Прямая передача данных между ПЛИС Virtex-7 по шине PCI Express

Ю. А. Румянцев, ООО НПО «Роста»,
rumyantsev@rosta.ru

Аннотация. В данной статье рассматривается передача данных по шине PCI Express с одновременным участием нескольких ПЛИС. В компьютерной системе, к PCI Express шине которой подключено несколько (в нашем случае 8) оконечных устройств (PCIe endpoints) ПЛИС запускается одновременно несколько транзакций передачи данных двух типов: А) DMA передача между ОЗУ и ПЛИС (чтение/запись) и Б) прямая передача данных между двумя ПЛИС (запись). Используя соединение PCI Express x4 Gen 2.0 при обращении в память была получена скорость записи 1451 МБ/с (90% от максимальной). Скорость записи данных между ПЛИС была равна 1603 МБ/с (99 % от максимальной) при длине пакетов 128 байт и 1740 МБ/с (99% от максимальной) при длине пакета 256 байт. Латентность передачи данных между ПЛИС зависит от количества промежуточных коммутаторов, и была равна 0,7 мкс для одного коммутатора и 1 мкс для трех. Также показано, что при одновременных передачах через общий канал скорость отдельных передач не уменьшается до тех пор, пока суммарная скорость передачи не превышает пропускную способность общего канала; затем канал используется на 100%, а его пропускная способность делится поровну между устройствами.

Ключевые слова: ПЛИС; FPGA; PCI Express;

1. Введение

PCI Express де факто стало стандартом передачи данных между CPU, системной памятью и аппаратными ускорителями (GPU, ПЛИС) в задачах High Performance Computing (HPC). Во-первых, шина PCI Express обладает небольшой латентностью, во-вторых, имеет высокую скорость передачи данных (около 7 ГБайт/с при соединении PCI Express x8 Gen 3.0). Наконец, шине PCIe присуща хорошая масштабируемость: обычно на материнских платах нет недостатка в разъемах шины PCI Express, к которой можно

подключить несколько плат ускорителей GPU или ПЛИС. Также в последнее время появились технические решения, позволяющие расширить шину PCI Express через кабельные соединения и подключать дополнительные периферийные устройства вне корпуса компьютера (1).

В современных HPC системах недостаточно иметь единственный аппаратный ускоритель. Уже стало привычным видеть две платы GPU на локальной PCI Express шине вычислительного узла. Для обеспечения обмена данными напрямую между GPU была разработана технология GPUDirect (2). Используя эту технологию, можно организовать обмен данными между устройствами GPU по шине PCI Express напрямую без использования оперативной памяти в качестве буфера, что позволяет существенно снизить накладные расходы на передачу данных.

Другие примеры множества ускорителей на шине PCI Express включают в себя системы, в которых одновременно работают и GPU и ПЛИС. В первом примере команда исследователей из Австралии собрали персональный компьютер из материнской платы Intel, процессора Core i7, платы GPU nVidia Tesla C2070 и платы Altera DE-530 с установленным кристаллом ПЛИС Stratix-IV (3). Они назвали его «Химера» в честь мифического чудовища Древней Греции Химеры, имеющий 3 головы (козел, змея, лев) на одном теле. Они успешно решили несколько задач (интегрирование методом Монте-Карло, поиск шаблона в 2D массиве) и работают над применением этой системы для анализа непрерывных гравитационных волн. Ключевой особенностью их проекта было то, что GPU и ПЛИС работали одновременно над одной задачей, и данные передавались от GPU к ПЛИС по шине PCI Express. Однако следует отметить, что этот обмен шел под управлением центрального процессора и через буфер в оперативной памяти.

Другая команда исследователей из Брюсселя собрала гибридный компьютер с платами GPU nVidia Tesla C2050 и ПЛИС Pico Computing EX-500 (4). Последняя плата может включать в себя от 1 до 6 кристаллов ПЛИС Xilinx Virtex6, каждая со своим собственным PCI Express интерфейсом к хосту. Подробности проекта пока неясны, сообщение доступно только в виде препринта.

Наконец команда разработчиков из компании Microsoft исследовала передачу данных напрямую между GPU и ПЛИС по шине PCI Express (5). В их системе были установлены платы GPU nVidia GeForce GTX 580 и ПЛИС Xilinx ML605 с 1 кристаллом Virtex6. Разработчики нашли способ инициировать прямую передачу данных между GPU и ПЛИС, используя CUDA API, технологию GPUDirect и доработки Linux драйвера устройства ПЛИС. Это позволило увеличить скорость и снизить латентность передачи данных по сравнению с подходом, использующим оперативную память в качестве промежуточного буфера. В данном случае GPU было ведущим устройством, а ПЛИС ведомым.

Шина PCI Express может быть использована и для передачи данных напрямую между ПЛИС. Компания Xilinx демонстрировала эту возможность (6). Инженеры Xilinx соединяли два кристалла ПЛИС напрямую по шине PCI Express без использования коммутаторов и вообще без компьютера с центральным процессором. Один кристалл самостоятельно конфигурировал свой PCI Express интерфейс, устанавливал связь со вторым и конфигурировал его. После чего было возможно передавать данные в обе стороны между двумя кристаллами ПЛИС. Этот подход демонстрирует принципиальную возможность передачи данных между ПЛИС по шине PCI Express, однако не может быть использован в случае, когда к компьютеру с центральным процессором подключены несколько ПЛИС через PCI Express коммутаторы.

В данной статье описывается передача данных в системе, содержащей множество ПЛИС на PCI Express шине. Обсуждаются вопросы одновременной передачи данных между системной памятью и несколькими ПЛИС, а также одновременной передачи данных между несколькими ПЛИС напрямую друг с другом. По нашим сведениям данное сообщение является первым в части описания передачи данных между ПЛИС через PCI Express шину компьютера.

2. Описание системы

Эксперименты проводились в следующей системе. В материнскую плату с процессором Intel Core i7 в разъем PCI Express 2.0 x8 устанавливался адаптер RHA-25 производства фирмы Роста (1), расширяющий PCI Express шину через кабельные соединения. На адаптере RHA-25 установлен PCI Express коммутатор (PLX Technology), три порта которого используются для внешних соединений: один ножевой upstream порт x8 Gen 3.0 и два кабельных downstream порта x4 Gen 3.0. К этой системе через 2 кабельных соединения PCI Express x4 Gen 3.0 подключался вычислительный блок RB-8V7 (1). Блок RB-8V7 имеет симметричную архитектуру и конструктивно состоит из двух плат RC-47. Каждая плата RC-47 имеет коммутатор PCI Express фирмы PLX с одним кабельным upstream портом и четырьмя downstream портами, каждый из которых соединен со своим ПЛИС Xilinx Virtex-7 (XC7V585T). Таким образом, в нашей системе к хосту по шине PCI Express с помощью одного адаптера RHA-25 подключалось 8 ПЛИС Virtex-7 (V7). Все ПЛИС соединялись с коммутатором PLX по интерфейсу PCI Express x4 Gen 2.0.

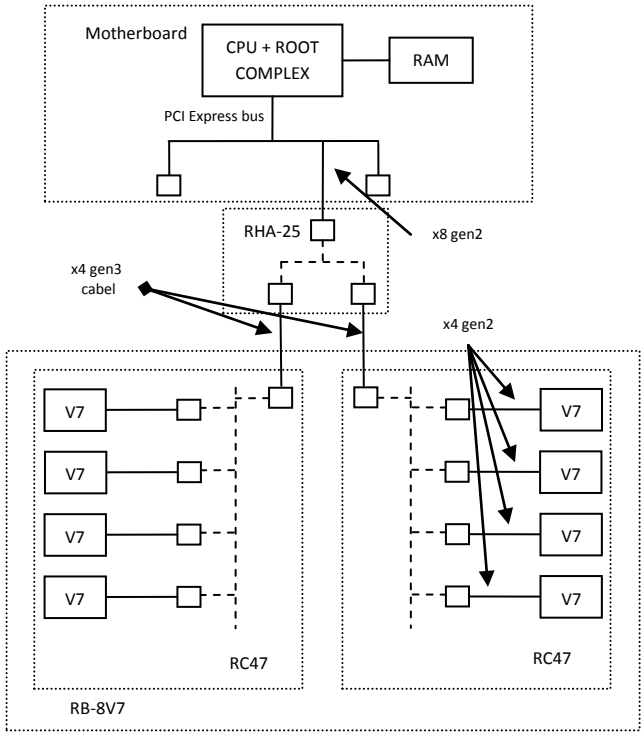

Рис. 1 Аппаратное обеспечение. Блок RB-8V7 подключен через кабельные PCIe соединения и адаптер RHA-25 к хост компьютеру

Внутри ПЛИС была реализована следующая схема (рис. 2). Проект использует PCI Express IP ядро фирмы Xilinx (7). Блок Rosta DMA Engine определяет функционал устройства на шине PCI Express. ПЛИС может выступать как в роли ведущего, так и ведомого устройства. Как ведомое устройство ПЛИС предоставляет центральному процессору доступ по чтению и записи к своим регистрам, а также может принимать большие пакеты данных от других устройств на шине (например, от других ПЛИС) с сохранением принятых данных в очереди EP_RX_FIFO. Как ведущее устройство ПЛИС способна обращаться в оперативную память компьютера в режиме DMA (чтение/запись). При этом при записи в память данные будут считываться из TX_FIFO, а при чтении из памяти – записываться в RX_FIFO. Также устройство способно генерировать транзакции записи по произвольному адресу на шине (например, для передачи данных в другие ПЛИС), в этом случае данные для передачи считываются из очереди EP_TX_FIFO. Организацией приема поступающих пакетов занимается автомат RX_STATE_MACHINE, за передачу пакетов отвечает автомат TX_STATE_MACHINE. Прием и передача пакетов могут идти одновременно.

Блок TX_ARBITER определяет, какой пакет генерировать для передачи следующим: абсолютный приоритет отдается генерации ответов в процессе чтения регистров центральным процессором, остальные пакеты (запросы на чтение/запись оперативной памяти или запросы на запись по произвольному адресу) планируются с равным приоритетом (round-robin).

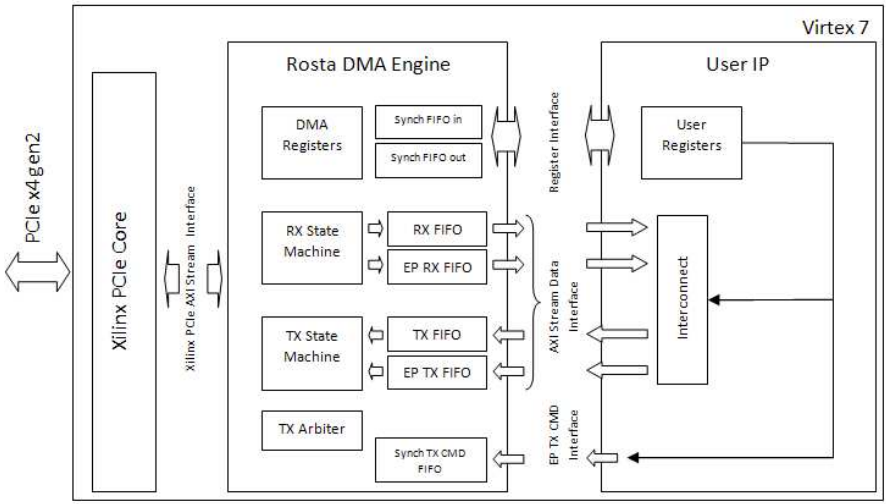

Рис. 2 Блок-схема проекта ПЛИС

Программированием DMA передачи данных между ПЛИС и ОЗУ занимается центральный процессор (путем записи в блок DMA_REGISTERS), а для управления процессом записи данных по произвольному адресу в другую ПЛИС есть внутренний аппаратный интерфейс EP TX CMD. Для доступа к регистрам пользователя в блоке User IP существует интерфейс Register Interface, а для блочной передачи данных между пространством PCI Express и пользовательской схемой есть четыре AXI Stream интерфейса, соединенных с очередями RX_FIFO, TX_FIFO, EP_RX_FIFO, EP_TX_FIFO. Наконец в сторону PCI Express ядра Xilinx блок Rosta DMA Engine поддерживает Xilinx PCIe AXI Stream Interface, имеющий ширину 64 бита. Блок работает на двух тактовых частотах: на частоте работы PCIe ядра Xilinx (250 МГц) – левая часть схемы, и на произвольной пользовательской частоте – правая часть схемы. Развязка по частотам происходит через очереди FIFO. Но во всех описываемых далее экспериментах пользовательская частота была равна частоте, на которой работало ядро PCIe (250 МГц).

Пользовательский блок User IP определяет поведение устройства на уровне приложения. В данной работе использовались несколько разных схем для разных назначений. Во-первых, использовалась схема для проверки корректности передачи данных между ПЛИС и ОЗУ. В этом случае в блоке

Interconnect выход RX_FIFO просто замыкался на вход TX_FIFO. Это позволяло записывать в оперативную память абсолютно те же данные, что и были считаны из нее. Программа на центральном процессоре (в дальнейшем просто хост) записывала данные в ПЛИС, считывала их, сравнивала и убеждалась в правильности сравнения данных.

Во-вторых, использовалась схема для измерения максимальной скорости передачи данных между ПЛИС и ОЗУ в обоих направлениях. Для этого из RX_FIFO данные, постоянно считывались, т.е. очередь всегда была пустая, и не было задержек при приеме данных из-за ее переполнения, а в TX_FIFO шла постоянная запись данных, т.е. не возникало задержек при передаче из-за недостатка данных в очереди.

В-третьих, была разработана схема для проверки корректности прямой передачи данных между ПЛИС. В блоке Interconnect была реализована схема коммутации выходов RX_FIFO и EP_RX_FIFO и входов TX_FIFO и EP_TX_FIFO. В первом случае выход RX_FIFO замыкался на вход TX_FIFO, а выход EP_RX_FIFO – на вход EP_TX_FIFO. Во втором случае выход RX_FIFO замыкался на вход EP_TX_FIFO, а выход EP_RX_FIFO – на вход TX_FIFO. Управлялась эта схема коммутации битом из одного из пользовательских регистров. В блок User Registers были добавлены регистры для управления интерфейсом EP TX CMD. В данном случае сам хост управлял передачей данных между ПЛИС, но вообще интерфейс EP TX CMD разрабатывался таким образом, чтобы сама схема ПЛИС могла инициировать передачу данных.

В четвертых, для измерения максимальной скорости передачи данных между ПЛИС была разработана специальная схема, которая при передаче данных постоянно записывала данные в EP_TX_FIFO, а при приеме постоянно вычитывала данные из EP_RX_FIFO. При этом внутри схемы был реализован аппаратный таймер, значения которого сохранялись и в дальнейшем отправлялись на хост. Интерфейсом EP TX CMD управлял хост через пользовательские регистры.

Наконец, для измерения латентности использовалась схема, передающая данные в другую ПЛИС. Принимающая данные ПЛИС сразу же записывала их обратно в то же устройство. В передающей ПЛИС одновременно с началом передачи запускался аппаратный таймер, который останавливался в тот момент, когда данные начинали поступать в очередь EP_RX_FIFO. Далее значение таймера можно было считать на хост через пользовательские регистры.

На хосте была установлена ОС Linux. Для работы с аппаратурой применялись драйверы и библиотеки собственной разработки.

3. Пропускная способность шины PCI Express

Прежде чем переходить к описанию экспериментов по измерению скорости передачи данных необходимо выяснить ее теоретический предел. Известно, что из-за применения кодирования 8B/10B максимальная теоретическая скорость передачи информации по одной линии PCI Express на частоте 2.5 ГГц (gen1) равна $V_{theory} = 2.0\ Гбит/с$. Для протокола второй генерации Gen 2.0 с частотой 5 ГГц эта скорость в 2 раза выше ($V_{theory} = 4.0\ Гбит/с$). Скорость передачи информации для третьей генерации выше еще в 2 раза и равна $V_{theory} = 8.0\ Гбит/с$ на одну линию (несмотря на то, что частота равна 8 ГГц, в протоколе третьей генерации применяется другой метод кодирования символов 128B/130B снижающий дополнительную нагрузку).

Однако данные передаются с чуть меньшей скоростью из-за того, что передача идет пакетами, включающими в себя дополнительную информацию (стартовые/стоповые биты, заголовок, контрольная сумма и тд). В итоге при передаче одного пакета транзакции записи помимо данных передаются дополнительные 20 байт, относящиеся к этому же пакету. Также по шине PCI Express передаются пакеты, вовсе не содержащие данные, а выполняющие чисто служебные функции. К таким можно отнести подтверждения о приеме пакетов с данными, требования повторить передачу в случае обнаружения несоответствия контрольной суммы, пакеты, обновляющие счетчики буферов свободного места в коммутаторах, и другие. Точно оценить их влияние на скорость передачи данных сложно (это зависит от конкретной реализации), однако можно в среднем примерно оценить их вклад как 3 дополнительных байта на 1 пакет с данными (8). Итого мы принимаем, что на передачу одного пакета с данными в среднем передается 23 дополнительных служебных байтов. Более подробно об этом написано в (8). В дальнейшем, если не оговорено обратное, под длиной пакета мы будет понимать количество данных в пакете.

По шине PCI Express данные могут передаваться пакетами разной длины. Максимальное количество данных при передаче одного пакета определяется параметром MAX_PAYLOAD_SIZE, значение которого равно степени двойки. У каждого устройства есть параметр MAX_PAYLOAD_SIZE_SUPPORTED, определяющий максимальный размер пакета, который может быть передан этим устройством. Конфигурационное ПО (программа BIOS) настраивает параметр MAX_PAYLOAD_SIZE для всех устройств в системе равным наименьшему из значений, поддерживаемых устройствами в системе. Как правило, современные чипсеты поддерживают размер пакетов до 128 байт, однако есть системы и с 256 байтами. В наших экспериментах параметр чипсета MAX_PAYLOAD_SIZE_SUPPORTED был равен 128 байт, и, несмотря на то, что устройства ПЛИС и PCI Express коммутаторы на платах RHA-25 и RC-47 поддерживали большие размеры пакетов (до 512), BIOS

настроила параметр MAX_PAYLOAD_SIZE для всех устройств в системе равным 128 байт.

Чем больше размер передаваемых пакетов, тем ближе пропускная способность приближается к теоретическому пределу передачи информации. Если теоретический предел принять равным 1, то по следующей формуле можно вычислить практический предел передачи данных в зависимость от размера пакета (см. табл. 1):

$$\xi = \frac{x}{x + 23}$$

$$V_{prac} = V_{theory} * \xi \qquad (1)$$

где x – размер пакета.

Таблица 1. Зависимость относительной скорости передачи данных от размера пакета

x, байт	4	8	16	32	64	128	256	512	1024
ξ	0,148	0,258	0,41	0,581	0,735	0,847	0,917	0,957	0,978

В табл. 2 приведено сравнение максимальных теоретических и практических скоростей передачи собственно данных для длины пакета равной 128 и 256 байт.

Таблица 2. Максимальные скорости передачи информации и данных на шине PCI Express для пакетов длиной 128 и 256 байт

Интерфейс	Ширина	V_{theory}, Гбит/с (10^9 *бит*)	V_{theory}, МБайт/с (2^{20} *байт*)	Payload 128	Payload 256
				V_{prac}, Мбайт/с (2^{20} *байт*)	V_{prac}, Мбайт/с (2^{20} *байт*)
PCIe 1.0	x1	2	238	201	218
	x2	4	476	403	436
	x4	8	953	806	873
	x8	16	1907	1612	1748
PCIe 2.0	x1	4	476	403	436
	x2	8	953	806	873
	x4	16	1907	1612	1748
	x8	32	3814	3225	3497
PCIe 3.0	x1	8	953	806	873
	x2	16	1907	1612	1748
	x4	32	3814	3225	3497
	x8	64	7629	6450	6995

4. Передача HOST-ПЛИС

При передаче данных между ПЛИС и системной памятью используется механизм прямого доступа в память. Пользовательское приложение на хосте подготавливает буфер в ОЗУ и делает системный вызов write или read. PCI Express драйвер устройства фиксирует страницы пользовательского буфера в памяти и использует механизм scatter/gather DMA. Список дескрипторов (пара адрес-длина) страниц записывается во внутреннюю память устройства ПЛИС, а затем ПЛИС сама обращается в память по адресам из этого списка. После завершения передачи данных ПЛИС генерирует прерывание, которое и завершает системный вызов. Когда пользовательское приложение намерено записать данные в ПЛИС, устройство обращается в ОЗУ с транзакциями чтения, а при чтении из ПЛИС устройство генерирует транзакции записи.

Далее везде будет подразумеваться, что слова «запись» и «чтение» относятся к ПЛИС, т.е. скорость записи подразумевает скорость процесса записи, начатого ПЛИС.

Транзакции записи, генерируемые на шине PCI Express, всегда однонаправлены. Инициатор транзакции записи формирует пакет, состоящий из заголовка и данных. Инициатор сам определяет размер данных, учитывая только ограничение MAX_PAYLOAD_SIZE. Все, что было сказано про скорость передачи данных в предыдущем параграфе, относится именно к транзакциям записи. Для транзакций записи скорость передачи данных легко оценить теоретически и измерить ее зависимость от размера передаваемого пакета.

С транзакциями чтения все немного сложнее. Инициатор транзакции чтения сначала формирует запрос на чтение – короткий пакет, состоящий только из заголовка. В этом пакете указывается, откуда (с какого адреса) и сколько необходимо считать данных. Максимальное количество данных, которое может быть запрошено за один раз определяется параметром MAX_READ_REQUEST_SIZE и обычно равно 4 КБ. Обычно периферийные устройства обращаются с запросами на чтение в ОЗУ, но могут запросить данные и из другого устройства. Когда устройство (периферийное или контроллер ОЗУ) получает запрос на чтение, сначала оно запрашивает требуемые данных из своей памяти, а затем возвращает их инициатору транзакции по шине PCI Express, генерируя ответные завершающие пакеты (completion request). При этом оно само определяет размер возвращаемых пакетов, опять же учитывая только ограничение MAX_PAYLOAD_SIZE. Инициатор транзакции не может повлиять на размер возвращаемых пакетов. Как правило, контроллер ОЗУ будет возвращать пакеты с длиной данных равной MAX_PAYLOAD_SIZE. Скорость передачи данных для транзакций чтения оценить сложно по нескольким причинам. Во-первых, генерируются пакеты двух типов, двигающиеся в противоположных направлениях – запросы на чтения и завершающие ответы. Во-вторых, на скорость передачи данных будут влиять задержки, возникающие при чтении из ОЗУ. Наконец, неясно, как проследить зависимость скорости передачи данных от количества данных в пакете. Поэтому в данной работе мы просто измеряли интегральное время от посылки первого запроса на чтение до прихода последнего байта данных, по нему вычисляли скорость чтения данных из ОЗУ и не прослеживали ее зависимость от размера пакета.

Шина PCI Express обеспечивает надежную передачу данных на уровне транзакций, т.е., пересылая данные, агенты (оконечные устройства и промежуточные коммутаторы) автоматически вычисляют контрольную сумму пакета, сравнивают ее с закодированной в самом пакете и требуют повторной передачи в случае обнаружения ошибки при передаче данных. Однако это не означает, что приложение на хосте или схема ПЛИС не могут сформировать и передать неправильные данные в результате ошибки программиста. Поэтому

для верификации корректности нашей схемы, включающей в себя приложение, драйвер и схему ПЛИС, были проведены проверки корректности передачи данных. Для этого использовалась первая схема User IP, в которой выход RX_FIFO замыкался на вход TX_FIFO. Размеры очередей FIFO были равны 4 КБ. Были проведены два эксперимента. В первом хост последовательно записывал в ПЛИС 4 КБ данных, затем их считывал и сравнивал. Второй эксперимент использовал тот факт, что тракты приема и передачи данных в ПЛИС могут работать параллельно. Хост сначала программировал обе операции записи и чтения, а затем ПЛИС начинала считывание данных из ОЗУ, и как только они поступали сначала в RX_FIFO, а затем и в TX_FIFO, сразу же начинала их запись обратно в ОЗУ. Это позволило передавать за раз намного больший объем данных по сравнению с размером очередей RX_FIFO и TX_FIFO (в нашем эксперименте 4 МБ). В обоих экспериментах сравнение переданных и принятых данных проходило успешно, что и позволило нам судить о правильности работы нашей схемы.

Эксперименты по измерению скорости передачи данных начались с измерения зависимости скорости записи в ОЗУ одного устройства ПЛИС от размера данных в передаваемых пакетах. Проводились эксперименты со значениями payload равными 8, 16, 32, 64 и 128 байт. В каждом эксперименте передавалось 4 МБ в одну сторону. Узким местом в тракте Virtex7 – ОЗУ было соединение PCI Express x4 Gen 2.0 между ПЛИС и коммутатором PCI Express на плате RC47 (см. рис. 1). Результаты представлены на рис. 3. Верхняя кривая Vtheory представляет собой теоретическую зависимость (формула 1) максимально достижимой скорости передачи данных от длины пакета (для интерфейса PCI Express x4 Gen 2.0). Кривая посередине Vhard представляет собой скорость, измеренную с помощью аппаратного таймера в ПЛИС и учитывающую только непосредственно передачу данных на шине PCI Express (время замерялось от начала передачи первого пакета с данными до конца передачи последнего). Наконец, кривая Vapp представляет собой скорость, измеренную в приложении на хосте (измерялось время выполнения системного вызова read).

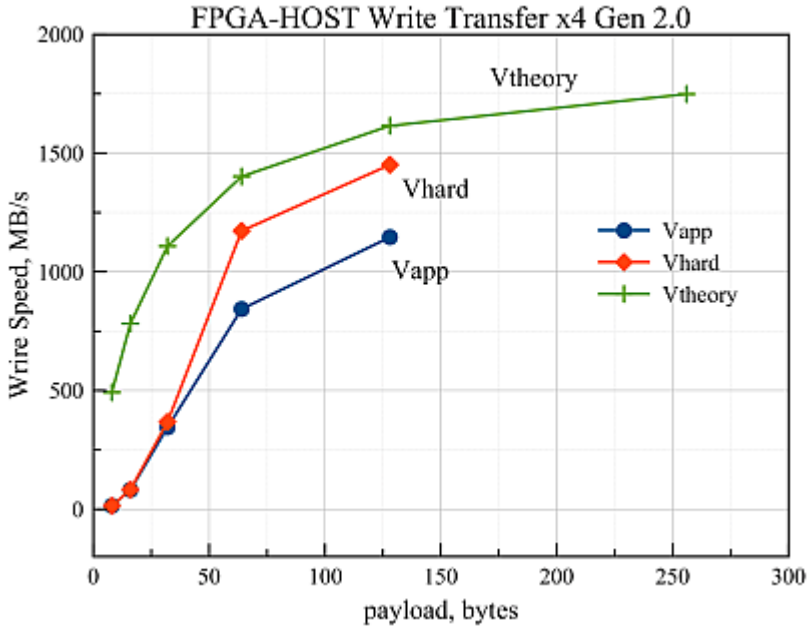

Рис. 3 Зависимость скорости записи в ОЗУ от длины пакета для интерфейса PCI Express x4 Gen 2.0

Из графика видно, что при значениях payload меньше 64 байт скорости Vhard и Vapp намного меньше Vtheory. Это объясняется тем, что в качестве приемника данных в данном случае выступает DDR память ОЗУ, обеспечивающая высокую скорость записи только в пакетном режиме (burst), передавая большое количество данных за одну транзакцию. Для пакета размером 128 байт Vhard = 1451 МБ/с, что составляет 90% от максимального значения 1612 МБ/с. Также видно, что скорость, измеренная в приложении Vapp (1146 МБ/с при payload=128) намного ниже Vhard для значений payload, начиная с 64 байт. Это связано с тем, что при передаче 4 МБ данных формируется порядка 1000 дескрипторов страниц, которые процессор записывает в ПЛИС. Эта начальная задержка (порядка 1 мс) существенно влияет на скорость передачи данных. Полное время выполнения системного вызова read для payload = 128 байт равно примерно 3,7 мс. Если из этого времени отнять начальную задержку в 1 мс, то получиться скорость примерно равная 1450 МБ/с, что совпадает со скоростью Vhard, измеренной аппаратно. В наших планах стоит изменение логики драйвера и схемы Rosta DMA Engine для уменьшения начальной задержки при программировании DMA передачи. Идея усовершенствования заключается в том, чтобы не передавать весь список дескрипторов страниц пользовательского буфера в ПЛИС, а вместо

этого сохранить его в доступной для ПЛИС области в ОЗУ. Тогда ПЛИС сама сможет вычитывать из ОЗУ дескрипторы, по которым уже будут передаваться данные. Процесс вычитывания дескрипторов из памяти и собственно передачу данных можно будет запустить параллельно, тем самым существенно сократив начальную задержку и как следствие, увеличив скорость передачи данных. Пока же мы будем ориентироваться на скорость, измеренную аппаратно. Также в последующих экспериментах не будут исследоваться передачи данных с payload меньше 128 байт.

В следующем эксперименте запускалось одновременно несколько передач данных с участием разных ПЛИС. Приложение на хосте запускало несколько потоков pthreads – по одному потоку на отдельную передачу. В каждом потоке ПЛИС сначала программировалась для чтения из ОЗУ, а затем на запись. Одновременность передачи достигалась использованием барьерной синхронизации в приложении. Размер передаваемого буфера был равен 4 МБ, размер пакетов при записи был равен 128 байт. В эксперименте были задействованы все 8 устройств ПЛИС Virtex7, содержащихся в блоке RB-8V7. Скорости записи данных в ОЗУ в зависимости от количества одновременно работающих ПЛИС представлены на рис. 4.

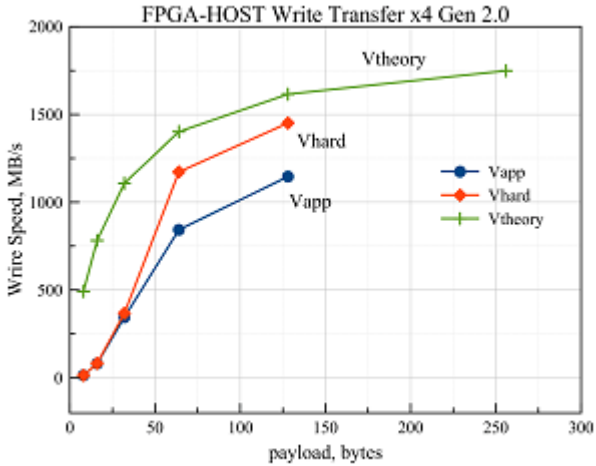

Рис. 4 Зависимость скорости записи в ОЗУ от количества одновременных транзакций

Две нижние кривые (V_average и V_average_app) отображают среднюю скорость записи, измеренную аппаратно и в приложении соответственно. Каждая отдельно взятая ПЛИС ограничена по скорости максимальной скоростью записи данных через интерфейс PCI Express x4 Gen 2.0, в нашем случае 1451 МБ/с. Кривые V_sum и V_sum_app – это суммы скоростей передачи данных отдельных устройств. Прямая Vmax = 3225 МБ/с

представляет собой максимальную скорость передачи данных через узкое место системы, ограничивающее скорость одновременной передачи данных. Таким узким местом является соединение PCI Express x8 Gen 2.0 адаптера RHA-25 с материнской платой компьютера. Для одной и двух одновременных передач скорость записи одинакова (1451 МБ/с), потому что суммарная скорость двух передач меньше Vmax. Начиная с трех передач, скорость записи отдельного устройства падает, однако суммарная скорость одинакова и равна Vmax. То, что на графике суммарная скорость для трех и более устройств превышает Vmax, объясняется «псевдоодновременностью» передачи данных. Как бы хорошо потоки не были параллельно распределены между ядрами центрального процессора, по шине PCI Express команды все равно идут последовательно. Поэтому какие-то устройства начинают передачу раньше, другие позже. Это приводит к тому, что в течение короткого промежутка времени вначале и в конце передачи данных общий канал соединения адаптера и материнской платы используется не всеми участвующими в передачи устройствами. Поэтому для них скорость передачи получается больше, а сумма всех скоростей превышает максимальную. Реально же получается, что общий канал используется на все 100%, а пропускная способность разделяется между устройствами в равных долях.

Аналогично можно рассмотреть случай чтения из ОЗУ (рис. 5).

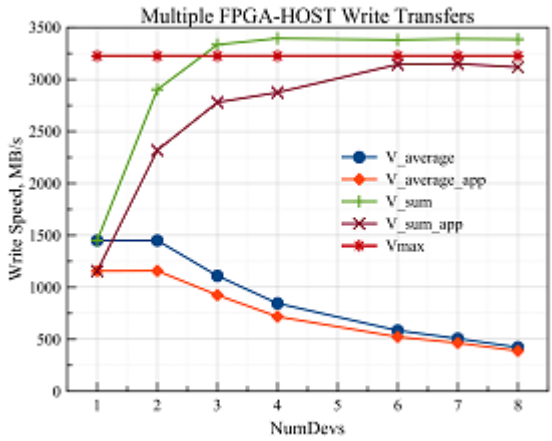

Рис. 5 Зависимость скорости чтения из ОЗУ от количества одновременных транзакций

Тут видно, что для передачи данных с участием от одного до трех устройств скорость чтения для каждого устройства одинакова и равна 1000 МБ/с. Начиная с четырех устройств, скорость ограничивается пропускной способностью канала связи с хостом.

5. Передача ПЛИС-ПЛИС

Запросы на запись и чтение памяти на шине PCI Express направляются в соответствии с адресом, закодированным в заголовках пакетов. Ведущее устройство способно сгенерировать пакет с произвольным адресом. Этот адрес может указывать в оперативную память, а может принадлежать области адресов, выделенных другому периферийному устройству. В последнем случае пакет запроса будет направлен от одного устройства к другому. В нашем случае таким образом передавались запросы на запись между разными ПЛИС. На чтение тесты не проводились.

Каждая ПЛИС получает от BIOS или от операционной системы диапазон адресов, по которым можно обратиться в это устройство. Чтобы запрограммировать передачу данных от ПЛИС А в ПЛИС В, приложение на хосте должно сообщить ПЛИС А (записать в соответствующий регистр в блоке User IP) базовый адрес ПЛИС В. В ПЛИС в блоке Rosta DMA Engine был реализован аппаратный интерфейс EP TX CMD, предназначенный для инициации передачи данных в другую ПЛИС изнутри схемы. Схема записывает данные для передачи в EP_TX_FIFO и передает через интерфейс EP TX CMD базовый адрес другого устройства и длину передачи. Далее передающий автомат TX_STATE_MACHINE в блоке Rosta DMA Engine начинает передачу данных из EP_TX_FIFO по указанному адресу. В принимающей ПЛИС данные записываются в очередь EP_RX_FIFO.

Для начала надо было убедиться в корректности передачи данных между ПЛИС. Для этого был поставлен следующий эксперимент. В передаче была задействована последовательность из всех 8 ПЛИС Virtex7, входящих в состав блока RB-8V7 (рис. 6). В первые 7 ПЛИС хост записывал базовые адреса устройств по следующей схеме. В первое устройство – базовый адрес второго, во втрое – третьего и тд. Первое устройство хост программировал на чтение данных из ОЗУ и на передачу считанных данных во второе устройство. Для этого в User IP блок коммутации Interconnect через регистры User IP настраивался на соединение выхода RX_FIFO с входом EP_TX_FIFO. Устройства 2-7 были настроены на передачу данных в следующие по цепочке. В них блок Interconnect был настроен на соединение EP_RX_FIFO с входом EP_TX_FIFO. Наконец, восьмое устройство было запрограммировано на передачу принятых в EP_RX_FIFO данных через TX_FIFO обратно в ОЗУ. После окончания передачи хост сравнивал данные. В данном эксперименте также измерялась скорость передачи данных в тракте. Она была равна скорости чтения из ОЗУ одним устройством ПЛИС.

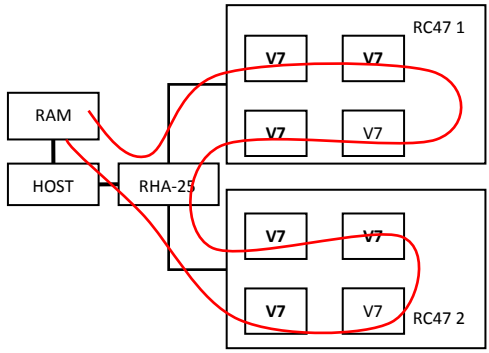

Рис. 6 Схема передачи данных HOST-ПЛИС-HOST

Удостоверившись в корректности передачи данных можно было переходить к измерению скорости. Из-за того, что максимальная длина пакетов, поддерживаемая чипсетом, была равна 128 байтам, параметр MAX_PAYLOAD_SIZE для всех коммутаторов и оконечных устройств в системе был установлен равным 128 байт. Поэтому по умолчанию передача данных между ПЛИС шла пакетами такой же длины. Однако было сделано наблюдение, что промежуточные PCI Express коммутаторы, находящиеся на платах RC-47 и адаптере RHA-25, а также интерфейсы PCI Express внутри ПЛИС поддерживали длину пакета равную 256 байт. При этом сам чипсет не участвовал в передаче данных между ПЛИС. Было сделано предположение, что если настроить MAX_PAYLOAD_SIZE для всех устройств в тракте передачи данных ПЛИС-ПЛИС равным 256, то можно будет запустить передачу пакетов длиной 256 байт, несмотря на то, что чипсет поддерживает только 128.

Для изменения параметра MAX_PAYLOAD_SIZE с помощью Linux команды setpci была проведена запись в регистр PCI Express Device Control для всех устройств и портов коммутаторов на платах RHA-25 и RC-47. Также был изменен блок Rosta DMA Engine так, чтобы можно было генерировать пакета длиной 256 байт. После чего действительно было возможно организовать передачу данных с блинной пакетов 256, и скорость передачи данных возросла.

Измерения скорости записи ПЛИС-ПЛИС были проведены для разного количества одновременных передач. Результаты представлены в таблице 3.

Таблица 3. Зависимость скорости записи ПЛИС-ПЛИС от количества одновременных передач при длинах пакетов 128 и 256 байт.

Количество передач	1	2	3	4	8.1	8.2
Скорость при payload = 128, МБ/с	1603	1549	1530	1520	1520	808
Скорость при payload = 256, МБ/с	1740	1704	1696	1685	1685	870

В первых четырех случаях передачи шли между устройствами на одной плате RC47. Максимальная скорость записи была получена во время одной передачи данных и составила 1603 МБ/с для длины пакета 128 байт и 1740 МБ/с для длины пакета 256 байт. В обоих случаях скорость составляла 99% от максимально возможной для соответствующей длины пакета.

Схемы взаимодействия устройств для случаев 8.1 и 8.2 представлены на рис.7.

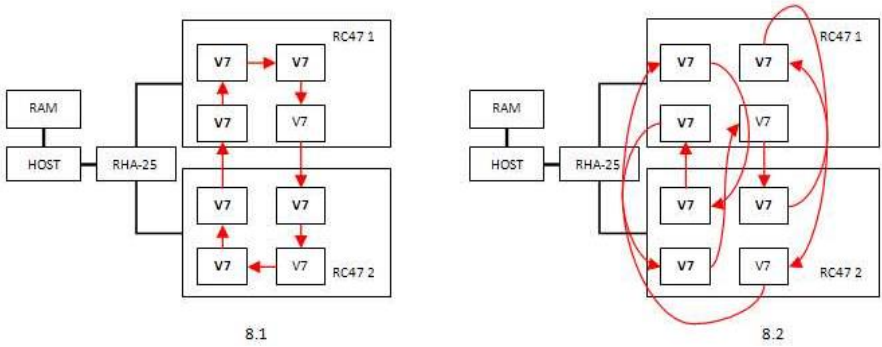

Рис. 7 Взаимодействие ПЛИС в случае 8ми одновременных передач

В случае 8.1 данные передавались последовательно от ПЛИС к ПЛИС сначала по кругу на одной плате, а затем через внешнее кабельное соединение и адаптер RHA-25 передача продолжалась в ПЛИС на другой плате. На второй плате данные также последовательно переписывались из одной ПЛИС в другую, а затем возвращались на первую. В итоге через адаптер RHA-25 в каждую сторону шло по одной передаче данных, и кабельное соединение PCI Express x4 Gen 3.0 никак не ограничивало ее скорость. В случае 8.2 данные передавались от одной ПЛИС к другой, но находящейся на другой плате. В итоге через адаптер RHA-25 шло 8 одновременных передач, по 4 в каждую сторону. Для длины пакета 128 байт скорость общего потока данных в одну сторону был равен 4*1520 = 6080 МБ/с, что превышает максимальную

скорость для канала PCI Express x4 Gen 3.0 равную 3225 МБ/с. Поэтому скорость канала должна была быть поделена поровну между устройствами, а средняя скорость передачи данных в каждой паре должна была бы стать 3225/4 = 806 МБ/с. Что подтверждается измеренным значением в 808 МБ/с. Поэтому можно утверждать, что канал связи плат между собой оказался задействован на 100%. Такие же рассуждения можно сделать и для передачи пакетов длиной 256 байт.

Наконец, был проведен эксперимент по измерению латентности передачи данных. Идея эксперимента была следующей. ПЛИС А посылает данные в ПЛИС В и запускает аппаратный таймер. Часы в разных устройствах сложно синхронизовать, поэтому используются таймер только в ПЛИС А. Как только ПЛИС В получает данные, она их сразу же начинает записывать обратно в ПЛИС А. Таймер останавливается, как только ПЛИС А получает первый байт данных. Значение таймера представляет собой удвоенную латентность передачи данных между блоком User IP одной ПЛИС и User IP другой. Латентность измерялась в двух случаях: при передачи между ПЛИС на одной плате RC-47 (рис. 8 А) и между ПЛИС на разных платах (рис. 8 В).

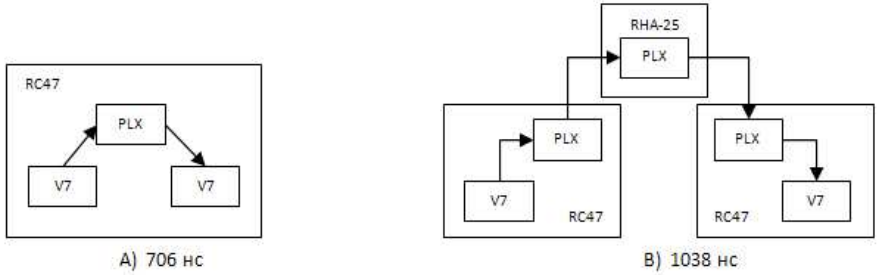

Рис. 8 Схемы передачи данных при измерениях латентности

При передаче данных между устройствами на одной плате RC-47 в тракте передачи был один коммутатор PLX, и задержка была равна 706 нс. При передаче с плату на плату было три промежуточных коммутатора, а задержка была равны 1038 нс. По этим данным можно определить задержку, возникающую в ПЛИС и вносимую коммутатором. Задержка в ПЛИС на приеме и передачи оказывается равной 270 нс, а в коммутаторе 166 нс, что хорошо согласуется с заявленной фирмой PLX Technology латентностью своих коммутаторов равной 150 нс.

6. Заключение

В данной работе была описана передача данных по шине PCI Express с одновременным участием нескольких ПЛИС. При записи в ОЗУ была получена скорость равная 90% от максимальной для соединения PCIe x4 Gen 2.0 при длине пакета 128 байт (1451 МБ/с). При чтении из ОЗУ скорость составила 1000 МБ/с. В случае одновременных передач ПЛИС-HOST скорость передачи данных не уменьшалась пока количество одновременных передач не насыщало узкий канал связи с хостом, в дальнейшем канал использовался на 100%, а его пропускная способность разделялась равномерно между устройствами.

Во время передачи ПЛИС-ПЛИС удалось запустить обмен пакетами длиной 256 байт, хотя чипсет хост компьютера поддерживал только 128. В этом случае удалось получить скорость равную 1740 МБ/с, что составляет 99% от максимальной скорости передачи интерфейса PCI Express x4 Gen 2.0 при длине пакета 256 байт. Также было показано, что возможно запустить несколько одновременных передач ПЛИС-ПЛИС, и пока суммарная скорость передачи не превышает пропускную способность общего канала, скорость отдельных передач не уменьшается, а затем канал используется на 100%, а его пропускная способность делится поровну между устройствами.

Была измерена латентность передачи данных ПЛИС-ПЛИС, которая составила 706 нс для одного промежуточного коммутатора и 1038 нс для трех.

Все это позволяет считать, что подход, основанный на использовании ПЛИС и IP ядра PCI Express интерфейса Xilinx и коммутаторов PLX Technology, может быть эффективно использован для организации обмена данными между большим количеством ПЛИС, подключенным к локальной PCI Express шине компьютера.

Список литературы

[1]. Rosta LTD, 2013. http://www.rosta.ru/.

[2]. nVidiaCorporation. GPUDirect. https://developer.nvidia.com/gpudirect.

[3]. Ra Inta, David J. Bowman, Susan M. Scott, "The “Chimera”: An Off-The-Shelf CPU GPGPU FPGA Hybrid Computing Platform", *International Journal of Reconfigurable Computing*, 2012.

[4]. Bruno da Silva, An Braeken, Erik H. D'Hollander, Abdellah Touhafi, Jan G. Cornelis, Jan Lemeire, "Performance and toolchain of a combined GPU/FPGA desktop", *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*, 2013.

[5]. Ray Bittner, Erik Ruf, Alessandro Forin, "Direct GPU/FPGA Communication Via PCI Express", *Cluster Computing*, 2013.

[6]. Sunita Jain, Guru Prasanna, "Point-to-Point Connectivity Using Integrated Endpoint Block for PCI Express Designs", Xilinx Corporation, XAPP869, 2007.

[7]. 7 Series FPGAs Integrated Block for PCI Express v1.7 Product Guide, Xilinx Corporation, 2012.

[8]. Alex Goldhammer, John Ayer, "Understanding Performance of PCI Express Systems", Xilinx Corporation, WP350, 2008.

Direct data transfer between FPGAs Virtex-7 via PCI Express bus

Yu.A. Rumyantsev <rumyantsev@rosta.ru>
Rosta Ltd, Moscow, Russia

Abstract. This article describes two types of data transfers via PCI Express bus involving several FPGA. The first one is a simultaneous DMA data transfer between the system memory and different FPGA chips. The second one is a simultaneous direct data transfer between different FPGA.

The data transfer speed was measured for both cases with results being about 99% from maximum speed for PCIe x4 Gen 2.0 link for the direct transfer between FPGAs (1603 MB/s for 128 bytes payload and 1740 MB/s for 256 bytes payload). The direct data transfer latency was also measured to be 0,7 us for one intermediate PCIe switch and 1 us for three intermediate switches.

Also the effect of simultaneous transfers on data transfer speed was studied with the result that, as long as the aggregate transfer speed does not overcome the shared link bandwidth, each transfer is performed on its maximum speed; after that the shared link utilization reaches 100% with its bandwidth being distributed equally between individual transfers.

Keywords: FPGA; PCI Express.

References

[9]. Rosta LTD, 2013. http://www.rosta.ru/.
[10]. nVidiaCorporation. GPUDirect. https://developer.nvidia.com/gpudirect.
[11]. Ra Inta, David J. Bowman, Susan M. Scott, "The “Chimera”: An Off-The-Shelf CPU GPGPU FPGA Hybrid Computing Platform", International Journal of Reconfigurable Computing, 2012.
[12]. Bruno da Silva, An Braeken, Erik H. D'Hollander, Abdellah Touhafi, Jan G. Cornelis, Jan Lemeire, "Performance and toolchain of a combined GPU/FPGA desktop", Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays, 2013.
[13]. Ray Bittner, Erik Ruf, Alessandro Forin, "Direct GPU/FPGA Communication Via PCI Express", Cluster Computing, 2013.
[14]. Sunita Jain, Guru Prasanna, "Point-to-Point Connectivity Using Integrated Endpoint Block for PCI Express Designs", Xilinx Corporation, XAPP869, 2007.
[15]. 7 Series FPGAs Integrated Block for PCI Express v1.7 Product Guide, Xilinx Corporation, 2012.
[16]. Alex Goldhammer, John Ayer, "Understanding Performance of PCI Express Systems", Xilinx Corporation, WP350, 2008.