Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчёт по лабораторной работе № 4**

Дисциплина: Автоматизация проектирования дискретных устройств (на английском языке)

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

(подпись)

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

(подпись)

Санкт-Петербург

2023

**Оглавление**

[1. Задание 3](#_Toc159544943)

[2. Ход решения 4](#_Toc159544944)

[2.1. Создание модуля на языке Verilog 4](#_Toc159544945)

[2.2. Создание теста 6](#_Toc159544946)

[2.3. Отладка устройства 8](#_Toc159544947)

[2.4. Настройка SignalTap 9](#_Toc159544948)

[2.5. Индивидуальное задание 11](#_Toc159544949)

[3. Вывод 11](#_Toc159544950)

# Задание

1. На языке SystemVerilog создать описание 10 разрядного двоичного счетчика - cnt\_10bits (файл - cnt\_10bits.sv)

* модуль счета задается на входе Module
* счет на сложение до значения Module-1
* вход асинхронного сброса rst\_n – активный уровень 0
* выход счетчика - [9:0] cnt
* выход переноса Cout – формируется логическая единица, когда значение на выходе счетчика = Module-1.Разработка описания модулей на SystemVerilog

1. На языке SystemVerilog создать описание комбинационного компаратора - cmp\_eq (файл - cmp\_eq.sv)

* Компаратор на равенство двух 10 разрядных
* Выход EQ – равен 1 при равенстве двух входных чисел.

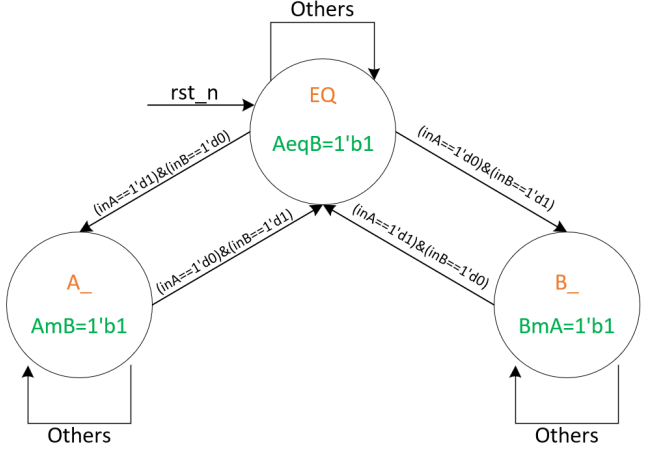
1. На языке SystemVerilog (максимально используя конструкции расширения SystemVerilog) создайте описание конечного автомата - fsm (файл - fsm.sv

* Конечный автомат (fsm) анализирует сигналы переноса двух счетчиков и формирует сигналы:

1. AeqB – равное количество сигналов переноса от счетчика А и счетчика В.
2. AmB – сигналы переноса от счетчика А появляются чаще.
3. BmA – сигналы переноса от счетчика В появляются чаще.

* Граф переходов приведен ниже

1. Состояния выделены оранжевым цветом
2. Выходные сигналы (в тех состояниях, где они равны 1, в остальных состояниях они должны быть равны 0) выделены зеленым цветом.



1. На языке SystemVerilog создать описание модуля верхнего уровня - lab\_MS\_SV1 (файл - lab\_MS\_SV1.sv)

# Ход решения

## Создание модуля на языке Verilog

На языке SystemVerilog была описана реализация 10-разрядного двоичного счётчика:

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

Автоматически созданное описание

Рис. 1 – Код файла cnt\_10bits.sv

На языке SystemVerilog было создано описание комбинационного компаратора.

Изображение выглядит как текст, снимок экрана, дисплей, Шрифт

Автоматически созданное описание

Рис. 2 – Код файла cmp\_eq.sv

На языке SystemVerilog было создано описание конечного автомата.

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

Автоматически созданное описание

Рис. 3 – Код файла fsm.sv

На языке SystemVerilog было создано описание модуля верхнего уровня.

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

Автоматически созданное описание

Рис. 4 – Код файла lab\_MS\_SV1.sv (1)

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

Автоматически созданное описание

Рис. 5 – Код файла lab\_MS\_SV1.sv (2)

Изображение выглядит как текст, диаграмма, снимок экрана, План

Автоматически созданное описание

Рис. 6 – Структура устройства в RTL-viewer

## Создание теста

На языке SystemVerilog напишем файл для тестирования созданного устройства:

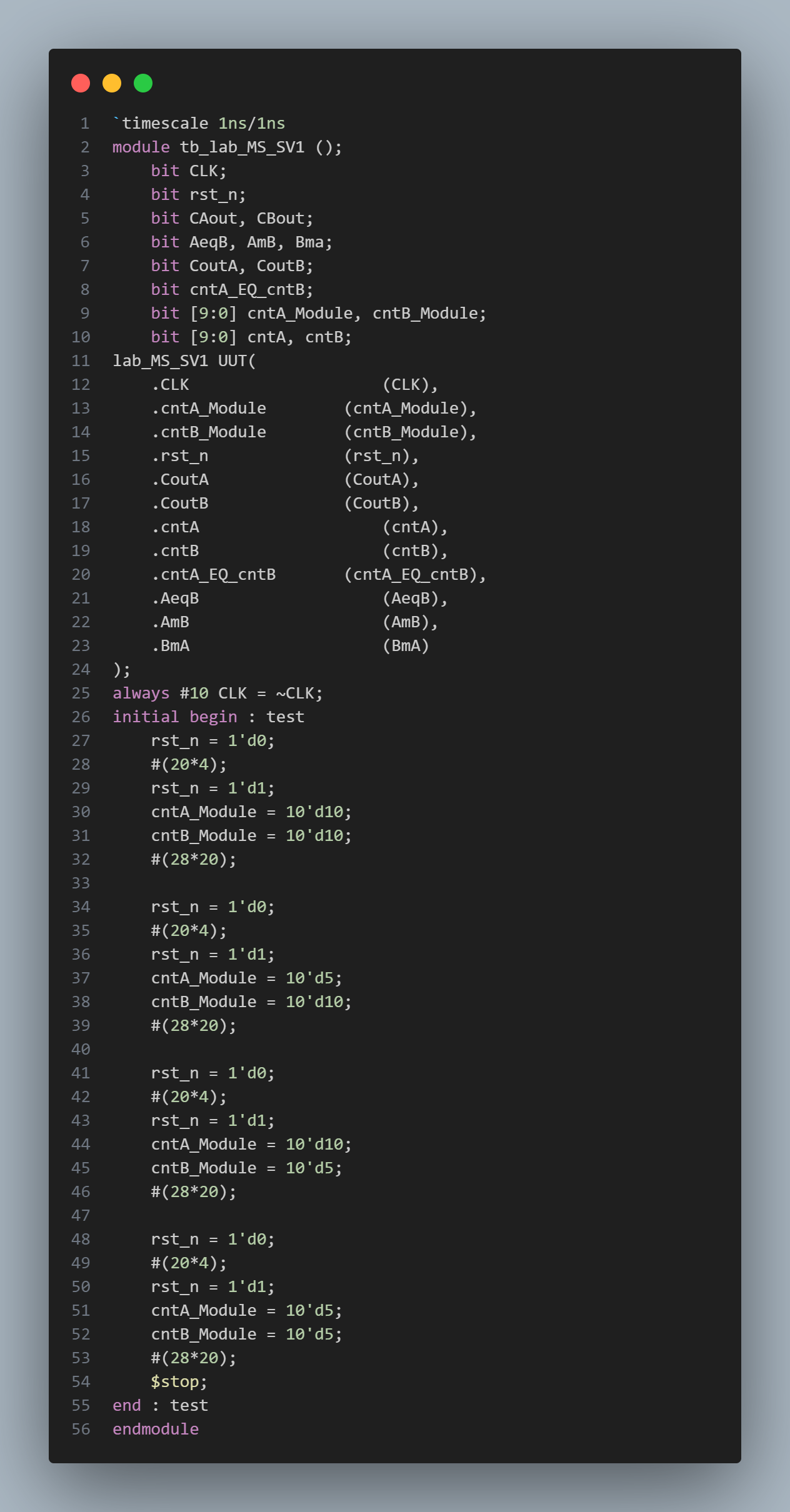


Рис. 7 – Код файла tb\_lab\_MS\_SV1.sv

Выполним моделирование в ModelSim:

Изображение выглядит как текст, стена, табло

Автоматически созданное описание

Рис. 8 – Симуляция проекта в ModelSim

Из диаграммы видно, что устройство работает корректно.

## Отладка устройства

Для отладки устройства создадим и настроим ISSPE модуль. Также необходимо создать файл верхнего уровня для отладки. Структура модуля для отладки приведена на (Рис. 1-8).



Рис. 9 – Описание файла верхнего уровня на языке SystemVerilog

После подключения файла можем убедиться, что устройство собрано верно.

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Автоматически созданное описание

Рис. 10 – Структура устройства в RTL-viewer

Создадим .sdc файл с требованиями к тактовому сигналу:

Изображение выглядит как текст

Автоматически созданное описание

Рис. 11 – Требования к тактовому сигналу

После компиляции посмотрим на временные характеристики модели:

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

Рис. 12 – Временные характеристики устройства

## Настройка SignalTap

Выполним предварительную настройку она SignalTap перед запуском симуляции:

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

Автоматически созданное описание

Рис. 13 – In System Sources and Probes Editor

Для отображения выводов конечного автомата создадим мнемоническую. Зададим следующие настройки Signal Tap:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рис. 14 – Настройки Signal Tap

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

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

Рис. 15 – Временные характеристики устройства

Зададим режимы работы для захвата данных в 4-х сегментах логического анализатора. Из приведенной диаграммы можно сделать вывод, что устройство работает правильно.

Изображение выглядит как стол

Автоматически созданное описание

Рис. 16 – диаграмма из ISSPE

Сравнив полученную средствами Signal Tap с той, которую до этого наблюдали в Modelsim, можем увидеть, что они совпадают.

## Индивидуальное задание

Вариант 12

Задача: настроить Signal Tap для формирования двух сегментов логического анализатора и задать на вход данные в соответствии с индивидуальным заданием.

Набор для сегмента 1: cntA\_Module = 28, cntB\_Module = 14

Набора для сегмента 2: cntA\_Module = 14, cntB\_Module = 28

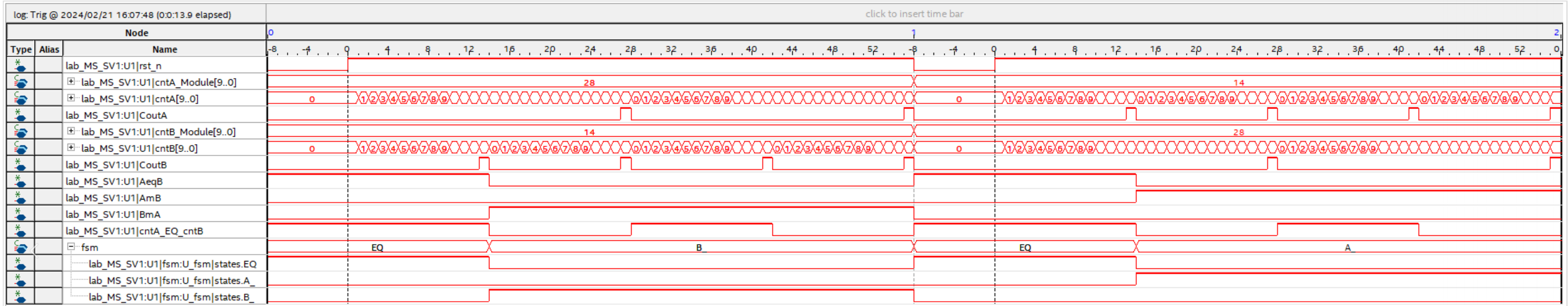


Рис. 17 – диаграмма из Signal Tap

* rst\_n = 0: Счётчик остаётся равным нулю, как и следует при активном сигнале сброса.
* rst\_n = 1: Рассмотрим первый набор значений (cntA\_Module = 28, cntB\_Module = 14). Через каждые 14 «тиков» на coutB подаётся сигнал, после чего счётчик сбрасывается. Аналогично, при значении cntA\_Module = 28 подаётся сигнал на coutA каждые 28 «тиков».

Из приведенной диаграммы видно, что устройство работает верно.

# Вывод

В рамках лабораторной работы была реализована и отлажена система на языке SystemVerilog, состоящая из 10-разрядного счетчика, компаратора на равенство и конечного автомата. Устройство предназначено для сравнения двух чисел с использованием конечного автомата.

Преимущества реализации:

* Модульность: Система разделена на модули, что упрощает ее разработку, тестирование и обслуживание.
* Повторное использование: Модули могут быть использованы в других проектах.
* Верификация: SystemVerilog позволяет использовать формальные методы верификации для проверки корректности работы системы.
* Отладка: SystemVerilog предоставляет инструменты для отладки системы, такие как SignalTap.

Использование таких средств как SignalTap позволило отслеживать сигналы на основе значений для формирования пары сегментов, задаваемых через Programmer. Такое представления удобно для отладки и помогает гораздо быстрее обнаружить проблему, протестировав различные входные данные.