САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ПЕТРА ВЕЛИКОГО

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчет

Пакеты Quartus и ModelSim. Лабораторная работа 3.

Дисциплина

«Технологии проектирования аппаратных средств компьютерных систем»

Выполнил:

Бараев Д.Р.

Группа: 3540901/02001

Преподаватель: А. П. Антонов

Санкт-Петербург

2021

Оглавление

[1 Lab 3\_1 Моделирование и внедрение проекта с ModelSim и Quartus Prime 5](#_Toc68528806)

[1.1 Цели 5](#_Toc68528807)

[1.2 Обзор проекта 5](#_Toc68528808)

[1.3 Создание проекта Quartus Prime. 5](#_Toc68528809)

[1.4 Запустите ModelSim Intel FPGA Starter Edition. 6](#_Toc68528810)

[1.5 Отладка проекта 8](#_Toc68528811)

[1.6 Файл верхнего уровня для отладки 9](#_Toc68528812)

[1.7 Добавление и настройка Signal Tap II 9](#_Toc68528813)

[1.8 Полная компиляция 11](#_Toc68528814)

[1.9 Отладка проекта 13](#_Toc68528815)

[1.10 Добавление модуля верхнего уровня 17](#_Toc68528816)

[1.11 Выводы 17](#_Toc68528817)

[2 Lab 3\_2 Самостоятельная работа 18](#_Toc68528818)

[2.1 Создание проекта 18](#_Toc68528819)

[2.2 Создать описание модулей на языке Verilog 18](#_Toc68528820)

[2.3 Моделирование в пакете ModelSim 23](#_Toc68528821)

[2.4 Тестирование и отладка на плате 24](#_Toc68528822)

[2.5 Программирование и проверка платы 28](#_Toc68528823)

[2.6 Выводы 29](#_Toc68528824)

1. **Lab 3\_1 Моделирование и внедрение проекта с ModelSim и Quartus Prime**

[Рисунок 1.1 - Окно создание проекта 5](#_Toc68528319)

[Рисунок 1.2 - Неправильная работа проекта в окне Wave 5](#_Toc68528320)

[Рисунок 1.3 - Результат перезапуска моделирования 6](#_Toc68528321)

[Рисунок 1.4 - Исправление фйла Lab3\_1.v 6](#_Toc68528322)

[Рисунок 1.5 - Окно Wave после перекомпиляции проекта. 6](#_Toc68528323)

[Рисунок 1.6 - Окно Memory Data и файл ss\_ascii.txt 7](#_Toc68528324)

[Рисунок 1.7 - Создание IP-компонента SP\_unit 7](#_Toc68528325)

[Рисунок 1.8 - Созданные IP-компоненты успешно добавлены в проект 8](#_Toc68528326)

[Рисунок 1.9 - Схема проекта в RTL Viewer 8](#_Toc68528327)

[Рисунок 1.10 - Добавленные узлы для анализа 9](#_Toc68528328)

[Рисунок 1.11 - Результат добавления ss\_udec: width = 7 9](#_Toc68528329)

[Рисунок 1.12 - Результат настройки таблицы 10](#_Toc68528330)

[Рисунок 1.13 - Тактовый сигнал для синхронизации 10](#_Toc68528331)

[Рисунок 1.14 - Добавили файл db\_Lab3\_1.sdc 11](#_Toc68528332)

[Рисунок 1.15 - Полная компиляция проекта 11](#_Toc68528333)

[Рисунок 1.16 - Timing Analyzer 11](#_Toc68528334)

[Рисунок 1.17 - Доступное оборудование - USB-Blaster 12](#_Toc68528335)

[Рисунок 1.18 - Запуск файла Lab3\_1.sof 13](#_Toc68528336)

[Рисунок 1.19 - Окно In-System Source and Probes Editor. Регулировка настроек 13](#_Toc68528337)

[Рисунок 1.20 - Настройка постоянного считывания данных 14](#_Toc68528338)

[Рисунок 1.21 - Проверка сигнала Reset 14](#_Toc68528339)

[Рисунок 1.22 - Временная диаграмма 15](#_Toc68528340)

[Рисунок 1.23 - Предупреждения и ошибки во время полной компиляции 15](#_Toc68528341)

[Рисунок 1.24 - In-System Sources and Probes Editor: Reset=1, Dir=1 15](#_Toc68528342)

[Рисунок 1.25 - SignalTap Logic Analyzer: временная диаграмма 15](#_Toc68528343)

[Рисунок 1.26 - Содержимое файла impl\_Lab3\_1.v 16](#_Toc68528344)

[Рисунок 1.27 - Структура устройства в RTL Viewer 16](#_Toc68528345)

**Lab 3\_2 Самостоятельная работа**

[Рисунок 2. 1 - Программная реализация модуля SYNCin 19](#_Toc68528435)

[Рисунок 2. 2 - Структура модуля SYNCin в RTL Viewer 19](#_Toc68528436)

[Рисунок 2. 3 - Программная реализация модуля CNT\_DIV 20](#_Toc68528437)

[Рисунок 2. 4 - Структура модуля CNT\_DIV в RTL Viewer 20](#_Toc68528438)

[Рисунок 2. 5 - Программная реализация модуля CNT 20](#_Toc68528439)

[Рисунок 2. 6 - Структура модуля CNT в RTL Viewer 21](#_Toc68528440)

[Рисунок 2. 7 - Создание модуля CODER 21](#_Toc68528441)

[Рисунок 2. 8 - Программная реализация модуля SYNCout 22](#_Toc68528442)

[Рисунок 2. 9 - Структура модуля SYNCout в RTL Viewer 22](#_Toc68528443)

[Рисунок 2. 10 - Программная реализация модуля верхнего уровня Lab3\_2 23](#_Toc68528444)

[Рисунок 2. 11 - Структура модуля верхнего уровня Lab3\_2 в RTL Viewer 24](#_Toc68528445)

[Рисунок 2. 12 - Программная реализация тестового модуля tb\_Lab3\_2.v 24](#_Toc68528446)

[Рисунок 2. 13 - Файлы в проекте ModelSim 25](#_Toc68528447)

[Рисунок 2. 14 - Временная диаграмма 25](#_Toc68528448)

[Рисунок 2. 15 - Содержимое памяти ROM 25](#_Toc68528449)

[Рисунок 2. 16 - Содержимое массивов памяти, используемых для перекодировки 7-сегментных кодов в без-знаковые десятичные цифры 25](#_Toc68528450)

[Рисунок 2. 17 – Модуль PLL\_unit 26](#_Toc68528451)

[Рисунок 2. 18 - Создание модуля отладчика на основе Altera In-System Sources and Probes 26](#_Toc68528452)

[Рисунок 2. 19 - Созданные модули 27](#_Toc68528453)

[Рисунок 2. 20 - Содержимое файла db\_Lab3\_2 27](#_Toc68528454)

[Рисунок 2. 21 - Структура модуля db\_Lab3\_2 в RTL Viewer 27](#_Toc68528455)

[Рисунок 2. 22 - Настройки SignalTap 28](#_Toc68528456)

[Рисунок 2. 23 - Окно In-System Sources and Probes Editor 28](#_Toc68528457)

[Рисунок 2. 24 - Содержимое файла impl\_Lab3\_2.v 29](#_Toc68528458)

[Рисунок 2. 25 - Структура модуля impl\_Lab3\_2 в RTL Viewer 29](#_Toc68528459)

[Рисунок 2. 26 - Окно Programmer 29](#_Toc68528460)

[Рисунок 2. 27 - Результат работы платы 30](#_Toc68528461)

# Lab 3\_1 Моделирование и внедрение проекта с ModelSim и Quartus Prime

## Цели

В ходе данной лабораторной работы предстоит ознакомиться со следующим:

1. Как создать проект в Quartus Prime
2. Как смоделировать проект с помощью ModelSim
3. Как отлаживать проект на плате с помощью ISSP и SignalTap II.
4. Как реализовать проект на плате.

## Обзор проекта

Входы для модуля верхнего уровня Lab3\_1:

* CLK - тактовый сигнал. Это вывод FPGA, подключенный к тактовому генератору на плате. Частота 25 МГц.
* RST - сигнал асинхронного сброса. Это вывод FPGA, подключенный к к переключателю на плате.
* DIR - направление счета (1 - счет ВНИЗ; 0 - счет ВВЕРХ). Это вывод FPGA, подключенный к переключателю на плате.

Выходы для модуля верхнего уровня Lab3\_1:

* [6:0] HEX - 7-сегментный код. Это контакты FPGA, подключенные к 7-сегментному дисплею на стадии разработки платы.
* [3:0] DIG - постоянное значение. Это выводы ПЛИС, подключенные к входам включения \ выключения 7-сегментных индикаторе.

Алгоритм реализации проекта:

* Делитель тактовой частоты (div\_cnt) делит входную тактовую частоту и выдает сигнал выполнения (cout).
* Значение деления (div\_par) является параметром.
* Счетчик (cnt\_val)
* двоично-десятичный счетчик
* синхронизируется по CLK
* включается сигналом выполнения (cout), приходящим от тактового делителя.
* значение для подсчета (cnt\_val) находится на входе DIR.
* Кодер
* Кодирует 4-битный двоично-десятичный код, поступающий от счетчика, в 7-сегментный код.
* Устанавливает постоянное значение «1000» для выходов DIG (крайняя левая цифра 7-сегментного индикатора горит, остальные выключены).

## Создание проекта Quartus Prime.

1. Создание проекта в Quartus Prime.

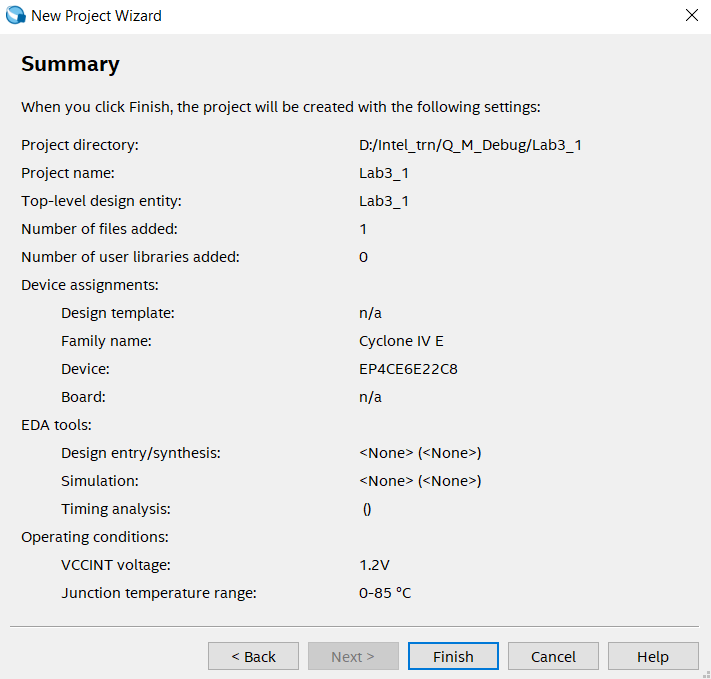


Рисунок 1.1 - Окно создание проекта

## Запустите ModelSim Intel FPGA Starter Edition.

* Щелкните правой кнопкой мыши в окне и выберите *Add to => Wave => Signals in region.*
* Щелкните правой кнопкой мыши сигнал tb\_ss и выберите *Radix => ASCII*.
* Щелкните правой кнопкой мыши сигнал tb\_hex и выберите *Radix => Hexadecimal.*
* Запустите моделирование.
* Обратите внимание на окно *Wave* - вы увидите, что проект работает некорректно.

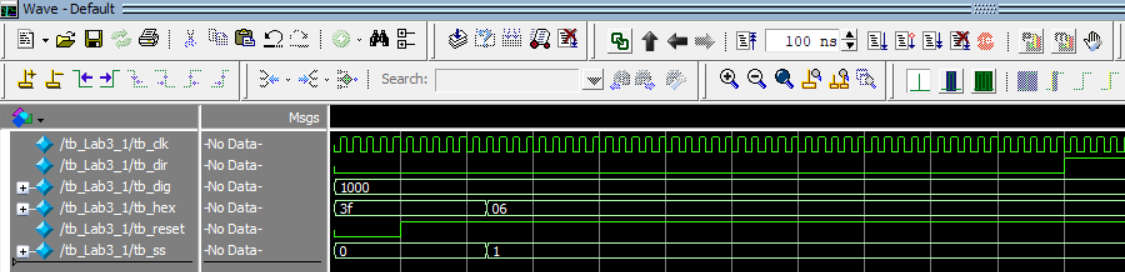


Рисунок 1.2 - Неправильная работа проекта в окне Wave

Вы можете увидеть значение 0 (ASCII), затем значение 1 (ASCII), и тогда кажется, что проект остановился. Это сохраняет значение 1 (ASCII) к концу моделирования.

Чтобы выяснить причину такого поведения, добавили в окно Wave несколько сигналов от модуля Lab3\_1 и перезапустили моделирование:

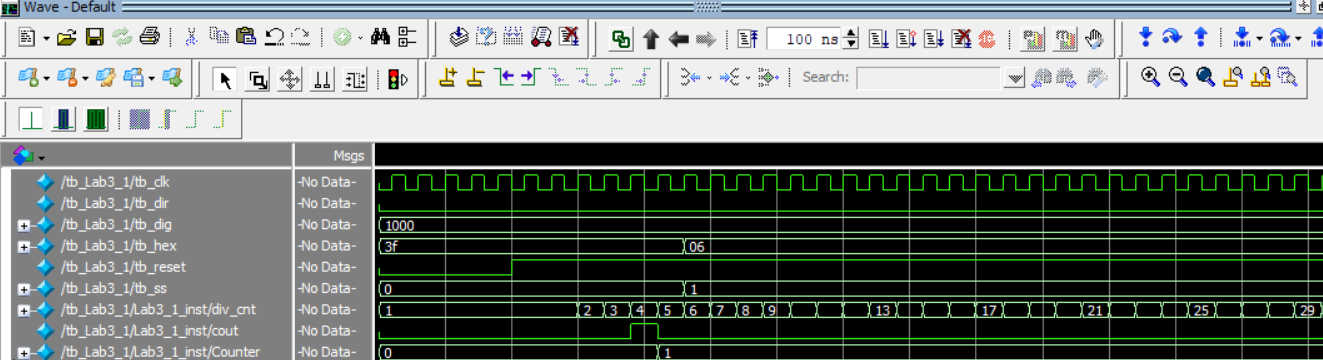


Рисунок 1.3 - Результат перезапуска моделирования

Видим, что вначале сигнал cout правильный (div\_cnt имеет значение 4, в соответствии с значением параметра в файле tb\_Lab3\_1.v).

Затем div\_cnt продолжает отсчет вместо того, чтобы заново начинать отсчет с 1.

Похоже, что Clock Divider (div\_cnt) в Lab3\_1.v не работает должным образом.

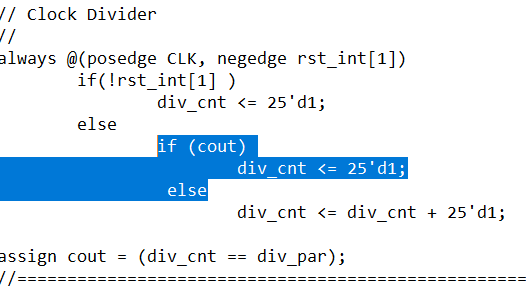


Рисунок 1.4 - Исправление файла Lab3\_1.v

Добавили недостающую часть кода в файле *Lab3\_1.v* и перекомпилировали файл.

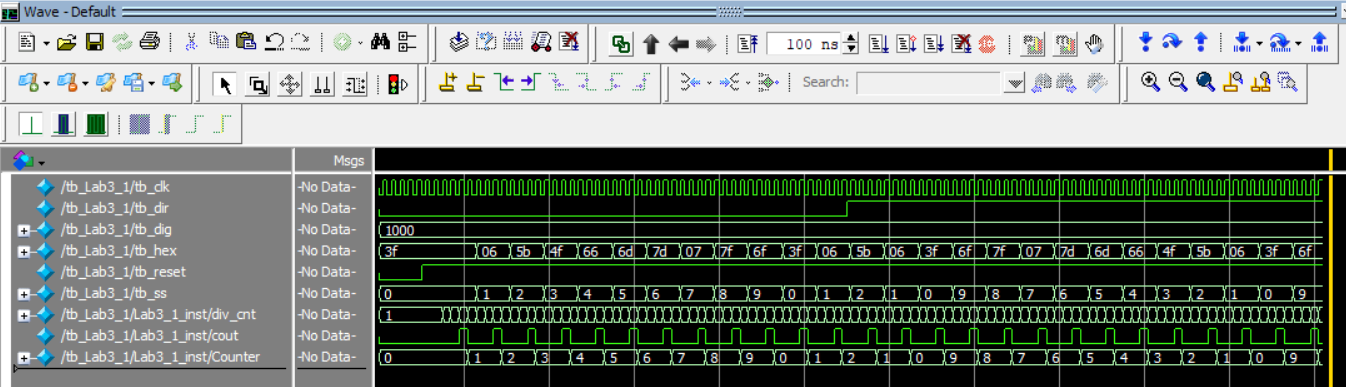


Рисунок 1.5 - Окно Wave после перекомпиляции проекта.

Проект пока работает исправно. На шине tb\_ss видим значения ASCII от 0 до 9. Счетчик считает вверх и вниз в соответствии со значениями DIR. Счетчик считается двоично-десятичным счетчиком.

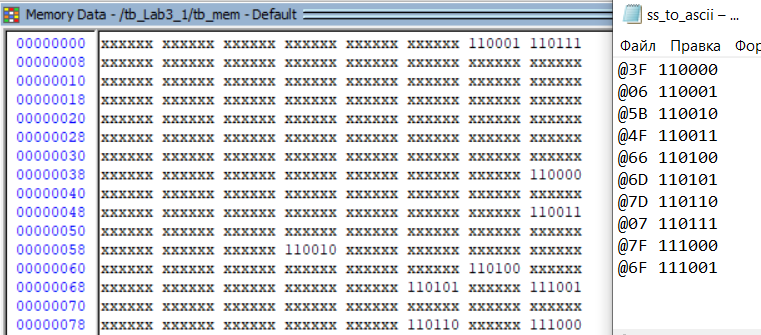


Рисунок 1.6 - Окно Memory Data и файл ss\_ascii.txt

В окне Memory Data вы можете увидеть массив памяти tb\_mem, инициализированный значениями из файла ss\_ascii.txt (Адрес в шестнадцатеричном формате. Данные в двоичном формате).

## Отладка проекта

1. Добавили IP-компоненты для отладки.

Некоторые IP-компоненты необходимо добавить в целях отладки. Это:

* ***Intel FPGA In-System Source & Probes*** - устройство будет использоваться для настройки (поиска) и просмотра (исследование) внутренних сигналов проекта (вместо использования реальных выводов FPGA).
* ***ALTPLL*** - единица будет использоваться для умножения входных часов. Умноженные часы будут использоваться в Signal Tap II.

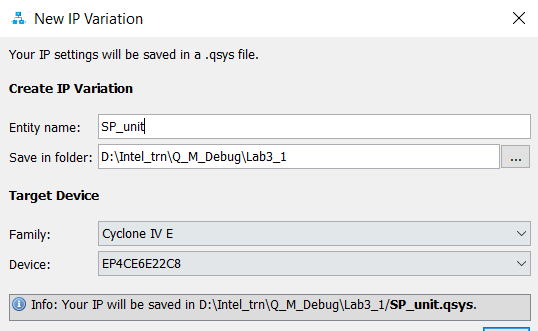


Рисунок 1.7 - Создание IP-компонента SP\_unit

После создание IP-компонентов убедились, что они были добавлены в проект.

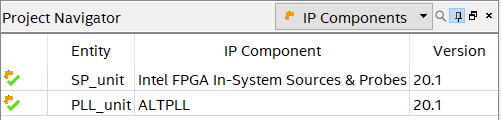


Рисунок 1.8 - Созданные IP-компоненты успешно добавлены в проект

## Файл верхнего уровня для отладки

Для отладки необходимо создать файл верхнего уровня. Файл *db\_Lab3\_1.v* предоставляется для лаборатории в папке *D:\Intel\_trn\Q\_M\_Debug\Lab3\_1.*

Файл верхнего уровня включает:

* Вход **CLK** — это входной контакт ПЛИС, связанный с тактовым генератором на плате.
* Атрибут **chip\_pin** устанавливает номер pin.
* altera\_attribute устанавливает стандарт ввода / вывода для pin.
* **Lab3\_1\_inst** - создание экземпляра сущности **Lab3\_1**.
* **SP\_unit\_inst** - создание экземпляра объекта **SP\_unit**.
* **PLL\_unit\_inst** - экземпляр объекта **PLL**\_**unit**.
* Выходная синхронизация (c0) **PLL\_unit\_inst** связана с сигналом **db\_clk\_high**, который не подключен к любым единицам в описании верхнего уровня. Сигнал **db\_clk\_high** будет использоваться как часы источник для логического анализатора Signal Tap II, который будет добавлен в проект позже.

1. Добавили в проект файл db\_Lab3\_1.v
2. Установили файл db\_Lab3\_1.v как файл верхнего уровня для проекта.

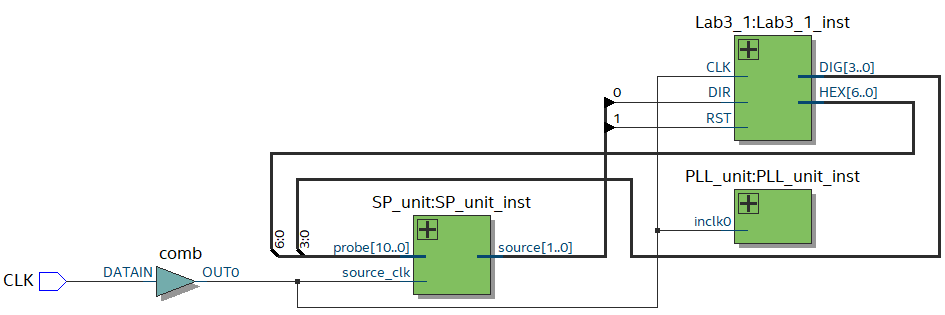


Рисунок 1.9 - Схема проекта в RTL Viewer

## Добавление и настройка Signal Tap II

Мы используем логический анализатор Signal Tap II для отладки. Настроили логический анализатор.

Запустили Signal Tap II (*Tools => Signal Tap Logic Analyzer*) и добавили несколько узлов для анализа.

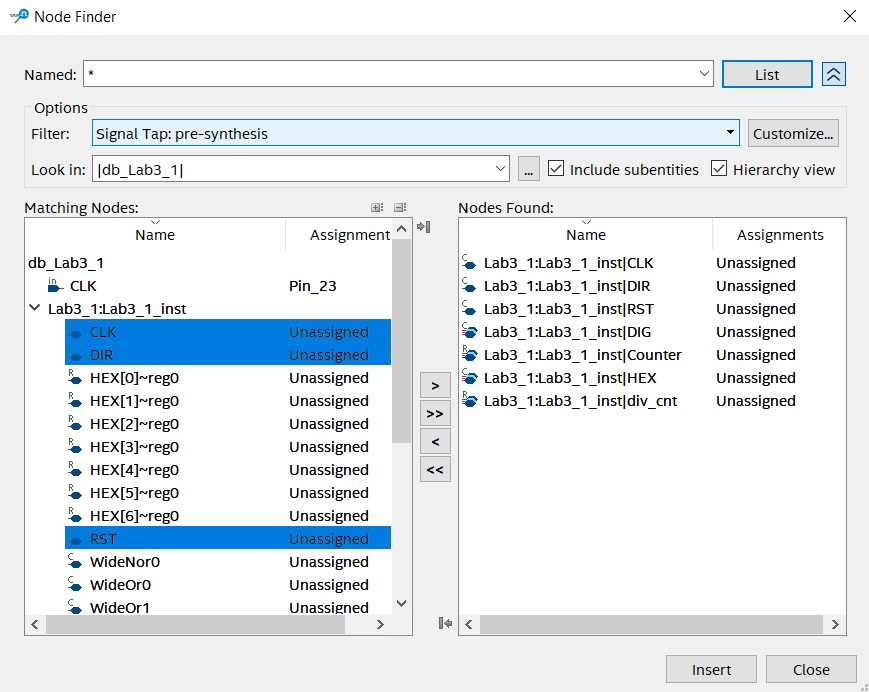


Рисунок 1.10 - Добавленные узлы для анализа

В окне *Mnemonic Table Setup* в таблицу *TABLE.stp* добавили ***ss\_udec: width = 7.***

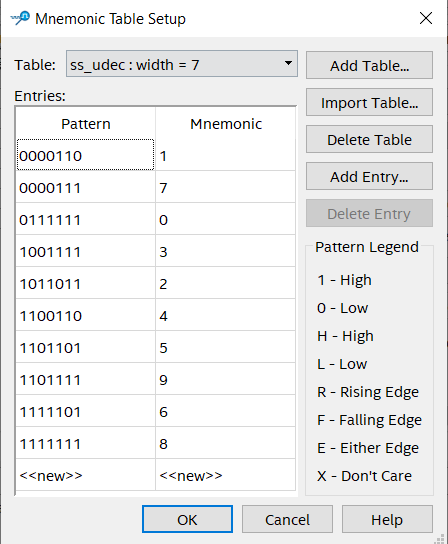


Рисунок 1.11 - Результат добавления ss\_udec: width = 7

Видим 7-сегментный код (левая часть таблицы) и соответствующую мнемонику, т.е. беззнаковые десятичные значения (правая часть таблицы). Таблица будет использоваться для упрощения процедура отладки.

1. Для *HEX [6..0] bus* выбрали ***ss\_udec: width = 7*** (*Bus Display Format => ss\_udec: width = 7*).

* Мнемоническая таблица будет использоваться для отображения значений на шине.

1. Для *div\_cnt* и *Counter* устанавливаем формат отображения шины - десятичное число без знака (*Bus Display Format => Unsigned decimal*).
2. В поле *Trigger Conditions* для сигнала *RST* установили нарастающий фронт (Rising Edge).

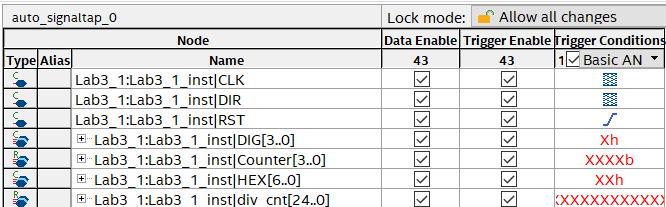


Рисунок 1.12 - Результат настройки таблицы

1. Назначили тактовый сигнал для синхронизации **Signal Tap II**.

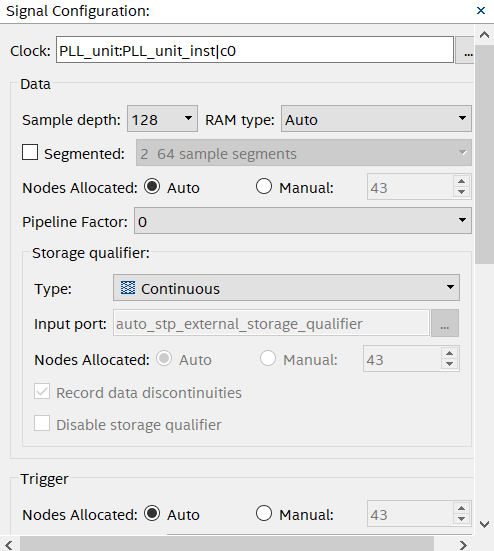


Рисунок 1.13 - Тактовый сигнал для синхронизации

1. Добавили файл в проект под именем **Lab3\_1.stp.**

## Полная компиляция

Для полной компиляции (для корректного анализа времени) нам нужен файл *.sdc* с временными ограничениями. Для этой лабораторной работы предоставляется файл ***db\_Lab3\_1.sdc***.

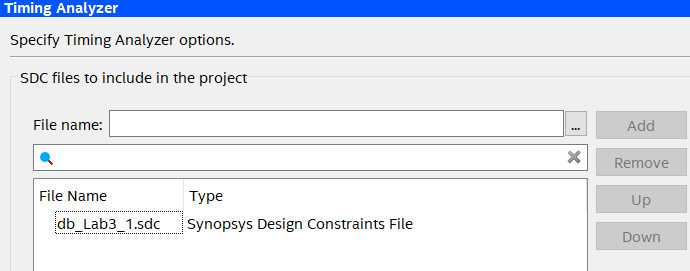


Рисунок 1.14 - Добавили файл db\_Lab3\_1.sdc

1. Запустили полную компиляцию проекта.

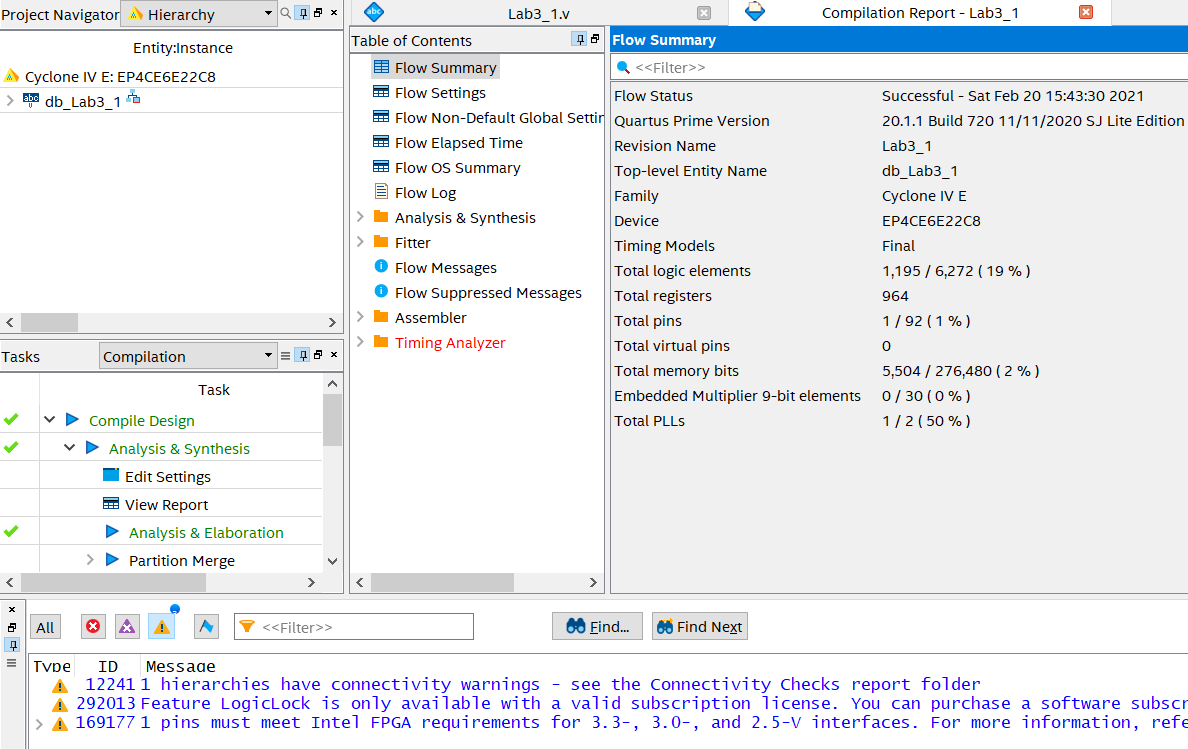


Рисунок 1.15 - Полная компиляция проекта

* Допускается наличие некоторых предупреждений, показанных на рисунке 15.

1. Проверили отчет компиляции (*Compilation Report => Timing Analyzer*).

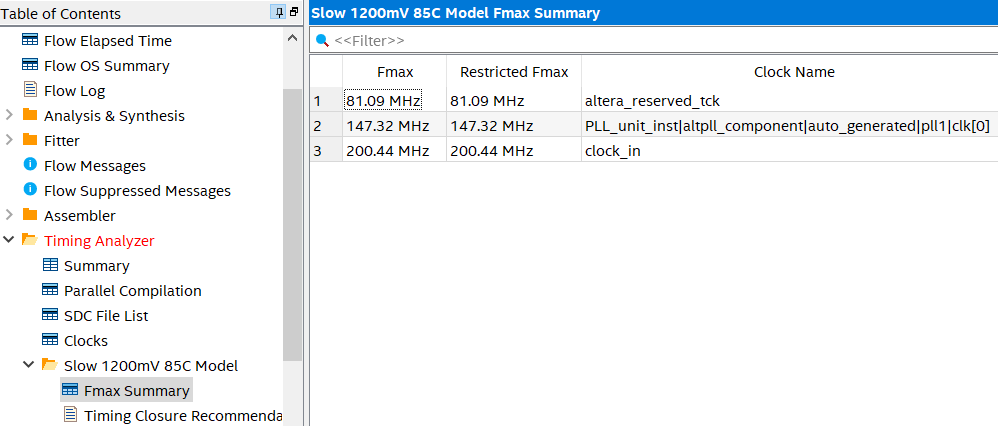


Рисунок 1.16 - Timing Analyzer

* Нарушений по времени нет.
* Допустимое значение Fmax для входа CLK (clock\_in) в самом медленном углу намного выше, чем требуется 25 МГц.
* Неограниченные пути — это входы JTAG для SignalTap II.

## Отладка проекта

Заходим в окно Programmer, выбираем необходимый интерфейс и загружаем на плату файл Lab3\_1.sof

Загружаем проект на плату.

Затем открываем окно In-System Source and Probes Editor и применяем необходимые настройки перед стартом.

1. Подключили плату к USB-порту ПК и включили её.

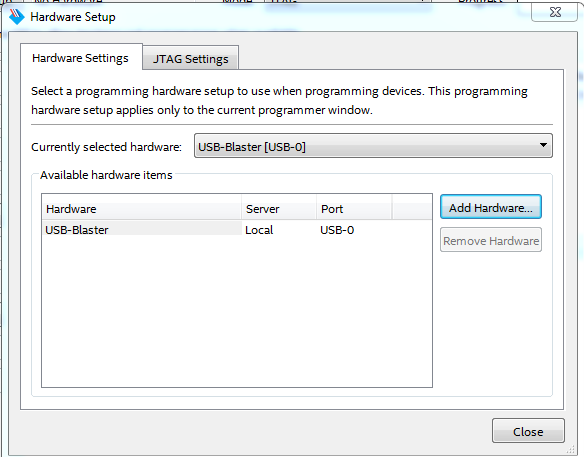


Рисунок 1.17 - Доступное оборудование - USB-Blaster

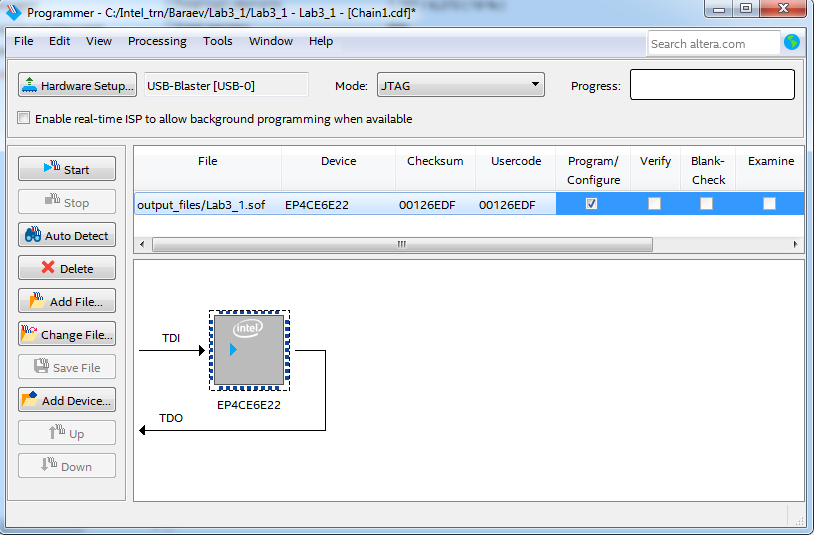


Рисунок 1.18 - Запуск файла Lab3\_1.sof

1. В окне In-System Source and Probes Editor отрегулировали все настройки, как было показано на рисунке в методичке.

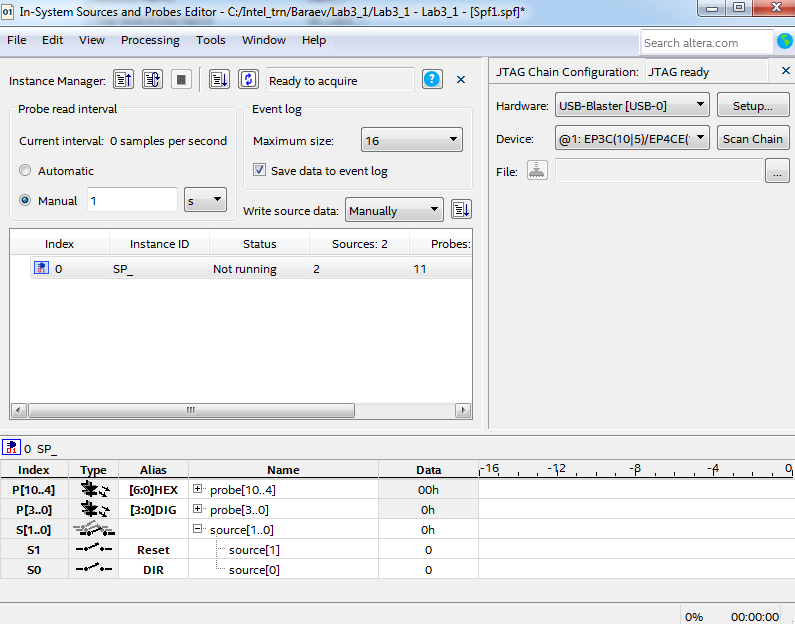


Рисунок 1.19 - Окно In-System Source and Probes Editor. Регулировка настроек

1. В том же окне настроили постоянное считывание данных. (**Processing => Continuously Read Probe Data)**

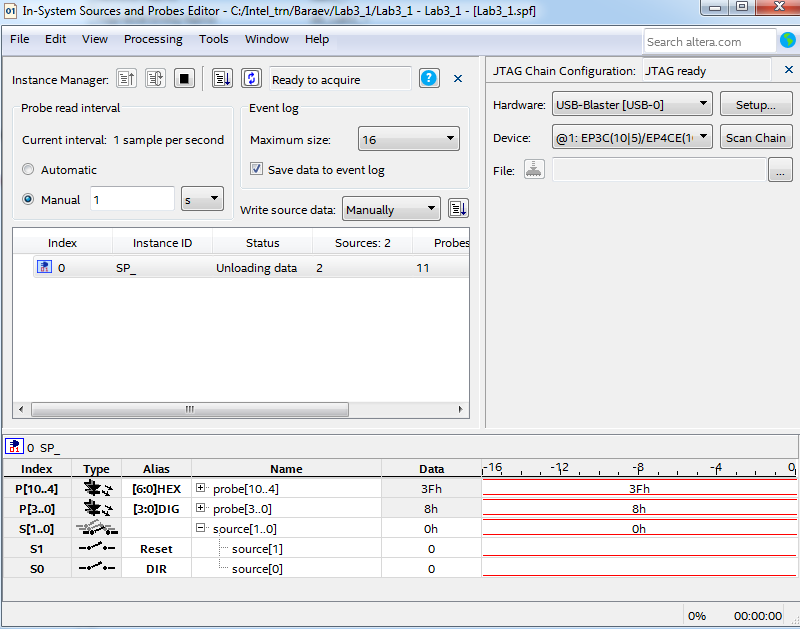


Рисунок 1.20 - Настройка постоянного считывания данных

Видим, что:

* Выходы шины HEX имеют постоянное значение 3F (это 7-сегментный код для 0).
* Это правильно, когда сигнал сброса равен 0 (активен).
* Выходы шины DIG имеют постоянное значение 8.
* Правильно: выбирается крайняя левая цифра 7-сегментного индикатора

Счетчик начинает работать после изменения значения Reset (source [1]) с 0 на 1.

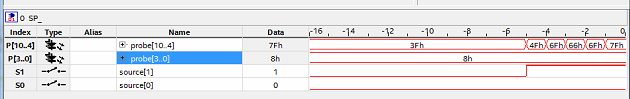


Рисунок 1.21 - Проверка сигнала Reset

1. В окне Signal Tap Logic Analyzer запускаем логический анализатор. Signal Tap будет ожидать фронт сигнала Reset. При изменении Reset из ISSPE получаем следующую временную диаграмму, снятую с реального устройства.

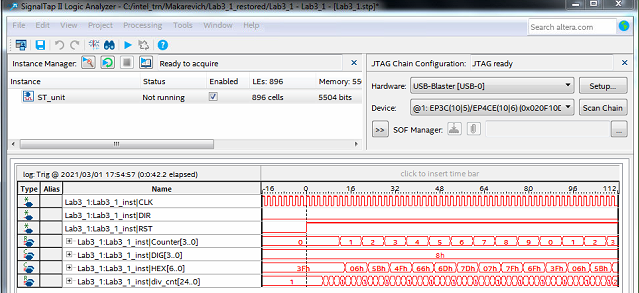


Рисунок 1.22 - Временная диаграмма

1. В файле db\_Lab3\_1.v добавляем параметр (25000000) для модуля Lab3\_1, как указано в задании. Это обеспечит период счётчика-делителя в 1 секунду. На плате установлен тактовый генератор в 25 МГц.
2. Запускаем полную компиляцию ещё раз.

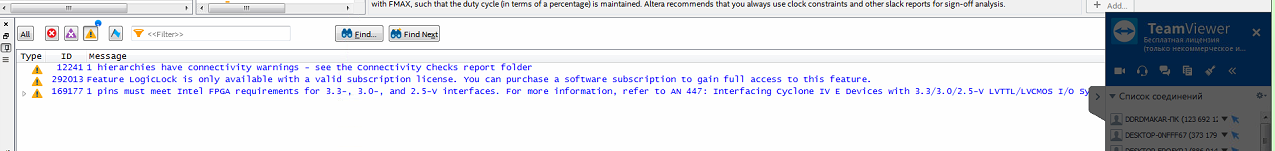


Рисунок 1.23 - Предупреждения и ошибки во время полной компиляции

1. Перепрошили плату и запустили окно In-System Sources and Probes Editor

Выставили сигналы Reset=0, Dir=0, потом Reset=1, Dir=0, и потом Reset=1, Dir=1.

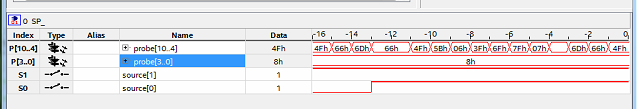


Рисунок 1.24 - In-System Sources and Probes Editor: Reset=1, Dir=1

Направление счёта изменилось, теперь в SignalTap поменяем значение div\_cnt, как указано в задании.

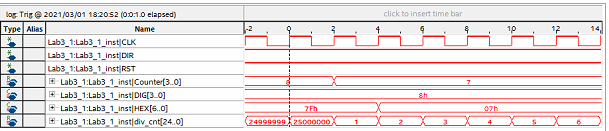


Рисунок 1.25 - SignalTap Logic Analyzer: временная диаграмма

Теперь изменим условия срабатывания триггера, как указано в задании.

Устройство работает правильно при всех комбинациях входных сигналов.

## Добавление модуля верхнего уровня

Добавим в проект файл impl\_Lab3\_1.v и назначим его файлом верхнего уровня. В нём содержится описание модуля, а также назначение выводов модуля реальным выводам микросхемы на тестовом стенде.

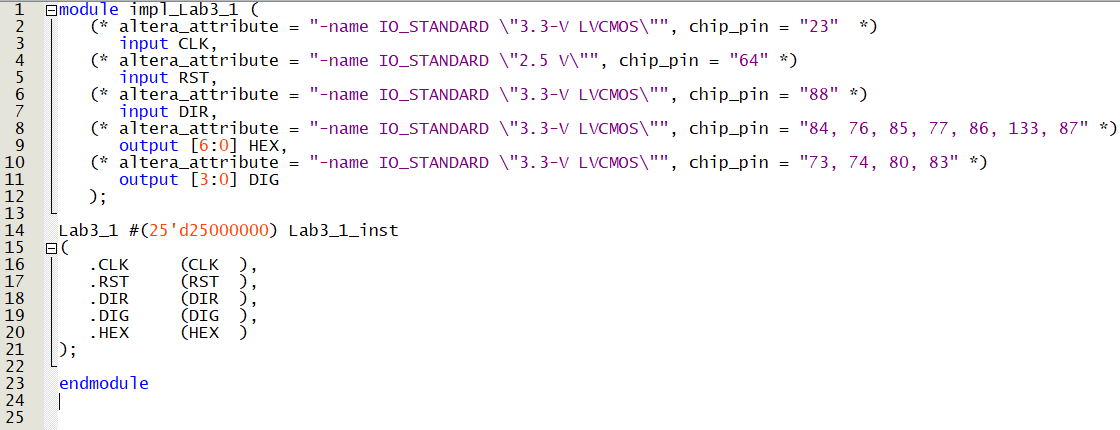


Рисунок 1.26 - Содержимое файла impl\_Lab3\_1.v

После полной компиляции проекта получаем следующую структуру устройства в RTL viewer.

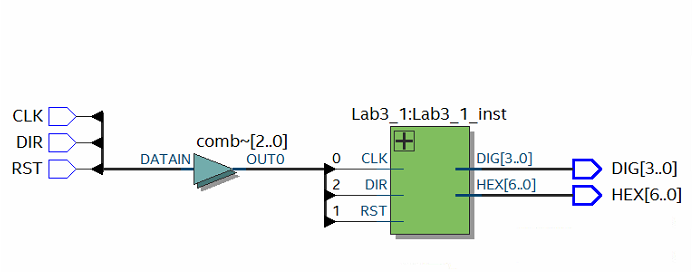


Рисунок 1.27 - Структура устройства в RTL Viewer

После прошивки тестового стенда можно наблюдать работу семисегментных индикаторов.

## Выводы

В ходе данной лабораторной работы научились:

1. Создавать проект в Quartus Prime
2. Моделировать проект с помощью ModelSim
3. Отлаживать проект на плате с помощью ISSP и SignalTap II.
4. Реализации проекта на плате

Устройство работает правильно, период счёта = 1с, устройство реагирует на переключатели Reset и Dir.

# Lab 3\_2 Самостоятельная работа

## Создание проекта

Микросхема FPGA - 10M50DAF484C6GES, семейство МАХ10.

## Создать описание модулей на языке Verilog

1. Создание модуля SYNCin.

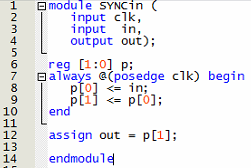


Рисунок 2. 1 - Программная реализация модуля SYNCin

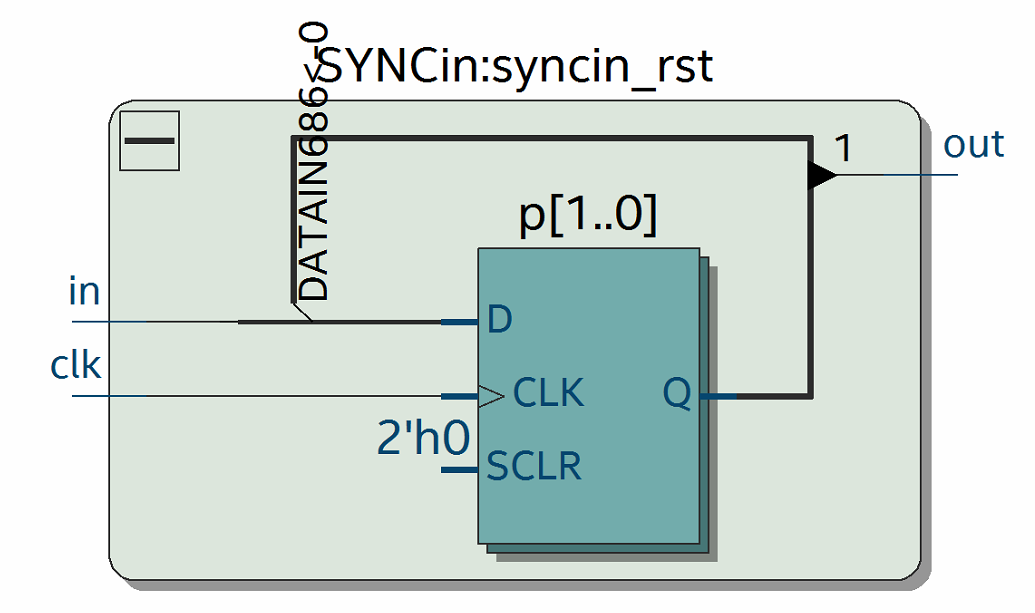


Рисунок 2. 2 - Структура модуля SYNCin в RTL Viewer

1. Создание модуля CNT\_DIV.

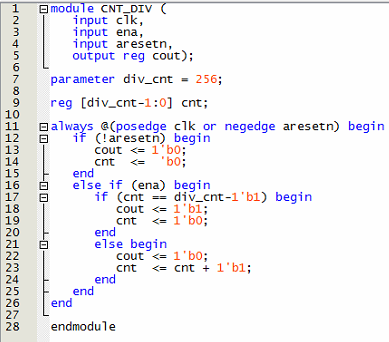


Рисунок 2. 3 - Программная реализация модуля CNT\_DIV

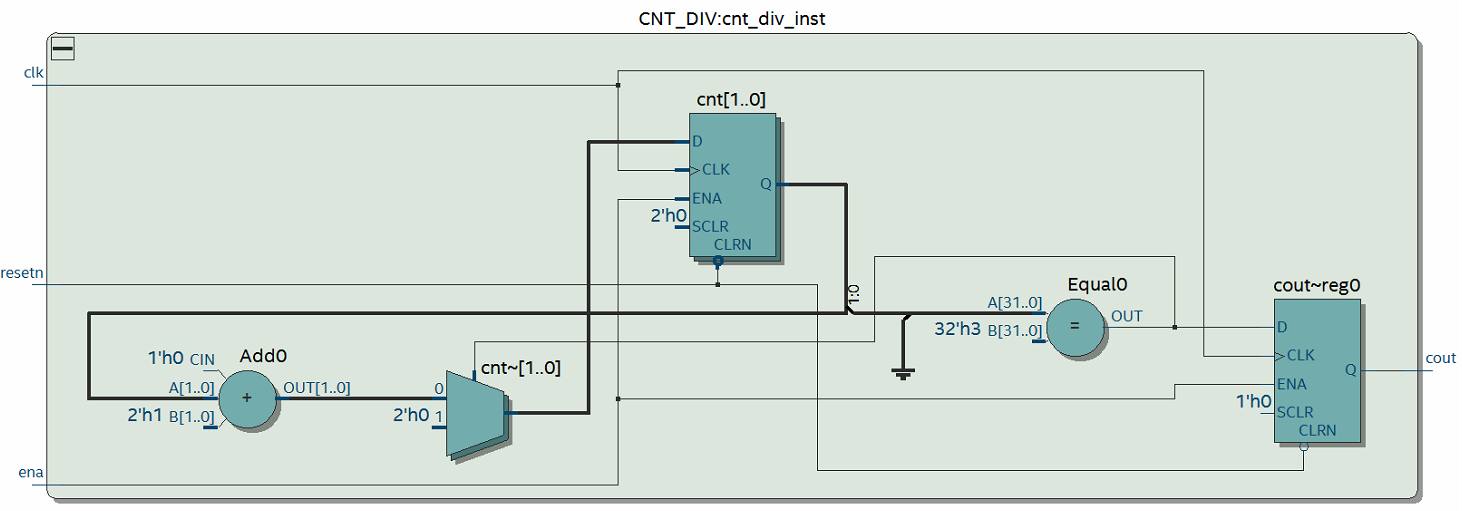


Рисунок 2. 4 - Структура модуля CNT\_DIV в RTL Viewer

1. Создание модуля CNT.

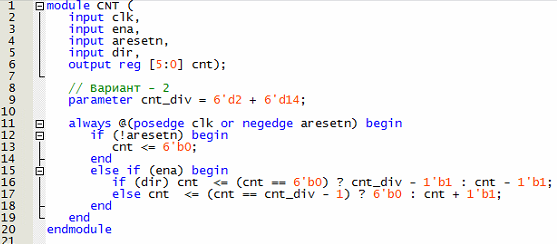


Рисунок 2. 5 - Программная реализация модуля CNT

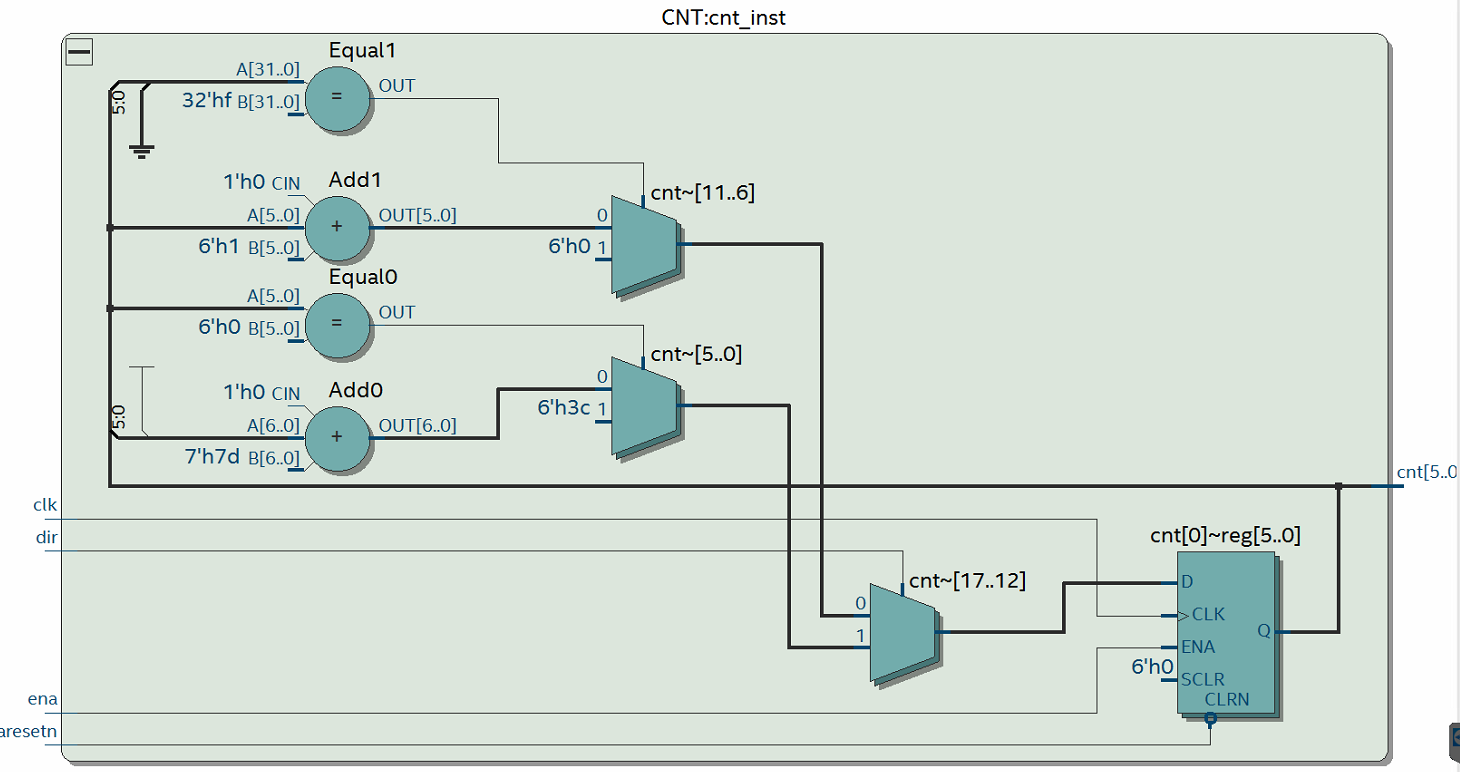


Рисунок 2. 6 - Структура модуля CNT в RTL Viewer

1. Создание модуля CODER.

Модуль CODER создаём на основе IP модуля ROM: 1-PORT.

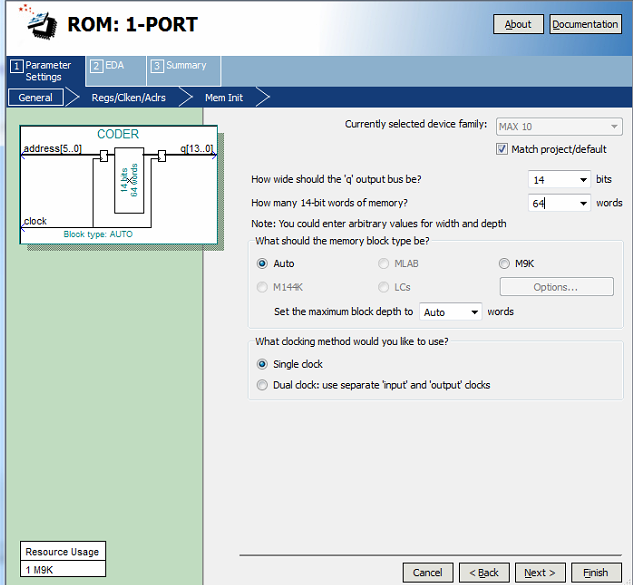


Рисунок 2. 7 - Создание модуля CODER

1. Создание модуля SYNCout.

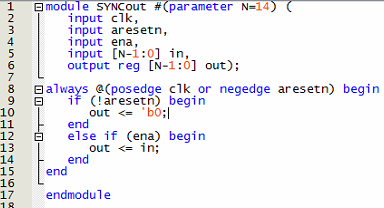


Рисунок 2. 8 - Программная реализация модуля SYNCout

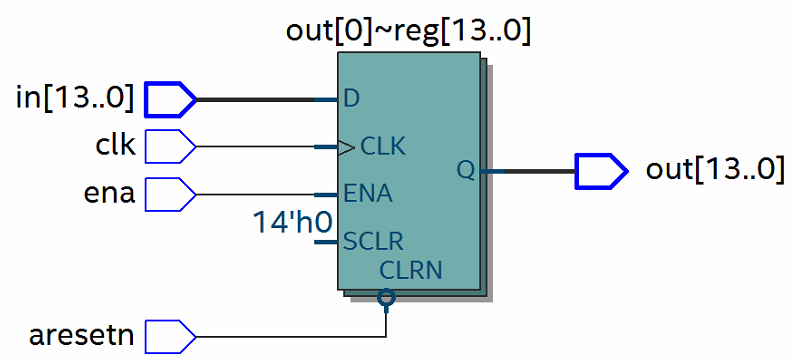
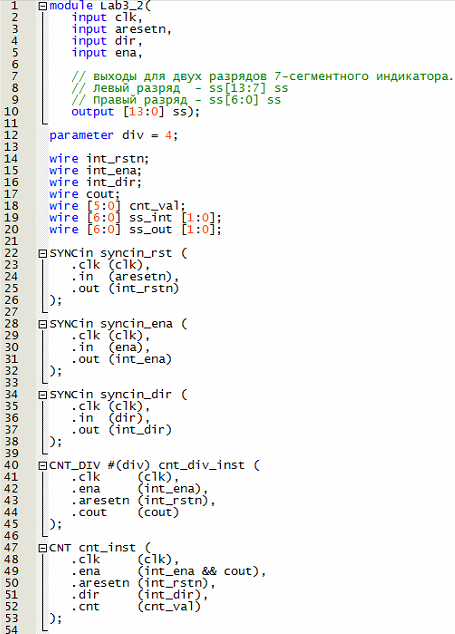


Рисунок 2. 9 - Структура модуля SYNCout в RTL Viewer

1. Создание модуля верхнего уровня Lab3\_2.



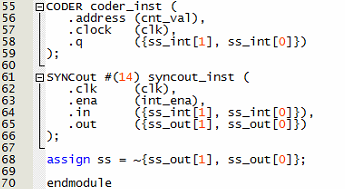


Рисунок 2. 10 - Программная реализация модуля верхнего уровня Lab3\_2

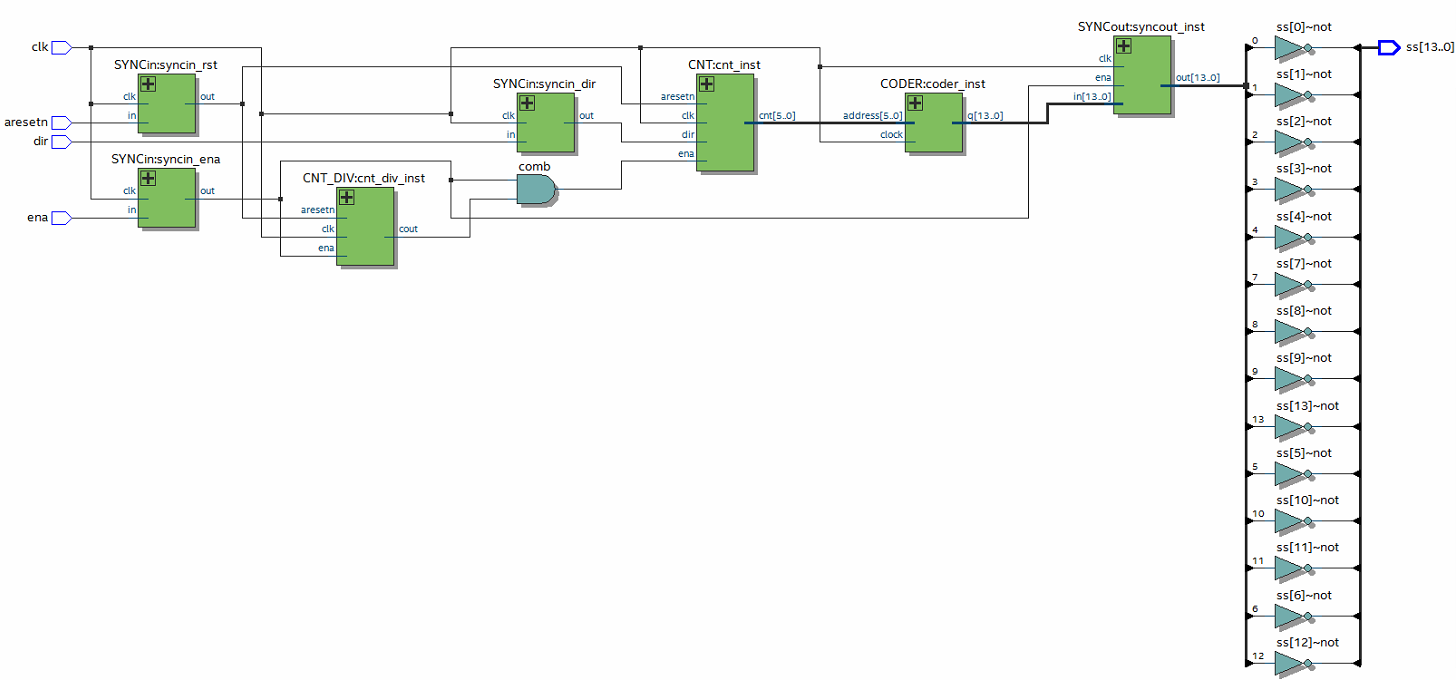


Рисунок 2. 11 - Структура модуля верхнего уровня Lab3\_2 в RTL Viewer

## Моделирование в пакете ModelSim

1. Создать файл tb\_Lab3\_2.v с тестом класса 1.

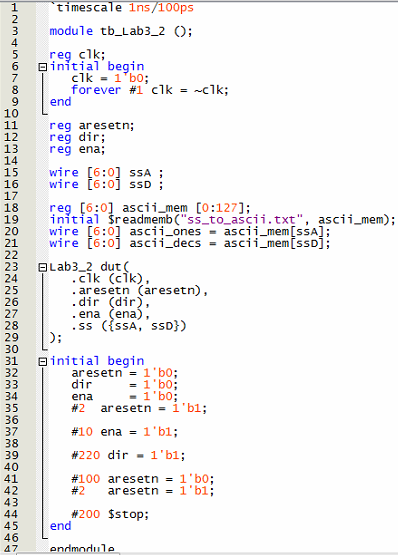


Рисунок 2. 12 - Программная реализация тестового модуля tb\_Lab3\_2.v

1. Для запуска теста создаём проект в ModelSim и скомпилируем в нём все исходные файлы.

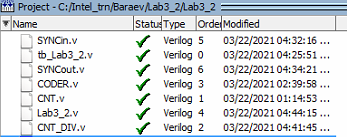


Рисунок 2. 13 - Файлы в проекте ModelSim

1. Добавляем библиотеку altera\_mf\_ver и запускаем симуляцию.

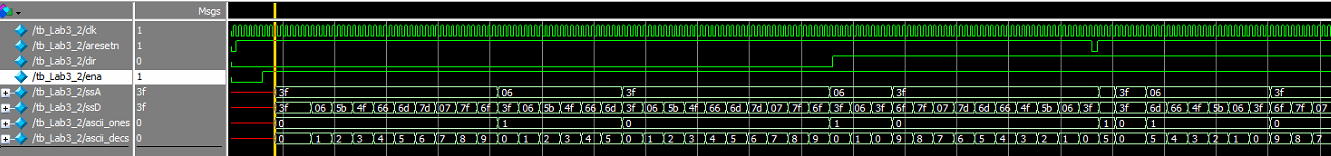


Рисунок 2. 14 - Временная диаграмма

Сигналы ascii\_decs и ascii\_ones на временной диаграмме отображают десятки и единицы в виде десятичных чисел.

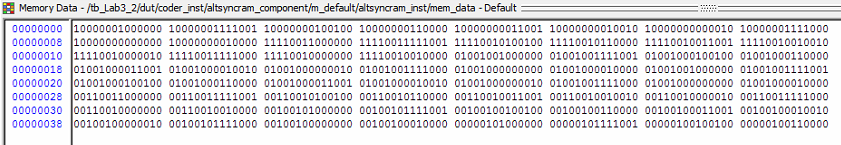


Рисунок 2. 15 - Содержимое памяти ROM



Рисунок 2. 16 - Содержимое массивов памяти, используемых для перекодировки 7-сегментных кодов в без-знаковые десятичные цифры

После тестирования видно, что:

* Правильно формируются все выходные данные для заданного модуля счёта.
* Сигнал разрешения работы
* Режимы сложения и вычитания
* Сигнал сброса

Следовательно, можно сделать вывод, что все модули работают правильно.

## Тестирование и отладка на плате

1. Для отладки проекта на плате необходимо создать модуль на основе ALTPL, удваивающий входную тактовую частоту.

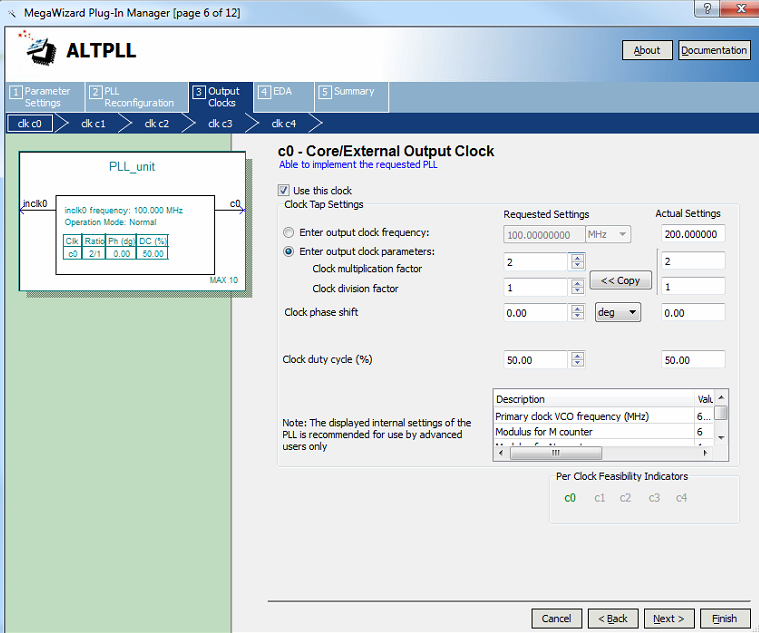


Рисунок 2. 17 – Модуль PLL\_unit

1. После этого нам для отладки требуется создать модуль на основе Altera In-System Sources and Probes.

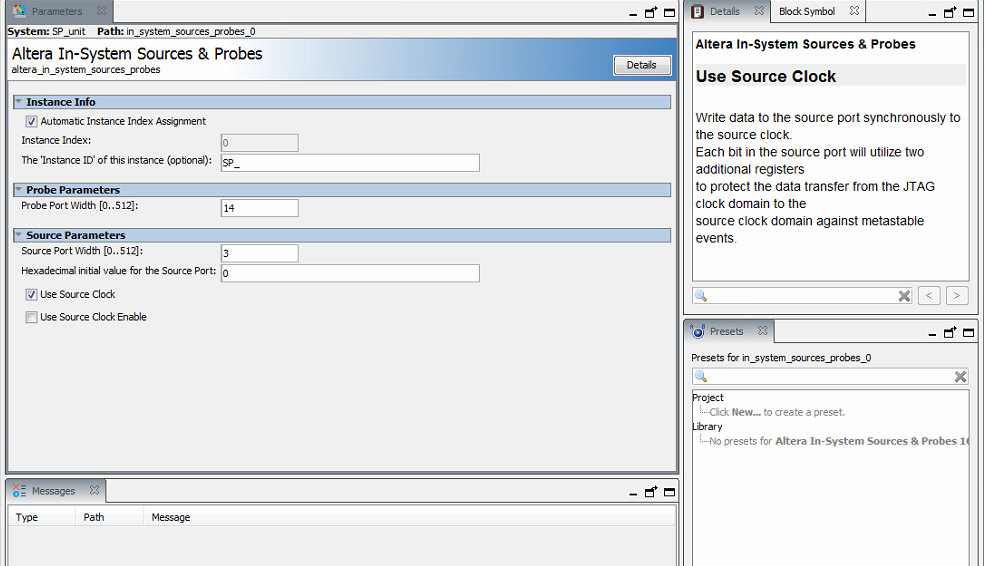


Рисунок 2. 18 - Создание модуля отладчика на основе Altera In-System Sources and Probes



Рисунок 2. 19 - Созданные модули

1. Добавили в проект файл db\_Lab3\_2

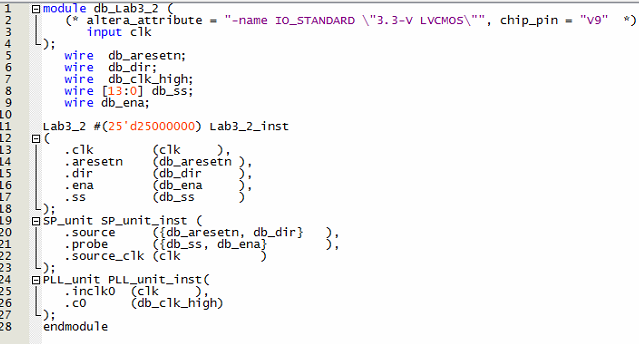


Рисунок 2. 20 - Содержимое файла db\_Lab3\_2

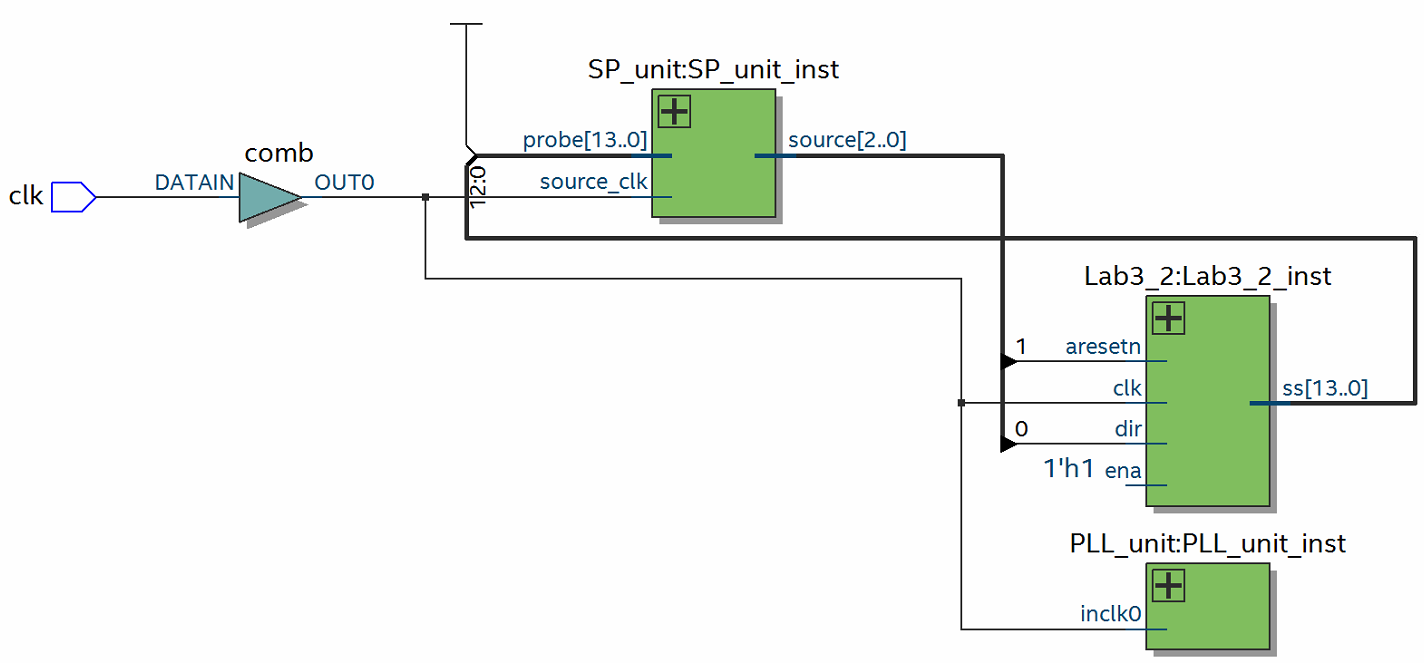


Рисунок 2. 21 - Структура модуля db\_Lab3\_2 в RTL Viewer

1. Настраиваем SignalTap, как требуется в задании.



Рисунок 2. 22 - Настройки SignalTap

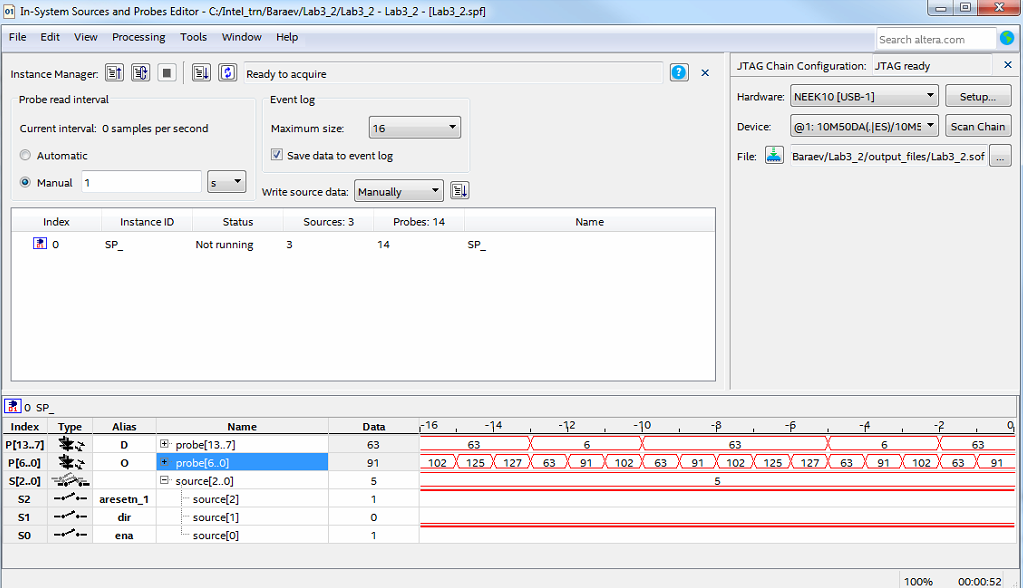


Рисунок 2. 23 - Окно In-System Sources and Probes Editor

1. Скомпилируем проект вместе с отладчиком и загрузим на тестовый стенд.
2. Разработаем файл impl\_Lab3\_2.v с описанием модуля верхнего уровня для реализации на плате.

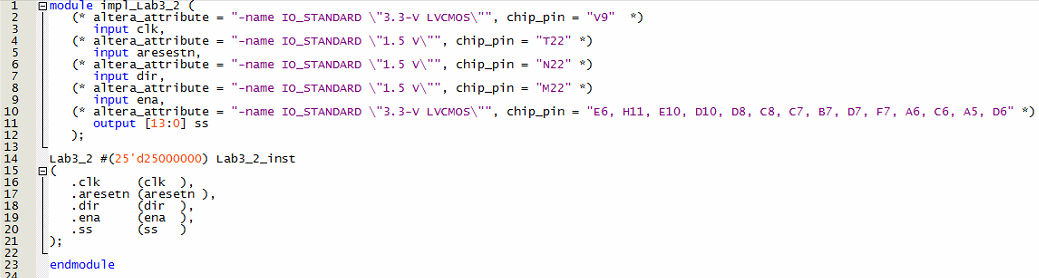


Рисунок 2. 24 - Содержимое файла impl\_Lab3\_2.v

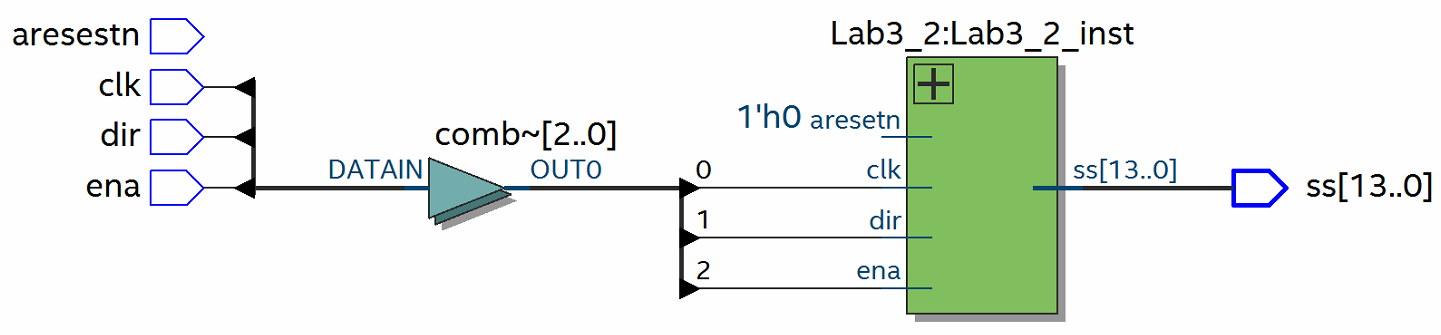


Рисунок 2. 25 - Структура модуля impl\_Lab3\_2 в RTL Viewer

## Программирование и проверка платы

1. Подключили плату к компьютеру.

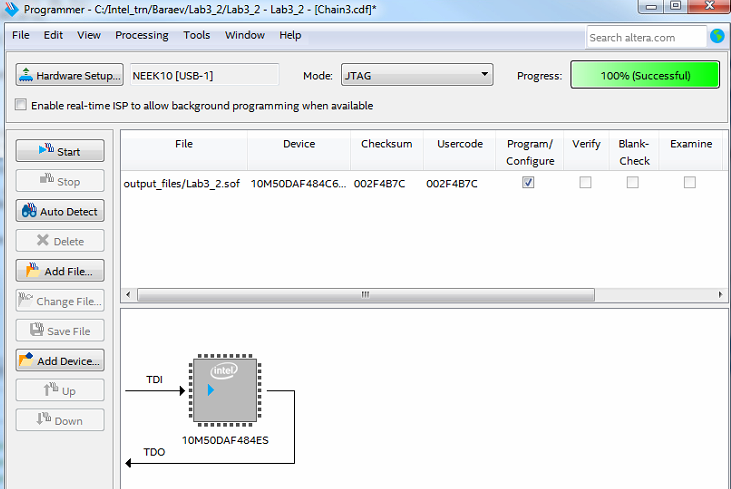


Рисунок 2. 26 - Окно Programmer

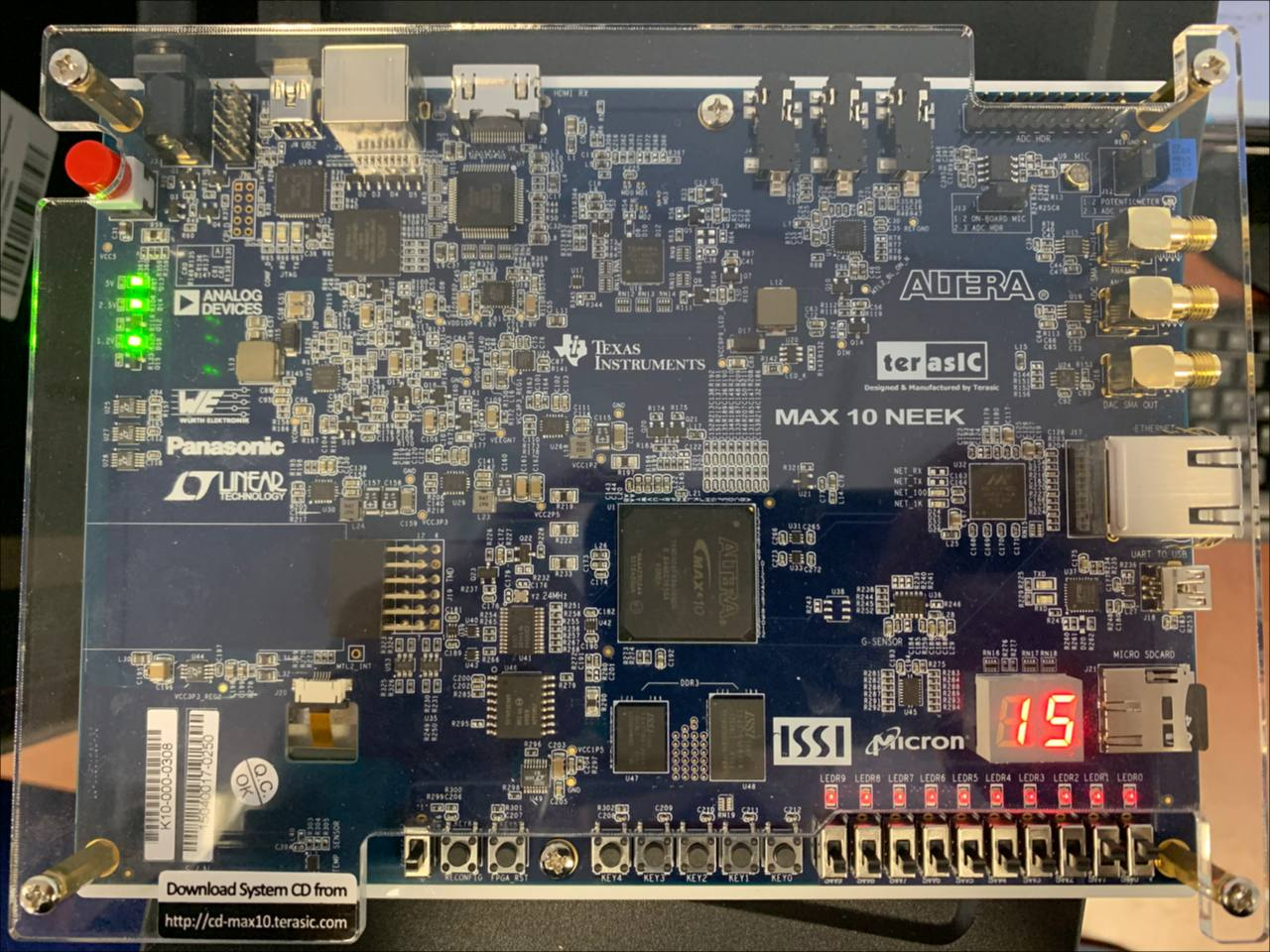


Рисунок 2. 27 - Результат работы платы

## Выводы

В ходе данной самостоятельной лабораторной работы создали проект в пакете Quartus, успешно осуществили моделирование в пакете ModelSim, осуществили тестирование и отладку на плате и осуществили реализацию на плате.