Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт кибернетики
Направление подготовки 09.03.01 «Информатика и вычислительная техника»
Кафедра информационных систем и технологий

БАКАЛАВРСКАЯ РАБОТА

Тема работы
Разработка устройства межинтерфейсного взаимодействия между AXI и SPI на ПЛИС

УДК 004.5 : 621.382.049.77

Студент

Группа	ФИО	Подпись	Дата
8В3А	Старшинов Владислав Сергеевич		

Руководитель

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент кафедры ИСТ	Мальчуков Андрей Николаевич	к.т.н., доцент		

КОНСУЛЬТАНТЫ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент кафедры менеджмента	Антонова Ирина Сергеевна	к.э.н.		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Инженер	Маланова Наталья Викторовна	к.т.н.		

ДОПУСТИТЬ К ЗАЩИТЕ:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Зав. кафедрой ИСТ	Мальчуков Андрей Николаевич	к.т.н., доцент		

Томск – 2017

ЗАПЛАНИРОВАННЫЕ РЕЗУЛЬТАТЫ ПО ОСНОВНОЙ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЕ ПОДГОТОВКИ БАКАЛАВРОВ 09.03.01 «ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА», ИК ТПУ, ПРОФИЛЬ «ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ, КОМПЛЕКСЫ, СИСТЕМЫ И СЕТИ»

Код результатов	Результат обучения (выпускник должен быть готов)
Профессиональные компетенции	
Р1	Применять базовые и специальные естественнонаучные и математические знания в области информатики и вычислительной техники, достаточные для комплексной инженерной деятельности.
Р2	Применять базовые и специальные знания в области современных информационных технологий для решения инженерных задач.
Р3	Ставить и решать задачи комплексного анализа, связанные с созданием аппаратно-программных средств информационных и автоматизированных систем, с использованием базовых и специальных знаний, современных аналитических методов и моделей.
Р4	Разрабатывать программные и аппаратные средства (системы, устройства, блоки, программы, базы данных и т. п.) в соответствии с техническим заданием и с использованием средств автоматизации проектирования.
Р5	Проводить теоретические и экспериментальные исследования, включающие поиск и изучение необходимой научно-технической информации, математическое моделирование, проведение эксперимента, анализ и интерпретация полученных данных, в области создания аппаратных и программных средств информационных и автоматизированных систем.
Р6	Внедрять, эксплуатировать и обслуживать современные программно-аппаратные комплексы, обеспечивать их высокую эффективность, соблюдать правила охраны здоровья, безопасность труда, выполнять требования по защите окружающей среды.
Универсальные компетенции	
Р7	Использовать базовые и специальные знания в области проектного менеджмента для ведения комплексной инженерной деятельности.
Р8	Владеть иностранным языком на уровне, позволяющем работать в иноязычной среде, разрабатывать документацию, презентовать и защищать результаты комплексной инженерной деятельности.
Р9	Эффективно работать индивидуально и в качестве члена группы, состоящей из специалистов различных направлений и квалификаций, демонстрировать ответственность за результаты работы и готовность следовать корпоративной культуре организации.
Р10	Демонстрировать знания правовых, социальных, экономических и культурных аспектов комплексной инженерной деятельности.
Р11	Демонстрировать способность к самостоятельной к самостоятельному обучению в течение всей жизни и непрерывному самосовершенствованию в инженерной профессии.

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт Кибернетики
Направление подготовки 09.03.01 «Информатика и вычислительная техника»
Кафедра Информационных Систем и Технологий

УТВЕРЖДАЮ:
Зав. кафедрой
_____ _____ Мальчуков А.Н.
(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

Бакалаврской работы

Студенту:

Группа	ФИ
8В3А	Старшинову Владиславу Сергеевичу

Тема работы:

Разработка устройства межинтерфейсного взаимодействия между AXI и SPI на ПЛИС.	
Утверждена приказом директора (дата, номер)	От 07.02.2017 № 709/с

Срок сдачи студентом выполненной работы:	31.05.2017

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе	Техническое задание к реализации устройства межинтерфейсного взаимодействия AXI-to-SPI.
Перечень подлежащих исследованию, проектированию и разработке вопросов	Изучение документации последовательных и параллельных интерфейсов; Выбор проектных решений и инструментов для программно-аппаратной реализации межинтерфейсного взаимодействия; Реализация устройства межинтерфейсного взаимодействия AXI-to-SPI на ПЛИС; Разработка методики тестирования и системы апробации устройства;

	Тестирование и апробация разработанного устройства межинтерфейсного взаимодействия AXI-to-SPI на ПЛИС; Проектирование, трассировка и компоновка элементов на печатной плате; Расчет ресурсоэффективности и ресурсосбережения; Анализ вредных производственных факторов.
Перечень графического материала	Структурная схема устройства; Блоки и ГСА автоматов блоков устройства; Диаграммы работы устройства; Фотографии устройства и осциллографа; УГО и посадочные места элементов для апробации устройства; Печатные платы разрабатываемого устройства.
Консультанты по разделам выпускной квалификационной работы	
Раздел	**Консультант**
Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	Доцент кафедры менеджмента Антонова И.С.
Социальная ответственность	Инженер Маланова Н.В.

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	19.09.2016

Задание выдал руководитель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент кафедры ИСТ	Мальчуков Андрей Николаевич	к.т.н., доцент		19.09.2016

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8В3А	Старшинов Владислав Сергеевич		19.09.2016

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Институт Кибернетики
Направление подготовки 09.03.01 Информатика и вычислительная техника
Уровень образования Бакалавриат
Кафедра Информационных систем и технологий
Период выполнения осенний / весенний семестр 2016/2017 учебного года
Форма представления работы:

Бакалаврская работа

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Срок сдачи студентом выполненной работы:	31.05.2017

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
21.02.2017	Проектирование устройства межинтерфейсного взаимодействия AXI-to-SPI	15
25.04.2017	Разработка устройства межинтерфейсного взаимодействия AXI-to-SPI	25
27.05.2017	Тестирование и апробация устройства межинтерфейсного взаимодействия AXI-to-SPI	25
22.05.2017	Проектирование печатной платы, компоновка и трассировка элементов на печатной плате	15
29.05.2017	Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	10
29.05.2017	Социальная ответственность	10

Составил преподаватель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент кафедры ИСТ	Мальчуков Андрей Николаевич	к.т.н., доцент		19.09.2016

СОГЛАСОВАНО:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Зав. кафедрой ИСТ	Мальчуков Андрей Николаевич	к.т.н., доцент		19.09.2016

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студенту:

Группа	ФИО
8В3А	Старшинову Владиславу Сергеевичу

Институт	Кибернетики	**Кафедра**	Информационных систем и технологий
Уровень образования	Бакалавр	**Направление/специальность**	09.03.01 «Информатика и вычислительная техника»

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
1. *Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих*	*Энергетические (стоимость в рублях на 1 кВт*ч для юрид. Лиц), информационные (час работы в интернете) и человеческие (согласно окладам научного руководителя и инженера-программиста)*
2. *Нормы и нормативы расходования ресурсов*	*Нормы из реальных осуществляемых затрат: потребление технических ресурсов, норма потребления электроэнергии согласно паспорту*
3. *Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования*	*Для юридических лиц в области образования социальные отчисления – 27,1%*
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. *Оценка коммерческого потенциала, перспективности и альтернатив проведения НИ с позиции ресурсоэффективности и ресурсосбережения*	*Оценка ресурсной, социальной эффективности НИ и потенциальных рисков.*
2. *Планирование и формирование бюджета научных исследований*	*На основании информации, представленной в научных статьях и публикациях, аналитических материалах, статистических бюллетенях и изданиях, нормативно-правовых документах, определить методику расчета экономической эффективности.*
3. *Определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования*	
Перечень графического материала *(с точным указанием обязательных чертежей)***:**	
Календарный план для выполнения научно-исследовательского проекта	

Дата выдачи задания для раздела по линейному графику	25.04.2017

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент кафедры менеджмента	Антонова Ирина Сергеевна	к.э.н.		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8В3А	Старшинов Владислав Сергеевич		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студенту:

Группа	ФИО
8В3А	Старшинову Владиславу Сергеевичу

Институт	Кибернетики	**Кафедра**	Информационных систем и технологий
Уровень образования	Бакалавр	**Направление/специальность**	09.03.01 «Информатика и вычислительная техника»

Исходные данные к разделу «Социальная ответственность»:	
1. *Описание рабочего места (рабочей зоны, технологического процесса, механического оборудования) на предмет возникновения:* – *вредных проявлений факторов производственной среды (метеоусловия, вредные вещества, освещение, шумы, вибрации, электромагнитные поля, ионизирующие излучения)* – *опасных проявлений факторов производственной среды (механической природы, термического характера, электрической, пожарной и взрывной природы)* – *негативного воздействия на окружающую природную среду (атмосферу, гидросферу, литосферу)* – *чрезвычайных ситуаций (техногенного, стихийного, экологического и социального характера)*	*Анализ и выявление вредных производственных факторов рабочей среды, а именно: электромагнитное излучение, микроклимат, освещение, шумы и прочие, влияющие на организм человека при разработке программного обеспечения в помещении учебной аудитории.* *Анализ и выявление опасных производственных факторов проектируемой среды, а именно: электробезопасность и пожаробезопасность.* *Утилизация люминесцентных ламп – основной источник загрязнения литосферы.* *Чрезвычайная ситуация техногенного характера для данного помещения – пожар.* *В качестве исходных данных использованы параметры рабочего помещения, в котором производилась разработка и условия труда при работе с персональным компьютером.*
2. *Знакомство и отбор законодательных и нормативных документов по теме*	*Выбор подходящих нормативов и документов (СанПиН 2.2.2/2.4.1340-03, СНиП 23-05-95, ГОСТ 6825-91, ГОСТ 12.1.003-83, СНиП 23-03-2003, СанПиН 2.2.4.548-96, ГОСТ 12.0.005-74, ГОСТ Р 51768-2001, 51057-01, ГОСТ 12.10.019 (с изм. №1)), для обеспечения соответствия условий труда Трудовому кодексу РФ.*
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. *Анализ выявленных вредных факторов проектируемой производственной среды в следующей последовательности:* – *физико-химическая природа вредности, её связь с разрабатываемой темой;* – *действие фактора на организм человека;* – *приведение допустимых норм с необходимой размерностью (со ссылкой на соответствующий нормативно-технический документ);* – *предлагаемые средства защиты (сначала коллективной защиты, затем – индивидуальные защитные средства)*	*Анализ выявленных вредных факторов труда разработчика-программиста: недостаточная освещенность рабочей зоны; отклонение параметров микроклимата в помещении; повышенный уровень шума; повышенный уровень излучения электромагнитных полей.*
2. *Анализ выявленных опасных факторов проектируемой произведённой среды в следующей последовательности* – *механические опасности (источники, средства*	*Анализ выявленных опасных производственных факторов рабочей среды, влияющих на организм человека при*

защиты; – *термические опасности (источники, средства защиты);* – *электробезопасность (в т.ч. статическое электричество, молниезащита – источники, средства защиты);* – *пожаровзрывобезопасность (причины, профилактические мероприятия, первичные средства пожаротушения)*	*разработке программного обеспечения в рабочем помещении учебной аудитории, а именно: опасность поражения электрическим током, опасность поражения статическим электричеством и пожароопасность.*
3. Охрана окружающей среды: – *защита селитебной зоны* – *анализ воздействия объекта на атмосферу (выбросы);* – *анализ воздействия объекта на гидросферу (сбросы);* – *анализ воздействия объекта на литосферу (отходы);* – *разработать решения по обеспечению экологической безопасности со ссылками на НТД по охране окружающей среды.*	*Утилизация используемой орг.техники и люминесцентных ламп.*
4. Защита в чрезвычайных ситуациях: – *перечень возможных ЧС на объекте;* – *выбор наиболее типичной ЧС;* – *разработка превентивных мер по предупреждению ЧС;* – *разработка мер по повышению устойчивости объекта к данной ЧС;* – *разработка действий в результате возникшей ЧС и мер по ликвидации её последствий*	*Чрезвычайная ситуация техногенного характера для данного помещения – пожар. Установка общих правил поведения и рекомендаций во время пожара, план эвакуации.*
5. Правовые и организационные вопросы обеспечения безопасности: – *специальные (характерные для проектируемой рабочей зоны) правовые нормы трудового законодательства;* – *организационные мероприятия при компоновке рабочей зоны*	*Основные проводимые правовые и организационные мероприятия по обеспечению безопасности трудящихся в учебных аудиториях.*
Перечень графического материала:	
При необходимости представить эскизные графические материалы к расчётному заданию (обязательно для специалистов и магистров)	*План эвакуации при пожаре* *Организация рабочего места*

Дата выдачи задания для раздела по линейному графику	25.04.2017

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Инженер	Маланова Наталья Владимировна	к.т.н		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8В3А	Старшинов Владислав Сергеевич		

РЕФЕРАТ

Пояснительная записка 153 страницы, 70 рисунков, 40 таблиц, 42 источника, 23 приложения.

Ключевые слова: ЦАП, межинтерфейсный адаптер, транзакции, AXI4-Lite, SPI, программирование блоков, натурное моделирование, САПР.

Объектом исследования является разработка межинтерфейсного взаимодействия между интерфейсами AXI и SPI на ПЛИС.

Цель работы – анализ алгоритмов работы блоков последовательного интерфейса SPI, параллельного интерфейса AXI4-Lite и разработка межинтерфейсного адаптера между интерфейсами AXI и SPI на ПЛИС.

В результате исследования изучена документация по работе последовательного интерфейса SPI, параллельного интерфейса AXI4-Lite и ЦАП MCP4922 и разработаны граф-схемы алгоритмов автомата работы блоков данных интерфейсов. В итоге были спроектированы 2 варианта исполнения устройства, один из которых был протестирован и апробирован в качестве генератора синусоидальных сигналов.

Основные конструктивные, технологические и технико-эксплуатационные характеристики: использование языка описания аппаратуры VHDL.

Результаты исследования и апробация: разработаны 2 исполнения устройства и была проведена апробация одного из исполнений в качестве генератора синусоидального сигнала.

Область применения: автоматизированная система управления технологическим процессом, системы передачи данных.

Значимость работы заключается в проектировании межинтерфейсного адаптера, который позволит передавать данные с ПК и преобразовать их в синусоиду с необходимой амплитудой, используя микросхему с ПЛИС и ЦАП, а также различные IP-ядра для интеграции в проект.

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ, СОКРАЩЕНИЯ И НОРМАТИВНЫЕ ССЫЛКИ

ПЛИС - программируемая логическая интегральная схема

ЦАП – цифро-аналоговый преобразователь

ПЭС – принципиальная электрическая схема

ГСА – граф-схема алгоритма

УГО – условно-графическое обозначение

ПЭС – принципиальная электрическая схема

САПР – система автоматизированного проектирования

SPI – serial peripheral interface

FPGA – field-programmable gate array

CPLD – complex programmable logic device

ОБЪЕКТ И МЕТОД ИССЛЕДОВАНИЯ

Объектом исследования является создание межинтерфейсного адаптера между интерфейсами AXI и SPI на ПЛИС.

Методом исследования является имитационное моделирование на специализированных САПР, натурное моделирование с использованием опытного образца, проектирование цифровых устройств в САПР.

Оглавление

ВВЕДЕНИЕ

В настоящее время остро стоит вопрос о передачи данных между различными устройствами, но проблема состоит в том, что используемые устройства подключаются по различным интерфейсам. Поэтому для выполнения данных задач следует использовать межинтерфейсные адаптеры для передачи необходимых данных.

На данный момент имеется несколько реализаций устройств межинтерфейсных взаимодействий. Они имеют различное быстродействие и требование к аппаратным ресурсам. Но к сожалению, не все реализации имеют открытый доступ, поэтому перед разработчиком стоит задача в выборе интерфейсов и создания адаптера, позволяющим передавать верные данные.

Целью работы является реализация межинтерфейсного адаптера AXI-to-SPI.

Задачи:

1. Изучение работы последовательных и параллельных интерфейсов;
2. Выбор процедуры для аппаратной реализации межинтерфейсного адаптера AXI-to-SPI на ПЛИС;
3. Реализация межинтерфейсного адаптера AXI-to-SPI;
4. Тестирование разработанного межинтерфейсного адаптера AXI-to-SPI на временных диаграммах;
5. Разработка системы апробации устройства и структурно-функциональной схемы для разрабатываемой системы;
6. Выбор компонентов для проведения апробации разработанного устройства;
7. Подключение soft-процессорного ядра Microblaze с последующим созданием IP-core для второго исполнения данного устройства;
8. Написать программу для генерации синусоиды для Microblaze;
9. Проведение апробации устройства на выбранных компонентах;
10. Проектирование устройства на печатной плате с созданием принципиальной схемы устройства, компоновкой и трассировкой элементов.

1. ОБЗОР СОВРЕМЕННЫХ МЕТОДОВ ПРОЕКТИРОВАНИЯ И РАЗРАБОТКИ ЦИФРОВЫХ УСТРОЙСТВ

1.1. Постановка задачи

Глобальная задача данной работы заключается в передаче в эфир аналогового сигнала определенной формы и определенной частоты с программным управлением при помощи терминала на ПК.

В ходе изучения данной задачи была выявлена основная проблема: нет возможности соединения терминала на ПК и передатчика, поэтому было принято решение о реализации устройства взаимодействия между ПК и передатчиком.

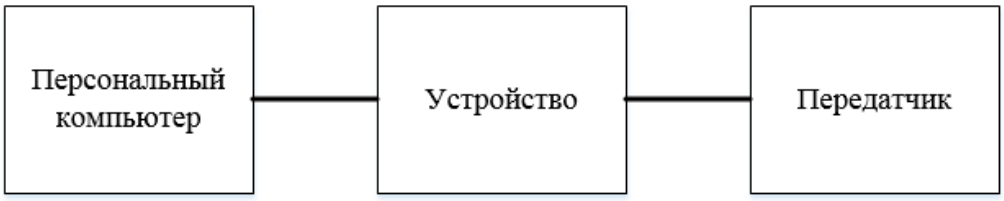


Рисунок 1 – Общая структура взаимодействия разрабатываемой системы

На ПК установлен терминал, в который оператор будет записывать данные и адрес, в который будут эти данные записываться.

Передатчик будет считывать данные с устройства и передавать преобразованный сигнал в эфир.

Для передачи данных с ПК на устройство будет использоваться протокол интерфейса AXI. Выбор был сделан на основании того, что данный интерфейс универсален, имеет открытую документацию, имеет высокую скорость передачи данных и небольшие задержки. Для передачи данных с устройства на передатчик будет использоваться протокол интерфейса SPI. Выбор был сделан на основании того, что данный интерфейс имеет высокую пропускную способность по сравнению с другими последовательными интерфейсами и имеет открытую документацию.

Устройство должно обеспечить преобразование параллельных сигналов в последовательные. Для этого между данными интерфейсами необходимо реализовать адаптер, соединяющий параллельный и последовательный протоколы интерфейсов.

Стоит упомянуть, что данные будут поступать в устройство с тактовой частотой 50 МГц, а с устройства на передатчик – с частотой 5 МГц. Данное преобразование позволит обеспечить необходимую производительность модуля и будет реализовано с помощью делителя частоты.

1.2. Обзор существующих решений

При анализе поставленных задач следует задаться выбором решения для их реализации. Выбор стоит осуществлять между ПЛИС и микроконтроллером.

Учитывая, что функционал ПЛИС не уступает микроконтроллеру, кроме того ПЛИС прошивается на уровне железе, практически по всей площади кристалла, а микроконтроллер прошивается на уровне программы для железа (сигналы проходят группами, от блока к блоку, от памяти к процессору, к оперативной памяти, от оперативной памяти к процессору, от процессора к портам ввода-вывода и т.д), ПЛИС выигрывает в быстродействии за счет своей архитектуры и более широких возможностях конвейерной обработки. Но при этом микроконтроллер выигрывает в простоте написания алгоритмов за счет того, что в ПЛИС разработчику приходится выполнять всю работу вручную. В итоге, в качестве устройства будет использоваться ПЛИС [1].

ПЛИС – электронный компонент, используемый для создания цифровых интегральных схем.

Логика работы ПЛИС определяется не на фабрике изготовителем микросхемы, а путем дополнительного программирования с помощью специальных средств: программаторов и программного обеспечения [2].

Внутри ПЛИС находятся некие базовые элементы, которые соединяются на основе конфигурационной записи. Возможные базовые элементы, вид и место хранения конфигурационной записи зависят от вида ПЛИС и от конкретной модели [3].

В ПЛИС выделяют два вида: CPLD и FPGA.

CPLD (сложные программируемые логические устройства) – ПЛИС, базовыми элементами которой являются макроячейки и простые логические

вентили (И(-НЕ)/ИЛИ(-НЕ)). Обычно содержит меньше базовых элементов, чем FPGA, но является более быстродействующей. Также обычно содержит энергонезависимую конфигурационную память прямо на кристалле, но имеет ограниченное число циклов конфигурирования. Также этот вид ПЛИС сейчас меньше применяется в промышленности, в отличие от FPGA.

FPGA (программируемая пользователем вентильная матрица) – ПЛИС, которые обычно имеют целый букет видов базовых блоков, это и настраиваемые логические элементы (таблицами истинности) и блоки сложения-умножения (Digital signal processing – DSP) и PLL (Phase-Locked Loop) для деления и умножения частоты и некоторые другие в зависимости от модели. Обычно имеют энергозависимую внутреннюю память и функционал для загрузки конфигурации с внешней энергонезависимой памяти [4].

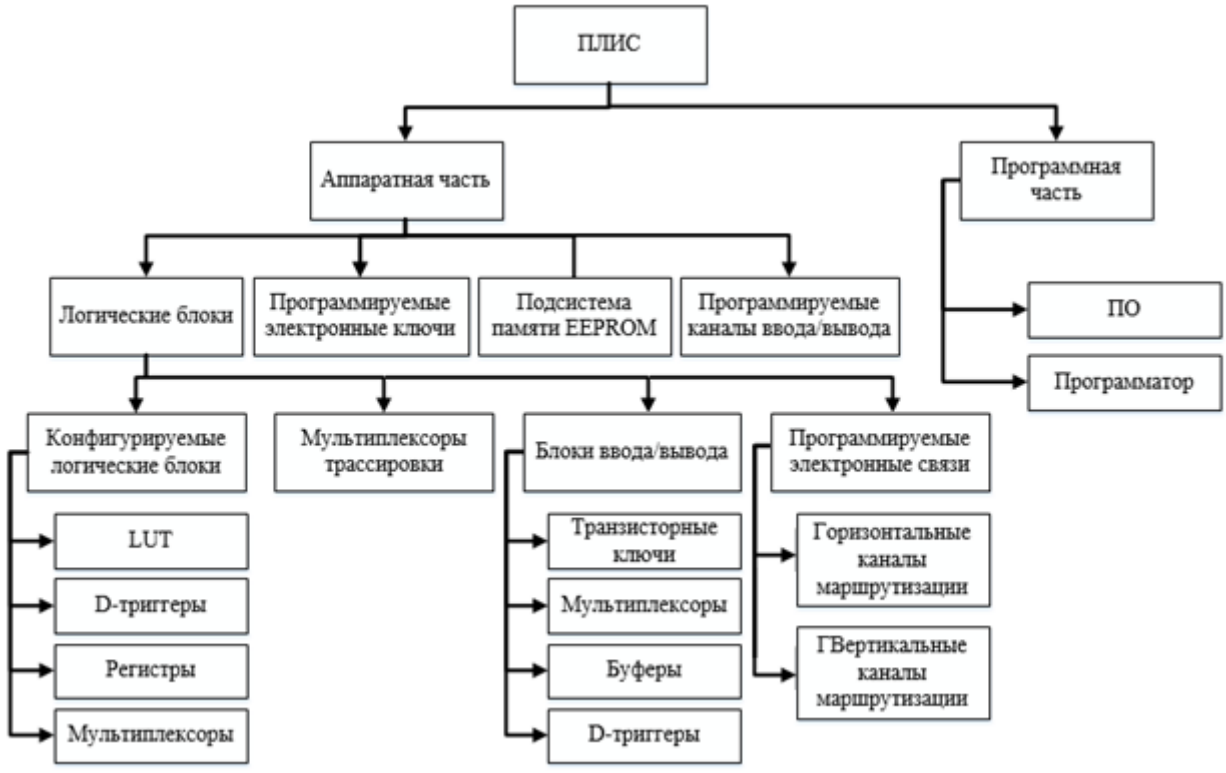


Рисунок 2 – Иерархия состава ПЛИС

Обычно, сама микросхема ПЛИС состоит из:

- конфигурируемых логических блоков, реализующих требуемую логическую функцию;

- программируемых электронных связей между конфигурируемыми логическими блоками;
- программируемых блоков ввода/вывода, обеспечивающих связь внешнего вывода микросхемы с внутренней логикой [2].

Рисунок 3 – ПЛИС Xilinx

Для программирования используются программатор и IDE (отладочная среда), позволяющие задать желаемую структуру цифрового устройства в виде принципиальной электрической схемы или программы на специальных языках описания аппаратуры [5].

1.3. Языки описания аппаратуры

Программирование на ПЛИС осуществляется с помощью языков описания аппаратуры. Существуют много языков описания аппаратуры, но на данный момент основными языками являются SystemVerilog и VHDL.

На верхнем уровне эти языки очень схожи – модель аппаратуры описывается в виде взаимодействующих блоков (модулей) и для каждого из них определяется интерфейс и реализация. Интерфейсы модулей описывают входные, выходные и двусторонние порты, благодаря которым модули соединяются друг с другом с целью обмена данными, а также управляющими сигналами. Реализация задает элементы внутреннего состояния и порядок вычисления значений выходных интерфейсов на основе этого состояния и значений входных портов, а также правила обновления внутреннего состояния [6].

При рассмотрении данных языков описания аппаратуры с точки зрения разработчика можно выявить следующие отличия:

Таблица 1 – Сравнение языков VHDL и SVerilog

	VHDL	SVerilog
Структурная организация	Интерфейс и архитектура, раздел описания и исполняемый раздел не пересекаются	Интегрированное тело модуля, декларации и параллельные операторы размещаются в модуле в произвольном порядке
Типы данных	Язык строгой типизации	Данные разных типов можно совмещать в одном выражении
Задержки	Минимальные, типичные и максимальные задержки нельзя описывать одновременно	Позволяет описывать все виды задержек
Создание пользовательских типов данных	Пользователь может вводить свои типы данных	Язык не позволяет вводить новые типы данных
Описание процесса (языковые особенности)	process	always
Основа языка	Ada	C
Кодирование	Громоздкий, но структурированный	Лаконичный, но смешанный в структуре

Несмотря на приведенные особенности в сравнении этих языков у каждого из них имеются свои преимущества и недостатки, поэтому нельзя склоняться к категоричному выбору одного из них, т.к. для разных задач рациональнее использовать тот язык описания аппаратуры, который будет более эффективным [7].

1.4. Среды разработки

На сегодняшний день существуют немного различных производителей микросхем, которые выпускают ПЛИС, такие как Xilinx, Altera, Microsemi и др. У каждой фирмы свои названия выпускаемых устройств и свои собственные системы проектирования, которые позволяют проектировать весь спектр цифровых устройств типа FPGA/CPLD. Основные отличия производителей устройств ПЛИС друг от друга заключается в архитектуре построения внутренних

программируемых комбинационных схем, способом загрузки программирования ПЛИС, емкостью логических элементов, числом эквивалентных вентилей, технологии изготовления кристаллов, различные типы корпусов ПЛИС и т. Д [4].

Каждая САПР может проектировать цифровые устройства на ПЛИС только под чипы своего производителя Quartus для Altera, ISE Design Suite и Vivado для Xilinx. Ввиду того, что в данной работе будет использоваться ПЛИС Spartan 3E от производителя Xilinx, следует рассмотреть САПР Xilinx Vivado Design Suite.

Интерфейс программы Xilinx Vivado представлен на рисунке 4.

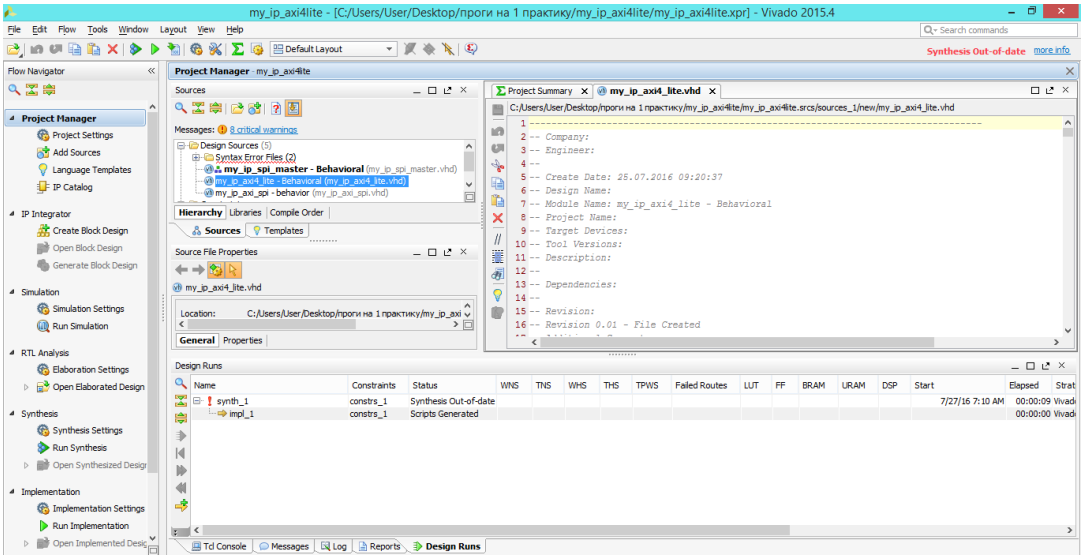


Рисунок 4 – Окно программы Xilinx Vivado

Для своевременного обнаружения возможных ошибок средства моделирования должны обеспечивать возможность контроля результатов каждого этапа процесса проектирования: создания исходных HDL-описаний, синтеза, размещения и трассировки в кристалл. Такой подход обеспечивает минимальное время разработки устройства и сокращает стоимость этого процесса, так как цена ошибки возрастает с каждым последующим шагом проектирования.

Среда моделирования ModelSim предназначена для проверки работоспособности проекта, описанного на одном из языков описания аппаратуры (HDL). Она включает в себя средства создания проекта, создания и редактирования исходных файлов проекта, компилятор, моделирующую программу и средства визуализации результатов моделирования (графический редактор и пр.). ModelSim поддерживает работу с тестовыми файлами на HDL

(testbench) или на языке TCL (командный язык для создания управляющих файлов). При этом среда универсальна и не привязана к конкретному производителю ПЛИС [10].

Интерфейс программы представлен на рисунке 5.

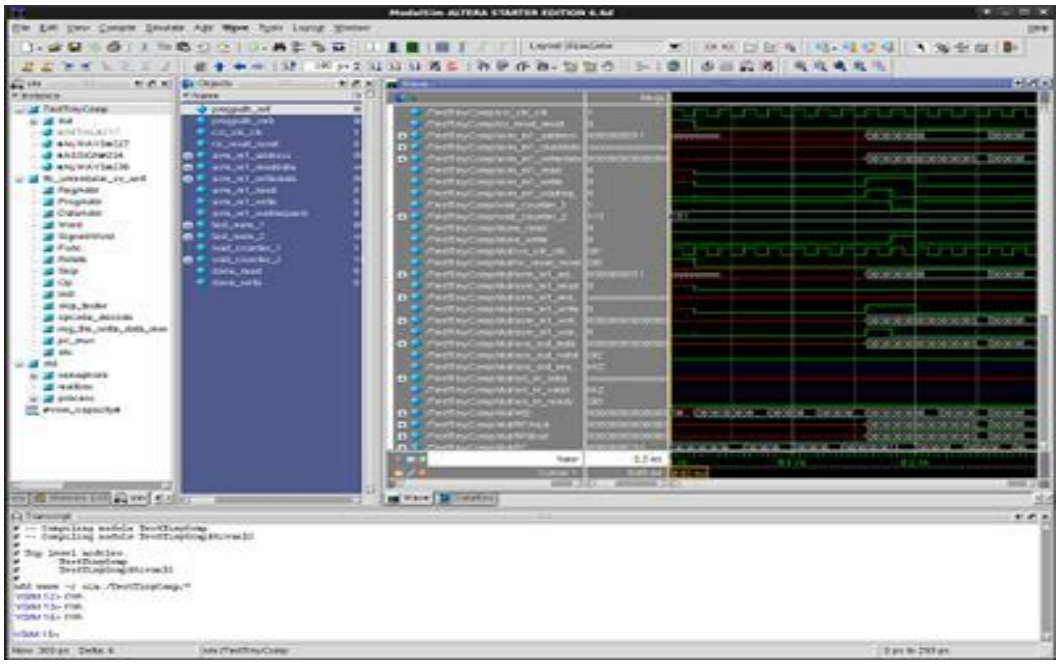


Рисунок 5 – Окно программы Modelsim 10.2c

1.5. Описание используемых интерфейсов

Для реализации поставленной задачи используются последовательный интерфейс SPI и параллельный интерфейс AXI4-Lite. Прежде чем переходить к программированию данных интерфейсов, необходимо произвести описание данных интерфейсов.

1.5.1 AXI4-Lite

Advanced Microcontroller Bus Architecture (AMBA) является параллельной шиной, предназначенной для объединения модулей в системе на кристалле (рис.6)

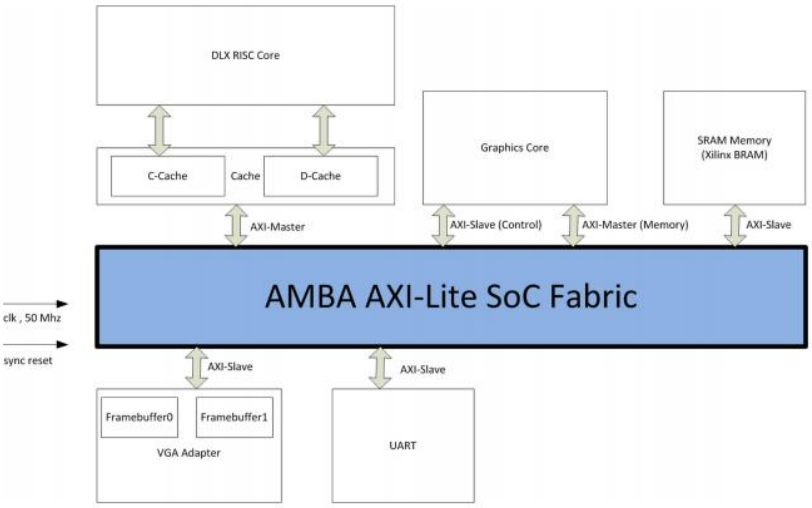


Рисунок 6 – Пример использования шины AMBA

AXI является частью ARM AMBA, семейства шин микроконтроллера и впервые введена в 1996 году. Первая версия AXI была впервые включена в AMBA 3.0, выпущенной в 2003 году, а AMBA 4.0, выпущенная уже в 2010 году, включает в себя вторую версию AXI, AXI4.

Преимущества шины AXI4:

- продуктивность – для стандартизации интерфейса AXI, разработчикам необходимо знать только обычный протокол для IP;
- гибкость – для оптимизации работы при передаче пакетов данных;
- доступность – при переходе на промышленный стандарт имеется доступ не только к каталогам IP Core, но и всемирному сообществу «ARM Partners» [11].

Имеется 3 типа AXI4 интерфейсов:

- AXI4 - для задач, где требуется высокая производительность;
- AXI4-Lite - для задач, где не требуется высокая пропускная способность памяти;
- AXI4-Stream - для потоковой передачи данных с высокой скоростью.

Основные отличия AXI4-Lite от полной версии следующие:

- Не поддерживается запись векторов данных. Перед каждой порцией данных на шину должен быть выставлен соответствующий адрес.
- Используется вся ширина шины данных. AXI4-Lite поддерживает шины данных размерностью 32 и 64 бита.
- Не поддерживается исключительный доступ к шине.

AXI4-Lite включает в себя 5 разных каналов:

- канал адреса чтения;
- канал адреса записи;
- канал чтения данных;
- канал записи данных;
- канал статуса записи операции.

Для передачи данных обычно используются 2 устройства: ведущее (master) и ведомое (slave). Данные могут передаваться в обоих направлениях между ведущим и ведомым устройствами одновременно. В AXI4-Lite позволяется передать 1 пакет данных за транзакцию [12].

На рисунке 7 показано, как совершается транзакция чтения AXI4 при использовании каналов адреса чтения и чтения данных между ведущим и ведомым устройствами.

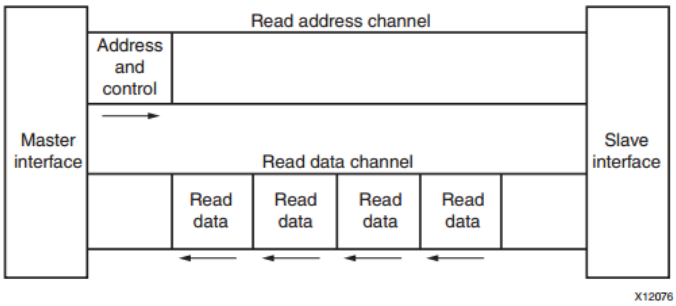


Рисунок 7 – Структура транзакции чтения по интерфейсу AXI

На рисунке 8 показано, как совершается транзакция записи AXI4 при использовании каналов адреса записи, записи данных и статуса записи операции между ведущим и ведомым устройствами.

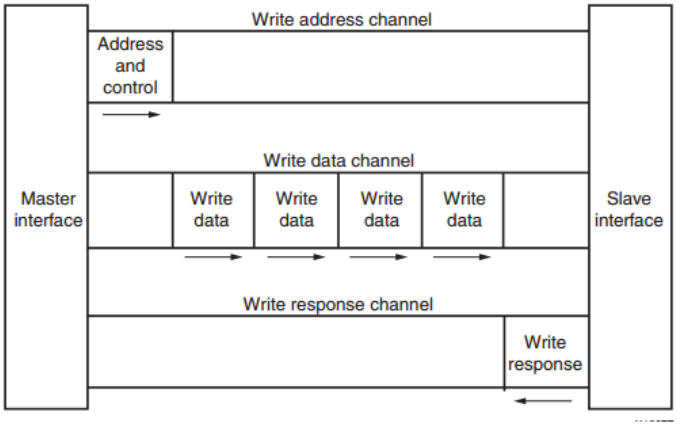


Рисунок 8 – Структура транзакции записи по интерфейсу AXI

Как показано на предыдущих рисунках, AXI4 предоставляет отдельные каналы данных и адреса для операций чтения, и записи для одновременной двунаправленной передачи данных. AXI4 требуется один адрес, который позволяет передать до 256 слов данных [12,13].

1.5.2 Serial Peripheral Interface (SPI)

SPI (Serial Peripheral Interface) – последовательный периферийный интерфейс, разработанный компанией Motorola для организации быстрого и простого в реализации обмена данными между компонентами системы – микроконтроллерами и периферийными устройствами.

В отличие от стандартного последовательного порта, SPI является синхронным интерфейсом, в котором любая передача синхронизирована с общим тактовым сигналом, генерируемым ведущим устройством (процессором). Принимающая (ведомая) периферия синхронизирует получение битовой последовательности с тактовым сигналом. К одному последовательному периферийному интерфейсу ведущего устройства-микросхемы может присоединяться несколько микросхем. Ведущее устройство выбирает ведомое для передачи, активируя сигнал «выбор кристалла» на ведомой микросхеме. Периферия, не выбранная процессором, не принимает участия в передаче по SPI [15].

На шине может быть одно ведущее устройство (master) и одно/несколько ведомых (slave).

Преимущества использования SPI:

- полнодуплексная передача данных по умолчанию;
- более высокая пропускная способность по сравнению с I2C;
- возможность произвольного выбора длины пакетов, длина пакетов не ограничена 8 битами;
- ведомым устройствам не нужен уникальный адрес.

Передача осуществляется пакетами. Длина пакета, как правило, составляет 1 байт (8 бит), при этом известны реализации SPI с иной длиной пакета, например,

4 бита. Ведущее устройство инициирует цикл связи установкой низкого уровня на выводе выбора подчиненного устройства того устройства, с которым необходимо установить соединение.

Подлежащие передаче данные ведущее и ведомое устройства помещают в сдвиговые регистры (рис. 9). После этого ведущее устройство начинает генерировать импульсы синхронизации на линии SCLK, что приводит к взаимному обмену данными. Передача данных осуществляется бит за битом от ведущего по линии MOSI и от ведомого по линии MISO. Передача осуществляется, как правило, начиная со старших битов, но некоторые производители допускают изменение порядка передачи битов программными методами. После передачи каждого пакета данных ведущее устройство, в целях синхронизации ведомого устройства, может перевести линию SS в высокое состояние [15].

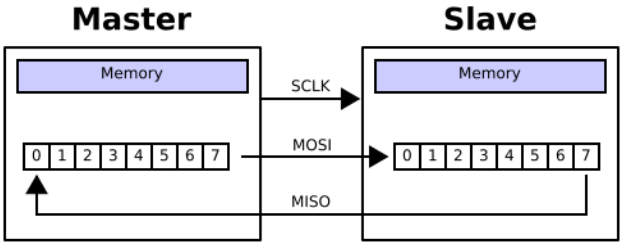


Рисунок 9 – Типичная структура транзакций по интерфейсу SPI

Обычно в SPI используется четырехпроводная схема подключения, но в простейшем случае к ведущему устройству подключено единственное ведомое устройство и необходим двусторонний обмен данными. В таком случае используется трехпроводная схема подключения [13, 15].

2. РАЗРАБОТКА УСТРОЙСТВА МЕЖИНТЕРФЕЙСНОГО ВЗАИМОДЕЙСТВИЯ AXI-TO-SPI НА ПЛИС

2.1. Структурно-функциональная схема первого исполнения устройства

Устройство межинтерфейсного взаимодействия AXI-to-SPI позволяет преобразовывать данные в параллельном виде в данные в последовательном виде. Данный модуль позволяет преобразовать данные, которые задает на ПК оператор, и передать на антенну сигнал определенной формы с необходимой частотой, используя микросхему с ПЛИС и ЦАП.

Структурно-функциональную схему устройства можно описать с помощью блоков: AXI_SLAVE, AXI_TO_SPI, SPI_MASTER, DLL/10. На рисунке 10 представлена структурно-функциональная схема первого исполнения устройства межинтерфейсного взаимодействия AXI-to-SPI.

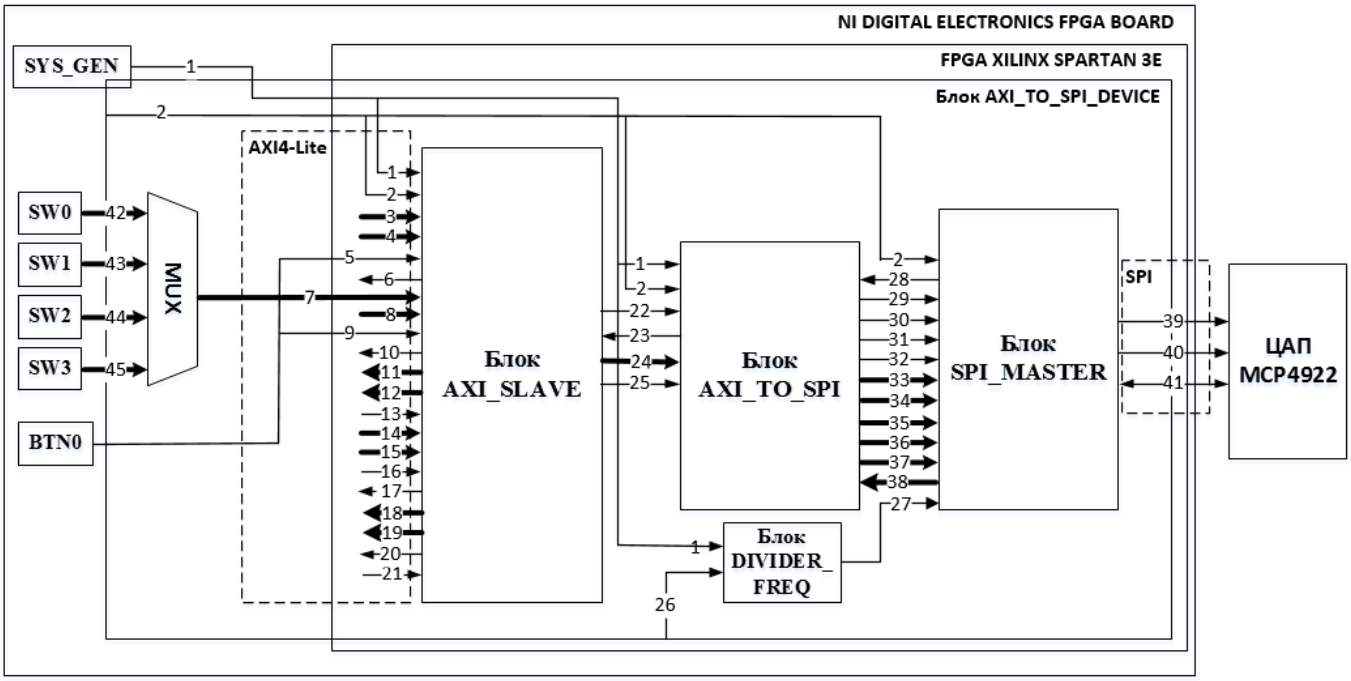


Рисунок 10 – Структурно-функциональная схема первого исполнения устройства

Связями между блоками являются следующие данные и управляющие сигналы:

1: S_AXI_ACLK – глобальный сигнал тактирования

2: S_AXI_RESETN – глобальный сигнал сброса

3: S_AXI_ AWADDR [31..0] – адрес записи

4: S_AXI_ AWPROT [2..0] – тип привилегий

5: S_AXI_AWVALID – актуальность адреса

6: S_AXI_AWREADY – готовность принять адрес

7: S_AXI_WDATA [31..0] – данные для записи

8: S_AXI_WSTRB [2..0] – строб данных

9: S_AXI_WVALID – актуальность данных

10: S_AXI_WREADY – готовность принять данные

11: S_AXI_BRESP [1..0] – статус произведенной операции записи

12: S_AXI_BVALID – актуальность данных при записи

13: S_AXI_BREADY – готовность принять статус записи

14: S_AXI_ARADDR [31..0] – адрес чтения данных

15: S_AXI_ARPROT [2..0] – тип привилегий для чтения

16: S_AXI_ARVALID – актуальность считываемого адреса

17: S_AXI_ARREADY – готовность считать данные

18: S_AXI_RDATA [31..0] – считанные данные

19: S_AXI_RRESP [1..0] – статус произведенной операции чтения

20: S_AXI_RVALID – актуальность считываемых данных

21: S_AXI_RREADY – готовность принять статус чтения

22: O_AXI_BUSY – сигнал занятости линии передачи

23: O_AXI_ENA – сигнал разрешения перехода в следующий блок

24: O_AXI_RX_DATA [31..0] – данные, передаваемые на блок AXI_TO_SPI

25: O_AXI_READY – завершение передачи данных

26: RESET – глобальный сброс блока DIVIDER_FREQ

27: CLK_OUT – выход тактовой частоты 5МГц

28: M_SPI_ENABLE – разрешение на передачу данных

29: M_SPI_BUSY – занятость линии передачи SPI

30: M_SPI_ADDR – установка адреса

31: M_SPI_RW1 – 1 команда чтения/записи

32: M_SPI_RW2 – 2 команда чтения/записи

33: M_SPI_CLK_DIV [31..0] – установка скорости

34: M_SPI_ TX_CMD1 [cmd_width-2..0] – 1 порция передаваемых команд

35: M_SPI_ TX_CMD2 [cmd_width-2..0] – 2 порция передаваемых команд

36: M_SPI_ TX_DATA1 [d_width-1..0] – 1 порция передаваемых данных

37: M_SPI_ TX_DATA2 [d_width-1..0] – 2 порция передаваемых данных

38: M_SPI_ RX_DATA [d_width-1..0] – Считываемые данные

39: M_SPI_SCLK – тактовая частота передаваемых сигналов 5МГц

40: M_SPI_SS_N – выбор ведомого устройства

41: M_SPI_SDIO – последовательная выдача команд и данных

42: DATA1 – 1 тестовый набор данных

43: DATA2 – 2 тестовый набор данных

44: DATA3 – 3 тестовый набор данных

45: DATA4 – 4 тестовый набор данных

В качестве основной задачи работы выступает передача аналогового сигнала в эфир, используя разрабатываемое устройство. Как было отмечено ранее, проблема заключается в прямом подключении ПК и передатчика, который будет выдавать аналоговый сигнал.

С помощью программного средства Xilinx Vivado и программатора данные будут передаваться на отладочную плату NI Digital Electronics FPGA Board через USB. Транслирование команд на ПЛИС осуществляется с помощью универсального интерфейса AXI4-Lite, к которому можно подключить разрабатываемое устройство.

Блок AXI_SLAVE принимает команды и адрес и с помощью многочисленных настроек осуществляет взаимодействие с ПК через ПЛИС на отладочной плате. Как видно из структурно-функциональной схемы, имеется 4 набора данных, которые выбираются из 4 движковых переключателей. В зависимости от выбранного переключателя происходит пересылка данных на AXI_SLAVE через мультиплексор по коду движкового переключателя. По кнопке BTN0 происходит передача сигнала готовности AXI_SLAVE принять данные. Данный блок является ведомым устройством, который принимает данные от ведущего устройства, которым выступает в данной ситуации ПК. Стоит уточнить, что частота приходящих команд на данный блок равно 50 МГц от SYS_GEN [12].

Для того, чтобы передать сигнал на антенну или осциллограф, используется передатчик. Для передатчика главным условием является преобразование сигнала с цифрового вида в аналоговый. Соответственно, нужен цифро-аналоговый преобразователь (ЦАП), поэтому используется блок ЦАП, в роли конкретного устройства ЦАП выступает MCP4922. В руководстве пользователя имеется карта регистров и интерфейс управления SPI с необходимыми временными диаграммами, которые и являются нашей целью. Учитывая, что шина AXI4-Lite является параллельной, а SPI является последовательной, стоит задаться вопросом преобразования параллельных данных в последовательные. Для этого необходимо создать межинтерфейсное взаимодействие между интерфейсом, по которому

передаются адрес и данные с ПК и интерфейсом, по которому передаются данные в следующий блок [15, 17].

Данные передаются с блока AXI_SLAVE на блок AXI_TO_SPI, являющимся межинтерфейсным взаимодействием и разбивает данные на инструкции и команды. В качестве инструкций выступают адреса регистров, в которые необходимо записать данные. Приходящие 32-битные команды разбиваются на 4 порции данных, в каждой из которых по 8 бит данных (по 1 байту). После всех необходимых преобразований байт данных передается на блок SPI_MASTER. После передачи байта данных освобождается флаг занятости BUSY и поступает флаг разрешения операции передачи ENABLE. По установлению данных сигналов происходит передача следующей порции данных (байта).

После передачи данных на блок SPI_MASTER они записываются в сдвиговый регистр, после чего выдача данных побитово. В качестве последовательности выдаваемых данных выступает бит инструкции и бит данных. Эта последовательность данных передается на блок передатчика ЦАП.

Для обеспечения требуемой производительности используется блок DLL/10, который делит частоту приходящих команд с ПК в параллельном виде в 10 раз, поэтому на блок AXI_SLAVE частота приходящих команд равна 50 МГц, а на блок SPI_MASTER частота приходящих сигналов равна 5 МГц.

После передачи данных на ЦАП данные, заданные в виде последовательности инструкций и команд (в цифровом виде) преобразуются в синусоидальный сигнал (аналоговый вид) и передается в эфир.

Данное устройство выступает в роли радиопередатчика аналогового сигнала на антенну. Апробация устройства будет проведена в качестве генератора синусоидального сигнала (приложение 3).

Для взаимодействия программы и микросхемы необходим базовый модуль – плата с ПЛИС, где имеется дополнительная плата с микросхемой ЦАП. Ножки микросхемы ЦАП физически подсоединяются к микросхеме ПЛИС на макетной плате, а подключение осуществляется с помощью щупов.

2.2. Функциональная схема первого исполнения устройства

Для разработки устройства необходимо для начала создать модуль межинтерфейсного взаимодействия. После компиляции написанных модулей создается функциональная схема устройства (приложение 4). Для написания модулей используется программное обеспечение Xilinx Vivado, которое поддерживает использование ПЛИС фирмы Xilinx. В данном программном обеспечении сделан упор в сторону написания исходного кода модулей и их последующее соединение в основном модуле, в программном обеспечении Quartus 11 обычно создаются в блоки, в которых записывается входные и выходные сигналы и реализуется автомат процессов. Эти входные и выходные сигналы соединяют между собой блоки и разработчик может расположить блоки в удобном ему порядке. В Xilinx Vivado данная функциональная схема была спроектирована программой после компиляции программ модулей.

Как видно из функциональной схемы, представленной выше, устройство состоит из 4 блоков, каждый из которых выполняет свою функцию.

Блок AXI_SLAVE принимает данные и адрес, на который записываются эти данные, от ПК, на котором оператор записывает эти данные и адрес в ПЛИС через программу-терминал по протоколу интерфейса AXI4-Lite. Осуществляемые между ними транзакции наглядно показывают роль данных устройств, в котором ПК является ведущим устройством, а блок AXI_SLAVE – ведомым устройством.

Блок AXI_TO_SPI выполняет роль преобразователя данных из параллельного вида в последовательный. Данный блок разделяет подаваемое с блока AXI_SLAVE 32-разрядное слово на 4 порции данных по 8 бит каждая, из которых 2 порции являются командами, выполняющие роль адресов необходимых для использования регистров, остальные 2 порции являются названиями этих регистров.

Блок SPI_MASTER выполняет побитовый вывод команд и данных в канал SDIO на ЦАП (команда, данные, команда, данные и т.д.) через последовательный интерфейс SPI. В данном случае имеется всего одно ведомое устройство, но

интерфейс SPI позволяет использовать несколько ведущих устройств, которые регулируется значением сигнала ss_n. Роль ведущего устройства выполняет блок SPI_MASTER, а ведомым – цифро-аналоговый преобразователь.

Блок DIVIDER_FREQ является делителем частоты на 10 приходящих команд на ПЛИС от ПК через блок AXI_SLAVE. То есть блок преобразует тактовый сигнал приходящих команд с 50 МГц в тактовый сигнал с частотой 5МГц.

2.3. Реализация блоков устройства

2.3.1. Блок AXI_SLAVE

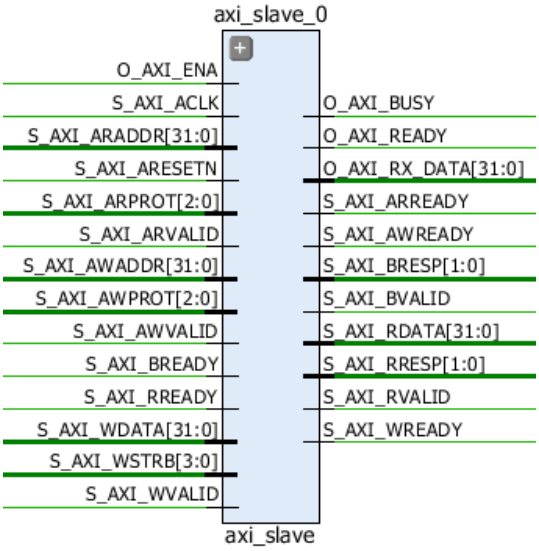


Рисунок 11 – Блок AXI_SLAVE

Таблица 2 – Сигналы блока AXI_SLAVE [13]

№ п/п	Название данных/сигнала	Источник/ Размерность	Описание сигнала
Сигналы блока AXI4-Lite Slave			
Глобальные сигналы			
1	S_AXI_ACLK	in std_logic	Глобальный сигнал тактирования частотой 50MHz
2	S_AXI_RESETN	in std_logic	глобальный сигнал сброса, активный уровень — низкий
Сигналы канала записи адреса			
3	S_AXI_ AWADDR	in std_logic_ vector (31 downto 0)	Адрес, по которому будут записываться данные
4	S_AXI_ AWPROT	in std_logic_	Сигнал устанавливает тип

		vector (2 downto 0)	привилегий и уровень безопасности для транзакций
5	S_AXI_AWVALID	in std_logic	Сигнал подтверждает, что на AWADDR выставлен актуальный адрес
6	S_AXI_ AWREADY	out std_logic	Сигнал отражает готовность устройства принять адрес записи и соответствующие сигналы управления
Сигналы канала записи данных			
7	S_AXI_WDATA	in std_logic_ vector (31 downto 0)	Данные для записи
8	S_AXI_WSTRB	in std_logic_ vector (2 downto 0)	Сигнал показывает, какие из байтов на WDATA будут записаны. В AXI4-Lite используется вся ширина шины, поэтому все линии WSTRB должны быть в высоком уровне
9	S_AXI_WVALID	in std_logic	Сигнал подтверждает, что на WDATA находятся актуальные данные
10	S_AXI_WREADY	out std_logic	Сигнал отражает готовность устройства принять данные и соответствующие сигналы управления
Сигналы канала статуса операции записи			
11	S_AXI_BRESP	out std_logic_ vector (1 downto 0)	Статус произведенной операции записи
12	S_AXI_BVALID	out std_logic	Сигнал подтверждает, что на BRESP находятся актуальные данные
13	S_AXI_BREADY	in std_logic	Сигнал отражает готовность устройства принять статус записи
Сигналы канала чтения адреса			
14	S_AXI_ARADDR	in std_logic_ vector (31 downto 0)	Адрес, по которому будут прочитаны данные
15	S_AXI_ARPROT	in std_logic_ vector (2 downto 0)	Сигнал устанавливает тип привилегий и уровень безопасности для транзакций
16	S_AXI_ARVALID	in std_logic	Сигнал подтверждает, что на ARADDR выставлен актуальный адрес
17	S_AXI_ARREADY	out std_logic	Сигнал отражает готовность устройства принять адрес чтения и соответствующие сигналы управления
Сигналы канала чтения данных			
18	S_AXI_RDATA	out std_logic_ vector (31 downto 0)	Прочитанные данные
19	S_AXI_RRESP	out std_logic_ vector (1 downto 0)	Статус произведенной операции чтения

20	S_AXI_RVALID	out std_logic	Сигнал подтверждает, что на RRESP находятся актуальные данные
21	S_AXI_RREADY	in std_logic	Сигнал отражает готовность устройства принять статус чтения
Сигналы для передачи данных на блок межинтерфейсного взаимодействия			
22	O_AXI_BUSY	out std_logic	Сигнал, который показывает, идет ли передача данных
23	O_AXI_ENA	in std_logic	Сигнал, который дает разрешение на передачу данных в следующий блок
24	O_AXI_RX_DATA	out std_logic_ vector (31 downto 0)	Данные, передаваемые на блок AXI_TO_SPI
25	O_AXI_READY	out std_logic	Сигнал, который сообщает о завершении передачи данных

Автомат блока состоит из 4 состояний. Переход между состояниями происходит с помощью определенных значений настроек сигналов AXI, необходимых для работы модуля.

В автомате имеются следующие состояния:

addr_wait: происходит предустановка устройства для начала передачи данных, также в данном состоянии обнуляются значения настроек валидности и готовности к передаче данных;

write state: происходит запись данных в регистр;

response: происходит ответ на получение данных и передача данных на следующий блок;

read_state: в данном состоянии считываются данные с регистра.

ГСА блока представлен на рис. 12:

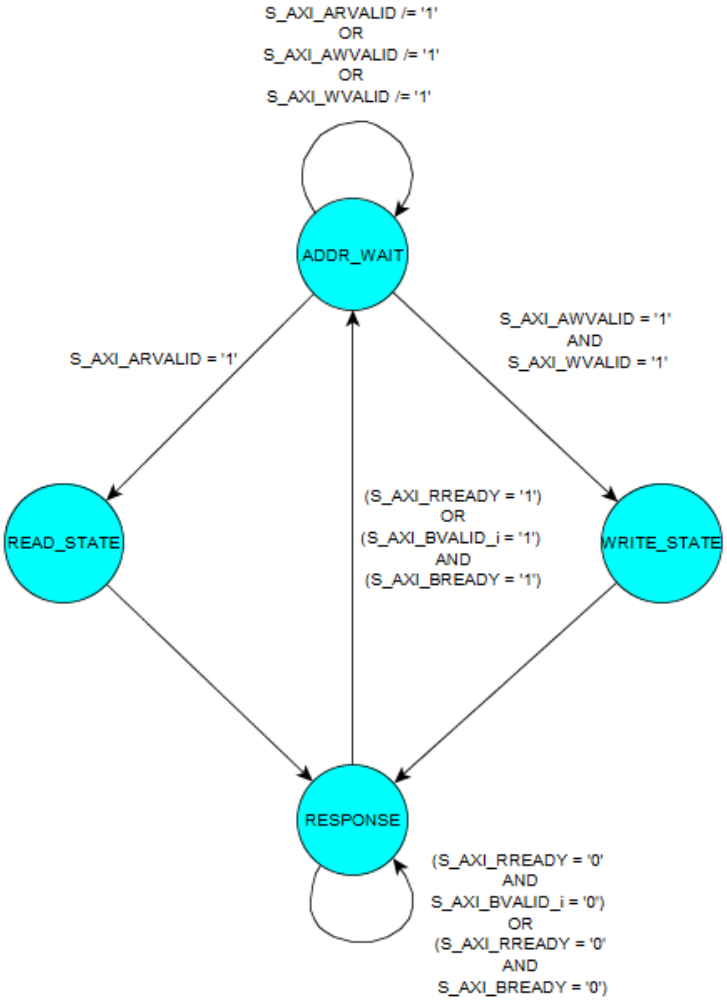


Рисунок 12 – ГСА автомата блока AXI_SLAVE

2.3.2. Блок SPI_MASTER

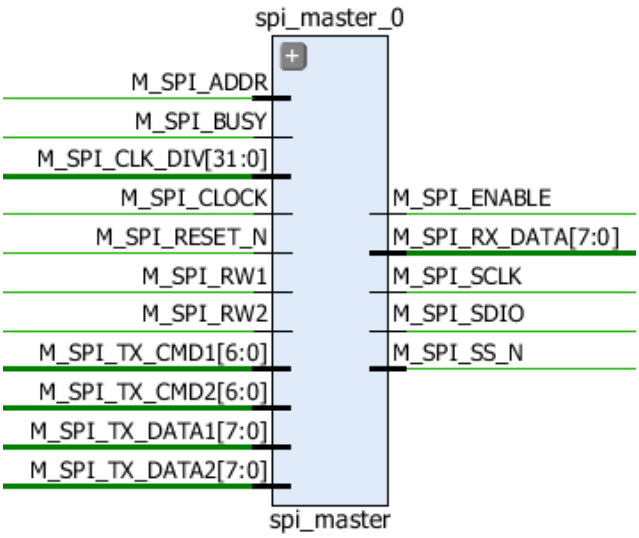


Рисунок 13 – Блок SPI_MASTER

Таблица 3 – Сигналы блока SPI_MASTER [15]

№ п/п	Название данных/сигнала	Источник/ Размерность	Описание сигнала
Сигналы блока SPI_MASTER			
1	M_SPI_ADDR	in integer	Установка адреса
2	M_SPI_BUSY	in std_logic	Показывает, передаются ли данные в данный момент
3	M_SPI_CLK_DIV	in integer range 0 to 31	Установка скорости
4	M_SPI_CLOCK	in std_logic	Глобальный сигнал тактирования частотой 5MHz
5	M_SPI_RESET_N	in std_logic	Глобальный сброс тактирования
6	M_SPI_RW1	in std_logic	1 команда чтения/записи
7	M_SPI_RW2	in std_logic	2 команда чтения/записи
8	M_SPI_TX_CMD1	in std_logic_ vector (cmd_width-2 downto 0)	1 порция передаваемых команд
9	M_SPI_TX_CMD2	in std_logic_ vector (cmd_width-2 downto 0)	2 порция передаваемых команд
10	M_SPI_TX_DATA1	in std_logic_ vector (d_width-1 downto 0)	1 порция передаваемых данных
11	M_SPI_TX_DATA2	in std_logic_ vector (d_width-1 downto 0)	2 порция передаваемых данных
12	M_SPI_ENABLE	out std_logic	Разрешение на передачу данных
13	M_SPI_RX_DATA	out std_logic _vector (d_width-1 downto 0)	Считываемые данные
Сигналы, подаваемые на ЦАП MCP4922			
14	M_SPI_SCLK	out std_logic	Тактовая частота передаваемых сигналов 5 МГц
15	M_SPI_SDIO	inout std_logic	Последовательная выдача инструкций и данных
16	M_SPI_SS_N	out std_logic_ vector(slaves-1 downto 0)	Выбор ведомого устройства

Автомат блока состоит из 6 состояний. В каждом состоянии прописаны свои процессы работы данного модуля/блока.

Автомат состоит из следующих состояний:

idle: предустановка блока для начала передачи данных, по глобальному сбросу переходит в это состояние;

execute: инициализация загрузки данных в буфер;

cmd_tx: запись команд чтения и записи, а также 2 порции команд/инструкций в буфер;

data_tx: запись 2 порций данных в буфер;

transfer: побитовая передача инструкций и данных из 2 буферов: сначала передается первые 2 байта (старшими битами вперед): сначала подается команда чтения/записи, потом инструкции, затем данные. После окончания передачи 2 байтов данных передаются следующие 2 байта;

tr_end: завершение передачи данных.

ГСА автомата блока представлен на рис. 14:

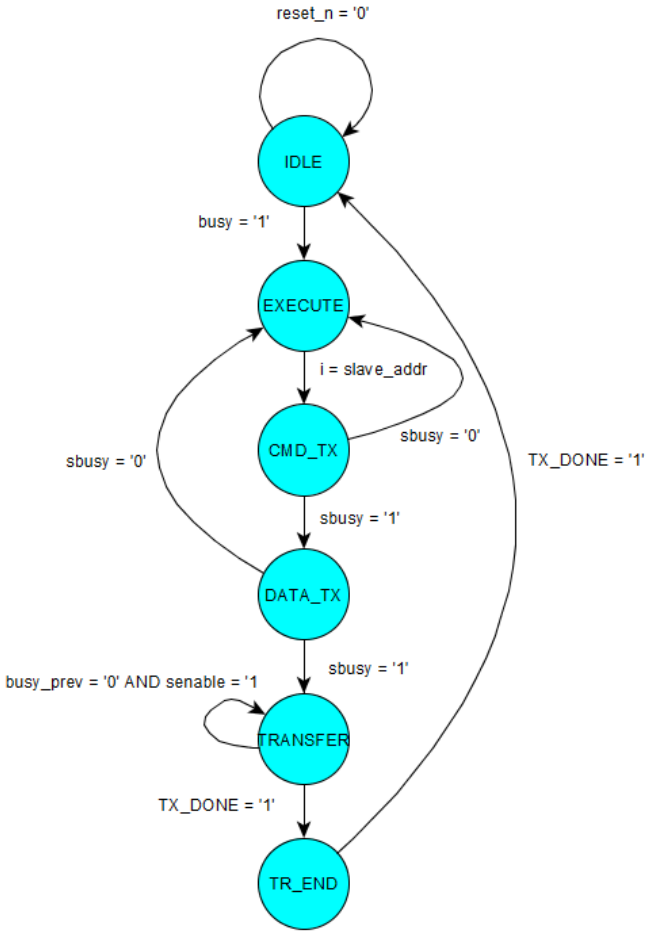


Рисунок 14 – ГСА автомата блока SPI_MASTER

2.3.3. Блок AXI_TO_SPI

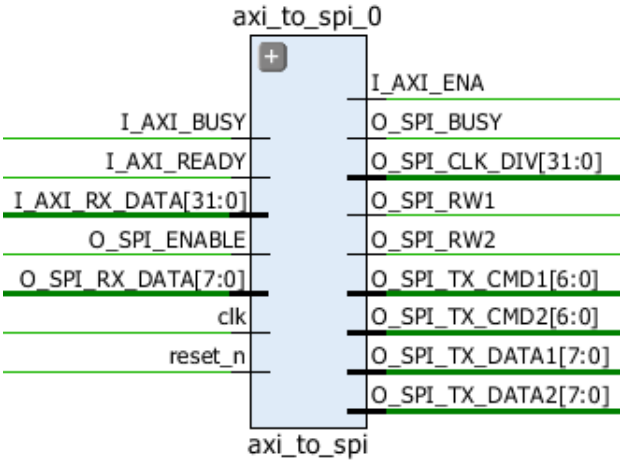


Рисунок 15 – Блок AXI_TO_SPI

Таблица 4 – Сигналы блока AXI_TO_SPI

№ п/п	Название данных/сигнала	Источник/ Размерность	Описание сигнала
1	I_AXI_BUSY	in std_logic	Сигнал, который показывает, идет ли передача данных
2	I_AXI_READY	in std_logic	Сигнал, который сообщает о завершении передачи данных
3	I_AXI_RX_DATA	in std_logic_ vector (31 downto 0)	Данные, получаемые с блока AXI_SLAVE
4	O_SPI_ENABLE	in std_logic	Настройка адреса для транзакции
5	O_SPI_RX_DATA	in std_logic_ vector (7 downto 0)	Данные, передаваемые на блок AXI_TO_SPI
6	CLK	in std_logic	Глобальный сигнал тактирования частотой 25MHz и сброс тактирования
7	RESET_N	in std_logic	Глобальный сигнал тактирования частотой 25MHz и сброс тактирования
8	I_AXI_ENA	out std_logic	Сигнал, который дает разрешение на передачу данных в следующий блок
9	O_SPI_BUSY	out std_logic	Показывает, передаются ли данные в данный момент
10	O_SPI_CLK_DIV	out integer range 0 to 31	Установка скорости
11	O_SPI_RW1	out std_logic	Чтение/запись 1 команды
12	O_SPI_RW2	out std_logic	Чтение/запись 2 команды
13	O_SPI_ TX_CMD1	out std_logic_	1 порция передаваемых команд

		vector (cmd_width-2 downto 0)	
14	O_SPI_ TX_CMD2	out std_logic_ vector (cmd_width-2 downto 0)	2 порция передаваемых команд
15	O_SPI_ TX_DATA1	out std_logic_ vector (d_width-1 downto 0)	1 порция передаваемых данных
16	O_SPI_ TX_ DATA2	out std_logic_ vector (d_width-1 downto 0)	2 порция передаваемых данных

В каждом состоянии задействованы свои процессы работы блока. Автомат блока имеет следующие состояния:

ready: предустановка блока для начала передачи данных: сброс блока, обнуление переменных;

axi_rx: запись принимаемых данных в буфер;

spi_tx: разбиение 32-битных данных на 2 команды чтения/записи, 2 порции инструкций и 2 порции данных;

spi_load: загрузка данных и переход в следующий блок.

ГСА блока представлен на рис. 12:

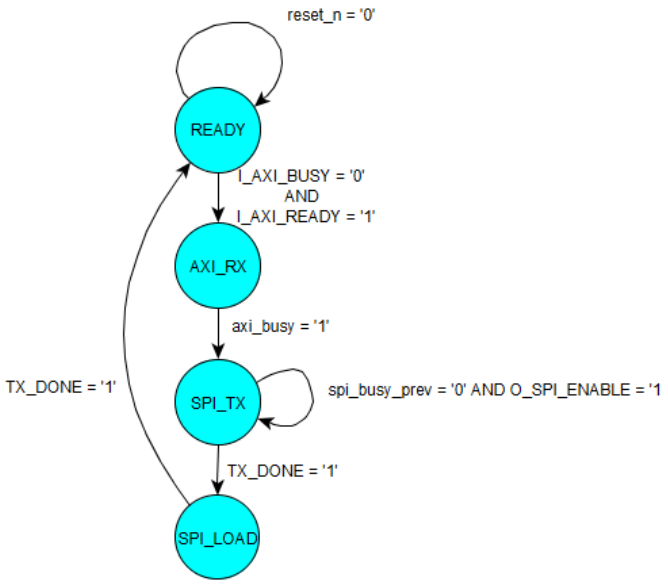


Рисунок 16 – ГСА автомата блока AXI_TO_SPI

В состоянии SPI_TX автомата происходит мультиплексирование данных, которые приходят с блока AXI_SLAVE. 32 бита данных разбиваются на 2 массива данных. В каждом массиве имеются команда чтения/записи (RW, чтение – ‘0’, запись – ‘1’), порция команд/инструкций (commands) и порция данных (data). Данное разбиение было обусловлено для формирования передачи 2 типов данных (инструкций и данных) по последовательному интерфейсу SPI.

	MSB					LSB
Bit Description:	*RW*	*Commands*	*Data*	*RW*	*Commands*	*Data*
Bits:	*31*	*30 - 24*	*23 - 16*	*15*	*14 - 8*	*7 - 0*

Рисунок 17 – Формат транзакций блока AXI_TO_SPI

Переход между передачами осуществляется по счетчику spi_busy_cnt:

0: осуществление передачи старшего байта данных message (31 DOWNTO 24);

1: осуществление передачи следующей порции данных message (23 DOWNTO 16);

2: осуществление передачи следующей порции данных message (15 DOWNTO 8);

3: осуществление передачи младшего байта данных message (7 DOWNTO 0) и установление сигнала TX_DONE = ‘1’ для подтверждения передачи данных.

2.3.4. Блок DIVIDER_FREQ

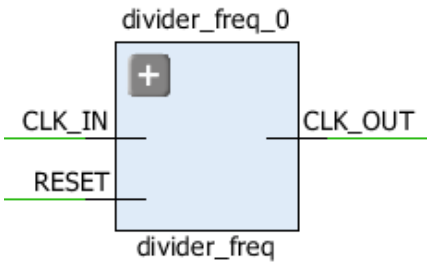


Рисунок 18 – Блок DIVIDER_FREQ

Таблица 5 – Сигналы блока DIVIDER_FREQ

№ п/п	Название данных/сигнала	Источник/ Размерность	Описание сигнала
1	CLK_IN	out std_logic	Сигнал, который показывает, идет ли передача данных
2	RESET	in std_logic	Сигнал, который дает разрешение на передачу данных в следующий блок
3	CLK_OUT	out std_logic_ vector (31 downto 0)	Данные, передаваемые на блок AXI_TO_SPI

Данный модуль представляет собой делитель частоты приходящих команд в 10 раз. Выходной сигнал данного модуля позволяет подать тактовую частоту передаваемых сигналов 5 МГц по последовательному интерфейсу SPI, что позволит обеспечить хорошую производительность данного устройства.

2.4. Структурно-функциональная и функциональная схемы второго исполнения устройства

Для автоматизации подаваемых тестовых наборов имеется возможность усовершенствовать первое исполнение устройства. Программный модуль AXI_TO_SPI_DEVICE необходимо обернуть в IP-core для подключения к другому устройству. Для того, чтобы реализовать программную часть устройства и наладить автоматическую подачу тестовых наборов, следует подключить soft-процессор Microblaze.

Структурно-функциональная схема устройства будет состоять из разработанного ранее модуля (обозначено на схеме FPGA module), soft-процессора Microblaze, динамической памяти DDR SDRAM, цифро-аналогового преобразователя MCP4922 и других элементов (приложение 5).

Собранный проект в Xilinx Vivado FPGA IP-core необходимо прошить через программатор в ПЛИС Xilinx Spartan 3E, главный модуль представлен в прил.19.

Тестовые наборы данных хранятся в динамической памяти DDR SDRAM. Данные из памяти будут поступать на ПЛИС, которая будет их считывать и преобразовывать, но управлять их работой также будет soft-процессор Microblaze.

Soft-процессор Microblaze позволит на программном уровне выводить на ЦАП MCP4922 тестовые наборы массивов данных и записывать их в необходимые регистры. Для того, чтобы устройство в дальнейшем можно было использовать, нужно написать программу для soft-процессора Microblaze на языке C/C++ для формирования синусоидального сигнала на выходе (приложение 20). Microblaze взаимодействует с модулем FPGA module через интерфейс AXI4-Lite для задействования необходимых для работы регистров. Все необходимые преобразования в модуле были описаны ранее [12, 15, 19].

Преобразованные данные передаются на ЦАП по интерфейсу SPI, который переводит последовательные данные в синусоидальный сигнал.

Убедиться в наличии данного сигнала можно с помощью осциллографа.

После добавления всех необходимых элементов в разрабатываемое устройство после компиляции формируется функциональная схема IP-core устройства межинтерфейсного взаимодействия (приложение 6).

2.5. Описание принципиальной схемы устройства

Для того, чтобы создать устройство на печатной плате, надо провести этапы компоновки и трассировки элементов на печатной плате. Для выполнения этих этапов надо сформировать нетлист, который создается после создания принципиальной схемы. Для начала создаются условно-графические обозначения элементов и посадочные места устройства (приложение 9). Библиотеки созданных элементов добавляются для создания принципиальной электрической схемы устройства (приложение 10). Спецификация элементов представлена в приложениях 11 и 12. Проектирование печатной платы происходит в программе P-CAD.

Основными элементами ПЭС являются микросхемы DD1 (преобразователь USB <-> UART FT232BM), DD2 (EEPROM), DD3 (ПЛИС Xilinx Spartan 3E), DD4 (цифро-аналоговый преобразователь MCP4922), DD5 (синхронная динамическая память DDR SDRAM AT25HP256).

Подключение ПК к устройству будет осуществляться через интерфейс USB (XP1), то есть с ЭВМ верхнего уровня.

Для того, чтобы передать данные на ПЛИС, необходимо наладить взаимодействие между ПЛИС (DD3) и преобразователем USB-UART (DD1) через сигналы TXD и RXD.

Данные и адреса, подаваемые на ПЛИС, хранятся в синхронной динамической памяти DDR SDRAM (DD5). Адресные сигналы A0-A11 и данные DQ0-DQ3 передаются на ПЛИС.

Для тестирования работы устройства по JTAG сигналы TCK, TDI, TDO подключены к разъему XP2 к отдельным гнездам, куда можно подключить необходимые щупы.

Сигналы выхода SPI nCS, SS_N и SDIO с ПЛИС передаются на входы ЦАП (DD4), на который подается опорное напряжение Vref и выдается синусоидальный сигнал Vouta. Для вывода нескольких синусоидальных сигналов имеется двухканальные выходы Vouta и Voutb.

Схема сброса включает резистор R1, конденсатор C3 и ключ SW1. При замыкании ключа (нажатие кнопки) на выходе RESET формируется сигнал равный уровню логического нуля, осуществляющий сброс микросхемы DD2.

Тактовую частоту задает кварцевый резонатор XTAL по сигналам Xin и Xout. Ферритовый стержень L1 устраняет помехи в источнике питания.

Для обеспечения работоспособности устройства, сглаживания помех и деления напряжений имеются многочисленные конденсаторы и резисторы. Для каждого источника питания подключены керамические и электролитические конденсаторы [14, 17, 18].

В приложениях 14 и 13 представлен общий вид двухсторонней печатной платы и компоновка элементов на ней, разведенные по принципиальной схеме (приложение 1) в среде ППП PCAD 2006. В приложениях 15 и 16 представлены верхний (Top) и нижний (Bottom) слои печатной платы.

2.6. Расчет потребления токов и мощности

Средний выходной ток ПЛИС Xilinx Spartan 3E составляет 80мА.

Напряжение питания схемы составляет 5В±5%. Ниже приведена таблица потребляемых токов для микросхем, входящих в ПЭС системы тестирования (табл. 6).

Таблица 6 – Таблица потребляемых токов

Микросхема	Потребляемый ток, мА	Количество
DD1	30	1
DD2	3,6	1
DD3	80	1
DD4	20	1
DD5	20	1

Напряжение питания всех микросхем составляет 5В.

Таким образом, общий потребляемый ток составляет:

$$Iпотр = 30+3,6+80+20+20 = 153.6 мА$$

Мощность, потребляемая схемой, составляет:

$$Рпотр = Iпотр * Uпит = 153,6*5 = 768 мВт.$$

Резисторы для понижения напряжения на микросхемы DD1 и DD2 нужны, т.к. выходное напряжение DD1 не соответствует напряжению питания ПЛИС DD3.

3. ТЕСТИРОВАНИЕ УСТРОЙСТВА МЕЖИНТЕРФЕЙСНОГО ВЗАИМОДЕЙСТВИЯ AXI-TO-SPI НА ПЛИС

3.1. Методика тестирования

Для проверки работы устройства межинтерфейсного взаимодействия AXI_to_SPI нужно разработать методику тестирования, в котором будут подобраны тестовые наборы, которые выведут различные данные на выход (табл. 7). Учитывая, как устройство будет работать в теории, были выбраны следующие тестовые наборы.

Ввиду отсутствия возможности использования ПЛИС новых поколений, тестовые наборы сигнала S_AXI_AWADDR в данной работе не имеют особого значения, поэтому для каждого тестового набора данных будет одинаковое значение входного сигнала S_AXI_AWADDR.

Таблица 7 – Тестовый набор данных

S_AXI_WDATA	SDIO
11223344	00010001001000100011001101000100
AC327513	10100110001100100111010100010011
26DFBE34	00100110110111111011111000110100
CBEFF3A2	11001011111011111111001110100010

Для тестирования разрабатываемого устройства необходимо провести симуляцию проекта в специализированной программе Modelsim. В данной программе можно произвести тестирование устройства/блока как вручную (оператор самостоятельно присваивает значение определенному сигналу и проверяет, как изменяются состояния других сигналов и буферов под его воздействием), так и автоматически (после написания testbench к блоку).

Для проверки работоспособности устройства необходимо начать с проверки работы блоков устройства [10].

В ходе работы будут представлены минимум 2 изображения тестирования: в 1 будут отображены все внутренние процессы, которые происходят в блоке, включая переход между состояниями автомата, во 2 будет представлены только

интерфейсные сигналы блоков (проведено тестирование при помощи написания программы testbench к каждому из 3 блоков) для более удобного оценивания результатов тестирования.

3.2. Тестирование работы устройства

В данном модуле собраны и соединены все блоки, участвующие в работе. Для проверки необходимо подать частоту 50МГц на тактовый сигнал S_AXI_ACLK, на SW0 набор данных 11223344, на SW1 набор данных AC327513, на SW2 набор данных 26DFBE34, на SW3 набор данных CBEFF3A2. По установлению BTN в единицу устанавливаются сигналы разрешения записи AWVALID и WVALID в 1, по которым происходит загрузка данных в S_AXI_WDATA и начало передачи в блок межинтерфейсного взаимодействия AXI_TO_SPI. Соответственно, были протестированы все наборы данных и были выведены все последовательные наборы данных SDIO. Получены следующие результаты:

Таблица 8 – Результаты полученных наборов данных

S_AXI_WDATA	SDIO (RW1/CMD1/DATA1/RW2/CMD2/DATA2)
11223344	0/0010001/00100010/0/0110011/01000100
AC327513	1/0100110/00110010/0/1110101/00010011
26DFBE34	0/0100110/11011111/1/0111110/00110100
CBEFF3A2	1/1001011/11101111/1/1110011/10100010

Тестирование работы каждого блока и наглядное тестирование всего устройства приведено в приложении 2.

Результаты тестирования всех наборов данных устройства AXI_TO_SPI_DEVICE приведены на рисунке 19 – 22.

Полученные временные диаграммы доказывают корректность работы устройства, результаты работы в таблице 8 полностью совпадают с теоретическими выходными данными в методике тестирования.

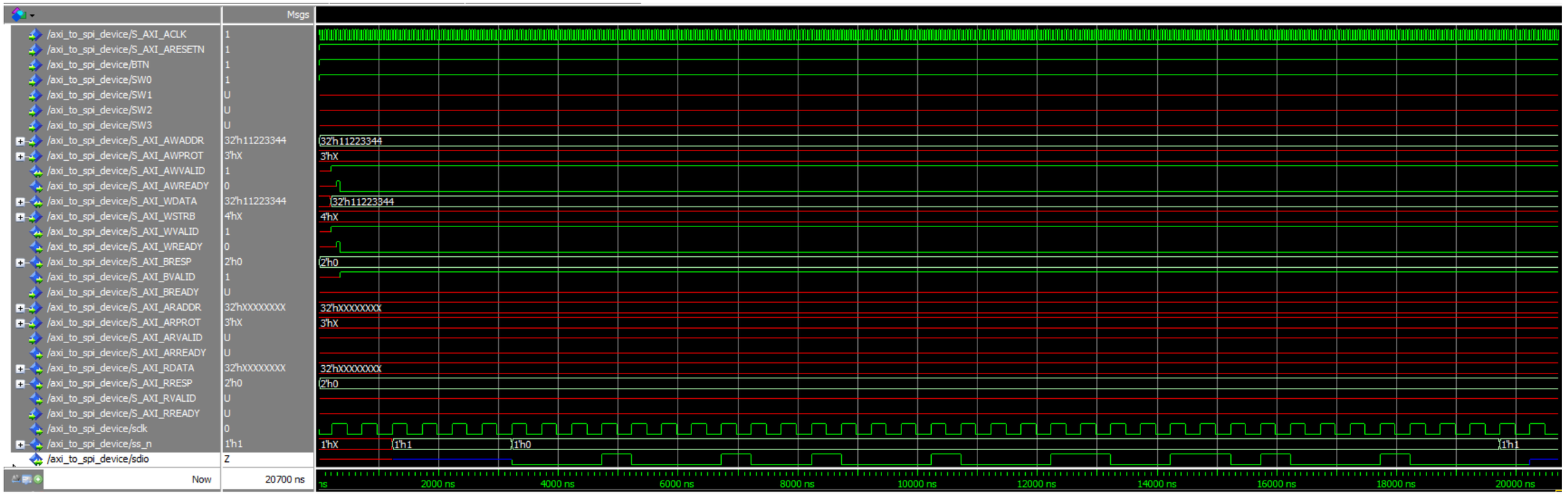


Рисунок 19 – Проверка работы первого тестового набора данных устройства AXI_TO_SPI_DEVICE

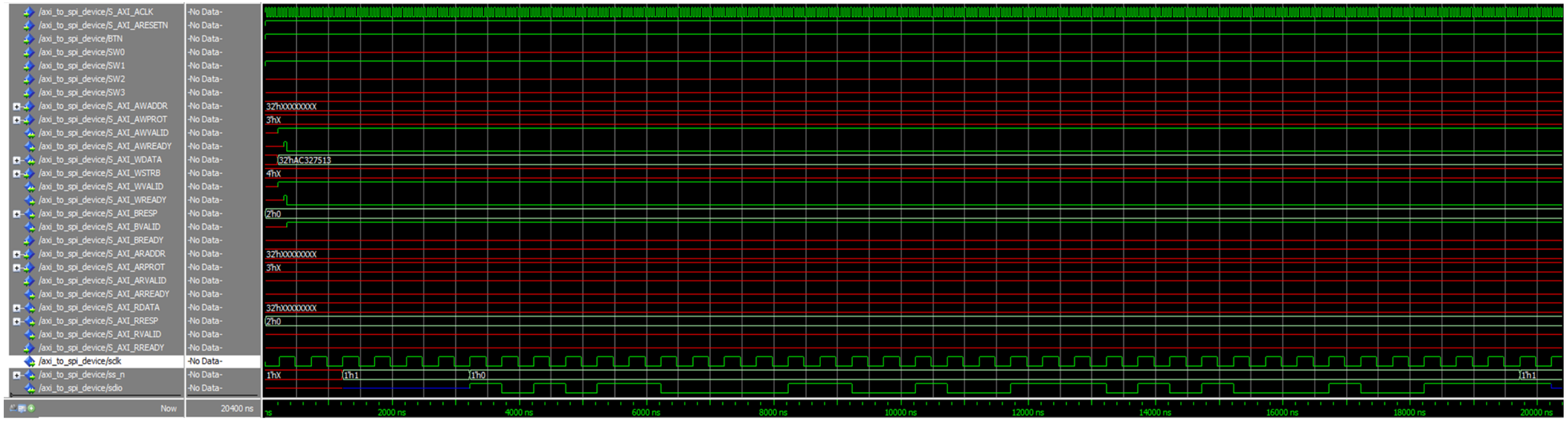


Рисунок 20 – Проверка работы второго тестового набора данных устройства AXI_TO_SPI_DEVICE

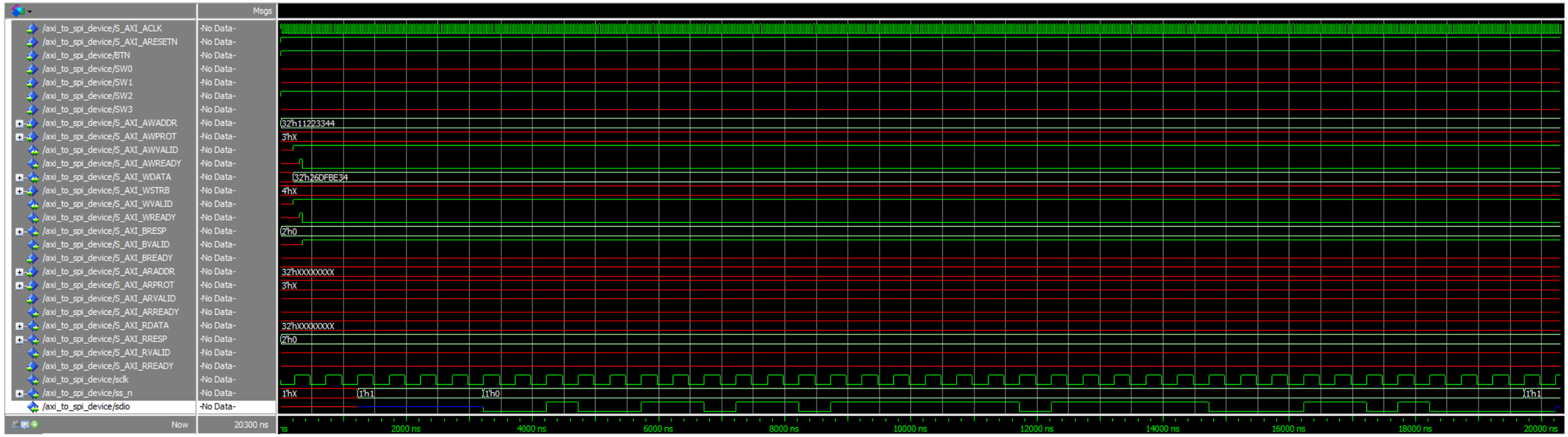


Рисунок 21 – Проверка работы третьего тестового набора данных устройства AXI_TO_SPI_DEVICE

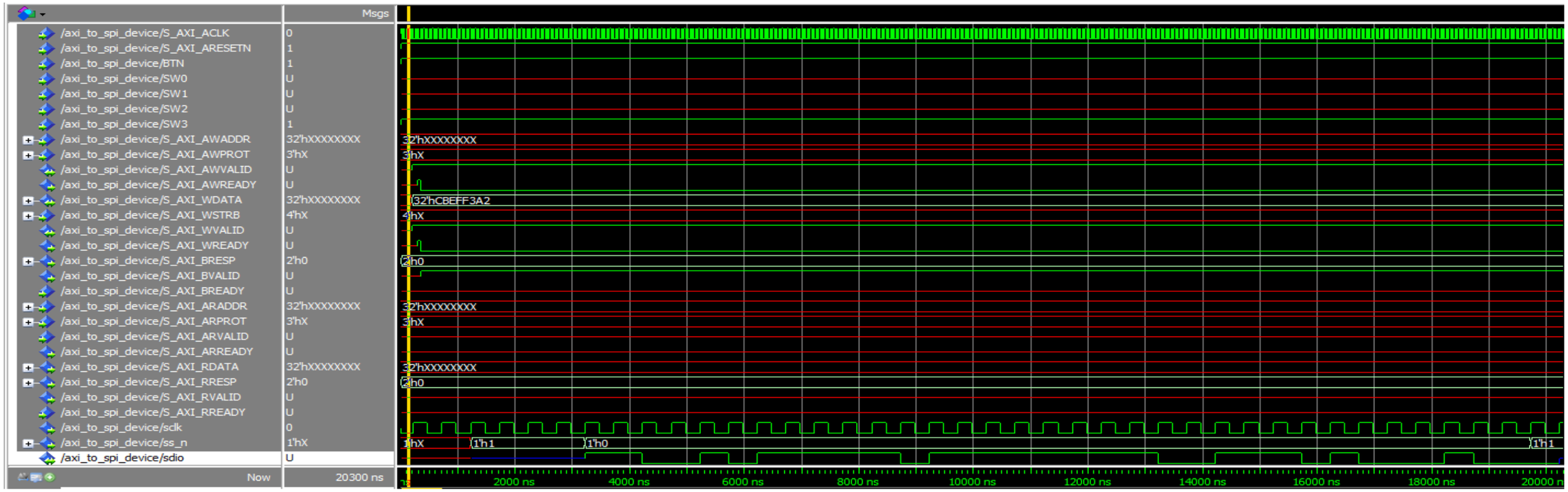


Рисунок 22 – Проверка работы четвертого тестового набора данных устройства AXI_TO_SPI_DEVICE

4. ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ

4.1. Оценка коммерческого потенциала и перспективности проведения научных исследований с позиции ресурсоэффективности и ресурсосбережения

4.1.1. Потенциальные потребители продукта

При создании продукта необходимо определить потенциального потребителя данной продукции. Ввиду того, что разработка является чисто практической и для того, чтобы ей воспользоваться, необходимы дополнительные знания и умения в области электроники, цифровой схемотехники и программирования, первичными потребителями продукции являются коммерческие организации, производящие аппаратное обеспечение в сферах, связанных с системами цифровой связи.

Таким образом, главным критерием сегментирования является специализация потенциального потребителя [22].

4.1.2. Анализ конкурентных технических решений

В качестве основных конкурентных технических решений были выбраны следующие разработки:

- межинтерфейсный адаптер с использованием ПЛИС (данная работа) (1),
- межинтерфейсный адаптер с использованием микроконтроллера (2),
- межинтерфейсный адаптер с использованием микропроцессора (3).

Результаты конкурентного анализа приведены в табл. 9:

Таблица 9 – Оценочная карта

Критерии оценки	Вес	Баллы			Конкурентоспособность		
		$Б_1$	$Б_2$	$Б_3$	$К_1$	$К_2$	$К_3$
Технические критерии оценки ресурсоэффективности							
1. Скорость работы	0,3	5	4	4	1,5	1,2	1,2
2. Универсальность	0,2	4	4	4	0,8	0,8	0,8
3. Удобство в эксплуатации	0,1	4	4	4	0,4	0,4	0,4
4. Потребность в ресурсах памяти	0,1	5	4	4	0,5	0,4	0,4
5. Функциональная мощность	0,1	5	4	5	0,5	0,4	0,5
Экономические критерии оценки эффективности							
6. Доступность	0,05	4	5	4	0,2	0,25	0,2
7. Послепродажное обслуживание	0,05	5	5	4	0,25	0,25	0,2
8. Цена	0,1	4	5	4	0,4	0,5	0,4
Итого:		36	35	33	4,55	4,2	4,1

- Под «универсальностью» подразумевается возможность взаимодействия с разными компонентами и взаимозаменяемость.
- Критерий «функциональная мощность» отражает наличие, либо отсутствие дополнительных возможностей.
- Под «доступностью» понимается то, насколько открытой является разработка.

Конкурентоспособность проекта 1 исполнения относительно проекта 2 исполнения:

$$K12 = \frac{K1}{K2} = \frac{4{,}55}{4{,}2} = 1{,}08(3)$$

Конкурентоспособность проекта 1 исполнения относительно проекта 3 исполнения:

$$K13 = \frac{K1}{K3} = \frac{4{,}55}{4{,}1} = 1{,}11$$

4.1.3. Технология QuaD

Оценочная карта для сравнения конкурентных технических решений приведена в табл. 10:

Таблица 10 – Оценочная карта

Критерии оценки	Вес	Баллы	Макс. балл	Отн. знач.	Ср.-взвеш. знач.
Показатели оценки качества разработки					
1. Скорость работы	0,3	100	100	1	0,3
2. Универсальность	0,2	80	100	0,8	0,16
3. Удобство в эксплуатации	0,1	80	100	0,8	0,08
4. Потребность в ресурсах памяти	0,1	100	100	1	0,1
5. Функциональная мощность	0,1	100	100	1	0,1
Показатели оценки коммерческого потенциала разработки					
6. Доступность	0,05	80	100	0,8	0,04
7. Законченность работы	0,05	90	100	0,9	0,045
8. Цена	0,1	80	100	0,8	0,08
				Итог:	0,885

Оценка качества и перспективности по технологии QuaD определяется по формуле:

$$П_{ср} = \sum В_i Б_i, \qquad (2)$$

где Пср – средневзвешенное значение показателя качества и перспективности научной разработки;

Bi – вес показателя (в долях единицы);

Бі – балл і-го показателя.

Можно заметить, что интегральный показатель конкурентоспособности данной разработки составляет 0,885, что является достаточно благоприятным для продолжения разработки [24].

4.1.4. SWOT-анализ

Описание сильных и слабых сторон проекта, выявление возможностей и угроз. Результаты первого этапа представлены в табл. 11:

Таблица 11 – Результаты первого этапа SWOT-анализа

	Сильные стороны 1. Наличие открытой документации к интерфейсам 2.Использование методологии проектирования в виде программного кода 3. Широкие возможности по масштабированию проекта 4. Высокая скорость работы	**Слабые стороны** 1. Сложность реализации программы относительно других конкурентов 2. Не все ПЛИС можно использовать в проекте 3. Проектирование в виде программного кода по сравнению с блочным проектированием менее наглядно.
Возможности 1. Появление дополнительного спроса на новый продукт 2. Повышение стоимости конкурентных разработок	B1C1: Возможность рассылки и выкладывания в общий доступ документации к разработке. B2C2C3C4: Появление новых оптимизаций, которые в будущем можно будет легко интегрировать в разработку, тем самым значительно улучшая ее конкурентоспособность.	B1C3: Программистам не нужно вдаваться в подробности схемотехнической работы, что оптимизирует их работу. B2C1: При всей своей сложности работы язык описания аппаратуры более понятен начинающим программистам.
Угрозы 1.Отсутствие мотивации к дальнейшему развитию проекта 2.Недостаток человеческих ресурсов по сравнению с конкурентными разработками.	У1С1: Документация не всегда описывается достаточно понятна для разработчика, нетривиальные интерфейсы, требующие большого опыта У2С3: Широкие возможности для масштабирования проекта доступны опытным разработчикам, возникновения спорных ситуаций при масштабировании проекта разными отделами предприятия	У1С1: Огромная трудоемкость работы и большое число разнообразных ошибок при разработке. При прекращении работы проект не сможет раскрыть своего потенциала. У2С3: Использование привычной методологии проектирования потребует большой работы для увеличения масштабности проекта, что увеличит его стоимость.

Интерактивная матрица проекта представлена в табл. 12 и 13:

Таблица 12 – Интерактивная матрица проекта (сильные стороны)

	Сил1	Сил2	Сил3	Сил4
В1	+	-	-	-
В2	-	+	+	+
У1	+	-	-	-
У2	-	-	+	-

Таблица 13 – Интерактивная матрица проекта (слабые стороны)

	Слаб1	Слаб2	Слаб3
В1	-	-	+
В2	+	-	-
У1	+	-	-
У2	-	-	+

Таким образом, можно сделать вывод, что проект необходимо развивать, применяя наиболее новые и оптимизированные интерфейсы, что позволит создать наиболее конкурентоспособную разработку устройства [20].

4.2. Определение возможных альтернатив проведения научных исследований

В качестве морфологических характеристик в данной работе можно выделить методологию проектирования, язык описания аппаратуры и режимы тактирования (подробней см. в основной части данной работы). Морфологическая матрица приведена в табл. 14:

Таблица 14 – Морфологическая матрица

Альтернативы	1	2	3
А. Методология проектирования	Блочная методология проектирования	Программирование	-
Б. Язык описания аппаратуры	Verilog HDL	VHDL	AHDL
В. Тактирование	Использование одного тактового сигнала для всех модулей	Отдельное тактирование каждого модуля	Отдельное тактирование входного и выходного модулей

В качестве методологии проектирования было выбрано проектирование в виде написания программного кода (А2), так как он обладает рядом преимуществ:

- создание блоков в проекте способствует появлению ошибок, не связанных с ошибками в написанной программе;
- не требуются знания основ схемотехники (устранение дребезга контакта при блочном проектировании);
- в больших проектах есть возможность путаницы в соединении выходов одного блока со входами других, при компиляции модулей, написанных программным кодом, функциональная схема формируется сама.

Выбор языка не имеет высокого значения в данном случае, так как все языки имеют примерно одинаковые возможности, но язык VHDL (Б2) имеет строгий контроль типов данных и его предпочитают программисты на предприятиях.

В случае с тактированием был остановлены выбор на отдельном тактировании входного и выходного модулей (В3) ввиду того, что этого требует реализация поставленной задачи: входные команды на модуль AXI_SLAVE должны приходить с частотой 50МГц, в то время как на SPI_MASTER команды приходят уже с частотой 5МГц. При других способах тактирования не будет осуществляться необходимая производительность устройства [23].

4.3. Планирование научно-исследовательских работ

4.3.1. Структура работ в рамках научного исследования

Перечень этапов и работ в рамках проведения научного исследования представлен в табл. 15:

Таблица 15 – Перечень этапов, работ и распределение исполнителей

Основные этапы	№	Содержание работ	Исполнитель
Разработка технического задания	1	Составление и утверждение технического задания	Р
	2	Календарное планирование работ по теме	И
Аналитический обзор	3	Подбор и изучение материалов по теме	Р,И
	4	Изучение уже существующих решений в данной области	И
	5	Выбор компонентов для реализации устройства	Р,И
Проектирование межинтерфейсного адаптера	6	Изучение технической документации последовательного и параллельного интерфейсов	И
	7	Составление структурно-функциональной схемы	И
	8	Проектирование ГСА автоматов блоков с указанием возможных состояний	И
Реализация межинтерфейсного адаптера	9	Выбор языка описания аппаратуры для программирования	И
	10	Программирование блоков СФС на основании ГСА автоматов к ним	И
	11	Устранение всех ошибок в программе и компиляция программы	И
Тестирование и апробация	12	Разработка методики тестирования устройства	И
	13	Тестирование программы на временных диаграммах	И
	14	Апробация разработанного устройства на железе	И
Компоновка и трассировка элементов на	15	Выбор ПО для проектирования	И
	16	Проектирование УГО и посадочных мест	И

печатной плате	17	Ввод принципиальной схемы	И
	18	Компоновка и трассировка элементов на печатной плате	И
Обобщение и оценка результатов	19	Оценка эффективности полученных результатов	Р,И
	20	Оценка целесообразности проведения дальнейших исследований по данной теме	Р,И

Р – Научный руководитель; И – Инженер-программист.

4.3.2. Определение трудоемкости выполнения работ

Оценим трудоемкость выполнения вышеозначенных работ. Для этого оценим минимальное и максимальное время выполнения каждой работы (табл. 40, приведена в приложении 21). Также произведем расчет ожидаемого значения трудоемкости по следующей формуле:

$$t_{\text{ож},i} = \frac{\left(3t_{min,i} + 2t_{max,i}\right)}{5}$$

где $t_{\text{ож, i}}$ – ожидаемая трудоемкость выполнения i-ой работы, чел.-дн.;

$t_{\text{min, i}}$ – минимально возможная трудоемкость выполнения i-ой работы (оптимистическая оценка: в предположении наиболее благоприятного стечения обстоятельств), чел.-дн.;

$t_{\text{max, i}}$ – максимально возможная трудоемкость i-ой работы (пессимистическая оценка: в предположении наиболее неблагоприятного стечения обстоятельств), чел.-дн.

Следует заметить, что исполнитель «Инженер-программист» задействован в каждой из перечисленных работ, а потому невозможно ускорение за счет параллельности их выполнения.

4.3.3. Разработка графика проведения научного исследования

В приложении 21 приведены временные показатели научного исследования.

На основе данной таблицы строится календарный план-график научного исследовательского проекта (рис. 69, прил. 22). График строится для максимального по длительности исполнения работ в рамках научно-исследо-

вательского проекта с разбивкой по месяцам и неделям за период дипломирования. Для удобной визуализации разработка календарного плана графика производится в программе Microsoft Project 2013 в виде диаграммы Ганта [25].

4.3.4. Бюджет научно-технического исследования

Расчет бюджета НТИ сводится к расчету материальных затрат, затрат на з/п руководителя и затрат на з/п инженера. При этом материальные затраты состоят только из расходных материалов и амортизации оборудования. Обе эти статьи будут учтены при расчете накладных расходов.

Средний оклад руководителя от ТПУ (доцент, к.т.н) составляет 23265 рубля (без учета районного коэффициента) Расчеты были произведены по формуле

$$З_{\text{срмес.}} = \frac{\sum_{n=1}^{12} З_{\text{мес}(n).}}{12} = \frac{293100}{12} = 23\,425 \text{ руб.}$$

Оклад дипломника составляет 6976 руб. (без учета районного коэффициента) (принято на основе данных с окладов профессорско-преподавательского состава и дипломников-студентов).

Исходя из объема нагрузки в размере двух консультаций в неделю, каждая по два астрономических часа, получаем 16 астрономических часов в месяц. Учитывая 6-ичасовой рабочий день с учетом 6-ти дневной рабочей недели, затраты на з/п руководителя составляют два среднедневных оклада в месяц, или 1458 рублей.

С учетом районного коэффициента, равного 30% от оклада, получается месячная заработная плата:

$$З_{\text{м}}^{\text{рук}} = 23\,425 * 1{,}3 = 30452{,}5 \text{ руб.},$$

$$З_{\text{м}}^{\text{разр}} = 6976 * 1{,}3 = 9068{,}8 \text{ руб.}$$

Зная месячную заработную плату каждого участника проекта, можно рассчитать соответствующую среднедневную заработную плату. Количество месяцев работы без отпуска принимается равным 11,2 (считается отпуск длиною 24 рабочих дня при 6-дневной рабочей недели):

$$З_{дн}^{рук} = \frac{30452{,}5 * 11{,}2}{219} = 1557{,}38 \text{ руб.},$$

$$З_{дн}^{разр} = \frac{9068{,}8 * 11{,}2}{219} = 462{,}92 \text{ руб.}$$

С учётом основной заработной платы, можно посчитать дополнительную заработную плату в размере 12 % от основной:

$$З_{дн}^{рук} = k_{доп} * З_{осн} = 0{,}12 * 30452{,}5 \ = 3\,654{,}3 \text{ руб.}$$

$$З_{дн}^{разр} = k_{доп} * З_{осн} = 0{,}12 * 9068{,}8 \ = 1\,397 \text{ руб.}$$

Величину отчислений во внебюджетные фонды определяется как:

$$З_{внеб} = k_{внеб}(З_{осн} + З_{доп}) = 0{,}271 * (9\,314 + 1\,397) = 2\,903 \text{ руб.}$$

Научных и производственных командировок в данном исследовании не производилось. Контрагентные расходы отсутствуют.

Затраты на специальное оборудование будет состоять из микросхем, ПЛИС/микроконтроллер/микропроцессор в зависимости от исполнения и печатной платы. Для апробации устройства имеется плата Elvis II, которая стоит 120300 руб. В совокупности при использовании ПЛИС данные затраты выйдут в размере 12000 рублей.

Материальные затраты учитываются с учетом количества использованной электроэнергии. Для юридических лиц стоимость 1 кВт*ч составляет 5,8 рублей. При умеренном пользовании компьютер средней мощности затрачивает 1,176 кВт в день в среднем. В стоимость включить стоимость канцтоваров $З_{матк} = 350$ руб., включающие стоимость бумаги и ручек [26].

$$З_{мат} = М_{д} * Д_{раб} * 6 * 5{,}8 + З_{матк}$$

$$З_{мат} = 1{,}176 \text{ кВт} * 154 \text{ дней} * 6 \text{ ч} * 5{,}8 \ \frac{\text{руб}}{\text{кВт} * \text{ч}} + 350 \text{ руб.} = \ 190510 \text{ руб.}$$

Накладные расходы рассчитаем как:

$$З_{накл} = (З_{внеб} + З_{доп} + З_{осн}) * k_{нр} = (9\,314 + 1\,397 + 2903) * 0{,}16 \ =$$

$$= 2\,178 \text{ руб.}$$

Бюджет затрат приведен в табл. 16:

Таблица 16 – Бюджет затрат по каждому исполнению НТИ

Наименование статьи	Сумма руб.		
	Исп. 1	Исп. 2	Исп. 3
1. Материальные затраты НТИ	190510	215820	210390
2. Затраты на спец. оборудование	132500	11400	12350
3. Затраты по основной з/п	68 770	69 570	68829
4. Затраты по доп. з/п	8814	7345	7456
5. Отчисления во внебюджетные фонды	20 740	21 740	21450
6. Затраты на научные и производственные командировки	0	0	0
7. Накладные расходы	12183	16 368	16096
8. Бюджет затрат НТИ	433435	312584	314232

4.4. Определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования

Интегральный финансовый показатель рассчитывается как:

$$I_{\text{финр}}^{\text{исп.}i} = \frac{\Phi_{\text{р},i}}{\Phi_{max}}$$

Используя данные таблицы 10 получаем:

$$I_{\text{финр}}^{\text{исп1}} = 0{,}748$$

$$I_{\text{финр}}^{\text{исп2}} = 1$$

$$I_{\text{финр}}^{\text{исп3}} = 0{,}988$$

Интегральный показатель ресурсоэффективности можно определить следующим образом:

$$I_{\text{р},i} = \sum a_i b_i$$

где $I_{\text{р},i}$ – интегральный показатель ресурсоэффективности для i-го варианта разработки,

a_i – весовой коэффициент i-го варианта разработки,

b_i – бальная оценка i-го варианта исполнения разработки, устанавливаемая экспертным путем по выбранной шкале оценивания,

n – число параметров сравнения.

Расчет интегральных показателей ресурсоэффективности приведен в табл.17:

Таблица 17 – Расчет интегральных показателей ресурсоэффективности

Критерии	Весовой коэф.	Исп.1	Исп.2	Исп.3
Скорость работы	0,3	5	4	2
Универсальность	0,25	5	5	5
Простота эксплуатации	0,1	5	5	3
Потребность в ресурсах памяти	0,1	4	3	3
Функциональная мощность	0,15	5	4	4
Итого:	1	4,9	4,25	3,25

Сравнительная эффективность разработок приведена в табл. 18:

Таблица 18 – Сравнительная эффективность разработок

Показатели	Исп1	Исп2	Исп3
Интегральный финансовый показатель разработки $I_{\text{финр}}$	0,988	1	0,748
Интегральный показатель ресурсоэффективности разработки $I_{\text{р}}$	4,9	4,25	3,25
Интегральный показатель эффективности I	4,959	4,25	4,3449

Сравнительная эффективность вариантов исполнения	1,1668	1	1,02233

Исходя из проведенного анализа, можно отметить, что Исполнение №1 является несколько более предпочтительным, нежели Исполнение №2 и №3. Несмотря на высокую стоимость, исполнение №1 имеет наибольший показатель ресурсоэффективности.

Таким образом, Исполнение №1, реализованное в данной работе, является несколько более дорогим, но и более качественным вариантом реализации проекта [21, 22, 23].

5. СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ

В любой научно-исследовательской и проектной деятельности немаловажную роль занимает такая область как безопасность труда и окружающей среды.

В понятие «социальная ответственность» входит следующее: состояние рабочего места, помещения, режим трудовой деятельности и обеспечение мероприятий по защите трудящихся в моменты чрезвычайных ситуаций регламентируются в соответствии с международным стандартом ICCSR26000:2011 «Социальная ответственность организации» [27]. Целью данного стандарта является принятие проектных решений, исключающих несчастные случаи на производстве и снижение негативных воздействий на окружающую среду.

Согласно данному стандарту такое понятие, как «социальная ответственность», означает ответственность организации за воздействие решений, которые были ею предложены, на общество и окружающую среду.

Раздел, посвященный социальной ответственности организации, включает в себя следующие составляющие: техногенная безопасность, региональная безопасность, организационные мероприятия обеспечения безопасности, особенности законодательного регулирования проектных решений и безопасность в чрезвычайных ситуациях.

Научно-исследовательский проект представляет собой разработку программного продукта в САПР Xilinx Vivado и предполагает большой объем работы с ПК, поэтому важным критерием безопасности является организация рабочего места и режима трудовой деятельности. К опасным факторам труда разработчика-программиста относятся: недостаточная освещенность рабочей зоны, отклонение параметров микроклимата в помещении и уровень шума, а к вредным факторам: излучение электромагнитных полей, электробезопасность и пожарная безопасность [28].

5.1. Характеристика рабочего места

Выпускная квалификационная работа студента выполнялась в десятом корпусе ТПУ на кафедре информационных систем и технологий. Рабочее место находится на четвертом этаже здания и представляет собой комнату длиной – 5 м., шириной – 4 м. и высотой – 3 м. Естественное освещение кабинета осуществляется посредством одного окна размерами 2,2 м. х 1,5 м. Дверь – металлическая, одностворчатая, черного цвета. Высота двери – 2 м., ширина - 1 м. Стены комнаты окрашены водоэмульсионной краской бежевого цвета. Потолок подвесной, плиточный. Пол покрыт линолеумом. Площадь кабинета составляет 20 $м^2$, объем – 60 $м^3$.

Согласно СанПиН 2.2.2/2.4.1340-03 [29], норма площади рабочего места с персональным компьютером составляет 4,5 $м^2$. В рассматриваемой аудитории установлено 4 рабочих места с персональными компьютерами и жидкокристаллическими экранами. Соответственно, на одного человека приходится 5 $м^2$, что соответствует вышеуказанным требованиям.

5.2. Вредные факторы производственной среды

5.2.1. Освещенность рабочей зоны

Рабочее (общее) освещение – это основное освещение, обеспечивающее нормальные условия для нахождения человека в помещении. Под нормальными понимаются условия жизнедеятельности человека, при которых он не напрягает зрение, чтобы выполнить любое действие, для которого данное помещение предназначено. [30].

Освещение в недостаточной степени может привести к напряжению зрения, ослаблению внимания и наступлению преждевременной утомленности. Ослепление, резь в глазах и раздражение могут быть вызваны чрезмерно ярким освещением. Свет на месте труда может создать сильные тени или отблески, а также дезориентировать работающего. Основным документом, регламентирующим нормы освещенности, является СНиП 23-05-95 [30].

Основным показателем качества освещения является освещенность E - поверхностная плотность светового потока. По характеристике зрительной работы труд программиста относится к разряду III подразряду Г (высокой точности), т.е. наименьший размер объекта различения от 0,3 до 0,5 мм (точка) [29]. Это значит, что нормативное значение освещенности рабочего места должно быть 200 лк (СНиП 23-05-95) [30].

Рассчитаем фактическую освещенность рассматриваемой учебной аудитории. Длина и ширина аудитории равны соответственно 5 и 4 м, высота – 3 м. Рассчитаем индекс помещения:

$$i = \frac{S}{h*(A+B)} \quad ,$$

где i – индекс помещения;

S – площадь помещения, м^2;

h – высота помещения, м;

A – длина помещения, м;

B – ширина помещения.

$$i = \frac{20}{3*(5+4)} = 0.7,$$

Исходя из значения индекса помещения, можно определить, что коэффициент использования рассматриваемого светового светильника с люминесцентными лампами равен 30% [31].

Рассчитаем освещенность по формуле, учитывая, что в аудитории 4 светильника по 4 лампы в каждом:

$$E_{\text{факт}} = \frac{N*n*\Phi_{\text{ст}}*\eta}{S*К_з*Z} \quad ,$$

где $E_н$ – фактическая освещенность;

N – число светильников в помещении;

n – число ламп в светильнике;

$\Phi_{ст}$ – величина стандартного светового потока, лм;

η – коэффициент использования светового потока;

S – площадь помещения;

$К_з$ – коэффициент запаса;

Z – коэффициент неравномерности освещения.

Зная, что $Ф_{ст}$ = 1450 лм для люминесцентных ламп дневной света ЛБЦ-30 (СНиП 23-05-95), $К_з$ для помещений с малым выделением пыли равен 1,5, а Z для люминесцентных ламп равен 1 рассчитаем значение фактической освещенности.

$$E_{\text{факт}} = \frac{4*4*1450*0{,}26}{20*1{,}5*1} = 201{,}1 \text{ лк},$$

Рассчитаем численную оценку разности между фактическим значением освещенности и нормативным.

$$\Delta E = \frac{(E_{\text{факт}} - E_{\text{н}})}{E_{\text{н}}} * 100\%,$$

где ΔЕ – показатель разности между фактической освещенностью и нормативной;

$Е_{факт}$ – фактическое значение освещенности;

$Е_н$ – нормативное значение освещенности.

$$\Delta E = \frac{(201{,}1 - 200)}{200} * 100\% = 0{,}5\%$$

Отсюда можно заключить, что в аудитории подходящая система освещения, так как сохраняется допустимое отклонение освещенности в 20% [30].

5.2.2. Производственный шум

Люди, которым приходится работать в условиях длительного шума, обычно имеют головные боли, раздражительность, сталкиваются со снижением памяти, повышенной утомляемостью, также у многих понижен аппетит, есть боли в ушах и т. д. Перечисленные факты снижают производительность, работоспособность человека, а также качество труда [32].

Шумовой фон помещения создают десять одновременно работающих компьютеров. Также возникает шум, исходящий от принтера или телефонных

аппаратов. Также источником шума является система вентиляции или шумы, поступающие извне помещения.

Во избежание негативных последствий от производственного шума, его необходимо регулировать в соответствие с нормами, которые указаны в ГОСТ 12.1.003-83 «ССБТ. Общие требования безопасности» [33].

Допустимые уровни звука и звукового давления для рабочего места разработчика-программиста согласно вышеуказанному ГОСТу 12.1.003-83 представлены в табл. 19.

Таблица 19 – Предельно допустимые уровни звука (ГОСТ 12.1.003-83 [33])

Вид трудовой деятельности/ Частоты	Уровни звука и звукового давления, дБ, в октавных полосах со среднегеометрическими частотами, Гц								
	31,5	63	125	250	500	1000	2000	4000	8000
Научная деятельность, проектирование, программирование, Рабочие места проектно-конструкторских бюро, программистов вычислительных машин и т.д.	86	71	61	54	49	45	42	40	38

Допустимый уровень звукового давления колеблется от 38 дБ до 86 дБ при частоте от 8000 Гц до 31,5 Гц, соответственно.

Для уменьшения воздействий шума можно использовать следующие методы, согласно СНиП 23-03-2003 [34]:

- экранирование рабочих мест, то есть установка перегородок между рабочими местами;
- установка оборудования, производящего минимальный шум.

Для снижения уровня шума, производимого персональными компьютерами, рекомендуется регулярно проводить их техническое обслуживание: чистка от пыли, замена смазывающих веществ; также применяются звукопоглощающие материалы.

5.2.3. Микроклимат помещения

Компьютеры могут привести к увеличению температуры и снижению относительной влажности в помещении. В СанПиН 2.2.4.548-96 установлены величины параметров микроклимата, создающие комфортные условия (табл. 20).

Работа программиста относится к легкой категории 1Б (СанПиН 2.2.4.548 – 96) [35]. В таблицах представлены данные показатели для теплого периода года (плюс 10 °С и выше) и для холодного периода года.

Таблица 20 – Оптимальные величины показателей микроклимата (СанПиН 2.2.4.548 – 96) [35]

Период года	Температура воздуха, °С	Температура поверхностей, °С	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	21-23	20-24	40-60	0,1
Теплый	22-24	21-25		0,1

Таблица 21 – Допустимые величины показателей микроклимата (СанПиН 2.2.4.548 – 96) [35]

Период года	Температура воздуха, °С	Температура поверхностей, °С	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	19-24	18-25	15-75	0,1-0,2
Теплый	20-28	19-29		0,1-0,3

Если температура воздуха отличается от нормальной, то время пребывания в таком помещении должно быть ограничено в зависимости от категории тяжести работ. Температура в рассматриваемом помещении в холодное время года может опускаться до 19-21 °С, а в теплое время года подниматься до 25-28 °С. Данные показатели соответствуют допустимым значениям температуры (табл. 21).

Таблица 22 – Рекомендуемое время работы при температуре воздуха ниже допустимых величин (СанПиН 2.2.4.548 – 96) [35]

Температура воздуха, °С	Время пребывания, не более, ч
17	6
18	7

Таблица 23 – Рекомендуемое время работы при температуре воздуха выше допустимых величин (СанПиН 2.2.4.548 – 96) [34, 35]

Температура воздуха, °С	Время пребывания, не более, ч
30,0	5
29,5	5,5
29,0	6

К мероприятиям по оздоровлению воздушной среды в производственном помещении относятся правильная организация вентиляции и кондиционирования воздуха, отопление помещений. В рассматриваемой аудитории вентиляция осуществляется естественным и механическим путём. В зимнее время в помещении предусматривается система отопления. Это обеспечивает нормальное состояние здоровья работников в аудитории.

5.2.4. Электромагнитное излучение

Электромагнитное излучение - распространяющееся в пространстве возмущение электрических и магнитных полей [36]. Источниками электромагнитного излучения в данном исследовании являются мониторы и системный блок.

Оценка величины уровней ЭМП, проведенная по паспортным данным компьютера и монитора, показала их соответствие нормам ТСО–03 и СанПиН 2.2.2/2.4.1340–03 [26]. В табл. 24 приведены нормы уровня ЭМП, которым соответствует техника в кабинете.

Таблица 24 – Допустимые уровни ЭМП, создаваемых ПК (СанПиН 2.2.2/2.4.1340–03 [26])

Наименование параметров		ВДУ ЭМП
Напряженность электрического поля	в диапазоне частот 5 Гц - 2 кГц	25 В/м
	в диапазоне частот 2 кГц - 400 кГц	2,5 В/м
Плотность магнитного потока	в диапазоне частот 5 Гц - 2 кГц	250 нТл
	в диапазоне частот 2 кГц - 400 кГц	25 нТл
Электростатический потенциал экрана видеомонитора		500 В

Для того, чтобы снизить воздействие таких видов излучения, рекомендуют применять такие мониторы, у которых уровень излучения понижен (MPR-II, TCO-92, TCO-99), а также установить защитные экраны и соблюдать режимы труда и отдыха.

5.3. Опасные факторы производственной среды

5.3.1. Поражение электрическим током

К опасным факторам относят поражение электрическим током согласно ГОСТ 12.0.003-74 [37]. Персональный компьютер питается от сети 220В переменного тока с частотой 50Гц. Помещение с ПЭВМ, где проводились описанные выше работы, относится к помещениям без повышенной опасности (ГОСТ 12.1.019 [39]).

К мероприятиям по предотвращению возможности поражения электрическим током относятся:

- при включенном сетевом напряжении работы на задней панели компьютера должны быть запрещены;
- все работы по устранению неисправностей должен производить квалифицированный персонал;
- необходимо постоянно следить за исправностью электропроводки.

5.3.2. Пожарная безопасность

Также к опасным факторам относится и пожарная безопасность (ГОСТ 12.0.003-74 [37]). Пожарная безопасность осуществляется системой пожарной защиты и системой предотвращения пожара.

По взрыво- и пожароопасности все помещения, согласно техническому регламенту НПБ 105-95 [38], делятся на 5 категорий, в зависимости от применяемых на производстве веществ и их количества. Рассматриваемая учебная аудитория относится к пожароопасной категории В [39].

Основные причины возникновения пожаров:

1. Нарушение правил пожарной безопасности;
2. Перегрузка электросети;
3. Неисправность прибора;
4. Разряд молнии и неисправность молниеотвода.

Для того что бы избежать возникновения пожара необходимо проводить следующие профилактические работы, направленные на устранение возможных источников возникновения пожара:

- периодическая проверка проводки;
- отключение оборудования при покидании рабочего места;
- проведение инструктажа работников о пожаробезопасности.

Для предотвращения пожара в аудитории с ПЭВМ имеется:

- углекислотный огнетушитель типа ОУ-2 (данный тип огнетушителя подходит для помещений с электрооборудованием (ГОСТ Р 51057-01[40]);
- Пожарная сигнализация ДИП-3СУ (извещатель пожарный, дымовой оптико-электронный точечный).

5.4. Региональная безопасность

Воздействие на литосферу предусматривает под собой утилизацию электронной техники: компьютеров, сканеров и т.п. Утилизация такого оборудования является достаточно сложной, так как такие они имеют сложную структуру. Непосредственная переработка большей части компонентов включает

в себя их сортировку, последующую гомогенизацию и отправку для повторного использования, т.е. с предварительным помолом или переплавкой.

При рассмотрении влияния процесса утилизации персонального компьютера были выявлены особо вредные выбросы согласно ГОСТ Р 51768-2001 [41]. В случае выхода из строя компьютеров, они списываются и отправляются на специальный склад, который при необходимости принимает меры по утилизации списанной техники и комплектующих. В настоящее время в Томской области утилизацией занимаются две компании: городской полигон и ООО НПП «Экотом». Утилизацией опасных бытовых отходов занимаются компании: ООО «Торем», ООО «СибМеталлГрупп».

5.5. Безопасность в чрезвычайных ситуациях

Одними из наиболее вероятных и разрушительных видов чрезвычайных ситуаций являются пожар или взрыв на рабочем месте.

Всякий работник при обнаружении пожара должен (ППБ 01-03 [42]):

- незамедлительно сообщить об этом в пожарную охрану;
- принять меры по эвакуации людей, каких-либо материальных ценностей согласно плану эвакуации;
- отключить электроэнергию, приступить к тушению пожара первичными средствами пожаротушения.

При возникновении пожара должна сработать система пожаротушения, передав на пункт пожарной станции сигнал о ЧС. В случае если система не сработала, то необходимо самостоятельно произвести вызов пожарной службы по телефону 101, сообщить точный адрес места возникновения ЧС и ожидать приезда специалистов.

Рабочее место располагается в 10 корпусе ТПУ 408 аудитория. Также представлен план эвакуации четвертого этажа 10 корпуса (приложение 23).

5.6. Правовые и организационные вопросы обеспечения безопасности

Работа программиста связана с постоянной работой за компьютером, следовательно, могут возникать проблемы, связанные со зрением. Также

неправильная рабочая поза может оказывать негативное влияние на здоровье. Таким образом, неправильная организация рабочего места может послужить причиной нарушения здоровья и появлением психологических расстройств.

Согласно СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы»:

- яркость дисплея не должна быть слишком низкой или слишком высокой;
- размеры монитора и символов на дисплее должны быть оптимальными;
- цветовые параметры должны быть отрегулированы таким образом, чтобы не возникало утомления глаз и головной боли.
- опоры для рук не должны мешать работе на клавиатуре;
- верхний край монитора должен находиться на одном уровне с глазом, нижний – примерно на 20° ниже уровня глаза;
- дисплей должен находиться на расстоянии 45-60 см от глаз;
- локтевой сустав при работе с клавиатурой нужно держать под углом 90°;
- каждые 10 минут нужно отводить взгляд от дисплея примерно на 5-10 секунд;
- монитор должен иметь антибликовое покрытие;
- работа за компьютером не должна длиться более 6 часов, при этом необходимо каждые 2 часа делать перерывы по 15-20 минут;
- высота стола и рабочего кресла должны быть комфортными.

Требования к организации рабочего места представлены на рис. 23.

Рисунок 23 – Организация рабочего места.

Для подведения общей картины по выполнению работы можно сделать вывод о том, что помещение удовлетворяет всем необходимым нормам для выполнения работы. Были приведены рекомендации для защиты от возможных угроз для безопасности жизнедеятельности, при соблюдении которых работа будет доведена до логического завершения

ЗАКЛЮЧЕНИЕ

Реализовано устройство межинтерфейсного взаимодействия между ПК и ЦАП. Данное устройство разработано в виде модулей в САПР Xilinx Vivado на языке описания аппаратуры VHDL в специальной среде моделирования и натурно на специально собранном макете.

При проектировании данного устройства был обоснован выбор проектного решения поставленной задачи, выбор используемого языка описания аппаратуры и составлена структурно-функциональная схема двух исполнений устройства.

При разработке данного устройства произошло ознакомление с программным продуктом Xilinx Vivado и средой тестирования Modelsim 10.2c, изучены методики программирования последовательных и параллельных интерфейсов и в итоге было осуществлено программирование модулей интерфейсов, а также протестирована работа данных модулей за счет программирования модулей тестирования (testbench) и произведено моделирование в специальной среде Modelsim.

Также была спроектирована печатная плата с выбранными компонентами: сконфигурированы библиотеки элементов и посадочные места, собрана принципиальная схема, проведена трассировка и компоновка элементов на печатной плате.

Кроме того, во втором исполнении устройства была проведена интеграция IP-ядер в единое устройство, изучен механизм интеграции soft-процессора Microblaze и написан программный код на языке C.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. ПЛИС (FPGA) и микроконтроллер. В чем разница? // Микропрогер [Электронный ресурс]. – URL: http://micro-proger.ru/2016/03/17/plis-fpga-i-mikrokontroller-v-chem-raznica/. (Дата обращения 12.01.2017).
2. Архитектура ПЛИС (FPGA) // Марсоход – open source hardware project [Электронный ресурс]. – URL: http://micro-proger.ru/2016/03/17/plis-fpga-i-mikrokontroller-v-chem-raznica/. (Дата обращения 12.01.2017).
3. Делаем таймер или первый проект на ПЛИС // Geektimes [Электронный ресурс]. – URL: https://geektimes.ru/post/256268/. (Дата обращения 12.01.2017).
4. FPGA/CPLD - ПЛИС (Программируемые Логические Интегральные Схемы) // FPGA-CPLD [Электронный ресурс]. – URL: http://www.fpga-cpld.ru/. (Дата обращения 12.01.2017).
5. ПЛИС // Национальная библиотека им. Н. Э. Баумана Bauman National Library [Электронный ресурс]. – URL: http://ru.bmstu.wiki/%D0%9F%D0%9B%D0%98%D0%A1/. (Дата обращения 12.01.2017).
6. Особенности языков описания архитектуры Verilog и VHDL // PARALLEL.RU [Электронный ресурс]. – URL: https://parallel.ru/fpga/hdl.html. (Дата обращения 13.01.2017).
7. Коротенькое сравнение VHDL и Verilog в помощь начинающим знакомство с ПЛИС // Geektimes [Электронный ресурс]. – URL: https://geektimes.ru/post/254108/. (Дата обращения 13.01.2017).
8. Vivado™ - Новое средство разработки XILINX // CTC INLINE GROUP [Электронный ресурс]. – URL: http://plis.ru/docum/sredstva_ razrabotki_i_ip-yadra_otladochnie_sredstva/vivado_-_novoe_sredstvo_ razra-botki. (Дата обращения 13.01.2017).
9. Проектирование для ПЛИС Xilinx с применением языков высокого уровня в среде Vivado HLS // Компоненты и технологии [Электронный ресурс]. –

URL: http://kit-e.ru/preview/pre_40_12_13_VHLS_Xilinx.php. (Дата обращения 13.01.2017).

10. Знакомство со средой моделирования ModelSim // Цифровая лаборатория FPGA / DSP [Электронный ресурс]. – URL: http://www.fpga.keoa.kpi.ua/ category/fpga/cad-pld/verilog-basics-laboratory-works. (Дата обращения 13.01.2017).
11. Ключев А.О., Ковязина Д.Р., Петров Е.В., Платунов А.Е. Интерфейсы периферийных устройств. – СПб.: СПбГУ ИТМО, 2010.
12. AXI Reference Guide [Электронный ресурс]. – URL: https://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf. (Дата обращения 13.01.2017).
13. AXI4-Lite IP Interface (IPIF) // Xilinx All Programmable [Электронный ресурс]. – URL: https://www.xilinx.com/products/intellectual-property/axi_lite_ipif.html. (Дата обращения 13.01.2017).
14. Овчинников В.А., Васильев А.Н., Лебедев В.В. Проектирование пе- чатных плат: Учебное пособие. 1-е изд. Тверь: ТГТУ, 2005. 116 с.
15. Интерфейс SPI // Microsin.net [Электронный ресурс]. – URL: http://microsin.net/programming/ARM/spi-interface.html. (Дата обращения 13.01.2017).
16. ИМС для цифро-аналоговых преобразователей // ЭКиС [Электронный ресурс]. – URL: http://www.ekis.kiev.ua/UserFiles/Image/pdfArticles/ Bul_AD_EKIS_03_2010.pdf. (Дата обращения 13.01.2017).
17. MCP4922 // Microchip [Электронный ресурс]. – URL: http://www.analog.com/media/en/technical-documentation/data-sheets/AD9122.pdf (Дата обращения 13.01.2017).
18. Самоучитель по P-CAD // P-CAD LIB [Электронный ресурс]. – URL: http://lib.qrz.ru/book/export/html/6676 (Дата обращения 13.01.2017).
19. MicroBlaze - семейство тридцатидвухразрядных микропроцессорных ядер, реализуемых на основе ПЛИС фирмы Xilinx // Рынок микроэлектроники

[Электронный ресурс]. – URL: http://www.compitech.ru/html.cgi/arhiv/03_09/stat_48.htm (Дата обращения 13.01.2017).

20. Волкова Л. Методика проведения SWOT-анализа // http://market.narod.ru/S_StrAn/SWOT.html.
21. Криницына З.В. Ресурсоэффективность отрасли: Учебное пособие / Криницына З.В. – Томск, издательство ТПУ, 2013. – 182 с.
22. Методическая поддержка центров коммерциализации технологий /под ред. А.Бретта, О.Лукши. – М.:ЦИПРА РАН, 2006. – 368 с.
23. Шепеленко Г.И. Экономика, организация и планирование производства на предприятии: Учебное пособие / Г. И. Шепеленко. – 2-е изд., доп. и перераб. – Ростов-на-Дону: МарТ, 2000. – 544 с. – ISBN 5-241-00014-3.
24. Технология оценки бизнеса QuaD // Центр Креативных Технологий [Электронный ресурс]. – URL: https://www.inventech.ru/technologies/quad/ (Дата обращения 13.01.2017).
25. Теория планирования ресурсов в Microsoft Project // Microsoft | Technet [Электронный ресурс]. – URL: https://blogs.technet.microsoft.com/project_ru/2014/09/30/microso/ (Дата обращения 13.01.2017).
26. Бюджет научно-технического исследования (НТИ) // Научная электронная библиотека [Электронный ресурс]. – URL: https://monographies.ru/en/book/section?id=12821 (Дата обращения 13.01.2017).
27. Международный стандарт ICCSR26000:2011 «Социальная ответственность организации»
28. ГОСТ 12.0.003-74 ССБТ. Опасные и вредные производственные факторы. Классификация. - М.: Издательство стандартов, 2001. – 4 с.
29. СанПиН 2.2.2/2.4.1340-03. Гигиенические требования к персональным электронно-вычислительным машинам и организации работы. – М.: Информационно-издательский центр Минздрава России, 2003. – 54 с.
30. СНиП 23-05-95. Естественное и искусственное освещение. – М.: Центр проектной продукции в строительстве, 2011. – 70 с.

31. ГОСТ 6825-91. Лампы люминесцентные трубчатые для общего освещения. – М.: Издательство стандартов, 1992. – 242 с.
32. Борьба с шумом на производстве: Справочник / Е.Я. Юдин, Л.А. Борисов; Под общ. ред. Е.Я. Юдина – М.: Машиностроение, 1985. – 400с.
33. ГОСТ 12.1.003-83. ССБТ. Общие требования безопасности. – М.: Издательство стандартов, 2002. – 13 с.
34. СНиП 23-03-2003. Защита от шума. – М.: Госстрой России, 2004. – 34 с.
35. СанПиН 2.2.4.548 – 96. Гигиенические требования к микроклимату производственных помещений. – М.: Информационно-издательский центр Минздрава России, 1997. – 20 с.
36. ГОСТ 12.1.019-79 ССБТ. Электробезопасность. Общие требования и номенклатура видов защиты – М.: Издательство стандартов, 1979. – 10 с.
37. ГОСТ 12.0.003-74 ССБТ. Опасные и вредные производственные факторы. Классификация. - М.: Издательство стандартов, 2001. – 4 с.
38. НПБ 105-95. Определение категорий помещений и зданий по взрывопожарной и пожарной опасности. / Шебеко Ю.Н. – М.: ВНИИПО, 1998. – 119 с.
39. СП 12.13130.2009. Определение категорий помещений, зданий и наружных установок по взрывопожарной и пожарной опасности. – М.: Проспект, 2010. – 32 с.
40. ГОСТ Р 51057-01. Огнетушители переносные. Общие технические требования. Методы испытаний. – М.: Издательство стандартов, 2001. – 48 с.
41. ГОСТ Р 51768-2001. Ресурсосбережение. Обращение с отходами. Методика определения ртути в ртутьсодержащих отходах. Общие требования. – М: Издательство стандартов, 2001. - 13 с.
42. ППБ 01-03. Правила пожарной безопасности в Российской Федерации. – М.: ФГУ ВНИИПО МЧС России, 2003. – 111 с.

СПИСОК ОСНОВНЫХ ПУБЛИКАЦИЙ

1) Старшинов В.С., Найбауэр Д.Ю. Детектор скрытой проводки [Электронный ресурс] // Ресурсоэффективным технологиям - энергию и энтузиазм молодых: сборник докладов V Всероссийской конференции студентов Элитного технического образования, Томск, 25-27 марта 2014. - Томск: ТПУ, 2014 - С. 164-165. - Режим доступа: http://portal.tpu.ru/science/konf/eto_conf/%D1%81%D0%B1%D0%BE%D1%80%D0%BD%D0%B8%D0%BA%20%D0%BA%D0%BE%D0%BD%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D0%B8%D0%B8%202014.pdf
2) Starshinov V. S. Robotic model car: theory and practice of construction and application // Лингвистика и межкультурная коммуникация: теория и методологические проблемы современного образования (для студентов, аспирантов и молодых ученых): сборник трудов II Российской научно-практической конференции с международным участием, Томск, 14-16 Мая 2014. - Томск: ТПУ, 2014 - Т. 1 - С. 72-75
3) Солнечное окно / А. К. Фомичев, В.С. Старшинов, И.А.Кремлев, И.С.Казакявичюс // Архитекторы будущего: сборник научных трудов Всероссийской научной школы по инженерному изобретательству, проектированию и разработке инноваций, 14-16 ноября 2014 г., г. Томск. — Томск : Изд-во ТПУ, 2014. — [С. 38-39].
4) Старшинов В.С. Проектирование цифровых устройств на программируемых логических интегральных схемах // Молодежь и современные информационные технологии: сборник трудов XIII Международной научно- практической конференции студентов, аспирантов и молодых ученых, Томск, 9-13 Ноября 2015. - Томск: ТПУ, 2016 - Т. 2 - С. 123-124.
5) Старшинов В. С. Возможность эксплуатации платы NI ELVIS II на базе Xilinx Spartan 3e в процессе обучения без использования Labview

[Электронный ресурс] // Молодежь и современные информационные технологии: сборник трудов XIV Международной научно-практической конференции студентов, аспирантов и молодых ученых: в 2 т., Томск, 7-11 Ноября 2016. - Томск: ТПУ, 2017 - Т. 1 - С. 24-25. - Режим доступа: http://portal.tpu.ru:7777/f_ic/files/science/activities/msit/msit2016/Sbornik_2016/Sbornik_MSIT_2016(Tom1).pdf

ПРИЛОЖЕНИЕ 1. Отладочная плата NI Digital Electronics FPGA Board

Любая конвертация данных приводит к потерям логическим ячеек, а для создания более гибких и занимающих меньше ячеек используются эквивалентные логические вентили, которые содержатся на плате NI Digital Electronics FPGA Board.

Данная плата является универсальным решением, поскольку на нем можно проектировать цифровые, аналоговые устройства и дополнительную обвязку за счет имеющихся на ней макетных плат.

Одним из главных компонентов платы NI Digital Electronics FPGA Board является ПЛИС Xilinx Spartan 3E (рис. 24).

Рисунок 24 – ПЛИС Xilinx Spartan 3E

Чип Xilinx Spartan 3E содержат:

- 500 000 системных функций
- 10,476 эквивалентных логических ячеек
- 73 000 распределенных битов памяти (массивы статической памяти, хранящие таблицы истинности)
- 360 000 блочных битов памяти
- 20 выделенных мультиплексоров
- 4 блока тактовой частоты
- 158 максимально используемых I/O (входов/выходов)
- 65 максимально дифференцирующих I/O пар

Данное ядро активно используется при разработке ввиду высоких технических характеристик и относительно невысокой стоимости, а также благодаря поддержи почти всех версий среды разработки Xilinx ISE и Vivado.

NI Digital Electronics FPGA Board содержит шесть каналов для ввода аналоговых входных сигналов AI0 – AI5, а также 32 цифровые входные (выходные) линии общего пользования – GPIO31. Внешний вид платы NI Digital Electronics FPGA Board (Elvis II) представлен на рисунке 25.

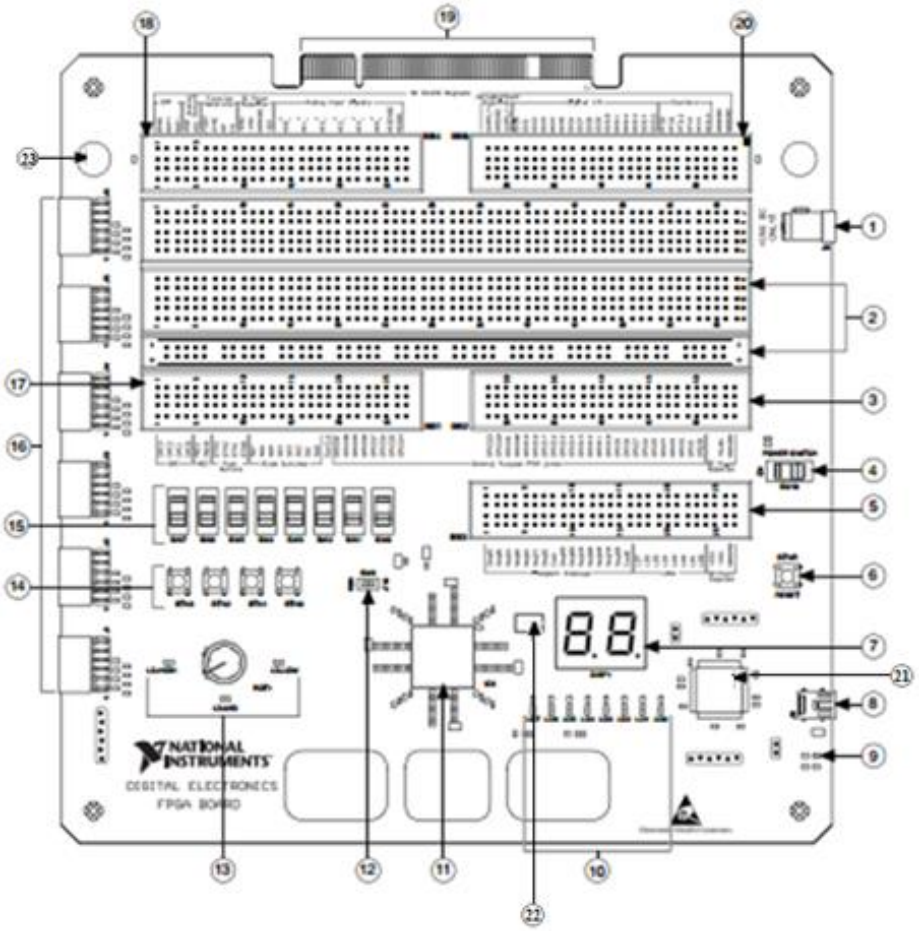


Рисунок 25 – Внешний вид платы NI Digital Electronics FPGA Board

Плата NI Digital Electronics FPGA Board включает в себя:

1 – разъем питания (15В) ;

2 – макетной платы для возможности создания дополнительной обвязки;

3 – блок макетной платы ВВ2;

4 – выключатель питания;

5 – блок макетной платы ВВ3;

6 – кнопка сброса;

7 – семисегментные индикаторы;

8 – USB-соединитель;

9 – LD-G-светодиод;

10 – светодиоды;

11 – FPGA Xilinx Spartan 3E;

12 – переключатель прошивки ПЛИС через ROM/JTAG ;

13 – датчик угла поворота с кнопкой;

14 – кнопки;

15 – движковые переключатели;

16 – 6 коннекторов типа PMOD (2x6);

17 – блок макетной платы BB1;

18 – блок макетной платы BB4;

19 – (NI ELVIS)-соединитель;

20 – блок макетной платы BB5;

21 – программатор;

22 – кварцевый генератор 50МГц;

23 – защита платы от статического электричества.

Описание сигналов, подаваемых на сигнальные зоны макетирования:

- BB1 – зона макетирования для подключения к ЦАП, АЦП, кнопкам, движковым переключателям, внешнему синхросигналу и линиям общего назначения ПЛИС.
- BB2 - зона макетирования для подключения к линиям общего назначения ПЛИС и источникам питания.
- BB3 - зона макетирования для управления семисегментным индикатором на два знакоместа, светодиодами, подключения к источникам питания
- BB4 - зона макетирования для подключения к сигналам NI ELVIS, включая аналоговые входные сигналы, аналоговые выходные сигналы,

функциональные генераторы, источники питания, цифровые входные/выходные сигналы.

- ВВ5 – зона макетирования для подключения к сигналам NI ELVIS, включая регулируемые источники питания, цифровые входные/выходные сигналы, выходные сигналы счётчиков, общие точки сигналов [1,2].

Плата NI Digital Electronics FPGA Board поддерживает два режима работы:

1) Автономный (обращение с помощью ПК к подразделу Автономный режим);

2) Режим NI Elvis – плата используется в качестве дополнительной макетной платы к рабочей станции NI Elvis (режим NI Elvis) [2].

ПРИЛОЖЕНИЕ 2. Тестирование устройства и отдельных блоков с описанием процессов обмена данными в используемых интерфейсах

Для передачи данных обычно используются 2 устройства: ведущее (master) и ведомое (slave). Данные могут передаваться в обоих направлениях между ведущим и ведомым устройствами одновременно.

Согласно методике тестирования устройства, на устройство были поданы 4 набора данных. Для того, чтобы удостовериться в правильности работы устройства, необходимо проверить работу каждого блока/модуля устройства. Для проверки работы блока в программе Modelsim 10.2c надо создать директорию библиотеки work, в которую будут загружаться файлы для тестирования. Кроме того, можно написать программу тестирования testbench для каждого модуля, в этом случае программисту не нужно устанавливать значения сигналов в нужное значение (программа делает это автоматически). При тестировании каждого блока 2 тестовых набора будут тестироваться загрузкой устройств модулей, 2 других будут тестироваться с помощью testbench.

Тестирование блока AXI_SLAVE

При глобальном сбросе устройства происходит обнуление переменных и буферов, в которых хранятся данные и адрес (обнуление значений сигналов WDATA и AWADDR, а также связанных с ними значений). По установке сигнала S_AXI_RESETN в высокое состояние и по установке сигналов S_AXI_AWVALID = ‘1’ и S_AXI_WVALID = ‘1’ должна происходить запись сигналов WDATA и AWADDR в буфер address и value соответственно. При этом автомат проходит состояния записи и передачи данных write_state и response. При передаче данных в состоянии response устанавливаются сигналы AWREARY и WREADY в высокое состояние, что говорит о готовности записать данные. При переходе из состояния записи в состояние отклика данные из буфера value достаются и передаются на O_AXI_RX_DATA и устанавливается флаг о готовности передать данные в следующий блок и обнуляются сигналы AWREADY и WREADY, что говорит о невозможности записать данные.

По окончанию готовности записи данных AWREADY происходит сброс адреса и данных. По установке сигнала ARREADY в высокое состояние происходит считывание адреса и данных и передача их на выходы ARADDR и RDATA соответственно.

По установлению сигнала S_AXI_ARVALID = ‘1’ считывание данных S_AXI_WDATA. После окончания состояния read_state данные записываются в S_AXI_RDATA.

Результаты тестирования блока AXI_SLAVE всеми тестовыми наборами приведены на рисунках 26 - 29 (для последних 2 тестовых наборов написана программа testbench – tb_axi_slave).

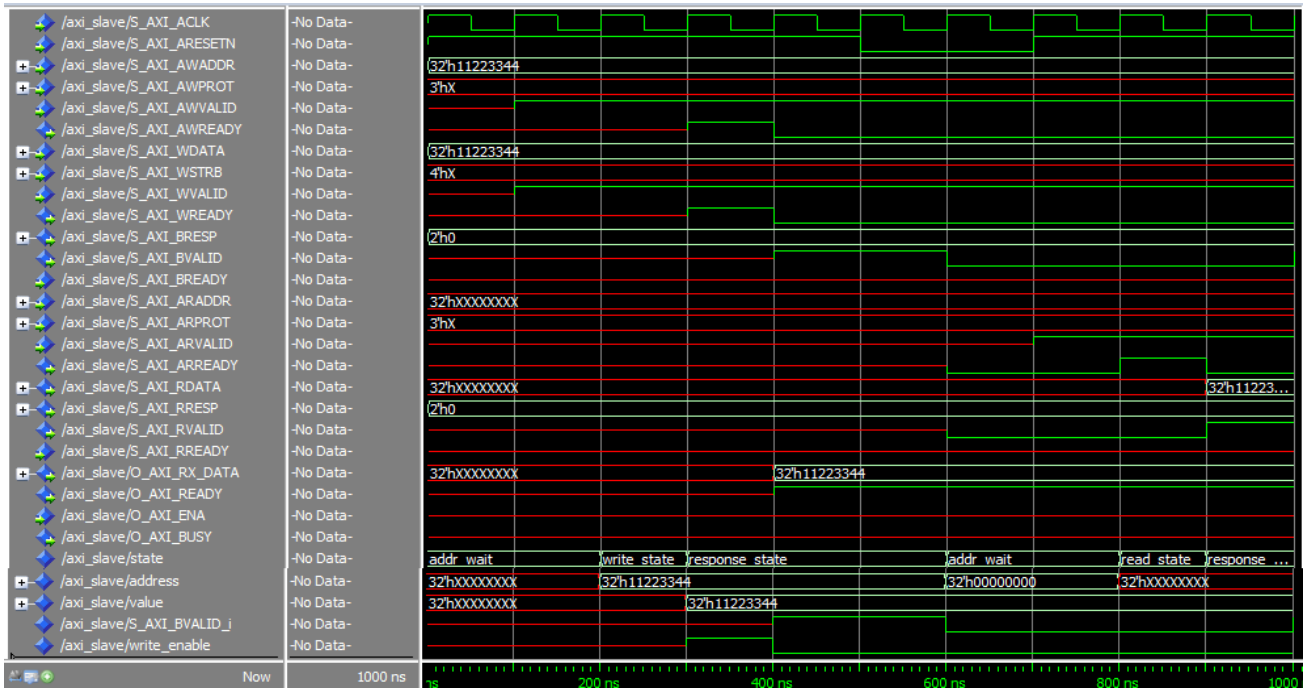


Рисунок 26 – Проверка работы блока AXI_SLAVE первым тестовым набором

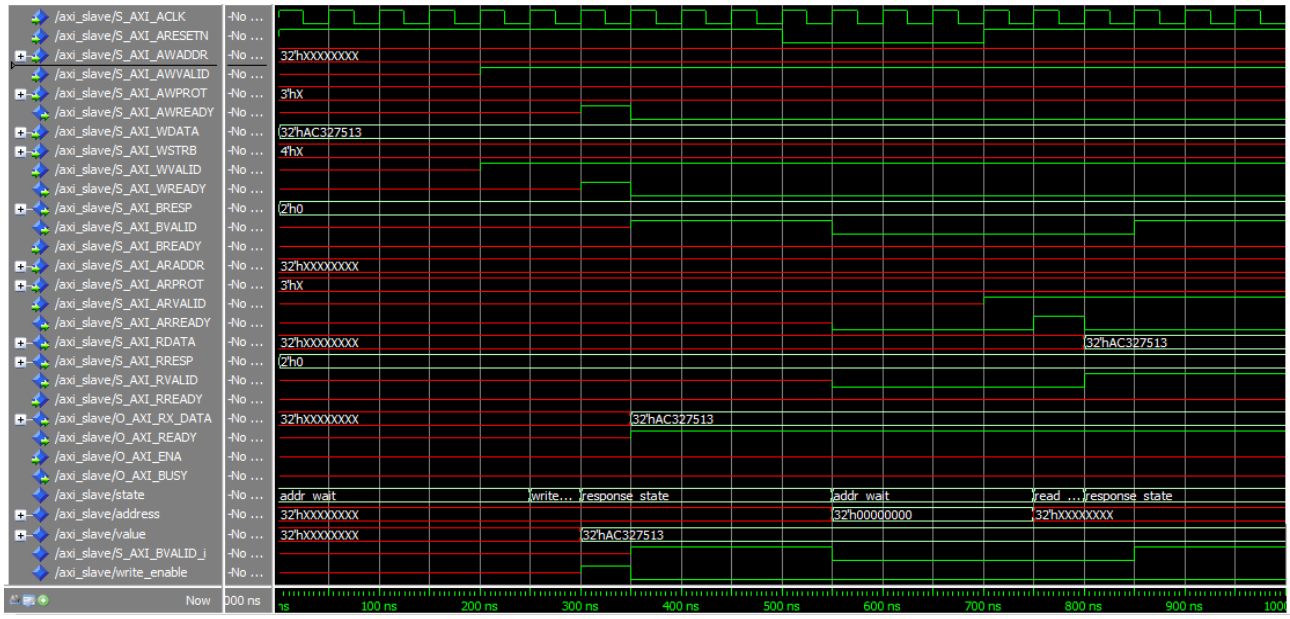


Рисунок 27 – Проверка работы блока AXI_SLAVE вторым тестовым набором

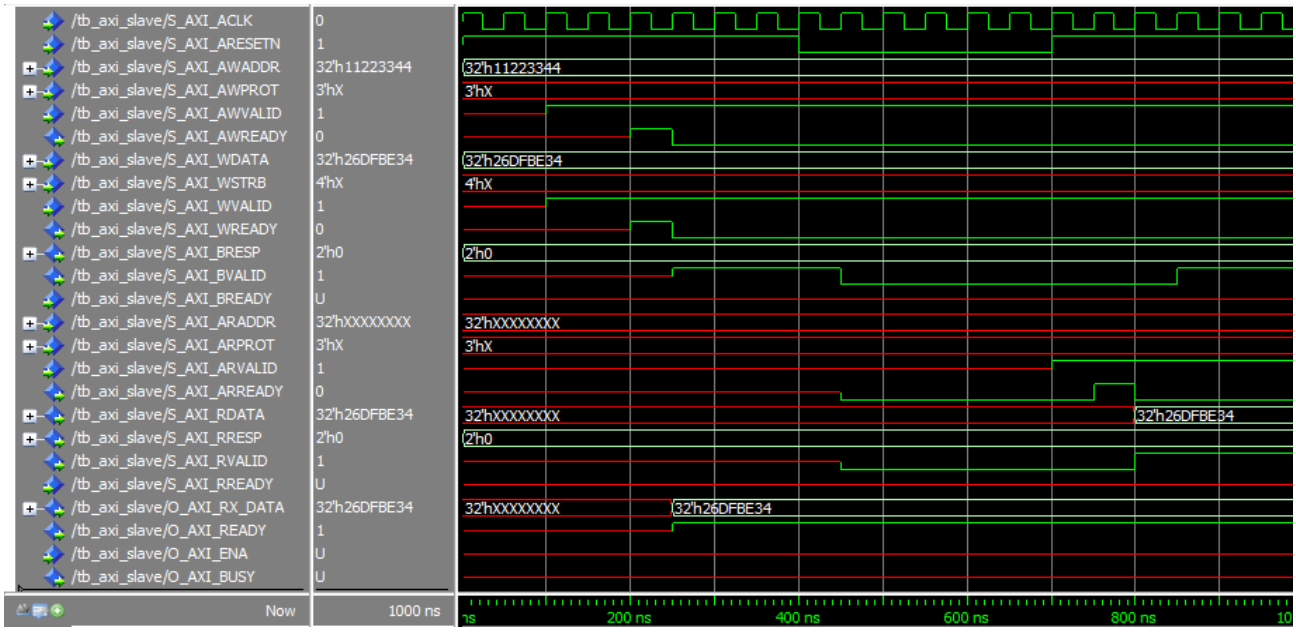


Рисунок 28 – Проверка работы блока AXI_SLAVE третьим тестовым набором

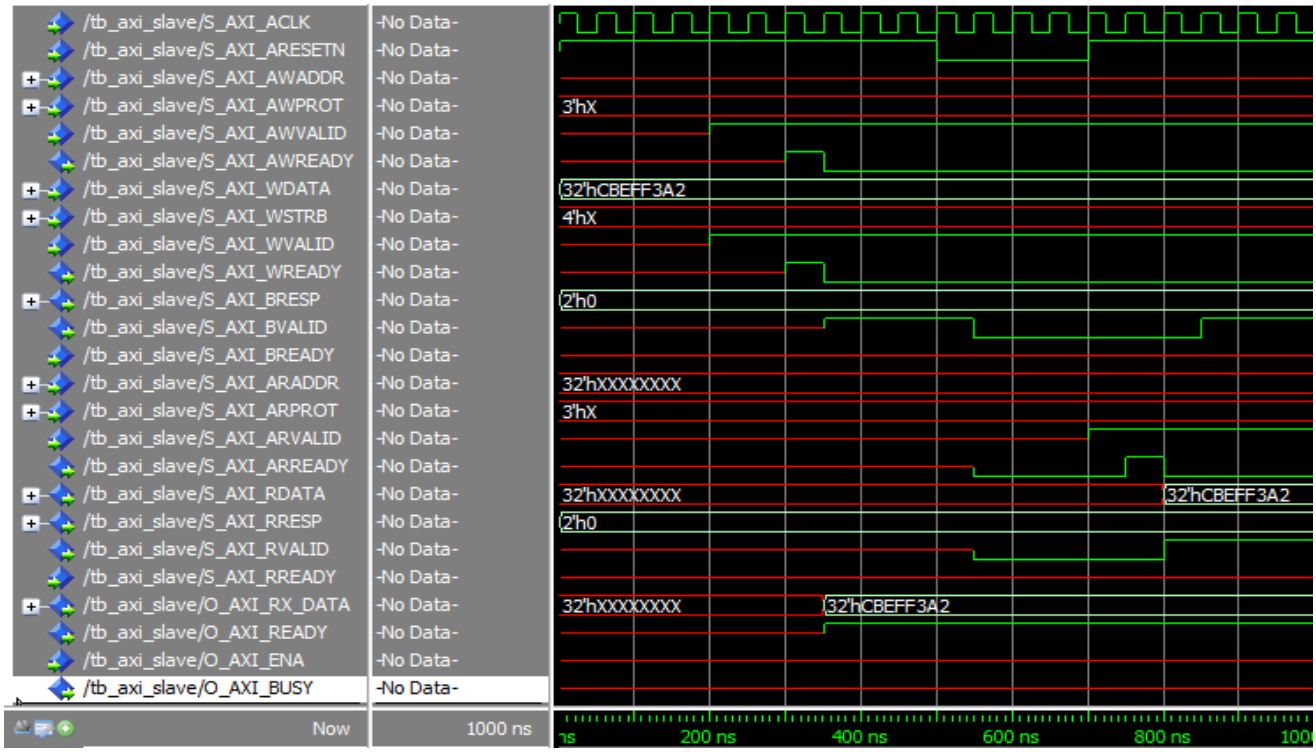


Рисунок 29 – Проверка работы блока AXI_SLAVE четвертым тестовым набором

Приведенная работа тестирования блока AXI_SLAVE четырьмя наборами данных верно отражает его работу на практике, данные, попадаемые на вход S_AXI_WDATA передаются на выход O_AXI_RX_DATA.

Тестирование блока AXI_TO_SPI

Для проверки блока AXI_TO_SPI следует принять, что блоком приняты данные на входе I_AXI_RX_DATA с сигнала выхода O_AXI_RX_DATA предыдущего блока AXI_SLAVE. По глобальному сбросу происходит переход в начальное состояние автомата и не происходит записывание входных данных в регистр. По установлению сигнала RESETN в высокое состояние автомат переходит в следующее состояние axi_tx, где устанавливается флаг axi_rrdy_load, сигнализирующий о возможности загрузки данных и установление сигнала axi_busy, свидетельствующем о работе блока. Соответственно, происходит записывание данных в буфер message для дальнейшей работы. При переходе в состояние tx_spi происходит разбиение данных в буфере на 6 частей (2 команды чтения/записи O_SPI_RW1 и O_SPI_RW2, 2 порции инструкций по 7 бит каждая

порция O_SPI_TX_CMD1 и O_SPI_TX_CMD2, 2 порции данных по 1 байту каждая порция O_SPI_TX_DATA1 и O_SPI_TX_DATA2). Вначале передается 1 команда чтения/записи и 1 порция инструкций, потом 1 порция данных, далее в таком же порядке 2 порция. Мультиплексируются данные с помощью счетчика занятости spi_busy_cnt, который производит счет по установлению сигнала O_SPI_ENABLE в высокое состояние. Как только передается 2 порция данных устанавливается флаг TX_DONE в высокое состояние, что сигнализирует о переходе в состояние spi_load. Данное состояние фиксирует завершение передачи данных из буфера и разрешает пересылку данных в следующий блок. Переход в состояние ready переходит автоматически.

Проверим разбитые данные: изначально были 32-битные данные 32’h11223344. Учитывая, что под порцию инструкций или данных отводится 1 байт, то 32-битные данные будут разбиты на 1 порцию команд 8’h11, 1 порцию данных 8’h22, 2 порцию команд 8’h33 и 2 порцию данных 8’h44. Ввиду того, что в команде старшим битом является команда чтения/записи, разработчиком было принято решение о выносе этой команды отдельно. При переводе в бинарную систему счисления старшими битами команд 8’h11 и 8’h33 являются нулями, то задается режим чтения, а команды не меняют своего значения при выносе данного бита.

Таблица 25 – Полученные наборы данных

S_AXI_WDATA	RW1	CMD1	DATA1	RW2	CMD2	DATA2
11223344	0	11_{16} 0010001_2	22_{16} 00100010_2	0	33_{16} 0110011_2	44_{16} 01000100_2
AC327513	1	$2C_{16}$ 0100110_2	32_{16} 00110010_2	0	75_{16} 1110101_2	13_{16} 00010011_2
26DFBE34	0	26_{16} 0100110_2	DF_{16} 11011111_2	1	$3E_{16}$ 0111110_2	34_{16} 00110100_2
CBEFF3A2	1	$4B_{16}$ 1001011_2	EF_{16} 11101111_2	1	73_{16} 1110011_2	$A2_{16}$ 10100010_2

Результаты тестирования блока AXI_TO_SPI приведены на рисунках 30-33 (для последних 2 тестовых наборов написана программа testbench – tb_axi_to_spi).

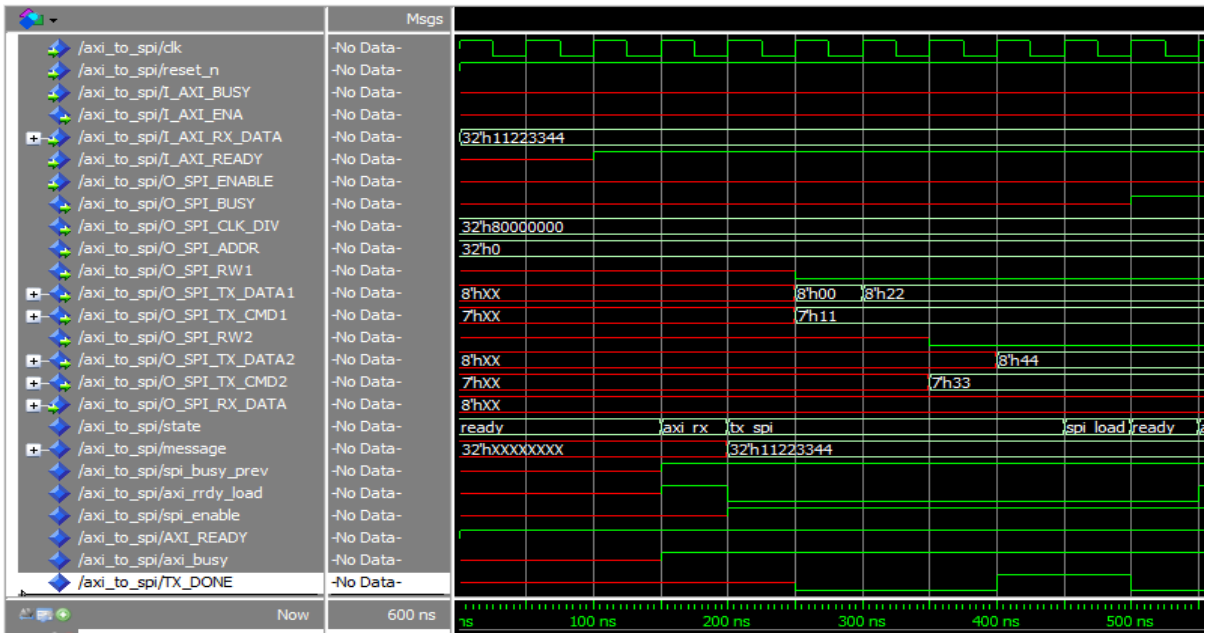


Рисунок 30 – Проверка работы блока AXI_TO_SPI 1 тестовым набором

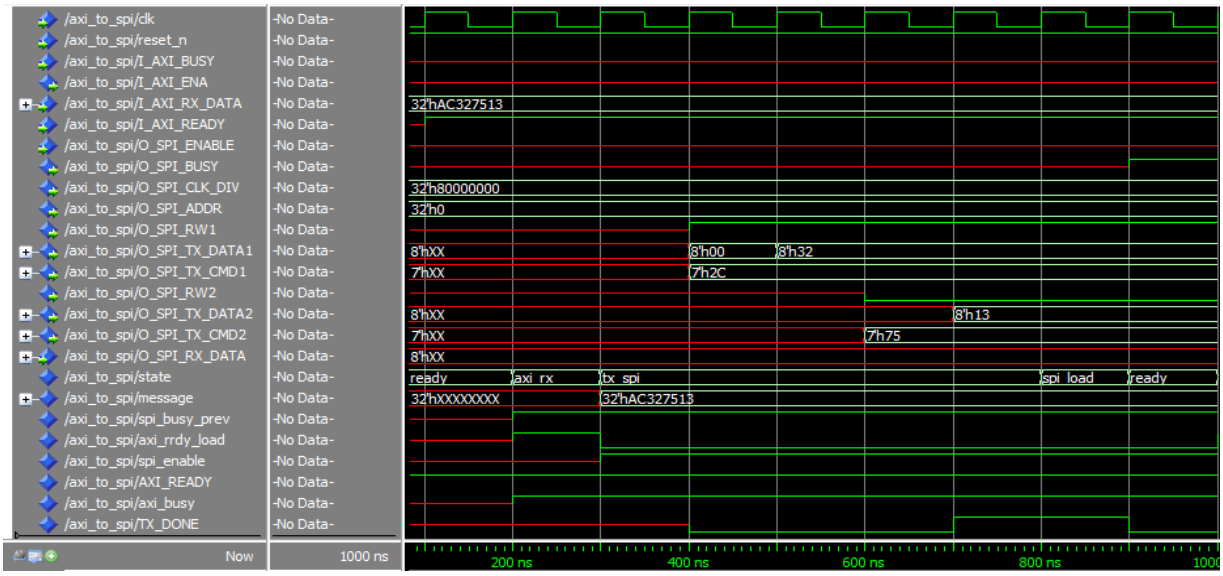


Рисунок 31 – Проверка работы блока AXI_TO_SPI 2 тестовым набором

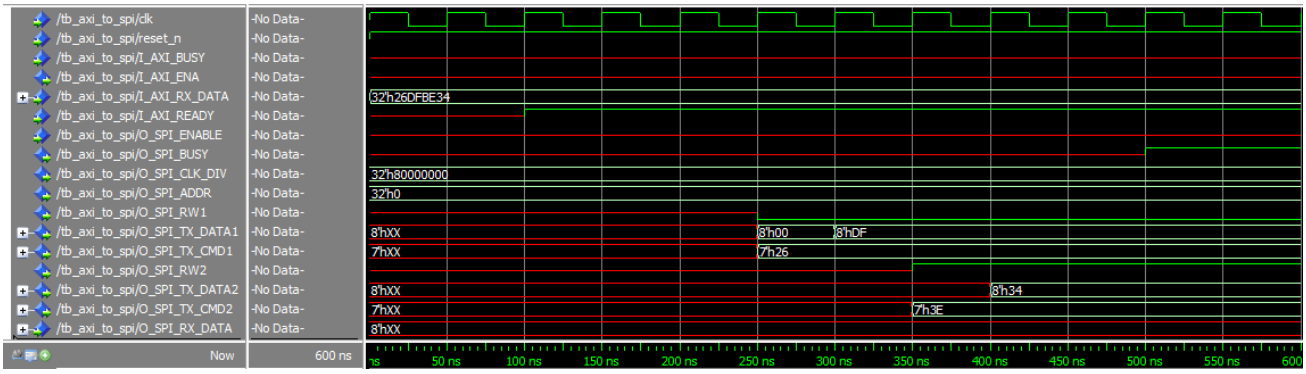


Рисунок 32 – Проверка работы блока AXI_TO_SPI 3 тестовым набором

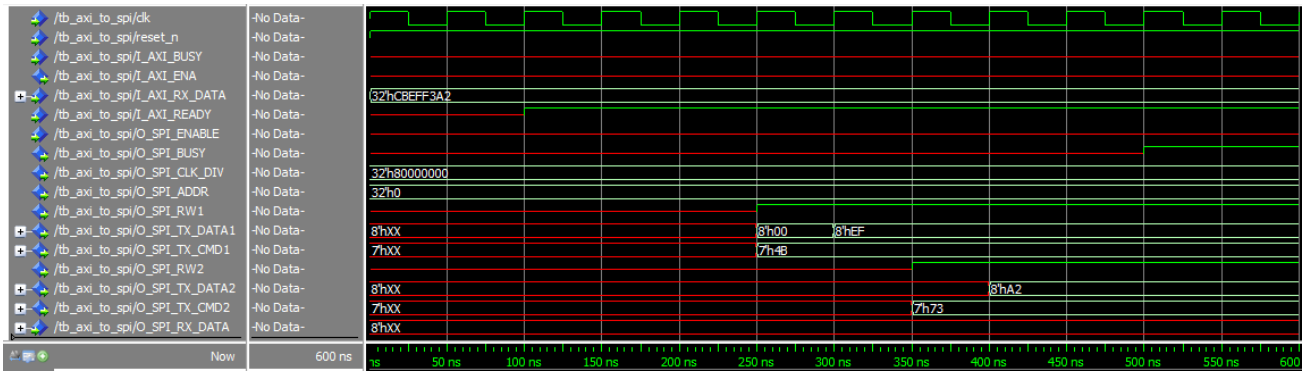


Рисунок 33 – Проверка работы блока AXI_TO_SPI 4 тестовым набором

Приведенная работа блока тестирования AXI_TO_SPI верно отражает его работу на практике, следовательно, блок работает верно.

Тестирование блока SPI_MASTER

Для тестирования данного блока следует принять во внимание входные параметры, полученные при тестировании предыдущего блока, а именно rw1 = ‘0’ и rw2 = ‘0’, tx_cmd1 = 7’h11, tx_cmd2 = 7’h33, tx_data1 = 8’h22, tx_data2 = 8’h44. В начальном состоянии автомата значение линии передачи по последовательному интерфейсу данных SDIO находится в высокоимпедансном состоянии (состоянии отображено синей линией). По техническому заданию команды по SPI должны передаваться с частотой 25 МГц. Поэтому на тактовый сигнал частоты clock приходит сигнал с выхода делителя частоты divider_freq. Для перехода в состояние готовности передачи необходимо установить сигнал busy в высокое состояние. Сигнал sbusy тоже устанавливается в высокое состояние для того, чтобы показать, что загружаются данные в буфер. Сигнал senable показывает

разрешению на загрузку в буфер, поэтому значение тоже устанавливается в высокое состояние. При переходе на следующее состояние cmd_tx происходит запись команд в соответствующие буфера команд (rw1 и rw2 в буфера rw_buffer1 и rw_buffer2 соответственно, tx_cmd1 и tx_cmd2 в буфера cmd_buffer1 и cmd_buffer2 соответственно). Как только произошла запись команд в буфера автомат переходит в следующее состояние data_tx, где происходит запись данных в буфера (tx_data1 и tx_data2 в d_buffer1 и d_buffer2 соответственно). Как только произошли записи команд и данных в буфера автомат переходит в состояние transfer, где происходит побитовая передача данных по последовательному каналу SPI через линию SDIO. Запись происходит следующим образом: сначала записываются старшие 2 байта: команда чтения/записи, 7 бит инструкций и байт данных, далее записываются оставшиеся 2 байта аналогично. Запись происходит старшими битами вперед.

Перед началом передачи устанавливается ss_n в нулевое положение, что позволяет разрешить передачу данных на ведомое устройство. При передаче 1 порции данных передаваемые данные 8’h22 записываются в rx_data, что говорит о завершении передачи 1 порции. Аналогично при передаче 2 порции 8’h44 записываются в rx_data. После завершения передачи устанавливается флаг TX_DONE в высокое состояние, что говорит об окончании транзакции.

Также необходимо провести проверку передаваемых данных по последовательному каналу. Так как было сказано, в каком порядке передавать данные, была определена правильная последовательность выводимых сигналов в последовательном виде:

0/0010001/00100010/0/0110011/01000100 , где

- 7’h11 = 2’b0010001;
- 8’h22 = 2’b00100010;
- 7’h33 = 2’b0110011;
- 8’h44 = 2’b01000100.

Автомат переходит в состояние tr_end, которое устанавливает все единицы на выход ss_n для недопущения передачи с последующим сбросом устройства и линию SDIO в высокоимпедансное состояние.

Результаты тестирования приведены в таблице:

Таблица 26 – Полученные наборы данных

S_AXI_WDATA	SDIO (RW1/CMD1/DATA1/RW2/CMD2/DATA2)
11223344	0/0010001/00100010/0/0110011/01000100
AC327513	1/0100110/00110010/0/1110101/00010011
26DFBE34	0/0100110/11011111/1/0111110/00110100
CBEFF3A2	1/1001011/11101111/1/1110011/10100010

Результаты тестирования блока SPI_MASTER приведены на рисунках 34-37 (с применением программы testbench).

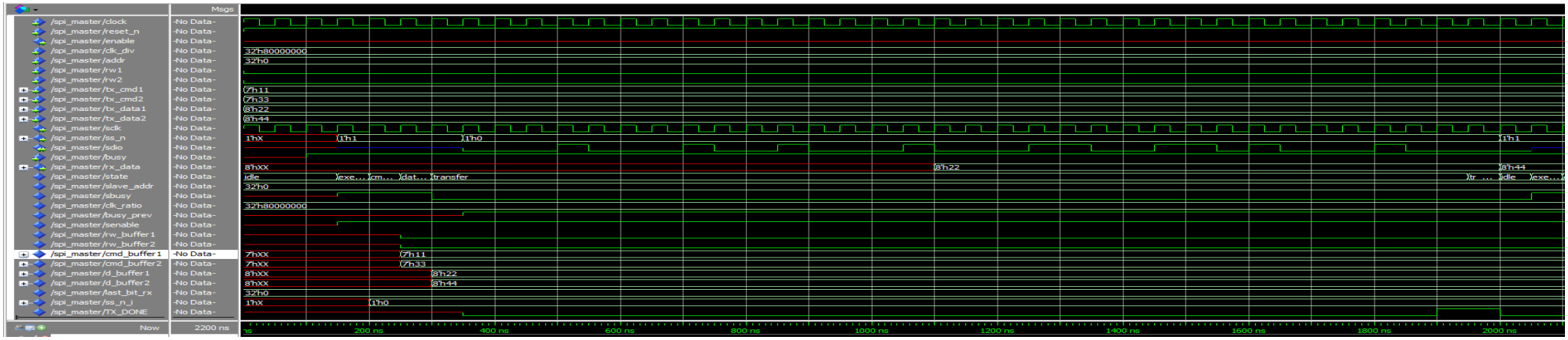


Рисунок 34 – Проверка работы блока SPI_MASTER 1 набором данных

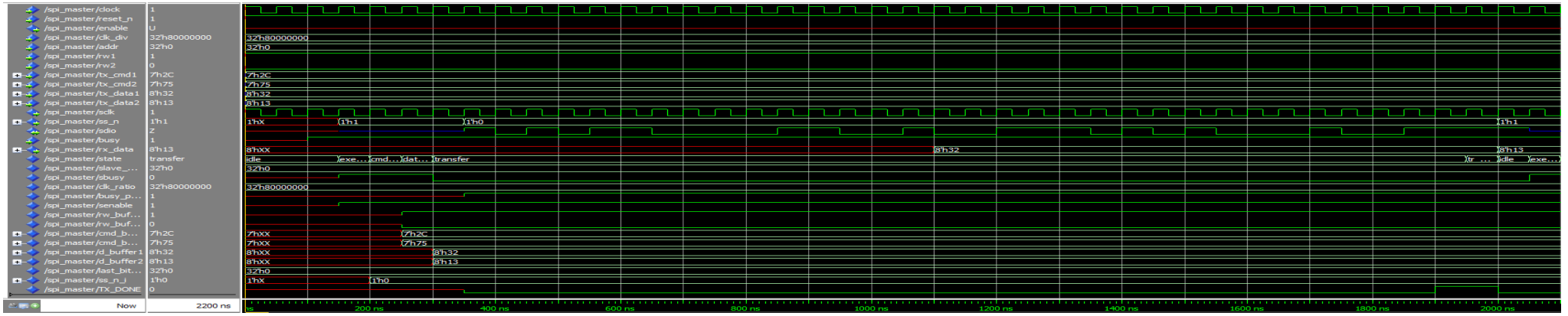


Рисунок 35 – Проверка работы блока SPI_MASTER 2 набором данных

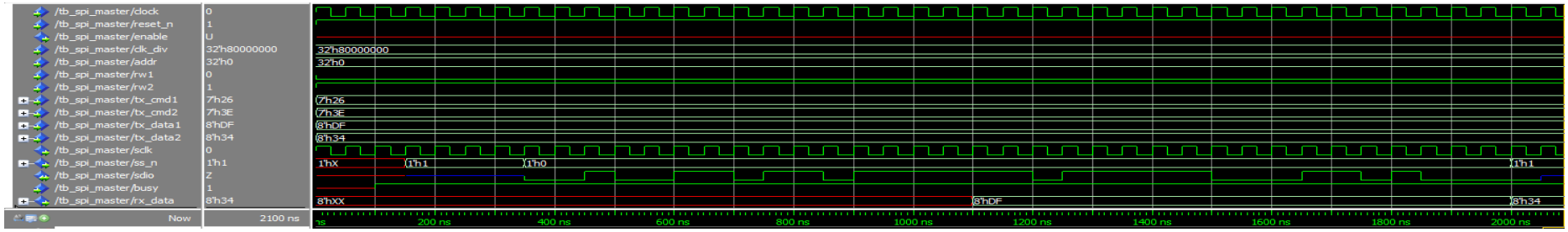


Рисунок 36 – Проверка работы блока SPI_MASTER 3 набором данных

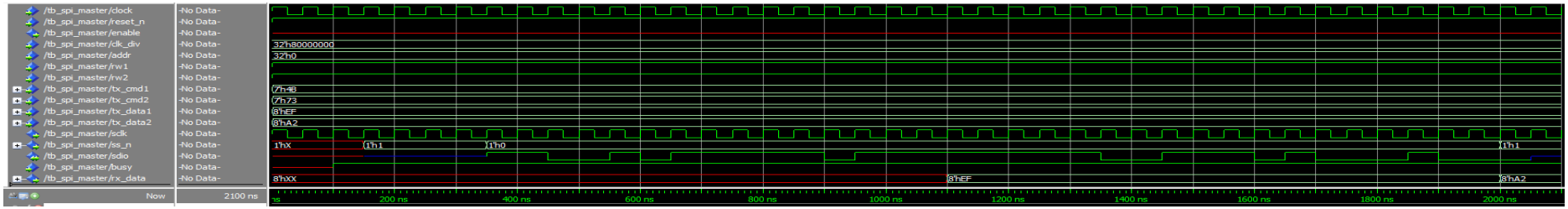


Рисунок 37 – Проверка работы блока SPI_MASTER 4 набором данных

Приведенная работа блока тестирования SPI_MASTER верно отражает его работу на практике, следовательно, блок работает верно.

Тестирование блока DIVIDER_FREQ

В данном блоке особая проверка не требуется. Необходимо лишь убедиться, что данный блок уменьшает тактовую частоту в 10 раз. Соответственно, период выходного сигнала CLK_OUT должен быть больше периода входного сигнала CLK_IN в 10 раз. Было выставлено значение периода CLK_IN равным 20нс. Как видно на рисунке 38, период CLK_OUT составляет 200нс, следовательно, делитель частоты работает верно.

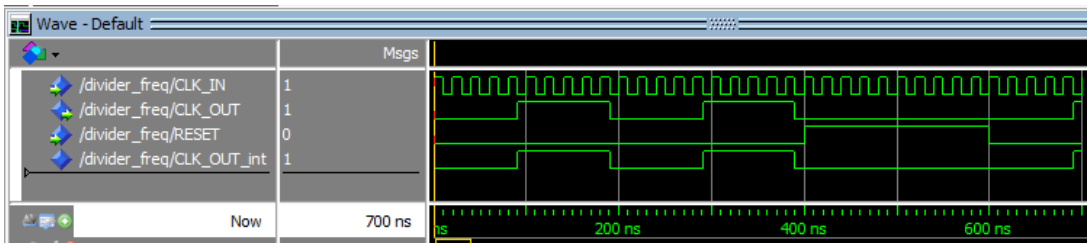


Рисунок 38 – Проверка работы блока DIVIDER_FREQ

Приведенная работа блока тестирования DIVIDER_FREQ верно отражает его работу на практике, следовательно, блок работает верно.

Тестирование устройства AXI_TO_SPI_DEVICE

В данном модуле собраны и соединены все блоки, участвующие в работе. Для проверки необходимо подать тактовую частоту 50МГц на тактовый сигнал S_AXI_ACLK, значение S_AXI_AWADDR не имеет никакого значения. При переключении движкового переключателя SW2 и нажатию кнопки BTN в S_AXI_WDATA с помощью мультиплексора приходит значение 32’h26DFBE34. По сигналам разрешения записи AWVALID и WVALID происходит дальнейшая передача. Как видно из рисунка 39, данные передаются на сигналы соединения AXI_RX_DATA_connection, далее они разбиваются на команды и данные, а после передаются по SDIO. Таким образом, организовано преобразование параллельных данных в последовательные.

AXI_RX_DATA_connection разбивается на RW1_connection = ‘0’, TX_CMD1_connection = 7’h26, TX_DATA1_connection = 8’hDF, RW2_connection = ‘1’, TX_CMD2_connection = 7’h3E, TX_DATA2_connection = 8’h34.

Так как было сказано, в каком порядке передавать данные, была определена правильная последовательность выводимых сигналов в последовательном виде:

0/0100110/11011111/1/0111110/00110100, где

- 7’h26 = 2’b0100110;
- 8’hDF = 2’b11011111;
- 7’h3E = 2’b0111110;
- 8’h34 = 2’b00110100.

Результат полного тестирования блока AXI_TO_SPI_DEVICE приведен на рисунке 39.

Рисунок 39 – Проверка работы устройства AXI_TO_SPI_DEVICE тестовым набором 26DFBE34

Устройство работает верно, так как сигнал SDIO на временных диаграммах полностью совпадают с расчетными последовательностями.

ПРИЛОЖЕНИЕ 3. Апробация устройства в качестве генератора синусоидального сигнала

В предыдущем приложении было проведено тестирование устройства программным способом, в ходе которого была подтверждена корректность работы устройства. Для того, чтобы протестировать устройство на железе, необходимо подобрать компоненты для проведения апробации. В качестве ПЛИС будет использоваться Xilinx Spartan 3E, встроенный в плату NI Electronics FPGA Board. В данной плате имеется макетная плата, в которую можно поставить другие микросхемы и собрать схему устройства.

Программу устройства необходимо прошить с помощью программных средств разработки Xilinx ISE и Xilinx IMPACT. Дополнительную память в данном устройстве использовать не имеет смысла, так как выбранные тестовые наборы не занимают большого объема. В качестве ЦАП будет использоваться MCP4922. С выхода ЦАП с помощью осциллографа будет сниматься синусоидальный сигнал в соотвествии с подаваемым на него тестовым набором.

Плата NI Electronics FPGA Board с собранной схемой устройства на макетной плате приведена на рисунках 40 и 41. Щуп питания осциллографа надо подвести к выходу ЦАП Vouta, а щуп «земли» к «земле». Для того, чтобы подать опорное напряжение, был собран резистивный делитель напряжения.

Тестовые наборы конфигурируются переключением движковых переключателей: 1 – на SW0, 2 – на SW1, 3 – на SW2, 4 – на SW3. По нажатию на клавишу BTN0 происходит пересылка тестового набора в блок AXI_SLAVE для получения последовательного набора данных. Последовательные данные в цифровом виде побитово передаются на ЦАП, который преобразует их в синусоидальный сигнал.

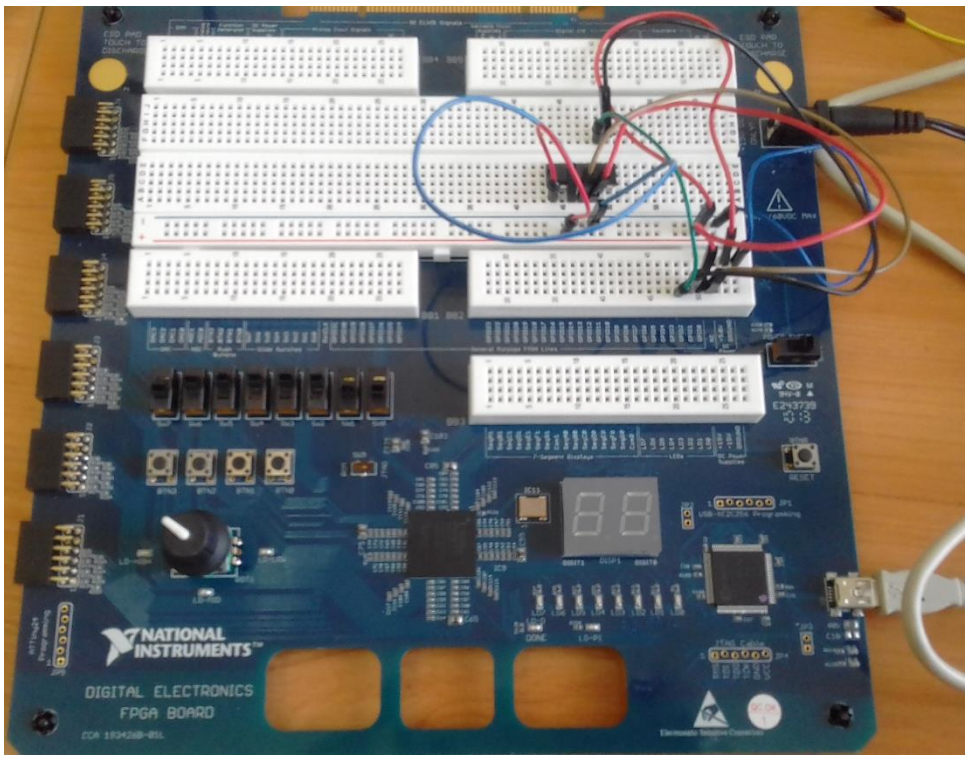


Рисунок 40 – Плата NI Digital Electronics Board

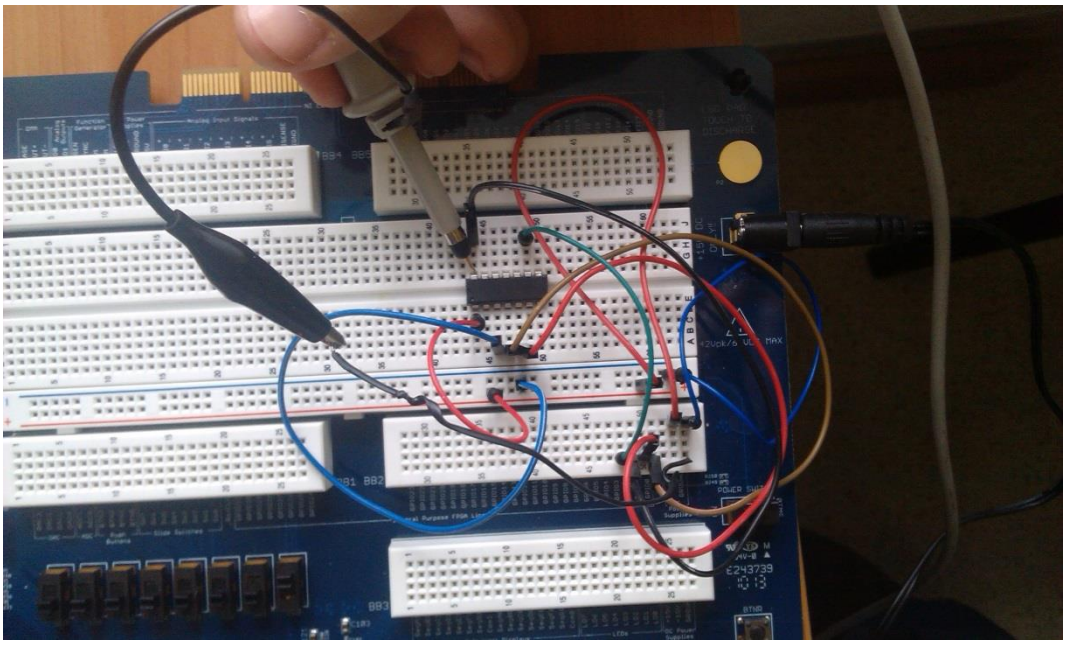


Рисунок 41 – Собранная схема устройства

Подача 1 тестового набора (11223344) осуществляется переключением движкового переключателя SW0 и нажатием BTN0 и результат принятого осциллографом синусоидального сигнала приведены на рисунках 42-43.

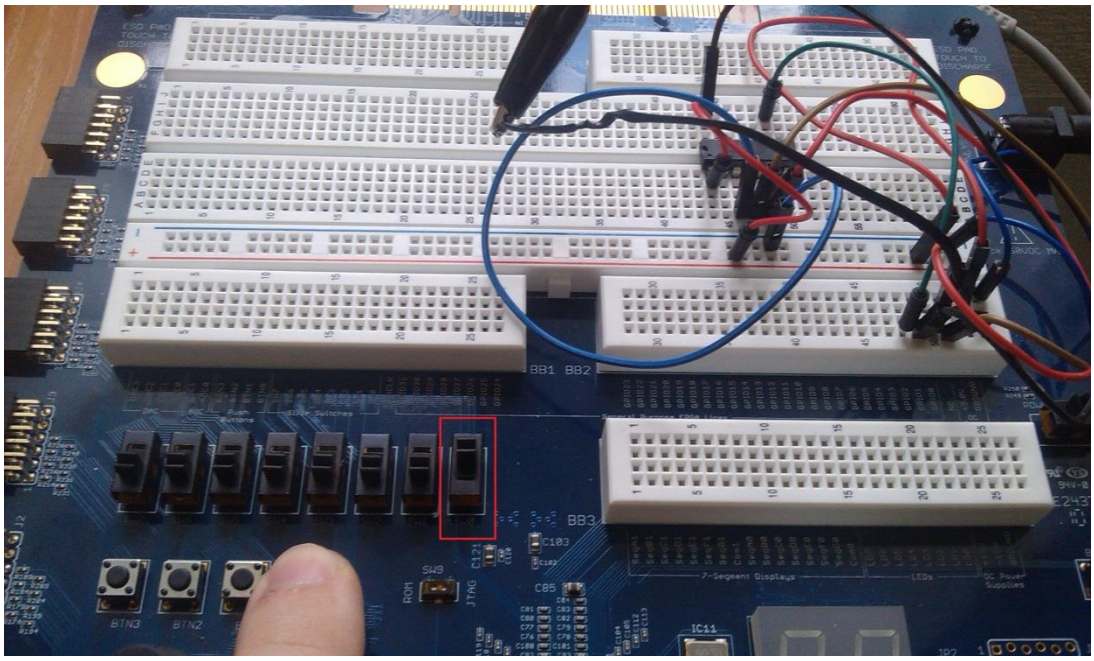


Рисунок 42 – Подача 1 тестового набора на ПЛИС

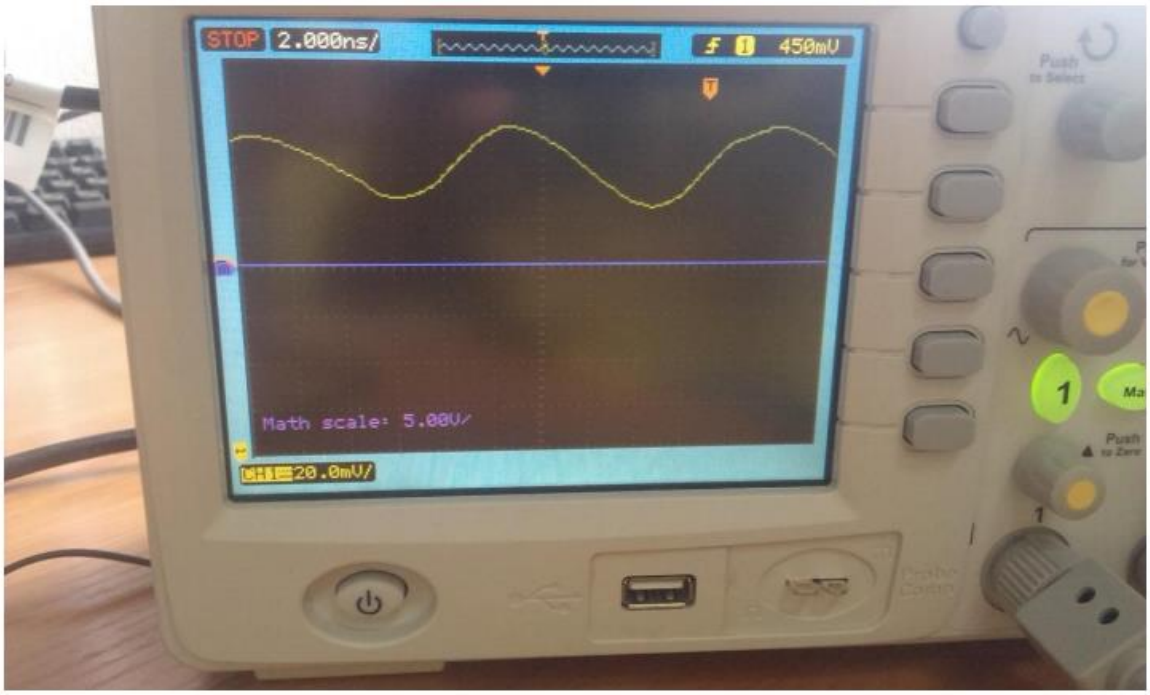


Рисунок 43 – Результат работы устройства при подаче 1 тестового набора

Подача 2 тестового набора (AC327513) осуществляется переключением движкового переключателя SW1 и нажатием BTN0 и результат принятого осциллографом синусоидального сигнала приведены на рисунках 44-45.

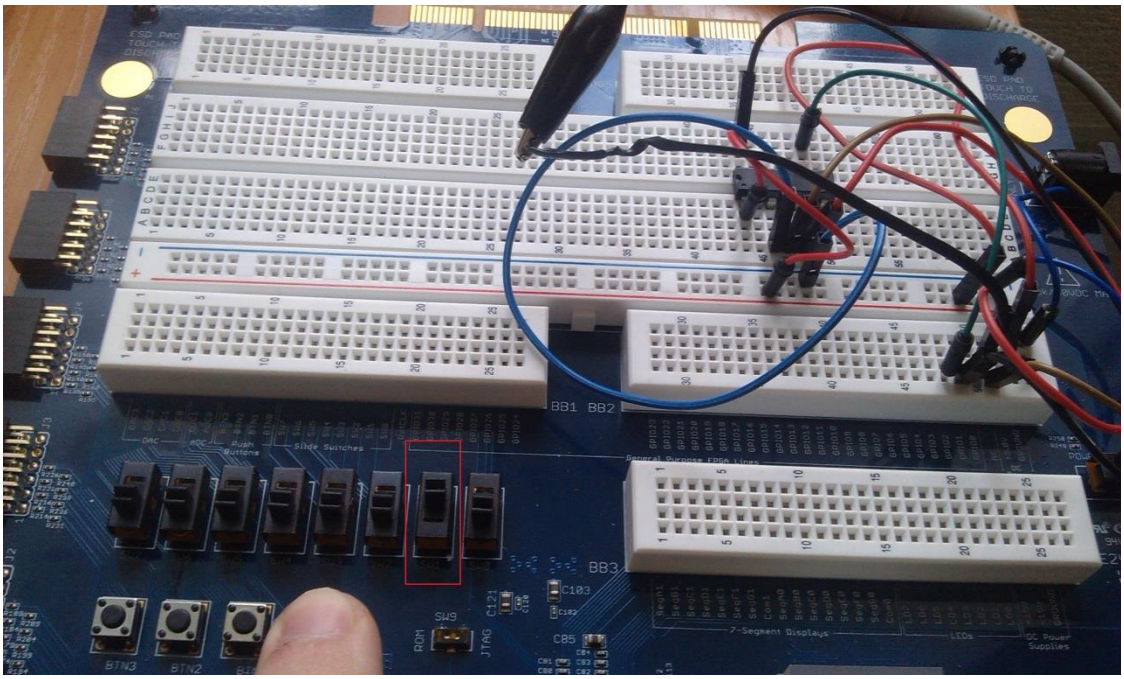


Рисунок 44 – Подача 2 тестового набора на ПЛИС

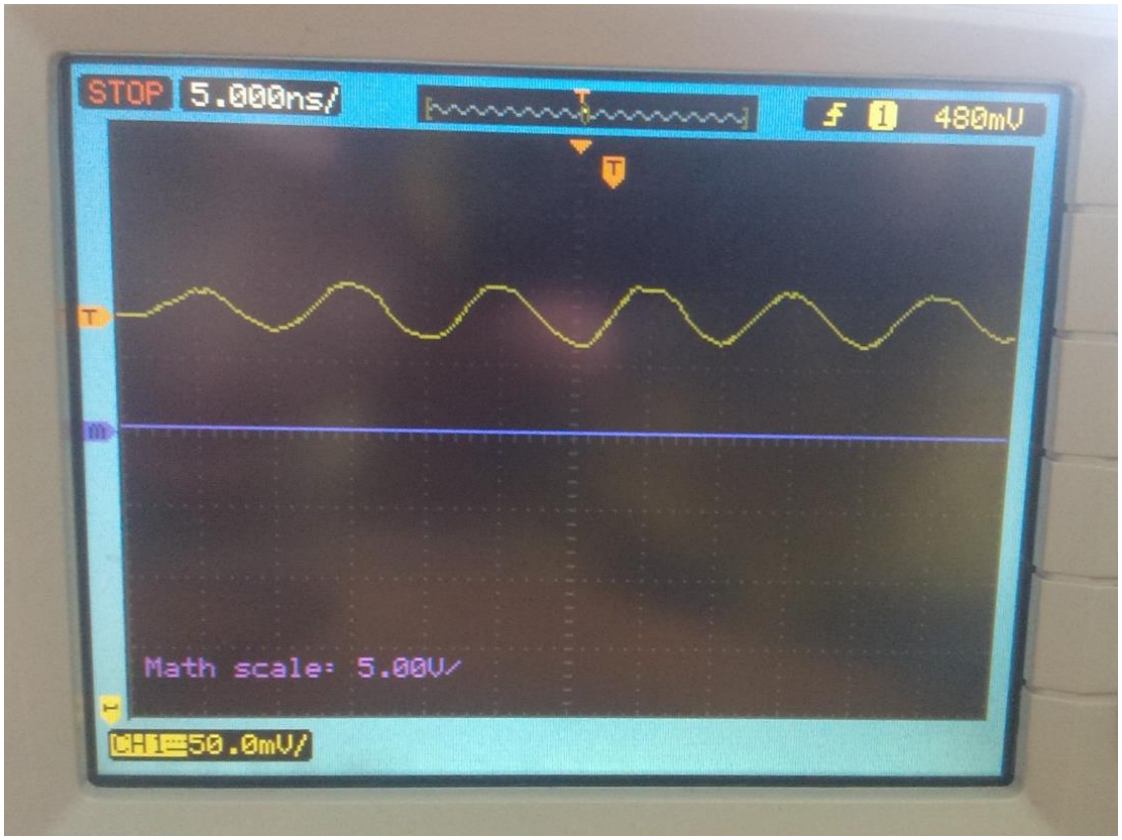


Рисунок 45 – Результат работы устройства при подаче 2 тестового набора

Подача 3 тестового набора (26DFBE34) осуществляется переключением движкового переключателя SW2 и нажатием BTN0 и результат принятого осциллографом синусоидального сигнала приведены на рисунках 46-47.

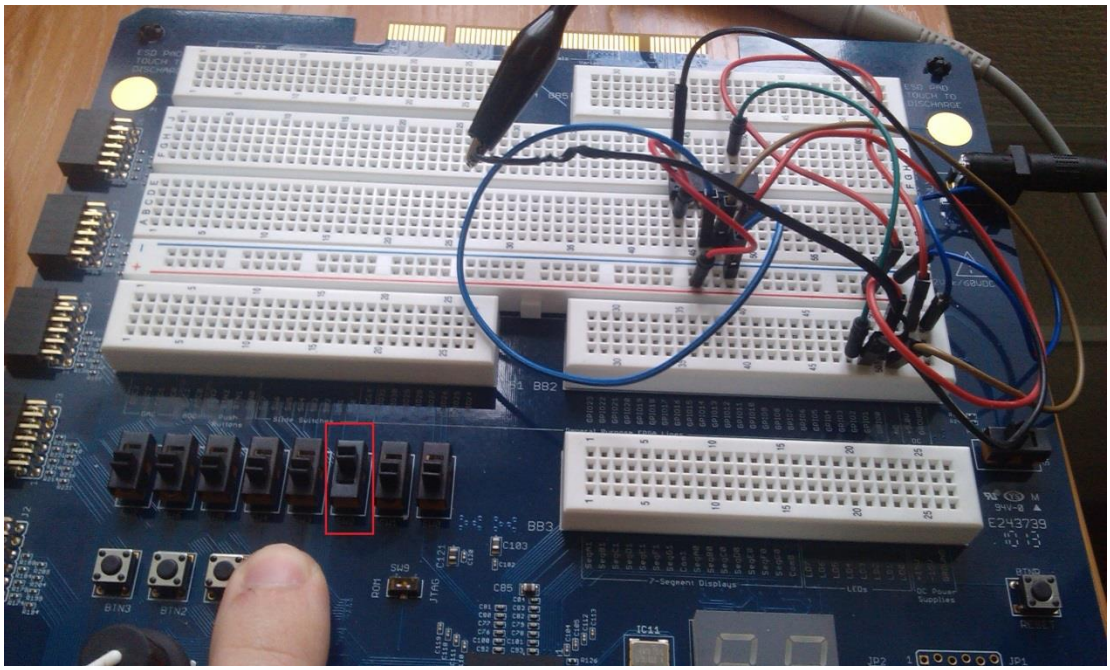


Рисунок 46 – Подача 3 тестового набора на ПЛИС

Рисунок 47 – Результат работы устройства при подаче 3 тестового набора

Подача 4 тестового набора (CBEFF3A2) осуществляется переключением движкового переключателя SW3 и нажатием BTN0 и результат принятого осциллографом синусоидального сигнала приведены на рисунках 48-49.

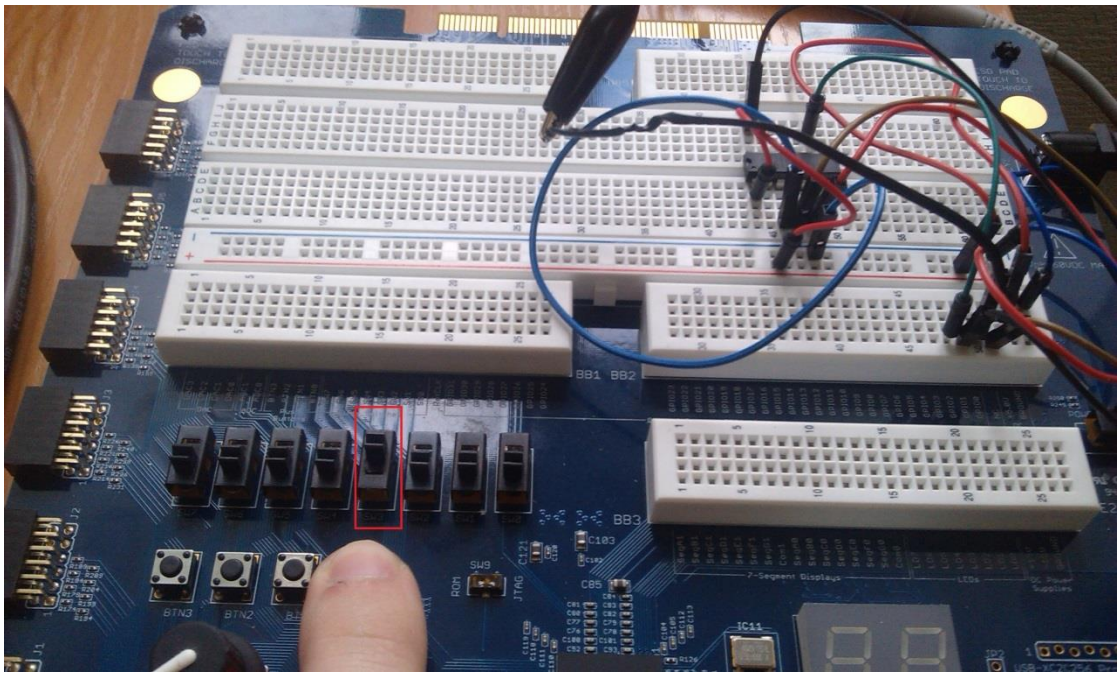


Рисунок 48 – Подача 4 тестового набора на ПЛИС

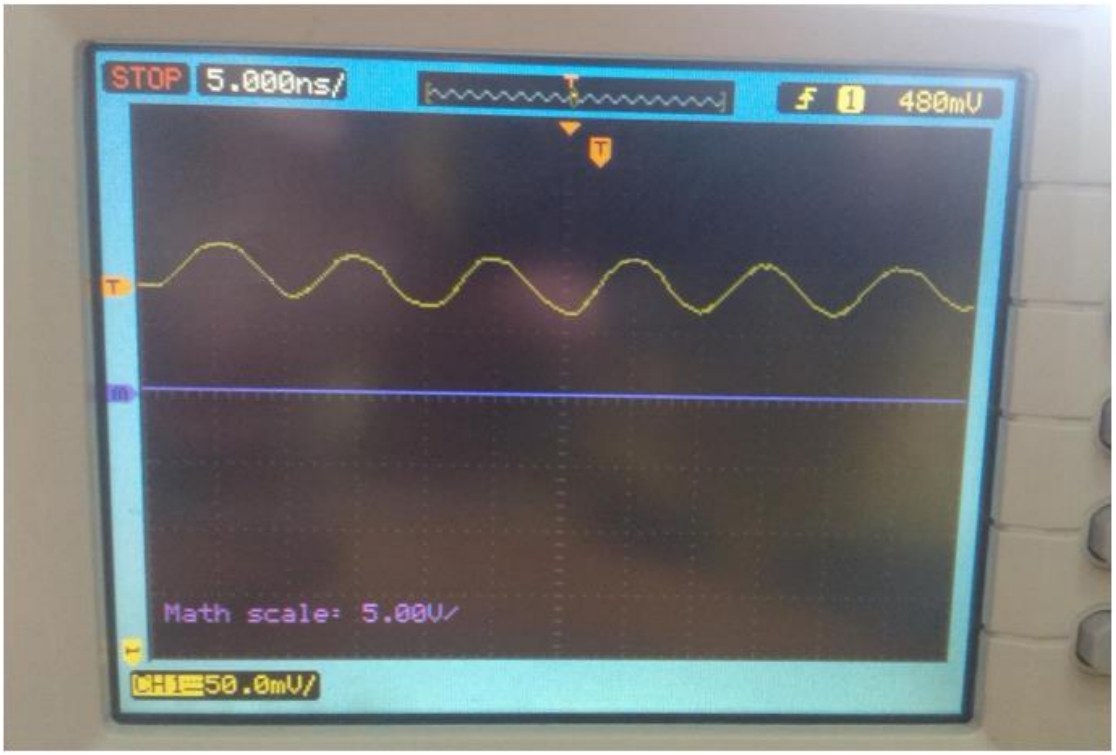


Рисунок 49 – Результат работы устройства при подаче 4 тестового набора

ПРИЛОЖЕНИЕ 4. Функциональная схема первого исполнения устройства

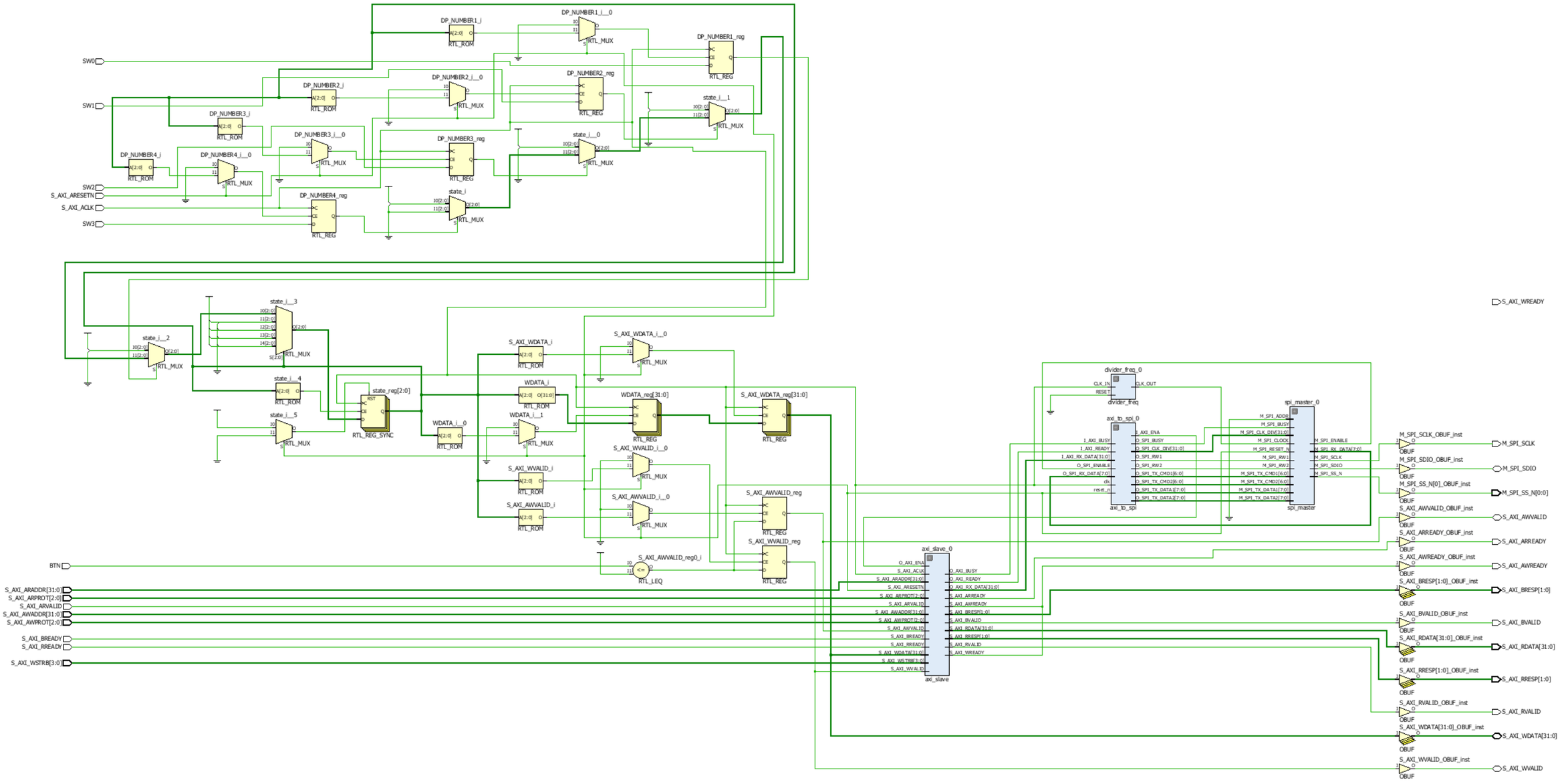


Рисунок 50 – Функциональная схема первого исполнения устройства

ПРИЛОЖЕНИЕ 5. Структурно-функциональная схема второго исполнения устройства

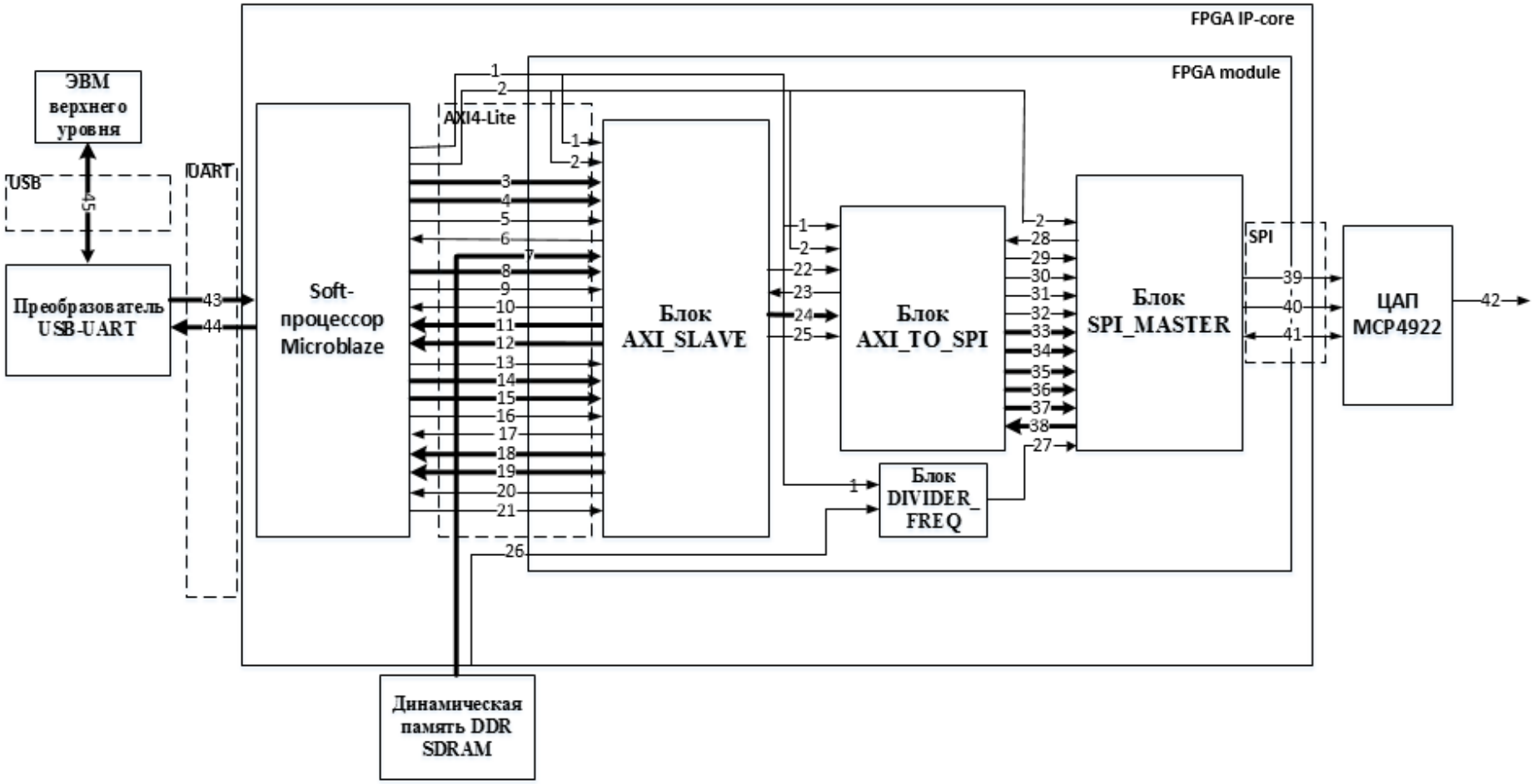


Рисунок 51 – Структурно-функциональная схема второго исполнения устройства

1: S_AXI_ACLK	19: S_AXI_RRESP [1..0]	37: M_SPI_TX_DATA2 [d_width-1..0]
2: S_AXI_RESETN	20: S_AXI_RVALID	38: M_SPI_RX_DATA [d_width-1..0]
3: S_AXI_ AWADDR [31..0]	21: S_AXI_RREADY	39: M_SPI_SCLK
4: S_AXI_ AWPROT [2..0]	22: O_AXI_BUSY	40: M_SPI_SS_N
5: S_AXI_AWVALID	23: O_AXI_ENA	41: M_SPI_SDIO
6: S_AXI_AWREADY	24: O_AXI_RX_DATA [31..0]	42: Vouta
7: S_AXI_WDATA [31..0]	25: O_AXI_READY	43: TxD
8: S_AXI_WSTRB [2..0]	26: RESET	44: RxD
9: S_AXI_WVALID	27: CLK_OUT	45: DATA
10: S_AXI_WREADY	28: M_SPI_ENABLE	
11: S_AXI_BRESP [1..0]	29: M_SPI_BUSY	
12: S_AXI_BVALID	30: M_SPI_ADDR	
13: S_AXI_BREADY	31: M_SPI_RW1	
14 S_AXI_ARADDR [31..0]	32: M_SPI_RW2	
15: S_AXI_ARPROT [2..0]	33: M_SPI_CLK_DIV [31..0]	
16: S_AXI_ARVALID	34: M_SPI_TX_CMD1 [cmd_width-2..0]	
17: S_AXI_ARREADY	35: M_SPI_TX_CMD2 [cmd_width-2..0]	
18: S_AXI_RDATA [31..0]	36: M_SPI_TX_DATA1 [d_width-1..0]	

ПРИЛОЖЕНИЕ 6. Функциональная схема второго исполнения устройства

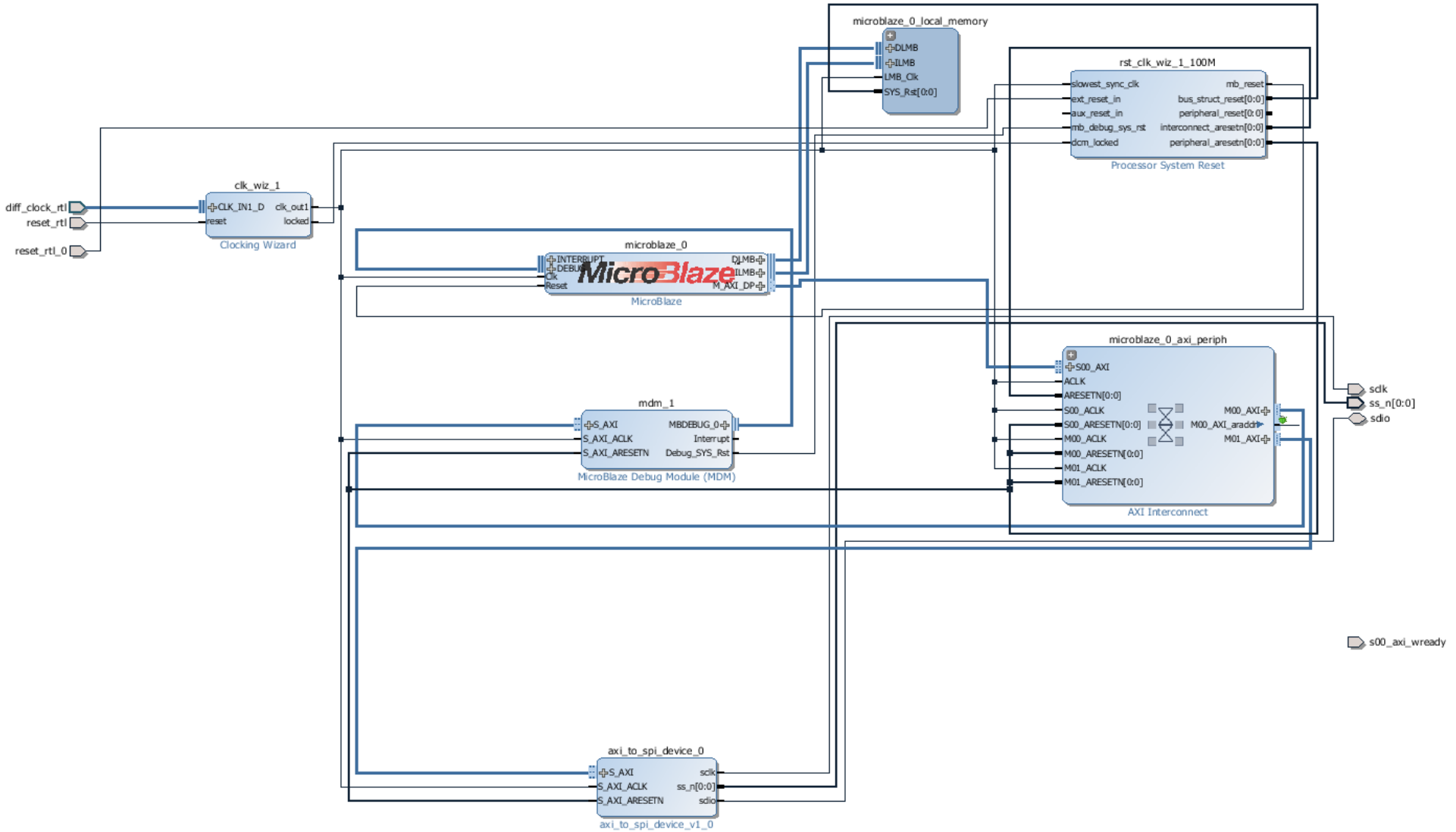


Рисунок 52 – Функциональная схема второго исполнения устройства (взаимодействие IP-ядер)

ПРИЛОЖЕНИЕ 7. Выбор элементной базы для первого и второго исполнений устройства

Для устройства в разных исполнениях необходимы разные компоненты для его создания.

Таблица 27 – Компоненты для первого исполнения устройства

Компонент/ микросхема	Название	Функция в устройстве
Отладочная плата	NI Electronics Digital FPGA Board	Содержит ПЛИС Xilinx Spartan 3E, в которую прошивается программа устройства
Цифро-аналоговый преобразователь	MCP4922	Считывает данные с ПЛИС в последовательном виде и преобразует в синусоидальный сигнал. Выбор данного ЦАП был осуществлен из-за дешевизны микросхемы.
Осциллограф		Считывает синусоидальный сигнал с выхода ЦАП

Таблица 28 – Компоненты для второго исполнения устройства

Компонент/ микросхема	Название	Функция в устройстве
ПЛИС	Xilinx Spartan 3E (без soft-процессора Miroblaze) / Zynq7000 (с soft-процессором)	Микросхема, в которую прошивается программа на ПК через USB.
Цифро-аналоговый преобразователь	MCP4922	Считывает данные с ПЛИС в последовательном виде и преобразует в синусоидальный сигнал. Выбор данного ЦАП был осуществлен из-за дешевизны микросхемы.
Синхронная динамическая память	DDR SDRAM A2S56D40 CTP	Удваивается скорость передачи данных без увеличения частоты тактового сигнала шины памяти
Энергонезависимое ПЗУ EEPROM	EEPROM AT93C46	Дополнительная память
Преобразователь USB-UART	FT232BM	Обеспечивает преобразование принимаемых данных по USB с последовательной передачей на UART
Ферритовый стержень		Устранение помех в схеме/согласно документации преобразователя USB-UART
Конденсаторы		Устранения помех сигналов
Резисторы		Делитель напряжения
Кварцевый резонатор		Подача тактовых сигналов на преобразователь USB-UART
Переключатель		Обеспечения сброса данных по нажатию
Печатная плата		На плате будет сконфигурировано устройство: собрана схема, проведена разводка платы
USB - разьем		Взаимодействия платы с ПК

ПРИЛОЖЕНИЕ 8. Использование soft-процессора Microblaze

MicroBlaze – soft-процессорное ядро, разработанное компанией Xilinx для использования в FPGA. MicroBlaze реализуется с помощью стандартной логики и блоков памяти ПЛИС. С точки зрения архитектуры, MicroBlaze очень похож на процессор с основанной на RISC DLX-архитектурой

MicroBlaze имеет универсальные средства связи с периферией, обеспечивающее возможность применять его в разнообразных встроенных приложениях. Для доступа к внутренней памяти ПЛИС (BRAM), MicroBlaze использует специальную шину LMB, что снижает нагрузку на другие шины. Подключение сопроцессора возможно средствами специального соединения, подобного FIFO — FSL (Fast Simplex Link). Интерфейс с сопроцессором может помочь ускорить работу алгоритмов с большим количеством вычислений, передав часть вычислений в созданный разработчиком (например, написанный на VHDL) аппаратный блок. Архитектура Microblaze приведена на рис. 54.

Рисунок 53 – Soft-процессорное ядро Microblaze

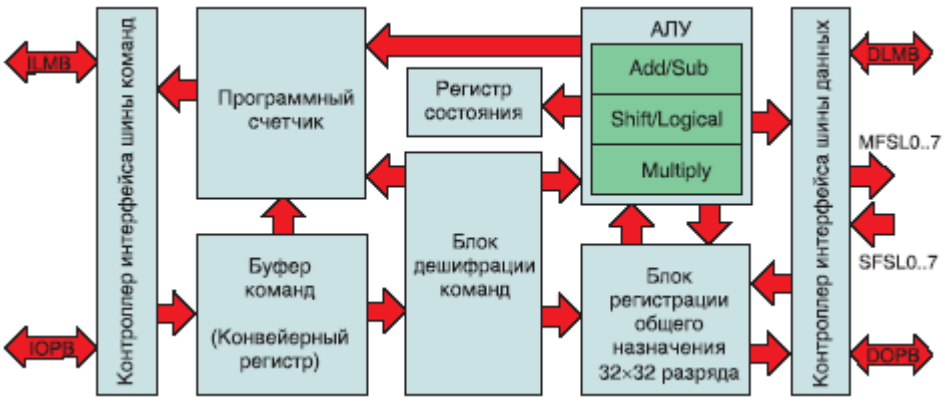


Рисунок 54 – Архитектура soft-процессорного ядра Microblaze

ПРИЛОЖЕНИЕ 9. Условно-графические обозначения и посадочные места элементов

Таблица 29 – Параметры компонента USB-разъема

Название компонента	Наименование корпуса	Тип выводов	Размер шага между выводами	Количество выводов	Габариты корпуса (длина и ширина)
USB - разъем	XP1	Штыревой	1.1 мм	4 вывода	10 x 5 мм

- Изображение УГО:

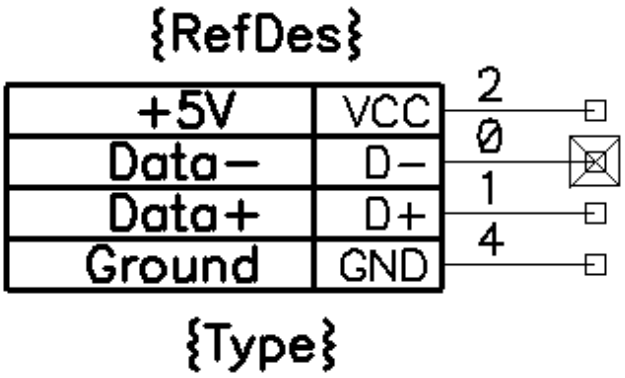


Рисунок 55 – Изображение УГО USB - разъема

- Изображение посадочного места:

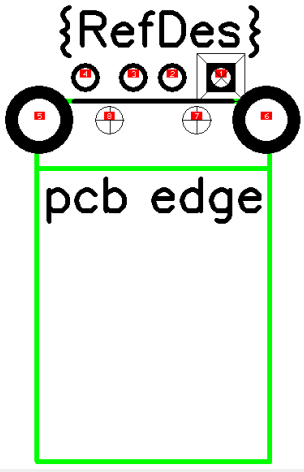


Рисунок 56 – Изображение посадочного места USB - разъема

Таблица 30 – Параметры компонента SOCKET

Название компонента	Наименование корпуса	Тип выводов	Размер шага между выводами	Количество выводов	Габариты корпуса (длина и ширина)
SOCKET	XP2	Штыревой	1.5мм	9 выводов	5 x 18 мм

- Изображение УГО:

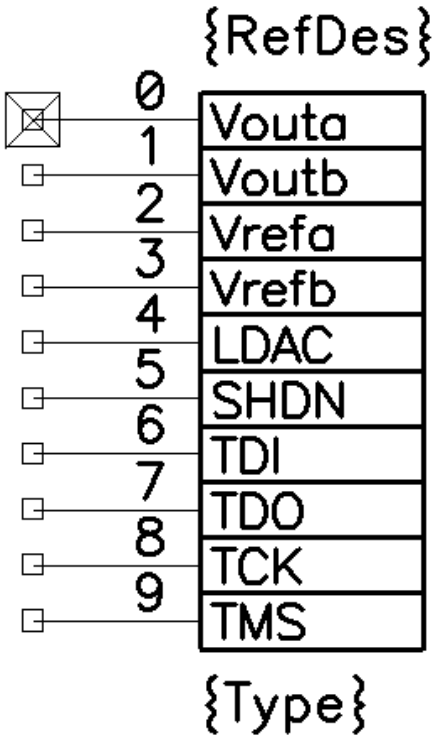


Рисунок 57 – Изображение УГО разъема на 10 выводов

- Изображение посадочного места:

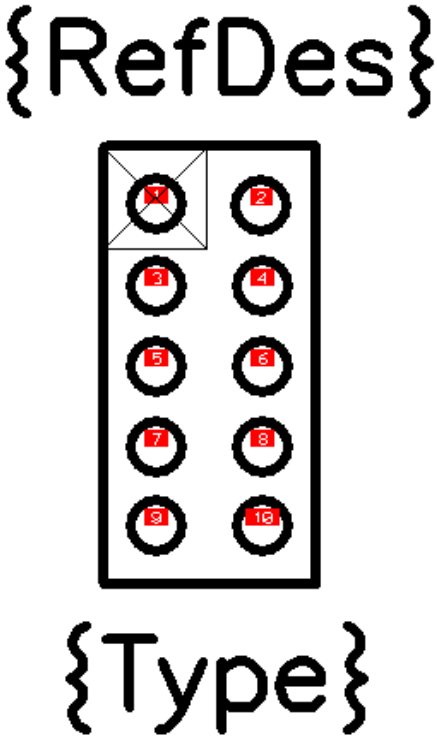


Рисунок 58 – Изображение посадочного места разъёма на 10 выводов

Таблица 31 – Параметры компонента EEPROM AT93C46

Название компонента	Наименование корпуса	Тип выводов	Размер шага между выводами	Количество выводов	Габариты корпуса (длина и ширина)
EEPROM AT93C46	8P3 – PDIP	Планарный	2.54 мм	8 выводов	7,8 x 9,2 мм

- Изображение УГО:

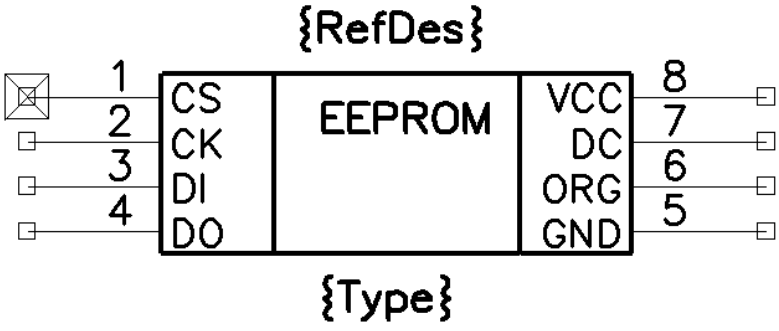


Рисунок 59 – Изображение УГО энергонезависимого ПЗУ EEPROM AT93C46

Таблица 32 – Наименование выводов микросхемы EEPROM AT93C46

Номер пина	Обозначение	Тип/полярность	Функция пина
1	CS	Вход	вход выбора чипа для включения тактовой частоты и данных
2	SK	Вход	вход тактовой частоты
3	DI	Вход	входные данные
4	DO	Выход	выходные данные
5	GND	Буфер	сигнал последовательных данных
6	NC		дополнительный пин
7	NC		дополнительный пин
8	VCC	Буфер	Питание +5В

- Изображение посадочного места:

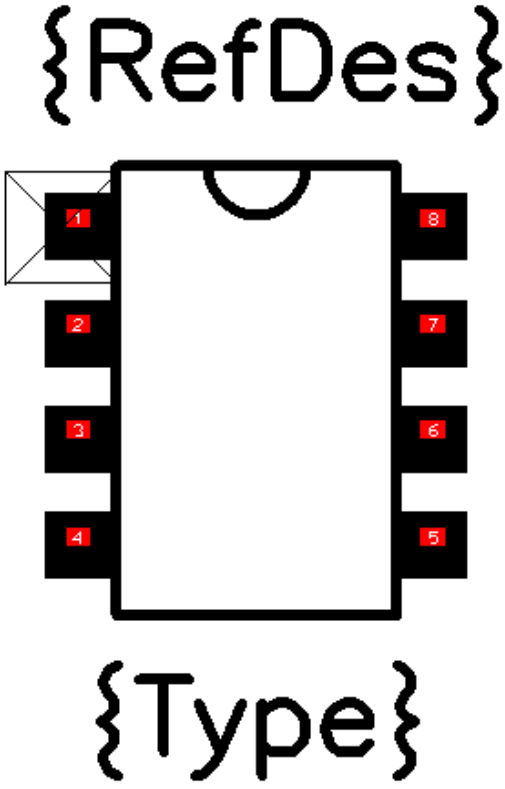


Рисунок 60 – Изображение посадочного места энергонезависимого ПЗУ EEPROM AT93C46

Таблица 33 – Параметры компонента ПЛИС Xilinx Spartan 3e

Название компонента	Наименование корпуса	Тип выводов	Размер шага между выводами	Количество выводов	Габариты корпуса (длина и ширина)
ПЛИС Xilinx Spartan 3e	TQ144	Планарный	0.54 мм	144 вывода	10 x 10 мм

- Изображение УГО:

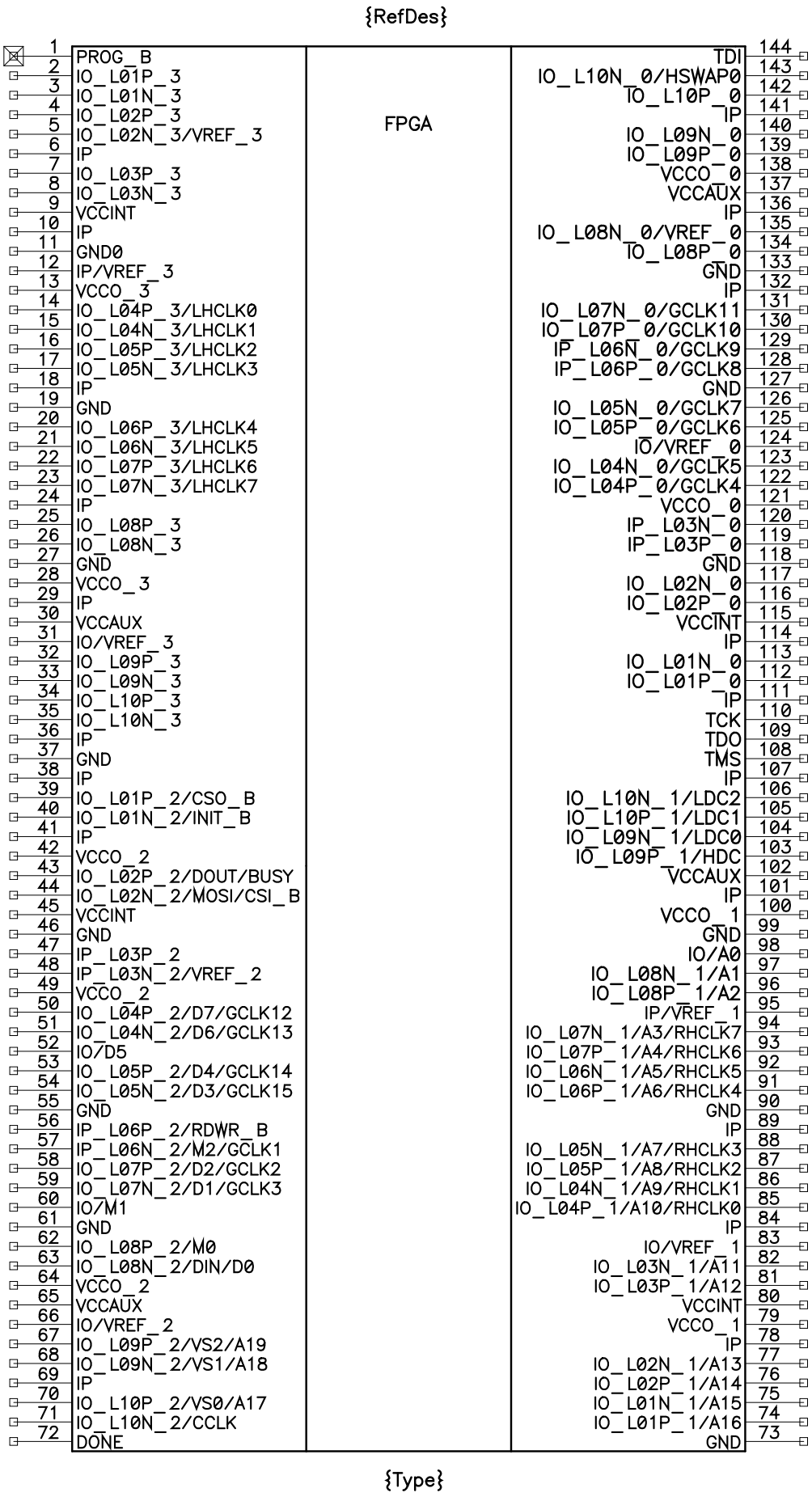


Рисунок 61 – Изображение УГО ПЛИС Xilinx Spartan 3e

- Изображение посадочного места:

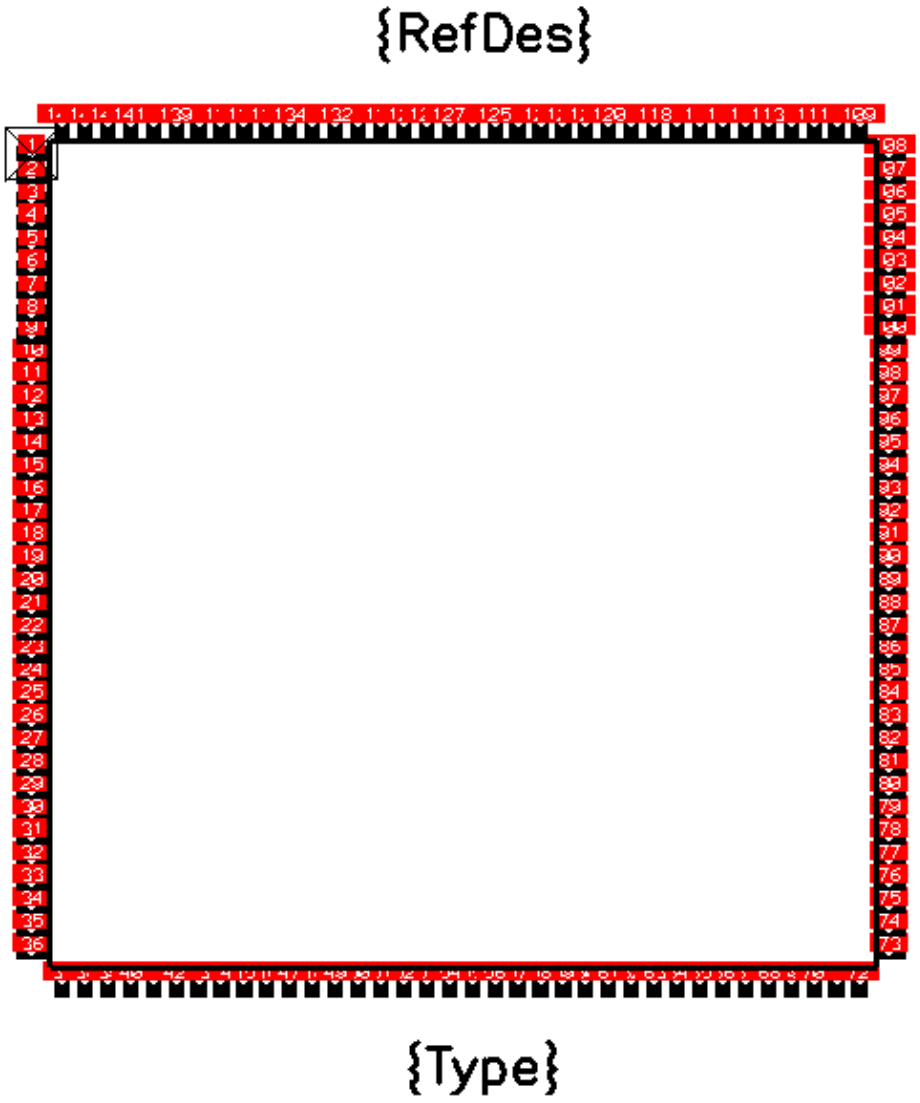


Рисунок 62 – Изображение посадочного места ПЛИС Xilinx Spartan 3e

Таблица 34 – Параметры компонента DDR SDRAM AT25HP256

Название компонента	Наименование корпуса	Тип выводов	Размер шага между выводами	Количество выводов	Габариты корпуса (длина и ширина)
DDR SDRAM AT25HP256	DIP-8	Планарный	0.5 мм	66 выводов	8 x 18 мм

- Изображение УГО:

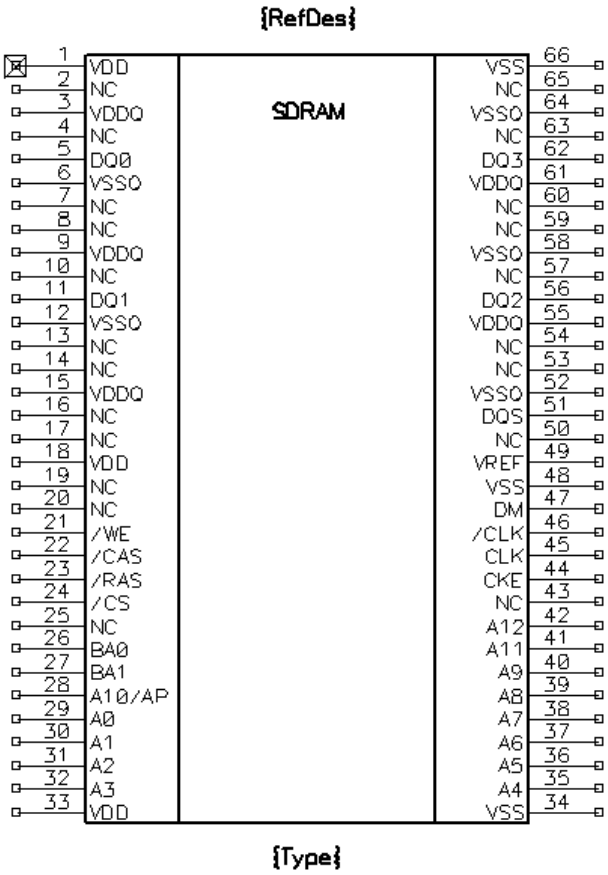


Рисунок 63 – Изображение УГО DDR SDRAM AT25HP256

Таблица 35 – Наименование выводов микросхемы DDR SDRAM AT25HP256

Сигнал	Тип	Полярность	Функция
CLK	Вход	Положит. фронт	Вход синхронизации системы. Все входы SDRAM срабатывают по положительному фронту CLK
CKE	Вход	Активный уровень – высокий	Активирует сигнал CLK высоким уровнем и отключает – низким. Отключение CLK низким уровнем CKE инициирует режимы энергосбережения и саморегенерации
/CS	Вход	Активный – низкий	Низкий уровень включает дешифратор команд, высокий – выключает. Когда дешифратор команд выключен, новые команды игнорируются и продолжается предыдущая операция
/RAS, /CAS, /WE	Вход	Активный – низкий	Состояние RAS, CAS и WE на положительном фронте CLK определяет операцию, которую будет выполнять SDRAM
BS0, BS1	Вход		Выбирает банк, который должен быть активным
A0 – A11	Вход		Во время цикла команды активации банка по переднему фронту CLK определяют адрес строки (RA0 – RA11). Во время цикла команды чтения или записи по переднему фронту CLK A0 – A9 и A11 определяют адрес строки (CA0 – CA9 и CA11). A10 используется для задания операции автоподзаряда в конце пакетного цикла чтения или записи. При высоком уровне A10

			задан автоподзаряд, а BS0 и BS1 определяют банк для подзаряда. При низком уровне A10 подзаряд отключен. В цикле подзаряда A10 в сочетании с BS0 и BS1 задает банк(и), в котором выполняется подзаряд. При высоком уровне A10 подзаряд выполняется во всех банках, независимо от состояния BS0 и BS1. При низком уровне A10 банк для подзаряда определяется BS0 и BS1
DQ0 – DQ15	Вход/ Выход		Входы/выходы данных работают так же, как в обычной динамической памяти
DQM LDQM UDQM	Вход	Активный уровень – высокий	Маска ввода/вывода данных переводит буферы линий DQ в третье состояние высоким уровнем. В 16-разрядных модулях LDQM и UDQM управляют буферами ввода/вывода младшего и старшего байтов соответственно. В режиме чтения маска DQM имеет задержку в два цикла и управляет выходными буферами как сигнал “Разрешение выхода”. Низкий уровень DQM включает выходные буферы, а высокий – отключает. В режиме записи DQM имеет нулевую задержку и действует, как маска слова, разрешая запись входных данных низким уровнем и блокируя ее высоким уровнем DQM
V_{DD}, V_{SS}	Вход		Питание и земля для входных буферов и логических схем ядра
V_{DDQ}, V_{SSQ}	Вход		Отдельные изолированные шины питания и земли для выходных буферов, обеспечивающие улучшенную помехоустойчивость к шумам

- Изображение посадочного места:

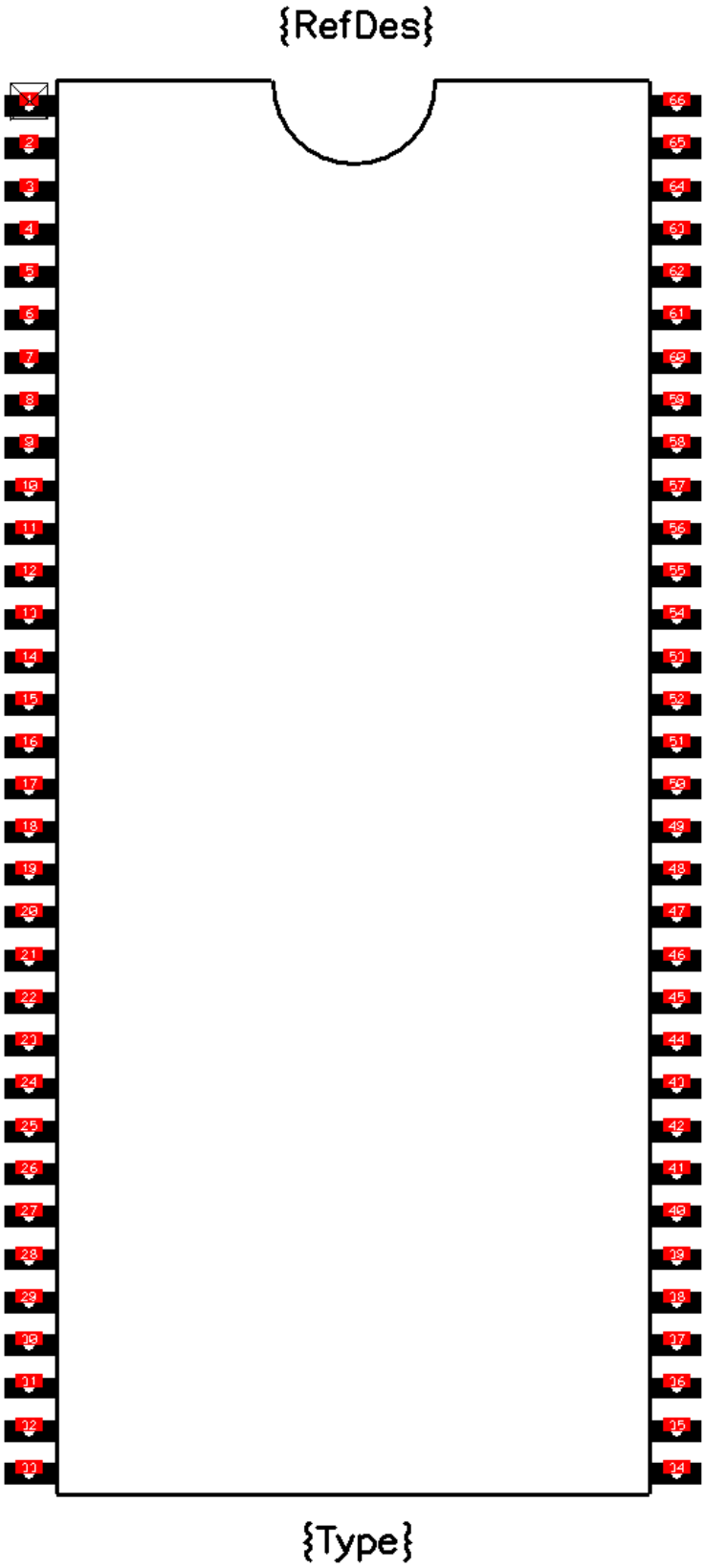


Рисунок 64 – Изображение посадочного места DDR SDRAM AT25HP256

Таблица 36 – Параметры компонента ЦАП MCP4922

Название компонента	Наименование корпуса	Тип выводов	Размер шага между выводами	Количество выводов	Габариты корпуса (длина и ширина)
ЦАП MCP4922	14-SOIC	Планарный	1.54 мм	14 выводов	6 x 14 мм

- Изображение УГО:

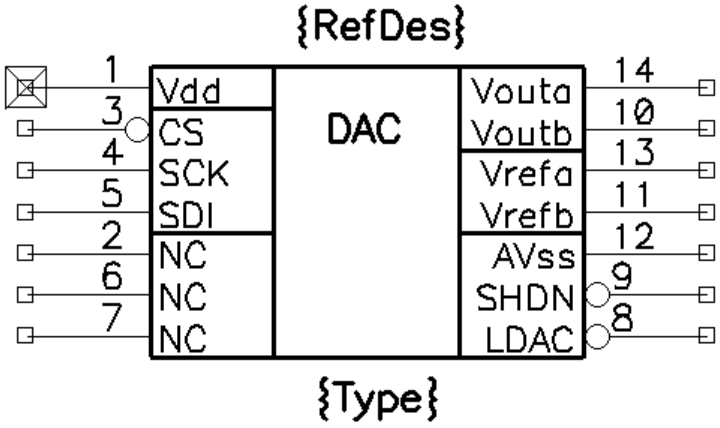


Рисунок 65 – Изображение УГО ЦАП MCP4922

Таблица 37 – Наименование выводов микросхемы ЦАП MCP4922

Номер пина	Обозначение	Тип	Полярность	Функция пина
1	V_{dd}	Вход		Вход источника питания
2	NC			Дополнительный пин
3	/CS	Вход	Инверсный	Вход выбора ведомого устройства для включения тактовой частоты и данных
4	SCK	Вход	Положит. фронт	Вход тактовой частоты
5	SDI	Вход/выход	Прямой	Сигнал последовательных данных
6	NC			Дополнительный пин
7	NC			Дополнительный пин
8	/LDAC	Вход	Инверсный	Вход синхронизации ЦАП
9	/SHDN	Вход	Инверсный	Вход аппаратного отключения
10	V_{outb}	Выход		Выходное напряжение канала В ЦАП
11	V_{refb}	Вход		Вход опорного напряжения канала В ЦАП
12	AV_{ss}	Вход		Пин аналогового контакта заземления
13	V_{refa}	Вход		Вход опорного напряжения канала А ЦАП
14	V_{outa}	Выход		Выходное напряжение канала А ЦАП

- Изображение посадочного места:

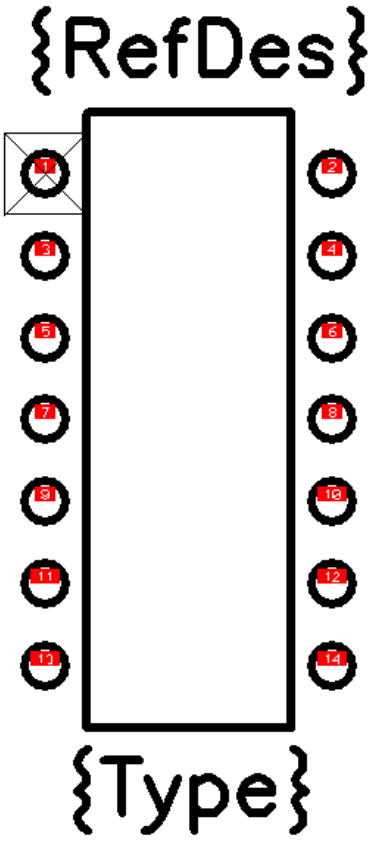


Рисунок 66 – Изображение посадочного места ЦАП MCP4922

Таблица 38 – Параметры компонента преобразователя FT232BM

Название компонента	Наименование корпуса	Тип выводов	Размер шага между выводами	Количество выводов	Габариты корпуса (длина и ширина)
Преобразователь USB-to-UART FT232BM	TQFP32	Планарный	0.54 мм	32	8 x 8 мм

- Изображение УГО:

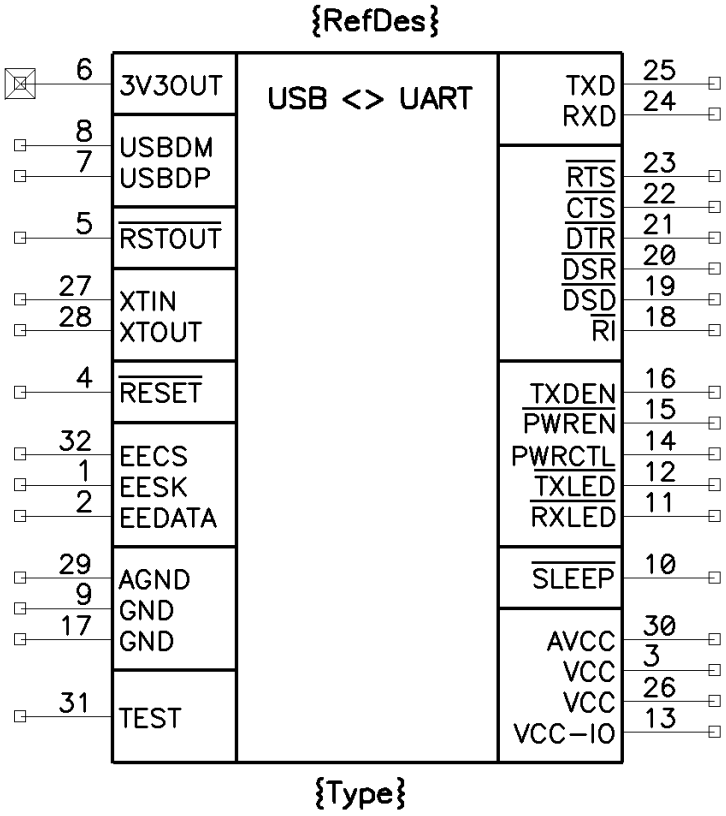


Рисунок 67 – Изображение УГО преобразователя USB-to-UART FT232BM

Таблица 39 – Наименование выводов микросхемы FT232BM

Номер пина	Обозначение	Тип	Полярность	Функция пина
1	EESK	Выход	Полож. фронт	Тактовый сигнал EEPROM
2	EEDATA	Вход/выход	Прямой	Данные по EEPROM
3	VCC			Питание
4	RESET#	Вход	Инверсный	Сброс данных
5	RSTOUT#	Выход	Инверсный	Выход сброса сигнала генератора
6	3V3OUT			Выход 3,3 В
7	USBDP	Вход/выход	Прямой	Положительный сигнал данных USB
8	USBDM	Вход/выход	Прямой	Отрицательный сигнал данных USB
9	GND			Подключение к земле
10	SLEEP#	Выход	Инверсный	Сигнал для перехода USB – RS232
11	RXLED#			Сигнал светодиодного индикатора для приема данных
12	TXLED#			Сигнал светодиодного индикатора для отправки данных
13	VCC-IO			Питание UART
14	PWRCTL	Вход	Прямой	Контроль питания
15	PWREN#	Выход	Инверсный	Включить опцию выключения интерфейса в EEPROM
16	TXDEN	Выход	Прямой	Разрешает передачу данных по RS485
17	GND			Подключение к земле
18	RI#	Вход	Инверсный	Входной индикатор контроля индикатора
19	DSD#	Вход	Инверсный	Входной сигнал обнаружения несущей данных
20	DSR#	Вход	Инверсный	Входной сигнал управления набором данных
21	DTR#	Выход	Инверсный	Выход контроля готовности терминала данных
22	CTS#	Вход	Инверсный	Вход очистки контроля отправки
23	RTS#	Выход	Инверсный	Запрос на отправку выходного сигнала управления
24	RXD	Вход	Прямой	Принимает данные по UART
25	TXD	Выход	Прямой	Передает данные по UART
26	VCC			Питание +5В
27	XTIN	Вход	Прямой	Вход для подключения к кварцевому резонатору
28	XTOUT		Прямой	Выход для подключения к кварцевому резонатору
29	AGND			Аналоговая земля
30	AVCC			Аналоговое питание
31	TEST	Вход	Прямой	Устанавливает устройство в тестовый режим
32	EECS	Вход/выход	Прямой	Выбор ведомого устройства EEPROM

- Изображение посадочного места:

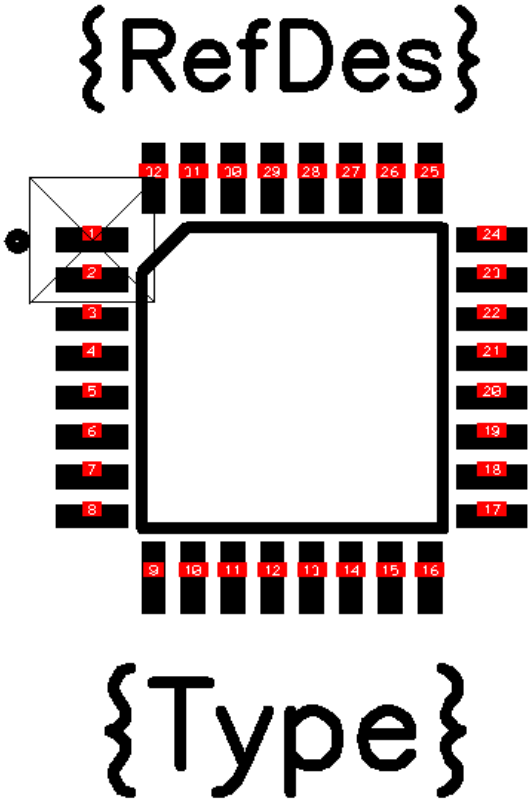
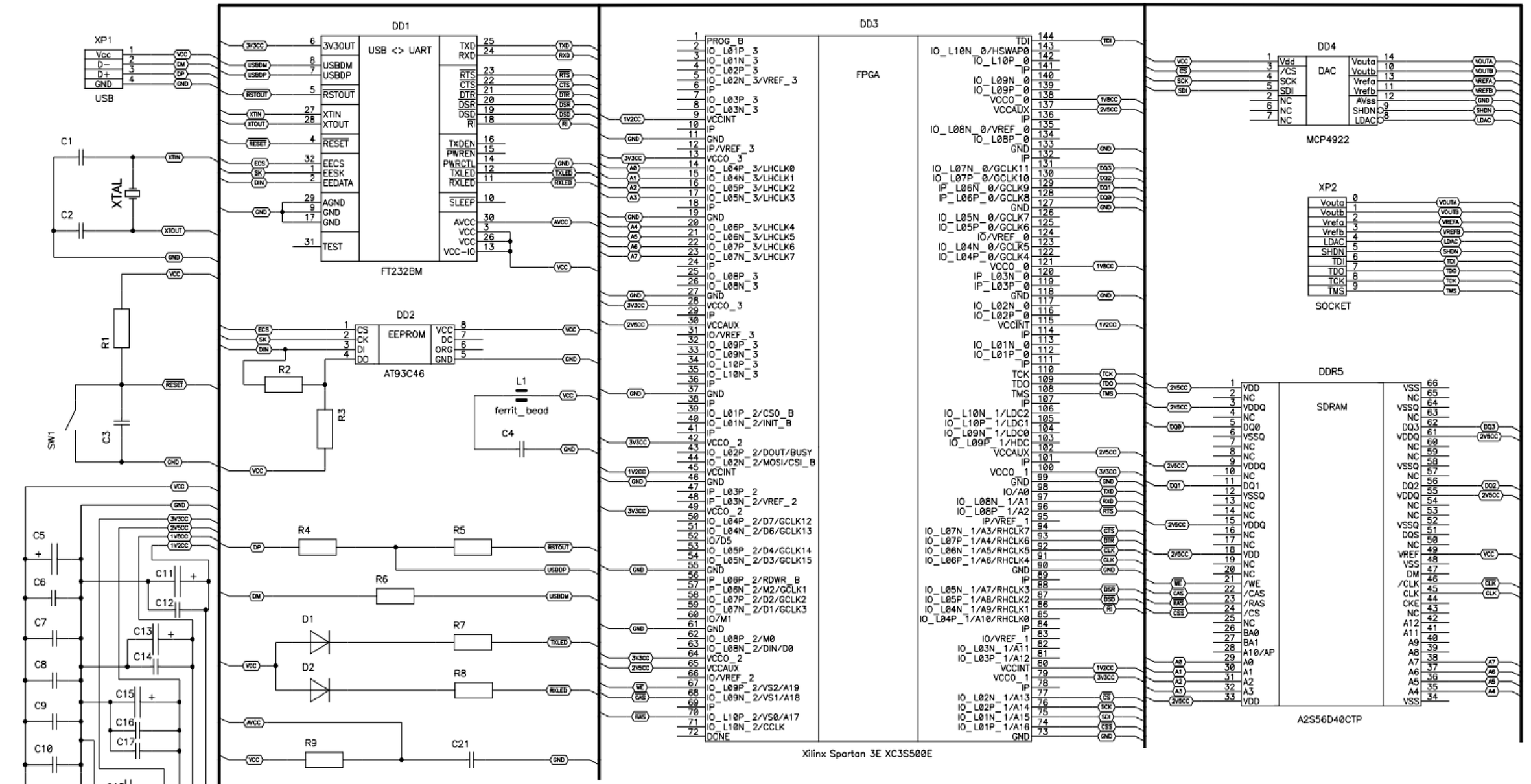


Рисунок 68 – Изображение контактной площадки преобразователя USB-to-UART FT232BM

ПРИЛОЖЕНИЕ 10. Принципиальная схема устройства

ФЮРА. XXXXXX.XXX



					ФЮРА. XXXXXX.XXX	
Изм.	Лист	№ докум.	Подп.	Дата	Устройство межинтерфейсного взаимодействия AXI- TO-SPI Принципиальная электрическая схема	Лит. У / Масса / Масштаб
Разраб.		Старшинов В.С.				
Пров.		Мальчуков А.Н.				Лист 1 / Листов 1
Т. контр.						
Н. контр.						ТПУ ИК Группа 8В3А
Утв.		Мальчуков А.Н.				

ПРИЛОЖЕНИЕ 11. Спецификация к ПЭС (часть 1)

Обозначение	Наименование	КОЛ
	Преобразователь USB - UART	
DD1	FT232BM	1
	Энергонезависимая память EEPROM	
DD2	AT93C46	1
	ПЛИС	
DD3	Xilinx Spartan 3E XC3S500E	1
	Цифро-аналоговый преобразователь	
DD4	MCP4922	1
	Синхронная динамическая память	
DD5	DDR SDRAM A2S56D40CTP	1
	Разъем	
XP1	USB A-001	1
XP2	PBD 10	
	Ферритовый фильтр	
L1	BL01RN1A2AB	1
	Кнопка	
SW1	SDTX - 210	1
	Кварцевый генератор	
XTAL	HC - 49S	1

					ФЮРА. XXXXXX.XXX ПЭ
Изм	Код	№докум.			
Разраб.		Старшинов В.С.			Устройство межинтерфейсного взаимодействия AXI-TO-SPI Перечень элементов
Проверил		Мальчуков А.Н.			
Т. контр.					
Н. контр.					
Утв.		Мальчуков А.Н.			

Лит	Масса	Масштаб
У		
Лист 1		Листов 2

ТПУ ИК
Группа 8В3А

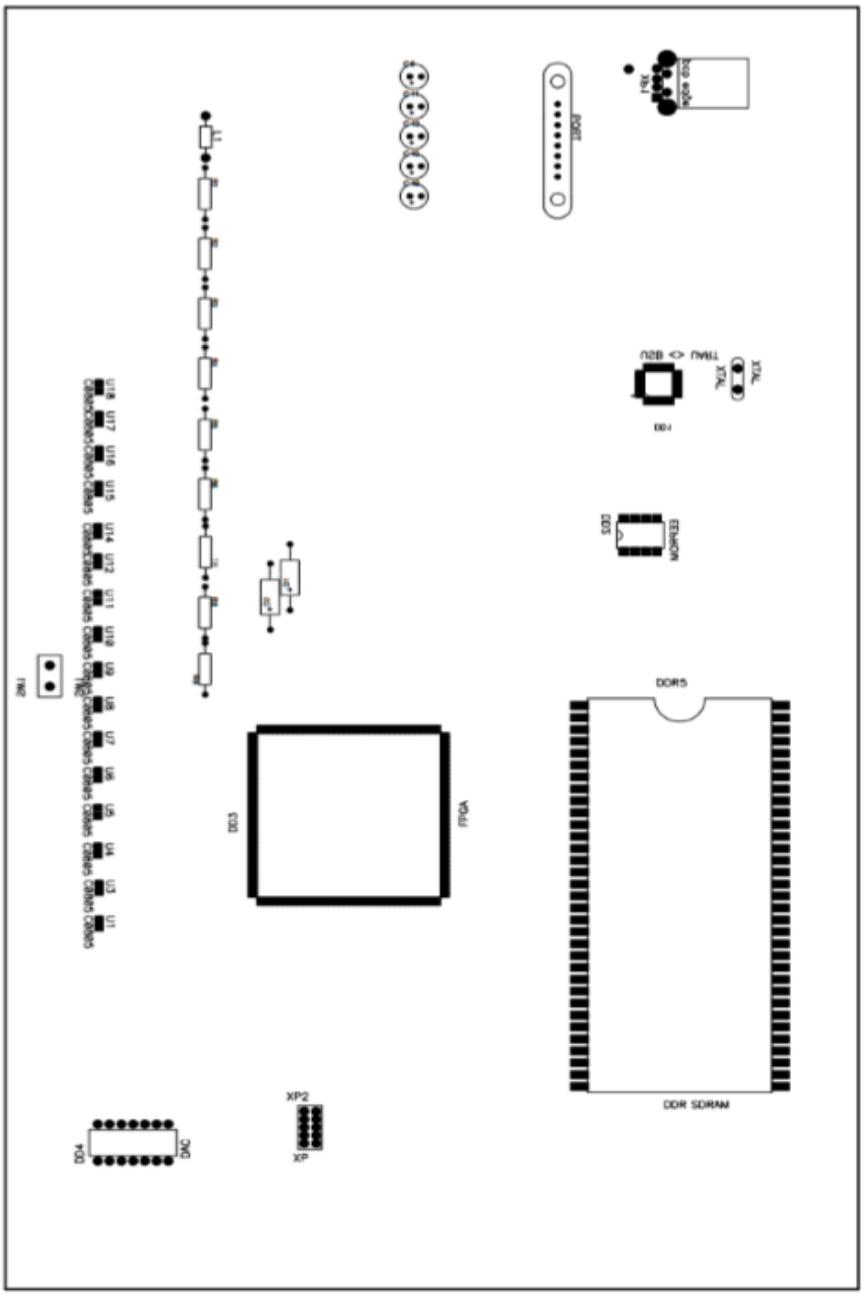
ПРИЛОЖЕНИЕ 12. Спецификация к ПЭС (часть 2)

Обозначение	Наименование	КОЛ
	Светодиоды	
D1-D2	АЛ307ЕМ	2
	Конденсаторы	
C5, C11, C13, C15, C18	К50-35 0,1мкФ	5
C3, C6-C10, C12, C14	К10-17 0.01 мкФ	8
C16-C17, C19-C20	К10-17 0.01 мкФ	4
C1-C2	К10-17Б 27 пФ	2
C4, C21	К73-9 33 нФ	2
	Резисторы	
R1, R3	MF-25 10 кОм	2
R2	MF-50 2,2 кОм	1
R4, R6	CF-100 27 Ом	2
R5	CF-50 1,5 кОм	1
R7, R8	C2-23 220 Ом	2
R9	CF25 470 Ом	1

					ФЮРА. XXXXXX.XXX ПЭ	
Изм	Код	№докум.				
Разраб.		Старшинов В.С.			Устройство межинтерфейсного взаимодействия AXI-TO-SPI Перечень элементов	Лит: У; Масса; Масштаб
Проверил		Мальчуков А.Н.				Лист 2 Листов 2
Т. контр.						
Н. контр.						ТПУ ИК Группа 8В3А
Утв.		Мальчуков А.Н.				

ПРИЛОЖЕНИЕ 13. Сборочный чертеж

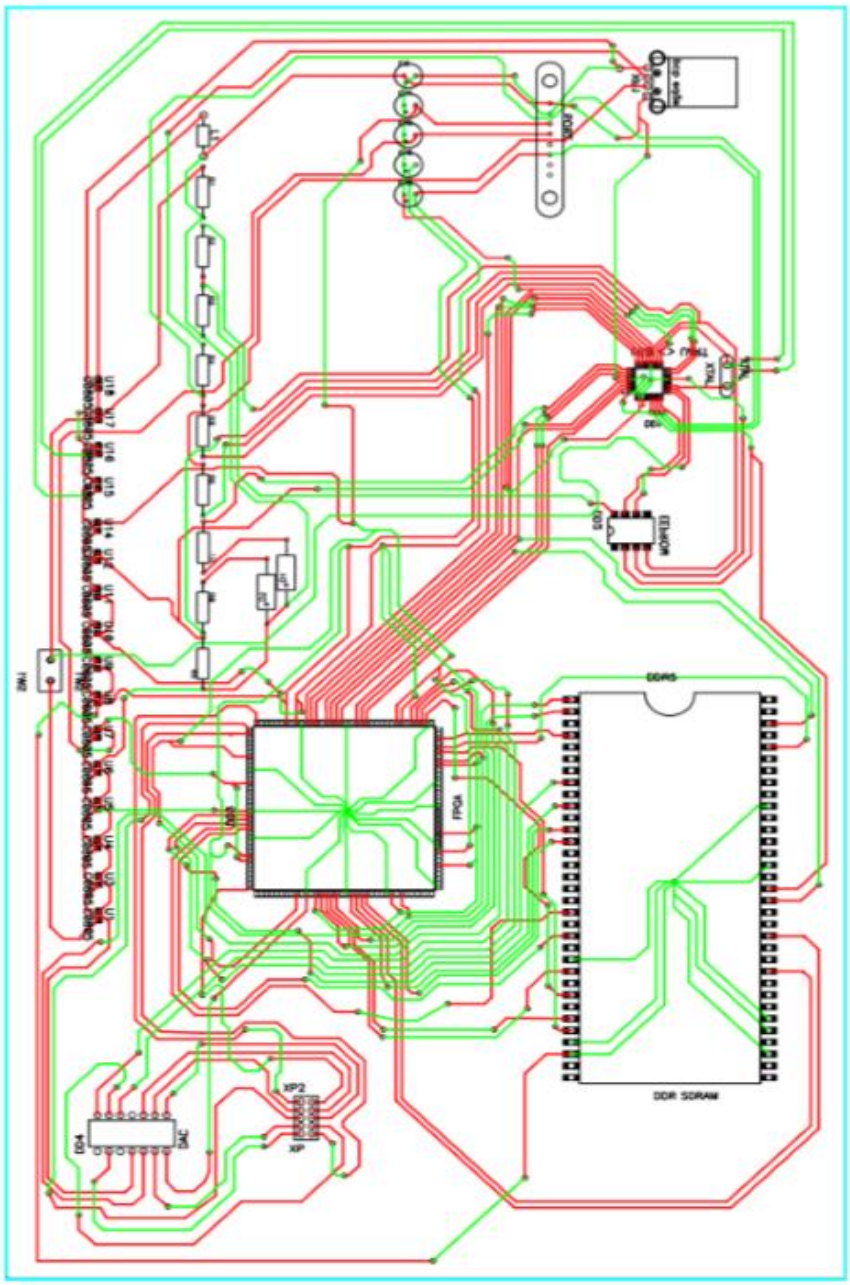
ФЮРА. XXXXXX.XXX



					ФЮРА. XXXXXX.XXX	
Изм	Код	№ докум.			Устройство межинтерфейсного взаимодействия AXI-TO-SPI Сборочный чертеж	Лит: У; Масса; Масштаб
Разраб.		Старшинов В.С.				Лист 1 Листов 1
Проверил		Мальчуков А.Н.				ТПУ ИК Группа 8В3А
Т. контр.						
Н. контр.						
Утв.		Мальчуков А.Н.				

ПРИЛОЖЕНИЕ 14. Общий вид печатной платы

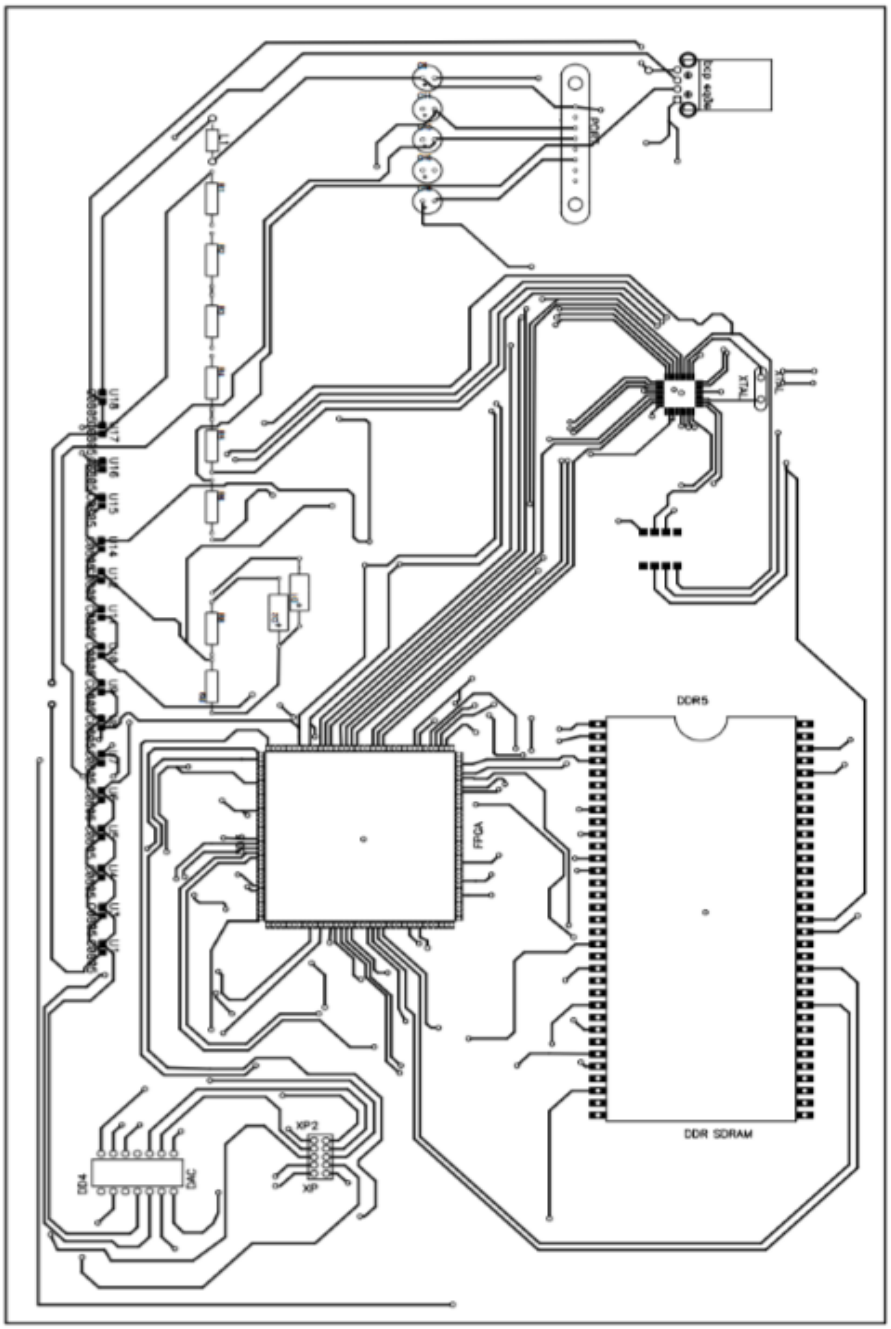
ФЮРА. XXXXXX.XXX



					ФЮРА. XXXXXX.XXX	
Изм	Кол	№ докум.			Устройство межинтерфейсного взаимодействия AXI-TO-SPI Печатная плата Общий вид	Лит: У; Масса; Масштаб
Разраб.		Старшинов В.С.				Лист 1 Листов 1
Провериł		Мальчуков А.Н.				ТПУ ИК Группа 8В3А
Т. контр.						
Н. контр.						
Утв.		Мальчуков А.Н.				

ПРИЛОЖЕНИЕ 15. Верхний слой печатной платы

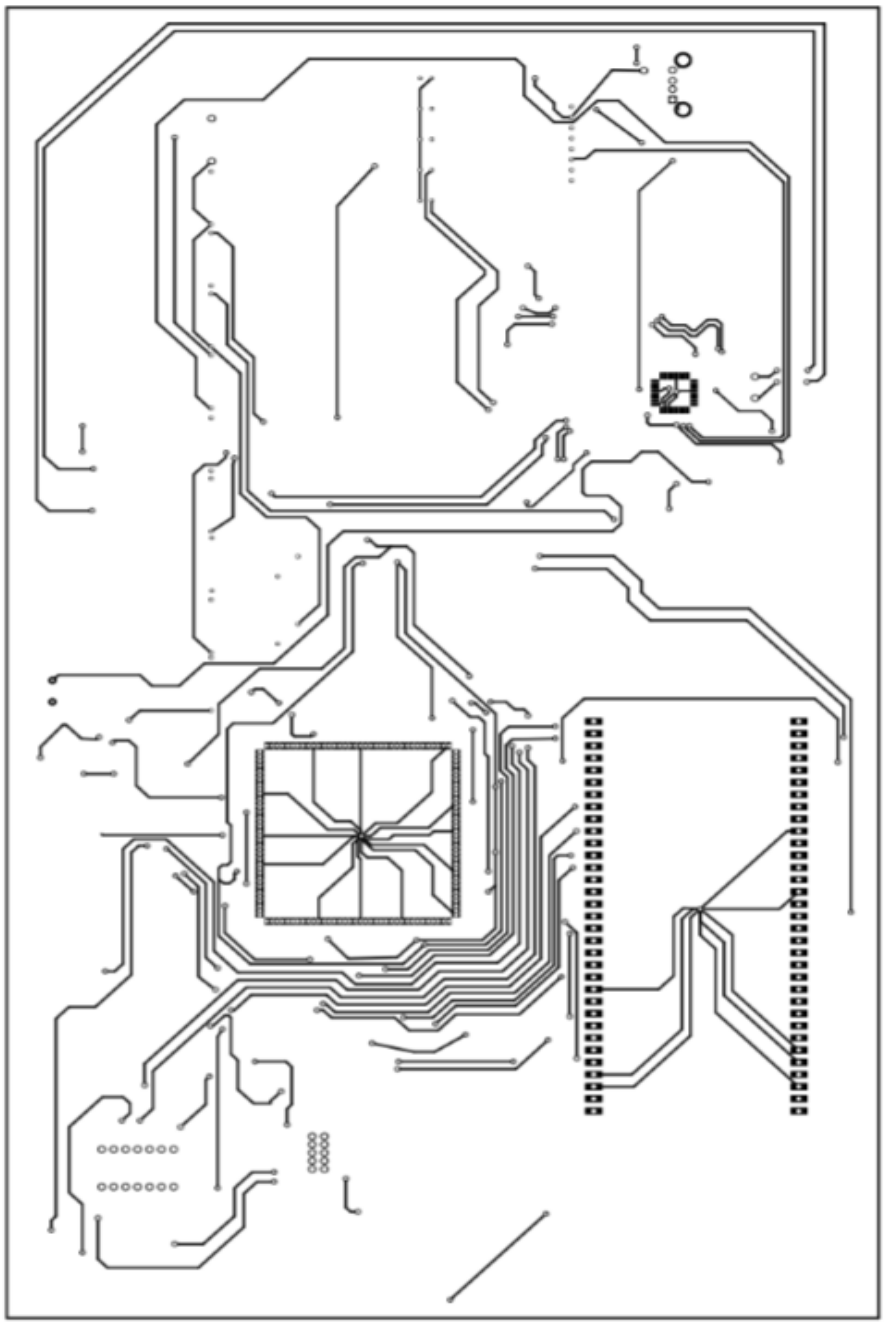
ФЮРА. XXXXXX.XXX



				ФЮРА. XXXXXX.XXX	
Изм	код	№ докум.			
Разраб.		Старшинов В.С.		Устройство межинтерфейсного взаимодействия AXI-TO-SPI Печатная плата Верхний слой	Лит: У; Масса; Масштаб
Проверил		Мальчуков А.Н.			Лист 1 Листов 1
Т. контр.					
Н. контр.					ТПУ ИК Группа 8В3А
Утв.		Мальчуков А.Н.			

ПРИЛОЖЕНИЕ 16. Нижний слой печатной платы

ФЮРА. XXXXXX.XXX



					ФЮРА. XXXXXX.XXX	
Изм	код	№докум.			Устройство межинтерфейсного взаимодействия AXI-TO-SPI Печатная плата Нижний слой	Лит: У; Масса; Масштаб
Разраб.		Старшинов В.С.				Лист 1 Листов 1
Проверил		Мальчуков А.Н.				
Т. контр.						
Н. контр.						ТПУ ИК Группа 8В3А
Утв.		Мальчуков А.Н.				

ПРИЛОЖЕНИЕ 17. Исходный код одного из модулей на языке VHDL

Модуль AXI_TO_SPI (axi_to_spi.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

  ---- объявление интерфейса ----
entity axi_to_spi is

GENERIC (
        data_width: integer := 8; -- ширина данных
        cmd_width: integer := 8;  -- ширина команд/инструкций
        CPOL: integer := 0;       -- полярность
        CPHA: integer := 0;       -- фаза
        slaves    : INTEGER:= 1   -- количество слейвов
        );

PORT(
        --- Глобальные переменные ---
       -- Тактовая частота работы
       clk:             IN STD_LOGIC;
       -- Глобальный сброс (активный - низкий)
       reset_n:           IN STD_LOGIC;
             --- Сигналы межинтерфейсного адаптера ---
       -- Сигнал занятости линии передачи
       I_AXI_BUSY:       IN STD_LOGIC;
       -- Сигнал разрешения передачи данных
       I_AXI_ENA:        OUT STD_LOGIC;
       -- Сигнал приемы данных с блока AXI_SLAVE
       I_AXI_RX_DATA:    IN STD_LOGIC_VECTOR (31 downto 0);
       -- Сигнал о готовности принять данные
       I_AXI_READY:      IN STD_LOGIC;
       -- Разрешение передачи данных на блок SPI_MASTER
       O_SPI_ENABLE:     IN STD_LOGIC;
       -- Сигнал занятости линии передачи на SPI_MASTER
       O_SPI_BUSY:       OUT STD_LOGIC;
  -- Установка скорости
       O_SPI_CLK_DIV:    OUT INTEGER;
       -- Установка адреса
       O_SPI_ADDR:       OUT INTEGER RANGE 0 TO slaves-1;
       -- Установка чтения/записи 1 порции данных/команд
       O_SPI_RW1:         OUT STD_LOGIC;
       -- Передаваемая первая порция данных
  O_SPI_TX_DATA1:    OUT STD_LOGIC_VECTOR (data_width-1 downto 0);
  -- Передаваемая первая порция команд
       O_SPI_TX_CMD1 :    OUT STD_LOGIC_VECTOR (cmd_width-2 downto 0);
       -- Установка чтения/записи 2 порции данных/команд
       O_SPI_RW2:         OUT STD_LOGIC;
       -- Передаваемая вторая порция данных
  O_SPI_TX_DATA2:    OUT STD_LOGIC_VECTOR (data_width-1 downto 0);
  -- Передаваемая вторая порция команд
       O_SPI_TX_CMD2 :    OUT STD_LOGIC_VECTOR (cmd_width-2 downto 0);
       -- Сигнал чтения данных
       O_SPI_RX_DATA:    IN STD_LOGIC_VECTOR (data_width-1 downto 0)
       );
end axi_to_spi;

-- внутренняя структура модуля
architecture Behavioral of axi_to_spi is
```

```
    TYPE machine IS(ready, axi_rx, tx_spi, spi_load);     -- состояния автомата
    SIGNAL state         : machine;                       -- установка флага
состояния
    SIGNAL message       : STD_LOGIC_VECTOR(31 DOWNTO 0); -- буфер хранимых данных
    SIGNAL spi_busy_prev : STD_LOGIC;                     -- предыдущий сигнал
занятости
    SIGNAL axi_rrdy_load : STD_LOGIC;                     -- готовность загрузки
    SIGNAL axi_busy      : STD_LOGIC;                    -- занятость линии передачи
    SIGNAL TX_DONE       : STD_LOGIC;                    -- флаг о завершении
передачи

begin
    PROCESS(clk, reset_n)
        VARIABLE spi_busy_cnt : INTEGER := 0;  --установка начального значения
счетчика мультиплексирования данных
      BEGIN
        IF(reset_n = '0') THEN     -- глобальный сброс
          spi_busy_cnt := 0;       -- счетчик на начало
          O_SPI_BUSY <= '0';       -- передачи данных нет
          axi_rrdy_load <= '0';    -- готовности загрузки тоже нет
          I_AXI_ENA <= '0';        -- и разрешения тоже нет
          TX_DONE <= '0';          -- да еще и флаг завершения передачи на нуле
          state <= ready;

        ELSIF(clk'EVENT AND clk = '1') THEN -- по переднему фронту

          CASE state IS -- начало работы автомата

            WHEN ready =>
              -- готовность загрузки данных
                IF (I_AXI_READY = '1') THEN
                    axi_rrdy_load <= '1';
                    axi_busy <= '1';
                    state <= axi_rx; -- переход в следующее состояние
                ELSE
                    state <= ready; -- зацикливание на текущем состоянии
                END IF;

           WHEN axi_rx =>
              -- запись принимаемых данных в буфер
                IF (axi_busy = '1') THEN
                    message <= I_AXI_RX_DATA; -- запись данных в регистр message
                    axi_rrdy_load <= '0'; -- загрузка завершена
                    state <= tx_spi; -- переход в следующее состояние
                ELSE
                    state <= ready; -- переход в предыдущее состояние
                END IF;

            WHEN tx_spi =>
               -- разбиение данных на инструкции и данные
               -- активизация счетчика мультиплексирования данных
                  spi_busy_prev <= O_SPI_ENABLE;
                  IF (spi_busy_prev = '1' AND O_SPI_ENABLE = '1') THEN
                      spi_busy_cnt := spi_busy_cnt + 1;
                  END IF;
              -- по флагу завершения работы перейти в следующее состояние
                  IF (TX_DONE = '1') THEN
                      state <= spi_load;
                  END IF;
              -- начало разбиения данных
                      CASE spi_busy_cnt IS
```

```
                        WHEN 0 =>
                     -- старшие 8 бит данных делятся на команду чтения записи и 1
порцию инструкций
                            O_SPI_RW1 <= message(31); -- команда чтения/записи
                            O_SPI_TX_CMD1  <= message(30 DOWNTO 24); -- порция
инструкций
                            O_SPI_TX_DATA1 <= (OTHERS => '0'); --данные пока не
передавать
                            TX_DONE <= '0';


                        WHEN 1 =>
                    -- следующие 8 бит отходят на 1 порцию данных
                            O_SPI_TX_DATA1 <= message(23 DOWNTO 16); --!!!!!
                            TX_DONE <= '0';


                        WHEN 2 =>
                    -- следующие 8 бит отводятся на 2 команду чтения/записи и 2
порцию инструкций
                            O_SPI_RW2 <= message(15);
                            O_SPI_TX_CMD2 <= message(14 DOWNTO 8);
                            TX_DONE <= '0';


                        WHEN 3 =>
                    -- следующие 8 бит отходят на 2 порцию данных
                            O_SPI_TX_DATA2 <= message(7 DOWNTO 0);
                            TX_DONE <= '1'; --флаг о завершении передачи данных


                        WHEN OTHERS => NULL; -- остальные по нулям
                     END CASE;


           WHEN spi_load =>
            -- загрузка в след блок
            IF(TX_DONE = '1') THEN
               TX_DONE <= '0';
               O_SPI_BUSY  <= '1'; -- передача данных в след блок
               state <= ready;  -- возвращение в начальное состояние
            END IF;

    END CASE;
   END IF;
 END PROCESS;

end Behavioral;
```

ПРИЛОЖЕНИЕ 18. Исходный код главного модуля/модуля тестирования на языке VHDL

Модуль AXI_TO_SPI_DEVICE (axi_to_spi_device.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

---- Объявление интерфейса ----
entity axi_to_spi_device is

    Generic (
            C_S_AXI_DATA_WIDTH : integer := 32;    -- Ширина данных
            C_S_AXI_ADDR_WIDTH : integer := 32;    -- Ширина адресного пространства
        -- диапазон адресного пространства
            C_BASEADDR : std_logic_vector := X"FFFFFFFF"; -- Максимально допустимый
адрес
            C_HIGHADDR : std_logic_vector := X"00000000"  -- Минимально допустимый
адрес
            slaves             : INTEGER  := 1;    -- число слейвов
            cmd_width          : INTEGER  := 8;    -- ширина команд
            d_width            : INTEGER  := 8;    -- ширина данных
            cpol               : INTEGER  := 0;    -- полярность
            cpha               : INTEGER  := 0     -- фаза
            );

    Port(
                        --- Глобальные параметры ---
        -- Глобальный тактовый сигнал
        S_AXI_ACLK      : in std_logic;
        -- Глобальный сигнал сброса. Данный сигнал активен при значении "0"
        S_AXI_ARESETN   : in std_logic;
        -- Кнопка «BTN0» на отладочной плате
        BTN             : in std_logic;
        -- Движковый переключатель «SW0» на отладочной плате
        SW0             : in std_logic;
        -- Движковый переключатель «SW1» на отладочной плате
        SW1             : in std_logic;
        -- Движковый переключатель «SW2» на отладочной плате
        SW2             : in std_logic;
        -- Движковый переключатель «SW3» на отладочной плате
        SW3             : in std_logic;
                        --- Входные порты ---
            --- Канал записи адреса ---
        -- Записываемый адрес (задается мастером, принимается слэйвом)
        S_AXI_AWADDR    : in std_logic_vector(C_S_AXI_ADDR_WIDTH-1 downto 0);
        -- Параметр допуска записываемого адреса. Данный сигнал идентифицирует
        -- привилегированный и защитный уровень транзакции, а также
        -- транзакции в качестве доступа к данным и доступа к командам
        S_AXI_AWPROT    : in std_logic_vector(2 downto 0);
        -- Достоверность записываемого адреса. Этот сигнал идентифицирует,
        -- что сигналы от мастера достоверны для записи адреса и данных
        S_AXI_AWVALID   : in std_logic;
        -- Готовность записи адреса. Этот сигнал идентифицирует, что слэйв
        -- готов принять адрес и подразумеваемые с ним сигналы
        S_AXI_AWREADY   : out std_logic;
            --- Канал записи данных ---
        -- Записываемые данные (записываются мастером, принимаются слэйвом)
        S_AXI_WDATA     : in std_logic_vector(C_S_AXI_DATA_WIDTH-1 downto 0);
        -- Запись стробов. Этот сигнал идентифицирует о действительных
```

```
        -- данных в стробе байтовых данных. Существует одна запись строба
        -- бит для каждого восьми битов шины данных записи
        S_AXI_WSTRB     : in std_logic_vector((C_S_AXI_DATA_WIDTH/8)-1 downto 0);
        -- Подтверждение записи. Данный сигнал подтверждает запись
        -- dданных и необходимых стробов
        S_AXI_WVALID    : in std_logic;
        -- Готовность записи данных . Этот сигнал говорит о готовности
        -- слэйвом принять записываемые данные
        S_AXI_WREADY    : out std_logic;
            --- Канал записи ответа/отклика/реакции ---
        -- Запись отклика данных. Этот сигнал присваивает статус
        -- записи транзакции
        S_AXI_BRESP : out std_logic_vector(1 downto 0);
        -- Подтверждение отклика данных. Данные сигнал символизирует, что канал
        -- сигнализирует о правильно записываемой реакции
        S_AXI_BVALID    : out std_logic;
        -- Готовность ответа. Этот сигнал говорит о готовности
        -- мастером принять отклик
        S_AXI_BREADY    : in std_logic;
                --- Канал чтения адреса ---
        -- Чтение адреса (инициируется мастером, принимается слэйвом)
        S_AXI_ARADDR    : in std_logic_vector(C_S_AXI_ADDR_WIDTH-1 downto 0);
        -- Тип защиты/доступа. Этот сигнал идентифицирует тип привилегии
        -- и защитного уровня транзакции, а также
        -- транзакции в качестве доступа к данным и доступа к командам
        S_AXI_ARPROT    : in std_logic_vector(2 downto 0);
        -- Достоверность считываемого адреса. Этот сигнал идентифицирует,
        -- что сигналы от мастера достоверны для считывания адреса и данных
        S_AXI_ARVALID   : in std_logic;
        -- Готовность чтения адреса. Этот сигнал идентифицирует, что слэйв
        -- готов считать адрес и подразумеваемые с ним сигналы
        S_AXI_ARREADY   : out std_logic;
                --- Канал чтения данных ---
        -- Считываемые данные (идентифицируются слэйвом)
        S_AXI_RDATA     : out std_logic_vector(C_S_AXI_DATA_WIDTH-1 downto 0);
        -- Чтение отклика данных. Этот сигнал присваивает статус
        -- чтения транзакции
        S_AXI_RRESP     : out std_logic_vector(1 downto 0);
        -- Подтверждение отклика данных. Данные сигнал символизирует, что канал
        -- сигнализирует о правильно считываемой реакции
        S_AXI_RVALID    : out std_logic;
        --- Готовность ответа. Этот сигнал говорит о готовности
        -- мастером принять считываемые данные и отклик
        S_AXI_RREADY    : in std_logic;

          ---- Выходные порты ----
         -- Сигналы SPI_MASTER –
         -- Тактовая частота передачи данных 5 МГц
         M_SPI_SCLK    : OUT    STD_LOGIC;
         -- Выбор ведомого устройства
         M_SPI_SS_N    : OUT    STD_LOGIC_VECTOR(slaves-1 DOWNTO 0);
         -- Линия передачи данных
         M_SPI_SDIO    : INOUT  STD_LOGIC
         );

END axi_to_spi_device;

      ---- Объявление внутренней структуры ----
architecture Behavioral of axi_to_spi_device is
type device_states is (waiting, data1, data2, data3, data4, response);
```

```
  signal state : device_states;    --- Сигнал соединения тактовой частоты
divider_freq с входной тактовой частотой SPI
    signal CLK_OUT_connection :  std_logic;
    -- Сигнал сброса тактовой частоты в divider_freq
    signal RESET : std_logic;
    -- Запись данных в регистр
    signal WDATA              :  std_logic_vector(31 downto 0);
    -- Код движкового переключателя
    signal DP_NUMBER :  std_logic_vector(3 downto 0);
    signal    AXI_READY_i  : std_logic;
    -- Сигнал соединения передаваемых данных с AXI на AXI_TO_SPI
    signal    AXI_RX_DATA_connection:   STD_LOGIC_VECTOR (31 downto 0);
    -- Сигнал соединения готовности передачи данных с AXI на AXI_TO_SPI
    signal    AXI_READY_connection:   std_logic;
    -- Сигнал соединения разрешения передачи с AXI на AXI_TO_SPI
    signal    AXI_ENA_connection:     std_logic;
    -- Сигнал соединения занятости канала передачи с AXI на AXI_TO_SPI
    signal    AXI_BUSY_connection:    std_logic;
    -- Сигнал соединения разрешения передачи с AXI_TO_SPI на SPI
    signal    SPI_ENABLE_connection:     STD_LOGIC;
    -- Сигнал соединения занятости канала передачи с AXI_TO_SPI на SPI
    signal    SPI_BUSY_connection:       STD_LOGIC;
    -- Сигнал соединения установки скорости передачи с AXI_TO_SPI на SPI
    signal    SPI_CLK_DIV_connection:    INTEGER;
    -- Сигнал соединения выбора ведомого устройства с AXI_TO_SPI на SPI
    signal    SPI_ADDR_connection:       INTEGER RANGE 0 TO slaves-1;
    -- Сигнал соединения передачи 1 команды чтения/записи с AXI_TO_SPI на SPI
    signal    SPI_RW1_connection:         STD_LOGIC;
    -- Сигнал соединения передачи 1 порции данных с AXI_TO_SPI на SPI
    signal    SPI_TX_DATA1_connection:    STD_LOGIC_VECTOR (d_width-1 downto 0);
    -- Сигнал соединения передачи 1 порции команд с AXI_TO_SPI на SPI
    signal    SPI_TX_CMD1_connection :    STD_LOGIC_VECTOR (cmd_width-2 downto 0);
    -- Сигнал соединения передачи 2 команды чтения/записи с AXI_TO_SPI на SPI
    signal    SPI_RW2_connection:         STD_LOGIC;
    -- Сигнал соединения передачи 2 порции данных с AXI_TO_SPI на SPI
    signal    SPI_TX_DATA2_connection:    STD_LOGIC_VECTOR (d_width-1 downto 0);
    -- Сигнал соединения передачи 2 порции команд с AXI_TO_SPI на SPI
    signal    SPI_TX_CMD2_connection :    STD_LOGIC_VECTOR (cmd_width-2 downto 0);
    -- Сигнал соединения считываемых данных с AXI_TO_SPI на SPI
    signal    SPI_RX_DATA_connection:    STD_LOGIC_VECTOR (d_width-1 downto 0);



-- объявление компонента AXI_SLAVE --
 COMPONENT axi_slave IS

 GENERIC(
         C_S_AXI_DATA_WIDTH            : integer             := 32;
         C_S_AXI_ADDR_WIDTH            : integer             := 32;
         C_BASEADDR                    : std_logic_vector    := X"FFFFFFFF";
         C_HIGHADDR                    : std_logic_vector    := X"00000000"
        );

 Port(

     -- Глобальные параметры
     S_AXI_ACLK      : in std_logic;
     S_AXI_ARESETN   : in std_logic;
```

```
                --- Канал записи адреса ---

     S_AXI_AWADDR    : in std_logic_vector(C_S_AXI_ADDR_WIDTH-1 downto 0);
     S_AXI_AWPROT    : in std_logic_vector(2 downto 0);
     S_AXI_AWVALID   : in std_logic;
     S_AXI_AWREADY   : out std_logic;
                --- Канал записи данных ---
     S_AXI_WDATA     : in std_logic_vector(C_S_AXI_DATA_WIDTH-1 downto 0);
     S_AXI_WSTRB     : in std_logic_vector((C_S_AXI_DATA_WIDTH/8)-1 downto 0);
     S_AXI_WVALID    : in std_logic;
     S_AXI_WREADY    : out std_logic;
                --- Канал записи отклика/ответа ---
     S_AXI_BRESP     : out std_logic_vector(1 downto 0);
     S_AXI_BVALID    : out std_logic;
     S_AXI_BREADY    : in std_logic;
                --- Канал чтения адреса ---
     S_AXI_ARADDR    : in std_logic_vector(C_S_AXI_ADDR_WIDTH-1 downto 0);
     S_AXI_ARPROT    : in std_logic_vector(2 downto 0);
     S_AXI_ARVALID   : in std_logic;
     S_AXI_ARREADY   : out std_logic;
                       --- Канал чтения данных ---
     S_AXI_RDATA     : out std_logic_vector(C_S_AXI_DATA_WIDTH-1 downto 0);
     S_AXI_RRESP     : out std_logic_vector(1 downto 0);
     S_AXI_RVALID    : out std_logic;
     S_AXI_RREADY    : in std_logic;
                       --- Взаимодействие со следующим блоком AXI_TO_SPI ---
     O_AXI_RX_DATA   : out std_logic_vector(C_S_AXI_DATA_WIDTH-1 downto 0);
     O_AXI_READY     : out std_logic;
     O_AXI_ENA       : in std_logic;
     O_AXI_BUSY      : out std_logic
  );
end COMPONENT axi_slave;

-- объявление компонента AXI_TO_SPI
COMPONENT axi_to_spi IS

    GENERIC (
            slaves    : INTEGER:= 1;
            cmd_width: integer := 8;
            data_width: integer := 8;
            CPOL: integer := 0;
            CPHA: integer := 0
            );

    PORT(
            -- Глобальные параметры
        clk:              IN STD_LOGIC;
        reset_n:           IN STD_LOGIC;
            -- Сигналы взаимодействия с блоком AXI межинтерфейсного адаптера
        I_AXI_BUSY:       IN STD_LOGIC;
        I_AXI_ENA:        OUT STD_LOGIC;
        I_AXI_RX_DATA:    IN STD_LOGIC_VECTOR (31 downto 0);
        I_AXI_READY:      IN STD_LOGIC;
            -- Сигналы взаимодействия с блоком SPI межинтерфейсного адаптера
        O_SPI_ENABLE:     IN STD_LOGIC;
        O_SPI_BUSY:       OUT STD_LOGIC;
     -- O_SPI_CPOL:       OUT INTEGER;
     -- O_SPI_CPHA:       OUT INTEGER;
        O_SPI_CLK_DIV:    OUT INTEGER;
        O_SPI_ADDR:       OUT INTEGER RANGE 0 TO slaves-1;
        O_SPI_RW1:        OUT STD_LOGIC;
```

```
        O_SPI_TX_DATA1:   OUT STD_LOGIC_VECTOR (data_width-1 downto 0);
        O_SPI_TX_CMD1 :   OUT STD_LOGIC_VECTOR (cmd_width-2 downto 0);
        O_SPI_RW2:        OUT STD_LOGIC;
        O_SPI_TX_DATA2:   OUT STD_LOGIC_VECTOR (data_width-1 downto 0);
        O_SPI_TX_CMD2 :   OUT STD_LOGIC_VECTOR (cmd_width-2 downto 0);
        O_SPI_RX_DATA:    IN STD_LOGIC_VECTOR (data_width-1 downto 0)
        );

end COMPONENT axi_to_spi;

-- объявление компонента SPI_MASTER
COMPONENT spi_master IS

  GENERIC(
    slaves    : INTEGER:= 1;
    cmd_width : INTEGER := 8;
    d_width   : INTEGER := 8;
    cpol      : INTEGER := 0;
    cpha      : INTEGER := 0
         );

  PORT(
        -- Глобальные параметры
      M_SPI_CLOCK    : IN     STD_LOGIC;
      M_SPI_RESET_N  : IN     STD_LOGIC;
        -- Сигналы приема данных от межинтерфейсного адаптера
      M_SPI_ENABLE   : OUT     STD_LOGIC;
   -- M_SPI_CPOL     : IN     STD_LOGIC;
   -- M_SPI_CPHA     : IN     STD_LOGIC;
      M_SPI_CLK_DIV  : IN     INTEGER;
      M_SPI_ADDR     : IN     INTEGER RANGE 0 TO slaves-1;
      M_SPI_RW1      : IN     STD_LOGIC;
      M_SPI_TX_CMD1  : IN     STD_LOGIC_VECTOR(cmd_width-2 DOWNTO 0);
      M_SPI_TX_DATA1 : IN     STD_LOGIC_VECTOR(d_width-1 DOWNTO 0);
      M_SPI_RW2      : IN     STD_LOGIC;
      M_SPI_TX_CMD2  : IN     STD_LOGIC_VECTOR(cmd_width-2 DOWNTO 0);
      M_SPI_TX_DATA2 : IN     STD_LOGIC_VECTOR(d_width-1 DOWNTO 0);
      M_SPI_SCLK     : OUT    STD_LOGIC;
      M_SPI_SS_N     : OUT    STD_LOGIC_VECTOR(slaves-1 DOWNTO 0);
      M_SPI_SDIO     : INOUT  STD_LOGIC;
      M_SPI_BUSY     : IN     STD_LOGIC;
      M_SPI_RX_DATA  : OUT    STD_LOGIC_VECTOR(d_width-1 DOWNTO 0)
       );

end COMPONENT spi_master;

-- объявление компонента DIVIDER_FREQ
COMPONENT divider_freq IS

PORT(
  -- Входная тактовая частота 250МГц
   CLK_IN: IN STD_LOGIC;
  -- Выходная тактовая частота 25МГц
   CLK_OUT: OUT STD_LOGIC;
  -- Глобальный сброс
   RESET: IN STD_LOGIC
     );

 end COMPONENT divider_freq;

begin
```

```
-- Соединение компонентов AXI и AXI_connection
  axi_slave_0:  axi_slave

    GENERIC MAP(
                C_S_AXI_DATA_WIDTH => C_S_AXI_DATA_WIDTH,
                C_S_AXI_ADDR_WIDTH => C_S_AXI_ADDR_WIDTH,
                C_BASEADDR => C_BASEADDR,
                C_HIGHADDR => C_HIGHADDR
                )

          PORT MAP(
                   S_AXI_ACLK     =>   S_AXI_ACLK,
                   S_AXI_ARESETN  =>   S_AXI_ARESETN,

                   S_AXI_AWADDR   =>   S_AXI_AWADDR,
                   S_AXI_AWPROT   =>   S_AXI_AWPROT,
                   S_AXI_AWVALID  =>   S_AXI_AWVALID,
                   S_AXI_AWREADY  =>   S_AXI_AWREADY,

                   S_AXI_WDATA    =>   S_AXI_WDATA,
                   S_AXI_WSTRB    =>   S_AXI_WSTRB,
                   S_AXI_WVALID   =>   S_AXI_WVALID,
                   S_AXI_WREADY   =>   S_AXI_AWREADY,

                   S_AXI_BRESP    =>   S_AXI_BRESP,
                   S_AXI_BVALID   =>   S_AXI_BVALID,
                   S_AXI_BREADY   =>   S_AXI_BREADY,

                   S_AXI_ARADDR   =>   S_AXI_ARADDR,
                   S_AXI_ARPROT   =>   S_AXI_ARPROT,
                   S_AXI_ARVALID  =>   S_AXI_ARVALID,
                   S_AXI_ARREADY  =>   S_AXI_ARREADY,

                   S_AXI_RDATA    =>   S_AXI_RDATA,
                   S_AXI_RRESP    =>   S_AXI_RRESP,
                   S_AXI_RVALID   =>   S_AXI_RVALID,
                   S_AXI_RREADY   =>   S_AXI_RREADY,

                   O_AXI_RX_DATA  =>   AXI_RX_DATA_connection,
                   O_AXI_READY    =>   AXI_READY_connection,
                   O_AXI_ENA      =>   AXI_ENA_connection,
                   O_AXI_BUSY     =>   AXI_BUSY_connection
                  );

 -- Соединение компонентов AXI_connection - AXI_TO_SPI и AXI_TO_SPI - SPI_connection
   axi_to_spi_0:  axi_to_spi

      PORT MAP    (   clk            =>   S_AXI_ACLK,
                      reset_n        =>   S_AXI_ARESETN,
                      I_AXI_BUSY     =>   AXI_BUSY_connection,
                      I_AXI_ENA      =>   AXI_ENA_connection,
                      I_AXI_RX_DATA  =>   AXI_RX_DATA_connection,
                      I_AXI_READY    =>   AXI_READY_connection,
                      O_SPI_ENABLE   =>   SPI_ENABLE_connection,
                      O_SPI_BUSY     =>   SPI_BUSY_connection,
                      O_SPI_CLK_DIV  =>   SPI_CLK_DIV_connection,
                      O_SPI_ADDR     =>   SPI_ADDR_connection,
                      O_SPI_RW1      =>   SPI_RW1_connection,
                      O_SPI_TX_DATA1 =>   SPI_TX_DATA1_connection,
                      O_SPI_TX_CMD1  =>   SPI_TX_CMD1_connection,
```

```
                    O_SPI_RW2      =>   SPI_RW2_connection,
                    O_SPI_TX_DATA2 =>   SPI_TX_DATA2_connection,
                    O_SPI_TX_CMD2  =>   SPI_TX_CMD2_connection,
                    O_SPI_RX_DATA  =>   SPI_RX_DATA_connection
                );


-- Соединение компонентов  SPI_connection - SPI
    spi_master_0:  spi_master

       GENERIC MAP    (
                      slaves         =>   slaves,
                      cmd_width      =>   cmd_width,
                      d_width        =>   d_width,
                      cpol           =>   cpol,
                      cpha           =>   cpha
                      )

         PORT MAP(
                    M_SPI_CLOCK           =>  CLK_OUT_connection,
                    M_SPI_RESET_N         =>  S_AXI_ARESETN,
                    M_SPI_ENABLE          =>  SPI_ENABLE_connection,
                    M_SPI_CLK_DIV         =>  SPI_CLK_DIV_connection,
                    M_SPI_ADDR            =>  SPI_ADDR_connection,
                    M_SPI_RW1             =>  SPI_RW1_connection,
                    M_SPI_TX_CMD1         =>  SPI_TX_CMD1_connection,
                    M_SPI_TX_DATA1        =>  SPI_TX_DATA1_connection,
                    M_SPI_RW2             =>  SPI_RW2_connection,
                    M_SPI_TX_CMD2         =>  SPI_TX_CMD2_connection,
                    M_SPI_TX_DATA2        =>  SPI_TX_DATA2_connection,
                    M_SPI_SCLK            =>  M_SPI_SCLK,
                    M_SPI_SS_N            =>  M_SPI_SS_N,
                    M_SPI_SDIO            =>  M_SPI_SDIO,
                    M_SPI_BUSY            =>  SPI_BUSY_connection,
                    M_SPI_RX_DATA         =>  SPI_RX_DATA_connection
                     );

  -- Соединение компонентов  AXI - DIVIDER_FREQ - SPI
    divider_freq_0:  divider_freq

        PORT MAP(
                    CLK_IN          =>  S_AXI_ACLK,
                    RESET           =>  RESET,
                    CLK_OUT         =>  CLK_OUT_connection
                );
   process (S_AXI_ACLK)
  begin
  if rising_edge(S_AXI_ACLK) then
  if S_AXI_ARESETN = '0' then
            state <= waiting;
            else

    case state is

            when waiting =>

            DP_NUMBER[0]  <= SW0;
            DP_NUMBER[1]  <= SW1;
            DP_NUMBER[2]  <= SW2;
            DP_NUMBER[3]  <= SW3;
```

```
                    case DP_NUMBER is
                    when “0001” =>
                        state <= data1;

                    when “0010” =>
                         state <= data2;

                    when “0100” =>
                         state <= data3;

                    when “1000” =>
                         state <= data4;
                   end case;

               when data1 =>
               WDATA <= X"11223344";
               state <= response;

               when data2 =>
               WDATA <= X"AC3275DE";
               state <= response;

               when data3 =>
               WDATA <= X"26DFBE34";
               state <= response;

               when data4 =>
               WDATA <= X"CBEFF3A2";
               state <= response;

               when response =>
               if (BTN <= '1') THEN
                      S_AXI_AWVALID <= '1';
                      S_AXI_WVALID <= '1';

               else
                      S_AXI_AWVALID <= '0';
                      S_AXI_WVALID <= '0';

              end if;
       WHEN OTHERS => NULL; -- остальные по нулям

               S_AXI_WDATA <= WDATA;


        end case;
       end if;
     end if;
  end process;

end Behavioral;
```

ПРИЛОЖЕНИЕ 19. Исходный код главного модуля IP-core второго исполнения устройства на языке Verilog

```
module design_1_wrapper
   (diff_clock_rtl_clk_n,
    diff_clock_rtl_clk_p,
    reset_rtl,
    reset_rtl_0,
    s00_axi_wready,
    sclk,
    sdio,
    ss_n);
  input diff_clock_rtl_clk_n;
  input diff_clock_rtl_clk_p;
  input reset_rtl;
  input reset_rtl_0;
  output s00_axi_wready;
  output sclk;
  inout sdio;
  output [0:0]ss_n;

  wire diff_clock_rtl_clk_n;
  wire diff_clock_rtl_clk_p;
  wire reset_rtl;
  wire reset_rtl_0;
  wire s00_axi_wready;
  wire sclk;
  wire sdio;
  wire [0:0]ss_n;

  design_1 design_1_i
       (.diff_clock_rtl_clk_n(diff_clock_rtl_clk_n),
        .diff_clock_rtl_clk_p(diff_clock_rtl_clk_p),
        .reset_rtl(reset_rtl),
        .reset_rtl_0(reset_rtl_0),
        .s00_axi_wready(s00_axi_wready),
        .sclk(sclk),
        .sdio(sdio),
        .ss_n(ss_n));
endmodule
```

ПРИЛОЖЕНИЕ 20. Исходный код части программы для soft-процессора Microblaze для второго исполнения устройства на языке С

```
#include <stdio.h>
#include <stdlib.h>
#include "xparameters.h"
#include "xil_cache.h"
#include "xspi.h"
#include "spi_header.h"
#include "xbasic_types.h"

#define SPI_DEVICE_ID           XPAR_SPI_0_DEVICE_ID
#define SPI_SELECT 0x01
#define LED_CHANNEL 1

int main()
{
	static XSpi Spi;

	int Status;
	u8 WriteBuffer[10];
	u8 ReadBuffer[10];
	XSpi_Config *ConfigPtr;   /* Отметка для конфигурации данных */

	Xil_ICacheEnable();
	Xil_DCacheEnable();

	xil_printf("---Entering---\n\r");

	 // Инициализация SPI протокола для готовности его использования
	ConfigPtr = XSpi_LookupConfig(SPI_DEVICE_ID);
	if (ConfigPtr == NULL) {
		return XST_DEVICE_NOT_FOUND;
	}

	Status = XSpi_CfgInitialize(&Spi, ConfigPtr,ConfigPtr->BaseAddress);
	if (Status == XST_SUCCESS)
	{
		xil_printf("Spi Initialize success!\n\r");
	}
	else
	{
		xil_printf("Spi Initialize failed!\n\r");
	}

// Устанавливаем устройство SPI в качестве ведущего и в режиме
// что сигнал выбора подчиненного устройства не переключается для каждого байта
передачи
	Status = XSpi_SetOptions(&Spi, XSP_MASTER_OPTION | XSP_MANUAL_SSELECT_OPTION);
	if (Status != XST_SUCCESS)
	{
		return XST_FAILURE;
	}

	// Установка ведомого устройства для того, чтобы он считал данные
	// read and written using the SPI bus
	XSpi_GetSlaveSelect(&Spi);

	Status = XSpi_SetSlaveSelect(&Spi, SPI_SELECT);
	if (Status != XST_SUCCESS) {
```

```
		return XST_FAILURE;
	}

	// Старт передачи задержек по SPI
	Status = XSpi_Start(&Spi);
	if (Status != XST_SUCCESS) {
		return XST_FAILURE;
	}

	//отключение глобальных задержек
	XSpi_IntrGlobalDisable(&Spi);

	//Выход АЦП для установки двунаправленной передачи данных по SDIO, so we
	//Установка регистров записи данных (по даташитам)
	WriteBuffer[0] = 0x00;
	WriteBuffer[1] = 0x00;
	WriteBuffer[2] = 0x99;
	Status = XSpi_Transfer(&Spi, WriteBuffer, ReadBuffer, 3);
	if (Status != XST_SUCCESS) {
		return XST_FAILURE;
	}

	//Считывание данных
	WriteBuffer[0] = 0x80;
	WriteBuffer[1] = 0x08;
	Status = XSpi_Transfer(&Spi, WriteBuffer, ReadBuffer, 3);
	Buffer[0] = ReadBuffer[2];
	if (Status != XST_SUCCESS) {
		return XST_FAILURE;
	}
	xil_printf("Buffer[2] = 0x%x\r\n", Buffer[2]);

	xil_printf("---Exiting main---\n\r");

	Xil_DCacheDisable();
	Xil_ICacheDisable();

	return 0;
}
```

ПРИЛОЖЕНИЕ 21. Временные показатели научно-исследовательского проекта

Таблица 40. – Временные показатели научного исследования

№	Содержание работ	Мин. время выполнения (дн.)			Макс. время выполнения (дн.)			Исполнители	Длительность работ в рабочих днях			Длительность работ в календарных днях		
		И1	И2	И3	И1	И2	И3		И1	И2	И3	И1	И2	И3
1	Составление и утверждение технического задания	4	4	4	5	5	5	Р	4	4	4	5	5	5
2	Подбор и изучение материалов по теме	8	8	8	16	16	16	Р, И	9	9	9	16	17	15
3	Изучение уже существующих решений в данной области	8	8	8	11	11	11	И	8	8	8	11	9	11
4	Выбор компонентов для реализации устройства	5	5	5	11	11	11	Р, И	9	9	9	11	10	11
5	Календарное планирование работ по теме	2	2	2	5	5	5	Р, И	4	4	4	5	5	5
6	Изучение технической документации интерфейсов	8	12	12	24	26	26	И	12	14	14	28	30	30
7	Составление структурно-функциональной схемы	5	6	4	9	11	8	И	6	7	5	11	13	10

Продолжение табл. 40.

8	Проектирование ГСА автоматов с указанием возможных состояний	3	5	4	8	10	9	И	4	6	5	10	12	12
9	Выбор языка описания аппаратуры для программирования	1	1	1	2	2	2	И	2	2	2	2	2	2
10	Программирование блоков структурно-функциональной схемы на основании ГСА к ним	12	15	14	18	24	20	И	15	17	16	24	28	26
11	Устранение всех ошибок в программе и ее компиляция	2	4	3	7	9	8	И	5	6	5	9	10	10
12	Разработка методики тестирования	2	4	3	4	6	5	И	4	4	4	6	7	6
13	Тестирование программы на временных диаграммах	4	4	4	6	6	6	И	4	4	4	6	6	6
14	Апробация разработанного устройства на железе	10	14	12	14	18	16	И	12	16	14	18	20	20

Продолжение табл. 40.

15	Выбор программного обеспечения для проектирования	1	1	1	1	1	1	И	1	1	1	1	1	1
16	Проектирование условного-графических обозначений и посадочных мест элементов	1	1	1	1	1	1	И	1	1	1	1	1	1
17	Ввод принципиальной схемы разрабатываемого устройства	1	1	1	2	2	2	И	1	1	1	2	2	2
18	Компоновка и трассировка элементов на печатной плате	4	4	4	9	9	9	И	5	5	5	9	10	10
19	Оценка эффективности полученных результатов	1	1	1	2	2	2	Р, И	1	1	1	2	2	2
20	Оценка целесообразности проведения дальнейших исследований по данной теме	1	1	1	2	2	2	Р, И	1	1	1	2	2	2
	Итого:											179	194	188

Р – Научный руководитель; И – Инженер - программист

ПРИЛОЖЕНИЕ 22. Календарный план для выполнения научно-исследовательского проекта

ДИАГРАММА ГАНТА

	Название задачи	Длительнс	Начало	Окончани	Предш
1	**Разработка технического задания**	**5 дней**	**Пн 03.10.16**	**Пт 07.10.16**	
2	Составление и утверждение технического задания	5 дней	Пн 03.10.16	Пт 07.10.16	
3	Составление технического задания закончено	0 дней	Пт 07.10.16	Пт 07.10.16	
4	**Аналитический обзор**	**43 дней**	**Пн 10.10.16**	**Ср 07.12.16**	
5	Подбор и изучение материалов по теме	16 дней	Пн 10.10.16	Пн 31.10.16	2
6	Изучение уже существующих решений в данной области	11 дней	Вт 01.11.16	Вт 15.11.16	5
7	Выбор компонентов для реализации устройства	11 дней	Ср 16.11.16	Ср 30.11.16	6
8	Календарное планирование работ по теме	5 дней	Чт 01.12.16	Ср 07.12.16	7
9	Аналитический обзор завершен	0 дней	Ср 07.12.16	Ср 07.12.16	
10	**Проектирование межинтерфейсного адаптера**	**45 дней**	**Чт 08.12.16**	**Ср 08.02.17**	
11	Изучение технической документации интерфейсов	24 дней	Чт 08.12.16	Вт 10.01.17	8
12	Составление структурно-функциональной схемы	11 дней	Ср 11.01.17	Ср 25.01.17	11
13	Проектирование ГСА автоматов с указанием возможных состояний	10 дней	Чт 26.01.17	Ср 08.02.17	12
14	Этап проектирования окончен	0 дней	Ср 08.02.17	Ср 08.02.17	
15	**Реализация межинтерфейсного адаптера**	**35 дней**	**Чт 09.02.17**	**Ср 29.03.17**	
16	Выбор языка описания аппаратуры для программирования	2 дней	Чт 09.02.17	Пт 10.02.17	13
17	Программирование блоков СФС на основании ГСА автоматов к ним	24 дней	Пн 13.02.17	Чт 16.03.17	16
18	Устранение всех ошибок в программе и ее компиляция	9 дней	Пт 17.03.17	Ср 29.03.17	17
19	Реализация завершена	0 дней	Ср 29.03.17	Ср 29.03.17	
20	**Тестирование, апробация**	**30 дней**	**Чт 30.03.17**	**Ср 10.05.17**	
21	Разработка методики тестирования	6 дней	Чт 30.03.17	Чт 06.04.17	18
22	Тестирование программы на временных диаграммах	6 дней	Пт 07.04.17	Пт 14.04.17	21
23	Апробация разработанного устройства на железе	18 дней	Пн 17.04.17	Ср 10.05.17	22
24	Этап тестирования, апробации закончен	0 дней	Ср 10.05.17	Ср 10.05.17	
25	**Компоновка и трассировка элементов на печатной плате**	**13 дней**	**Чт 11.05.17**	**Пн 29.05.17**	
26	Выбор ПО для проектирования	1 день	Чт 11.05.17	Чт 11.05.17	23
27	Проектирование УГО и посадочных мест элементов	1 день	Пт 12.05.17	Пт 12.05.17	26
28	Ввод принципиальной схемы разрабатываемого устройства	2 дней	Пн 15.05.17	Вт 16.05.17	27
29	Компоновка и трассировка элементов на печатной плате	9 дней	Ср 17.05.17	Пн 29.05.17	28
30	Этап компоновки и трассировки завершен	0 дней	Пн 29.05.17	Пн 29.05.17	
31	**Обобщение и оценка результатов**	**4 дней**	**Вт 30.05.17**	**Пт 02.06.17**	
32	Оценка эффективности полученных результатов	2 дней	Вт 30.05.17	Ср 31.05.17	29
33	Оценка целесообразности проведения дальнейшиз исследований по данной теме	2 дней	Чт 01.06.17	Пт 02.06.17	32
34	Результаты работы подведены	0 дней	Пт 02.06.17	Пт 02.06.17	

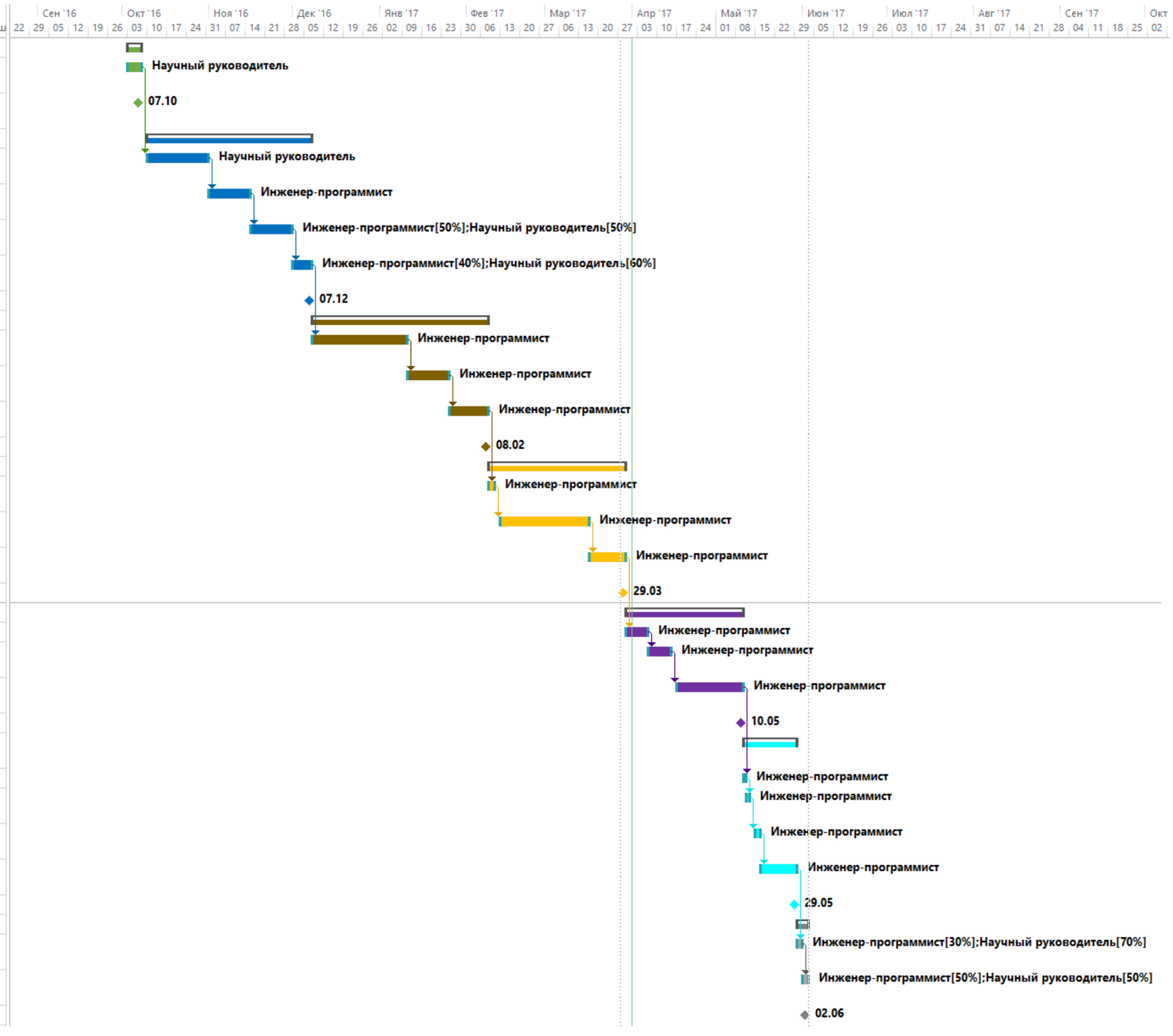


Рисунок 69 – Календарный план для выполнения научно-исследовательского проекта

ПРИЛОЖЕНИЕ 23. План эвакуации при чрезвычайных ситуациях

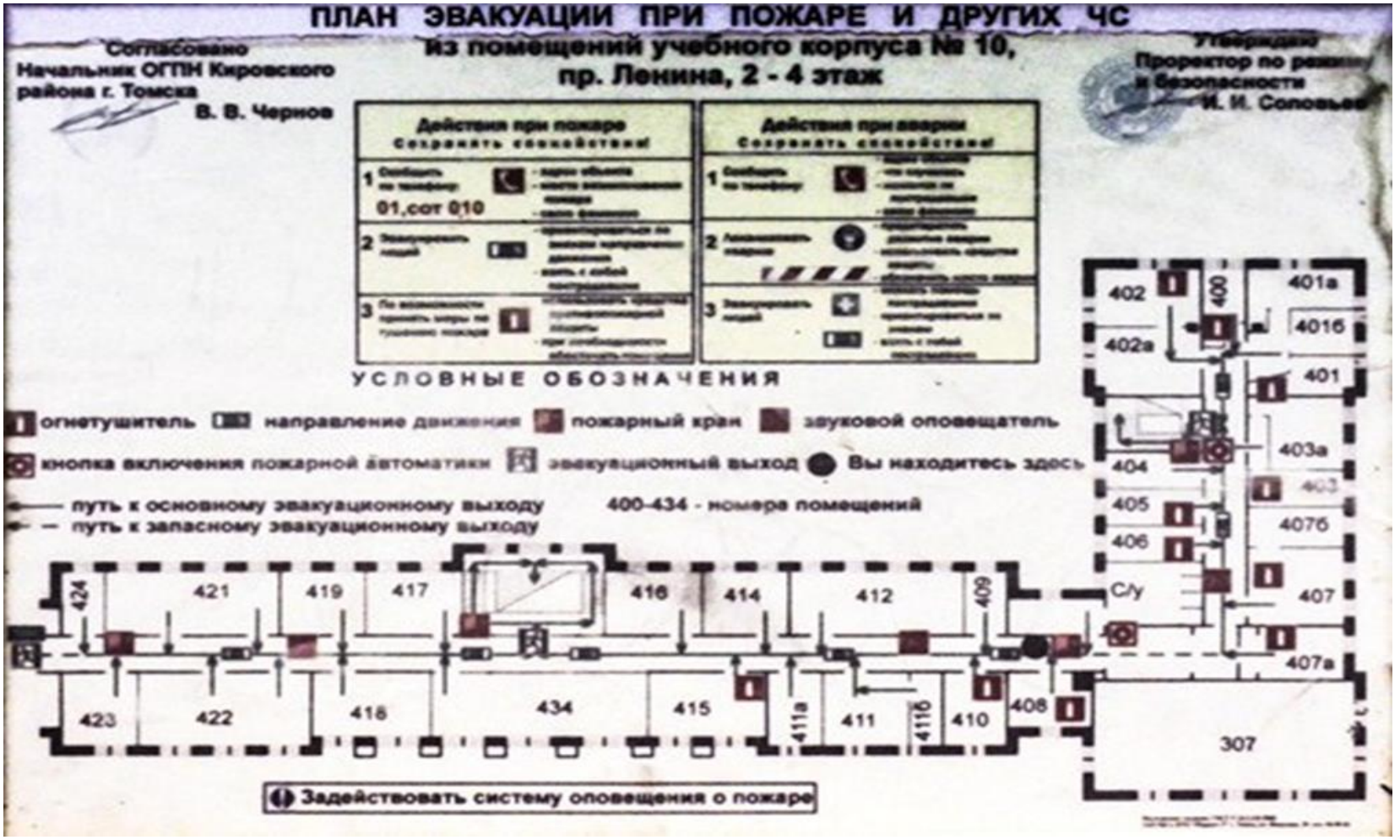


Рисунок 70 – План эвакуации при чрезвычайных ситуациях