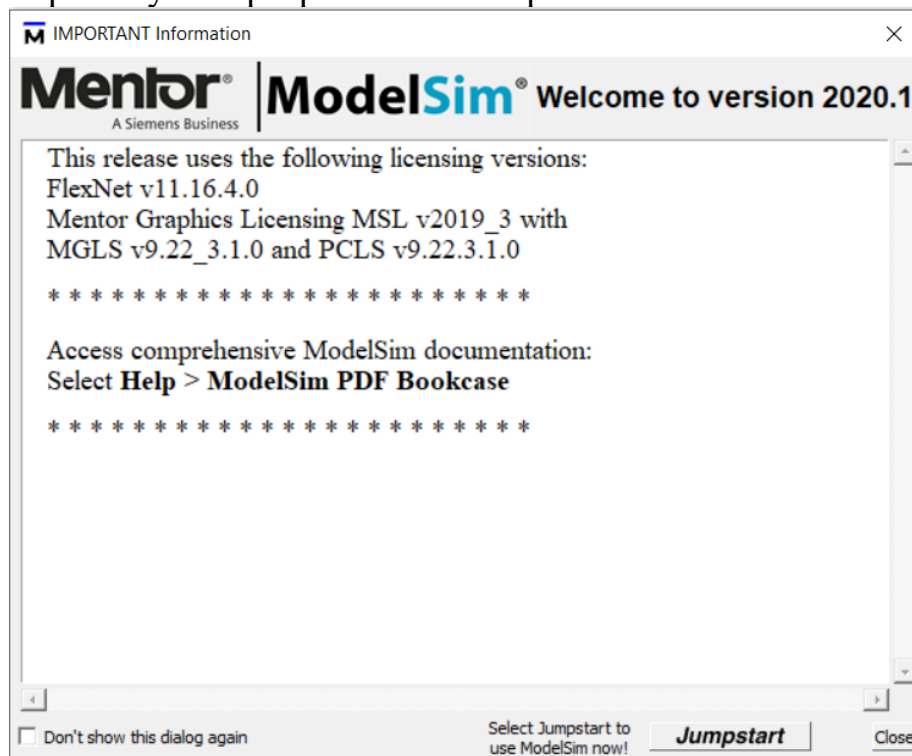
endmodule
```

Рисунок 3 - заведомо выданная часть модуля

Изобразите комбинационную схему на рис.2 в области, выделенной красным цветом на рисунке 3.

5) Работа с ModelSim.

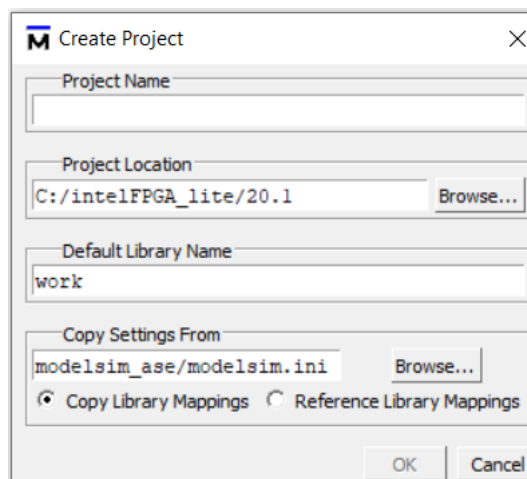
- При запуске программы вас встретит окно



- Нажимаем jumpstart

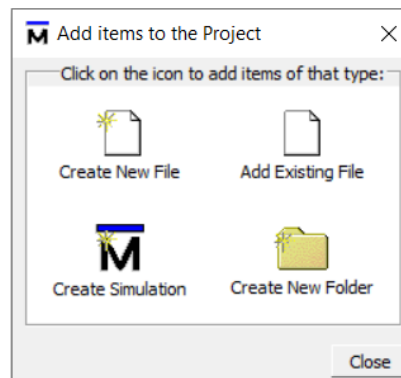


- Выбираем создать проект, причем Project name может быть любым

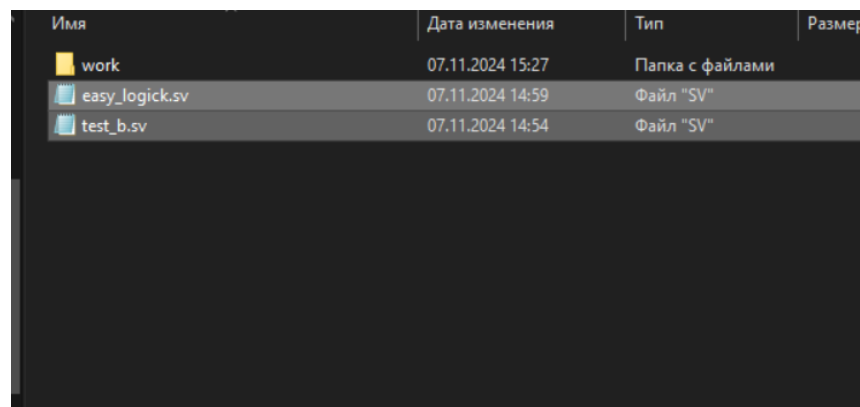


Project location мы ставим папку в которой у нас лежать ранее созданные 2 файла. Остальное без изменений

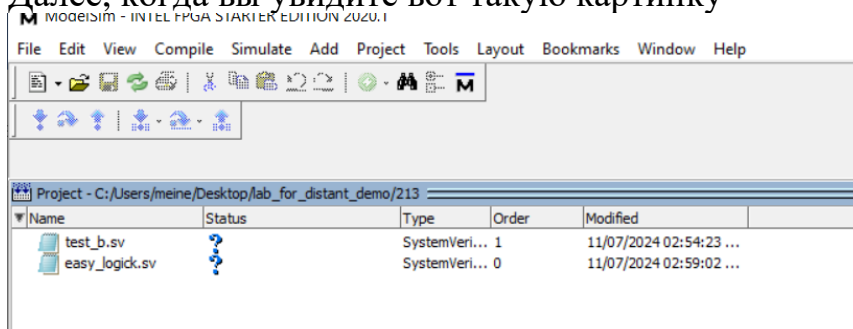
- Выбираем Add existing file



- Зажав кнопку Ctrl выбираем нужные нам 2 файла

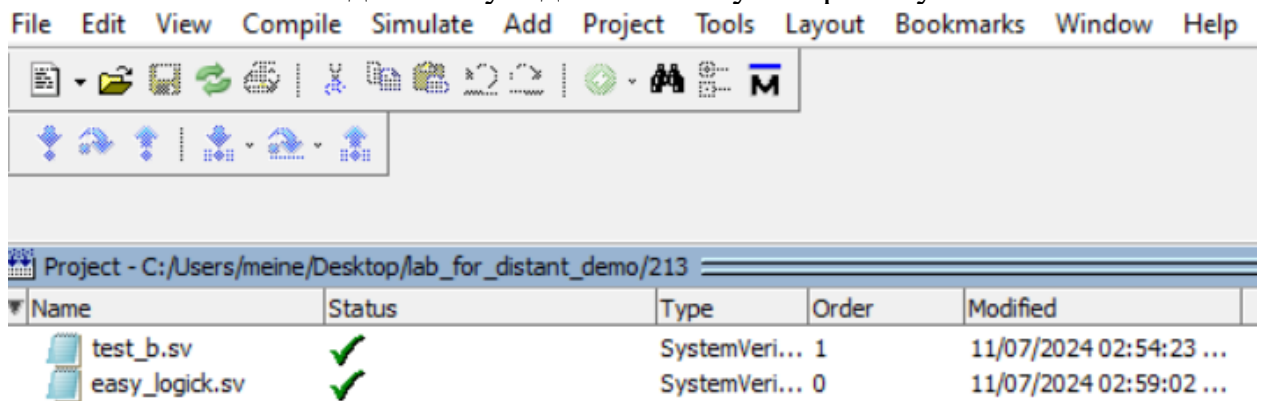


- Далее, когда вы увидите вот такую картинку



Compile=>compile all

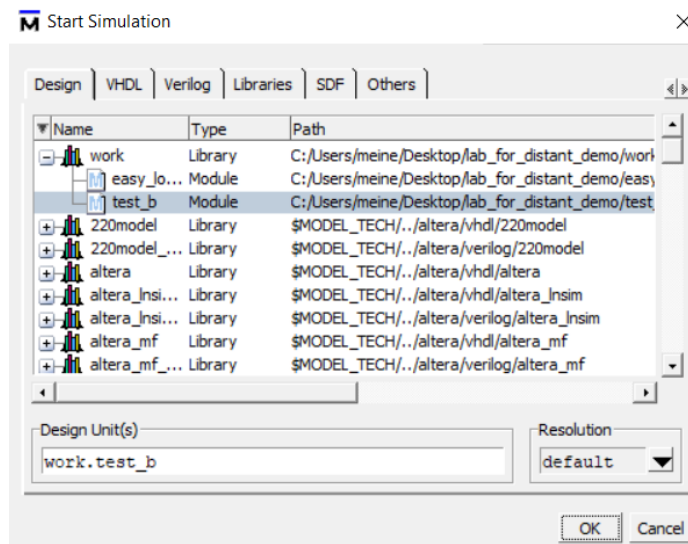
- После чего Вы должны увидеть вот такую картинку



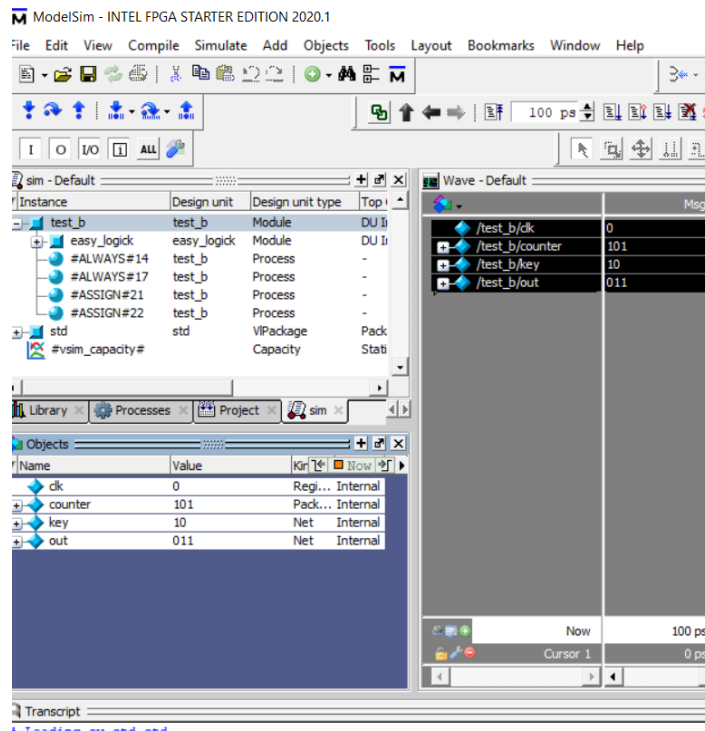
В случае, если вы получили красный крестик, а не зеленую галочку, значит в том, что вы написали есть ошибка и вам стоит перепроверить всё.

А в случае, если полученная картина совпадает, вы должны выполнить следующие действия:

- Simulate => start simulation

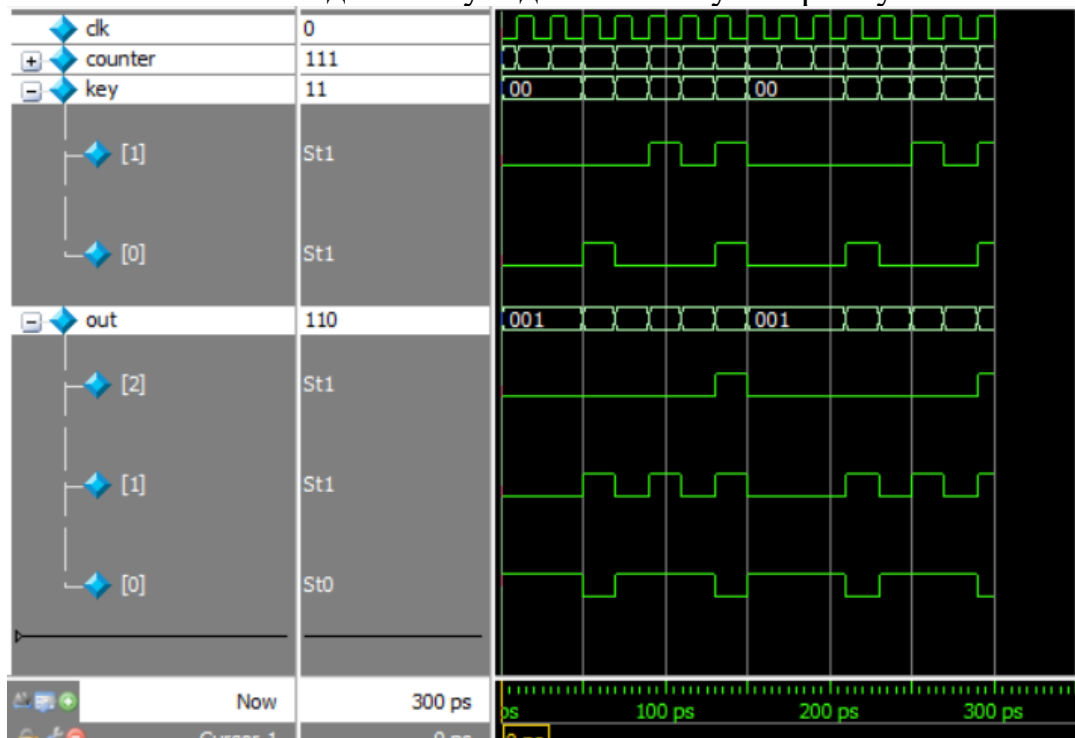


- Ищем папку work и файл тестирования => ok



- Вам необходимо выбрать objects и начать симуляцию в simulation => run => run 100, перед этим изменить время с 100 до 300.

Вы должны увидеть вот такую картину



- В случае, если получены другие эюры, вам следует провести работу над ошибками.

Контрольные вопросы:

1. Структура модуля - описать методы построения.
2. Как обозначаются типы данных в языке system Verilog?
3. Побитовые операторы - описать и привести в пример 3 шт.