И.В. Белянин, П.Ю. Петраков (ЗАО «МЦСТ»),
д.т.н. В.М. Фельдман (ПАО «ИНЭУМ им. И.С. Брука)

I. Belyanin, P. Petrakov, V. Feldman

ФУНКЦИОНАЛЬНАЯ ОРГАНИЗАЦИЯ И АППАРАТУРА СЕТЕВОГО ВЗАИМОДЕЙСТВИЯ МОДУЛЕЙ В ВЫЧИСЛИТЕЛЬНОМ КЛАСТЕРЕ НА БАЗЕ МИКРОПРОЦЕССОРОВ С АРХИТЕКТУРОЙ «ЭЛЬБРУС»

FUNCTIONAL ORGANIZATION AND HARDWARE MEANS OF NETWORK INTERCONNETION OF MODULES IN COMPUTER CLUSTER BASED ON «ELBRUS» MICROPROCESSORS

Статья посвящена проблеме формирования функциональной уровневой структуры сетевого взаимодействия вычислительных модулей в кластере на базе микропроцессоров с архитектурой «Эльбрус». Приведены особенности реализации устройств, соответствующих сетевым уровням.

The article is devoted to problems of developing layered functional organization for network interaction of computer cluster modules based on the microprocessors with «Elbrus» architecture. Features of device implementation appropriate *to the functional layers are described.*

Ключевые слова: вычислительная сеть, вычислительный кластер, сетевой уровень, логический уровень, сетевой коммутатор, контроллер сетевого взаимодействия.

Keywords: computer network, computing cluster, network layer, logical layer, *network switch, network interconnect controller.*

Введение

Вычислительный кластер в этой статье рассматривается как объединенная на основе сетевой функциональной организации группа вычислительных модулей (узлов), которая образует для пользователей единый вычислительный ресурс. Использование кластерного

подхода позволяет значительно уменьшить время выполнения задачи путем ее разделения на параллельные потоки, каждый из которых обрабатывается отдельным вычислительным модулем.

Интегральная производительность кластера зависит не только от производительности его узлов, но и от качества и скорости связей между ними. Существует большое количество решений по объединению отдельных машин в вычислительную сеть с использованием стандартных высокоскоростных интерфейсов, таких как 10G Ethernet, InfiniBand, RapidIO и других, однако проведенный авторами анализ показал, что применение любого из них существенно увеличивает стоимость оконечной аппаратуры. Кроме того, принималось во внимание требование разработчиков программного обеспечения по поддержке удаленных DMA-обращений, которому соответствуют не все стандартные сетевые интерфейсы. Исходя из этого, было решено разработать собственную функциональную организацию сетевого взаимодействия вычислительных модулей кластера со своим стеком протоколов и реализующую ее аппаратуру. Особенности построения сетевого взаимодействия приведены ниже.

1. Принцип сетевого взаимодействия вычислительных модулей в составе кластера

В традиционных терминах организации компьютерных сетей функции взаимодействия вычислительных модулей в составе кластера можно отнести к четырем уровням – физическому, сетевому, транспортному и прикладному (рис. 1).

В общем случае допускается любая физическая среда сетевого взаимодействия при условии, что она обеспечивает достаточную пропускную способность и время доступа. В приведенной на рис. 1 схеме на различных уровнях задействованного соединения могут выполняться контроль целостности пакетов, кодирование сигналов (например, 8B\10B) и помехозащитное кодирование (например, с использованием CRC).

Функции сетевого и транспортного уровней поддерживаются связанной с узлом аппаратурой двух типов – сетевыми коммутаторами и контроллерами сетевого взаимодействия (в дальнейшем «контроллерами») соответственно. Сетевые коммутаторы, которые могут быть выполнены в виде отдельных устройств или интегрированы в контроллер, выполняют маршрутизацию и передачу данных между смежными узлами кластера. Контроллеры осуществляют обмен «end-to-end» между объектами системного программного обеспечения (ПО), а также между абонентами, обеспечивающими прикладным процессам услуги транспортировки. Для каждого из этих двух уровней сформированы протокол, специфицирующий услуги уровня, словарь протокольных единиц данных и алгоритм взаимодействия объектов (сущностей) данного уровня в процессе обеспечения услуг.

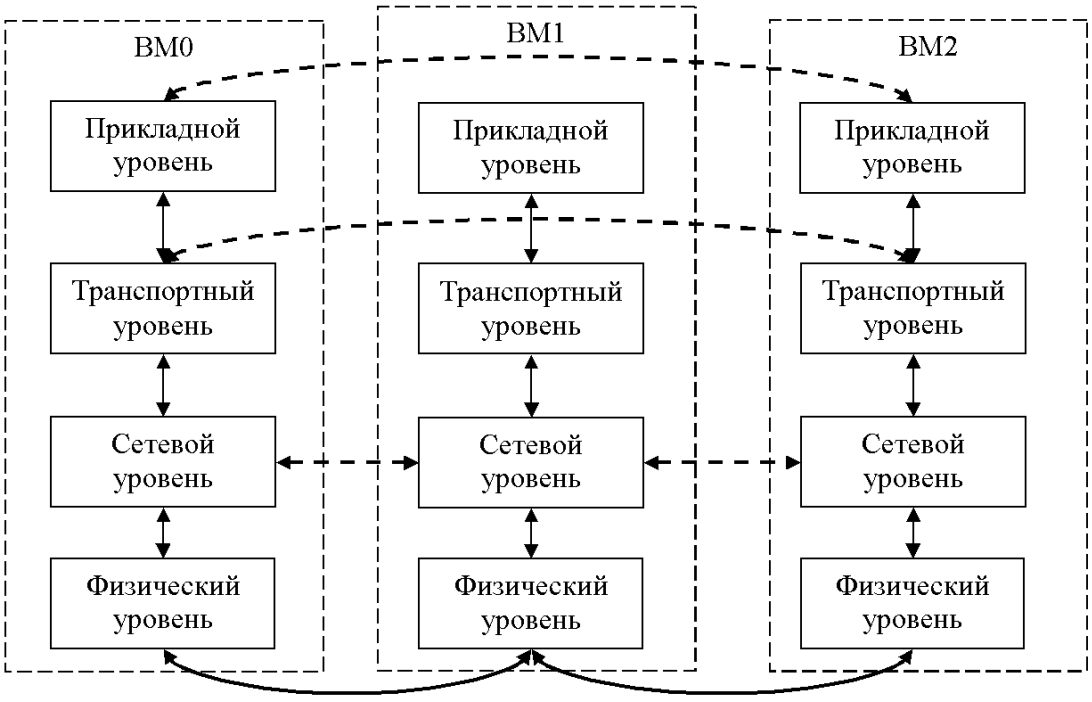
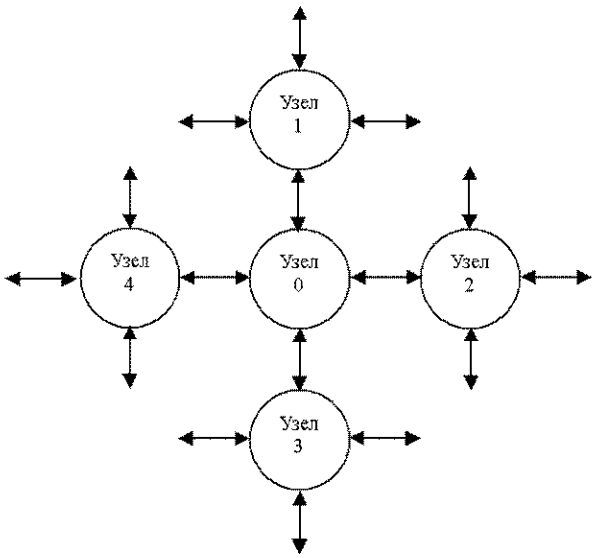


Рис. 1. Схема взаимодействия вычислительных модулей ВМ0 и ВМ2 с маршрутизацией через транзитный модуль ВМ1

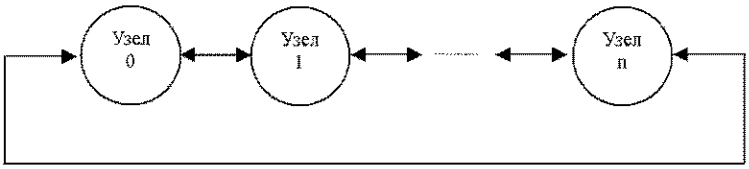
Несколько абонентов могут находиться в одном вычислительном модуле, с

вычислительным модулем могут быть связаны несколько контроллеров и/или сетевых коммутаторов. Сетевой коммутатор может иметь несколько внутренних портов, через

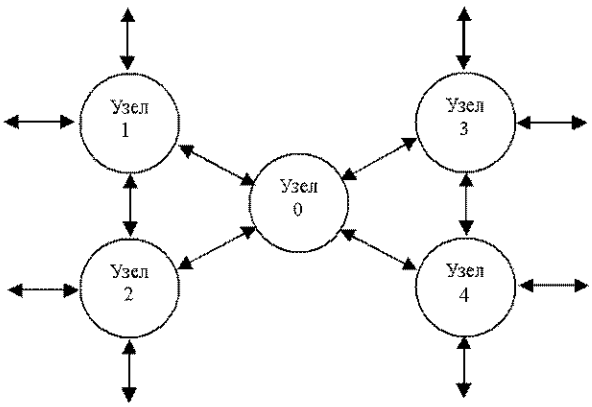
Звезда



Кольцо



Бабочка



которые получают доступ в сеть абоненты его узла, и внешних портов, связывающих с аппаратурой других узлов кластера через соединения физического уровня. Для обеспечения адресной доставки данных каждому абоненту кластерной сети в процессе инициализации сети присваивается уникальный идентификатор (в рамках одного кластера поддерживается до 255 абонентов), а в каждом сетевом коммутаторе формируется

таблица маршрутизации. Возможные варианты топологии кластера представлены на рис. 2. Масштабируемость сети сверх 255 абонентов обеспечивается отдельными межсетевыми шлюзами.

Рис. 2. Варианты топологии кластера

На прикладном уровне связь через абонентов позволяет задействовать вычислительные ресурсы других узлов или предоставлять им свои ресурсы. В системное ПО прикладного уровня входит драйвер, который отвечает за настройку сетевой аппаратуры и наличие актуальной информации о ресурсах сети. В этой своей функции он использует локализованный в сетевом коммутаторе стандартный блок пространства конфигурации Element_Config_Space с регистрами, содержащими информацию о типе, возможностях, статусе узла и определяющими некоторые функции управления, причем это пространство доступно не только для локальных обращений, но и для обращений от других узлов.

2. Сетевой уровень

На сетевом уровне выполняется взаимодействие между сетевыми коммутаторами, основанное на использовании находящейся в каждом из них таблицы маршрутизации, которая ассоциирует идентификаторы функционирующих в сети абонентов с номерами портов коммутатора. Она может формироваться вручную или в автоматическом режиме. В первом случае возможно использование различных методов, например, формирование и обновление всех таблиц одним привилегированным абонентом или включение каждым абонентом своего идентификатора во все таблицы. При автоматическом режиме используется механизм аппаратной трансляции изменений в конкретной таблице по всей сети.

Количество портов сетевого коммутатора определяется топологией сети, например, для реализации кластера с топологией 3D-тор каждому узлу необходимо иметь шесть внешних каналов сетевого взаимодействия. Поэтому в данном проекте устройство разрабатывалось с учетом необходимости введения различного количества портов. Вариант блок-схемы сетевого коммутатора с четырьмя портами приведен на рис. 3.

Пользовательские процессы могут задать через сетевой драйвер наличие нескольких виртуальных каналов в каждом порту (например, три канала VC0-VC2 на рис. 3), каждому из которых соответствует буфер обмена [1]. Виртуальные каналы позволяют обеспечивать должное качество обслуживания различных типов трафика и строить многомерные сети. Для организации использования виртуальных каналов в каждый порт введена регистровая структура, ассоциирующая класс трафика с определенным виртуальным каналом (TC/VC Map). Связь портов этого устройства с физической средой осуществляется через интерфейс WLink, разработанный компанией МЦСТ для высокоскоростных соединений. Допускается возможность использования различных физических соединений для разных портов.

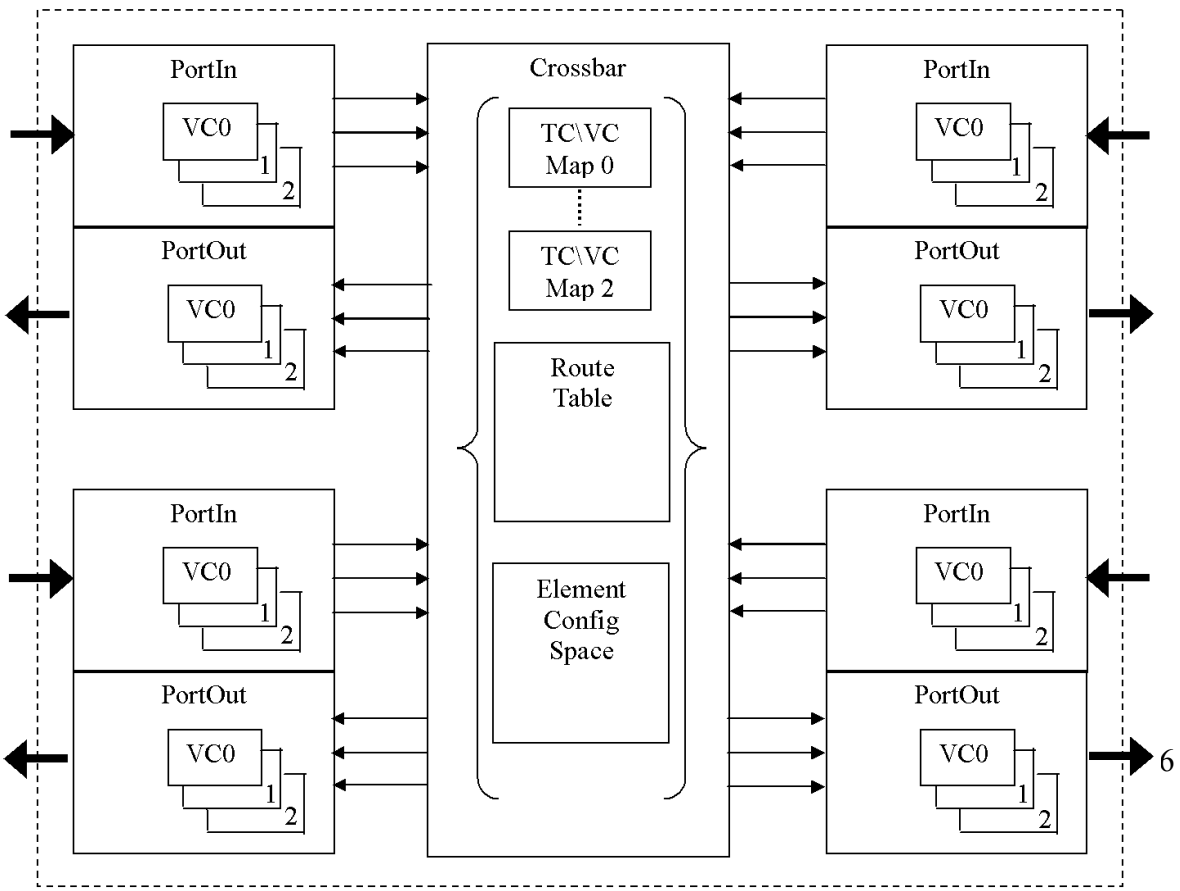


Рис. 3. Блок-схема сетевого коммутатора

Порт состоит из двух блоков – PortIn и PortOut. Коммутатор принимает пакет через внешний интерфейс блока PortIn, помещая его целиком в буфер приема, соответствующий номеру заданного в пакете виртуального канала. Если принятый пакет содержит обращение к внутренним регистрам сетевого коммутатора, то коммутатор формирует ответ и отправляет его в блок PortOut этого же порта. Если же принятый пакет представляет транзитный трафик, то коммутатор направляет его в другой выходной порт в соответствии с таблицей маршрутизации (Route Table) и состоянием регистров в блоке Element Configuration Space, причем виртуальный канал определяется в соответствии со структурой TC\VC Map выходного порта. Связь сетевого коммутатора с контроллером сетевого взаимодействия осуществляется через один из портов.

3. Транспортный уровень

Каждый абонент в сети имеет собственный идентификатор Base_Device_ID, используемый для адресации обменов, которая может быть статической или динамической. При статической адресации подразумевается, что абоненты получают идентификаторы, задаваемые локальным ПО, – в этом случае контроль уникальности идентификаторов и их задание лежат на операторе кластера. При динамическом задании адресов в сети присутствуют привилегированные абоненты, именуемые «хостами», которые отвечают за обнаружение новых абонентов, присваивание адресов, конфигурацию и подключение к сети. Взаимодействие хостов друг с другом осуществляется при инициализации сети их программным обеспечением. Абонент может быть настроен только одним хостом. Чтобы при этом избежать конфликтов между хостами, в регистровое пространство настраиваемого устройства введен регистр Host_Base_Device_ID. Хост захватывает контроль над абонентом, внося в этот регистр

свой идентификатор.

Пересылка данных между абонентами (сервисная единица данных) задается дескриптором обмена, формируемым в системной памяти запросчика и определяющим устройство назначения, объем пересылаемых данных, их расположение в адресных пространствах узлов отправителя и получателя, а также другие параметры. Пересылаемые данные могут располагаться сегментировано в адресных пространствах. В этом случае дескриптор обмена содержит ссылку на список дескрипторов сегментов.

Единицей взаимодействия абонентов на транспортном уровне является «обмен» – операция по пересылке заданной порции данных «end-to-end» из памяти одного узла в память другого. Поддерживаются следующие варианты обменов:

●Put – операция записи в память удаленного узла по адресу, указанному в дескрипторе обмена. Ограничение на размер данных – до 1 Гб.

●Get – операция чтения памяти удаленного узла по адресу, указанному в дескрипторе обмена. Ограничение на размер данных– до 1 Гб.

●Message – операция передачи сообщения в определенную структуру (почтовый ящик) абонента. Номер ящика, если у него их несколько, указывается в дескрипторе обмена. Ограничение на размер сообщения – до 4 Кб.

●Maintenance – служебная операция управления и поддержки сети, выполняющая обращение (чтение или запись) в регистровое пространство конкретного элемента кластера. Операции этого типа доступны только системному сетевому программному обеспечению.

Все операции транспортного уровня предполагают контроль доставки: на каждую выполняемую пересылку должен возвращаться ответ со статусом завершения. Если он не приходит в течение некоторого времени, заданного значением регистра Timeout_Value в регистровом пространстве абонента, то данный запрос следует повторить [1]. Операция

завершается при получении ответа с положительным или отрицательным статусом либо при достижении счетчиком повторов максимального значения, также определенного в пространстве регистров абонента. Во втором случае абонент завершает транзакцию самостоятельно, формируя отчет об ошибке.

Контроллер сетевого взаимодействия (Node Interconnect Controller, NIC) необходим для обеспечения функций транспортного уровня, а в интегральной конфигурации может включать сетевой коммутатор. Из соображений универсальности устройство разрабатывалось таким образом, чтобы либо напрямую подключаться к процессорам вычислительного узла, либо – через стандартный PCI-Express интерфейс в качестве карты расширения. Причем в варианте многопроцессорной оконечной системы возможно подключение контроллера к нескольким процессорам одновременно для более эффективного использования внешних каналов связи. Возможна реализация, предполагающая подключение к процессору вычислительного модуля через его внешние связи (интерфейс IO-link), каждая из которых обеспечивает пропускную способность до 8 Гбит/с. В варианте с картой расширения возможно выполнение устройства на базе ПЛИС Altera с использованием возможности HardIP PCI-Express.

Для системного ПО вычислительного модуля контроллер представляет собой PCI устройство, поддерживающее два способа задания параметров процесса пересылки данных (трансфера) – PIO и DMA, причем оба они могут применяться одновременно. В первом случае параметры и данные для трансфера задаются значениями соответствующих полей блока PIO-регистров. (PIO Register Block) Так как он находится в самом устройстве и имеет конечную величину, то при использовании этого способа вводится ограничение на количество передаваемых данных (в представляемой реализации оно составляет 256 двойных слов). Операции управления и поддержки сети категории Maintenance можно задать, только используя PIO-регистры.

При использовании DMA-способа вся информация о последовательности обменов находится в системной памяти в виде кольцевого буфера, содержащего очередь дескрипторов. Каждый из его элементов представляет собой логическую структуру, определяющую параметры трансфера, такие как устройство назначения, размер обмена, расположение передаваемых данных в адресных пространствах отправителя и получателя.

Устройство рассчитано на применение в многопроцессорной системе, где (соответственно числу обслуживаемых процессоров) независимо выполняются несколько процессов, количество которых – это параметр масштабирования, зависящий от конкретной реализации и отображаемый в специальном информационном регистре. Каждый процесс обеспечивается определенными ресурсами – блоком PIO-регистров, очередью DMA-дескрипторов, почтовым ящиком для приема обменов типа Message. Вследствие этого выполняемый в данном узле процесс может независимо осуществлять обмены с процессами остальных узлов сети.

Блок-схема устройства в варианте подключения к трем процессорам через интерфейсы IOlink представлена на рис. 4. В модуле DMA Dispatcher происходит коммутация внутренних интерфейсов модулей контроллера, через которые идет обмен с системной памятью, с модулями IO Link Core, преобразующими внутренний интерфейс в системный.

Рис. 4. Блок-схема контроллера сетевого взаимодействия в варианте применения в многопроцессорной оконечной системе с тремя IOLink интерфейсами

Descriptor Fetch Block это модуль, отвечающий за получение параметров, необходимых для разных обслуживаемых процессов. Параметры могут быть вычислены с использованием значений регистров блока Register Block, либо получены из дескрипторов в системной памяти с использованием системного интерфейса, доступ к которому осуществляется через модули IOLink Core. На основе вычисленных параметров в модуле Packet Injection формируются отправляемые в сеть пакеты в формате обменов транспортного уровня. Входящие пакеты обрабатываются в модулях Service Control и Packet Injection. В первом из них происходит обработка только сервисных пакетов. Модуль Packet Ejection обрабатывает все другие вводимые сетью пакеты и формирует транзакции записи/чтения в системную память.

4. Прикладной уровень

При загрузке операционной системы сетевой драйвер определяет, что к данному вычислительному модулю подключен контроллер сетевого взаимодействия. Из статусных регистров контроллера в Register Block драйвер определяет его характеристики и затем конфигурирует это устройство – выделяет ему требуемую память, настраивает очереди дескрипторов, открывает сетевые порты и выполняет другие действия. После этого драйвер получает доступ к сетевым обменам транспортного уровня посредством maintenance-операций.

Если данный вычислительный модуль является привилегированным (хостом), т.е. отвечает за конфигурацию и работу всей сети, то драйвер обязан выполнить обход всех сетевых устройств, используя maintenance-операции. При этом он узнает о присутствующих в сети устройствах, настраивает те из них, которые не захвачены другим

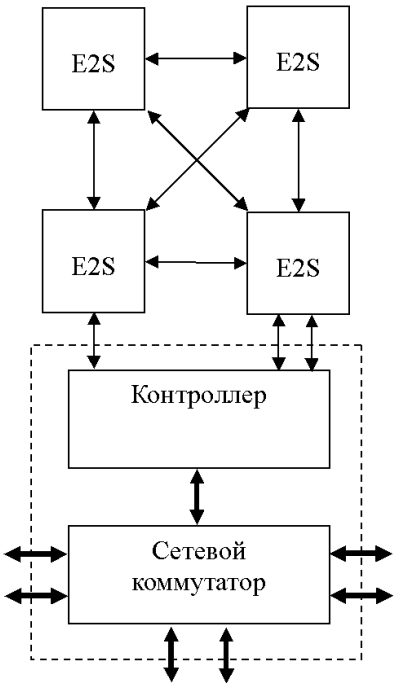
хостом, и получает информацию о вычислительных ресурсах сети. Настройка сетевых элементов производится в соответствии с правилами взаимодействия на транспортном и сетевом уровнях. Если же вычислительный модуль не является хостом, то сетевой драйвер ожидает того момента, когда подключенный к нему контроллер сетевого взаимодействия будет настроен одним из действующих в кластере хостов.

Через сетевой драйвер получает доступ к сетевому взаимодействию пользовательское приложение, исполняемое в вычислительном модуле. Основой особенностью (и достоинством) кластера является возможность взаимодействия его узлов при решении общей глобальной задачи, которая реализуется с использованием программных библиотек распределенных вычислений, например, открытых библиотек MPI (Message Passing Interface) или SHMEM (Symmetric Hierarchical Memory Access). Кроме этого, существует немало других программных продуктов. Пример взаимодействия представлен на рис. 5, где приведена ситуация, когда пользователь (оператор кластера) запускает в хосте ID0 приложение, предполагающее распределенные вычисления. Программное обеспечение хоста с использованием библиотеки делит исходную задачу на определенное количество подзадач и запрашивает соответствующие вычислительные ресурсы у своего сетевого драйвера, который в ответ выдает идентификаторы узлов, доступных для сетевого взаимодействия. В данном случае это узлы 1, 2, 3, 4, 5 и 6. Получив доступ к вычислительным ресурсам кластера, приложение запускает в них подзадачи. После выполнения подзадачи каждый узел сообщает результаты хосту ID0, который выдает пользователю результаты выполнения исходной задачи.

Рис. 5. Схема взаимодействия узлов кластера на прикладном уровне

5. Реализация сетевого взаимодействия в вычислительном кластере на основе микропроцессоров с архитектурой «Эльбрус»

Представленная функциональная организации сетевого взаимодействия, обладая универсальностью, сформирована применительно к кластеру, узел которого состоит из



четырех процессорных модулей на базе микропроцессоров «Эльбрус-4С» (четырехъядерная система на кристалле [2]), объединенных высокоскоростными межпроцессорными линками в единую ccNUMA систему [3]. Контроллер сетевого взаимодействия и сетевой коммутатор интегрированы в составе отдельного устройства, подключаемого непосредственно к процессорам и имеющего до шести внешних портов. Принципиальная схема такого узла приведена на рис. 6.

Рис. 6. Общая схема реализации вычислительного модуля в кластере на микропроцессорах «Эльбрус-4С»

Для реализации функций кластера был написан сетевой драйвер, выполняющий настройку контроллера и коммутатора, а в варианте хоста – всей сети. Драйвер поддерживает оба типа формирования таблиц маршрутизации – автоматический и ручной. Кроме того, он включает интерфейс, через который происходит управление ресурсами, используемыми пользовательскими приложениями.

В качестве основы программного обеспечения, поддерживающего взаимодействие вычислительных модулей в кластере на уровне пользовательских процессов, выбрана библиотека MPI, модифицированная для работы с сетевым драйвером. Одной из главных особенностей этой модификации является возможность работы с сетевым контроллером напрямую, после того как тот выделит приложению требуемые ресурсы. Поддержка данной библиотеки позволяет исполнять на разработанном кластере большое количество существующих пользовательских приложений.

Заключение

Представленные результаты являются первым опытом компании ЗАО «МЦСТ» в проектировании вычислительных кластеров на базе собственных микропроцессоров. В процессе работы возник ряд проблем, среди которых можно отметить сложность формирования функциональной организации и соответствующего ей протокольного стека, поддерживающих любую топологию сети, реализацию сетевого контроллера на ПЛИС с заданной производительностью, моделирование и верификацию кластера. Тем не менее, основные проблемы были решены, и в настоящее время в стадии отладки находится вычислительный узел, представленный в предыдущем разделе статьи. Суммарное число вычислительных узлов в кластере предполагается равным 64, топология сети – 3D-тор. Сетевое взаимодействие осуществляется контроллером NIC с тремя процессорными линками и шестью внешними портами, реализованным на FPGA Cyclone

V.

Благодарности

Авторы выражают благодарность за содействие в написании статьи А.В. Брегеру, внесшему основной вклад в разработку прикладного уровня кластера, и А.А. Иванову, принявшему активное участие в разработке сетевой аппаратуры.

Литература

1. Mohapatra P. Wormhole routing techniques for directly connected multicomputer systems //ACM Computing Surveys (CSUR), 1998, т. 30, №3. С. 374-410.

2. Микропроцессор Эльбрус-4С, МЦСТ, 2014 [Электронный ресурс]. http://www.mcst.ru/mikroprocessor-elbrus4s. Дата обращения: 06.04.2015.

3. Ким А.К., Перекатов В.И., Ермаков С.Г. Микропроцессоры и вычислительные комплексы семейства «Эльбрус». – СПб.: Питер, 2013. 272 с.