|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА - Российский технологический университет»**  **РТУ МИРЭА** | | |
| [**Институт перспективных технологий и индустриального программирования**](https://www.mirea.ru/docs/250117) | | |
| **Кафедра наноэлектроники** | | |

|  |  |
| --- | --- |
| **ОТЧЕТ** | |
| **ПО ПРЕДДИПЛОМНОЙ ПРАКТИКЕ** | |
|  | |
| **Тема: «Модуль сбора и обработки данных от КМОП биосенсоров»** | |
| Студент группы ЭНБО-01-21 | Матевосов И.В. |
| Руководитель практики  *доцент, к.т.н.* | Певцов Е.Ф. |

|  |  |  |
| --- | --- | --- |
| «Отчет представлен к рассмотрению» | «\_\_» \_\_\_\_\_\_202\_\_ г. |  |
|  |  |  |
| «Отчет утвержден.  Допущен к защите.» | «\_\_\_» \_\_\_\_\_\_\_\_\_\_202\_\_ г. |  |

Москва, 2025

Оглавление

[Введение 3](#_Toc198232470)

[Постановка задачи 3](#_Toc198232471)

[Выбор инструментов и сред разработки 5](#_Toc198232472)

[Этапы разработки системы 9](#_Toc198232473)

[Верификация и моделирование 14](#_Toc198232474)

[Анализ результатов 17](#_Toc198232475)

[Заключение 19](#_Toc198232476)

# Введение

Целью прохождения преддипломной практики являлась разработка цифровой системы сбора и обработки данных с КМОП-биосенсоров, реализованной на языке описания аппаратуры Verilog. Основной задачей было создание и верификация модуля, способного принимать данные от аналогово-цифрового преобразователя (АЦП) через интерфейс SPI, выполнять математическую обработку сигнала в режиме реального времени, производить логарифмическое преобразование, а также формировать результат для последующей передачи или анализа.

В ходе практики особое внимание было уделено вопросу преобразования сигнала биосенсора в логарифмическую шкалу. Такая форма представления данных позволяет наиболее эффективно отслеживать динамику изменения порогового напряжения на выходе чувствительного элемента, связанного с концентрацией исследуемого вещества.

Проект охватывает полный цикл цифровой обработки сигнала: от моделирования SPI-протокола и буферизации с помощью FIFO, до визуализации результатов с применением Python и библиотеки matplotlib. Дополнительно в ходе работы были освоены средства симуляции (Icarus Verilog), анализа временных диаграмм (GTKWave), работы с системой контроля версий Git и оформлением проекта в виде репозитория GitHub.

Практика стала неотъемлемой частью подготовки к выполнению выпускной квалификационной работы и позволила на практике применить полученные знания в области цифровой схемотехники, микросистемной электроники и обработки данных.

# Постановка задачи

В современных микросистемах, ориентированных на биомедицинские измерения, особое внимание уделяется разработке компактных и энергоэффективных узлов сбора и обработки данных. Одним из таких примеров являются КМОП-биосенсоры, которые преобразуют воздействие внешней среды (например, изменение ионной концентрации) в электрический сигнал, чаще всего представленный в виде малых токов или напряжений. Эти сигналы требуют последующей цифровой обработки, включающей преобразование, фильтрацию и анализ отклонений.

В рамках преддипломной практики ставилась задача разработать модуль сбора и обработки данных с КМОП-биосенсоров, пригодный для реализации на программируемой логике (ПЛИС), и выполняющий следующие функции:

* Приём данных от АЦП по SPI-интерфейсу.
* Математическая обработка в реальном времени.
* Оценка сдвига логарифмической функции.
* Буферизация данных с использованием FIFO.
* Визуализация результатов.

АЦП подключается к внешнему аналоговому выходу биосенсора и передаёт оцифрованные значения через последовательный интерфейс. Изначально работа велась с 12-битным потоком, затем формат был расширен до 16 бит для увеличения точности. Сигнал, принятый по SPI, преобразуется в напряжение в милливольтах, затем подаётся на блок аппроксимации логарифмической функции. В частности, реализовано приближённое вычисление натурального логарифма на базе кусочно-линейной интерполяции. Это позволяет отслеживать экспоненциальные изменения, характерные для физики МОП-структур. Важной целью обработки является оценка сдвига значений ln(V) между соседними измерениями:

Такой сдвиг может отражать изменение порогового напряжения транзистора, связанное с воздействием биологического агента.

Результаты обработки хранятся в буфере FIFO, откуда могут быть считаны либо для передачи в ПК, либо для дальнейшего использования в системе. FIFO реализован в 16-битном формате и управляется сигналами wr\_en и rd\_en.

С помощью внешнего скрипта на Python, эмулирующего UART-приём данных, реализовано построение графика логарифмического сдвига Δx. Это позволяет продемонстрировать поведение модели в виде кривой, имитирующей реальный отклик биосенсора. Система охватывает полный цифровой путь от аналогового сигнала до визуального представления данных, и может быть масштабирована под конкретные задачи анализа встраиваемых сенсорных устройств.

# Выбор инструментов и сред разработки

В процессе выполнения практики было принято решение использовать инструменты, ориентированные на разработку и моделирование цифровых систем на языке описания аппаратуры Verilog, а также на обработку и визуализацию данных, полученных в результате моделирования.

В качестве основного языка проектирования цифровой логики был выбран Verilog HDL (Hardware Description Language). Язык позволяет описывать параллельную структуру аппаратных модулей, синтезируемых на программируемых логических интегральных схемах (ПЛИС). Verilog активно применяется в промышленной и исследовательской практике, поддерживается большинством САПР, и предоставляет необходимые средства для описания интерфейсов, автоматов состояний, регистров, буферов и арифметических блоков.

В качестве симулятора был использован Icarus Verilog — свободно распространяемая система для компиляции Verilog-кода и запуска временного моделирования. Она позволяет запускать тестбенчи, выполнять отладку логики и генерировать файлы временных диаграмм.



Рисунок 1 – Логотип программного обеспечения Icarus Verilog.

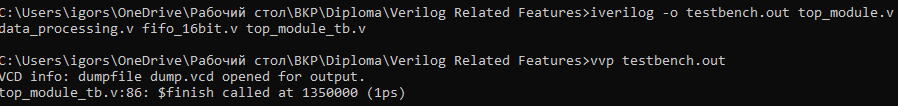


Рисунок 2 – Пример запуска симуляции с помощью команды iverilog.

На рис. 2 показаны команды, исполняемые в ходе симуляции работы модулей, написанных на Verilog. -o testbench.out – вывод результатов и название файла вывода, дальше аргументы – названия файлов, которые содержат модули, использующиеся в коде. Команда vvp выдаёт результат в виде сигналов, которые затем можно визуально отобразить.

Для анализа поведения сигналов во времени использовалась программа GTKWave, читающая файлы формата .vcd, сгенерированные симулятором. В GTKWave удобно отслеживать фронты тактовых импульсов, сигналы чтения и записи, переходы между состояниями в регистрах и буферах. GTKWave предоставила ключевые подтверждения корректной работы SPI-интерфейса, FIFO и блока обработки данных.

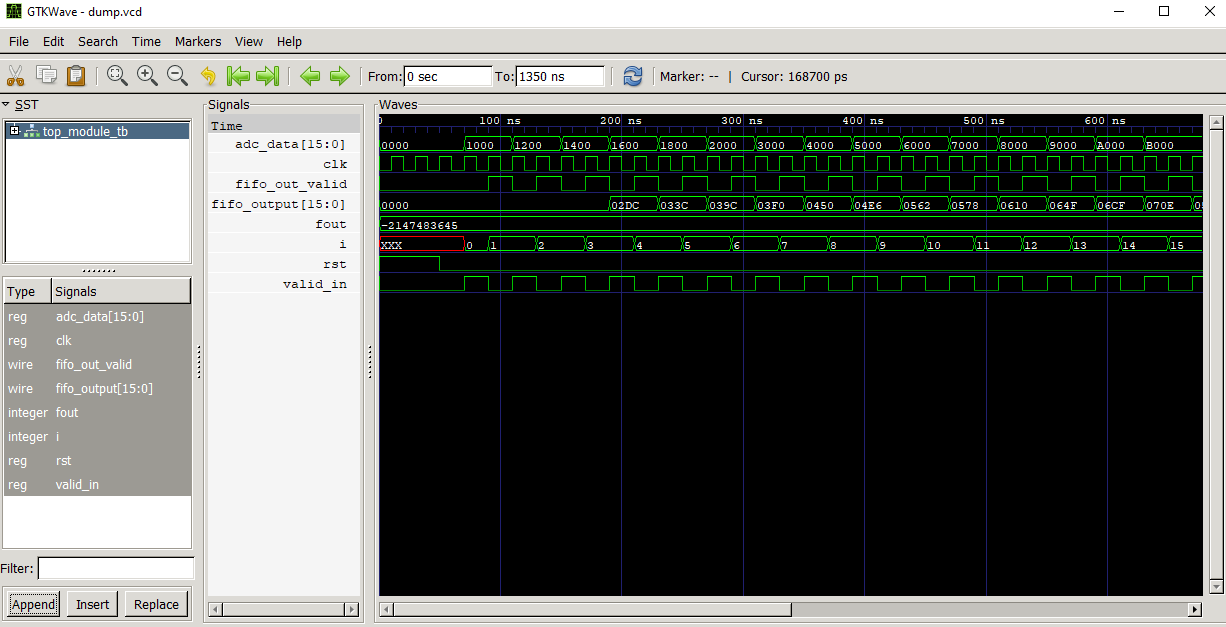


Рисунок 3 – Рабочий интерфейс программы GTKWave, на временной диаграмме результаты выполнения testbench.out.

Разработка кода велась в редакторе VS Code, настроенном с поддержкой синтаксиса Verilog и автозапуска симуляций. Использование расширений (например, Verilog-HDL/SystemVerilog) упростило отладку и повысило читаемость кода.

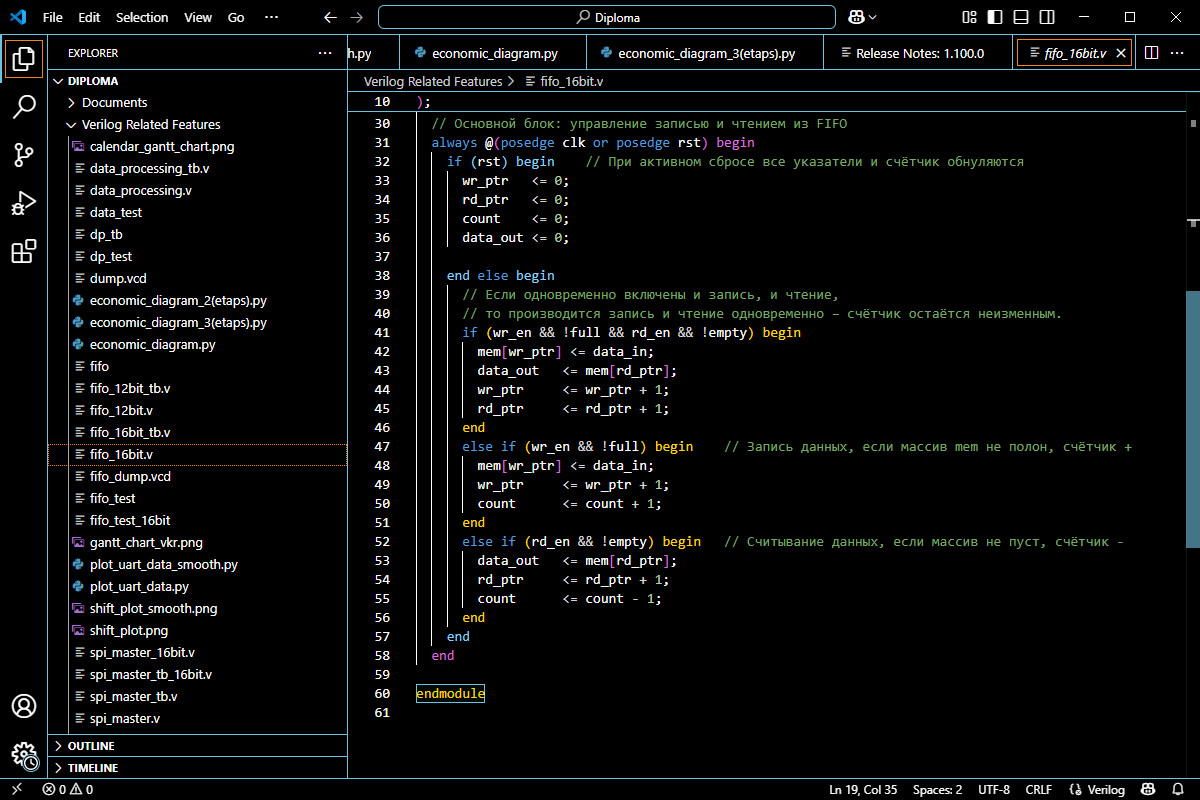


Рисунок 4 – Интерфейс редактора кода Visual Studio Code.

Для обеспечения сохранности проекта и контроля версий использовалась система управления версиями Git. Все ключевые файлы проекта (модули, тестбенчи, результаты) хранились в репозитории GitHub. Актуальная ссылка на репозиторий: https://github.com/SuNGviN/Diploma. Это обеспечило удобный способ отката к предыдущим версиям, а также продемонстрировало владение инструментами командной разработки.

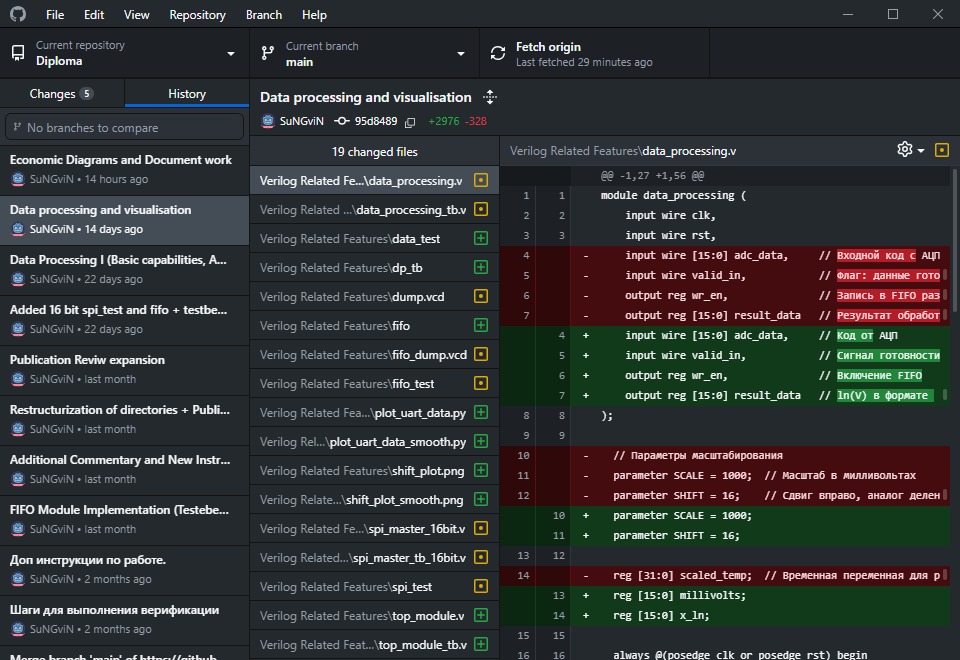


Рисунок 5 – Интерфейс приложения GitHub, версии для рабочего стола.

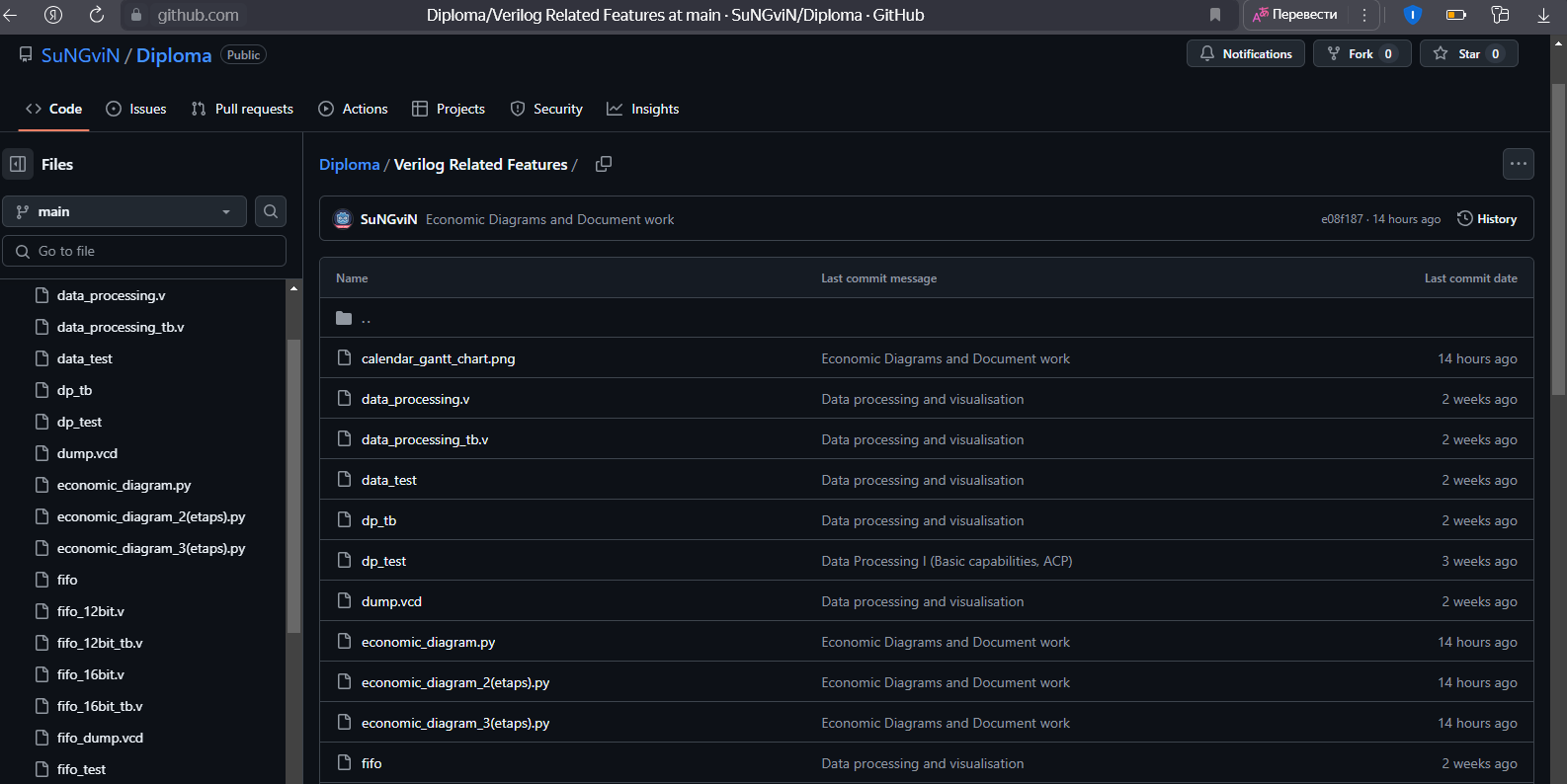


Рисунок 6 –Интерфейс веб-версии GitHub с репозиторием дипломной работы.

На этапе визуализации полученных цифровых результатов применялись Python 3.11 и библиотека matplotlib. Данные, полученные из Verilog (моделируемая UART-передача), сохранялись в текстовый файл и далее использовались для построения графиков логарифмического сдвига Δx.

# Этапы разработки системы

На первом этапе был создан модуль, реализующий приём данных с внешнего АЦП по интерфейсу SPI (Serial Peripheral Interface). Этот протокол широко используется для синхронного обмена данными между микросхемами. Изначально модуль работал с 12-битными словами, затем был модифицирован под 16-битный формат, более характерный для высокоточных АЦП.

Ключевые компоненты SPI-модуля:

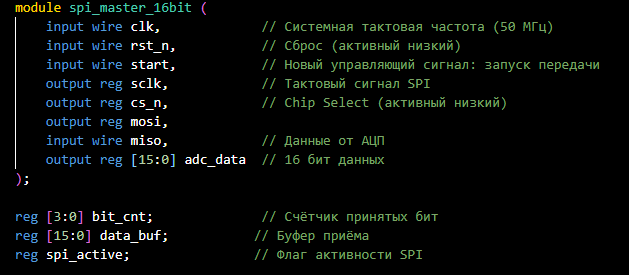


Рисунок 7 – Входные и выходные сигналы модуля spi\_master.

Сигналами являются sclk, cs\_n, mosi, miso. В модуле также присутствует счётчик битов и буфер приёма входных битов из miso.

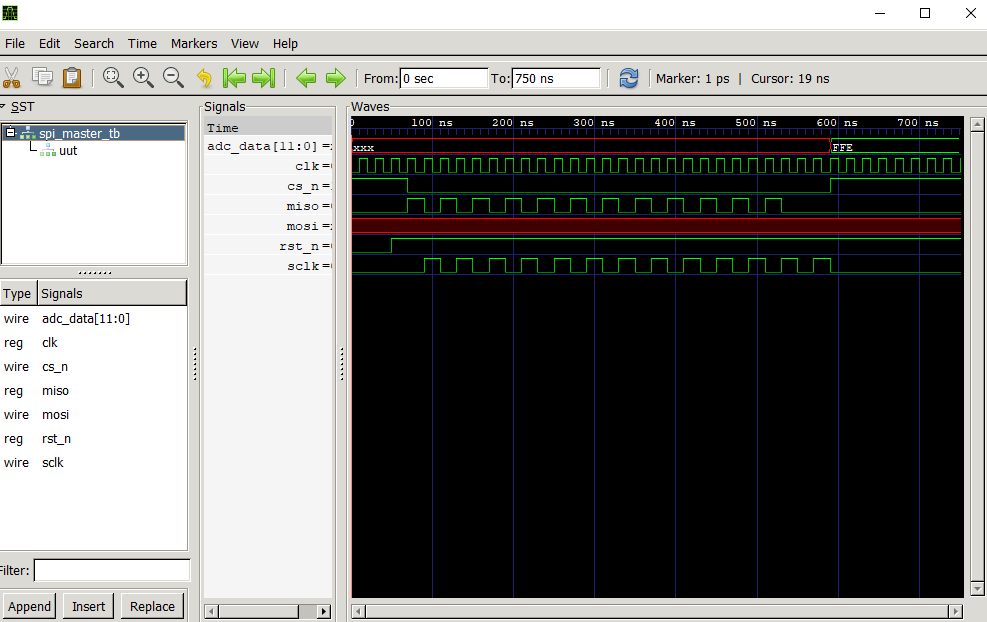


Рисунок 8 – Временная диаграмма работы модуля spi\_master.

Модуль был протестирован с помощью Verilog-тестбенча, имитирующего передачу данных. Диаграммы GTKWave подтвердили корректность работы счётчика, захвата битов и формирования готового результата.

Следующим этапом стал модуль FIFO (First In, First Out), предназначенный для временного хранения результатов обработки. Буфер обеспечивает раздельное управление чтением и записью, позволяет сгладить разницу в скорости поступления и обработки данных.

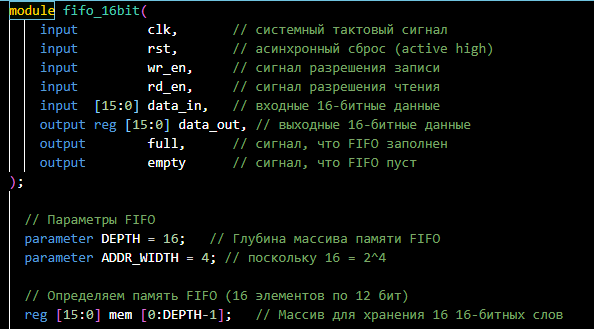


Рисунок 9 – Внутренние параметры модуля fifo.

Особенности реализации модуля: ширина данных 16 бит, управляющими сигналами являются wr\_en, rd\_en, full, empty, чтение данных происходит только, когда empty = 0.

Модуль также прошёл тестирование с записью и считыванием нескольких значений. GTKWave отразил работу указателей записи/чтения и заполнение буфера.

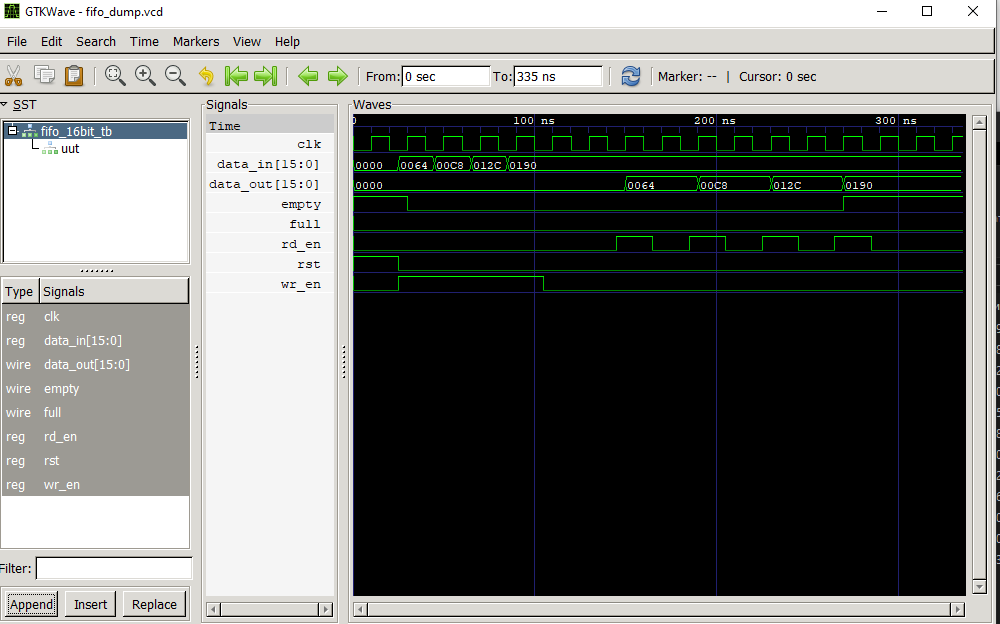


Рисунок 10 – Чтение поступающих данных и запись буфер fifo.

Центральной частью системы стал модуль, выполняющий перевод кода АЦП в физическое значение напряжения и последующую аппроксимацию логарифма этого значения. Работа велась в формате фиксированной запятой (8.8 – 8 бит отводится под целую часть числа, 8 – под часть после запятой), что позволило избежать необходимости в аппаратной арифметике с плавающей точкой.

Обработка включает:

* Перевод кода в милливольты:

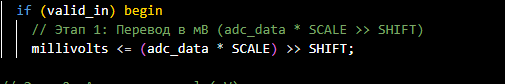


Рисунок 11 – Перевод бинарного кода из adc\_data в милливольты. SCALE = 1000, SHIFT = 16.

* Аппроксимацию ln(V) через кусочно-линейные зависимости (вычисление с коэффициентами a, b в разных интервалах значений)

Вычисление сдвига:

Также была реализована логика автоматического хранения предыдущего значения x\_prev, что позволило рассчитывать сдвиг между измерениями в реальном времени.

Для объединения всех компонентов был создан модуль top\_module.v, включающий:

* SPI-приёмник
* Модуль обработки
* FIFO-буфер

Сигналы были согласованы между модулями:

adc\_data → обработка

result\_data → FIFO

fifo\_output → UART / ПК

Также был добавлен механизм автоматического чтения FIFO, имитирующий последовательную передачу данных наружу.

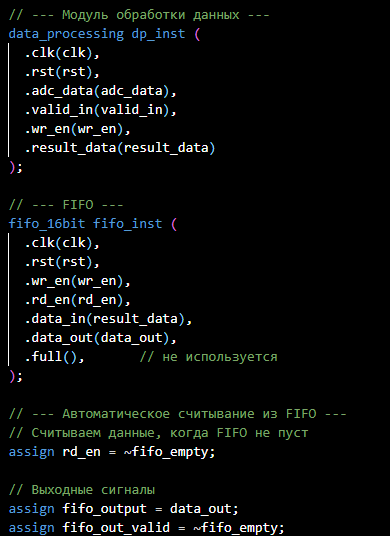


Рисунок 12 – Инстанцирование модулей fifo и data\_processing внутри top\_module, вывод данных, если буфер fifo не пуст.

На заключительном этапе был создан симуляционный интерфейс вывода, данные из FIFO автоматически записывались в файл uart\_output.txt, где каждое значение представляло собой логарифмический сдвиг Δx. Далее использовался Python-скрипт для чтения файла и построения графика.

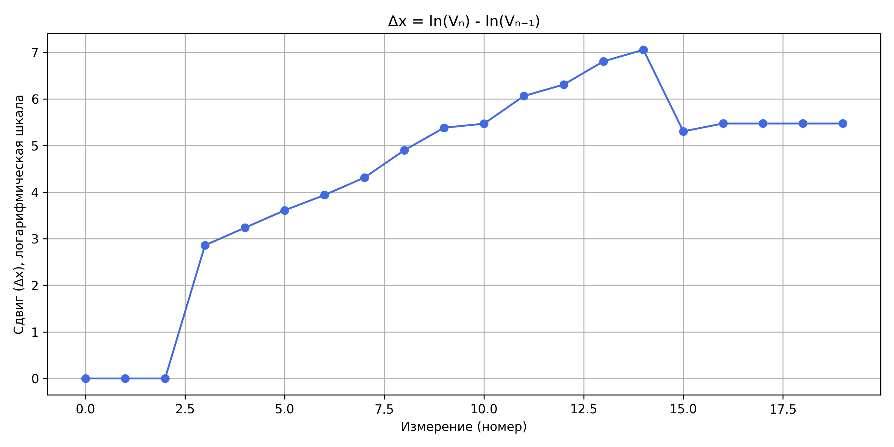


Рисунок 13 – Результат сбора и обработки информации.

На итоговом графике видна фоновая стабильность сигнала, резкий рост при увеличении напряжения на участке измерений №4–14, плато (насыщение) и последующий спад сигнала.

# Верификация и моделирование

На каждом этапе разработки всех основных модулей системы была проведена поэтапная верификация каждого блока с использованием инструментов симуляции и анализа временных диаграмм. Основная цель верификации — убедиться в корректной работе цифровой логики, соответствии поведения ожиданиям и адекватности получаемых результатов.

Сначала верификация производилась на уровне SPI-модуля. С помощью Verilog-тестбенча в линию miso подавалась имитированная 16-битная последовательность, например 1010101010101010. Наблюдая в GTKWave сигналы cs\_n, sclk и состояние счётчика, удалось убедиться, что каждый бит корректно принимается и сдвигается в буфере.

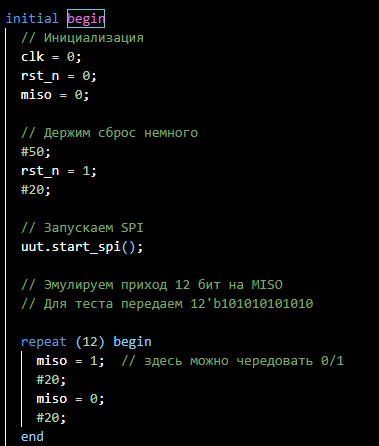


Рисунок 14 – Тестбенч для верификации SPI модуля.

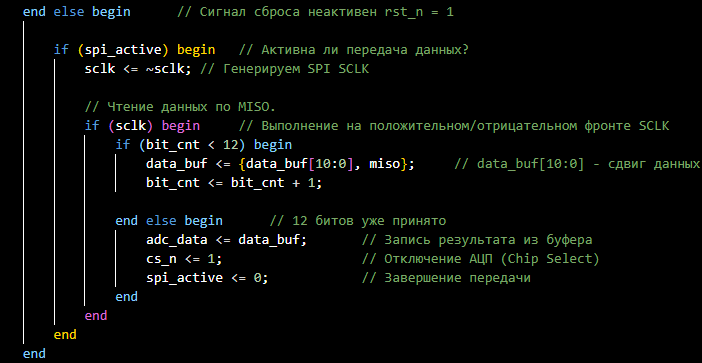


Рисунок 15 – Процесс чтения данных по miso.

В момент окончания приёма формировался сигнал adc\_data, фиксирующий полное 16-битное значение (см. рис. 16). Сигнал cs\_n автоматически переходил в высокий уровень, завершая транзакцию (см. рис. 17).

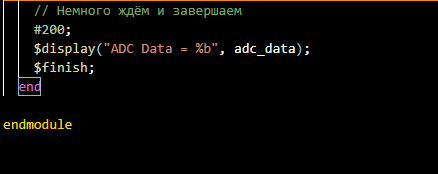


Рисунок 16 – Ожидание 200 нс, и вывод данных с помощью $display.

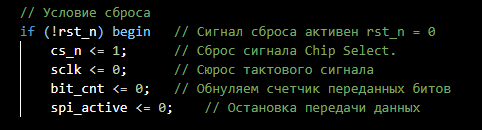


Рисунок 17 – Завершение процесса передачи данных в spi\_module.

Следующим этапом стало моделирование FIFO, обеспечивающего буферизацию между асинхронными потоками записи и чтения. В отдельном тестбенче выполнялась запись и считывание четырёх различных значений. С помощью GTKWave были проверены нарастание сигнала full при заполнении буфера, появление empty = 0 при записи, правильная последовательность выдачи данных при чтении

Особое внимание уделялось проверке расчётов внутри модуля обработки. В качестве теста использовались значения АЦП: 0x4000, 0x8000, 0xC000, 0xFFFF, соответствующие напряжениям примерно от 0.4 В до 1.6 В.

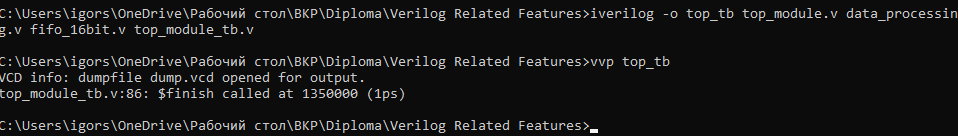


Рисунок 18 – Команды, необходимые для исполнения верификации top\_module.

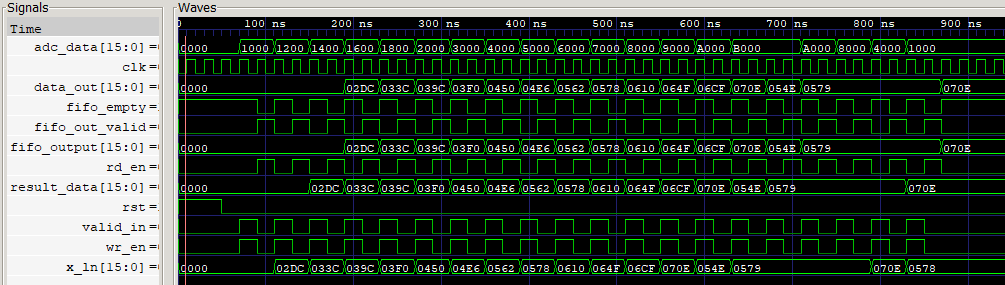


Рисунок 19 – Временная диаграмма из GTKWave модуля top\_module.

В GTKWave отслеживались сигналы:

millivolts — правильность масштабирования

x\_ln — значение логарифма

result\_data — сдвиг Δx

wr\_en — момент готовности нового результата

Поскольку в рамках практики не осуществлялась работа с реальной ПЛИС, было принято решение смоделировать передачу результата через UART. Для этого при каждом fifo\_out\_valid = 1 значение fifo\_output записывалось в файл uart\_output.txt, запись осуществлялась средствами Verilog ($fdisplay), каждая строка файла соответствовала одному измерению Δx

Для построения графика был разработан скрипт на Python с использованием библиотеки matplotlib. На основе содержимого uart\_output.txt создавался график зависимости логарифмического сдвига Δx от номера измерения.

# Анализ результатов

Для тестирования обработки и визуализации была использована последовательность из 20 входных значений adc\_data. Поведение графика отражает типичный отклик биосенсора при воздействии внешнего агента (например, ионов, молекул или температуры).

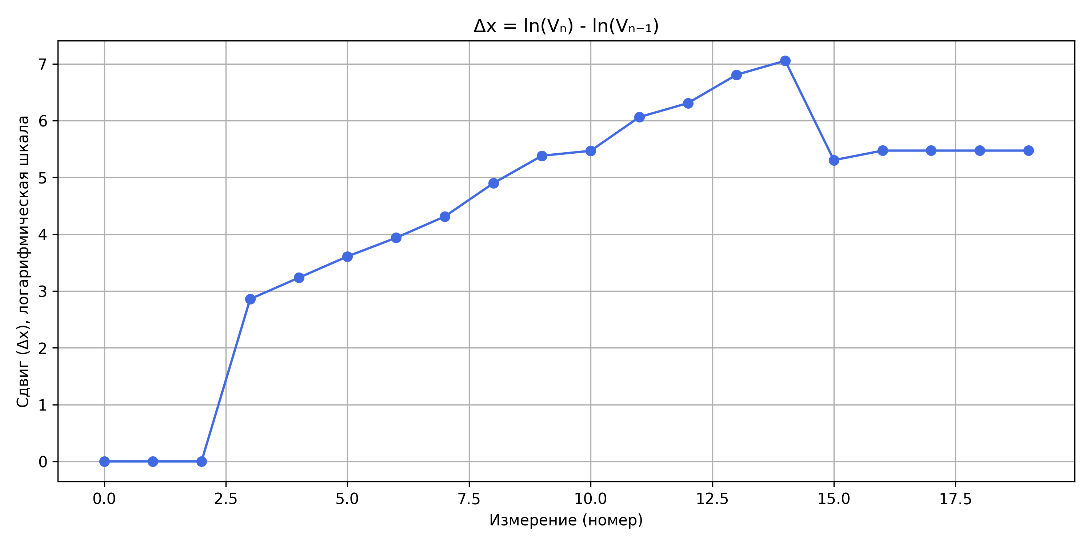


Рисунок 20 – Зависимость значений сдвига от № измерений.

Построенный график показывает изменение логарифма входного напряжения по сравнению с предыдущим измерением:

Когда значения Δx близки к нулю, сигнал стабилен, входное напряжение не изменяется, затем наблюдается быстрый рост Δx, соответствующий экспоненциальному подъёму напряжения. После насыщения логарифм выравнивается, изменения Δx становятся минимальными.

Такой график легко интерпретируется, является устойчивым к шумам и может служить показателем изменения физико-химических параметров среды.

Достоинства выбранного подхода:

* Логарифмическое преобразование усиливает малые изменения на низких уровнях сигнала.
* Переход от абсолютных значений к дифференциальному анализу (Δx) позволяет отсекать постоянную составляющую
* Модуль может быть использован с любым типом биосенсоров, выдающих аналоговый отклик

Формат хранения результатов в FIFO и организация передачи по UART (или через лог-файл) делает систему легко встраиваемой в реальное микропроцессорное или ПЛИС-решение. Возможные пути масштабирования:

1. Подключение нескольких каналов
2. Применение фильтрации
3. Построение диаграмм в реальном времени

# Заключение

В ходе преддипломной практики была выполнена полноценная разработка цифровой подсистемы сбора и обработки данных с КМОП-биосенсоров, реализованной на языке описания аппаратуры Verilog. Система охватывает все ключевые этапы обработки сигнала — от приёма оцифрованных данных по SPI-интерфейсу до расчёта логарифмического сдвига и формирования результирующего потока информации.

В процессе работы были успешно достигнуты следующие результаты:

* Освоены принципы работы интерфейса SPI и реализован приём 16-битных данных от внешнего источника (АЦП);
* Создан модуль буферизации (FIFO), обеспечивающий временное хранение результатов;
* Разработан математический модуль обработки, включающий преобразование данных в милливольты, аппроксимацию натурального логарифма и расчёт его сдвига между последовательными измерениями;
* Интегрирован верхнеуровневый модуль top\_module, объединяющий все элементы системы в единую архитектуру;
* Проведено тестирование с помощью симулятора Icarus Verilog и анализ временных диаграмм в GTKWave;
* Организован вывод результатов обработки в файл, эмулирующий передачу по UART;
* Выполнена визуализация логарифмического сдвига Δx с использованием Python и библиотеки matplotlib;

Полученные графики подтверждают корректность работы системы и возможность анализа динамики отклика биосенсора.

В процессе практики были также освоены современные средства командной и индивидуальной инженерной работы: система контроля версий Git, работа с GitHub, среда Visual Studio Code, а также приёмы структурирования проекта и подготовки отчётной документации.

Полученные навыки и программно-аппаратные наработки послужили основой для выполнения выпускной квалификационной работы и продемонстрировали готовность к решению инженерных задач в области цифровой обработки сигналов и микросистемной электроники.