HDL-реализация сетевого контроллера Gigabit Ethernet

Алексей Гребенников (г. Актау, Казахстан)

Статья содержит описание сетевого контроллера Gigabit Ethernet на языке Verilog и его реализацию в среде Xilinx PlanAhead для отладочной платы SP605, содержащей ПЛИС семейства Spartan 6. Симуляция проекта проводилась в среде ModelSim.

Введение

Сетевой интерфейс Ethernet получил очень широкое распространение при построении локальных сетей. По мере развития технологии скорость передачи данных увеличивалась, и версия 1 Гб/с является самой распространённой на данный момент. Достоинствами этого интерфейса являются высокая скорость передачи данных, возможность связи между двумя любыми устройствами и простота построения сети.

Основные положения

Общие принципы сетевого взаимодействия описываются моделью OSI (Open Systems Interconnection), согласно которой существует семь уровней связи открытых систем. При этом сетевой контроллер работает на двух нижних уровнях – физическом (physical layer) и канальном (data link layer). Структурная схема контроллера изображена на рисунке 1. Физический уровень, как правило, выполняется в виде отдельной микросхемы (далее по тексту PHY), так как выполняет цифровые и аналоговые функции. На плате SP605 установлена микросхема PHY Alaska M88E1111. К сожалению, подробная документация на эту микросхему отсутствует в свободном доступе в Интернете, поэтому с принципами работы физического уровня можно ознакомиться на примере аналогичной микросхемы DP83865 [1] фирмы National Semiconductor.

Интерфейс PHY с передающей средой называется MDI (Media Dependent Interface) и, как следует из названия, зависит от выбранной среды передачи. На плате SP605 используется витая пара. Канальный уровень Ethernet MAC (далее по тексту MAC) реализован на ПЛИС как часть системы на кристалле. Интерфейс между PHY и MAC не зависит от среды передачи данных, но зависит от режима работы контроллера. На рисунке 1 обозначены основные типы этого интерфейса:

- MII (Media Independent Interface) – интерфейс передачи данных для скоростей 10 и 100 Мб/с. При этом режиме работы данные передаются полубайтами (nibble);
- GMII (Gigabit Media Independent Interface) – интерфейс передачи данных для скорости 1 Гб/с. Данные передаются байтами;
- RGMII (Reduced Gigabit Media Independent Interface) – в этом режиме работы данные передаются полубайтами, но на обоих фронтах синхросигнала.

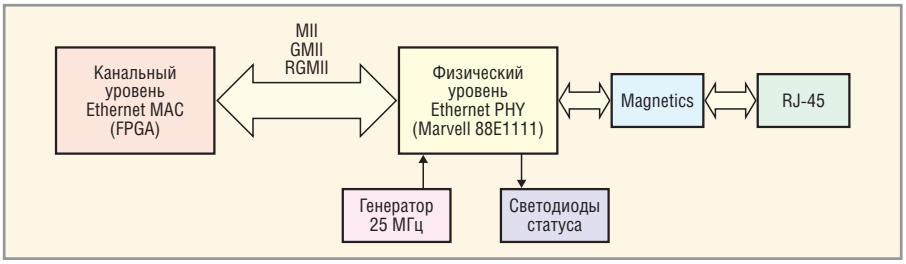


Рис. 1. Структура сетевого контроллера

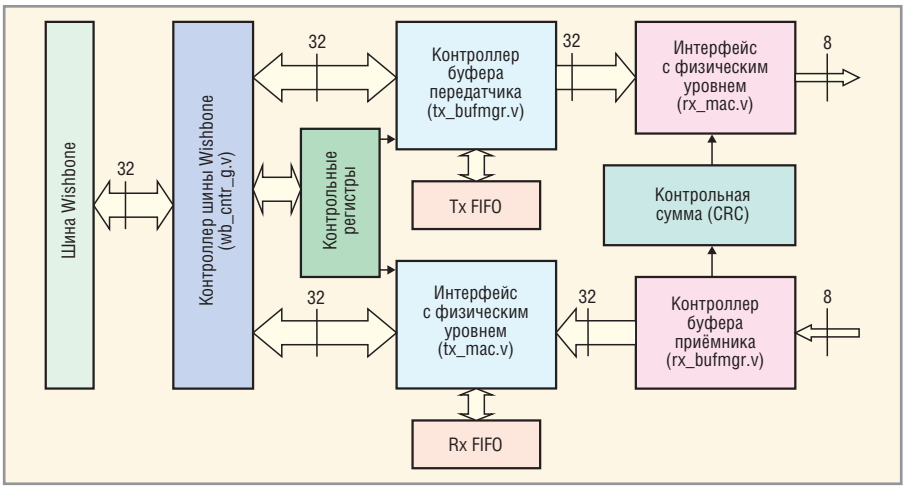


Рис. 2. Блок-схема канального уровня

Контроллер может обмениваться данными в полудуплексном и полнодуплексном режиме. В первом случае одновременно возможен только приём или передача. В случае одновременной активации нескольких передатчиков возникает коллизия. При полнодуплексном режиме коллизии исключены.

Версия контроллера, описанная в данной статье, поддерживает только полнодуплексный режим с интерфейсом GMII. Все исходные файлы проекта приведены в архиве *verilog_sources.zip* на сайте журнала. Рассмотрим более подробно работу канального уровня контроллера, реализованную на ПЛИС.

Структура канального уровня контроллера

За основу при построении модели был взят контроллер фирмы National Semiconductor DP83820 [2]. Блок-схема модели показана на рисунке 2. Обмен данными с внешней памятью производится через шину Wishbone, широко используемую при построении систем на кристалле. Подробная спецификация шины приведена в [3].

Управление схемой производится при помощи регистров, расположенных в файле *ops_regs.v*. Эти регистры находятся в адресном пространстве

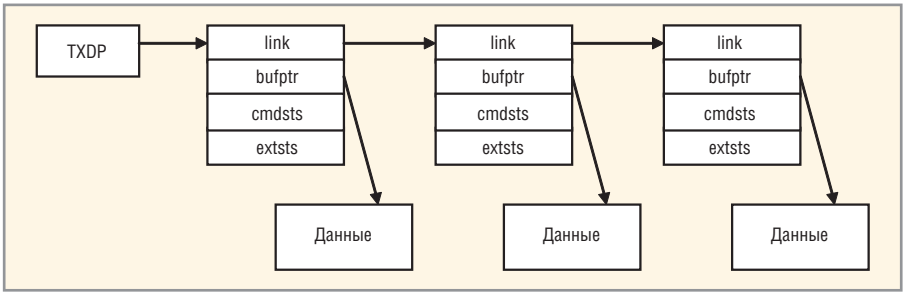


Рис. 3. Структура дескрипторов

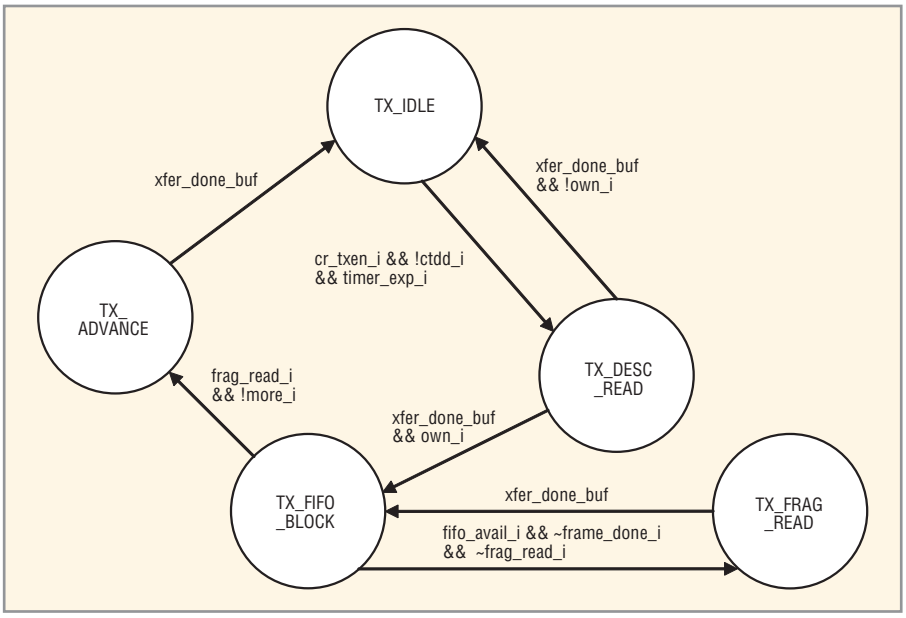


Рис. 4. Конечный автомат контроллера буфера передатчика

шины с базовым адресом *0xFFF80000*. Регистр *cr* содержит флаги *tx_en* и *rx_en*, разрешающие работу передатчика и приёмника соответственно. Передатчик считывает данные из внешней памяти в свою внутреннюю память типа FIFO и затем передаёт данные блоку интерфейса с физическим уровнем. Для проверки целостности данных вычисляется контрольная сумма CRC. Приёмник после получения данных от PHY записывает их во внутреннюю память FIFO и затем во внешнюю память после получения управления шиной Wishbone. В случае ошибки контрольной суммы пакет данных игнорируется.

Рассмотрим более подробно работу отдельных блоков канального уровня.

Контроллер буфера передатчика

Контроллер буфера передатчика состоит из двух частей – управляющей схемы и конечного автомата, расположенных в файлах *tx_bufmgr.v* и *tx_fsm.v* соответственно. Задачей контроллера является чтение данных из внешней памяти во внутреннюю память FIFO и последующая передача этих данных блоку интерфейса с физическим уровнем.

Организация буфера данных одинакова для приёмника и передатчика и изображена на рисунке 3. Каждый блок данных имеет свой дескриптор, состоящий их четырёх 32-битных слов:

- *link* – указатель на дескриптор следующего блока данных;
- *bufptr* – указатель на данные этого блока;
- *cmdsts* – слово состояния и команд данного блока;
- *extsts* – дополнительное слово состояния;

Один блок содержит один пакет данных.

Схема конечного автомата буфера передатчика изображена на рисунке 4. В начальный момент времени автомат находится в режиме ожидания (*TX_IDLE*). Следующее состояние контроллера – чтение дескриптора блока данных из внешней памяти, но для перехода в это состояние необходимо выполнить нескольких условий. Внешняя управляющая программа – драйвер – должна инициализировать указатель *txdp*, который является адресом первого слова дескриптора блока во внешней памяти. При этом флаг *ctdd* сбрасывается в нулевое значение, указывая на то, что загружен новый указатель. Затем драйвер устанавливает в единичное значение нулевой бит управляющего регистра *cr*. Этот бит разрешает работу передатчика, и в контроллере буфера он называется *cr_txen_i*. Затем по таймеру конечный автомат переходит в состояние чтения одного дескриптора *TX_DESC_READ*. Задержка введена для того, чтобы в режиме ожидания контроллер не перегружал шину Wishbone слишком частыми запросами на чтение.

В состоянии *TX_DESC_READ* контроллер считывает четыре 32-битных слова из внешней памяти. После окончания цикла чтения анализируется 31-й бит слова *cmdsts*, который в конечном автомате называется *own_i* и является признаком принадлежности дескриптора. Этот бит устанавливается в единицу производителем данных и в ноль – потребителем данных. Для дескрипторов передатчика производителем данных является драйвер, а контроллер является потребителем данных. Для дескрипторов приёмника – наоборот. Поэтому, как видно из рисунка 4, если бит *own_i* равен единице, значит, он был модифицирован драйвером и в данный момент принадлежит контроллеру.

Затем конечный автомат переходит в состояние подготовки чтения одного блока данных в память FIFO *TX_FIFO_BLOCK*. Если же бит *own_i* равен нулю, это значит, что дескриптор всё ещё принадлежит драйверу и не может быть использован контроллером. В этом случае автомат возвращается в состояние ожидания на время, заданное константой *TX_TIMER_EXP*, или 16'h0100 циклов тактового сигнала в данном примере. Циклы чтения дескриптора повторяются до тех пор, пока драйвер не установит в единицу 31-й бит слова *cmdsts*, т.е. бит *own_i*.

В состоянии конечного автомата *TX_FIFO_BLOCK* происходит подготовка к чтению блока данных. Во-первых, проверяется наличие свободного места в FIFO, т.е. равенство единице бита *fifo_avail_i*. Порог установки бита *fifo_avail_i* конфигурируется битами 8–15 регистра *txcfg*, который в блоке контроллера называется *flth* (fill threshold). Этот регистр находится в файле регистров по глобальному адресу шины Wishbone (*WB_BASE + 8'h28*). Биты *frame_done_i* и *frag_read_i* устанавли-

ваются в нулевое значение перед началом цикла чтения памяти. Полный цикл чтения пакета данных может включать в себя несколько переходов состояний *TX_FIFO_BLOCK* – *TX_FRAG_READ*. После считывания полного пакета данных бит *frame_done_i* устанавливается в единичное значение.

Во время чтения данных в состоянии *TX_FRAG_READ* отслеживается уровень заполнения FIFO. При достижении порога *drth* (drain threshold) или после считывания полного пакета данных, если его размер меньше порогового значения, устанавливается бит *tx_start_o*, разрешающий блоку интерфейса с физическим уровнем начать передачу данных. После окончания передачи блока бит *frag_read_i* устанавливается в единичное значение, и конечный автомат переходит в следующее состояние – *TX_ADVANCE*.

Заметим, что изображённый на рисунке 4 бит *more_i* в данной реализации контроллера всегда равен нулю. Этот бит используется при более сложной организации буферной памяти, когда один пакет данных размещается в нескольких дескрипторах.

В состоянии *TX_ADVANCE* контроллер обновляет значение дескриптора в основной памяти, обнуляя бит *own_i*, и передаёт дескриптор драйверу. В случае успешной передачи пакета данных устанавливает 27-й бит слова *cmdsts* – признак успешности передачи. Затем проверяется значение поля *link*. Если оно равно нулю, значит, дескриптор был последним в цепочке, изображённой на рисунке 3. В этом случае конечный автомат переходит в состояние *TX_IDLE* и ждёт обновления указателя *txdp* драйвером, т.е. момента сброса бита ctdd_i. Если же поле link не равно нулю, автомат загружает это значение в регистр *txdp*, сбрасывает бит *ctdd_i*, устанавливает бит *timer_exp_i* и переходит в состояние *TX_IDLE*. Однако в этом случае сразу же осуществляется переход в состояние *TX_DESC_READ* (если в момент перехода драйвер не обнулил бит разрешения передачи *cr_txen_i*). Таким образом, организация связанных буферов данных позволяет драйверу инициализировать сразу несколько пакетов данных и не вмешиваться в процесс передачи до полного его окончания.

Контроллер буфера передатчика работает в тесном взаимодействии с блоком интерфейса с физическим уровнем. Рассмотрим более подробно работу этого блока.

Интерфейс передатчика с физическим уровнем

Модуль этого интерфейса расположен в файле *tx_mac.v*. К этому модулю относится и блок подсчёта контрольной суммы *crc_gen_tx.v*. Основой интерфейса является конечный автомат, изображённый на рисунке 5. В начальный момент времени автомат находится в режиме ожидания *TXMAC_IDLE*. После получения сигнала *tx_start_i* от контроллера буфера автомат переходит в состояние *TXMAC_START*, т.е. начинает передачу данных.

В состоянии *TXMAC_START* проверяется режим передачи данных и готовность среды. Данный контроллер поддерживает только полнодуплексный режим передачи на скорости 1 Гб/с. При этом коллизии в линии отсутствуют; *phy_crs_i* (carrier sense) – сигнал от PHY о готовности среды передачи.

Передача пакета данных Ethernet начинается в состоянии *TXMAC_PRE-*

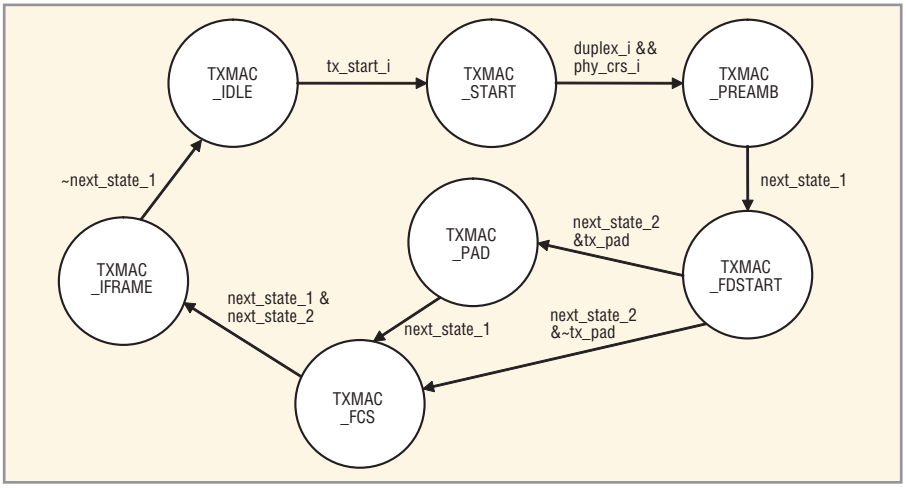


Рис. 5. Конечный автомат интерфейса передатчика с физическим уровнем

Преамбула	7 байт
Разделитель (SFD)	1 байт
Адрес приёмника	6 байт
Адрес передатчика	6 байт
Длина/тип	2 байта
Данные	46-1500 байт
Дополнение (Pad)	
Контрольная сумма	4 байта

Рис. 6. Структура пакета IEEE 802.3

AMB. Основой интерфейса Ethernet является стандарт IEEE 802.3. Последняя редакция стандарта вышла в 2008 г., и его подробное описание приведено в [4]. Согласно этому стандарту, данные передаются пакетами, структура которых изображена на рисунке 6. Преамбула, разделитель, дополнение и контрольная сумма – это служебные данные, генерируемые непосредственно модулем *tx_mac*. Всё остальное генерируется протоколами более высокого уровня и просто инкапсулируется в пакет Ethernet. В состоянии конечного автомата *TXMAC_PREAMB* передаётся преамбула, которая состоит из семи байтов со значением 8'h55, т.е. чередованием единиц и нулей. После преамбулы передаётся разделительный байт 8'hD5. Этот байт также представляет собой последовательность единиц и нулей, но заканчивается двумя единицами. Преамбула и SFD необходимы для синхронизации передатчика и приёмника.

После этого автомат переходит в состояние *TXMAC_FDSTART*. В этом состоянии происходит передача полезной информации, полученной от контроллера буфера, на скорости 1 Гб/с в полнодуплексном режиме. Особенностью этого режима является то, что данные передаются по 8 бит на частоте 125 МГц, причём частота для PHY генерируется на канальном уровне. Заметим, что в режиме приёма частота 125 МГц для канального уровня генерируется в PHY.

Полезные данные, как видно из рисунка 6, также структурированы. Первые шесть байт – это MAC-адрес приёмника, следующие шесть байт – MAC-адрес передатчика. После этого два байта – длина/тип*. Далее идут данные протоколов более высокого уровня. Если длина пакета меньше 46 байт, он дополняется до этого минимального размера байтом 8'hFF.

С момента передачи первого байта значимых данных, т.е. адреса приёмника, включается модуль подсчёта контрольной суммы. Подробно алгоритм подсчёта контрольной суммы описан в стандарте IEEE 802.3-2008, а также в других источниках в Интернете. После окончания передачи значимых данных передаётся контрольная сумма в состоянии *TXMAC_FCS*, гарантирующая целостность данных.

После окончания передачи пакета необходимо выдержать минимальный промежуток времени до начала передачи следующего пакета. Это время равно времени передачи 96 бит данных и выдерживается в состоянии конечного автомата *TXMAC_IFRAME*. После этого автомат снова переходит в состояние ожидания *TXMAC_IDLE*.

Фрагмент симуляции работы передатчика изображён на рисунке 7. Контроллер буфера считывает данные из основной памяти в состоянии *tx_fsm_state* = *TX_FRAG_READ*. После окончания процесса чтения конечный автомат *tx_fsm* переходит в состояние *TX_FIFO_BLOCK* и активирует линию *tx_start_i*. При получении этого сигнала конечный автомат *txmac_fsm* выходит из режима ожидания и начинает передачу пакета данных. В момент передачи первого байта преамбулы активируется линия *tx_en_o*, которая включает передатчик в PHY. В схеме есть две тактовые частоты – *clk_i* (30 МГц), основная частота контроллера буфера, и *tx_clk_i* (125 МГц), частота блока *tx_mac*.

Приёмная часть контроллера

Приёмная и передающая части контроллера во многом схожи. Приёмник, как это видно из рисунка 2, состоит из блока интерфейса с физическим уровнем, расположенным в файле *rx_mac.v*, и контроллера буфера приёмника (файлы *rx_bufmgr.v* и *rx_fsm.v*). К блоку *rx_mac.v* также относится модуль контроля чётности *crc_gen.v*. Алгоритм работы передающего и приёмного модулей контрольной суммы отличается только временем выборки сигнала. Модуль *crc_gen* обрабатывает данные на отрицательном фронте синхросигнала, а модуль *crc_gen_tx* – на положительном. Приёмник может принимать все пакеты или только те, у которых MAC-адрес приёмника совпадает с адресом контроллера. Этот параметр контролируется регистром *rfcr*.

Преамбула и SFD принимаются, но не копируются в память FIFO. После начала приёма значимых данных начинается копирование 32-битных слов в память FIFO. Однако до получения контрольной суммы, которая передаётся последним словом пакета, неизвестно, достоверны данные или нет. По окончании приёма пакета полученная контрольная сумма проверяется на основе вычисленной. В случае их равенства пакет фиксируется как достоверный. В противном случае бит достоверности сбрасывается:

**В ранних редакциях стандарта IEEE 802.3 это поле определялось как длина, тогда как в стандарте Ethernet это поле всегда обозначало тип. В стандарте IEEE 802.3-2008 это поле называется длина/тип.*

```
if (crc_value != crc_val_acq)
data_valid[psize_wr[4:0]] <=
1'b0;
else
data_valid[psize_wr[4:0]] <=
1'b1;
```

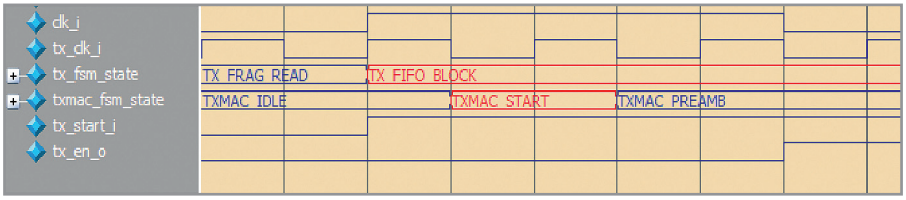


Рис. 7. Фрагмент симуляции работы передатчика

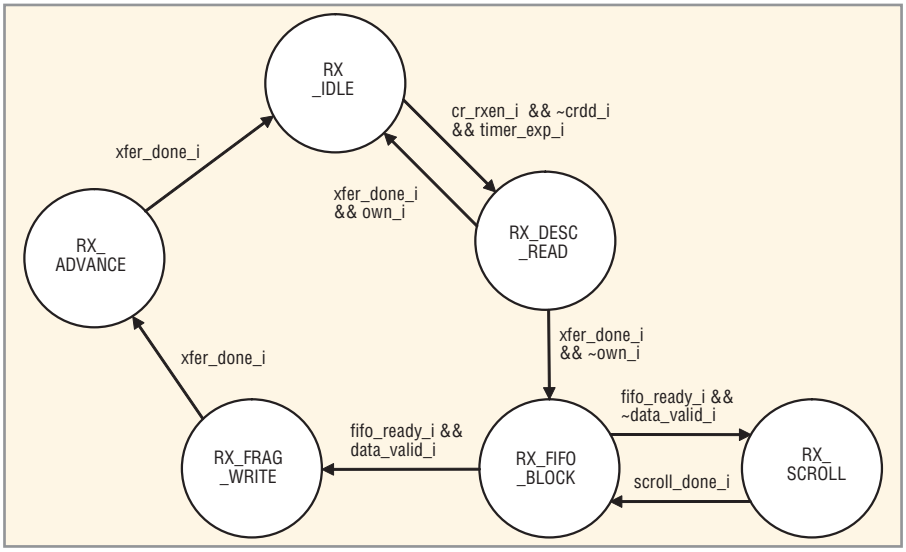


Рис. 8. Конечный автомат контроллера буфера приёмника

Приёмник, так же как и передатчик, содержит два конечных автомата – *rxmac_fsm* и *rx_fsm*. Рассмотрим более подробно работу автомата блока контроллера буфера приёмника, схема которого изображена на рисунке 8.

В начальный момент времени автомат находится в режиме ожидания *RX_IDLE*. Для его перехода в активное состояние требуется инициализация со стороны драйвера. Организация памяти приёмника такая же, как и у передатчика (см. рис. 3). Драйвер сначала инициализирует указатель дескриптора *rxdp*, тем самым сбрасывая бит *crdd* в нулевое значение. После этого драйвер устанавливает бит разрешения приёма в регистре *cr* и, по истечении времени таймера, конечный автомат переходит в состояние чтения одного дескриптора из основной памяти *RX_DESC_READ*. После окончания цикла чтения – установки бита *xfer_done_i* в единичное состояние – проверяется бит *own_i* дескриптора. Поскольку для приёмника производителем данных является сам контроллер, в соответствии с описанной выше логикой при-

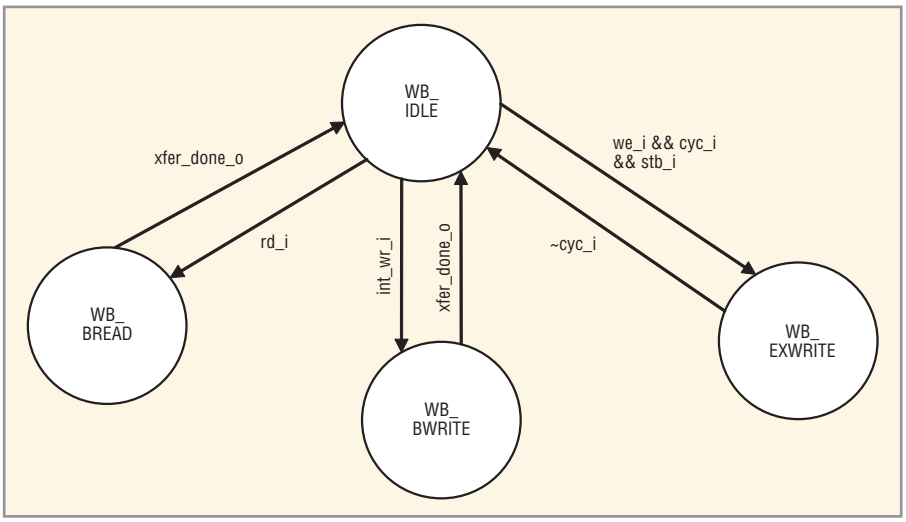


Рис. 9. Конечный автомат контроллера шины Wishbone

надлежности дескрипторов, бит *own_i* у дескриптора, принадлежащего контроллеру, должен быть равен нулю. В противном случае автомат возвращается в режим ожидания и затем периодически, по истечении времени таймера, считывает дескриптор из внешней памяти.

В случае, если бит *own_i* равен нулю, автомат переходит в следующее состояние – *RX_FIFO_BLOCK*. В этом состоянии автомат ожидает окончания приёма пакета блоком интерфейса с физическим уровнем, который записывает данные в память FIFO. При совпадении вычисленной и полученной контрольных сумм автомат переходит в состояние *RX_FRAG_WRITE* и копирует данные из FIFO в адресное пространство внешней памяти, указанное в текущем дескрипторе. Если контрольные суммы не совпадают, блок данных не записывается во внешнюю память. Однако этот блок данных уже содержится в памяти FIFO, поэтому конечный автомат переходит в состояние *RX_SCROLL*, в котором происходит увеличение адреса чтения на размер принятого пакета.

Приёмник может хранить в памяти до 32 пакетов данных, прежде чем они будут скопированы во внешнюю память. Для этого в модуле *rx_mac* организован массив 16-битных слов, хранящий размер принятого пакета, и массив битов достоверности данных:

```
reg [15:0] packet_size [0:31];
reg data_valid [0:31];
```

Доступ к этому массиву осуществляется по принципу памяти FIFO, т.е. имеется адрес для чтения и для записи. При получении пакета инкрементируется адрес записи, а при копировании пакета во внешнюю память инкрементируется адрес чтения.

После успешного копирования пакета данных во внешнюю память конечный автомат переходит в состояние *RX_ADVANCE*. В этом состоянии происходит обновление текущего дескриптора – бита принадлежности *own_i*, бита успешности приёма пакета и поля размера принятого пакета, куда записывается фактический размер принятого пакета данных в байтах. В этом состоянии также проверяется поле дескриптора *link*. Если это поле равно нулю, автомат переходит в состояние *RX_IDLE* и ждёт дальнейшего обновления указателя дескриптора *rxdp* драйвером. Если же поле *link* не равно нулю, то это ненулевое значение копируется в регистр *rxdp*, сбрасывается бит *crdd_i* и автомат также переходит в состояние ожидания. Но в связи с тем, что указатель дескриптора уже был обновлён, сразу же происходит переход в состояние *RX_DESC_READ*, в котором копируется новый дескриптор. Таким образом, драйвер может инициализировать массив связанных дескрипторов и затем блоками считывать данные из внешнего буфера.

Как упоминалось выше, обмен данными с внешней памятью происходит через шину Wishbone. Рассмотрим более подробно работу контроллера этой шины.

Контроллер шины Wishbone

Модуль контроллера шины расположен в файле *wb_cntr_g.v*. Этот модуль имеет в своём составе конечный автомат, изображённый на рисунке 9. В состоянии *WB_IDLE* автомат находится в режиме ожидания; *WB_BREAD* – цикл чтения данных, инициированный модулями *rx_bufmgr* или *tx_bufmgr*; *WB_BWRITE* – цикл записи данных, инициированный модулями *rx_bufmgr* или *tx_bufmgr*. Эти два состояния используются для чтения/записи дескрипторов и данных; *WB_EXWRITE* – запись данных в адресное пространство контроллера внешними устройствами. В этом состоянии внешний драйвер инициализирует управляющие регистры контроллера, которые находятся в адресном пространстве шины Wishbone. Обмен данными осуществляется 32-битными словами.

Система на кристалле

Поскольку сетевой контроллер обменивается данными с внешней памятью и требует инициализации, для тестирования его работы требуется ряд вспомогательных устройств. На рисунке 10 изображена блок-схема системы на кристалле с указанием верхних модулей всех устройств. Верхний модуль системы на кристалле находится в файле *sopc.v*.

Входной дифференциальный тактовый сигнал 200 МГц преобразуется в две частоты – 30 и 125 МГц. Частота 125 МГц используется в цепях передатчика в блоке интерфейса с физическим уровнем (*tx_mac*) и также является тактовым сигналом для PHY. Для передачи тактовой частоты на внешний вывод ПЛИС (clock forwarding) используется примитив *ODDR2*, шаблон которого имеется в среде PlanAhead. Приёмная цепь интерфейса физического уровня (*rx_mac*) тактируется внешней частотой 125 МГц, генерируемой PHY. Все остальные модули работают на частоте 30 МГц.

Контроллер памяти обеспечивает чтение/запись памяти через шину Wishbone и также имеет встроенную память, реализованную на внутренних блоках памяти ПЛИС. Этого объёма вполне достаточно для тестирования сетевого контроллера.

Задачей контроллера шины Wishbone является декодирование адреса и мультиплексирование шины соответствующим образом. Сетевой контроллер может обмениваться данными с памятью в режиме чтения и записи. Контроллер UART может считывать данные из памяти, а также записывать данные как в память, так и в регистры сетевого контроллера, в зависимости от адреса. При построении системы использовался UART, описанный в [5].

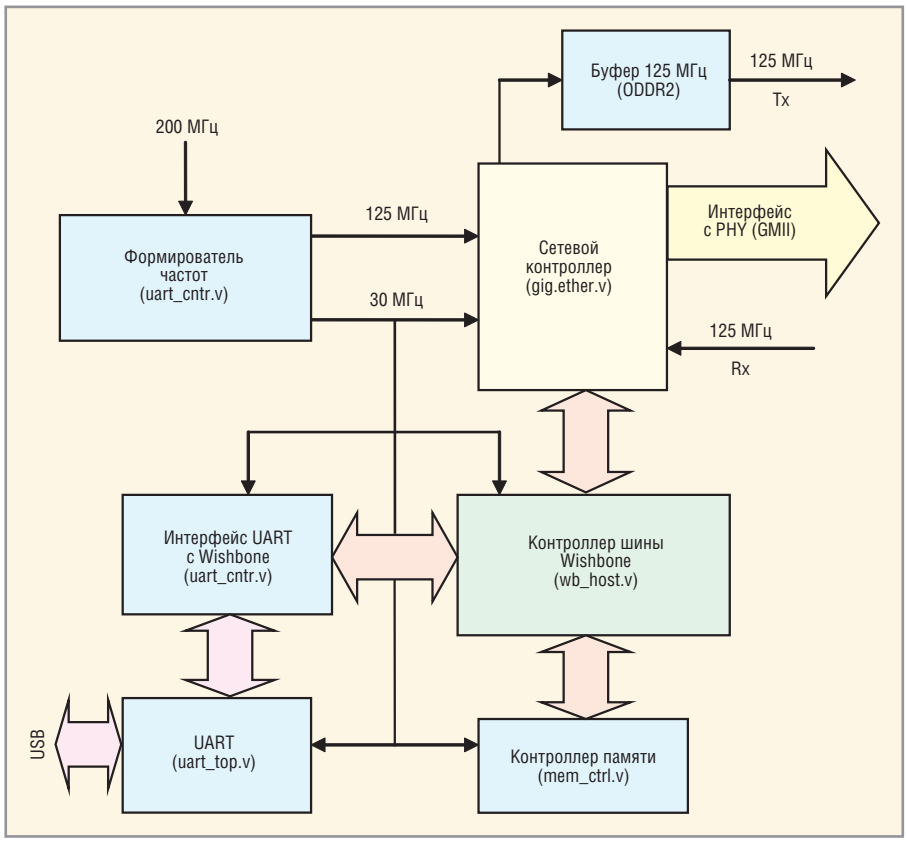


Рис. 10. Блок-схема системы на кристалле

Поэтому потребовался дополнительный модуль для обеспечения обмена данными UART через шину Wishbone. Контроллер UART декодирует несколько команд, формат которых описан в файле *uart_cntr.v*.

Для проверки работоспособности сетевого контроллера использовалась программа Wireshark и программа связи с UART, исходные коды которой содержатся в архиве *XilinxCOM.zip*, а также в приложении к статье [5]. Во время тестирования плата SP605 подключалась к компьютеру через интерфейс Gigabit Ethernet и два разъёма USB. Один разъём USB – это JTAG, второй разъём USB – мост USB-UART. Данные, передаваемые платой SP605, контролируются программой Wireshark, а данные, передаваемые компьютером, фиксируются программой XilinxCOM.

В начальный момент времени UART инициализирует указатели *rxdp* и *txdp* в модулях контроллера *rx_bufmgr* и *tx_bufmgr* соответственно. Затем в основной памяти инициализируются дескрипторы для буферов чтения и записи. После этого UART разрешает работу приёмника/передатчика путём записи соответствующего значения в регистр *cr*. Затем данные, переданные контроллером (т.е. блок данных, записанный контроллером UART в область памяти, заданную дескриптором передатчика), фиксируются программой Wireshark. И, соответственно, данные, переданные компьютером, могут быть считаны через UART-область памяти, заданной дескриптором приёмника.

Заключение

Модель сетевого контроллера, описанная в статье, предназначена для образовательных целей и позволяет изучить как основы построения подобных устройств, так и принципы работы канального и физического уровней модели OSI. Сетевой контроллер был синтезирован для ПЛИС Spartan 6 в составе отладочной платы SP605 и показал удовлетворительные результаты при тестировании.

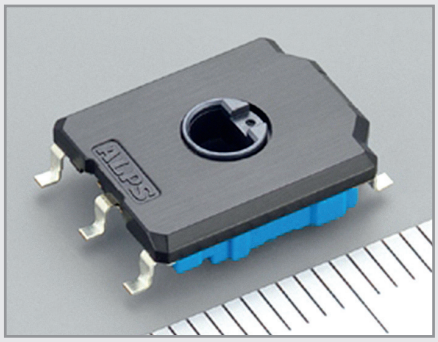
Литература

1. DP83865 Gig PHYTER V 10/100/1000 Ethernet Physical Layer.
2. DP83820 10/100/1000 Mb/s PCI Ethernet Network Interface Controller.
3. WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores. www.opencores.org.
4. IEEE Std 802.3-2008. Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specification.
5. *Гребенников А.* HDL-реализация асинхронного приёмопередатчика. Современная электроника. 2011. № 4.

Новости мира

Компактный датчик угловых перемещений

Компания Alps Electric Europe выпускает RDC90, резистивный датчик угловых перемещений с полым валом. Размеры элемента 11,0 × 13,7 × 3,2 мм. Он отличается большой долговечностью (до 100 миллионов циклов) и микролинейностью ±0,1%. Среди остальных технических параметров общее сопротивление 3,3 кОм ± 30%, абсолютная линейность ±2,2% (опорная точка 2,5 В) и эффективный электрический угол ±40°. Типичными областями применения являются джойстики для радиоуправления, измерение углов отклонения шарниров роботов, измерение углов и положений в коммерческих приложениях.



www.alps.com

Микросхема управления светодиодами

Компания International Rectifier выпускает микросхему импульсного источника питания IRS2548D. Чип, в котором объединены схема коррекции коэффициента мощности и полумостовой драйвер, имеет, по утверждению производителя, КПД более 88% при нагрузке 40 В/1,3 А. Элемент имеет функцию широтно-импульсного снижения яркости менее чем до 2% световой отдачи и целый ряд функций защиты, в том числе PFC- и полумостовую защиту от перегрузок, а также защиту от электростатических разрядов. Кроме того IRS2548D содержит также перестраиваемый генератор, внутренний MOSFET, внутренний ограничительный стабилитрон 15,6 В на питание.

www.irf.com