Лабораторная работа №1

Создание проектов в среде Quartus. Простейшие логические схемы. Программная симуляция логических схем

1. Цель работы

Целью работы является:

- Изучение среды Quartus
- Изучение основных понятий языка Verilog
- Изучение среды моделирования ModelSim-Altera
- Создание простого проекта и его моделирование

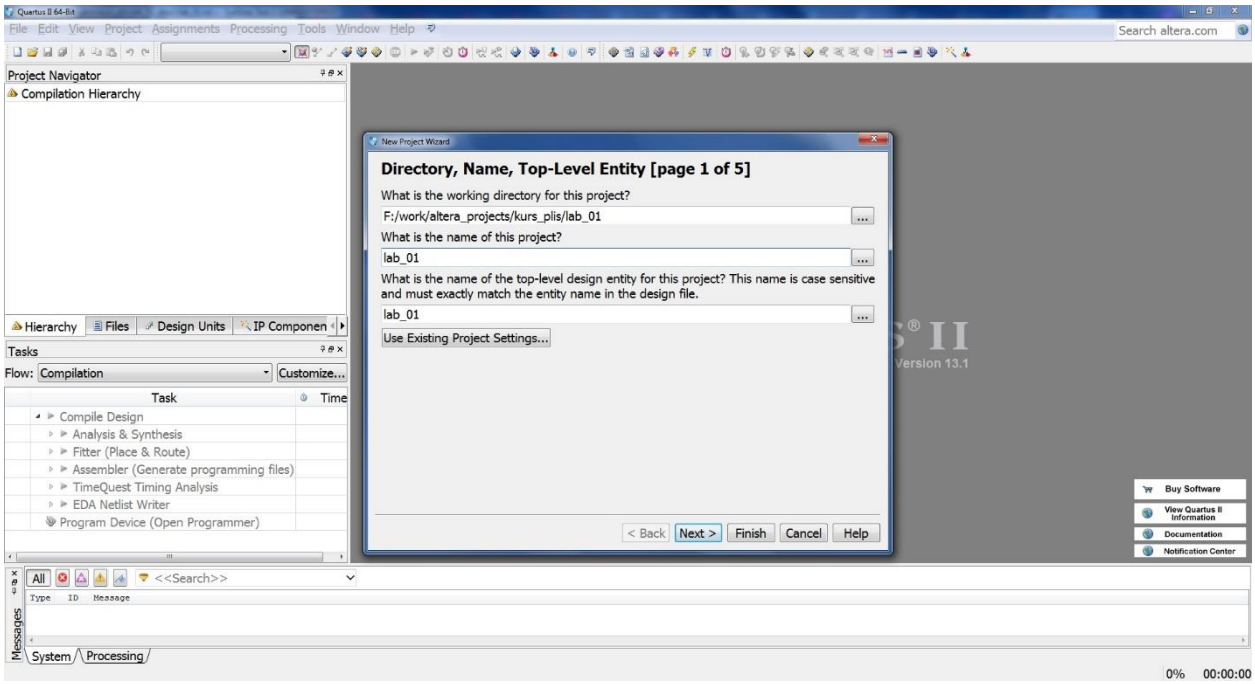
2. Создание проекта в среде Quartus II 13.1

Описание работы:

Для создания проекта в среде Quartus необходимо выполнить следующие шаги:

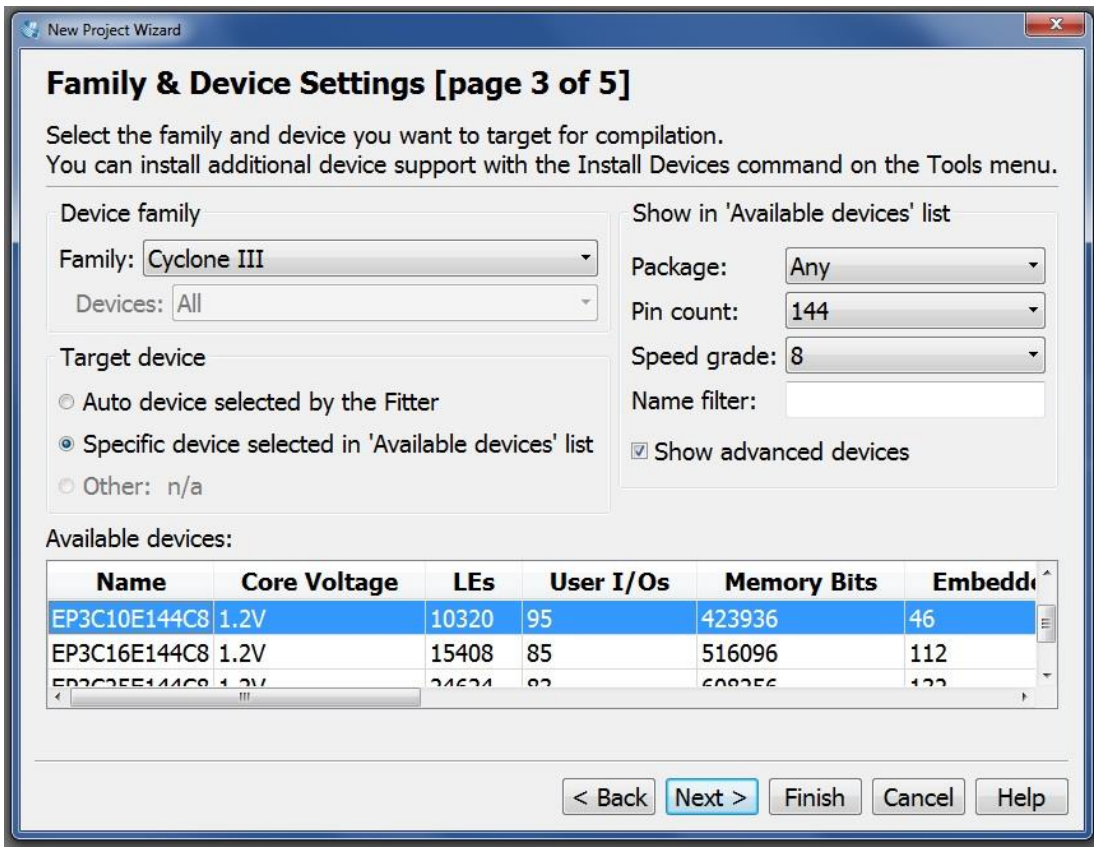
1. Запустите среду разработки Quartus II 13.1.

2. Откройте мастер создания новых проектов (File / New Project Wizard...)



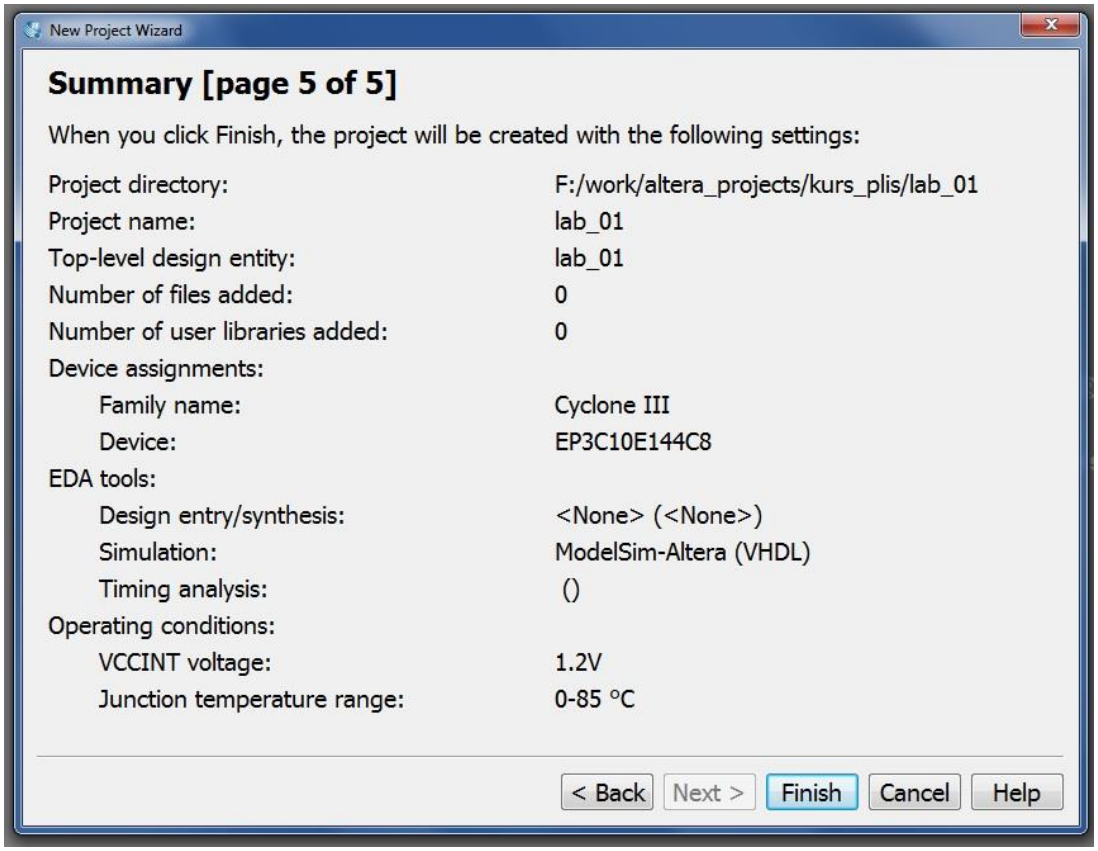
3. В соответствующих полях необходимо выбрать рабочую папку для нового проекта, имя проекта и имя главного модуля. Нажмите кнопку Next. Следующую вкладку можно пропустить. Нажмите кнопку Next еще раз.

4. Далее нужно выбрать микросхему ПЛИС с которой собираетесь работать. В рамках данного курса вы будете работать с платами, на которых установлена микросхема Cyclon III EP3C10E144C8. Выберите её в списке в нижней части окна. Чтобы ограничить это список можно выбрать параметры этой микросхемы в выпадающих списках в верхней части окна. Нажмите кнопку Next.

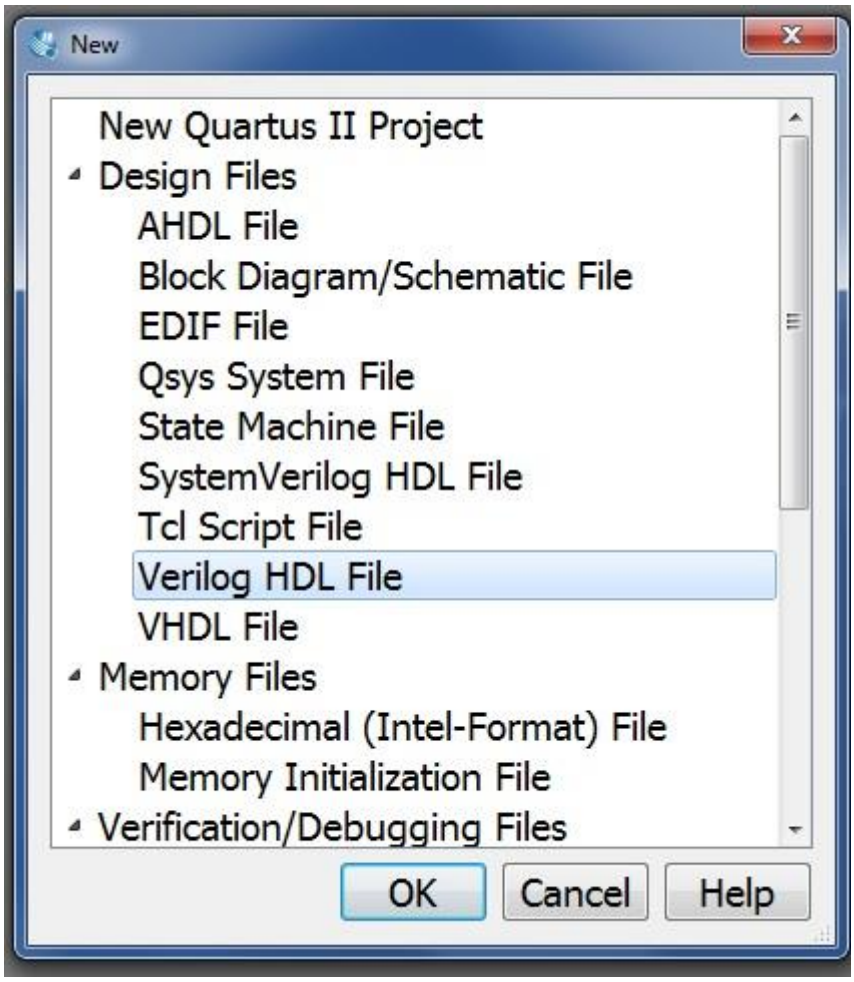


5. Следующую вкладку в текущей работе можно пропустить. Нажмите кнопку Next.

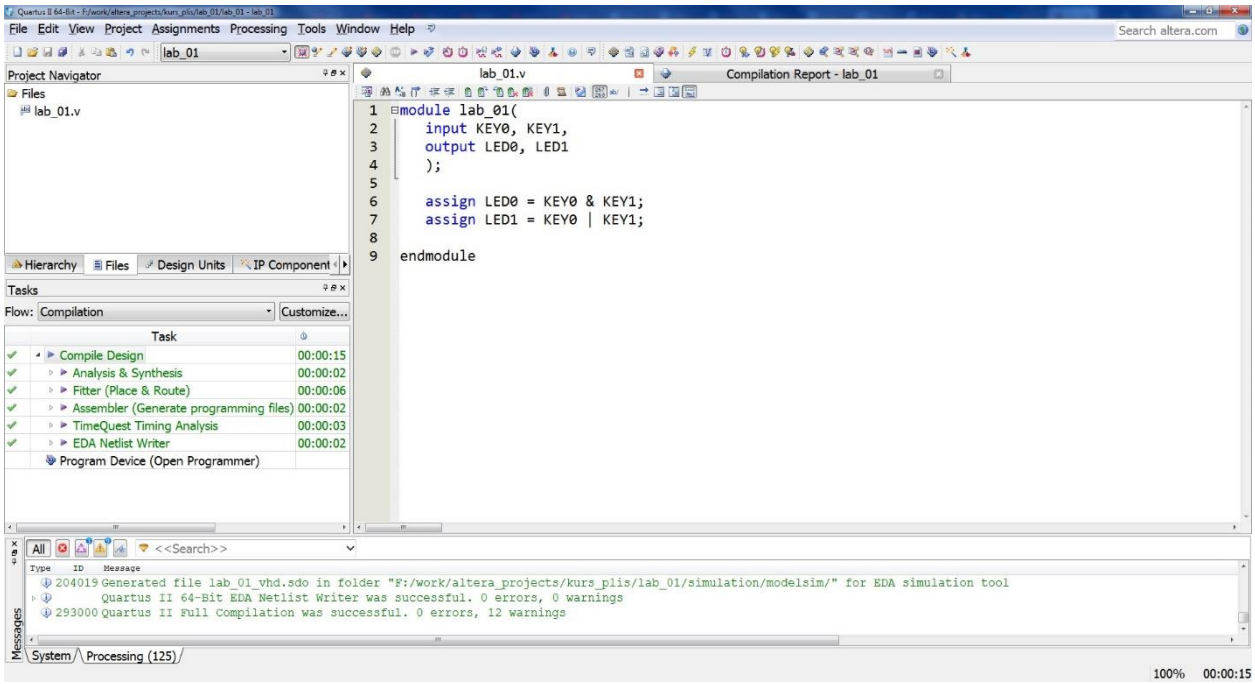
6. На последней вкладке приведены выбранные на предыдущих этапах параметры. Если всё правильно нажмите кнопку Finish.



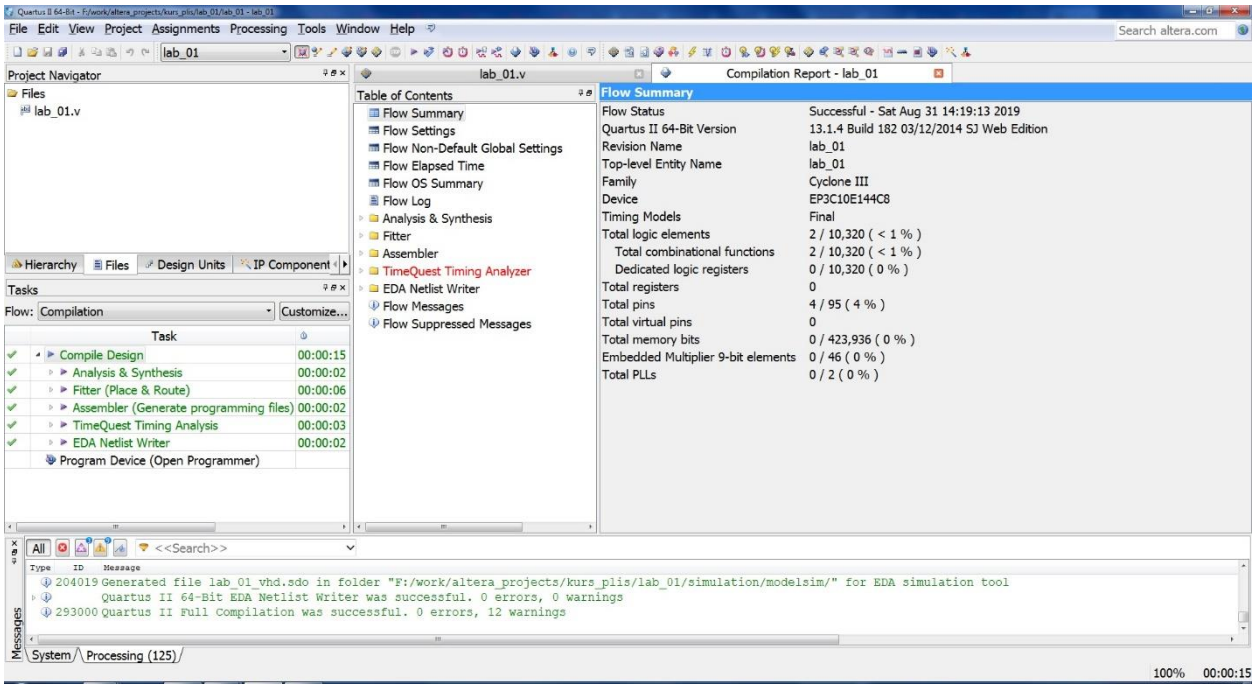
7. Создайте новый Verilog-файл. То есть выберите пункт меню File / New... и в появившемся окне выберите Verilog HDL File. Нажмите OK.



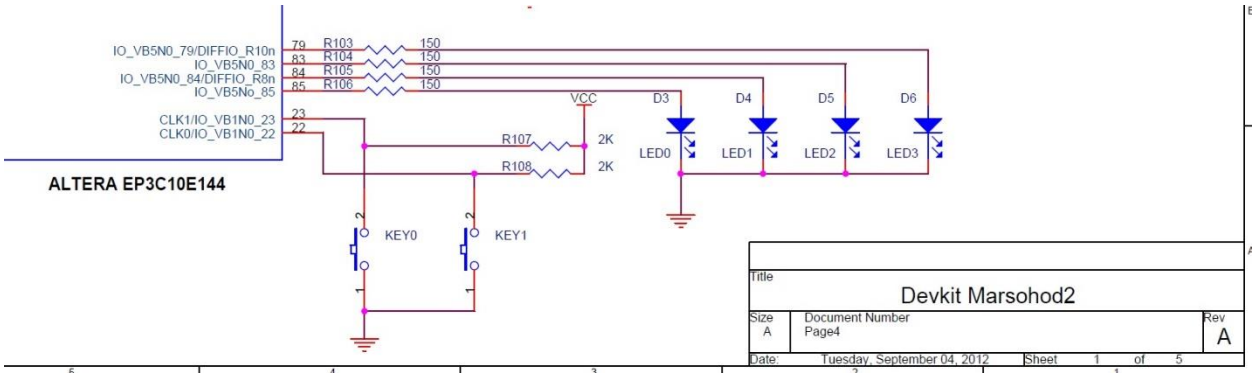
8. Наберите текст на языке Verilog, описывающий модуль, согласно вашему варианту. Сохраните его в рабочую папку проекта. Имя модуля должно совпадать с именем главного модуля, указанного при создании проекта.



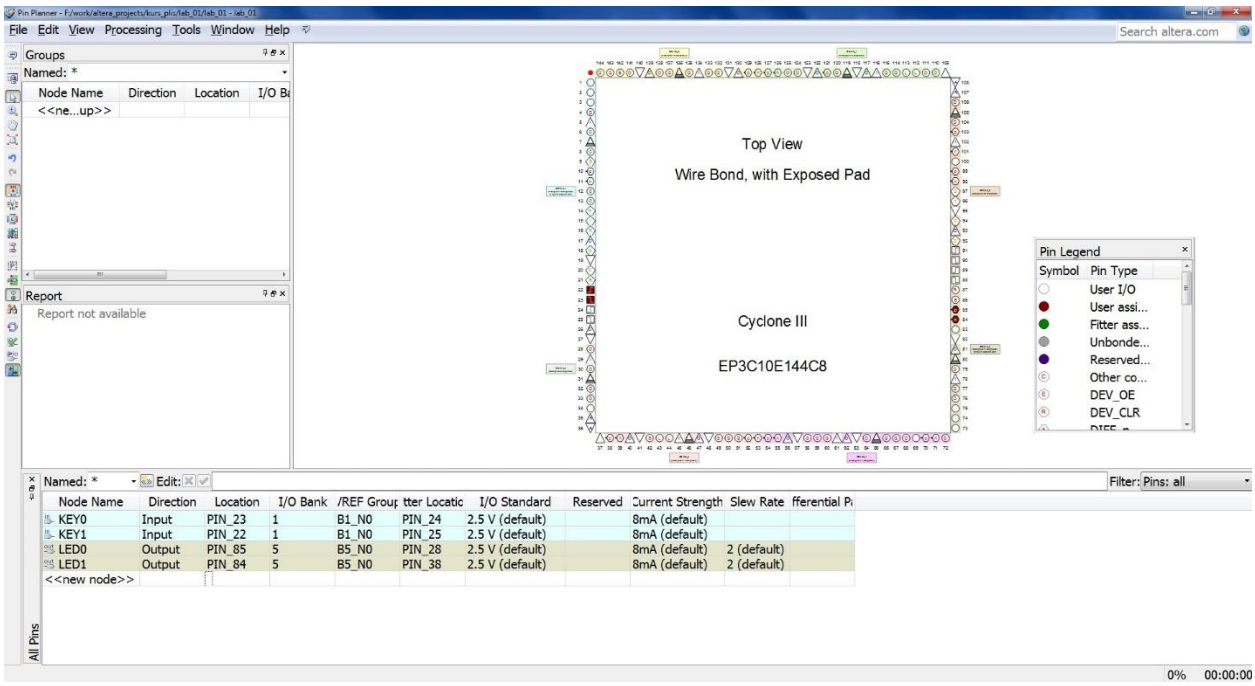
9. Убедитесь, что модуль компилируется выбрав пункт меню Processing / Start Compilation. Внизу должно залогироваться успешное завершение компиляции.



10. Теперь надо сделать так, чтобы логические выводы модуля соответствовали определённым ножкам микросхемы. Для этого изучите схему макетной платы и определите, к каким физическим ножкам подключены кнопки и светодиоды.

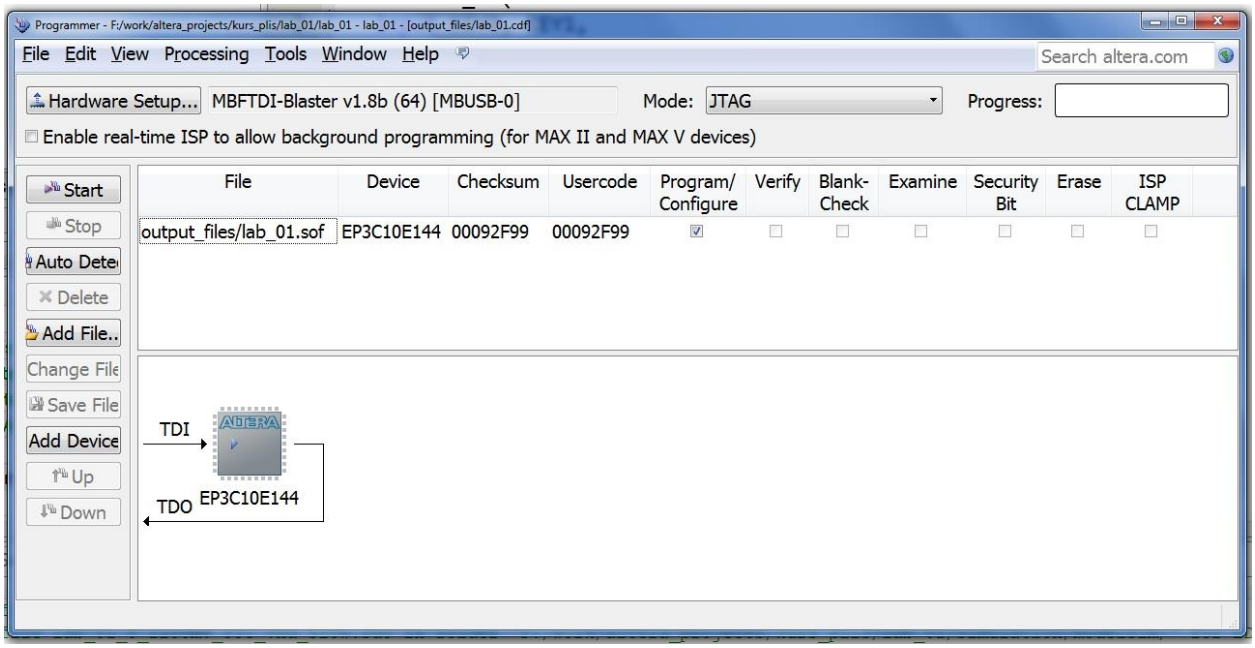


11. С помощью пункта меню Assignments / Pin Planner запустите интерфейс, который позволяет назначить соответствие между ножками и выводами модуля. Для каждого логического вывода заполните наименование ножки в столбце Location. Убедитесь в правильности направлений в столбце Direction. Закройте интерфейс.



12. Перекомпилируйте проект с помощью пункта меню Processing / Start Compilation.

13. Присоедините плату с ПЛИС к компьютеру. С помощью пункта меню Tools / Programmer запустите программатор. Убедитесь, что в поле Hardware Setup... выбрано MBFTDI-Blaster v1.8b (64) [MBUSB-0]. Нажмите кнопку Start. После этого в поле Progress должна появиться надпись на зеленом фоне 100% (Successful). На плате должны загореться светодиоды.



14. Убедитесь, что светодиоды правильно реагируют на нажатия кнопок. Учтите, что согласно схеме подключения, нажатая кнопка соответствует нулевому сигналу на соответствующем входе, а отпущенная - единице.

3. Описание конструкций языка Verilog

Рисование схем в графическом редакторе неудобно и отнимает много времени. И, зачастую, такие схемы проблематичны в отладке. Поэтому для разработки устройств на базе FPGA применяются языки описания аппаратуры HDL (Hardware Description Language). В рамках данного курса мы познакомимся с языком Verilog (и его модификацией – SystemVerilog).

Ниже приведены основные конструкции языка.

Для объявления переменной типа провод используется ключевое слово wire:

```
wire a;
```

Для того, чтобы соединить провода применяется непрерывное присвоение assign (другие типы присвоений будут рассмотрены позднее):

```
wire a,b;
assign a=b;
```

При описании схем используется разделение на модули. Каждый модуль имеет входы и выходы типа wire:

```
module module_AND (input a, input b, output f);
assign f=a&b;
endmodule
```

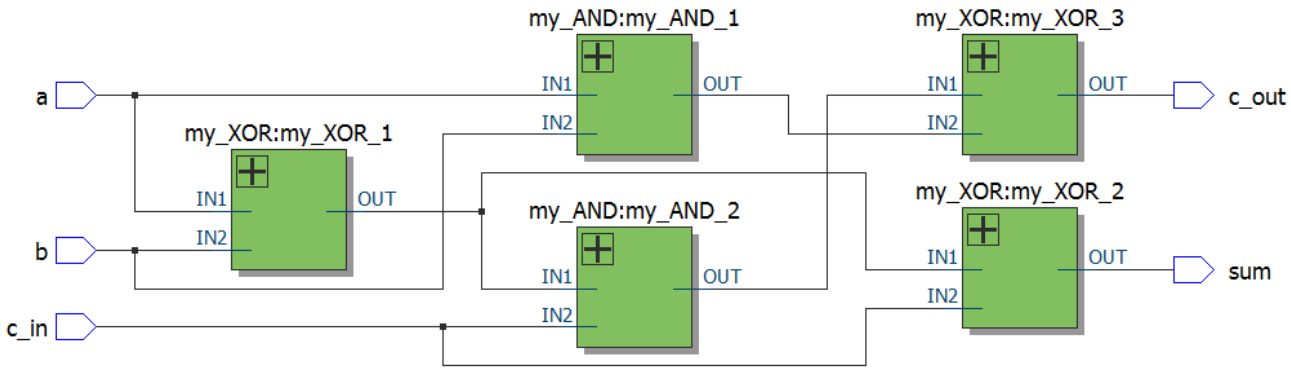
Приведенный выше модуль реализует логическую функцию И. Модули можно соединять в иерархическую структуру. Пример такого соединения:

```
module module_AND (
input IN1,
input IN2,
output OUT
);

assign OUT = IN1 & IN2;

endmodule

module module_XOR(
input IN1,
input IN2,
output OUT
);

assign OUT = IN1 + IN2;

endmodule

module module_sum (
input a,
input b,
input c_in,
output sum,
output c_out
);

wire s1, s2,s3;

module_XOR my_XOR_1 (.IN1 (a), .IN2 (b), .OUT (s1) );
module_XOR my_XOR_2 (.IN1 (s1), .IN2 (c_in), .OUT (sum) );
module_AND my_AND_1 (.IN1 (a), .IN2 (b), .OUT (s3) );
module_AND my_AND_2 (.IN1 (s1), .IN2 (c_in), .OUT (s2) );
module_XOR my_XOR_3 (.IN1 (s2), .IN2 (s3), .OUT (c_out) );

endmodule
```

Модуль module_sum описывает одноразрядный сумматор на основе логических элементов И (AND) и ИСКЛЮЧАЮЩЕЕ-ИЛИ (XOR). Для этого внутри основного модуля (top level) создаются несколько экземпляров модулей module_XOR и module_AND.

Экземпляр модуля описывается следующим образом: сначала пишется название модуля, экземпляр которого нужен, затем указывается имя данного экземпляра и далее описываются подключения сигналов: точка, имя сигнала модуля и в скобках имя проводника, который к нему подключен.

Посмотреть получившуюся схему можно через меню Tools/Netlist Viewrs/RTL Viewer.



С помощью HDL Verilog цифровая система может быть описана на структурном и поведенческом уровнях. Приведённый выше код описывает схему на структурном уровне. Структурное описание представляет собой описание системы в виде совокупности компонентов и связей между ними. Поведенческое описание представляет собой описание системы при помощи задания зависимости вход-выход.

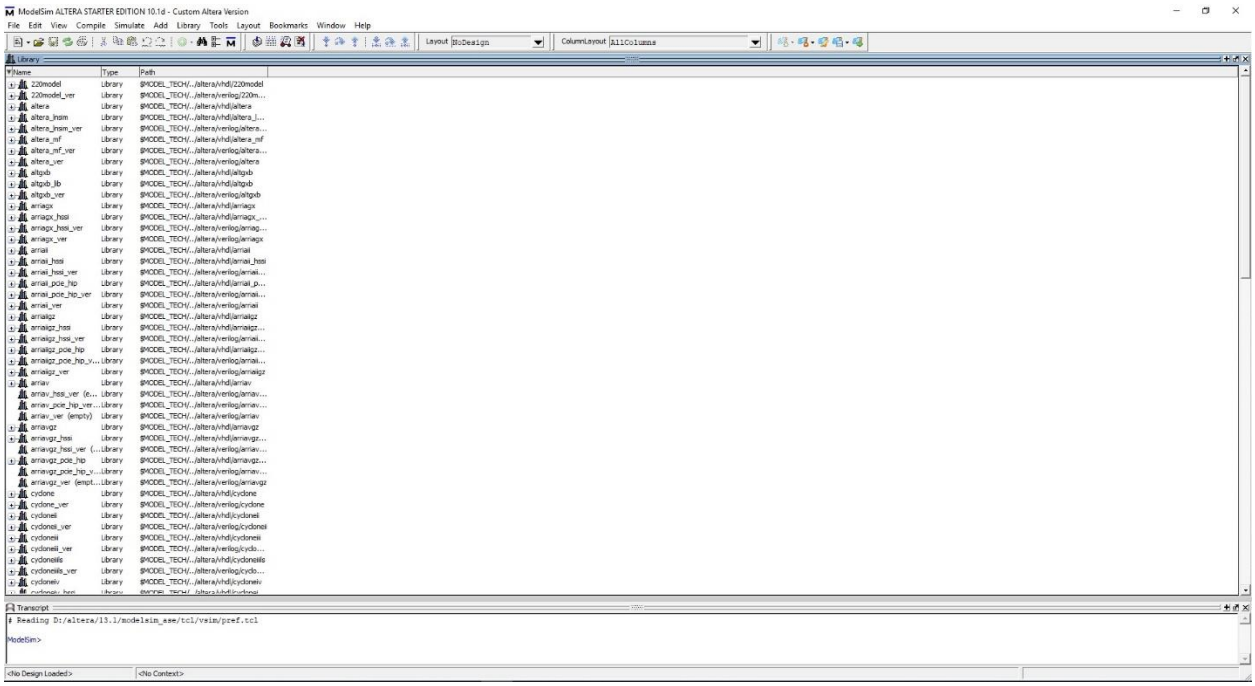
Ниже приведён код описания схемы сумматора на поведенческом уровне.

```
module module_sum (
input a,
input b,
input c_in,
output sum,
output c_out
);

assign sum = ((a^b)& c_in);
assign c_out = ((a^b) ^ c_in) ^ (a&b);

endmodule
```

4. Программная симуляция логических схем

Задание:

1. Запустите среду моделирования ModelSim-Altera 10.1d



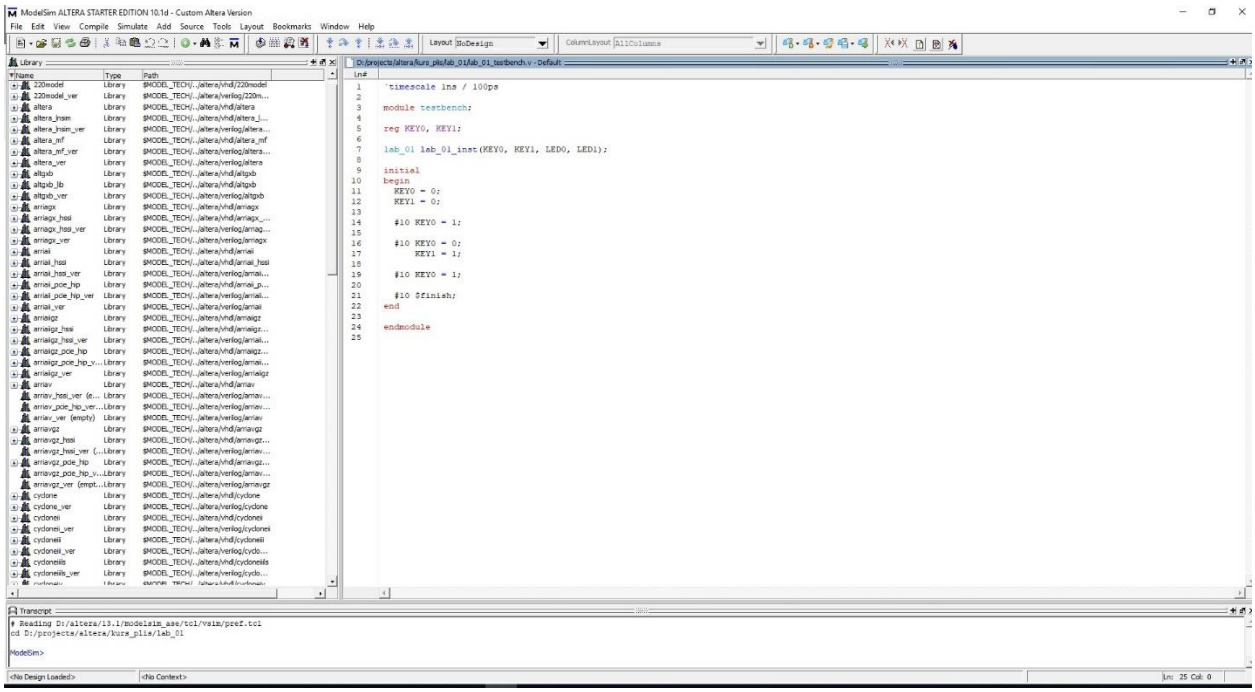
2. С помощью пункта меню File / Change Directory… выберите рабочую директорию проекта, созданного вами в процессе лабораторной работы №1.

3. В этой директории уже находится модуль, которых вам необходимо проверить. Теперь надо создать модуль, который будет осуществлять проверку. Для этого с помощью пункта меню
File / New / Source / Verilog создайте новый Verilog-файл. В нём напишите код для моделирования вашей схемы.

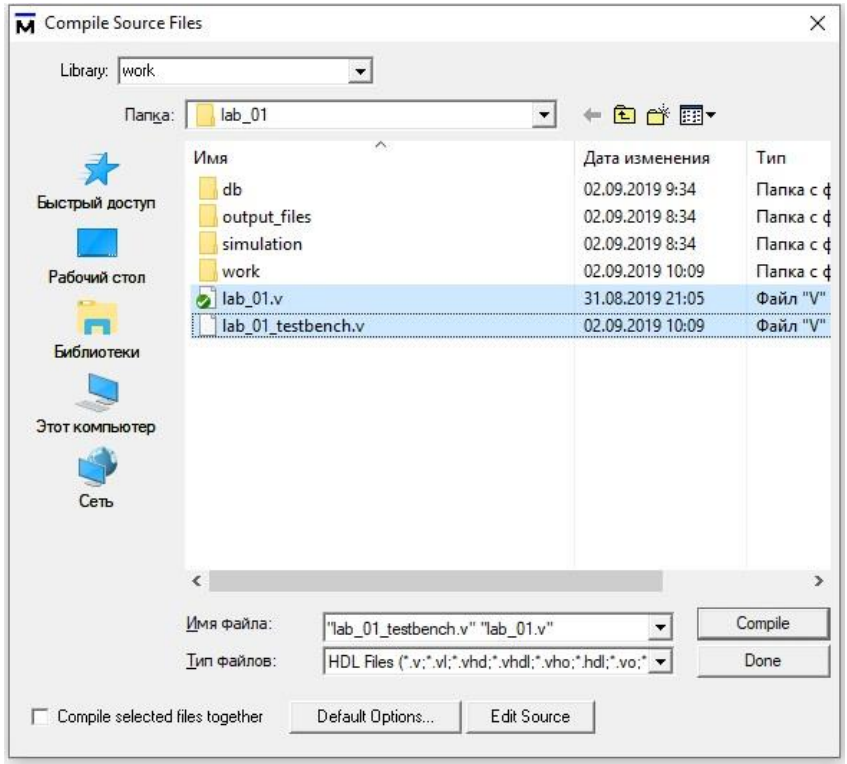
Пример такого файла приведён ниже.

```
 `timescale 1ns / 100ps
module testbench;
reg KEY0, KEY1;
lab_01 lab_01_inst(KEY0, KEY1, LED0, LED1);
initial
begin
  KEY0 = 0;
  KEY1 = 0;
  #10 KEY0 = 1;
  #10 KEY0 = 0;
      KEY1 = 1;
  #10 KEY0 = 1;
  #10 $finish;
end
endmodule
```

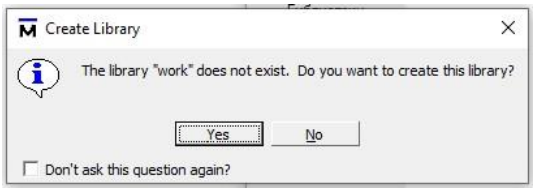
Сохраните его в рабочую директорию.



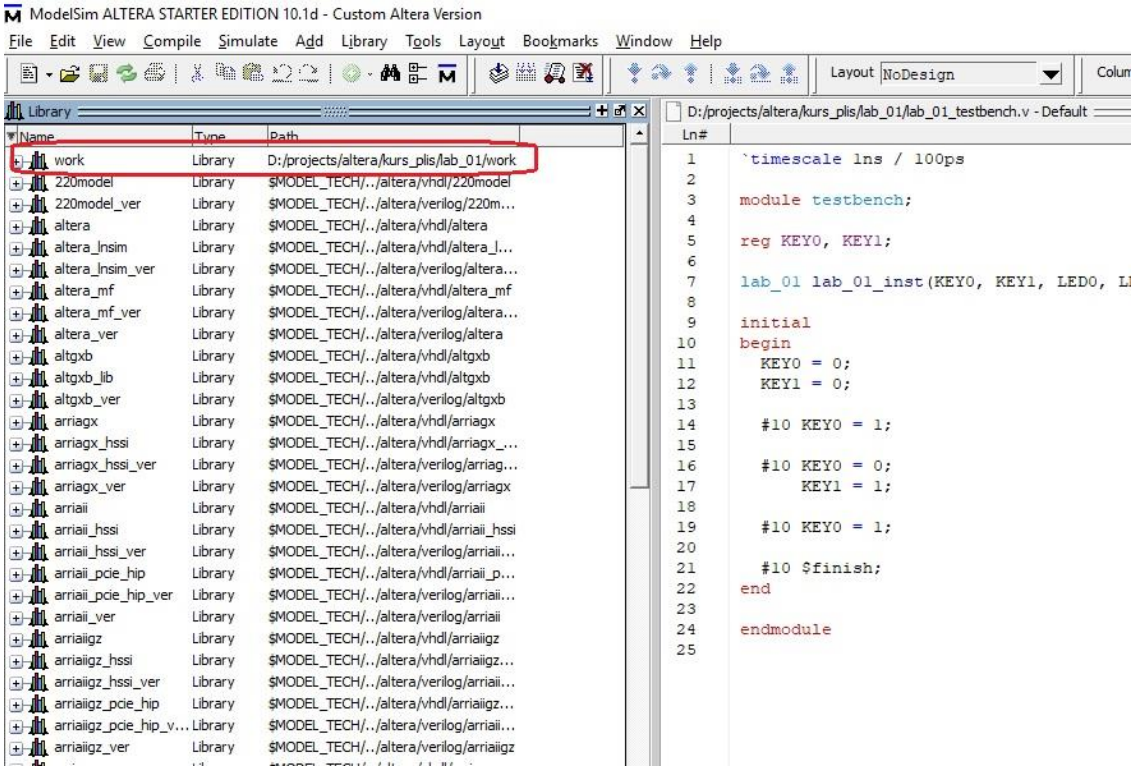
4. Скомпилируйте файл тестируемого модуля вместе с тестирующим модулем. Для этого используйте пункт меню Compile / Compile… Чтобы файлы скомпилировались вместе надо выбрать оба файла и нажать Compile.



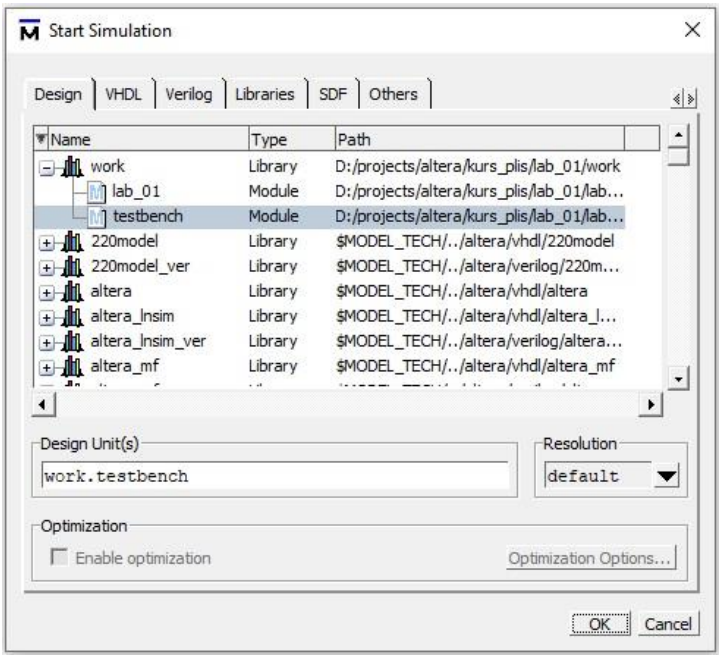
5. Если вы компилируете в первый раз, то вам будет предложено создать библиотеку work в рабочей директории. Нажмите Yes.



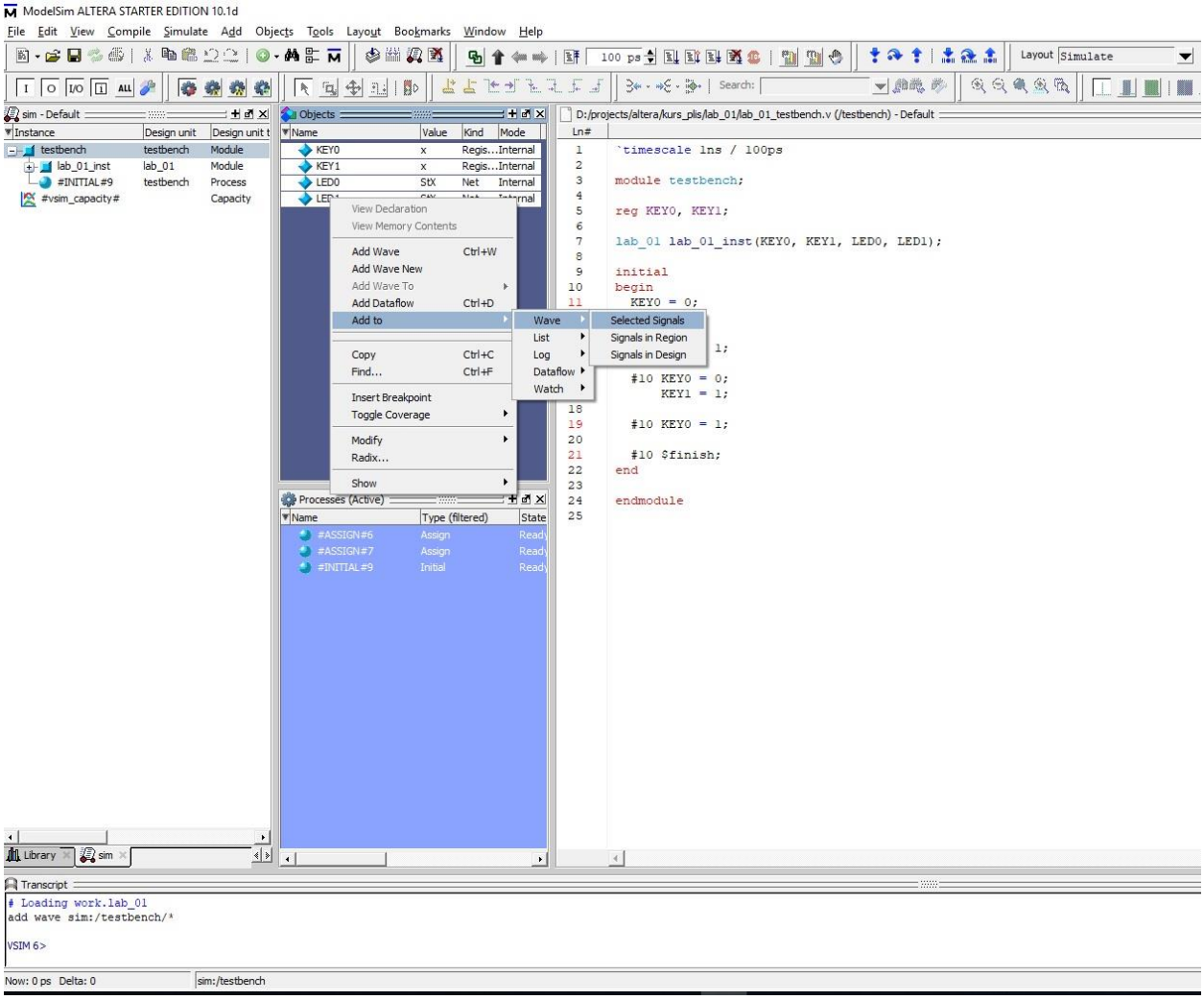
6. Убедитесь в отсутствии ошибок компиляции и в том, то библиотека work появилась в списке библиотек.

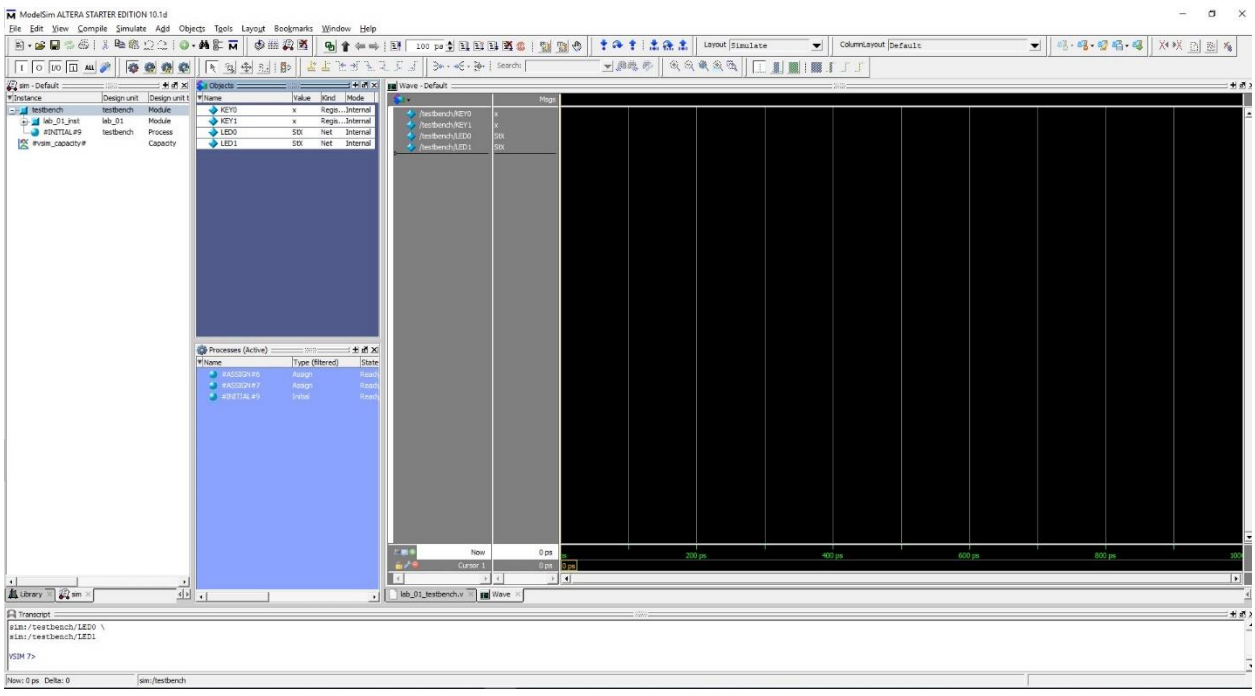


7. Запустите симуляцию выбрав пункт меню Simulate / Start Simulation… В появившемся окне выберите тестирующий модуль. Нажмите OK.

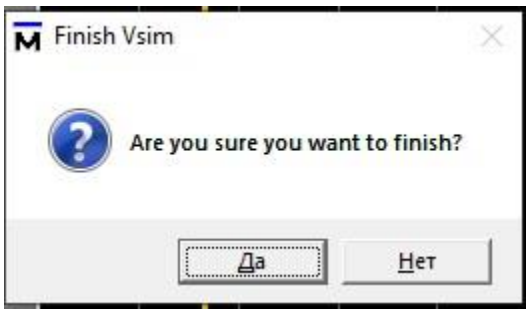


8. В появившемся окне Objects выберите сигналы, которые хотите отслеживать и, с помощью контекстного меню Add to / Wave / Selected Signals, добавьте их в окно Wave.

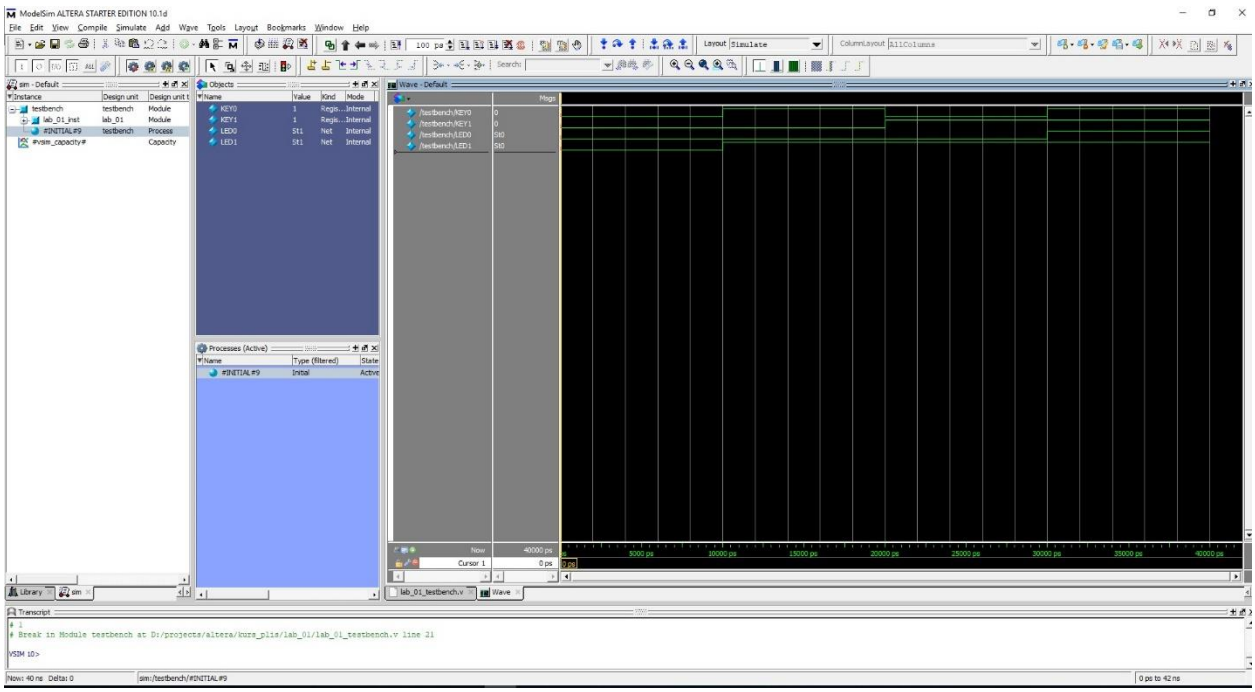




9. Выберите пункт меню Simulate / Run / Run -All. В появившемся окне с предложением закончить работу с ModelSim нажмите Нет.



10. Через некоторое время в окне Wave появятся графики, отображающие поведение выбранных сигналов.



11. Поведением сигналов KEY0 и KEY1 управляет тестирующий модуль. Видно, что они последовательно проходят через все возможные комбинации значений. Сигналы LED0 и LED1 – это выходы тестируемого модуля. Убедитесь, что они работают в соответствии с внутренней логикой тестируемого модуля.

5. Задание

1. Описать на языке Verilog схему в соответствии с вариантом (на структурном уровне).
2. Скомпилировать схему и загрузить в плату.
3. Проверить работоспособность схемы по таблице истинности.
4. Описать схему другим способом (на поведенческом уровне).
5. Скомпилировать и загрузить в плату.
6. Проверить работоспособность схемы по таблице истинности.
7. Провести моделирование полученных схем в ModelSim.

Варианты заданий

1	$F=x+y*z$	11	$F=z+\overline{x}*\overline{y}$	21	$F=\overline{y}+\overline{x}*\overline{z}$
2	$F=x+y*\overline{z}$	12	$F=z+x*\overline{y}$	22	$F=\overline{y}+x*\overline{z}$
3	$F=y+x*z$	13	$F=\overline{x}+y*z$	23	$F=\overline{z}+\overline{x}*\overline{y}$
4	$F=y+\overline{x}*z$	14	$F=\overline{x}+y*\overline{z}$	24	$F=\overline{z}+x*\overline{y}$
5	$F=z+x*y$	15	$F=\overline{y}+x*z$		
6	$F=z+\overline{x}*v$	16	$F=\overline{y}+\overline{x}*z$		
7	$F=x+\overline{y}*\overline{z}$	17	$F=\overline{z}+x*y$		
8	$F=x+y*\overline{z}$	18	$F=\overline{z}+\overline{x}*y$		
9	$F=y+\overline{x}*\overline{z}$	19	$F=\overline{x}+\overline{y}*\overline{z}$		
10	$F=v+x*\overline{z}$	20	$F=\overline{x}+y*\overline{z}$		