Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчёт по лабораторной работе № 10**

Дисциплина: Автоматизация проектирования дискретных  
устройств (на английском языке).

Выполнил студент гр. 5130901/10101 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.Л. Симоновский (подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.П. Антонов (подпись)

“14” апреля 2024 г.

Санкт-Петербург

2024

Оглавление

[1. Список иллюстраций: 2](#_Toc164008074)

[2. Алгоритм работы проекта: 3](#_Toc164008075)

[3. Ход работы: 3](#_Toc164008076)

[4. Вывод: 12](#_Toc164008077)

# Список иллюстраций:

[Рис. 2.1. Схема разрабатываемого устройства. 3](#_Toc164008081)

[Рис. 3.1. RTL Viewer для my\_master. 5](#_Toc164008082)

[Рис. 3.2. RTL Viewer для my\_slave. 5](#_Toc164008083)

[Рис. 3.3. RTL Viewer модуля my\_Dslave. 6](#_Toc164008084)

[Рис. 3.4. Platform Designer. 6](#_Toc164008085)

[Рис. 3.5. Component Type для my\_master. 7](#_Toc164008086)

[Рис. 3.6. Files для my\_master. 7](#_Toc164008087)

[Рис. 3.7. Выбор файла симуляции для my\_master. 7](#_Toc164008088)

[Рис. 3.8. Signals & Interfaces для my\_master. 7](#_Toc164008089)

[Рис. 3.9. Signal & Interfaces для slave. 7](#_Toc164008090)

[Рис. 3.10. Signal & Interfaces для Dslave. 8](#_Toc164008091)

[Рис. 3.11. Элементы в Platform Designer. 8](#_Toc164008092)

[Рис. 3.12. Настройка сигналов clk. 8](#_Toc164008093)

[Рис. 3.13. Подключения в модуле. 8](#_Toc164008094)

[Рис. 3.14. Символ системы. 9](#_Toc164008095)

[Рис. 3.15. Анализ проблемных подключений. 9](#_Toc164008096)

[Рис. 3.16. System with PD Interconnect. 9](#_Toc164008097)

[Рис. 3.17. Schematic. 9](#_Toc164008098)

[Рис. 3.18. Подключение файлов к проекту. 10](#_Toc164008099)

[Рис. 3.19. RTL Viewer. 10](#_Toc164008100)

[Рис. 3.20. Результат тестирования. 11](#_Toc164008101)

[Рис. 3.21. Результат тестирования со смещением. 11](#_Toc164008102)

[Рис. 3.22. RTL Viewer. 12](#_Toc164008103)

[Рис. 3.23. Signal Tap. 12](#_Toc164008104)

# Алгоритм работы проекта:

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

Изображение выглядит как текст, снимок экрана, диаграмма, Прямоугольник

Автоматически созданное описание

Рис. 2.1. Схема разрабатываемого устройства.

# Ход работы:

Выполним создание проекта со стандартными настройками, после чего создадим описания модулей my\_master, my\_slave и my\_Dslave.

Модуль my\_master будет выглядеть следующим образом:

Изображение выглядит как текст, снимок экрана, меню

Автоматически созданное описание

Модуль my\_master функционирует как главное устройство в системе Avalon Memory-Mapped (MM). Он управляет передачей данных от мастера к другим компонентам.

Модуль работает на основе конечного автомата (FSM), который имеет четыре состояния: initSM, del1, wr1D, del2.

* initSM: Начальное состояние.
* del1: Задержка для ожидания данных от мастера Avalon MM (чтобы отделить циклы записи по шине, это не обязательно, но так будет наглядно при просмотре waveform).
* wr1D: Ожидание завершения операции записи данных от мастера Avalon MM.
* del2: Дополнительная задержка после завершения операции записи.

RTL Viewer выглядит следующим образом:

Изображение выглядит как линия, диаграмма, снимок экрана, График

Автоматически созданное описание

Рис. 3.1. RTL Viewer для my\_master.

Модуль my\_slave выглядит следующим образом:

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Модуль my\_slave в интерфейсе Avalon Memory-Mapped (MM) функционирует как подчинённое устройство, принимая данные от мастера и передавая их через выходной сигнал coe\_s0\_Dout. Он использует тактовый сигнал csi\_clk для синхронизации операций и сигнал сброса rsi\_reset для инициализации внутренних состояний. Модуль содержит 8-битный регистр данных rg\_DATA, который обновляется при каждом положительном фронте csi\_clk, если активирован сигнал записи avs\_s0\_write. При активации сигнала сброса регистр rg\_DATA сбрасывается в ноль. Данные, хранящиеся в регистре rg\_DATA, передаются через выходной сигнал coe\_s0\_Dout. Сигнал avs\_s0\_waitrequest всегда устанавливается в ноль, что означает отсутствие запроса на ожидание со стороны подчинённого устройства.

Посмотрим, как выглядит диаграмма этого модуля в RTL Viewer:

Изображение выглядит как линия, диаграмма, текст, снимок экрана

Автоматически созданное описание

Рис. 3.2. RTL Viewer для my\_slave.

Далее разработаем модуль my\_Dslave:

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Модуль my\_Dslave является простым устройством в системе, которое принимает данные от мастера Avalon MM и передаёт их через интерфейс Conduit. Когда мастер отправляет данные, my\_Dslave сохраняет их во внутреннем регистре и затем передаёт через выходной порт coe\_s0\_Dout через интерфейс Conduit без каких-либо изменений. Это позволяет эффективно передавать данные от мастера Avalon MM к другим частям системы, используя my\_Dslave в качестве посредника, без необходимости дополнительной обработки или изменений данных.

RTL Viewer выглядит следующим образом:

Изображение выглядит как линия, диаграмма, снимок экрана, текст

Автоматически созданное описание

Рис. 3.3. RTL Viewer модуля my\_Dslave.

Перейдем в Platform Disigner:

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

Автоматически созданное описание

Рис. 3.4. Platform Designer.

Добавим в PD ранее созданные компоненты:

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

Автоматически созданное описание

Рис. 3.5. Component Type для my\_master.

Изображение выглядит как текст, Шрифт, число, линия

Автоматически созданное описание

Рис. 3.6. Files для my\_master.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рис. 3.7. Выбор файла симуляции для my\_master.

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

Автоматически созданное описание

Рис. 3.8. Signals & Interfaces для my\_master.

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

Автоматически созданное описание

Рис. 3.9. Signal & Interfaces для slave.

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

Автоматически созданное описание

Рис. 3.10. Signal & Interfaces для Dslave.

Теперь добавим модули в систему:

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

Автоматически созданное описание

Рис. 3.11. Элементы в Platform Designer.

Выполним настройку моудля clk:

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

Автоматически созданное описание

Рис. 3.12. Настройка сигналов clk.

Подключим все компоненты друг к другу:

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

Автоматически созданное описание

Рис. 3.13. Подключения в модуле.

Произведем анализ разработанного модуля:

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

Автоматически созданное описание

Рис. 3.14. Символ системы.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рис. 3.15. Анализ проблемных подключений.

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

Рис. 3.16. System with PD Interconnect.

Изображение выглядит как текст, снимок экрана, дисплей, диаграмма

Автоматически созданное описание

Рис. 3.17. Schematic.

Как видим все выполнено в соответствии с заданием.

Выполним генерацию и подключим получившиеся файлы к проекту:

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

Автоматически созданное описание

Рис. 3.18. Подключение файлов к проекту.

Далее создадим файл верхнего уровня:

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

Автоматически созданное описание

RTL Viewer данного проекта приведен ниже:

Изображение выглядит как текст, диаграмма, линия, снимок экрана

Автоматически созданное описание

Рис. 3.19. RTL Viewer.

Выполним тестирование разработанного файла, используя следующий тестовый модуль:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Запустим тестовый файл и получим следующий результат:

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

Автоматически созданное описание

Рис. 3.20. Результат тестирования.

Как мы видим, данные доставляются в соответствии с заданными адресами. А при несовпадении адреса отправляются в my\_Dslave, где внутренний счетчик увеличивается на 1, что соответствует заданию.

Поменяем сигнал data\_d – смещение записываемого значения, относительно адреса на 18 (в соответствии с вариантом и повторим запуск тестируемого модуля:

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Рис. 3.21. Результат тестирования со смещением.

Как мы видим, все работает корректно. Теперь перейдем к тестированию непосредственно на плате. Для этого разработаем следующий модуль:

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

Автоматически созданное описание

RTL\_Viewer разработанного модуля выглядит следующим образом:

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Автоматически созданное описание

Рис. 3.22. RTL Viewer.

Выполним компиляцию разработанного модуля и посмотрим на результат в Signal Tap:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рис. 3.23. Signal Tap.

Полученный результат совпадает со схемой, полученной в Model Sim, что свидетельствует о корректности разработанного устройства.

# Вывод:

В ходе выполнения лабораторной работы была создана структура проекта с использованием Platform Designer. Этот инструмент предоставляет графический интерфейс для интеграции и настройки IP-блоков, что упрощает и ускоряет процесс разработки в сравнении с ручным написанием кода.

Преимущества Platform Designer включают в себя:

1. Ускорение разработки: Платформа предлагает готовые IP-блоки, которые могут быть легко интегрированы в проект, сокращая время разработки.
2. Упрощение процесса: Использование графического интерфейса позволяет избежать сложностей в написании и отладке кода, особенно для начинающих разработчиков.
3. Модульность и повторное использование: IP-блоки могут быть использованы в разных проектах, что способствует повторному использованию кода и упрощает поддержку проектов.
4. Визуализация системы: Платформа предоставляет графическое представление структуры проекта, что облегчает понимание и анализ системы.

Таким образом, использование Platform Designer обеспечивает эффективное управление проектом, ускоряет его разработку и повышает его надежность за счет готовых компонентов и удобного интерфейса.