О Т Ч Е Т
О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

по теме: ***«*ИССЛЕДОВАНИЕ И СОЗДАНИЕ СИСТЕМНОГО АППАРАТНО-ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПЕРЕДАЧИ ДАННЫХ ПО ЛИНИЯМ ROCKETIO ДЛЯ СИСТЕМ НА КРИСТАЛЛАХ ПЛИС ФИРМЫ XILINX*»***

Научный руководитель **С.С. Андреев**

Москва 2007

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современный уровень элементной базы ПЛИС позволяет создавать системы на кристалле, сочетающие в себе архитектурные решения традиционных микропроцессорных систем и, в то же время, специализированных устройств обработки данных. Так в ПЛИС фирмы Xilinx, серий Virtex-2Pro и Virtex-4, присутствуют погруженные в кристалл микропроцессорные ядра PowerPC 405, которые позволяют создавать системы на кристалле (СнК) с поддержкой множества различных широко распространенных периферийных устройств. Это дает возможность разрабатывать системы из многих FPGA с использованием подходов, хорошо зарекомендовавших себя при построении вычислительных кластеров из универсальных рабочих станций. Каждый из кристаллов FPGA выступает при этом в роли узла вычислительного кластера.

Интеграция многих FPGA в единую вычислительную систему по хорошо известным кластерным технологиям имеет как достоинства, так и недостатки. К недостаткам относятся довольно скромные показатели производительности используемой при этом коммуникационной сети Ethernet, то есть не очень высокая пропускная способность (100 МБ/с) и весьма высокая латентность передачи данных (60 мкс) между различными FPGA .

При построении вычислительных кластеров общего назначения сходная проблема решается путем оснащения кластера дополнительной сетью высокой производительности, например, Myrinet или Infiniband. Для случая узлов кластера, оснащенных специализированной вычислительной мощностью, необходимость в дополнительной сети высокой производительности еще более очевидна.

Специализированные аппаратные ядра приемопередатчиков RocketIO входят в состав рассматриваемых FPGA и вполне пригодны для реализации адаптеров коммуникационных сетей Myrinet [7] и Infiniband [8]. Однако, буквальное повторение одной из этих коммуникационных технологий в FPGA вряд ли целесообразно. Согласно сложившейся многолетней практике кластерных технологий, программные средства работы с коммуникационными сетями высокой производительности довольно слабо интегрированы в ОС вычислительного узла. Работа с оборудованием таких сетей происходит посредством нестандартных, только этим сетям присущих, библиотек доступа. Единого стандарта, такого, как Ethernet для tcp/ip, аппаратная реализация которого значительно повышала бы совместимость сети высокой производительности с каким-либо общеупотребительным программным обеспечением, в любом случае, не существует. По этой причине реализация на базе RocketIO собственной сети высокой производительности ничем не хуже, а во многом и лучше, чем прямое повторение Myrinet или Infiniband. Сеть собственной разработки можно постараться оптимизировать по сравнению с более универсальными решениями, как по объему используемого оборудования, так и по латентности коммуникаций. В данном проекте латентность передачи

сообщения составляет 0,4 мкс (у Myrinet 6 мкс при примерно одинаковых показателях предельной пропускной способности - более 220МБ/с).

Роль этой сети при построении кластеров из СнК на базе FPGA та же, что и роль сетей, подобных Myrinet, при построении кластеров из рабочих станций общего назначения.

Первым шагом к разработке такой сети высокой производительности на базе RocketIO является рассматриваемое в настоящем документе IP core rocketmem.

Целью работы является исследование и создание системного аппаратно-программного обеспечения (ПО) передачи данных по линиям RocketIO для системы на кристалле (СнК) ПЛИС платы RMB-411. Созданное аппаратно-программное обеспечение обеспечивает взаимодействие программных приложений СнК плат RMB-411, соединенных между собой кабельными линиями при помощи интерфейсов RocketIO.

В состав комплексного решения входят: IP core контроллера линка RocketIO, рассчитанное на применение протокола Aurora и шины PLB, программные тесты разработанного контроллера для standalone OS Xilinx XPS, и решение по использованию разработанного IP core в рамках Xilinx XPS для СнК, создаваемых в ПЛИС платы RMB-411.

По результатам НИР подготовлен отчет, состоящий из трех разделов.

В первом разделе приводятся общие сведения о реализованном устройстве, кратко описывается программистская модель. Также описана тестовая конфигурация СнК ПЛИС RMB-411, необходимая и достаточная для проверки функционирования IP core контроллера линка. Приведена структура и основные модули устройства rocketmem.

Во втором разделе приводятся сведения о порядке реализации устройства штатными средствами XPS. Перечислены основные стандартные компоненты и технологии, использованные в процессе реализации.

В третьем разделе разобраны основные архитектурные ограничения реализации, то есть показано, в какой мере и как именно совершенствование функций устройства потребует переработки его внутреннего строения. Приводятся параметры производительности, достигнутые в текущем варианте реализации.

По результатам НИР сделаны выводы, которые приведены в Заключении.

1. Общие сведения.

Устройство rocketmem представляет собой устройство управления линком на базе RoketIO и предназначено для передачи данных между компьютерами, соединенными друг с другом такими линками. Настоящая версия реализует одноканальный линк.

1.1. Тестовая конфигурация СнК.

Для отладки и тестирования устройства была создана тестовая конфигурация СнК, представленная ниже. Эта конфигурация не является примером типовой конфигурации, в которой устройство будет применяться на практике. Это – намеренно упрощенная до предела тестовая платформа, предназначенная лишь для демонстрации работоспособности устройства, тестирования и измерения производительности.

СнК включает в себя два устройства rocketmem, связанных кабелем loopback для передачи данных друг другу. Это позволяет, в частности, тестировать одновременную передачу и прием данных одним и тем же устройством. В реальной системе кабелем будут соединяться устройства разных СнК.

Одно из устройств тестовой СнК подключено к входу запроса некритического прерывания процессора и, тем самым, способно вызывать прерывания. Это позволяет тестировать логику запроса прерываний. В реальной системе выход запроса прерывания устройства rocketmem будет подключаться не непосредственно к входу запроса прерывания процессора, а к входу запроса прерывания некоторого контроллера прерываний.

При тестировании передачи данных по каналам DMA требуется наличие адресуемого на PLB диапазона памяти, не являющегося локальным для устройства rocketmem. В тестовой СнК в качестве такого диапазона памяти используется буферный блок памяти другого устройства rocketmem. В реальной системе, скорее всего, в этом качестве будет выступать внешняя по отношению к кристаллу оперативная память, подключенная к PLB собственным контроллером, или же PLB BRAM, которая в тестовой СнК не предусмотрена.

1.1. Структура и основные модули тестовой конфигурации системы на кристалле платы RMB-411.

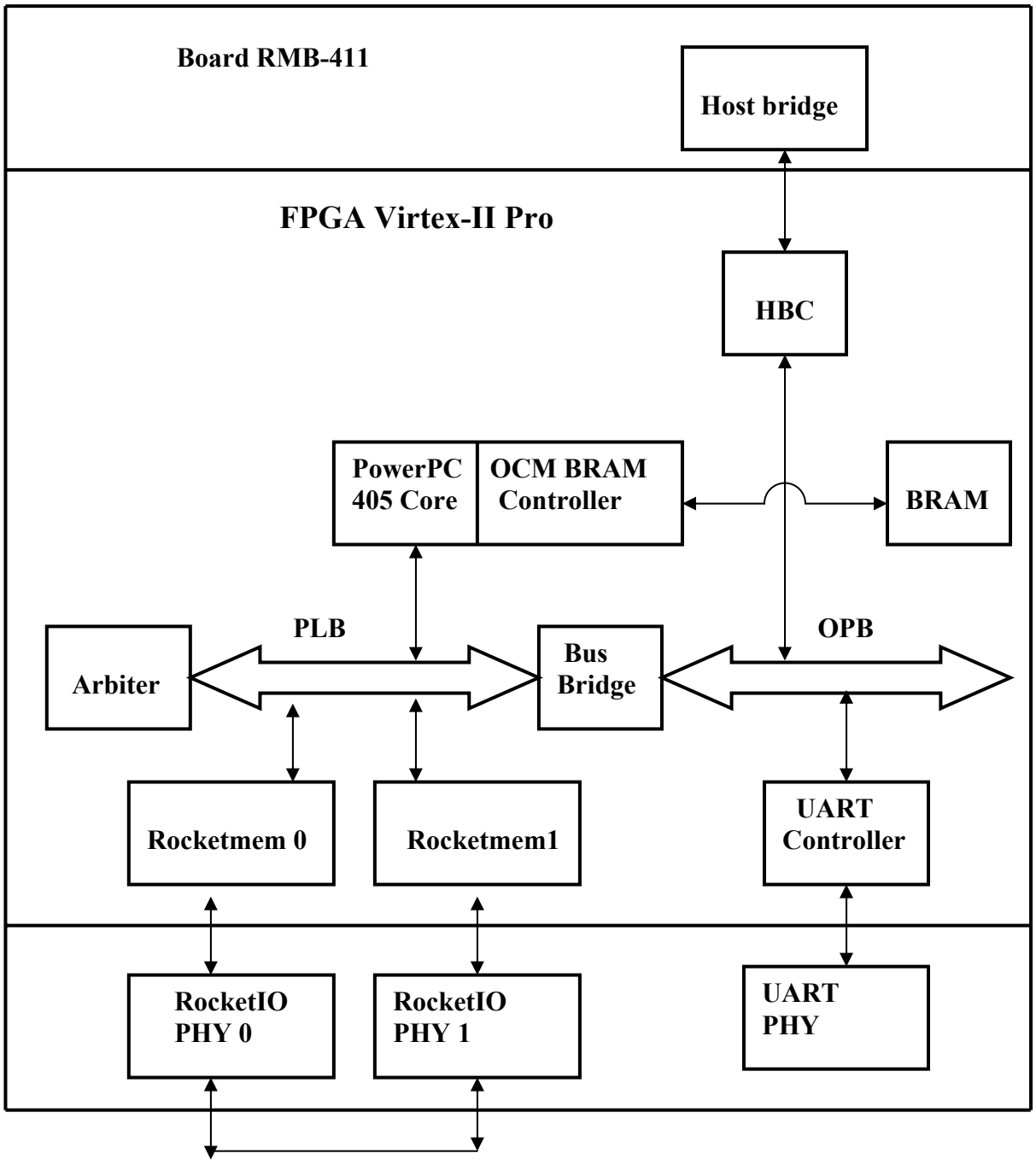


Рисунок 1.1 Тестовая конфигурация системы на кристалле RMB-411

Структура и основные модули тестовой конфигурации системы на кристалле RMB-411, приведены на рисунке 1.1. Где приняты следующие обозначения:

BOARD RMB-411 – Плата RMB-411;

FPGA Virtex-II Pro – ПЛИС Virtex-II Pro;

PowerPC 405 Core – ядро процессора PowerPC 405 внутри ПЛИС;

BRAM – блочная память ПЛИС;

OCM BRAM Controller – контроллер, подключаемый к интерфейсу OCM процессора и обеспечивающий доступ к блочной памяти ПЛИС Virtex-II Pro;

PLB (Processor Local Bus) – локальная шина процессора;

OPB (On-Chip Peripheral Bus) – шина периферийных устройств;

Arbiter – арбитр шины PLB или OPB;

Bus Bridge – мост между шиной PLB и OPB;

Host bridge – мост с host-машиной;

HBC (Host bridge controller) – контроллер моста с host-машиной;

UART универсальный асинхронный приемопередатчик;

UART Controller – контроллер UART;

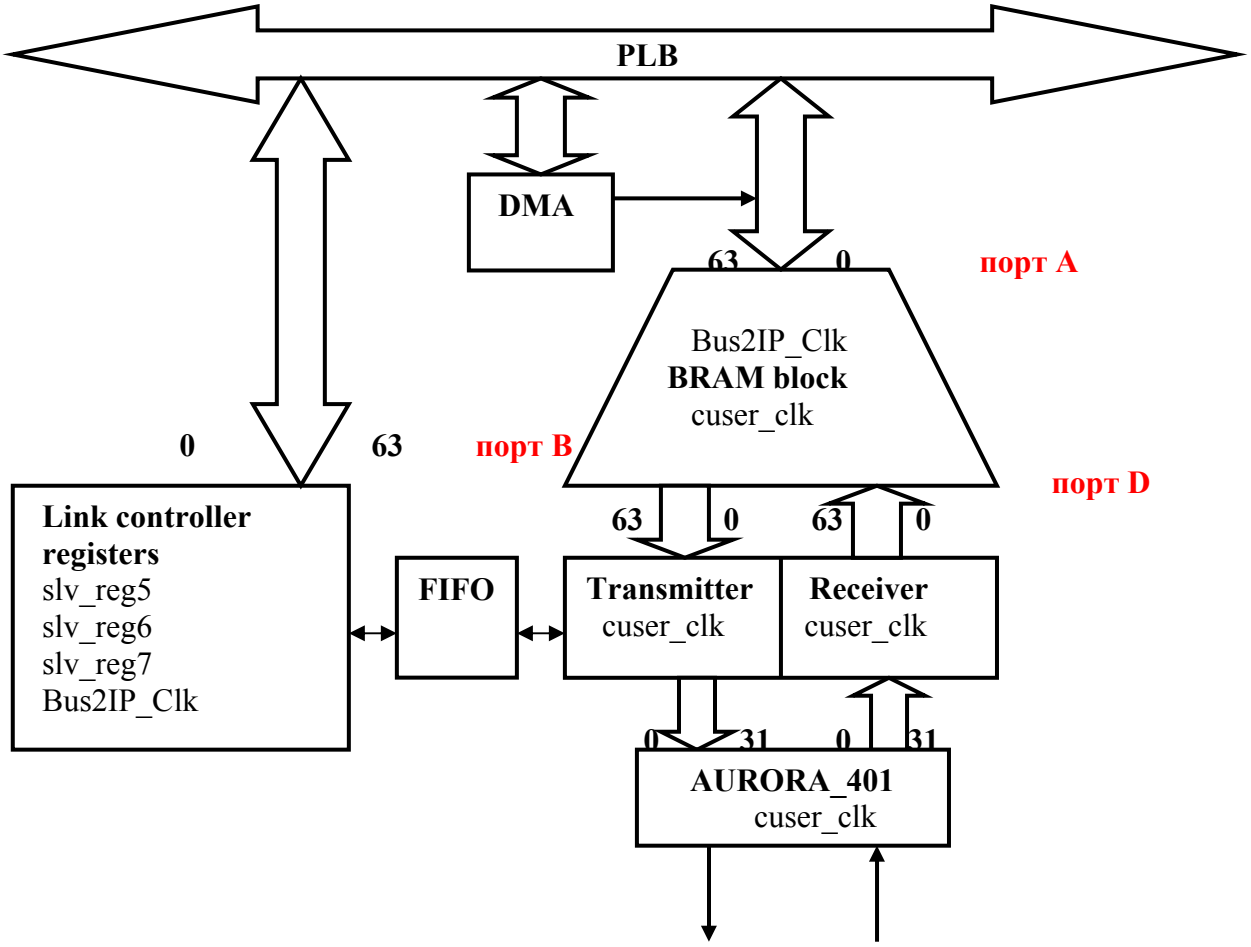
UART PHY – порт UART:

Rocketmem 0, Rocketmem 1 – устройства обмена данными по линку на базе RocketIO:

RocketIO PHY 0, RocketIO PHY 1 – порт RocketIO

1.2. Общая структура и принципы функционирования устройства.

Устройство включает в себя буферный блок памяти (BRAM block), контроллер линка (Link controller), контроллер прерываний (не показан) и контроллер DMA.



Передача данных в линк происходит из буферного блока памяти устройства. Для запуска передачи сообщения в линк передающий процессор должен выполнить некоторую последовательность действий с регистрами контроллера линка. Обмен происходит без участия процессора. Завершение обмена сигнализируется соответствующим битом в одном из регистров контроллера. Инициатор обмена (передатчик) указывает при запуске обмена как адрес и длину передаваемых данных, так и адрес, начиная с которого данные надо разместить в памяти приемника. **Прием данных из линка** происходит в буферный блок памяти устройства. Для приема данных из линка процессор приемника ничего делать не должен – данные сами разместятся в памяти, начиная с адреса, указанного инициатором обмена (передатчиком). Обмен происходит без участия процессора. При завершении приема сообщения на приемной стороне возникает некритическое прерывание. Блок памяти устройства имеет размер 64К байт и адресуется на PLB.

Для эффективного копирования данных между буферным блоком памяти устройства и любым адресуемым на шине PLB диапазоном адресов, не относящимся к данному устройству, в состав устройства включаются два контроллера DMA. Эти контроллеры не участвуют непосредственно в обмене сообщениями через линк. Их основное назначение – обеспечить возможность обмена данными между буферным блоком памяти и другими запоминающими

устройствами в составе системы со скоростями, значительно превосходящими скорость переписи данных в программном цикле.

Конкретный формат доступных программисту регистров вместе с примерами их использования описан в Приложении 1 (Руководство программиста).

2. *Использованные стандартные компоненты и технологии*.

Устройство создавалось стандартными средствами XPS, при помощи диалога “Create – Import Peripheral”, то есть включает PLB IPIF [1]. Шина PLB была выбрана как наиболее быстрая. При создании устройства были заказаны следующие компоненты: 8 slave-регистров, Reset/MIR, Simple DMA, одна линия запроса прерывания, один дополнительный адресный диапазон памяти. Slave-регистры использовались для реализации регистров контроллера линка, дополнительный адресный диапазон – для реализации буферного блока памяти на базе BRAM. При этом получаются два файла с исходным текстом – rocketmem.vhd и user_logic.vhd. В файле rocketmem.vhd потребовалось внести два изменения значений констант:

constant DEV_BURST_PAGE_SIZE : integer := 65536; - задание размера максимальной порции данных, передаваемой за одно обращение к DMA, и

```
constant DMA_LENGTH_WIDTH_ARRAY : INTEGER_ARRAY_TYPE :=
 (
  0 => 17,   -- dma channel 0 transfer byte counter width
  1 => 17    -- dma channel 1 transfer byte counter width
 );
```

- задание ширины счетчиков DMA в битах. Также потребовалось добавить внешние порты:

```
CH_RXP:  in   std_logic;
CH_RXN:  in   std_logic;
CH_TXP:  out  std_logic;
CH_TXN:  out  std_logic;
AURORA_REFCLK  : in  std_logic;
```

для внешних контактов дифференциальной линии rocketio и для выхода опорного таймера, тактирующего контроллер этой линии. Поскольку высокочастотный (125 МГц и более) опорный таймер должен быть дифференциальным, пришлось использовать стандартное, рекомендованное Xilinx устройство DIFF_INPUT_BUF (Xilinx solution record 19539). Необходимый тактирующий выход получается из дифференциальных входов подключением этого устройства через system.mhs (см. Руководство по включению устройства в состав системы).

В mpd – файл устройства были внесены добавления, касающиеся добавленных портов, а также порта запроса прерывания. В pao – файл были добавлены ссылки на упоминаемые ниже исходные тексты, использованные в устройстве.

Все остальные изменения и добавления вносились в файл user_logic.vhd.

Для реализации регистров управления контроллером линка необходимо было решить проблему перехода из области тактирования PLB в область тактирования внутреннего контроллера линии RocketIO. С этой целью использовались стандартные, полученные при помощи Coregen, асинхронные FIFO на базе BRAM [2], в количестве 3-х штук (для передачи команд во внутренний контроллер, для выдачи сигнала завершения передачи из внутреннего контроллера и для выдачи запроса прерывания по завершению приема из внутреннего контроллера).

Для реализации буферного блока памяти требовалось организовать трехпортовую память на базе BRAM (один порт – на доступ через PLB, один на выдачу данных и один – на прием). Использованный в устройстве модуль трехпортовой памяти на базе BRAM является минимально переработанным модулем четырехпортовой памяти, предоставляемым Xilinx в составе XAPP 228 [3].

Для реализации внутреннего контроллера линии использовалось свободно распространяемое Xilinx в виде исходных текстов IP Core Aurora [4]. Исходный вариант – версия 1.8.1, доработанный MTI, одноканальный, 4-байтовый, написанный на VHDL, был подвергнут лишь косметическим доработкам: добавлен один generic для упрощения выбора высокой или низкой частоты тактирующих входов, и внесены небольшие изменения в модуль тактирования, для упрощения выбора тактирующего входа CLK или CLK2, а также добавлен сигнал для тактирования BRAM.

При необходимости расширить канал это легко может быть сделано выбором другого варианта IP Core Aurora – например, четырехканального или восьмиканального. Общая логика устройства, реализованная в user_logic.vhd, при этом пострадать не должна.

Конкретная последовательность действий по включению устройства в проект XPS приводится в Приложении 2 (Руководство по включению устройства в состав системы).

3. *Архитектурные ограничения реализации*.

Устройство реализует единственный вариант операции обмена данными – **одностороннюю запись по инициативе передатчика**. Все другие типичные режимы, такие, как одностороннее чтение и/или двусторонние обмены, могут быть легко реализованы программно с использованием прерываний по концу приема сообщения. Например, это прерывание можно использовать для программного запроса встречной записи, что позволяет выполнить одностороннее чтение.

Использование шины PLB для интерфейса к устройству перестанет оправдывать себя при существенном расширении канала. Уже сейчас, для одноканальной линии, реально достигаемая максимальная скорость «накачки» данных в буферную память устройства при помощи PLB DMA примерно равна скорости передачи данных по линии.

Таблица измерений скорости и латентности передачи данных

	Передача по линку	**Передача по DMA**
Скорость передачи данных (МБ/с) Длина сообщения 64 КБ	**265**	**252**
Скорость передачи данных (МБ/с) Длина сообщения 8 Б	**20**	**6,2**
Латентость передачи данных (мкс) Длина сообщения 8 Б	**0,4**	**1,3**

Примечание. При измерении латентности для запуска обмена по линку использовалась запись в один регистр, для обмена по DMA – в два регистра.

Накладной расход протокола AURORA составляет 8 байт на сообщение, не считая времени синхронизации таймера 1 раз на 10000 байт [4].

Протокол Aurora не предусматривает обнаружения искажений передаваемых данных и повторов передачи при таких искажениях. Используемая конфигурация оборудования не выявила искажений при 30-часовом тесте, но в принципе помнить об этой проблеме следует.

Протокол Aurora не предусматривает старт-стопного по приему режима передачи данных. Все данные, передаваемые в линию, должны приниматься в ритме передачи. В настоящей версии старт-стоп по приему не требуется, поскольку в качестве приемника выступает BRAM, доступ в

которую происходит по выделенному порту. Однако, это верно лишь для режима обмена «точка-точка». Любая попытка организовать аппаратный коммутатор сообщений для настоящей технологии потребует добавить логику старт-стопа. Программная коммутация сообщений возможна уже сейчас, но ее реализация приведет к многократному росту латентности. Можно утверждать, что любая попытка организации на базе предложенной технологии коммутируемой сети потребует существенно пересмотреть и усложнить реализацию.

ЗАКЛЮЧЕНИЕ

В ходе выполнения НИР получены следующие научно-технические результаты.

Была определена необходимая и достаточная тестовая конфигурация СнК платы RMB-411 для тестирования устройства rocketmem и измерения его производительности. Разработаны:

- контроллер линка RocketIO СнК платы RMB-411, обеспечивающего интерфейс к шине PLB и использование протокола Aurora, при этом программному приложению обеспечивается интерфейс RDMA для взаимодействия с программным приложением, расположенным на противоположном конце линка.

- способ включения разработанного IP core в проекты СнК платы RMB-411, разрабатываемые в рамках Xilinx XPS. При этом обеспечивается способ использования IP core Aurora, делающий разрабатываемое ядро не зависимым от ширины линка RocketIO.

- набор тестовых приложений контроллера для standalone OS Xilinx XPS.

Проведены исследования работоспособности устройства, в том числе 30-часовой тест непрерывной работы с контролем правильности передаваемых данных.

Путем измерения фактически достижимых параметров производительности установлено, что при использовании линков на базе RocketIO с числом каналов, больших одного, шина PLB, и вообще предлагаемая в рамках XPS шинная инфраструктура интеграции устройств в СнК, не адекватна скоростям передачи данных в линках. Потребуется переход на сетевые системы интеграции СнК, аналогичные Hypertransport и/или PCI Express.

Таким образом, можно утверждать, что основная цель проведенной НИР достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Xilinx LogiCORE PLB IPIF (v2.02a) DS448 Product Specification.
2. Xilinx LogiCORE FIFO Generator (v2.1) User Guide UG175.
3. Xilinx Application note Xapp228. Quad-Port Memories in Virtex Devices.
4. Aurora_401 Reference Design (v1.8). User Guide UG041.
5. Xilinx OS and Libraries Document Collection. EDK 8.1i.
6. RocketIO Transceiver User Guide UG024.
7. http://www.myri.com
8. http://www.mellanox.com

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ 1. ***Руководство программиста***.

1. Программно доступные регистры.

Все регистры, доступные программисту для управления устройством, отображены в адресное пространство памяти процессора. Каждый регистр характеризуется смещением от **базового адреса устройства**, который оно (устройство) получает при генерации компьютера. Регистры сгруппированы по функциям в блоки, каждый блок имеет фиксированное смещение от базового адреса, а регистр в блоке – фиксированное смещение от начала блока.

Блок буферной памяти устройства размером 64К байт имеет базовый адрес, не связанный с базовым адресом устройства. Адрес блока буферной памяти задается отдельно при генерации компьютера.

Все приводимые ниже примеры фрагментов программ касаются программирования в среде XPS, с использованием Standalone BSP [5]. Оба упомянутых выше базовых адреса доступны в файле xparameters.h проекта XPS как XPAR_ROCKETMEM_N_BASEADDR и XPAR_ROCKETMEM_N_AR0_BASEADDR, соответственно, где N – порядковый номер устройства в системе.

Замечание. Здесь и далее под **словом** будем понимать 32-разрядное значение, под **двойным словом** – 64 – разрядное.

2.1. Блок регистров контроллера линка.

Смещение от базового адреса: 0 (0x0)

Все регистры – 32-разрядные слова, работа с ними должна происходить словными командами записи в память и чтения из памяти. Побайтовый доступ не допускается.

В приведенной ниже таблице регистры имеют названия RegX, где X – смещение данного регистра относительно базового адреса в словах.

Reg10			
01	2 15	16	31
	откуда	**куда**	

Регистр запуска передачи.

Reg12			
0	17	18	31
		сколько	

Регистр длины передаваемого сообщения.

Reg15	
0 30	31
	TF

Регистр завершения передачи.

Регистр длины передаваемого сообщения.

Смещение от начала блока: 12 слов.

Доступ: только запись.

Формат: 14 младших разрядов – длина передаваемого сообщения в двойных словах. Для буферного блока памяти в 64К байт максимальное значение – 8192.

Регистр завершения передачи.

Смещение от начала блока: 15 слов.

Доступ: только чтение.

Формат: единица младшего разряда означает, что последняя запущенная передача сообщения завершена. TF – передатчик свободен.

Регистр запуска передачи.

Смещение от начала блока: 10 слов.

Доступ: только запись.

Формат: регистр состоит из двух полуслов по 16 разрядов. Каждое полуслово содержит в 14 младших разрядах смещение в буферном блоке памяти, измеренное в двойных словах. Старшее полуслово содержит смещение передаваемого сообщения в буферном блоке памяти передатчика. Младшее полуслово содержит смещение в буферном блоке памяти приемника, в которое надо поместить принятое сообщение.

Запись в этот регистр запускает передачу сообщения и обнуляет бит готовности в регистре завершения передачи.

Прочие регистры контроллера.

Блок регистров контроллера линка имеет размер в 16 слов, считая от начала блока. Некоторые из этих регистров могут предоставлять отладочную информацию в режиме «только чтение». Использовать их программист не должен, поскольку их интерфейс не стабилен от версии к версии.

Пример использования.

Пусть при генерации компьютера базовый адрес устройства получился равным 0x30000000. Тогда для работы с регистрами контроллера в начале программы следует объявить словный указатель на блок регистров:

```
/* slave register block memory address: */
  int * volatile reg0 = (int*)0x30000000;
```

Также можно было написать:

```
#include “xparameters.h”
……
   int * volatile reg0 = (int*)XPAR_ROCKETMEM_0_BASEADDR;
```

Пусть теперь мы хотим переписать 20 двойных слов из собственной буферной памяти, расположенных в ней со смещением в 5 двойных слов от ее начала, в буферную память приемника, начиная с 7-го слова от ее начала.

```
reg0[12] = 20;
reg0[10] = (5 << 16) | 7;
```

Ожидание конца выдачи данных можно записать так:

```
while (!(reg0[15]&1 )){};
```

Не убедившись в завершении предыдущего обмена, следующий запускать нельзя.

2.2. Блок регистров общего управления.

Смещение от базового адреса: 64 слова (0x00000100)

Регистры описаны в: [1], в разделе Interface Description of PLB IPIF Services, PLB IPIF Reset/MIR service.

Необходимые для работы определения констант и функций доступа находятся в исходных текстах драйвера устройства rocketmem, в директории “drivers” дистрибутивного проекта.

2.3. Блок регистров управления контроллером прерываний.

Смещение от базового адреса: 128 слов (0x00000200)

Регистры описаны в: [1], в разделе Interface Description of PLB IPIF Services, PLB IPIF Interrupt service.

Необходимые для работы определения констант и функций доступа находятся в исходных текстах драйвера устройства rocketmem, в директории “drivers” дистрибутивного проекта. Там же, в файле rocketmem_selftest.c, имеется пример того, что нужно делать для разрешения прерываний от устройства, и как должен выглядеть обработчик.

Пример использования.

Пусть программа желает обрабатывать прерывания от завершения приема сообщения. Обработчик должен выглядеть примерно так:

```
// Located in: ppc405_0/include/xparameters.h
#include "xparameters.h"
#include "xexception_l.h"
#include "rocketmem.h"
#include "stdio.h"
```

```
void IHandler(void * baseaddr_p)
{
  Xuint32 baseaddr;
  Xuint32 IntrStatus;

  baseaddr = (Xuint32) baseaddr_p;

  /*
   * Get status from IP Interrupt Status Register.
   */
  IntrStatus = ROCKETMEM_mReadReg(baseaddr,
ROCKETMEM_INTR_ISR_OFFSET);

  xil_printf("IP Interrupt! Status register (ISR) value : 0x%08x \n\r", IntrStatus);

  /*
   * Clear IP interrupts by toggle write back to IP ISR register.
   */
  ROCKETMEM_mWriteReg(baseaddr, ROCKETMEM_INTR_ISR_OFFSET,
IntrStatus);
}
```

При этом в начале программы должен быть примерно такой код:

```
    XExc_Init();
    XExc_RegisterHandler( XEXC_ID_NON_CRITICAL_INT, IHandler, reg0 );
    XExc_mEnableExceptions( XEXC_ALL );

   /*
    * Enable all possible interrupts and clear interrupt status register(s)
    */

   n = ROCKETMEM_mReadReg(reg0, ROCKETMEM_INTR_ISR_OFFSET);
   xil_printf("   - IP (user logic) interrupt status : 0x%08x \n\r", n);
   xil_printf("   - clear IP (user logic) interrupt status register\n\r");
```

```
ROCKETMEM_mWriteReg(reg0, ROCKETMEM_INTR_ISR_OFFSET, n);
n = ROCKETMEM_mReadReg(reg0, ROCKETMEM_INTR_DISR_OFFSET);
xil_printf("   - Device (peripheral) interrupt status : 0x%08x \n\r", n);
xil_printf("   - clear Device (peripheral) interrupt status register\n\r");
ROCKETMEM_mWriteReg(reg0, ROCKETMEM_INTR_DISR_OFFSET, n);
xil_printf("   - enable all possible interrupt(s)\n\r");
ROCKETMEM_EnableInterrupt(reg0);
```

Прерыванию от завершения приема сообщения соответствует бит 0x2 регистра состояния (ISR).

2.4. Блок регистров управления контроллером DMA.

Смещение от базового адреса: 192 слова (0x00000300)

Регистры описаны в: [1], в разделе Interface Description of PLB IPIF Services, PLB IPIF DMA service.

Необходимые для работы определения констант и функций доступа находятся в исходных текстах драйвера устройства rocketmem, в директории “drivers” дистрибутивного проекта. Примеры запуска канала DMA – там же, в файле rocketmem_selftest.c. Реализован режим Simple DMA.

Прочие описанные в [1], в разделе Interface Description of PLB IPIF Services стандартные блоки регистров устройства в настоящем устройстве не реализованы.

3. Тестовые примеры.

В директории «*TestApp_Memory/src****» дистрибутивной системы имеется набор тестов устройства, которые могут использоваться также в качестве примеров программирования. Набор снабжен краткими пояснениями в файле*** *rmb_readme.txt****, который расположен там же.***

ПРИЛОЖЕНИЕ 2. *Руководство по включению устройства в состав системы*

1. Общие сведения.

Устройство rocketmem представляет собой устройство управления линком roketio. Предназначено для передачи данных между компьютерами, соединенными друг с другом такими линками. Настоящая версия реализует одноканальный линк.

Устройство включает в себя буферный блок памяти, контроллер линка, контроллер прерываний и контроллер DMA.

Устройство предназначено для использования в составе систем, создаваемых при помощи XPS. В комплект поставки включается готовая дистрибутивная система.

2. Порядок включения устройства в систему.

1). Для подготовки устройства к использованию следует скопировать в директорию **edk_user_repository/MyProcessorIPLib/drivers** установленной версии XPS поддиректорию **rocketmem_v1_00_a** из директории **drivers** дистрибутивной системы. Также следует скопировать в директорию **edk_user_repository/MyProcessorIPLib/pcores** установленной версии XPS поддиректорию **rocketmem_v1_00_a** из директории **pcores** дистрибутивной системы. Это обеспечит появление устройства в EDK User Repository установленной версии XPS.

2). Создав проект системы при помощи XPS, следует скопировать в директорию **pcores** этого проекта все файлы, находящиеся непосредственно в директории **pcores** дистрибутивной системы, имена которых начинаются на "**fifo_generator_**". Эти файлы используются при сборке устройства rocketmem.

3). Затем следует в файл **etc/fast_runtime.opt** создаваемого проекта, в раздел "program ngdbuild", добавить строку: "**-sd ../pcores**", например:

Было:

-uc <design>.ucf; # ucf constraints
<design>.ngd; # Name of NGD file. Filebase same as design filebase
End Program ngdbuild

Стало:

-uc <design>.ucf; # ucf constraints
<design>.ngd; # Name of NGD file. Filebase same as design filebase
-sd ../pcores;
End Program ngdbuild

Это – указание транслятору VHDL на местоположение дополнительных файлов, речь о которых шла в предыдущем пункте.

4). Для настоящего устройства необходимо устройство DIFF_INPUT_BUF, которое, если у Вас его нет, следует предварительно скопировать в **hw/XilinxProcessorIPLib/pcores** установленной версии XPS. Устройство доступно для свободного скачивания на сайте Xilinx.

5). Теперь следует штатными средствами XPS добавить устройство rocketmem из EDK User Repository в создаваемую систему, подключить его к шине PLB как "master-slave", и задать адреса. При задании адресов придется задавать границы двух адресных диапазонов: от **C_BASEADDR** до **C_HIGHADDR** включительно, и от **C_AR0_BASEADDR** до **C_AR0_HIGHADDR** включительно. Оба диапазона должны быть размером в 64Кбайт, их стартовые адреса должны быть кратны 64К байт. **C_BASEADDR** задает базовый адрес устройства, а **C_AR0_BASEADDR** – базовый адрес блока буферной памяти устройства (см. Руководство программиста). Оба эти значения впоследствии, при генерации программного обеспечения системы, автоматически появятся как константы в файле **ppc405_0/include/xparameters.h** создаваемого проекта.

6). Затем следует выйти из XPS и продолжить вносить изменения в файлы создаваемого проекта вручную, а именно:

7). В файл system.mhs, в раздел внешних портов, добавить внешние порты для линий rocketIO и тактирующего их дифференциального таймера, например:

Было:

```
PORT sys_clk_pin = dcm_clk_s, DIR = I, SIGIS = DCMCLK
 PORT sys_rst_pin = sys_rst_s, DIR = I
```

Стало:

```
PORT sys_clk_pin = dcm_clk_s, DIR = I, SIGIS = DCMCLK
 PORT sys_rst_pin = sys_rst_s, DIR = I
 PORT timer_clk_p_0_pin = timer_clk_p_0, DIR = I
 PORT timer_clk_n_0_pin = timer_clk_n_0, DIR = I
 PORT timer_clk_p_1_pin = timer_clk_p_1, DIR = I
 PORT timer_clk_n_1_pin = timer_clk_n_1, DIR = I
 PORT ch2_rxp_pin = ch2_rxp, DIR = I
 PORT ch2_rxn_pin = ch2_rxn, DIR = I
 PORT ch2_txp_pin = ch2_txp, DIR = O
 PORT ch2_txn_pin = ch2_txn, DIR = O
```

8). В разделе устройства rocketmem вставить ссылки на эти порты:

Было:

```
BEGIN rocketmem
 PARAMETER INSTANCE = rocketmem_0
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_BASEADDR = 0x30000000
 PARAMETER C_HIGHADDR = 0x3000ffff
 PARAMETER C_AR0_BASEADDR = 0x30010000
 PARAMETER C_AR0_HIGHADDR = 0x3001ffff
 BUS_INTERFACE MSPLB = plb
END
```

Стало:

```
BEGIN rocketmem
 PARAMETER INSTANCE = rocketmem_0
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_BASEADDR = 0x30000000
 PARAMETER C_HIGHADDR = 0x3000ffff
 PARAMETER C_AR0_BASEADDR = 0x30010000
 PARAMETER C_AR0_HIGHADDR = 0x3001ffff
 BUS_INTERFACE MSPLB = plb
# ===========================================
 PORT CH_RXP = ch2_rxp
 PORT CH_RXN = ch2_rxn
 PORT CH_TXP = ch2_txp
 PORT CH_TXN = ch2_txn
 PORT AURORA_REFCLK = exp_clk_input_0
 PORT IP2INTC_Irpt = EICC405EXTINPUTIRQ
END
```

О строке, предшествующей “END”, см. ниже.

8). Включить в систему дифференциальный таймер.

Для этого добавить в system.mhs следующий раздел:

```
BEGIN DIFF_INPUT_BUF
 PARAMETER INSTANCE = DIFF_INPUT_BUF_0
 PARAMETER HW_VER = 1.00.a
 PARAMETER INPUT_BUS_WIDTH = 1
 PORT DIFF_INPUT_P = timer_clk_p_0
 PORT DIFF_INPUT_N = timer_clk_n_0
 PORT SINGLE_ENDED_INPUT = exp_clk_input_0
END
```

9). Соединить выход запроса прерывания устройства с соответствующим входом на процессоре, для чего добавить строку в раздел процессорного блока:

Было:

```
PORT BRAMISOCMCLK = sys_clk_s
 PORT BRAMDSOCMCLK = sys_clk_s
 PORT CPMC405CLOCK = proc_clk_s
END
```

Стало:

```
PORT BRAMISOCMCLK = sys_clk_s
 PORT BRAMDSOCMCLK = sys_clk_s
 PORT EICC405EXTINPUTIRQ = EICC405EXTINPUTIRQ
 PORT CPMC405CLOCK = proc_clk_s
END
```

Замечание. До сих пор примеры приводились для случая включения в состав системы единственного устройства rocketmem, в предположении, что описываемые действия легко обобщить на случай нескольких устройств. В случае настоящего пункта это не так. При изготовлении системы с несколькими устройствами rocketmem для подключения нескольких выходов запроса прерывания к входу запроса прерывания на процессоре следует использовать логику “OR” или какой-либо стандартный контролер прерываний. В дистрибутивной системе, включающей в себя два устройства rocketmem, к линии запроса прерываний на процессоре подключено только нулевое устройство, соответственно, только оно способно генерировать прерывания по завершению приема сообщения.

10). Модификация файла system.mhs на этом завершена, осталось модифицировать файл data/system.ucf. Требуется внести в него информацию о номерах контактов линий rocketio и линий дифференциального таймера, например:

```
NET timer_clk_p_0_pin LOC=H18;
 NET timer_clk_n_0_pin LOC=J18;
NET timer_clk_p_0_pin IOSTANDARD = LVDS_25_DCI;
 NET timer_clk_n_0_pin IOSTANDARD = LVDS_25_DCI ;

NET "ch2_rxp_pin" LOC = A27;
NET "ch2_rxn_pin" LOC = A26;
NET "ch2_txp_pin" LOC = A28;
```

NET "ch2_txn_pin" LOC = A29;

3. ***Выбор дифференциального таймера для тактирования линий RocketIO***.

Архитектура FPGA Virtex2pro накладывает жесткие ограничения на взаимное расположение встроенных контроллеров RocketIO и тактирующих их таймеров на кристалле [6]. Каждый контроллер RocketIO имеет 4 входа тактирования, два для низкочастотных таймеров (до 125Мгц) и два для таймеров с частотой 125Мгц и выше. Входы эти называются REFCLK и REFCLK2 (низкочастотные) и BREFCLK и BREFCLK2 (высокочастотные). Рабочая частота линии получается умножением частоты таймера на 20. Выбор низкочастотного или высокочастотного входа определяется частотой используемого таймера, а выбор номера соответствующего входа (CLK или CLK2) – расположением таймера на кристалле: выбирать надо тот из входов, который ближе к таймеру. При использовании таймера низкой частоты требование близости таймера к входу тактирования имеет характер рекомендации. При использовании таймера высокой частоты, напротив, соблюдение требования **непосредственной** близости таймера к входу тактирования совершенно обязательно, причем использовать надо дифференциальный таймер. Дифференциальные буфера, при подключении к которым внешнего генератора получается дифференциальный таймер, и контроллеры rocketio расположены на кристалле таким образом, что для каждого из контроллеров существует ровно один дифференциальный буфер, расположенный в непосредственной близости от одного из входов этого контроллера (BREFCLK или BREFCLK2). Таким образом, по номерам внешних контактов линии RocketIO можно совершенно однозначно определить номера внешних контактов дифференциальных таймеров, пригодных для высокочастотного тактирования данной линии по каждому из входов (BREFCLK и BREFCLK2).

Таблица соответствия внешних контактов линий RocketIO и внешних контактов дифференциальных буферов приводится в документе [6], в главе “Digital Design Considerations”, в разделе “BREFCLK”, таблица “BREFCLK pin numbers”. Приведем пример использования этой таблицы для платы RMB-411.

Плата RMB-411 снабжена кристаллом Virtex2Pro размером 20, в корпусе FF1152. Этому случаю соответствует строка таблицы:

TOP		BOTTOM	
BREFCLK	BREFCLK2	BREFCLK	BREFCLK2
H18/J18	J17/H17	AK18/AL18	AL17/AK17

Читается это так: «Для тактирования линий RocketIO, расположенных в верхней части кристалла, следует использовать дифференциальный таймер с внешними входами H18/J18, по входу BREFCLK, или же дифференциальный таймер с внешними входами J17/H17 – по входу BREFCLK2, для линий из нижней части кристалла – соответственно, AK18/AL18 по BREFCLK или

AL17/AK17 – по BREFCLK2». Из описания RMB-411 мы знаем, что внешние генераторы подключены разработчиками платы к дифференциальным буферам H18/J18, а также AK18/AL18. Следовательно, в нашем распоряжении имеются два дифференциальных таймера, первый из которых годится для тактирования линий rocketio из верхней части кристалла по входу BREFCLK, второй – для тактирования линий из нижней части кристалла, по тому же входу.

Соответствие между входами тактирования линий RocketIO и дифференциальными буферами таймеров известно программе map в составе Xilinx ISE, и при его нарушении шаг «map» сборки системы завершается аварийно, причем в system.log выдается рекомендация о том, какой именно таймер следовало бы использовать для данной линии и данного ее входа тактирования. Если указанный дифференциальный таймер на данной плате не задействован, следует выбрать для тактирования другой вход.

Выбор того или иного входа тактирования (BREFCLK или BREFCLK2) подразумевает внесение изменений в схему устройства rocketmem. Это изменение достигается выбором варианта строки в файле **pcores/rocketmem_v1_00_a/data/rocketmem_v2_1_0.pao** из дистрибутивного проекта. Строка находится близко к концу файла, и имеет вид

lib rocketmem_v1_00_a ck_aurora_4_bref vhdl

если требуется выбрать вход тактирования BREFCLK, или же

lib rocketmem_v1_00_a ck_aurora_4_bref2 vhdl

если требуется выбрать вход тактирования BREFCLK2.

4. ***Настройка параметров линии RocketIO***.

О настраиваемых параметрах линии rocketio подробно рассказано в документе [6]. Для наилучшего учета конкретных вариантов кабельной линии (длина кабеля, тип разъема) может потребоваться изменение трех параметров:

TERMINATION_IMP (значение сопротивления согласующего резистора приемника),

TX_DIFF_CTRL (значение напряжения сигнала),

TX_PREEMPHASIS.

Для изменения значений этих параметров следует найти в файле **aurora_401.vhd** контекстным поиском generic map компонента GT_CUSTOM, и вручную внести требуемые изменения.

СПИСОК ИСПОЛНИТЕЛЕЙ

Исполнитель НИР	С.С. Андреев
Исполнитель НИР	А.О. Лацис
Исполнитель НИР	Е. А. Плоткина