|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ РАДИОЭЛЕКТРОНИКИ И ЛАЗЕРНОЙ ТЕХНИКИ

КАФЕДРА ТЕХНОЛОГИИ ПРИБОРОСТРОЕНЯ (РЛ-6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ **11.05.01 Радиоэлектронные системы и комплексы**

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОЙ РАБОТЕ***

***ПО ДИСЦИПЛИНЕ***

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_Цифровые устройства и микропроцессоры\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

***НА ТЕМУ:***

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_Алгоритмы шифрования в цифровом логическом\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_анализаторе на базе ПЛИС\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_**Д. В. Омаров\_**\_\_\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_**Д. А. Семеренко**\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2022 г.*М**инистерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_\_ г.

# Задание на выполнение курсовой работы

по дисциплине: «Цифровые устройства и микропроцессоры»

Студент группы РЛ6-71

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Омаров Денис Владимирович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Тема курсовой работы: \_\_Алгоритмы шифрования в цифровом логическом анализаторе\_\_\_

на базе ПЛИС \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направленность КР (учебная, исследовательская, практическая, производственная, др.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР): Кафедра

График выполнения работы: 25% к \_\_\_ нед., 50% к \_\_\_ нед., 75% к \_\_ нед., 100% к \_\_\_ нед.

***Задание***: Разработать логический анализатор цифровых сигналов на базе ПЛИС\_\_\_\_

\_1. Не менее 16 каналов, частота измерения 100 кГц на канал\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_2. Разработать протокол обмена сообщениями\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_3. Разработать ПО для компьютера \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Оформление курсовой работы:***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Расчетно-пояснительная записка на \_\_\_\_\_ листах формата А4.

Дата выдачи задания « \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

**Руководитель курсовой работы**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Д. А. Семеренко

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_**Д.В. Омаров

(Подпись, дата) (И.О.Фамилия)

Оглавление

[Введение 3](#_Toc120674950)

[Глава 1. Обзор существующих решений. 5](#_Toc120674951)

[Глава 2. Функциональная и принципиальная схема электрическая схема устройства. 7](#_Toc120674952)

[Глава 3. Алгоритм работы ПЛИС и основные узлы. 10](#_Toc120674953)

[Глава 4. Результаты исследований. 19](#_Toc120674954)

[Заключение 29](#_Toc120674955)

[Литература 31](#_Toc120674956)

[Приложение 1. Блок-схема верхнего уровня 32](#_Toc120674957)

[Приложение 2. Блок-схема управляющего модуля 34](#_Toc120674958)

[Приложение 3. Временная диаграмма приёмника UART 35](#_Toc120674959)

[Приложение 4. Временна диаграмма приёмника UART 36](#_Toc120674960)

[Приложение 5. Блок-схема архиватора 37](#_Toc120674961)

[Приложение 6. Временная диаграмма EDGE\_CODER 38](#_Toc120674962)

[Приложение 7. Временные диаграммы модуля RLE сжатия 39](#_Toc120674963)

[Приложение 8. Временна диаграмма CBUF 43](#_Toc120674964)

[Приложение 9. Вид приложения для ПК 44](#_Toc120674965)

[Приложение 10. Практическое исследование по выбору дискретизирующей частоты 45](#_Toc120674966)

[Приложение 11. Практическое исследование эффективности работы применённой конфигурации 47](#_Toc120674967)

# Введение

В большинстве случаев объём и скорость входных данных логического анализатора настолько велики, что никакая схема памяти либо не может вместить в себя такой объём, либо не может работать на такой высокой частоте. Для решения этой проблемы и проблемы предельной скорости передачи применяют методы кодирования (сжатия) информации.

Целью данной работы является выбор методов кодирования и исследование их эффективности, а также разработка цифровой схемы логического анализатора на базе ПЛИС Altera Cyclone IV EP4CE6E22C8N.

Логический анализатор – это электронный прибор, который улавливает и отображает несколько сигналов от цифровой системы или цифровой схемы.

Логический анализатор широко применяется в решении задач цифровой электроники. Одна из наиболее распространённых задач, решаемых с помощью логического анализатора, отладка различных цифровых конструкций, в том числе ИС. Логические анализаторы могут обнаруживать аппаратные дефекты, которые не обнаруживаются при моделировании. Логические анализаторы могут управляться программно и аппаратно от внешних воздействий, что позволяет использовать их удалённо в недоступных для человека местах, например в ракете или в спутниковой системе.

Основная проблема построения логического анализатора состоит в большом объёме входных данных. Из этого факта вытекает ряд конструктивных проблем: высокая скорость работы приёмного, обрабатывающего, передающего устройства, большой объём памяти, высокая надёжность (даже один пропущенный бит информации сигнала является критической ошибкой).

Проблема высокой скорости работы приёмного и обрабатываемого устройств решается путём выбора соответствующего устройства например: ПЛИС (Altera Cyclone IV скорость работы порядка 300 – 400 МГц), микроконтроллеры, процессоры, ASIC и иные интегральные схемы с подобранной частотой работы соответствующей требуемой частоте входных сигналов.

Проблема высокой скорости передающего устройства решается путём выбора соответствующего протокола передачи: USB, SPI, I2C, USART, CAN, Ethernet, PCI, ISA и иные протоколы передачи с требуемой скоростью передачи. Как правило объём входных данных сильно больше предельных скоростей многих интерфейсов, для этого применяются разные методы кодирования информации.

Проблема большого объёма памяти решается путём выбора соответствующей интегральной схемы памяти (SRAM, DRAM, SDRAM, DDR SDRAM, EEPROM) или построение схемы запоминающего устройства на архитектуре выбранного устройства (построение схемы ЗУ на ПЛИС, использование встроенной памяти ПЛИС или МК).

# Глава 1. Обзор существующих решений

На данный момент существует множество готовых логических анализаторов с разными характеристиками. Однако, анализ таких коммерческих решений усложняется из-за отсутствия в свободном доступе документации. Поэтому анализ данных устройств был осуществлён по доступным изображениям внутреннего устройства.

Salae Logic 8 имеет 8 каналов с частотой дискретизации до 24 МГц и пропускной способностью 24 Мб/сек. Схема Salae Logic 8 состоит из микропроцессора с высокоскоростным интерфейсом от компании CYPRESS, схема памяти EEPROM ATMLH432, шинный формирователь от NXP. Судя по схеме данного устройства, можем предположить следующий алгоритм его работы: сигналы с исследуемых каналов поступают на шинный формирователь, откуда сигналы поступают на микроконтроллер, обрабатываются, записываются в двухканальную память, откуда в последующем данные считываются микроконтроллером и отправляются по USB в ПК.

RDC2-0064 Logic Analyzer логический анализатор разработки. Имеет 32 канала максимальная частота выборки в буферном режиме 72 МГц и 18 МГц в потоковом, максимальное число выборок в буферном режиме 240 тысяч и 16 миллиардов в потоковом. Данное устройство не имеет корпуса и имеет открытую спецификацию, от чего мы можем легко рассмотреть его схему. На плате имеются: микроконтроллер STM32F722VCT6, стабилизатор тока и напряжения LM1117GS-3.3, защитные диоды USBLC6-2SC6. Схема имеет защиту по току и напряжению, а вся работа устройства происходит на микроконтроллере, учитывая два режима работы устройства, можем предположить, что микроконтроллер считывает значение с сигнальных входов, преобразует, записывает во внутреннюю память, а после при необходимости передает данные по одному из интерфейсов.

Hantek 4032L логический анализатор фирмы Hantek. Имеет 32 анализируемых каналов, частота дискретизации 400 МГц имеет 2 Гб памяти, поддерживает анализ основных передающих интерфейсов: SPI, UART, I2C; имеет настраиваемые уровни входов. Для работы подключается через USB 2.0 к ПК, имеет собственное приложение. Внутреннюю структуру Hantek 4032L и алгоритм его работы найти не удалось.

Как видно, почти все логические анализаторы имеют одну структуру и алгоритм работы и по большей части отличается только качеством элементов, более продуманной схемами защиты и стабилизации питания, а также оптимизацией алгоритмов работы устройства.

Основной принцип работы большинства логических анализаторов представлен на Рисунке 1

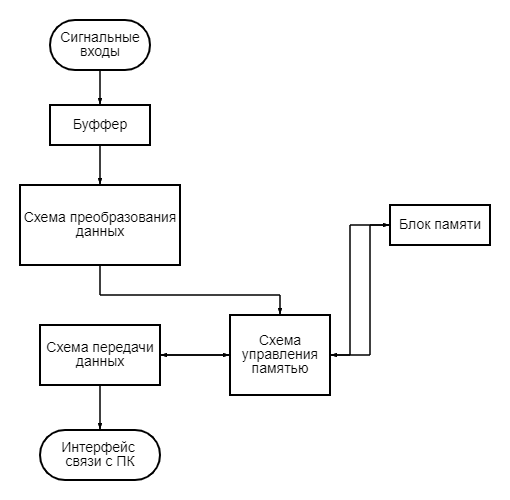


Рисунок 1 – Функциональная схема принципа работы логических анализаторов представленных на рынке

Сигнал со входов считывается в буфер, для повышения устойчивости работы устройства, затем считывается и обрабатывается для дальнейшего хранения (или передачи) схемой преобразования данных, после эти данные записываются в запоминающее устройство, для последующих считывания и передачи схемой передачи данных по соответствующему интерфейсу. Как видно из представленных реализаций, работу нескольких блоков могут брать на себя отдельные схемы, например в RDC2-0064 Logic Analyzer работа всего устройства осуществляется на одном микроконтроллере.

# Глава 2. Функциональная схема электрическая устройства

Разработка устройства была проведена на базе ПЛИС Altera Cyclone IV EP4CE6E22C8N, ввиду её высокой скорости работы и наличия встроенной памяти. В качестве передающего интерфейса использовался UART для упрощения процесса разработки. Для связи ПК и ПЛИС использовался модуль USB–UART адаптера на базе микросхемы FT232. Декодирование и наглядное представление сигналов виде графиков осуществлялось с помощью написанной программы для ПК.

На сигнальных входах ПЛИС установлен промежуточный буфер, чтобы обеспечить дальнейшую работу с определёнными сигналами, т.к. сигнал на входе может принимать промежуточные значения или начать изменяться в момент преобразования.

В качестве метода кодирования данных был разработан алгоритм шифрования на базе стандартного алгоритма RLE. Е.Р. Пантелеев описывает, что в стандартной алгоритме кодирования длин серий (RLE–алгоритм) предлагается вместо исходной последовательности слов отправлять слово и число его повторений в последовательности [5, с. 15][5]. Такой алгоритм наиболее эффективен в случае, когда сигнал меняется медленно и представляет собой длинные последовательности. В случае, когда сигнал меняется быстро и представляет собой короткие последовательности (меньше принятой длины кода), такой алгоритм резко теряет в эффективности сжатия. Поэтому в случае быстроменяющегося сигнала будет отправляться непреобразованная последовательность. Чтобы различать преобразованную последовательность от непреобразованной в старший бит кода будет записан ноль или единица.

Чтобы не отправлять кодируемое слово, как предлагает Е.Р. Пантелеев, в качестве входных данных будем использовать не значение сигнала, а длину уровня сигнала (расстояние между фронтами сигнала) [5, с. 15][5]. Такую операцию выполняет детектор фронта сигнала.

Модифицированное RLE-кодирование выполняет RLE кодировщик. На входе устройства имеется 16 независимых каналов, такие данные затем необходимо объединить в один поток, чтобы отличить такие последовательности в 4 старших бита кода, за исключением бита типа преобразования, записывается номер канала. Таким образом, при использовании 8 битного кода, рабочими остаётся 3 бита. Такой код только увеличит входные данные, поэтому будем использовать 16 битный код.

Такой код в итоге придётся передавать по UART, который имеет 8 битный код. Для выравнивания данных применяется кольцевой буфер, который принимает 16 битный код, а выдаёт 8 битный. Такой модуль также согласует разные скорости записи и считывания.

Поскольку UART имеет скорость передачи в худшем случае меньше, чем скорость поступления данных, запишем данные в память типа FIFO, которые будут считываться приёма-передатчиком UART по требованию.

Данные с FIFO считываются модулем приёма-передачи UART и отправляются на выходные ножки ПЛИС, на USB–UART адаптер. Который отправляет данные в ПК по USB. Где данные декодируются и представляются в удобном пользователю виде.

Для управления схемой применяется управляющая схема (управляющий модуль), которая инициирует начало и конец работы схемы, а также отображает внутренние флаги состояния схемы на диодах. Управление работой устройства осуществляется аппаратно – с помощью кнопок, а также програмно – с ПК по UART отправляется соответствующая команда.

Такая схема отображается в функциональной схеме устройства Рисунок 2.

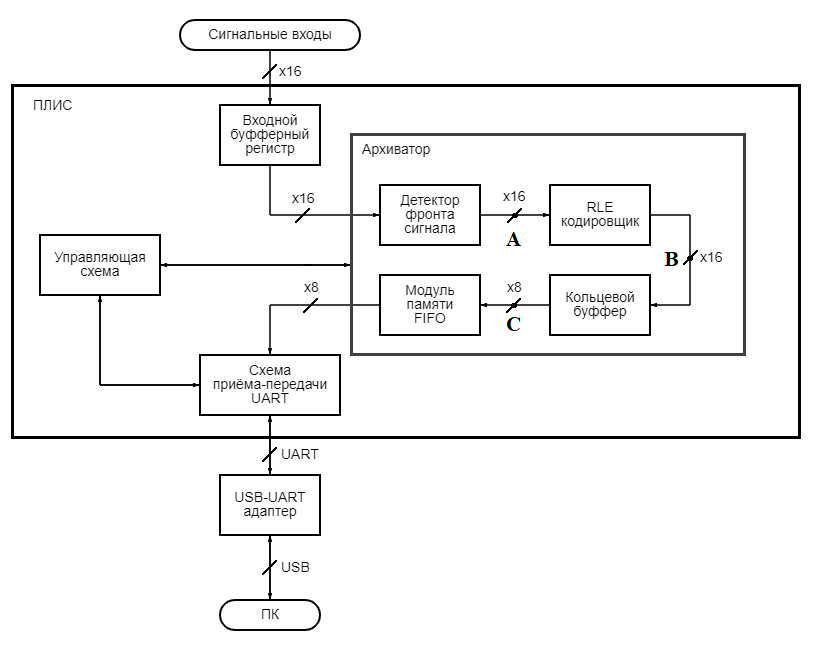


Рисунок – Функциональная схема устройства

# Глава 3. Алгоритм работы ПЛИС и основные узлы

Основной алгоритм работы ПЛИС задаётся её функциональной схемой Рисунок 2. Рассмотрим работу и реализацию каждого блока ПЛИС.

Блок-схема файла верхнего уровня представлена в Приложении А. Схема соединений основных модулей стр.1 и схема формирования основных рабочих частот, а также схема формирования исследующих сигналов стр.2 (Подробнее смотреть Главу 4).

Схема формирования основных рабочих частот основывается на работе внутреннего модуля PLL. Модуль PLL и делитель формируют три основных частоты:

Таблица – Основные формируемые частоты

|  |  |
| --- | --- |
| Тактирующая частота (rdclk) |  |
| Рабочая частота модуля uart (uart\_clk) |  |
| Частоту дискретизации (clk) |  |

Также в блок-схеме верхнего уровня соединены основные модули: ARCHIVATOR, USART, CONTROL\_CIRCUIT. Шины данных и флага готовности записи модулей архиватора и приёма-передатчика соединены через мультиплексоры, для будущего управления управляющей схемой на основе конечного автомата, который в рамках данной работы разработан не был, поэтому мультиплексоры работают в статическом режиме, их работа эквивалентна простому проводу.

Общий алгоритм работы данных модулей прост: когда приёма-передатчик готов получить данные, он отправляет сигнал архиватору, тот по готовности отправляет байт данных и сигнал готовности данных, после чего цикл повторяется. Как только внутренний буфер архиватора переполняется поднимается флаг переполнения – новые данные в буфер не записываются, и после передачи уже собранных данных поднимается флаг конца передачи.

Флагами, а также внешними управляющими сигналами управляет модуль CONTROL\_CIRCUIT. Схема модуля CONTROL\_CIRCUIT представлена в Приложении Б. Флаги переполнения буфера, статуса работы, отсутствие данных в буфере, готовности передатчика выводятся на светодиоды в соответствии со схемой, при этом флаг переполнения буфера защёлкивается на триггере, который сбрасывается по сигналу сброса.

Если на приёмник пришла управляющая команда, то она обрабатывается в модуле valider и при декодировании инвертируется сигнал сброса. Модуль просто сравнивает по сигналу значение принятого байта с байтом команды (0xAB) и инвертирует значение выхода.

Также присутствует запуск с кнопки, однако он разрешается только после подключения к ПК, т.к. в других случаях устройство работать не должно.

Модуль приёма-передачи UART состоит из двух отдельных модулей приёма и передачи. Модуль тактируется частотой работы uart\_clk, которая в четыре раза больше частоты передачи UART 512 кГц. Передатчик не нуждается в такой высокой частоте, однако для работы приёмника была предусмотрена именно такая частота, для запаса по рассинхронизации генераторов (как частотной, так и фазовой).

Модуль приёма начинает захват на середине уровня сигнала (на втором импульсе частоты передачи, умноженной на четыре), для увеличения надёжности работы. После того как на первом фронте рабочей частоты был получен стартовый бит, дождёмся его появления на втором фронте, иначе идентифицируем его как слишком короткий и переходим в состояние ожидания. Если стартовый бит в данном случае был принят повторно, начинаем заносить в регистр данных каждое четвёртое значение принятого бита, включая этот. Если бы частота работы была в два раза меньше, то был бы высокий риск сбиться и пропустить какой-нибудь бит или всю последовательность, если бы, например частота генератора внешнего UART передатчика была бы немного выше нашего приёмника. Данную проблему поясняют иллюстрации на Рисунках 3–4.

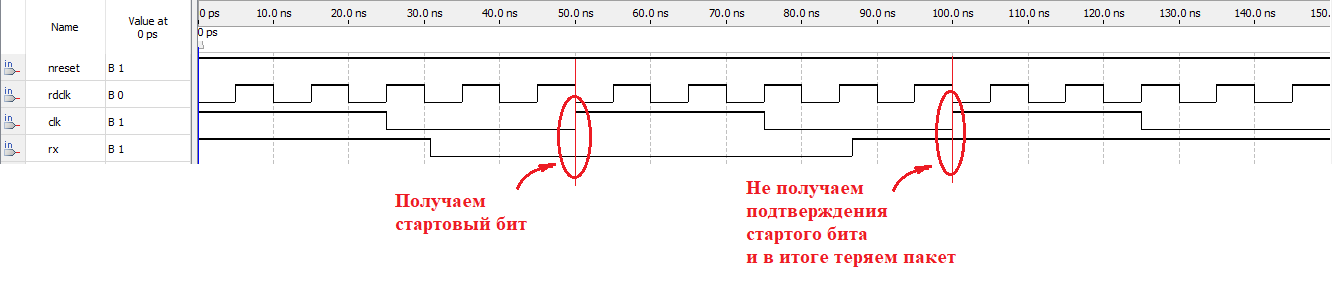


Рисунок – Временная диаграмма приёмника, тактируемого двойной частотой, при рассинхронизации внутренних генераторов



Рисунок – Временная диаграмма приёмника, тактируемого четверной частотой, при рассинхронизации внутренних генераторов

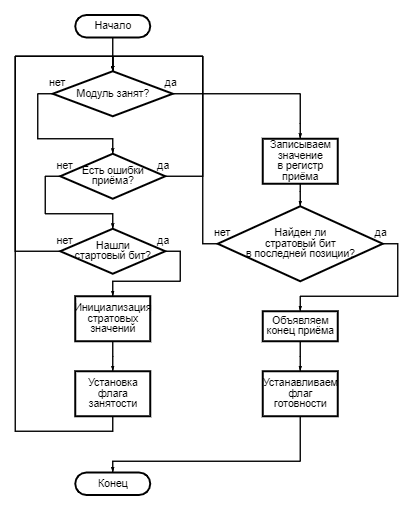


Рисунок 5 – Общий алгоритм работы модуля приёмника UART

В случае, когда частота работы в четыре раза больше частоты передачи, мы вводим дополнительное время погрешности, которое может быть потрачено на рассинхронизацию по фазе и частоте между внутренними генераторами приёмника и передатчика. Как видно из этих диаграмм минимальная задержка обеспечивается при частоте в три раза большей, однако т.к. проектирование осуществляется на двоичной логике удобней и понятней вести подсчёт по степеням двойки. И при потребности в устранении большей погрешности рассинхронизации можно увеличить длину задержки , увеличив величину счёта и рабочую частоту. По той же причине мы не фиксируем стартовый бит на первом фронте, т.к. в случае если генератор передатчика медленней – будет наблюдаться обратная ситуация, которая приведёт к потере битов и пакетов. Временная диаграмма модуля приведена в Приложении В.

Модуль передачи данных выводит на линию TX стартовый бит, значение из регистра данных, сдвинутого на число отправленных битов, стоповый бит, выдерживает каждое значение 4 такта частоты uart\_clk, после чего переходит в режим ожидания новых данных. Временна диаграмма модуля передатчика приведена в Приложении Г.

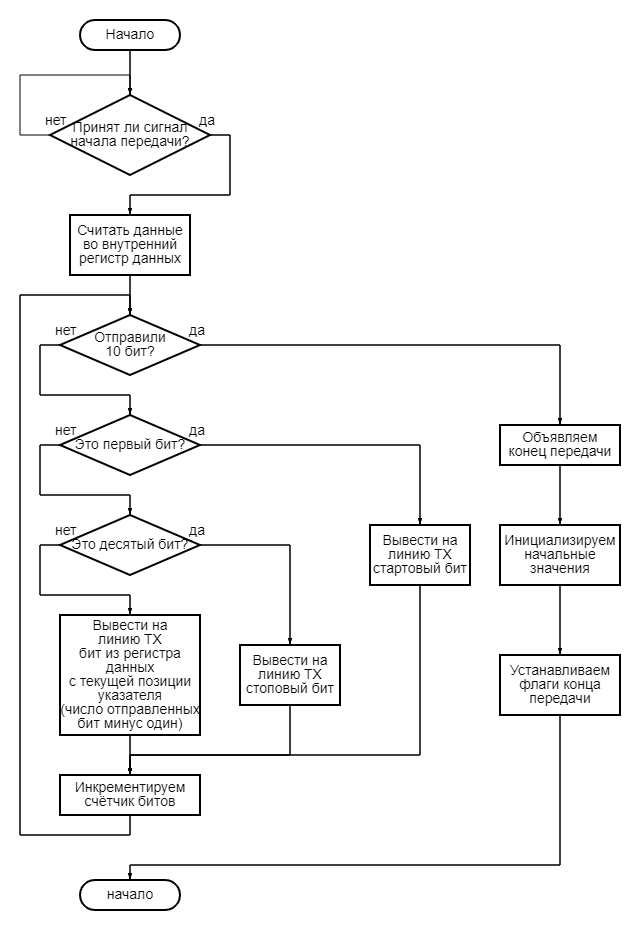


Рисунок 6 – Общий алгоритм работы модуля передачи UART

Общий принцип работы модуля архиватора изображён на Рисунке 2 и изложен в Главе 2.

Сжатие данных начинается с сортировки входных данных в более удобный формат, для этого вместо непреобразованных данных, будем отсылать лишь сигнал о их изменении. Более упорядоченные и однообразные данные поддаются лучшему кодированию. Эту задачу, а также задачу входного буфера реализует модуль EDGE\_CODER. Входной сигнал поступает на модуль EDGE\_CODER, который записывает значения сигналов в первый регистр по такту частоты дискретизации, по следующему такту он пробрасывает значение из первого регистра во второй и снова записывает первый регистр входными сигналами. Выход модуля представляет собой попарную операцию XOR двух регистров. Временная диаграмма представлена в Приложении Е (на функциональной схеме точка A).

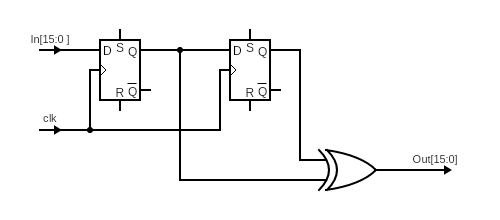


Рисунок 7 – Синтезированная RTL схема модуля EDGE\_CODER

Подготовленные данные нужно сжать. Алгоритм кодирования реализует модуль RLE\_CODER, представляющий собой набор параметризованных модулей RLE\_CODER\_CODER, осуществляющих преобразование над каждым каналом независимо (по модулю на канал).

Независимые модули кодирования выдают коды по готовности каждого отдельного канала, т.е. в совершенно случайные отрезки времени и может случаться так, что в один отрезок времени готовы данные от каждого канала. Для решения этой проблемы тактирование каждого модуля RLE\_CODER\_CODER сдвинуто по времени, но для обеспечения правильной работы этот сдвиг минимален. После того, как модуль одного канала завершил преобразование, он выставляет на шине данных преобразованный код и поднимает флаг готовности передачи данных. Напомним, что начало преобразования для каждого следующего канала сдвинуто по времени относительно предыдущего и т.к. время преобразования у каждого канала фиксировано, то следующий модуль не начнёт передачу, пока не завершилась передача предыдущего.

Каждый модуль RLE\_CODER\_CODER имеет 16-ти битовую шину выходных данных и один флаг готовности данных. Однако, физически мы имеем только один передатчик, поэтому 16 независимых потоков данных нам нужно объединить в один. Для этого применяется мультиплексор. В качестве адреса мультиплексор принимает сигнал с приоритетного шифратора, входами которого будут флаги готовности независимых преобразователей RLE\_CODER\_CODER. Таким образом, на шине выходных данных будет находиться значение преобразованных данных из последнего закончившего преобразование модуля

Основной принцип работы модуля кодирования отдельного канала RLE\_CODER\_CODER состоит в подсчёте подряд идущих нулей, и в случае нахождении единицы – отправки последовательности из 11-ти битов несжатой последовательности. С временными диаграммами RLE\_CODER\_CODER и RLE\_CODER можно ознакомиться в Приложении Ж (Временная диаграмма модуля RLE\_CODER показана относительно точки B функциональной схемы).

Синхронизацией скоростей преобразования данных и записи их в память, а также разбиением 16-ти битной последовательности в 8-и битную, занимается модуль кольцевого буфера CBUF. Кольцевой буфер реализует структуру памяти FIFO или очередь. Данный модуль основан на структуре «Память с указателями начала и конца». Модуль содержит два регистра с позициями начала и конца данных. По внешнему сигналу записи модуль записывает половину двухбайтового слова в ячейку памяти по указателю и инкрементирует значение регистра, для второй половины слова операция повторяется. Чтение осуществляется таким же образом, но только с одной операцией над одним байтом. Для контроля за правильной работой указателей введены внутренние флаги и их проверка на каждом цикле чтения/записи. В случае переполнения модуль уходит в заблокированное состояние с выводом соответствующего флага. С временными диаграммами можно ознакомиться в Приложении З (Временная диаграмма показана относительно точки С функциональной схемы).

В качестве модуля памяти был синтезирован модуль FIFO памяти на внутренних M9K блоках с помощью встроенных средств Quartus «MegaWizard Plug-in Manager». Для блокировки поступления новых данных была введена схема единожды срабатываемой защёлки, которая блокирует поступление новых данных в случае переполнения FIFO. Схема блокирования изображена на Рисунке 13.

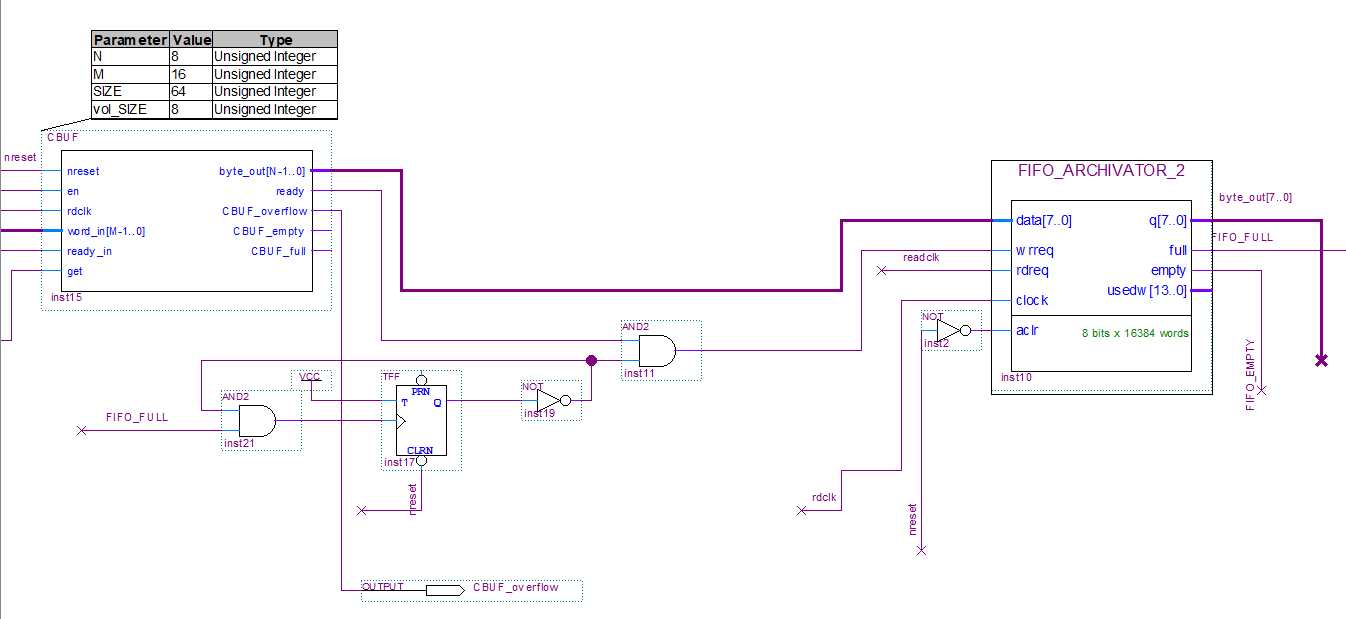


Рисунок – Схема блокирования FIFO

В качестве сигнала ready модуля архиватора используется входной сигнал считывания данных, разрешаемый при наличии в FIFO данных.

Спроектированное устройство осуществляет обработку, кодирование и передачу данных. Однако, оно не осуществляет их удобную пользователю обработку и выдачу. Другими словами, логический анализатор выполняет только половину необходимых процессов. Для организации полного цикла работы необходим ПК с соответствующим программным обеспечением. Поскольку процесс получения и предоставления данных специфичен для организации работы с устройством потребовалась бы работа с несколькими программами со сложной инструкцией. Более того, данные передаются в закодированном виде с нестандартным алгоритмом. Для облегчения процесса использования, отладки и настройки было принято разработать простое приложение в QT для работы с устройством.

Устройство осуществляет передачу данных по последовательному порту (COM-порту). Для работы с COM-портами применяется WinAPI (встроенная библиотека «windows.h»). Изначально приложение было написано полностью на WinAPI, включая пользовательский интерфейс, однако в последующем было решено продолжить разработку приложения с помощью встроенных средств QT для более удобной работы с графиками. Встроенные средства QT также могут обеспечить взаимодействие с COM-портами, однако не имеют возможность тонкой настройки baudrate частоты порта, предоставляя лишь неизменяемый список уже сформированных значений.

Для чтения был создан новый QThread поток (с созданием соответствующего наследованного класса), в котором в цикле по байту как из файла из COM-порта считываются байты и записываются в промежуточный бинарный файл.

Настройка COM-порта производилась в функции SerialBegin() согласно стандартной процедуре с заполнением основных структур DCB и COMMTIMEOUTS соответствующими значениями. Для работы используется COM3 порт, созданный драйвером соответствующего FT232 модуля.

Основные функции работы с портами и файлами WinAPI вынесены в файлы COMp\_WinAPI.c/.h.

Для обработки данных было создано две функции. Первая функция pre\_RLE\_adressed\_decode() обеспечивает первичную обработку данных. Функция открывает файл со считанными из COM-порта данными, считывает по два байта из последовательности и записывает их в конец бинарного файла с номером, соответствующим номеру канала считанной последовательности (старшие 4 бита слова за исключением первого бита).

После того как основные файлы были собраны, можно начать обработку данных отдельно по каналам. Для этого была написана функция RLE\_decode(), однако из-за изменения архитектуры ПЛИС (Подробнее смотреть Главу 4) отпала необходимость в отдельных файлах последовательности, т.к. отсутствует необходимость в декодировании LZ. Поэтому дальнейшая обработка проходила при формировании графиков, для увеличения скорости работы приложения.

Конечную обработку и представление данных пользователю выполняет функция create\_graph(). Функция открывает соответствующий файл канала, считывает два байта, проверяет старший бит, декодирует слово соответствующим алгоритмом. В случае единицы в старшем бите декодированной последовательностью будут младшие 11 бит. В случае нуля последовательность будет представлять собой последовательность повторений последнего декодированного отсчёта длинной в число равное значению младших 11 бит кода.

Все считанные отсчёты заносятся в QVector, плюс малое отклонение для построения вертикальных линий. При записи каждого отсчёта в дополнительный QVector заносятся координаты отсчётов (+1 к прошлому отсчёту, начиная с нулевого). После чего вектора передаются в методы созданных виджетов-графиков. Работа с графиками осуществлялась с помощью библиотеки QCustomPlot.

Алгоритм работы с программой:

1. После подключения устройства и ПК следует нажать кнопку «Connect». После удачного подключения в первую строчку выведется сообщение об удачном подключении с номером порта.
2. После подключения к устройству и подключения его к исследуемым каналам следует нажать кнопку «Send START» приложение отправит команду начала преобразования (0xAB), после чего будет запущена передача данных.
3. При загорании флага конца передачи на ПЛИС или при желании пользователя завершить преобразование досрочно следует повторно нажать «Send START» (команды начала и конца одинаковы)
4. Чтобы закрыть поток чтения, который может продолжаться нажать кнопку «Disconect», если не все данные успели записаться или возникли сбои в работе. Данный шаг можно пропустить, убедиться в окончании передачи пользователь может, посмотрев характеристики файла первичного чтения данных (временного бинарного файла), если его размер увеличивается – передача не закончена.
5. После окончания передачи нажимаем кнопку «Conv file» по этой команде будет произведено декодирование и сформированы основные файлы для построения графиков. После выполнения этой команды во второе поле TextLabel будет выведено число принятых данных (длина сжатой последовательности). Данную кнопку разрешено нажимать повторно при необходимости.
6. После того как файлы сформировались (приложение не даст делать что-либо пока файлы преобразуются) нажимаем кнопку «Create graph», после чего будут сформированы 16 интерактивных временных диаграмм сигналов. Каждый график в углу помечен номером канала и количеством собранных отсчётов. После завершения формирования графиков в третье поле TextLabel будет выведено общее число декодированных отсчётов, в четвёртое поле будет выведена степень сжатия в процентах. Чтобы перевести программу в изначальное состояние – нажать Clear. Вид окна приложения представлен в Приложении И.

# Глава 4. Результаты исследований

## Результаты исследования выбора частоты дискретизации

В данном исследовании рассматриваются предельные режимы работы устройства при различных значениях частоты дискретизации. Результатом работы будет соответствующий выбор частоты дискретизации устройства. Данные исследований приведены в Приложении К.

Под высокой частотой сигнала подразумевается быстроменяющийся сигнал, такой что работа алгоритма не оптимальна. Под низкочастотным сигналом подразумевается медленно меняющийся сигнал (вовсе не меняющийся), такой сигнал алгоритм сжимает максимально сильно.

Случаи, в которых передача завершилась из-за переполнения, означают что внутренняя память была заполнена, работа алгоритма была остановлена для передачи уже сформированных отсчётов. В случаях, когда внутренняя память не была переполнена, означают что передача была остановлена по желанию пользователя и могла продолжаться бесконечно (т.к. скорость передачи в данных случаях превышала скорость генерации данных).

Как видим из экспериментов на высоких и низких частотах (Таблицы 2–5) практически полученные значения степени сжатия алгоритма согласуются с теоретически рассчитанными для крайних случаев. Степень сжатия медленноменяющегося сигнала настолько велика, что передача такого сигнала почти не занимает ресурсы передатчика. Из таблиц очевидно, что передача такого сигнала может осуществляться в непрерывном режиме почти для любой частоты дискретизации. Более того, при более детальном рассмотрении данного процесса, было обнаружено, что внутренняя память не успевает заполниться больше, чем на пару десятков байт, для такого сигнала.

Также, из (Таблиц 2–4) можно заметить, что начиная с частоты дискретизации в , устройство может работать в непрерывном режиме даже на высокочастотном сигнале. При работе на более высоких частотах дискретизации память заполняется быстрее, чем освобождается, что приводит к переполнению. В случае переполнения удаётся стабильно собрать 45056 отсчётов. Если работа в непрерывном режиме не имеет особого значения, следует выбирать частоту дискретизации в пределах 100–200 кГц. На данных частотах осуществлялась наиболее стабильная передача, в отличие от высоких значений частоты от 0,6 до 1,5 МГц. Однако, эти ошибки возникают из-за нарушения отношения величин частоты дискретизации и тактирующей частоты, и если повысить значение тактирующей частоты, то устройство может работать и на высоких частотах. Однако, значение тактирующей уже достаточно велико, порядка 100 МГц, и бездумное увеличение её значения приводит к артефактам передачи (в переданной последовательности отсутствуют некоторые значения, либо обнаруживаются переходные значения). Внутренний модуль памяти FIFO ПЛИС имеет ограничения частоты работы, которые чётко не указаны в документации. Однако, экспериментально было выявлено, что не стоит увеличивать тактирование данного модуля выше 100 МГц.

Более того, при работе в непрерывном режиме, устройство теоретически может работать бесконечно, однако при длительной работе (в рамках запуска одного исследования) могут возникать артефакты передачи, что при высоком требовании к позиции переданных байтов может привести к грубым искажениям полученных данных.

Исходя, из всего вышеописанного можно сделать вывод, что оптимальная работа устройства достигается при значениях частоты дискретизации в 100–200 кГц. Выберем частоту 100 кГц, с более высоким отношением тактирующей к частоте дискретизации.

Также, было проведено исследование работы при неодновременной загруженности каналов (Таблиц 6–7). В рамках этого исследования на 4 из 16 сигнальных каналов был подан высокочастотный сигнал, в то время как остальные каналы были заземлены. Для случаев с переполнением было стабильно собрано 167216 отсчётов, для остальных случаев работа была остановлена пользователем и могла продолжаться далее, подобно с ранее рассмотренными исследованиями. Степень сжатия во всех случаях составила . Это достаточно высокая степень сжатия, к примеру работа простых статических алгоритмов над растровыми изображениями составляет примерно такие же значения.

Разработанный алгоритм сжатия повысил максимальную частоту дискретизации для работы в непрерывном режиме в 4 раза, при занятости четверти каналов, что согласуется с теорией, ведь алгоритм работает над каждым каналом независимо, а значит общее количество данных в секунду будет рассчитываться как сумма данных в секунду от каждого канала. То есть, если каналы заземлены и не используются, количество их выходных данных пренебрежимо мало по сравнению с активными каналами. Данное исследование моделирует работу устройства в среднем случае.

## Результаты исследования эффективности работы применённой конфигурации

Результатом данного эксперимента является получение зависимости степени сжатия от частоты сигнала, а также получение диапазона сжимаемых частот. Данные исследования приведены в Приложении Л.

Для исследования работы устройства в рабочих условиях, а также для получения значений степени сжатия в промежуточных случаях, смоделируем работу устройства при измерении сигнала передающего интерфейса. Для этого создадим невысокую частоту с определённой скважностью, модулированную высокочастотным сигналом. Такое исследование моделирует работу UART при передаче с задержками между пакетами. Такое испытание наиболее подходяще для выбранного алгоритма, т.к. если принять задержку кратную длине несжатой последовательности, можно сразу перейти к области работы данного алгоритма. Ведь, как не трудно убедиться, при иных значениях исследуемого сигнала мы либо будем попадать в область отрицательного сжатия, либо быстро попадём в область максимального сжатия, выбрав не верный шаг.

Как видно из (Таблиц 8–9) и соответствующих им графиков начиная с определённой граничной эквивалентной частоты (10 кГц) степень сжатия алгоритма начинает увеличиваться и при частоте 0,75 кГц начинает работать с максимальной эффективностью, даже переходя в потоковый режим. На графиках наблюдаются несколько скачкообразные изменения коэффициента сжатия, скорее всего частота модулированного сигнала была подобрано несколько не точно и фронта соседних частот оказались на недостаточном удалении друг от друга, это также может зависеть от момента запуска исследования. Однако, в целом мы получили достаточно достоверные данные с шагом коэффициента сжатия в 10–15%.

Как видно из графика 2, полученные значения формируют экспоненциальную зависимость. Попробуем аппроксимировать полученные значения экспонентой. Для этого воспользуемся встроенным инструментом MATLAB «curveFitter». Введём адреса таблиц полученных данных. Т.к. ожидаемая зависимость может иметь отрицательные значения, одной экспоненты мало, введём ещё одну и будем аппроксимировать двух-экспоненциальной аппроксимацией. Получим, что зависимость с аппроксимацией имеет вид:

Параметр ошибки модели: степень детерминации

При этом стоит учитывать, что степень сжатия не может превысить или быть равна , а также необходимо помнить, что для любой частоты выше степень сжатия будет равна . График аппроксимации приведён на Рисунке 9.

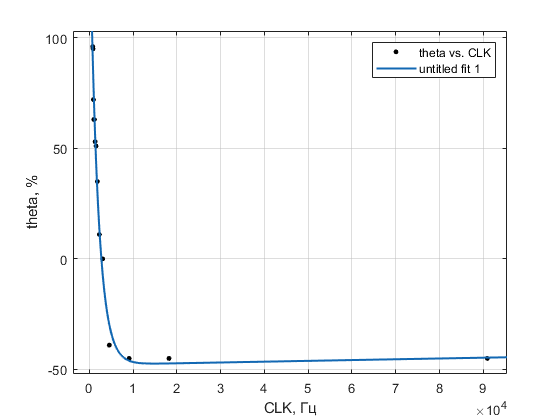


Рисунок – График двух-экспоненциальной аппроксимации зависимости коэффициента сжатия от частоты сигнала

Переломная точка роста данного графика составляет и начиная с этой точки начинается активный рост. По сути, мы можем управлять данной зависимостью изменяя длину кода. При увеличении длины кода, ожидается смещение к оси ординат всех отсчётов (непропорционально), т.е. уменьшение частоты, при которой начинается рост (если рассматривать зависимость в отражённом виде справа на лево) функции коэффициента сжатия. Это значит, что увеличение длины кода приведёт не к увеличению общего коэффициента сжатия (можно рассматривать как нормированную площадь под кривой в диапазоне исследуемых частот), а наоборот – к его уменьшению. Однако, увеличение длины кода также приведёт к увеличению скорости роста функции, что в предельном случае приведёт функцию к виду единичного импульса. Что, казалось бы, тоже плохо, однако если ограничить работу алгоритма только на этом интервале частот, то получим алгоритм с очень высоким значением степени сжатия. Остальную часть функции можно аппроксимировать другими методами или вариацией этого метода с другой длиной кода и более плавной характеристикой. На деле получить такие зависимости не получиться, потому что функция на самом деле дискретная, однако рассчитать приближённые значения вполне реально.

Эксперимент показал, что разработанный алгоритм осуществляет сжатия частот до и в некотором смысле можно расширить диапазон до . Результаты данного исследования могут быть полезны в дальнейших разработках и оптимизации алгоритмов сжатия (применительно работы логического анализатора и не только).

# Заключение

В данной работе был рассмотрен процесс проектирования логического анализатора и его основных блоков. В результате работы были разработаны алгоритмы работы устройства и его блоков, была разработана архитектура устройства на ПЛИС с использованием языка описания аппаратуры Verilog и встроенного схемного редактора Quartus, был разработан метод сжатия данных на базе стандартного алгоритма RLE, было разработано пользовательское приложение на ПК, а также проведены испытания предельных режимов работы устройства, рабочих режимов работы и получены зависимости эффективности работы разработанных алгоритмов сжатия, проведены исследования на тему текущих и предполагаемых параметров кода.

По итогу работу было получено работающее устройство для анализа логических сигналов. Схема устройства предполагает использование одной микросхемы ПЛИС, однако это очень неэффективно. Весь потенциал схема раскрывает после введения модулей RAM-памяти, в идеале в связке с контроллером. В такой схеме ПЛИС может заполнять одну или несколько схем памяти сжатыми с помощью простых потоковых алгоритмов данными, которые затем контроллер будет сжимать более сильным, возможно статическим, алгоритмом сжатия и передавать по высокоскоростному интерфейсу: USB в случае связи с ПК или флешкой, DDR в случае связи с памятью, процессором, другими платами ПЛИС, и иными. Модуль интерфейса в таком случае может быть встроен в контроллер, либо размещено IP-ядро модуля интерфейса на ПЛИС.

В конечной реализации устройство может иметь схемы защиты по питанию и настраиваемые по заданному уровню входные IO буферы с программной подтяжкой.

Также, возможно использование более продвинутых алгоритмов сжатия, либо разработка новых. В этом могут быть полезны данные исследований применённого алгоритма и полученные зависимости. Вполне возможно применение метода deflate или его вариаций, когда несколько алгоритмов работают «одновременно» и выбирается тот, чьё мгновенно сжатие выше.

В архитектуре проекта на ПЛИС могут быть внесены оптимизирующие исправления. Многие модули работают не оптимально, временные задержки некоторых из них не были рассчитаны строго. Многие модули требуют двойного хэндшейка, ждут хэндшейка, находятся в неизвестном состоянии во время простоя. Строгая синхронизация и договорённость синхронной работы всех модулей с учётом конечного вида архитектуры решат эти проблемы, а значит повысят скорость работы и уменьшат занимаемые ресурсы ПЛИС.

Решение вышеописанных проблем повлияет не только на количество снимаемых отсчётов, но и на частоту работы устройства. То есть предлагаемые модификации способны повысить частоту дискретизации входных сигналов, вплоть до нескольких десятков или сотен мегагерц.

Также улучшить проект может разработка управляющего конечного автомата в CONTROL\_CIRCUIT с введением множества режимов работы, увеличения списка команд. Сделать приложение для ПК более приветливым пользователю: введение описаний, выпадающих списков портов, пролистываемый график (как в Logic 2.4.1). Всё это увеличит управляемость и востребованность пользователями.

Для удобства, схема может быть дополнена програмно управляемыми синтезатором частот, генератором сигналов, ШИМ-генератором. Эти модули не займут больших ресурсов ПЛИС, а значит при той же стоимости конечного устройства возможно расширение его функциональности.

В данной работе также была рассмотрена теория основных алгоритмов сжатия без потерь и приведены обоснования их эффективности касательно данного проекта.

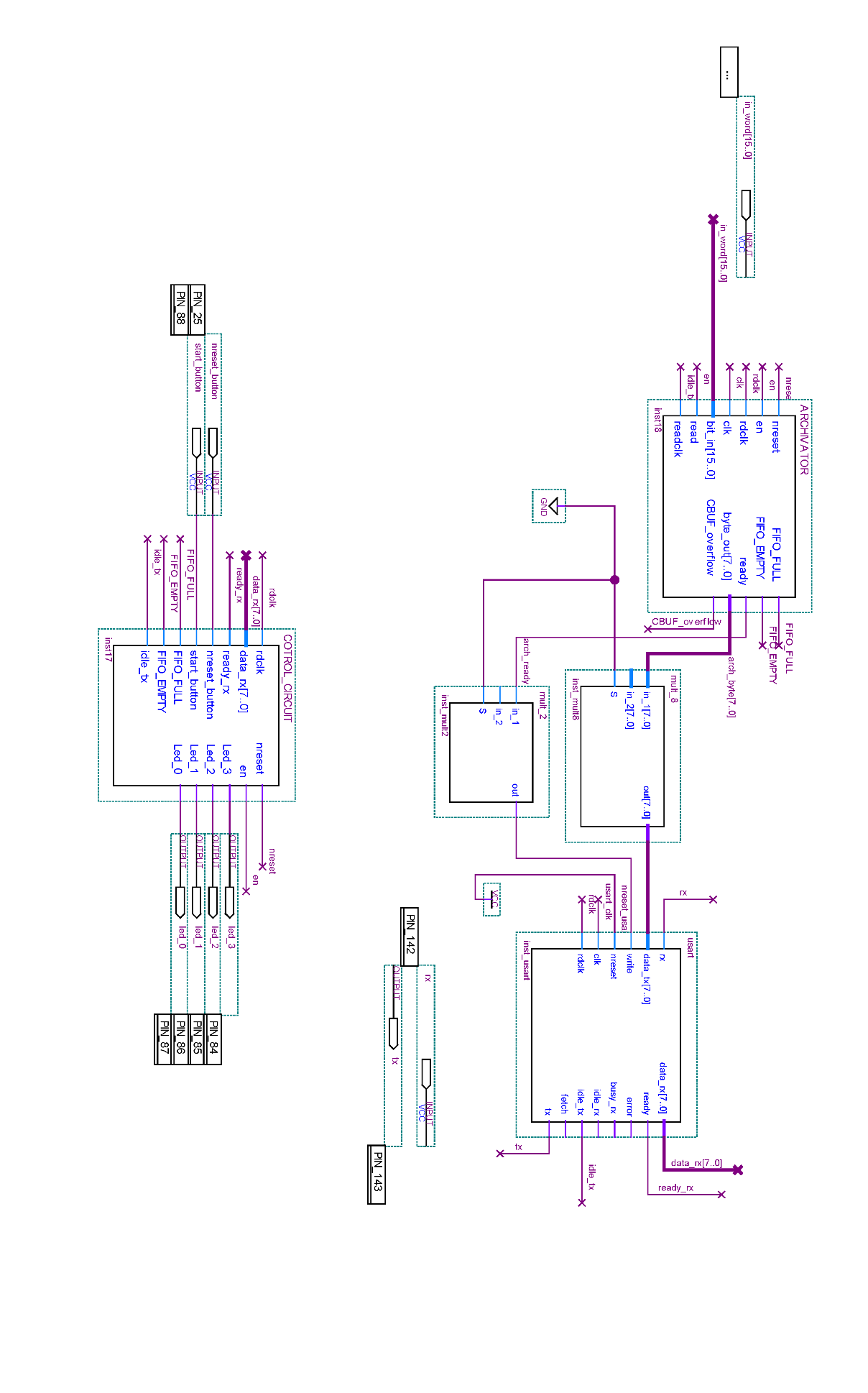
Результаты данной работы могут быть использованы в научно-популярных или учебных целях.

# Литература

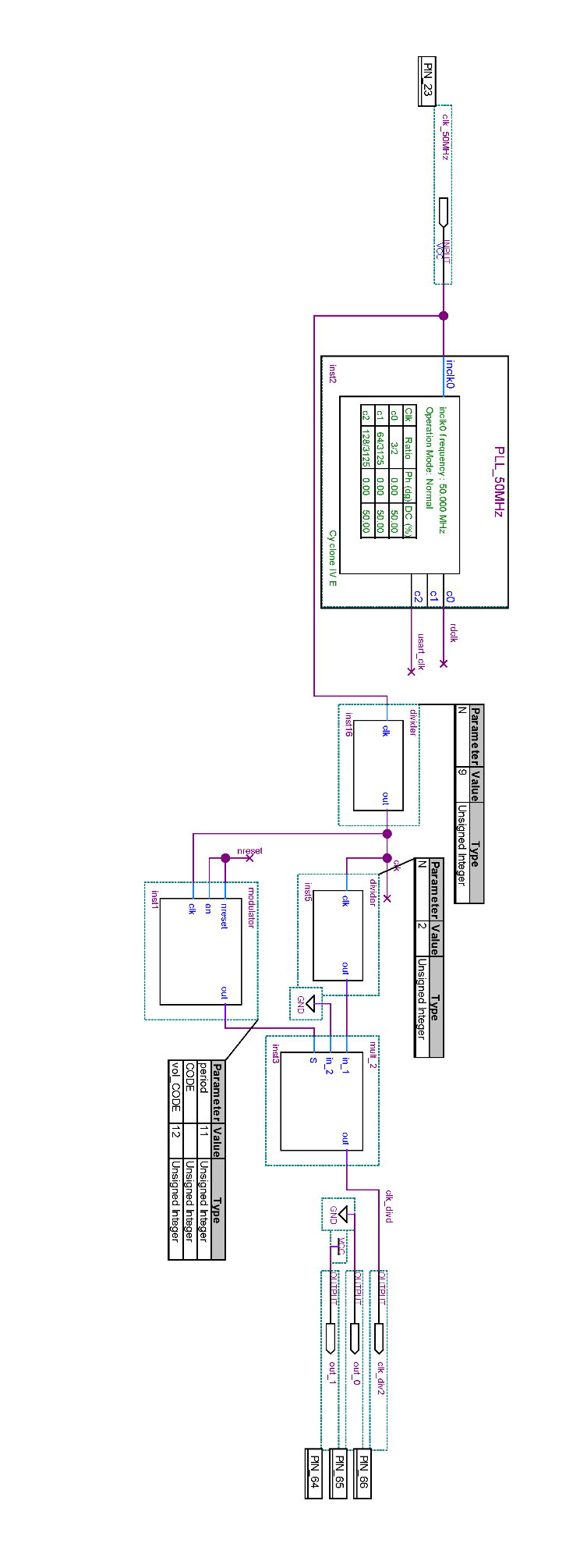
1. Cyclone IV Device Handbook: CYIV-51001-1.7 //ALTERA
2. Single- and Dual-Clock FIFO Megafunction: User Guide//ALTERA
3. Стешенко В.Б., Попова Т.В., Малашевич Д.Б. Основы HDL Verilog как средтсва проектирования цифровых устройств: Уч.пос./ Под ред. А.И. Сухопарова. – М.: МИЭТ, 2006. – 136.: ил.
4. Краткий курс HDL Иосиф Каршенбойм [Электронный ресурс] – режим доступа: URL: <http://iosifk.narod.ru/hdl_coding/verilog.htm> (Дата обращения 30.11.2022)
5. Пантелеев К.Р. Алгоритмы сжатия данных без потерь: учебное пособие для вузов / Е. Р. Пантелеев, А. Л. Алыкова. – Санкт-Петербург : Лань, 2001. –172 с. : ил.

# Приложение А. Блок-схема верхнего уровня

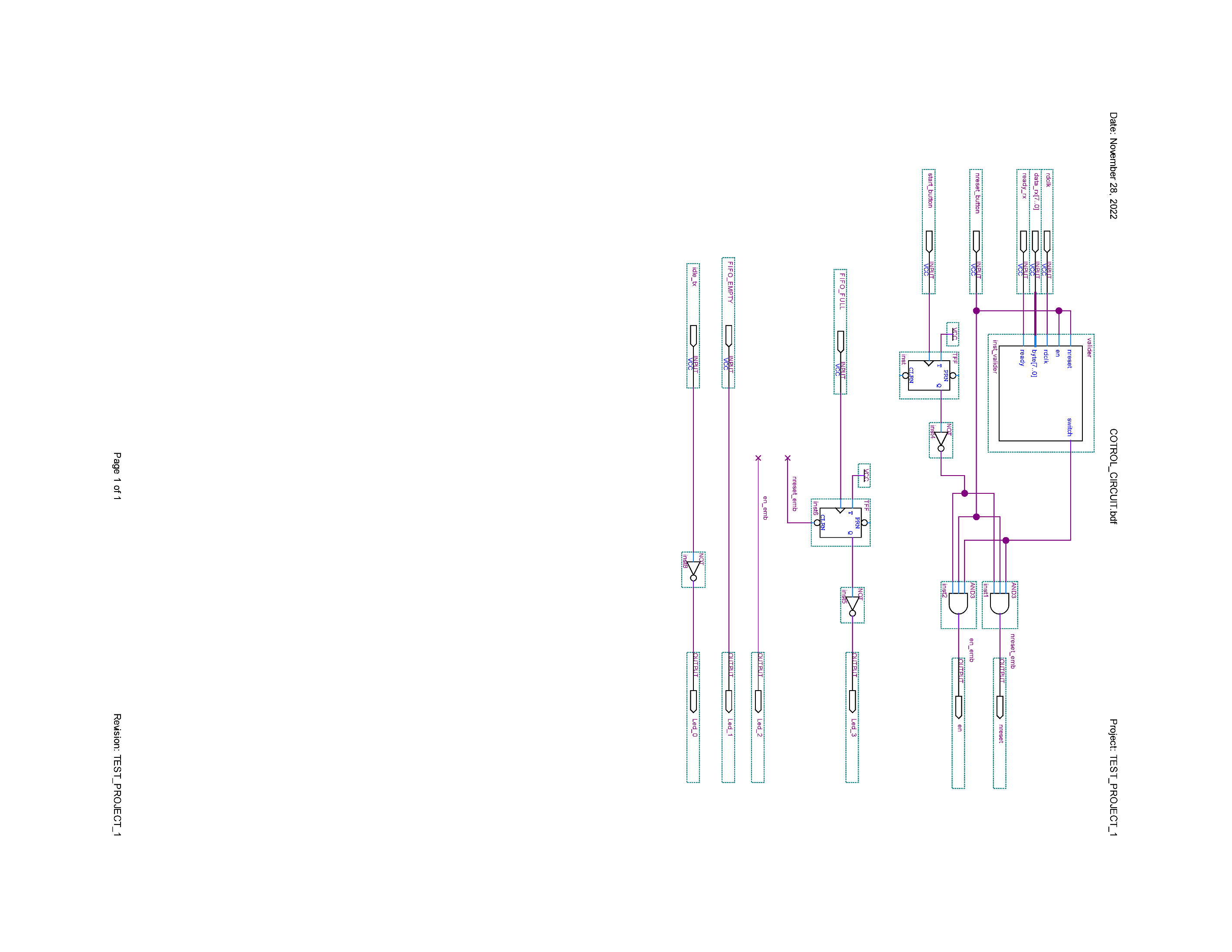
## Приложение А.1 Блок-схема верхнего уровня (стр. 1)



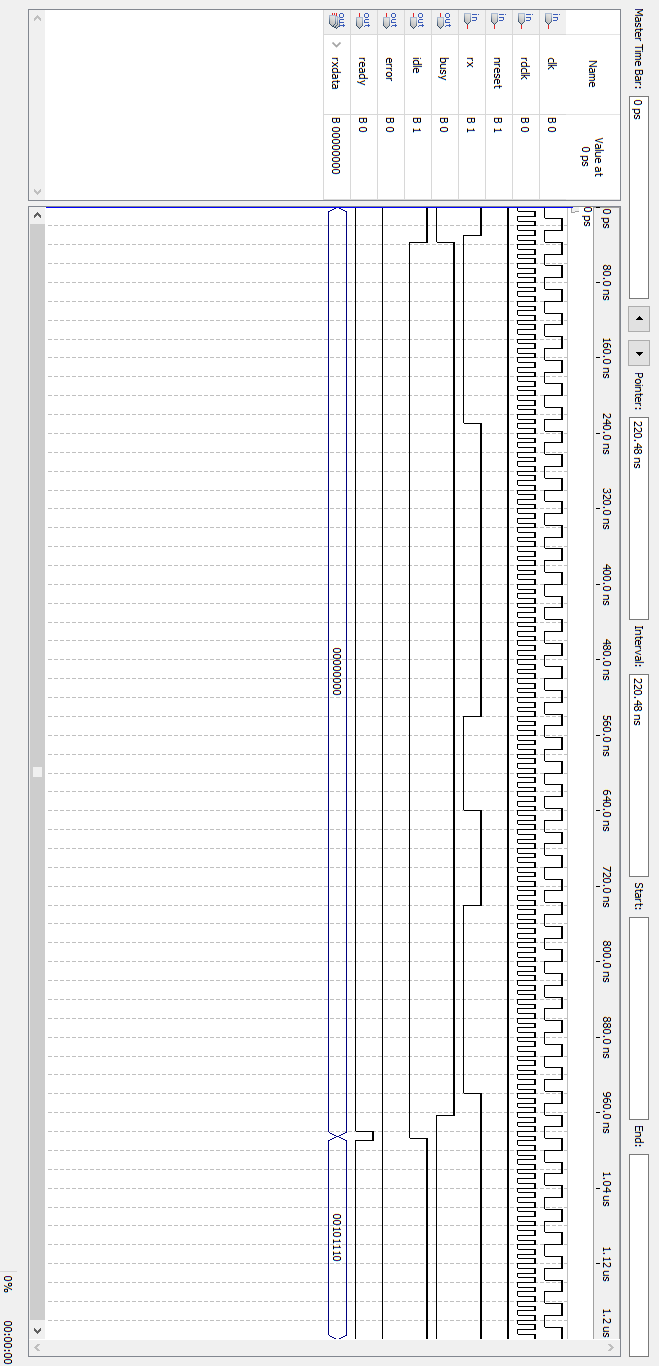
## Приложение 1.2 Блок-схема верхнего уровня (стр. 2)



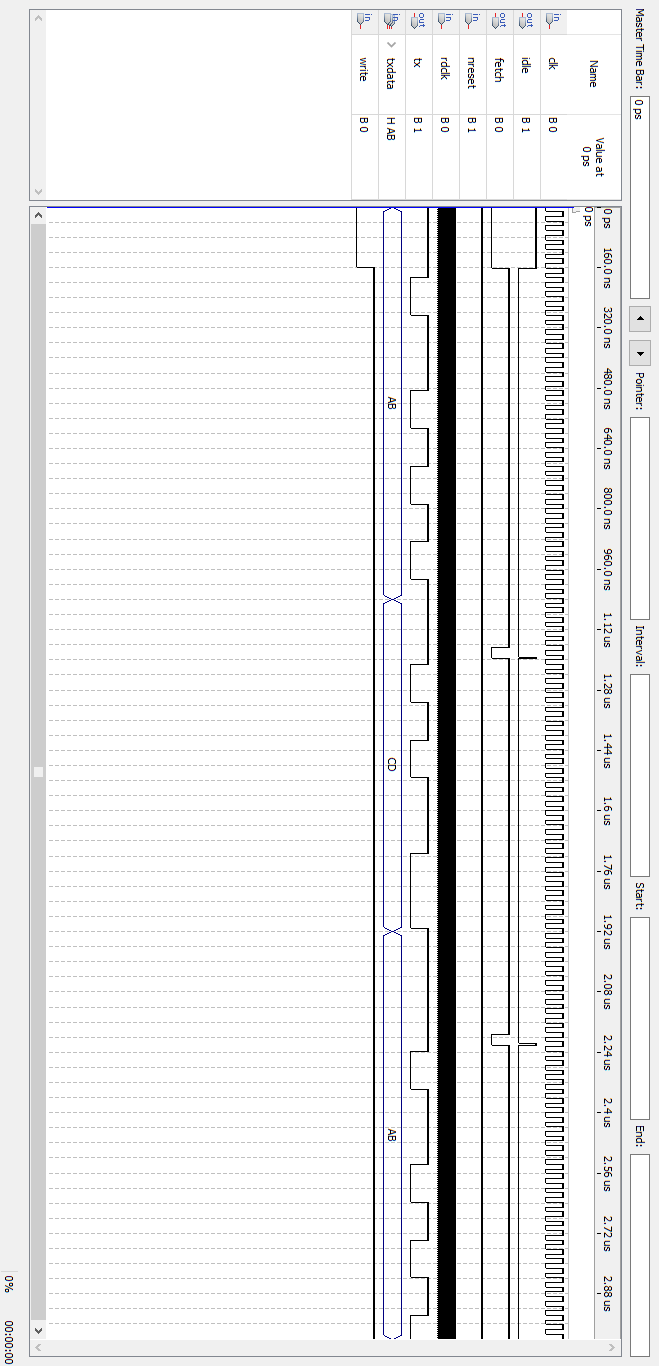
# Приложение Б. Блок-схема управляющего модуля



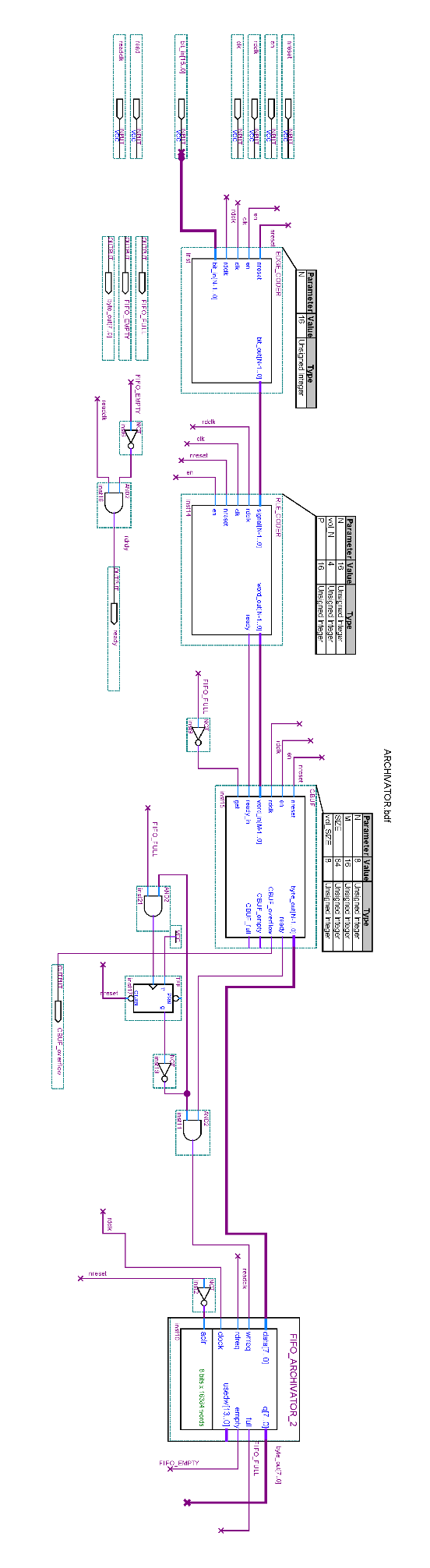
# Приложение В. Временная диаграмма приёмника UART



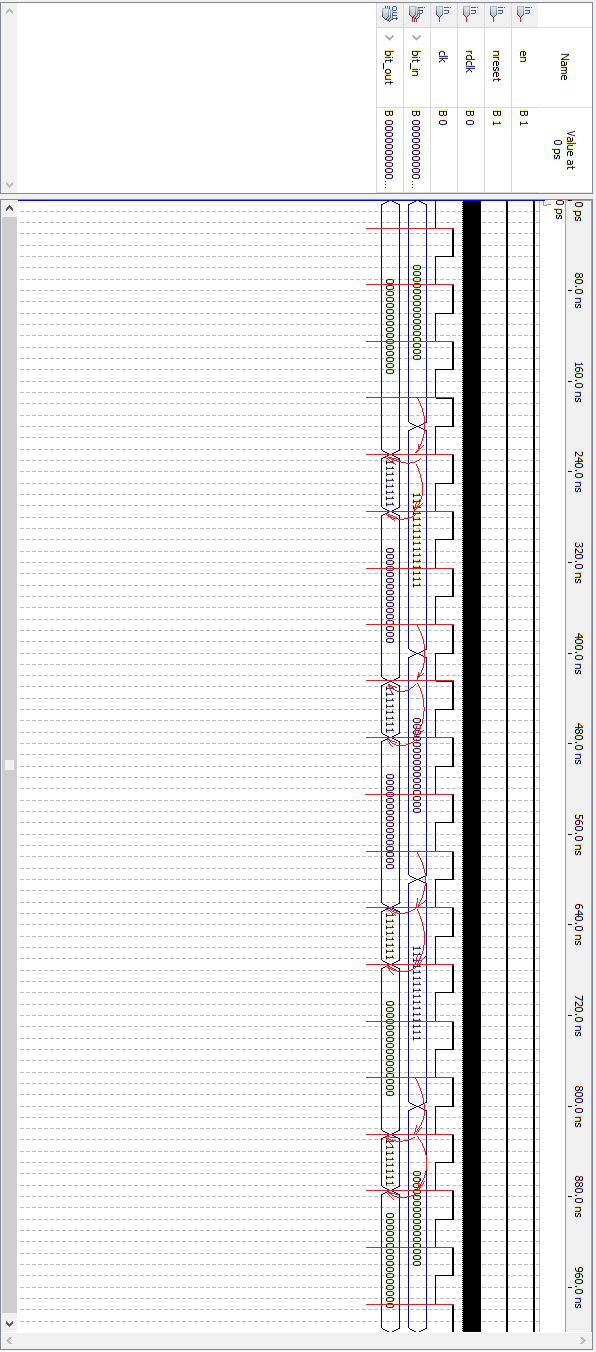
# Приложение Г. Временна диаграмма передатчика UART



# Приложение Д. Блок-схема архиватора

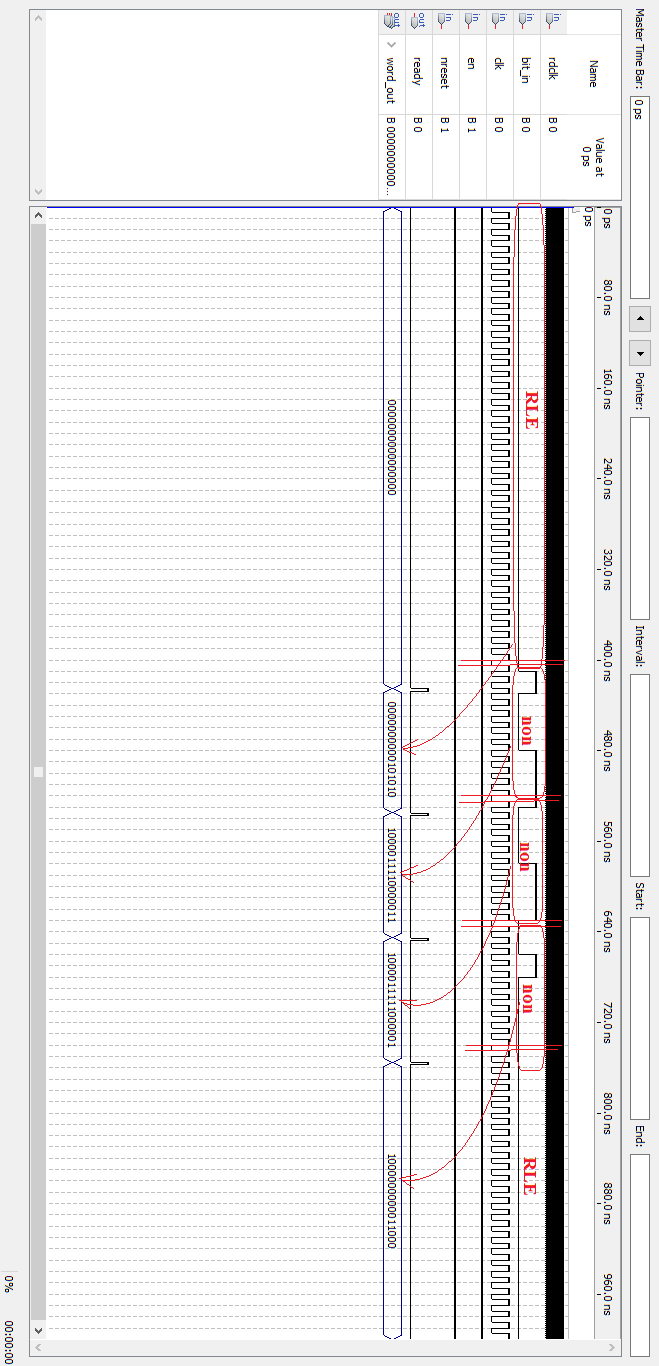


# Приложение Е. Временная диаграмма EDGE\_CODER

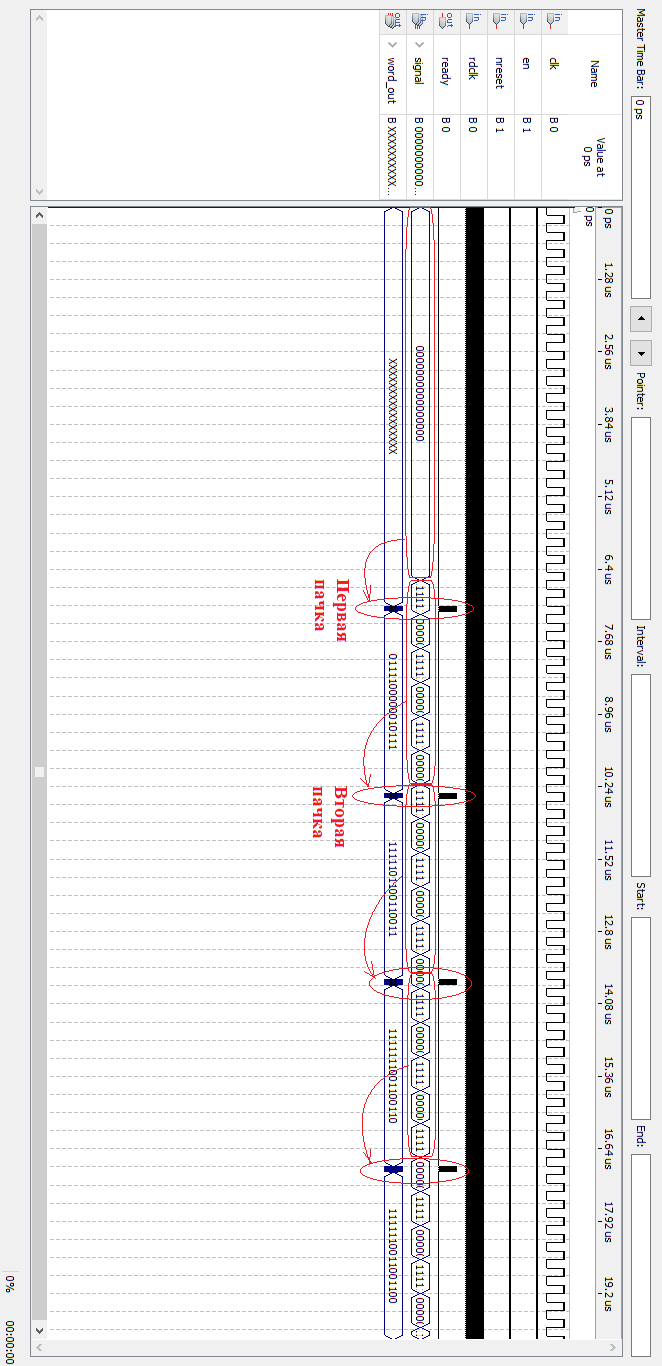


# Приложение Ж. Временные диаграммы модуля RLE сжатия

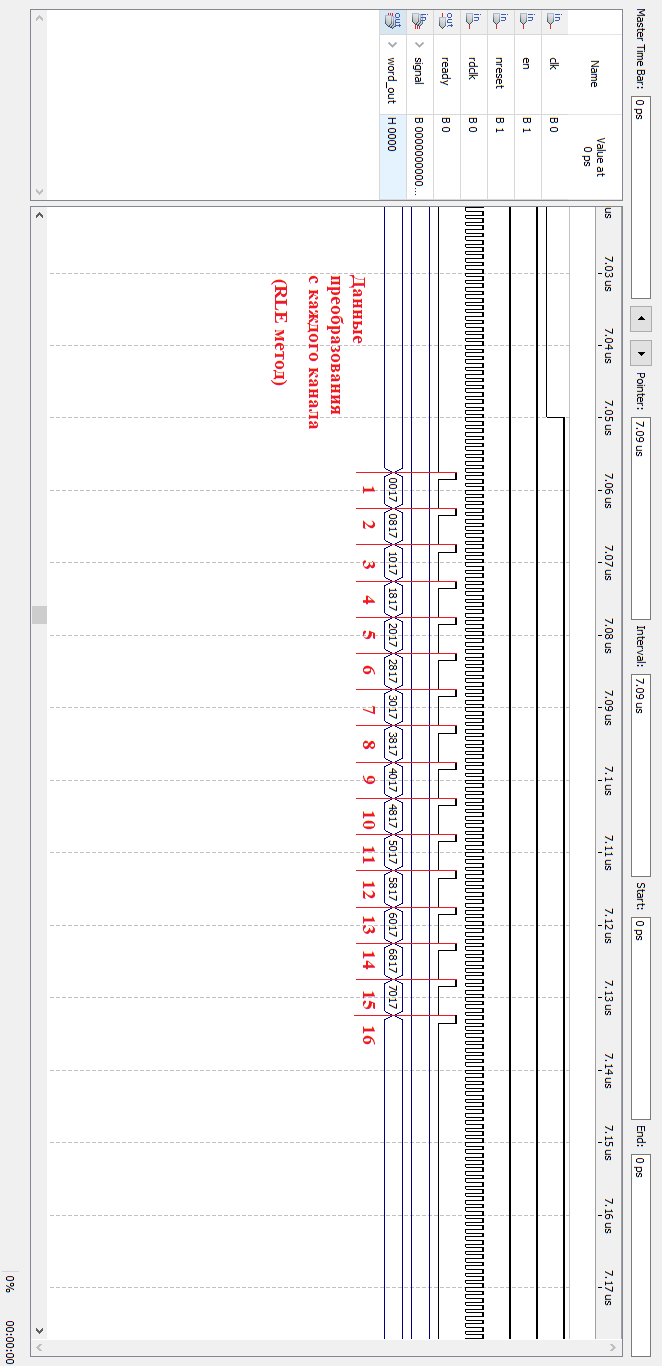
## Приложение Ж.1. Временная диаграмма модуля RLE\_CODER\_CODER



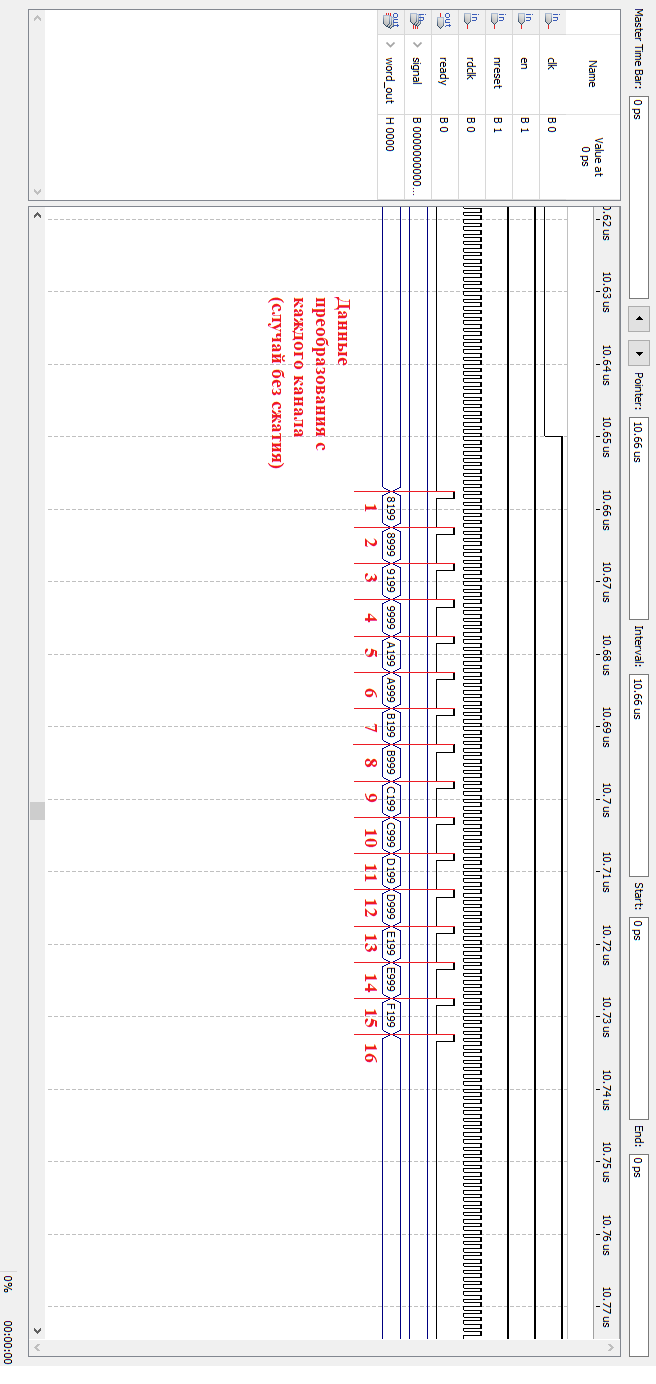
## Приложение Ж.2. Временная диаграмма модуля RLE\_CODER (Часть 1. Общая временна диаграмма (точка B))



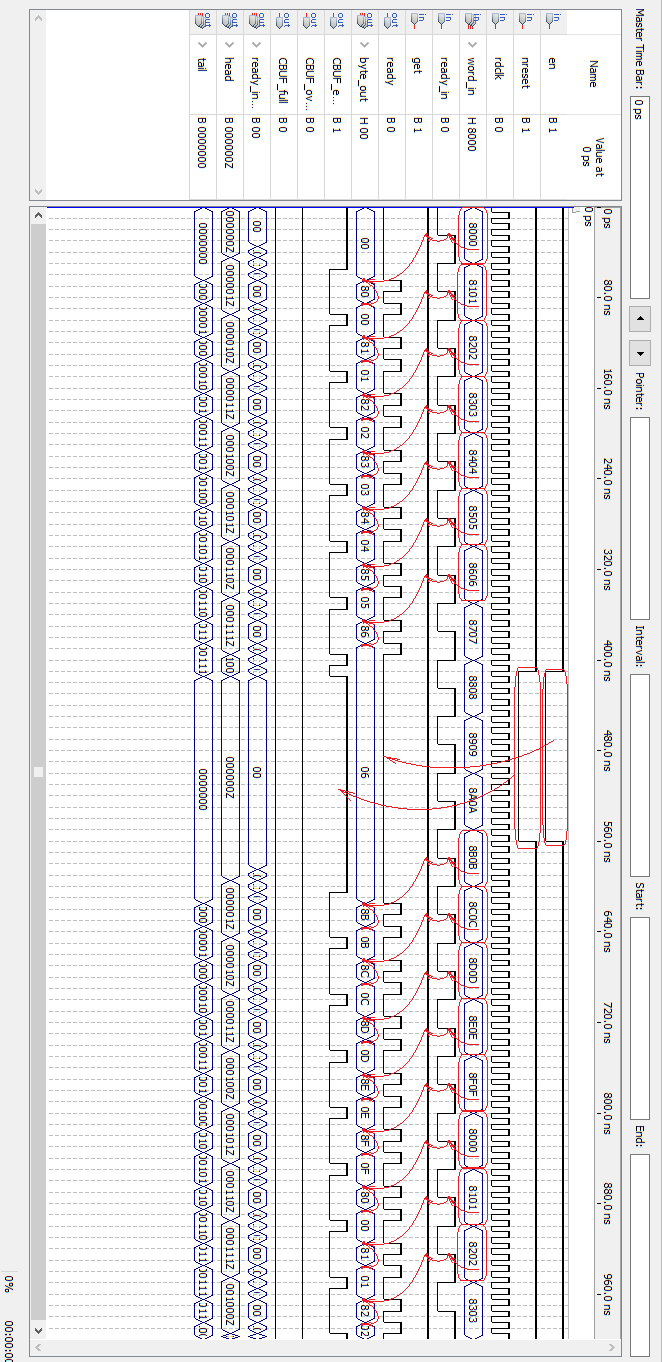
## Приложение Ж.3. Временная диаграмма модуля RLE\_CODER (Часть 2. Увеличенная временна диаграмма участка первой пачки передачи в HEX-коде)



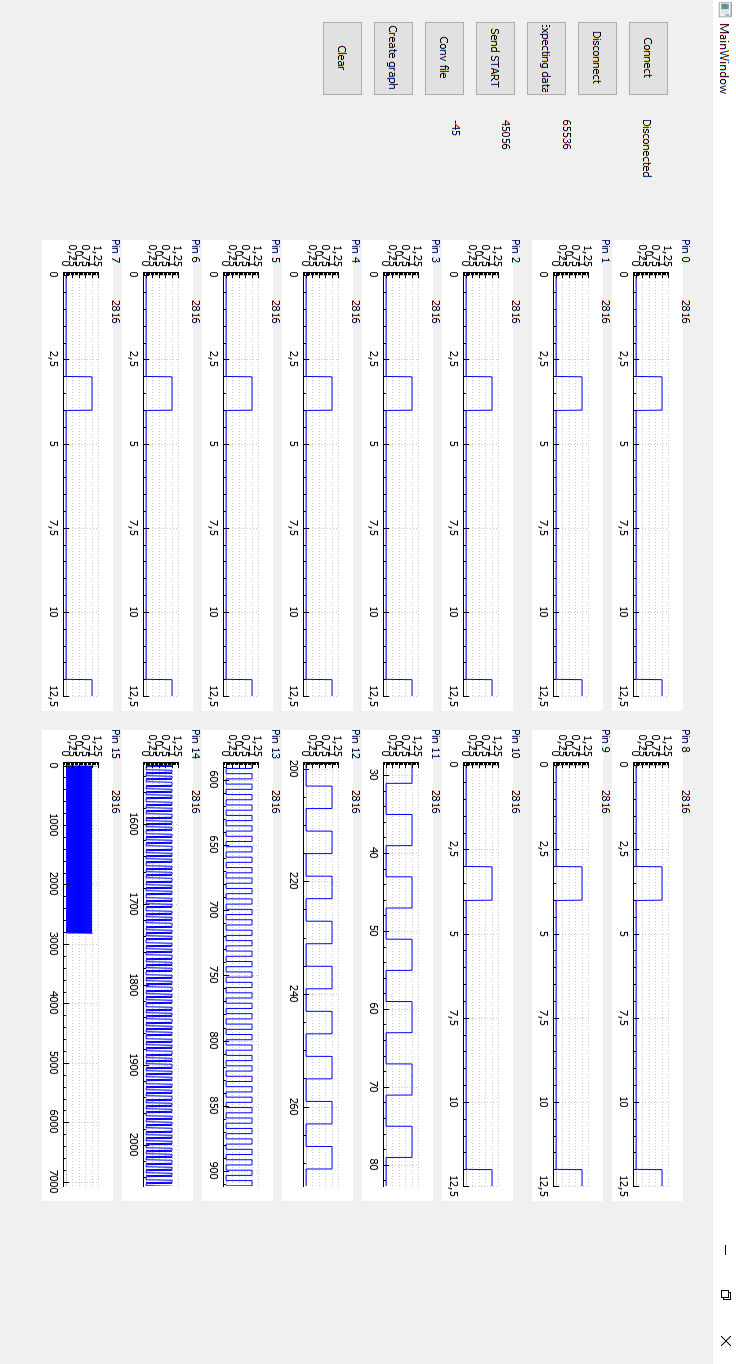
## Приложение Ж.4. Временная диаграмма модуля RLE\_CODER (Часть 3. Увеличенная временна диаграмма участка второй пачки передачи в HEX-коде)



# Приложение З. Временна диаграмма CBUF (точка C)



# Приложение И. Вид приложения для ПК



# Приложение К. Результаты исследования выбора частоты дискретизации

## Приложение К.1. Результаты эксперимента по выбору дискретизирующей частоты (Часть 1)

Обозначения, принятые в исследовании:

– коэффициент делителя (частота дискретизации получается путём преобразования частоты 50 МГц делителем, представляющим собой –разрядный счётчик);

– Частота дискретизации;

– Число отсчётов сжатой последовательности;

– Число отсчётов несжатой последовательности;

– Степень сжатия:

Таблица 2 – Эксперимент исследования работы устройства на высокой частоте сигнала (Часть 1)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер эксперимента | 1 | 2 | 3 | 4 | 5 |
|  | 5 | 6 | 7 | 8 | 9 |
|  | 1562,5 | 781,25 | 390,625 | 195,312 | 97,656 |
|  | 65536 | 65536 | 65536 | 65536 | 65536 |
|  | 45056 | 87398 | 45056 | 45056 | 45056 |
|  | -45% | 25% | -45% | -45% | -45% |
| Передача завершилась из-за переполнения? | ДА | ДА | ДА | ДА | ДА |
| Ошибки передачи | ДА | ДА | НЕТ | НЕТ | НЕТ |

Таблица 3 – Эксперимент исследования работы устройства на высокой частоте сигнала (Часть 2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер эксперимента | 6 | 7 | 8 | 9 |
|  | 10 | 11 | 12 | 13 |
|  | 48,828 | 24,414 | 12,207 | 6,103 |
|  | 65536 | 65536 | 129024 | 181403 |
|  | 45056 | 45056 | 88704 | 95772 |
|  | -45% | -45% | -45% | -45% |
| Передача завершилась из-за переполнения? | ДА | ДА | НЕТ | НЕТ |
| Ошибки передачи | НЕТ | НЕТ | НЕТ | НЕТ |

## Приложение К.2. Результаты эксперимента по выбору дискретизирующей частоты (Часть 2)

Таблица 4 – Эксперимент исследования работы устройства на низкой частоте сигнала (Часть 1)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер эксперимента | 1 | 2 | 3 | 4 | 5 |
|  | 5 | 6 | 7 | 8 | 9 |
|  | 1562,5 | 781,25 | 390,625 | 195,312 | 97,656 |
|  | 97280 | 97488 | 97280 | 63232 | 133632 |
|  | 8357472 | 12439795 | 8373760 | 8057168 | 17035424 |
|  | 98% | 99% | 98% | 99% | 99% |
| Передача завершилась из-за переполнения? | НЕТ | НЕТ | НЕТ | НЕТ | НЕТ |
| Ошибки передачи | ДА | ДА | НЕТ | НЕТ | НЕТ |

Таблица 5 – Эксперимент исследования работы устройства на низкой частоте сигнала (Часть 2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер эксперимента | 6 | 7 | 8 | 9 |
|  | 10 | 11 | 12 | 13 |
|  | 48,828 | 24,414 | 12,207 | 6,103 |
|  | 19712 | 41472 | 14080 | 5120 |
|  | 2489328 | 5270800 | 1739965 | 589888 |
|  | 99% | 99% | 99% | 99% |
| Передача завершилась из-за переполнения? | НЕТ | НЕТ | НЕТ | НЕТ |
| Ошибки передачи | НЕТ | НЕТ | НЕТ | НЕТ |

Таблица 6 – Эксперимент исследования работы устройства на высокой частоте сигнала при использовании 4 каналов из 16 (Часть 1)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер эксперимента | 1 | 2 | 3 | 4 | 5 |
|  | 5 | 6 | 7 | 8 | 9 |
|  | 1562,5 | 781,25 | 390,625 | 195,312 | 97,656 |
|  | 65536 | 65536 | 65744 | 65536 | 65536 |
|  | 45056 | 95956 | 167359 | 167216 | 167216 |
|  | -45% | 31% | 60% | 60% | 60% |
| Передача завершилась из-за переполнения? | ДА | ДА | ДА | ДА | ДА |
| Ошибки передачи | ДА | ДА | НЕТ | НЕТ | НЕТ |

## Приложение К.3. Результаты эксперимента по выбору дискретизирующей частоты (Часть 3)

Таблица 7 – Эксперимент исследования работы устройства на высокой частоте сигнала при использовании 4 каналов из 16 (Часть 2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер эксперимента | 6 | 7 | 8 | 9 |
|  | 10 | 11 | 12 | 13 |
|  | 48,828 | 24,414 | 12,207 | 6,103 |
|  | 97280 | 65536 | 65536 | 65536 |
|  | 266188 | 167216 | 170648 | 167216 |
|  | 63% | 60% | 61% | 60% |
| Передача завершилась из-за переполнения? | НЕТ | НЕТ | НЕТ | НЕТ |
| Ошибки передачи | НЕТ | НЕТ | НЕТ | НЕТ |

# Приложение Л. Результаты исследования эффективности работы применённой конфигурации

## Приложение Л.1. Результаты эксперимента по исследованию эффективности работы устройства (Часть 1)

В данном исследовании приняты следующие обозначения:

– коэффициент заполнения равный , и при равных и кратных ему :

– Отношение длительности логических «0» к «1»;

– Эквивалентная частота сигнала;

– Число отсчётов сжатой последовательности;

– Число отсчётов несжатой последовательности;

– Степень сжатия:

Таблица 8 – Испытание конечной конфигурации с частотой дискретизации 100 кГц модулированным сигналом (Часть 1)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Номер эксперимента | 1 | 2 | 3 | 4 | 5 | 6 |
|  | 1 | 0,5 | 0,33 | 0,25 | 0,2 | 0,16 |
|  | 0 | 1 | 2 | 3 | 4 | 5 |
|  | 10,00 | 4,50 | 3,00 | 2,27 | 1,80 | 1,50 |
|  | 65536 | 65536 | 65536 | 65536 | 65536 | 65536 |
|  | 45056 | 47056 | 66135 | 73664 | 101472 | 136400 |
|  |  |  |  |  |  |  |
| Передача завершилась из-за переполнения? | ДА | ДА | ДА | ДА | ДА | ДА |
| Ошибки передачи | НЕТ | НЕТ | НЕТ | НЕТ | НЕТ | НЕТ |

## Приложение Л.2. Результаты эксперимента по исследованию эффективности работы устройства (Часть 2)

Таблица 9 – Испытание конечной конфигурации с частотой дискретизации 100 кГц модулированным сигналом (Часть 2)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Номер эксперимента | 7 | 8 | 9 | 10 | 11 | 12 |
|  | 0,142 | 0,125 | 0,111 | 0,100 | 0,090 | 0,083 |
|  | 6 | 7 | 8 | 9 | 10 | 11 |
|  | 1,30 | 1,10 | 1,00 | 0,90 | 0,80 | 0,75 |
|  | 65600 | 65536 | 65536 | 97584 | 97280 | 287744 |
|  | 141520 | 181057 | 181328 | 357372 | 1962812 | 8780864 |
|  |  |  |  |  |  |  |
| Передача завершилась из-за переполнения? | ДА | ДА | ДА | ДА | ДА | НЕТ |
| Ошибки передачи | НЕТ | НЕТ | НЕТ | НЕТ | НЕТ | НЕТ |

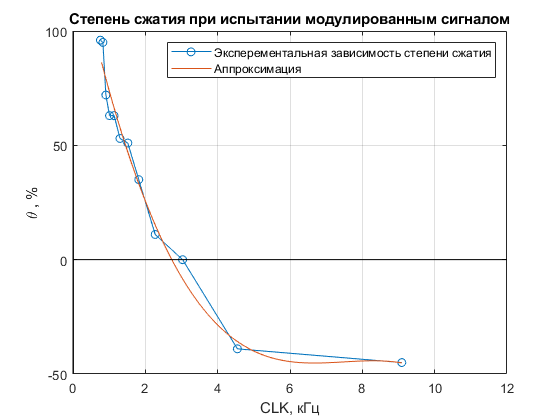


Рисунок 10 – График степени сжатия от эквивалентной частоты

## Приложение Л.3. Результаты эксперимента по исследованию эффективности работы устройства (Часть 3)

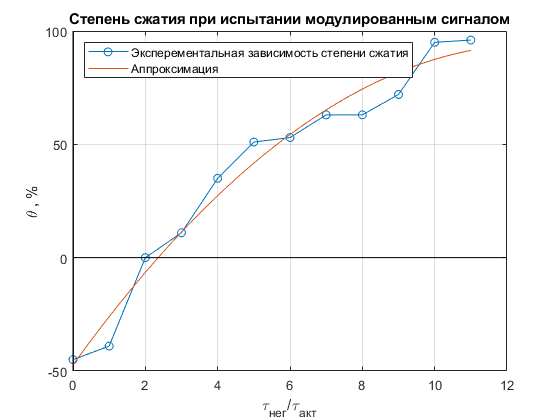


Рисунок 11 – График степени сжатия от отношения длительности уровней периода сигнала

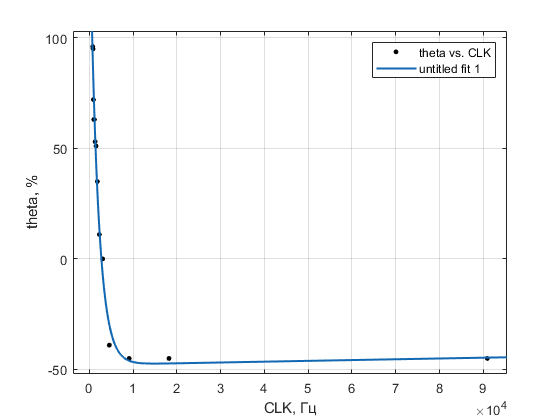


Рисунок 12 – График двух-экспоненциальной аппроксимации зависимости коэффициента сжатия от частоты сигнала