Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчёт по лабораторной работе Lab\_MS\_SV\_2**

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 3](#_Toc160975806)

[1.1. Цель работы 3](#_Toc160975807)

[1.3. Вариант по заданию 4](#_Toc160975808)

[2. Ход решения 5](#_Toc160975809)

[2.1. Создание LFSR модуля 5](#_Toc160975810)

[2.2. Создание модуля гистограммы 7](#_Toc160975811)

[2.3. Создание модуля верхнего уровня 10](#_Toc160975812)

[2.4. Создания модуля для тестирования на плате 12](#_Toc160975813)

[2.5. Настройка Signal Tap II 12](#_Toc160975814)

[2.6. Доработка модуля гисторграммы 13](#_Toc160975815)

[2.7. Тестирование на плате 14](#_Toc160975816)

[3. Вывод 16](#_Toc160975817)

**Список иллюстраций**

[Рис. 1 – LFSR модуль 6](#_Toc160975883)

[Рис. 2 – Структура LFSR модуля в RTL Viewer 6](#_Toc160975884)

[Рис. 3 – Тестовый файл для модуля LFSR 7](#_Toc160975885)

[Рис. 4 – Моделирование тестового файла средствами ModelSim (вар. 1) 7](#_Toc160975886)

[Рис. 5 – Моделирование тестового файла средствами ModelSim (вар. 2) 7](#_Toc160975887)

[Рис. 6 – Проверка на полный проход цикла (сигнал LFSR\_CYCLE) 8](#_Toc160975888)

[Рис. 7 – Модуль гистограммы 8](#_Toc160975889)

[Рис. 8 – Структура модуля гистограммы в RTL Viewer 9](#_Toc160975890)

[Рис. 9 – Тестовый файл для модуля гистограммы 10](#_Toc160975891)

[Рис. 10 – Результаты моделирования тестового файла модуля гистограммы 10](#_Toc160975892)

[Рис. 11 – Данные в памяти 11](#_Toc160975893)

[Рис. 12 – Модуль верхнего уровня 11](#_Toc160975894)

[Рис. 13 – Тестовый файл для модуля верхнего уровня 12](#_Toc160975895)

[Рис. 14 – Данные в памяти гисторграммы 12](#_Toc160975896)

[Рис. 15 – Модуль db для тестирования на плате 13](#_Toc160975897)

[Рис. 16 – Настройка окна Signal Tap II 13](#_Toc160975898)

[Рис. 17 – Временные характеристики устройства 13](#_Toc160975899)

[Рис. 18 – Настройка ISSP 13](#_Toc160975900)

[Рис. 19 – 14](#_Toc160975901)

[Рис. 20 – исправленный модуль гистограммы 14](#_Toc160975902)

[Рис. 21 – Схема исправленного модуля гистограммы в RTL Viewer 15](#_Toc160975903)

[Рис. 22 – Результаты моделирования обновлённого тестового файла модуля гистограммы 15](#_Toc160975904)

[Рис. 23 – Данные в памяти (2) 15](#_Toc160975905)

[Рис. 24 – Память при RST = 1 16](#_Toc160975906)

[Рис. 25 – Память при RST = 0 16](#_Toc160975907)

[Рис. 26 – Результаты моделирования средствами Signal Tap II 16](#_Toc160975908)

# Задание

## Цель работы

Лабораторная работа выполняется по индивидуальным заданиям. Теория и приведенные ниже (на английском языке) примеры могут быть использованы как образцы для составления собственных исходных кодов проектируемых модулей и тестов. В работе следует использовать конструкции SystemVerilog.

* 1. **Программа работы:**
* Разработать оптисание LFSR по индивидуальному заданию (номер задания = номеру в списке группы)
* Разработать тест для проверки LFSR и провести моделирование
* По результатам моделирования в ModelSim **необходимо доказать**, что период повторения равен 2N-1, где N – максимальная степень полинома из задания (ОБРАТИТЕ ВНИМАНИЕ: в некоторых вариантах задания, теоретически, может быть ошибка, тогда этот период будет меньше 2N-1. Если у Вас так получилось, то надо, прежде всего, проверить правильность своего описания, и только после этого обсудить это с преподавателем).
* Разработать модуль для построения гистограммы
* Разработать тест для проверки модуля построения гистограммы и провести моделирование
* По результатам моделирования в ModelSim необходимо доказать, что для входных данных, поступающих со счетчика (шаг счета = номеру студента в группе) получена правильная гистограмма в модуле памяти.
* Разработать модуль верхнего уровня lab\_MS\_SV3, содержащий LFSR и модуль построения гистограммы.
* Разработать тест для проверки модуля верхнего уровня иерархии (tb\_lab\_MS\_SV3).
* По результатам моделирования необходимо доказать, что период повторения LSFR равен 2N-1, где N – максимальная степень полинома из задания и модуль построения гистограммы создает правильную гистограмму.
* Разработать модуль верхнего уровня для отладки db\_lab\_MS\_SV3, содержащий: модуль lab\_MS\_SV3; модуль SP\_unit (модуль, обеспечивающий возможность задания входных управляющих сигналов без использования кнопок на плате). Модуль должен содержать подключение только к тактовому сигналу на плате.
* Настроить логический анализатор для проведения исследования и отладки реализуемого на плате db\_lab\_MS\_SV3.
* Провести анализ работы db\_lab\_MS\_SV3 и доказать (зафиксировав результаты снимками экрана), что:
  + Модуль управляется входными сигналами RST и ENA
  + Правильно формируется гистограмма
  + Формируемые псевдослучайные данные отображаются аналогично изображению на Figure 17.
* Изменение модуля histogram\_unit
  + В модуле histogram\_unit (см. Figure 7) замените память mem\_in, созданную как массив, на параметризированный (IP) блок RAM:1\_PORT
    - 10 бит – разрядность слова, 1024 слова, Read-During Write – OLD DATA;
    - Включите Allow in-System-Memory-Content Editor и задайте Instance ID = RAM0
  + Осуществите моделирование в пакете ModelSim (не забудьте подключить библиотеку altera\_mf\_ver) и убедитесь в правильности работы модуля.
* В модуле db\_lab\_MS\_SV3 настройте в логическом анализаторе 16 сегментов (по 16 отсчетов, положение – в центре) условие захвата – достижение генератором значения, равного Вашему номеру в списке группы.
* Запустите ISSP, убедитесь, что RST=1 и ENA= 1;
* Запустите IMCE: считайте данные из модуля памяти с Instance ID = RAM0. Зафиксируйте их и объясните полученные результаты.
* Откройте окно Логического анализатора SignalTapII и запустите захват данных
* Откройте окно ISSP, установите RST=0
* Откройте окно IMCE: считайте данные из модуля памяти с Instance ID = RAM0. Зафиксируйте их и объясните полученные результаты.
* Откройте окно Логического анализатора SignalTapII, зафиксируйте и приведите в отчете временные диаграммы, убедитесь в том, что:
  + временная диаграмма похожа на приведенную ниже
  + формируемые генератором данные, повторяются во всех 16-х сегментах.
  + объясните почему на выходе mem\_out мы видим числа 0, 1, 2, …15.
    - если увеличить количество сегментов для захвата данных, то до какого занчения будут увеличиваться эти числа?
      * Как это объяснить?
      * Докажите это с использованием SignalTapII

## Вариант по заданию

|  |  |  |
| --- | --- | --- |
| **№ варианта** | **Полином для реализации** | **Тип реализации**  **Тип логического элемента обратной связи** |
| 6 |  | Фибоначчи  XOR |

# Ход решения

## Создание LFSR модуля

В соответствии с заданием разработаем описание LSFR модуля на языке SystemVerilog для следующего полинома:

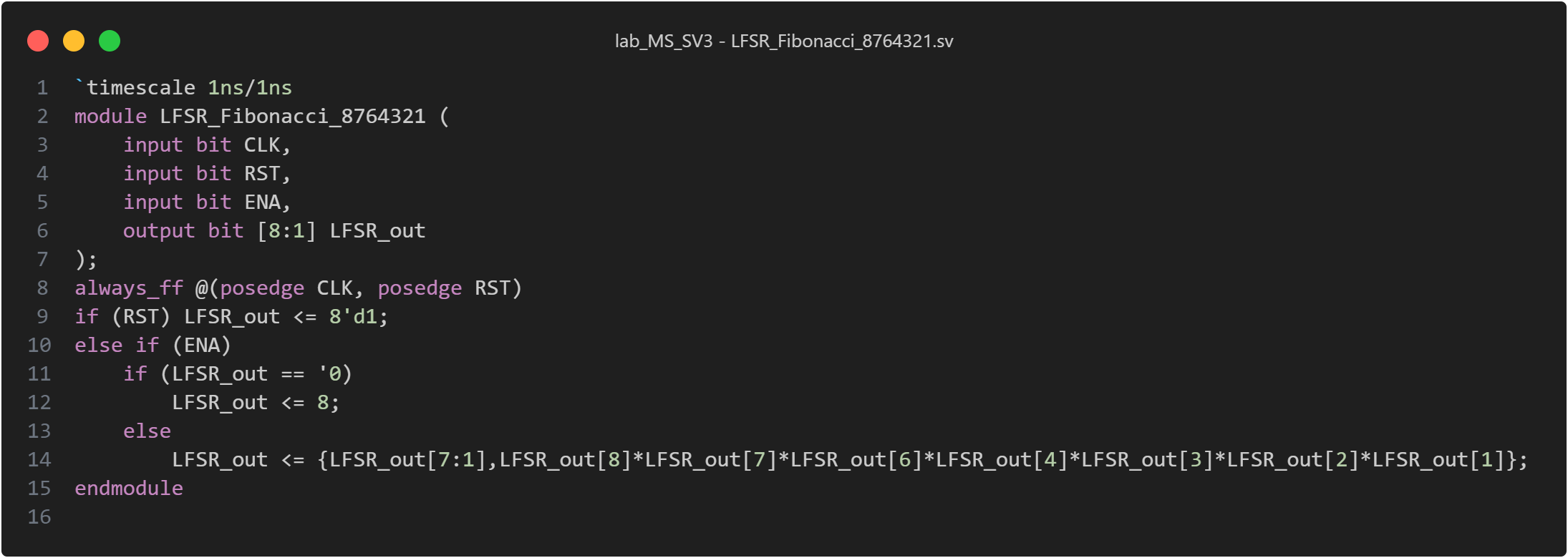


Рис. 1 – LFSR модуль

Данный модуль – сдвигающий регистр, с асинхронным сбросом. Сброс выполняется в 1 т. к. в противном случае LFSR перестанет работать.

После компиляции данного модуля можем увидеть, что его структура в RTL Viewer выглядит следующим образом:

*Изображение выглядит как диаграмма, линия, снимок экрана, График

Автоматически созданное описание*

Рис. 2 – Структура LFSR модуля в RTL Viewer

Напишем тест для только что созданного модуля LFSR:

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рис. 3 – Тестовый файл для модуля LFSR

Данный модуль позволит нам посчитать период. Т. к. степень полинома 8, мы ожидаем период равный .

Для проверки ожидаемых результатов протестируем работу модуля средствами ModelSim. При этом получим следующую waveform’у:

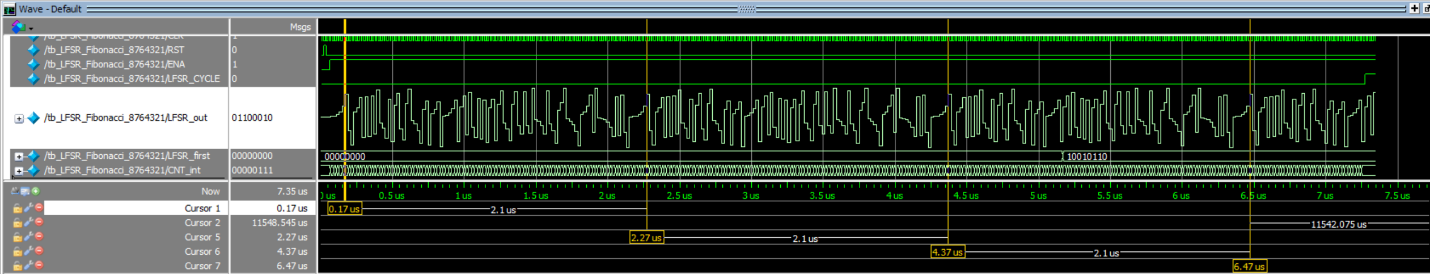


Рис. 4 – Моделирование тестового файла средствами ModelSim (вар. 1)

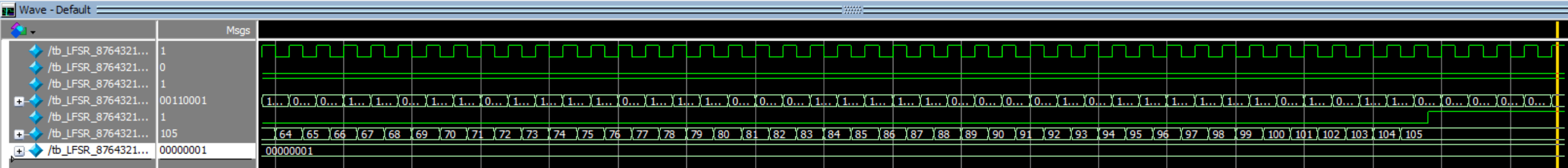


Рис. 5 – Моделирование тестового файла средствами ModelSim (вар. 2)

Как видно из полученной waveform, ожидаемый период не был получен. Однако, сигнал LFSR\_CYCLE (3-й снизу сигнал) при подаче последнего сигнала перешёл в состояние 1. Это значит, что был пройден полный цикл. Фрагмент кода с модулем LFSR\_CYCLE представлен ниже:

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

Автоматически созданное описание

Рис. 6 – Проверка на полный проход цикла (сигнал LFSR\_CYCLE)

## Создание модуля гистограммы

Теперь создадим модуль для создания гистограммы:

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

Автоматически созданное описание

Рис. 7 – Модуль гистограммы

Здесь MAX\_NUMBER – параметр, определяющий максимальное число в гистограмме, а SIZE – размерность данных в гистограмме.

Данный модуль использует память mem\_arr, в которой каждый такт по введенному адресу (d\_in) добавляется единица, также присутствует возможность отчистки памяти, однако для полной отчистки понадобится MAX\_NUMBER тактов.

Получившаяся RTL схема выглядит следующим образом:

*Изображение выглядит как линия, антенна

Автоматически созданное описание*

Рис. 8 – Структура модуля гистограммы в RTL Viewer

Как можно заметить, тут нет как таковой памяти, все строится на регистрах. Причина этого в том, что мы пытаемся одновременно читать значения из памяти и записывать в неё, из-за чего Quartus решает, что это лучше сделать, используя регистровые схемы.

Теперь напишем тест для созданного модуля:

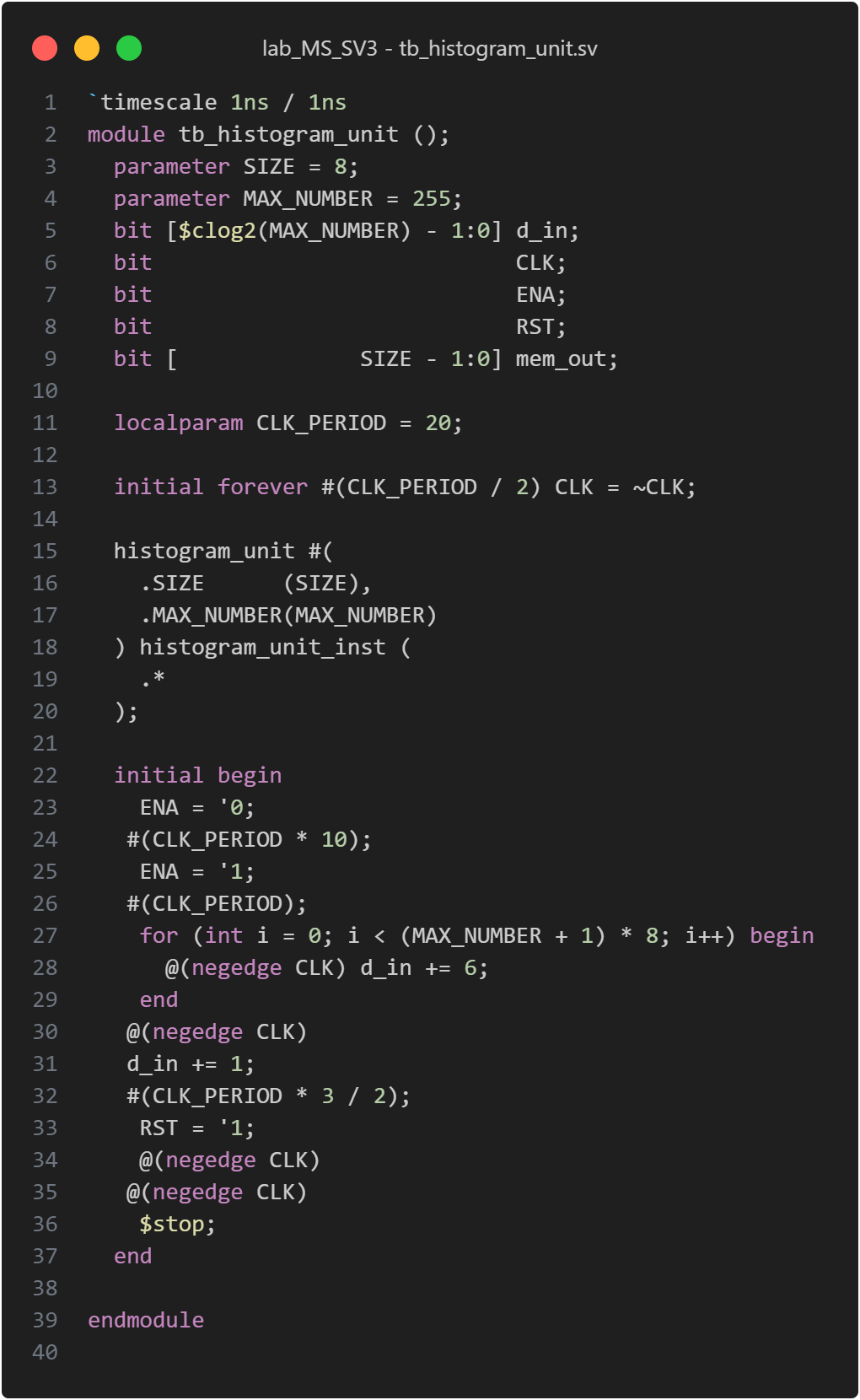


Рис. 9 – Тестовый файл для модуля гистограммы

При прогонке данного тестового модуля средствами ModelSim получим следующую временную диаграмму:

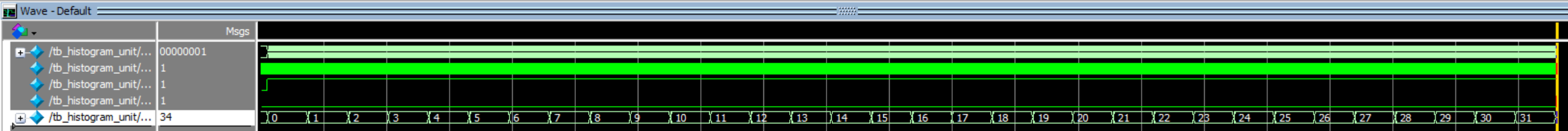


Рис. 10 – Результаты моделирования тестового файла модуля гистограммы

Здесь же, открыв вкладку Memory List, можем увидеть данные в памяти:

Изображение выглядит как текст, снимок экрана, число

Автоматически созданное описание

Рис. 11 – Данные в памяти

Как мы видим, модуль гистограммы корректно обрабатывает входную последовательность, а также очищает данные по RST.

## Создание модуля верхнего уровня

Теперь напишем модуль верхнего уровня, объединив LFSR и модуль гистограммы, чтоб проверить, что в LFSR все случайные числа равновероятны:

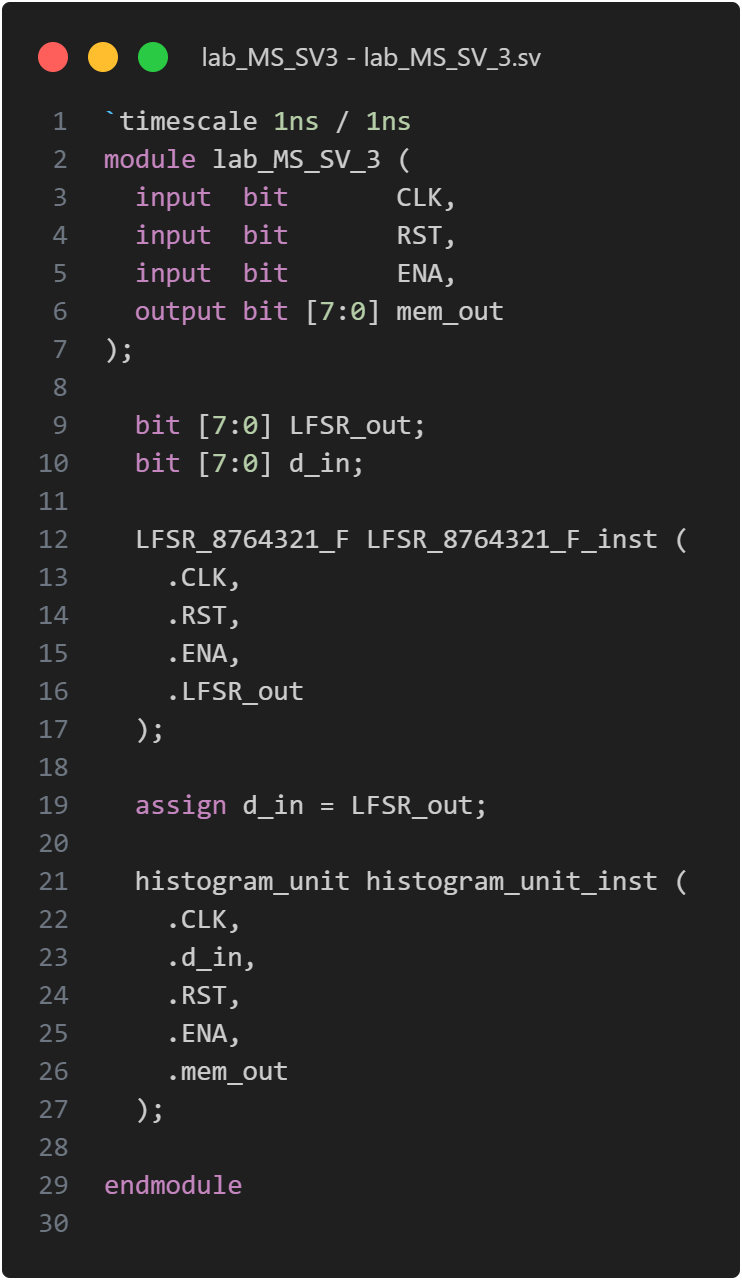


Рис. 12 – Модуль верхнего уровня

Аналогично тому. Как делали с предыдущими модулями, напишем тестовый модуль:

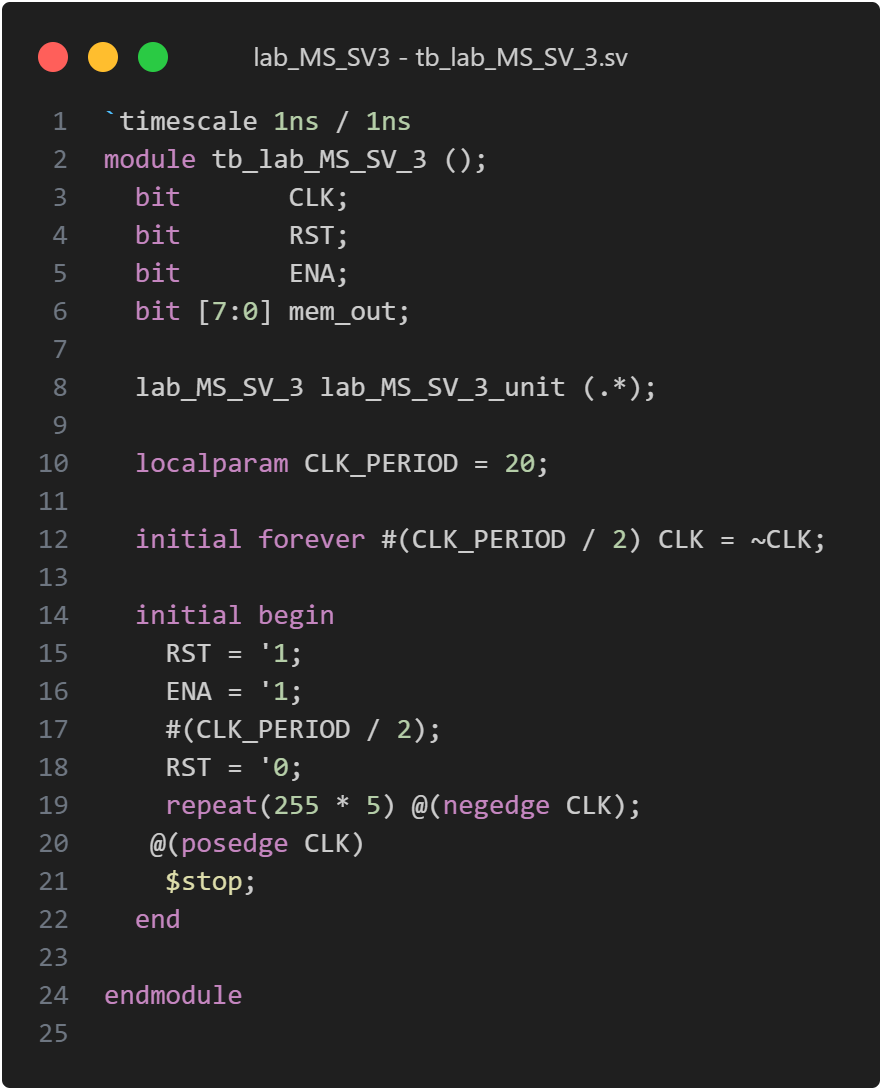


Рис. 13 – Тестовый файл для модуля верхнего уровня

Мы 12 раз повторяем период, который был получен ранее, ожидая, что в гистограмме все значения от 1 до 255 будут равны 5. Проверим это:

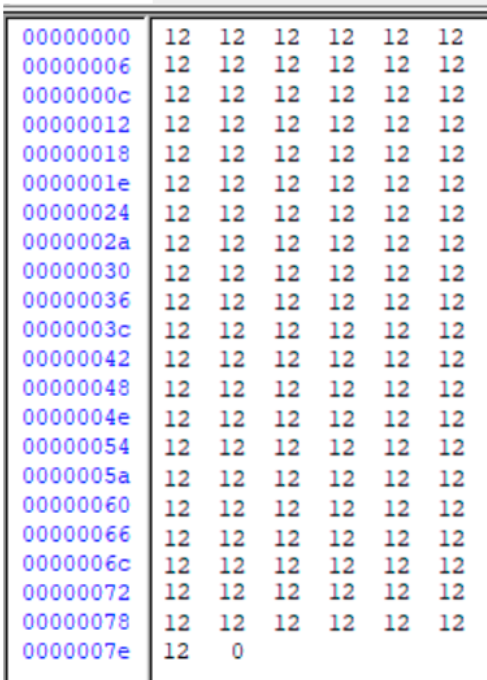


Рис. 14 – Данные в памяти гисторграммы

Как мы видим, результат соответствует ожиданиям.

## Создания модуля для тестирования на плате

Теперь разработаем модуль для тестирования на плате:

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

Автоматически созданное описание

Рис. 15 – Модуль db для тестирования на плате

## Настройка Signal Tap II

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

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рис. 16 – Настройка окна Signal Tap II

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

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 17 – Временные характеристики устройства

Теперь запустим и выполним проверку корректности работы программы на плате. Выполним загрузку разработанного модуля на плату и запустим тестирование, переведя RST в 0:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рис. 18 – Настройка ISSP

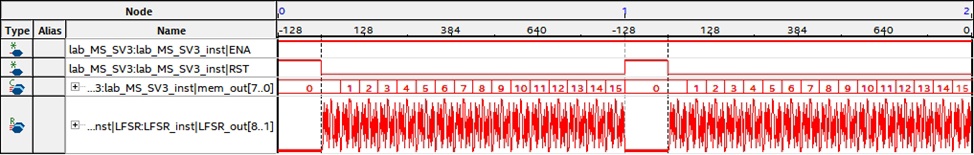


Рис. 19 –

Как мы видим, LFSR выдает случайные числа, а mem\_out увеличивается с 0 до 13 т.е. ведется подсчет циклов с начала работы.

## Доработка модуля гисторграммы

Теперь в соответствии с заданием в модуле histogram необходимо заменить mem\_arr на однопортовую память. Проблема состоит в том, что в один такт необходимо выполнить считывание значения из памяти и на основании этого записать в ту же ячейку новые данные. Выполнить это за один такт нереально, поэтому необходимо добавить PLL, который умножит внутреннюю частоту. Схема будет следующей: на фронте clk сохраняется значение на входе d\_in, после чего на спаде clk\_50 (clk с частотой в 2 раза большей) мы загрузим на вход памяти адрес, а на второй спад мы на вход памяти поместим обновленные данные:

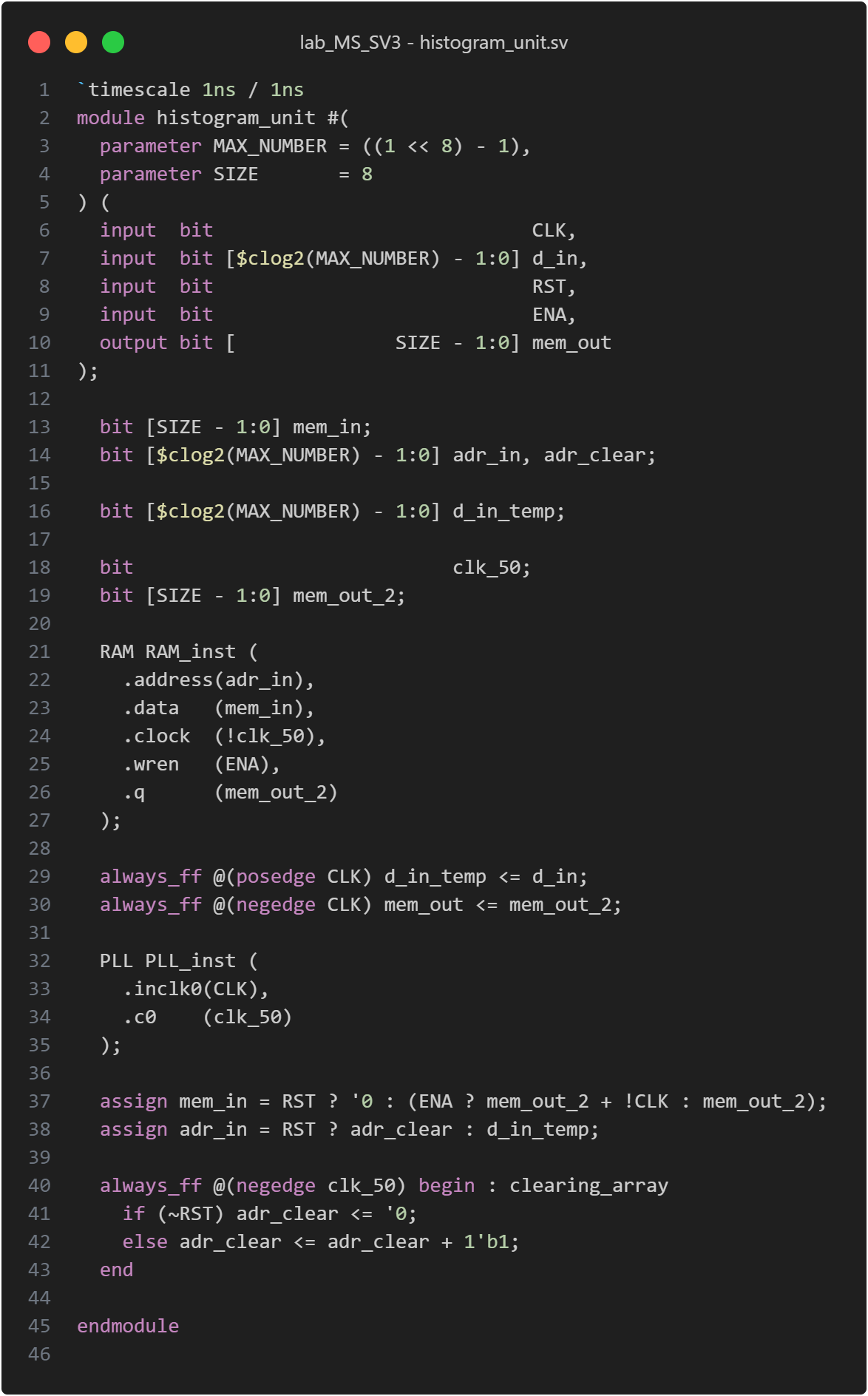


Рис. 20 – исправленный модуль гистограммы

Теперь память будет стираться в 2 раза быстрее (по 2 адреса за такт). Посмотрим на то, как выглядит RTL схема модуля:

*Изображение выглядит как диаграмма, линия, снимок экрана, План

Автоматически созданное описание*

Рис. 21 – Схема исправленного модуля гистограммы в RTL Viewer

Как мы видим, такой способ действительно помог избавиться от регистровой схемы памяти, однако сильно усложнил проект т. к. требует PLL.

Теперь необходимо повторить тестирование этого модуля, чтоб проверить корректность его работы:

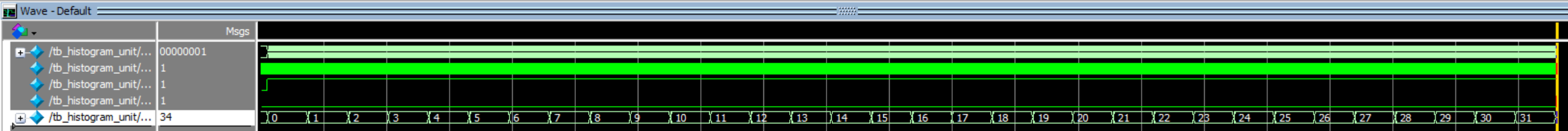


Рис. 22 – Результаты моделирования обновлённого тестового файла модуля гистограммы

Здесь же, открыв вкладку Memory List, можем увидеть данные в памяти:

Изображение выглядит как текст, снимок экрана, число

Автоматически созданное описание

Рис. 23 – Данные в памяти (2)

Как мы видим, модуль гистограммы корректно обрабатывает входную последовательность, а также очищает данные по RST.

Тестирование модуля верхнего уровня также повторило предыдущие тесты, поэтому перейдем сразу к тестированию на плате.

## Тестирование на плате

Важной особенностью новой памяти является то, что её можно посмотреть через ISMC. Если сделать это при RST = 1, мы увидим следующий результат:

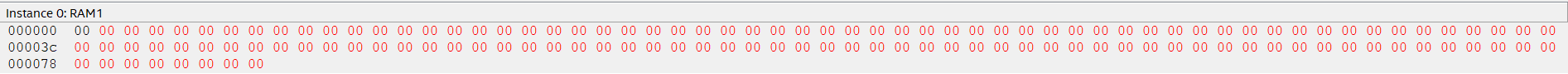


Рис. 24 – Память при RST = 1

И это логично т. к. память постоянно сбрасывается. Теперь поставим RST в 0 и считаем память:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 25 – Память при RST = 0

Как мы видим, 0 ячейка осталась 0 т. к. наш генератор не должен выдавать это число, а вот остальные ячейки имеют случайные значения, хотя ожидалось что они будут одинаковыми. Это связано с тем, что ISMC не работает на достаточных частотах и не может выгрузить всю память сразу, а делает это постепенно, откуда и возникают отличия.

В Signal Tap II увидим:

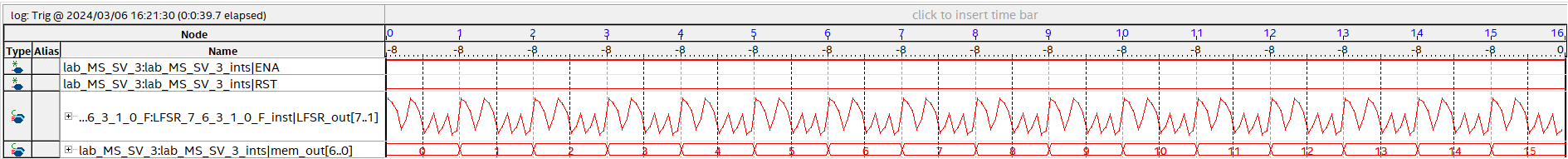


Рис. 26 – Результаты моделирования средствами Signal Tap II

Мы видим, что было зафиксировано 16 отрезков, когда встречалось требуемое число. Оно встречается раз в цикл (как и ожидалось), а также соседние измерения имеют одинаковую форму т. к. цикл формирует одинаковую последовательность чисел.

# Вывод

В результате выполнения лабораторной работы были успешно разработаны модули с использованием SystemVerilog, что позволило использовать параметризацию для легкой адаптации кода под различные конфигурации. Это значительно упростило разработку, делая ее более гибкой и обеспечивая легкость поддержки проекта при внесении изменений.

Дополнительное удобство заключается в эффективном использовании инструментов In-System Sources and Probes Editor и SignalTap II для отладки. Эти инструменты обеспечили быстрое выявление и исправление ошибок, что существенно ускорило процесс разработки и обеспечило успешное завершение проекта.

Практическое применение этого опыта раскрывается в контексте разработки сложных цифровых систем, таких как генераторы псевдослучайных последовательностей с последующим анализом данных. Применение SystemVerilog и современных инструментов отладки делает цифровое проектирование более гибким и адаптивным к изменяющимся требованиям проекта.