МИНОБРНАУКИ РОССИИ

Государственное образовательное учреждение высшего профессионального образования

«Санкт-Петербургский государственный электротехнический университет

«ЛЭТИ» имени В.И.Ульянова (Ленина)»

(СПбГЭТУ)

**Факультет Радиотехники и телекоммуникаций**

**Кафедра Телевидения и видеотехники**

*К защите допустить*:

**Заведующий кафедрой**

**Дипломная работа**

**На тему:** Разработка ядра передающей части системы коррекции ошибок в IP-телевидении для ПЛИС Xilinx

*Дипломант \_\_\_\_\_\_\_\_\_\_\_ А.С. Антипов*

*Руководитель \_\_\_\_\_\_\_\_\_\_\_ К.В.Иванов*

*Консультант по экономическому*

*обоснованию \_\_\_\_\_\_\_\_\_\_\_ Г.В. Голигузова*

*Консультант по БЖД \_\_\_\_\_\_\_\_\_\_\_ С.В. Петухова*

Санкт-Петербург

2012 г.

Реферат

93 стр., 48 рис., 5 табл., 4 прил., 9 ист.

Цифровой программируемый генератор импульсов, цифро-аналоговая система с использованием программируемой логики.

Разработана принципиальная электрическая схема цифрового генератора сигналов произвольной формы на основе выбранных электронных компонентов. Разработана схема работы программируемой логики. Спроектировано и реализовано управляющее программное обеспечение. Использованы следующие среды моделирования и разработки: MicroCap 8, P-CAD 2006, Altium Designer 08, Quartus II 8.1, Qt Creator 1.1. Разработанный генератор импульсов полностью соответствует требованиям технического задания.

В тексте диссертации приведено руководство программиста и руководство пользователя. Приведен обзор существующих аналогов, их достоинства и недостатки. В приложении приведен исходный код управляющего ПО, а также код «прошивки» микросхемы программируемой логики.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 4

1 СТРУКТУРА ЛОКАЛЬНОЙ СЕТИ ДЛЯ ПЕРЕДАЧИ IP- ТЕЛЕВИДЕНИЯ 6

1.1 ОСОБЕННОСТИ РЕАЛИЗАЦИИ IP-ТЕЛЕВИДЕНИЯ 6

1.2 ИЕРАРХИЯ ПРОТОКОЛОВ 9

2 ФОРМИРОВАНИЕ МЕДИА ПОТОКА IP-ТЕЛЕВИДЕНИЯ 21

2.1 СТРУКТУРА ТРАНСПОРТНОГО ПОТОКА MPEG 21

2.2 СТРУКТУРА МЕДИА ПАКЕТОВ 23

3 СИСТЕМА КОРРЕКЦИИ ОШИБОК 25

3.1 НАЗНАЧЕНИЕ СХЕМЫ FEC 25

3.2 ФОРМИРОВАНИЕ RTP ЗАГОЛОВКА МЕДИА ПАКЕТОВ 26

3.3 ФОРМИРОВАНИЕ СХЕМЫ FEC 26

3.4 СТРУКТУРА FEC ПАКЕТА 29

3.5 ФОРМИРОВАНИЕ ЗАГОЛОВКОВ FEC-ПАКЕТОВ 29

3.5.1 ФОРМИРОВАНИЕ RTP ЗАГОЛОВКА 29

3.5.1 ФОРМИРОВАНИЕ FEC ЗАГОЛОВКА 30

4 43

ВВЕДЕНИЕ

В современном мире компьютерные сети выполняют важнейшую роль, так как являются основой глобального информационного обмена. Компьютерные сети повседневно используются для передачи гигантских объемов разнообразной информации: файлов данных, электронной почты, мультимедиа данных, голосового трафика и т.д. Сети, построенные на базе стека протоколов TCP/IP, называются IP-сетями. Под стеком протоколов TCP/IP понимается стандартизованный набор сетевых протоколов, который позволяет создавать как небольшие локальные, так и глобальные сети.

На сегодняшний день IP-сети глубоко интегрированы во все сферы деятельности человека, не является исключением и передача видеосигналов по IP-сетям. Передача видеосигналов по IP-сетям часто используется для реализации таких сервисов как IPTV. IPTV принято называть цифровую технологию многопрограммного интерактивного телевизионного вещания в IP-сети с помощью пакетной передачи видеоданных по IP-протоколу (Video over IP)

По сравнению с обычным телевидением, IPTV позволяет потребителю не только в надлежащем качестве просматривать один из десятков или сотен каналов, но и предоставляет ряд дополнительных услуг, обеспечивающих обратную связь между телезрителем и источником контента. Это дает нам право утверждать, что у данного сервиса есть огромное будущее.

Как правило, из вещательных центров передаются так называемые пакеты каналов, содержащие несколько десятков различных телевизионных сигналов. Поскольку IP-сеть использует пакетный способ передачи информации, то телевизионные сигналы необходимо предварительно подготовить для передачи по IP-сети. В процессе такой подготовки производится обработка телевизионного сигнала алгоритмами сжатия (такими, как MPEG-2, MPEG-4/H.264), формирование транспортного потока MPEG, имеющего пакетную структуру и последующая инкапсуляция образованных транспортных пакетов в IP-пакеты, последовательность которых образует видеопоток, содержащий рассматриваемый телевизионный сигнал [1]. К сожалению, не всегда можно дать гарантию, что все пакеты, отправленные из вещательного центра, достигнут своего получателя.

Одним из решений данной проблемы является опережающая коррекция ошибок (Forward Error Correction, FEC). Речь идет о коррекции потерь IP-пакетов, которая позволяет реконструировать утерянные пакеты без необходимости в их дополнительной передаче. Правда, увеличивающаяся надежность системы не дается бесплатно. Использование FEC требует дополнительных данных, внедряемых в IP поток, что приводит к созданию второго потока. Второй поток используется только для FEC данных и увеличивает ширину, занимаемую каналом.

В данной дипломной работе будет рассмотрена двумерная схема FEC и разработана синтезируемая Verilog HDL – модель ядра передающей части системы IP-телевидения, содержащего одномерную схему FEC. Использование данного модуля в системах IP-телевидения позволит снизить пакетные потери, что в свою очередь приведет к повышению качества декодируемого видеосигнала. Данный критерий является основным в конкурентной борьбе между провайдерами.

1. сТРУКТУРА ЛОКАЛЬНОЙ СЕТИ ДЛЯ ПЕРЕДАЧИ

IP-ТЕЛЕВИДЕНИЯ

1.1 ОСОБЕННОСТИ РЕАЛИЗАЦИИ IP-ТЕЛЕВИДЕНИЯ

В настоящее время IP- телевидение прочно вошло в нашу жизнь. Под IP-телевидением понимается следующее:

* трансляцию телевизионных каналов посредством IP-сети;
* видео по запросу (VoD) и ее разновидность оплаченного видео по запросу (SVoD);
* виртуальный кинотеатр (NVoD);
* виртуальный видеомагнитофон (nPVR);
* оплата за просмотр (PPV);
* просмотр программ со сдвигом во времени (TSTV);
* просмотр скаченных видеофильмов (FTV).

Весь спектр услуг IP- телевидения можно условно разделить на две группы: общего пользования и персональные. К первым относится, прежде всего, просмотр какого-либо канала или только определенной передачи (абонент оплачивает только время, в течение которого транслируется передача). Еще одна массовая услуга – это "виртуальный кинотеатр". Один за другим, по расписанию, провайдер выпускает в эфир популярные фильмы, а пользователь выбирает удобное для себя время и покупает интересующий его сеанс. К массовым услугам можно отнести и предоставление какой-либо справочной информации на экране телевизора – прогноза погоды или курса валют.

Другая группа услуг предоставляется конкретному подписчику. Прежде всего, это "видео по запросу" – выбор фильма или телепередачи из каталога провайдера и их просмотр с любым количеством пауз и повторов в течение определенного времени с момента заказа. Сюда же относятся time-shift (пауза в трансляции и последующий просмотр с места остановки) и "виртуальный видеомагнитофон" (запись передачи, идущей в определенное время).

Для реализации этих услуг оператор использует следующие режимы передачи информации в IP сети: unicast, broadcast, multicast.

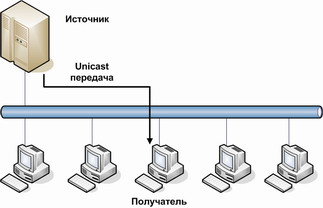
 В IP-сетях видеопотоки доставляются по адресному принципу из вещательного центра до пользовательского оборудования телезрителя, где адрес получателя содержится в заголовках соответствующих IP-пакетов. Такой способ передачи называется одноадресным (unicasting – одноадресная рассылка). Unicast используется для предоставления персональных услуг: передачи видео по требованию и видеоконференцсвязи. Абонент заказывает персональный контент, предназначенный только для него и, соответственно, только сам получает заказанную услугу. При одновременном просмотре своих заказов несколькими пользователями их трафик суммируется на участке от источника – файлового сервера, на котором находятся требуемые передачи, до абонентской линии – например, порта на оборудовании DSLAM (Digital Subscriber Line Access Multiplexer – xDSL мультиплексере доступа), причем это имеет место, даже если все абоненты заказали одну и ту же услугу (рассылку).

Рисунок 1 – Unicast режим передачи

Число абонентов, которые могут получать unicast трафик одновременно, ограничено доступной в магистральной части сети шириной потока (скоростью потока). Для случая Gigabit Ethernet сети теоретическая максимальная ширина потока данных может приближаться к 1 Гбит/с за вычетом полосы, необходимой для передачи служебной информации и технологических запасов оборудования. Предположим, что в магистральной части сети мы можем для примера выделить не более половины полосы для сервисов, которым требуется unicast трафик. Легко подсчитать для случая 5Мбит/с на телевизионный канал MPEG2, что число одновременно получающих unicast трафик абонентов не может превышать 100.

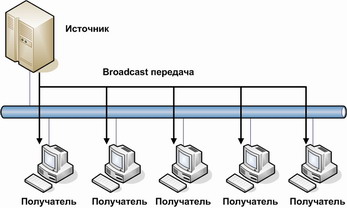
При широковещательном способе передачи информационных потоков (broadcasting – широковещание) одна копия, вещаемой телевизионной программы, передается всем пользователям, т.е. два разных пользователя будут просматривать одни и те же передаваемые ТВ программы.

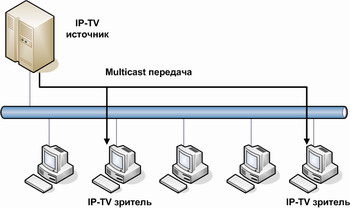
Рисунок 2 – Broadcast режим передачи

Для этого режима используются адреса, заканчивающиеся на 255, например, 192.168.1.255. Если передавать видео в режиме broadcast, то сеть будет нагружена данными вне зависимости от того, принимает их пользователь или нет. Поэтому данный режим применяется только для передачи каких-либо служебных сообщений.

Из широковещательных служб необходимо выделить службу коммутируемого широковещания (multicasting – многоадресная рассылка), когда из всего списка вещаемых ТВ программ группе пользователей доставляется только часть ТВ программ, выбранная самими пользователями[1]. Multicast можно назвать самым важным режимом для IP-телевидения. Для идентификации групп каналов используется специально зарезервированный для этих целей при разработке протокола IP диапазон адресов – от 224.0.0.0 до 239.255.255.255, кроме последнего байта 255 (класс D).

Multicast предусматривает передачу информации от источника к абонентским мультиплексорам или коммутаторам одним потоком, транслируя далее ее только на те порты, которые эту информацию заказывали. Multicast позволяет существенно сэкономить полосу пропускания в транспортной сети, не требуя отдельного потока для каждого канала к каждому зрителю. Конечно, все участвующие в описанном процессе устройства должны поддерживать работу в режиме multicast.

Рисунок 3 – Multicast режим передачи

Загрузка магистральной части сети multicast трафиком зависит только от числа транслируемых в сети каналов. В ситуации с Gigabit Ethernet сетью, предположив, что половину магистрального трафика мы можем выделить под multicast передачу, мы получаем около 100 телевизионных MPEG-2 каналов, каждый имеющий скорость потока данных 5 Мб/сек, независимо от числа абонентов.

1.2 ИЕРАРХИЯ ПРОТОКОЛОВ

Для передачи данных, в частности видеоизображений сжатых согласно существующим стандартам, таким как MPEG-2/MPEG-4 и H.264, используется стандартизированный набор протоколов, также называемый стеком протоколов. Протоколы обеспечивают многоуровневую подготовку сжатого видеоизображения для передачи по сети связи. В данной работе в качестве сети передачи данных используется IP-сеть, построенная на базе стека протоколов TCP/IP.

Многоуровневый характер сетевых процессов проводит к необходимости рассмотрения многоуровневых моделей телекоммуникационных сетей. В качестве эталонной утверждена семиуровневая эталонная сетевая модель взаимодействие открытых систем (OpenSourceInitiative – OSI), в которой все сетевые процессы, реализуемые открытой системой, разбиты на взаимно подчиненные уровни. В данной модели обмен информацией может быть представлен в виде семиуровневого стека, представленного в таблице 1 [2].

Семиуровневая модель OSI Таблица 1

|  |  |  |
| --- | --- | --- |
| № уровня | Наименование  Уровня | Содержание |
| 7 | Уровень приложений | Предоставление услуг на уровне конечного пользователя: почта, теледоступ и пр. |
| 6 | Уровень представления данных | Интерпретация и сжатие данных |
| 5 | Уровень сессии | Аутентификация и проверка полномочий |
| 4 | Транспортный уровень | Обеспечение корректной сквозной пересылки данных |
| 3 | Сетевой уровень | Маршрутизация и ведение учета |
| 2 | Канальный уровень | Передача и прием пакетов, определение аппаратных адресов |
| 1 | Физический уровень | Собственно кабель или физический носитель |

Основная идея модели OSI заключается в том, что каждому уровню (в том числе и транспортной среде) отводится конкретная роль. Благодаря этому общая задача передачи данных подразделяется на отдельные, легко обозримые задачи [2].

Формализованные правила, определяющие последовательность и формат сообщений, которыми обмениваются сетевые компоненты, лежащие на одном уровне, но в разных узлах, называются протоколом.

Модули, реализующие протоколы соседних уровней и находящиеся в одном узле, также взаимодействуют друг с другом в соответствии с четко определенными правилами и с помощью стандартизованных форматов сообщений, которые называются интерфейсом. Таким образом, протоколы определяют правила взаимодействия модулей одного уровня в разных узлах, а интерфейсы определяют правила взаимодействия модулей соседних уровней в одном узле [3].

Большое число уровней, используемых в модели, обеспечивает декомпозицию информационно-вычислительного процесса на простые составляющие. В свою очередь, увеличение числа уровней вызывает необходимость включения дополнительных связей в соответствии с дополнительными протоколами и интерфейсами. Для передачи видео контента по IP-сети важную роль играют первые четыре уровня.

**Уровень 1, физический уровень модели (Physical layer)**

Физический уровень модели определяет характеристики физической сети передачи данных, которая используется для межсетевого обмена. Это такие параметры, как напряжение в сети, сила тока, число контактов на разъемах, электрические, механические, функциональные и процедурные параметры для физической связи в системах. Физическая связь и неразрывная с ней эксплуатационная готовность являются основной функцией 1-го уровня модели OSI [2]. На данном уровне осуществляются как соединения с физическим каналом, так и расторжение, управление каналом, а также им определяется скорость передачи данных, топология сети, механические и электрические характеристики, требуемые для подключения, поддержания соединения и отключения физической цепи. Также здесь определяются правила передачи каждого бита через физический канал связи.

**Уровень 2, канальный (Data Link layer)**

Канальный уровень представляет собой комплекс процедур и методов управления каналом передачи данных, организованный на основе физического соединения.

Основной задачей канального уровня является проверка доступности среды передачи. Другой задачей канального уровня является реализация механизмов обнаружения и исправления ошибок. Для этого на канальном уровне биты группируются в наборы, называемые кадрами.

Канальный уровень обеспечивает корректность передачи каждого кадра, помещая специальную последовательность бит в начало и конец каждого кадра для его выделения, а также вычисляет контрольную сумму, обрабатывая все байты кадра определенным способом и добавляя контрольную сумму к кадру.

Необходимо отметить, что функция исправления ошибок не является обязательной для канального уровня, поэтому в некоторых протоколах этого уровня она отсутствует [3].

К канальному уровню отнесены протоколы, определяющие соединение, протоколы взаимодействия между драйверами устройств и устройствами, с одной стороны, а с другой стороны, между операционной системой и драйверами устройств [2].

Технология канального уровня определяют проводные/беспроводные соединения и электрические сигналы на физическом уровне, формат кадров и протоколы управления доступом к среде [2].

На канальном уровне и физическом уровне широкое применение получили технологии Ethernet, ATM, SDH и др.

**Уровень 3, сетевой (Network layer)**

Сетевой уровень - устанавливает связь между двумя пользователями в вычислительной сети или между двумя сетями. Вычислительные сети являются пакетными сетями, т.е. информация в них передается в виде пакетов, имеющих заголовок, содержащий служебную информацию и поле полезной нагрузки, состоящее из передаваемой по сети информации. Информация необходимая для доставки пакетов от одного пользователя к другому содержится в поле служебной информации, в соответствии с ней пакет передается по вычислительной сети [2].

Основная задача сетевого уровня – маршрутизация данных (передача данных между сетями). Обеспечение доставки пакетов между различными вычислительными сетями выполняют устройства, называемые маршрутизаторами. Маршрутизаторы определяют, для какой сети предназначен пакет, на основе содержащегося в его заголовке адреса, после чего направляют этот пакет в заданную сеть, через соответствующий интерфейс.

К сетевому уровню относятся протоколы, которые отвечают за отправку и получение данных, где определяются отправитель и получатель, и необходимая информация для доставки пакета по сети [2]. К таким протоколам относится протокол IP, на сегодняшний день, являющийся самым распространенным протоколом сетевого уровня модели OSI.

Протокол IP используется для негарантированной доставки данных от одного узла сети к другому. Это означает, что на уровне этого протокола (третий уровень модели OSI) не дается гарантий надежной доставки пакета до адресата. В частности, пакеты могут прийти не в том порядке, в котором были отправлены или не прибыть вовсе. Гарантии безошибочной доставки пакетов дают протоколы более высокого (транспортного) уровня модели OSI. Для доставки пакетов сформированных согласно протоколу IP – IP пакетов между вычислительными сетями используется адресация, регламентированная данным протоколом – IP адресация [2]. IP адресация включает IP-адреса получателя и отправителя пакета. Она строго регламентирует какое устройство или группа устройств в сети является получателями IP- пакетов, поскольку каждое сетевое устройство, функционирующее на сетевом уровне, имеет свой IP- адрес.

Вся информация о пути, по которому должен пройти пакет, определяется по состоянию сети в момент прохождения пакета. Эта процедура называется маршрутизацией в отличие от коммутации, используемой для предварительного установления маршрута следования отправляемых данных.

Принцип маршрутизации является одним из тех факторов, которые определяют гибкость сети Internet и ее преимущества по отношению к другим сетевым технологиям. Маршрутизация представляет собой ресурсоемкую процедуру, так как предполагает анализ каждого пакета, который проходит через шлюз или маршрутизатор, в то время как при коммутации анализируется только управляющая информация, устанавливается канал (физический или виртуальный), и все пакеты пересылаются по этому каналу без анализа маршрутной информации. Однако при неустойчивой работе сети пакеты могут пересылаться по различным маршрутам и затем собираться в единое сообщение. При коммутации путь придется устанавливать заново для каждого пакета, и при этом потребуется больше накладных затрат, чем при маршрутизации [2].

Существует несколько версий протокола IP. В настоящее время используется версия Ipv4. Фактически в заголовке пакета определены все основные данные, необходимые для перечисленных выше функций протокола IP: адрес отправителя, адрес получателя, общая длина пакета и тип пересылаемой датаграммы.

Используя данные заголовка, машина может определить, на какой сетевой интерфейс отправлять пакет. Если IP- адрес получателя принадлежит одной из ее сетей, то на интерфейс этой сети пакет и будет отправлен, в противном случае пакет отправят на другой шлюз.

На основе протокола транспортного уровня IP-модель производит экскапсуляцию информации из IP-пакета и ее передачу на модуль обслуживания соответствующего транспорта [2].

**Уровень 4, Транспортный (Transport layer)**

Транспортный уровень обеспечивает передачу данных между любыми узлами сети с требуемым уровнем надежности. А также реализует механизмы транспортировки, гарантированности доставки и управления информационным потоком (с целью предотвращения переполнения системы данными из другой системы).

Протоколы, функционирующие на транспортном уровне, связывают нижние уровни (физический, канальный и сетевой) с верхними уровнями, которые часто реализуются программными средствами. Этот уровень разделяет средства формирования данных в сети от средств их передачи. Здесь осуществляется разделение информации по определенной длине и уточняется адрес назначения [2]. Транспортный уровень позволяет мультиплексировать передаваемые сообщения или соединения. Мультиплексирование сообщений позволяет передавать сообщения одновременно по нескольким линиям связи, а мультиплексирование соединений – передает в одной посылке несколько сообщений для различных соединений.

Транспортный уровень отвечает за надежность доставки данных: после проверки контрольной суммы принимается решение о сборке сообщения в одно целое. Если сетевой уровень определяет только правила доставки информации, то транспортный – отвечает за целостность доставленных данных [2].

На данном уровне используются протоколы TCP (Transmission Control Protocol –протокол контрольной доставки) и UDP (User Datagram Protocol – протокол пользовательских датаграмм).

Протокол TCP – это транспортный механизм, в соответствии с которым формируется поток данных, состоящий из TCP-пакетов [4]. Процессу формирования потока данных предшествует установление соединения между двумя сетевыми устройствами, между которыми производится передача информации, за счет этого обеспечивается гарантия, что соответствующий адресат получит информацию. После установки соединения начинается передача данных, в процессе которой осуществляется периодический контроль безошибочности передачи информации, в соответствии с механизмами протокола TCP [2]. Данный протокол гарантирует, что приложение получит данные точно в такой же последовательности, в какой они были отправлены, и без потерь.

В отличие от протокола TCP, протокол UDP не гарантирует доставку информации, содержащейся в поле полезной нагрузки UDP-пакета. Согласно протоколу UDP, перед началом передачи информации не происходит установки предварительного соединения между отправителем и получателем информации, а также не производится контроль канала передачи информации в процессе обмена информацией. Эта особенность позволяет протоколу UDP гораздо быстрее и эффективнее доставлять данные для приложений, работающих в реальном времени, таких как IPTV и IP-телефония, однако эта же особенность накладывает высокие требования к надежности работы канала передачи информации, а также возлагает механизмы контроля возможных ошибок на протоколы более высоких уровней [4].

Заголовок UDP-датаграммы (сообщения) имеет вид, показанный на рисунке 4.

32 бита

|  |  |
| --- | --- |
| Source Port | Destination Port |
| Length | Checksum |
| Application Data | |

Рисунок 4 – Структура заголовка UDP-сообщения

Поле Length определяет общую длину сообщения. Поле Checksum служит для контроля целостности данных. Приложение, которое использует протокол UDP, должно поддерживать целостность данных, анализируя поля Checksum и Length. Кроме этого, при обмене данных по UDP прикладная программа сама должна заботиться о контроле получения данных адресатом. Обычно это достигается обменом подтверждениями о доставке между прикладными программами [2].

С ростом популярности мультимедийных приложений реального времени потребовалось стандартизовать уже существующие более и менее одинаковые транспортные протоколы реального времени. Для этой цели был разработан протокол RTP (Real Time Protocol – протокол передачи трафика реального времени).

RTP занимает странное положение в стеке протоколов. Было решено, что RTP будет принадлежать пользовательскому пространству и работать поверх UDP, чтобы использовать его возможности мультиплексирования и контрольного суммирования. Но RTP может использоваться и поверх любой другой сетевой транспортной среды.

Основной функцией RTP является уплотнение нескольких потоков реального масштаба времени в единый поток пакетов UDP. Поскольку RTP использует обычный UDP, его пакеты не обрабатываются маршрутизаторами каким-либо особым образом, если только не включены свойства доставки с качеством обслуживания уровня IP. В частности, нет никаких гарантий касательно доставки, дрожания (неустойчивой синхронизации) и т.д.

Каждый пакет, посылаемый с потоком RTP, имеет номер, на единицу превышающий номер своего предшественника. Такой способ нумерации позволяет получателю определить пропажу пакетов. Однако повторная передача в данном случае не является хорошим решением, поскольку это займет много времени, и повторно переданный пакет окажется никому не нужным. Поэтому протокол RTP не осуществляет управление потоком, контроль ошибок, и в стандарте не предусмотрены никакие подтверждения и механизмы запроса повторной передачи.

Еще одна функция, необходимая приложениям реального времени, - это отметки времени. Идея состоит в том, чтобы позволить источнику связать отметку времени с первым символом каждого пакета. Отметки времени ставятся относительно момента начала передачи потока, поэтому важны только интервалы между отметками. Абсолютные значения, по сути дела никакой роли не играют. Такой механизм позволяет приемнику буферизовать небольшое количество данных и проигрывать каждый отрезок спустя правильное число миллисекунд после начала потока независимо от того, когда на самом деле приходит пакет, содержащий данный отрезок. За счет этого не только снижается эффект джиттера(флуктуаций, неустойчивости синхронизации), но и появляется возможность синхронизации между собой нескольких потоков.

RTP поддерживает одновременную доставку по многим адресам, если мультикастинг поддерживается нижележащим сетевым уровнем.

На рисунке 5 представлен фиксированный RTP-заголовок, который содержит ряд полей, идентифицирующих такие элементы, как формат пакета, порядковый номер, источники, границы и тип полезной нагрузки. За фиксированным заголовком могут следовать другие поля, содержащие дополнительную информацию о данных.

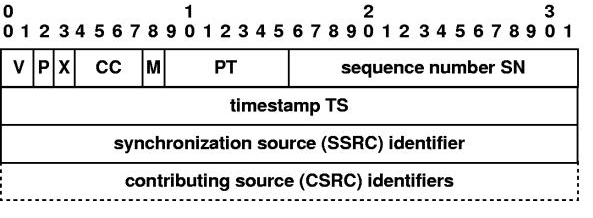


Рисунок 5 – Фиксированный RTP-заголовок

Первые 12 октетов присутствуют во всех RTP-пакетах, в то время как список CSRC-идентификаторов присутствует только, когда пакет формируется смесителем. Поля имеют следующие назначения:

* V (2 бита). Поле версии.

Это поле идентифицирует версию протокола RTP. В настоящее время в это поле записывается код 2. Значение 1 использовалось в опытной версии RTP, а код 0 - в аудио приложении "vat".

* P (1 бит). Поле заполнения.

Это поле сигнализирует о наличии заполняющих октетов в конце полезной нагрузки. Если Р = 1, пакет содержит один или более дополнительных октетов-заполнителей в конце поля данных (заполнители не являются частью поля данных). Последний октет заполнителя содержит число октетов, которые должны игнорироваться. Заполнитель нужен при использовании некоторых алгоритмов шифрования при фиксированном размере блоков или при укладке нескольких RTP-пакетов в один UDP.

* X (1 бит). Поле расширения заголовка.

Если бит Х=1, далее следует фиксированный заголовок, за которым размещается одно расширение заголовка.

* CC(4 бита). CSRC count – поле числа отправителей.

Это поле содержит число идентификаторов отправителей, чьи данные находятся в пакете, причем сами идентификаторы следуют за основным заголовком.

* M (1 бит). Поле маркера.

Интерпретация маркера определяется типом полезной нагрузки. Для видео предполагается разрешить выделять в потоке пакетов существенные события, такие как границы кадра. В случае передачи голоса он задает начало речи после периода молчания.

* PT (7 бит). Поле типа полезной нагрузки.

Это поле идентифицирует тип полезной нагрузки и формат поля данных RTP-пакета, включая сжатие и шифрование. В стационарном состоянии отправитель использует только один тип полезной нагрузки в течение сеанса, но он может его изменить в ответ на изменение условий, если об этом сигнализирует протокол управления передачей в реальном времени (Real-Time Transport Control Protocol).

* Sequence Number (16 бит). Поле порядкового номера.

Каждый источник начинает нумеровать пакеты с произвольного номера, увеличиваемого затем на единицу с каждым посланным пакетом данных RTP. Это позволяет обнаружить потерю пакетов и определить порядок пакетов с одинаковой отметкой о времени. Несколько последовательных пакетов могут иметь одну и ту же отметку о времени, если логически они порождены в один и тот же момент, как, например, пакеты, принадлежащие к одному и тому же видеокадру.

* Timestamp (32 бита). Временная метка.

Временная метка соответствует времени стробирования для первого октета в информационном RTP-пакете. Время стробирования должно быть получено от часов, показания которых увеличиваются монотонно и линейно, чтобы обеспечить синхронизацию и вычисление временного разброса. Разрешающая способность часов должна быть достаточной для обеспечения приемлемой точности синхронизации (одного тика на видео кадр обычно не достаточно). Частота часов зависит от формата данных и задается статически в профайле, в спецификации поля данных, или динамически средствами, выходящими за пределы спецификации протокола RTP. Если RTP-пакеты генерируются периодически, используется временная привязка, определенная задающим генератором стробирования, а не показаниями системных часов.

Начальное значение временной метки является случайным. Несколько последовательных RTP-пакетов могут иметь идентичные временные метки, если логически они генерируются одновременно (например, относятся к и тому же видео кадру).

* Synchronization Source (SSRC) Identifier (32 бита). Поле идентификатора источника синхронизации.

Этот идентификатор выбирается случайным образом, так чтобы в пределах одной RTP-сессии не было двух равных SSRC-кодов. Все приложения должны быть способны выявлять случаи равенства SSRC-кодов. Если отправитель изменяет свой транспортный адрес, он должен также сменить и SSRC-идентификатор.

* Contributing Source (CSRC) identifiers. CSRC-список: от 0 до 15 элементов, по 32 бита каждый

CSRC-список идентифицирует источники информации, которые внесли свой вклад в поле данных пакета. Число идентификаторов задается полем CC. Если число источников больше 15, только 15 из них могут быть идентифицированы.

**Уровень 5, Сеансовый (Session layer)**

На сеансовом уровне осуществляется управление сеансами (сессиями) связи между двумя взаимодействующими прикладными пользовательскими процессами (пользователями). Определяется начало и окончание сеанса связи: нормальное или аварийное; определяется время, длительность и режим сеанса связи, точки синхронизации для промежуточного контроля и восстановления при передаче данных, восстанавливается соединение после ошибок во время сеанса связи без потери данных.

Кроме того, сеансовый уровень содержит дополнительно функции управления паролями, подсчета платы за пользование ресурсами сети, управления диалогом, синхронизации и отмены связи в сеансе передачи после сбоя вследствие ошибок в нижерасположенных уровнях [2].

**Уровень 6, Представления данных (Presentation layer)**

Уровень представления данных управляет представлением данных в необходимой для программы пользователя форме, осуществляет генерацию и интерпретацию взаимодействия процессов, кодирование/декодирование данных, в том числе компрессию и декомпрессию данных (преобразование данных из промежуточного формата сессии в формат данных приложения).

На рабочих станциях могут использоваться различные операционные системы. Каждая из них имеет свою файловую систему, свои форматы хранения и обработки данных. Задачей данного уровня является преобразование данных при передаче информации в формат, который используется в информационной системе. При приеме данных 6-й уровень представления выполняет обратное преобразование. Таким образом, появляется возможность организовать обмен данными между станциями, на которых используются различные операционные системы.

Компрессия, или упаковка, данных сокращает время передачи данных. Кодирование передаваемой информации обеспечивает защиту ее от перехвата.

**Уровень 7, Прикладной уровень (Application layer)**

Прикладной уровень – уровень прикладных программ или приложений, определяет протоколы обмена данными этих прикладных программ – в его ведении находятся прикладные сетевые программы, обслуживающие файлы, а также выполняются вычислительные, информационно-поисковые работы, логические преобразования информации, передача почтовых сообщений и т.п. Одна из задач этого уровня – обеспечить удобный интерфейс пользователя.

Таким образом, из структуры семиуровневой модели OSI видно, что уровень с меньшим номером предоставляет услуги смежному с ним верхнему уровню и пользуется для этого услугами смежного с ним нижнего уровня[2].

Каждый уровень выполняет собственное формирование пакета, добавляя заголовок и концевые блоки к сообщению, поступившему с более высокого уровня. Это приводит к появлению шести наборов заголовков и концевых блоков к тому моменту, когда сообщение готово к передаче по сети (рисунок 6). По мере того как данные передаются с верхнего уровня на нижний, протокол каждого уровня добавляет собственный заголовок, включающий необходимую служебную информацию. Все заголовки и концевые блоки затем передаются физическому уровню, который может добавить свою порцию служебной информации для передачи по физическому каналу связи.

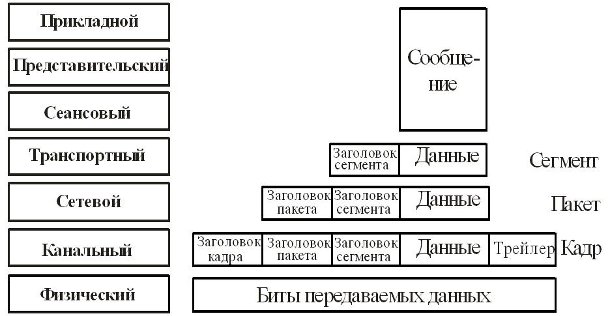


Рисунок 6 – Формирование заголовков в модели OSI

1. ФОРМИРОВАНИЕ МЕДИА ПОТОКА IP-ТЕЛЕВИДЕНИЯ

2.1 СТРУКТУРА ТРАНСПОРТНОГО ПОТОКА MPEG

Системная спецификация MPEG-2 [5] описывает объединение элементарных потоков одной или нескольких телевизионных программ в единый поток данных, удобный для записи или передачи по каналам цифровой связи. Надо отметить, что стандарт MPEG-2 не определяет защиту от ошибок, возможных при записи или передаче, хотя он, конечно, предусматривает такую возможность, облегчая защиту за счет оптимального выбора параметров потока. MPEG-2 регламентирует две возможных формы единого потока данных – это программный поток и транспортный поток.

Транспортный поток состоит из пакетов фиксированной длины – транспортных пакетов, размер которых составляет 188 байт. Программный поток состоит из пакетов более длинных, чем транспортный пакет [6]. В данной работе рассмотрены вопросы связанные только с транспортным потоком, поскольку именно он используется для передачи цифрового телевидения по каналам связи, в частности по IP-сетям.

Транспортный поток имеет иерархическую структуру, схематично представленную на рисунке 7.

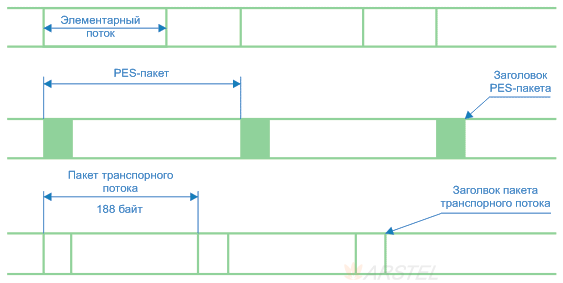


Рисунок 7 – Структура ES, PES и транспортного потоков MPEG-2

Согласно этой иерархической структуре, выделяют три уровня: уровень элементарного потока, уровень элементарного пакетированного потока и уровень транспортного потока [1].

Первый шаг на пути получения единого потока - формирование пакетного элементарного потока PES (Packetised Elementary Stream), представляющего собой последовательность PES-пакетов (рисунок 7). Каждый пакет состоит из заголовка и данных пользователя, или полезной нагрузки, которая представляет собой фрагменты исходного элементарного потока. PES-пакеты могут быть переменной длины.

Транспортный поток может объединять пакетные элементарные потоки, переносящие данные нескольких программ с независимыми временными базами. Он состоит из коротких пакетов фиксированной длины (188 байтов). Элементарные потоки видео, звука и дополнительный данных (например, телетекст) разбиваются на фрагменты, равные по длине полезной нагрузке транспортного пакета (184 байта) и мультиплексируются в единый поток. Этот процесс подчиняется ряду ограничений:

* Первый байт каждого PES-пакета элементарного потока должен быть первым байтом полезной нагрузки транспортного пакета.
* Каждый транспортный пакет может содержать данные лишь одного PES-пакета.

Если PES-пакет не имеет длину, кратную 184 байтам, то один из транспортных пакетов не заполняется данными PES-пакета полностью. В этом случае избыточное пространство заполняется полем адаптации. Транспортные пакеты, переносящие разные элементарные потоки, могут появляться в произвольном порядке, но пакеты, принадлежащие одному элементарному потоку, должны следовать в транспортном потоке в хронологическом порядке, т.е. в порядке их «нарезания» из PES-пакетов. Структура транспортного потока оптимизирована для условий передачи данных в каналах связи с шумами. Это проявляется, прежде всего, в небольшой длине пакетов.

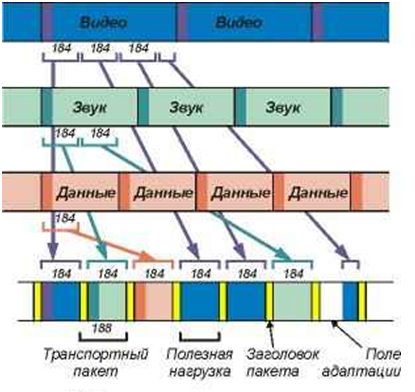


Рисунок 8 – Формирование транспортного пакета

Транспортный пакет начинается с 4-байтного заголовка, первый байт которого – синхронизирующий (число 47 в шестнадцатеричном коде). Это значение не является уникальным и может появляться в других полях транспортного пакета. Однако тот факт, что заголовки всегда следуют с интервалом в 188 байтов, упрощает определение начала пакета. В заголовке содержится служебная информация, а именно о принадлежности данного транспортного пакета определенному PES-потоку, указатель начала PES-пакета в данном транспортном пакете, информация о штампах времени PCR (Program Clock Reference – ссылка на программные часы) [1].

2.2 СТРУКТУРА МЕДИА ПАКЕТОВ

При формировании информационного потока соблюдается иерархия последовательной инкапсуляции пакетов всех уровней от транспортных пакетов до кадров канального уровня модели OSI [1].

На рисунке 9 показана последовательная инкапсуляция транспортных пакетов в пакеты нижележащих уровней. Протокол каждого последующего уровня добавляет собственный заголовок, включающий необходимую служебную информацию. Все заголовки и концевые блоки затем передаются физическому уровню, который добавляет свою порцию служебной информации для передачи по Ethernet сети.



Рисунок 9 – Кадр в формате MPEG с использованием протоколов RTP/UDP/IP для передачи по сети Ethernet

Мультимедийное приложение может состоять из нескольких аудио-, видео-, текстовых и некоторых других потоков. Они прописываются в библиотеке RTP, которая, как и само приложение, находится в пользовательском пространстве. В нашем случае в качестве полезной нагрузки для RTP выступают транспортные пакеты MPEG. Так как 10/100 Мбит/с Ethernet ограничивает размер кадра 1500 байт. C вычетом заголовков размер полезной нагрузки RTP составляет 1500 – 14 – 20 – 8 – 12 = 1446 байт. Этого размера достаточно для инкапсуляции 7 транспортных пакетов по 188 байт каждый. В нашем случае используется как раз 7 транспортных пакетов для оптимального использования полезной нагрузки RTP. После помещения транспортных пакетов в пакет RTP, RTP пакеты отправляются в сокет. Термин "сокет" (socket) обозначает одновременно библиотеку сетевых интерфейсов и оконечное устройство канала связи (точку связи), через которое процесс может передавать или получать данные. На другом конце сокета генерируются UDP-пакеты, которые внедряются в IP-пакеты. Теперь остается передать IP-пакеты по сети. Если компьютер подключен к локальной сети Ethernet, IP-пакеты для передачи помещаются в поле полезной нагрузки Ethernet-кадра.

1. . СИСТЕМА КОРРЕКЦИИ ОШИБОК

Как и в любой среде передачи, в компьютерных сетях существуют помехи и шумы, ведущие к потерям и искажениям данных. Поскольку современные компьютерные сети являются сетями пакетной передачи данных, искажения отдельных бит в пакете могут привести к потере пакета целиком ввиду расхождения контрольной суммы бит. Потери пакетов также могут происходить из-за перегрузок коммутаторов, «проседания» пропускной способности каналов из-за слишком большого числа одновременно подключенных пользователей, переполнения входных буферов сетевых устройств и т.д.

Большинство сетевых приложений крайне чувствительны к потерям данных. Для некоторых сетевых приложений проблема потерь пакетов решается путем повторной передачи источником потерянных пакетов по запросу получателя – так называемая схема ARQ (Automatic Repeat reQuest). Однако такая схема неприемлема для передачи видео в режиме реального времени, поскольку запрос и повторная отправка пакета являются длительными операциями. Кроме того данная схема реализуется транспортным протоколом TCP. В нашем случае на транспортном уровне используются совместно протоколы UDP и RTP, поэтому для решения проблемы потерь пакетов Pro-MPEG CoP #3 предлагает использовать схему упреждающей коррекции ошибок Forward Error Correction (FEC).

3.1 НАЗНАЧЕНИЕ СХЕМЫ FEC

Схемы коррекции, такие как FEC, позволяют реконструировать утерянные пакеты без необходимости в их дополнительной передаче, но весьма накладны, ввиду использования дополнительной мощности в полосе пропускания.

Для передачи видео в режиме реального времени с использованием протокола UDP по IP-сетям потеря любого пакета будет приводить к ухудшению качества декодированного изображения. При просмотре полученного изображения можно будет наблюдать следующие искажения: разбиение части изображения на блоки, заморозки кадров, пропуск кадров и другие неприятные вещи.

Схема FEC впервые упоминается в RFC 2733. Этот документ определяет формат полезной нагрузки для RTP пакета, который позволяет использовать схему FEC при передаче видео в реальном времени. Основные преимущества этой схемы заключаются в том, что она может использоваться с любыми стандартами видеоформатов медиа пакетов (MPEG, SDI, SDTI…),инкапсулированными в RTP-пакет; возможно использование традиционных кодов коррекции ошибок, таких как код четности, коды Хэмминга, Рида – Соломона. Для генерирования поля полезной нагрузки FEC пакета необходимо использовать не более 24 последовательных медиа пакетов.

Для описания восстановления пакетных потерь Pro-MPEG CoP #3 предлагает расширить существующий RFC 2733, что позволит применять помимо последовательных еще и непоследовательные медиа пакеты, расположенные на интервалах, больших чем 24 пакета, для генерирования поля полезной нагрузки FEC пакета.

Целью дипломной работы является разработка ядра передающей части системы коррекции ошибок, поэтому помимо медиа пакетов, структура которых описана выше, будут передаваться дополнительные FEC-пакеты для восстановления утерянных медиа пакетов. Так как схема FEC примененяется на транспортном уровне, то это внесет ряд ограничений на формирование RTP-заголовка для медиа пакетов.

3.2 ФОРМИРОВАНИЕ RTP ЗАГОЛОВКА МЕДИА ПАКЕТОВ

RFC3550 и RFC2250 накладывают следующие дополнительные ограничения на RTP заголовок медиа пакетов:

• Бит P (дополнительный) должен быть обнулен. Это означает, что в полезной нагрузке не будет никаких дополнительных байтов.

• Бит X (расширение) должен быть обнулен. Это означает, что не будет расширения заголовка.

• Бит M (Маркер) должен всегда обнуляться. Это означает, что во время сеанса передачи не будет никаких разрывов в потоке.

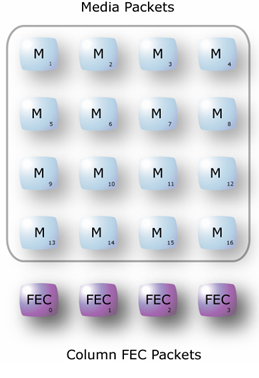
• Поле CC (число CSRC) должно быть обнулено. Это подразумевает, что отсутствуют записи в поле CSRC .

• Значение поля SSRC не будет использоваться получателем, таким образом, отправитель свободен в выборе значений для заполнения данного поля.

• Не существует никаких требований для присвоения начального порядкового номера пакета. Этот номер будет присвоен в произвольном порядке, как предложено в RFC3550.

3.3ФОРМИРОВАНИЕ СХЕМЫ FEC

Различают два типа FEC коррекции: FEC коррекция по столбцам и FEC коррекция по строкам.

Коррекция ошибок по столбцам обеспечивает исправление последовательной потери пакетов в пределах данного столбца. Любой потерянный пакет, принадлежащий к конкретному столбцу, сможет быть восстановлен за счет соответствующего FEC-пакета. Коррекция ошибок по столбцу идеальна для исправления ошибок разрыва соединения и случайных ошибок.

Коррекция ошибок по строкам обеспечивает исправление непоследовательной потери пакетов в пределах строки. Она обычно используется для исправления случайных ошибок.

Рисунок 10 – Коррекция ошибок по столбцам

Коррекция ошибок по столбцам часто называют одномерной коррекций ошибок. Коррекция же по столбцам и строкам – это двумерная коррекция, так как исправление ошибок происходит сразу в двух размерностях. Она обеспечивает более устойчивую схему для защиты от ошибок, способную справиться со случайными ошибками, а также исправить больше ошибок, чем схемы коррекции по столбцам и по строкам в отдельности.

Процесс формирования полезной нагрузки FEC-пакетов основан на использовании матрицы, размер которой определяется двумя параметрами L и D, L – интервал между непоследовательными пакетами, используемыми для формирования FEC-пакета, D – глубина матрицы. Этот размер выбирается в зависимости от числа искаженных пакетов, которые подлежат восстановлению, а также исходя из требований к линии и заданного качества передачи. FEC-пакеты вычисляются при помощи логической функции XOR (исключающее или) из медиа пакетов, располагающихся вдоль столбца или вдоль строки матрицы.

Как только FEC-пакеты будут сформированы, они сразу же передаются вместе с медиа пакетами получателю, правда отдельно на разные UDP порты.

Как определено в Pro-MPEG CoP #3, медиа пакеты передаются на порт n, в то время как FEC-пакеты по столбцам передаются на порт n+2 , а FEC-пакеты по строкам на порт n+4. Это позволяет обеспечивать трансляцию одновременно для всех получателей: поддерживающих и не поддерживающих систему коррекции ошибок. Получатели, поддерживающие эту систему, могут использовать дополнительные FEC-потоки для восстановления пропавших и поврежденных пакетов. Получатели, не поддерживающие FEC, просто проигнорируют эти потоки.

Рисунок 11– Передача RTP потоков

На рисунке 12 изображен пример медиа и FEC-пакетов, дошедших до получателя. Получатель должен по максимуму использовать доступные FEC-пакеты для восстановления недостающих медиа пакетов исходного потока видео.



Рисунок 12 – Пакеты дошедшие до получателя

Для того чтобы обеспечить функциональную совместимость и упростить реализацию схемы FEC, необходимо задать определенные значения для параметров L и D. Производители должны поддерживать все комбинации этих значений в определенных пределах, но при этом могут выходить за рамки предложенного диапазона при желании.

L\*D <= 100

1 <= L <= 20

4 <= D <= 20

Эти ограничения применяются для обоих FEC-потоков. Нужно, правда, отметить, что использование второго потока желательно только в тех случаях, когда L>=4.

Диапазон значений, предложенный выше, характерен для транспортного потока MPEG-2. Остальные видеоформаты также могут использовать FEC-схему, но уже с другими рекомендованными значениями.

В рамках данной дипломной работы мною будет реализована на языке Verilog HDL одномерная схема опережающей коррекции ошибок, когда FEC-пакет будет формироваться из четырех последовательных медиа пакетов.

3.4 СТРУКТУРА FEC ПАКЕТА

Пакет FEC построен путем размещения FEC заголовка и полезной нагрузки FEC в полезной нагрузки RTP, как показано на рисунке 13.

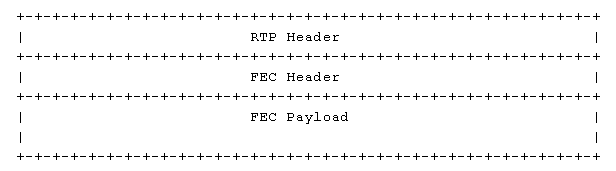


Рисунок 13 – Структура FEC пакета

3.5 ФОРМИРОВАНИЕ ЗАГОЛОВКОВ FEC ПАКЕТОВ

3.5.1 ФОРМИРОВАНИЕ RTP ЗАГОЛОВКА

RFC2733 накладывает следующие ограничения на значения полей в RTP заголовке FEC пакета. Он указывает, что значения полей Р, Х, М, и CC вычисляются из медиа пакетов, но из-за ограничений в разделе 3.2 значения этих полей будут равны нулю.

Кроме этого будут приняты следующие дополнительные ограничения:

• Значение поля SSRC не будет использоваться приемником, следовательно, будет равно нулю.

• Поле time stamp не будет использоваться приемником, следовательно, тоже будет равно нулю.

3.5.2 ФОРМИРОВАНИЕ FEC ЗАГОЛОВКА

Изначально в RFC 2733 были описаны первые 12 байт заголовка FEC. Для того чтобы учесть расширение схемы коррекции ошибок, предложенное Pro-MPEG CoP #3, FEC заголовок должен быть изменен, как показано на рисунке 3 [7].

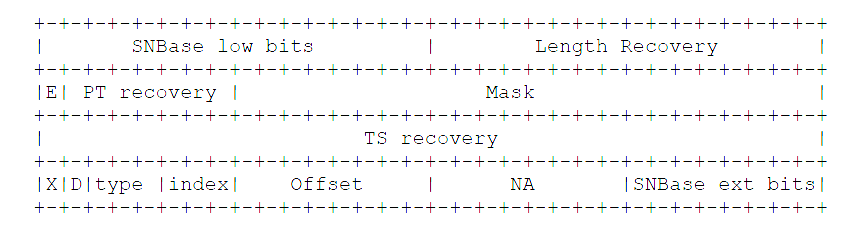


Рисунок 14 – Заголовок FEC протокола

В RFC2733 определяются значения следующих полей :

• SNBase low bits (16 бит): минимальное количество последовательных пакетов, связанных с FEC пакетом. Для транспортных потоков MPEG2 поле порядкового номера из 16 бит достаточно, так что этот параметр должен содержать полный порядковый номер. Для протоколов с более длинными порядковыми номерами в этой области будут содержаться младшие 16 бит порядкового номера.

• Length Recovery (16 бит) Длина Восстановления: это поле используется для определения длины любых медиа пакетов, связанных с пакетом FEC.

• PT Recovery (7 бит) Восстановление полезной нагрузки: это поле используется для определения типа полезной нагрузки любых медиа пакетов, связанных с пакетом FEC.

• TS Recovery (16 бит) Восстановление временной метки: это поле используется для восстановления временной метки любых медиа пакетов, связанных с пакетом FEC.

Ниже приведены поля, значения которых отличаются от значений приведенных в RFC2733, или которые являются новыми:

• E (1 бит) : В RFC2733 это поле должно быть установлено в '0'. Согласно Pro-MPEG CoP #3 это поле устанавливается в '1', чтобы указать, что заголовок продлен.

• Mask (24 бит) Маска: В RFC2733 маска используется, чтобы выбрать, какие пакеты используются для формирования FEC пакета. Определение маски подразумевает сложную связь между пакетами данных и FEC пакетом, что в свою очередь усложняет реализацию. Для простоты Pro-MPEG CoP #3 предлагает установить поле Mask в ноль, а его роль будет выполнять поле NA.

• X (1 бит): Этот бит зарезервирован для будущих расширений заголовка и должен быть установлен в ноль, чтобы соответствовать версии заголовка FEC, описанной в Pro-MPEG CoP #3.

• D (1 бит): Этот бит показывает к какому именно типу FEC пекетов (по строкам или по столбцам) принадлежит данный пакет. Этот бит должен быть установлен в 0, если коррекция ошибок осуществляется по столбцам и в 1, если коррекция осуществляется по строкам.

• Type (3 бита) Тип: Это поле показывает, какой код исправления ошибок используется. Это может быть XOR (тип = 0), Хэмминга (тип = 1), Рида-Соломона (тип = 2). Также могут быть использованы и другие методы кодирования. Если придерживаться Pro-MPEG CoP #3, то оборудование должно использовать при кодировании только логическую функцию XOR.

• Index (3 бита) Индекс: Это поле используется для более сложных кодов защиты от ошибок. Для метода с XOR коррекцией, один FEC пакет соответствует группе медиа пакетов, следовательно, поле индекса всегда будет содержать 0.

• Offset (8 бит) Смещение: Это поле соответствует периоду для выбора медиапакетов, связанных с этим пакетом FEC. Оно в точности соответствует параметру L для FEC-пакетов, вычисляемых по столбцам (первый поток FEC). Для FEC-пакетов, вычисляемых по строкам (второй поток FEC) этот параметр всегда должна быть константа. Это поле должно сохранять свое значение в течение сессии для каждого потока FEC.

• NA (8 бит): Это поле указывает количество медиа пакетов, связанных с FEC-пакетом, и в точности соответствует параметру D для FEC пакетов, вычисляемых по столбцам, или параметру L для FEC пакетов, вычисляемых по строкам. Это поле должно сохранять свое значение в течение сессии для каждого потока FEC.

• SNBase ext bits (8бит): Это поле зарезервировано для использования с протоколами, которые требуют расширенных порядковых номеров больше, чем 16 бит. Для транспортных потоков MPEG2 поле порядкового номера из 16 бит достаточно, так что этот параметр должен быть установлен в 0. Для протоколов с более длинными порядковыми номерами это поле будет содержать следующие 8бит порядкового номера, после тех, что уже записаны в поле SNBase low bit.

1. Структурная схема передающей части системы IP-телевидения

На рисунке 15 представлена структурная схема передающей части системы IP-телевидения.

 Рисунок 15 – Структурная схема передающей части системы IP-телевидения

В общем случае этапы формирования сигнала IP-телевидения включают оцифровку сигнала, его компрессию, формирование транспортного потока и инкапсуляцию.

Первый этап – оцифровка, требуется в том случае, если исходные видео- и аудиосигналы существуют в аналоговой форме. Например, если речь идет о старых фильмах, хранимых на пленочных носителях, или если в IP-сеть требуется передать сигнал от местного источника. Оцифровка аналоговых сигналов выполняется кодерами сжатия, оборудованными соответствующими входными интерфейсами. Что касается крупных студий, то там телепрограммы уже давно формируются в цифровой форме. Для оцифровки видео используются 10-битные АЦП(Аналогово-цифровой преобразователь), частота дискретизации сигнала яркости составляет 13,5 МГц, а цветоразностных – вдвое меньше, то есть 6,75 МГц. Легко посчитать, что при использовании 10-битного АЦП результирующая скорость некомпрессированного видеопотока составляет 270 Мбит/с.

10\*13.5+10\*6.75+10\*6.75 = 270 Мбит/с.

Для передачи по транспортным каналам цифровой видеосигнал необходимо компрессировать. Эта операция осуществляется в ядре видеокодирования MPEG-2, H.264,VC1.

Затем происходит формирование транспортного потока и дальнейшая его инкапсуляция для передачи по сети Ethernet.

В нашем случае все этапы, кроме оцифровки, реализуются на базе

ПЛИС (программируемая логическая интегральная схема). В качестве ПЛИС используется схема фирмы Xilinx.

Сформированный сигнал IP-телевидения требуется передать в сеть, для этого используется MAC(Media Access Controller), организующий доступ к среде. MAC – микросхема канального (второго) уровня модели OSI. MAC обеспечивает адресацию и механизмы управления доступом к каналам, что позволяет нескольким терминалам или точкам доступа общаться между собой в многоточечной сети (например, в локальной вычислительной сети).

Устройство физического уровня (Ethernet PHY) обеспечивает кодирование данных, поступающих от MAC-подуровня для передачи их по транспортной среде, синхронизацию передаваемых данных, а также прием и декодирование данных в узле-приемнике.

Разъем RJ-45 осуществляет подключение к сети Ethernet.

Разрабатываемая мною синтезируемая Verilog HDL –модель модуля одномерной коррекции ошибок будет включена в схему передающей части системы IP-телевидения, представленную на рисунке 15, на место формирователя пакетов IPTV.

1. функциональная схема модуля коррекции ошибок передающей части системы IP-ТЕЛЕВИДЕНИЯ

Основной функцией модуля коррекции ошибок является формирование двух потоков: основного – содержащего медиа пакеты и дополнительного – содержащего FEC-пакеты для восстановления утерянных медиа пакетов.



Рисунок 16 – Функциональная схема модуля коррекции ошибок передающей части системы IP-телевидения

Данный модуль включает в себя три подмодуля: два первых подмодуля формируют соответствующие пакеты, адаптер объединяет эти пакеты в один поток, передаваемый по сети Ethernet.

На первые два подмодуля от формирователя транспортных потоков вприходят три сигнала:

* Ts\_parallel – транспортный поток, состоящий из пакетов по 188 байт;
* Ts\_parallel\_strobe – строб, показывающий начало каждого байта;
* Strobe\_47 – строб, выделяющий первый байт каждого пакета.

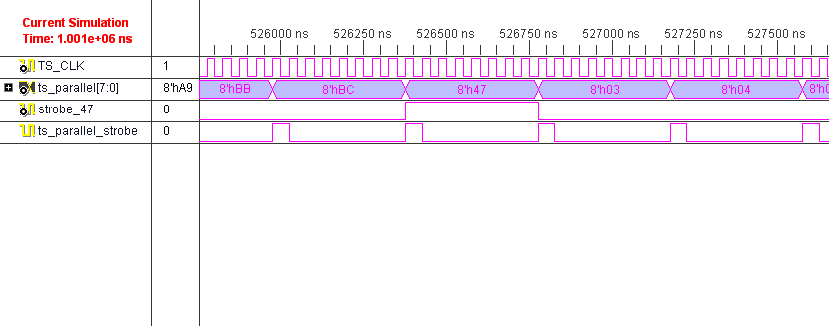


Рисунок 17 – Диаграммы входных сигналов

В каждом из этих подмодулей транспортный поток делится на пачки по 7 транспортных пакетов. При приходе сигналов ack\_fec и ack\_rtp к пачкам по 7 пакетов добавляются соответствующие сетевые заголовки и готовые Ethernet пакеты выводятся в сеть.

На третий подмодуль – адаптер – с первых двух поступают следующие сигналы:

* tx\_data\_rtp, tx\_data\_fec – потоки медиа и FEC-пакетов;
* tx\_data\_rtp\_valid, tx\_data\_fec\_valid – сигналы, показывающие в какой момент времени возможна отправка медиа и FEC-пакетов на MAC контроллер;
* tx\_data\_rtp\_start, tx\_data\_fec\_start – сигналы, выделяющие первые байты пакетов;
* tx\_data\_rtp\_stop, tx\_data\_fec\_stop – сигналы, выделяющие последние байты пакетов;

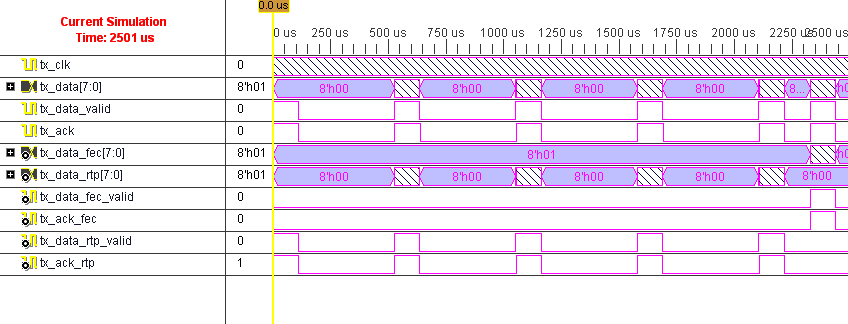


Рисунок 18 – Диаграммы выходных сигналов

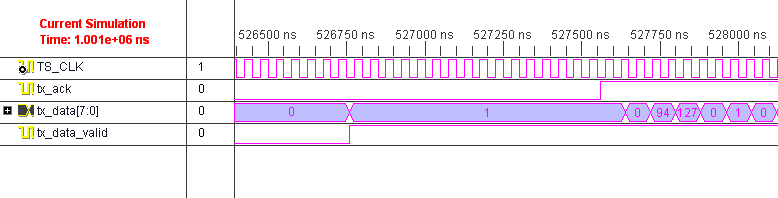
Третий подмодуль – адаптер – объединяет потоки tx\_data\_rtp и tx\_data\_fec в один общий поток tx\_data, сигналы tx\_data\_fec\_valid и tx\_data\_rtp\_valid в сигнал tx\_data\_valid, сигналы tx\_data\_rtp\_stop и tx\_data\_fec\_stop в tx\_data\_stop, сигналы tx\_data\_rtp\_start и tx\_data\_fec\_start в tx\_data\_start. Сформированные в адаптере сигналы мы отправляем на MAC контроллер. 

Рисунок 19

С MAC контроллера на адаптер приходит сигнал ack, подтверждающий готовность MAC контроллера осуществить передачу пакетов в сеть. В адаптере из ack формируются сигналы ack\_fec и ack\_rtp, отправляемые на соответствующие модули.

5.1 ФОРМИРОВАТЕЛЬ МЕДИА ПАКЕТОВ

Если рассмотреть первый подмодуль подробнее, то он включает в себя две памяти: ROM – для хранения заголовков протоколов (в нашем случае ее длина будет 54 байт), RAM – для записи в нее 14 транспортных пакетов MPEG.



Рисунок 20 – Схема формирователя медиа пакетов

Модуль RAM имеет два независимых порта, которые позволяют доступ к единому пространству памяти, созданному на основе определяемых пользователем значений ширины и глубины. Глубина памяти составляет 12 байт, ширина – 4098 байт, этого достаточно, чтобы одновременно в памяти могло храниться два последовательных медиа пакета, исключая заголовки сетевых протоколов. Память разделена на два банка по 2049 байт каждый, переключение с одного банка на другой происходит по смене значения сигнала ts\_wr\_bank.

Полезная нагрузка медиа пакета состоит из 7 транспортных пакетов по 188 байт, что равняется 1316 байт.

Порт A работает только на запись на частоте 27 МГц (T = 37нс), порт B только на чтение на частоте 12,5 МГц (T = 80 нс). Как только в память будут записаны первые 7 транспортных пакетов, банк переключится, и запись следующих 7 пакетов будет осуществляться в другой раздел памяти. Одновременно с переключением банка возникнет строб frame\_written, разрешающий чтение из памяти первых 7 транспортных пакетов. После записи следующих 7 транспортных пакетов в память, банк снова переключается и все действия повторяются заново.

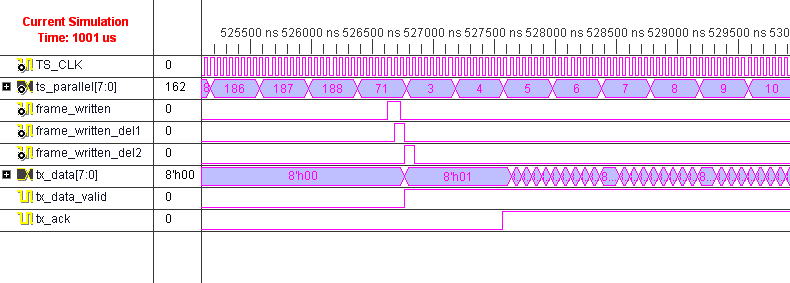


Рисунок 21 – Диграммы вывода пакета на MAC контроллер

Скорость записи медленнее, чем скорость чтения, поэтому пакет будет считан из памяти до момента следующего переключения банка.

К 7 пакетам, считанным из памяти, при помощи мультиплексора будут добавлены заголовки сетевых протоколов, хранящихся в ROM.

Адрес записи 7 транспортных пакетов в RAM начинается не с нуля, а с 54, так как значения адреса соответствуют значениям счетчика для формирования медиа пакета. В медиа пакете байты с 0 по 53 будут заполнены значениями заголовков, хранящихся в ROM, начиная с 54 будут записываться значения полезной нагрузки, считываемые из RAM.

На следующем такте после строба frame\_written возникает сигнал готовности пакета tx\_data\_rtp\_valid, после прихода сигнала ack\_rtp, разрешающего вывод сигнала в сеть, запускается процесс инкапсуляции.

5.2 ФОРМИРОВАТЕЛЬ МЕДИА ПАКЕТОВ



Рисунок 22 – Схема формирователя FEC-пакетов

Второй подмодуль так же, как и первый содержит две памяти: RAM и ROM. ROM, как и в первом модуле, выполняет функцию хранения заголовков. Для FEC-пакета под заголовки отводится 70 байт.

RAM используется для формирования содержимого FEC-пакета. В качестве RAM выбрана двухпортовая память, первый порт которой работает на чтение и запись, а второй – только на чтение. Глубина памяти составляет 12 байт, ширина – 4098 байт. Память, также как и в первом подмодуле, разделена на два банка по 2049 байт каждый, переключение с одного банка на другой происходит по смене значения сигнала ts\_wr\_bank.

Полезная нагрузка FEC-пакета состоит из 7 транспортных пакетов по 188 байт. FEC-пакеты вычисляются при помощи логической функции XOR, используя полезную нагрузку четырех последовательных пакетов. Механизм формирования этого пакета следующий.

На первом этапе в память записываются 7 транспортных пакетов 1-го FEC-пакета (сигнал ts\_parallel\_r1). Запись осуществляется побайтно по адресам с 69 по 1315 по сигналу ts\_parallel\_strobe\_del1. После записи 1-го FEC-пакета формируется строб ts\_valid\_strobe, равный ”1” во время записи транспортных пакетов со 2-го по 4-й FEC-пакеты. Во время отсутствия строба ts\_valid\_strobe суммирование пакетов не производится.

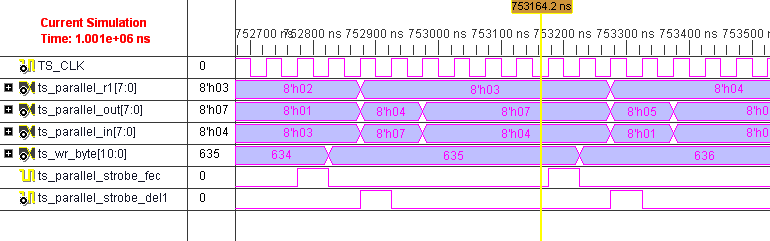


Рисунок 23 – Диаграммы суммирования пакетов

На втором этапе при записи транспортных пакетов со 2-го по 4-й FEC-пакетов на первом такте происходит считывание из памяти уже записанного по этому адресу значения и суммирование по модулю 2 с только что пришедшим значением. На втором такте происходит запись просуммированного значения в эту же ячейку памяти.

Таким образом, к приходу 5 пакета в памяти записана сумма по модулю 2 полезных нагрузок четырех FEC-пакетов. Как только в память будет записана эта сумма, банк переключится, и запись следующих 4 FEC-пакетов будет производиться в другой раздел памяти.

FEC-пакет будет готов к выводу в сеть в тот же момент времени, что и четвертый медиа пакет. Однако разрывать вывод пакета запрещено, поэтому следует задержать вывод FEC-пакета на некоторое время. Время записи медиа пакета в память составляет 37нс\*1316\*8=389536 нс, а время чтения из памяти 80нс\*1370=109600 нс. Как видно из приведенных значений, чтение происходит в 3,5 раза быстрее. Для того чтобы вывести FEC-пакет между 4-м и 5-м медиа пакетами, стоит задержать его вывод относительно конца 4-го медиа пакета на 109600 нс.

1. Технико – экономическое обоснование дипломной работы

4.1 КОНЦЕПЦИЯ ЭКОНОМИЧЕСКОГО ОБОСНОВАНИЯ РАБОТЫ

Целью настоящей дипломной работы является разработка ядра передающей части системы коррекции ошибок в IP-телевидении для ПЛИС Xilinx (ПЛИС – программируемая логическая интегральная схема).

Все большее распространение цифрового телевидения в России оборачивается ужесточением конкуренции операторов IPTV. Основным «орудием» конкурентной борьбы провайдеры выбирают качество услуг.

Как и в любой среде передачи, в компьютерных сетях существуют помехи и шумы, ведущие к потерям и искажениям данных. Поскольку современные компьютерные сети являются сетями пакетной передачи данных, то будут происходить потери пакетов целиком. Для передачи видео в режиме реального времени по IP-сетям потеря любого пакета будет приводить к ухудшению качества декодированного изображения.

Необходимость применения разрабатываемого модуля, заключается в максимальном снижении пакетных потерь, что в свою очередь должно привести к повышению качества изображения при просмотре IP-телевидения.

4.2 КРАТКОЕ ТЕХНИЧЕСКОЕ ОПИСАНИЕ СИСТЕМЫ

В программном модуле осуществляется формирование двух транспортных потоков: основного – содержащего медиа пакеты, и дополнительного – содержащего FEC-пакеты с контрольной суммой, по которому на приемной части системы коррекции ошибок можно будет восстанавливать медиа пакеты, потерянные в сети.

4.3 ОРГАНИЗАЦИООНЫЙ ПЛАН ВЫПОЛНЕНИЯ РАБОТЫ

Трудоемкость выполнения научно-исследовательской работы(НИР) определяется фактическими затратами труда, выраженными в человеко-днях, по перечню видов работ, при составлении которого учитывалось смысловое содержание каждого вида работ и взаимосвязь между ними, в частности логическая упорядоченность последовательности их выполнения. Перечень работ, их распределение по видам исполнителей приведен в таблице 2.

На основе трудоемкости выполнения работы по разработке системы рас­считываются издержки на оплату труда ее исполнителей, являющиеся одной из основных статей калькуляции себестоимости разработки.

Таблица 2

|  |  |  |
| --- | --- | --- |
| Наименование работ | Трудоемкость, чел./дни | |
|  | Руководитель | Дипломант |
| Разработка и уточнение технического задания | 1 | 3 |
| Изучение и обзор научно-технической литературы | - | 15 |
| Разработка структуры решения задачи | 1 | 15 |
| Разработка структурной схемы передающей части системы | - | 1 |
| Разработка функциональной схемы передающей части системы | - | 1 |
| Программная реализация | - | 20 |
| Моделирование и варификация | 2 | 20 |
| Составление технической документации | 2 | 10 |
| Сдача работы | 2 | 4 |
| Итого | 8 | 89 |

Смета затрат на выполнение проекта. Себестоимость разработки будет определяться по фактическим затратам. Расчет себестоимости осуществляется по следующим статьям:

1. Материалы с учетом транспортно-заготовительных расходов;

2. Издержки на амортизацию ПК и оргтехники;

3. Основная и дополнительная заработная плата;

4. Отчисления на социальные нужды;

5. Накладные расходы;

В основе определения заработной платы непосредственных исполнителей является определение трудоемкости работ (таблица 6.1), связанных с разработкой дипломной работы.

Основная и дополнительная заработная плата исполнителей разработки, т.е. руководителя и дипломанта, рассчитываются на основании следующих данных:

• Трудоемкость выполнения работ (Таблица 6.1) Трук = 8 чел/дней,

Тдип = 89 чел/дней.

• Дневная ставка научного руководителя Друк = 850 руб. 00коп.

• Дневная ставка инженера Ддип = 400 руб. 00коп.

• Процент дополнительной заработной платы 12%.

• Процент отчислений на социальные нужды 34,2%.

• Процент накладных расходов 33%.

Основная заработная плата исполнителей рассчитывается по формуле:

Сзо = Трук \* Друк + Тдип \* Ддип

Сзо = 8 \* 850 + 89 \* 400 = 6800 + 35600 = 42400 руб. 00 коп.

Дополнительная заработная плата составляет 12% от основной заработной платы:

Сзд = Сзо \* 0,12

Сзд = 5088 руб. 00 коп.

К статье «Отчисления на социальные нужды» относятся:

* Отчисления на социальное страхование;
* Отчисления на пенсионное обеспечение;
* Отчисления на медицинское страхование и т.д.;

Отчисления на социальные нужды составляют 34,2% от суммы основной и дополнительной заработной платы:

Ссн = ( Сзо + Сзд ) \* 0,342 = 16240 руб. 90 коп.

Накладные расходы составляют 33% от суммы основной и дополнительной заработной платы:

Снр = (Сзо + Сзд ) \* 0,33

Снр = 15671 руб. 04 коп.

Расчет количества и стоимости материалов с учетом транспортно-заготовительных расходов: на статью «Материалы» относятся расходы на основные и вспомогательные материалы и комплектующие изделия, которые могут понадобиться при выполнении разработки.

4.4 КАЛЬКУЛЯЦИЯ РАСХОДОВ

4.4.1 РАСХОДЫ ПО СТАТЬЕ «МАТЕРИАЛЫ»

Калькуляция расходов по статье «Материалы» приведена в таблице 3

Таблица 3

|  |  |  |  |
| --- | --- | --- | --- |
| Материалы | Кол-во | Цена, руб. | Сумма, руб. |
| Бумага для оргтехники, пачка | 2 | 150 | 300 |
| Картридж для принтера | 1 | 3280 | 3280 |
| Компакт диск (CD-RW), | 1 | 45 | 45 |
| USB-флеш-накопитель | 1 | 350 | 350 |
| Канцелярские товары |  |  | 350 |
| ИТОГО: | | | 4325 |
| Транспортные расходы (15%) | | | 648,75 |
| ВСЕГО: | | | 4973,75 |

4.4.2 ИЗДЕРЖКИ НА АМОРТИЗАЦИЮ ПК И ОРГТЕХНИКЕ

Амортизация оборудования (в данном случаеэто ПК и принтер) определяется по государственным нормам линейным методом. За год сумма амортизации составит:

Аг = Коб \* НАМ/100

где Коб - балансовая стоимость оборудования,

Нам - норма амортизации, равная 20%.

За рабочий день сумма амортизации составит:

Ад = Аг / N

где N - количество рабочих дней в 2011 году, равное 248 дням.

Балансовая стоимость ПК на базе AMD Turion 64 X2 - 23000 руб.,

принтера Xerox Phaser 3117- 2400 руб.

Амортизация оборудования за год – Аг = 5080 руб.

Амортизация оборудования за день – Ад = 20 руб. 50 коп.

Амортизация оборудования за время разработки дипломного работы

САМ = Ад \* 100 = 2050 руб.

На основе полученных данных по отдельным статьям затрат составляется калькуляция себестоимости разработки в целом по форме, представленной в таблице 4.

4.4.3 СЕБЕСТОИМОСТЬ ВЫПОЛНЕНИЯ ДИПЛОМНОЙ РАБОТЫ

Себестоимость выполнения дипломной работы в целом представлена в

таблице 4.

Таблица 4

|  |  |
| --- | --- |
| Статья затрат | Сумма, руб. |
| Материалы | 4973,75 |
| Издержки на амортизацию ПК и оргтехники. | 2050 |
| Основная заработная плата | 42400 |
| Дополнительная заработная плата | 5088 |
| Отчисления на социальные нужды | 16240,90 |
| Накладные расходы | 15671,04 |
| ИТОГО: | 86423,69 |

Цена данной дипломной разработки рассчитывается по формуле:

Цпр = Сзо + Сзд + Ссн + Снр + Матер +. САМ + Жел.приб. + НДС,

где Сзо - основная заработная плата,

Сзд - дополнительная заработная плата,

Ссн - отчисления на социальные нужды,

Снр - накладные расходы,

Матер. - расходы на материалы,

САМ - амортизация оборудования,

НДС - налог на добавочную стоимость, равный 18%,

Жел.приб. - прибыль, которую планируем, равная 35%.

Цпр =86423,69 + (86423,69 \* 0,35) +[(86423,69 + 86423,69 \* 0,35) \* 0,18]= 137672 руб. 94 коп.

4.5 ОЦЕНКА ЭКОНОМИЧЕСКОЙ ЭФФЕКТИВНОСТИ

4.5.1 НАУЧНО-ТЕХНИЧЕСКИЙ ЭФФЕКТ

Разрабатываемый модуль является ядром ПЛИС фирмы Xilinx, которая, в свою очередь, является элементом передающей системы коррекции ошибок в IP-телевидении. Как объект для продажи может рассматриваться в комплексе всей системы (передающей части IP телевидения).

Область применения разработки ограничивается только данной системой, но не ограничивается использованием ПЛИС конкретной фирмы. Данный программный модуль с незначительными изменениями может применяться также и с рядом других ПЛИС.

4.6 ВЫВОДЫ

В данном разделе была проанализирована экономическая составляющая дипломного проекта.

При анализе основных затрат очевидно, что максимальные расходы при разработке модуля приходятся на материалы и на основную заработную плату.

Исходя из расчета, себестоимость конечного продукта составляет: 86423 руб. 69 коп. Эта сумма может послужить базой для дальнейшего ценообразования при коммерческом распространении продукта.

В связи с растущей популярностью IP телевидения и большой загруженностью IP сетей передающая часть с встроенной системой коррекции ошибок должна быть востребована на рынке. Разработанный программный модуль может работать на большинстве ПЛИС фирмы Xilinx, которая является достаточно распространенной. Поэтому затраты на работу целесообразны и окупят себя.

В качестве потребителей могут рассматриваться Интернет-провайдеры, транслирующие какой-либо контент по сети. Аналогов данному модулю пока не существуют, но есть отдельные сводные библиотеки, которыми можно воспользоваться для создания аналога или усовершенствования самой схемы синхронизации, вследствие, технического прогресса.

1. безопасность жизнедеятельности

5.1 ВВЕДЕНИЕ

Целью настоящей дипломной работы является разработка ядра передающей части системы коррекции ошибок в IP-телевидении для ПЛИС Xilinx (ПЛИС – программируемая логическая интегральная схема).

Исследования проводились за персональным компьютером, поэтому в этом разделе будет проведен анализ вредных и опасных факторов, сопровождающих работу оператора ПЭВМ.

5.2 ТРЕБОВАНИЕ ПО ЭЛЕКТРОБЕЗОПАСНОСТИ ПРИ РАБОТЕ НА ПЭВМ

Место, где ведется проектирование, является помещением без повышенной опасности(ПУЭ 1.1.13).

Уровень звука не должен превышать 60 дБА (ГОСТ 12.1.003.-83). Температура и влажность в помещении должны соответствовать ГОСТ 12.1.005-88 (в холодное время - +18 - +20С и 40-60%; в теплое время - +21 - +23C и 40-60%). Освещенность рабочего места должна составлять не менее 400 лк СНиП 23-05-95

В ПЭВМ, подключаемой к сети, питание напряжения 220В. В соответствие с ПУЭ 1.1.3 ПЭВМ является установкой менее 1000В. При ее эксплуатации, контрольно – профилактических и регулировочных работах, производимых с ПЭВМ, необходимо строго соблюдать соответствующие меры предосторожности:

1. Перед включением ПЭВМ в сеть питания проверять исправность сетевого соединительного шнура и соединение зажима защитного заземления ПЭВМ с шиной защитного заземления;
2. Соединение зажима защитного заземления ПЭВМ с шиной защитного заземления производить раньше других присоединений к ПЭВМ, а отсоединение – после всех отсоединений;
3. При ремонте ПЭВМ замену любого элемента, монтаж или демонтаж производить только при отключенном от сети питания сетевом соединительном шнуре;
4. Руководствоваться техникой безопасности из руководства пользователя ПЭВМ (в комплект поставки не входит).

При работе с ЭВМ на человека воздействует вредное электромагнитное излучение, которое может вызвать ряд нарушений здоровья человека.

Требования по безопасной работе человека-оператора согласно СанПиН 2.2.2./2.4 1340 – 03.

Одними из основных элементов персонального компьютера являются системный блок и монитор. Они могут представлять опасность для здоровья человека. Системный блок закрыт сверху металлической крышкой и внутри него находится блок питания, к которому подводится питание напряжением 220В, поэтому возникает необходимость в заземлении. Заземление возможно осуществить через шнур питания, так как это предусмотрено конструкцией шнура, заканчивающегося вилкой с заземленным контактом. Работающий монитор является источником вредного излучения, состоящего из:

* оптических лучей (видимых, и/к, у/ф);
* электромагнитного излучения от самых низких частот (порядка 5 Гц) до – частот порядка сотен кГц и выше, которое опасно само по себе
* рентгеновского излучения, особенно у мониторов с большими размерами ЭЛТ;
* электростатического поля, возникающего между экраном и оператором.

Для электростатических полей, согласно ГОСТ 12.1.045-84, предельная напряженность поля, при которой допускается работать в течение часа, равна 60 кВ/м. В течение рабочей смены разрешается работать без специальных мер защиты при напряженности 20 кВ/м. Для электрического поля промышленной частоты в соответствии с ГОСТ 12.1.002-84 предельно допустимый уровень напряженности электрического поля, пребывание в котором не допускается без применения специальных мер защиты, равен 25 кВ/м. Согласно стандарту допускается пребывание персонала без специальных средств защиты в течение всего рабочего дня в электрическом поле напряженностью до 5 кВ/м. Для магнитных полей в соответствии с СанПиН 2.2.4.1191-03 установлена напряженность поля в течение рабочей смены при работе с магнитными установками и магнитными материалами равной 8 кА/м.

Поэтому монитор должен иметь гигиенические сертификаты, включающие в себя оценку визуальных параметров и подтверждающие не превышение допустимых норм излучения.

5.3 АНАЛИЗ ВРЕДНЫХ И ОПАСНЫХ ФАКТОРОВ, СОПРОВОЖДАЮЩИХ РАБОТУ НА ПЭВМ

Рабочее место при работе с ЭВМ должно отвечать требованиям СанПиН 2.2.2/2/3/1340-03, а именно:

- высота стола с клавиатурой должна составлять 62-88 см над уровнем пола, а высота экрана 90-128 см;

- расстояние от экрана до края стола 40-115 см;

- наклон экрана от -15 до + 20 градусов по отношению к нормальному его положению;

- положение спинки кресла оператора должно обеспечивать наклон тела назад 97-121 градус.

Перед началом работы следует убедиться в исправности розеток, вилок, шнура внешним осмотром. Экран следует установить таким образом, чтобы смотреть на него под прямым углом. Лучше смотреть сверху вниз, т.е. нижний край должен быть ближе к оператору. Если на экране монитора отсвечивают блики, надо установить монитор так, чтобы не было бликов. Расстояние от глаз до экрана должно быть 50-70 см. клавиатура должна располагаться не ближе, чем на 10см от края стола.

Согласно санитарных правил и норм, должны также выполняться следующие требования:

* Помещения, в которых располагаются персональные компьютеры, должны быть оборудованы одноместными столами;
* Столы и стулья должны иметь регулировочные параметры и соответствовать росту.

Так как болезнь глаз является доминирующим профессиональным заболеванием, то особое внимание следует обратить на освещенность рабочего места. Искусственное освещение в помещениях должно осуществляться системой общего равномерного освещения согласно СНиП 23-05-95.Искусственное освещение, которое необходимо использовать, приведено в таблице 7.1

Нормируемые показатели освещения Таблица 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Помещение | Освещенность при комбинированном освещении , лк | Освещенность при общем освещении, лк | Коэффициент пульсации освещенности, %, не более | Показатель дискомфорта, не более |
| Проектные залы и комнаты, конструкторские, чертежные бюро | 600/400 | 500 | 10 | 40 |

Правильный выбор цветовой гаммы оформления пользовательского интерфейса, структура экрана пользователя влияют на зрение. Продолжительная работа с ЭВМ может вызвать нарушения зрения, так и функциональные нарушения работы внутренних органов.

При работе с ЭВМ возможны следующие нарушения:

* Резь в глазах, быстрая утомляемость глаз (астенопатия);
* Проблемы с фокусировкой зрения;
* Боль в спине, пояснице, в области шеи и др.;
* Боль в плечах, руках и кистях (запястный синдром);
* Стенокардия;
* Стрессы;
* Сыпь на лице, «стягивание кожи», аллергические реакции;
* Хронические головные боли;
* Раздражительность;
* Нарушение сна;
* Возбудимость и депрессия.

Продолжительность непрерывной работы на компьютере без регламентированного перерыва не должна превышать 3-х часов. При 8 часовом рабочем дне следует делать перерыва в работе, общей продолжительностью 70 минут. Недопустимо суммирование регламентированных перерывов. Для снижения утомленности глаз, следует выполнять профилактические упражнения.

Персонал, работающий с ПЭВМ, должен иметь группу по технике безопасности не ниже 3(ПТЭ).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Жуков С. В. Алгоритмы и устройства мультиплексирования в системах цифрового телевидения: Дисс. … канд. техн. наук /СПбГЭТУ «ЛЭТИ», 2008
2. Максимов Н. В., Попов И. И. Компьютерные сети, М.: ФОРУМ, 2008
3. Кузин А.В. Компьютерные сети: учебное пособие. 3-е изд.: ФОРУМ, 2010
4. Олифер В. Г., Олифер Н. А. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 3-е изд. – СПб.: Питер, 2006
5. ISO/IEC13818-2 Generic Coding of Moving Pictures and Associated Audio: Video, (MPEG-2 Video Specification), May, 1994
6. Кривошеев М. И., Федунин В. Г. Интерактивное телевидение. – М.: Радио и связь, 2000
7. Pro-MPEG Code of Practice #3 release 2 //URL: http://www.pro-mpeg.org/documents/wancop3.pdf
8. RFC 2250 //URL: http://www.ietf.org/
9. RFC 2733 //URL: http://www.ietf.org/
10. RFC 3550 //URL: http://www.ietf.org/

ПРИЛОЖЕНИЕ – СХЕМА РАБОТЫ ЯДРА ПЕРЕДАЮЩЕЙ ЧАСТИ СИСТЕМЫ

Модуль основной:

`include "vconst.v"

module client\_my(

reset,

TS\_CLK,

ts\_parallel,

ts\_parallel\_strobe,

strobe\_47,

tx\_data,

tx\_data\_valid,

tx\_data\_start,

tx\_data\_stop,

tx\_ack,

tx\_clk

);

input reset;

output [7:0] tx\_data;

output tx\_data\_valid;

output tx\_data\_start;

output tx\_data\_stop;

input tx\_ack;

input tx\_clk;

input TS\_CLK;//input, will bу strobed (1/8) for parallel output

input [7:0] ts\_parallel;

input ts\_parallel\_strobe;

input strobe\_47;

/// ///

wire [7:0] tx\_data\_fec;

wire [7:0] tx\_data\_rtp;

wire tx\_data\_fec\_valid;

wire tx\_data\_rtp\_valid;

wire tx\_ack\_rtp;

wire tx\_ack\_fec;

wire clk\_ena = 1;

wire clk = tx\_clk;

wire [15:0] fec\_number;

wire tx\_data\_start\_fec;

wire tx\_data\_start\_rtp;

wire tx\_data\_stop\_fec;

wire tx\_data\_stop\_rtp;

reg [3:0] ts\_parallel\_strobe\_rtp\_fec;

initial ts\_parallel\_strobe\_rtp\_fec = 0;

always@(posedge TS\_CLK)

ts\_parallel\_strobe\_rtp\_fec <= {ts\_parallel\_strobe\_rtp\_fec[2:0], ts\_parallel\_strobe};

client\_my\_rtp uut (

.reset(reset),

.tx\_data\_rtp(tx\_data\_rtp),

.tx\_data\_rtp\_valid(tx\_data\_rtp\_valid),

.tx\_data\_start(tx\_data\_start\_rtp),

.tx\_data\_stop(tx\_data\_stop\_rtp),

.tx\_ack(tx\_ack\_rtp),

.tx\_clk(tx\_clk),

.TS\_CLK(TS\_CLK),

.ts\_parallel(ts\_parallel),

.ts\_parallel\_strobe\_rtp(ts\_parallel\_strobe\_rtp\_fec[3]),

.strobe\_47(strobe\_47),

.fec\_number(fec\_number)

);

client\_my\_fec eut (

.reset(reset),

.tx\_data\_fec(tx\_data\_fec),

.tx\_data\_fec\_valid\_del(tx\_data\_fec\_valid),

.tx\_data\_start(tx\_data\_start\_fec),

.tx\_data\_stop(tx\_data\_stop\_fec),

.tx\_ack(tx\_ack\_fec),

.tx\_clk(tx\_clk),

.TS\_CLK(TS\_CLK),

.ts\_parallel(ts\_parallel),

.ts\_parallel\_strobe\_fec(ts\_parallel\_strobe),

.strobe\_47(strobe\_47),

.fec\_number(fec\_number)

);

adapter adapt (

.clk(tx\_clk),

.clk\_ena(1),

.valid\_rtp(tx\_data\_rtp\_valid),

.valid\_fec(tx\_data\_fec\_valid),

.ask\_rtp(tx\_ack\_rtp),

.ask\_fec(tx\_ack\_fec),

.ask(tx\_ack),

.valid\_union(tx\_data\_valid),

.tx\_data\_start\_fec(tx\_data\_start\_fec),

.tx\_data\_start\_rtp(tx\_data\_start\_rtp),

.tx\_data\_stop\_fec(tx\_data\_stop\_fec),

.tx\_data\_stop\_rtp(tx\_data\_stop\_rtp),

.tx\_data\_start(tx\_data\_start),

.tx\_data\_stop(tx\_data\_stop),

.channal1(tx\_data\_rtp),

.channal2(tx\_data\_fec),

.channal\_union(tx\_data)

);

endmodule

Формирователь медиа пакетов:

module client\_my\_rtp(

// asynchronous reset

reset,

// direct connection to the Ethernet core's transmitter client interface.

tx\_data\_rtp,

tx\_data\_rtp\_valid,

tx\_data\_start,

tx\_data\_stop,

tx\_ack,

tx\_clk,

//transport stream input

TS\_CLK,

ts\_parallel,

ts\_parallel\_strobe\_rtp,

strobe\_47,

fec\_number

);

parameter FIVE\_UDP\_PACKETS = 24'd6579;//5263;

parameter PAYLOAD\_OFFSET = 54;//offset for 7 MPEG TP from video encoding

input reset;

output [7:0] tx\_data\_rtp;

output tx\_data\_rtp\_valid;

output tx\_data\_start;

output tx\_data\_stop;

output fec\_number;

input tx\_ack;

input tx\_clk;

input TS\_CLK;//input, will bу strobed (1/8) for parallel output

input [7:0] ts\_parallel;

input ts\_parallel\_strobe\_rtp;

input strobe\_47;

// create a synchronous reset in the transmitter clock domain

reg tx\_pre\_reset; initial tx\_pre\_reset = 0;

reg tx\_reset; initial tx\_reset = 0;

// Create synchronous reset in the transmitter clock domain.

always @(posedge tx\_clk, posedge reset)

begin

if (reset === 1'b1)

begin

tx\_pre\_reset <= 1'b1;

tx\_reset <= 1'b1;

end

else

begin

tx\_pre\_reset <= 1'b0;

tx\_reset <= tx\_pre\_reset;

end

end

reg [10:0] ts\_wr\_byte;

initial ts\_wr\_byte <= 11'd1201;

reg ts\_wr\_bank; initial ts\_wr\_bank = 0;

reg ts\_parallel\_strobe\_del; initial ts\_parallel\_strobe\_del = 0;

reg frame\_written, frame\_written\_del1, frame\_written\_del2;

initial frame\_written = 0;

reg [15:0] ip\_identification; initial ip\_identification = 16'd0;

reg [15:0] sequence\_number; initial sequence\_number = 16'd0;

always@(posedge TS\_CLK) begin

ts\_parallel\_strobe\_del <= ts\_parallel\_strobe\_rtp;

if(ts\_parallel\_strobe\_rtp) begin

if(strobe\_47 & (ts\_wr\_byte > 11'd1200)) begin//more than 6 but less thab 7 TS packets

ts\_wr\_byte <= PAYLOAD\_OFFSET-1;//after ETH, IP, UDP usw. - for next writing

ts\_wr\_bank <= ~ts\_wr\_bank;

frame\_written <= 1;

ip\_identification <= ip\_identification + 1;

sequence\_number <= sequence\_number + 1;

end

else ts\_wr\_byte <= ts\_wr\_byte + 1;

end

if(frame\_written\_del1) frame\_written <= 0;

end

// sequence number fo FEC

reg [15:0] fec\_number; initial fec\_number = 16'd0;

reg [23:0] clk\_ena\_divider; initial clk\_ena\_divider <= 24'd0;

always@(posedge TS\_CLK or posedge reset) begin

if(reset == 1)begin

clk\_ena\_divider <= 24'd0;

end

else

if(ts\_parallel\_strobe\_rtp) begin

if(clk\_ena\_divider == FIVE\_UDP\_PACKETS) begin

clk\_ena\_divider <= 24'd0;

fec\_number <= fec\_number + 4;

end

else clk\_ena\_divider <= clk\_ena\_divider+1;

end

end

reg [23:0] tx\_clk\_divider;//if no input signal, generate 1 packet pro second as test, can be DISABLED if no USE\_ONE\_SECOND\_TEST\_PACKETS

initial tx\_clk\_divider <= 24'd0;

wire tx\_valid\_strobe = (tx\_clk\_divider < 24'd1370);

wire waiting\_ack;

wire tx\_data\_start = (tx\_clk\_divider == 0) & ~tx\_reset;// need for macmii

wire tx\_data\_stop = (tx\_clk\_divider == 1369) & ~tx\_reset;// need for macmii

always @(posedge tx\_clk) begin

frame\_written\_del1 <= frame\_written;

frame\_written\_del2 <= frame\_written\_del1;

if(frame\_written\_del1 & ~frame\_written\_del2) begin

if(~tx\_data\_rtp\_valid) tx\_clk\_divider <= 0;

end

else if(~waiting\_ack & tx\_valid\_strobe) tx\_clk\_divider <= tx\_clk\_divider+1;

end

assign waiting\_ack = (tx\_clk\_divider == 0) & ~tx\_reset & ~tx\_ack;

assign tx\_data\_rtp\_valid = (tx\_valid\_strobe & ~tx\_reset); //len = 1370 = 0x55A = 1356(IP) + 14 Ethernet header + ???4 Checksum???

wire [7:0] doutb;

/\*ts\_dpram\*/

fec\_dp\_ram fec\_dp\_ram (

.clka(TS\_CLK), //input clka;

.addra({ts\_wr\_bank, ts\_wr\_byte[10:0]}), //input [11 : 0] write addra;

.dina(ts\_parallel), //input [7 : 0] dina;

.wea(ts\_parallel\_strobe\_del), //input wea;

.clkb(tx\_clk), //input clkb;

.addrb({~ts\_wr\_bank, tx\_clk\_divider[10:0]}), //input [11 : 0] read addrb;

.doutb(doutb) //output [7 : 0] doutb;

);

reg[7:0] pkt [(PAYLOAD\_OFFSET-1):0];

assign tx\_data\_rtp = (tx\_clk\_divider < PAYLOAD\_OFFSET) ? pkt[tx\_clk\_divider] : doutb;

reg [15:0] ip\_checksum;

initial ip\_checksum = 0;

always @(posedge tx\_clk) ip\_checksum <= checksum(0);

always @(\*) begin

// MAC header

pkt[0] <= 8'h01;// MAC destination 1 multicast

pkt[1] <= 8'h00;// MAC destination 2 multicast

pkt[2] <= 8'h5e;// MAC destination 3 multicast - 5e from 239. ...

pkt[3] <= 8'h7f;// MAC destination 4 multicast - 7f for energia (IP[2] = 255)

pkt[4] <= 8'h00;// MAC destination 5 multicast - 00 for energia (IP[3] = 0)

pkt[5] <= 8'h01;// MAC destination 6 multicast - 01 for energia (IP[4] = 1)

pkt[6] <= 8'h00;// MAC source 1 00

pkt[7] <= 8'hc0;// MAC source 2 c0

pkt[8] <= 8'hdf;// MAC source 3 df

pkt[9] <= 8'hfa;// MAC source 4 fa

pkt[10] <= 8'h41;// MAC source 5 8f

pkt[11] <= 8'h41;// MAC source 6 2b

pkt[12] <= 8'h08;//Type (IP)

pkt[13] <= 8'h00;

//IP header

pkt[14] <= 8'h45;//Version and header length

pkt[15] <= 8'h00;//Type of service

pkt[16] <= 8'h05;

pkt[17] <= 8'h4c;//054c Total length = 1356 bytes = 1336(UDP) + 20(IP)

pkt[18] <= ip\_identification[15:8];//8'hc7;

pkt[19] <= ip\_identification[7:0];//8'h74;// c774 identification

pkt[20] <= 8'h00;

pkt[21] <= 8'h00;//0000 - flags

pkt[22] <= 8'd254;//TTL = 254

pkt[23] <= 8'd17;//Protocol = 17 UDP

pkt[24] <= ip\_checksum[15:8];

pkt[25] <= ip\_checksum[7:0];

pkt[26] <= 8'd10;//Source IP[1]

pkt[27] <= 8'd2;//Source IP[2]

pkt[28] <= 8'd20;//Source IP[3]

pkt[29] <= 8'd2;//Source IP[4]

pkt[30] <= 8'd239;//Destination multicast IP[1]

pkt[31] <= 8'd255;//Destination multicast IP[2]

pkt[32] <= 8'd0;//Destination multicast IP[3]

pkt[33] <= 8'd1;//Destination multicast IP[4]

//UDP header

pkt[34] <= 8'h04;

pkt[35] <= 8'ha1;//Source UDP port = 0xF88 = 3976

pkt[36] <= 8'h20;

pkt[37] <= 8'h04;//Destination UDP port (2004h=8196)

pkt[38] <= 8'h05;

pkt[39] <= 8'h38;//Length 1336(0538) bytes = 1328 (payload) + 8 (UDP header)

pkt[40] <= 8'h00;

pkt[41] <= 8'h00;//Checksum, if not used can be set to 0x0000

//RTP header

pkt[42] <= 8'h80;//ver=2, p,x,cc=0

pkt[43] <= 8'h21;//M=0,Mpeg-II transport streams(33), 21h=33

pkt[44] <= sequence\_number[15:8];

pkt[45] <= sequence\_number[7:0];//25040 +1 random number

pkt[46] <= 8'h00;

pkt[47] <= 8'h00;

pkt[48] <= 8'h00;

pkt[49] <= 8'h00;//timestamp

pkt[50] <= 8'h00;

pkt[51] <= 8'h00;

pkt[52] <= 8'h00;

pkt[53] <= 8'h00;//SSRC

end

function [19:0] checksum;

input dummy;

begin

checksum = 0;

checksum = checksum + {pkt[14],pkt[15]};//14-begin of IP header

checksum = checksum + {pkt[16],pkt[17]};

checksum = checksum + {pkt[18],pkt[19]};

checksum = checksum + {pkt[20],pkt[21]};

checksum = checksum + {pkt[22],pkt[23]};

//24 and 25 are checksum

checksum = checksum + {pkt[26],pkt[27]};

checksum = checksum + {pkt[28],pkt[29]};

checksum = checksum + {pkt[30],pkt[31]};

checksum = checksum + {pkt[32],pkt[33]};//33 is end of IP header

checksum = {4'h0, checksum[15:0]} + {16'b0,checksum[19:16]};

checksum = {4'h0, checksum[15:0]} + {16'b0,checksum[19:16]};

checksum = ~checksum;

end

endfunction

endmodule

Формирователь FEC-пакетов:

`include "vconst.v"

module client\_my\_fec(

// asynchronous reset

reset,

// direct connection to the Ethernet core's transmitter client interface.

tx\_data\_fec,

tx\_data\_fec\_valid\_del,

tx\_data\_start,

tx\_data\_stop,

tx\_ack,

tx\_clk,

//transport stream input

TS\_CLK,

ts\_parallel,

ts\_parallel\_strobe\_fec,

strobe\_47,

fec\_number

);

`define FOUR\_UDP\_PACKETS 24'd5263

`define FEC\_PACKET 24'd1386

`define FIRST\_UDP\_PACKET 24'd1315

parameter PAYLOAD\_OFFSET = 70;

parameter D\_FEC = 8'h04;

parameter OFFSET = 8'h01;

// COLUMN\_FEC

parameter PORT\_FEC = 16'd8198;// Destination UDP port(if column fec=2006h=8198)

parameter TYPE\_FEC = 8'h00;// X=0; D=0 if column;type=0(XOR);index=0 for XOR

// create a synchronous reset in the transmitter clock domain

reg tx\_pre\_reset; initial tx\_pre\_reset = 0;

reg tx\_reset; initial tx\_reset = 0;

input reset;

input tx\_ack;

input tx\_clk;

output [7:0] tx\_data\_fec;

output tx\_data\_fec\_valid\_del;

output tx\_data\_start;

output tx\_data\_stop;

input TS\_CLK;//input, will bу strobed (1/8) for parallel output

input [7:0] ts\_parallel;

input ts\_parallel\_strobe\_fec;

input strobe\_47;

input [15:0] fec\_number;

always@(posedge tx\_clk, posedge reset)

begin

if (reset == 1) begin

tx\_pre\_reset <= 1'b1;

tx\_reset <= 1'b1;

end

else begin

tx\_pre\_reset <= 1'b0;

tx\_reset <= tx\_pre\_reset;

end

end

// adress

reg [10:0] ts\_wr\_byte;

initial ts\_wr\_byte <= 11'd1201;

wire clk\_ena;

assign clk\_ena = ts\_parallel\_strobe\_fec;

always@(posedge TS\_CLK or posedge reset) begin

if(clk\_ena) begin

if (strobe\_47 & (ts\_wr\_byte > 11'd1200)) begin //more then 6 but less than 7 TS packet

ts\_wr\_byte <= PAYLOAD\_OFFSET-1; //after Eth,IP,UDP,RTP usw.- for next writing

end

else ts\_wr\_byte <= ts\_wr\_byte + 1;

end

end

//sum

reg [7:0] ts\_parallel\_r; initial ts\_parallel\_r = 0;

reg [7:0] ts\_parallel\_r1; initial ts\_parallel\_r1 = 0;

reg frame\_written, frame\_written\_del1, frame\_written\_del2; initial frame\_written = 0;

reg ts\_parallel\_strobe\_del; initial ts\_parallel\_strobe\_del = 0;

reg ts\_parallel\_strobe\_del1; initial ts\_parallel\_strobe\_del1 = 0;

reg ts\_wr\_bank; initial ts\_wr\_bank = 0;

always@(posedge TS\_CLK) begin

ts\_parallel\_strobe\_del <= ts\_parallel\_strobe\_fec;

ts\_parallel\_strobe\_del1 <= ts\_parallel\_strobe\_del;

ts\_parallel\_r <= ts\_parallel;

ts\_parallel\_r1 <= ts\_parallel\_r;

end

reg [15:0] ip\_identification; initial ip\_identification = 16'd0;

reg [15:0] sequence\_number; initial sequence\_number = 16'd0;

reg [23:0] clk\_ena\_divider; initial clk\_ena\_divider <= 24'd0;

wire ts\_valid\_strobe;

wire [7:0] ts\_parallel\_out;

wire [7:0] ts\_parallel\_in;

assign ts\_valid\_strobe = (clk\_ena\_divider > `FIRST\_UDP\_PACKET);

always@(posedge TS\_CLK or posedge reset) begin

if(reset == 1)begin

clk\_ena\_divider <= 24'd0;

end

else

if(clk\_ena) begin

if(clk\_ena\_divider == `FOUR\_UDP\_PACKETS) begin

clk\_ena\_divider <= 24'd0;

ip\_identification <= ip\_identification + 1;

sequence\_number <= sequence\_number + 1;

ts\_wr\_bank <= ~ts\_wr\_bank;

frame\_written <= 1;

end

else clk\_ena\_divider <= clk\_ena\_divider+1;

end

if(frame\_written\_del1) frame\_written <= 0;

end

assign ts\_parallel\_in = (ts\_valid\_strobe) ? ts\_parallel\_r1^ts\_parallel\_out : ts\_parallel\_r1;

// delay

wire tx\_clk\_ena = 1;

wire fr\_domen\_tx\_clk = frame\_written\_del1 & ~frame\_written\_del2; // !!!

always@(posedge tx\_clk) begin

frame\_written\_del1 <= frame\_written;

frame\_written\_del2 <= frame\_written\_del1;

end

// gen. address

reg [23:0] tx\_clk\_divider\_addrd; // if no input signal, generate 1 packet pro second as test, can be DISABLED if no USE\_ONE\_SECOND\_TEST\_PACKETS

initial tx\_clk\_divider\_addrd <= 24'd0;

always@(posedge tx\_clk) begin

if(~tx\_data\_fec\_valid\_del)

tx\_clk\_divider\_addrd <= 0;

else if(~waiting\_ack & tx\_data\_fec\_valid\_del) // !!

tx\_clk\_divider\_addrd <= tx\_clk\_divider\_addrd+1;

end

assign waiting\_ack = (tx\_clk\_divider\_addrd == 0) & ~tx\_reset & ~tx\_ack;

// valid\_fec

reGenerator #(.DEL\_PACK(`FEC\_PACKET\*2), .COUNTER\_WIDTH(13)) rg1 (

.clk(tx\_clk),

.clk\_ena(tx\_clk\_ena),

.src(fr\_domen\_tx\_clk),

.sink(dffExpand),

.control(0));

reGenerator #(.DEL\_PACK(`FEC\_PACKET), .COUNTER\_WIDTH(11)) rg2 (

.clk(tx\_clk),

.clk\_ena(tx\_clk\_ena),

.src(dffExpand),

.sink(tx\_data\_fec\_vd\_start),

.control(0));

reGenerator #(.DEL\_PACK((`FEC\_PACKET)-1), .COUNTER\_WIDTH(11)) rg\_wa (

.clk(tx\_clk),

.clk\_ena(tx\_clk\_ena),

.src(tx\_ack),

.sink(tmp0),

.control(1),

.tx\_data\_start(tx\_data\_start),

.tx\_data\_stop(tx\_data\_stop));

assign tx\_data\_fec\_valid\_del = tmp0 | tx\_data\_fec\_vd\_start;

/\*ts\_dpram\*/

wire [7:0] doutb;

fec2\_dpram fec2\_dpram (

.clka(TS\_CLK), //input clka;

.addra({ts\_wr\_bank, ts\_wr\_byte[10:0]}), //input [11 : 0] write addra;

.dina(ts\_parallel\_in), //input [7 : 0] dina;

.douta(ts\_parallel\_out), //output [7 : 0] douta]

.wea(ts\_parallel\_strobe\_del1), //input wea;

.clkb(tx\_clk), //input clkb;

.addrb({~ts\_wr\_bank, tx\_clk\_divider\_addrd[10:0]}), //input [11 : 0] read addrb;

.doutb(doutb) //output [7 : 0] doutb;

);

reg[7:0] pkt [(PAYLOAD\_OFFSET-1):0];

assign tx\_data\_fec = (tx\_clk\_divider\_addrd < PAYLOAD\_OFFSET) ? pkt[tx\_clk\_divider\_addrd] : doutb;

wire [15:0] ip\_checksum = checksum(0);

always @(\*) begin

//FEC header 1386

// MAC header

pkt[0] <= 8'h01;// MAC destination 1 multicast

pkt[1] <= 8'h00;// MAC destination 2 multicast

pkt[2] <= 8'h5e;// MAC destination 3 multicast - 5e from 239. ...

pkt[3] <= 8'h7f;// MAC destination 4 multicast - 7f for energia (IP[2] = 255)

pkt[4] <= 8'h00;// MAC destination 5 multicast - 00 for energia (IP[3] = 0)

pkt[5] <= 8'h01;// MAC destination 6 multicast - 01 for energia (IP[4] = 1)

pkt[6] <= 8'h00;// MAC source 1 00

pkt[7] <= 8'hc0;// MAC source 2 c0

pkt[8] <= 8'hdf;// MAC source 3 df

pkt[9] <= 8'hfa;// MAC source 4 fa

pkt[10] <= 8'h41;// MAC source 5 8f

pkt[11] <= 8'h41;// MAC source 6 2b

pkt[12] <= 8'h08;//Type (IP)

pkt[13] <= 8'h00;

//IP header

pkt[14] <= 8'h45;//Version and header length

pkt[15] <= 8'h00;//Type of service

pkt[16] <= 8'h05;

pkt[17] <= 8'h5c;//055c Total length = 1372 bytes = 1352(UDP) + 20(IP)

pkt[18] <= ip\_identification[15:8];//8'hc7;

pkt[19] <= ip\_identification[7:0];//8'h74;// c774 identification

pkt[20] <= 8'h40;

pkt[21] <= 8'h00;//0000 - flags

pkt[22] <= 8'd254;//TTL = 254

pkt[23] <= 8'd17;//Protocol = 17 UDP

pkt[24] <= ip\_checksum[15:8];

pkt[25] <= ip\_checksum[7:0];

pkt[26] <= 8'd10;//Source IP[1]

pkt[27] <= 8'd2;//Source IP[2]

pkt[28] <= 8'd20;//Source IP[3]

pkt[29] <= 8'd2;//Source IP[4]

pkt[30] <= 8'd239;//Destination multicast IP[1]

pkt[31] <= 8'd255;//Destination multicast IP[2]

pkt[32] <= 8'd0;//Destination multicast IP[3]

pkt[33] <= 8'd1;//Destination multicast IP[4]

//UDP header

pkt[34] <= 8'h04;

pkt[35] <= 8'ha1;//Source UDP port = 0xF88 = 3976

pkt[36] <= PORT\_FEC[15:8];

pkt[37] <= PORT\_FEC[7:0];//Destination UDP port (2004h=8196, if column fec=2006h=8198, row fec=2008h=8200)

pkt[38] <= 8'h05;

pkt[39] <= 8'h48;//Length 1352(0548) bytes = 1344 (payload) + 8 (UDP header)

pkt[40] <= 8'h00;

pkt[41] <= 8'h00;//Checksum, if not used can be set to 0x0000

//RTP header

pkt[42] <= 8'h80;//ver=2, p,x,cc=0

pkt[43] <= 8'h60;//M=0,Mpeg-II transport streams(33), 60h=96

pkt[44] <= sequence\_number[15:8];

pkt[45] <= sequence\_number[7:0];

pkt[46] <= 8'h00;

pkt[47] <= 8'h00;

pkt[48] <= 8'h00;

pkt[49] <= 8'h00;//timestamp

pkt[50] <= 8'h00;

pkt[51] <= 8'h00;

pkt[52] <= 8'h00;

pkt[53] <= 8'h00;//SSRC

//FEC header

pkt[54] <= fec\_number[15:8];

pkt[55] <= fec\_number[7:0];//SNBase low

pkt[56] <= 8'h00;

pkt[57] <= 8'h00;//length recovery

pkt[58] <= 8'h80;//E=1 The heading is prolonged, PT=0

pkt[59] <= 8'h00;

pkt[60] <= 8'h00;

pkt[61] <= 8'h00;//mask=0

pkt[62] <= 8'h00;

pkt[63] <= 8'h00;

pkt[64] <= 8'h00;

pkt[65] <= 8'h00;//TS\_Recovery

pkt[66] <= TYPE\_FEC;//X=0; D=0 if column, D=1 if row;type=0(XOR);index=0 for XOR

pkt[67] <= OFFSET;//offset=L=1 for my column FEC or offset = 1 for row FEC

pkt[68] <= D\_FEC;//NA=4

pkt[69] <= 8'h00;//SNbase=0

end

function [19:0] checksum;

input dummy;

begin

checksum = 0;

checksum = checksum + {pkt[14],pkt[15]};//14-begin of IP header

checksum = checksum + {pkt[16],pkt[17]};

checksum = checksum + {pkt[18],pkt[19]};

checksum = checksum + {pkt[20],pkt[21]};

checksum = checksum + {pkt[22],pkt[23]};

//24 and 25 are checksum

checksum = checksum + {pkt[26],pkt[27]};

checksum = checksum + {pkt[28],pkt[29]};

checksum = checksum + {pkt[30],pkt[31]};

checksum = checksum + {pkt[32],pkt[33]};//33 is end of IP header

checksum = {4'h0, checksum[15:0]} + {16'b0,checksum[19:16]};

checksum = {4'h0, checksum[15:0]} + {16'b0,checksum[19:16]};

checksum = ~checksum;

end

endfunction

endmodule

Формирователь задержки:

module reGenerator(

clk,

clk\_ena,

///

src,

sink,

control,

tx\_data\_start,

tx\_data\_stop

);

parameter DEL\_PACK = 13;

parameter COUNTER\_WIDTH = 13;

input clk, clk\_ena;

///

input src, control;

output sink;

output tx\_data\_start, tx\_data\_stop;

///

reg vf\_del;

initial vf\_del <= 0;

wire pulseEnd = (vf\_del^control) & ~(src^control);

always@(posedge clk) begin

if(clk\_ena) begin

vf\_del <= src;

end

end

///

reg [12:0] del\_divider;

reg dffExpand;

reg carry;

initial begin

del\_divider <= 8'd0;

dffExpand <= 0;

carry <= 0;

end

wire count\_ena = clk\_ena & dffExpand;

always@(posedge clk) begin

if(pulseEnd & clk\_ena)

del\_divider <= 0;

else if(count\_ena)

begin

if(del\_divider == DEL\_PACK-2) begin

del\_divider <= 0; carry <= 1; end

else begin

del\_divider <= del\_divider+1; carry <= 0; end

end

end

wire tffv = clk\_ena & (pulseEnd | carry);

always@(posedge clk) begin

if(tffv) begin

dffExpand <= pulseEnd;

end

end

/// /// out

assign sink = dffExpand;

assign tx\_data\_stop = carry;

assign tx\_data\_start = pulseEnd;

endmodule

Адаптер:

module adapter(

clk,

clk\_ena,

valid\_rtp,

valid\_fec,

ask\_rtp,

ask\_fec,

ask,

tx\_data\_start\_fec,

tx\_data\_start\_rtp,

tx\_data\_stop\_fec,

tx\_data\_stop\_rtp,

/// output

valid\_union,

tx\_data\_stop,

tx\_data\_start,

// datastreams

channal1,

channal2,

channal\_union

);

input clk, clk\_ena;

input valid\_rtp;

input valid\_fec;

output ask\_rtp;

output ask\_fec;

input ask;

output valid\_union;

///

input tx\_data\_start\_fec;

input tx\_data\_start\_rtp;

input tx\_data\_stop\_fec;

input tx\_data\_stop\_rtp;

output tx\_data\_stop;

output tx\_data\_start;

///

input [`WIDTH-1:0] channal1;

input [`WIDTH-1:0] channal2;

output [`WIDTH-1:0] channal\_union;

wire sel;

assign sel = valid\_rtp | ~valid\_fec;

assign valid\_union = valid\_rtp | valid\_fec;

assign tx\_data\_start = tx\_data\_start\_fec | tx\_data\_start\_rtp;

assign tx\_data\_stop = tx\_data\_stop\_fec | tx\_data\_stop\_rtp;

/// stream immediate

assign channal\_union = sel ? channal1 : channal2;

// demux

assign ask\_fec = ask & valid\_fec;

assign ask\_rtp = ask & valid\_rtp;

endmodule

Файл с константами:

// file : `include "vconst.v"

`timescale 1ns / 100ps

`define COUNT\_RTP 40

`define COUNT\_FEC `COUNT\_RTP\*4

`define WIDTH 8

`define SEL\_DELAY 9+1