Контроллеры Fast Ethernet для встроенных применений

Первая часть данной статьи продолжает обзор, произведенный А. Сигаевым в нашем журнале («КиТ» № 2'2000), во второй части статьи описывается пример разработки MAC-контроллера в FPGA.

Иосиф Каршенбойм

lk@lmail.loniis.spb.su

Введение

Вместо введения адресую читателя к статье А. Сигаева [1], где дан достаточно подробный анализ существующих решений по применению встроенных решений для Ethernet 10/100 для микроконтроллеров. Однако хочется несколько дополнить этот обзор, особенно в части решений для 100Base-X, 100Base-FX.

Потребность создания недорогих контроллеров со встроенными аппаратными средствами доступа непосредственно к сети Ethernet заставляет разработчиков искать такие технические решения, которые могли бы быть реализованы в нелегких условиях нашей действительности. Многократно обсуждалось, что для отечественных разработчиков недоступны многие технологии, которые широко используются во всем мире. И главная наша беда — это невозможность для большинства фирм продавать свою интеллектуальную собственность — IP. Таким образом, поскольку мы не имеем возможности конкурировать на мировом рынке IP и продавать наши проекты за те деньги, которые они стоят на самом деле, сразу же возникают трудности при разработке. Многие западные фирмы работают по такому принципу: разрабатываем только то, что хорошо умеем делать сами, а все остальное — покупаем как IP, отлаженное, без ошибок и с сопровождением. Это позволяет сосредоточить усилия и значительно экономить время. Если для западной фирмы купить отлаженный проект, аналогичный описанному ниже, за $30 тыс. не составляет труда (или купить зашифрованную мегафункцию за пару тыс. долл.), то отечественный разработчик стремится получить то, что он не знает, из открытых источников в интернете или разработать сам. Что касается открытых проектов, то здесь есть много мнений. Но, в любом случае, все ошибки в работе такого проекта будут находиться и исправляться пользователем проекта, одна за одной. Ибо цель таких открытых проектов — реклама, а не конечный работающий продукт. В том случае, когда разработчик берется «быстренько» сделать недостающий в его изделии блок, возникает много вопросов.

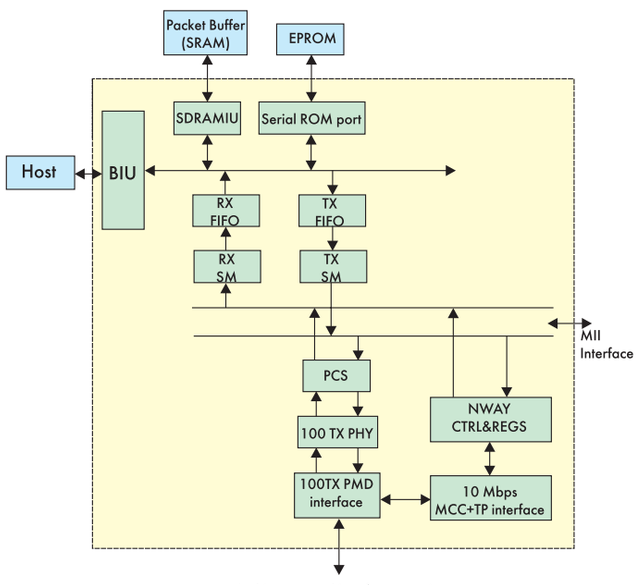

Рис.1 Architecture and interface overview

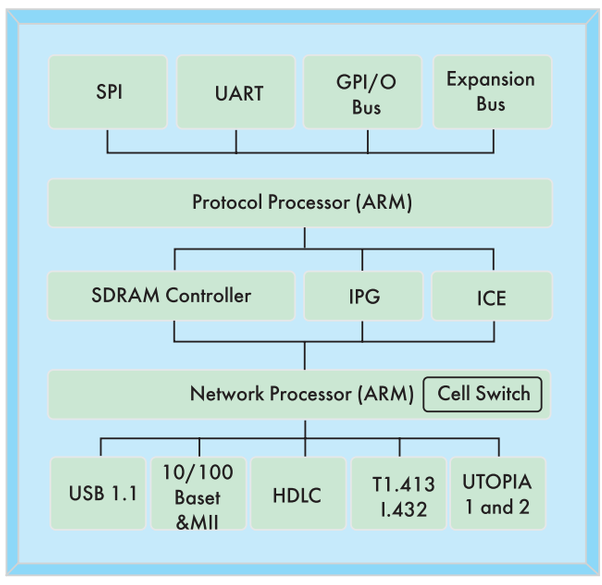

Рис. 2 Helium 210 System Interfaces

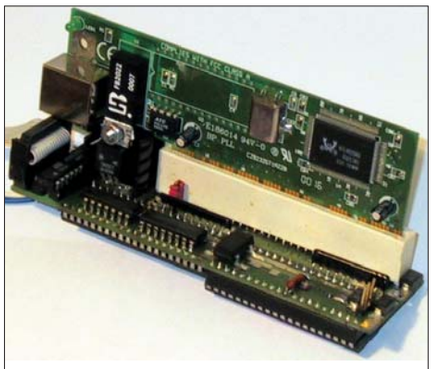

Рис. 3

Особенно много вопросов по тематике доступа непосредственно к сети Ethernet встречается в телеконференциях. А то, что большая часть этих вопросов остается без ответа, привело автора к написанию данной статьи. Без сомнения, в дальнейшем интерес к данной тематике будет только усиливаться. Этому будет способствовать, с одной стороны, необходимость интеграции разрабатываемых устройств в существующие локальные и глобальные сети, а с другой стороны — удешевление микросхем FPGA и программных инструментов для них. В статье приводится описание проекта MAC-контроллера, выполненного в FPGA, рассмотрены вопросы стыка MAC-контроллера с хостом, а также приводится подробное описание интерфейса MII и назначение его сигналов на примере микросхемы трансивера LXT972 фирмы Level One.

И еще несколько слов от автора. Прошли те времена, когда специалисты годами разрабатывали какую-то узкую область. Сегодня разработчику приходится делать все и сразу. Мощные программные средства позволяют новичкам использовать знания и опыт, накопленные предыдущими поколениями инженеров. Большие фирмы, где есть возможность работать узкоспециализированным разработчикам, теперь можно сосчитать по пальцам. А для большинства отечественных фирм характерна ситуация, когда разработчик аппаратуры выполняет весь комплекс работ по разработке, наладке и сопровождению изделия. Часто изделия выполняются на заказ, и поэтому приходится осваивать новые направления. Естественно, что каждое задание может быть реализовано несколькими способами. Для аппаратчика, например, легче, а главное привычнее реализовать устройство, выполняющее свои функции в «железе». А программисты пишут в конференцию, что им легче написать много-много программ для железки, но пусть там будет готовая «железка», в данном случае — сетевая карта.

Поэтому данная статья должна рассматриваться как рекомендации по выбору направления разработки. Кроме того, не надо забывать и о стоимости изделия. И кроме затрат на комплектующие изделия необходимо также учитывать и дополнительные затраты. Конечно, микропроцессор со встроенным контроллером — это очень перспективное решение. Но во что встанет программное обеспечение для работы с ним и сколько времени и сил понадобится для того, чтобы научиться с ним работать.

Именно поэтому автор попытался написать статью так, чтобы она была понятна специалисту с минимальным опытом в данном вопросе.

Обзор

К стандартным решениям данного вопроса можно отнести использование микросхемы контроллера сети 10/100 Fast Ethernet **MX98726EC** фирмы Macronix International (**www.macronix.com**). Данные микросхемы представляют собой полное решение проблемы стыка с сетью. Блок-схема данной микросхемы приведена на рис. 1.

Существуют и более «экзотические» решения, например такие, как двухпроцессорная микросхема Helium 210 фирмы Virata (**www.virata.com**), блок-схема которой представлена на рис. 2. Здесь один из процессоров обрабатывает нижние уровни протокола, а другой — верхние.

Но не всегда возможно применить такие контроллеры, так как появляется зависимость только от одного производителя или одного поставщика микросхем и потому цена на данные микросхемы может быть неустойчива.

Однако существует еще несколько решений данного вопроса, не получивших освещения в предыдущей обзорной статье.

Как говорилось во введении, многим отечественным разработчикам написать уникальные программы гораздо проще, чем изготовить «самодельное» устройство. Такая ситуация характерна для изделий, выпускающихся малыми сериями. Именно поэтому появляются и такие нестандартные решения вопроса, как описанное ниже.

Устройство реализовано следующим образом: для стыка с сетью применена стандартная сетевая карта, а интерфейсные сигналы для управления картой формируются на специальном контроллере. Такое решение было применено В. П. Константиновым (vp@tts.lt), и, судя по тому, как живо оно обсуждалось в конференции fido7.ru.embedded, чувствуется, что это решение многих не оставило равнодушным. Вот фрагмент из его письма:

«При разработке все было направлено на получение минимальной стоимости. Она и получилась минимально возможной на сегодня для такого типа устройств.

Абсолютные цифры комплектации сильно варьируются от тиража и аппетитов поставщиков комплектующих... Наверное $25 — это сегодня реальность.

Блок схема это просто три квадратика:

контроллер PCI,

контроллер сетевых протоколов,

контроллер Пользователя.

Тактовая частота PCI шины и контроллеров 24 МГц.»

Фотография данного устройства приведена на рис. 3.

В данной конструкции многих разработчиков привлекает то, что нет необходимости проводить большую разработку аппаратных средств, а основные затраты труда приходятся на разработку программ для микроконтроллеров. Учитывая то, что для многих разработчиков это не представляет большой трудности, данный вариант для многих применений может быть достаточно привлекательным. Основной трудноразрешимой проблемой здесь является следующее: для каждого типа сетевой карты, примененной в данной конструкции, необходимо писать собственный программный драйвер, учитывающий особенности работы конкретной микросхемы — контроллера сети. При замене карты на карту другого типа или при обновлении карты производителем работу по программированию необходимо будет выполнять заново.

И, последнее, о чем необходимо сказать — процессор со встроенным контроллером MII — RISC-процессор RC32355 фирмы IDT. Данный процессор имеет интерфейсы Ethernet, ATM, USB, TDM и два 16550, поддерживает различные типы памяти. Производительность — 195 Mips при тактовой частоте до 150 МГц. Процессор RC32355 представляет собой хорошее решение для платформ VoIP/VoDSL (см. рис. 4). Более подробно с документацией на данный процессор можно ознакомиться на сайте фирмы **www.idt.com** или на сайте дистрибьютора **www.efo.ru**.

Другой путь: контроллер MAC-адресов в FPGA + трансивер

Разные задачи — разные способы реализации проекта

Из приведенных выше обзоров можно сделать следующий вывод — задача пользовате-

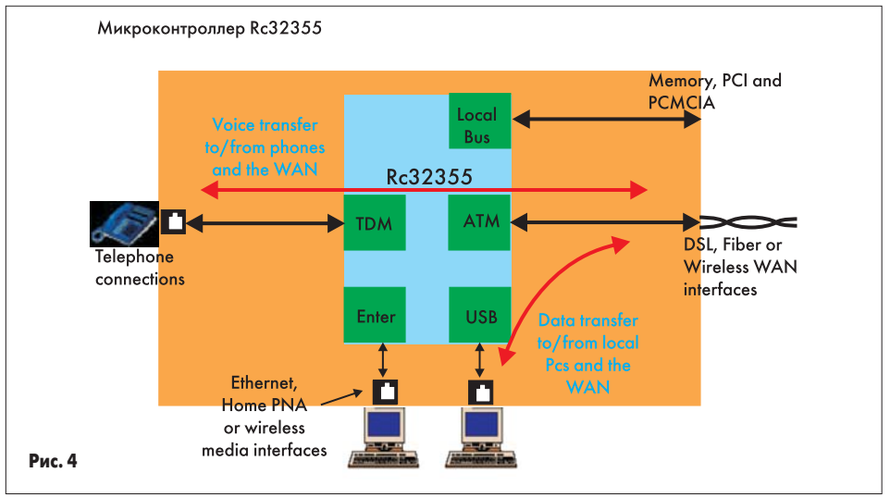

Рис. 4

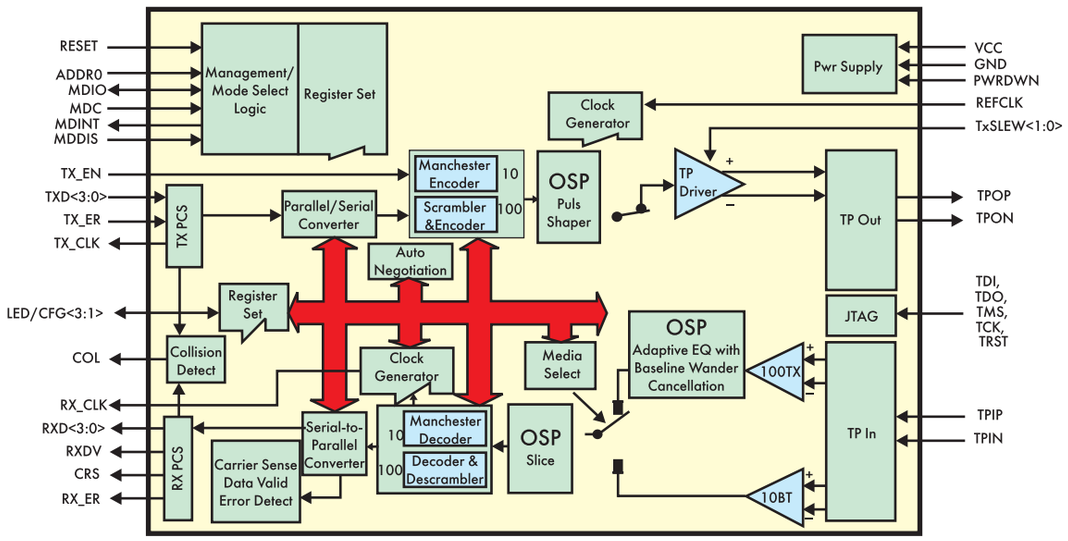

Рис. 5

ля диктует способы ее решения. Невозможно одними и теми же средствами оптимально решить все задачи. Причем здесь необходимо рассматривать не только то устройство, которое мы хотим подключить к сети, но и саму сеть, ее быстродействие, режим работы и загрузку. Так, например, для реализации связи 8-разрядного микроконтроллера, находящегося на ненагруженной 10-мегабитной сети, будет достаточно очень простых и дешевых решений. С другой стороны, если нам необходим узел, одновременно производящий обработку пакетов TCP-IP и их передачу в сети 100 Мбит/с, то тогда проще будет применить процессор со встроенным MAC-контроллером. Если же нам требуется передавать данные потоком, а число логических соединений по протоколам верхних уровней велико, то, скорее всего, ресурсов одного процессора будет мало и его использование будет в такой задаче нецелесообразно. Поэтому применим другое решение данной задачи. Контроллер MAC-адресов реализуем в FPGA, а для стыка с физической линией применим микросхему-трансивер.

Реализация контроллера MAC-адресов в FPGA

Контроллер MAC-адресов (MAC-контроллер), выполненный в FPGA, имеет те же преимущества, что и любой специализированный вычислительный узел. При выполнении конкретной задачи производительность специализированного вычислительного узла оказывается значительно выше производительности стандартных микропроцессоров. Обработка пакетов TCP-IP может производиться в хост-процессоре или в массиве процессоров. Кроме того, часть функций формирования пакетов TCP-IP может производиться и в FPGA. Более того, в FPGA может быть встроен процессорно-аппаратный узел формирования пакетов верхнего уровня. Такая архитектура построения контроллера может дать значительный выигрыш по производительности по сравнению с рассмотренными выше решениями.

В данной статье будет описан только контроллер MAC-адресов.

Контроллер MAC-адресов имеет канал передачи данных, канал приема данных и канал служебного интерфейса.

В канале передачи данных MAC-контроллер формирует всю диаграмму передачи данных, автоматически подсчитывает контрольную сумму (CRC) при передаче данных и добавляет ее к кадру данных. Так же автоматически производится проверка на правильность передаваемой информации по числу передаваемых слов и ведется контроль правильности обслуживания FIFO.

В канале приема данных MAC-контроллер определяет адрес кадра, приходящего из сети, и если этот кадр имеет адрес MAC-контроллера или широковещательный адрес, то такой кадр принимается. При приеме кадра производится подсчет контрольной суммы, производится подсчет числа принятых слов, определяются ошибки при работе с FIFO. Так же производится подсчет числа целых принятых кадров, находящихся в FIFO.

Трансивер

Функциональные возможности

Для реализации проекта был выбран трансивер со стандартным протоколом MII, со стандартными регистрами, в котором адресация и расположение управляющих битов так же задается стандартно. Подобные микросхемы трансиверов выпускаются многими фирмами — AMD, Level One, Cirrus Logic и др.

Рассмотрим трансивер LXT972 фирмы Level One (www.level1.com). Блок-схема этого трансивера приведена на рис. 5.

LXT972 — двухскоростной Fast Ethernet Transceiver на основе протокола CSMA-CD

LXT972 — однопортовый приемопередатчик локальной сети Fast Ethernet 10/100, который поддерживает работу в сети 10 и 100 Мбит/с. Он соответствует всем требованиям IEEE 802.3. LXT972 может непосредственно управлять как линией 100Base-TX (до 140 метров), так и линией 10Base-T (до 185 метров) и поддерживает прикладные программы для обоих типов линий. Он обеспечивает простое сопряжение контроллера доступа к среде 10/100 (MACs) с Media Independent Interface (MII). Интерфейс MII позволяет получить доступ к расширенному набору регистров управления трансивером. LXT972 поддерживает стандартный протокол CSMA/CD или дуплексную работу на 10 и 100 Мбит/с. Его эксплуатационный режим может быть установлен с помощью автопереговоров, параллельного обнаружения или ручного управления. При включении питания LXT972 читает свои выводы конфигурации, чтобы проверить принудительные параметры настройки. Если режим принудительной конфигурации не установлен, то будет использовано обнаружение auto-negotiation/parallel, чтобы автоматически определить эксплуатационные режимы линии. Если устройство PHY с другой стороны линии связи поддерживает автопереговоры, LX972 будет вести автопереговоры с ним, используя импульсы Fast Link Pulse (FLP) Bursts. Если партнер PHY не поддерживает автопереговоры, LXT972 автоматически обнаружит либо присутствие импульсов связи на частоте (10 Мбит/с PHY), либо символы неактивного состояния в линии (100 Мит/с PHY) и соответственно установит свои эксплуатационные режимы.

Трансивер использует единственный разъем типа RJ-45 и для 10Base-T, и для 100Base-TX. Программируемые выводы (драйверы светодиодов) позволяют настроить режим индикации, требуемый пользователем. LXT972 изготовлен по технологии CMOS и требует только одно напряжение питания — 3,3 В. Трансивер имеет малое энергопотребление (обычно 300 мВт). Корпус — 64-выводной Low-profile Quad Flat Package (LQFP).

Описание сигналов трансивера

Трансивер поддерживает все сигналы канала приема-передачи данных (MII), сигналы служебного канала (MII_MI), а так же сигналы связи с линией и дополнительные сигналы. На примере данного трансивера рассмотрим сигналы интерфейса MII и MII_MI, поскольку это необходимо для описания работы MAC-контроллера. Описание интерфейса состоит из следующих разделов: описание сигналов, их логической связи, их электрических характеристик и механических характеристик

(размеров, контактов и пр.). Далее будут описаны только два первых пункта, как наиболее общие для всех выпускаемых трансиверов.

Интерфейс MII

Интерфейс MII предназначен для связи MAC-контроллера с трансивером и состоит из двух частей: собственно канала приема-передачи данных (MII) и служебного канала управления (MII_MI).

Канал передачи данных (MII) содержит следующие сигналы:

TXD[3... 0]: Transmit Data — группа параллельных сигналов данных, которые поступают в трансивер из MAC. Они выдаются синхронно относительно TX_CLK.

TX_EN: Transmit Enable. MAC устанавливает этот сигнал, когда установлены достоверные данные на TXD. Этот сигнал должен быть синхронизирован с TX_CLK.

TX_ER: Transmit Error. Сообщает о том, что в передаваемом потоке данных произошла ошибка. Этот сигнал должен быть синхронизирован с TX_CLK. Трансивер имеет возможность передавать в линию сигнал ошибки TX_ER, получаемый от MAC. Когда MAC устанавливает TX_ER, LXT972 установит кодовую комбинацию «H» на выводах TPOP/N. Типичная ситуация для установления сигнала ошибки по передаче такова: хост не успел заполнить буфер передачи данными, кадр еще не закончен, а буфер считан весь. Передача еще не остановлена и в линию передаются «пустые» данные, которые не будут соответствовать кадру верхнего уровня. Если не устанавливать сигнал TX_ER, то неправильно сформированный кадр уйдет в линию и будет принят приемником. Далее приемник, произведя проверку кадра по контрольной сумме, все же отбракует принятый кадр. Но если установить сигнал TX_ER, то на приемной стороне кадр будет отбракован еще до конца приема кадра, то есть в момент получения сигнала ошибки.

TX_CLK: Transmit Clock. TX_CLK вырабатывается в трансивере и передается в MAC. TX_CLK = 2,5 МГц для операций на 10 Мбит/с, TX_CLK = 25 МГц — для 100 Мбит/с.

Канал приема данных (MII) содержит следующие сигналы:

RXD [3... 0]: Receive Data — группа параллельных сигналов данных, которые выдаются из трансивера в MAC синхронно относительно RX_CLK.

RX_DV: Receive Data Valid. Трансивер устанавливает этот сигнал, когда получает достоверный пакет данных и, соответственно, выдает достоверные данные на RXD. Этот вывод синхронен с RX_CLK. Изменения сигнала по времени зависят от того, какой режим по быстродействию используется в линии:

- Для режима 100TX RX_DV устанавливается с первым нибблом Start of Frame Delimiter (SFD) «5D» и остается установленным до последнего ниббла пакета данных.
- Для режима 10BT полная преамбула усечена.

RX_ER: Receive Error. Трансивер сообщает о том, что в приемном потоке данных произошла ошибка. Этот вывод синхронен с RX_CLK. На приемной стороне, когда LXT972 находится в режиме 100TX и получает недопустимый символ из сети, он устанавливает RX_ER и код «1110» на выводах RXD.

RX_CLK: Receive Clock. Вырабатывается в трансивере и передается в MAC. RX_CLK = 2,5 МГц для операций на 10 Мбит/с, RX_CLK = 25 МГц — для 100 Мбит/с.

Канал данных (MII) содержит дополнительные сигналы:

COL: Collision Detected. Трансивер устанавливает этот вывод, когда обнаружено столкновение на линии. Вывод остается высоким во время столкновения на линии. Этот сигнал асинхронный и неактивен в течение дуплексной операции.

CRS: Carrier Sense. Опрос несущей — асинхронный вывод. В течение полудуплексной операции (бит 0.8 = 0) трансивер устанавливает этот вывод во время передачи или приема пакетов данных. В течение дуплексной операции (бит от 0.8 = 1) CRS установлен при приеме. Установка бита CRS производится асинхронно относительно RX_CLK. CRS сбрасывается при потере несущей частоты, синхронной с RX_CLK.

Служебный канал (MI) содержит следующие сигналы:

MDDIS: Management Disable. Когда MDDIS = 1, MDIO заблокирован для операций чтения и записи. При включении питания или при сбросе, когда MDDIS = 0, выводы управления интерфейса микросхемы устанавливают соответствующие биты регистра как «заданные по умолчанию значения». После того, как цикл power-up/RESET закончен, управление битами регистров возвращается последовательному каналу MDIO.

MDC: Management Data Clock. Частота для последовательного канала данных MDIO. Максимальная частота — 8 МГц.

MDIO: Management Data. Двунаправленный последовательный канал данных для связи с регистром STA трансивера.

MDINT: Management Data Interrupt. Когда прерывание для трансивера разрешено, то есть в регистре 18 трансивера бит 01 = 1, активный низкий сигнал на этом выводе указывает изменение состояния. Прерывание сбрасывается при чтении Регистра 19 трансивера. Установка бита 18.1 = 1 дает возможность устройству запросить прерывание через вывод MDINT. Активный низким уровнем сигнал на этом выводе указывает изменение состояния LXT972. Прерывания могут быть вызваны четырьмя условиями:

- законченные автопереговоры (Auto-negotiation complete);
- изменение состояния режима быстродействия (Speed status);
- изменение состояния режима дуплекс (Duplex status);
- изменение состояния связи (Link status).

Сигналы связи с линией:

TPOP, TPON — положительный и отрицательный выводы для подключения к витой паре. При работе в 100Base-TX или 10Base-T TPOP/N выдают в линию импульсы, соответствующие 802.3.

TPIP, TPIN — положительный и отрицательный выводы для подключения к витой паре. При работе в 100Base-TX или 10Base-T TPOP/N получают из линии дифференциальные импульсы, соответствующие 802.

Дополнительные сигналы

RESET — сброс. Этот активный при низком уровне сигнал действует по «или» с битом Сброса регистра управления (регистр 0, бит 15). Цикл сброса LXT972 продлен до 258 мс (номинал), после этого сигнал сброса неактивен.

ADDR0 устанавливает адрес устройства по служебному интерфейсу управления MII_MI.

PAUSE — когда этот сигнал установлен, LXT972 находится в состоянии Pause в течение автопереговоров.

PWRDWN: Power Down. Когда этот сигнал установлен, LXT972 находится в режиме пониженного потребления.

CLK — опорная частота 25 МГц.

Описание работы трансивера

Интерфейс данных MII

LXT972 поддерживает стандартный Media Independent Interface (MII). MII состоит из интерфейса данных и интерфейса управления (Management Interface). Интерфейс Данных MII пропускает данные между LXT972 и Media Access Controller (MAC). Отдельные параллельные шины предусмотрены для передачи и для приема данных. Этот интерфейс работает в режимах 10 и 100 Мбит/с. Быстродействие устанавливается автоматически, как только эксплуатационные режимы сетевой связи будут определены. В течение операции 100Base-X LXT972 передает и получает 5-битовые символы из линии связи, а в MII данные передаются и принимаются в виде 4-битовых нибблов. Такая кодировка данных позволяет передавать и принимать сигналы управления синхронно с сигналами данных. При отсутствии данных на передачу, при старте пакета, при окончании пакета и при ошибках в линии в поток данных, передаваемых в линию трансивером, вставляются соответствующие управляющие кодовые комбинации.

Конфигурация через аппаратный интерфейс управления

LXT972 обеспечивает аппаратный интерфейс управления для применений, где использование интерфейса MDIO нежелательно. Для аппаратного интерфейса управления задействованы три вывода драйверов светодиодов и светодиоды подключаются к этим выводам одним своим выводом, а другим выводом либо на GND, либо на 3,3 В. Чтобы установить конфигурацию устройства, выводы опрашиваются при включении питания или при действии сигнала «Сброс», а затем они используются для управления светодиодами.

Конфигурация через Management Interface

LXT972 обеспечивает и интерфейс MDIO, и аппаратный интерфейс управления для конфигурации устройства.

MDIO Management Interface

LXT972 поддерживает IEEE 802.3 MII Management Interface, также известный как интерфейс Management Data Input/Output (MDIO). Этот интерфейс позволяет устройствам верхнего уровня управлять состоянием LXT972. Интерфейс MDIO состоит из физического подключения, определенного протокола, который требуется для работы данных устройств, и некоторого набора регистров, которые требуются по стандарту IEEE 802.3.

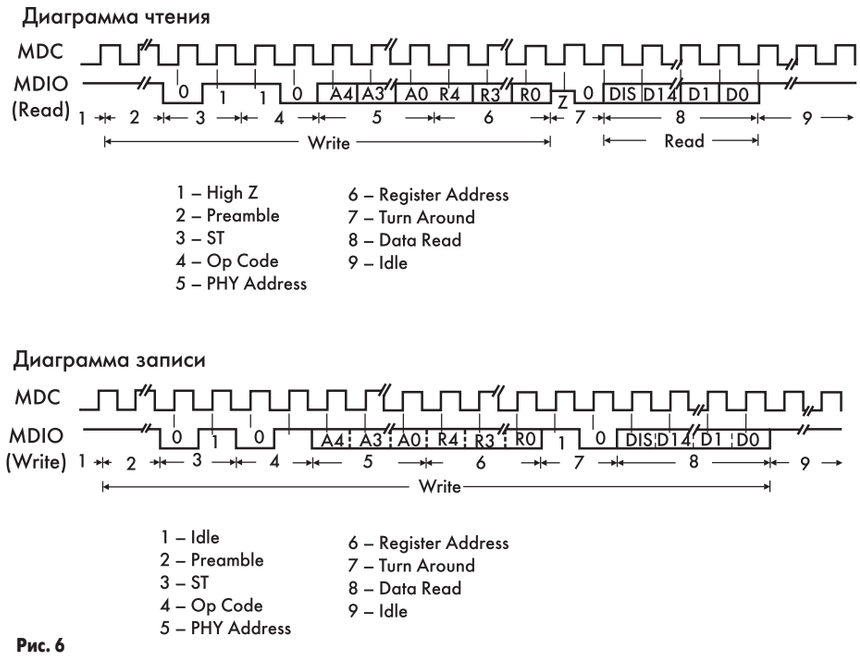

Рис. 6

Таблица 1. Структура полей кадра MAC по IEEE 802.3
MAC: Media Access Control
(Standard reference: ISO/IEC 8802-3)

Длина в октетах (octets)	Поле
7	Преамбула (Preamble)
1	Признак начала кадра – SFD (Start Frame Delimiter)
2 или 6	Адрес назначения (Destination Address)
2 или 6	Адрес отправителя (Source Address)
2	Длина (Length). Длина этого поля – 2 октета, они показывают число октетов данных LLC в поле данных.
N от 46 до 1500	Данные (LLC DATA). Поле данных содержит последовательность из N октетов данных. Минимальный размер этого поля необходим для правильного выполнения операций CSMA/CD. Если необходимо, то к данным могут быть присоединены дополнительные биты заполнения поля до минимально необходимой величины.
	Поле заполнения (PAD)
4	Поле контроля данных – контрольная сумма (FCS: Frame Check Sequence)

Поле кадра: Преамбула (Preamble)

биты 0 _ _ _ _ _ _ 7	Preamble field
10101010	7 октетов с данными 55H

Поле кадра: Признак начала кадра — SFD

биты 0 _ _ _ _ _ _ 7	SFS field
10101011	1 октет с данными D5H

Поле кадра: Поле адреса (Address)

биты 0 _ _ _ _ _ _ 7	Поле адреса 16 бит
октет 1, бит 1	I/G (Individual/Group) I=0 G=1
октеты 1-2	15-разрядное поле адреса

биты 0 _ _ _ _ _ _ 7	Поле адреса 48 бит
октет 1, бит 1	I/G (Individual/Group) I=0 G=1
октет 1, бит 2	U/L (Universal/Local) U=0 L=1
октеты 1-6	46-разрядное поле адреса

LXT972 также поддерживает дополнительные регистры для расширенных функциональных возможностей. LXT972 имеет множество внутренних 16-разрядных регистров. Определенные биты регистра описываются формой записи «X.Y», где X — номер (0-31) регистра, а Y — номер (0-15) бита.

Физический интерфейс состоит из линии данных (MDIO) и линии синхрочастоты (MDC). Операция этого интерфейса управляется выводом разрешения MDDIS. Когда установлено состояние MDDIS = 1, операции чтения и записи MDIO заблокированы, и аппаратный интерфейс управления обеспечивает первичное управление конфигурацией. Когда MDDIS = 0, порт MDIO открыт как для чтения, так и для записи, а аппаратный интерфейс управления не используется.

Адресация MDIO

Протокол позволяет одному контроллеру связываться с двумя чипами LXT972. Вывод ADDR0 может быть установлен в 1 или в 0, и это определяет адрес чипа.

Структура кадра MDIO

Физический интерфейс состоит из линии данных (MDIO) и линии частоты (MDC). Структура кадра показана на рис. 6 (чтение и запись).

Кадр начинается с преамбулы, состоящей из 32 «1» (можно передавать «1» постоянно — это и будет исходным состоянием до начала кадра), затем передается стартовая комбинация из двух битов «01», затем передаются два бита, соответствующие коду операции («10» для чтения и «01» для записи данных), далее передается физический адрес трансивера A[4... 0] и регистра трансивера R[4... 0]. Таким образом, возможно обслуживание нескольких трансиверов по интерфейсу MI. Микросхема LXT972 имеет только один адресный разряд, что позволяет непосредственно подключить к шине только две микросхемы. После передачи адреса в цикле записи данных передается код «10», после чего передатчик передает 16-разрядное слово данных. Далее передатчик приходит в исходное состояние. В цикле чтения данных после передачи адреса передатчик переводит свой выход в третье состояние и, после задержки в 2 бита, необходимых для завершения переходного процесса на шине, начинает прием 16-разрядного слова данных. Адресуемая микросхема после приема адреса подключается к шине и после задержки в 2 бита начинает передавать требуемые данные. После передачи данных микросхема переводит свои выходы в третье состояние и отключается от шины. Передатчик приходит в исходное состояние после приема данных от микросхемы.

Операции на 100 Мбит/с

Как было сказано выше, трансивер работает и в режиме 10 Мбит/с, и в режиме 100 Мбит/с. Работу трансивера в режиме 10 Мбит/с здесь описывать не будем. Рассмотрим более подробно режим работы 100 Мбит/с. Данные между трансивером и MAC-контроллером передаются кадрами в соответствии с IEEE 802.3. Структура полей MAC-кадра по IEEE 802.3 показана в табл. 1.

В течение операции 100Base-X LXT972 передает и получает 5-разрядные символы из линии связи. На рис. 7 показана структура стандартного пакета. Когда MAC не передает данные, LXT972 выдает в линию символы Idle. В режиме 100TX LXT972 производит скремблирование и передачу данных, используя кодирование MLT-3. Данные, полученные из сети в коде MLT-3 дескремблируются, декодируются и передаются в MII MAC. Более подробно об этом можно прочесть в документации [2].

Как показано на рис. 7, MAC-контроллер начинает каждую передачу с преамбулы. Как только LXT972 обнаруживает начало преамбулы, которую он получает из MAC-контроллера, он передает в линию Start-of-Stream Delimiter (SSD, символы J и K). Затем он кодирует и передает остальную часть пакета, включая остальную часть преамбулы, SFD, данные пакета и CRC. Как только пакет из MAC-контроллера закончился, LXT972 передает End-of Stream-Delimiter (ESD, символы T и R) и затем возвращается к передаче символов Idle. Когда LXT972 получает из линии недопустимые символы, он устанавливает RX_ER.

Реализация MAC-контроллера в FPGA

Описываемый здесь MAC-контроллер был разработан для шлюза интернет-телефонии. Основной режим работы MAC-контроллера — 100 Мбит/с, дуплекс. При работе в дуплексе нет столкновений пакетов в линии, поэтому обработка коллизий поддержана только под управлением от хоста, однако при необходимости требуемые узлы легко могут быть реализованы аппаратно. Режим работы на 100 Мбит/с для интернет-шлюза подразумевает большой поток информации как на прием, так и на передачу. Число пакетов определяется числом

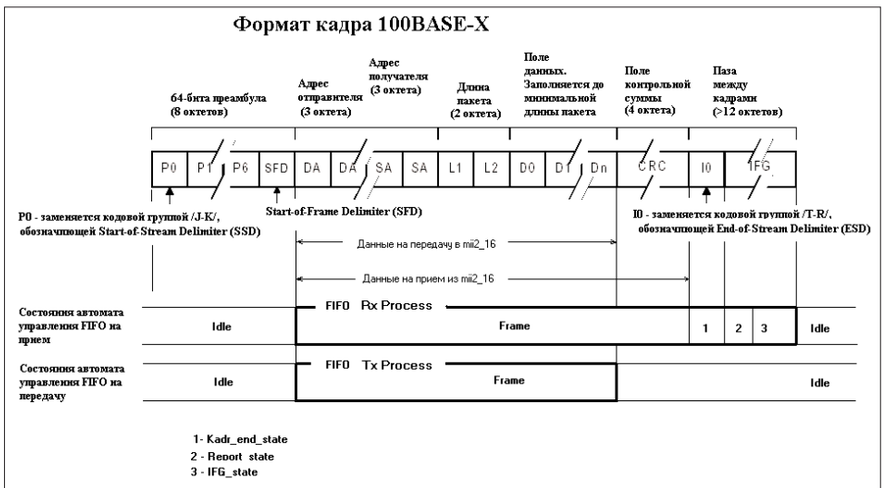

Рис. 7

одновременно ведущихся телефонных разговоров, а объем пакетов определяется типом примененного протокола сжатия. Таким образом, для данного устройства требовалось реализовать контроллер, передающий и принимающий большое количество пакетов и требующий как можно меньше ресурсов от хост-машины.

Контроллер может быть выполнен в различных вариантах, которые определяются требованиями к производительности. Типовой набор блоков, входящих в состав контроллера, приведен на рис. 8.

Контроллер состоит из узла согласования интерфейса с хост-машиной, канала передачи данных, канала приема данных и служебного канала. При необходимости к контроллеру в каналы приема и передачи данных могут быть добавлены блоки DMA. Контроллер был реализован как в варианте с 16-разрядной шиной, так и в варианте с 32-разрядной шиной. Поскольку схема для варианта с 16-разрядной шиной отличается от 32-разрядной только в части стыка с интерфейсом MII, то эта часть проекта реализована в двух вариантах: с 16-разрядной шиной — mii2_16 и с 32-разрядной шиной — mii2_32. Остальная часть схемы каналов приема и передачи данных выполнена с параметризацией, и поэтому одинакова для обоих вариантов исполнения.

Контроллеры DMA и блок сопряжения с хост-машиной выполнены стандартно и в данной статье не рассматриваются.

Канал передачи состоит из блока FIFO и передающей части блока mii2_16 — стыка с интерфейсом MII. Канал приема также состоит из блока FIFO и части mii2_16.

Блоки FIFO

Было разработано два варианта блоков FIFO: первый вариант с двумя полями памяти приведен на рис. 9, второй вариант FIFO с тегами — на рис. 10-11.

Вариант с двумя блоками памяти выполнен стандартно: у каждого поля памяти есть счетчик адресов для записи данных, счетчик адресов для чтения данных, мультиплексор адресов, управляемый из блока управления и общий для обеих полей памяти мультиплексор данных, передаваемых в линию. Мультиплексор данных также управляется из блока управления.

При работе в полудуплексном режиме имеют место столкновения пакетов в линии. Большая часть столкновений происходит при передаче начала пакета. Поэтому первый вариант позволяет производить повторную передачу после столкновения пакетов в линии без перезагрузки FIFO. Недостатком такого FIFO является слабое использование памяти, и поэтому такой блок FIFO применяется в устройствах, работающих с небольшим объемом передаваемой и принимаемой информации, работающих, в основном, в полудуплексном режиме.

Перед тем, как рассмотреть второй вариант FIFO, необходимо сказать несколько слов о тех трудностях, с которыми можно столкнуться в том случае, когда поток данных велик, и необходимо создать устройство, работающее в долговременном непрерывном режиме. Если применить классическую организацию FIFO с глубиной меньшей, чем длина одного минимального пакета, и успевать обрабатывать его быстрее, чем в FIFO может попасть следующий пакет (например, на приеме), то особых проблем не возникает. Если же применить классический вариант FIFO с глубиной большей, чем длина одного минимального пакета, то в FIFO может одновременно находиться информация от нескольких пакетов. Такая ситуация может произойти при большом потоке данных через контроллер. При любом сбое или рассогласовании в устройстве управления, отслеживающем правильность приема информации из FIFO, текущий пакет будет потерян. Мало того, произведя сброс FIFO, мы потеряем и следующий пакет или его часть. Если в момент сброса идет прием очередного пакета, то и он будет потерян, что, в свою очередь, приведет к следующей ошибке в устройстве управления и т. д. И, таким образом, синхронизация данных возможна только в том случае, когда в момент сброса нет приема данных. А это значит, что нужно отслеживать паузу в приеме пакетов (IFG) и успеть сбросить FIFO до начала прихода следующего пакета. Чтобы избежать подобной ситуации, небходимо «привязать» служебную ин-

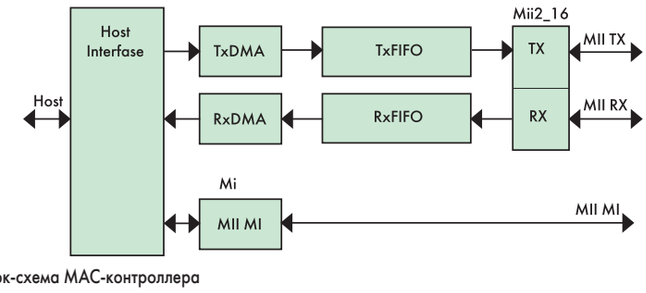

Рис. 8. Блок-схема MAC-контроллера

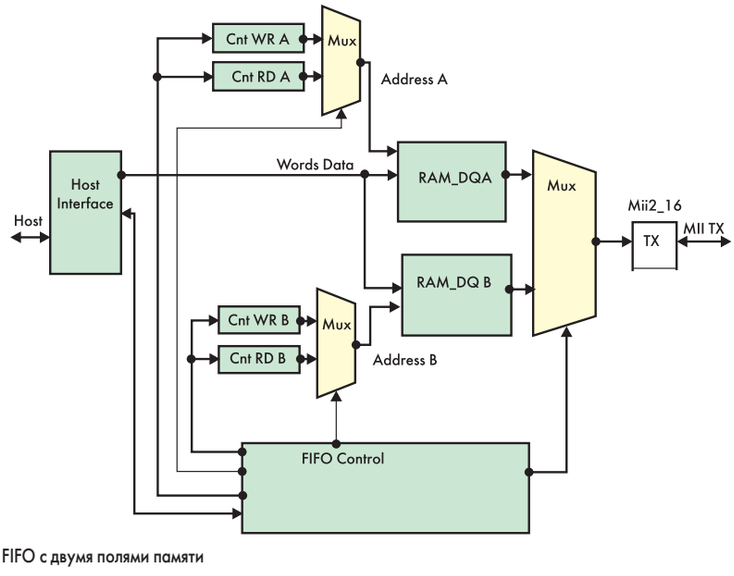

Рис. 9. FIFO с двумя полями памяти

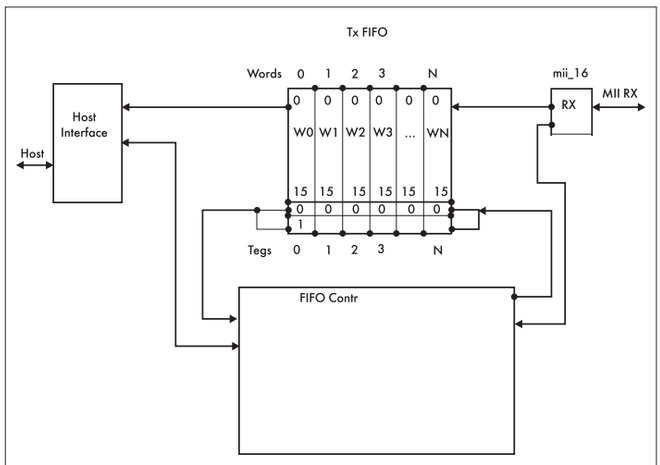

Рис. 10. FIFO на передачу с тегами

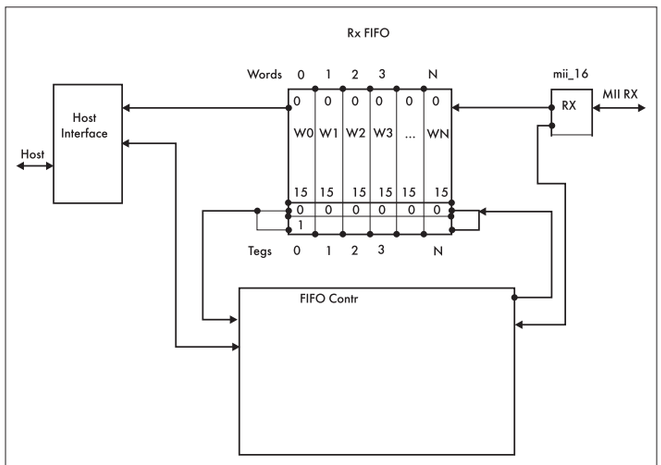

Рис. 11. FIFO на прием с тегами

формацию к словам данных, находящимся в FIFO. Такой вариант FIFO и был применен при разработке — FIFO с тегами. Здесь принято следующее решение: к слову данных разрядностью 16 или 32 бита добавляются еще 2 бита, которые записываются и читаются из FIFO одновременно со словом данных в дополнительные поля FIFO, как показано на рис. 10–11.

Такой вариант FIFO имеет хорошее использование памяти и ориентирован на прием или на передачу большого количества информации. Более того, если на стороне записи и чтения есть резерв по времени хотя бы на одну операцию чтения или записи, то можно передавать через FIFO и служебную информацию. На стороне записи служебная информация помещается в FIFO с записью в тег признака «служебная», а на стороне чтения такая информация извлекается из FIFO устройством управления и помещается в соответствующий регистр служебной информации.

Рассмотрим кодировку тегов в приемном и передающем FIFO:

Кодировка тегов в приемном FIFO

```
teg_out[3... 0] == B»0101»;   -- разрешение записи данных
teg_out[3... 0] == B»0110»;   -- разрешение записи слова «начала» пакета
teg_out[3... 0] == B»1001»;   -- разрешение записи слова «конца» пакета
teg_out[3... 0] == GND;       -- пустой, в такое слово данных может быть помещена любая служебная информация
```

Кодировка тегов в передающем FIFO:

```
teg_in[3... 0] = B»0101»;   -- разрешение записи данных
teg_in[3... 0] = B»1010»;   -- разрешение записи слова управления
teg_in[3... 0] = B»0110»;   -- разрешение записи слова длины пакета
teg_in[3... 0] = B»1001»;   -- разрешение записи слова «конца» пакета
teg_in[3... 0] = GND ;      -- пустой
```

Наличие информации о начале и конце пакета позволяет стыковать такие FIFO с каналами DMA напрямую, так как не требуется никакой дополнительной обработки по сигналам управления. Кроме того, для упрощения обслуживания приемного FIFO и повышения надежности работы в него введена блокировка чтения данных после получения признака конца пакета. Это позволяет производить чтение данных из приемного FIFO в хост-машину пакетами произвольной длины. Блокировка снимается после чтения регистра статуса. Таким образом, при сбое структуры данных в FIFO теряется только один пакет, а другие пакеты потеряны быть не могут. Для передающего FIFO дополнительно введен счетчик передаваемых слов и схема управления, которая проверяет число переданных слов в пакете. Если количество переданных в пакете слов не соответствует тому, что записывалось в счетчик, то вырабатывается сигнал ошибки по числу передаваемых слов.

У блоков FIFO есть еще одна важная задача — развязка синхрочастот. Приемная часть FIFO и блока mii2_16 работают на частоте приема (2,5 или 25 МГц в зависимости от режима работы линии). Передающая часть FIFO и блока mii2_16 работает на частоте передачи (те же 2,5 или 25 МГц). Частоту приема и частоту передачи MAC-контроллер получает из трансивера, и эти частоты могут отличаться друг от друга. Хост-машина работает на своей, системной частоте, например 25 или 33 МГц.

Взаимодействие блоков

Реализация проекта построена по иерархическому принципу. Взаимодействие блоков осуществляется таким образом: выдаем сигнал управления — получаем сигнал подтверждения о выполнении. Получив сигнал подтверждения, снимаем сигнал управления. Каждый верхний блок выдает сигналы управления только на один блок — нижний по иерархии — и получает сигналы о выполнении только от одного блока — нижнего по иерар-

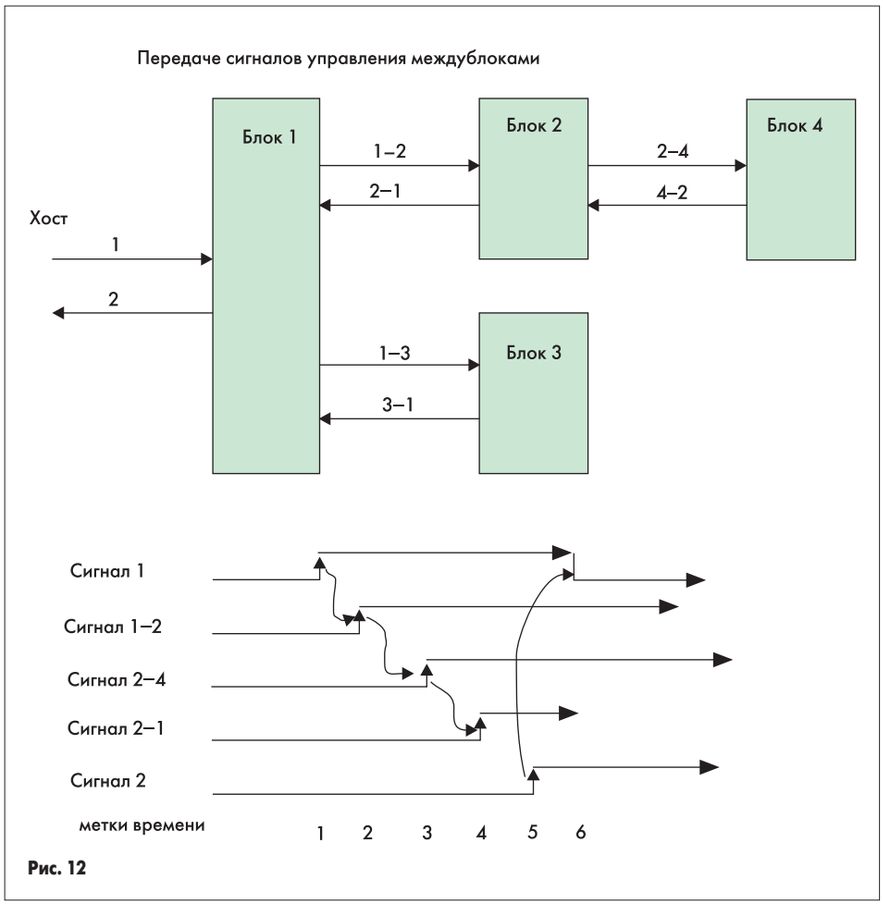

Рис. 12

хии. Если блок не обрабатывает сигналы, то они проходят через этот блок транзитом. Каждый блок, если необходимо, выполняет привязку своих входных сигналов к своей синхрочастоте. На рис. 12 приведено изображение четырех блоков, функционирующих по такому принципу.

Например, процесс передачи начинается с выдачи из хоста основного сигнала управления Сигнал1 = 1. Этот сигнал вызывает начало передачи, причем его длительность должна быть больше длительности, необходимой для того, чтобы этот сигнал был принят и обработан следующим блоком — блоком 2. Когда нижний по иерархии блок выполняет указанное ему действие, то он возвращает верхнему блоку сигнал о выполнении процедуры. Таким образом, блок 1 передает сигнал управления в блок 2, а тот в свою очередь передает сигнал управления в блок 4. Блок 4 передает свой ответный сигнал в блок 2, а блок 2 — в блок 1. Блок 1 возвращает хосту Сигнал2, сообщающий о завершении процедуры. При получении сигнала подтверждения хост снимает сигнал управления. Передача следующего пакета может начаться только после снятия и повторной установки сигнала Сигнал1. Такой подход позволяет легко каскадировать блоки, при этом синхронизация управляющих сигналов получается автоматически.

Небольшое отступление о «строгом» и «нестрогом» проектировании «железа»

Сначала здесь автор по привычке написал «проектирование аппаратуры». Но слово «железо» в его теперешнем понимании подходит гораздо больше к тому, о чем пойдет речь в этом отступлении. Сегодня словом «железо» определяется комплекс аппаратной программируемой логики в FPGA и программ в управляющих микропроцессорах. Программистам не надо объяснять, что такое try-catch, или почему надо сначала проверить, существует ли файл, а потом уже его открывать. Уже давно настали времена, когда при проектировании схем в FPGA должен применяться принцип «патронов не жалеть». К сожалению, у некоторых аппаратчиков от «староглиняных времен» сохранился принцип «простоты схемы». Могу процитировать принцип таких разработчиков: «если монету подбросить, то она упадет и точно встанет на ребро, если, конечно, не зависнет в воздухе». Других ситуаций, по их мнению, быть не должно. «Просто не должно быть и все». И, поэтому, все другие ситуации их схемы не распознают и не обрабатывают. Однако бывает, что даже монеты падают по-другому. И вот тут начинаются сложности с отладкой. Можно сделать «железо» «нестрогим», то есть разрешить ему работать только под управлением программы от хоста, а для обнаружения ошибок в работе встроить в управляющую программу микропроцессора дополнительные модули. Таким образом, чтобы отладить одни программы, необходимо иметь другие отлаженные программы. Однако, по мнению автора, отследить циклограмму работы хоста, который, например, раз в полчаса передает неправильные пакеты, довольно трудоемко. Чтобы проконтролировать, успевает ли хост вовремя «подкачать» данные в FIFO, необходима специальная ветвь в программном обеспечении. Автор же предпочитает делать «строгую» аппаратуру, то есть такую аппаратуру, в которую встроены дополнительные узлы контроля, которые могут фиксировать ошибочные состояния, возникающие в работе узла, как на этапе отладки программ, так и в рабочем режиме. Поскольку современные FPGA обычно имеют одинаковые корпуса на всю серию микросхем, то есть возможность на этапе отладки использовать микросхему с большими ресурсами, и в них заложить аппаратные узлы контроля. Затем, при переходе к серийным изделиям, можно удалить избыточные узлы контроля. Примером реализации такого подхода к проектированию может служить, например, счетчик передаваемых слов в передающем FIFO. Если число слов, передаваемых хостом, не будет соответствовать тому числу, которое будет записано в служебный регистр перед началом передачи кадра, то блок контроля FIFO выдаст прерывание и запишет в контрольный регистр причину, вызвавшую сбой в работе. Такой же принцип применен и для контроля опустошения буфера по передаче. Если хост-процессор не успеет «подкачать» данные в буфер, то зафиксируется признак ошибки по передаче. Обобщая, можно сказать, что отладка «железа» превращается по большей части в отладку программ, часть из которых выполняется в хост-процессоре или в управляющем микроконтроллере, а другая часть этих программ представляет собой описание схем в FPGA. Поэтому время и затраты труда на отладку аппаратуры должны распределяться между этими программами равномерно.

Окончание следует.

Литература

1. А.Сигаев. Embedded Internet // «Компоненты и Технологии». 2000. № 2.
2. Н. Олифер, В. Олифер. Высокоскоростная технология Fast Ethernet (IEEE 802.3u). Центр Информационных Технологий (lxt972d.pdf).
3. Л. Куин, Р. Рассел. Fast Ehernet. BHV. 1998.