Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчёт по курсовой работе CR\_1**

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. Задание 4](#_Toc166158062)

[1.1. Цель работы 4](#_Toc166158063)

[2. Ход решения 5](#_Toc166158064)

[2.1. Модуль GEN 5](#_Toc166158065)

[2.1.1. Создание модуля 5](#_Toc166158066)

[2.1.2. Тестирование 7](#_Toc166158067)

[2.2. Модуль RaF 8](#_Toc166158068)

[2.2.1. Создание модуля 8](#_Toc166158069)

[2.2.2. Тестирование 9](#_Toc166158070)

[2.3. Модуль CR\_1 10](#_Toc166158071)

[2.3.1. Создание модуля 10](#_Toc166158072)

[2.3.2. Тестирование 12](#_Toc166158073)

[2.4. Создание модуля для тестирования на плате 14](#_Toc166158074)

[2.5. Настройка Signal Tap II 15](#_Toc166158075)

[2.6. Тестирование на плате средствами Signal Tap II 16](#_Toc166158076)

[2.7. Создание модуля имплементации и запуск на плате 17](#_Toc166158077)

[3. Вывод 18](#_Toc166158078)

**Список иллюстраций**

[Рис. 1 – Структура разрабатываемого устройства 4](#_Toc166158079)

[Рис. 2 – Модуль интерфейса lab\_MS\_SV5\_interface 5](#_Toc166158080)

[Рис. 3 – Модуль gen 6](#_Toc166158081)

[Рис. 4 – RTL Viewer модуля gen 6](#_Toc166158082)

[Рис. 5 – Тестовый файл первого класса для модуля gen 7](#_Toc166158083)

[Рис. 6 – Моделирование тестового файла средствами ModelSim (gen\_tb) 7](#_Toc166158084)

[Рис. 7 – Модуль RaF 8](#_Toc166158085)

[Рис. 8 – RTL Viewer модуля RaF 8](#_Toc166158086)

[Рис. 9 – Тестовый файл первого класса для модуля RaF 9](#_Toc166158087)

[Рис. 10 – Моделирование тестового файла средствами ModelSim (RaF\_tb) 9](#_Toc166158088)

[Рис. 11 – Модуль CR\_1 10](#_Toc166158089)

[Рис. 12 – Настройка модуля FIFO 11](#_Toc166158090)

[Рис. 13 – RTL Viewer модуля CR\_1 11](#_Toc166158091)

[Рис. 14 - Тестовый файл первого класса для модуля CR\_1 12](#_Toc166158092)

[Рис. 15 – Тестирование модуля CR\_1 до сигнала full 13](#_Toc166158093)

[Рис. 16 – Тестирования модуля CR\_1 до сигнала empty 13](#_Toc166158094)

[Рис. 17 – Тестирование модуля CR\_1 до максимального значения генератора 13](#_Toc166158095)

[Рис. 18 – Тестирования модуля CR\_1 на сигнале RST 13](#_Toc166158096)

[Рис. 19 – Модуль db для тестирования на плате 14](#_Toc166158097)

[Рис. 20 – RTL Viewer для тестового файла модуля CR\_1 14](#_Toc166158098)

[Рис. 21 – Настройка окна Signal Tap II 15](#_Toc166158099)

[Рис. 22 – Мнемоническая таблица 15](#_Toc166158100)

[Рис. 23 – Временные характеристики устройства 15](#_Toc166158101)

[Рис. 24 – Появление сигнала full при тестировании на плате 16](#_Toc166158102)

[Рис. 25 – Появление сигнала empty при тестировании на плате 16](#_Toc166158103)

[Рис. 26 – Появление значения 127 (максимального) при тестировании на плате 16](#_Toc166158104)

[Рис. 27 – Отключение работы при тестировании на плате 16](#_Toc166158105)

[Рис. 28 – Сброс при тестировании на плате 16](#_Toc166158106)

[Рис. 29 – Модуль имплементации 17](#_Toc166158107)

[Рис. 30 – RTL Viewer для impl\_CR\_1 17](#_Toc166158108)

# Задание

## Цель работы

Разработать устройство, структура которого будет выглядеть седеющим образом:

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Автоматически созданное описание

Рис. – Структура разрабатываемого устройства

Он состоит из следующих элементов:

* GEN – генератор случайных чисел.
* FIFO – IP модуль очереди.
* RaF – модуль для поиска максимального числа, среди поданных на вход.
* lab\_3s – модуль для вывода результата модуля RaF на 7-сегментный индикатор.

Для этого модуля разработать тестовый файл, программу отладки на плате и итоговый проект для интеграции устройства.

# Ход решения

## Модуль GEN

### Создание модуля

Начнем разработку данного проекта с модуля GEN. Он должен работать по следующему алгоритму:

Если ( и и ) выполняется:

Модуль формирует сигнал , работает генератор псевдослучайных чисел (данные выдаются на выход )

Если условие ( и и ) не выполняется:

Модуль формирует сигнал , генератор псевдослучайных чисел находится в состоянии ожидания (данные выдаются на выход )

В качестве генератора псевдослучайных чисел используется модуль LFSR\_8\_7\_6\_4\_3\_2\_1\_F, разработанный ранее в ходе лабораторных:

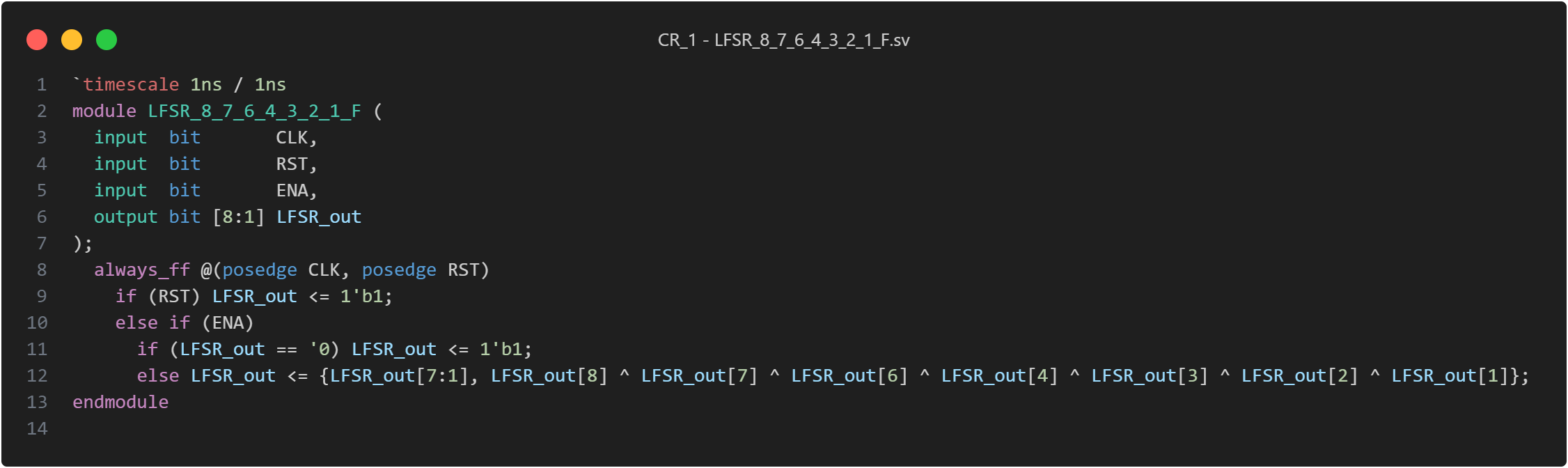


Рис. – Модуль интерфейса lab\_MS\_SV5\_interface

В свою очередь модуль GEN является лишь контролирует, когда модуль будет работать, а когда нет. Он будет выглядеть так, как показано на рис. 3 ниже.

Этот модуль принимает на вход сигналы, которые показаны на рис. 1. Далее объявляется модуль LFSR\_8\_7\_6\_4\_3\_2\_1\_F. Сигнал RST у него будет принимать постоянное значение нуля, т. е. никогда не будет сброшен, благодаря ENA будет выполняться остановка генерации, а на выход data будет подаваться результат.

Логика включения ENA и сигнала wrreq написана в блоке always\_ff и соответствует поставленному заданию.

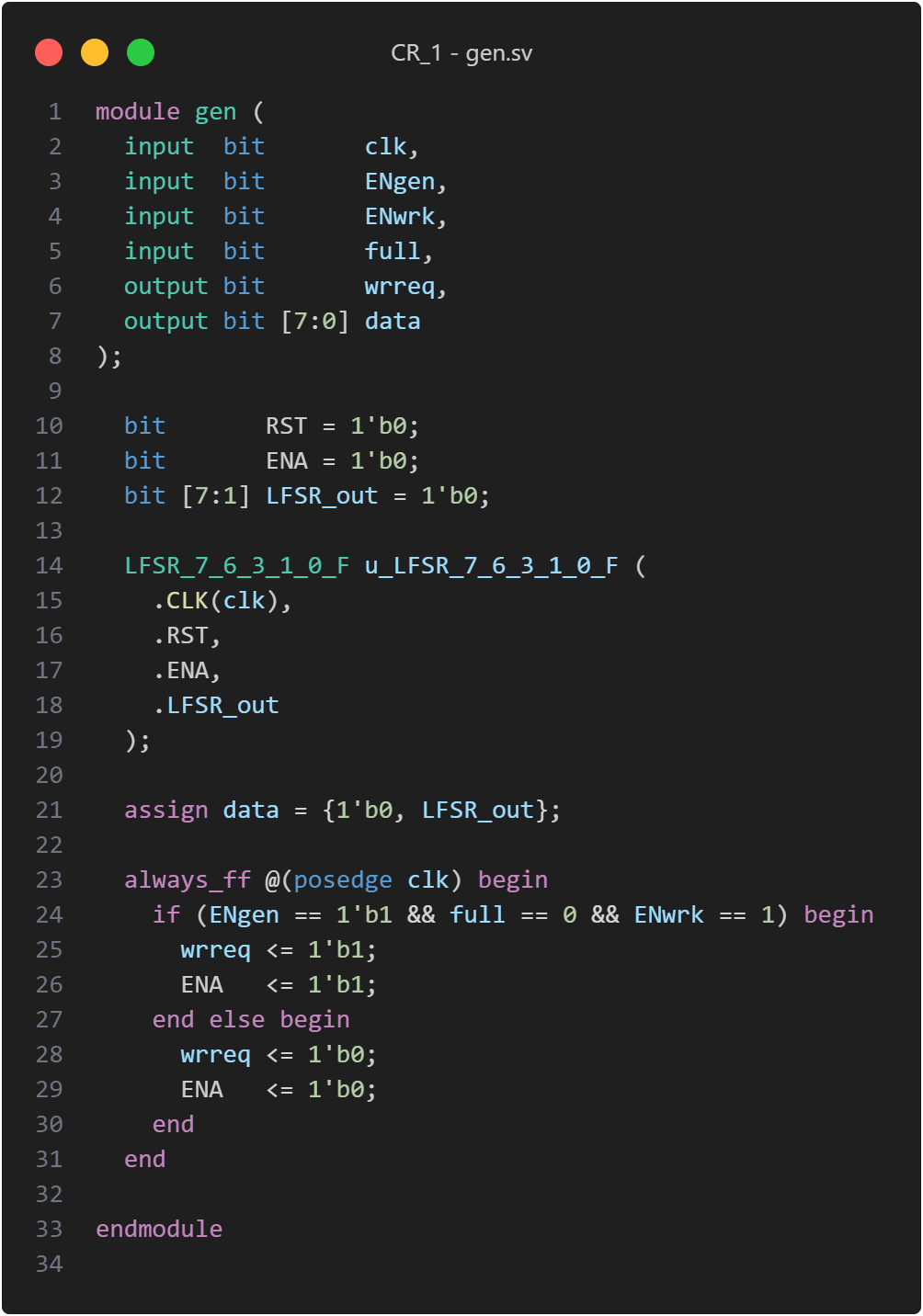


Рис. – Модуль gen

Выполним компиляцию и посмотрим на получившуюся RTL схему:

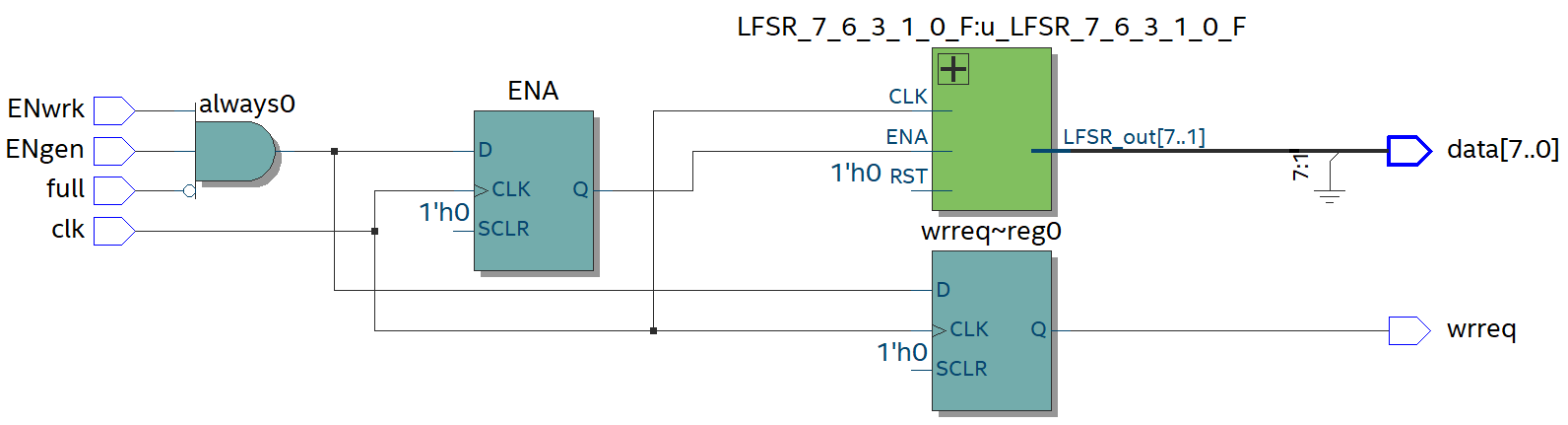


Рис. 4 – RTL Viewer модуля gen

Она соответствует ожиданиям, теперь необходимо провести тестирование данного модуля, для этого разработаем тест первого класса, он будет выглядеть следующим образом:

### Тестирование

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

Автоматически созданное описание

Рис. – Тестовый файл первого класса для модуля gen

В этом тесте мы проверяем, что счет происходит только при заданных требованиях, а также продолжается, когда требования вновь выполняются.

Запустим его и посмотрим на результат:

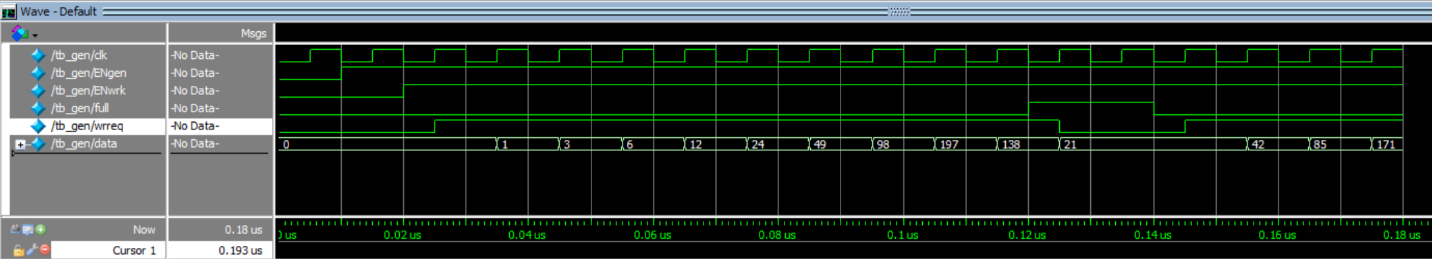


Рис. – Моделирование тестового файла средствами ModelSim (gen\_tb)

Как можно заметить, тест соответствует разработанным требованиям.

## Модуль RaF

### Создание модуля

Следующим разработаем модуль RaF, который будет считывать данные из FIFO и сохранять максимальное число. Алгоритм выглядит следующим образом:

* Если ( и и )
  + Модуль считывает данные из FIFO (формирует сигнал )
  + Ищет среди принимаемых от FIFO данных максимальное значение
  + Передает текущее максимальное значение на выход
* Если условие ( и и) не выполняется
  + Модуль формирует сигнал
  + Модуль находится в состоянии ожидания (текущее максимальное значение выдается на выход)
* Сигнал RST асинхронно сбрасывает найденное максимальное значение в 0

Его описание на языке System Verilog будет выглядеть следующим образом:

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

Автоматически созданное описание

Рис. – Модуль RaF

Посмотрим на RTL Viewer разработанного модуля:

Изображение выглядит как диаграмма, текст, снимок экрана, линия

Автоматически созданное описание

Рис. 8 – RTL Viewer модуля RaF

### Тестирование

Далее необходимо провести тестирование исследуемого модуля. Для этого разработаем тест первого уровня:

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

Автоматически созданное описание

Рис. – Тестовый файл первого класса для модуля RaF

В этом тесте мы подаем на вход q какое-то значение, но т.к. условия не выполняются записано оно не будет, после чего подаем все значения для выполнения условия и теперь запись должна быть выполнена. После чего пытаемся перезаписать меньшим значением.

Далее происходит сброс и пытаемся записать после сброса другое значение.

Запустим данный тест и посмотрим на результат тестирования:

Изображение выглядит как Мультимедийное программное обеспечение, линия, снимок экрана

Автоматически созданное описание

Рис. 10 – Моделирование тестового файла средствами ModelSim (RaF\_tb)

Как мы видим все результаты тестирования совпадают с ожиданиями.

Модуль FIFO мы возьмем из IP библиотеки, разрабатывать его не нужно, а модуль для вывода значений на 7-сегментный дисплей мы возьмем из лабораторной работы прошлого семестра, это значит мы готовы перейти к разработке основного модуля, представленном на рис. 1.

## Модуль CR\_1

### Создание модуля

Данное описание на языке System Verilog приведено ниже:

Изображение выглядит как снимок экрана, текст

Автоматически созданное описание

Рис. – Модуль CR\_1

Как можно заметить это просто описание соединение разработанных модулей а также очереди и модуля, лабораторной номер 3.

Очередь будет иметь следующие настройки:

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

Автоматически созданное описание

Рис. 12 – Настройка модуля FIFO

Выполним компиляцию и посмотрим на RTL Viewer, чтоб убедиться в корректности построенной схемы:

Изображение выглядит как диаграмма, линия, снимок экрана, План

Автоматически созданное описание

Рис. 13 – RTL Viewer модуля CR\_1

Как мы видим, полученная схема похожа на Рис. 2.1. что свидетельствует о корректности разработанного модуля. Теперь выполним тестирование этого модуля, для этого разработает следующий тест первого класса:

### Тестирование

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рис. 14 - Тестовый файл первого класса для модуля CR\_1

Данный тест проверяет работу модуля по следующему алгоритму:

* Запись в FIFO до появления сигнала ()
* Чтение из FIFO до появления сигнала ()
* Чтение и запись FIFO до появления на выводе максимального значения   
  ()
* Запрет работы ()
* Сброс максимального значения ()

Проведем первый тест, его результат выглядит следующим образом:

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

Автоматически созданное описание

Рис. – Тестирование модуля CR\_1 до сигнала full

Как мы видим, все работает корректно и данные выдаются до появления сигнала full. В bc данные не записываются т.к. он не активирован.

Теперь перейдем к следующему тесту:

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

Автоматически созданное описание

Рис.  – Тестирования модуля CR\_1 до сигнала empty

Система корректно показывает и на этом тесте, выдавая содержимое очереди до сигнала empty. В bc данные аналогично не попадают т.к. он не активирован.

Теперь запустим систему в стандартном для неё режиме работы:

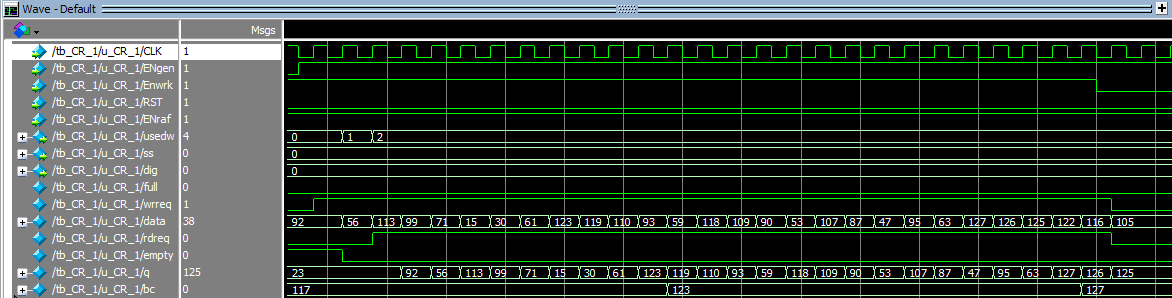


Рис. – Тестирование модуля CR\_1 до максимального значения генератора

Как мы видим, система работает корректно и действительно сохраняет значение максимума.

Теперь проверим сигнал RST:

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

Автоматически созданное описание

Рис. – Тестирования модуля CR\_1 на сигнале RST

На рисунке выше видно, что устройство корректно работает и с этими сигналами.

Стоит отметить, что во время тестирования выходы ss и dig оставались нулевыми. Это связано с тем, что модуле вывода данных на 7-сегментный индикатор стоит делитель, чтоб избавиться от мигания индикатора от частных обновлений. Его значение составляет около 10000, что много меньше количество тактов в тесте.

Изменим это, передав в него значение 1, чтоб при проверке на плате уже видеть нормальные сигналы.

## Создание модуля для тестирования на плате

Перейдем непосредственно к тестированию на плате, для этого создадим следующий модуль, благодаря которому сможем менять входные значение, используя ISSPE и смотреть на результат, используя Signal Tap II:

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

Автоматически созданное описание

Рис. – Модуль db для тестирования на плате

Его RTL Viewer приведен ниже:

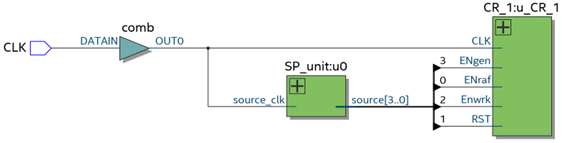


Рис. 20 – RTL Viewer для тестового файла модуля CR\_1

## Настройка Signal Tap II

Для ввода значений в модуль будем использовать ISSP, там же будем смотреть результат. Дополнительно добавим Signal Tap II, в котором будем получать значения по изменению состояний и получать результат в виде названия состояния и адреса:

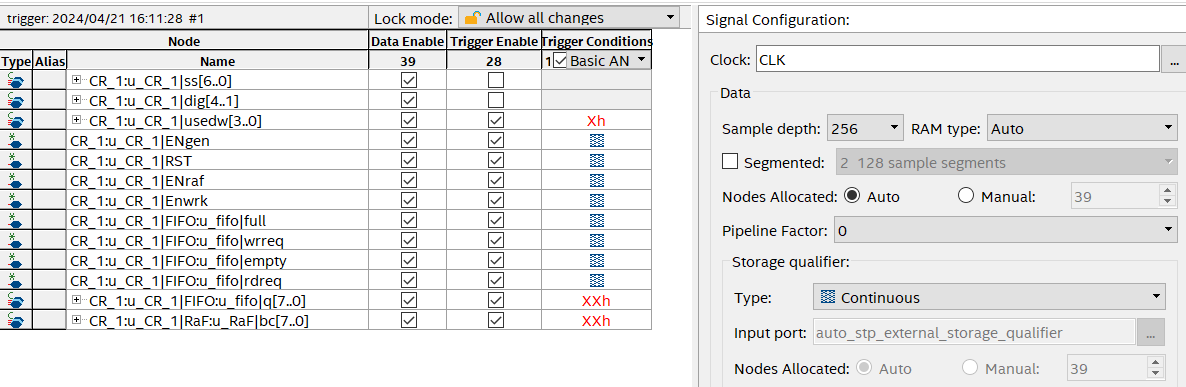


Рис. – Настройка окна Signal Tap II

Для выхода ss создадим маску, чтоб смотреть значение, которое он выводит:

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

Автоматически созданное описание

Рис. – Мнемоническая таблица

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. 23 – Временные характеристики устройства

## Тестирование на плате средствами Signal Tap II

Теперь запустим и выполним проверку корректности работы программы на плате. Выполним загрузку разработанного модуля на плату и рассмотрим интересующие нас случаи:

Запись в FIFO до появления сигнала ():

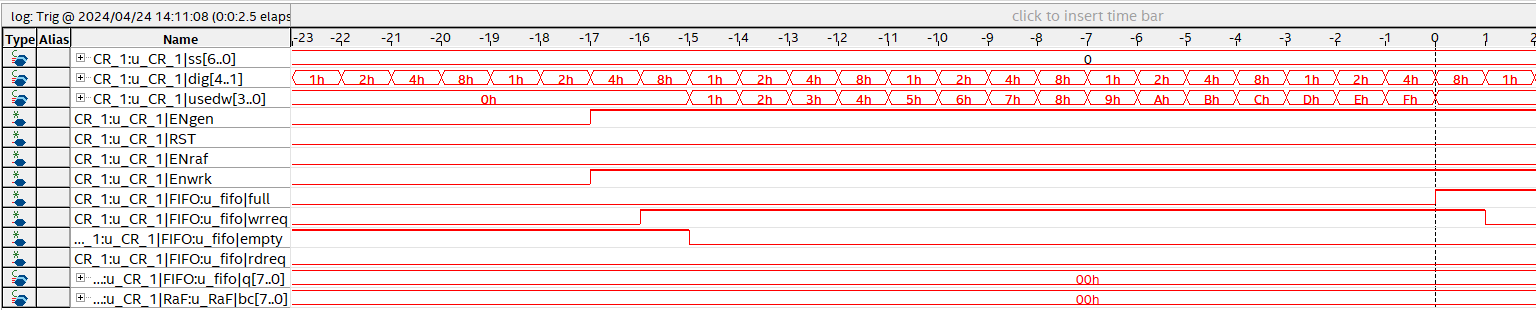


Рис. – Появление сигнала full при тестировании на плате

Чтение из FIFO до появления сигнала ():

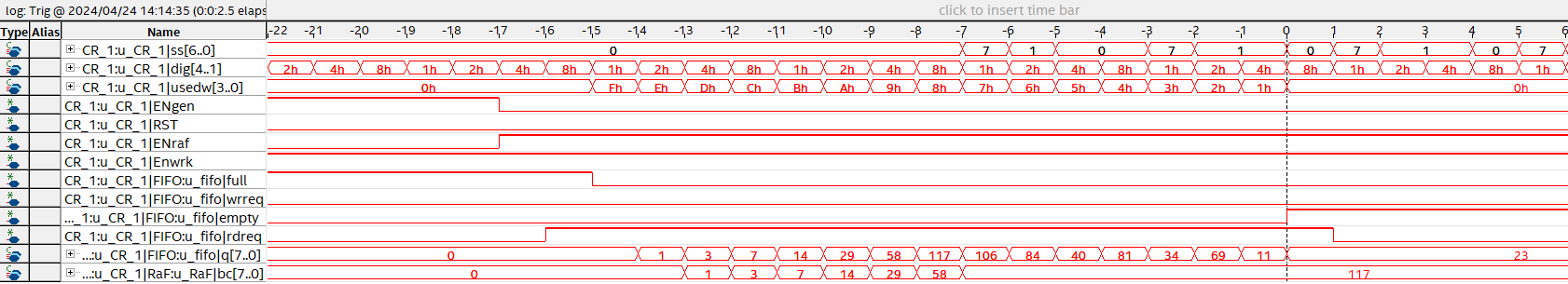


Рис. – Появление сигнала empty при тестировании на плате

Чтение и запись FIFO до появления на выводе максимального значения   
():

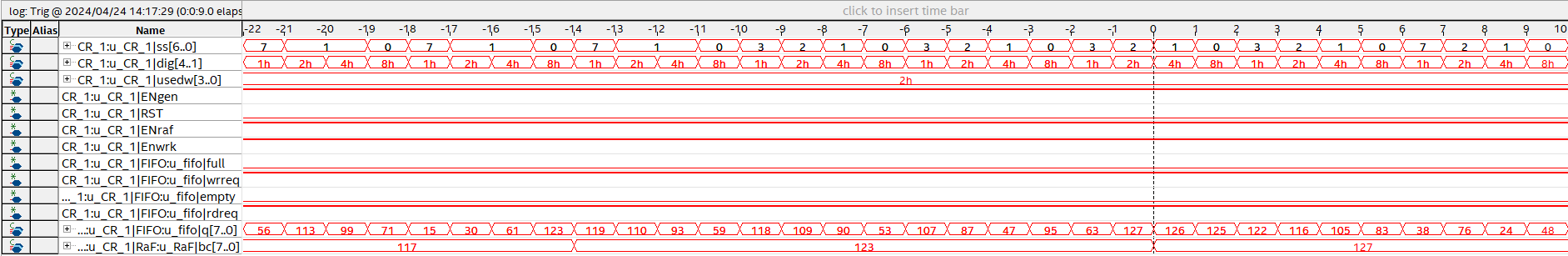


Рис. – Появление значения 127 (максимального) при тестировании на плате

Запрет работы ():

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Автоматически созданное описание

Рис. – Отключение работы при тестировании на плате

Сброс максимального значения ():

Изображение выглядит как текст, линия, Шрифт, число

Автоматически созданное описание

Рис. – Сброс при тестировании на плате

## Создание модуля имплементации и запуск на плате

Как мы видим, все работает корректно. Раз устройство работает верно, перейдем к созданию модуля имплементации:

Как мы видим, все работает корректно. Раз устройство работает верно, перейдем к созданию модуля имплементации:

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

Автоматически созданное описание

Рис. – Модуль имплементации

Также не забудем выполнить изменения в модуле вывода значений на 7-сегментный индикатор, поставив делитель обратно на 10000. Выполним компиляцию и посмотрим на получившуюся RTL схему:

Изображение выглядит как диаграмма, линия, снимок экрана, План

Автоматически созданное описание

Рис. 30 – RTL Viewer для impl\_CR\_1

Запишем получившуюся программу на плату. Результат работы продемонстрирован преподавателю.

# Вывод

В результате выполнения курсовой работы была успешно осуществлена разработка заданного устройства с использованием System Verilog в качестве основного инструмента. Кроме того, произведено комплексное тестирование устройства как на симуляторе, так и на плате. Этот процесс охватывал все этапы разработки - от первоначального проектирования до завершения работы над полностью функциональным устройством, готовым к применению. Полученный результат свидетельствует о стабильной работоспособности разработанного устройства, что подтверждает успешное завершение проекта. Эти навыки разработки и тестирования электронных устройств могут быть полезными для будущих проектов в области аппаратной разработки, системной интеграции и производства электроники.