Реализация многоочередного реконфигурируемого инжекционного конвейера в адаптере высокоскоростной коммуникационной сети как решение проблемы организации эффективного взаимодействия адаптера с множеством процессов

К.А. Курочкин

АО «НИЦЭВТ»

В настоящей статье рассматривается проблема организации эффективного взаимодействия множества процессов, выполняемых на современных многоядерных вычислительных системах, с адаптером высокоскоростной коммуникационной сети «Ангара» как с устройством PCI Express. В качестве решения данной проблемы предлагается замена множества стандартных инжекционных конвейеров многоочередным реконфигурируемым конвейером, позволяющим гибко распределять аппаратные ресурсы такого конвейера между инжектирующими процессами и обеспечивающим максимальную пропускную способность для произвольного числа процессов при минимальных затратах аппаратуры по сравнению со стандартным подходом. В статье приводится описание микроархитектуры данного конвейера и сравнение предложенной реализации с существующим подходом, а также даётся оценка эффективности нового решения.

Ключевые слова: адаптер коммуникационной сети, PCI Express, инжекционный конвейер, кольцевой буфер, PIO, масштабируемость, реконфигурация.

1. Введение

Учитывая тенденции современной вычислительной техники ко все большей параллезации и наращиванию количества одновременно выполняющихся процессов на узле, интерфейс взаимодействия процессов с коммуникационным адаптером, пропускная способность которого растёт меньшими темпами становится тонким местом, ограничивающим производительность сети вычислительного кластера в целом. Обычно адаптер коммуникационной сети выполняется в форм-факторе карты расширения PCI Express [3, 8, 9], что является наиболее универсальным и коммерчески оправданным решением, однако влечет за собой дополнительные накладные расходы на организацию доступа процессов к коммуникационной сети, особенно при пересылке пакетов малого размера (до 64 байт). Удаленность адаптера от вычислительных ядер, а также необходимость поддерживать протокол взаимодействия системы с периферийным устройством PCI Express приводит к усложнению аппаратной и программной частей сетевого стека, отвечающих за передачу пакетов в сеть. Реализация адаптера коммуникационной сети и вычислительных ядер на одном кристалле позволяет обойти все эти проблемы [1], однако данная технология доступна лишь крупным компаниям и позволяет строить суперкомпьютерные кластеры с использованием оборудования только этого производителя, являясь, по сути, узкоспециализированным решением.

Основная же масса коммуникационных адаптеров реализует стандартный PCI Express интерфейс, используя два режима взаимодействия процессов с периферийным устройством — PIO и DMA. PIO (от англ. Programmed Input/Output) - является режимом, при котором процессор сам непосредственно пишет данные для отправки в область памяти, отображенную на ресурсы адаптера. При таком режиме тратится время CPU, однако обеспечивается минимальная задержка на передачу пакета в сеть. DMA (от англ. Direct memory Access) — режим, при котором процесс для передачи пакетов в сеть лишь указывает устройству, где они лежат в памяти, после чего их передача будет осуществлена самим устройством без участия самого процессора. Под словом «указывает» в данном случае подразумевается передача

процессом (в режиме PIO) специального сообщения устройству с информацией о начальном адресе и размере сетевого пакета, который устройству необходимо скопировать из памяти хост-системы. Такое сообщение обычно называется дескриптором сетевого пакета. На рисунке 1 проиллюстрирована разница между PIO и DMA пересылками данных, где видно, что для DMA передачи необходимо прохождение 3 транзакций по PCI Express шине между периферийным устройством и системной памятью, тогда как в PIO режиме посылается только 1 пакет.

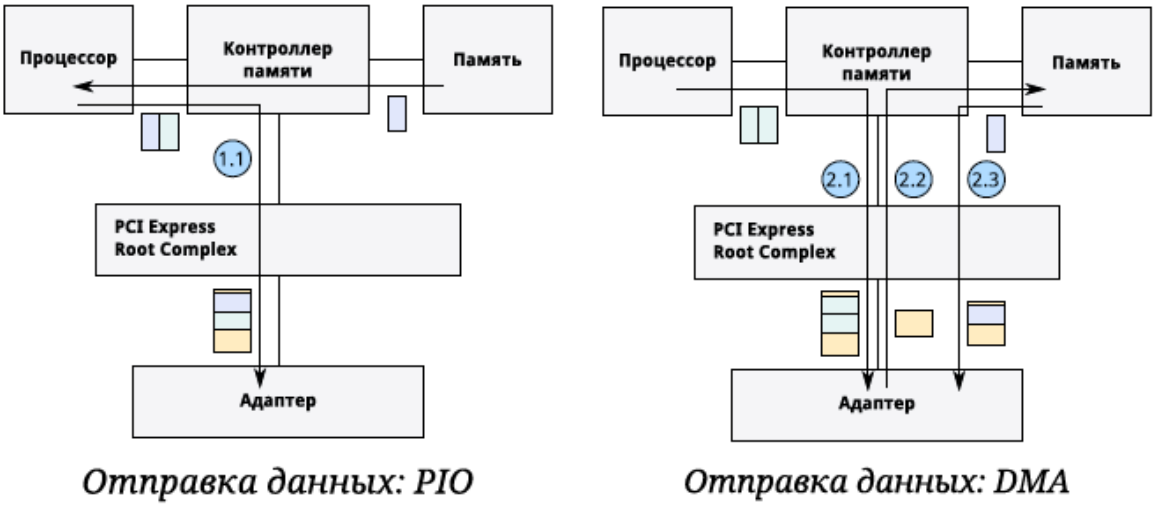

Рис. 1. Отличие PIO и DMA

Вследствие указанных выше различных особенностей, обычно, для коротких сообщений, где важна задержка передачи, используется PIO режим, а для длинных сообщений, где решающую роль играет пропускная способность — режим DMA. В некоторых архитектурах адаптеров для пакетов, передаваемых в режиме PIO, не выделяется отдельная аппаратная очередь, а реализуется общий механизм инжекции дескрипторов, где в зависимости от типа дескриптора сетевой пакет будет либо сформирован сразу (когда тело пакета содержится в самом дескрипторе), либо после чтения тела пакета из памяти в DMA режиме. Так, например, устроен адаптер сети Tianhe Express [2].

Для поддержки одновременной работы нескольких процессов в современных адаптерах реализуется множество аппаратных очередей для хранения дескрипторов и множество механизмов для передачи сообщений с низкой задержкой в режиме PIO. Примером такой архитектуры может служить адаптер Aries сети суперкомпьютера Cray Gemini, где реализовано 127 FMA-окон (Fast Memory Access) для передачи пакетов до 64 байт и 4 DMA канала, называемых BTE (Block Transfer Engine) для асинхронной передачи данных объемом до 4 ГБ [3].

В адаптере коммуникационной сети «Ангара» [10] также используется механизм очередей (инжекционных конвейеров) для отправки пакетов в сеть. В адаптере на базе СБИС ЕС8430 реализовано 10 очередей для инжекции пакетов в сеть в режиме PIO.

Дизайн большинства аппаратных очередей ориентирован на эксклюзивное использование одним процессом, так как предполагает некий механизм подсчёта кредитов для предотвращения переполнения буфера. Поскольку чтение кредитной информации непосредственно из адаптера является дорогой операцией, используется некоторый периодически обновляемый кэш, относительно которого вычисляется возможность записи очередного пакета.

С ростом количества процессов на узле, пропорциональное увеличение данных аппаратных блоков затруднено, так как это усложняет схему их синхронизации и взаимодействия (и, как следствие, physical design), увеличивает количество требуемых ресурсов для physical design, увеличивает расчётное энергопотребление и, в конечном счёте, неоправданно увеличивает стоимость кристалла, что приводит либо к необходимости применения программных методов разделения одного общего аппаратного ресурса между несколькими программными потоками, либо к нахождению иного способа организации аппаратных механизмов, отличного от

простого их копирования. Попыткой увеличить масштабируемость аппаратных конвейеров является методика совмещения нескольких очередей в одной FIFO-памяти, применяемая в маршрутизаторе Extoll[4]. Она позволяет использовать один блок RAM и внутреннюю логику FIFO между несколькими контекстами очередей, сберегая тем самым ограниченные ресурсы ПЛИС. Авторами заявляется экономия аппаратных ресурсов до 4 раз при использовании многоочередного FIFO-модуля по сравнению с обычными FIFO.

Еще одним методом, повышающим масштабируемость взаимодействия, является подход, освещенный в недавном совместном патенте Intel и Cray [5]. В данном подходе на стороне периферийного устройства имеется 1МБ аппаратной памяти для хранения сетевых пакетов. Этот 1МБ делится на подблоки по 64Б каждый. Также имеется логика для поддержания до 160 инжекционных контекстов. На каждый контекст может быть выделено от 1 до 1024 подблоков. Таким, образом имеется фикисированная аппаратная часть, которая однако рассчитана на 160 одновременно выполняющихся процесса, причем эти аппаратные ресурсы могут быть гибко распределены между инжектирующими процессами в зависимости от требуемой каждому доли пропускной способности канала PCIE.

Данные методики в той или иной степени были рассмотрены применительно к адаптеру сети Ангара, однако, из-за некоторых особенностей и ограничений архитектуры программного и аппаратного уровней потребовалось нахождение своего решения проблемы построения аппаратной схемы взаимодействия адаптера со множеством процессов перспективных вычислительных систем.

2. Описание модели взаимодействия процессов с адаптером сети Ангара

Согласно традиционной программной модели взаимодействия хост-системы с периферийным устройствами, в адресном пространстве системы выделяются регионы памяти, путем записи или чтения в которые процессы осуществляют доступ к ресурсам данного устройства. Адаптер сети Ангара реализует несколько таких регионов[11], одним из которых является регион, отвечающий за инжекцию пакетов в основные инжекционные конвейеры. В составе адаптера находятся несколько (от 16 до 64) основных аппаратных инжекционных конвейеров, задача которых принять и сохранить пакеты, полученные через PCI Express интерфейс от вычислительных процессов и затем передать их в сеть для дальнейшей маршрутизации на целевой узел. На адресное пространство данного региона линейно отображаются внутренние буферы инжекционных конвейеров, и для организации поточной записи эти области организованы в виде кольцевого буфера. Запись в область кольцевого буфера осуществляется по последовательным адресам и при записи по индексу, превышающему размер области, происходит перезапись ячеек буфера. Т.е. при записи данных по адресам N-3, N-2, N-1, N, N+1 (N – суммарное кол-во ячеек буфера), запись по адресу N+1 будет осуществлена по 0 ячейке буфера. Таким образом, буфер «закольцован» - за последней ячейкой снова следует 0. Данная схема аллокации представлена на рисунке 2.

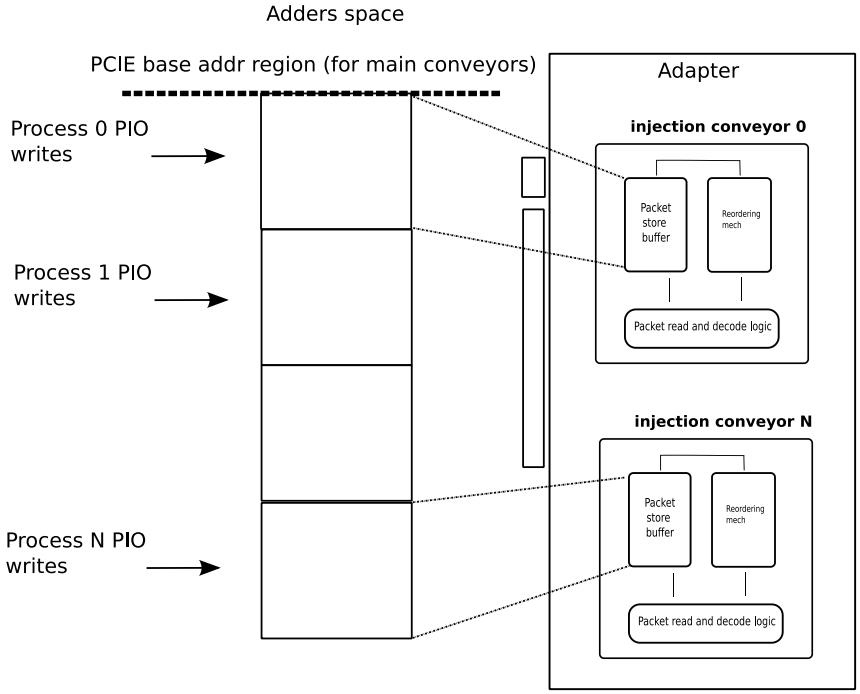

Рис. 2. Схема аллокации инжекционных конвейеров

Запись данных процессом происходит с помощью выполнения инструкций типа MOV архитектуры x86 и является PIO (Programmed Input/Output). PIO, тем не менее, не обеспечивает максимального темпа передачи таких сообщений без применения режима WC (Write Combining), реализованного в архитектурах современных процессоров Intel и AMD. Режим WC подразумевает наличие специального набора буферов емкость обычно в 64 байта (1 кэш строка современных процессоров), которые накапливают в себе отдельные транзакции по 8 байт и выдают затем «собранные» (от англ. combined) транзакции с минимальными накладными расходами на системную шину. Особенность данных буферов состоит в том, что порядок записей на выходе из этого буфера может отличаться от программного порядка – так называемый weak ordering тип памяти [6, 7]. Именно для восстановления изначального программного порядка записей в структуре инжекционного конвейера предусмотрен аппаратный механизм «Reordering mech».

Для реализации семантики кольцевого буфера инжектирующему процессу помимо имеющего программного указателя записи необходимо наличие указателя чтения. В архитектуре адаптера предусмотрен отдельный аппаратный механизм сброса по определенному адресу в память системы указателей чтения всех конвейеров в режиме DMA.

Структура инжекционного конвейера адаптера представлена на рисунке 3. Как видно из рисунка конвейер состоит из трех основных подблоков. Подблок «Packet store buffer» является набором RAM памятей и в нем сохраняются данные, пришедшие по PCI Express интерфейсу, подблок “Reordering mech” реализует логику продвижения указателя записи для буфера «Packet store buffer». Блок «Packet read and decode logic» на основании этого указателя записи, осуществляет чтение и декодирование заголовков пакетов, а далее при условии наличия пакета целиком в буфере, осуществляет его чтение и передачу его далее в сеть. Также этот блок реализует указатель чтения данного буфера, который в свою очередь необходим для контроля потока, для исключения случая переполнения буфера. Таким образом, в инжекционном конвейере осуществляется абстракция FIFO-очереди (или по-другому – кольцевого буфера), тогда как в реальности отдельные записи в буфер могут приходить по произвольным адресам из-за режима WC.

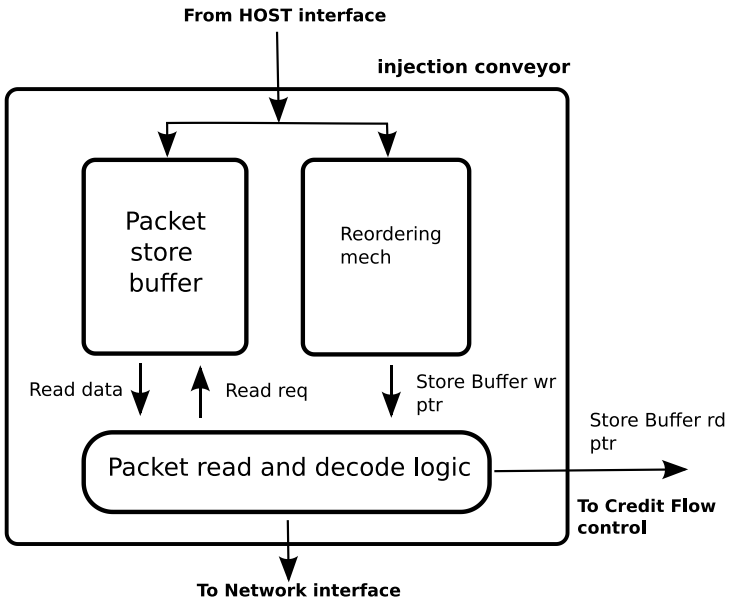

Рис. 3. Структура инжекционного конвейера

В составе адаптера имеется набор инжекционных конвейеров. Они связаны между собой демультиплексером входного потока со стороны PCIE на входе и round robin арбитром на выходе, как представлено на рисунке 4.

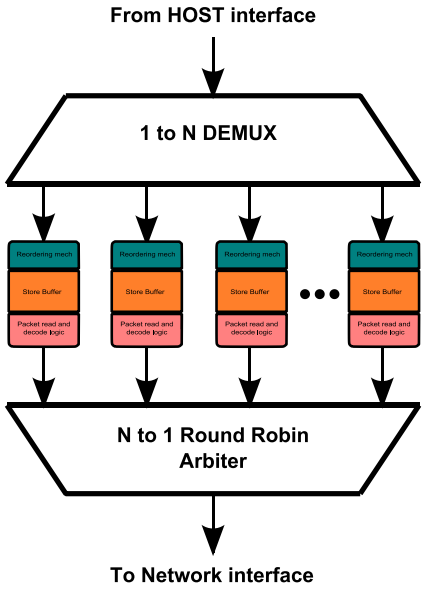

Рис. 4. Инжекционные конвейеры

Основная проблема данного подхода заключается в том, что, в сущности, последовательный поток данных разделяется на N параллельных потоков и затем снова сходится в один. В каждый момент времени ресурсы N -1 инжекционных конвейеров будут простаивать. Более того размеры каждого конвейера не могут быть уменьшены вследствие необходимости поддержать максимальную пропускную способность для одного инжектирующего процесса. Минимальная глубина буфера инжекционного конвейера рассчитывается по следующей формуле:

$$D_{injbuf} = 2 \times Lat_{PCIE} \times BW_{PCIE} + Delay$$

Где Lat_{PCIE} - задержка шины PCI Express, BW_{PCIE} – пропускная способность шины PCI Express, Delay – задержка обновления указателя записи, введена для ограничения данного вида трафика по эжекционному каналу, измеряется в байтах и показывает через сколько байт принятой информации следует сбросить на хост обновленные указатели чтения для данного

конвейера. Подставив в эту формулу значения для интерфейса PCI Express Gen2 и значение Delay = 2048 байт, получим размер буфера, равный:

$$D_{injbuf} = 2 \times 250ns \times 64 \frac{Gb}{s} + 2048B = 6048B$$

И округлив данное значение вверх до степени двойки для удобства использования кольцевого буфера, получим:

$$D_{injbuf} = 8192B = 8KB, \quad (1)$$

Что составляет довольно значительную величину для внутрикристалльной памяти. Также значительная часть логики и RAM памяти используется в механизме Reordering, который также дублируется по всем конвейерам. Все эти факторы в совокупности приводят к очень плохой масштабируемости данной схемы. Учитывая тенденции современной вычислительной техники ко все большему увеличению одновременно выполняющихся процессов на узле и относительную неэффективность программных методов синхронизации процессов, данная схема требует существенного пересмотрения для использования адаптера с будущими вычислительными платформами с числом процессов от 128 до нескольких тысяч.

3. Описание микроархитектуры многоочередного реконфигурируемого инжекционного конвейера

Решением проблемы масштабируемости инжекционных конвейеров предлагается замена набора обычных конвейеров многоочередным конвейером. Основной идеей многоочередного конвейера является разделение общих ресурсов, таких как механизм реордеринга, буфер хранения данных, декодер заголовков пакетов, механизм чтения пакетов из памяти между контекстами нескольких процессов посредством их арбитража с помощью round robin арбитров. Общая структура многоочередного конвейера представлена на рисунке 5.

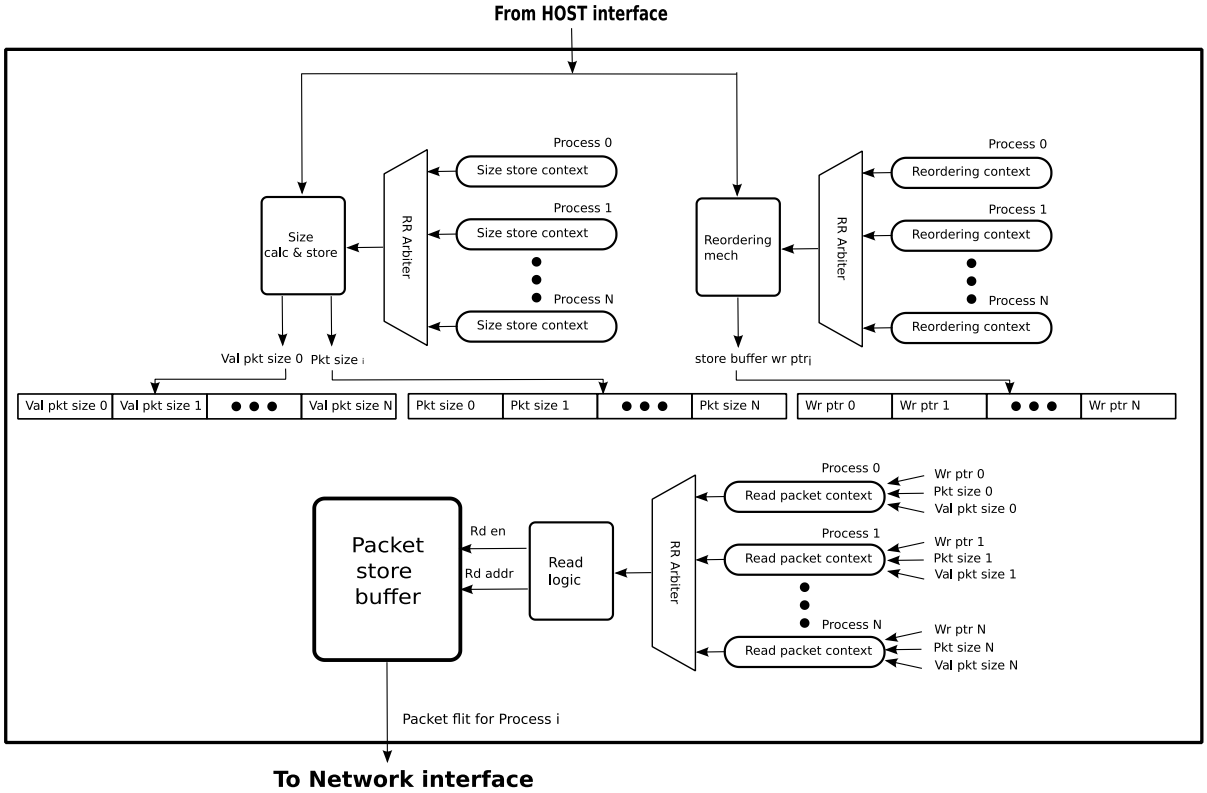

Рис. 5. Схема многоочередного инжекционного конвейера

Конвейер разбит на три главные части – блок «Size calc&store», «Reordering mech» и блок «Packet store buffer». Блоки «Size calc&store» и «Reordering mech» работают параллельно, предоставляя для каждого процесса указатель записи соответствующей ему части буфера пакетов «Packet store buffer» и размер текущего пакета, который необходимо считать из этого

буфера. На основании указателей чтения и записи буфера пакетов, а также текущего размера пакетов для каждого процесса формируется флаг готовности читать сетевой пакет из буфера по следующей формуле:

$$Rdy\_to\_read_i = (Busy\_space_i \geq Pktsize_i) \& Val\_pktsize_i, \quad (2)$$

Где $Rdy\_to\_read_i$ - готовность i-го процесса читать пакет из памяти, $Busy\_space_i$ – количество занятых слотов в буфере данных, $Pktsize_i$ и $Val\_pktsize_i$ – значение предвыбранного размера пакета и действительность предвыбранного размера пакета. Значение $Busy\_space_i$ контекст процесса получает через механизм “Reordering mech”, а значения $Pktsize_i$ и $Val\_pktsize_i$ - через механизм “Size calc&store”.

Из множества контекстов процессов, для которых выполняется условие (2) по round-robin алгоритму выбирается один процесс, контекст которого передается в блок “read logic”. В данный контекст входят:

1. Номер процесса;
2. Текущий указатель записи буфера данного процесса;
3. Размер считываемого сетевого пакета для данного процесса;

На основании этих данных блок “read logic” вычитывает необходимый сетевой пакет из буфера и передает его дальше по конвейеру в сеть. После окончания чтения текущего процесса сеанс выбора нового процесса повториться, и из буфера будет вычитан пакет уже другого процесса. Таким образом, аппаратные ресурсы равномерно распределяются между всеми активными контекстами процессов.

Принципиальным отличием многоочередного конвейера от традиционной версии является наличие механизма «size calc&store». В традиционной схеме декодирование заголовка пакета происходит после чтения его из буфера пакетов. Применение такого же принципа в многоочередном конвейере приводит к возникновению принципиального ограничения пропускной способности конвейера. Дело в том, что при попытке организовать доступ нескольких очередей к такой схеме возникает проблема простоя процессов в ожидании считанного заголовка пакета предыдущим процессом. Задержка на чтение данных из буфера пакетов составляет 4 такта с момента подачи строба чтения и соответствующего адреса чтения для данного процесса. Это время данный процесс находится в режиме ожидания заголовка, который затем необходимо декодировать и сравнить с занятым местом в буфере, на что тратится дополнительно 2 такта. Итого 6 тактов процесс находится в режиме ожидания, прежде чем сформировать еще один запрос на чтение уже всего пакета при выполнении условия (2). Возможным решением данной проблемы является либо выдача спекулятивных запросов чтение при неизвестном пока условии (2), либо же выдача запросов на чтение заголовков для других процессов. Оба этих вариантов приводят к сильному усложнению либо к алгоритмов арбитража, либо к дополнительной логике отброса преднакачанных флитов в случае невыполнения условия (2). В любом случае дизайн оказывается сильно перегружен логикой, что отрицательно сказывается на его временных характеристиках. Поэтому согласно принципу simple is clever было предложен третий вариант организации чтения флитов из памяти.

Устройство механизма «Size calc&store» представлено на рисунке 7. Основой данного механизма является RAM память размеров пакетов. Каждый входной флит проходит через декодер заголовка пакетов, который, исходя из полей заголовка, вычисляет размер сетевого пакета. Это вычисленное значение пишется в память store size buffer. Глубина данной памяти равна глубине основного store буфера инжекционного конвейера. Для определения валидности ячеек в данной памяти вместе с размером пишется дополнительный разряд адреса. Память размеров читается по хранящимся для каждого процесса указателям чтения данной памяти, который также расширен на 1 разряд. Изначально значение указателя памяти для процесса равно 0, а сама память инициализируется значением 1’b1 в старшем разряде. Действительность ячейки при чтении определяется совпадением старших разрядов указателя и ячейки памяти. Действительно, при записи со стороны хоста пакета, первая запись будет осуществлена по 0 адресу и в старший разряд 0 ячейки будет записано 0 значение. Введение данного механизма позволяет избежать обнуления валидностей ячеек памяти, что приводит к общему упрощению схемы. Расширение на один адресный разряд влечет изменение только в схеме аллокации

инжекционного конвейера в адресном пространстве системы – для каждого конвейера будет выделено в два раза больше памяти, что, однако, никак не сказывается на эффективности организации кольцевого буфера. Инжектирующий процесс будет писать с адресов 0 по N-1, а с 0 по 2*N -1. При записи в основной буфер пакетов данный разряд просто игнорируется.

При чтении действительного размера пакета, указатель чтения памяти размеров пакетов увеличивается следующим образом:

$$Pktsize_{rd\_ptr\_next} = Pktsize_{rd\_ptr\_old} + Pktsize_{out}, \quad (3)$$

Данная схема позволяет предвыбрать размеры двух пакетов для каждого процесса.

С введением отдельного буфера для размеров пакетов, схема организации чтения флитов из основного буфера заметно упрощается, пакеты из него вычитываются сразу и целиком.

В многочередном конвейере в основном буфере используется суммарно 8 КБ памяти, разделяемой между всеми процессами, однако для поддержки однопоточной производительности требуется, чтобы одному процессу были доступны все 8КБ (согласно (1)). Для этого в многоочередном конвейере предусмотрен механизм реконфигурации конвейера. Данный механизм позволяет статически разбить данный конвейер на меньшее, чем максимально возможное число контекстов, однако каждому контексту будет доступно больше чем 8Кб\ Nmax. Реконфигурация конвейера осуществляется путем записи в специальные программно доступные регистры адаптера. Регистр состоит из N бит, где N – максимально доступное число процессов в конвейере. i-ый бит в данном реконфигурационном регистре обозначает активность i-го контекста процесса, т.е. к примеру, для 8-очередного конвейера значение 8’b1111_1111 будет соответствовать 8 активным процессам по 8Кб\ Nmax памяти на каждый. Значение 8’b0101_0101 будет соответствовать 4 активным процессам и 8Кб\(Nmax*2) на каждый. Следует заметить, что в данной конфигурации объединяются памяти четного и нечетного контекстов, причем активным является всегда четный контекст и комбинация 8’b1010_1010 не допускается. Далее согласно тому же принципу действительны разбиения 8’b0001_0001 и 8’b0001_0001. В последнем случае одному 0 процессу доступна вся память инжекционного конвейера. Данная схема позволяет гибко распределять ресурсы конвейера, например, также доступны разбиения 8’b0101_0001 или 8’b0001_1101, когда разным процессам выделяется разное количество памяти. Однако необходимо отметить, что этот механизм является статическим и для реконфигурации конвейера требуется его полный сброс и приостанов всех инжектирующих процессов, а также аллокация новой конфигурации в памяти.

4. Результаты

На данный момент полностью реализована RTL-модель многоочередного инжекционного конвейера, которая была промоделирована в среде IUS от Cadence. Тестовое окружение для данного блока было также написано на языке Verilog и представляет собой поведенческую модель инжектирующих процессов, включая имитацию работы механизма Write Combining. Для оценки эффективности предложенного решения введены следующие метрики:

BW_{raw} - суммарная пропускная способность инжекционного конвейера с учетом головных флитов. При частоте работы схемы = 500 Mhz, теоретическое пиковое значение $BW_{\max}$ = 64 Гбит/с, что соответствует выдаче действительного флита сетевого пакета из конвейера на каждом такте. Следует, однако, учесть, что при передаче сетевого пакета далее в сеть к нему добавляется хвостовой флит с информацией канального уровня, и, таким образом, целевая пропускная способность, меньше которой пропускная способность конвейера быть не должна равна:

$$BW_{target} = \frac{pktsize}{pktsize+1} * BW_{\max}$$

При условии работы инжектирующих процессов на максимально возможной пропускной способности график зависимости пропускной способности от размера инжектируемых пакетов должен стремиться к этой величине. Любое отклонение в меньшую сторону будет свидетельствовать о том, что инжекционный конвейер будет являться бутылочным горлышком на инжекционном тракте, что является недопустимым.

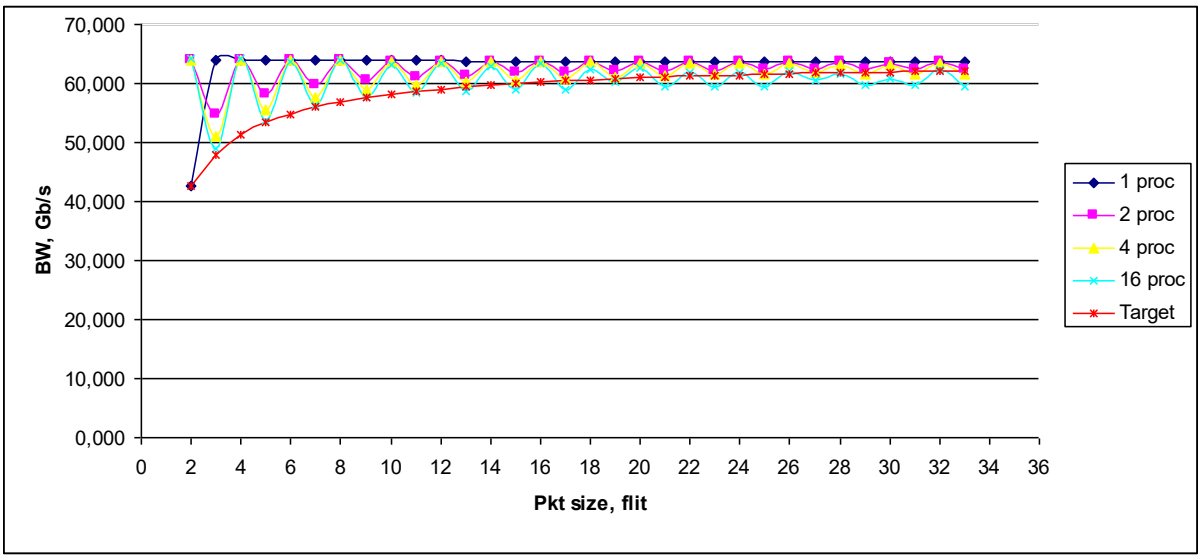

Рис. 6. Зависимость BW от размера пакета для многоочередного инжекционного конвейера

На рисунке 6 представлен график такой зависимости для многоочередного инжекционного конвейера, из которого следует, что данная характеристика конвейера близка к теоретической. Также из графика заметны падения пропускной способности для количества процессов больше одного для пакетов нечетной длины. Эта особенность работы связана с наличием в RAM памяти буфера механизмов ecc и как следствие техники multycycle, которая не позволяет читать из памяти подряд по адресам с совпадающей четностью. Т.к. контексты процессов при работе блока равномерно переключаются после каждого считанного пакета, для нечетных длин пакетов это правило нарушается и механизм чтения пакетов из памяти вынужден вставлять такт простоя для избежания ошибки работы схемы.

Следующей по важности метрикой эффективности инжекционного конвейера является значение задержки Lat от загрузки канала. В результате проведения замеров оказалось, что в независимости от числа одновременно инжектирующих процессов и степени загрузки канала задержка остается постоянной и составляет 14 тактов. Это является хорошим показателем и указывает на то, что в схеме не теряются такты при переключении контекста инжектирующих процессов.

Так как инжекция пакетов через PCI Express интерфейс осуществляется путем опустошения WC-буфера, размер транзакций почти всегда будет равен длине кэш строки, и сетевые пакеты, большие по длине, чем 64Б, будут записываться в буфер инжекционного конвейера кусками. Выдача пакета в сеть произойдет только после того, как он полностью попадет в буфер, что естественным образом скажется на задержке. На рисунке 7 представлена зависимость задержки от длины пакета, где видно, что задержка растет ступенчато с увеличением длины пакета по вышеупомянутым причинам.

Рис. 7. График зависимости задержки от длины пакета

В таблице 1 приведена информация о затрачиваемых ресурсах ПЛИС (данные предоставлены для ПЛИС фирмы Xilinx семейства Virtex7) для разного количества доступных контекстов процессов, доступных в многоочередном конвейере, а также приведена информация для стандартного подхода.

Таблица 1. Сравнение потребления аппаратных ресурсов

	1	2	4	8	16	32
Multi Queue injection conveyor						
LUT/FF	-	913	1295	1920	3142	3959
RAM	-	5	5	5	5	5
Freq, Mhz	-	311.917	306.077	296.621	270,911	251.74
Set of standart injection conveyors						
LUT/FF	1432	2864	5728	11456	22912	45824
RAM	6	12	24	48	96	192
Freq, Mhz	217.647	217.647	217.647	217.647	217.647	217.647

Данные результаты следует анализировать с поправкой на несколько фактов. Во-первых, в многоочередном инжекционном конвейере используется иной механизм реордеринга, занимающий меньше аппаратных ресурсов по сравнению с механизмом в стандартном конвейере. Во-вторых, в многочередном конвейере нет части выходной логики, отвечающей за мультиплексирование сетевого пакета на несколько направлений, а также логики декодирования и преобразования некоторых полей заголовков сетевых пакетов. Объем данной логики может быть оценен в пределах 500-1000 LUT/FF, и это значение будет являться константной прибавкой для многоочередгого конвейера с любым числом очередей.

Тем не менее, учитывая данные обстоятельства, все равно можно сделать вывод о том, что многоочередной конвейер во много раз превосходит стандартный подход по показателям масштабируемости, превосходя стандартную схему более чем в 10 раз. Это позволяет отодвинуть порог масштабируемости инжекционных конвейеров на порядок и использовать существующую схему кольцевого буфера для организации взаимодействия адаптера с будущими вычислительными узлами.

Литература

1. Dong Chen, Noel A., Eisley Philip, Heidelberger, Robert M., Senger Yutaka Sugawara The IBM Blue Gene/Q Interconnection Fabric, IEEE 2012

2. Zhengbin Pang, Min Xie, Jun Zhang, Yi Zheng, Guibin Wang, Dezun Dong, Guang Suo The TH Express high performance interconnect networks, Front. Comput. Sci., 2014
3. Bob Alverson, Edwin Froese, Larry Kaplan, Duncan Roweth Cray XC ® Series Network, WP-Aries01-1112, 2012
4. Dipl. Inf Benjamin Ulrich Geibaus. Hardware Support for Efficient Packet Processing. Heidelberg, Mannheim, 2012 P. 12-23
5. Mark Debbage, Yatin M. Mutha Sending Packets using optimized PIO write sequences without SFENCEs, Patent Application Publication, 2015
6. Advanced Micro Devices AMD64 Architecture Programmer's Manual Volume 2: System Programming, May 2013 P. 161-179
7. Intel® 64 and IA-32 Architectures Software Developer's Manual Combined Volumes: 1, 2A, 2B, 2C, 3A, 3B and 3C, February 2014 P. 2310-2313, 2317-2318
8. http://www.mellanox.com/pdf/whitepapers/IB_Intro_WP_190.pdf
9. Mark S. Birrittella, Mark Debbage, Ram Huggahalli, James Kunz, Tom Lovett, Todd Rimmer, Keith D. Underwood, Robert C. Zak Intel® Omni-Path Architecture, IEEE, 2015
10. Корж А.А, Макагон Д. В., Бородин А. А., Жабин И. А., Куштанов Е. Р., Е. Л., Сыромятников Е. Л., Черемушкина Е. В.Отечественная коммуникационная сеть 3D-тор с поддержкой глобально адресуемой памяти для суперкомпьютеров транспетафлопсного уровня производительности, ПАВТ, 2010
11. Дмитрий Макагон, Евгений Сыромятников Сети для суперкомпьютеров, Открытые Системы №7, 2011