Сетевой адаптер Corundum на основе ПЛИС

По материалам [1]

Введение

Контроллер сетевого интерфейса (network interface controller или NIC) является шлюзом, через который компьютер взаимодействует с сетью. NIC создаёт мост между программным стеком и сетью, определяющий сетевой интерфейс. Набор функций этого интерфейса и их реализации развиваются очень быстро в результате потребностей в повышении скорости линий и расширении возможностей NIC в части поддержки высокопроизводительных распределенных вычислений и виртуализации. Рост скорости в линии ведёт к необходимости реализации многих функций NIC на аппаратном уровне. В то же время новые сетевые функции, такие как точное управление передачей для множества очередей, требуют реализации передовых протоколов и архитектуры сетей.

В соответствии с потребностями в открытой платформе для разработки протоколов и архитектуры сетей на реальных скоростях передачи данных предложена платформа с открытым исходным кодом для создания прототипов NIC на основе FPGA. Эта платформа, названная Corundum [1], способна работать со скоростью не менее 94 Гбит/с, имеет открытый исходный код и вместе с драйвером может применяться в полном сетевом стеке. Решение является переносимым и компактным, поддерживая множество разных устройств и сохраняя ресурсы для дополнительной настройки даже в небольших устройствах. Модульное и расширяемое устройство Corundum позволяет оптимизировать программно-аппаратные решения для создания и тестирования сетевых приложений в реальных условиях.

Предпосылки

Мотивы разработки Corundum становятся понятными, если рассмотреть распределение функций сетевого интерфейса между программной и аппаратной частью имеющихся NIC. Аппаратные функции NIC делятся на две основные категории. Первая категория включает простые функции разгрузки, принимающие на себя часть работы процессора (CPU) по обработке каждого пакета, такие как расчёт контрольной суммы или свёртки (hash) и сегментация, что позволяет сетевому стеку выполнять групповую (batch) обработку пакетов. Ко второй категории относятся функции, которые нужно выполнять на аппаратном уровне NIC для обеспечения высокой производительности и беспристрастности. К таким функциям относится управление потоками, распределение нагрузки и временные метки. Традиционно аппаратные возможности NIC реализуются с помощью фирменных микросхем, ориентированных на конкретные задачи (application-specific integrated circuit или ASIC). Это позволяет достичь высокой производительности при малой стоимости. Однако расширяемость таких ASIC сильно ограничена, а цикл разработки для расширения функциональности достаточно дорог и продолжителен [2]. Чтобы преодолеть эти ограничения, было разработано множество интеллектуальный (smart) и программных NIC. Интеллектуальные сетевые адаптеры обеспечивают программируемость обычно за счёт предоставления множества программируемых вычислительных ядер и аппаратных примитивов. Эти ресурсы могут применяться для выгрузки с хоста различных прикладных, сетевых и связанных с виртуализацией операций. Однако интеллектуальные NIC не всегда подходят для высокоскоростных линий и аппаратные функции могут быть ограничены [2].

Программные NIC обеспечивают большую гибкость за счёт программной реализации сетевых функций, минуя большую часть выгрузки в оборудование. Это обеспечивает быструю разработку и тестирование функций, но требует больше ресурсов CPU хоста и не всегда позволяет работать со скоростью линии. Кроме того, присущий программам на основе прерываний элемент случайности не позволяет создавать сетевые приложения, требующие точного управления передачей [3]. Тем не менее, во многих исследовательских проектах [4]-[7] были программно реализованы новые функции NIC путём изменения сетевого стека или применения схем обхода ядра, таких как DPDK[1] [8].

Сетевые адаптеры на основе ПЛИС (FPGA) сочетают свойства NIC на основе ASIC и программных NIC - они могут работать со скоростью линии, обеспечивают малые задержки и точную синхронизацию, сохраняя достаточно короткий цикл разработки новых функций. Разработано достаточно фирменных NIC на основе ПЛИС. Например, компания Alibaba создала полностью настраиваемый сетевой адаптер ПЛИС на основе RDMA, который применяется для аппаратной реализации протокола прецизионного контроля перегрузок (High precision congestion control protocol или HPCC) [9]. Имеется и коммерческая продукция, например от Exablaze[2] [9] и Magmio [10]. К сожалению, коммерческие сетевые адаптеры на основе ПЛИС (как и ASIC NIC) часто используют закрытые фирменные функции, которые невозможно изменить. Это ограничивает гибкость и применимость таких NIC при разработке новых сетевых приложений. Доступные коммерчески высокопроизводительные компоненты DMA, такие как ядро Xilinx XDMA, ядра QDMA, ускорительное ядро Atomic Rules Arkville DPDK [12] не предоставляют полностью настраиваемого оборудования для контроля потока передаваемых данных. Ядро Xilinx XDMA предназначено для выгрузки приложений, поэтому обеспечивает сильно ограниченную функциональность очередей и не предоставляет простого метода управления планированием передачи. Ядро Xilinx QDMA и ускорительное ядро Atomic Rules Arkville DPDK ориентированы на сетевые приложения, поддерживая небольшое число очередей и предоставляя драйверы DPDK. Однако поддерживаемое число очередей (2K в XDMA и до 128 в Arkville) невелико и нет простого метода точного управления передачей пакетов.

Имеется проект с открытым кодом NetFPGA [13], однако он предоставляет лишь инструменты для базовой обработки пакетов на основе FPGA и не предназначен для разработки NIC. Кроме того, эталонная модель NetFPGA NIC использует фирменное ядро Xilinx XDMA, не предназначенное для сетевых приложений. Замена ядра Xilinx XDMA в эталонном NIC для NetFPGA на Corundum расширяет возможности и повышает гибкость создания прототипов.

Решения по обработке пакетов на основе FPGA включают Catapult [14] с выгрузкой приложений и FlowBlaze [15] с реализацией механизмов «сопоставление-действие» (match-action) на основе ПЛИС. Однако эти платформы оставляют реализацию стандартных функций NIC за ASIC и работают в режиме «насадки на провод» (bump-in-the-wire), не обеспечивая явного управления планировщиком и очередями в NIC.

[1]Data Plane Development Kit - комплект для разработки плоскости данных.
[2]Приобретена компанией Cisco Systems.

Проект Corundum является полностью открытым и может работать со стандартным сетевым стеком хоста на реальных скоростях линии. Поддерживаются тысячи очередей передачи, связанных с расширяемыми планировщиками передачи для тонкого и точного управления потоками. Это обеспечивает мощную и гибкую платформу с открытым кодом для разработки сетевых приложений, сочетающих аппаратные и программные функции.

Реализация Corundum

Corundum имеет ряд уникальных архитектурных свойств. Во-первых, состояния аппаратных очередей сохраняются в оперативной памяти (RAM) FPGA, что позволяет поддерживать тысячи очередей с индивидуальным управлением. Эти очереди связаны с интерфейсами, а каждый интерфейс может иметь несколько портов, в которых могут применяться свои независимые планировщики передачи. Такой подход обеспечивает очень тонкое управления передачей пакетов. Модуль планировщика можно изменить или заменить полностью для реализации иных схем планирования, включая экспериментальные. В сочетании с синхронизацией часов по протоколу PTP это обеспечивает возможность планирования по времени, включая высокоточный доступ TDMA[1].

В Corundum используется модульное решение с параметризацией. Многие конфигурационные и структурные параметры, включая число интерфейсов и портов, число очередей, размер памяти, тип планирования и т. п., можно задать во время синтезирования через параметры Verilog. Эти параметры раскрываются в регистрах конфигурации, которые драйвер считывает для определения конфигурации NIC, что позволяет использовать один драйвер для разных плат и конфигураций, не изменяя его.

Решение поддерживает компоненты DMA PCIe для интерфейса ядра Xilinx Ultrascale PCIe hard IP. Поддержка интерфейса PCIe TLP, обычно применяемая в FPGA, не реализована и оставлена на будущее. Это позволит расширить набор подходящих ПЛИС.

Corundum занимает достаточно малую площадь, оставляя достаточно места для дополнительной логики даже в сравнительно мелких FPGA. Например, Corundum для ExaNIC X10 [10] (2 порта 10G с интерфейсом PCIe gen 3 x8 и 512-битовым внутренним путём данных) занимает менее четверти ресурсов, доступных на одном из самых мелких Kintex Ultrascale FPGA (KU035).

Высокоуровневый обзор

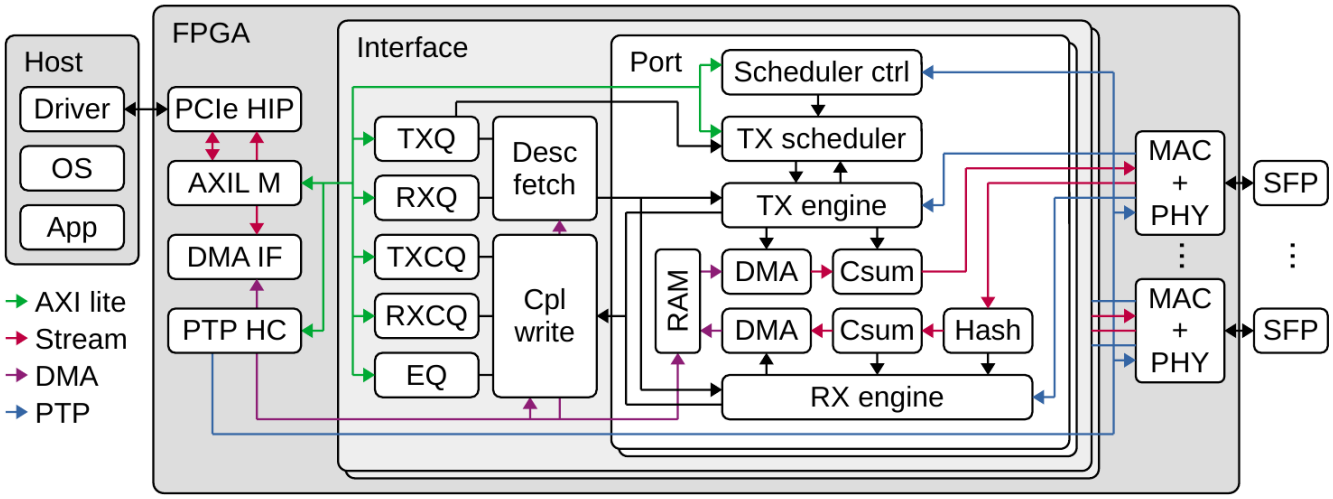


Рисунок 1. Блок-схема Corundum NIC.

PCIe HIP - ядро PCIe hard IP, AXIL M - AXI lite master, DMA IF - интерфейс DMA, PTP HC - аппаратные часы PTP, TXQ - менеджер очередей передачи, TXCQ - менеджер очередей завершения передачи, PTP, RXQ - менеджер очередей приёма, RXCQ - менеджер очередей завершения приёма, EQ - менеджер событий в очередях, MAC + PHY - контроллер доступа к среде Ethernet (MAC) и уровень физического интерфейса (PHY).

Блок-схема Corundum NIC показана на рисунке 1. На верхнем уровне NIC состоит из 3 основных вложенных модулей. Модуль верхнего уровня содержит компоненты поддержки и интерфейсов, включая ядро PCI express hard IP и интерфейс прямого доступа к памяти (DMA), аппаратные часы PTP, компоненты интерфейса Ethernet с MAC, PHY и сериализаторами[2]. Модуль верхнего уровня включает также 1 или несколько экземпляров интерфейсных модулей, каждый из которых соответствует сетевому интерфейсу операционной системы (например, eth0). Каждый интерфейсный модуль включает логику управления очередью, а также логику дескрипторов и завершения обработки. Логика обработки очереди поддерживает состояния для всех очередей NIC - передача, завершение передачи, приём, завершение приёма, события в очереди. В каждом интерфейсном модуле содержится 1 или несколько экземпляров модулей порта. Каждый модуль порта предоставляет потоковый интерфейс AXI в MAC и содержит планировщик передачи, механизмы передачи и приёма, пути данных для передачи и приёма, а также память (RAM) для временного сохранения входящих и исходящих пакетов при операциях DMA.

Для каждого порта планировщик передачи в модуле порта выбирает очереди, назначенные для отправки. Планировщик передачи генерирует команды для механизма передачи, который координирует операции на пути передачи. Модуль планировщика является гибким функциональным блоком, который можно изменить или заменить произвольным планировщиком (например, по событиям). По умолчанию планировщик просто использует круговой обход (round robin). Для всех портов, связанных с одним интерфейсным модулем, используется общий набор очередей передачи, который представляется как один унифицированный интерфейс с операционной системой. Это позволяет перемещать потоки из порта в порт или распределять нагрузку между несколькими портами просто сменой установок планировщика передачи без влияния на остальной сетевой стек. Динамическое сопоставление очередей с портами, задаваемое планировщиком, является уникальным свойством Corundum, позволяющим исследовать новые протоколы

[1]Time Division Multiple Access - множественный доступ с разделением по времени.
[2]Узел преобразования параллельных данных (шина) в последовательные (поток).

и архитектуру сети, включая параллельные сети, такие как P-FatTree [16], и сети с оптической коммутацией, такие как RotorNet [17] и Opera [18].

В направлении приёма входящие пакеты проходят через модуль хэширования потока (Hash) для определения целевой приёмной очереди и генерации команд для машины приёма (RX engine), координирующей операции на приёмном пути данных. Поскольку все порты одного интерфейсного модуля используют общий набор приёмных очередей, входящие потоки от разных портов сливаются в общий набор очередей. В NIC можно добавить настраиваемые модули для предварительной обработки и фильтрации входящих пакетов до их передачи на шину PCIe.

Компоненты NIC соединены между собой через несколько разных интерфейсов, включая AXI lite, AXI stream и сегментируемый интерфейс с памятью для операций DMA, который будет описан ниже. AXI lite служит для пути управления от драйвера к NIC. Он применяется для инициализации и настройки компонентов NIC, а также для управления указателями очередей при операциях приёма и передачи. Интерфейсы AXI stream служат для передачи внутри NIC пакетизированных данных, включая пакеты уровня передачи PCIe (PCIe transmission layer packet или TLP) и кадры Ethernet. Сегментированный интерфейс с памятью служит для соединения интерфейса PCIe DMA с трактом данных NIC, а также логикой дескрипторов и завершения обработки.

Большая часть логики NIC работает в пользовательском домене синхронизации PCIe с номинальной частотой 250 МГц для всех современных вариантов исполнения. Для взаимодействия с MAC применяются асинхронные буферы FIFO, работающие в соответствующих доменах синхронизации приёма и передачи с частотой 156,25 МГц для 10G, 390,625 МГц для 25G и 322,266 МГц для 100G.

Управление конвейерными очередями

Обмен пакетами данных между Corundum NIC и драйвером осуществляется через очереди дескрипторов и завершения. Очереди дескрипторов формируют коммуникационный канал от хоста к NIC для передачи сведений о местах хранения отдельных пакетов в памяти системы. Очереди завершения формируют коммуникационный канал от NIC к хосту для передачи сведений о завершённых операциях и связанных с ними метаданных. Очереди дескрипторов и завершения реализованы в форме кольцевых буферов в доступной через DMA памяти системы, а оборудование NIC поддерживает требуемые сведения о состоянии очередей. Данные состояния включают адрес DMA для кольцевого буфера, указатели на производителя и потребителя, с также ссылку на соответствующую очередь завершения. Состояние дескриптора для каждой очереди помещается в 128 битов.

Логика управления очередями для Corundum NIC должна эффективно сохранять и поддерживать состояния тысяч очередей. Ото требует хранить состояние очереди в блочном ОЗУ (block RAM или BRAM[1]) или ultra RAM (URAM[2]) на FPGA. Поскольку для состояния требуется 128 битов ОЗУ, а блоки URAM имеют размер 72x4096, для хранения 4096 очередей нужно 2 экземпляра URAM. Использование экземпляров URAM позволяет расширять логику управления очередями до 32768 очередей на интерфейс.

Для поддержки высокой пропускной способности от NIC требуется параллельная обработка дескрипторов. Поэтому логика управления очередями должна отслеживать множество выполняющихся операций, передавая обновлённые указатели очередей драйверу при завершении операций. Требуемое для отслеживания выполняемых операций состояние значительно меньше состояния дескриптора и поэтому может сохраняться в триггерах (flip-flop) и распределенном ОЗУ.

В NIC используется 2 модуля диспетчера очередей - queue_manager для управления очередями дескрипторов host-to-NIC и cpl_queue_manager для управления очередями завершения NIC-to-host. Модули незначительно различаются в части обработки указателей и генерации событий. Из-за сходства модулей здесь описан лишь модуль queue_manager.

Массив BRAM или URAM, применяемый для хранения данных о состоянии очередей, требует задержки в несколько тактов для каждой операции чтения, поэтому в queue_manager используется конвейерная архитектура для облегчения одновременного выполнения нескольких операций. Конвейер поддерживает 4 типа операций - чтение и запись регистра, запрос на постановку или извлечение из очереди. Операции доступа к регистрам через интерфейс AXI позволяют драйверу инициализировать состояние очереди и предоставлять указатели на выделенную память хоста, а также обращаться к указателям производителя и потребителя при обычных операциях.

Планировщик передачи

По умолчанию в Corundum NIC применяется простой планировщик передачи с круговым обходом, реализованный в модуле tx_scheduler_rr. Планировщик передаёт команды механизму передачи для запуска операций передачи из очередей NIC. Планировщик содержит базовые состояния для всех очередей, буфер FIFO для хранения текущих активных очередей и выполнения кругового обхода, а также таблицу операций для отслеживания выполняющихся передач.

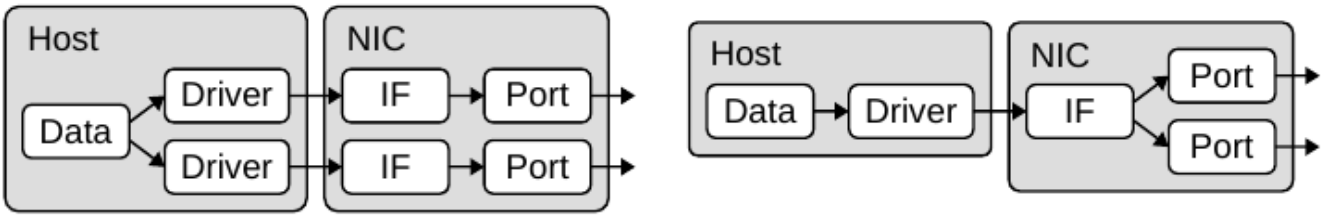
Подобно логике управления очередями планировщик хранить сведения о состоянии очередей в BRAM или URAM на FPGA, что позволяет расширять планировщик для поддержки большого числа очередей. Планировщик передачи тоже использует конвейерную обработку для сокрытия задержки доступа к памяти. Модуль планировщика передачи включает 4 основных интерфейса - интерфейс регистра AXI lite и 3 потоковых интерфейса. Интерфейс AXI lite позволяет драйверу менять параметры планировщика, а также включать и отключать очереди. Первый потоковый интерфейс обеспечивает получение событий (doorbell) от логики управления очередями, когда драйвер помещает пакеты для передачи в очередь. Второй потоковый интерфейс передаёт команды отправки, генерируемые планировщиком для машины передачи. Каждая команда включает индекс очереди, из которой нужно передавать, а также тег для отслеживания выполняющихся операций. Третий потоковый интерфейс возвращает планировщику сведения о статусе операции передачи, сообщая размер переданного пакета или причину отказа (пустая или отключённая очередь).

Планировщик передачи можно расширить или заменить для реализации любых алгоритмов планирования. Это позволяет использовать Corundum как платформу для оценки экспериментальных алгоритмов планирования, таких как SENIC [4], Carousel [5], PIEO [19], Loom [7]. Можно также предоставить дополнительные входя в модуль планировщика

[1]См. https://nandland.com/lesson-15-what-is-a-block-ram-bram/
[2]См. https://en.wikipedia.org/wiki/UltraRAM

передачи, включая обратную связи от пути приёма, которые позволяют реализовать новые протоколы и методы контроля перегрузок, такие как NDP [6] и HPCC [9]. Соединение планировщика с аппаратными часами PTP позволяет поддерживать TDMA для реализации RotorNet [17], Opera [18] и других моделей коммутации каналов.

Порты и интерфейсы



(a) Традиционный сетевой адаптер с программным назначением портов. *(b) Corundum NIC с аппаратным назначением портов.*

Рисунок 2. Сравнение архитектуры портов и интерфейсов.

Уникальной архитектурной особенностью Corundum является разделение портов и сетевых интерфейсов, позволяющее связать с одним интерфейсом несколько портов. Большинство современных NIC поддерживают 1 порт на интерфейс, как показано на рисунке 2(a). Когда сетевой стек помещает пакет в очередь передачи на сетевом интерфейсе, пакет попадает в сеть через связанный с этим интерфейсом порт. Однако в Corundum с интерфейсом может быть связано несколько портов, поэтому выбор порта для передачи пакета в сеть может определяться оборудованием при извлечении из очереди, как показано на рисунке 2(b).

Все порты, связанные с одним модулем сетевого интерфейса, используют общий набор очередей передачи и представляются операционной системе как один унифицированный интерфейс. Это позволяет переносить потоки из порта в порт, изменяя лишь настройки планировщика передачи без воздействия на остальные части сетевого стека. Динамическое сопоставление выходных очередей с портами, управляемое планировщиком, позволяет исследовать новые протоколы и архитектуру сетей, включая параллельные сети, такие как P-FatTree [16], и сети с оптической коммутацией, такие как RotorNet [17] и Opera [18].

Путь данных и механизмы приёма и передачи

В пути данных Corundum применяются отображаемые на память и потоковые интерфейсы. Поток AXI служит для переноса пакетов Ethernet между модулями DMA порта, Ethernet MAC, а также модулями контрольной суммы (Csum) и расчёта хэш-значения (Hash). Поток AXI служит также для соединения ядра PCIe hard IP с ведущим модулем PCIe AXI lite и модулем интерфейса PCIe DMA. Для подключения интерфейсного модуля PCIe DMA, модулей DMA в портах, а также логики очередей дескрипторов и завершения с внутренней памятью (RAM) используется сегментированный интерфейс с памятью.

Разрядность потоковых интерфейсов AXI определяется требуемой пропускной способностью. Логика ядра пути данных (кроме Ethernet MAC) работает в пользовательском домене синхронизации PCIe с частотой 250 МГц. Поэтому потоковые интерфейсы AXI с ядром PCIe hard IP должны соответствовать разрядности интерфейса аппаратного ядра - 256 битов для PCIe gen 3 x8 и 512 битов для PCIe gen 3 x16. На стороне Ethernet разрядность интерфейса соответствует разрядности интерфейса MAC, если частота 250 МГц не слишком мала для обеспечения нужной пропускной способности. Для Ethernet 10G разрядность интерфейса MAC составляет 64 бита при частоте 156,25 МГц и он может быть соединён с доменом синхронизации 250 МГц той же разрядности. Для Ethernet 25G интерфейс MAC имеет разрядность 64 бита при частоте 390,625 МГц, что требует его преобразования в разрядность 128 битов для обеспечения нужной пропускной способности при частоте 250 МГц. Для Ethernet 100G в Corundum применяются аппаратные ядра CMAC Xilinx 100G на Ultrascale Plus FPGA. Интерфейс MAC имеет разрядность 512 битов при частоте 322,266 МГц и подключён к домену синхронизации 250 МГц с разрядностью 512 битов, поскольку для обеспечения пропускной способности 100 Гбит/с нужна частота около 195 МГц.

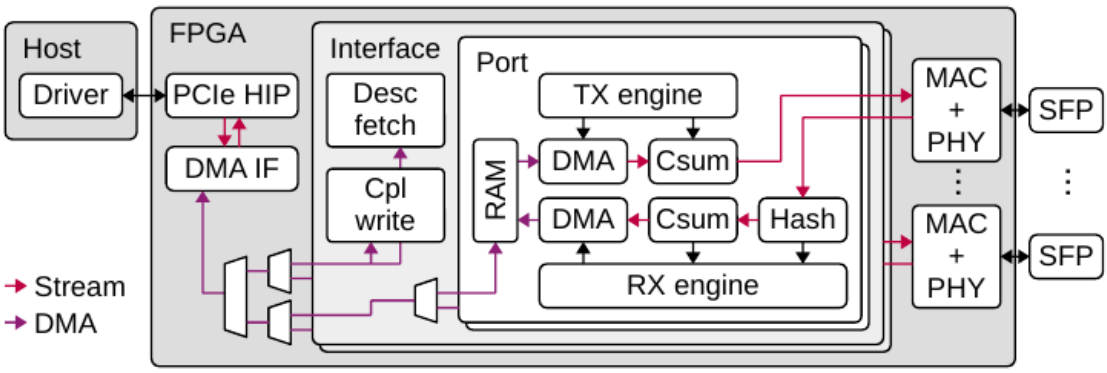


Рисунок 3. Упрощённый вариант рисунка 1 с путями данных.

Блок-схема пути данных NIC показана на рисунке 3. Ядро PCIe hard IP (PCIe HIP) соединяет NIC с хостом. Два потоковых интерфейса соединяют интерфейсный модуль PCIe DMA с ядрами PCIe hard IP. Один интерфейс служит для запросов чтения и записи, другой - для чтения данных. Модуль интерфейса PCIe DMA соединяется с модулем выборки дескрипторов, модулем завершения записи, модулями временной памяти (RAM), а также механизмами RX и TX через набор набор мультиплексоров интерфейса DMA. В направлении интерфейса DMA мультиплексоры комбинируют команды переноса DMA от нескольких источников. В обратном направлении они маршрутизируют передачу откликов о состоянии. Мультиплексоры также управляют сегментированными интерфейсами с памятью при чтении и записи. Мультиплексор верхнего уровня объединяет трафик дескрипторов с трафиком пакетов данных, предоставляя дескрипторам более высокий приоритет. Затем пара мультиплексоров объединяет трафик от нескольких интерфейсных модулей. Дополнительный мультиплексор внутри каждого интерфейсного модуля объединяет трафик пакетов данных от нескольких экземпляров портов.

Машины приёма и передачи отвечают за координацию операций при передаче и приёме пакетов. Эти машины могут обслуживать множество обрабатываемых пакетов для повышения пропускной способности. Как показано на рисунке 1, машины приёма и передачи соединены с несколькими модулями тракта приёма и передачи данных, включая модули DMA, модули выгрузки контрольных сумм и хэширования, логику обработки дескрипторов и завершения, а также а также интерфейсы временных меток Ethernet MAC.

Машина передачи отвечает за координацию операций отправки пакетов. Она обрабатывает запросы передачи от планировщика передачи для конкретных очередей. После низкоуровневой обработки с использованием механизма PCIe DMA пакет проходит через модуль контрольной суммы, MAC и PHY. После отправки пакета машина передачи получает метку PTP от MAC, создаёт запись о завершении и передаёт её модулю завершения записи.

Приёмная машина отвечает за координацию операций получения пакетов. Входящие пакеты проходят через PHY и MAC. После низкоуровневой обработки, включающей хэширование и установку метки времени, машина приёма будет выдавать 1 или несколько запросов на запись машине PCIe DMA для записи данных пакета в память хоста. По завершении записи машина приёма создаёт запись о завершении и передаёт её модулю завершения.

Модули чтения и завершения записи работают аналогично машинам приёма и передачи. Эти модули обрабатывают запросы дескрипторов и завершения чтения/записи от машин приёма и передачи, выдают запросы на размещение в очереди и извлечение из неё диспетчерам очередей для получения адреса элемента очереди в памяти хоста, а также выдают запросы к интерфейсу PCIe DMA для доставки данных. Модуль завершения записи отвечает также за обработку событий в очередях завершения передачи и приёма, путём включения их в подходящую очередь событий и внесения записей о событиях.

Сегментированный интерфейс с памятью

Для обеспечения высокой производительности DMA через PCIe в Corundum применяется сегментированный интерфейс с памятью. Интерфейс расщеплён на сегменты с максимальным размером 128 битов и разрядностью вдвое больше разрядности потокового интерфейса AXI для ядра PCIe hard IP. Например, для PCIe gen 3 x16 с 512-битовым потоковым интерфейсом AXI из аппаратного ядра PCIe будет применяться 1024-битовый сегментированный интерфейс с 8 сегментами по 128 битов. Такой интерфейс обеспечивает лучшее соответствие импеданса при использовании одного интерфейса AXI, что позволяет более полно использовать канал PCIe за счёт устранения противодействия из-за выравнивания с машине DMA и арбитража в логике соединений. В частности, интерфейс гарантирует, что интерфейс DMA может выполнять полноразрядное чтение или запись без выравнивания в каждом такте. Кроме того, использование простого двухпортового ОЗУ, выделенного для трафика одного направления, устраняет противоречия между путями чтения и записи.

Каждый сегмент работает подобно интерфейсу AXI lite, но использует 3 интерфейса вместо 5. Один канал предоставляет адрес и данные для записи, другой - адрес для чтения, третий - считываемые данные. В отличие от AXI, всплески (burst) и переупорядочение не поддерживаются, что упрощает логику интерфейса. Соединительные компоненты (мультиплексоры) отвечают за сохранение порядка операций даже при доступе к нескольким блокам RAM. Сегменты действуют независимо друг от друга с отдельными соединениями для управления потоками данных и экземплярами логики упорядочения соединений. Операции маршрутизируются по сигналам выбора, а не по декодированию адресов. Это избавляет от необходимости назначать адреса и позволяет использовать параметризуемые компоненты соединений, маршрутизирующие операции с минимальной настройкой конфигурации.

Байтовые адреса отображаются на адреса сегментированного интерфейса и младшие биты адреса задают байтовую «дорожку» (lane) в сегменте, следующие - сегмент, а старшие - адрес слова для этого сегмента. Например, в 1024-битовом сегментированном интерфейсе с 8 сегментами по 128 битов 4 младших бита адреса задают «дорожку», следующие 3 - сегмент, а оставшиеся адресную шину для сегмента.

Драйвер устройства

Corundum NIC соединяется с сетевым стеком ядра Linux через модуль ядра. Этот модуль отвечает за инициализацию NIC, регистрацию интерфейсов ядра, выделение доступных через DMA буферов для очередей дескрипторов и завершения, обработку прерываний устройства и передачу сетевого трафика между ядром и NIC.

NIC использует пространство регистров для раскрытия таких параметров, как число интерфейсов, портов, очередей и планировщиков, размер максимального передаваемого блока (maximum transport unit или MTU), поддержка меток PTP и выгрузки. Драйвер считывает регистры при инициализации, что даёт ему возможность настроить себя и зарегистрировать интерфейсы ядра в соответствии с конфигурацией NIC. Эта возможность автоматического определения означает слабую связь между драйвером и NIC, драйверу обычно не нужно менять тракт данных при работе с разными платами FPGA, вариантами устройства Corundum и значениями параметров.

Литература

[1] Alex Forencich, Alex C. Snoeren, George Porter, George Papen. Corundum: An Open-Source 100-Gbps NIC. 2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). https://www.fccm.org/past/2020/proceedings/2020/pdfs/FCCM2020-65FOvhMqzyMYm99lfeVKyl/580300a038/580300a038.pdf, https://github.com/corundum/corundum

[2] D. Firestone, A. Putnam, S. Mundkur, D. Chiou, A. Dabagh, M. Andrewartha, H. Angepat, V. Bhanu, A. Caulfield, E. Chung, H. K. Chandrappa, S. Chaturmohta, M. Humphrey, J. Lavier, N. Lam, F. Liu, K. Ovtcharov, J. Padhye, G. Popuri, S. Raindel, T. Sapre, M. Shaw, G. Silva, M. Sivakumar, N. Srivastava, A. Verma, Q. Zuhair, D. Bansal, D. Burger, K. Vaid, D. A. Maltz, and A. Greenberg, “Azure accelerated networking: SmartNICs in the public cloud,” in 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18). Renton, WA: USENIX Association, Apr. 2018, pp. 51-66. https://www.usenix.org/conference/nsdi18/presentation/firestone

[3] B. Stephens, A. Akella, and M. M. Swift, “Your programmable NIC should be a programmable switch,” in Proceedings of the 17th ACM Workshop on Hot Topics in Networks, ser. HotNets ’18. New York, NY, USA: Association for Computing Machinery, 2018, p. 36-42. https://www2.cs.uic.edu/~brents/docs/panic.hotnets18.pdf

[4] S. Radhakrishnan, Y. Geng, V. Jeyakumar, A. Kabbani, G. Porter, and A. Vahdat, “SENIC: Scalable NIC for end-host rate limiting,” in 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14). Seattle, WA: USENIX Association, 2014, pp. 475-488. https://www.usenix.org/system/files/conference/nsdi14/nsdi14-paper-radhakrishnan.pdf

[5] A. Saeed, N. Dukkipati, V. Valancius, V. The Lam, C. Contavalli, and A. Vahdat, “Carousel: Scalable traffic shaping at end hosts,” in Proceedings of the Conference of the ACM Special Interest Group on Data Communication, ser. SIGCOMM ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 404-417. https://www.comm.utoronto.ca/~jorg/teaching/ece1545/papers/Carousel-Sigcomm2017-paper.pdf

[6] M. Handley, C. Raiciu, A. Agache, A. Voinescu, A. W. Moore, G. Antichi, and M. Wójcik, “Re-architecting datacenter networks and stacks for low latency and high performance,” in Proceedings of the Conference of the ACM Special Interest Group on Data Communication, ser. SIGCOMM ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 29-42. [Online]. https://discovery.ucl.ac.uk/id/eprint/10068163/1/ndp.pdf

[7] B. Stephens, A. Akella, and M. Swift, “Loom: Flexible and efficient NIC packet scheduling,” in 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19). Boston, MA: USENIX Association, Feb. 2019, pp. 33-46. https://www.usenix.org/system/files/nsdi19-stephens.pdf

[8] Data plane development kit, https://www.dpdk.org/

[9] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh, and et al., “HPCC: High precision congestion control,” in Proceedings of the ACM Special Interest Group on Data Communication, ser. SIGCOMM ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 44-58. https://liyuliang001.github.io/publications/hpcc.pdf

[10] Cisco Nexus SmartNIC https://www.cisco.com/c/en/us/products/interfaces-modules/nexus-smartnic/index.html

[11] Magmio https://www.magmio.com/product

[12] Atomic rules, http://www.atomicrules.com/

[13] N. Zilberman, Y. Audzevich, G. A. Covington, and A. W. Moore, “NetFPGA SUME: Toward 100 Gbps as research commodity,” IEEE Micro, vol. 34, no. 5, pp. 32-41, Sep. 2014. https://github.com/NetFPGA/netfpga.git, https://www.cl.cam.ac.uk/~nz247/publications/zilberman2014sume.pdf

[14] A. M. Caulfield, E. S. Chung, A. Putnam, H. Angepat, J. Fowers, M. Haselman, S. Heil, M. Humphrey, P. Kaur, J.-Y. Kim, and et al., “A cloud-scale acceleration architecture,” in The 49th Annual IEEE/ACM International Symposium on Microarchitecture, ser. MICRO-49. IEEE Press, 2016. https://www.cs.utexas.edu/ftp/dburger/papers/MICRO16.pdf

[15] S. Pontarelli, R. Bifulco, M. Bonola, C. Cascone, M. Spaziani, V. Bruschi, D. Sanvito, G. Siracusano, A. Capone, M. Honda, F. Huici, and G. Siracusano, “FlowBlaze: Stateful packet processing in hardware,” in 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19). Boston, MA: USENIX Association, Feb. 2019, pp. 531-548. https://www.usenix.org/conference/nsdi19/presentation/pontarelli

[16] W. M. Mellette, A. C. Snoeren, and G. Porter, “P-FatTree: A multi-channel datacenter network topology,” in Proceedings of the 15th ACM Workshop on Hot Topics in Networks, ser. HotNets ’16. New York, NY, USA: Association for Computing Machinery, 2016, p. 78-84. https://cseweb.ucsd.edu//~snoeren/papers/ptree-hotnets16.pdf

[17] W. M. Mellette, R. McGuinness, A. Roy, A. Forencich, G. Papen, A. C. Snoeren, and G. Porter, “RotorNet: A scalable, low-complexity, optical datacenter network,” in Proceedings of the Conference of the ACM Special Interest Group on Data Communication, ser. SIGCOMM ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 267-280. https://cseweb.ucsd.edu/~gmporter/papers/sigcomm17-rotornet.pdf

[18] W. M. Mellette, R. Das, Y. Guo, R. McGuinness, A. C. Snoeren, and G. Porter, “Expanding across time to deliver bandwidth efficiency and low latency,” in 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20). Santa Clara, CA: USENIX Association, Feb. 2020. https://www.usenix.org/system/files/nsdi20-paper-mellette.pdf

[19] V. Shrivastav, “Fast, scalable, and programmable packet scheduler in hardware,” in Proceedings of the ACM Special Interest Group on Data Communication, ser. SIGCOMM ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 367-379. https://web.ics.purdue.edu/~vshriva/papers/pieo_2019.pdf

Николай Малых

nmalykh@protokols.ru