RapidIO – коммутационная структура последовательного типа*

Н.Слепов
nslepov@online.ru

Логический уровень. Спецификация управления потоком

Эта часть логической спецификации [2и] описывает управление потоком при параллельной и последовательной передаче данных в отсутствие и при наличии перегрузок.

Перегрузки

В системах на основе коммутационных структур могут возникать несколько типов перегрузок, классифицируемых по длительности события: сверхкороткие, короткие, средние и длинные. Перегрузки могут возникнуть внутри и между коммутаторами, а также между коммутаторами и конечными точками. Концептуально перегрузка возникает в выходном порте, который старается передать все данные соединенному с ним устройству, но получает информации больше, чем способен передать. Избыточные данные могут скапливаться до тех пор, пока их объем не превысит емкость буфера выходного порта коммутатора. После этого перегрузка распространяется на другие устройства, подключенные к соответствующему входу коммутатора (рис.15). Поэтому борьба за конкретное соединение в системе может влиять на способность системы передавать данные, не связанные с оспариваемым соединением. Это крайне нежелательный режим работы системы для многих приложений. Длительность периода действия перегрузки определяет размер воздействия перегрузки на систему.

Сверхкороткие перегрузки характеризуются очень малой длительностью – до 500 нс. В системе RapidIO эти события адекватно отрабатываются благодаря буферизации внутри устройств на любой стороне звена связи и наличию механизма повтора передачи, описанного в Спецификациях физического уровня [2г, 2е]. Эта комбинация добавляет эластичности каждому звену описанной системы. Влияние на систему сверхкоротких перегрузок мало или незаметно.

Короткие перегрузки длятся от десятков до сотен микросекунд. Они крайне вредно влияют на производительность системы в целом. Управление этим типом перегрузок требует определенных средств, фиксирующих их появление и использующих некий механизм снижения количества данных, вводимых конечными точками в зону перегрузки системы. Если это можно сделать вовремя, то зона перегрузки останется локализованной (до тех пор, пока вызванный этой перегрузкой затор не расчистится), т.е. не нанесет ущерба другим частям системы.

Средние перегрузки представляют собой частые серии коротких перегрузок, длящиеся секунды или минуты. Этот тип событий указывает на несбалансированность нагрузки, передаваемой через коммутационную структуру. Устранение этого типа перегрузки требует программного механизма балансировки нагрузки для реконфигурации структуры или системы в целом.

Длинные перегрузки представляют собой ситуацию, когда система не имеет ресурсной емкости для обслуживания всех запросов. Эта ситуация корректируется модернизацией или заменой всей системы.

В разработанной спецификации [2и] рассмотрена проблема управления короткими перегрузками.

Пример возникновения перегрузки показан на рис. 15. Корни проблемы – в большом количестве независимых потоков транзакций запросов, каждый из которых имеет взрывной и пространственно локальный характер, но не превышает возможной емкости звена передачи или конечных точек. Благодаря случайной комбинации таких потоков требования к полосе пропускания конкретного звена (обычно в середине многокаскадной топологии, в нашем случае в выходном порте коммутатора 3) превышают в течение какого-то периода времени емкость звена, учитывая концентрацию в нем потоков ***а***, ***в*** и ***с***. В результате буфер этого выходного порта будет переполнен транзакциями, что приведет к активации механизма управления потоком на звеньях предыдущих коммутаторных каскадов. Выходные буферы коммутаторов 1 и 2 также заполнятся пакетами. Пакеты потока запроса транзакций (например, поток d) также не достигнут выходного порта

* Продолжение, начало см.: “ЭЛЕКТРОНИКА: НТБ”, 2006, №5, с.76.

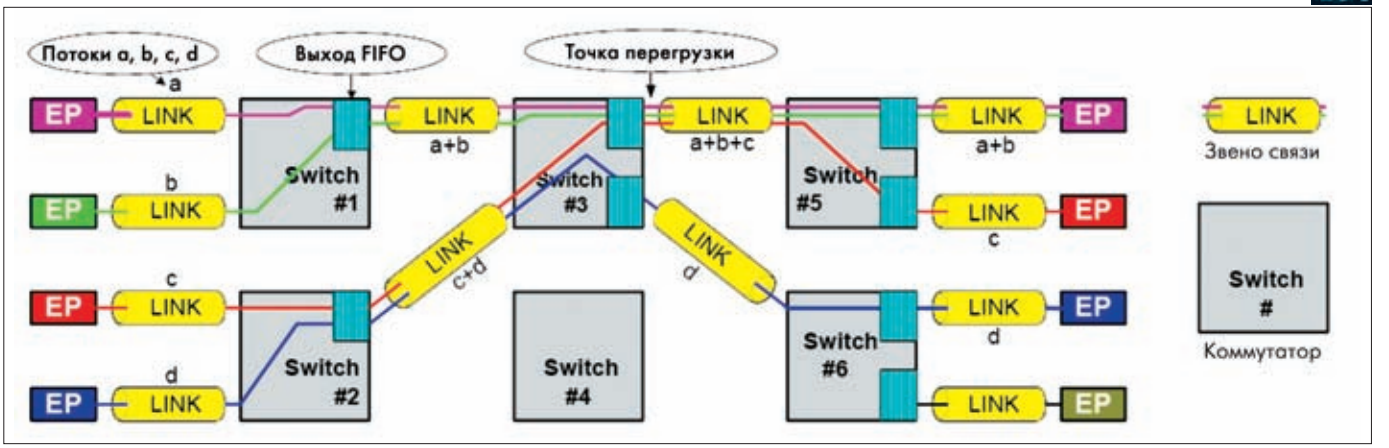

Рис. 15. Пример возникновения перегрузки

(из-за заполненного буфера потоков в коммутатор 3) и окажутся в состоянии ожидания, приводя к дополнительным потерям производительности системы.

Аналогичное явление происходит, когда конечная точка не может обработать входной поток и применяет управление потоком для остановки входного потока пакетов.

Операции по управлению потоком

Схемы управления применяются для снижения влияния короткой перегрузки на передачу пакетов. Эти схемы используют явные сообщения управления потоком типа "выключить передачу" (XOFF) и "включить передачу" (XON), переносимые с помощью *пакетов управления перегрузкой* (CCP), которые требуют квитирования. Пакет XOFF CCP используется для остановки выбранных потоков перед конечными точками их источников, а пакеты XON CCP – для возобновления этих потоков с тех же точек, как только затор рассосался. Методы обнаружения перегрузки зависят от реализации и от внутренней структуры буферизации пакетов (учитывается емкость коммутатора). Так, для коммутатора с буферизованным выходным портом (см. рис. 15) перегрузка возникает тогда, когда будут превышены некоторые уровни "паводка" (Watermark), но это не единственный способ обнаружения перегрузки. Некоторые из этих методов описаны в [2и, Appendix A].

Собственно операция управления потоком реализуется единственной транзакцией FLOW CONTROL (управление потоком). Эта транзакция генерируется коммутатором или конечной точкой для управления потоком данных. Этот механизм обратно совместим с устройствами, унаследованными от RapidIO в той же системе.

Требования к физическому уровню. Для поддержки эффективного контроля к физическому уровню системы предъявляются определенные требования. Так, топология системы должна быть такой, чтобы транзакция управления потоком могла быть передана назад к любому источнику, пославшему запрос на управление потоком. Этот обратный путь может или повторять путь транзакции запроса к точке перегрузки, или использовать другой путь в зависимости от требований к системе.

Передача транзакций управления потоком. Транзакции управления потоком рассматриваются как потоки независимого и наиболее важного трафика в системе. Они всегда передаются при первой возможности и за счет всех других потоков трафика. Согласно спецификации физического уровня для этого требуется помечать пакеты, добавляя поле приоритета "prio"=11 и поле "crf"=1 (если оно поддерживается). Однако при прохождении потока CCP назад к конечной точке источника XOFF и XON CCP могут быть сброшены (потеряны) элементами, обслуживающими прямой поток, в случае недостаточной емкости буфера. Эти транзакции используют

нормальный формат пакета, применяемого для контроля ошибок.

Механизм отложенного XOFF. Из-за возможности потерять пакет XON в коммутационной структуре должен быть предусмотрен механизм отложенного XOFF для рестарта отложенных потоков, которые были выключены (благодаря прохождению XOFF), но не были включены (ввиду потери XON) в конечных точках. Детали этого механизма зависят от реализации, однако конечные точки должны иметь достаточно средств, чтобы обрабатывать, а не игнорировать отложенные потоки. Этот механизм должен нормально работать со стандартными XON/XOFF CCP при обработке коротких перегрузок.

Список управляемых потоков. Элементы, посылающие XOFF, должны хранить список остановленных потоков, например хотя бы для того, чтобы отобрать потоки, требующие рестарта. Следовательно, должен формироваться "Список управляемых потоков", включающий (кроме прочей информации, зависящей от реализации) также ID потоков.

Счетчики XOFF/XON. Для ряда потоков, терминируемых конечными точками, должны быть использованы счетчики числа отключений/включений потоков. Так как число потоков может быть непредсказуемо велико, то счетчики могут обслуживать агрегированные группы потоков, хотя для простоты лучше использовать один счетчик на поток. Работа счетчика (он должен быть реверсивным) может быть основана на принципе инкремента (XOFF) – декремента (XON), тогда поток считается активным, если ассоциированный с ним счетчик содержит "0". Если активируется механизм отложенного XOFF, то счетчик сбрасывается и поток стартует вновь.

0111	tgtdestinationID	XON/XOFF	rsrv	flowID	SOC
4	8(tt=0b00), 16(tt=0b01)	1	7	7	1

Рис. 16. Формат пакета CCP (type 7) логического уровня

Формат пакета управления потоком

Формат пакета FLOW CONTROL (пакет type 7 для управления потоком) используется коммутатором RapidIO или ОЭ конечной точки для остановки (XOFF) и запуска (XON) потока трафика. Единственный поток запроса транзакций управления потоком оперирует пакетами CCP. Пакеты типа 7 не имеют поля полезной нагрузки данных и не генерируют пакетов отклика. Источник пакетов контроля потока должен установить бит SOC (источник перегрузки) на "0", если это коммутатор, и на "1", если это конечная точка. Этот бит информативен по сути и может использоваться для идентификации отказов конечных точек. Формат пакета CCP приведен на рис. 16.

Описание его полей приводится ниже:

- 0111 (Type 7) – пакет типа 7 (4 бита);
- XON/XOFF – поле (1 бит) для остановки (XOFF) и запуска (XON) потока трафика;
- flowID – поле (7 бит) – ID потока запроса транзакций;

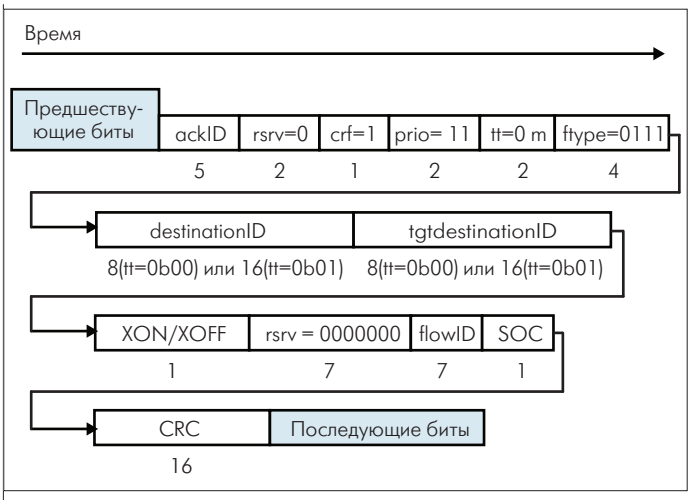

Рис. 17. Формат пакета управления потоком при последовательной передаче

- destinationID – поле (8 или 16 бит) указывает, какая конечная точка CCP назначается для "sourceID" пакета, который вызвал генерацию данного CCP;
- tgtdestinationID – поле (8 или 16 бит), объединенное с полем flowID, указывает, какие потоки запроса транзакций должны быть задействованы в соответствии с полем destinationID пакета, который вызвал генерацию данного CCP;
- SOC – поле (1 бит) – переключатель источника перегрузки: "0" – коммутатор, "1" – конечная точка.

На рис. 17 и 18 приведены полные форматы пакетов управления потоком при последовательной и параллельной передаче. Поле "destinationID" пакета CCP на рис. 17 есть поле "sourceID" для пакетов, ассоциированных с перегрузкой, и является точкой назначения для транзакции потока управления. Поле tgtdestinationID есть поле destinationID для пакетов, ассоциированных с перегрузкой, и было точкой назначения для этих пакетов. Поле tgtdestinationID используется как точка назначения для пакетов управления потоком, чтобы идентифицировать поток запроса транзакций, в соответствии с которым нужно действовать. Размер этого поля может быть 8 бит для малого транспортного формата (tt=0b00) или 16 бит для большого транспортного формата (tt=0b01).

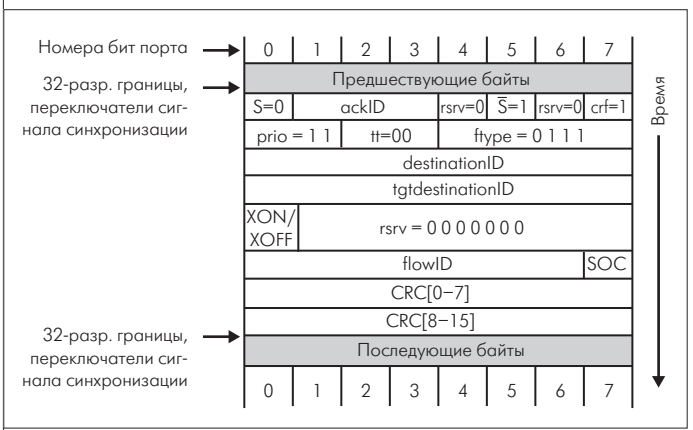

Рис. 18. Формат пакета управления потоком при параллельной передаче

ЛОГИЧЕСКИЙ УРОВЕНЬ. ПОТОКОВАЯ ПЕРЕДАЧА ДАННЫХ

Эта часть логической спецификации [2к] была разработана для повышения эффективности, гибкости и увеличения независимости протоколов с целью минимизировать ресурсы, необходимые для поддержки структуры взаимосвязи в плане передачи данных, и сохранить совместимость и полноту взаимодействия с остальными спецификациями RapidIO. Рациональность такой оптимизации в том, что сама платформа RapidIO обладает значительно большими возможностями, чем предполагалось вначале.

Существует ряд схем, позволяющих инкапсулировать кадры одного протокола в поля другого протокола (см., например, RFC1226, RFC1234, RFC1701). Логическая спецификация потоковой передачи данных определяет механизм транспортировки (из конца в конец цепи связи) любого протокола через стандартный интерфейс RapidIO и адресные связи между элементами. Принятая методология инкапсуляции обеспечивает одновременное использование различных протоколов сетевого уровня на одном и том же звене и дает общее решение для связи различных хостов, мостов и коммутаторов, обеспечивающее передачу широкого спектра типов данных.

Потоковая передача данных имеет следующие особенности.

- Инкапсуляция протокола не зависит от типа инкапсулируемого протокола.
- Используется механизм сегментации и повторной сборки протокольного блока данных (PDU), допускающий размер PDU до 64 кбайт.
- Поддерживаются сотни классов трафика и тысячи потоков данных между конечными точками.
- Поддерживаются одновременные потоки PDU с интерливингом между конечными точками.
- Обеспечивается гладкое взаимодействие с другими спецификациями RapidIO.
- Спецификация пакетов не зависит от типа связи с другими устройствами на физическом уровне коммутационной структуры.
- Протоколы и форматы пакетов не зависят от физической топологии связи. Протоколы работают на топологиях типа "точка-точка", "кольцо", "шина", "коммутируемая многомерная сеть", "дуплексное последовательное соединение" и т.д.
- Протокол потоковой передачи требует упорядоченной передачи и приема пакетов, неупорядоченная доставка пакета недопустима.
- Если некоторые устройства имеют ограничения на ширину полосы и задержку при выполнении определенных операций, то RapidIO не препятствует специальным реализациям системы, которые налагают эти ограничения на всю систему.

Система потоковой передачи данных

На рис. 19 приведен пример блок-схемы системы связи на основе RapidIO, требующей использования протокола инкапсуляции. На ней показан ряд типичных трактов данных, использующих как специализированные, так и широко распространенные (стандартные) интерфейсы. Все они связаны с коммутационной структурой RapidIO.

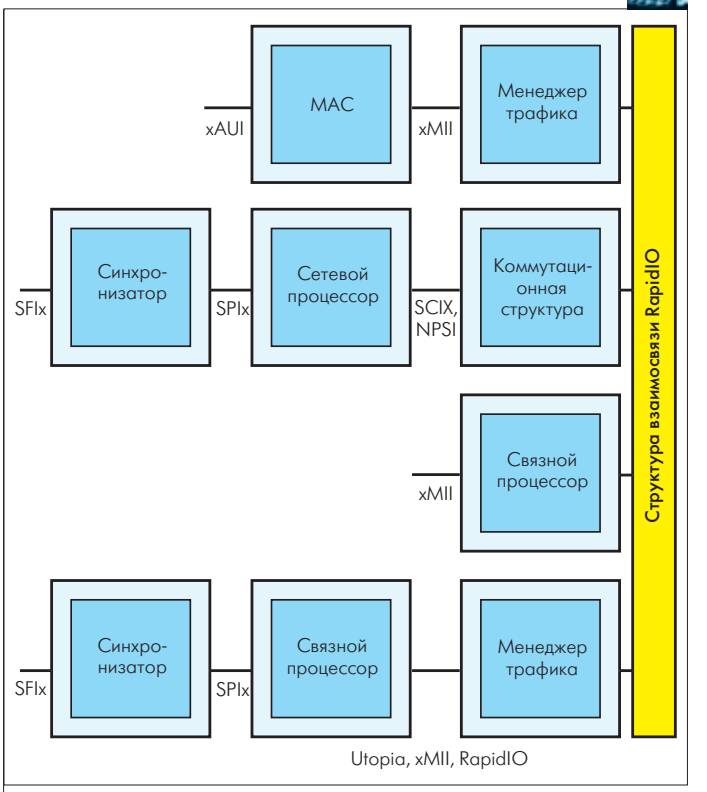

Рис. 19. Блок-схема системы сетевой связи на основе RapidIO

Потоки данных представлены логическими соединениями между входными и выходными портами. Эти соединения перекрывают однонаправленную передачу блоков PDU, которые разделены интервалами, зависящими от времени прихода данных на входной порт. В общем случае передача может быть двунаправленной и независимой. Общий вход может обслуживать один поток, сотни, тысячи и миллионы потоков одновременно в зависимости от уровня классификации PDU.

Транзакции потоковой передачи отличаются от большинства других транзакций RapidIO двумя особенностями: они способны передавать больше типоразмеров данных и не требуют квитирования с помощью пакетов отклика. Спецификация потоковой передачи поддерживает самые разные устройства, отличающиеся интерфейсами, протоколами и степенью сложности.

Потоки трафика между конечными пользователями идентифицируются потоковыми ID, которые зависят от количества уровней в реализации протокола (т.е. количества уровней PDU). В потоке могут быть разные PDU, либо связанные между собой, либо нет, однако *поток трафика* представляет собой последовательность *упорядоченных* PDU.

Логический уровень потоковой передачи использует *ID виртуального потока* (VSID), позволяющий однозначно идентифицировать и одновременно управлять многими потоками трафика (потоками упорядоченных PDU). Формирова-

ние VSID осуществляется в процессе классификации PDU. Сложность процесса пропорциональна сложности системы передачи, реализуемой для данного приложения. VSID объединяет поля ID источника (source) и ID назначения (destination), класса сервиса (class of service) и поле "streamID".

Поток запроса транзакций определяется как упорядоченная последовательность транзакций запроса, охватывающая только PDU от заданного источника до назначения (на что указывают ID источника и назначения транзакций). Каждый пакет в этом потоке имеет одну и ту же пару ID источника-назначения. Все потоки трафика отображаются на потоки запроса транзакций. Эти потоки могут быть использованы совместно с транзакциями других логических уровней RapidIO, и поэтому взаимосвязь между потоками, классами трафика, виртуальными очередями и всеми потоками запроса транзакций RapidIO зависит от реализации. Допускаются множественные потоки запроса транзакций, причем каждый из таких потоков отличается своим ID ("flowID"), введенным в спецификации ввода-вывода [2a]. Этот ID представляет собой самый нижний уровень управления трафиком в системе RapidIO.

Классы сервиса и виртуальные очереди

Хотя системы потоковой передачи могут поддерживать тысячи потоков данных, эти потоки фактически мультиплексируются в один пакет транспортного уровня. В качестве механизма принятия решения, какие потоки могут использовать общие ресурсы, выступает виртуальная очередь. Для более профессионального управления потоками трафика вводится понятие идентификатора (ID) класса сервиса (CoS) через поле "cos". Связь между параметрами "flowID" и "cos" зависит от реализации.

На входе RapidIO тысячи потоков могут комбинироваться в несколько виртуальных выходных очередей (VoQ), используя ID назначения "Dest. ID" и "cos" (которые являются частями VSID), как показано на рис. 20. Поле "cos", определенное этой спецификацией, имеет длину 1 байт и определяет максимальное число буферных структур данных (равное 256), которое может быть использовано конкретными устройствами.

На рис. 20 показано (на примере "Потока B"), что виртуальные потоки при потоковой передаче могут смешиваться

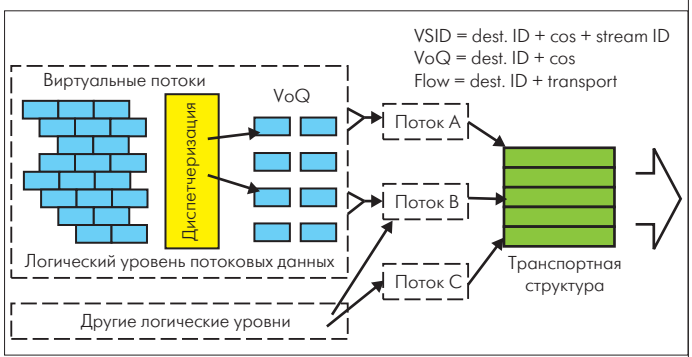

Рис. 20. Схема отображения виртуальных потоков данных на системный вход

с другими потоками логического уровня, прежде чем попадут на транспортный уровень. Обратный процесс в точке назначения осуществляется по схеме, противоположной той, что показана на рис. 20, только вместо виртуальных выходных очередей используются виртуальные входные (ViQ).

Протокол передачи потоковых данных

Передача потоковых данных значительно отличается от передачи данных в компьютерных системах с каналом ПДП. Основное отличие в том, что данные в процессе передачи многократно инкапсулируются. Например, они могут вначале инкапсулироваться в пакеты Ethernet, которые, в свою очередь, инкапсулируются в пакеты TCP/IP. Другая особенность в том, что RapidIO обрабатывает тысячи потоков трафика, часть которых генерируется не устройствами системы RapidIO, а удаленными пользователями (ПК, соединенными с ЛВС). Все это требовало разработки нового протокола передачи потоковых данных.

Этот протокол использует только транзакции запроса. Как и многие протоколы доставки данных, они перекладывают гарантии доставки данных на протокол верхнего уровня (это же относится и к управлению при возникновении ошибок и перегрузок в сети), избавляясь от необходимости использовать транзакции отклика.

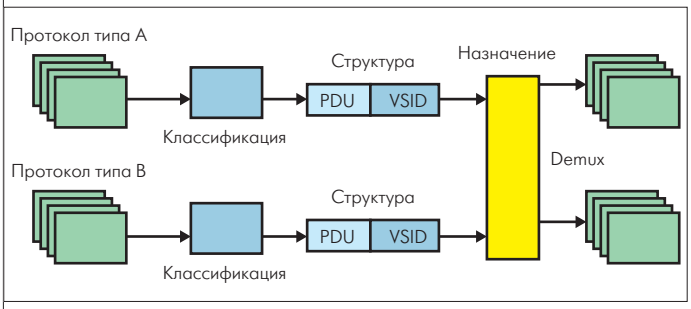

Рис. 21. Схема виртуальных потоков, использующих VSID

Операции потоковой передачи

Операции потоковой передачи (в отличие от предыдущих) синхронизируются и относятся к классу синхронных, а значит, потоки (будучи виртуальными) могут быть повторно использованы для передачи различных данных.

Операция потоковой передачи состоит из отдельных транзакций потоковой передачи. Последовательность транзакций передает PDU между двумя конечными точками. Так как транзакции отклика не используются, то не нужно ждать уведомлений о приеме, и скорость передачи может быть существенно увеличена.

Поток использует уникальный ID VSID, позволяющий обрабатывать поток при прохождении RapidIO. VSID создается путем "классификации" PDU, проводимой один раз при входе в систему RapidIO (рис. 21), после чего обработка PDU не зависит от протокола. VSID используется также в точке назначения для "реклассификации" PDU, после чего данные возвращаются

в исходный вид и становятся протокольно-зависимыми. Такая модель виртуальной адресации позволяет обойтись без использования выравнивающих буферов и других ресурсов.

VSID – это связка, объединяющая несколько полей: ID источника/назначения (source/destination), "cos" (качество обслуживания) и ID потока ("streamID"). Так, объединение полей "destination ID+cos+streamID" представляет уникальный ID источника, а полей "source ID+cos+streamID" – уникальный ID назначения.

PDU и их сегментация

Поток трафика состоит из последовательности связанных и упорядоченных PDU, передаваемых также последовательно и с сохранением упорядоченности. Если PDU не связаны между собой, из них можно сформировать другие потоки или их можно мультиплексировать в общий поток. В зависимости от состояния коммутационной структуры допускается несколько потоков PDU, но структура должна гарантировать, что порядок доставки PDU при этом не нарушится.

Большие PDU должны *сегментироваться* при передаче, чтобы удовлетворить ограничения RapidIO на максимальную длину полезной нагрузки в 256 байт, и *повторно собираться* в исходный PDU при приеме, реализуя функцию SAR (как в технологии ATM). Максимальный размер PDU определяется при конфигурации в регистре CAR. Размер блока, используемого в процессе сегментации, определяется параметром MTU (максимальный передаваемый блок), который должен согласовываться со всеми элементами, участвующими в процессе SAR.

Транзакции, участвующие в поточной передаче, также именуются сегментами. Типичная последовательность составлена из сегментов трех типов: *начального сегмента*, *сегментов продолжения* и *конечного сегмента*. Первые два обычно имеют размер MTU, а последний – переменный размер, так как содержит остаток PDU. Если размер PDU меньше MTU, то PDU передается одним сегментом переменного размера, зависящего от PDU. Все сегменты PDU должны иметь одни и те же значения параметров "flowID" и "cos".

Начальный сегмент содержит поля, необходимые для идентификации VSID и "открытого" *контекста сегментирования*. Этот контекст определяется как комбинация "source ID" и "flowID" и используется приемником для повторной сборки данного PDU. Использование этой комбинации позволяет каждой паре "источник-назначение" иметь одно PDU для каждого "flowID". VSID используется при открытии процесса сегментации и в точке назначения для воссоединения данного PDU с его потоком, так как сегменты продолжения и конечный сегмент не несут этой информации. После получения последнего сегмента контекст сегментирования закрывается. В тот пери-

од, когда контекст открыт, ни поток, ни PDU, ассоциированные с контекстом, не должны изменяться. Так как одновременно существует большое количество источников PDU и контекстов на один источник, то число состояний контекстов, которое должно обрабатываться в точке назначения, потенциально очень велико. Число максимально поддерживаемых контекстов указано в регистре CAR.

Для увеличения эффективности передачи информация о том, какой блок PDU содержится в каком пакете, не включена в конкретный заголовок. Эта ситуация требует, чтобы передатчик выпустил конкретную последовательность, начиная с первого блока PDU и кончая последним блоком, а транспортная структура доставила эту последовательность к логическому уровню потоковой передачи в том же порядке.

Пример сегментирования пакета. Рассмотрим процесс разбиения PDU на сегменты. Пусть, например, длина PDU – 21 байт, а MTU=8 байтов, тогда получаем три сегмента: начальный (8 байтов), продолжение (8 байтов) и конечный (5 байтов). Конечный сегмент должен быть дополнен 1 байтом (pad) для выравнивания по длине полуслова (2 байта). В результате, в заголовке этого сегмента (см. ниже) нужно указать, что бит P=1 (присутствует байт дополнения – pad) и бит O=1 (число байтов полезной нагрузки нечетное – odd). Эти биты позволяют четко определить длину PDU в точке назначения. Подробные правила сегментации PDU и их сборки приведены в разделе 3.2.5 стандарта [2к].

Классы сервиса и потоки трафика

ID виртуального потока разбивается на три части: порт (задается ID источника/назначения), класс (задается полем "cos") и поток (задается полем "streamID"). Такой ID характерен как для отдельных потоков, так и для групп трафика. В точках входа, выхода и внутри системы (при дефиците ресурсов) такой трафик может быть снова разделен и поставлен в очередь, используя вторую часть ID "класс".

В транспортной структуре коммутация пакета осуществляется по ID назначения и ID отображенного потока ("flowID"), см. выше. Полный ID класса сервиса (CoS ID) является подмножеством VSID. Он состоит из ID источника/назначения (или порта входа/выхода) и поля "cos". При этом входная очередь должна быть основана на "destination ID+cos", а выходная очередь должна быть основана на "source ID+cos", как показано на рис. 22. Включение ID источника/назначения в CoS ID позволяет приобщить класс сервиса к образованию пары "источник-назначение". Полем "cos" следует пользоваться, начиная с MSB (нулевой бит) и используя необходимое число бит в зависимости от числа поддерживаемых классов (нулевой бит для двух классов, биты 0 и 1 для четырех, биты 0, 1 и 2 для восьми и т.д.).

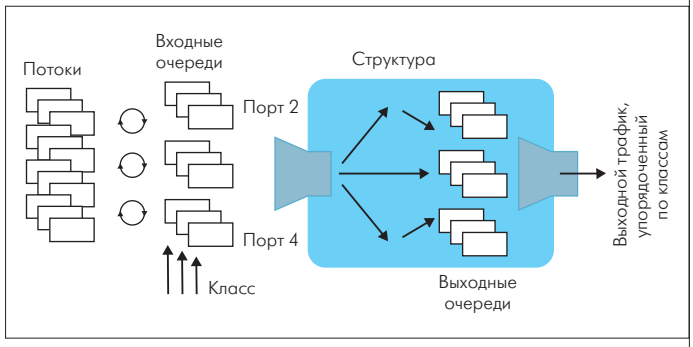

Рис. 22. Сортировка трафика на основе CoS ID

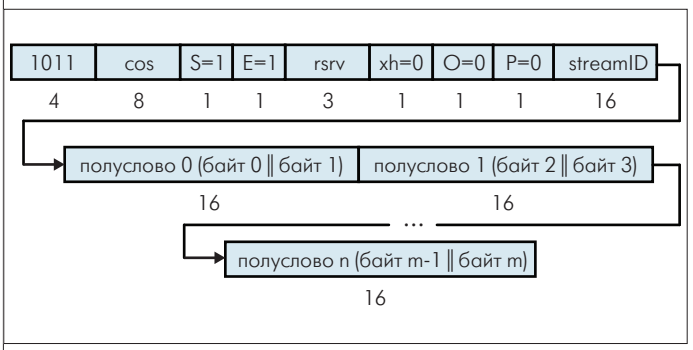

Рис. 23. Пример сегмента в формате пакета потоковой передачи данных

Формат пакета потоковых данных

Форматом пакетов потоковых данных является формат транзакции DATA STREAMING (type 9). Пакеты этого типа всегда имеют полезную нагрузку данных, пока PDU не терминировано. В отличие от других логических спецификаций RapidIO длина полезной нагрузки данных выравнивается по полуслову (2 байта), а не по двойному слову (8 байтов). Общая длина заголовка одного сегмента пакета составляет 20 бит. Формат односегментного пакета приведен на рис. 23 и содержит следующие поля:

- 1001 (9) – преамбула (4 бита), указывает, что формат пакета – "type 9";
- cos – класс сервиса (8 бит), используется в точке назначения для доопределения потока трафика;
- S (Start) – поле (1 бит), указывает (если S=1), что этот пакет – начальный сегмент нового PDU;
- E (End) – поле (1 бит), указывает (если E=1), что этот пакет – конечный сегмент данного PDU;
- rsrv – резервное поле (3 бита), устанавливается на "0" отправителем, игнорируется получателем;
- xh – расширенный заголовок (1 бит), дает возможность расширить заголовок пакета type 9 в будущем;
- O (odd) – поле (1 бит), указывает (если O=1) на нечетное число полуслов в полезной нагрузке;
- P (pad) – поле (1 бит), указывает (если P=1), что 1 байт был добавлен в полезную нагрузку для выравнивания ее по границе полуслова (позволяет (см. пример выше) отправителю передавать нечетное число байтов в пакете);
- streamID – ID потока трафика (16 бит).

Если оба поля S и E равны "1", то такой PDU полностью содержится в одном пакете. Размер полезной нагрузки такого пакета может соответствовать MTU или быть меньше ее (n и

m определяются размером PDU). Только этот тип пакета содержит поле "xh".

Для пакета, состоящего из нескольких сегментов, существует три типа заголовков: для начального (S=1) и конечного (E=1) сегментов и для сегмента продолжения (S=0, E=0). Заголовки этих типов сегментов отличаются структурой полей от заголовка на рис. 23. Так, для начального сегмента и сегмента продолжения вместо связки полей "xh-O-P" (3 бита) используется связка "0" (фиксированный бит 0)-"rsrv" (2 бита), а для конечного сегмента – связка "0"-O-P.

ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ VSID

Механизм идентификации виртуального потока с помощью 32-разрядного ключа (VSID) обеспечивает ряд особенностей системы RapidIO [2к, Annex A], а именно:

- механизм управления трафиком при входе и внутри системы;
- независимые от протокола теги для реклассификации пакетов на выходе системы;
- механизм гибкой (на уровне "субпорта") адресации;
- независимость в управлении буферами.

Адрес порта, используемый в VSID, это или ID назначения, или ID источника, в зависимости от того, где находится пакет: на входе или на выходе системы. ID назначения/источника составляют пару, что позволяет устанавливать другие поля ("class" и "streamID") независимо от того, как они используются с другими парами. Использование VSID может меняться в зависимости от сложности системы и требований со стороны приложений.

Классификация пакетов

Все PDU требуют классификации при входе в систему. Анализируются поля PDU и формируется маршрутная информация. Сформированный VSID является 32-разрядным тегом (дескриптором), а не адресом порта. В точке назначения он может быть использован для повторной связки PDU с буфером назначения. Это можно сделать либо прямой адресацией, либо с помощью таблицы перекодировки с одним ключом. VSID дает ясно очерченный и независимый от протокола путь сортировки трафика и виртуальный механизм управления буферным пулом. Без использования виртуального тега пакет должен был бы подвергнуться реклассификации, основанной на той части PDU, которая зависит от протокола. В мультисервисных платформах это приводит к череде процессов, повторяющих то, что уже было сделано у источника.

Адресация на уровне субпорта

Простейшее использование VSID – демультиплексирование трафика в разнесенные субпорты в точке назначения. Это

позволяет разделить трафик по протоколам или по ряду субпортов с тем же протоколом.

Приложение типа DSLAM. Пусть линейная карта содержит 128 пользовательских портов. Система может представлять каждый из них как независимый пункт назначения системы RapidIO, требующий использования исключительно большого числа ID назначения и увеличенного заголовка. Альтернативой этому может быть инкапсуляция ATM-трафика в 128 VSID (по одному на каждый порт). Линейная карта тогда займет единственный порт в системе RapidIO. VSID может также быть использован как адрес для распараллеливания трафика на различные шины (типа UTOPIA), ведущие к различным пользовательским портам. Кроме того, такая схема имеет преимущества при восстановлении после сбоев. Тогда при сбое в линейной карте нужно будет восстановить в таблице маршрутизации системы одну ячейку (отображающую единственный порт), а не все 128 субпортов.

Приложение типа VOIP. VSID можно использовать для разделения трафика всего на два канала, один – для управляющего процессора, обрабатывающего управляющие сообщения, а второй – для сетевого процессора, который распространяется на все цифровые сигнальные процессоры (DSP). VSID мог бы содержать адрес DSP назначения для того, чтобы затем разгрузить сетевой процессор на распределение. VSID мог бы также дополнительно содержать канал пользователя в рамках DSP-демультиплексирования трафика.

Виртуальная выходная очередь

Приложения, работающие с большим числом потоков, могут использовать поле "класс" ("class") для регулирования входа в систему (известного как виртуальная выходная очередь). Например, интерфейс системы RapidIO может содержать 256 очередей для 64 портов назначения с четырьмя классами трафика. Трафик для каждой точки назначения того же класса должным образом "взвешивается". Взвешивание между классами может быть уникальным для каждого приложения. Трафик при этом сохраняется отсортированным по назначению. Если трафик был просто выгружен в 4 очереди, а порт назначения дал сбой, то этот трафик можно перекинуть на другой порт, в противном случае его можно сбросить на время, пока порт не будет восстановлен, или переадресовать. Сохраняя трафик отсортированным по назначению на входе системы, можно переадресовать это назначение с минимальными потерями трафика.

Буферы при формировании виртуальной выходной очереди могут быть расширены до 2 и даже до 16 K в зависимости от размера системы и того, как много различных классов трафика обслуживается. Это управление входом системы может оказаться простым механизмом для добавления системе качества обслуживания (QoS) путем использования ID назначения и части "class" в VSID. Заметим, что это можно сделать, не применяя "streamID" для демультиплексирования в точке назначения.

Системные требования

Использование VSID определяется тремя элементами в системе: источником, обрабатывающей структурой и назначением.

Мост ATM UTOPIA-RapidIO. Этот мост классифицирует трафик, используя поле VPI как порт назначения, а VCI как адрес субпорта. Он отображает весь трафик (type 9) на единственный поток RapidIO, устанавливая класс "0" и данное значение "streamID" в качестве VCI для коммутаторов на потоке. Порт назначения использует эту часть "streamID" в VSID как аппаратный адрес субпорта.

Сетевой процессор (NP) содержит звено связи уровня OC-48, агрегирующее трафик множества 1 Мбайт/с портов, размещенных на линейных картах. NP классифицирует трафик каждого пользователя на два класса: приоритетный – для голоса (используя RTP) и неприоритетный – для всех остальных. Это устанавливает поле "класс" на "0" или "1", данный порт отводится соответствующей линейной карте, а "streamID" отводится желаемому субпорту.

Интерфейс CSIX-RapidIO. Пакет CSIX содержит поля "назначения" и "класса" (источник в интерфейсном чипе является предустановленным параметром). Поле "streamID" занимает первые 16 бит полезной нагрузки CSIX. Пакет RapidIO легко формируется из этой информации. Интерфейс системы содержит множество виртуальных выходных очередей, две на каждый порт назначения. Так как интерфейс CSIX-NP также является сегментированным интерфейсом, то PDU собирается вновь в виртуальных очередях, пока доступно достаточное количество информации, чтобы сформировать требуемое MTU в системе RapidIO. Эта система отображает данный класс на приоритетный или неприоритетный поток. Данный порт назначения использует указанное "streamID" для отображения соответствующего трафика на корректный субпорт пользователя. Каждый субпорт содержит два класса очередей для того, чтобы собрать трафик, как только он пройдет повторную сборку.

10-гигабитный MAN-интерфейс. Специализированный классификационный процессор создает 32-разрядный VSID на основе IP, TCP/UDP и информации приложений. Этот тег присоединяется перед SPI4.2-пакетом. Интерфейсом к системе служит мост SPI4.2-RapidIO, который содержит виртуальные выходные очереди. Пунктом назначения служит процессор, который поддерживает только память и логические транзак-ции ввода/вывода. Интерфейсный мост RapidIO-процессор содержит буферы сегментации-сборки и таблицы перекодировки (look up tables) ассоциированных машин, которые отображают VSID на адреса буфера ПДП (DMA), и наоборот.

Система содержит множество таких обрабатывающих карт для поддержки трансляции адреса, шифрования или функции межсетевого экрана. Источник классифицирует трафик, основанный на том, какие используются приложения. Путем размещения адреса буфера в точке назначения и присвоения "streamID" формируется соединение. Создается таблица источника с требованиями "дерева поиска" для заданного протокола, устанавливая в результате VSID.

В точке назначения VSID может использоваться на аппаратном уровне или его можно гибко отображать на виртуальные буферы. В любом случае сам источник должен быть достаточно интеллектуальным, чтобы назначить VSID в соответствии с тем, что нужно в точке назначения. Это не вызывает вопросов, так как источнику в любом случае нужно классифицировать пакет для определения точки назначения. VSID может быть использован для разделения трафика по протоколам, субпортам, классам сервиса или по требуемому количеству очередей. Если назначение управляет большим количеством буферов, VSID позволяет назначению использовать единый, независимый от протокола ключ, чтобы заново отобразить данный трафик и полностью абстрагироваться от любого управления буферами.

ЛОГИЧЕСКИЙ УРОВЕНЬ. СПЕЦИФИКАЦИЯ ПОДДЕРЖКИ МУЛЬТИКАСТИНГА

Концепция дублирования единственного сообщения и посылки его многим избранным назначениям известна как *мультикастинг* (multicast). Она полезна во многих системах. Наиболее эффективна аппаратная реализация. Внутри системы RapidIO возможность дублирования сообщения становится все более масштабной с ростом числа конечных точек [2л]. Так как число конечных точек растет с числом коммутаторов, установленных в системе, то поддержка мультикастинга описывается только для коммутаторов, а конечные точки не затрагиваются. Возможный учет конечных точек описан в Приложении А [2л].

Спецификация поддержки мультикастинга ограничивается транзакциями запроса, которые не требуют отклика (например, транзакциями типа SWRITE [2а]). Это объясняется тем, что реализация поддержки получения транзакций отклика внутри коммутатора, который, как правило, не принимает во внимание протоколы логического уровня, сложна и проблематична.

Для коммутатора возможность послать одно сообщение по нескольким назначениям можно реализовать различными путями в зависимости от потребностей системы. Есть, однако, две причины, мотивирующие определение общего интерфейса и режима работы мультикастинга в системе. Во-первых, без описания стандартного интерфейса и режима его работы широкий круг возможных реализаций не позволит использовать общие программные драйверы мультикастинга. Во-вторых, без стандартных определений для интерфейса и режима работы невозможно гарантировать взаимодействие различных компонентов, поддерживающих мультикастинг. При определении общего интерфейса необходимо определить этот стандартный интерфейс на определенном уровне абстракции, чтобы избежать ограничений гибкости реализации. В помощь этому в спецификации [2л] приводятся примеры использования такого интерфейса. В любом случае механизм мультикастинга должен быть простым, компактным, надежным, масштабируемым и совместимым со всеми физическими уровнями.

Режим работы мультикастинга

В системе RapidIO могут сосуществовать коммутаторы, не поддерживающие и поддерживающие мультикастинг. Единственное требование к ним в том, чтобы они могли маршрутизировать ID назначения, используемые для мультикастинга транзакций.

Размножение пакета. Операция мультикастинга состоит из размножения отдельного пакета так, чтобы его могли получить несколько конечных точек. Это размножение осуществляется коммутатором, а не конечной точкой. Это значит, что возможности размножения пакетов расширяются с ростом числа коммутаторов.

Организация мультикастинга. Каждый коммутатор отдельно может быть запрограммирован управлять рассылкой размноженных пакетов к его выходным портам. Эти пакеты сами по себе не модифицируются в процессе размножения, а просто передаются через соответствующие порты. Спецификация [2л] адресует пакеты запроса мультикастинга только для транзакций, не требующих отклика. Это существенно упрощает поддержку мультикастинга коммутаторами RapidIO, которым не нужно собирать отклики от других типов операций RapidIO. Примерами транзакций, которые могут работать в режиме мультикастинга, являются транзакции NWRITE и SWRITE логической спецификации ввода/вывода. Транзакции мультикастинга, которые требуют отклика, имеют режим работы, определяемый конкретной реализацией.

СПИСОК СОКРАЩЕНИЙ (окончание)

RapidIO – открытый стандарт коммутационной структуры последовательного типа, разработан Ассоциацией RTA (8.04) для встраиваемого компьютерного оборудования; определяет 3 уровня: логический, транспортный и физический, для которого предлагается XAUI-совместимый интерфейс со скоростями 1,25; 2,5; 3,125 Гбит/с (линейный код 8B/10B); может объединять до 4 полос трафика (пропускная способность 20 Гбит/с).

RTA – RapidIO Trade Association – Профессиональная ассоциация RapidIO, www.RapidIO.org (разработчик коммутационной структуры последовательного типа RapidIO).

SAR – Segmentation And Reassembly – сегментация и сборка – процессы, происходящие на подуровне SAR ATM, который осуществляет сегментацию данных (или блока PDU) на 48-байтные блоки для передачи их в ATM-ячейках с последующей сборкой для восстановления исходного формата данных (или блока PDU) на другом конце звена данных.

SPI – System Packet Interface – системный пакетный интерфейс (например, SPI-4.2 компании Silicon Logic Engineering).

UTOPIA (Utopia) – Universal Test and Operations Physical layer Interface – универсальный интерфейс для тестирования и операций физического уровня.

VCI – Virtual Channel Identifier – идентификатор виртуального канала – идентификатор, используемый коммутаторами для установления виртуального канала; в ATM – идентификатор виртуального канала в рамках, ограниченных заданным виртуальным маршрутом VPI.

VoIP (VOIP) – Voice (services) Over Internet Protocol – передача голоса с помощью IP-протокола.

VPI – Virtual Path Identifier – идентификатор виртуального маршрута/пути – идентификатор, используемый коммутаторами для установления виртуального маршрута/пути; в ATM – часть заголовка ячейки, определяющая совместно с VCI виртуальное соединение.

XAUI – 10 Gigabit Attachment Unit Interface – интерфейс подключаемого блока/устройства 10-гигабитного Ethernet – интерфейс между двумя 10-гигабитными подуровнями расширителя (XGXS), позволяющий увеличить зону действия интерфейса XGMII для 10 Гбит/с.

Операции мультикастинга

Операции мультикастинга имеют два управляющих типа – *маска мультикастинга* (multicast mask) и *группа мультикастинга* (multicast group). Множество конечных точек назначения, которые получают соответствующий пакет мультикастинга, называется группой мультикастинга. Каждая такая группа ассоциируется с уникальным ID назначения. Он позволяет коммутатору RapidIO определить, что полученный пакет подлежит размножению для мультикастинга.

Маска мультикастинга – это значение, которое управляет тем, с какими выходными портами ассоциируется одна или многие группы мультикастинга. Концептуально маска мультикастинга – регистр с одним разрядом на каждый выходной порт коммутатора. Существует один набор таких масок на весь коммутатор. Всем маскам в коммутаторе присвоены уникальные последовательные ID-номера начиная с "0".

Пример конфигурации мультикастинга показан на рис. 24. В этом примере в качестве ID назначения, используемого для мультикастинга, выбран ID 0x80. Допустим, что с этим ID на входе "0" коммутатор ассоциирует выходы "1–4". Группа мультикастинга A составлена из произвольно выбранных конечных точек: 0x10, 0x15, 0x16 и 0x17. ID 0x80 при конфигурации коммутатора ассоциирован с маской мультикастинга номер 2. Группа и маска мультикастинга привязаны к глобальной карте системных адресов (через регистры CSR). Конфигурирование коммутатора для размножения пакета осуществляется в два этапа. На первом этапе список выходных портов коммутатора помещается в маску мультикастинга. На втором один или несколько ID назначения, которые представляют группы мультикастинга, ассоциируются с маской мультикастинга номер 2 данного коммутатора. Коммутатор посылает пришедший пакет по всем выходным портам, для которых в маске мультикастинга стоят "1". Операция ассоциации осуществляется с использование регистров CSR.

ЛОГИЧЕСКИЙ УРОВЕНЬ. ГЛОБАЛЬНАЯ ОБЩАЯ ПАМЯТЬ

Эта часть логической спецификации [2д] описывает глобальную распределенную общую память (GSM). Стандарт RapidIO поддерживает ее, несмотря на то что ориентирован на программную модель пакетной передачи сообщений. Программная модель, ориентированная на GSM, как правило, используется компьютерными системами общего пользования с мультиобработкой, которые требуют аппаратной поддержки когерентности кэш-памяти. Здесь дополнение GSM допускает существование под управлением одного протокола как распределенной обработки ввода/вывода, так и мультиобработки систем общего пользования.

Поддержка моделей когерентности, кэш-памяти со сквозной записью, каталогов памяти (или их эквивалентов) для ускорения удаленного доступа к памяти является прерогативой конечных точек (МП, памяти и, возможно, устройств ввода-вывода), использующих операции RapidIO. Причем RapidIO

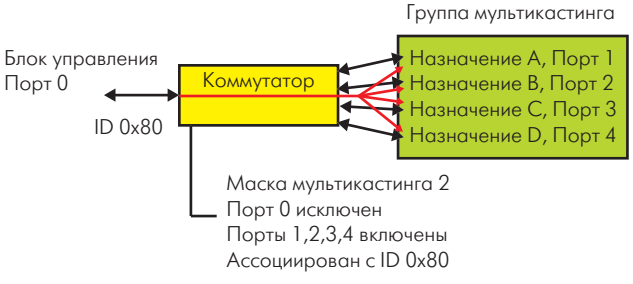

Рис. 24. Пример конфигурации мультикастинга

обеспечивает эти операции, чтобы можно было формировать большое разнообразие систем, использующих широкий диапазон программных моделей: от основанных на строгой непротиворечивости, через модели с полной упорядоченностью хранения данных до моделей со слабой упорядоченностью. Взаимодействие между конечными точками, поддерживающее различные модели когерентности/кэширования/каталогизации, при этом не гарантируется. Однако группы конечных точек с согласованными моделями могут соединяться с другими группами, согласованными с другими моделями, на той же коммутационной структуре RapidIO. Различные группы могут взаимодействовать, используя операции ввода/вывода и передачи сообщений RapidIO.

Система памяти

Для программной модели GSM память физически может располагаться в разных местах в машине, оставаясь доступной для всех ОЭ. Широко используемые архитектуры имеют адресуемую общую память, использующую вещательную передачу транзакций, известную как *отслеживающий протокол* на базе шины (или отслеживающая шина – snoopy bus). Обычно существует центральный контроллер памяти, для которого все устройства являются однородными и равнодоступными. На рис. 25 показана такая типичная система общей памяти на основе отслеживающей шины. Суперкомпьютеры, машины с массовым параллелизмом и кластеризованные машины, имеющие распределенную систему памяти, используют для поддержки ее когерентности методы, отличные от вещательной передачи, например CC-NUMA (кэш-когерентный неоднородный доступ к памяти), так как отслеживающий протокол вещательного типа в этих машинах не эффективен.

Для RapidIO выбрана относительно простая схема поддержки когерентности на основе каталога памяти. В этом методе каждый контроллер памяти отслеживает, где в системе находится самая последняя копия каждого элемента данных. RapidIO предлагает набор специфических для шины ISA средств управления кэш-памятью, и ОС поддерживает такие операции, как сброс содержимого блока кэш-памяти и механизмы синхронизации буфера динамической переадресации (TLB). Для уменьшения требуемого каталожного заголовка архитектура оптимизирована относительно малых кластеров для 16 процессоров, известных как *домены когерентности*. Концепция таких доменов позволяет множественным когерентным группировкам существовать в такой взаимосвязи в виде жестко связанных обрабатывающих кластеров.

Ниже представлены особенности спецификации RapidIO GSM, призванные удовлетворить требованиям различных приложений.

- Поддержка системной архитектуры CC-NUMA для обеспечения модели GSM, учитывая, что физические соображения вынуждают использовать интерфейс с топологией точка-точка вместо традиционной шины.

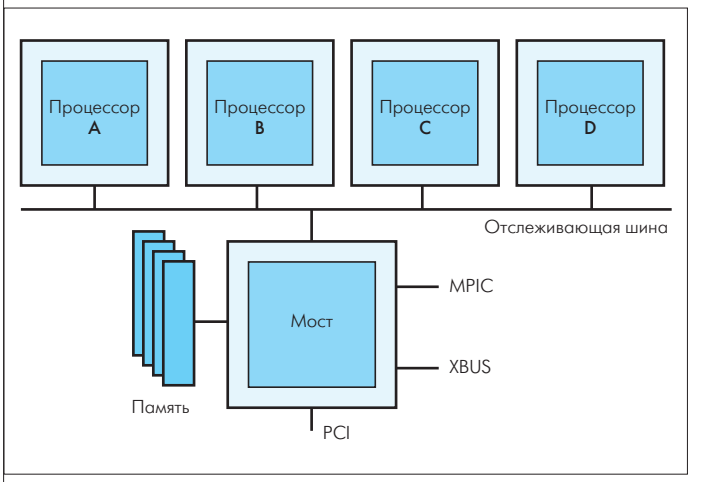

Рис. 25. Система общей памяти с отслеживающей шиной

- Размер заявок процессорной памяти соответствует или меньше гранулярности кэш-когерентности (размер гранул когерентности зависит от типа МП или от реализации).
- Набор команд RapidIO поддерживает различные операции управления кэш-памятью (в том числе сброс содержимого блока кэш-памяти). Эти функции позволяют обеспечить работу стандартных приложений и ОС.
- Протоколы и форматы пакетов не зависят от физической топологии связи, а протоколы работают на топологиях "точка-точка", "кольцо", "шина", "коммутируемая много-мерная сеть", "последовательное дуплексное соединение" и т.д.
- Протоколы обрабатывают прием и передачу неупорядоченных данных.
- Поддерживаются 48- и 64-разрядные адреса, которые могут потребоваться в будущем.

Модели системы с различными типами ОЭ, описанные в [2д, гл. 2], аналогичны описанным выше, см. также [2а, гл. 2].

Операции GSM

Для поддержки кэш-когерентных операций GSM на кэшируемом пространстве памяти используется набор транзакций. Операции GSM основаны на размере гранул когерентности. Изменение гранул когерентности в системе не изменяет операционных протоколов, но изменяет размер полезной нагрузки данных. Исключением являются операции "flush" и "I/O read", которые могут потребовать субгранулы когерентности для поддержки когерентного ввода-вывода и операций кэш-памяти со сквозной записью. Операции сброса содержимого блока кэш-памяти ("flush") могут не иметь полезной нагрузки данных для поддержки инструкций манипуляции с кэш-памятью.

Некоторые транзакции посылаются многим адресатам в процессе завершения какой-то операции. Эти транзакции могут быть посланы или как ряд направленных транзакций, или как одна транзакция, если транспортный уровень имеет возможность мультикастинга. Возможность мультикастинга и операции определяются в соответствующих спецификациях RapidIO транспортного уровня.

Всего в гл. 3 спецификации [2д] описано 19 операций 10 типов: считывание данных, считывание инструкций, считывание для внесения в кэш-память, объявление данных кэш-памяти недействительными, операция отбраковки гранул когерентности, два типа операций на TLB, два типа операций на кэш-памяти (включая операции типа " flush") и операции ввода/вывода. Указанные операции требуют посылки транзакции отклика. Подробное описание этих операций и форматы соответствующих пакетов см. в спецификации [2д].

ЛОГИЧЕСКИЙ УРОВЕНЬ. НАДЕЖНОСТЬ И УПРАВЛЕНИЕ ОШИБКАМИ

Систему RapidIO с точки зрения надежности относят к классу систем с архитектурой, устойчивой к отказам, а с точки зрения режима функционирования – к системам класса "24/7", т.е. допускающим обслуживание 24 часа в сутки, 7 дней в неделю. В плане возможностей резервирования система поддерживает практически все схемы. Она допускает горячую замену карт и блоков и обеспечивает интенсивный мониторинг параметров и отказов, критичных к надежности элементов. Мониторинг параметров проводится на предмет не только выхода параметров за границы допуска, но и деградации параметров. Система обеспечивает изоляцию отказов, выбор пути передачи в сети с обходом проблемных узлов, а также реализует управление качеством обслуживания.

Спецификация управления ошибками [2з] описывает, с одной стороны, требования ко всем физическим и логическим уровням, выполнение которых необходимо для своевременного и эффективного обнаружения ошибок, с другой – режим работы устройства в случае, если ошибка обнаруживается, а также то, как регистры "Port *n* Control" CSR, осуществляющие это, управляются программно и аппаратно. Как было указано выше, обнаружение ошибок проводится с помощью различных схем контроля четности и использования CRC – контроля с помощью циклических избыточных кодов. К сожалению, схема CRC, основанная на добавлении избыточных бит, вычисленных при кодировании, не позволяет исправлять ошибки. Кодов, исправляющих ошибки, в системе не используется. Однако исправление ошибок проводится с помощью интенсивного использования известного механизма запроса на повтор осуществляемой операции, реализуемого с помощью транзакций отклика.

СПЕЦИФИКАЦИЯ ЗАГРУЗКИ И ВЗАИМОДЕЙСТВИЯ СЕТЕВЫХ УСТРОЙСТВ RapidIO

Спецификации RapidIO устанавливают рамки для осуществления широкого круга реализаций. Так, спецификация [2ж] описывает стандартный набор решений для запуска и начальной загрузки (инициализации) системы, а также проектирования устройств и систем, обеспечивающих их взаимодействие в рамках RapidIO. Она освещает следующие вопросы: исследование и инициализация системы, требования к устройствам RapidIO, взаимодействие RapidIO с шиной PCI и обрабатывающих элементов с GSM.

Инициализация системы RapidIO

Простой вариант инициализации системы состоит в том, чтобы загрузить во все устройства, подключенные к системе, информацию, необходимую для инициализации, в момент начальной загрузки системы, используя загрузочные ROM или аналогичные устройства. В RapidIO используется другой более гибкий метод, учитывающий факт возможного изменения структуры систем при наличии горячей замены и использования устройств типа "plug-and-play".

Этот метод использует один или несколько хост-процессоров, которые сначала обследуют конфигурацию системы, а затем проводят процедуру инициализации. При этом хост сначала запрашивает соседний коммутатор, направляющий запрос на тот выходной порт, к которому через агента подключено устройство, хранящее код начальной загрузки в локальном ROM. Заполучив его, хост начинает обследовать, что подключено к каждому порту коммутатора. Все подключенные устройства должны иметь регистр CSR, содержащий информацию, нужную при обследовании. Когда все устройства системы будут идентифицированы и получат уникальные ID базового устройства, хост осуществит их конфигурацию, требуемую для конкретного приложения. После этого устройства получают возможность формировать запросы.

Требования к устройствам RapidIO

Сети RapidIO построены вокруг двух основных типов устройств – строительных блоков (рис. 26):

- конечных точек – источников и стоков для пакетов;
- коммутаторов – средств передачи пакетов между портами без их анализа.

К ним примыкают все устройства, поддерживающие обслуживание транзакций и взаимодействие с приложениями.

Прежде всего, все устройства разделены на три класса. Для первого класса требования минимальны и растут с каждым классом. Классы могут иметь дополнительные или

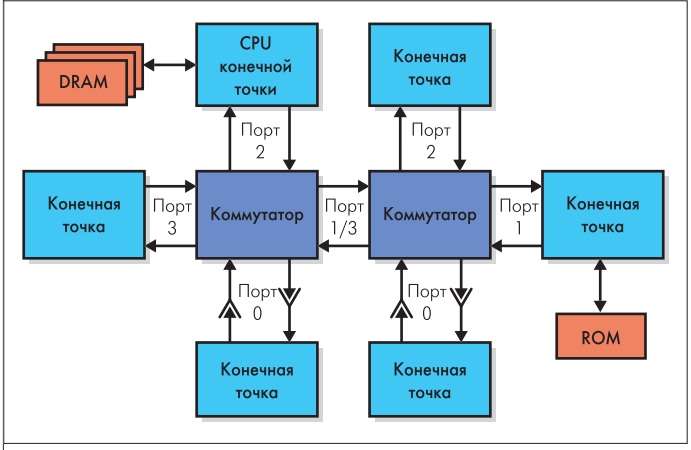

Рис. 26. Основные сетевые строительные блоки RapidIO

опциональные функции. Однако все устройства должны удовлетворять и поддерживать:

- один или более портов 8/16 LP-LVDS и/или порты 1x/4x LP-Serial;
- ID-поля малых (8-битных) транспортных устройств;
- процедуры восстановления поврежденного пакета или символа управления;
- протокол повторной посылки пакета;
- управление потоком для узких мест (throttle) в портах физического уровня типа 8/16 LP-LVDS;
- упорядочение транзакций по "flowID";
- внутреннее обслуживание ошибок (для коммутаторов);
- максимальный размер пакета 276 байт (для коммутаторов);
- максимальный размер полезной нагрузки данных 256 байт (для конечных точек).

Каждое устройство должно содержать следующие регистры CAR: с ID и информацией об устройстве и сборке (Assembly), с особенностями ОЭ и с операциями в конечных точках над источниками и назначениями.

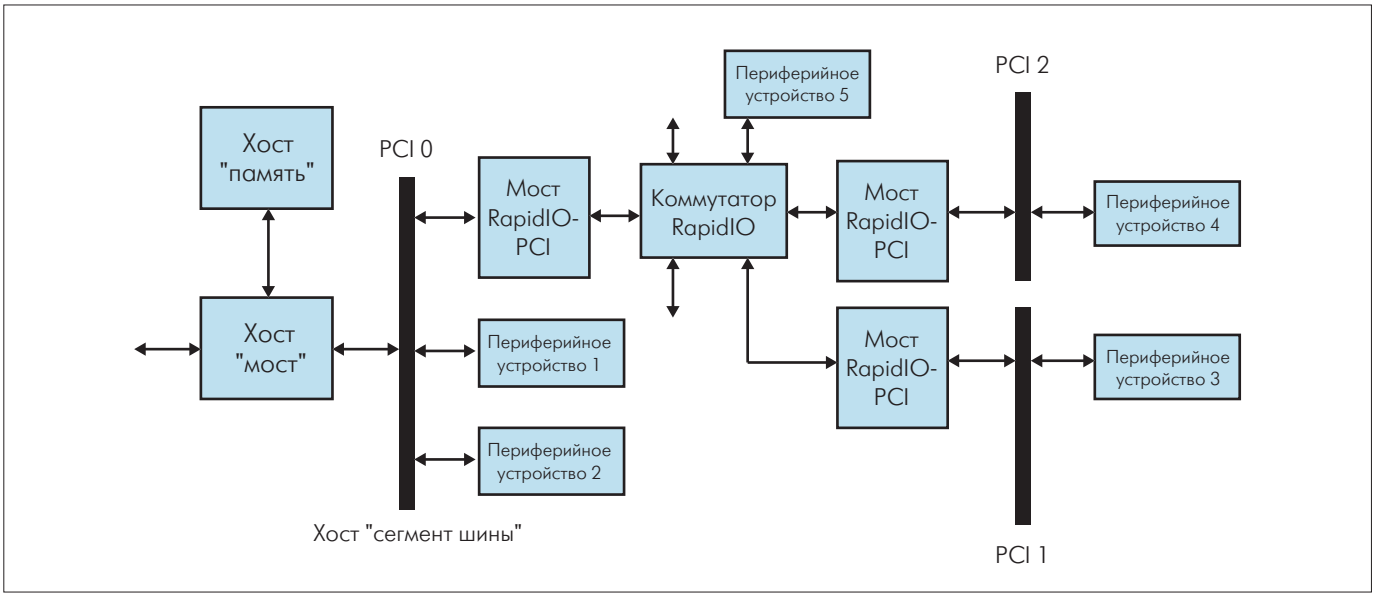

Рис. 27. Пример связи системы RapidIO с шиной PCI

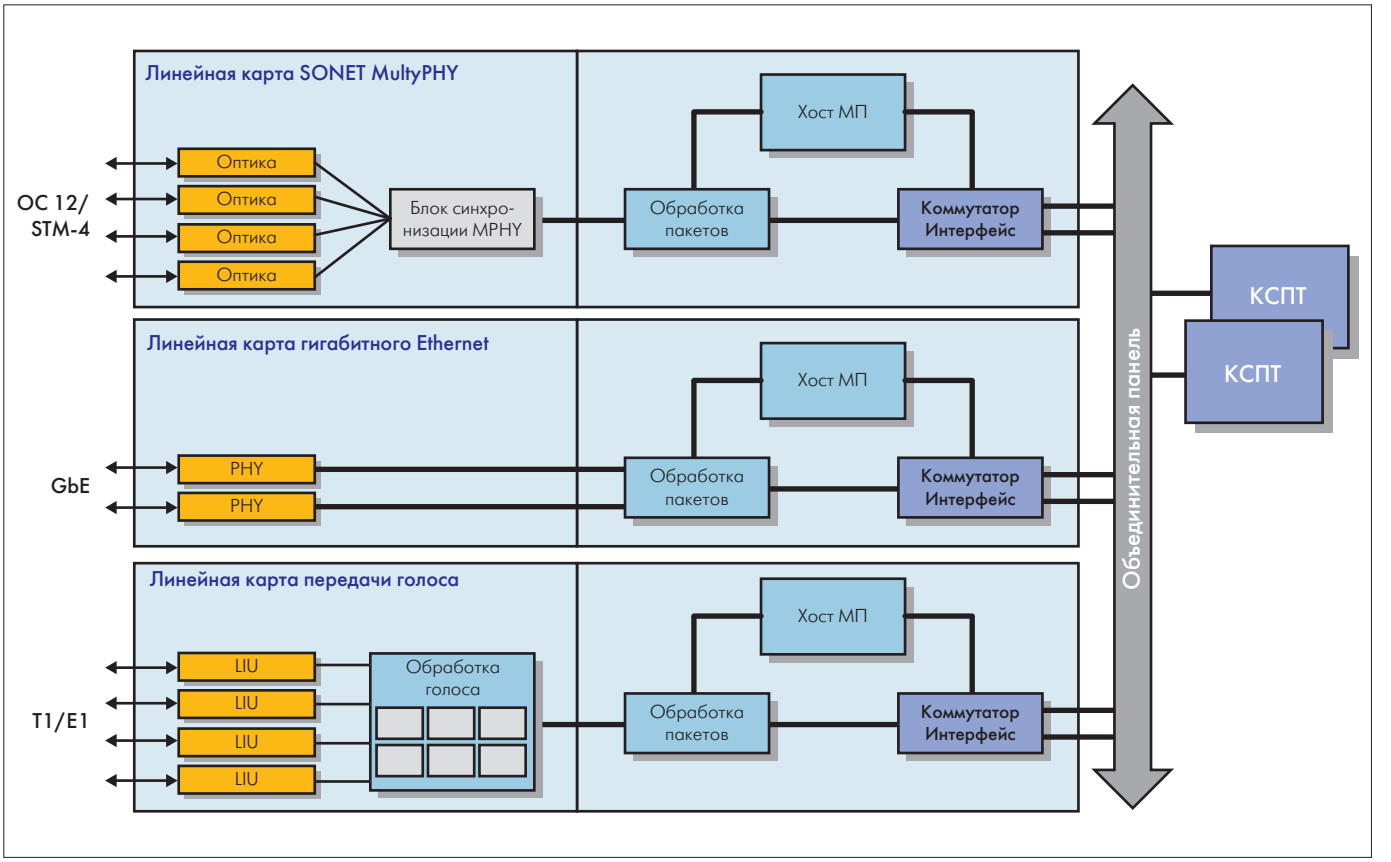

Рис. 28. RapidIO как мультисервисный коммутатор

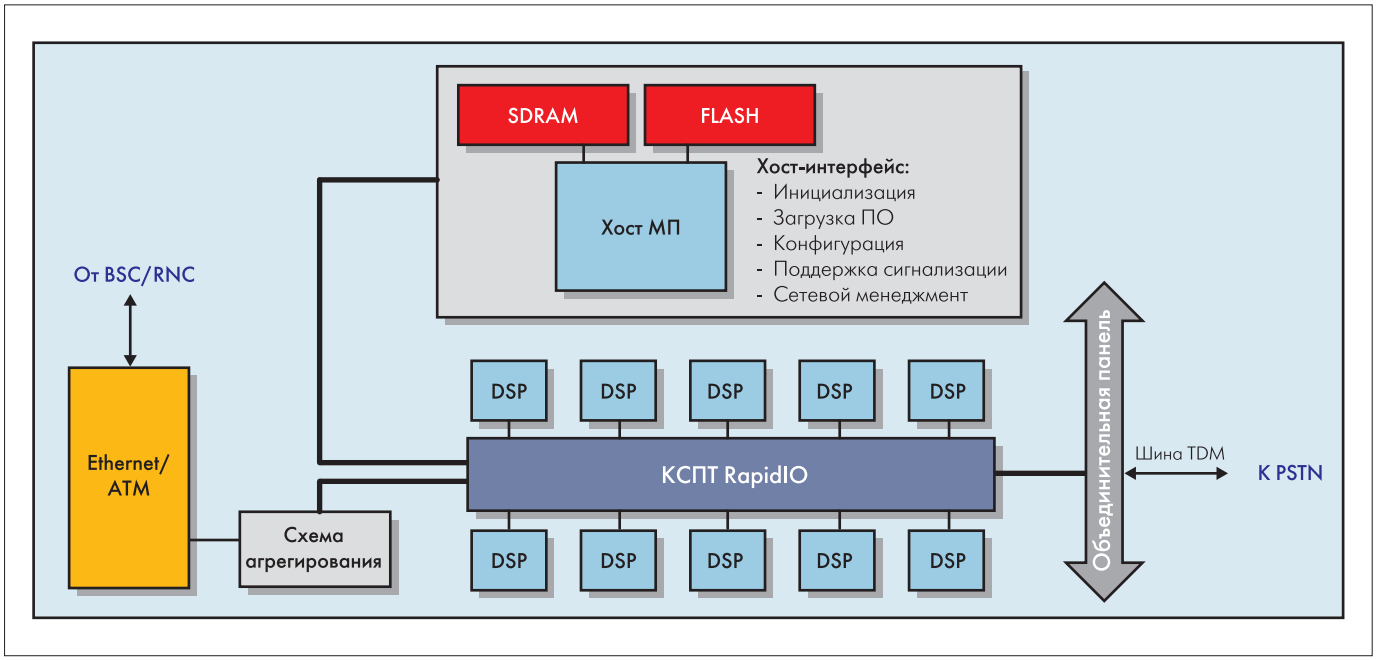

Рис. 29. Мобильный центр коммутации, использующий RapidIO

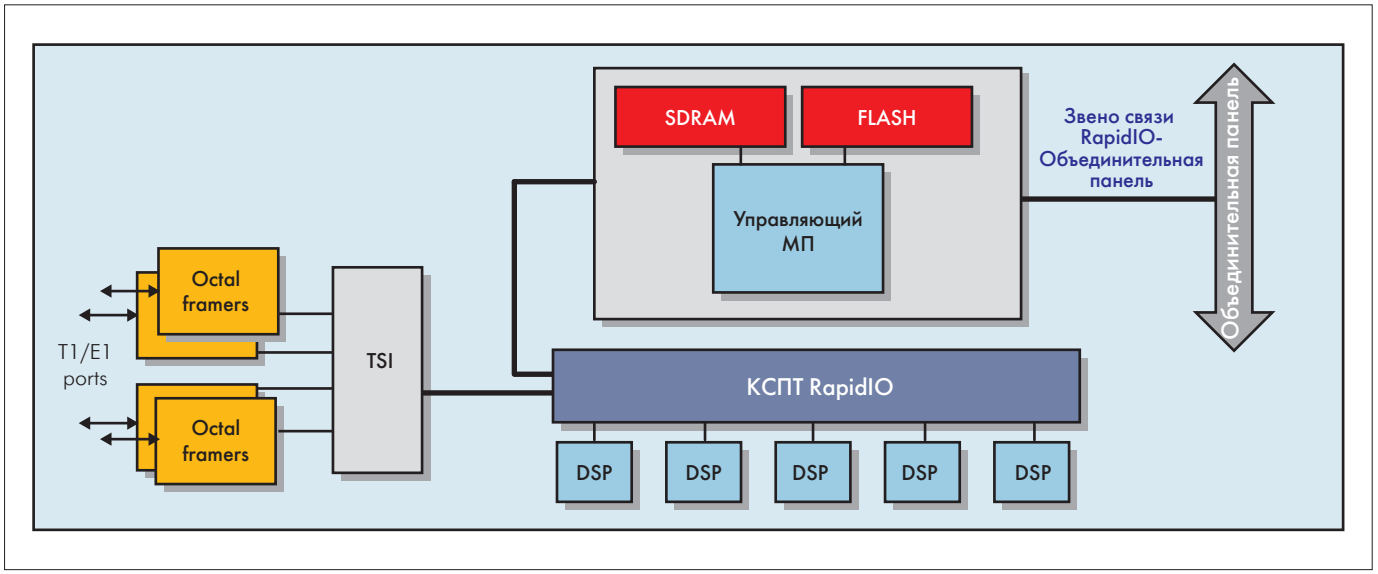

Рис. 30. Коммутаторы голосовых шлюзов цифровых абонентских линий

Первый класс должен дополнительно поддерживать 34-разрядную адресацию. Второй и третий классы отличаются только более широким набором операций.

Взаимодействие RapidIO с шиной PCI

Система RapidIO допускает работу с шиной PCI. Однако RapidIO и PCI имеют разные протоколы и требуют использования функций трансляции для передачи транзакций между собой. Для этого необходимо применять ОЭ типа "мост". На рис. 27 показана типичная схема использования сегментов шин PCI (с навешенными на них периферийными устройствами) и мостов: простых и типа RapidIO-PCI, позволяющих связать периферийные устройства разного типа с коммутатором системы RapidIO.

Для транзакций, которые должны передаваться между RapidIO и PCI, необходимо отобразить адресные пространства, определенные на шине PCI, на пространства RapidIO, транслировать типы PCI-транзакций в операции RapidIO и обеспечить поддержку требованиям производитель/потребитель шины PCI как для версии 2.2, так и для PCI-X.

Взаимодействие ОЭ с GSM

ОЭ должны удовлетворять различным требованиям при работе с GSM. Протокол GSM считает ОЭ интегрированным элементом, включающим локальную память и контроллеры памяти и ввода-вывода, что усложняет требования при реализации взаимодействия. RapidIO описывает пять ОЭ: процессор-память, только память, только процессор, ввод-вывод и коммутатор. Из них только коммутатор полностью готов к работе с GSM. Остальные (кроме ввода-вывода) требуют дополнительного описания части протокола для реализации каждого из них. Устройства вво-

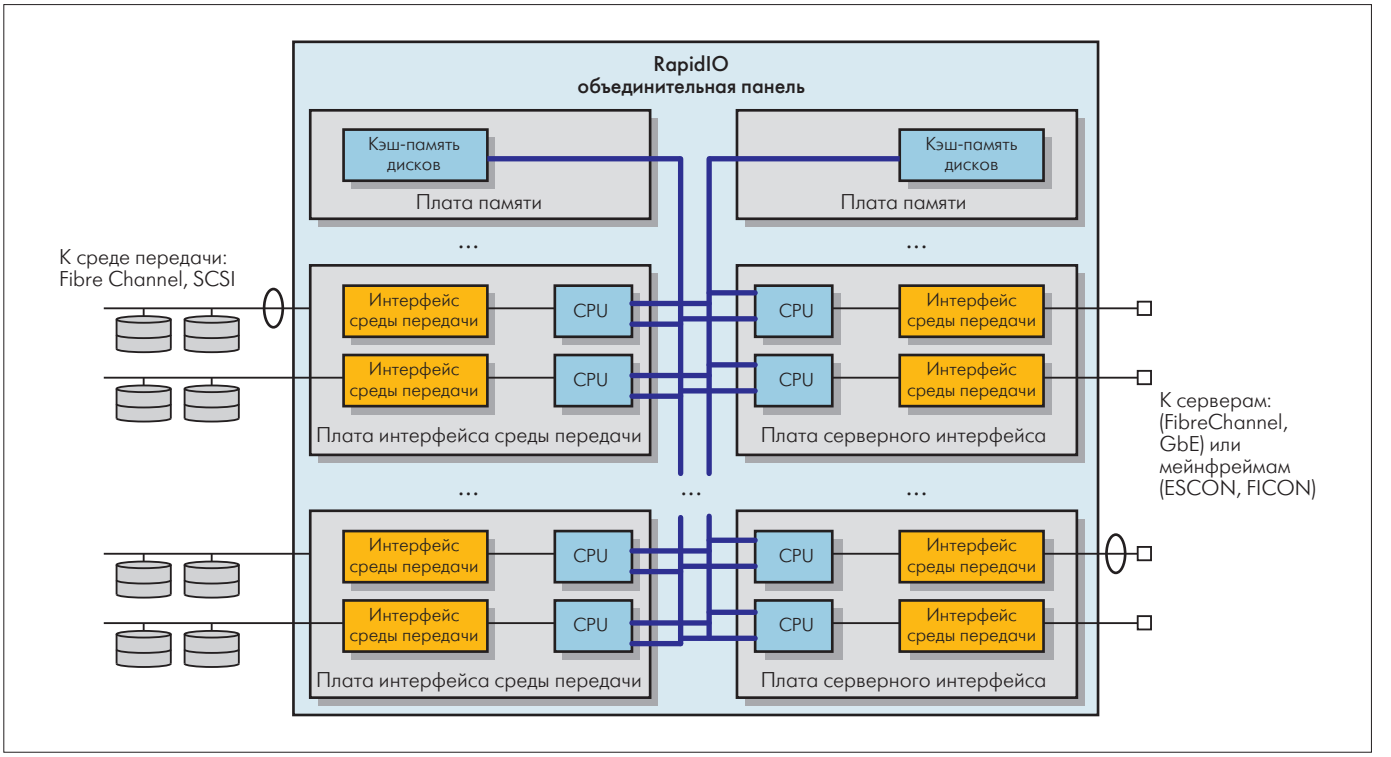

Рис. 31. RapidIO в качестве коммутатора системы хранения данных предприятия

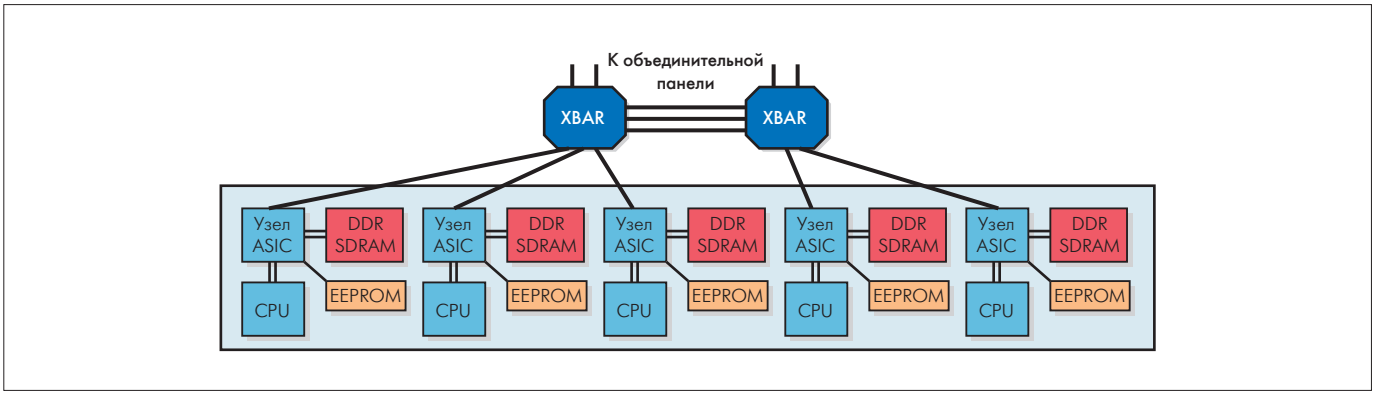

Рис. 32. Схема коммутатора системы обработки сигналов и изображений

да-вывода практически не участвуют в работе GSM, хотя и имеют возможность считывать (I/O Read) и записывать (Data Cache Flush) данные в память (более подробно см [2ж]).

ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ СИСТЕМЫ RapidIO В ПРИЛОЖЕНИЯХ

Существуют различные приложения, использующие RapidIO [5]. К ним относятся:

- мультисервисные коммутаторы (рис. 28), позволяющие осуществлять компьютерную обработку стандартных потоков T1/E1 цифровых АТС, гигабитного Ethernet, сформированного в ЛВС, и потоков SONET/SDH магистральных систем дальней связи;
- центры коммутации мобильной связи (рис. 29), включая сети сотовой и транкинговой связи, отличающиеся интенсивным использованием цифровых сигнальных процессоров (DSP);
- коммутаторы голосовых шлюзов цифровых абонентских линий и линий доступа с мультиплексорами DSLAM (рис. 30), отличающихся насыщенной смесью каналов передачи голоса, видео и данных;
- коммутаторы систем хранения данных предприятий и крупных информационных центров (рис. 31), использующих массивы типа RAID и другие устройства дисковой и ленточной памяти большой емкости;
- коммутаторы систем обработки сигналов и изображений (рис. 32);
- платформы типа ATCA для встраиваемых технологий в научных, промышленных и военных приложениях (см. [1]), где RapidIO используется для связи между чипами, картами и объединительной панелью ATCA (стандарт PICMG 3.5 RapidIO).