Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчёт по лабораторной работе Lab\_PD2**

Дисциплина: Автоматизация проектирования дискретных устройств (на английском языке)

Выполнил студент гр. 5130901/10101 М.Т. Непомнящий

(подпись)

Руководитель А.А. Федотов

(подпись)

Санкт-Петербург

2024

# Оглавление

[1. Оглавление 2](#_Toc162458719)

[1. Задание 4](#_Toc162458720)

[2. Ход работы 4](#_Toc162458721)

[2.1. Создание проекта 4](#_Toc162458722)

[Подготовка проекта 4](#_Toc162458723)

[Начало работы в PD 5](#_Toc162458724)

[2.2. Настройка компонентов 6](#_Toc162458725)

[Настройка clk\_0 6](#_Toc162458726)

[Настройка sc\_fifo\_0 6](#_Toc162458727)

[Настройка MyST\_source\_0 и MyST\_sink\_0 7](#_Toc162458728)

[2.3. Подключение тактового сигнала 8](#_Toc162458729)

[2.4. Подключение Avalon-ST интерфейсов 8](#_Toc162458730)

[2.5. Анализ системы 10](#_Toc162458731)

[Проверка блока 10](#_Toc162458732)

[Анализ с помощью Schematic 11](#_Toc162458733)

[2.6. Генерация системы 12](#_Toc162458734)

[2.7. Подключение файлов к проекту 13](#_Toc162458735)

[3. Тестирование проекта 14](#_Toc162458736)

[3.1. Тестирование средствами ModelSim 14](#_Toc162458737)

[Создание тестового файла 14](#_Toc162458738)

[Симуляция средствами ModelSim 15](#_Toc162458739)

[3.2. Тестирование средствами Signal Tap II 16](#_Toc162458740)

[Создание файла для отладки 16](#_Toc162458741)

[Настройка Signal Tap II 17](#_Toc162458742)

[Тестирование на плате средствами Signal Tap II 17](#_Toc162458743)

[4. Вывод 19](#_Toc162458744)

**Список иллюстраций**

[Рис. 1 – Структура проекта 4](#_Toc162458745)

[Рис. 2 – Детали проекта 4](#_Toc162458746)

[Рис. 3 – Задания пути к библиотеке IP 5](#_Toc162458747)

[Рис. 4 – Исходное окно PD 5](#_Toc162458748)

[Рис. 5 – Добавление компонентов 6](#_Toc162458749)

[Рис. 6 – Настройка компонента clk 6](#_Toc162458750)

[Рис. 7 – Настройка компонента sc\_fifo 7](#_Toc162458751)

[Рис. 8 – Система после переименования компонентов 7](#_Toc162458752)

[Рис. 9 – Подключение тактового сигнала (1) 8](#_Toc162458753)

[Рис. 10 – Подключение тактового сигнала (2) 8](#_Toc162458754)

[Рис. 11 – Подключение Avalon-MM интерфейсов 8](#_Toc162458755)

[Рис. 12 – Экспорт выводов 9](#_Toc162458756)

[Рис. 13 – Проверка поля Messages на отсутствие ошибок 9](#_Toc162458757)

[Рис. 14 – Символ системы 10](#_Toc162458758)

[Рис. 15 – Show System with QSYS Interconnect 10](#_Toc162458759)

[Рис. 16 – Анализ проблемных подключений 11](#_Toc162458760)

[Рис. 17 – Schematic (фильтр по in) 11](#_Toc162458761)

[Рис. 18 – Schematic (фильтр по clk) 11](#_Toc162458762)

[Рис. 19 – Schematic 12](#_Toc162458763)

[Рис. 20 – Предустановки окна Genreration 12](#_Toc162458764)

[Рис. 21 – Проверка успешности генерации HDL 12](#_Toc162458765)

[Рис. 22 – Подключение файлов к проекту 13](#_Toc162458766)

[Рис. 23 – Синтаксис файла Lab2\_top.sv 13](#_Toc162458767)

[Рис. 24 – Схема проекта в RTL Viewer 13](#_Toc162458768)

[Рис. 25 – Тестовый файл tb\_lab\_PD2\_top.sv 14](#_Toc162458769)

[Рис. 26 – Тестовый файл tb\_lab\_PD2\_top.sv 15](#_Toc162458770)

[Рис. 27 – Моделирование проекта средствами ModelSim 15](#_Toc162458771)

[Рис. 28 – Файл для отладки модуля верхнего уровня 16](#_Toc162458772)

[Рис. 29 – Схема проекта с добавлением SP\_unit в RTL Viewer 16](#_Toc162458773)

[Рис. 30 – Настройка окна Signal Tap II 17](#_Toc162458774)

[Рис. 31 – Временные характеристики устройства 17](#_Toc162458775)

[Рис. 32 – Результат SignalTap II 17](#_Toc162458776)

[Рис. 33 – Анализ рабочей папки проекта 18](#_Toc162458777)

# Задание

Средствами Platform Designer создать структуру проекта, представленную на рисунке ниже:

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рис. 1 – Структура проекта

# Ход работы

## Создание проекта

### Подготовка проекта

Создадим проект, установив следующие значения:

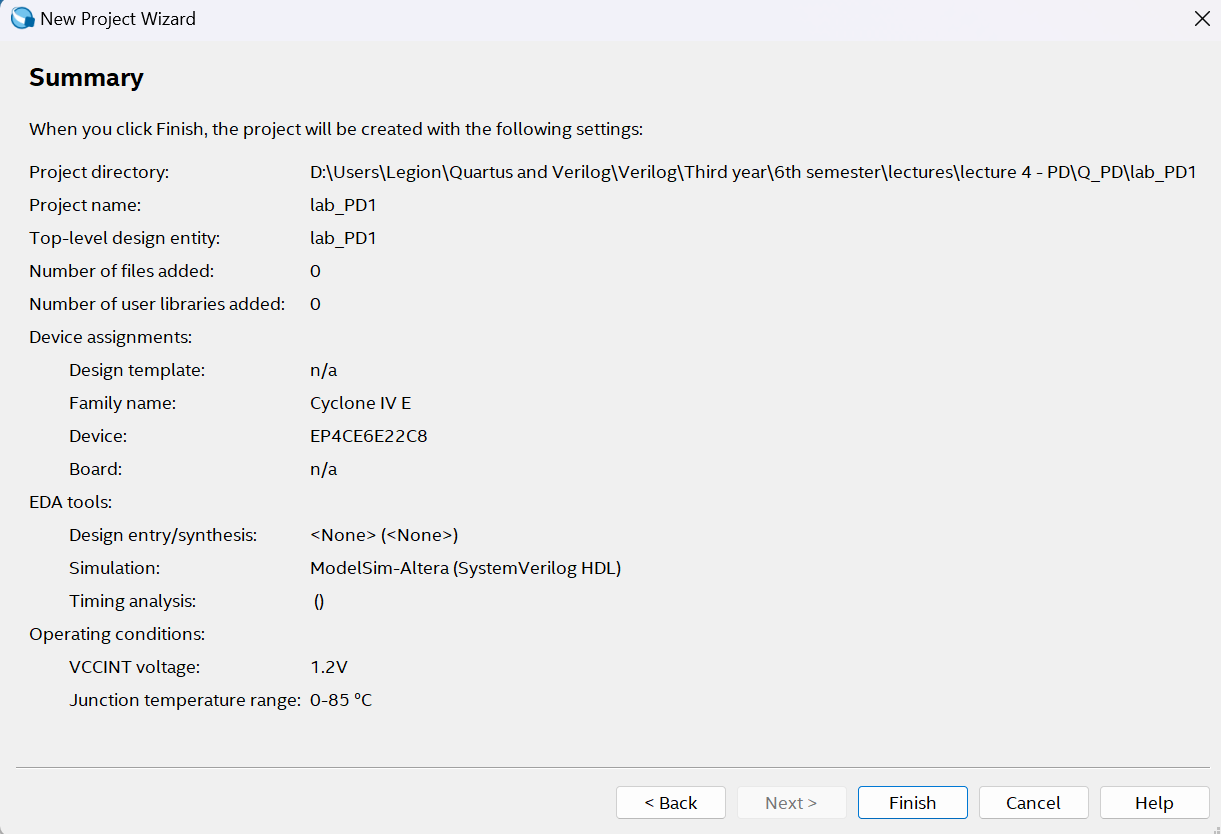


Рис. 2 – Детали проекта

Перейдём по пути Tools → Options → IP Settings → IP Catalog Search Locations и зададим путь к библиотеке IP:

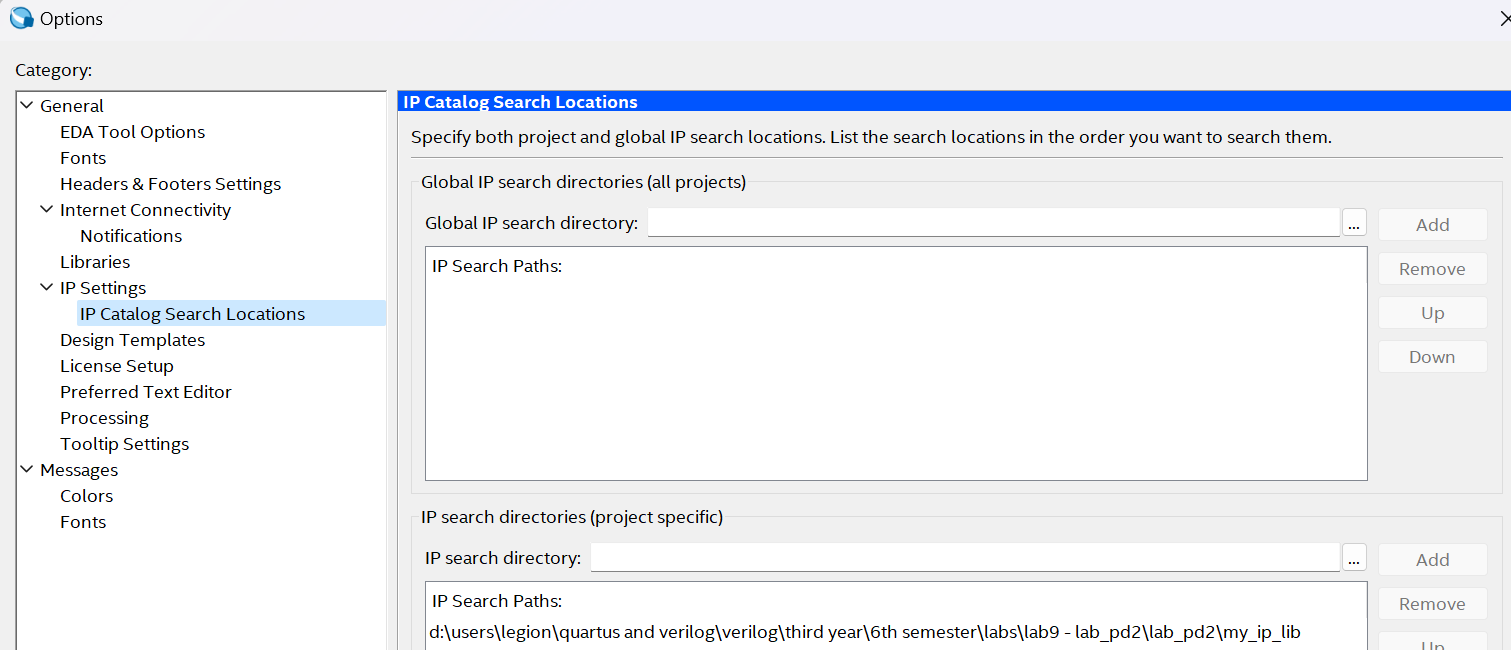


Рис. 3 – Задания пути к библиотеке IP

### Начало работы в PD

Откроем PD и сохраним систему:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рис. 4 – Исходное окно PD

Добавим компоненты: MyST\_source\_component, Avalon-ST Single Clock FIFO, MyST\_sink\_component. Таким образом, получим следующую картинку (в окне Hierarchy слева отображаются все добавленные компоненты):

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рис. 5 – Добавление компонентов

Наличие ошибок связано с тем, что настройка модулей не производилась, т. к. она будет рассмотрена дальше.

## Настройка компонентов

### Настройка clk\_0

Переименуем компонент **clk\_0** в **clk** и зададим значение Reset synchronous edges = Deassert

Изображение выглядит как текст, программное обеспечение, Шрифт, Значок на компьютере

Автоматически созданное описание

Рис. 6 – Настройка компонента clk

### Настройка sc\_fifo\_0

Переименуем компонент **dc\_fifo\_0** в **sc\_fifo**. Зададим значение Bits per symbol = 4:

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рис. 7 – Настройка компонента sc\_fifo

Запись данных будет происходить следующим образом: 100 – счёт на сложение, 200 – счёт на вычитание.

### Настройка MyST\_source\_0 и MyST\_sink\_0

Переименуем MyST\_source\_0 и MyST\_sink\_0 в MyST\_source и MyST\_sink соответственно. Получившаяся структура будет выглядеть следующим образом:

Изображение выглядит как текст, программное обеспечение, число, снимок экрана

Автоматически созданное описание

Рис. 8 – Система после переименования компонентов

## Подключение тактового сигнала

Выделим интерфейс clk компонента clk, и, открыв его соединения, выберем подключение ко всем тактовым входам:

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

Рис. 9 – Подключение тактового сигнала (1)

Подключим тактовый сигнал, выполнив Filter → Clock and Reset Interfaces, убедимся, что соединения выполнены. Также, подключим сигнал Reset, выполнив System → Create Global Reset Network, и убедимся, что соединения для reset также выполнены:

Изображение выглядит как текст, число, программное обеспечение, Шрифт

Автоматически созданное описание

Рис. 10 – Подключение тактового сигнала (2)

## Подключение Avalon-ST интерфейсов

Выполним Filter → Avalon-MM Interfaces и выберем соединения так, как показано на картинке ниже

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рис. 11 – Подключение Avalon-MM интерфейсов

Проведём экспорт выводов путём задания имён для выделенных модулей в столбце Export:

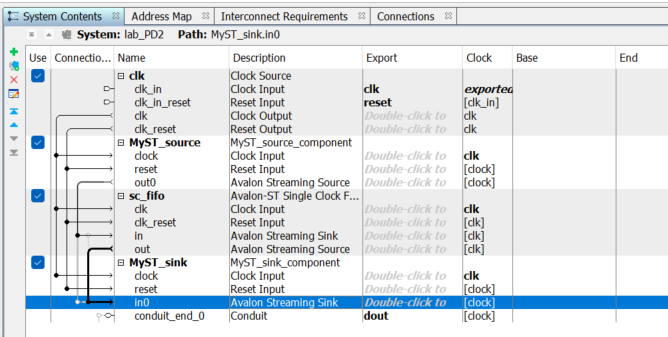


Рис. 12 – Экспорт выводов

Убедимся в том, что система не содержит ошибок и в поле Messages есть только 1 информационное сообщение:

Изображение выглядит как программное обеспечение, текст, Мультимедийное программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рис. 13 – Проверка поля Messages на отсутствие ошибок

## Анализ системы

### Проверка блока

Выполним View → Block Symbol и убедимся в том, что символ системы построен правильно:

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рис. 14 – Символ системы

Выполним команду System → Show System with PD Interconnect (Show System with QSYS Interconnect). Проверим, что новых модулей не появилось.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рис. 15 – Show System with QSYS Interconnect

Выполним View → Clock domains Beta, выберем режим отображения Reset. Заметим, что проблемных подключений не выявлено:

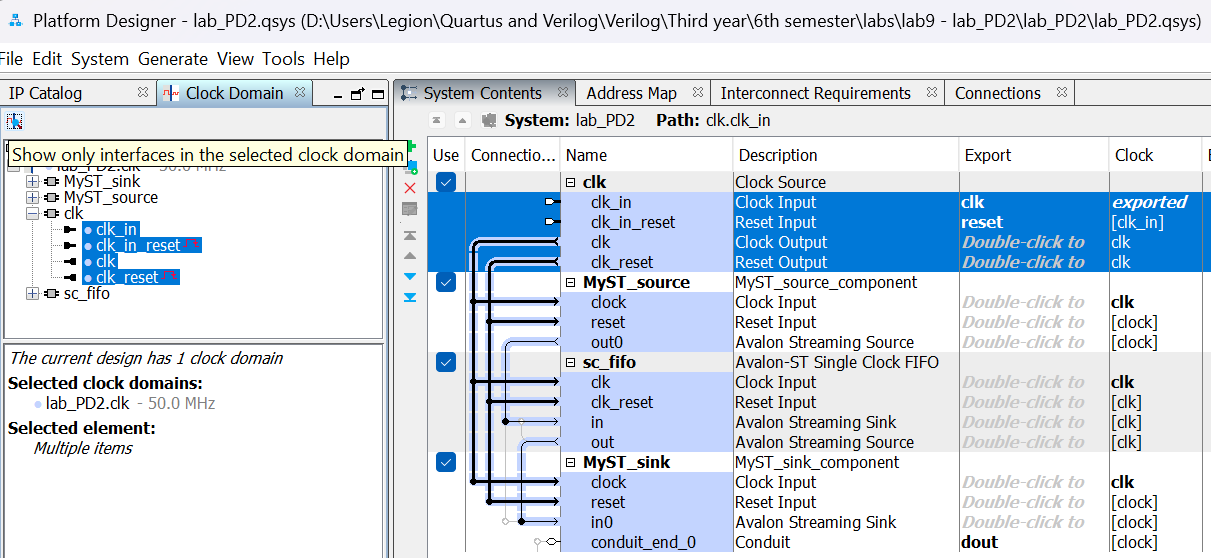


Рис. 16 – Анализ проблемных подключений

### Анализ с помощью Schematic

Выполним View → Schematic, в качестве фильтра введём in и убедимся в том, что система синхронизации и каналы ST системы подключены верно:

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Автоматически созданное описание

Рис. 17 – Schematic (фильтр по in)

Теперь введём в качестве фильтра clk, чтобы проверить, что шины Avalon MM подключены верно:

Изображение выглядит как текст, снимок экрана, дисплей, Шрифт

Автоматически созданное описание

Рис. 18 – Schematic (фильтр по clk)

Полный блок системы будет выглядеть следующим образом:

Изображение выглядит как текст, диаграмма, снимок экрана, Прямоугольник

Автоматически созданное описание

Рис. 19 – Schematic

## Генерация системы

Выполним PD → Generate HDL и укажем следующие предустановки для генерации:

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рис. 20 – Предустановки окна Generration

Удостоверимся в том, что генерация прошла успешно:

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рис. 21 – Проверка успешности генерации HDL

## Подключение файлов к проекту

Подключим файлы к проекту в Quartus

Изображение выглядит как текст, программное обеспечение, число, веб-страница

Автоматически созданное описание

Рис. 22 – Подключение файлов к проекту

Синтаксис файла Lab2\_top.sv:

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рис. 23 – Синтаксис файла Lab2\_top.sv

Выполним анализ и синтез проекта средствами QP и убедимся в правильности схемы средствами RTL Viewer:

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Автоматически созданное описание

Рис. 24 – Схема проекта в RTL Viewer

# Тестирование проекта

## Тестирование средствами ModelSim

### Создание тестового файла

Добавим тест первого класса для созданного проекта:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рис. 25 – Тестовый файл tb\_lab\_PD2\_top.sv

Укажем созданный файл в качестве основного тестового файла, который будет выполняться при симуляции средствами ModelSim:

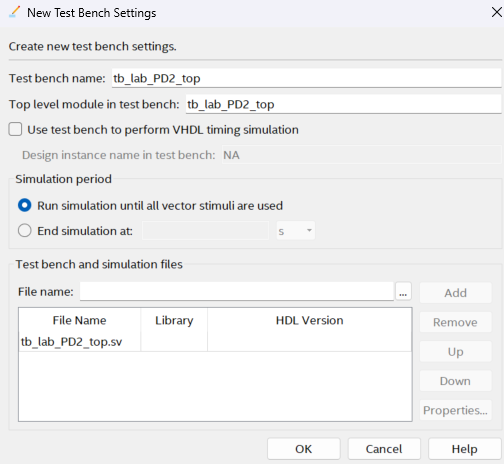


Рис. 26 – Тестовый файл tb\_lab\_PD2\_top.sv

### Симуляция средствами ModelSim

Выполним компиляцию проекта средствами ModelSim. Для этого запустим waveLab.do файл:

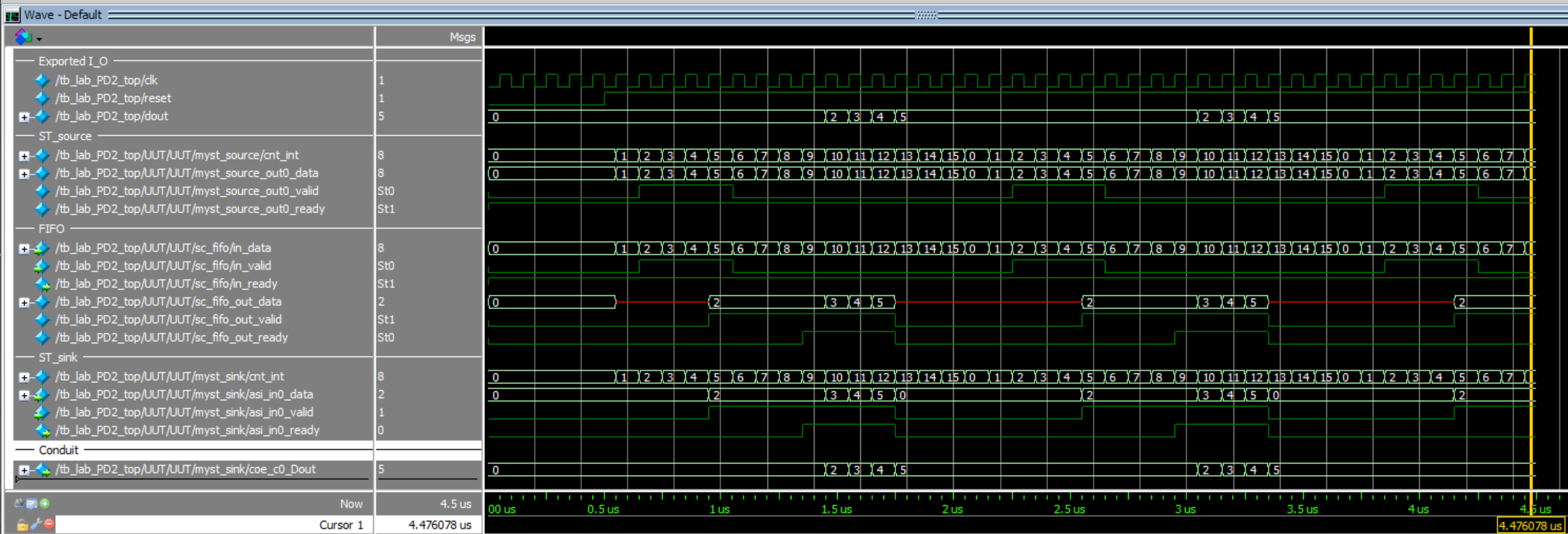


Рис. 27 – Моделирование проекта средствами ModelSim

**Компонент ST\_source:**

Видим, что у ST\_source идут данные (содержимое счётчика выводится в виде данных). Сигнал valid обрабатывает по 16 тактов за цикл, после чего сбрасывается и начинает отсчёт сначала.

**Компонент ST\_sink:**

Аналогично, у ST\_sink есть данные и сигнал, который показывает, что циклы размером до 16 тактов.

**Компонент FIFO:**

Сигнал in\_ready все время показывает, что он находится в состоянии ready, поскольку он не заполнен (количество элементов, которые можем передать = 4, это мы задавали, когда настраивали компонент на Рис. 7). Соответственно, в какой-то момент получаем данные, при этом (с маленькой задержкой) сигнал in\_valid становится = 1, тем самым показывая, что он считывает данные. После получения 4 элементов он снова переходит в значение 0. Через какое-то время на sc\_fifo\_out\_valid поступает сигнал 1, который показывает, что данные начали поступать и FIFO готов. Сигнал sc\_fifo\_out\_valid будет = 1 и данные будут поступать до того момента, пока приёмник не скажет, что он готов sc\_fifo\_out\_ready = asi\_in0\_ready = 0. Пока этот сигнал = 1 на sc\_fifo\_out\_data будет поступать 4 порции данных ( 2, 3, 4, 5, эти данные поступят только когда сигнал asi\_in0\_ready = 0). После этого FIFO будет опустошён (красная линия на временной диаграмме). Ждём следующего периода, чтобы снова заполнить FIFO, ждём пока приёмник будет готов принимать и так далее…

## Тестирование средствами Signal Tap II

### Создание файла для отладки

Создадим файл db\_lab\_PD2\_top.sv для отладки модуля lab\_PD2\_top:

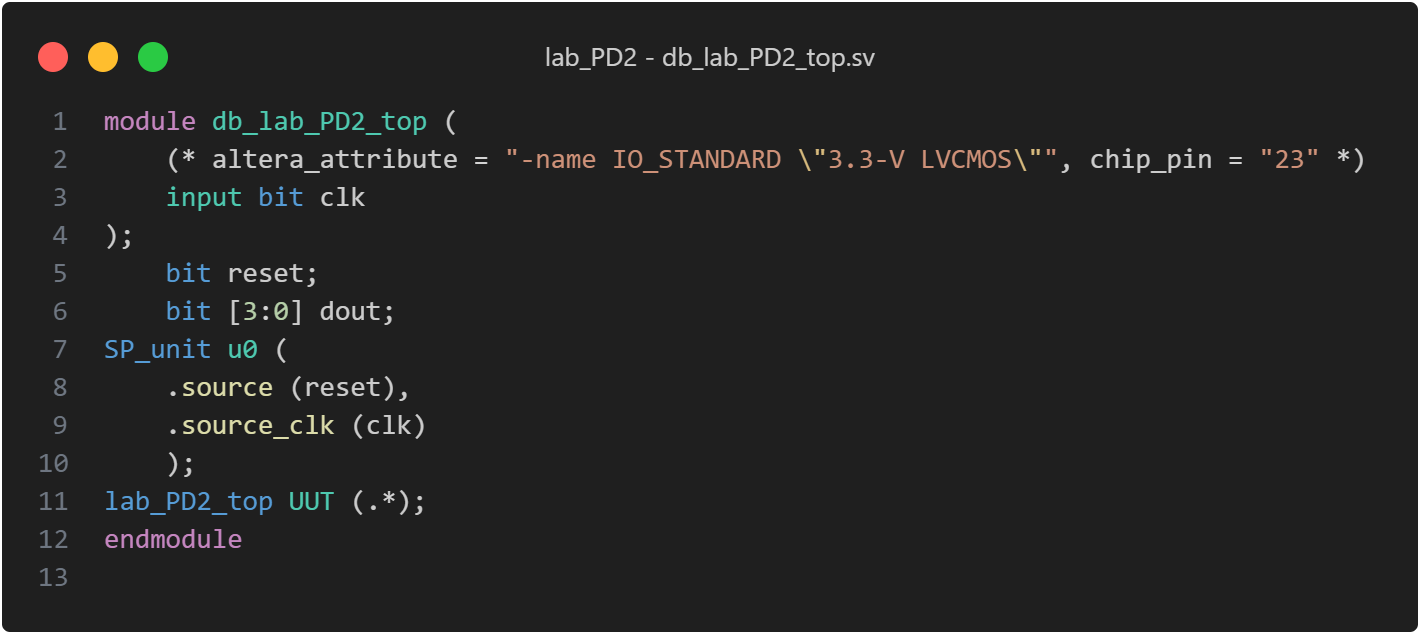


Рис. 28 – Файл для отладки модуля верхнего уровня

Создадим модуль ISSPE, укажем файл db\_lab\_PD3\_top.sv файлом верхнего уровня и убедимся в том, что схема, получаемая в результате компиляции, будет верной:

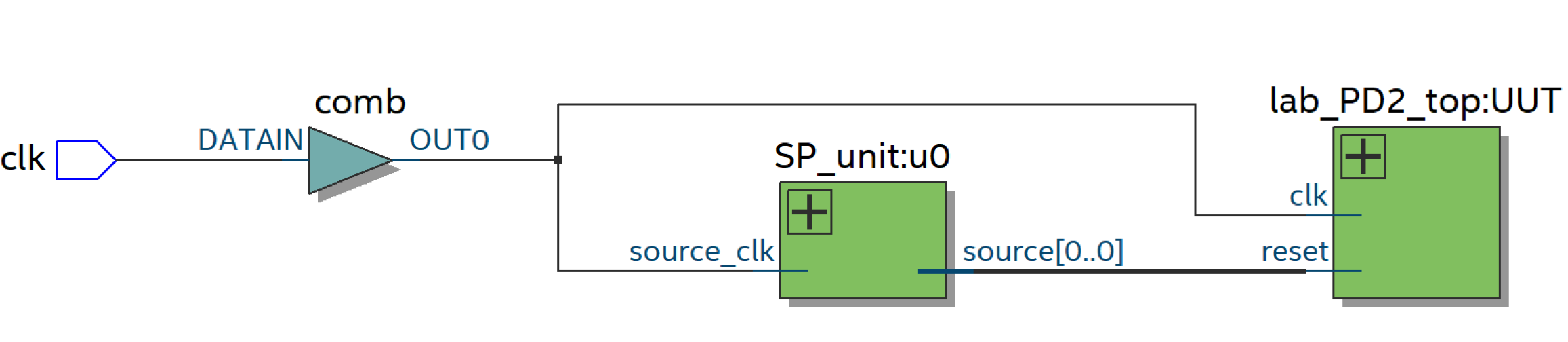


Рис. 29 – Схема проекта с добавлением SP\_unit в RTL Viewer

Как видно из схемы SP\_unit добавлен корректно.

### Настройка Signal Tap II

Добавим и настроим Signal Tap II, чтобы произвести отладку на плате:

Изображение выглядит как текст, число, линия, программное обеспечение

Автоматически созданное описание

Рис. 30 – Настройка окна Signal Tap II

\*Момент захвата данных синхронизируем по reset

### Тестирование на плате средствами Signal Tap II

Выполним полную компиляцию. В отчете о компиляции видно, что устройство удовлетворяет временным параметрам.

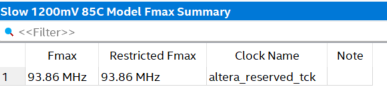


Рис. 31 – Временные характеристики устройства

Теперь запустим и выполним проверку корректности работы программы на плате. Выполним загрузку разработанного модуля на плату и запустим тестирование:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рис. 32 – Результат Signal Tap II

Полученная временная диаграмма совпадает с той, что была получена в ходе тестирования проекта средствами ModelSim (Рис. 27). Данные поступают и передаются на приёмник корректно.

Проверим, что в системных путях есть все необходимые файлы:

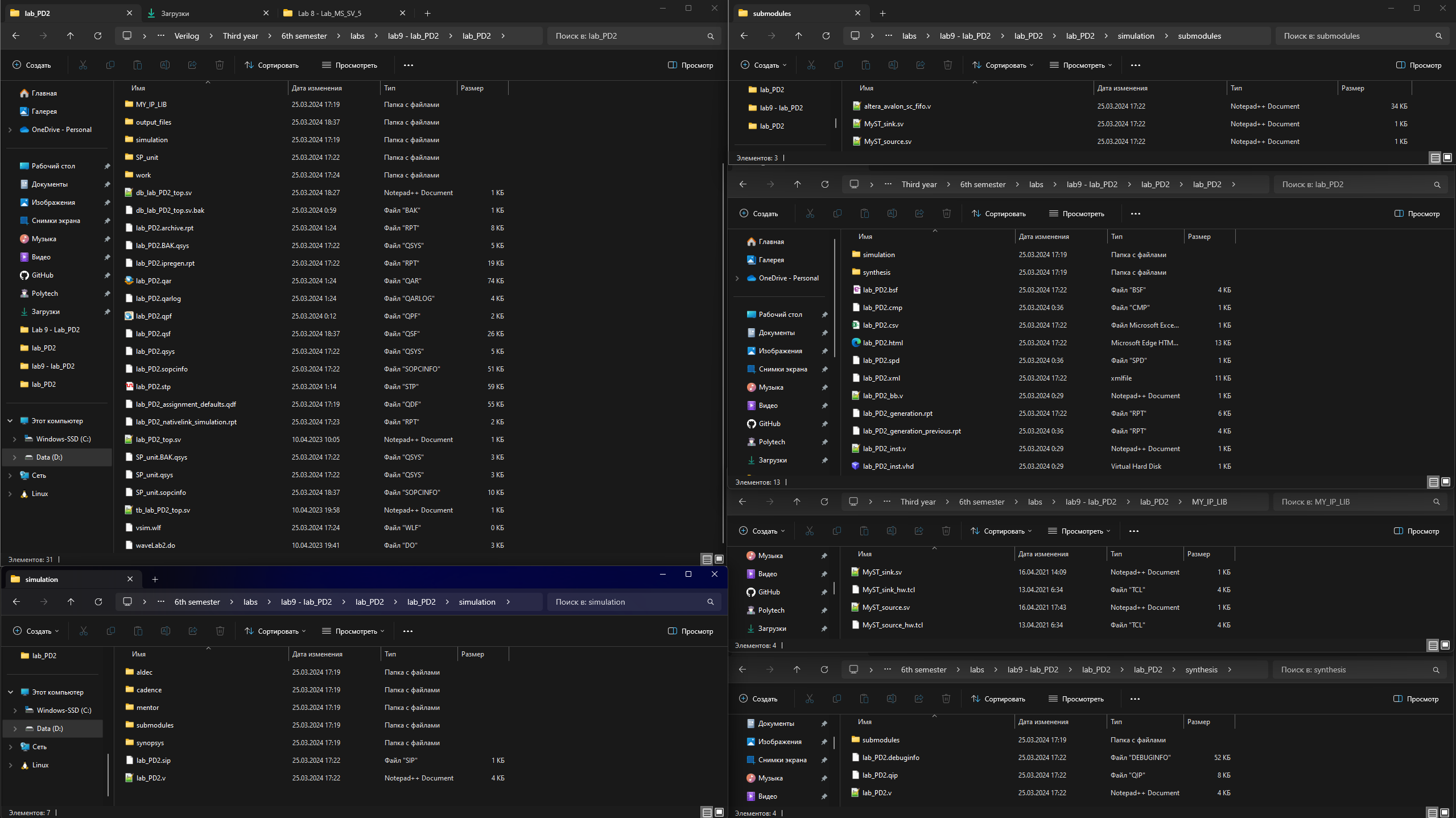


Рис. 33 – Анализ рабочей папки проекта

# Вывод

В ходе лабораторной работы была успешна система, представленная на Рис. 1.

Система демонстрирует эффективную передачу данных между источником (ST\_source) и приемником (ST\_sink) с использованием буферизации через FIFO. Модуль ST\_source функционирует как источник данных, генерируя выходные данные и уведомляя о их готовности через активацию сигнала aso\_out0\_valid. Сигнал aso\_out0\_valid периодически активируется, каждые 16 тактов, что указывает на появление новых данных для передачи.

С другой стороны, модуль ST\_sink работает как приемник данных, готовый принимать данные только при наличии готовности от модуля ST\_source. Он активизирует сигнал asi\_in0\_ready для уведомления об источнике о своей готовности принять данные и обновляет свой выходной сигнал coe\_c0\_Dout только при наличии действительных данных от ST\_source.

Промежуточное звено, FIFO, обеспечивает буферизацию данных между источником и приемником. Сигнал in\_valid активируется при поступлении новых данных для записи в FIFO, в то время как сигнал out\_valid указывает на готовность FIFO к передаче данных приемнику. FIFO успешно передает данные в ST\_sink только тогда, когда приемник готов принять их.

Таким образом, система обеспечивает синхронную и надежную передачу данных между источником и приемником, используя FIFO для управления потоком данных и согласования скоростей передачи.