|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ высшего образования |
| **«Национальный исследовательский ядерный университет «МИФИ»** |
| ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ |
| КАФЕДРА «КОМПЬЮТЕРНЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ» (№12) |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**на курсовую учебно-исследовательскую работу по дисциплине**

**СХЕМОТЕХНИКА ЦИФРОВЫХ УСТРОЙСТВ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Тема: SerDes with align feature** | | | | |
| Студент | Кротов Евгений Геннадьевич | | Группа | С21-501 |
|  | ФИО | |  |  |
| Студент | | Косолобов Дмитрий Сергеевич | | Группа | С21-501 |
|  | | ФИО | |  |  |
|  |  | |  |  |
| Руководитель | | Решетько Валерий Михайлович, старший преподаватель НИЯУ МИФИ | | |
|  | | ФИО, степень, звание, должность | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  |  | Кротов Е.Г. |
| Студент | подпись |  | Косолобов Д.С |
|  | подпись |  | ФИО |
| Руководитель |  |  | Решетько В.М. |

подпись

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель |  |  | Сапрыкина В.С. |

подпись

**Москва, 2023**

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc148356518)

[1. Serializer/Deserializer 4](#_Toc148356519)

[1.1. Общая функция 4](#_Toc148356520)

[1.2. Align Feature 5](#_Toc148356521)

[1.3. Язык VHDL 5](#_Toc148356522)

[2. АЛГОРИТМ РАБОТЫ SERDES БЛОКОВ 7](#_Toc148356523)

[2.1. Разработка алгоритма 7](#_Toc148356524)

[2.2. Разработка программы 7](#_Toc148356525)

[2.3.Тестирование программы 10](#_Toc148356526)

[ЗАКЛЮЧЕНИЕ 13](#_Toc148356527)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 14](#_Toc148356528)

[Приложение 1 15](#_Toc148356529)

[Код блока сериализации 15](#_Toc148356530)

[Код блока десериализации 16](#_Toc148356531)

[Приложение 2 17](#_Toc148356532)

[Тесты к блоку сериализации 17](#_Toc148356533)

[Тесты к блоку десериализации 18](#_Toc148356534)

# ВВЕДЕНИЕ

Сериализатор/Десериализатор (SerDes) — пара функциональных блоков, обычно используемых в высокоскоростной связи, для преобразования данных между последовательным и параллельным интерфейсами в обоих направлениях. Термин «SerDes» относится к обобщённым интерфейсам, используемым в различных технологиях и приложениях. Основное использование SerDes — обеспечение высокоскоростной передачи данных по одиночной линии или по дифференциальной паре для уменьшения количества контактов ввода/вывода микросхемы и межсоединений между компонентами.

Цель данной работы — создание SerDes блоков с выравниванием, которые можно протестировать в среде моделирования.

Для достижения поставленной цели в работе решаются следующие задачи:

* изучение теоретического материала по языку VHDL, симулятору ModelSim и САПР Quartus;
* проектирование SerDes блоков с выравниванием;
* написание кода для SerDes блоков с выравниванием;
* тестирование работы SerDes блоков с выравниванием.

# 1. Serializer/Deserializer

## 1.1. Общая функция

Основная структура SerDes разделяется на два функциональных блока: блок параллельно-последовательного преобразователя (PISO, Parallel In Serial Out) и блок последовательно-параллельного преобразователя (SIPO, Serial In Parallel Out), изображенных на Рис.1.1. Есть четыре различных архитектуры SerDes:

— SerDes с параллельным тактированием;

— SerDes с внутренним тактированием;

— 8b/10b SerDes (или более сложные коды);

— SerDes с чередованием битов.

Блок параллельно-последовательного преобразователя PISO обычно имеет параллельный тактовый вход, набор входных линий данных, и вход фиксации данных. Он может использовать внутреннюю или внешнюю фазовую автоподстройку частоты (ФАПЧ) для умножения входящего параллельного тактового сигнала до частоты тактового сигнала последовательного выхода. Простейшая форма PISO имеет только сдвиговый регистр, который принимает параллельные данные с параллельным тактированием, и выдвигает его с более высокой последовательной тактовой частотой. Реализации могут также использовать двойную буферизацию регистра, чтобы избежать метастабильности при передаче данных между частями с разной тактовой частотой.

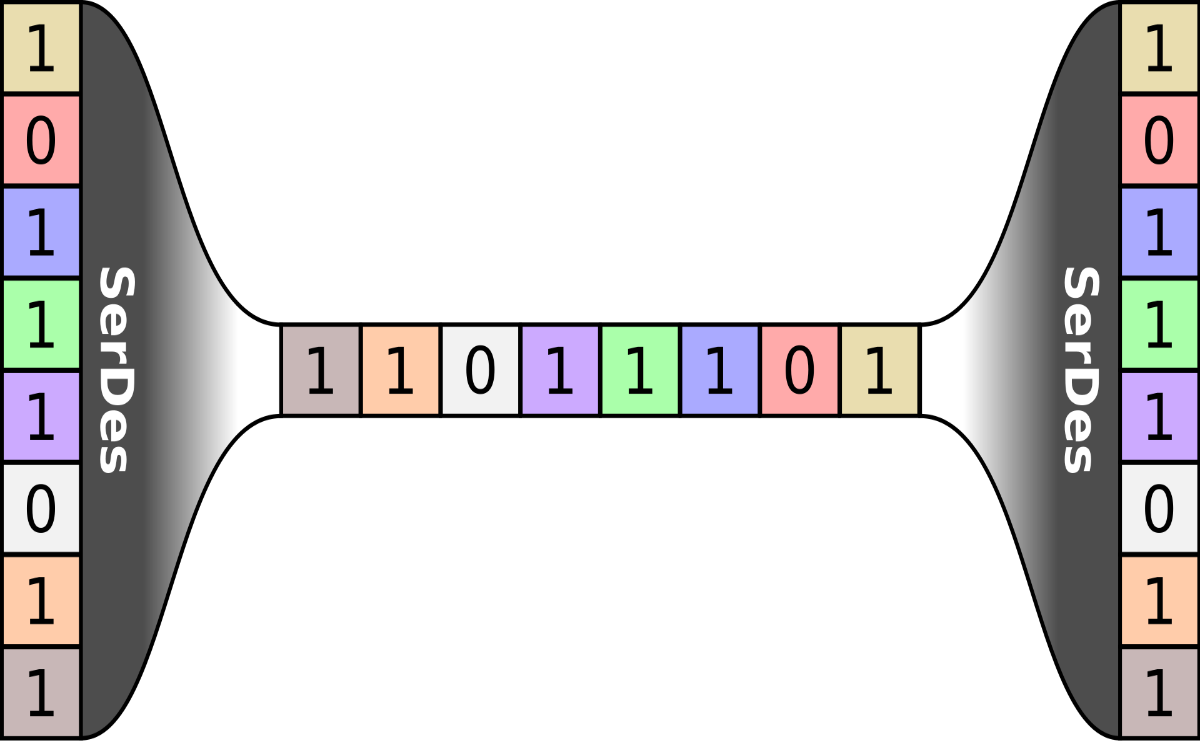


Рис. 1.1 Принцип работы SerDes.

SIPO, блок последовательно-параллельного преобразователя, обычно имеют выход входящей тактовой частоты, набор выходных линий данных и выходные защелки данных. Приёмный тактовый сигнал могут быть восстановлены из данных по методу восстановления тактового сигнала. Однако SerDes, которые не передают тактовые сигналы, используют образцовые такты для синхронизации ФАПЧ с необходимой частой TX, во избежание низкочастотных гармоник, присутствующих в потоке данных. Блок SIPO понижает входящую тактовую частоту для параллельного выхода. Реализации, как правило, имеют два регистра, подключенных как двойной буфер. Один регистр используется для тактирования последовательного потока, а другой используется для хранения данных более медленной, параллельной стороны.

## 1.2. Align Feature

Alignfeature (или Align Feature) — это функция или механизм, который помогает выравнивать и синхронизировать данные, передаваемые через SerDes. Выравнивание данных важно для того, чтобы правильно интерпретировать их на приемной стороне и избежать ошибок при передаче данных на высоких скоростях.

Некоторые типы SerDes включают блок кодирования/декодирования. Целью такого кодирования и декодирования является, как правило, изменение статистических свойств передаваемого сигнала для облегчения восстановления тактового сигнала в приёмнике и обеспечения синхронизации фреймов. Также кодирование может обеспечивать баланс по постоянному току.

## 1.3. Язык VHDL

VHDL – это язык описания аппаратуры (HDL), позволяющий моделировать поведение и структуру цифровых систем на различных уровнях абстракции, начиная с системного уровня и заканчивая уровнем логических вентилей, для целей проектирования, документирования и верификации. С 1987 года VHDL стандартизован Институтом инженеров по электротехнике и электронике (IEEE) в виде стандарта IEEE Std 1076, последней версией которого является IEEE Std 1076-2019. Для моделирования аналоговых и смешанных сигнальных систем был разработан стандартизованный IEEE HDL на основе VHDL под названием VHDL-AMS (официально IEEE 1076.1).

VHDL назван в честь программы Министерства обороны США, в рамках которой он был создан, - Very High Speed Integrated Circuits Program (VHSIC). В начале 1980-х годов в рамках программы VHSIC был проведен поиск нового языка программирования HDL для использования при проектировании интегральных схем, которые предполагалось создать. Результатом этих усилий стал VHDL версии 7.2, выпущенный в 1985 году. В следующем году началась работа по его стандартизации в качестве стандарта IEEE.

# 2. АЛГОРИТМ РАБОТЫ SERDES БЛОКОВ

## 2.1. Разработка алгоритма

Разработка алгоритма блока сериализатора состоит из следующих этапов:

1. При асинхронном сбросе (i\_rstb = '0'), сдвиговый регистр r\_data и счетчик r\_count обнуляются, и сигналы o\_data\_ena и o\_data устанавливаются в '0'.

2. При наступлении положительного фронта тактового сигнала i\_clk и при условии, что сброс синхронизации (i\_sync\_reset) не активен (i\_sync\_reset = '1'), происходит процесс сериализации данных.

3. Если счетчик r\_count равен 0, то данные из i\_data записываются в сдвиговый регистр r\_data, и самый старший бит (i\_data(7)) устанавливается на выходе o\_data. Сигнал o\_data\_ena устанавливается в '1', и счетчик r\_count увеличивается на 1.

4. Если счетчик r\_count меньше 8, то на выходе o\_data появляется следующий бит из сдвигового регистра r\_data, и сигнал o\_data\_ena остается '1'. Счетчик r\_count увеличивается на 1.

5. Как только счетчик r\_count достигает значения 8, сигнал o\_data\_ena устанавливается в '0', и процесс сериализации завершается.

Таким образом, устройство берет 8-битные данные, поочередно отправляет их бит за битом на выход o\_data, и управляющий сигнал o\_data\_ena указывает, когда данные на выходе действительны.

Разработка алгоритма блока десериализатора состоит из следующих этапов:

1. В начале, при сбросе (i\_rstb = '0'), сигналы r\_data\_enable, r\_count, o\_data\_valid устанавливаются в '0'., сдвиговый регистр r\_data и выходной сигнал o\_data устанавливаются в 8 нулей.

2. При наступлении положительного фронта тактового сигнала i\_clk и при условии, что сброс синхронизации (i\_sync\_reset) не активен (i\_sync\_reset = '1'), начинается процесс десериализации данных.

3. Выходной сигнал o\_data\_valid устанавливается равным r\_data\_enable, что означает, что данные на выходе действительны, только если сигнал r\_data\_enable равен '1'.

4. Если r\_data\_enable равен '1', то 8-битное слово r\_data передается на выход o\_data.

5. Если активен входной сигнал i\_data\_ena = '1', означающий, что в десериализатор переданы корректные данные, происходит процесс десериализации. Бит данных i\_data добавляется в младший конец сдвигового регистра r\_data, а счетчик r\_count увеличивается на 1.

6. Если счетчик r\_count достигает значения 7 (что означает, что все 8 бит данных были десериализованы), то счетчик r\_count сбрасывается в 0, и сигнал r\_data\_enable устанавливается в '1', что указывает, что данные готовы к выводу.

7. Если счетчик r\_count меньше 7, счетчик r\_count увеличивается на 1, и сигнал r\_data\_enable устанавливается в '0', указывая, что данные еще не готовы к выводу.

8. Если входной сигнал i\_data\_ena не активен (i\_data\_ena = '0'), сигнал r\_data\_enable устанавливается в '0', и процесс десериализации приостанавливается.

Таким образом, устройство принимает последовательные биты данных и десериализует их в 8-битное слово, которое становится доступным на выходе, когда все 8 бит данных были десериализованы.

Условное графическое обозначение представлено на рисунке 2.1.

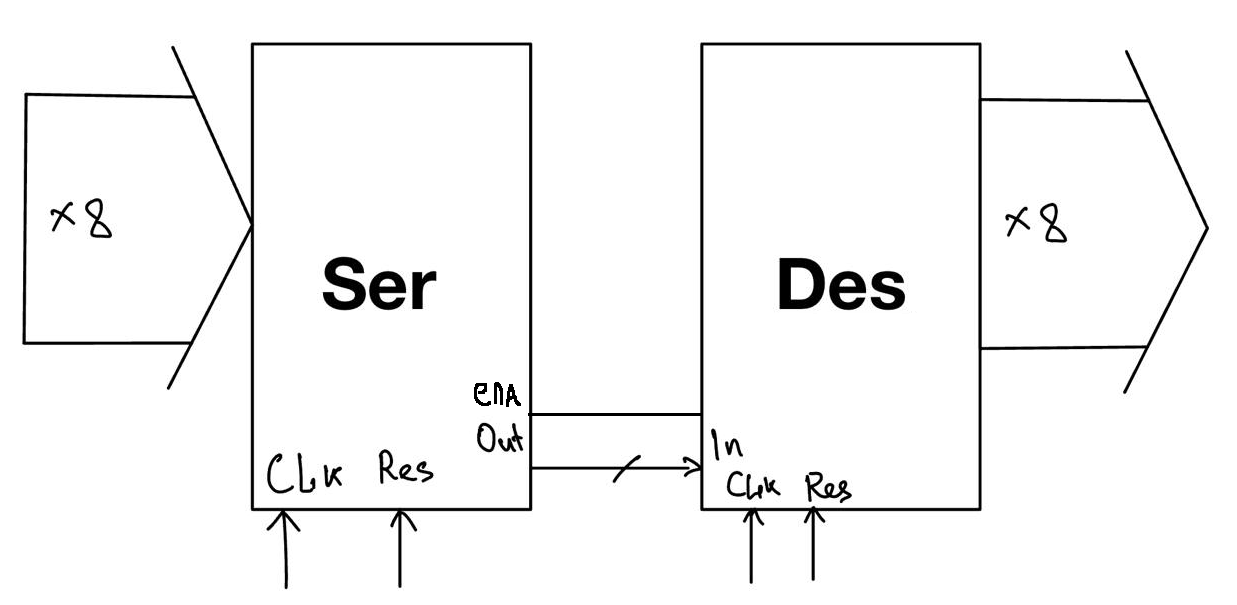


Рис. 2.1. УГО блоков сереализации и десереализации.

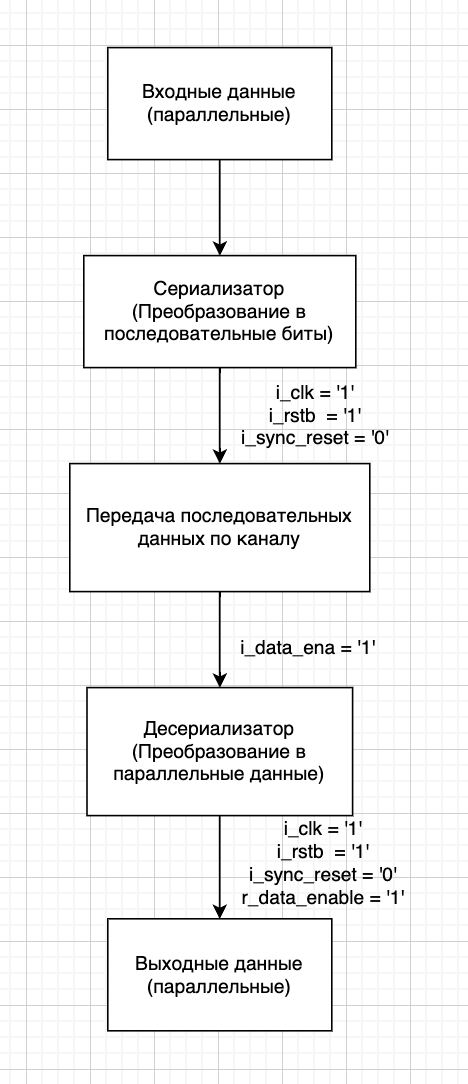


Рис. 2.2. Внутренняя блок схема блоков сереализации и десереализации.

## 2.2. Разработка программы

Разработка программы устройства SerDes разделена на два блока:

* Serializer;
* Deserializer.

*Serializer.*

Модуль для сериализации восьми бит данных в один бит. Ниже описан список портов ввода-вывода и режимы работы этого модуля.

Список портов ввода-вывода:

*i\_clk (вход)* - тактовый сигнал (clock), используется для синхронизации операций в модуле.

*i\_rstb (вход)* - сигнал асинхронного сброса (reset), используется для сброса состояния модуля.

*i\_sync\_reset (вход)* - сигнал синхронного сброса, который сбрасывает счетчик и управление данными.

*i\_data (вход)* - входные параллельные данные, 8-битный вектор данных для сериализации.

*o\_data (выход)* - выходной бит данных после сериализации (1 бит).

*o\_data\_ena (выход)* - сигнал разрешения данных, указывает, что данные на выходе действительны.

Режимы работы cериализатора:

*Сброс (Reset):*

Когда сигнал i\_rstb (асинхронный сброс) равен '0', сериализатор находится в режиме сброса.В этом режиме сигналы r\_data, r\_count, o\_data, и o\_data\_ena устанавливаются в исходные значения:

r\_data устанавливается в нулевое значение (все биты '0').

r\_count устанавливается в 0.

o\_data\_ena устанавливается в '0'.

o\_data устанавливается в '0'.

*Режим ожидания (Idle Mode):*

Пока сигнал i\_sync\_reset равен '1', сериализатор находится в режиме ожидания.

r\_count остается равным 0.

o\_data\_ena устанавливается в '0'.

*Режим сериализации (Serialization Mode):*

Когда сигнал i\_sync\_reset равен '0', сериализатор находится в режиме сериализации.

На каждом такте i\_clk имеем:

Если r\_count равен 0 (начало новой последовательности), входные данные i\_data записываются в регистр r\_data, и на выходе o\_data устанавливается старший бит входных данных (i\_data(7)). Также o\_data\_ena устанавливается в '1', указывая на наличие сериализованных данных.

Если r\_count находится в диапазоне от 1 до 7 (сериализация оставшихся битов), на выходе o\_data устанавливаются оставшиеся биты из регистра r\_data. Также o\_data\_ena устанавливается в '1'.

Если r\_count достигает 8 (все 8 бит данных сериализованы), o\_data\_ena устанавливается в '0', указывая на конец последовательности.

*Deserializer.*

Модуль для десериализации последовательных данных в восемь бит параллельных. Ниже описан список портов ввода-вывода и режимы работы этого модуля.

i\_clk (вход) - тактовый сигнал (clock), используется для синхронизации операций в модуле.

*i\_rstb (вход)* - сигнал асинхронного сброса (reset), используется для сброса состояния модуля.

*i\_sync\_reset (вход)* - сигнал синхронного сброса, который сбрасывает счетчик и управление данными.

*i\_data (вход)* - входные последовательные данные для десериализации.

*i\_data\_ena (вход)* - Вход, указывающий, разрешено ли записывать данные. Если активен (равен 1), данные будут записаны в десериализатор.

*o\_data (выход)* - сигнал, который представляет десериализованные данные. Этот сигнал будет содержать 8 бит данных.

*o\_data\_valid (выход)* - сигнал указывающий на доступность данных в выходе десериализатора. Если активен (равен 1), данные в o\_data действительны.

Режимы работы деcериализатора:

*Сброс (Reset):*

Когда сигнал i\_rstb (асинхронный сброс) равен '0', десериализатор находится в режиме сброса.

В этом режиме сигналы r\_data\_enable, r\_count, r\_data, o\_data\_valid и o\_data устанавливаются в исходные значения:

r\_data\_enable устанавливается в '0'.

r\_count устанавливается в 0.

r\_data устанавливается в нулевое значение (все биты '0').

o\_data\_valid устанавливается в '0'.

o\_data устанавливается в нулевое значение (все биты '0').

*Режим ожидания (Idle Mode):*

Пока сигнал i\_sync\_reset равен '1' или сигнал i\_data\_ena равен '0', десериализатор находится в режиме ожидания.

r\_count остается равным 0.

r\_data\_enable устанавливается в '0'.

*Режим десериализации (Deserialization Mode):*

Когда сигнал i\_sync\_reset равен '0' и сигнал i\_data\_ena равен '1', десериализатор находится в режиме десериализации.

Каждый раз, когда поступает новый бит данных i\_data, он добавляется к существующим данным r\_data справа (сдвиг на один бит влево) для формирования последовательности байт r\_data. Счетчик r\_count увеличивается на 1 с каждым новым битом данных. Когда r\_count достигает 7 (последний байт данных), сигнал r\_data\_enable устанавливается в '1', и данные готовы к выводу. Данные на выходе (o\_data) становятся доступными только после того, как r\_data\_enable установлен в '1'. После этого они передаются на выход o\_data, и сигнал o\_data\_valid устанавливается в '1', указывая на наличие десериализованных данных.

*Выходной режим (Output Mode):*

Когда сигнал o\_data\_valid равен '1', десериализированные данные доступны на выходе o\_data. Сигнал r\_data\_enable также остается в '1', чтобы обеспечить стабильность выходных данных. В этом режиме данные могут быть использованы в других частях цифровой схемы.

## Тестирование программы

Файлы SerDes\_Testbench используются для проверки работоспособности и соответствия спецификации модуля SerDes. В этих тестах были предприняты следующие действия для проверки модуля:

1. Создание сигналов и компонентов.

Вначале определены сигналы, которые будут подключены к портам модуля Serializer. Эти сигналы включают тактовый сигнал i\_clk, сигнал сброса i\_rstb, сигнал синхронного сброса i\_sync\_reset, а также входные данные i\_data и выходные сигналы o\_data\_ena и o\_data.

1. Генерация тактового сигнала.

В тестбенче создан процесс, который генерирует тактовый сигнал i\_clk. Этот сигнал переключается между '0' и '1' с периодом 10 наносекунд. Это обеспечивает тактовый сигнал, который используется для управления операциями в модуле.

1. Генерация входных данных и сброса.

Тестовые случаи для сериализатора:

В тестбенче определен процесс, который генерирует входные данные i\_data и сигнал сброса i\_rstb.

Test Case 1 Data Input: входные данные i\_data устанавливаются в "11001010".

Test Case 2 Reset: сигнал сброса i\_rstb сначала устанавливается в '0', а затем возвращается в '1' после некоторой задержки.

Тестовые случаи для десериализатора:

Test Case 1 Reset: сначала проводится сброс десериализатора, устанавливая i\_rstb в '0', а затем снимая сброс, устанавливая i\_rstb в '1'.

Test Case 2 Data Input: отправляется последовательность данных с разными значениями i\_data и сигналом разрешения данных i\_data\_ena. Эти данные демонстрируют, как десериализатор обрабатывает данные входные биты.

Test Case 3 Data Invalid: устанавливается i\_data в '1' без разрешения данных, чтобы продемонстрировать, как десериализатор обрабатывает недействительные данные.

Test Case 4 Synchronous Reset: активируется синхронный сброс, устанавливая i\_sync\_reset в '1', а затем снимаете сброс, устанавливая i\_sync\_reset в '0'.

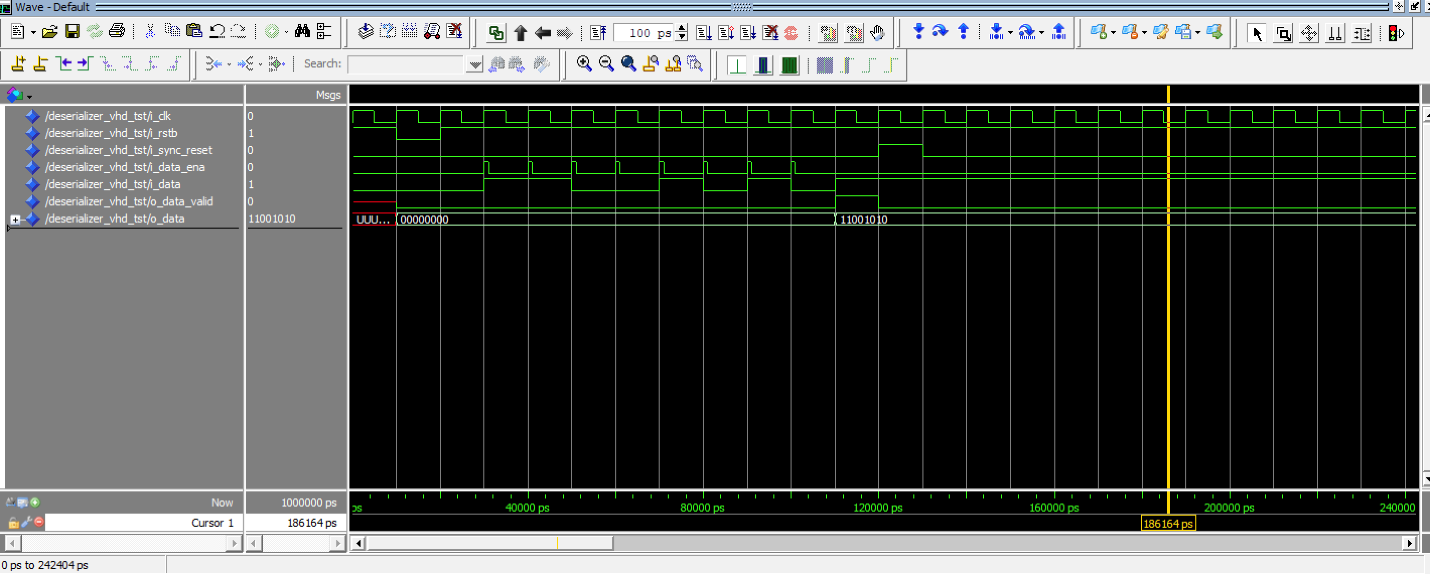


Рис.2.3. Временная диаграмма Des блока.

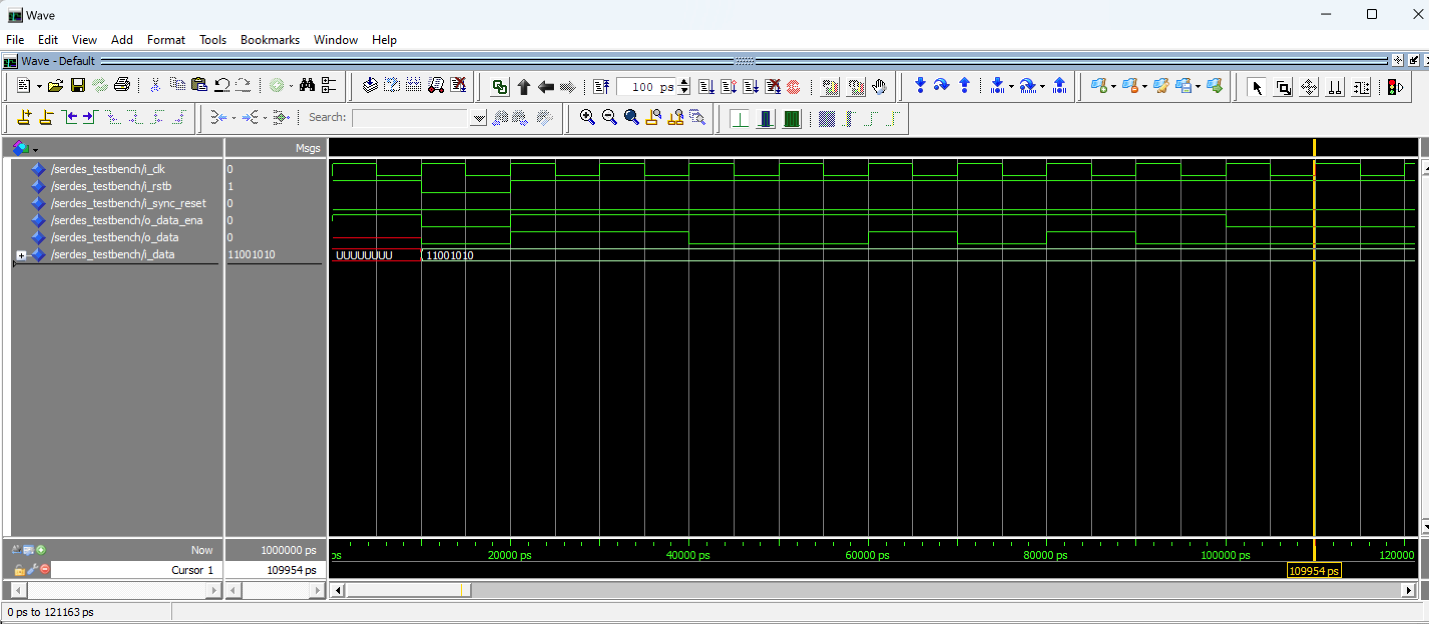


Рис.2.4. Временная диаграмма Ser блока.

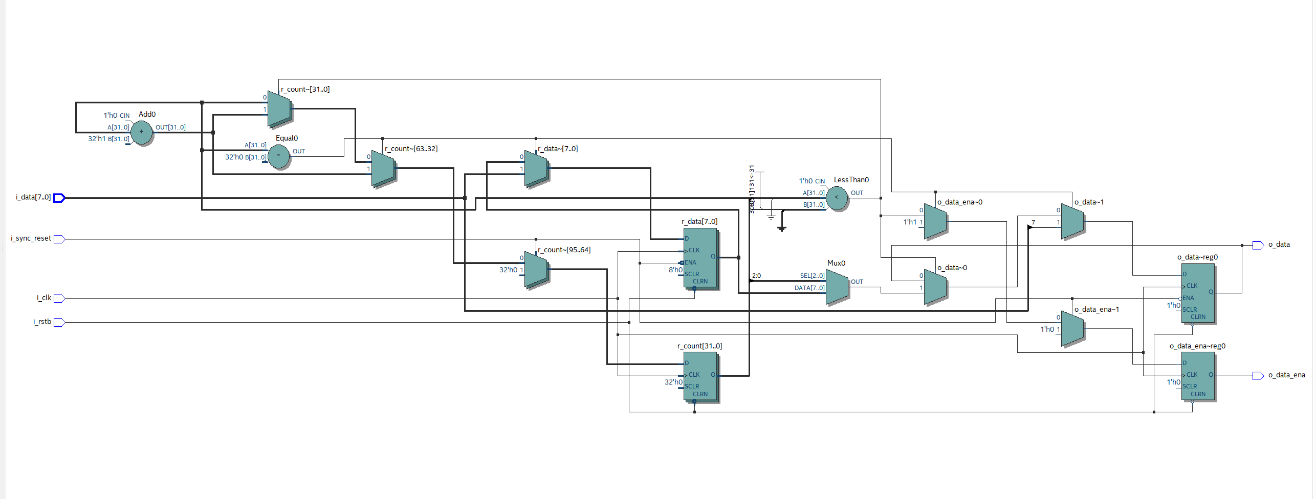


Рис.2.5 RTL-схема сереализатора.

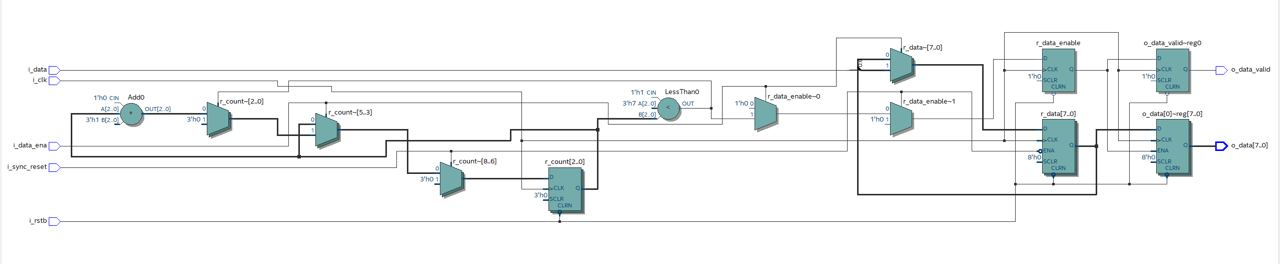


Рис.2.6 RTL-схема десереализатора.

# ЗАКЛЮЧЕНИЕ

В ходе курсовой работы был разработаны модули сериализации и десереализации SerDes с функцией выравнивания на языке VHDL. Работа модулей соответствует заявленной спецификации.

В ходе работы были получены навыки работы с языком описания оборудования VHDL, приобретен опыт работы с программным обеспечением Quartus Prime Standart, ModelSim и опыт работы в команде.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. SerDes [Электронный ресурс]. — Режим доступа: Интернет: https://ru.wikipedia.org/wiki/SerDes (дата обращения: 01.10.2023)
2. ModelSim - Altera Edition [Электронный ресурс]. — Режим доступа: Интернет: http://altera.ru/soft\_modelsim-ae.html (дата обращения: 29.09.2023)
3. FPGA Design Software - Intel® Quartus® Prime [Электронный ресурс]. — Режим доступа: Интернет: https://www.intel.com/content/www/us/en/products/details/fpga/development-tools/quartus-prime.html. (дата обращения: 29.09.2023)
4. VHDL [Электронный ресурс]. — Режим доступа: Интернет: https://ru.wikipedia.org/wiki/VHDL (дата обращения: 29.09.2023)

# Приложение 1

## Код блока сериализации

## Код блока десериализации

# Приложение 2

## Тесты к блоку сериализации

## Тесты к блоку десериализации

