Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчёт по лабораторной работе № 10**

Дисциплина: Языки описания аппаратных средств вычислительных систем.

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

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Федотов

(подпись)

Санкт-Петербург

2023

Оглавление

[1. Задача: 3](#_Toc151334601)

[2. Решение: 5](#_Toc151334602)

[3. Вывод: 13](#_Toc151334603)

# Задача:

На языке Verilog разработать:

* Модуль ss\_cntr – параметризированный модуль управления динамическим отображением для 4-разрядного 7-сегментного индикатора.
* Модуль lab\_2s – оберточный модуль, осуществляющий подключение к выводам компонента ss\_cntr выводов, на плате; задание константных значений выводам компонента ss\_cntr; задание коэффициента деления счетчика.

***Модуль ss\_cntr:***

**Входы:**

* сlk – тактовый сигнал.
* rst\_n – синхронный сброс, активный уровень – 0.
* [3:0] A – вход данных.
* [3:0] B – вход данных.
* [3:0] C – вход данных.
* [3:0] D – вход данных.

**Выходы:**

* [6:0] ss – выходы данных для 7-сегментного индикатора.
* [4:1] dig – выходы управления включением разрядов 4-разрядного 7-сегментного индикатора.

Изображение выглядит как текст, диаграмма, План, схематичный

Автоматически созданное описание

Рис. 1.1 – Структура устройства ss\_cntr

**Состав устройства:**

* rst\_rg – синхронизатор сигнала сброса, компонент должен содержать 2 последовательно включенных триггера.
* d\_rg – компонент, содержащий 4 регистра для хранения 4-х наборов входных данных.
* MUX2 – параметризированный мультиплексор 4(Nбит) =>1(N бит), реализованный как комбинационная схема. Параметр N – разрядность мультиплексора (базовое значение = 4).
* b2ss – преобразователь двоичного кода в 7-сегментный реализованный как комбинационная схема.
* cnt\_div – счетчик-делитель с параметризированным коэффициентом деления.
* FSM – конечный автомат, обеспечивающий управление блоком MUX и формирование сигналов включения разрядов 7-сегментного индикатора.

Изображение выглядит как текст, круг, Шрифт, диаграмма

Автоматически созданное описание

Рис. 1.2 – Граф переходов модуля FSM

* ss\_rg – регистр хранения выходных данных 7-сегментного индикатора.
* dig\_rg – регистр хранения сигналов управления 7-сегментного индикатора [4:1] dig).

***Модуль lab\_2s:***

**Входы:**

* сlk – тактовый сигнал.
* rst\_n – синхронный сброс, активный уровень – 0.

**Выходы:**

* [6:0] ss – выходы данных для 7-сегментного индикатора.
* [4:1] dig – выходы управления включением разрядов 4-разрядного 7-сегментного индикатора.

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Автоматически созданное описание

Рис. 1.3 – Структура устройства lab\_2s

**Состав устройства:**

* ss\_rst – компонент разработанного модуля.

# Решение:

Реализуем каждое из устройств, начнем с FSM, это конечный автомат, граф которого приведен на Рис. 1.2, реализуем его следующим образом:

Изображение выглядит как текст, снимок экрана, число, дисплей

Автоматически созданное описание

Рис. 2.1 – FSM на Verilog

Выполним компиляцию и синтез, чтоб удостовериться, что схема действительно скомпилировалась, как конечный автомат:

Изображение выглядит как текст, диаграмма, снимок экрана, График

Автоматически созданное описание

Рис. 2.2 – RTL Viewer

Далее реализуем тестовый модуль FSM\_tb:

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

Автоматически созданное описание

Рис. 2.3 – FSM\_tb на Verilog

Это тест 2 класса, все ожидаемые значения записаны прямо в тесте, для удобства. Помимо самих переходов проверена работа входа ENA и rst\_ni. Все тесты выполняются корректно, wave выглядит следующим образом:

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

Автоматически созданное описание

Рис. 2.4 – Wave для FSM\_tb

На ней видно, что модуль работает корректно, в том числе асинхронный сброс. Далее реализуем модуль MUX:

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

Автоматически созданное описание

Рис. 2.5 – MUX на Verilog

Этот модуль выполнен в соответствии с этой схемой:

Изображение выглядит как текст, Шрифт, диаграмма, белый

Автоматически созданное описание

Рис. 2.6 – Схема модуля MUX

Удостоверимся, что полученная схема действительно комбинационная, как этого требует задание:

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Автоматически созданное описание

Рис. 2.7 – RTL Viewer модуля MUX

Выполним тестирование этого модуля, используя тестовый модуль MUX\_tb:

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рис. 2.8 – Тестовый модуль MUX\_tb

Данный тестовый модуль берет значения из входного файла и сравнивает со значением из выходного. Модуль MUX работает корректно, все ожидаемые данные совпали с реальными. Wave выглядит следующим образом:

Изображение выглядит как снимок экрана, электроника, линия, компьютер

Автоматически созданное описание

Рис. 2.9 – Wave для MUX\_tb

Реализуем модуль ss\_rg:

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

Автоматически созданное описание

Рис. 2.10 – ss\_rg на Verilog.

Реализуем модуль dig\_rg:

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

Автоматически созданное описание

Рис. 2.11 – dig\_rg на Verilog.

Реализуем модуль d\_rg:

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

Автоматически созданное описание

Рис. 2.12 – d\_rg на Verilog.

Модуль rst\_rg взят из lab\_1s:

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

Автоматически созданное описание

Рис. 2.13 – rst\_rg на Verilog

Модуль b2ss возьмем из lab3\_2:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рис. 2.14 – b2ss на Verilog

Модуль cnt\_div возьмем из lab\_1s:

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

Автоматически созданное описание

Рис. 2.15 – cnt\_div на Verilog

Создадим структурное описание модуля ss\_cntr:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рис. 2.16 – ss\_cntr на Verilog

Это параметризированное устройство, где параметр – коэффициент деления. Реализуем модуль-обертку lab\_2s:

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

Автоматически созданное описание

Рис. 2.17 – lab\_2s на Verilog

Выполним компиляцию и получим следующую схему на RTL Viewer:

Изображение выглядит как диаграмма, линия, План, снимок экрана

Автоматически созданное описание

Рис. 2.18 – RTL Viewer для lab\_2s

Создадим тест 1 класса для этого модуля:

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

Автоматически созданное описание

Рис. 2.19 – lab\_2s\_tb на Verilog

Запустим его и получим следующую Wave:

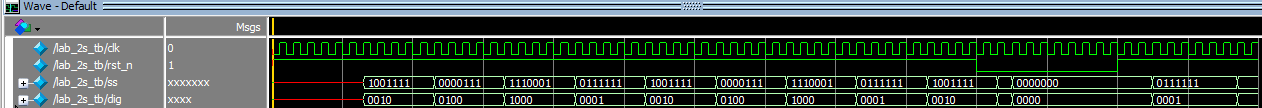


Рис. 2.20 – Wave для модуля lab\_2s\_tb

Устройство работает в соответствии с ожиданиями.

# Вывод:

В результате выполнения лабораторной работы было разработано устройство для динамической индикации на семисегментном индикаторе с использованием языка Verilog. Это устройство позволяет выводить любое значение на индикатор, принимая входные данные и управляя отображением на семисегментном индикаторе.

Применение на практике включает создание параметризированного модуля управления динамическим отображением для 4-разрядного 7-сегментного индикатора (ss\_cntr). Также разработан оберточный модуль (lab\_2s), который осуществляет подключение к выводам компонента ss\_cntr, задание константных значений выводам и коэффициента деления счетчика.

Программирование на языке Verilog значительно упрощает разработку сложных устройств по сравнению с использованием блочных диаграмм. Моделирование с использованием ModelSim дает возможность провести более комплексное тестирование устройства, что ускоряет процесс разработки.

Плюсы:

1. Упрощение разработки сложных устройств.
2. Возможность параметризации модулей для адаптации к различным требованиям.
3. Более комплексное тестирование с использованием ModelSim.
4. Корректное завершение процессов.

Таким образом, использование языка Verilog и инструментов моделирования, таких как ModelSim, предоставляет эффективные средства для разработки и тестирования сложных устройств динамической индикации.