Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчёт по лабораторной работе № 5**

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

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

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Федотов (подпись)

“09” марта 2024 г.

Санкт-Петербург

2024

Оглавление

[1. Список иллюстраций: 2](#_Toc160856894)

[2. Цель упражнения: 3](#_Toc160856895)

[3. Алгоритм работы проекта: 3](#_Toc160856896)

[4. Решение: 3](#_Toc160856897)

[5. Вывод: 10](#_Toc160856898)

# Список иллюстраций:

[Рис. 4.1. RTL структура LFSR модуля. 3](#_Toc160856902)

[Рис. 4.2. Вейформа для модуля. 4](#_Toc160856903)

[Рис. 4.3. RTL Viewer для разработанного модуля. 5](#_Toc160856904)

[Рис. 4.4. Результат тестирования модуля. 6](#_Toc160856905)

[Рис. 4.5. Данные в памяти. 6](#_Toc160856906)

[Рис. 4.6. Результат тестирования модуля. 7](#_Toc160856907)

[Рис. 4.7. Данные в памяти гистограммы. 8](#_Toc160856908)

[Рис. 4.8. Настройки Signal Tap II. 8](#_Toc160856909)

[Рис. 4.9. Настройки ISSP. 8](#_Toc160856910)

[Рис. 4.10. Signal Tap II, результат. 8](#_Toc160856911)

[Рис. 4.11. RTL схема модуля. 9](#_Toc160856912)

[Рис. 4.12. Результат тестирования модуля. 9](#_Toc160856913)

[Рис. 4.13. Данные в памяти. 10](#_Toc160856914)

[Рис. 4.14. Настройки Signal Tap II. 10](#_Toc160856915)

[Рис. 4.15. Память при RST = 1. 10](#_Toc160856916)

[Рис. 4.16. Память при RST = 0. 10](#_Toc160856917)

[Рис. 4.17. Signal Tap II. 10](#_Toc160856918)

# Цель упражнения:

Пройти цикл проектирования в рамках пакетов Quartus и ModelSim, включая следующие этапы:

* Создание проекта.
* Разработка описания модулей с использованием конструкций расширения SystemVerilog.
* Разработка теста на языке SystemVerilog и моделирование.
* Отладка проекта.

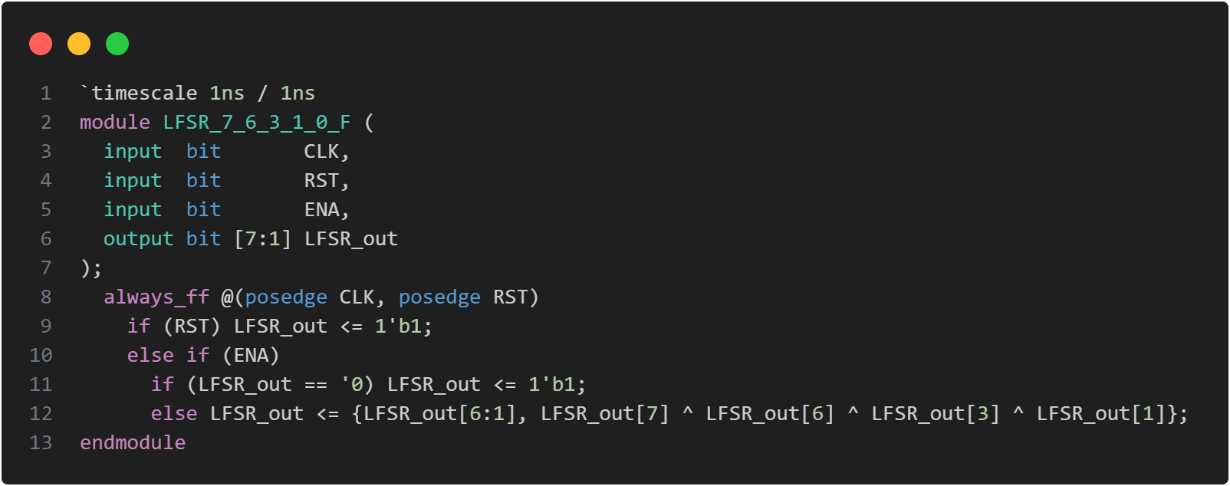
# Алгоритм работы проекта:

Разработать описание LFSR по следующему полиному (вариант 18):

Типом реализации логического элемента в обратной связи – XOR.

# Решение:

Разработаем LFSR модуль:



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

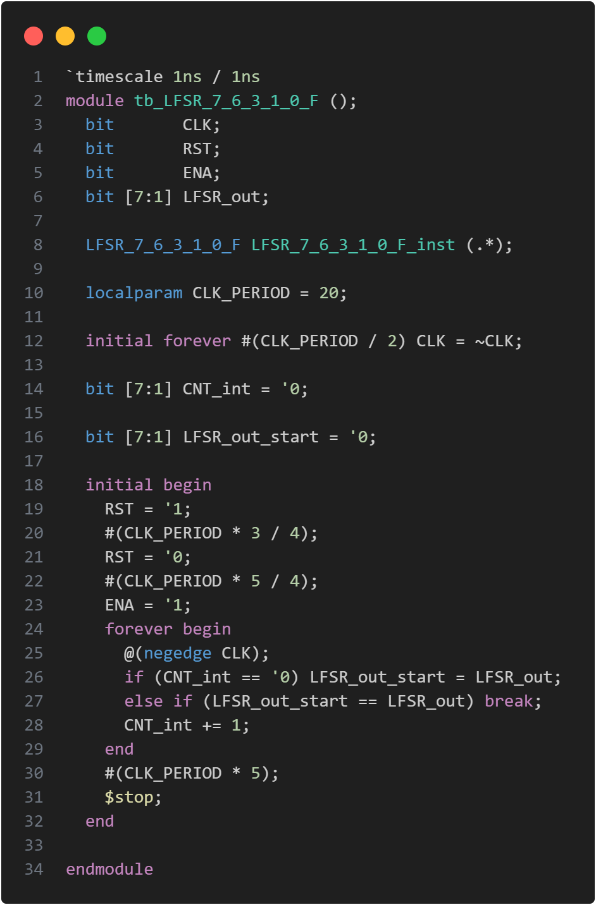
Получившаяся RTL структура выглядит следующим образом:

Изображение выглядит как диаграмма, линия, снимок экрана, График

Автоматически созданное описание

Рис. 4.1. RTL структура LFSR модуля.

Теперь составим тестовый модуль для данного модуля:



Данный модуль позволит нам посчитать период. Т.к. степень полинома 7, мы ожидаем период равный 127 (.

Получившаяся вейформа выглядит следующим образом:

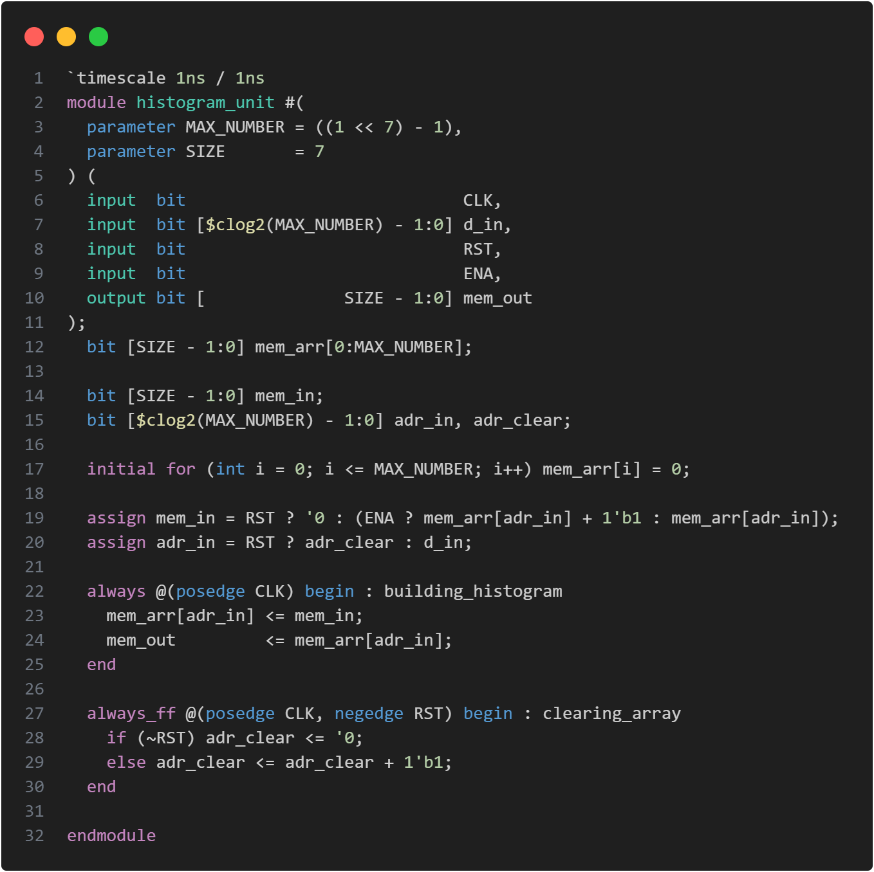
Изображение выглядит как снимок экрана, электроника, Мультимедийное программное обеспечение, программное обеспечение

Автоматически созданное описание

Рис. 4.2. Вейформа для модуля.

Как мы видим, мы получили ожидаемый период в 127 единиц.

Теперь создадим модуль для создания 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:

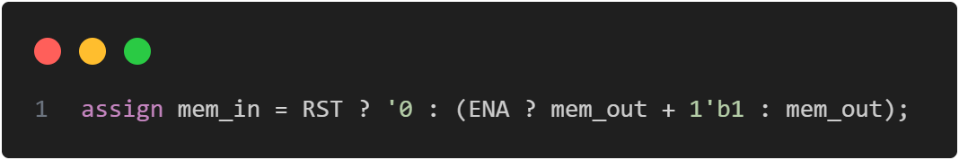
Изображение выглядит как линия, антенна

Автоматически созданное описание

Рис. 4.3. RTL Viewer для разработанного модуля.

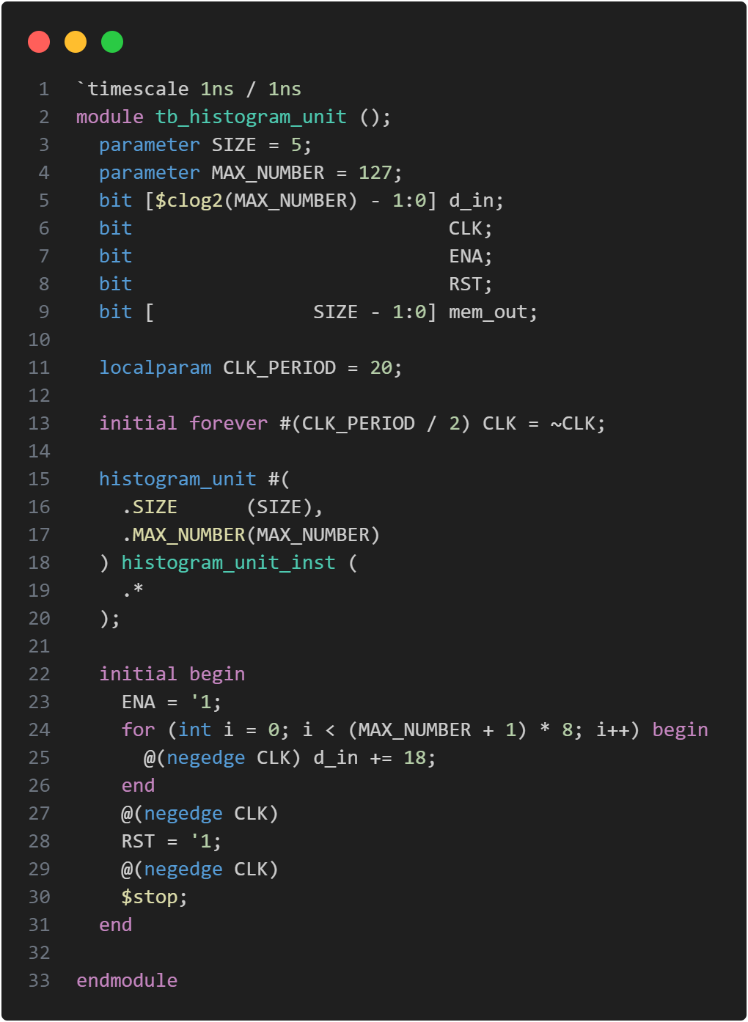
Как можно заметить, тут нет как таковой памяти, все строится на регистрах. Причина этого в том, что мы пытаемся одновременно читать значения из памяти и записывать в неё, из-за чего квартус решает, что это лучше сделать, используя регистровые схемы.

Это можно исправить, заменив строку 19 на следующую:



Однако это приведет к тому, что вместо увеличения заданного значения на 1, мы будем записывать результат суммы в следующую цифру, поданную на вход. В контексте нашей задачи это практически не повлияет на результат (если мы изначально ожидаем корректность разработанного модуля), однако по моему мнению лучше использовать схему на регистрах. Как от неё избавиться будет рассмотрено позже.

Теперь напишем тест для созданного модуля:



Запустим этот тестовый модуль:

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

Автоматически созданное описание

Рис. 4.4. Результат тестирования модуля.

Посмотрим данные в памяти:

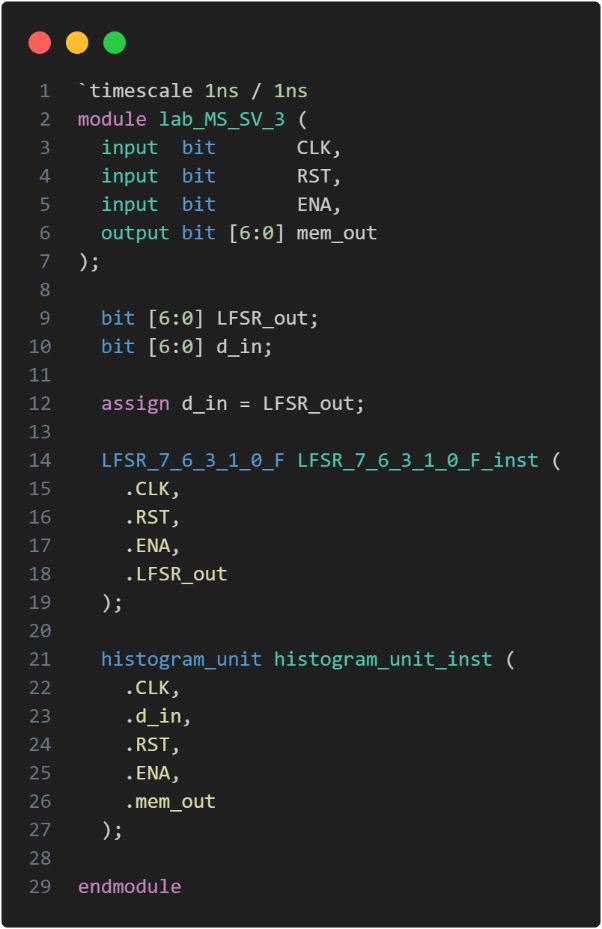
Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

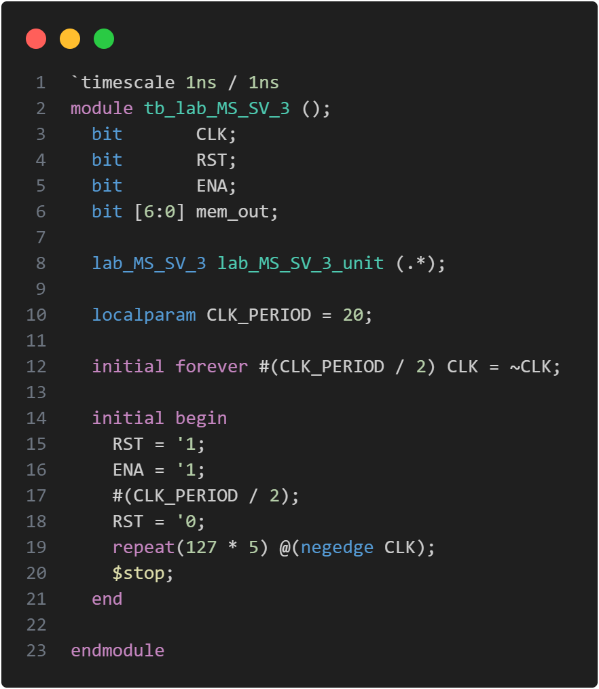
Рис. 4.5. Данные в памяти.

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

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



Теперь напишем тестовый модуль:



Мы 5 раз повторяем период, который был получен ранее, ожидая, что в гистограмме все значения от 1 до 127 будут равны 5. Проверим это:

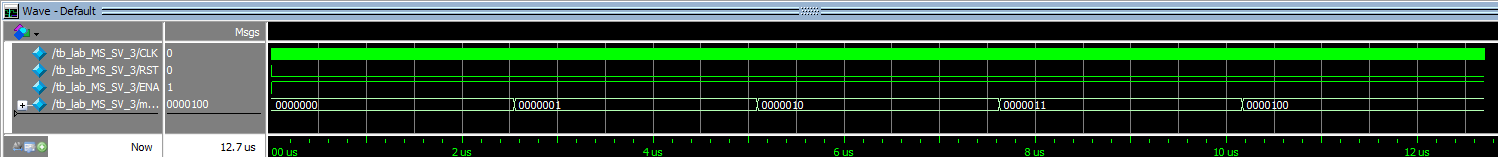


Рис. 4.6. Результат тестирования модуля.

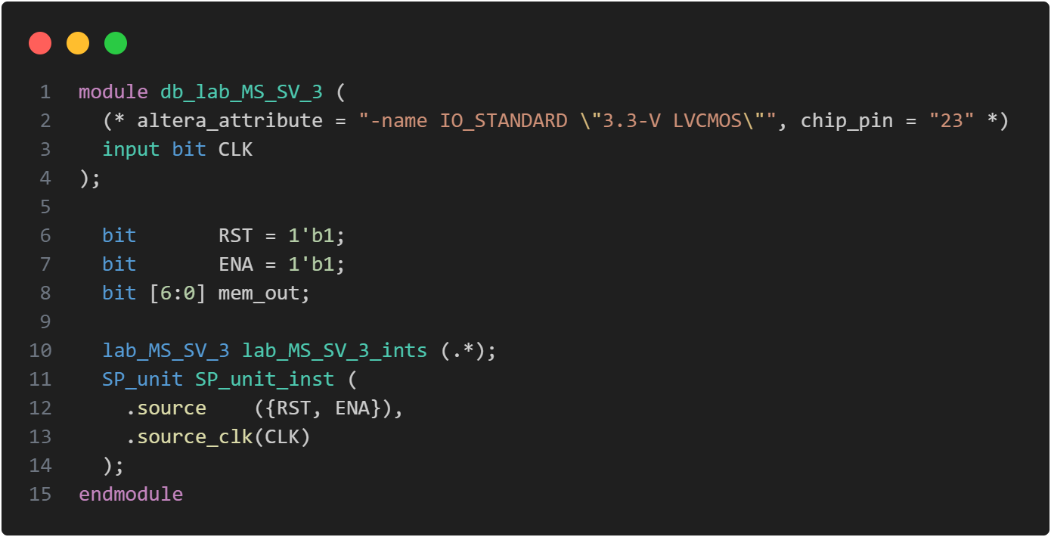
Изображение выглядит как текст, снимок экрана, число

Автоматически созданное описание

Рис. 4.7. Данные в памяти гистограммы.

Как мы видим, результат соответствует ожиданиям.

Теперь разработаем модуль для тестирования на плате:



Используя ISSP, будем редактировать сигналы RST и ENA, а благодаря Signal Tap II наблюдать за результатом:

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

Рис. 4.8. Настройки Signal Tap II.

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. 4.9. Настройки ISSP.

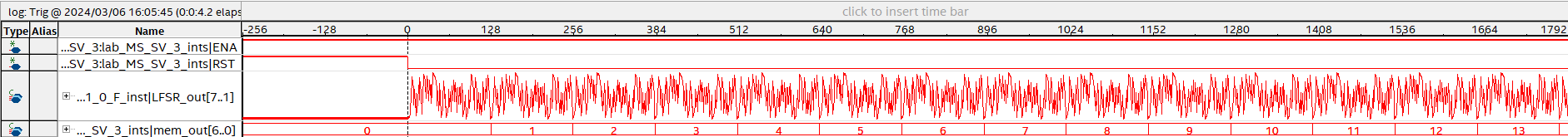
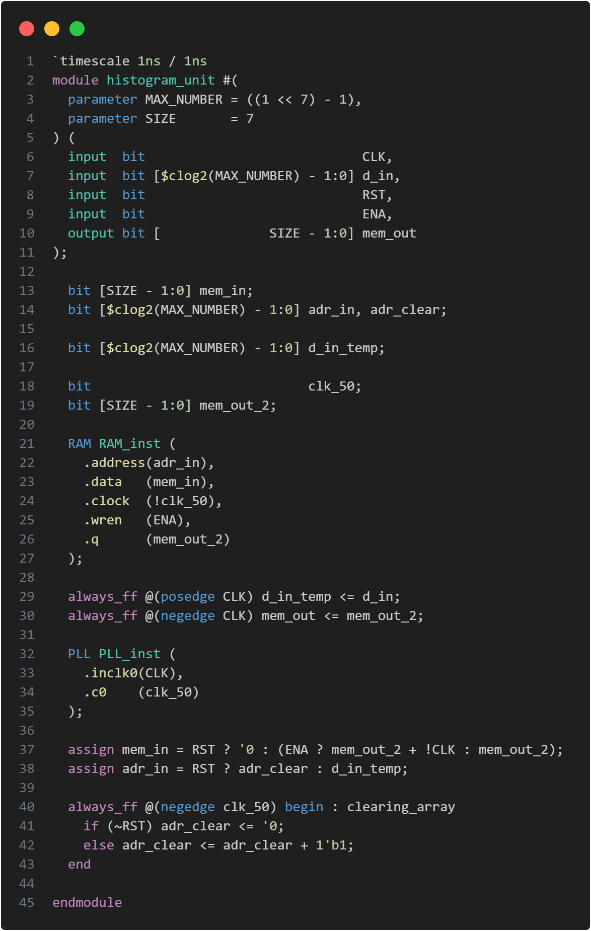


Рис. 4.10. Signal Tap II, результат.

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

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:



Стоит отметить, что 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. 4.11. RTL схема модуля.

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:

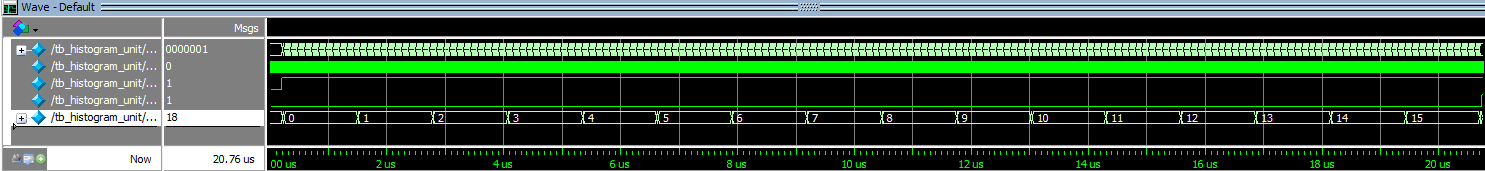


Рис. 4.12. Результат тестирования модуля.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рис. 4.13. Данные в памяти.

Как мы видим, результаты соответствуют ожиданиям.

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

Настроим Signal Tap II следующим образом:

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Автоматически созданное описание

Рис. 4.14. Настройки Signal Tap II.

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

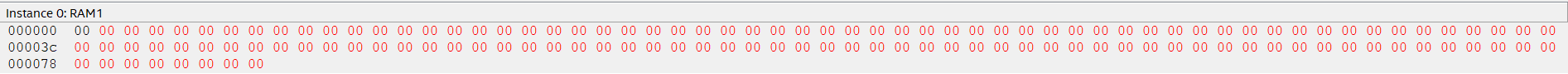


Рис. 4.15. Память при RST = 1.

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. 4.16. 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:

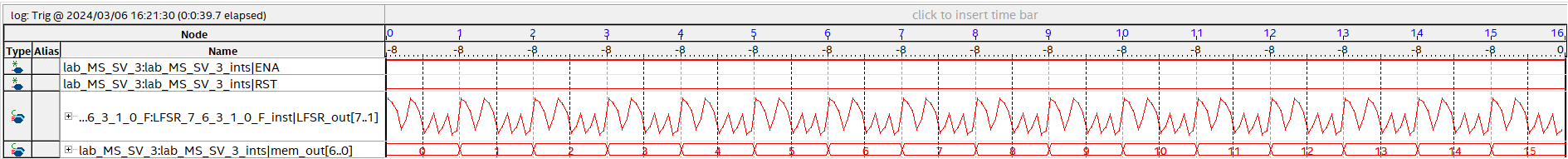


Рис. 4.17. Signal Tap II.

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 ходе лабораторной работы успешно пройден цикл проектирования, начиная с создания проекта и разработки модулей с использованием расширений SystemVerilog. Использование SystemVerilog предоставило широкий спектр новых возможностей по сравнению с Verilog, облегчая процесс разработки и улучшая читаемость кода.

Отладка проекта осуществлялась с помощью инструментов In-System Sources and Probes Editor и SignalTap II, что значительно повысило эффективность процесса. Эти инструменты позволили быстро выявить и исправить ошибки, что является ключевым аспектом при работе с любым проектом.