МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт Факультет приборостроения, информационных технологий и электроники	Кафедра «Информационная безопасность систем и технологий»
Специальность 100503 –	Информационная безопасность автоматизированных систем
Специализация –	Защищенные автоматизированные системы управления

ДИПЛОМНЫЙ ПРОЕКТ

на тему

УСТРОЙСТВО КРИПТОГРАФИЧЕСКОГО ПРЕОБРАЗОВАНИЯ ИНФОРМАЦИИ

Студент _____ Проценко Екатерина Дмитриевна

Руководитель

Ведущий инженер АО «ПНИЭИ» _____ Краснов В.А.

Нормоконтролёр _____ Иванов А.П.

Рецензент

Ведущий инженер АО «ПНИЭИ» _____ Печилейский С.В.

Работа допущена к защите (протокол заседания кафедры от 23.06.16 № 14)

Заведующий кафедрой _____ С.Л. Зефиров

Работа защищена с отметкой Отлично (протокол заседания ГЭК от 24.06.16 № 14)

Секретарь ГЭК _____ А.П. Иванов

Пенза 2016

Проверка ВКР – 10 баллов

С.Л. Зефиров

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

АМКЗИ – аппаратный модуль криптографической защиты информации

БКЗИ – блок криптографической защиты информации

ВС – вычислительная система

КИ – ключевая информация

КО – канальное оборудование

ЛВС – локальная вычислительная сеть

МКЗ – модуль криптографической защиты

ОЗУ – оперативное запоминающее устройство

ООД – оконечное оборудование данных

ПЛИС – программируемые логические интегральные схемы

РГИ – регистр информации

РГК – регистр ключа

СС – статусное сообщение

ТЗ – техническое задание

УК – устройство контроля

УКПИ – устройство криптографического преобразования информации

РЕФЕРАТ

Пояснительная записка содержит 61 с., 13 рис., 12 табл., 6 источников, 2 прил.

УСТРОЙСТВО КРИПТОГРАФИЧЕСКОГО ПРЕОБРАЗОВАНИЯ ИНФОРМАЦИИ, АППАРАТУРА КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ, БЛОК КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ, ШИФРО ГАММА, КРИПТОГРАФИЧЕСКИЙ АЛГОРИТМ

Цель работы – разработка функционально законченного устройства и реализация его на ПЛИС. Устройство должно обеспечивать приём IP пакетов, криптографическое преобразование используя несекретный криптографический алгоритм «Уступ», передачу зашифрованных пакетов по сетям общего пользования.

Разработка устройства криптографического преобразования информации производилось в соответствии с заданием. Средствами для разработки схемотехники устройства являлись среда Xilinx ISE 9.0. Разработка структуры и схемотехники ПЛИС, проводилась на языке VHDL.

Разработано изделие позволяющее:

– вести шифрование/расшифрование открытых/закрытых данных, используя несекретный алгоритм «Уступ» со скоростью до 100 Мбит/с;

– фильтровать Ethernet пакеты по MAC адресам, по IP адресу приемника;

– работать как самостоятельно, так и в комплексе с другими устройствами, обеспечивая наибольшую функциональность.

СОДЕРЖАНИЕ

Введение 6

1 Обзор существующих устройств криптографического преобразования информации 9

1.1 Криптомаршрутизатор многопротокольный Т-247 9

1.2 Криптомаршрутизатор серии «КМ» (КМ-03, КМ-07) 10

1.3 Ethernet сервер NPort 6250-M-SC 11

1.4 Программный комплекс Crypton IPMobile 13

2 Характеристики ПЛИС Spartan 6 15

3 Структура аппаратного модуля криптографической защиты информации (АМКЗИ) 20

3.1 Функционирование устройства 22

3.2 Контроль функционирования устройства 24

4 Разработка структуры устройства криптографической защиты информации 26

4.1 Взаимодействие с внешним процессором 26

4.2 Описание структуры устройства криптографического преобразования информации 27

5 Описание криптографического алгоритма «Уступ» 29

6 Описание функциональных модулей ПЛИС 33

6.1 Блок криптографической защиты информации 33

6.2 Контроллер Ethernet 35

6.3 Serial Interface 38

7 Описание работы MicroBlase 39

7.1 Предпусковой контроль 39

7.2 Заполнение регистров проверок 40

7.3 Рабочий режим 40

8 Описание работы устройства по структурной схеме 43

8.1 Инициализация 43
8.2 Заполнение регистров проверок 43
8.3 Изменение регистров проверок 44
8.4 Обработка команд управления 44
8.5 Установка ключа 46
8.6 Установка синхроданных 46
8.7 Запись открытых/закрытых данных 47
8.8 Чтение результата 48
9 Разработка функциональной схемы шифратора 49
9.1 Регистр ключа (РГК) 49
9.2 Регистр информации (РГИ) 50
9.3 Узлы подстановки и замены F1 и F2 51
10 Описание работы шифратора по функциональной схеме 54
10.1 Предварительное усложнение ключа 54
10.2 Выработка шифрогаммы 54
11 Разработка функциональной схемы устройства управления шифратором 55
VHDL представление устройства управления 55
VHDL представление шифратора 56
VHDL представление счетчика тактов 56
12 Расчет быстродействия устройства 58
Заключение 60
Список использованных источников 61
Приложение А 62
Приложение Б 65

ВВЕДЕНИЕ

Вопрос защиты информации является на сегодняшний день не просто актуальным, а жизненно важным с точки зрения личной и финансовой безопасности. Без знания и квалифицированного применения современных технологий, стандартов, протоколов и средств защиты информации невозможно достигнуть требуемого уровня информационной безопасности [1]. Криптографические методы – одни из наиболее эффективных средств защиты информации, особенно передаваемой по протяженным линиям связи.

В настоящее время широкое распространение получили локальные вычислительные сети, и как следствие возникла потребность в защите информации циркулирующей в подобных сетях.

Наиболее популярной технологией передачи данных в ЛВС стала технология Ethernet. С другой стороны, существует значительный комплекс наработок для шифрования данных, который в силу определенных обстоятельств не поддерживает данный интерфейс. В качестве решения этой проблемы могут выступать устройства преобразования трафика. Подобные устройства так же могут производить его фильтрацию и необходимую обработку.

На практике защита информации представляет собой комплекс регулярно используемых средств и методов, принимаемых мер и осуществляемых мероприятий с целью систематического обеспечения требуемой надежности информации, генерируемой, хранящейся и обрабатываемой на каком-либо объекте информационной системы, а также передаваемой по каналам. Защита должна носить системный характер, то есть для получения наилучших результатов все разрозненные виды защиты информации должны быть объединены в одно целое и функционировать в составе единой системы, представляющей собой слаженный механизм взаимодействующих элементов, предназначенных для выполнения

определенных задач по обеспечению информационной безопасности. Иначе говоря, при организации обеспечения информационной безопасности закономерно возникает необходимость в создании комплексной системы защиты информации, включающей в себя:

- организационную защиту информации;
- правовую защиту информацию;
- программно-аппаратную защиту информации;
- инженерно-техническую защиту информации.

Требования к данной работе как раз заключается в создании немаловажной части комплекса защиты информации – программно-аппаратного блока ее защиты.

Вопрос использования шифровальных методов в вычислительных системах (ВС) стал в наше время особенно актуальным.

Во-первых, расширилась сфера применения компьютерных сетей, таких как World Wide Web, с помощью которых передаются огромные объемы информации государственного, военного, коммерческого и личного характера, не дающего возможности доступа к ней сторонних лиц.

Во-вторых, появление современных сверхмощных компьютеров, продвинутых технологий сетевых и нейронных вычислений делает возможным дискредитацию шифровальных систем еще вчера считавшихся совершенно безопасными.

Бала доказана обоснованность проблемы защиты информации в современных системах, перейдем к вопросу реализации криптографического алгоритма.

В последнее время широкое распространение получили программируемые логические интегральные схемы (ПЛИС). На ПЛИС разрабатываются множество цифровых устройств различного назначения. В настоящее время наблюдается быстрое развитие рынка

телекоммуникационных систем, для которого остро стоит проблема защиты информации от несанкционированного доступа при передаче ее по открытым каналам связи. На сегодняшний день известно множество методов защиты данных, самым надежным и распространенным из них является криптографическое кодирование информации.

При проектировании специализированных устройств передачи данных реализуемых на ПЛИС актуальна задача реализации криптоалгоритмов на этой же элементной базе. Так как отказ от применения внешних микросхем реализующих тот или иной криптоалгоритм и реализация этого алгоритма на одной ПЛИС вместе с проектируемым устройством обладает рядом преимуществ. Во-первых, увеличивается надежность проектируемого устройства, обусловленная меньшим количеством внешних соединений. Во-вторых, уменьшаются габаритные размеры и энергопотребление устройства, за счет сокращение количества элементов схемы. В-третьих, появляется возможность замены криптографического алгоритма без внесения изменений в схему устройства.

1 Обзор существующих устройств криптографического преобразования информации

1.1 Криптомаршрутизатор многопротокольный Т-247

В настоящий момент на рынке представлено множество коммерческих устройств обеспечивающих криптографическое преобразование информации. Подобные устройства пользуются спросом как для решения специфических задач возникающих перед разработчиками защищенных телекоммуникационных сетей передачи данных, так и для поддержки старого оборудования.

Рассмотрим наиболее известные решения в этой области.

Программный комплекс в составе субмодуля защищенного (СМЗ), модуля криптографической защиты (МКЗ) Т-247К, субмодуля незащищенного (СМН) и консоли управления предназначен для создания защищенных виртуальных корпаративных сетей с мезанизмами обеспечения безопасности на сетевом уровне на базе ведомственных сетей и каналов передачи данных, а так же сетей и каналов передачи общего пользования.

Обеспечивает независимую маршрутизацию в защищенной и незащищенных зонах и криптографическую защиту с использованием протокольно-независимого проходного шифратора, высокую степень криптографической защиты от скрытых каналов утечки при работе в сетях общего пользования, в том числе в интернете.

Возможности криптомаршрутизатора Т-247:

– обеспечивает дискреционный и мандатный механизмы разграничения доступа;

– идентификацию и аутентификацию источников информации;

– автоматическую криптографическую защиту передаваемой информации;

– маскировку трафика;

– локальное и защищенное удаленное управление настройками маршрутизации и безопасности;

– защиту от несанкционированного доступа к аппаратуре и информации;

– ввод и хранение ключевой информации.

Характеристики:

– количество внешних интерфейсов: Ethernet STP-1, Ethernet STP/AUI-1,RS-232-1 (СМЗ); Ethernet STP-1, Ethernet STP/AUI-1,RS-530 (V-35)-2,RS-232-1 (СМН);

– скорость маршрутизации 11000 пакетов/с (пакеты 64 байта);

– скорость шифрованного потока не менее 6 Мбит/с в каждую сторону;

– режим работы: круглосуточный, без принудительного охлаждения;

– потребляемая мощность не более 90 Вт;

– масса 60 кг;

– размеры 482,6х457,5х266 мм.

Проанализировав данное устройство можно сделать вывод о нецелесообразности его использования для решения поставленных в проекте задач. Устройство имеет низкую скорость (6 Мбит/с), и очень большие массо-габаритные характеристики (масса 60 кг, размеры 482,6х457,5х266 мм). Так же устройство имеет высокую потребляемую мощность.

1.2 Криптомаршрутизатор серии «КМ» (КМ-03, КМ-07)

Предназначен для обеспечения криптографической защиты информации при передаче TCP/IP-протоколов данных в ведомственных сетях. Разработан на основе телекоммуникационной составляющей

технологии «Дионис».

Возможности:

– скорость шифрования до 20 Мбит/с;

– базовая конфигурация- 3 интерфейса Ethernet (включая оптический), 2 интерфейса RS-232;

Характеристики:

– дополнительные функции безопасности, расширенная система мониторинга и трассировки;

– передача интегрированных потоков видео/голос/данные с возможностью установления приоритета трафика;

– потребляемая мощность не более 30 Вт;

– габаритные размеры:412x293x67 мм;

– масса не более 3,5 кг.

Рассмотренное устройство так же не может быть использовано для решения поставленных задач, в виду низкой скорости, а так же из-за больших массо-габаритных характеристик

1.3 Ethernet сервер NPort 6250-M-SC

Ethernet сервер с криптозащитой данных 2-портовый терминальный сервер для подключения двух устройств с последовательным интерфейсом RS-232/422/485 к управляющему компьютеру или сетевому концентратору через сеть Ethernet по многомодовому оптоволокну.

NPort 6250-M-SC - это сервер устройств 3 в 1, поддерживает RS-232, RS-422 и RS-485, каждый из которых может быть выбран во время конфигурации сервера.

NPort 6250-M-SC поддерживает протоколы SSL и SSH, которые обеспечивают шифрование данных перед их передачей по сети. Кроме того, сервер оснащён системой аппаратного кодирования информации, поддерживающей алгоритмы DES/3DES/AES, что повышает общую

производительность системы. Состав и параметры устройства приведены в таблице 1.

Таблица 1 – Основные технические характеристики ЕКОН134

Последовательные порты	
Последовательные порты	RS-232/422/485 – 2
Разъем	DB-9 (Male)
Защита от импульсных помех	15 кВ
Максимальная скорость передачи данных	921600 бит/с
Управление направлением передачи данных	ADDC® (автоматическое)
Передаваемые сигналы	
RS-232	Tx, Rx, RTS, CTS, DTR, DSR, DCD, GND
RS-422	Tx+, Tx-, Rx+, Rx-, GND
RS-485 (2-проводный)	Data+, Data-, GND
RS-485 (4-проводный)	Rx+, Rx-, Tx+, Tx-, GND
Интерфейс Ethernet	
Сетевые протоколы	ICMP, IP, TCP, UDP, DHCP, BOOTP, Telnet, DNS, SNMP V1/V2c/V3, HTTP, SMTP, ARP, PPPoE
Гальваническая изоляция	1.5 кВ
Оптический интерфейс	
Оптические порты (многомодовые)	1 x SC
Тип оптоволоконного кабеля	50/125 мкм, 62.5/125 мкм
Динна волны	1300 нм
Дальность передачи	5 км
Электропитание	
Напряжение питания	=12...48 В

Рассмотренное устройство имеет низкую скорость последовательных интерфейсов, а так же слишком большое напряжение питания. Так же в устройство реализована поддержка беспроводных сетей, а это не допустимо

при использовании в составе аппаратуры защиты данных.

1.4 Программный комплекс Crypton IPMobile

Программный комплекс Crypton IPMobile предназначен для организации виртуальной частной сети (VPN). Crypton IP Mobile является VPN-агентом, который позволяет удаленным мобильным пользователям устанавливать обмен зашифрованной информацией друг с другом [2].

Crypton IP Mobile обеспечивает автоматическое шифрование и расшифрование всей передаваемой информации. Шифрование информации производится по алгоритму ГОСТ 28147-89.

Crypton IP Mobile не позволяет передавать информацию в открытом виде, т.е. Интернет можно использовать только как транспортную среду для передачи информации. Таким образом, Crypton IPMobile позволяет обмениваться данными только внутри VPN. Для полноценной работы с Интернет предусмотрена возможность временного отключения, если это не противоречит установленной политике безопасности. Есть возможность фильтрации IP-трафика, в случае если Crypton IPMobile работает в режиме криптомаршрутизатора. Так же следует отметить, что данный комплекс поддерживает обработку фрагментированных пакетов.

Основные характеристики программного комплекса Crypton IPMobile:

- поддержка Windows 2000/2003/XP;
- увеличение скорости работы до 80 Мбит/с;
- возможность работы в режиме маршрутизатора;
- возможность работы с фрагментированным IP трафиком;
- возможность работы более чем с одним сетевым адаптером;
- поддержка сетевых карт с функцией Checksum Offload;
- графический интерфейс для механизма конфигурации;
- настраиваемая многоуровневая система протоколирования работы.

Рассмотренное устройство так же не может быть использовано для

решения поставленных задач, в виду низкой скорости. Так же при работе с Crypton IP Mobile становятся недоступными компьютеры сетевого окружения в локальной сети из-за резкого снижения скорости коммуникаций при шифровании трафика.

Ни одно из рассмотренных устройств не подходит для передачи зашифрованных данных в кластерную систему, реализованную на ПЛИС, в виду больших габаритов, низкой скорости передачи данных и излишней функциональности. Так же в разрабатываемое устройство должно осуществлять фильтрацию зашифрованного Ethernet трафика и реализация подобных функций на существующих аналогах зачастую невозможно или труднореализуема.

Принимая во внимание выводы, приведенные выше, требуется разработать собственное устройство криптографического преобразования информации. В качестве целевого интерфейса будет использоваться Serial Interfase, так как он обеспечивает высокую скорость (до 3.2Гбит/с) и надежность передачи данных, устройство должно реализовывать криптографический алгоритм «Уступ». Поддержка данного интерфейса интегрирована во многие ПЛИС последних моделей и широко используется для передачи данных в кластерных системах обработки данных реализованных на ПЛИС. Так же при проектировании устройства передачи данных реализуемого на ПЛИС актуальна задача реализации криптографических алгоритмов на этой же элементной базе.

2 Характеристики ПЛИС Spartan 6

Исходя из требований ТЗ, необходимо выбрать ПЛИС отвечающую как требованиям надежности, так и способную реализовать функции разрабатываемого устройства. Проведя исследования продукции фирм Xilinx, Altera, Actel, Atmelи др., было принято решение использовать ПЛИС фирмы Xilinx линейки Spartan 6.

Семейство Spartan 6 включает два подсемейства оптимизированных для решения разных задач.

Семейство LX оптимизировано для приложений с низкой стоимостью. ПЛИС этого подсемейства поддерживают до 147 логических ячеек, 4.8 Мб памяти, интегрированные контроллеры памяти, специализированные ЦОС блоки, высокопроизводительные аппаратные IP с возможностью конфигурации [3].

Семейство LXT расширяет семейство LX за счет внедрения до восьми 3.2 Gbps GTP трансиверов и интеграции PCI Express блока, полученного от семейства Virtex для обеспечения сочетания наименьших рисков с минимальной стоимостью. Подробные характеристики семейства LXT приведены в таблице 2.

Таблица 2 – Характеристики семейства Spartan 6 LXT

Кристалл (XC6S…)	LX25T	LX45T	LX75T	LX100T
Секции (LUT)	3 758	6 822	11 662	15 822
Логические ячейки	24 051	43 661	74 637	101 261
Триггеры	30 064	54 576	93 296	126 576
Распределенная память (кбит)	229	401	692	976
Блоки памяти BRAM (по 18 кбит)	52	116	172	268
Общая емкость BRAM (кбит)	936	2 088	3 096	4 824
Блоки синхронизации (CMT)	2	4	6	6
Контакты	250	296	348	498
DSP48A1	38	58	132	180

Продолжение таблицы 2

Кристалл(XC6S…)		LX25T	LX45T	LX75T	LX100T
PCI Express блок		1	1	1	1
Контроллер памяти		2	2	4	4
Трансивер GTP		2	4	8	8
Коммерческий (C)		-2,-3,-4	-2,-3,-4	-2,-3,-4	-2,-3,-4
Индустриальный (I)		-2,-3	-2,-3	-2,-3	-2,-3
Конфигурационная память (Мбит)		6.4			
Корпус	Размер	Максимальное число пользовательских контактов / Трансиверы GTP			
CSG324	15x15	190/2	190/4		
CSG484	19x19		296/4	292/4	296/4
FG(G)484	23x23	250/2	296/4	268/4	296/4
FG(G)676	27x27			348/8	376/8
FG(G)900	31x31				498/8

Основные характеристики семейства Spartan -6 [4]:

– технология производства: 45 нм;

– LUT спроектированы с 6 входами и двумя триггерами для конвейерного использования;

– LUT с гибкой конфигурацией, могут быть сконфигурированы как логика, ОЗУ или сдвигающий регистр;

– напряжение питания ядра: 1,2В. или 1,0В. (только LX, с классом быстродействия 1L).

– низкое энергопотребление в статике и динамике:

– 60% снижение статического потребления (по сравнению с Spartan-3A);

– 50% снижение динамического потребления (по сравнению с Spartan-3A);

– поддержка большого количества стандартов ввода-вывода;

– поддержка нескольких физических интерфейсов PHY через ресурсы ввода/вывода.

Блок управления и синтеза сигналов синхронизации (CMT) поддерживает:

– низкий уровень шума, гибкую систему тактирования;

– модуль цифрового управления синхронизацией (DCM) устраняющий расфазировку синхронизирующих импульсов и искажение скважности;

– фазовую подстройку частоты;

– синтез частоты с одновременным умножением, делением и сдвигом фаз;

– 32 цепи глобальных тактовых сигналов;

– региональный ввод/вывод тактовых сигналов;

– увеличенный блок памяти емкостью до 4.8 Мбит;

– каждый блок RAM емкостью 18 кбит, можно сконфигурировать как два блока по 9 кбит;

– возможность побайтовой записи;

– независимый выбор ширины порта чтения и записи (в конфигурации ОЗУ).

Встроенный контроллер памяти обеспечивает:

– поддержку DDR, DDR2, DDR3 и LPDDR3;

– полосу пропускания до 12,8 Гбит/с;

– поддержку многопортовой шинной структуры с независимой внутренней памятью;

Технология SelectIO™:

– прием/передача данных со скоростью 1050 Мбит/с по каждой дифференциальной паре контактов;

– программируемый ток выходных каскадов до 24 мА;

– поддержка 1,2 В-3,3 В стандартов ввода-вывода;

– поддержка функции «Горячей» замены;

– поддержка HSTL и SSTL интерфейсов;

– содержит предварительный сумматор для поддержки работы симметричных фильтров.

– расширенные возможности конфигурации включают в себя:

– поддержку многофункциональных Flash подключенных через JTAG;

– мультизагрузку для удаленного обновления прошивки;

– уникальный идентификационный номер кристалла – DNA;

– шифрование конфигурационной последовательности по алгоритму AES;

– определение и исправление ошибок в битовом потоке.

– GTP трансивер поддерживает:

– скорость до 3,2 Гбит/с;

– высокоскоростные интерфейсы: Serial ATA, Aurora, 1G Ethernet, PCI Express®, OBSAI, CPRI, EPON, GPON, Display Port, and XAUI;

– низкое энергопотребление менее 150 мВ для 3,2 Гбит/с;

– поддержка объединения каналов;

– программируемый передающий предварительный корректор;

– динамическая реконфигурация модуля.

– PCI-Express блок поддерживает:

– работу с GTP трансивером;

– встроенное аппаратное IP ядро, освобождающее логические ресурсы пользователя и уменьшающее энергопотребление.

Режим энергосбережения:

– поддерживает состояние режима ожидания и может быть сконфигурирован для пробуждения от любого количества пользовательских ножек;

– поддерживает программную настройку.

– встраиваемые системы включают:

– процессор MicroBlaze 7.0;

– контроллер памяти, обеспечивающий полосу пропускания до 12,8 Гбит/с;

Проанализировав линейки ПЛИС семейства Spartan 6, наиболее подходящем был выбран ПЛИС на кристалле LX 45T. Выбор обусловлен соотношением таких параметров, как наличие GTP трансиверов, и достаточным количеством секций. Также цена выбранного ПЛИС является приемлемой для проекта, а характеристики надежности отвечают требованиям ТЗ.

3 Структура аппаратного модуля криптографической защиты информации (АМКЗИ)

В АМКЗИ передача, прием и контроль Ethernet кадров, производит выполнение команд от ООД, хранение настроечных данных (MAC- и IP-адресов), выдача СС на ООД, управление процессом функционирования всего полукомплекта.

Отсек состоит из микросхемы, реализующей физический уровень интерфейса Ethernet (PHY) и ПЛИС, необходимой для настройки параметров работы данного интерфейса и приема/передачи данных.

В качестве микросхемы физического уровня Ethernet используется микросхема 88E1111-B2-BAB1 фирмы Marvel. Отличается низким энергопотреблением (0.75Вт) и малыми размерами (9х9 мм), автоматически определяет направление приемо-передающих линий.

Serial Interface осуществляет связь с другими ПЛИС или же аналогичными устройствами поддерживающими данный интерфейс. Поддерживает работоспособность физической линии и обеспечивает целостность передаваемых данных. Структура АКЗ представлена на рисунке 1.

Основными элементами схемы является ПЛИС XC6SLX45 серии Spartan 6 фирмы Xilinx, на которой реализовано устройство криптографического преобразования информации, подробное описание представлено в пункте 4. Дополнительно к ПЛИС подключена микросхема энергонезависимой памяти (FRAM) FM25V20 фирмы RAMTRON для хранения массива случайных чисел, используемых при работе программного датчика случайных чисел, так же в состав входят интерфейс последовательных портов для ввода КИ.

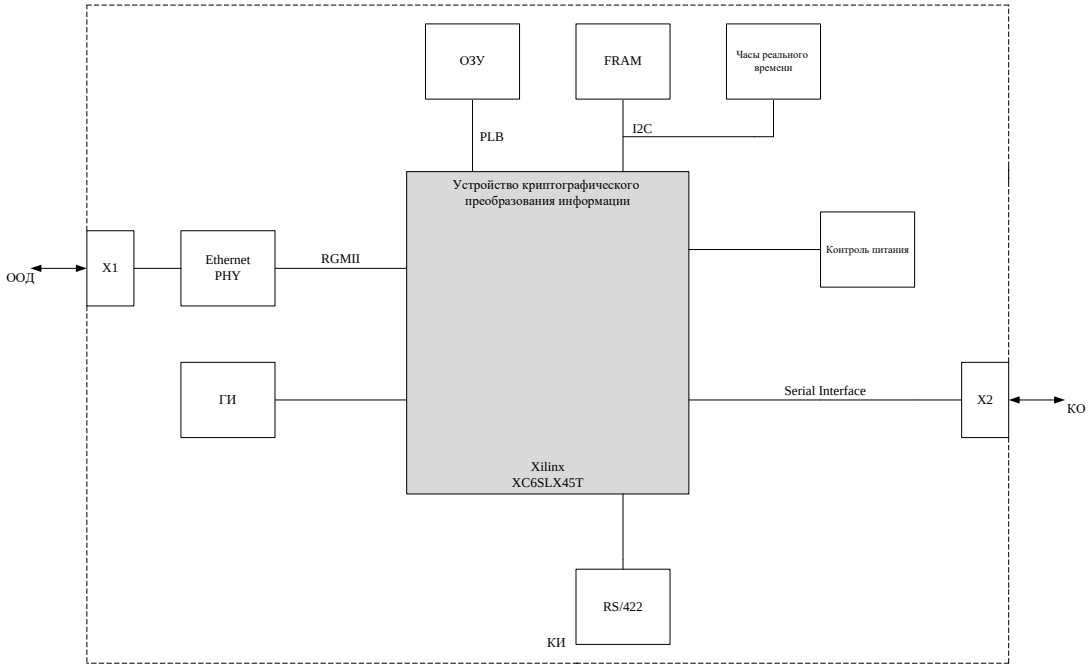


Рисунок 1 – Структура аппаратного модуля криптографической защиты информации

Информационный обмен данными ПЛИС и энергонезависимая память FRAM осуществляют по интерфейсу I2C. В процессе обмена данными ПЛИС выступает в роли «ведущего», а энергонезависимая память FRAM «ведомого». I2C представляет собой последовательную шину данных для связи интегральных схем, использующую две двунаправленные линии связи. В оперативном запоминающем устройстве (ОЗУ). во время работы устройства хранятся входные, выходные и промежуточные данные, обрабатываемые микропроцессором.

При получении пакета для шифрования ПЛИС производит проверку MAC-адресов, выравнивание по длине, затем пакет отправляется в устройство криптографического преобразования информации.

При получении расшифрованного пакета из блока, осуществляется передача его на адрес ООД. При получении сообщения от канального отсека о необходимости управления скоростью обмена – ПЛИС формирует Ethernet-кадр в соответствии с IEEE 802.3x и отправляет его на ООД.

3.1 Функционирование устройства

Устройство взаимодействует с ООД по протоколу Ethernet стандарта 100Base-T (витая пара) в соответствии с IEEE 802.3.

Устройству присвоены уникальные MAC и IP-адреса. MAC-адрес присваивается однократно при изготовлении изделия без возможности изменения, а для замены IP-адреса предусмотрена соответствующая команда.

В рабочем режиме полукомплект при приеме Ethernet-кадра проверяет MAC-адрес получателя с установленными командой «Регистрация MAC-адресов» адресами ООД или с собственным уникальным MAC-адресом. При несовпадении ни с одним из вышеуказанных адресов устройство уничтожает Ethernet-кадр.

Кроме того, устройство в рабочем режиме проверяет адрес отправителя пакета. Адрес должен совпадать с адресом ООД, с которого пришла команда «Переход в рабочий режим».

В случае если тип Ethernet-кадра указывает, что в нем находится IP-пакет от ООД, и MAC-адрес получателя совпадает с одним из адресов КО, то считается, что данный пакет предназначен для шифрования.

В этом случае ПЛИС добавляет перед IP-заголовком служебные данные (96 бит), которые состоят из собственного MAC-адреса полукомплекта, отправляющего пакет, и текущего времени суток. Формат служебного заголовка представлен на рисунке 2. Служебные данные предназначены для защиты от навязывания устаревшими пакетами противником.

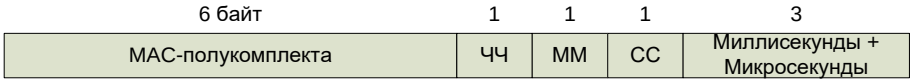


Рисунок 2 – Формат служебного заголовка

Затем пакет выравнивается по длине (дополняется случайными

значениями), после чего Ethernet-кадр направляется на шифрование.

Выравнивание производится таким образом, чтобы длина зашифрованного пакета была равна одной из четырех градаций, представленных в таблице 3.

Таблица 3 – Градация зашифрованных пакетов

Градация	Длина исходного IP-пакета	Длина зашифрованного IP-пакета
1	29…104	128
2	105…488	512
3	489…1000	1024
4	1001…1476	1500

При получении Ethernet-кадра от ООД устройство проверяет тип пакета. При совпадении типа кадра с 0x8808 изделие проверяет поля (MAC-адреса) на принадлежность к протоколу управления скоростью потока (в соответствии с IEEE 802.3x). В случае совпадения, полукомплект формирует на абонентском стыке и отправляет в ООД Ethernet-кадр управления потоком. Остальные пакеты расшифровываются.

Пакет считается ложным, если при проверке служебных данных обнаруживается факт получения дублированного или устаревшего (вышедшего за пределы доверительного интервала) пакета.

Расшифрованные пакеты устройство направляет на ООД, устанавливая MAC-адрес отправителя значением MAC-адреса КО, получившего пакет из канала связи.

Во время работы полукомплект ведет следующую статистику:

– общее количество пакетов, принятых полукомплектом от ООД;

– количество пакетов, принятых от ООД и отбракованных полукомплектом;

– общее количество пакетов, принятых полукомплектом от КО;

– количество пакетов, принятых полукомплектом от КО и отбракованных как дублированные;

– количество пакетов, принятых полукомплектом от КО и отбракованных из-за не попадания в доверенный интервал;

3.2 Контроль функционирования устройства

Устройство автоматически при включении питания проводит пусковой контроль: функциональный контроль своих внутренних узлов и тестирование, необходимое для выполнения специальных требований к изделию. Результат контроля высылается в статусном сообщении по абонентскому стыку на ООД.

Командой запроса состояния можно впоследствии запросить данный результат контроля.

При необходимости проведения функционального тестирования во время работы с ООД может быть отправлена соответствующая команда. При проведении функционального тестирования по команде штатная работа изделия (шифрование/расшифровывание) приостанавливается до окончания выполнения операции тестирования.

Для отображения результатов контроля, а также текущего состояния изделия, от каждого полукомплекта на лицевой поверхности выведены два индикатора – НОРМА и АВАРИЯ. В таблице 4 приведено описание состояния изделия, отображаемое данными индикаторами.

Таблица 4 – Состояние изделия, отображаемое индикаторами

Индикатор НОРМА	Индикатор АВАРИЯ	Описание	Код состояния
Выключен	Выключен	Производится функциональное тестирование (в начале тестирования индикаторы включаться на 1/2 секунды)	0xFF
Выключен	Горит	Обнаружена неисправность/ошибка ввода КИ	0xE0

Продолжение таблицы 4

Индикатор НОРМА	Индикатор АВАРИЯ	Описание	Код состояния
Мигает 1 раз/сек	Мигает 1 раз/сек, синхронно с НОРМОЙ	Ожидание ввода КИ	0x41
Мигает Плавно	Выключен	Ожидание команды перехода в рабочий режим	0x00
Горит	Выключен	Рабочий режим	0x01

4 Разработка структуры устройства криптографической защиты информации

4.1 Взаимодействие с внешним процессором

Структура устройства, представленная на рисунке 2, состоит из внешнего устройства представленного процессором MicroBlaze с тактовой частотой 300 МГц и системной шины PLB, работающей на 100 МГц. Процессор выполняет общее управление устройством, сброс, инициализацию, подачу входных пакетов данных, чтение результата, наблюдение за сигналами состояния[5].

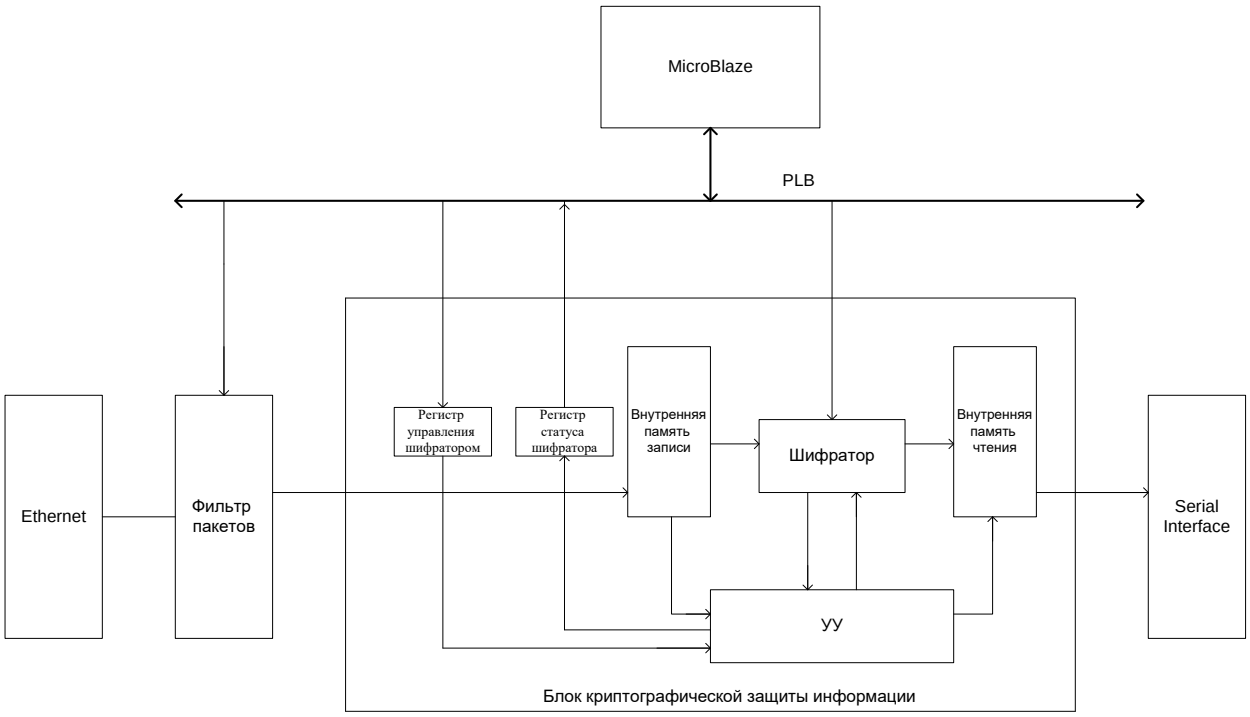


Рисунок 2 – Структура устройства криптографического преобразования информации

MicroBlaze получает доступ к внешним устройствам через высокоскоростную локальную шину PLB (Processor Local Bus). Интерфейс PLB обеспечивает раздельные 32 битную адресную и 64 битную данных шины для передачи данных и команд. Таким образом, проектируемое устройство будет отображаться в адресном пространстве процессора, благодаря чему мы получаем простой и удобный способ управления им.

Следует выделить два адресных блока, доступных к чтению/записи извне:

– Register Base Address (0xFFFFD000): внутренняя память чтения/записи и регистры устройства управления шифратором;

– Ustup Base Address (0xFFFFC800): блок памяти RAM, находящийся непосредственно в шифраторе.

4.2 Описание структуры устройства криптографического преобразования информации

Исходя из структуры устройства приведенной на рисунке 2 модуль можно разделить на внутренние компоненты, которые реализованы на ПЛИС и внешние, к которым относятся процессором MicroBlaze.

К внутренним компонентам относятся функциональные блоки реализованные в ПЛИС (в дальнейшем «Блок криптографического преобразования информации»).

Устройство криптографического преобразования информации обеспечивает приём IP пакетов, криптографическое преобразование используя несекретный криптографический алгоритм «Уступ», передачу зашифрованных пакетов по сетям общего пользования.

Внутри устройства реализованы следующие функциональные модули:

Контроллер Ethernet осуществляет преобразование сигналов поступающих от PHY контроллера в информацию пригодную для обработки. В этом компоненте накапливаются принятые кадры. Так же все передаваемые кадры записываются во внутреннюю память компонента.

Serial Interface служит для преобразования поступающих в него данных в последовательные сигналы на линии связи.

Устройство управления шифратором принимает входные сигналы от MicroBlaze. Основными модулями устройства управления являются блоки RAM, представляющие собой адресное пространство шифратора, а также

схема дешифрации команд и счетчик тактов, в которой хранит значения регистров проверок (MAC, IP адреса). Регистр управления шифратором содержит поля для сигналов сброса, инициализации.

Регистр статуса шифратора содержит информацию о текущем состоянии устройства.

Регистры управления шифратором содержит регистры управления, статуса, идентификаторов

Регистр чтения данных из внутренней памяти читает результат шифрования/расшифрования.

Регистр записи данных во внутреннюю память записывает входные для шифратора данные.

5 Описание криптографического алгоритма «Уступ»

Согласно заданию на дипломное проектирование, шифратор должен реализовывать криптографический алгоритм «Уступ». Описание и схема которого представлены на рисунке 3.

Ключом алгоритма является двоичная последовательность длины 512 бит.

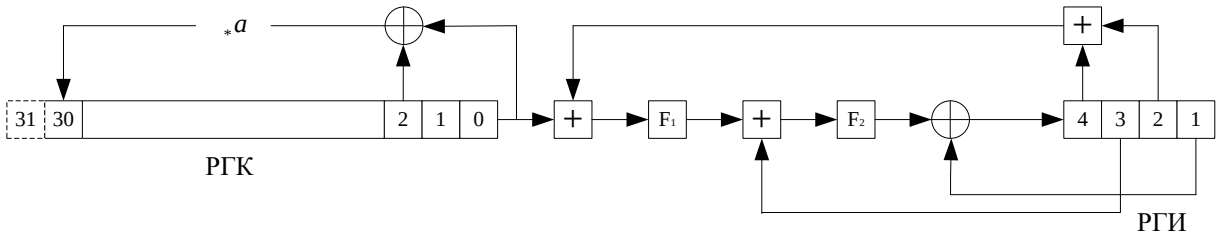


Рисунок 3 – Криптогрфическая схема выработки шифрогаммы

Криптографическая схема содержит следующие узлы:

РГИ - информационный регистр для записи 4 слов по 16 бит исходной информации;

РГК — ключевой регистр для записи 32 слов по 16 бит ключа;

$\oplus$ - 16-разрядный сумматор для поразрядного сложения слов по mod 2;

+ - 16-разрядный сумматор для сложения слов по mod 216;

В РГК реализуется рекуррентная последовательность над GF(2^{16}), описываемая неприводимым над GF(2^{16}) характеристическим полиномом $g(x) = x^{31} \oplus \alpha * x^2 \oplus \alpha$, где α - примитивный элемент поля GF(2^{16}), порожденного примитивным над GF(2^{16}) полиномом $h(x) = x^{16} \oplus x^{12} \oplus x^3 \oplus x \oplus 1, \alpha = (0,...,0,1,0)$.

Умножение описывается следующим образом. Пусть Y=(y15, ... , Уо) - элемент поля, тогда

$$\alpha * Y = \begin{cases} (y_{14}, \dots, y_0), \text{если } y_{15} = 0 \\ (y_{14}, \dots, y_{11} \oplus 1, y_{10}, \dots, y_2 \oplus 1, y_1, y_0 \oplus 1, 1), \text{если } y_{15} = 1 \end{cases}$$

Каждая подстановка F1 и F2 представляет собой суперпозицию преобразований:

F1 - старший и младший байты преобразуются с помощью подстановки f, а в получившемся слове их значения меняются местами;

F2 - старший и младший байты преобразуются с помощью подстановки f, а получившееся слово циклически сдвигается на 3 разряда в сторону старших разрядов.

Значения подстановки f (16-i+j) приведены в таблице 5.

Таблица 5 – Значения подстановки на узлах F1, F2

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	253	35	104	221	79	174	163	14	29	160	121	38	92	233	131
1	52	152	108	247	114	159	27	2	106	110	78	212	34	153	97	244
2	202	211	95	234	51	59	54	130	147	200	10	224	26	100	69	194
3	113	185	99	3	149	115	226	58	148	7	186	180	133	181	253	10
4	152	63	232	122	78	98	109	196	125	207	83	22	42	241	120	169
5	167	151	64	82	75	198	67	238	230	147	127	184	246	245	172	145
6	142	39	65	198	154	32	150	49	96	176	23	239	86	170	210	80
7	66	216	228	134	21	161	98	56	242	22	40	62	13	18	3	100
8	39	197	175	87	162	126	78	254	146	123	155	91	37	165	250	19
9	187	229	73	117	57	136	124	55	0	68	189	169	140	191	48	16
10	45	208	190	25	71	20	218	28	255	9	239	50	254	137	61	81
11	43	118	237	235	182	69	158	8	72	236	10	103	85	5	163	111
12	204	141	36	166	15	116	119	1	183	222	197	24	87	195	21	156
13	79	177	116	171	223	178	88	107	201	173	162	94	44	74	95	231
14	164	220	34	47	289	30	243	144	103	192	168	83	213	12[r]	77	46
15	70	179	4	53	11	249	105	140	248	33	4	209	205	255	234	217

Алгоритм работает следующим образом.

В начале ключ записывается в РГК. Одновременно в РГИ записывается синхропосылка длины 64 бит. Выработка синхропосылки осуществляется с помощью рекуррентной последовательности с начальным состоянием, индивидуальным для каждого аппарата. Рекуррента порождается примитивным полиномом $f(x)=x^{127} \oplus x^{63} \oplus 1$ над полем $GF(2^{16})$.

Далее, при каждом перезапуске, рекуррента прорабатывает 127 тактов. Первые 64 знака нового заполнения используются в качестве синхропосылки.

После записи ключа и синхропосылки схема в течение 64 тактов работает в режиме предварительного усложнения по схеме, изображенной на рисунке 4. При этом РГК используется как память из 32 ячеек, работающая в режиме "считывание-запись". Из РГК на узел сложения по mod 2^{16} поступает слово из ячейки с номером, определяемым пятью младшими разрядами слова 4-ой ячейки РГИ, а на его место записывается слово из первой ячейки РГИ.

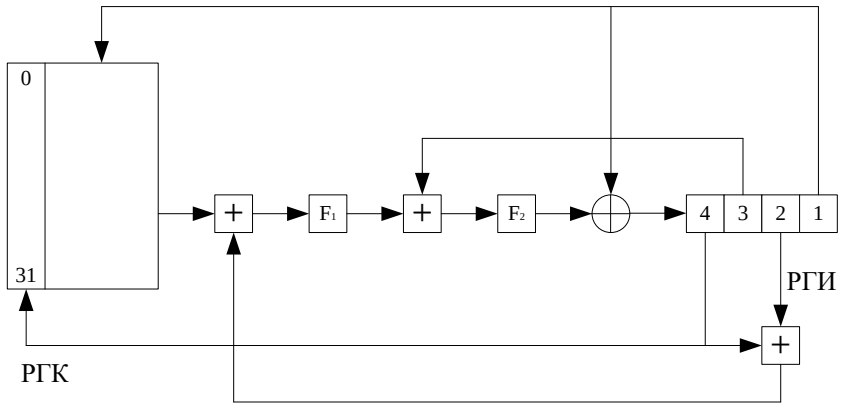


Рисунок 4 – Криптографическая схема работы алгоритма предварительного усложнения

Далее начинается рабочий режим. РГК работает в режиме ОЛЗ на 31 ячейку в соответствии со схемой, изображенной на рисунке 3. Обозначим:

$a_1(l)$, $a_2(l)$, $a_3(l)$, $a_4(l)$ - слова состояния регистра РГИ после i-го такта;

$x_0(1)$, ... , $x_{31}(1)$ - слова состояния регистра РГК после 1-го такта.

Уравнения преобразования алгоритма, приведенного на рисунке 4, имеют вид:

$$a_k^{(i)} = a_{k+1}^{(i-1)} \text{ при } k = 1 \div 3;$$

$$a_4^{(i)} = a_1^{(i-1)} \oplus F_2\left(F_1\left(a_2^{(i-1)} + a_4^{(i-1)} + x_R^{(i-1)}\right) + a_3^{(i-1)}\right),$$

где R – $a_4^{(i-1)}$ приведенное по модулю 32 (R≡ $a_4^{(i-1)}$mod32;

$$x_j^{(i)} = x_j^{(i-1)}, \text{при } j \neq R;$$

$$x_R^{(i)} = a_1^{(i-1)}.$$

Уравнения преобразования алгоритма, приведенного на рисунке 3, имеют вид:

$$a_k^{(i)} = a_{k+1}^{(i-1)} \text{ при } k = 1 \div 3;$$

$$a_4^{(i)} = a_1^{(i-1)} \oplus F_2\left(F_1\left(a_2^{(i-1)} + a_4^{(i-1)} + x_0^{(i-1)}\right) + a_3^{(i-1)}\right),$$

$$x_j^{(i)} = x_{j+1}^{(i-1)}, \text{при } j = 0 \div 29;$$

$$x_{30}^{i} = \alpha * (x_0^{(i-1)} \oplus x_2^{(i-1)}.$$

После каждых 32 тактов работы схемы (один цикл) слова $a_4(32k)$, $a_3(32k)$, a_2 и 12 старших разрядов слова $a_1(32k)$ (k>l) заполнения РГИ используются в качестве гаммы шифрования.

6 Описание функциональных модулей ПЛИС

6.1 Блок криптографической защиты информации

Блок криптографической защиты информации является главной составляющей устройства, где происходят основные процессы криптографического преобразования информации и передача зашифрованных пакетов по сетям общего пользования. Блок состоит из регистров устройства управления шифратором, регистров внутренней памяти чтения/записи и непосредственно самого шифратора, подробное описание которых представлено в таблицах 6, 7 и 8.

Таблица 6 – Список регистров устройства управления шифратором

Название	Адрес	Доступ	Описание	Назначение
Регистр управления шифратором	Register Base Address + 0x04	R/W	Содержит поля для сигналов сброса, инициализации	Установка в ‘1’ младшего бита, сбрасывает шифратор на любой выполняемой им операции. Возвращение нулевого значения равносильно начальной инициализации устройства
Регистр статуса шифратора	Register Base Address + 0x00	RO	Содержит информацию о текущем состоянии устройства	Доступный только для чтения регистр, счетчики ошибок, отладочную информацию о состоянии шифратора
Регистр-счетчик готовых пакетов	Register Base Address + 0x08	RO		Инкремент значения данного регистра говорит о успешном шифровании/расшифровании очередного пакета данных. Счетчик декрементируется в случае прочтении ненулевого значения.

Регистры управления внутренней памятью чтения (RFF Register Address) содержат информацию управления, статуса, идентификаторов.

Адресное пространство: Register Base Address + 0x100.

Регистры управления внутренней памятью записи (WFF Register Address) содержат информацию управления, статуса, идентификаторов. Адресное пространство: Register Base Address + 0x300.

Регистр чтения данных из внутренней памяти (RFF Data Address) по адресу Register Base Address + 0x200 читает результат шифрования/расшифрования.

Регистр записи данных во внутреннюю память (WFF Data Address) по адресу Register Base Address + 0x400 записывает входные для шифратора данные.

Таблица 7 –Регистры внутренней памяти чтения/записи

Название	Смещение от базового адреса	Доступ	Сокращение
Регистр сброса	Register Address + 0x00	Write	RST
Регистр идентификаторов	Register Address + 0x00	Read	MIR
Регистр статуса	Register Address + 0x04	Read	STATUS
Регистр данных	Data Address + 0x00		PFIFO_DATA

Регистр сброса. Сброс WFF, RFF происходит, когда по адресу регистра записывается значение 0x0000000A.

Регистр идентификации. Регистр, доступный только для чтения содержит идентификационную информацию о данном блоке, которая может быть использована для проверки корректности всей системы.

Регистр статуса. Регистр статуса, доступен только для чтения, содержит информацию о заполнении WFF, RFF

Регистр данных. Запись любого значения в этот регистр для WFF – инициирует загрузку этих данных в область памяти WFF, в случае если оно уже не полное. Чтение из регистра данных для RFF – инициирует выгрузку

данных из области памяти RFF, если оно не пустое.

Адресное пространство регистров шифратора , размещено начиная с Ustup Base Address и имеет длину 32 двойных слова. Память представляет собой две параллельно размещенные RAM, расположенные непосредственно в шифраторе.

Таблица 8 – Адресное пространство шифратора

Смещение	Длина поля (байт)	Доступ	Название	Описание
0x00	8	R/W	Синхроданные	Начальное заполнение регистра РГИ
0x08	32	R/W	Ключ	Начальное заполнение РГК
0x48	8	R/W	Открытые/закрытые данные	Данные для шифрования/расшифрования
0x50	8	RO	Иммитозащитная вставка	Передаются с зашифрованным пакетом
0x58	8	RO	Результат шифрования/расшифрования	Выработанная ШГ, наложенная на входные данные
0x78	4	R/W	Статус шифратора	Показывает завершение текущей операции
0x7C	4	WO	Управление шифратором	Команды, управляющие работой шифратора

6.2 Контроллер Ethernet

Данный компонент представляет собой стандартный компонент библиотеки фирмы Xilinx с оболочкой осуществляющей управление ядром. Входы осуществляющие управление ядром описаны в таблице 9.

Таблица 9 – Контакты ядра Ethernet.

Сигнал	Направление	Описание	Значение
glbl_rstn rx_axi_rstn tx_axi_rstn	Вход	входы сброса, подключаются к глобальной цепи, возвращают компонент в начальное состояние	
rx_axi_clk tx_axi_clk	Вход	входы тактирования приемной и передающей части компонента. rx_axi_clk задает PHYконтроллер	125 Мгц
rx_enable tx_enable	Вход	входы разрешения работы приемной и передающей части	VCC
x_axis_mac_tdata[7:0]	Вход	Очередной байт передаваемого кадра	
tx_axis_mac_tvalid	Вход	Управляющий сигнал для tx_axis_mac_tdata. Подтверждает корректность данных на входе	
tx_axis_mac_tlast	Вход	Управляющий сигнал для tx_axis_mac_tdata. Сообщает о конце передаваемого кадра	
tx_axis_mac_tuser	Вход	Сообщает об ошибках в передаваемом кадре	
tx_axis_mac_tready	Выход	Выставляется когда передаваемые данные отправлены	
rx_axis_mac_tdata[7:0]	Выход	Очередной байт приятого кадра	
rx_axis_mac_tvalid	Выход	Управляющий сигнал для rx_axis_mac_tdata. Подтверждает корректность данных на выходе	
rx_axis_mac_tlast	Выход	Сообщает о конце принимаемого кадра	

Продолжение таблицы 9

Сигнал	Направление	Описание	Значение
rx_axis_mac_tuser	Выход	Сообщает об ошибках в принимаемом кадре	
rx_axis_filter_tuser	Выход	Выход используется для отсеивания кадров по MACадресу	Не используется
tx_statisitics_vector[31:0] rx_statistics_vector[27:0]	Выход	Содержат статистику о прнятых/переданных кадрах	
pause_req	Вход	Включает паузу при передаче кадра	
speedis100 speedis10100	Выход	Сообщает скорости соединения установленной на PHYконтроллере	
mdc	Выход	Частота для PHY контроллера	2.5Мгц
mdio_i	Вход	Входной сигнал для связи с PHY контроллером	VCC если не используется
mdio_o	Выход	Выходной сигнал для связи с PHY контроллером	
mdio_t	Выход	Управляет шиной MDIO	0 – К шине подключается mdio_o 1 – К шине подключается mdio_i
rgmii_txd[3:0]	Выход	Данные передаваемые на PHY	
rgmii_tx_ctl	Выход	Управляющий сигнал для PHY	
rgmii_txc	Выход	Частота для PHY	
rgmii_rxd[3:0]	Вход	Данные принятые от PHY	
rgmii_rx_ctl	Вход	Управляющий сигнал от PHY	
rgmii_rxc	Вход	Частота от PHY	

Так же с ядром используется блочная память, организованная по

принципу FIFO, в которой сохраняются принятые и передаваемые кадры. Память рассчитана на хранение одного кадра размером до 2 Кб.Контроллер MDIOинтерфейса, контроллер RGMII, блоки синхронизации частоты и сигнала сброса, блок декодирования регистра статуса ядра, управляющие автомат AXI шины. Опционально подключаются блоки: генератор шаблонов и блок проверки кадров, которые могут быть использованы для выполнения функции ping. Для тактирования блока используется DCM.

6.3 Serial Interface

Последовательный интерфейс для передачи данных использует одну сигнальную линию, по которой информационные, биты передаются друг за другом последовательно. В SI одно устройство является ведущим (управляющим) и задаёт синхронизацию для ведомого устройства. Последовательная передача позволяет сократить количество сигнальных линий и увеличить дальность связи.

При передаче каждому байту предшествует старт-бит, сигнализирующий приемнику о начале посылки, за которым следуют биты данных и, возможно, бит паритета (четности). Завершает посылку стоп-бит, гарантирующий паузу между посылками. Старт-бит следующего байта посылается в любой момент после стоп-бита, то есть между передачами возможны паузы произвольной длительности. Старт-бит, имеющий всегда строго определенное значение (логический 0), обеспечивает простой механизм синхронизации приемника по сигналу от передатчика. Подразумевается, что приемник и передатчик работают на одной скорости обмена. Внутренний генератор синхронизации приемника использует счетчик-делитель опорной частоты, обнуляемый в момент приема начала старт-бита.

7 Описание работы MicroBlase

7.1 Предпусковой контроль

Микроконтроллер MicroBlase выполняет функции устройства контроля (УК) и управления. Он осуществляет обработку команд управления, запись ключевой и случайной информации в криптографический преобразователь, а так же хранит настройки устройства преобразования интерфейсов и обеспечивает целостность хранимой информации.

После подачи питания на устройство, микроконтроллер в течение 10 секунд проводит проверку работоспособности и сохраняет статистику.

В предпусковой контроль входят:

- включение на 1 секунду и выключение НОРМА и АВАРИЯ;
- проведение функционального тестирования процессором MicroBlaze представлено на рисунке 6:
  - сброс Read, Write FIFO;
  - сброс шифратора;
  - установка ключа;
  - установка синхроданных;
  - запись маркера начала пакета в WWR;
  - запись открытых/закрытых данных в WWR;
  - чтение маркера начала пакета из RWW;
  - чтение открытых/закрытых данных из RWW.
- прием команд от ООД и отправка СС на ООД;
- успешный и корректный ввод КИ;
- ввод команды перехода в рабочий режим;
- установка часов.

Так как интерфейс I2C не имеет обратной связи в режиме ведомого,

его проверка осуществляется чтением значений регистров ошибок интерфейса.

Для тестирования RAM и FRAM памяти, в них записываются последовательность чисел, затем считывается и проверяется на совпадение.

Вся хранимая в УК информация хранится в соответствующих зонах, каждая зона имеет контрольную сумму хранения, в течении предпускового контроля контрольные суммы проверяются. Если зона повреждена, происходит цикл восстановления зоны, после чего проверка повторяется и выносится решение о целостности зоны.

В процессе предпускового контроля таймер должен несколько раз досчитать до фиксированного значения, при этом проверяется затраченное на это количество тактов. Результат предпускового контроля показан на рисунке 5.

7.2 Заполнение регистров проверок

После предпускового контроля, УК выставляет УКПИ сигнал о готовности к работе, после этого через интерфейс I2Cв блок преобразования интерфейсов передаются значения регистров проверок. Если же начальные значения отсутствуют, в регистр статуса УК заносятся соответствующие значения и цикл передачи не происходит.

7.3 Рабочий режим

Если УК успешно передало начальные значения, то оно переходит в рабочий режим, и ожидает от УКПИ команд управления. Если команды управления отсутствуют, то микроконтроллер раз в 1 секунду производит обновление статистики работы устройства. Новые значения статистики передаются в УК и сохраняются во внутренней памяти.

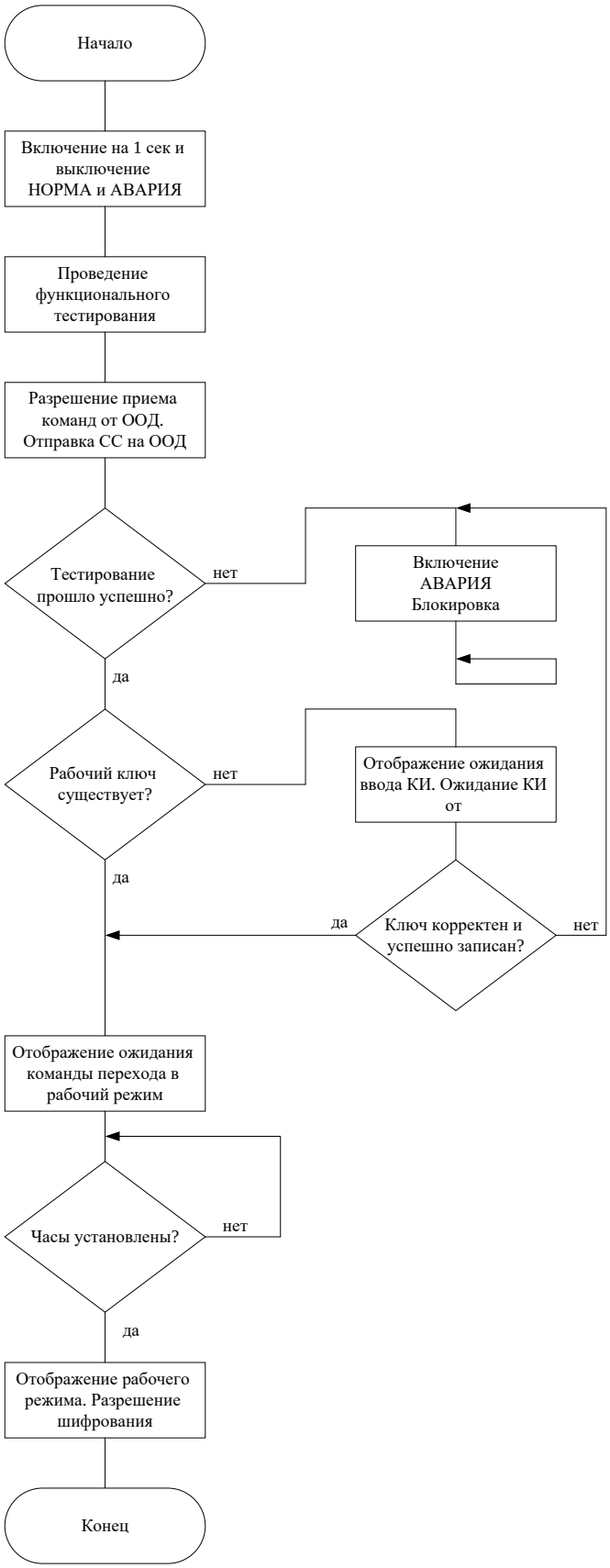


Рисунок 5 – Предпусковой контроль

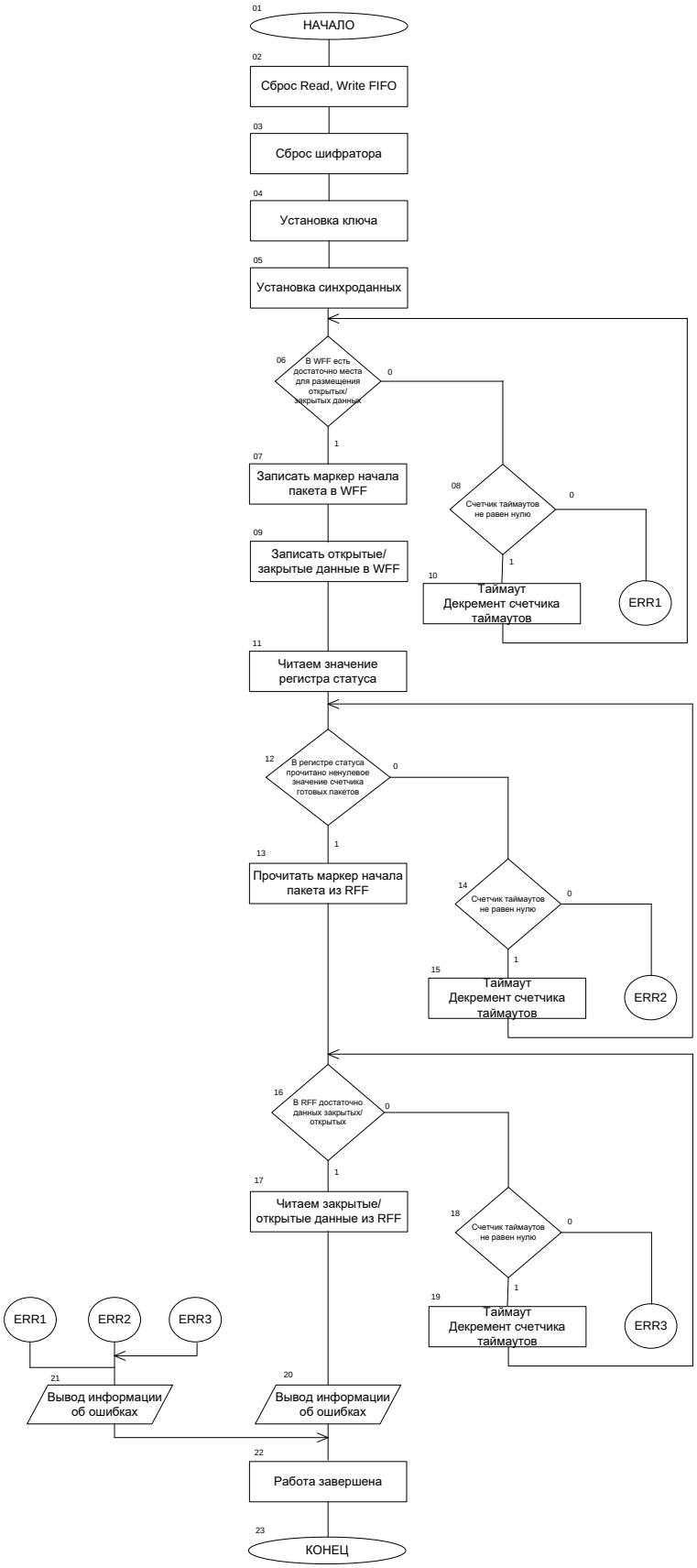


Рисунок 6 – Проведение функционального тестирования процессором MicroBlaze

8 Описание работы устройства по структурной схеме

8.1 Инициализация

Процесс, предназначенный для приведения всех блоков устройства в начальное состояние – состояние готовности к шифрованию/расшифрованию очередных данных.

Сразу после подачи питания в течении не менее чем 250 микросекунд вырабатывается сигнал сброса, который подается на все участки схемы и приводит блок в начальное состояние. После этого схема ожидает сигнала готовности от всех компонентов. Время ожидания не превышает 100 микросекунд.

– Сброс WFF и RFF. Осуществляется путем записи в регистр сброса FIFO управляющего сигнала 0x0000000A. По этому сигналу, все данные, до этого момента времени, размещенные в области памяти RFF и WFF освобождаются, флаги статуса сбрасываются на значения по умолчанию.

– Сброс шифратора. Осуществляется путем записи в регистр управления шифратора значения 0x00000001. По этому сигналу, шифратор прекращает любую текущую операцию, сбрасывает текущий введенный ключ, все флаги и триггеры переводятся в значение по умолчанию.

8.2 Заполнение регистров проверок

Сразу после инициализации и получения прерывания от устройства контроля (УК), устройство криптографического преобразования информации (УКПИ) запрашивает начальное заполнение регистров проверки кадра.

Последовательность запросов к УК:

– запрос состояния микроконтроллера (команда 01h). Чтение 16 слов, начиная с адреса C2h.

– запрос MAC адреса источника (команда 06h). Чтение 3 слов, начиная с адреса 00h.

– запрос MAC адреса получателя (команда 07h). Чтение 3 слов, начиная с адреса 06h.

– запросIPадреса устройства (команда 08h). Чтение 2 слов, начиная с адреса 08h.

– запрос UDPпорта отправителя (команда 09h). Чтение слова по адресу 0Ah

После получения всех начальных значений регистров, они в течение 25 тактов сохраняются в регистрах УКПИ и схема переходит в состояние ожидания кадров для передачи.

8.3 Изменение регистров проверок

В случае если устройство запускается первый раз, либо начальные значения были стерты из памяти микроконтроллера после инициализации, блок переходит в режим ожидания команды перевода в рабочий режим.

После получения команды, УКПИ передает в УК значения MAC адреса источника, получателя, IP адрес устройства, UDP порт отправителя. После завершения цикла обмена с УК, вырабатывается сигнал сброса и происходит инициализация устройства с новыми параметрами.

После перехода в рабочий режим устройство выполняет свою основную функцию – преобразование и шифрование пакетов.

8.4 Обработка команд управления

В рабочем режиме кроме пакетов с целевой информацией, в устройство могут приходить команды управления. Все команды управления приходят только по стыку Ethernet. В ответ на команду устройство выдает квитанцию B8F0F0C1h если команда обработана успешно, и квитанцию B8F1F1D3h если произошла ошибка при обработке команды управления.

Команды управления:

– команда перехода в рабочий режим;

– команда запроса статистики;

– команда смены IP;

– команда стирания;

– команда расширенного контроля;

Команда перехода в рабочий режим.

Команда переводит устройство в рабочий режим. В этом режиме устройство преобразует пакеты полученные по интерфейсу Ethernet в интерфейс Rocket и наоборот.

Команда запроса статистики.

Команда имеет формат, представленный в таблице 10 и в поле данных содержит константу A8C5E341h.

Таблица 10 – Формат команды управления

UDP – порт отправителя	UDP– порт получателя
Длина UDP – сообщения	Контрольная сумма
Данные	

После получения команды, устройство в течении 100мс должно выдать статистику в устройство, которое её запросило. В статистику входят как регистры ПЛИС, так и регистры УК полученные после старта устройства.

Команда смены IP.

Команда имеет формат, представленный в таблице 9. И в поле данных содержит константу D9F53C12h.

Команда предназначена для смены IPадреса устройства. После получения новый IPпередается в УК, а устройство перезапускается.

Команда стирания.

Команда имеет формат, представленный в таблице 9. И в поле данных содержит константу 5533C34Ah.

После получения команды стирания, УКПИ транслирует команду в УК. После подтверждения стирания от УК устройство перезапускается и

ждет команды перевода в рабочий режим.

Команда расширенного контроля.

Команда имеет формат, представленный в таблице 10. И в поле данных содержит константу F13C4A52h.

8.5 Установка ключа

Шифрование/расшифрование информации проводится на ключе, который необходимо занести в шифратор перед началом всех остальных операций. Временная диаграмма записи ключа представлена на рисунке 7. Ключ представляет из себя последовательность из 64 байт и заносится в адресное пространство шифратора по 4 байта по адресу, начиная с Ustup Base Address + 0x08.По команде «Занести ключ в шифратор» - запись слова 0x00000001 по адресу Ustup Base Address + 0x7C, ключ из внутреннего RAM в течение 32 тактов переносится в РГК. Об успешной установке ключа, говорит значение 0x00000003 размещенное по адресу Ustup Base Address + 0x7C. Только дождавшись этого сигнала, можно переходить к следующему шагу. В противном случае, по истечении таймаута на занесение ключа, необходимо сбросить шифратор и повторить попытку.

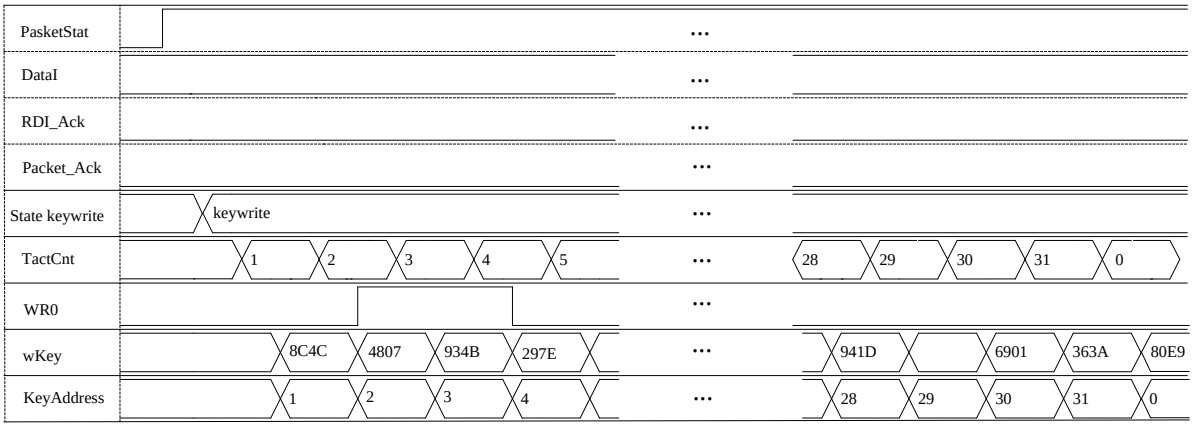


Рисунок 7 – Временная диаграмма записи ключа

8.6 Установка синхроданных

Еще один массив данных, участвующий в шифровании/расшифровании – синхроданные. Представляют из себя 8-ми

байтную последовательность информации. Временная диаграмма записи синхроданных представлена на рисунке 8. Заносятся в адресное пространство шифратора, начиная с адреса Ustup Base Address + 0x00 последовательно по 4 байта. По команде «Сменить синхроданные» - запись слова 0x00000002 по адресу Ustup Base Address + 0x7C, синхроданные переносятся в РГИ за 1 такт.

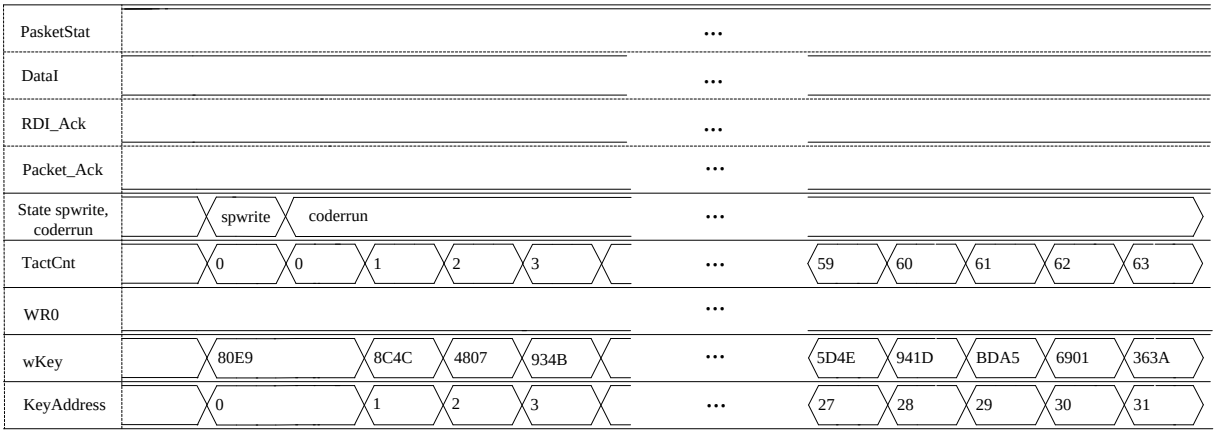


Рисунок 8 – Временная диаграмма записи синхроданных

8.7 Запись открытых/закрытых данных

После успешного занесения ключа и синхроданных в шифратор, можно подавать открытые/закрытые данные (в зависимости от выбранного режима работы). Перед записью необходимо убедиться, что WFF готово к получению очередного пакета, т.е. не пусто. Это операция осуществляется путем проверки флага FULL в регистре статуса WFF, затем длина входного пакета данных сравнивается с числом свободных ячеек WFF, данную информацию предоставляет все тот же регистр статуса, расположенный по адресу WFF_Register_Address + 0x04.

Каждый входящий пакет обрамляется контрольным словом. Пакеты становятся в очередь, ожидая завершения шифрования/расшифрования предыдущих. Временная диаграмма шифрования пакетов представлена на рисунке 9.

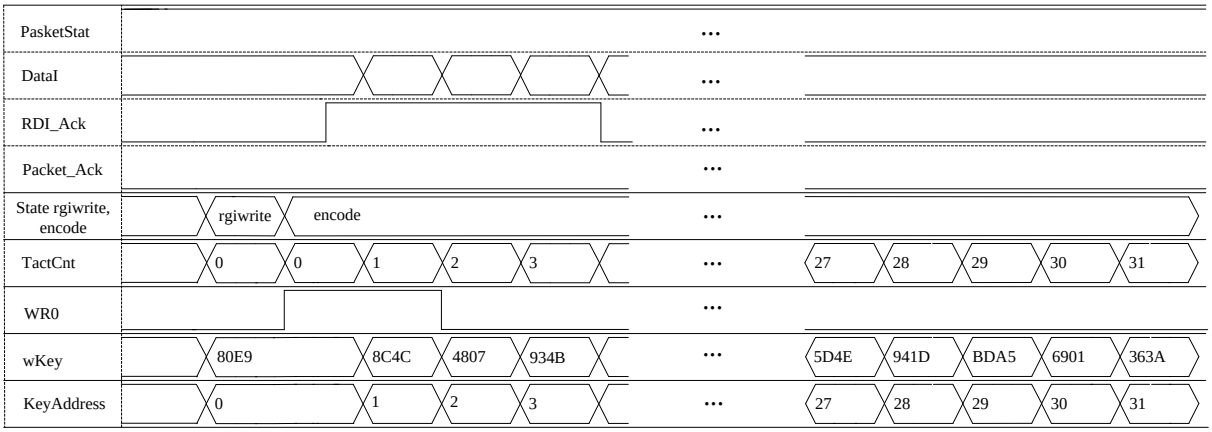


Рисунок 9 – Временная диаграмма шифрования пакетов данных

8.8 Чтение результата

Программа процессора должна вести постоянный мониторинг за регистром готовых пакетов, расположенному по адресу Register_Base_Address + 0x08. Чтение ненулевого значения, говорит об успешном шифровании/расшифровании очередного пакета данных. В этом случае, убедившись, что регистр статуса RFF в не содержит ‘1’, читаем из регистра данных RFF_Data_Address контрольное слово, формат которого аналогичен контрольному слову, создаваемому при записи. Выделив из него длину и проверив наличие указанного числа данных результата в RFF из того же регистра статуса, приступаем к чтению выходного пакета данных из RFF. Чтение происходит по словам.

9 Разработка функциональной схемы шифратора

9.1 Регистр ключа (РГК)

Исходя из криптографической схемы алгоритма шифрования «Уступ», в схеме шифратора можно выделить следующие основные блоки:

– РГК, представляющий из себя блок памяти RAM на 32 слова;

– РГИ – сдвигающий регистр на 4 слова;

– узлы подстановки и замены F1, F2 – ПЗУ на 256 байт.

Функциональная схема блока криптографической защиты информации изображена в приложении А.

РГК, представляет блок двух портовой памяти на 32 слова, изображенный на рисунке 10.

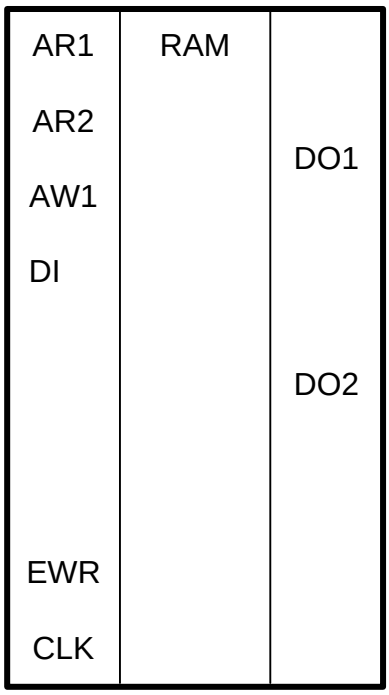


Рисунок 10 – Условное изображение РГК

Запись осуществляется по переднему фронту тактового сигнала CLK при наличии высокого уровня на ножке EWR. Чтение происходит асинхронно, независимо от разрешающих сигналов.

Описание блока на VHDL:

```
Entity RamR is
```

```
Port ( -- Адрес чтения и записи
   aAddrWr, aAddrRd1, aAddrRd2 : IN std_logic_vector(4 downto 0);
   -- Записываемые данные
   wWrData : IN word;
   -- Разрешение записи, тактирование
   bEWr, bClk : IN std_logic;
   -- Считываемые данные
   wReadData1, wReadData2 : OUT word );-- end port
End entity RamR;
```

9.2 Регистр информации (РГИ)

РГИ представляет сдвигающий регистр из 4-х 16-ти битных слов. Условное изображение РГИ показано на рисунке 11.

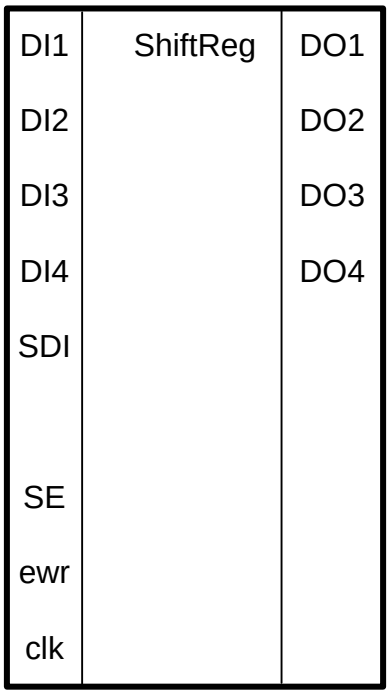


Рисунок 11 – Условное изображение РГИ

Данные в РГИ заносятся по переднему фронту сигнала CLK при наличии высокого уровня на входе EWR. Запись инициируется извне сигналом на замену синхроданных. При наличии высокого уровня на входе SE, с каждым очередным тактом происходит сдвиг на 16 разрядов в сторону

D1, на место D4 поступает слово-заместитель SDI. Чтение из регистра происходит асинхронно независимо от разрешающих сигналов.

Описание блока на VHDL:

```
entity ShiftRG is
generic (   -- Размер регистра
      constant RGSIZE : integer := 4 );
port (      -- Данные, записываемые в регистр
      wiA1, wiA2, wiA3, wiA4 : IN word;
      -- Слово, "заполнитель" при сдвиге
      wRepA : IN word;
      -- Бит, разрешающие запись
      bEWr : IN std_logic;
      -- Бит, разрешающий сдвиг
      bEShift : IN std_logic;

      -- Тактирование
      bClk : IN std_logic;
      -- Данные читаемые из регистра
      woA1, woA2, woA3, woA4 : OUT word  );-- end port
end ShiftRG;
```

Полное VHDL описание данного модуля можно найти в приложении Б.

9.3 Узлы подстановки и замены F1 и F2

Данные элементы представляют собой ПЗУ на 256 байт. Условное изображение ПЗУ показано на рисунках 12 и 13.

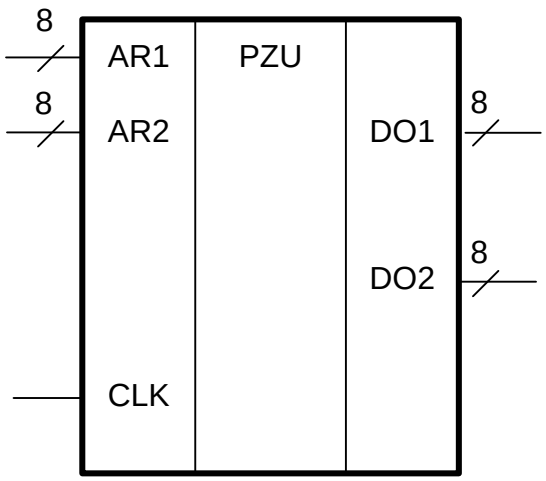


Рисунок 12 – Условное изображение ПЗУ

Входы AR1, AR2 являются адресами выборки значений из ПЗУ, выборка производится с каждым очередным возрастающим фронтом сигнала CLK. На узле подстановки F1 порядок байт на выходе меняется местами, на узле F2 слово с выхода ПЗУ попадает на регистр сдвига, представленный на рисунке 12.

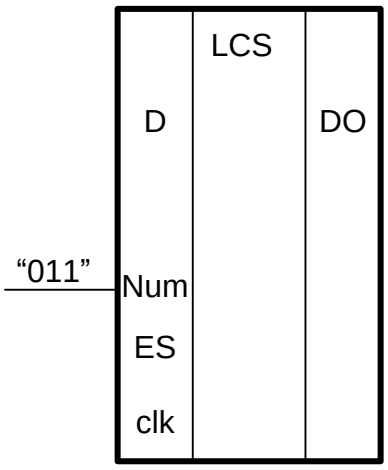


Рисунок 13 – Условное изображение ПЗУ

С каждым возрастающим фронтом сигнала CLK, при наличии высокого уровня на входе ES, выход DO выдает значение D, сдвинутое на 3 разряда влево.

Описание блока на VHDL:

```
-- Циклический сдвиг влево
```

```
function LCS ( wSrcWord : IN word; bLen : IN byte ) return word is
    variable wTmpWord, wTmpWord2 : word := X"0000";
    variable nLen : integer range 0 to 7;
Begin
  nLen := CONV_INTEGER( bLen ) - 1;
  -- Циклический сдвиг влево на 3 разряда
  wTmpWord2( nLen downto 0 ) := wSrcWord( 15 downto (15-nLen) );
  wTmpWord := SHL( wSrcWord, bLen ) or wTmpWord2;
   return wTmpWord;
End function;
```

10 Описание работы шифратора по функциональной схеме

10.1 Предварительное усложнение ключа

Работа шифратора инициируется прочтением из WFF контрольного слова, содержащего режим работы и длину пакета данных. Первым делом входные данные (ключ, синхроданные) переносятся из временного блока RAM в РГК и РГИ, затем необходимо выполнить предварительное усложнение ключа.

Алгоритм предварительного усложнения ключа запускается записью по адресу Ustup Base Address + 0x7C значения 0x00000005. На его выполнение требуется 64 такта работы системы. По завершении усложнения ключа по адресу Ustup Base Address + 0x78 записывается значение 0x00000001, говорящее об успешном завершении работы алгоритма. После чего, шифратор переходит в режим ожидания новых команд.

10.2 Выработка шифрогаммы

Запись по адресу Ustup Base Address + 0x7C команды 0x00000004. В данном режиме схема прорабатывает 32 такта.

По завершении работы, сформированная шифрогамма накладывается на первые 8 байт пакета открытой/закрытой информации, размещенных по адресу Ustup Base Address + 0x48. Результат гаммирования помещается по адресу Ustup Base Address + 0x58, после чего по адресу Ustup Base Address + 0x78 записывается значение 0x00000002 – свидетельство успешного завершения алгоритма и размещения результата шифрования/расшифрования в адресном пространстве шифратора. После чего, шифратор переходит в режим ожидания новых команд.

11 Разработка функциональной схемы устройства управления шифратором

Устройство управления шифратором принимает входные сигналы от процессора MicroBlaze. Основными модулями устройства управления являются блоки RAM, представляющие собой адресное пространство шифратора (блоки и карта адресов описаны выше), а также схема дешифрации команд и счетчик тактов.

Процессор записывает в память устройства управления ключ и синхроданные. Дешифратор команд анализирует данные записанные по адресу управляющего слова и передает команду в шифратор, удерживая ее необходимое число тактов. Полученный от шифратора результат размещается в адресном пространстве, откуда будет записан в RFF.

Далее будут представлены VHDL описания основных блоков[6].

VHDL представление устройства управления

```
entity comm_hu is
   port (
         -- Тактирование схемы
         clk : IN STD_LOGIC;
         -- Сигнал сброса
          reset : IN STD_LOGIC;
         -- Разрешение записи
         bEWr : IN STD_LOGIC;
         -- Разрешение чтения
         bERd : IN STD_LOGIC;
                     -- Адрес для занесения исходных данных
         aAddr : IN std_logic_vector (4 DOWNTO 0);
         -- Входные данные
         d_in : IN std_logic_vector (31 DOWNTO 0);
          -- Выходные данные
```

```
        d_out : OUT std_logic_vector (31 DOWNTO 0) );
end comm_hu;
```

VHDL представление шифратора

```
Entity coder_ustup is
   Port
   (
        -- Тактовый импульс
        bClk : IN std_logic;
              -- Новое заполнение для РГИ и РГК
 bNewDataReady : IN std_logic;
              -- Заменить РГИ
 bChangeRGI : IN std_logic;
              -- Старт "Алгоритма предварительного усложнения"
 bStartAPU : IN std_logic;
              -- Разрешение записи в РГК во время АПУ
bRGKEnWrite : IN std_logic;
              -- Сигнал на выработку новой шифрующей гаммы
bProcNewGamma : IN std_logic;
             -- i-слово ключа
wKey : WORD;
              -- Массив синхроданных
awSData : IN word_arr ( 4 downto 1 );
              -- Результирующая шифрующая гамма
        awCryptoGamma : OUT word_arr ( 4 downto 1 ) );-- end port
  End entity coder_ustup;
```

VHDL представление счетчика тактов

```
 -- Счетчик тактов
  TACT_COUNT : process( clk )
  Begin
       if ( clk'event and clk = '1' ) then
```

```
        if ( bChangeKey       = '1' or
         bHU_StartAPU     = '1' or
         bHU_ProcNewGamma = '1' ) then g_nTactCnt <= g_nTactCnt + 1;
    else g_nTactCnt <= 0; end if;
        if ( bCryptGammaReady = '1' or
         bImmitDataReady  = '1' ) then
      g_nTactCnt2 <= g_nTactCnt2 + 1;
    else
      g_nTactCnt2 <= 0;
    end if;
   end if;
End process TACT_COUNT;
```

12 Расчет быстродействия устройства

Быстродействие устройства криптографического преобразования информации складывается из быстродействия шифратора и преобразования пакетов переменной длины, равной размеру Ethernet кадра и измеряется мегабитами переданной зашифрованной информации в секунду.

Посчитаем максимальную пропускную способность УКПИ.

Компонент Ethernet работает на максимальной частоте 100 МГц. Разрядность шины данных интерфейса RGMII составляет 8 бит. Таким образом, за один такт можно передать 8 бит данных Умножим 8 бит на рабочую частоту 100 МГц и получаем максимальную скорость интерфейса Ethernet 1000 Мбит/сек.

Так как на преобразование пакета затрачивается время, в течение которого пакеты находятся во внутренней памяти, все внутренние шины имеются разрядность 16 бит. Это позволяет обработать и передать пакет за время, которое в 2 раза меньше времени приема пакета по интерфейсу RGMII. Таким образом, после приема первого пакета, и до получения второго, устройство способно полностью обработать и передать пакет, на следующий уровень системы. Это позволяет уменьшить задержки и избежать накапливания пакетов в памяти.

Шифрование происходит блоками по 64 бита. Расчет количества тактов, необходимый для подготовки и шифрования одного блока информации представлен в таблицах 11 и 12.

Таблица 11 – Длительность основных операций в тактах при подготовке шифратора

Название операции	Длительность (такт)
Запись ключа	32
Запись синхроданных	1
Прогон	64
Очистка статуса	1
Итого	98

Таблица 12 – Длительность основных операций в тактах в рабочем режиме

Название операции	Длительность (такт)
Замена синхроданных	1
Замена открытых данных	1
Выработка ШГ	32
Гаммирование	1
Запись результата	1
Очистка статуса	1
Итого	37

Частота работы ПЛИС – 100 МГц, следовательно, время, затрачиваемое на 1 такт равно 10 нс.

Время шифрования одного блока информации 37×10 нс = 370 нс

Время подготовки шифратора к работе 98×10нс = 0,98 мкс

Время шифрования 1 мегабита 370×128000 = 47,36 мс

Общее время шифрования 1 мегабита 378,88 мс + 0,98 мкс =378,88 мс;

Так как скорость системы зависит от самого медленного компонента, поэтому скорость всего устройства определяют по скорости блока криптографической защиты информации. Скорость шифрования равна 60 Мбит/с, что удовлетворяет требованиям технического задания.

ЗАКЛЮЧЕНИЕ

Результатом выполнения дипломного проекта стала разработка «Устройства криптографического преобразования информации» - реализация криптографического алгоритма «Уступ» на элементной базе Spartan 6 фирмы Xilinx. Спроектированная система позволяет вести приём IP пакетов, криптографическое преобразование, передачу зашифрованных пакетов по сетям общего пользования со скоростью до 100 Мбит/с как самостоятельно, так и в составе другой аппаратуры.

В процессе выполнения дипломного проекта был освоен цикл проектирования систем на базе ПЛИС: проектирование функциональных и структурных схем, описание их на языке VHDL, описание временных диаграмм, а так же составление алгоритмов.

Таким образом, все пункты технического задания были выполнены в полном объеме.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Шаньгин В.Ф. Информационная безопасность компьютерных систем и сетей: учебное пособие. – М.: ИД «ФОРУМ»: ИНФРА-М, 2011. – 416 с.

2 Аппаратно-программные СКЗИ КРИПТОН на защите АРМ. /Information Security/ Информационная безопасность - URL: http://www.itsec.ru/articles2/company/apparatn-prog_SKZI_KRIPTON_na_ zasch _ARM. №2, 2005.

3 Проценко Е.Д. Сравнительный анализ характеристик ПЛИС типа FPGA Xilinx семейств Spartan-6 и Virtex-6 (статья принята для публикации).

4 Зотов В. Особенности архитектуры нового поколения высокопроизводительных ПЛИС FPGA фирмы Xilinx серии Spartan-6 // Компоненты и технологии. URL: http://www.kit-e.ru/articles/plis/2009_09_62.php. № 8, 2009.

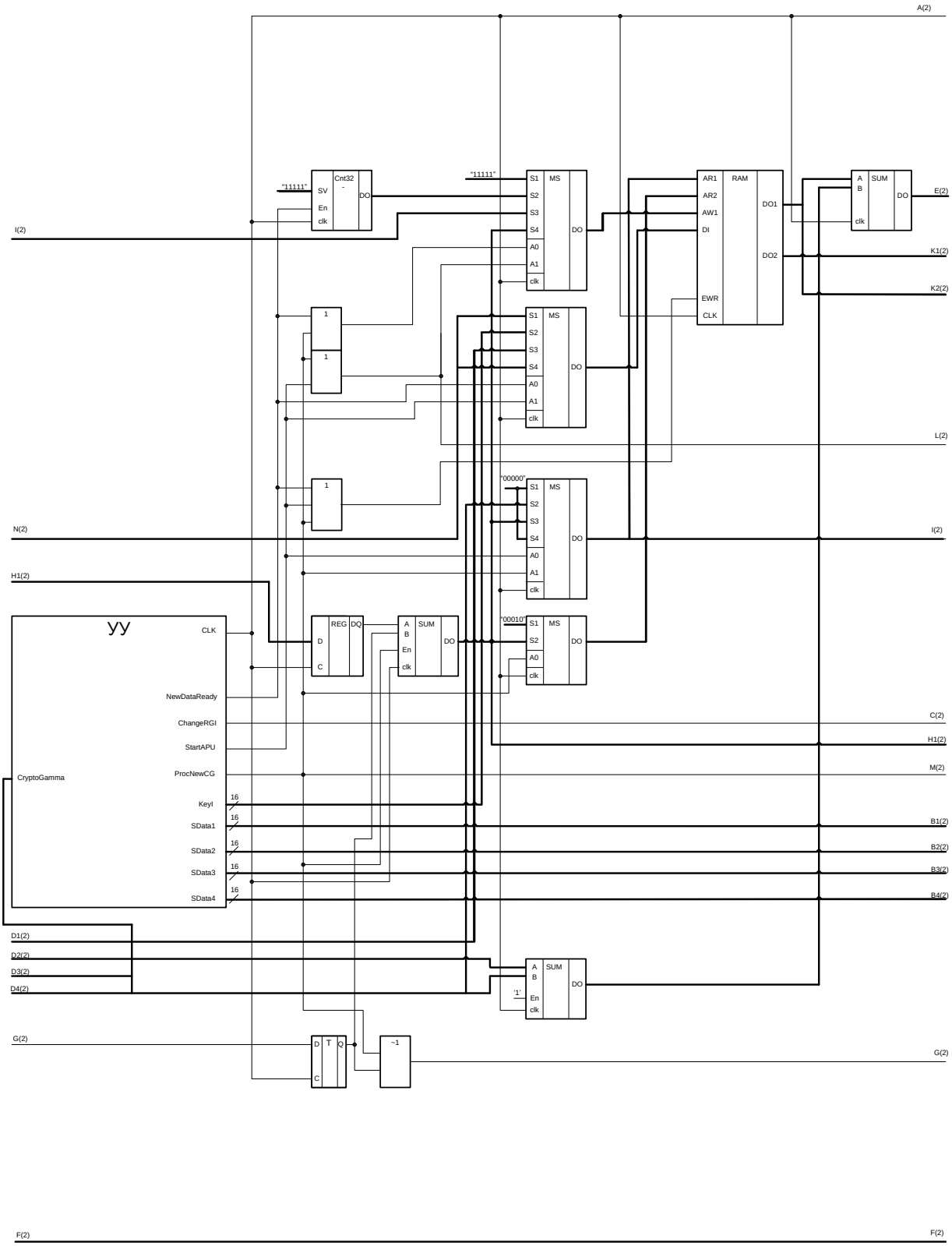
5 Построение систем на кристалле с подчиненным процессорным ядром MicroBlaze на ПЛИС фирмы Xilinx //Компоненты и технологии. URL: http://www.kit-e.ru/assets/files/pdf/2007_09_115.pdf. №9, 2007.

6 Суворова Е.А., Шейнин Ю.Е. Проектирование цифровых систем на VHDL: учебное пособие. – «БХВ-Петербург», 2003г. – 565 с.

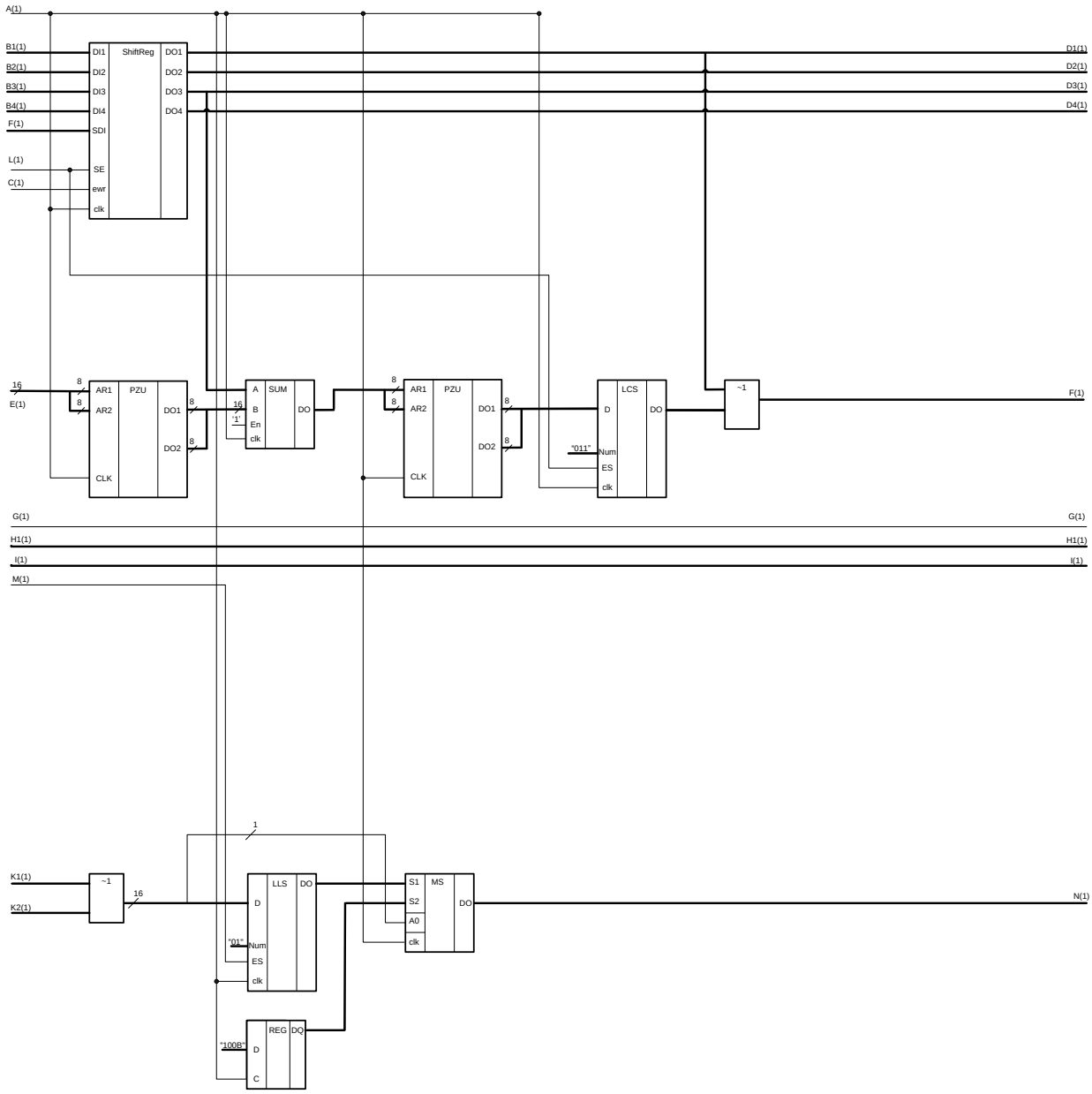
ПРИЛОЖЕНИЕ А

Схема электрическая функциональная блока криптографической защиты информации. Шифратор

ПГУ 1.100503.001 Э2



					ПГУ 1.100503.001 Э2		
Изм.	Лист	№ докум	Подпись	Дата			
Разраб		Проценко Е.Д.			Схема электрическая функциональная Блок криптографической защиты информации Шифратор	Литера	Лист 1 / Листов 2
Пров		Краснов В.А..					
Н. Контр.		Иванов А.П.				Гр. 11ПИ1	
Утв							



					ПГУ 1.100503.001 Э2	
Изм.	Лист	№ докум	Подпись	Дата		
Разраб		Проценко Е.Д.			Схема электрическая функциональная Блок криптографической защиты информации Шифратор	Литера: — / Лист: 2 / Листов: 2
Пров		Краснов В.А..				
Н. Контр.		Иванов А.П.				Гр. 11ПИ1
Утв						

ПРИЛОЖЕНИЕ Б

Листинг VHDL-описания схемы шифратора

```
-- \brief  Модуль осуществляет взаимодействие
--       между компонентом HardUstup и управляющим модулем ЗАС

_____________________________________________________________________

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
_____________________________________________________________________
library ustup_v2_00_a;
use ustup_v2_00_a.coder_ustup;
use ustup_v2_00_a.RamR;

use types.all;
_____________________________________________________________________
entity comm_hu is
     port (
           -- Тактирование схемы
           clk : IN STD_LOGIC;

           -- Сигнал сброса
           reset : IN STD_LOGIC;

           -- Разрешение записи
           bEWr : IN STD_LOGIC;

           -- Разрешение чтения
           bERd : IN STD_LOGIC;

                       -- Адрес для занесения исходных данных
           aAddr : IN std_logic_vector (4 DOWNTO 0);

           -- Входные данные
           d_in : IN std_logic_vector (31 DOWNTO 0);

            -- Выходные данные
           d_out : OUT std_logic_vector (31 DOWNTO 0) );
end comm_hu;

architecture Struct of comm_hu is

_____________Декларация_констант_____________________________________

-- Длина ключа
constant KEY_LEN : integer := 32;

-- Длина синхроданных
constant SDATA_LEN : integer := 4;

-- Максимальная длина открытой информации
constant OPEN_DATA_MAXLEN : integer := 4;

-- Максимальная длина открытой информации в памяти
```

```
constant OPEN_DATA_ADDRMAXLEN : integer := OPEN_DATA_MAXLEN/2;
_____________________________________________________________________
-- Адрес, по которому поступает управляющее слово
constant CONTROL_WORD_ADDR : integer := 31;

-- Адрес, по которому размещается открытая информация
constant OPEN_DATA_ADDR : integer := ( (KEY_LEN)/2 + (SDATA_LEN)/2 );

-- Свободные адреса, там будут размещены данные для формирования
иммитовставки
constant FREE_DATA_ADDR : integer := (OPEN_DATA_ADDR +
OPEN_DATA_ADDRMAXLEN);

-- Число свободных ячеек
constant FREE_DATA_LEN : integer := 2;

-- Адрес, по которому размещается результирующая гамма
constant RES_DATA_ADDR : integer := (FREE_DATA_ADDR + FREE_DATA_LEN);

-- Адресное пространство 2^ADDR_SPACE
constant ADDR_SPACE : integer := 5;
_____________________________________________________________________

-- Число тактов, необходимое для подготовки к АПУ
constant READY_APU_TACT_LEN : integer := 1;

-- Число тактов необходимое для работы алгоритма предварительного
усложнения
constant APU_WORK_MAXLEN : integer := 128 + READY_APU_TACT_LEN;

-- Число тактов, во время которых РГК работает только в режиме чтения
constant RGK_ONLY_READ : integer := 0;

-- Число тактов необходимое для работы алгоритма выработки гаммы шифра
constant PROC_GAMMA_LEN : integer := 64;

-- Число тактов необходимое для выработки части иммитовставки
constant IMMIT_DATAREADY_LEN : integer := 32;

-- Число тактов, необходимое для наложения штфрогаммы на открытую
информацию
constant CRYPT_LEN : integer := 2;

constant WRITE_RES_LEN : integer := 2;

_________________Декларация_сигналов_____________________________________

-- Данные, читаемые из ОЗУ
signal dwReadWord1, dwReadWord2 : std_logic_vector (31 DOWNTO 0);

-- i-е слово ключа
signal wHU_Key : WORD;

-- Синхроданные и результирующая ШГ
signal awHU_SData, awHU_ResGamma : word_arr( SDATA_LEN downto 1 );
```

```
-- Входные данные для ОЗУ
signal dwRamDataWr : std_logic_vector (31 DOWNTO 0);
signal  aRamAddrWr,  aRAMAddrRd1,  aRAMAddrRd2  :  std_logic_vector(4
downto 0 ) := "00000";

-- Указатели на адреса в ОЗУ УУ
signal aKeyPtr   : std_logic_vector(4 downto 0 ) := "00010";
signal   aOpenPtr      :   std_logic_vector(4   downto   0   )   :=
CONV_STD_LOGIC_VECTOR( OPEN_DATA_ADDR, ADDR_SPACE );
signal   aImmitPtr   :   std_logic_vector(4   downto   0   )   :=
CONV_STD_LOGIC_VECTOR( FREE_DATA_ADDR, ADDR_SPACE );
signal   aResPtr         :   std_logic_vector(4   downto   0   )   :=
CONV_STD_LOGIC_VECTOR( RES_DATA_ADDR,  ADDR_SPACE );

-- Разрешение записи в ОЗУ
signal bERamWr : std_logic := '0';

-- Флаги управления работой
signal bChangeKey, bChangeSData,
       bHU_ProcNewGamma, bHU_StartAPU, bHU_RGKWrite,
       bCryptGammaReady,  bImmitDataReady,  bGetImmit  :  std_logic  :=
'0';

-- Зашифрованные данные
signal dwCryptData, dwImmitData : std_logic_vector (31 DOWNTO 0);

-- Триггер для передачи за 2 такта
signal bParityTact : std_logic := '0';

signal nAddr : integer range 0 to KEY_LEN;

-- Флаги, указывающие на завершение работы алгоритмов усложнения и
выработки шифрогаммы
-- Флаг прочтения регистра статуса
signal bSetKeyComplete, bAPUComplete, bCryptFinish : std_logic := '0';

_____________________Управление_счетчиком_тактов_____________________
-- Счетчик тактов
signal g_nTactCnt, g_nTactCnt2 : integer range 0 to 255;
signal nRGKOnlyReadLen : integer range 0 to 127 := RGK_ONLY_READ;
_____________________________________________________________________
begin

     -- Два блока оперативной памяти для хранения данных,
     -- поступающих от основного процессора
  RAM_GEN : FOR i IN 0 TO 1 GENERATE

     RAM_I : entity ustup_v2_00_a.RamR
     port map (
                 bClk => clk,
                 bEWr => bERamWr,

                        -- Адреса чтения и записи
                    aAddrWr  => aRamAddrWr,
```

```
                      aAddrRd1 => aRamAddrRd1,
                     aAddrRd2 => aRamAddrRd2,

                      wWrData       =>  dwRamDataWr(16*i  +  15  downto
    16*i),
                      wReadData1   =>   dwReadWord1(16*i   +   15   downto
    16*i),
                      wReadData2 => dwReadWord2(16*i + 15 downto 16*i)
    );

  END GENERATE RAM_GEN;
```

```
    -- Шифратор, осуществляющий выработку ШГ по алгоритму "Уступ"
  HUstup_I : entity ustup_v2_00_a.coder_ustup
  port map (
             bClk          => clk,
             bNewDataReady => bChangeKey,
             bChangeRGI    => bChangeSData,
             bStartAPU     => bHU_StartAPU,
             bRGKEnWrite   => bHU_RGKWrite,
             bProcNewGamma => bHU_ProcNewGamma,
             wKey          => wHU_Key,
             awSData       => awHU_SData,

             -- Выходные данные
             awCryptoGamma => awHU_ResGamma );
```

```
  -- Счетчик тактов
  TACT_COUNT : process( clk )
  Begin

    if ( clk'event and clk = '1' ) then

      if ( bChangeKey       = '1' or
           bHU_StartAPU     = '1' or
           bHU_ProcNewGamma = '1' ) then g_nTactCnt <= g_nTactCnt + 1;
      else g_nTactCnt <= 0; end if;

      if ( bCryptGammaReady = '1' or
           bImmitDataReady  = '1' ) then
        g_nTactCnt2 <= g_nTactCnt2 + 1;
      else
        g_nTactCnt2 <= 0;
      end if;

    end if;

  End process TACT_COUNT;
```

```
-- Управление шифрованием
CONTROL : process ( clk )
variable nA : integer range 0 to 128 := 0;
Begin
  if ( clk'event and clk = '1' ) then
```

```
    if ( nAddr = CONTROL_WORD_ADDR and bEWr = '1' ) then

      case d_in( 3 downto 0 ) is
        -- Передать полученные данные в шифратор, начать шифрование
        when "0001" => bChangeKey <= '1';

        -- Заменить синхроданные на новые
        when "0010" => bChangeSData <= '1';

        when "0011" => nRGKOnlyReadLen <= CONV_INTEGER( d_in(7 downto
4) );

        -- Выработать новую ШГ
        when "0100" => bHU_ProcNewGamma <= '1'; bGetImmit <= '1';

        -- Выполнить предварительное усложнение
        when "0101" => bHU_StartAPU <= '1';

        when others =>
                 nA := 0;

      end case; -- d_in( 3 downto 0 )

    end if; -- ( aAddr = CONTROL_WORD_ADDR )

    -- Определяет длительность передачи синхроданных в шифратор
    -- Синхроданные передаются за 1 такт
    if ( bChangeSData = '1' ) then
      bChangeSData <= '0';
    end if; -- if ( bChangeSData = '1' )

    -- Определяет длительность передачи ключа
    if ( (bChangeKey = '1' and g_nTactCnt = (KEY_LEN-1) ) or reset =
'1' ) then
      bChangeKey <= '0';

      -- Ключ установлен
      if ( reset /= '1' ) then bSetKeyComplete <= '1'; end if;
    end if;

    -- Сбросить флаг установки ключа
    if ( bSetKeyComplete = '1' ) then
      bSetKeyComplete <= '0';
    end if;

    -- Определяет длительность работы АПУ в режиме без записи
    if ( bHU_StartAPU = '1' and g_nTactCnt = nRGKOnlyReadLen ) then
      bHU_RGKWrite <= '1';
    end if;

    -- Определяет длительность работы АПУ
    if ( (bHU_StartAPU = '1' and g_nTactCnt = APU_WORK_MAXLEN) or
reset = '1' ) then
```

```
      bHU_StartAPU <= '0';

      bHU_RGKWrite <= '0';

      -- Предварительное усложнение завершено,
      -- установить соответствующий бит в регистре статуса
      if ( reset /= '1' ) then bAPUComplete <= '1'; end if;

    end if;

    -- Длительность удержания флага об окончании АПУ
    if ( bAPUComplete = '1' ) then
      bAPUComplete <= '0';
    end if;

    -- Определяет длительность работы получения части иммитовставки
    if ( (bGetImmit = '1' and g_nTactCnt = IMMIT_DATAREADY_LEN) or
reset = '1' ) then

      bGetImmit <= '0';

      -- Команда на получение части иммитовставки
      if ( reset /= '1' ) then
        bImmitDataReady <= '1';
      end if;

    end if;

    -- Определяет длительность работы АВГШ
    if ( (bHU_ProcNewGamma = '1' and g_nTactCnt = (PROC_GAMMA_LEN-1))
or reset = '1' ) then

      -- Команда на получение гаммы шифра
      bHU_ProcNewGamma <= '0';

      -- Выработка ШГ завершена
      if ( reset /= '1' ) then
        bCryptGammaReady <= '1';
      end if;

    end if;

    -- Определяет длительность записи части иммитовставки
    if ( bImmitDataReady = '1' and g_nTactCnt2 = (CRYPT_LEN-1) ) then
      -- Флаг, указывающий на готовность части иммитовставки
      bImmitDataReady <= '0';
    end if;

    -- Определяет длительность гаммирования данных
    if ( (bCryptGammaReady = '1' and g_nTactCnt2 = (CRYPT_LEN-1)) or
reset = '1' ) then
      bCryptGammaReady <= '0';

      -- Данные зашифрованы, выставить уведомление в регистре статуса
      if ( reset /= '1' ) then bCryptFinish <= '1'; end if;
```

```
    end if;

    -- Сигнализация о завершении шифрования (изменение регистра
статуса)
    if ( bCryptFinish = '1' ) then
      bCryptFinish <= '0';
    end if;

  end if; -- ( clk'event and clk = '1' )
End process CONTROL;

_____________________Вычисление_адреса_записи_результата_______________

ADDR_CALC : process( clk )
variable bResPtr : std_logic := '0';
Begin
  if ( clk'event and clk = '1' ) then

____________________Шифрование_данных_и_запись_результата_______________

    -- Сформирована часть иммитовставки
    if ( bImmitDataReady = '1' ) then

      if  (  aImmitPtr  <  CONV_STD_LOGIC_VECTOR(  (FREE_DATA_ADDR  +
FREE_DATA_LEN - 1), 5 ) ) then
        aImmitPtr <= aImmitPtr + 1;
      else
        bResPtr := '1';
      end if;
    end if; -- if ( bImmitDataReady = '1' ) then

    -- Сформирована ШГ
    if ( bCryptGammaReady = '1' ) then

      -- Сместить указатель на открытые данные
      if  (  aOpenPtr  <  CONV_STD_LOGIC_VECTOR(  OPEN_DATA_ADDR  +
(OPEN_DATA_MAXLEN/2-1), 5 ) ) then
        aOpenPtr <= aOpenPtr + 1;
      else
        bResPtr := '1';
      end if;

      -- Сместить указатель на результирующую ШГ
      if  (  aResPtr  <  CONV_STD_LOGIC_VECTOR(  (RES_DATA_ADDR  +
(OPEN_DATA_MAXLEN/2-1)), 5 ) ) then
        aResPtr <= aResPtr + 1;
      else
        bResPtr := '1';
      end if;

    end if; -- if ( bCryptGammaReady = '1' )

    -- В случае сброса или добавления новых данных, сбросить указатели
    if ( reset = '1' or bChangeKey = '1' or bResPtr = '1' ) then
```

```
      aOpenPtr   <= CONV_STD_LOGIC_VECTOR( OPEN_DATA_ADDR, ADDR_SPACE
);
      aImmitPtr  <= CONV_STD_LOGIC_VECTOR( FREE_DATA_ADDR, ADDR_SPACE
);
      aResPtr    <= CONV_STD_LOGIC_VECTOR( RES_DATA_ADDR,  ADDR_SPACE
);
      bResPtr := '0';
    end if;

  End if; -- if ( clk'event and clk = '1' )
End process ADDR_CALC;

_____________________Передача_за_два_такта_____________________________

TRNSM_TACT_CNT : process( clk )
Begin

  if ( clk'event and clk = '1' ) then

    -- Работа с данными, передаваемыми за 2 такта
    if ( bCryptGammaReady = '1' or bChangeKey = '1' or bImmitDataReady
= '1' ) then

      bParityTact <= not bParityTact;

    else bParityTact <= '0'; end if; -- if ( bCryptGammaReady = '1' or
bTrnsmKey = '1' )

  end if; -- if ( clk'event and clk = '1' )

End process TRNSM_TACT_CNT;
_____________________________________________________________________

-- Адрес чтения ключа
KEY_ADDR_CALC : process( clk )
Begin

  if ( clk'event and clk = '1' ) then

    -- Указатель на слово передаваемого в шифратор ключа
    if ( bChangeKey = '1' ) then

      aKeyPtr <= aKeyPtr + bParityTact;

    else aKeyPtr <= "00010"; end if; -- if ( bChangeKey = '1' )

  end if; -- if ( clk'event and clk = '1' )

End process KEY_ADDR_CALC;

_____________________Управление_ОЗУ_____________________________
-- Запись разрешена по сигналу основного процессора, либо для
сохранения результата
-- Запись в регистр статуса разрешена по сигналам щавершения работы
АПУ, АВГШ, установки ключа
```

```
bERamWr <= '0' when ( reset = '1' ) else
           bEWr or bAPUComplete or
           bCryptGammaReady     or
           bSetKeyComplete      or
           bCryptFinish         or bImmitDataReady;

-- Адрес записи и входные данные:
--    30 - адрес регистра статуса
aRamAddrWr <= aAddr     when ( bEWr = '1' )             else
              aResPtr   when ( bCryptGammaReady = '1' ) else
              aImmitPtr when ( bImmitDataReady  = '1' ) else
              "11110";

-- Данные для записи в ОЗУ:
--    Запись извне по сигналу разрешения записи,
--    Запись результата шифрования по сигналу готовности криптогаммы
--     Запись значения в регистр статуса по сигналам завершения работы
алгоритмов шифр-я
dwRamDataWr <= d_in        when ( bEWr             = '1' ) else
               dwImmitData when ( bImmitDataReady  = '1' ) else
               X"00000001" when ( bAPUComplete     = '1' ) else
               X"00000003" when ( bSetKeyComplete  = '1' ) else
               X"00000002" when ( bCryptFinish     = '1' ) else
               dwCryptData;

-- Адрес чтения из ОЗУ. Чтение открытой информации, синхроданных
aRamAddrRd1 <= aOpenPtr when ( bCryptGammaReady = '1' ) else
               aKeyPtr  when ( bChangeKey = '1' ) else
               "00000";

-- Чтение ключа, чтение внешним процессором
-- Адрес чтения из ОЗУ
aRamAddrRd2 <= aAddr when ( bERd = '1' ) else
               "00001";

____________________Передача_ключа_____________________________________
-- Данные, передаваемые в шифратор
wHU_Key <= dwReadWord1(15 downto 0)   when ( bChangeKey = '1' and
bParityTact = '0' ) else
           dwReadWord1(31 downto 16);

_________________Передача_синхроданных_____________________________
awHU_SData(1) <= dwReadWord1(31 downto 16);
awHU_SData(2) <= dwReadWord1(15 downto  0);
awHU_SData(3) <= dwReadWord2(31 downto 16);
awHU_SData(4) <= dwReadWord2(15 downto  0);

___________________Управление_шифрованием_____________________________

-- Шифрование открытой информации
dwCryptData <= GetDWORD( awHU_ResGamma(1), awHU_ResGamma(2)   ) xor
dwReadWord1 when ( bCryptGammaReady = '1' and bParityTact = '0' ) else
               GetDWORD( awHU_ResGamma(3), awHU_ResGamma(4)   ) xor
dwReadWord1;
```

```
-- Часть иммитовставки
dwImmitData <= GetDWORD( awHU_ResGamma(1), awHU_ResGamma(2) ) when (
bImmitDataReady = '1' and bParityTact = '0' )  else
               GetDWORD( awHU_ResGamma(2), awHU_ResGamma(3) );
```

```
-- Чтение данных внешним процессором
d_out <= dwReadWord2 when ( bERd = '1' ) else
         X"00000000";

nAddr <= CONV_INTEGER(aAddr);
```

```
end architecture Struct;
```

coder_ustup.vhd

```
-- \brief  Основной модуль программы
```

```
library IEEE;

use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
```

```
library ustup_v2_00_a;
use ustup_v2_00_a.RamR;
use ustup_v2_00_a.ShiftRG;
```

```
-- Узел замены
use NodeChange.all;

-- Библиотека деклараций используемых типов данных
use types.all;
```

```
Entity coder_ustup is
      Port
      (
            -- Тактовый импульс
            bClk : IN std_logic;

                -- Новое заполнение для РГИ и РГК
                bNewDataReady : IN std_logic;

                -- Заменить РГИ
                bChangeRGI : IN std_logic;

                -- Старт "Алгоритма предварительного усложнения"
                bStartAPU : IN std_logic;

                -- Разрешение записи в РГК во время АПУ
                bRGKEnWrite : IN std_logic;
```

```
                -- Сигнал на выработку новой шифрующей гаммы
                bProcNewGamma : IN std_logic;

                -- i-слово ключа
                wKey : WORD;

                -- Массив синхроданных
                awSData : IN word_arr ( 4 downto 1 );

            -- Результирующая шифрующая гамма
            awCryptoGamma : OUT word_arr ( 4 downto 1 ) );-- end port

  End entity coder_ustup;

Architecture Struct of coder_ustup is

_____________________Используемые_компоненты_____________________________

-- ОЗУ на 32 слова
  component RamR is

      Port ( -- Адрес чтения и записи
             aAddrWr, aAddrRd1, aAddrRd2 : IN std_logic_vector(4 downto
0);

             -- Записываемые данные

                  wWrData : IN word;

             -- Разрешение записи, тактирование
             bEWr, bClk : IN std_logic;

             -- Считываемые данные
             wReadData1, wReadData2 : OUT word );-- end port
  End component RamR;
_____________________________________________________________________

component  ShiftRG
generic (  -- Размер регистра
           constant RGSIZE : integer := 4 );

port (     -- Данные, записываемые в регистр
     wiA1, wiA2, wiA3, wiA4 : IN word;

     -- Слово, "заполнитель" при сдвиге
     wRepA : IN word;

     -- Бит, разрешающие запись
     bEWr : IN std_logic;

     -- Бит, разрешающий сдвиг
     bEShift : IN std_logic;

     -- Тактирование
     bClk : IN std_logic;
```

```
      -- Данные читаемые из регистра
      woA1, woA2, woA3, woA4 : OUT word     );-- end port

end component ShiftRG;
_____________________________Декларация_сигналов_____________________________
constant KEY_SIZE : integer := 32;

 -- Тип массив, для размещения значений подстановки
type TABLE_256 is array (0 to 255) of integer range 0 to 255;

-- Значения подстановки F ( для служебного использования )
signal aNodeData : TABLE_256 := ( );

_______________________________Управление_РГИ_______________________________
-- Выходы РГИ
signal wSR_Ao1, wSR_Ao2, wSR_Ao3, wSR_Ao4 : WORD;

signal bSR_EShift : std_logic := '0';
signal wSR_RepA : word;

_______________________________Управление_РГК_______________________________
-- Адреса записи и чтения РГК
signal aRM_AddrRd1 : std_logic_vector (4 DOWNTO 0) := "00000";
signal aRM_AddrRd2 : std_logic_vector (4 DOWNTO 0) := "00010";
signal aRM_AddrWr  : std_logic_vector (4 DOWNTO 0) := "00000";

-- Разрешение записи в РГК
signal bRM_EWr : std_logic := '0';

-- Слово, читаемое из РГК
signal wRM_ReadData1, wRM_ReadData2, wRM_WriteData, wXorData : WORD;

-- Задержка
signal wDelayFF : word;

signal g_bXor : std_logic := '1';
signal bDelay, bDelay2 : std_logic := '0';
signal bRGKEWr : std_logic := '0';
_____________________________________________________________________________

Begin

  -- Описание РГИ1
     RGI1_I : entity ustup_v2_00_a.ShiftRG
     Port map (
                 wiA1    => awSData(1),
                 wiA2    => awSData(2),
                 wiA3    => awSData(3),
                 wiA4    => awSData(4),
                 bClk    => bClk,
                 wRepA   => wSR_RepA,
                 bEWr    => bChangeRGI,
                 bEShift => bSR_EShift,
```

```
                    -- Выход
                    woA1 => wSR_Ao1,
                    woA2 => wSR_Ao2,
                    woA3 => wSR_Ao3,
                    woA4 => wSR_Ao4 );

-- Описание РГК1
RGK1_I : entity ustup_v2_00_a.RamR
Port map (
      wWrData  => wRM_WriteData,
      bClk     => bClk,
      aAddrRd1 => aRM_AddrRd1,
      aAddrRd2 => aRM_AddrRd2,
      aAddrWr  => aRM_AddrWr,
      bEWr     => bRM_EWr,

 -- Выход
      wReadData1 => wRM_ReadData1,
            wReadData2 => wRM_ReadData2 );

_____________________Описание_процессов_____________________________
  -- Задержка для подготовки данных
  DELAY_CNT : process( bClk )
  Begin

    if ( bClk'event and bClk = '1' ) then

      -- Поступление новых данных, предварительное усложнение или
выработка ШГ
      if ( bNewDataReady = '1' or bStartAPU = '1' or bProcNewGamma =
'1' ) then
        bDelay <= '1';
      else
        bDelay <= '0';
      end if; --        if ( bNewDataReady = '1' or bStartAPU = '1' or
bProcNewGamma = '1' ) then

      if ( bStartAPU = '1' and bDelay = '1' ) then
        bDelay2 <= '1';
      else
        bDelay2 <= '0';
      end if; -- if ( bStartAPU = '1' and bDelay = '1' ) then

    end if;

  End process DELAY_CNT;

_______________Вычисление_адресов_чтения_и_записи_ОЗУ_____________

-- Управление РГК

  -- Управление разрешением на запись в РГК
  -- Адрес записи в ОЗУ: при записи ключа - уменьшается с 31 до 0,
  -- Во время работы АПУ = адресу чтения
  -- Данные для записи в ОЗУ ( ключ - при новом заполнении,
```

```
  --                              1-е слово РГИ во время работы АПУ )
  RAM_WRITE_CNTRL : process ( bClk )
  Begin

    if ( bClk'event and bClk = '1' ) then
      --  Начальное заполнение РГК (ключ записывается с последнего
адреса)
      -- Используется для формирования адреса записи в ОЗУ
      if ( bNewDataReady = '1' ) then
        aRM_AddrWr <= aRM_AddrWr - '1';

      -- Выработка ШГ
      -- Указатели на адресы РГК, пермещаются, иммитируя сдвиг ОЗУ
      elsif ( bProcNewGamma = '1' ) then

        if ( bDelay = '1' ) then aRM_AddrWr <= aRM_AddrWr + g_bXor;
end if;
        bRGKEWr <= bRGKEWr xor '1';

      -- Разрешить запись в РГК, спустя указанное число тактов
      -- Адрес записи = адресу чтения из РГК
      -- Данные для записи - слово, выдвигаемое из РГИ
      elsif ( bStartAPU = '1' and bDelay = '1' ) then
--        aRM_AddrWr <= aRM_AddrRd1(4 downto 0);
        aRM_AddrWr <= aRM_AddrRd1;

        bRGKEWr <= bRGKEWr xor '1';

      else
        bRGKEWr    <= '0';
        aRM_AddrWr <= "11111";
      end if; -- if ( bNewDataReady = '1' )

      -- Получение слова, умножаемого в поле GF(2^16)
      wXorData <= MulAlfa( wRM_ReadData1 xor wRM_ReadData2 );

    End if; -- ( bClk'event and bClk = '1' )

  End process RAM_WRITE_CNTRL;
```

```
  -- Управление сдвигом
  ENABLE_SHIFT : process ( bClk )
  Begin

    if ( bClk'event and bClk = '1' ) then

      if ( (bStartAPU = '1' and bDelay2 = '1') or bProcNewGamma = '1'
) then
        -- Разрешение сдвига РГИ
        bSR_EShift <= bSR_EShift xor '1';

      else bSR_EShift <= '0'; end if;

    end if;
```

```
  End process ENABLE_SHIFT;
```

```
  -- Узел формирования слова-вдвигателя в РГИ
  NODE1 : process( bClk )
    variable wTmpWord, wTmpWord2, wTmpWord3, wRepA2 : word;
    variable wA4, wA2 : word;
  Begin
    if ( bClk'event and bClk = '1' ) then

      if ( (bStartAPU = '1' and bDelay2 = '1') or bProcNewGamma = '1'
) then
          wA2 := wSR_Ao3;
          wA4 := wSR_RepA;
       else
          wA2 := wSR_Ao2;
          wA4 := wSR_Ao4;
       end if;

       wTmpWord := wA2 + wA4 + wRM_ReadData1;

      -- Подстановка F1
      wDelayFF(15downto                          8)                    <=
CONV_STD_LOGIC_VECTOR(aNodeData(CONV_INTEGER(wTmpWord(     7     downto
0))),8);
      wDelayFF(           7           downto           0)           <=
CONV_STD_LOGIC_VECTOR(aNodeData(CONV_INTEGER(wTmpWord(15         downto
8))),8);

      wTmpWord2 := wDelayFF + wSR_Ao3;

      -- Подстановка F2
      wTmpWord3(           7           downto           0)           :=
CONV_STD_LOGIC_VECTOR(aNodeData(CONV_INTEGER(wTmpWord2(     7     downto
0))),8);
      wTmpWord3(15              downto              8)              :=
CONV_STD_LOGIC_VECTOR(aNodeData(CONV_INTEGER(wTmpWord2(15         downto
8))),8);

      -- Слово, вдвигаемое в РГИ
      wRepA2   := LCS(wTmpWord3, X"03") xor wSR_Ao1;
      wSR_RepA <= wRepA2;

      -- Адрес чтения из РГК
      -- Во время АПУ - младшие 5 разрядов A4
      if ( bStartAPU = '1' ) then

        if   ( bDelay2 = '0' ) then aRM_AddrRd1 <= wSR_Ao4( 4 downto 0
);
        else aRM_AddrRd1 <= wRepA2( 4 downto 0 ); end if;

      elsif ( bProcNewGamma = '1' ) then

        g_bXor <= g_bXor xor '1';
```

```
        aRM_AddrRd1 <= aRM_AddrRd1 + g_bXor;
        aRM_AddrRd2 <= aRM_AddrRd2 + g_bXor;

      else
        g_bXor      <= '1';
        aRM_AddrRd1 <= "00000";
        aRM_AddrRd2 <= "00010";
      end if;

    end if; -- if ( bClk'event and bClk = '1' )

  End process NODE1;

_____________________Цепь_замен_____________________________________________

  bRM_EWr <= '1'     when ( bNewDataReady = '1' ) else
             bRGKEWr when ( bRGKEnWrite = '1' or bProcNewGamma = '1' )
else
             '0';

  wRM_WriteData <= wKey      when ( bNewDataReady = '1' ) else
                   wSR_Ao1   when ( bStartAPU = '1' )     else
                   wXorData;

_________________Работа_РГИ_____________________________________________

  -- Выдача результирующей гаммы
  awCryptoGamma(1) <= wSR_Ao1;
  awCryptoGamma(2) <= wSR_Ao2;
  awCryptoGamma(3) <= wSR_Ao3;
  awCryptoGamma(4) <= wSR_Ao4;
_____________________________________________________________________________

End architecture Struct;
_____________________________end_of_coder_ustup.vhd_________________________

_____________________________NodeChange.vhd_________________________________

-- \brief  Реализация функций, осуществляющих подстановки F1, F2


_____________________________________________________________________________
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

library work;
use work.all;

-- Библиотека деклараций используемых типов данных
use types.all;
_____________________________________________________________________________
Package    NodeChange is

      -- Циклический сдвиг влево
```

```
      function LCS ( wSrcWord : IN word; bLen : IN byte ) return word;

      -- Умножение в поле GF(2^16)
      function MulAlfa ( wSrcWord : IN word ) return word;

end package NodeChange;
```

_____

```
package    body NodeChange is
```

_____

```
-- Циклический сдвиг влево
function LCS ( wSrcWord : IN word; bLen : IN byte ) return word is
    variable wTmpWord, wTmpWord2 : word := X"0000";
    variable nLen : integer range 0 to 7;
  Begin

    nLen := CONV_INTEGER( bLen ) - 1;

    -- Циклический сдвиг влево на 3 разряда
     wTmpWord2( nLen downto 0 ) := wSrcWord( 15 downto (15-nLen) );
     wTmpWord := SHL( wSrcWord, bLen ) or wTmpWord2;

    return wTmpWord;
  End function;
```

_____

```
-- Умножение в поле GF(2^16)
function MulAlfa ( wSrcWord : word ) return word is
-- Примитивный элемент поля GF (0x100B)
constant ALFA : WORD := X"100B";
-- Переменная для хранения старшего разряда умножаемого слова
variable bY15 : std_logic;
variable wResWord : word;
Begin
      -- Получить старший разряд слова
      bY15 := wSrcWord(15);

      -- Сдвиг влево на 1 разряд
      wResWord := SHL( wSrcWord, X"1" );

      if ( bY15 = '1' ) then
            wResWord := wResWord xor ALFA;
      end if;

      return wResWord;
End function MulAlfa;
```

_____

```
end package     body NodeChange;
```

_____end_of_NodeChange.vhd_____

_____RamR.vhd_____

```
-- \brief  Реализация модуля ОЗУ на 64 слова
--  \remark  Запись  осуществляется  по  переднему  фронту  тактового
импульса (bClk)
--                    при наличии высокого уровня на входе разрешения
bEWr
_____________________________________________________________________

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

library work;
use work.all;

use types.all;
--
_____________________________________________________________________

-- ОЗУ на 32 слова
Entity RamR is

     Port ( -- Адрес чтения и записи
            aAddrWr, aAddrRd1, aAddrRd2 : IN std_logic_vector(4 downto
0);

            -- Записываемые данные
                 wWrData : IN word;

            -- Разрешение записи, тактирование
            bEWr, bClk : IN std_logic;

            -- Считываемые данные
            wReadData1, wReadData2 : OUT word );-- end port

End entity RamR;
_____________________________________________________________________
Architecture Struct of RamR is

  -- Размер     ОЗУ
  constant RAMSIZE : integer := 32;

-- Определение типа для хранимых данных
type Data is array ( 0 to RAMSIZE - 1 ) of word;

 -- Массив слов, хранящихся в ОЗУ
signal awRAMData : Data;

Begin
_____________________________________________________________________
-- Запись данных по адресу
WRITE_DATA : process ( bClk )
Begin
     -- Принят тактовый импульс
     if ( bClk = '1' and bClk'event ) then
```

```
      -- Если установлено разрешение на запись...
      if ( bEWr = '1' ) then
            -- Записать в память данные по адресу aAddrWr
            awRAMData( CONV_INTEGER(aAddrWr) ) <= wWrData;
      end if;

      end if;-- ( bClk = '1' and bClk'event )
      end process WRITE_DATA;
--
_____________________________________________________________________
___-

-- Чтение данных из ОЗУ
wReadData1 <= awRAMData( CONV_INTEGER(aAddrRd1) );
wReadData2 <= awRAMData( CONV_INTEGER(aAddrRd2) );

end architecture Struct;

_____________________end_of_RamR.vhd_____________________________
_____________________ShiftRG.vhd_____________________________

-- \brief   Реализация регистра сдвига вправо, емкостью 4 двухбайтных
слова
-- \date    31.07.2007
-- \remark Работа регистра:
--     bEWr       - разрешение записи в регистр
--     bEShift - разрешение сдвига (сдвиг вправо, в 4 слово заносится
wRepA)
-- \note    При   одновременной   установке   нескольких   управляющих
сигналов,
--      приоритет операций: запись - сдвиг.

--
_____________________________________________________________________
______

library IEEE;
use IEEE.STD_LOGIC_1164.all;

library work;
use work.all;

-- Библиотека деклараций используемых типов данных
use types.all;


_____________________________________________________________________

entity ShiftRG is
      generic (  -- Размер регистра
                 constant RGSIZE : integer := 4 );

      port (     -- Данные, записываемые в регистр
            wiA1, wiA2, wiA3, wiA4 : IN word;

            -- Слово, "заполнитель" при сдвиге
```

```
            wRepA : IN word;

            -- Бит, разрешающие запись
            bEWr : IN std_logic;

            -- Бит, разрешающий сдвиг
            bEShift : IN std_logic;

            -- Тактирование
            bClk : IN std_logic;

            -- Данные читаемые из регистра
            woA1, woA2, woA3, woA4 : OUT word     );-- end port

end ShiftRG;

architecture Struct of ShiftRG is

-- Определение типа для хранимых данных
type Data is array ( 0 to RGSIZE-1 ) of word;
  -- Массив слов, хранящихся в ОЗУ
Signal aRGData : Data;

begin
     process    (bClk)
     Begin
          -- Принят тактовый импульс
          if ( bClk = '1' and bClk'event ) then

               -- Занести полученные данные в регистр
               if ( bEWr = '1' )     then
                    aRGData (0) <= wiA1;
                    aRGData (1) <= wiA2;
                    aRGData (2) <= wiA3;
                    aRGData (3) <= wiA4;
               end if;

               -- Сдвиг хранящихся данных и их вывод
               if ( bEShift = '1' )  then
                    aRGData (0) <= aRGData (1);
                    aRGData (1) <= aRGData (2);
                    aRGData (2) <= aRGData (3);
                    aRGData (3) <= wRepA;
               end if;
          end if;--( bClk = '1' )
     end process;

     -- Выдать хранящиеся данные
     woA1 <= aRGData (0);
     woA2 <= aRGData (1);
     woA3 <= aRGData (2);
     woA4 <= aRGData (3);

end Struct;
```

```
_________________end_of_ShiftRG.vhd_____________________________________

_____________________________types.vhd_____________________________________
-- \brief  Библиотека, описывающая используемые типы данных
_______________________________________________________________________________

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
_______________________________________________________________________________

package TYPES is

  -- Размер байта
     constant BYTESIZE : integer := 8;

     -- Размер слова
     constant WORDSIZE : integer := 16;

     -- Размер двойного слова
     constant DWORDSIZE : integer := 32;

     -- Определение типа байт
     subtype byte is std_logic_vector (7 downto 0);

     -- Определение типа слово
     subtype word is std_logic_vector (WORDSIZE-1 downto 0);

  -- Определение типа двойное слово
     subtype DWORD is std_logic_vector (DWORDSIZE-1 downto 0);

     -- Определение типа массив слов
  type word_arr is array ( NATURAL range <> ) of word;
_______________________________________________________________________________
     function GetDWORD ( wLeftWord, wRightWord : IN word ) return
DWORD;
_______________________________________________________________________________

End package TYPES;
_______________________________________________________________________________
package body TYPES is
_______________________________________________________________________________

     -- Собрать двойное слово
     function GetDWORD ( wLeftWord, wRightWord : IN word ) return
DWORD is
     variable dwResault : DWORD;
     variable nCnt2 : integer := 0;
     Begin
          -- Записать младшее слово
          for nCount IN 0 to 15 loop
```

```
                dwResault ( nCount ) := wRightWord ( nCount );
            end loop;

            -- Записать старшее слово
            for nCount IN 16 to 31 loop
                dwResault ( nCount ) := wLeftWord ( nCnt2 );
                nCnt2 := nCnt2 + 1;
            end loop;

            return dwResault;
        End function GetDWORD;
```

```
end package    body TYPES;
```

_____end_of_types.vhd_____

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

УДК 004.056.55

Политехнический институт
Факультет приборостроения, информационных технологий и электроники
Выпускающая кафедра:
Информационная безопасность систем и технологий
Учебная группа 11ПИ1

УТВЕРЖДАЮ
Зав. кафедрой ИБСТ
к.т.н., доцент С.Л. Зефиров
«27» 04 2016 г.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

студентки группы 11ПИ1 Проценко Екатерины Дмитриевны

Специальность 100503 – Информационная безопасность автоматизированных систем

Тема дипломного проекта: Устройство криптографического преобразования информации.

Руководитель ВКР
Ведущий инженер НИОИ-57 АО «ПНИЭИ»
Краснов Всеволод Александрович

Нормоконтролер
к.т.н., доцент
Иванов А.П.

1 Цели и задачи ВКР

Цель работы - Разработка функционально законченного устройства и реализация его на ПЛИС. Устройство должно обеспечивать приём IP пакетов, криптографическое преобразование используя несекретный криптографический алгоритм «Уступ», передачу зашифрованных пакетов по сетям общего пользования.

Устройство предназначено для приёма IP пакетов, криптографическое преобразование используя несекретный криптографический алгоритм «Уступ», передачу зашифрованных пакетов по сетям общего пользования.

2 Тактико-технические требования к выполнению ВКР

Технические требования:

– устройство криптографического преобразования информации должно обеспечивать приём IP пакетов, криптографическое преобразование используя несекретный криптографический алгоритм «Уступ», передачу зашифрованных пакетов по сетям общего пользования со скоростью не менее 100 Мбит/с;

– работа устройства не должна оказывать влияние на быстродействие всей системы в целом;

– устройство криптографического преобразования информации должно иметь возможность работать в пакетном режиме;

– входные и выходные пакеты данных должны содержаться в буфере внутренней памяти размером не более 2 Кб;

– по завершении преобразования пакета, результат должен отображаться в регистре-счетчике готовых пакетов;

– устройство должно обеспечивать фильтрацию Ethernet кадров по IP адресам отправителя и получателя;

– устройство должно иметь собственный IP адрес, на который будут приходить команды управления;

– работа устройства должна соответствовать требованиям, описанным в криптографическом алгоритме «Уступ»;

– устройство криптографического преобразования информации должно обеспечивать надежный процесс шифрования/расшифрования информации;

– в случае возникновения критической ошибки, должны выставляться соответствующие флаги регистра статуса, либо инициироваться прерывание процесса;

– устройство криптографического преобразования информации должно быть реализовано на элементной базе аналогичной элементам, размещенным на отладочной плате Xilinx Spartan 6 LX45T;

– при выполнении дипломного проекта проанализирован алгоритм работы устройства криптографического преобразования информации включающий такие этапы как:

– инициализация;

– заполнение регистров проверок;

– изменение регистров проверок;

– обработка команд управления;

– отправка пакета по линии на блок криптографической защиты информации;

– установка ключа;

– установка синхроданных;

– запись открытых/закрытых данных;

– чтение результата.

В ВКР должны быть рассмотрены следующие вопросы:

– обзор устройств криптографического преобразования информации;

– характеристики ПЛИС семейства Spartan 6 фирмы Xilinx;

– описание криптографического алгоритма «Уступ»;

– описание структуры аппаратного модуля криптографической защиты информации;

– разработка структурной схемы устройства криптографического преобразования информации;

– описание работы устройства криптографического преобразования информации по структурной схеме;

– разработка функциональной схемы блока криптографической защиты информации;

– описание работы блока криптографической защиты информации по функциональной схеме;

– разработка функциональной схемы устройства управления блоком криптографической защиты информации;

– расчет быстродействия устройства криптографического преобразования информации.

3 Требования к разрабатываемой документации

Отчет по ВКР выполняется в соответствии с требованиями ГОСТ 7.32.

Перечень и содержание графической части ВКР:

– блок криптографической защиты информации. Алгоритм «Уступ». Криптографическая схема – 1л., ф. А1 (плакат);

– структура аппаратного модуля криптографической защиты информации – 1л., ф. А1 (плакат);

– структура устройства криптографического преобразования информации – 1л., ф. А1 (плакат);

– работа шифратора на примере шифрования одного пакета данных. Временная диаграмма – 1л., ф. А1 (плакат);

– устройство криптографического преобразования информации. Алгоритм работы предпускового контроля – 1л., ф. А1 (плакат);

– устройство криптографического преобразования информации. Алгоритм проведения функционального тестирования процессором MicroBlaze– 1л., ф. А1 (плакат);

Электронная копия отчета и графической части ВКР должна быть представлена на магнитном или оптическом носителе.

4 Сроки выполнения ВКР

Тема утверждена приказом ректора ПГУ № _____ от «___»_____ 2016 г.

Дата выдачи задания « 20 » февраля 2016 г.

Время дипломного проектирования с 18.04.16 г. по 10.06.16 г.

Задание к исполнению принял «___» _____ 2016 г.

Исполнитель ВКР _____ Проценко Е.Д.

Ректору Пензенского
государственного университета
А. Д. Гулякову
студентки 5 курса ФПИТЭ
очной формы обучения
Проценко Екатерины Дмитриевны

заявление.

Я, Проценко Екатерина Дмитриевна, студентка 5 курса очной формы обучения специальности 10.05.03 – «Информационная безопасность автоматизированных систем», прошу провести проверку с использованием системы «Антиплагиат» дипломного проекта на тему «Устройство криптографического преобразования информации», выполненной мной самостоятельно, на содержание элементов плагиата.

Все прямые заимствования из печатных и электронных источников, а также из защищенных ранее выпускных квалификационных работ, кандидатских и докторских диссертаций имеют соответствующие ссылки.

Я ознакомлен с действующим Стандартом университета «Выпускная квалификационная работа обучающихся по образовательным программам высшего образования – программам бакалавриата, программам специалитета и программам магистратуры», согласно которому обнаружение плагиата является основанием для недопуска ВКР к защите и отчисления из университета.

– /Е.Д. Проценко/
01.06.2016 г.

ВКР представлена на проверку 01.06.2016 г.

Руководитель ВКР_____ /В.А. Краснов/

Протокол

проверки на оригинальность в системе «Антиплагиат.ВУЗ» выпускной квалификационной работы на тему «Устройство криптографического преобразования информации» студентки 5 курса Е.Д. Проценко специальности 10.05.03 – «Информационная безопасность автоматизированных систем».

Руководитель ВКР ведущий инженер АО «ПНИЭИ» Краснов Всеволод Александрович.

Отчет о проверке № 1

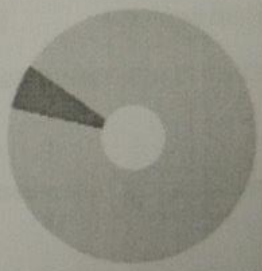
ФИО: Иванов Алексей Петрович
дата выгрузки: 02.06.2016 23:58:31
пользователь: ap_ivanov@mail.ru / ID: 1267386
отчет предоставлен сервисом «Антиплагиат»
на сайте http://www.antiplagiat.ru

Информация о документе

№ документа: 35
Имя исходного файла: ВКР_Проценко_ЕД_11ПИ1.pdf
Размер текста: 921 кБ
Тип документа: Не указано
Символов в тексте: 102142
Слов в тексте: 11641
Число предложений: 314

Информация об отчете

Дата: Отчет от 02.06.2016 23:58:33 - Последний готовый отчет
Комментарии: не указано
Оценка оригинальности: 93.52%
Заимствования: 6.48%
Цитирование: 0%



Оригинальность: 93.52%
Заимствования: 6.48%
Цитирование: 0%

Руководитель ВКР В.А. Краснов

Секретарь ГЭК А.П.Иванов

08.06.2016 г.

О Т З Ы В

на выпускную квалификационную работу студента

Пензенского государственного университета

Проценко Екатерины Дмитриевны

Тема проекта: Устройство криптографического преобразования информации

Целью выпускной квалификационной работы Проценко Е.Д. является разработка функционально законченного устройства и реализация его на ПЛИС. Устройство должно обеспечивать приём IP пакетов, криптографическое преобразование используя несекретный криптографический алгоритм «Уступ», передачу зашифрованных пакетов по сетям общего пользования.

Работа выполнена в полном объёме, а разработанное устройство соответствует требованиям технического задания.

В техническом задании на выпускную квалификационную работу ставилась задача разработать устройство криптографического преобразования информации на отладочной плате Xilinx Spartan 6 LX45T.

В ходе дипломного проектирования была разработана структурная схема устройства и функциональная схема шифратора, работа которого соответствует требованиям, описанным в криптографическом алгоритме «Уступ», была разработана схемотехника ПЛИС на языке VHDL, описывающая работу шифратора, в конце работы было рассчитано быстродействие устройства, которое соответствует требованиям, описанным в техническом задании.

Автором опубликована статья на тему «Сравнительный анализ характеристик ПЛИС типа FPGA Xilinx семейств Spartan-6 и Virtex-6» в сборнике трудов научно-практической конференции «Актуальные проблемы науки и образования» студентов, аспирантов и профессорско-

преподавательского состава университета, проходившей с 25 по 29 апреля 2016 года.

Положительными сторонами проекта явились применение современной элементной базы и соответственно современных методов проектирования.

Дипломный проект выполнен в полном соответствии с заданием и заслуживает оценки "отлично", а студент Проценко Е.Д. достойна присвоения квалификации специалиста по информационной безопасности по специальности 10.05.03 – "Информационная безопасность автоматизированных систем".

Количество баллов: 15

Руководитель ВКР,
Ведущий инженер АО «ПНИЭИ» Краснов В.С.

Р Е Ц Е Н З И Я

на выпускную квалификационную работу студента

Пензенского государственного университета

Проценко Екатерины Дмитриевны

Тема проекта: Устройство криптографического преобразования информации

В данной работе рассмотрена возможность создания концептуальной модели устройства, позволяющего обеспечить преобразование информации, используя несекретный криптографический алгоритм «Уступ», а так же передачу зашифрованных пакетов по сетям общего пользования.

Актуальность темы обусловлена широким спектром применения в системах и сетях связи оборудования, построенного с использованием технологии Ethernet, и потребностью создания высокоскоростных систем защиты и обработки информации циркулирующей в таких сетях.

Дипломный проект Проценко Е.Д. – хороший пример реализации программно-аппаратных комплексов защиты информации с использованием программируемых логических интегральных схем (ПЛИС) фирмы Xilinx.

В ходе дипломного проектирования была разработана структурная схема устройства и функциональная схема шифратора, работа которого соответствует требованиям, описанным в криптографическом алгоритме «Уступ», была разработана схемотехника ПЛИС на языке VHDL, описывающая работу шифратора, в конце работы было рассчитано быстродействие устройства, которое соответствует требованиям, описанным в техническом задании.

В содержательном плане работа охватывает как описание структуры, так и подробное описание назначения и работы составляющих систему компонентов.

Особенностями дипломного проекта является комплексный подход к проблеме разработки и моделирования устройства на базе программируемых

логических интегральных схем. К достоинствам работы можно отнести детальную проработку функциональных возможностей разработанного устройства, а так же высокую производительность всей системы в целом. Также необходимо отметить, что разработка является частью реального проекта.

В качестве недочета стоит отметить, что для выработки синхроданных в устройстве не реализован специальный программный датчик случайных чисел.

Выпускная квалификационная работа Проценко Е.Д. может быть представлена к защите на заседании Государственной аттестационной комиссии и заслуживает оценки «отлично».

Количество баллов: 9

Рецензент
Ведущий инженер АО «ПНИЭИ» С.В. Печилейский