Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчёт по лабораторной работе Lab\_PD3**

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](#_Toc162361320)

[2. Задание 5](#_Toc162361321)

[3. Ход работы 6](#_Toc162361322)

[3.1. Создание проекта 6](#_Toc162361323)

[Подготовка проекта 6](#_Toc162361324)

[Начало работы в PD 6](#_Toc162361325)

[3.2. Настройка сигналов 8](#_Toc162361326)

[Настройка clk 8](#_Toc162361327)

[Настройка sc\_fifo\_0 8](#_Toc162361328)

[Настройка MyST\_source\_0 и MyST\_sink\_0 8](#_Toc162361329)

[3.3. Подключение сигналов 9](#_Toc162361330)

[Подключение тактового сигнала 9](#_Toc162361331)

[Подключение сигнала Reset 9](#_Toc162361332)

[3.4. Подключение Avalon-MM интерфейсов 10](#_Toc162361333)

[3.5. Экспорт выводов 11](#_Toc162361334)

[3.6. Настройка my\_master 11](#_Toc162361335)

[3.7. Анализ системы 13](#_Toc162361336)

[Проверка блока 13](#_Toc162361337)

[Анализ с помощью Schematic 14](#_Toc162361338)

[Генерация системы 14](#_Toc162361339)

[Анализ подключенных файлов 15](#_Toc162361340)

[3.8. Подключение файлов к проекту 17](#_Toc162361341)

[4. Тестирование проекта 19](#_Toc162361342)

[4.1. Тестирование средствами ModelSim 19](#_Toc162361343)

[Создание тестового файла 19](#_Toc162361344)

[Симуляция средствами ModelSim 20](#_Toc162361345)

[4.2. Тестирование средствами Signal Tap II 21](#_Toc162361346)

[Создание файла для отладки 21](#_Toc162361347)

[Настройка Signal Tap II 21](#_Toc162361348)

[Тестирование на плате средствами Signal Tap II 22](#_Toc162361349)

[5. Дополнительное задание 23](#_Toc162361350)

[5.1. Изменение значений по варианту 23](#_Toc162361351)

[Изменение адресов slave 23](#_Toc162361352)

[Изменение компонента master 23](#_Toc162361353)

[5.2. Тестирование средствами ModelSim 24](#_Toc162361354)

[Симуляция средствами ModelSim 24](#_Toc162361355)

[5.3. Тестирование средствами Signal Tap II 24](#_Toc162361356)

[Тестирование на плате средствами Signal Tap II 24](#_Toc162361357)

[6. Вывод 26](#_Toc162361358)

**Список иллюстраций**

[Рис. 1 – Структура проекта 5](#_Toc162361273)

[Рис. 2 – Задания пути к библиотеке IP 6](#_Toc162361274)

[Рис. 3 – Исходное окно PD 6](#_Toc162361275)

[Рис. 4 – Добавление компонентов 7](#_Toc162361276)

[Рис. 5 – Переименование компонентов 7](#_Toc162361277)

[Рис. 6 – Настройка компонента clk 8](#_Toc162361278)

[Рис. 7 – Настройка компонента sc\_fifo 8](#_Toc162361279)

[Рис. 8 – Система после переименования компонентов 9](#_Toc162361280)

[Рис. 9 – Подключение тактового сигнала (1) 9](#_Toc162361281)

[Рис. 10 – Подключение тактового сигнала (2) 9](#_Toc162361282)

[Рис. 11 – Подключение сигнала Reset 10](#_Toc162361283)

[Рис. 12 – Подключение Avalon-MM интерфейсов 10](#_Toc162361284)

[Рис. 13 – Фиксация адресов 10](#_Toc162361285)

[Рис. 14 – Назначение правильных адресов для компонентов 11](#_Toc162361286)

[Рис. 15 – Проверка корректности адресов 11](#_Toc162361287)

[Рис. 16 – Экспорт выводов 11](#_Toc162361288)

[Рис. 17 – Настройка компонента my\_master 12](#_Toc162361289)

[Рис. 18 – Проверка системы 12](#_Toc162361290)

[Рис. 19 – Символ системы 13](#_Toc162361291)

[Рис. 20 – Анализ проблемных подключений 13](#_Toc162361292)

[Рис. 21 – Show System with QSYS Interconnect 14](#_Toc162361293)

[Рис. 22 – Schematic 14](#_Toc162361294)

[Рис. 23 – Предустановки окна Genreration 15](#_Toc162361295)

[Рис. 24 – Проверка успешности генерации HDL 15](#_Toc162361296)

[Рис. 25 – Файл компонента my\_master (1) 15](#_Toc162361297)

[Рис. 26 – Файл компонента my\_master (2) 16](#_Toc162361298)

[Рис. 27 – Файл компонента my\_slave 16](#_Toc162361299)

[Рис. 28 – Файл компонента my\_slave 17](#_Toc162361300)

[Рис. 29 – Подключение файлов к проекту 17](#_Toc162361301)

[Рис. 30 – Синтаксис файла lab\_PD3\_top.sv 18](#_Toc162361302)

[Рис. 31 – Схема проекта в RTL Viewer 18](#_Toc162361303)

[Рис. 32 – Тестовый файл tb\_lab\_PD2\_top.sv 19](#_Toc162361304)

[Рис. 33 – Тестовый файл tb\_lab\_PD2\_top.sv 19](#_Toc162361305)

[Рис. 34 – Моделирование проекта средствами ModelSim 20](#_Toc162361306)

[Рис. 35 – Файл для отладки модуля верхнего уровня 21](#_Toc162361307)

[Рис. 36 – Схема проекта с добавлением SP\_unit в RTL Viewer 21](#_Toc162361308)

[Рис. 37 – Сигналы логического анализатора 21](#_Toc162361309)

[Рис. 38 – Конечный автомат 22](#_Toc162361310)

[Рис. 39 – Настройка окна Signal Tap II 22](#_Toc162361311)

[Рис. 40 – Временные характеристики устройства 22](#_Toc162361312)

[Рис. 41 – Результат SignalTap II 22](#_Toc162361313)

[Рис. 42 – Изменение адресов для slave’ов 23](#_Toc162361314)

[Рис. 43 – Изменённое окно Address Map 23](#_Toc162361315)

[Рис. 44 – Изменённый модуль master 23](#_Toc162361316)

[Рис. 34 – Моделирование проекта средствами ModelSim 24](#_Toc162361317)

[Рис. 51 – Временные характеристики устройства 24](#_Toc162361318)

[Рис. 52 – Результат SignalTap II 25](#_Toc162361319)

# Задание

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

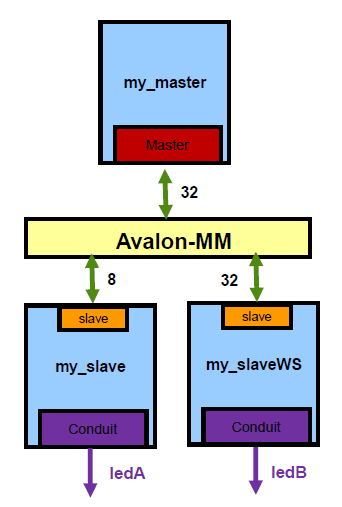


Рис. 1 – Структура проекта

Один из slave’ов – 32-битный, другой – 8-битный → понадобится динамическое выравнивание адресов.

Структура проекта выглядит следующим образом:

* Ведущий осуществляет запись словами по 32 бита
* Ведомый my\_slave–8 разрядный. Один цикл записи 32 разрядного слова от Ведущего будет преобразован системой соединений в 4 цикла записи 8 разрядными словами (на время этих четырёх циклов Ведущий будет приостановлен – он получит сигнал waiterquest от системы соединений).
* Ведомый my\_slaweWS–32 разрядный. Он, по получению от Ведущего сигнала write,выставляет (на один период тактового сигнала) сигнал waitrequest–приостанавливает Ведущего на один период тактового сигнала. Затем осуществляет запись данных.

# Ход работы

## Создание проекта

### Подготовка проекта

Создадим проект и зададим путь к библиотеке IP:

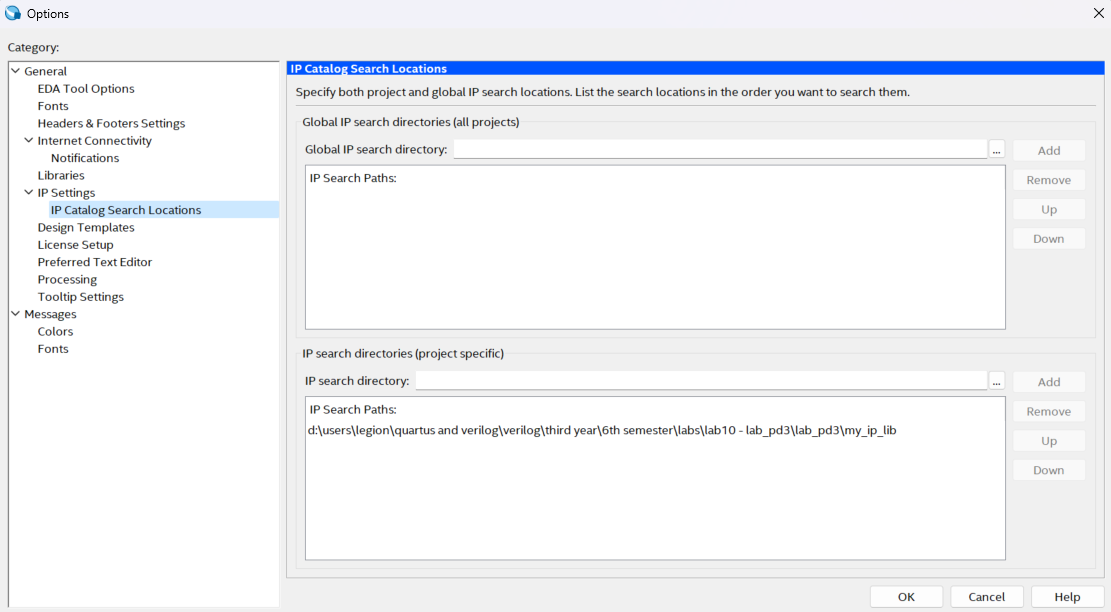


Рис. 2 – Задания пути к библиотеке IP

### Начало работы в PD

Откроем PD и сохраним систему:

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

Автоматически созданное описание

Рис. 3 – Исходное окно PD

Добавим компоненты: my\_master, my\_slave, my\_slaveWS. 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. 4 – Добавление компонентов

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

Переименуем компоненты (уберём постфикс `\_0`):

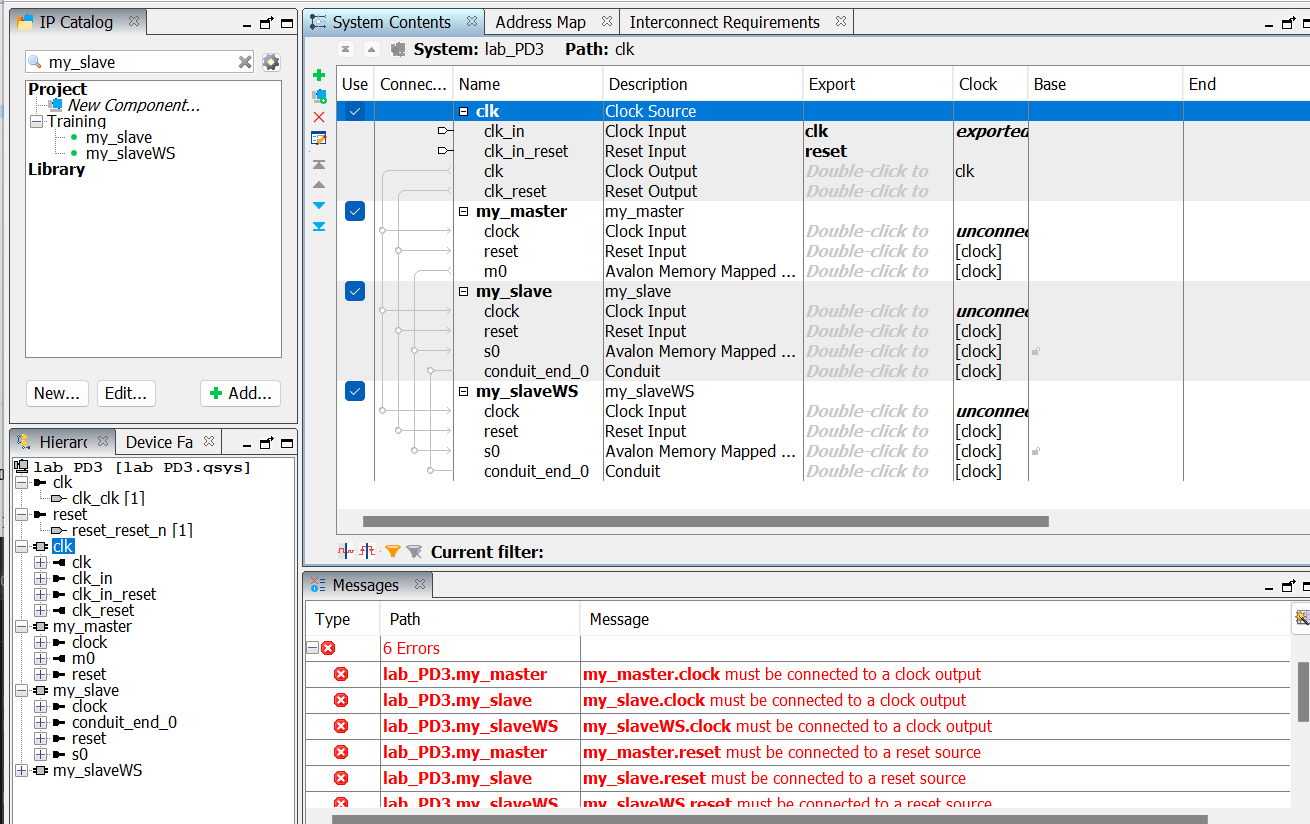


Рис. 5 – Переименование компонентов

## Настройка сигналов

### Настройка clk

Зададим значение Reset synchronous edges = Deassert

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, веб-страница

Автоматически созданное описание

Рис. 6 – Настройка компонента clk

### Настройка sc\_fifo\_0

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 сигналов

### Подключение тактового сигнала

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

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

Автоматически созданное описание

Рис. 9 – Подключение тактового сигнала (1)

Выполним Filter → Clock and Reset Interfaces и убедимся, что соединения выполнены корректно:

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

Автоматически созданное описание

Рис. 10 – Подключение тактового сигнала (2)

### Подключение сигнала Reset

Выполним System → Create Global Reset Network и убедимся, что система выглядит корректно, сигнал Reset подключен:

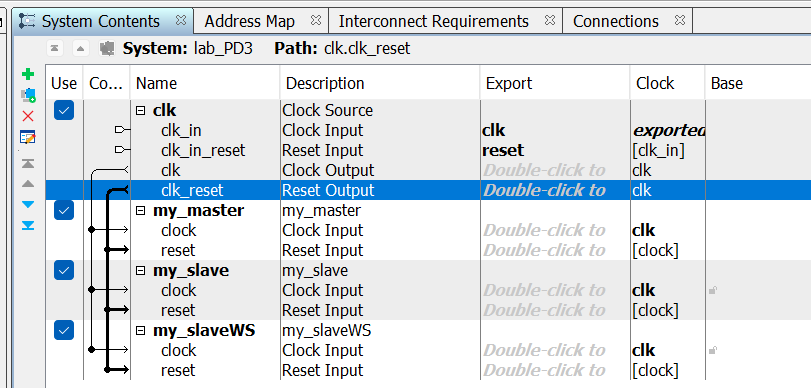


Рис. 11 – Подключение сигнала Reset

## Подключение Avalon-MM интерфейсов

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

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рис. 12 – Подключение Avalon-MM интерфейсов

Заметим, что в окне Msgs высвечивается ошибка, указывающая на то, что базовые адреса заданы неправильно. Поменяем адреса, проверим, что они верны, и зафиксируем их (нажать символ замка у каждого из адресов):

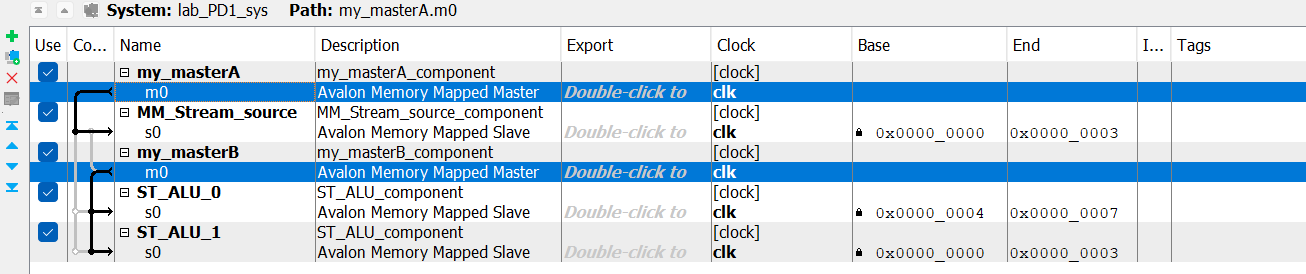


Рис. 13 – Фиксация адресов

Откроем окно закладок View → Address Map. В окне появятся Ведущие (master) и ведомые (slave) шины, т. е. столбцы и строки Avalon MM соответственно. Проверим, что они выведены корректно. Для этого компоненту my\_slave.s0 назначим базовый адрес = 0 и выполним System → Assign Base Address:

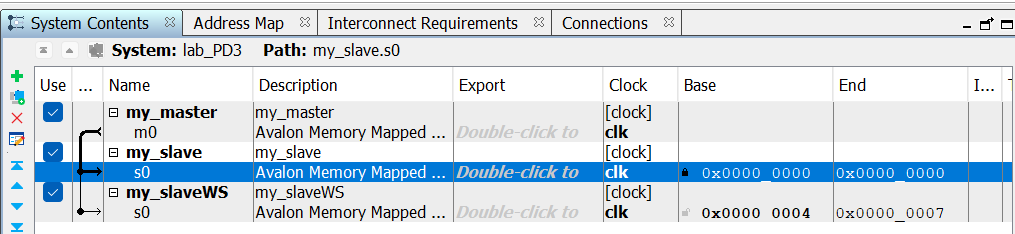


Рис. 14 – Назначение правильных адресов для компонентов

Вкладка Address Map будет выглядеть следующим образом:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рис. 15 – Проверка корректности адресов

Запомним базовые адреса:

* my\_slave.s0 = 0,
* my\_slaveWS.s0 = 0.

\*Позже он будут указываться при настройке модулей my\_slave и my\_slveWS соответственно

## Экспорт выводов

Проведём экспорт выводов путём задания имён для выделенных модулей в столбце Export (значение dout\_a и dout\_b в соотвыетствующем столбце выделенных строк):

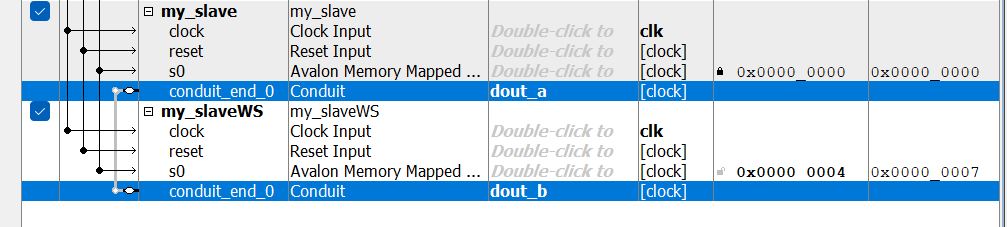


Рис. 16 – Экспорт выводов

## Настройка my\_master

Зададим следующие значения:

* address\_1 = 0
* data\_2 = 287454020, это соответствует 32’h11223344, что будет удобно при запуске моделирования и последующем рассмотрении результатов на временной диаграмме
* address\_2 = 4
* data\_2 = 61166, что соответствует 32’h0000eeee (принцип задания такого числа аналогичен data\_1, для удобства анализа передачи: big или little endian)

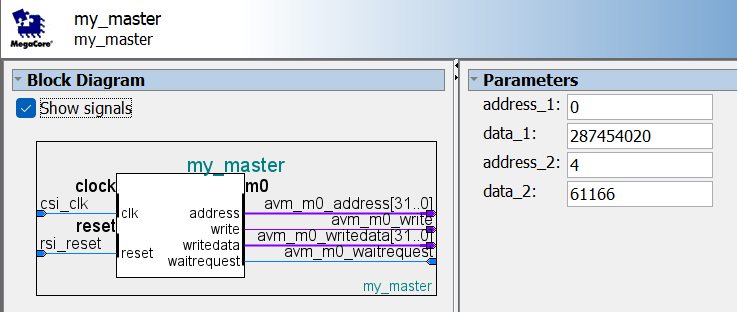


Рис. 17 – Настройка компонента my\_master

Заметим, что значения address\_1 и address\_2 соответствуют значениям slave’ов.

Убедимся в том, что система выглядит корректно и в поле Messages нет ошибок:

Изображение выглядит как текст, снимок экрана, число, дисплей

Автоматически созданное описание

Рис. 18 – Проверка системы

## Анализ системы

### Проверка блока

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

Изображение выглядит как текст, снимок экрана, дисплей, число

Автоматически созданное описание

Рис. 19 – Символ системы

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

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

Автоматически созданное описание

Рис. 20 – Анализ проблемных подключений

Выполним команду System → Show System with PD Interconnect (Show System with QSYS Interconnect). Проверим, был добавлен только модуль mm\_interconnect\_0.

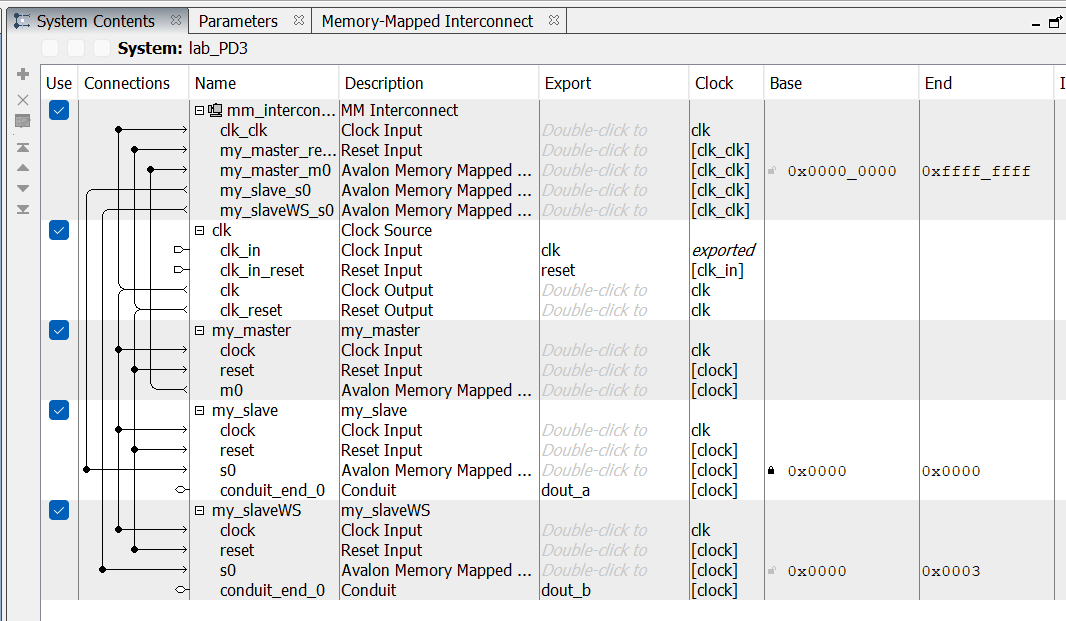


Рис. 21 – Show System with QSYS Interconnect

### 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:

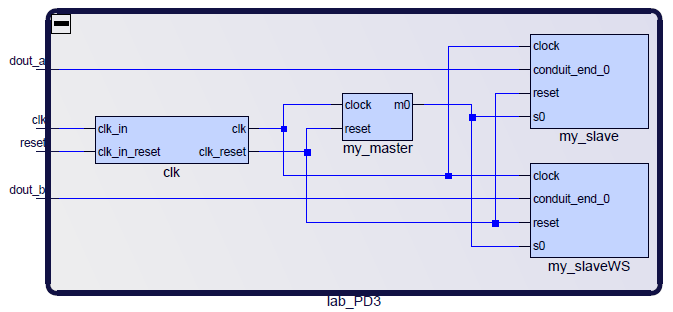


Рис. 22 – Schematic

### Генерация системы

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

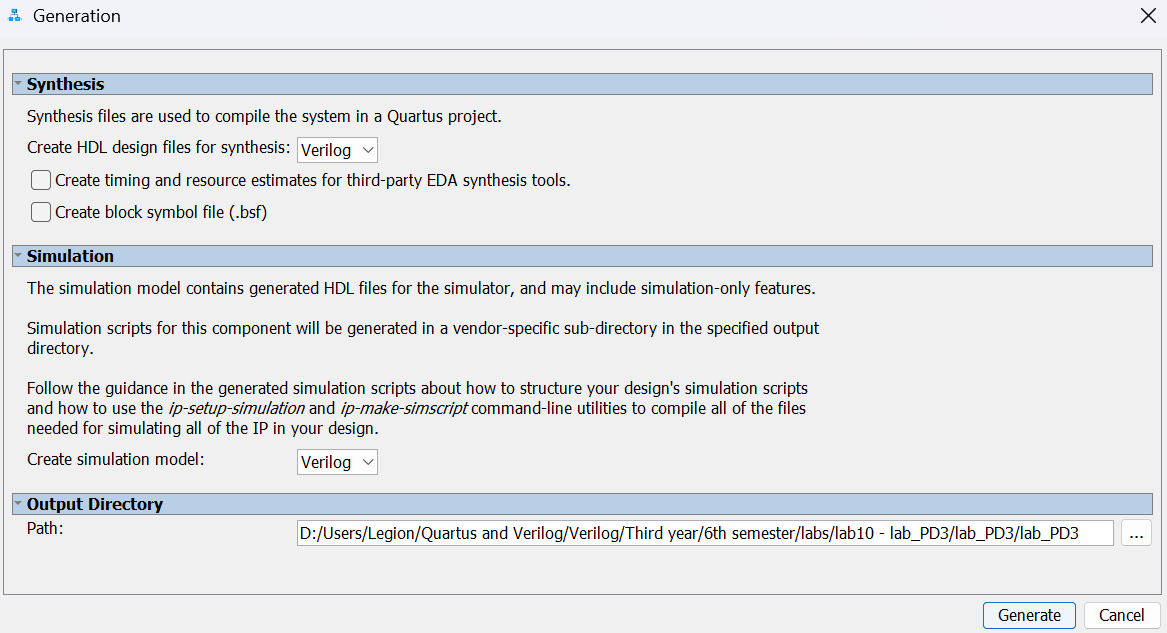


Рис. 23 – Предустановки окна Genreration

Удостоверимся в том, что генерация прошла успешно:

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

Автоматически созданное описание

Рис. 24 – Проверка успешности генерации HDL

### Анализ подключенных файлов

Таким образом, получили 3 файла: my\_master.sv, my\_slave.sv и my\_slaveSV:

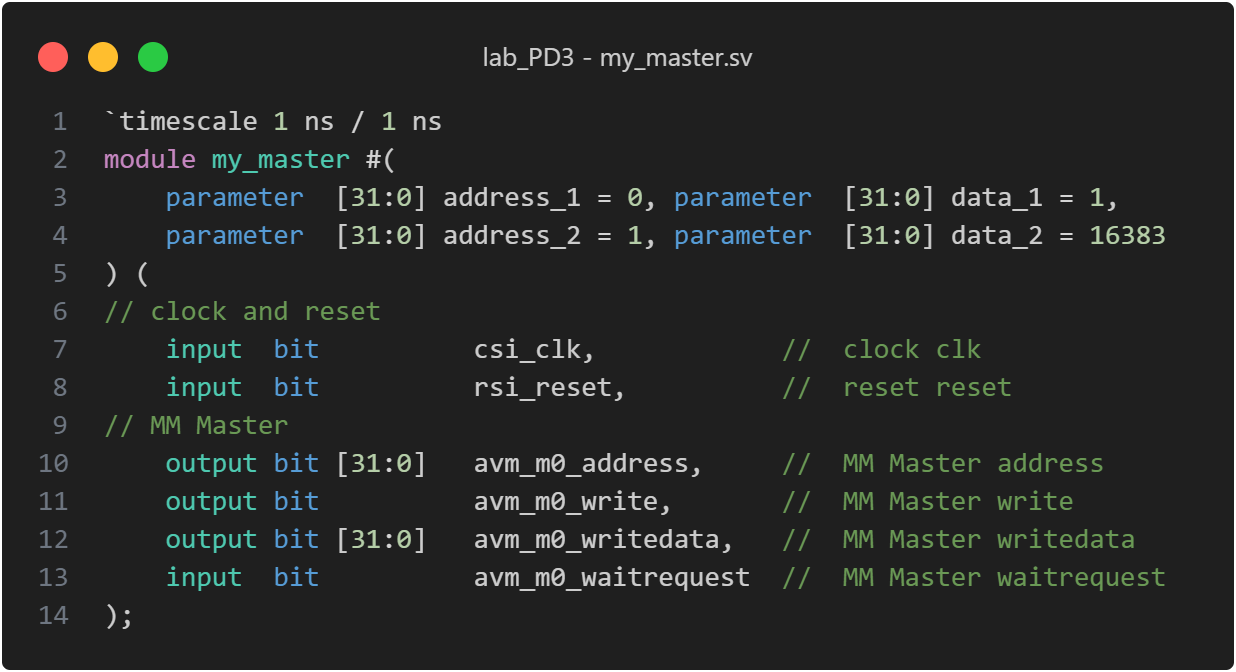


Рис. 25 – Файл компонента my\_master (1)

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

Автоматически созданное описание

Рис. 26 – Файл компонента my\_master (2)

Файл my\_maser содержит анализ waitrequest (формируются сигналы либо по параметрам, которые мы задаём, либо значение = 255 в промежутках, которое отвечает за «пустой обмен», значение именно такое, чтобы было нагляднее при моделировании средствами Signal Tap II), интерфейсы clock, reset и master.

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

Автоматически созданное описание

Рис. 27 – Файл компонента my\_slave

Модуль my\_slave (8-битный) – берёт 8-битные данные, у него есть интерфейс MM Slave. Но он НЕ умеет формировать сигнал waitrequest, он занулён. Осуществляет по сигналу write запись тех данных, которые приходят с шины writedata во внутренний регистр.

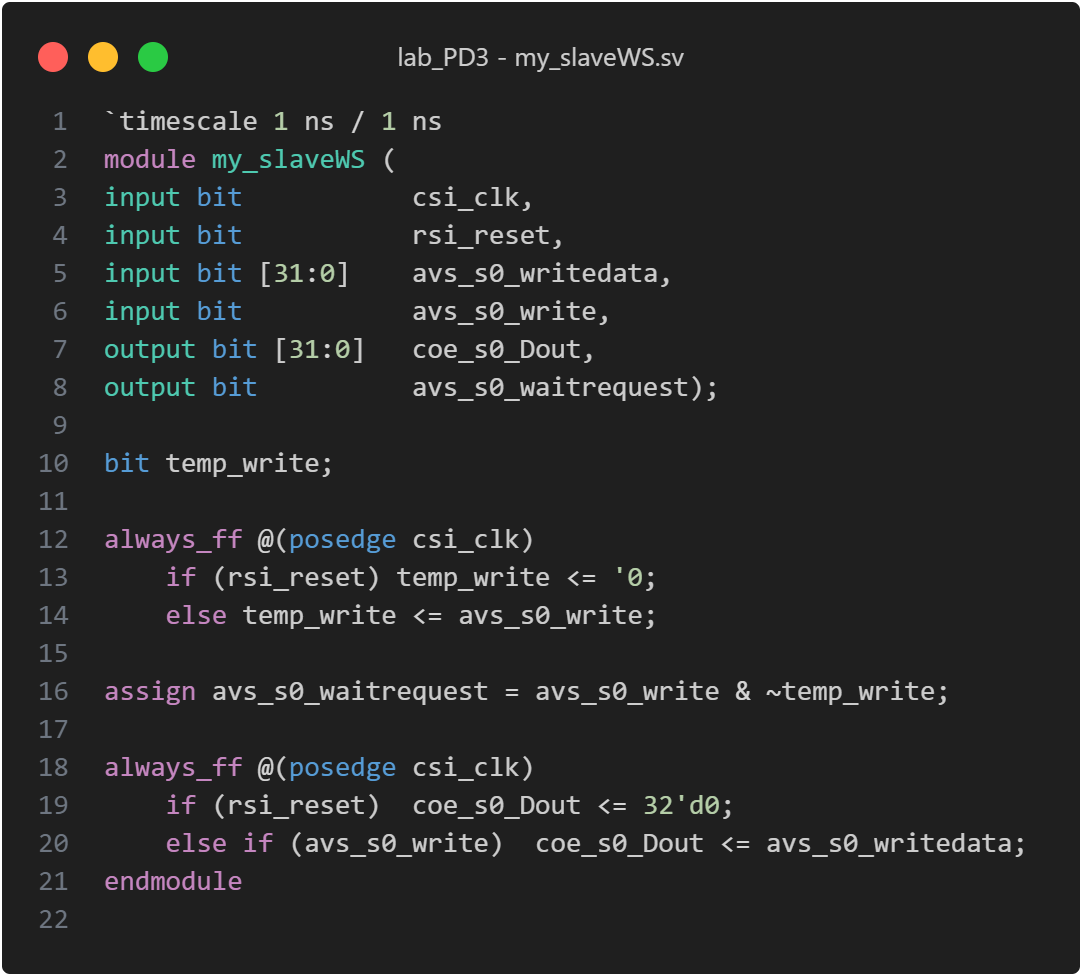


Рис. 28 – Файл компонента my\_slave

Модуль my\_slaveWS (32-битный) – умеет формировать сигнал waitrequest, получив запрос write, он его записывает в регистр, и только на следующем такте убирает ,этот сигнал, т. е. сделана задержка в 1 такт.

## Подключение файлов к проекту

Подключим файлы к проекту в Quartus

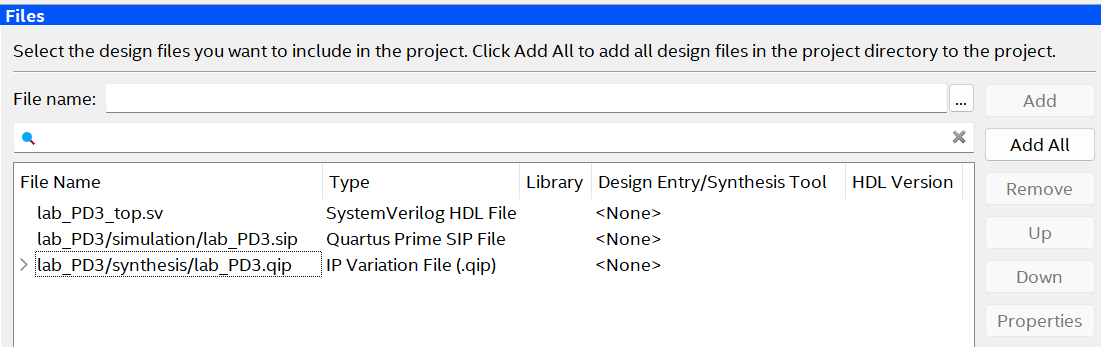


Рис. 29 – Подключение файлов к проекту

Синтаксис файла Lab2\_top.sv:

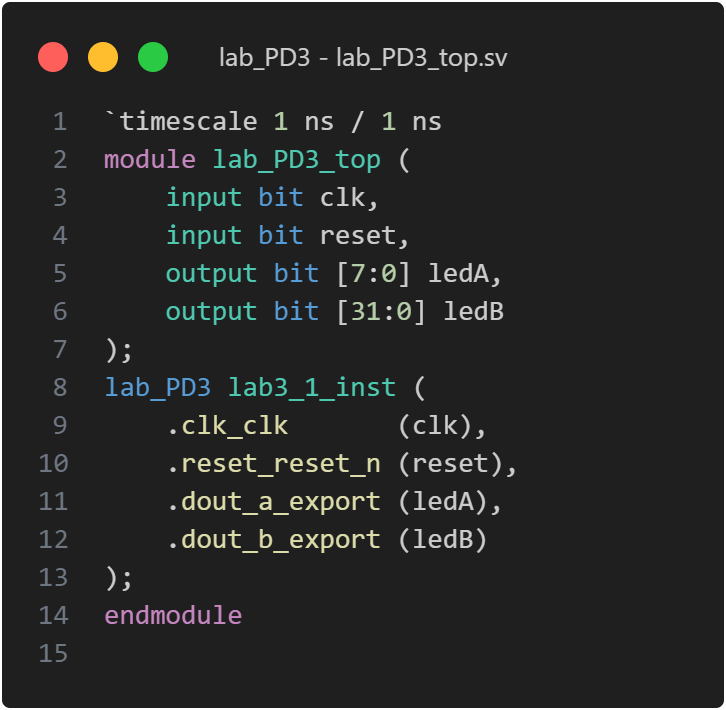


Рис. 30 – Синтаксис файла lab\_PD3\_top.sv

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

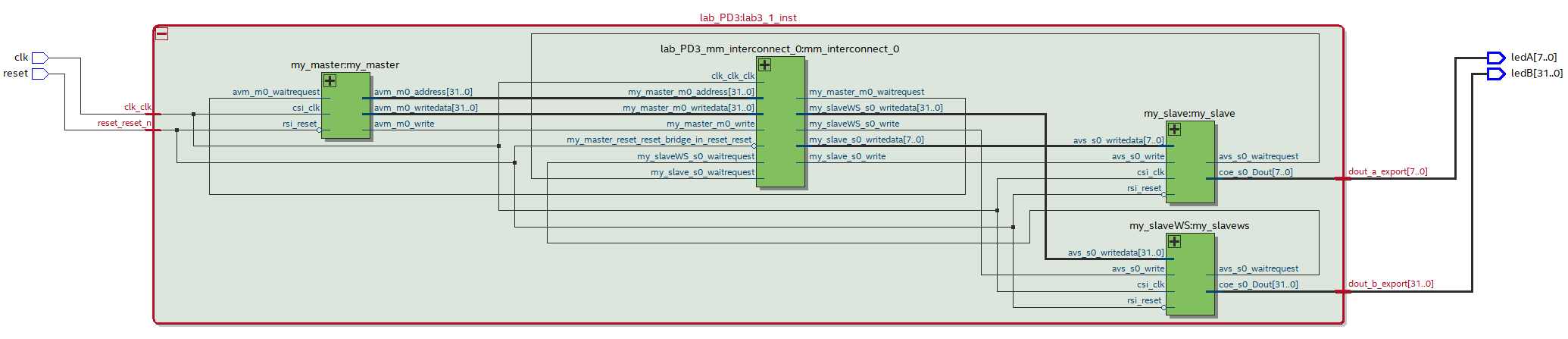


Рис. 31 – Схема проекта в RTL Viewer

Можем увидеть, что полученная в RTL Viewer схема совпадает с той, что была задана по условию (в зелёном блоке отображается тот фрагмент системы, который был создан средствами PD).

# 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. 32 – Тестовый файл tb\_lab\_PD2\_top.sv

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

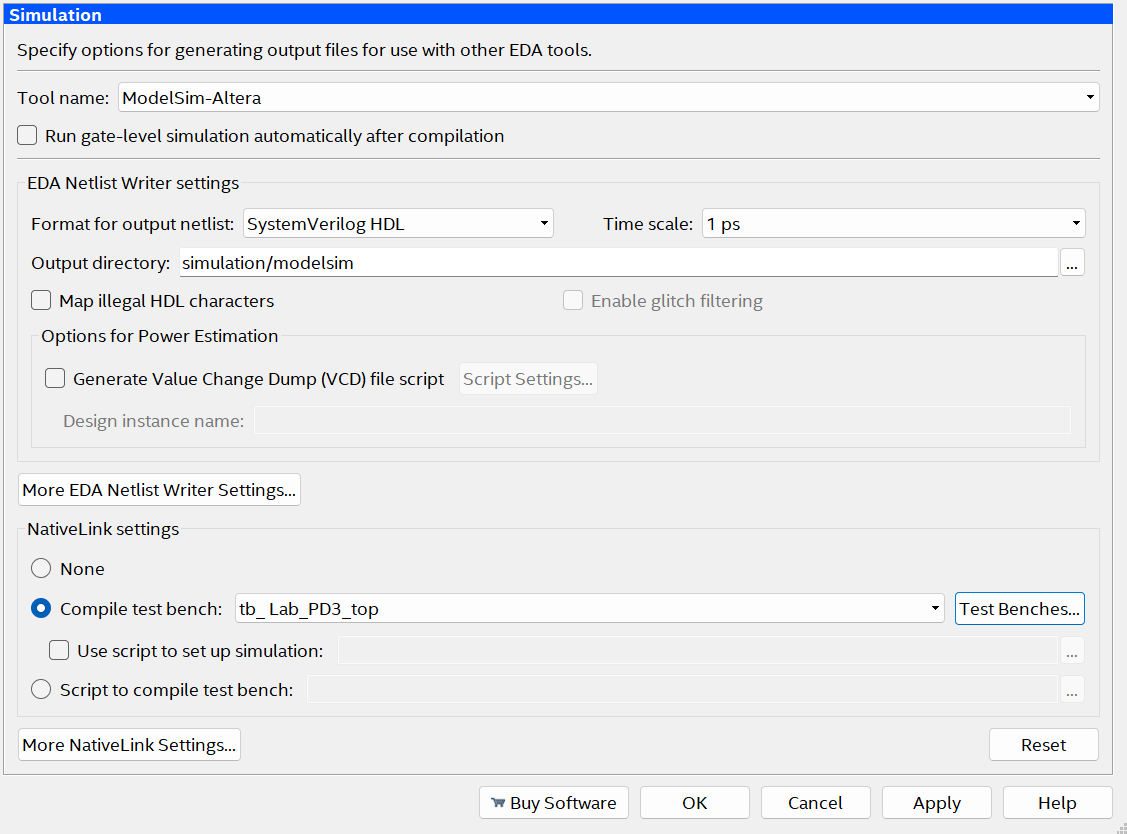


Рис. 33 – Тестовый файл tb\_lab\_PD2\_top.sv

### Симуляция средствами ModelSim

Выполним компиляцию проекта средствами ModelSim. Для этого запустим waveLab2.do файл:

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

Автоматически созданное описание

Рис. 34 – Моделирование проекта средствами ModelSim

**master**

Формирует записи по неким адресам. В начале это тот самый адрес ff, который является заполнителем, пока не поступит обычный адрес. Затем, он начинает формировать запись по 0-му адресу (проходит инициализация + задержка после начала моделирования) По 0-му адресу записываются значения, указанные в writedata (32’h11223344). 32 разряда выкинуто

**my\_slave (8-битный)**

32 разряда было выкинуто, но, поскольку этот slave – 8-битный, у него появляется сигнал wr1D и начинают формироваться данные (сначала младшие 44, потом 33, потом 22, наконец - 11). Во время этого у master возникает сигнал wr1D (wait request). Это означает, что в то время, пока байты 44, 33, 22, 11 последовательно передаются, мастер заторможен. Этот сигнал, Wr1D, сформировала система меж соединений (она увидела, что было подано 4 байта = 32 бита и надо притормозить master, пока производится запись этих байтов в 8-битный my\_slave).

**my\_slaveWS (32-битный)**

Далее осуществляется «пустой» цикл (со значениями ff). После этого master осуществляет запись по адресу = 4 (значение writedata = 32’h0000eeee). Здесь заметим, что 8-битный my\_slave получает данные в виде `ee`, но у него нету никакого признака write, в отличие от 2-го (32-битного) my\_slaveWS. Соответственно, только 2-ой slave получит значения, но он зафиксирует эти значения только в тот момент, когда master перестанет подавать сигнал waitrequest. В этот момент произойдёт фиксация этих значений и master перейдёт в состояние ended.

Таким образом, происходит последовательная передача в режиме little endian 32-битных данных с выставлением waitrequest’а. Происходит выставление waitrequest’а также и в режиме big endian, но уже по инициативе slave (32-битного).

## Тестирование средствами Signal Tap II

### Создание файла для отладки

Создадим файл db\_lab\_PD3\_top.sv для отладки модуля lab\_PD3\_top:

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

Автоматически созданное описание

Рис. 35 – 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:

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Автоматически созданное описание

Рис. 36 – Схема проекта с добавлением SP\_unit в RTL Viewer

Как видно из схемы SP\_unit добавлен корректно.

### Настройка Signal Tap II

Выберем сигналы для логического анализатора:

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

Автоматически созданное описание

Рис. 37 – Сигналы логического анализатора

Добавим к сигналам конечный автомат:

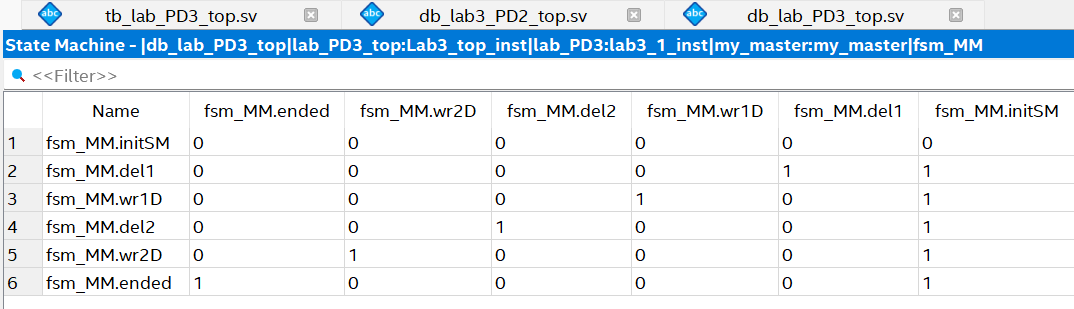


Рис. 38 – Конечный автомат

Настроим окно Signal Tap II так, как показано на рисунке ниже:

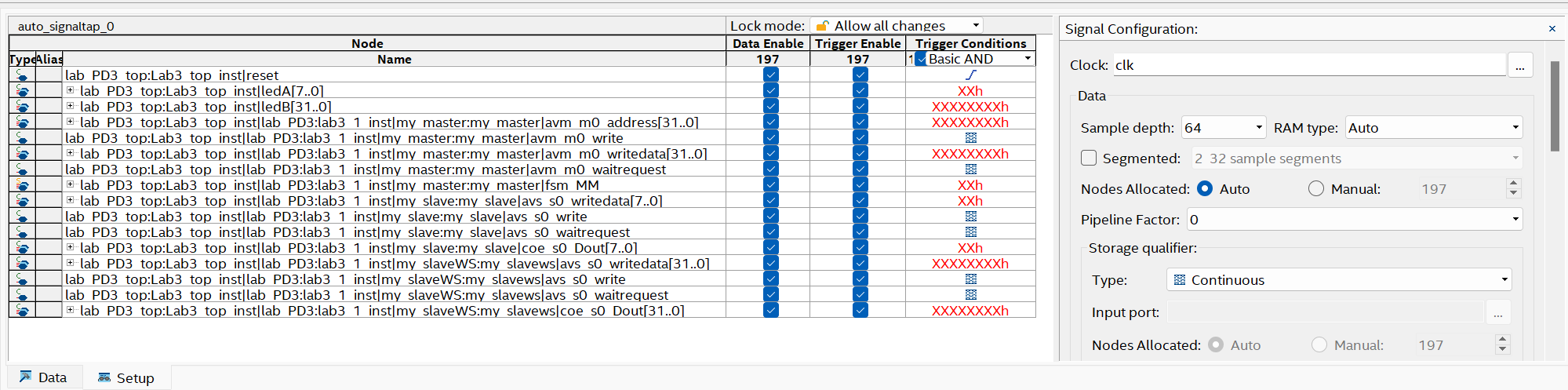


Рис. 39 – Настройка окна Signal Tap II

### Тестирование на плате средствами Signal Tap II

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

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рис. 40 – Временные характеристики устройства

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

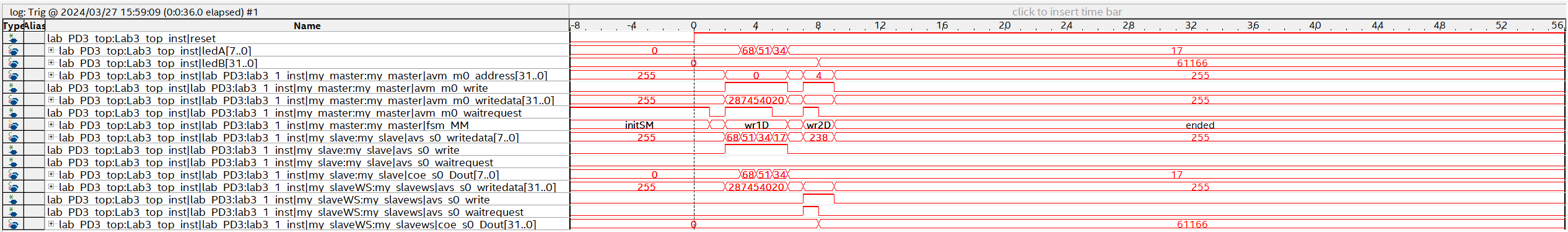


Рис. 41 – Результат SignalTap II

Полученная временная диаграмма совпадает с той, что была получена в ходе тестирования проекта средствами ModelSim (Рис. 34). Данные поступают и передаются на приёмник корректно.

# Дополнительное задание

## Изменение значений по варианту

### Изменение адресов slave

В соответствии с вариантом в списке группы (6 вариант) установим значения адресов для модулей my\_slave = 2410 = 1816 и my\_slaveWS = 4810 = 3016 соответственно.

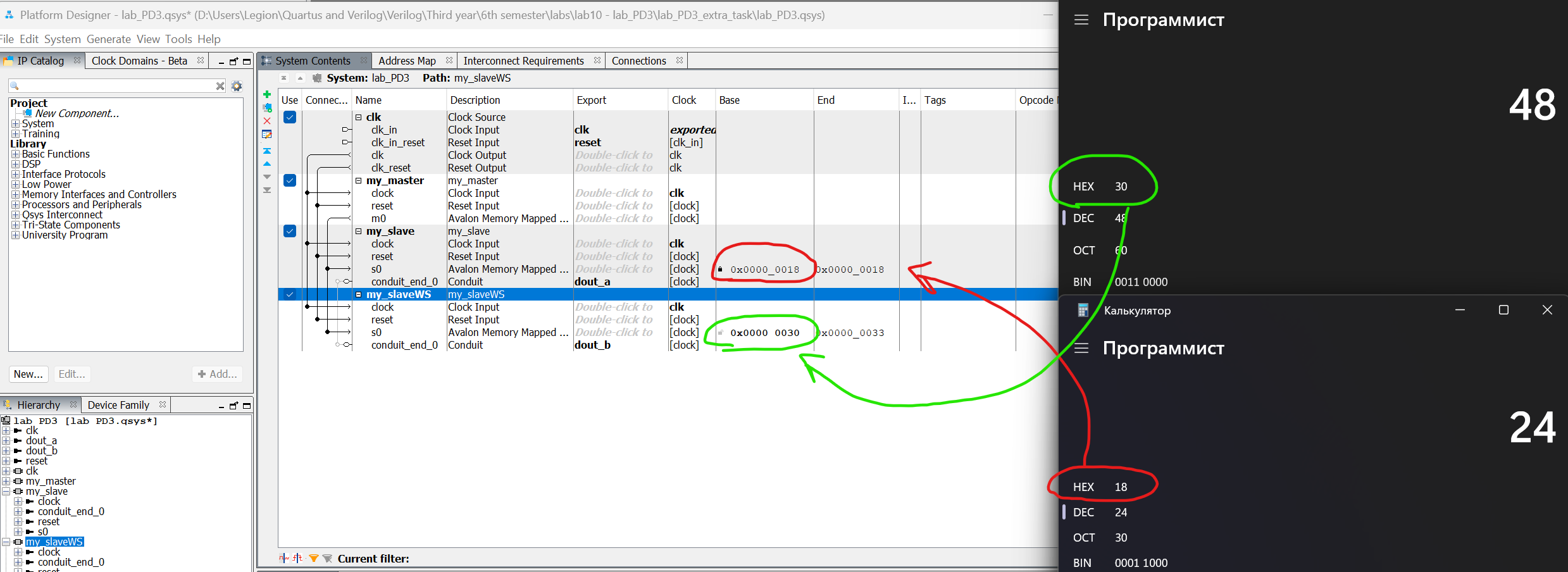


Рис. 42 – Изменение адресов для slave’ов

Окно Adress Map будет выглядеть следующим образом:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рис. 43 – Изменённое окно Address Map

### Изменение компонента master

Так как модуль master работает вместе со slave’ами, необходимо также поменять адреса внутри самого компонента master,установив следующие значения:

* address\_1 = 2410
* data\_1 = 287454020 (не изменяется)
* address\_2 = 4810
* data\_2 = 6 (номер варианта)

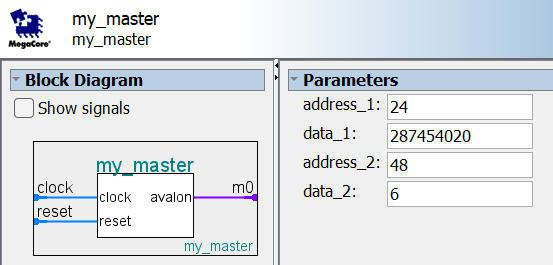


Рис. 44 – Изменённый модуль master

## Тестирование средствами ModelSim

### Симуляция средствами ModelSim

Выполним компиляцию проекта средствами ModelSim. Для этого запустим waveLab2.do файл:

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

Автоматически созданное описание

Рис. 34 – Моделирование проекта средствами ModelSim

Заметим, что временная диаграмма корректно обрабатывает изменённые адреса и значения, запись wr1D теперь происходит по адресу 2410 = 1816 (происходит запись данных data\_1 = 11223344, это значение мы оставляли таким же, как и в первой симуляции), а запись wr2D по адресу 4810 = 3016 (осуществляется запись данных data\_2 = 6, это значение согласно номеру варианта). Для удобства добавлены курсоры с пометками о том, что происходит на каждом из этапов, так, например становится понятно, что запись одного байта длится 0.1 мкс (или 100 ns). Отсюда следует, что цикл записи my\_slave = 0.4 мкс (или 400 ns), my\_slaveWS = 0.2 мкс (или 200 ns). Также, видим, что задержка обработки для некоторых сигналов составляет 0.1 мкс (или 100 ns).

## Тестирование средствами Signal Tap II

### Тестирование на плате средствами Signal Tap II

Настройки Signal Tap II остаются такими же, единственное, что изменяется, некоторые адреса сигналов, так как для дополнительного задания был создан отдельный проект.

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

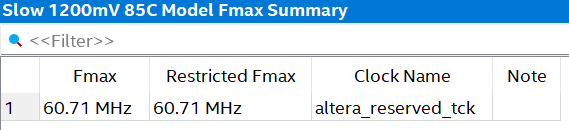


Рис. 51 – Временные характеристики устройства

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

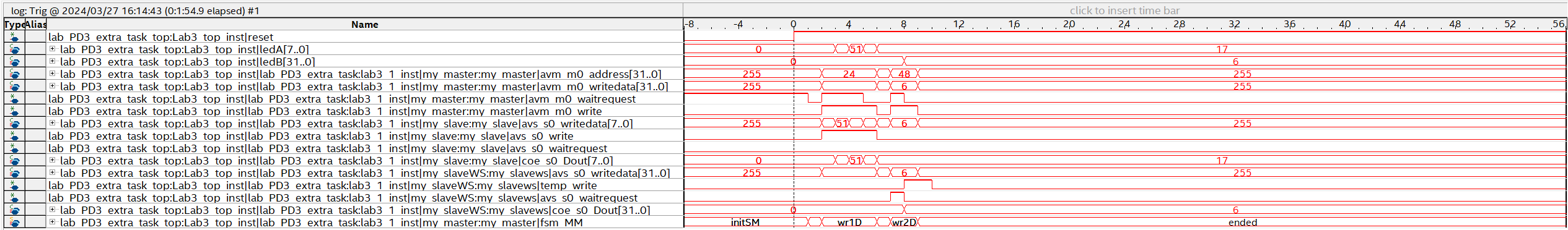
****

Рис. 52 – Результат SignalTap II

Полученная временная диаграмма совпадает с той, что была получена в ходе тестирования проекта средствами ModelSim (Рис. 34). Данные поступают и передаются на приёмник корректно.

# Вывод

В ходе лабораторной работы успешно разработан и скомпилирован проект в среде ModelSim, включающий в себя master-модуль и два slave-модуля: my\_slave и my\_slaveWS. Анализ временной диаграммы подтвердил корректность передачи данных между модулями.

В `my\_slave`, являющемся 8-битным устройством, подается 32 бита данных, что требует организации очереди для их записи. Именно поэтому возникает необходимость в использовании сигнала wr1D, чтобы обработать эти данные порциями по 8 бит.

С другой стороны, второй slave (my\_slaveWS) является 32-битным устройством, что позволяет обрабатывать 32-битные данные сразу, без необходимости в организации очереди. Поэтому для него используется сигнал wr2D, и данные обрабатываются непосредственно.

Внесенные изменения в адреса и данные были успешно учтены в проекте, и временная диаграмма верно отображает процесс обмена данными между master- и slave-модулями, учитывая их особенности и требования к обработке данных.